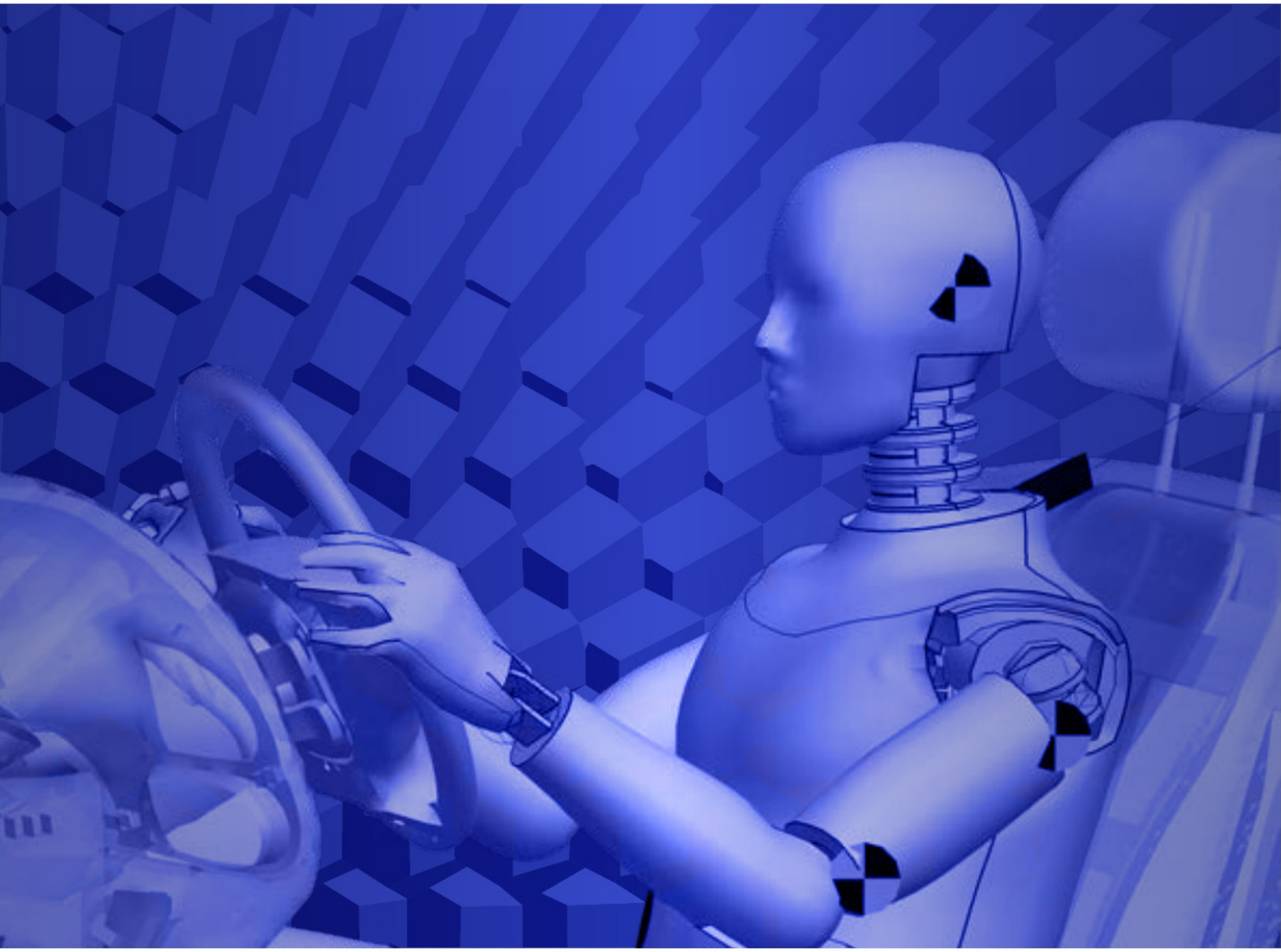
PRIMER
Version 16.0



Oasys Ltd

The Software House of ARUP

For help and support from Oasys Ltd please contact:

UK

The Arup Campus
Blythe Valley Park
Solihull
B90 8AE
United Kingdom
Tel: +44 121 213 3399
Email: dyna.support@arup.com

China

Arup
39/F-41/F
Huaihai Plaza
1045 Huaihai Road (M)
Xuhui District
Shanghai 200031
China
Tel: +86 21 3118 8875
Email: china.support@arup.com

India

Arup
Ananth Info Park
Hi-Tec City
Madhapur Phase-II
Hyderabad 500 081, Telangana
India
Tel: +91 40 44369797 / 98
Email: india.support@arup.com

Web: www.arup.com/dyna

or contact your local Oasys Ltd distributor.

LS-DYNA, LS-OPT and LS-PrePost are registered trademarks of Livermore Software Technology Corporation

Preamble **0.1**
Abstract 0.1
Development status 0.1
Memory requirements 0.1
Output devices 0.1
Revision history 0.2
Text conventions used in this manual 0.4
1 Running PRIMER **1.1**
1.1 Starting the code 1.1
1.2 Selecting a graphics device 1.1
1.3 If PRIMER will not start in "screen-menu" mode on your display 1.3
1.4 Running PRIMER in "Text-Only" mode. 1.4
1.5 Command Line arguments 1.5
2 Using Screen Menus **2.1**
2.0 Using the PRIMER screen menu system 2.1
2.1 Basic screen menu layout 2.1
2.2 Mouse and keyboard usage for screen-menu interface 2.2
2.3 Dialogue input in the screen menu interface 2.8
2.4 Window management in the screen interface 2.8
2.5 Using standard "file filter" boxes. 2.20
2.6 Obtaining Help and advice. 2.22
2.7 Error and Warning messages 2.23
2.8 Dealing with crashes 2.23
2.9 Quick Pick Function 2.27
2.10 Using Parameters in Edit Panels. 2.31
2.11 Formulae in Edit panels 2.32
2.12 Operational Hierarchy 2.33
2.13 Selecting Entities for Operations 2.33
2.14 Undo 2.52
3 Model manipulation **3.1**
3.0 How PRIMER treats "models" 3.1
3.1 MODEL > CREATE 3.3
3.2 MODEL > READ 3.4
3.3 MODEL > WRITE 3.18
3.4 MODEL > MERGE 3.36
3.5 MODEL > COPY Copying models internally. 3.50
3.6 MODEL > DELETE Deleting internal models 3.50
3.7 MODEL > RENUMBER Renumbering models and/or their contents 3.51
3.8 Model Contents 3.65
3.9 MODEL > CHECK 3.71
3.10 Operations on models 3.107
3.11 Viewing models 3.107
3.12 Memory management and usage. 3.107
3.13 Include Files 3.109
3.14 INCLUDE transform 3.124
3.15 MODEL > BUILD 3.128
3.16 Model Modified 3.166
3.17 Model POST 3.178
4 Model visualisation **4.1**
4.0 Visualisation and labelling. 4.1
4.1 Basic drawing commands: LI(ne), HI(dden line), SH(aded image) 4.2
4.2 Data Plotting Commands: 4.18
4.3 Controlling Model Visibility 4.44
4.4 Controlling Entity visibility and labelling 4.46
4.5 BLANKING Controlling entity visibility 4.53
4.6 Dynamic Labelling 4.58
4.7 Predictive Picking and Menu "Hover Over" 4.62
5 Keywords **5.1**
5.0 Index to keywords 5.1
5.1 Keywords 5.3
5.2 Databases: Importing data from Pre-defined database files 5.401
5.3 Contact Penetration Checking 5.409
5.4 Contouring panel gaps for sliding contact 5.426
5.5 Contact Penetration Fixing 5.428
5.6 Contact gap fixing 5.437
5.7 Tied contact fixing 5.439
6 Tools **6.1**
6.0 TOOLS panels 6.1
6.1 AIRBAGS 6.4
6.2 ASSIGN MASS 6.53
6.3 ATTACHED Displaying what is "attached to" things 6.66

6.4 BILL OF MATERIALS 6.71
6.5 BLANKING Setting entity visibility. 6.82
6.6 CHANGING UNITS 6.83
6.7 CHECK Running the model checker, and setting its options. 6.86
6.8 CLIPBOARD 6.87
6.9 COAT ENTITY: Coating entities with shells or segments 6.101
6.10 COMPARE: Comparing FE and CAD data 6.106
6.11 COMPOSITE 6.109
6.12 CONNECTIONS 6.123
6.13 CUT SECTIONS 6.262
6.14 DUMMIES Positioning Occupants 6.301
6.15 DUMMY AND SEATSQUASH 6.339
6.16 EJECTION MITIGATION SCRIPT 6.345
6.17 EXPLODE 6.348
6.18 FIND AND SKETCH 6.350
6.19 FMH Free Motion Headform 6.360
6.20 GROUPS 6.375
6.21 HIC Area Calculator 6.393
6.22 ICFD Setup script 6.403
6.23 INCLUDE Controlling *INCLUDE files. 6.405
6.24 INSTRUMENT PANEL PENDULUM 6.406
6.25 LOAD PATHS 6.414
6.26 Luggage Retention Automation Script (ECE-R17) 6.416
6.27 MACROS 6.419
6.28 MASS PROPERTY CALCULATOR 6.429
6.29 MEASURE Measuring the distance and angles between nodes and points on the screen. 6.434
6.30 MECHANISM Creating and analysing mechanisms 6.446
6.31 MESHING 6.485
6.32 NODE IMPORT 6.528
6.33 ORIENT Translating, rotating, scaling, reflecting, projecting 6.530
6.34 OTHER 6.557
6.35 PEDESTRIAN MARKUP 6.574
6.36 REMOVE Delete unwanted, model clean-up, node merging and duplicate elimination. 6.620
6.37 RIGIDIFY 6.631
6.38 SCRIPT Using JavaScript in PRIMER 6.634
6.39 Seat Belt Anchorage Automation Script (ECE-R14) 6.635
6.40 SEAT-BELTS Fitting seatbelts and related elements. 6.639
6.41 SEAT FOAM COMPRESSION 6.754
6.42 Sled Test Automation Script (ECE-R17) 6.772
6.43 Text Edit. External editing of any keyword 6.775
6.44 VOLUME CALCULATOR 6.777
6.45 XREFS Cross references viewer 6.782
7 Part Tree and Table **7.1**
7.0 Part tree and table. 7.1
7.1 PART TREE 7.2
7.2 PART TABLE 7.16
7.3 PART COMPARE 7.28
8 Images **8.1**
8.0 LASER: Introduction to Laser Plotting 8.1
8.1 Controlling laser plotting using the Laser Plotting panel 8.3
8.2 MARGINS... Modifying laser paper size on the page. 8.8
8.3 Creating Encapsulated Postscript (EPS) files. 8.8
8.4 Notes on laser plotting 8.9
8.5 Raster Images 8.9
8.6 3D PDF 8.15
8.7 WebGL 8.17
8.8 Read background image and watermark 8.18
9 Viewing Controls **9.1**
9.0 VIEWING CONTROL 9.1
9.1 PRIMER coordinate space systems and view layout. 9.2
9.2 The Viewing Control box 9.4
9.3 Using the "Compass Rose" 9.8
9.4 Dynamic Viewing (Using the mouse to change views). 9.10
9.5 Further commands in the Viewing Menu 9.14
9.6 Saved properties 9.24
9.7 Accelerated Graphics 9.34
9.8 Support for 3D mouse 9.42
9.9 Special 3D graphics driver options. 9.42
10 Scripting **10.1**
Introduction 10.1
10.1 Using JavaScript in PRIMER. 10.1

10.2 A Brief Tutorial on JavaScript in PRIMER 10.5
11 Quick Find 11.1
Introduction 11.1
Fuzzy Matching 11.3
Search Terms 11.4
Keyword Menus 11.5
LS-DYNA Manual 11.6
Model Entities 11.7
Tutorials 11.9
Options 11.10
APPENDICES A.1
APPENDIX I: Standard Object Names and Acronyms A.2
APPENDIX IIa: Dummy "tree" file format. B.1
APPENDIX IIb: Mechanism file format. B.15
APPENDIX IIc: "Positions" in Dummy and Mechanism data. B.21
APPENDIX IId: The Dummy Angles File (.daf) B.23
APPENDIX III: Origami "tree" file example C.1
APPENDIX IV: Airbag Folding example D.1
APPENDIX V: Seatbelt "Tree" File Structure E.1
APPENDIX VI: Format translation during MODEL> READ F.1
APPENDIX VII: Format translation during MODEL> WRITE G.1
APPENDIX VIII: "Curve" file formats H.1
APPENDIX IX: Primer database format I.1
APPENDIX X: Headform "tree" file example J.1
APPENDIX XI: Target and Position "tree" file example K.1
APPENDIX XII: Dialogue (typed in) Command Syntax L.1
APPENDIX XIII: Summary of "oa_pref", command-line and Environment Variable settings M.1
APPENDIX XIV: Automated model build from command line N.1
APPENDIX XV: Finding model mass properties O.1
APPENDIX XVI: XML format for model build P.1
APPENDIX XVII: MAT100 <DT> added mass for solid spotwelds Q.1
APPENDIX XVIII: Morphing keyword format R.1
Technical Topics to do with Graphics S.1
1. Window Managers S.1
2 Graphics S.2
3 Controlling Graphics in Oasys Ltd. LS-DYNA environment S.12
Installation organisation T.1
Version 16.0 Installation structure T.1
JaDe: The JavaScript debugger U.1
Viewing the script files and functions U.1
Adding/removing breakpoints U.1
Running the script U.2
Printing the value of a variable U.3
The call stack U.4
Exceptions U.5
Licences used in software V.1
Expat V.1
FreeType V.1
FFmpeg V.3
Jpeg V.3
Libcurl V.3
Libfame V.4
Libgif V.4
Libpng V.4
Libxlsxwriter V.5
Openssl V.7
PCRE V.8
PDFHummus V.9
POV-Ray V.9
SmoothSort V.9
Spidermonkey V.10
Win-iconv V.14
Zlib V.14
Fonts on Linux W.1
RedHat / CentOS 7 W.1
SUSE 12 W.1

Preamble

Abstract

Many pre-processors are available for mesh-building and general modelling, but their support for non-linear input data is only partial: specialist data such as load-curves, joints, complex material models or the INCLUDE file structure of the data can be lost when an existing analysis is taken back into them for modification, or when models are merged.

In addition there are several specialist functions peculiar to particular types of analysis, such as occupant positioning, which are not provided satisfactorily by general-purpose pre-processors.

PRIMER is designed to solve these problems. It is capable of reading, processing and writing out the entire LS-DYNA version R9.0 onwards keyword input deck, with no exceptions or omissions: no information is lost during processing. It will also read and write several other common formats.

Input decks may be visualised directly, and any number of input models may be merged intelligently into a single output model; with the additional ability to move, delete, edit and check models, parts or individual components in the process.

In addition PRIMER15.0 provides several specialist features for LS-DYNA analysis. These are features such as occupant positioning, belt fitting, airbag folding, spotwelding, massing-up and de-penetration of contacts.

Development status

This manual documents PRIMER15.0. The code is still being developed, and this version provides full compatibility with LS-DYNA release R9.0.

Memory requirements

Memory is allocated dynamically, so the amount required rises in proportion to the amount of data being manipulated. In release 15 tests show that memory required to read in and display data is approximately 750MBytes per 1,000,000 elements in the model in the 64 bit version. Operations such as Model Merge, Spotwelding and Contact checking can easily double these requirements, therefore for "real world" usage, and to allow a margin for future expansion, we would recommend the following:

- **64 bit version**: 1.5 Gbytes of memory for each 1,000,000 nodes and elements in the model.

Please contact Oasys Ltd if you would like advice about specifying a computer for PRIMER usage.

Output devices

The code supports the following graphical devices:

- Open GL 3-D generic graphics - this is the recommended choice on all computer platforms.
- X_Windows Colour, greyscale & monochrome is still supported on a "legacy" basis, but no longer developed.
- Postscript (Adobe 2.0) Laser driver.
- Portable Document Files (PDF)
- Bitmap (.bmp), JPEG (.jpg), GIF (.gif) and PNG (.png) image files

There is also a "no graphics" (tty) mode that can be run in batch, but this has only a limited subset of the full command set.

Revision history

Released with Primer 16.0

Rev 0 March 2019

Released with Primer 15.0

Rev 0 May 2018

Released with Primer 14.0

Rev 0 March 2017

Released with Primer 13.0

Rev 0 March 2016

Released with Primer 12.0

Rev 0 October 2014

Released with Primer 11.0

Rev 0 March 2013

Released with Primer 10.0

Rev 0 May 2011

Released with Primer 9.4:

Rev 0 Nov 2009

Released with Primer 9.3 rc2:

Rev 0 January 2008

Released with Oasys Ltd. LS-DYNA environment 9.2 Software suite:

Rev 0 March 2006

Released with Oasys Ltd. LS-DYNA environment 9.1 Software suite:

Rev 0 November 2004: Release of version 9.1 software

Released with Oasys Ltd. LS-DYNA environment 9.0 Software suite:

Rev 0 November 2003: Release of version 9.0 software

Released with Oasys Ltd. LS-DYNA environment 8.1 Software suite:

Rev 0 April 2001: Release of version 8.1 software

- New Version 8.1 features;

- Manual reorganised, and converted to HTML for web browser access from inside code.

Releases with Oasys Ltd. LS-DYNA environment 8.0a Software suite:

Rev 0 October 2000 Interim release of version 8.0a software

Releases with Oasys Ltd. LS-DYNA environment 8.0 Software suite:

Rev 0 January 2000 Initial release of version 8.0 software

Releases with Oasys Ltd. LS-DYNA environment 7.1 Software suite:

Rev 0 October 1998 Initial release of version 7.1 software

- New Version 7.1 features;

- Manual reorganisation for improved layout

Releases with Oasys Ltd. LS-DYNA environment 7.0 Software suite:

- Rev 0 June 1997 Initial release of manual for version 1.0 software.
- Rev 1 March 1998 Updated for pre-release 7.0b software
- Rev 2 April 1998 Updated for official 7.0b release

Text conventions used in this manual

Typefaces

Three different typefaces are used in this manual:

Manual text	This typeface is used for text in this manual.
`Computer type`	This one is used to show what the computer types. It is also used for equations, keywords (eg **`*PART`**) etc.
`Operator type`	This one is used to show what you must type.
Button text	This one is used for screen menu buttons (eg **APPLY**)

Notation

Triangular, round and square brackets have been used as follows:

Triangular To show generic items, and special keys. For example:

```
<list of integers> <filename> <data component>

<return> <control Z> <escape>
```

Round To show optional items during input, for example:

```
<command> (<optional command>) (<optional number>)
```

And also to show defaults when the computer prompts you, eg:

```
Give new value (10) :

Give model number (12) :
```

Square To show advisory information at computer prompts, eg

```
Give filename: [.key] :

PRIMER_MANAGER >>> [H for Help] :
```

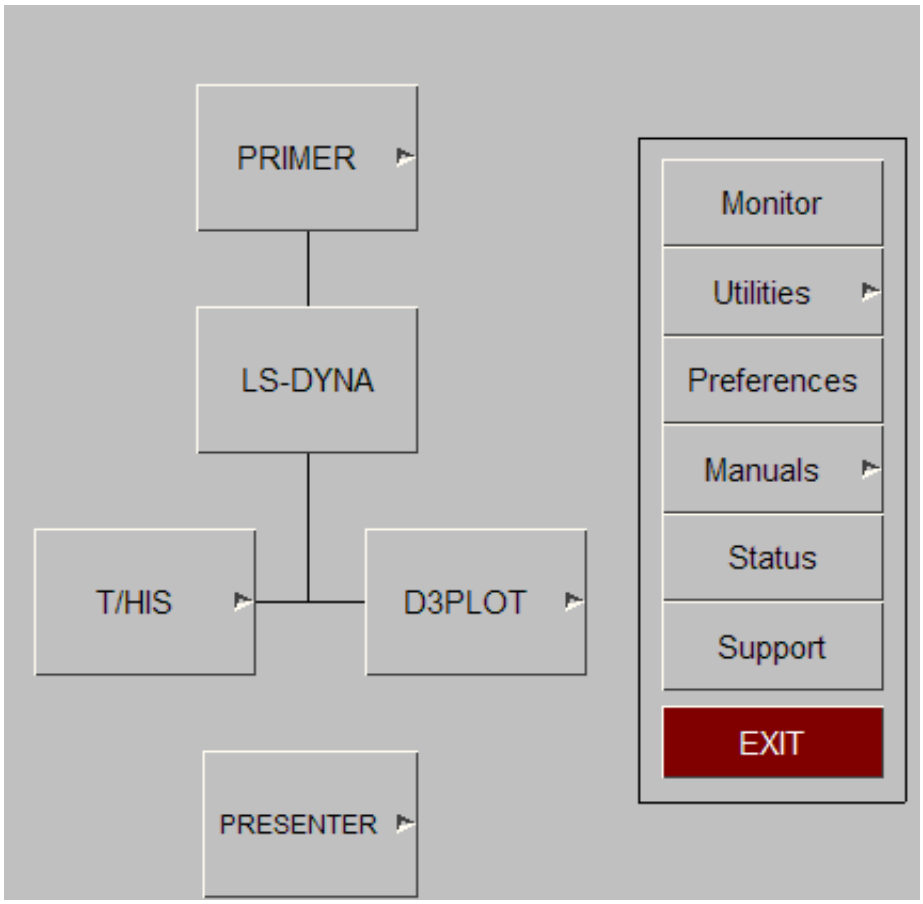
Also to show implicit commands, eg

```
[ORIENT] TRANSLATE <entity> <number of values>
```

1 Running PRIMER

1.1 Starting the code

PRIMER is run from the PRIMER button in the Shell:



Users who are running on a device without a window manager should use the `PR` option in the command-line Shell. This will mean that the programme runs in command-line mode only, ie no graphics, which may be suitable for batch usage.

If your system has been customised locally you may have to use some other command or icon: consult your system manager in this case.

PRIMER may be run both locally on your machine and remotely, in client/server mode, using the remote machine (client) to display on the local screen (server). For remote usage it will be necessary to set the host's **`DISPLAY`** environment variable to point to the server, and to enable remote display on the server: see section 1.3 if you have problems doing this.

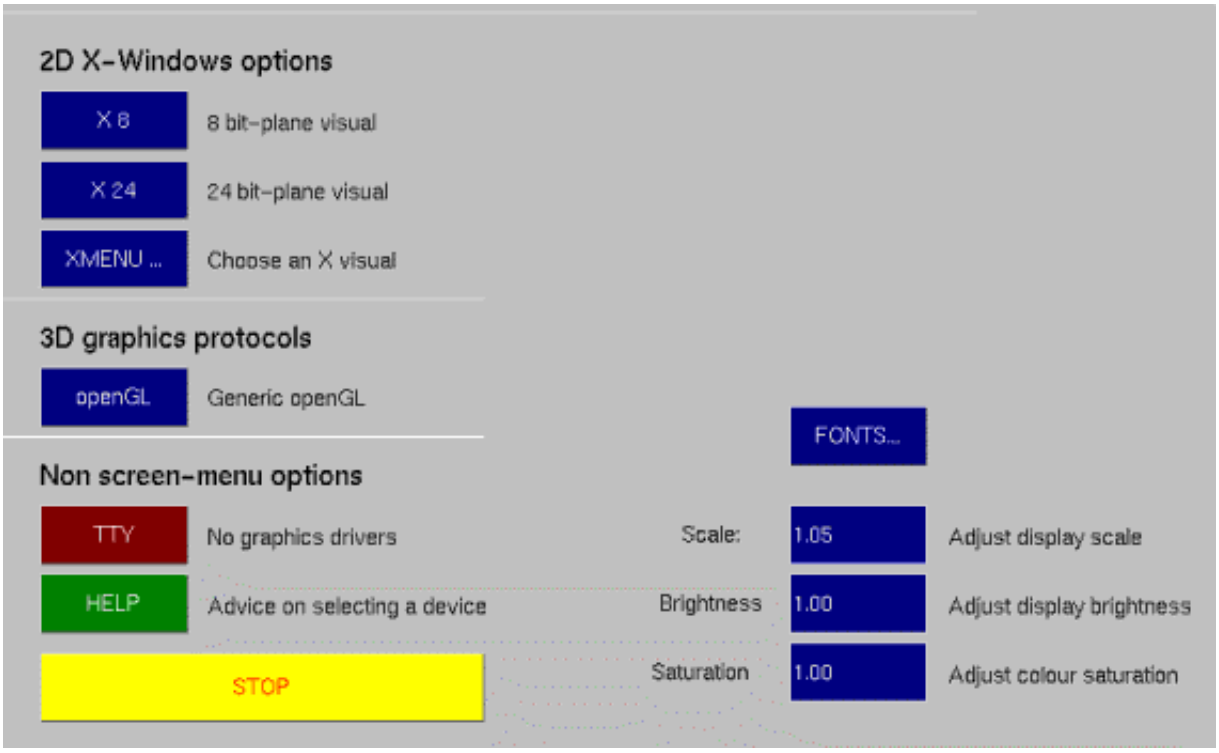
1.2 Selecting a graphics device

This panel is not normally mapped.

If you see it when you start PRIMER it means that the command which launches the code, or the oa_pref giving preferences, needs to be configured to define a default graphics device.

On Windows	This panel is not normally mapped, and PRIMER starts under OpenGL automatically

On Unix / Linux When PRIMER starts it will normally be configured to use OpenGL graphics automatically. If, exceptionally, it is not you will see the device selection panel:



The actual devices available will depend on your machine type and the graphics options that have been installed. Most workstations will provide both X11 and OpenGL graphics, but older machines may have a more limited range of options.

OpenGL Selects the 3D OpenGL device, using hardware acceleration if available. This is the best choice of all if it is available on your system.

X8 Selects an X11 visual with 8 bit-planes. Shading will be limited, as only 256 colours will be available

X24 Selects an X11 visual with 24 bit-planes. Over 16 million colours will be available, so shading will be better - this is the recommended choice under X-Windows.

XMENU Lets you choose from all the visuals available on your system. Use only if one of the standard options doesn't work - you may need to ask Oasys Ltd for advice.

Note: If you always want to start PRIMER with the same graphics device you can do so by defining a "**`-d=<device>`**" command-line argument. This will bypass the device selection panel above. See Command Line Arguments below for more details. The same effect can be obtained by setting the preference **`graphics_type`**

The other command options on this panel are:

FONTS Allows you to choose font typefaces, style and size. (May also be set interactively from the Options >, Menu Attributes pulldown window.)

TTY Invokes text-only mode with no graphics or menus

HELP Gives online advice about using this panel

STOP Stops the PRIMER session.

There are also three settings that control the appearance of the screen menu interface (but not the graphical images of your model). These are:

Scale Controls the effective scale of the display used for the menu interface.

The menu system for PRIMER was designed for a high resolution (1280 x 1024) display of at least 17" size. On smaller screens and/or lower resolution displays it can be a bit over-sized leading to some panels being too small for their contents. The "scale" value can be used to factor the physical size of the display: values greater than 1.0 will make it appear to be larger, so text and buttons will shrink making more of them fit into panels.

This variable may also be set using the environment variable **DISPLAY_FACTOR**. Valid settings being a number in the range 0.5 to 2.0, or the word "automatic". For example:

`setenv DISPLAY_FACTOR 1.2` (C shell syntax)

`DISPLAY_FACTOR=automatic; export DISPLAY_FACTOR` (Bourne shell)

The "automatic" setting calculates a factor based on your physical screen size: you can still overwrite it in this front panel.

(May also be set interactively from the Options >, Menu Attributes pulldown window.)

Brightness Controls screen menu colour brightness.

On some displays the colours in the screen menu come out garishly bright. This value may be set in the range 0.0 to 1.0 to control this brightness: 1.0 being light, 0.0 being dark. It too may be set as above with the environment variable **DISPLAY_BRIGHTNESS**.

Saturation Controls screen menu colour saturation.

As with brightness the colour saturation of the screen menu may be set in the range 0.0 to 1.0 (totally grey to fully saturated colours). Again there is an environment variable **DISPLAY_SATURATION** which may be used to set this globally as described above.

You may need to experiment a bit to find the right values for your particular display but, once found, the environment variables are probably the best way to set them. There is the additional advantage that they will also apply to T/HIS and D3PLOT.

NOTE: From PRIMER version 9.2 onwards the Display Factor, Font attributes, and left-handedness may be set interactively using the Options >, Menu Attributes pulldown menu

A complete listing of all possible command line arguments, environment variables and "oa_pref" file configuration options may be found in Appendix XIII

1.3 If PRIMER will not start in "screen-menu" mode on your display

You may be running on a device with a window manager, but still only get the command-line prompt (and probably no menu driven Shell either).

On a system running an X11 based window manager, generally Unix, this is almost certainly because of one or both of the following setup errors:

- The DISPLAY environment variable has not been set up, or has been set incorrectly. This tells the X11 window manager where to place windows, and it must be set to point to the server's screen. Its generic setup string is:

  `setenv DISPLAY <hostname>:<display number>` (C shell syntax)

  Where <hostname> is the server's name or internet address, for example:

  `setenv DISPLAY :0` (Default display :0 on this machine)

  `setenv DISPLAY tigger:0` (Default display :0 on machine "tigger")

  `setenv DISPLAY 69.177.15.2:0` (Default display :0, address 69.177.15.2)

  You may have to use the raw network address if the machine name has not been added to your **`/etc/hosts`** file, or possibly the "yellow pages" server hosts file.
- Your machine (strictly the X11 "server") has not been told to accept window manager requests from remote machines. This is usually the case when you are trying to display from a remote machine over a network, and you get the message similar to:

```
Xlib: connection to "<hostname>" refused by server

Xlib: Client is not authorised to connect to server
```

In this case go to a window with a Unix prompt on your machine, and type:

```
xhost +
```

Which tells your window manager to accept requests from any remote client. It will produce a confirmatory message, which will be something like:

```
access control disabled, clients can connect from any host
```

If you are still having problems getting graphics to work...

A more detailed explanation of networked graphics, hardware, X11 and associated topics is given in Technical Topics to do with Graphics. It contains a trouble-shooting guide, and a full description of how to use X11-based graphics. If you are still having problems after reading that please see your system manager, or contact Oasys Ltd for advice and help.

1.4 Running PRIMER in "Text-Only" mode.

All the previous sections assume that you want to use PRIMER's Graphical User Interface (GUI) "menu system". However PRIMER can also run with no graphics, although this is usually only the case when it is used in batch or from a script file.

This is referred to as "Text-Only" mode, and is invoked by specifying device "TTY" or "Batch" (there is no difference between the two).

No interactive graphics are available, nor can bitmaps or laser files be generated. Only the limited dialogue command set (see Appendix X11) can be used, which restricts the functionality of the code to the following operations:

- Read, Write, Copy and Delete models
- Orient functions (translate, rotate, etc) on a restricted range of object types
- Data Transfer (moving properties from model A to B)
- Bill of Materials (BOM) operations
- Assign Mass (massing up) operations
- Material database import capability
- Model summary

The full Text-Only command list is described in Appendix X11

All Text-Only commands may also be used in command files - see Using Command Files.

1.4.1 Starting PRIMER in Text-Only mode.

This may be done in one of two ways:

- **By adding the command line option `"-d=batch"` or `"-d=tty"` to the execution sequence that invokes PRIMER.**

  This is normally done only in shell scripts intended for batch running. The effect is to start PRIMER with no graphics or menu interface, and all command input and output is assumed to be via the controlling terminal. This and other dialogue arguments are discussed in more detail in Appendix X11.
- **By selecting device TTY on the PRIMER front device selection panel.**

  This will close down the graphics window and revert to terminal input and output as above.

1.4.2 Typing in commands when running in Text-Only mode.

Dialogue commands are typed at the **Primer_Manager >>>** prompt, and then at subsequent sub-menu prompts.

The complete dialogue command list, structure and syntax is given in Appendix X11, but a brief description is:

- The command tree is hierarchical, with **Primer_Manager** as the top level.
- Preceding any command with "**/**" (forward slash) means "return to top level before executing it".
- Any command can be aborted with "**q**" (for Quit)
- Help can be obtained anywhere with "**h**" (for Help)

Commands may be given in sequence on a line, or individually in response to each sub-menu prompt. For example the following two command sequences are identical in effect: they both read LS-Dyna keyword file "test_model.key" into model #2.

Commands on individual lines	**PRIMER_MANAGER >>> [H for Help]** `READ` **READ >>> [H for Help]** `DK` **Give LS-DYNA keyword filename (.key) [H for Help]:** `test_model` **Give model number: (1)** `2`
Concatenated commands on a single line	**PRIMER_MANAGER >>> [H for Help]** `READ DK test_model 2`

When giving commands you need only only type enough letters for the command to be unique in that context, subject to a minimum of 2 letters. In the example above **READ** could have been abbreviated to **RE.**

The full command-line syntax and a list of commands is given in APPENDIX XII: Dialogue (typed in) Command Syntax

1.4.3 Using "command files" and macros

As well as typing in commands you can run pre-built files of commands, referred to as command files.

These are invoked using the "**-cf=<*filename*>**" option on the command line, and they operate as follows:
- The commands in <*filename*> are executed in sequence, exactly as if they had been typed in.
- At the end of <*filename*> one of two things happens:
  - If "**-exit**" was also specified on the command line then PRIMER terminates normally
  - Otherwise it reverts to interactive input at the command prompt

Therefore a typical batch invocation for PRIMER (in Unix) might be something like:

```
$<pathname>/primer14_x64.exe -d=batch -cf=my_command_file -exit
```

This will run in text-only mode, execute the commands in file "my_command_file" and terminate normally at the end of that file.

It is also possible to run "macro files" of stored commands from the command line. Macros are described in section 6.27MACROS, and running them from the command line via the "**-ml=<*filename*>**" command is described in Appendix XIII

1.5 Command Line arguments

This section has been moved to Appendix XIII Command line Arguments.

2 Using Screen Menus

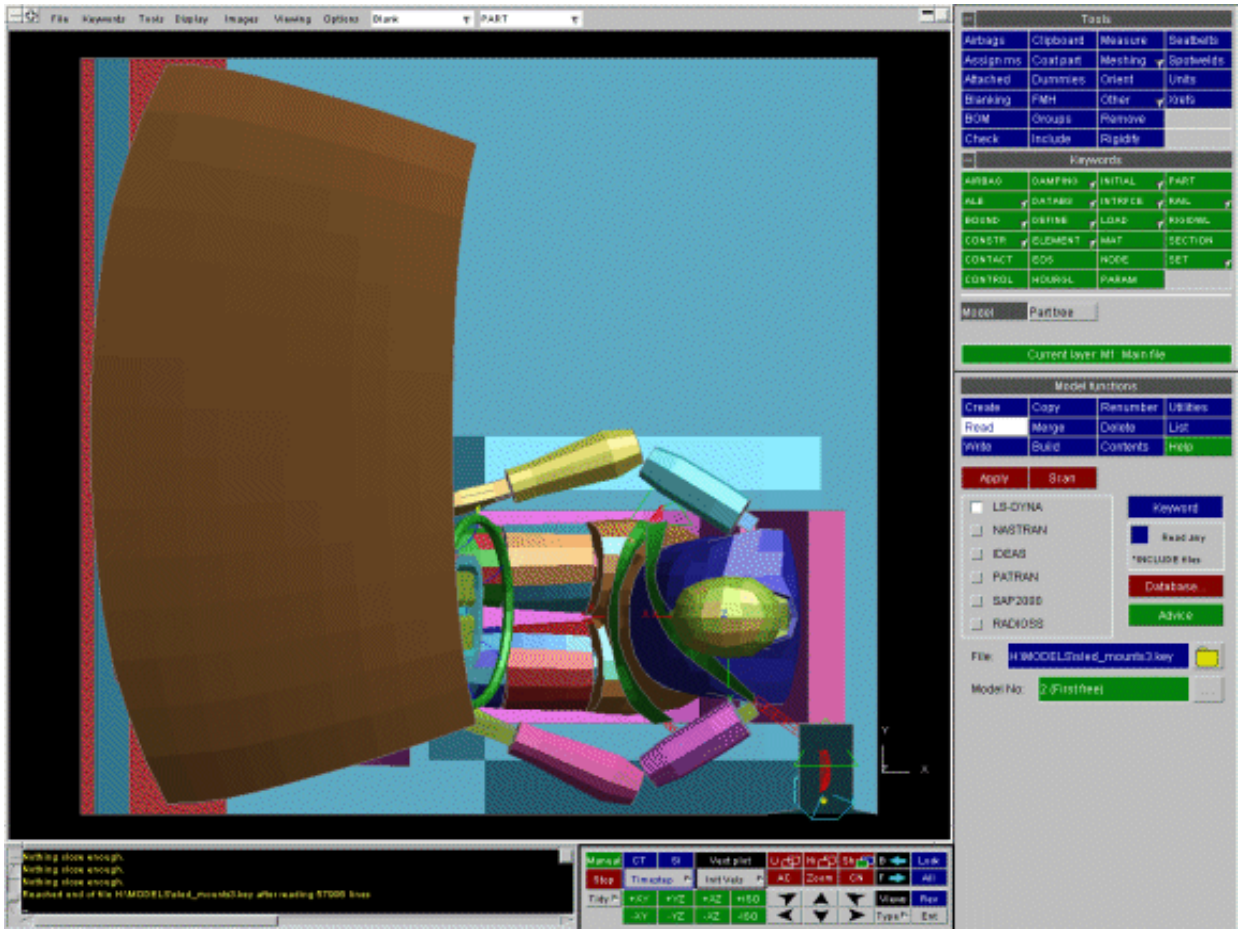
2.1 Basic screen menu layout
2.2 Mouse and keyboard usage
2.3 Dialogue input
2.4 Window management
2.5 Using file selection boxes
2.6 Obtaining Help and Advice
2.7 Error and Warning messages.
2.8 Dealing with crashes
2.9 Quick Pick
2.10 Using Parameters in edit panels
2.11 Formulae in edit panels
2.12 Operational Hierarchy
2.13 Selecting Entities for Operations

2.0 Using the PRIMER screen menu system

PRIMER has both a screen-menu and a "command-line" user interface, but the latter gives only limited functionality. It is assumed that interactive users will be using the menu-interface, which can be driven entirely using the mouse.

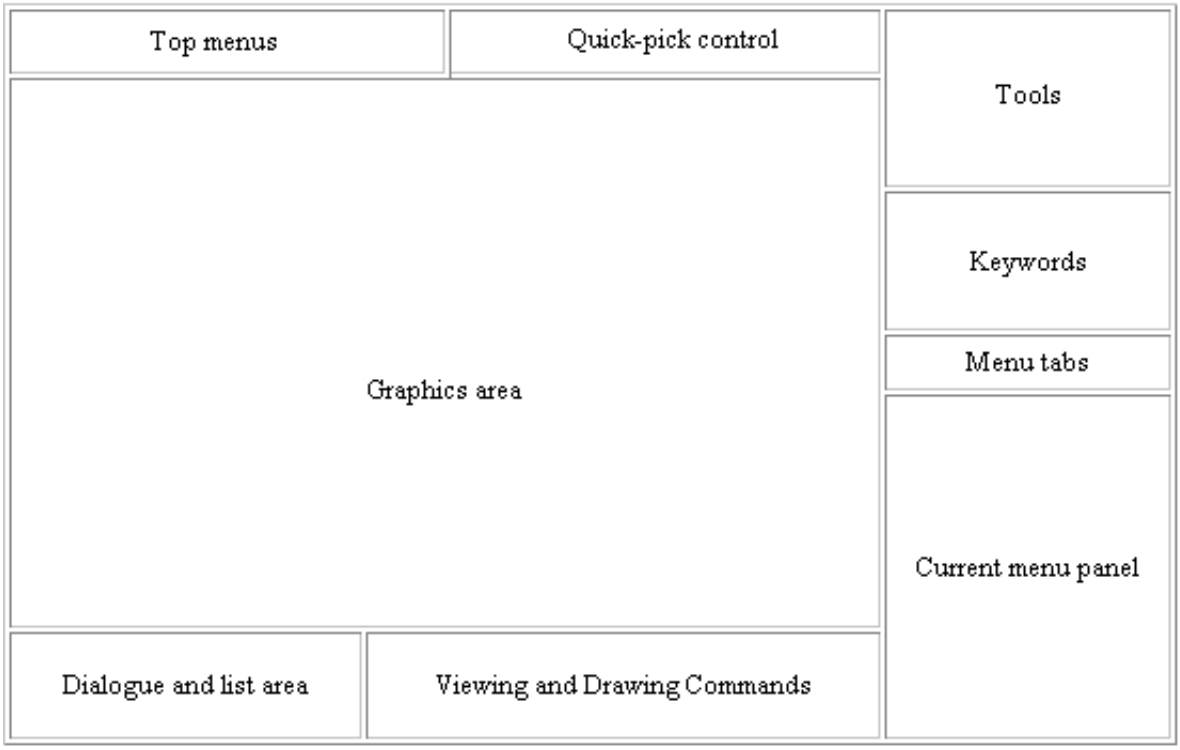
2.1 Basic screen menu layout

PRIMER runs within a single window, owned by the window manager, which has several sub-windows inside it. A typical PRIMER session will look like this:



The various sub-windows always exist within the master window, and may be moved and resized at will inside it. They will keep their relative size and position as the master window is changed in size and/or shape, and will reappear after the main window is de-iconised.

The default layout of the main sub-windows is as follows:



These windows cannot be dismissed. A brief description of their functions is:

- "Top menus" This is included in the same window as the Graphics area. It allows access to some of the basic options.
- "Quick Pick control" controls the mouse action when applied within the graphics area.
- "Graphics area" Is where graphics are drawn.
- "Tools" This menu provides access to many different functions available in PRIMER.
- "Keywords"This provides access to all the Keywords supported by PRIMER.
- "Menu Tabs" These control which option is displayed in the current menu panel. Model and Part Tree will always be available in addition to selected options.
- "Current Menu Panel" Displays the menu for the option currently selected by the menu tabs.
- "Dialogue & list" Allows "command-line" input and output, also provides a listing area for messages.
- "Viewing and Drawing Commands" provides all aspects of view control: direction, perspective, scale, etc and contains the drawing commands and their settings.

While you are free to re-position these master windows it is recommended that you keep to this default layout. This is because when further sub-windows appear their position and size is designed assuming this layout, and aims to obscure as little useful information as possible. The TIDY command in the "Viewing and Drawing" box will restore the screen layout to this default state.

2.2 Mouse and keyboard usage for screen-menu interface

Most screen-menu operations are driven with the left mouse button only, but there are exceptions:

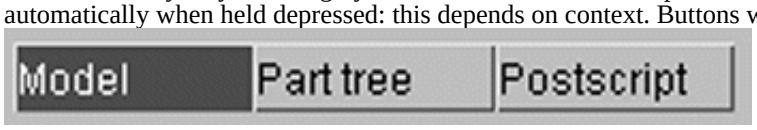
- "Popup" menus are invoked with the right mouse button; those in the Top menus, Tools and Keywords areas can also be invoked with the left mouse button.
- Text in the dialogue area and text boxes requires keyboard entry;
- Text strings saved in the cursor "cut" buffer may be "pasted" into dialogue areas and text boxes using the middle mouse button.
- Dynamic viewing (<meta key> + <mouse button>) uses the three mouse buttons to distinguish different modes. Section 9 describes viewing.
- Some specialist functions use different mouse buttons for particular functions.

- Screen-picking uses:
  Left button to select;
  Middle button to reject most recent selection
  Right button to reject what is under the cursor.
- See also section on Quick Pick

2.2.1 Buttons

Screen buttons are depressed by clicking on them, but action only takes place when the mouse button is released, so it is safe to drag the (depressed) mouse around the screen.

Buttons may be set (ie depressed) by PRIMER itself, for example the "**MODEL**" one below, to indicate that this option is in force. They may also be greyed out, to indicate that the option is not currently available. Some buttons repeat automatically when held depressed: this depends on context. Buttons with "..." after them will invoke sub-menus.



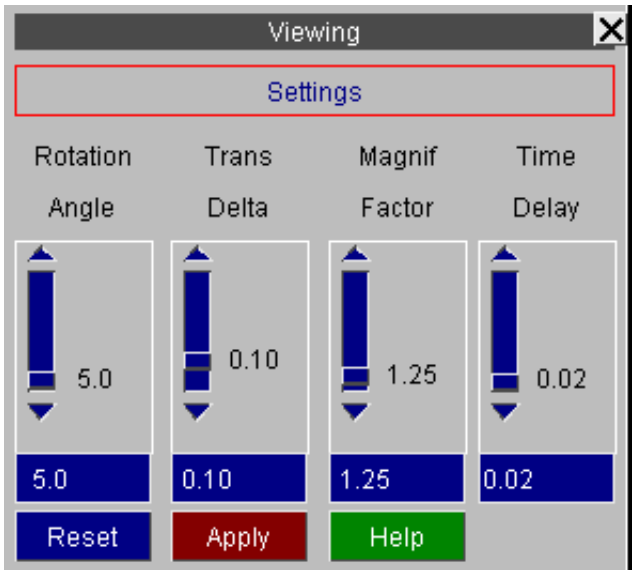
The primitive "widgets" in the menu interface are used as follows:

"Popup" window invocation: Buttons with an ">" symbol may be selected normally with the left mouse button, but if the *right* mouse button is depressed over them it will invoke a "popup" window. Move the cursor into this window to make a selection, or move elsewhere and click a button to deactivate the popup.

2.2.2 Sliders

Sliders are moved by clicking on the slider button itself and then dragging it to a new position.

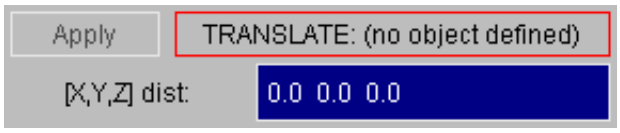
They may also be moved automatically by clicking on, and holding down, one of the arrows at either end. Using the left mouse button for this advances the slider by 1 unit, the middle button by 10, and the right button by 100.



2.2.3 Text boxes

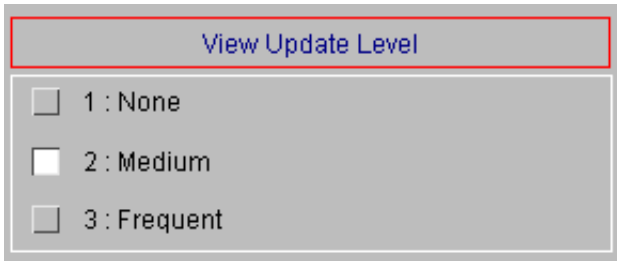
To enter text in a text box: first make it "live" by clicking on it, then type in text, then type `<return>` to enter the string. Clicking on a "live" box for a second time is exactly the same as typing `<return>`, so clicking twice on a box effectively enters its current contents. You can use the left and right arrow keys for line editing within a box: text entry takes place after the current cursor position. Control U (^U) will delete the entire text box contents.

You can "drop" the current X-Windows cut/paste buffer contents into a text box with the middle mouse button, just as you would in a shell (terminal) window.



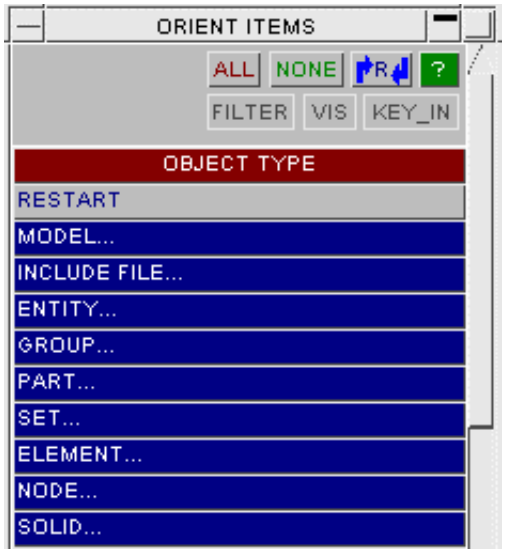
2.2.4 Radio boxes

A "radio" set is provided where only one selection is possible from a range of options. In this example the view "update" frequency has been set to level 2. To select click anywhere on the row of the relevant option, any previously selected item will be deselected.



2.2.5 Menu selections

Menus of items are used when you need to make one or more selections from a (potentially) long list. Click on the row you want to select: clicking on a row that is already selected will have the effect of deselecting it.



Where selecting more than 1 item would be valid you can "drag" (click, hold down and move) down the menu to select multiple items. Alternatively the <click> (start of range) .. <shift><click> (end of range) method (cf Windows) may be used.

When the list is too long to display in the window you can use the vertical scroll-bars to move up and down it. A mouse scroll wheel can also be used to move up and down in these panels. The filter button allows a subset of the selected entities to be offered, e.g. only those parts of a particular material type. See section 6.2 for more information on filtering. The menus are refreshed automatically after creation, editing or deletion of data; alternatively the [R] button can be used to refresh the button.

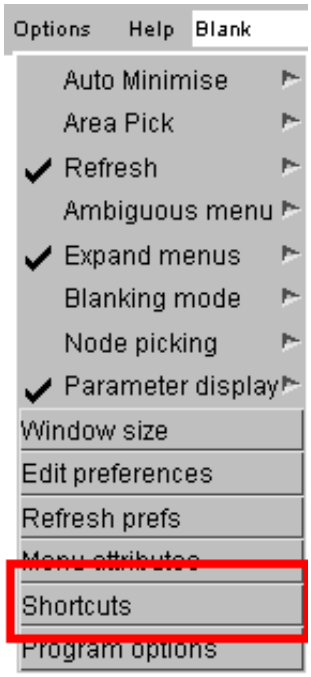
By default menus will expand horizontally when you move the mouse into them in order to show more of their contents. This is described in section 2.4.4.3 below

2.2.6 Shortcut keys

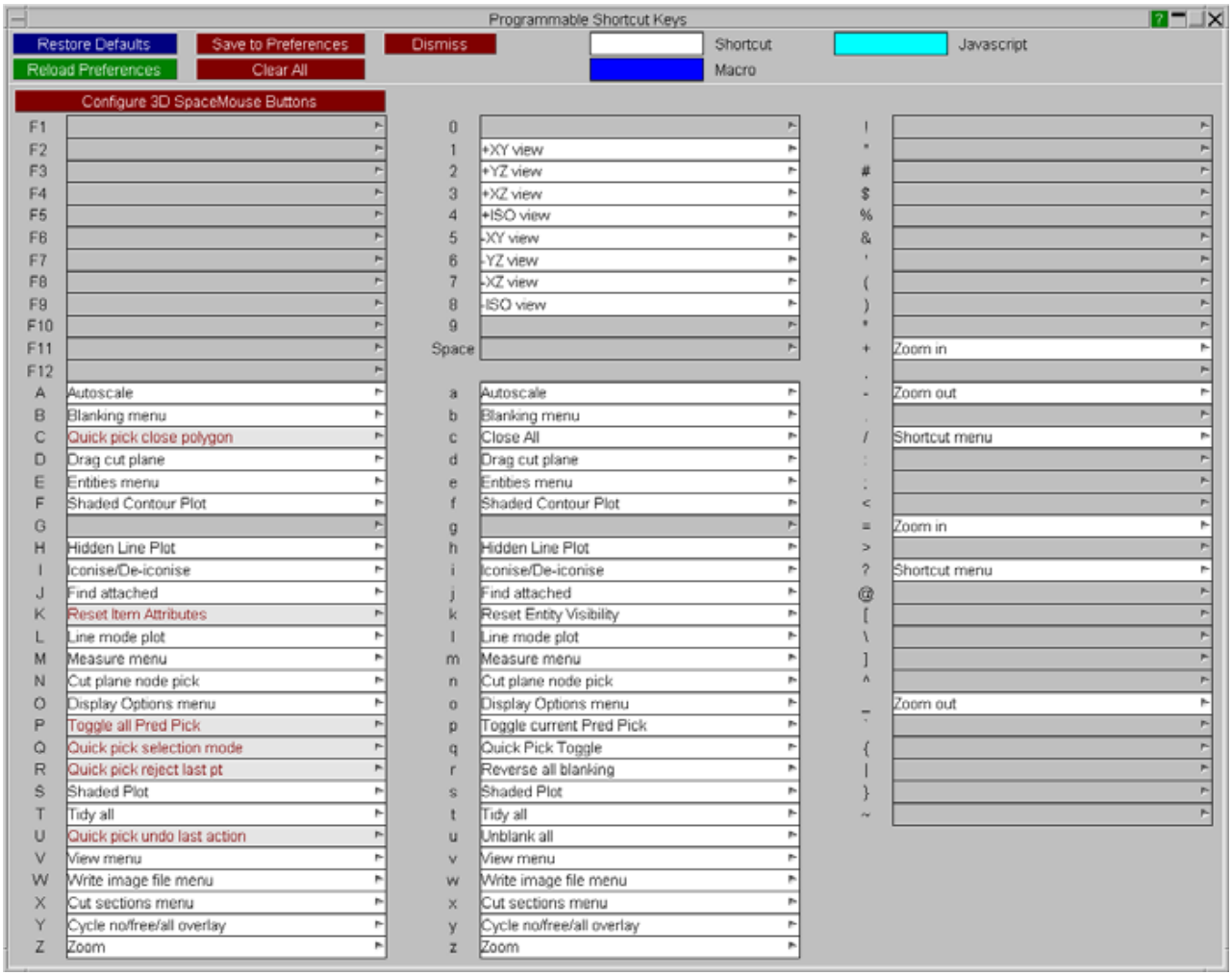
Some panels and actions can be accessed through pre-programmed shortcuts and from v9.4 the keys they are assigned to are customizable.

In v9.4 a number of new pre-programmed shortcuts have been added, including the top menu panels, all the contour buttons and the Lock and Centre buttons. Javascripts and Command Files can also be assigned to a key.

A listing of the available shortcuts and the keys they are assigned to can be brought up by pressing either the '?' key (by default) or accessing it through the Options top menu.

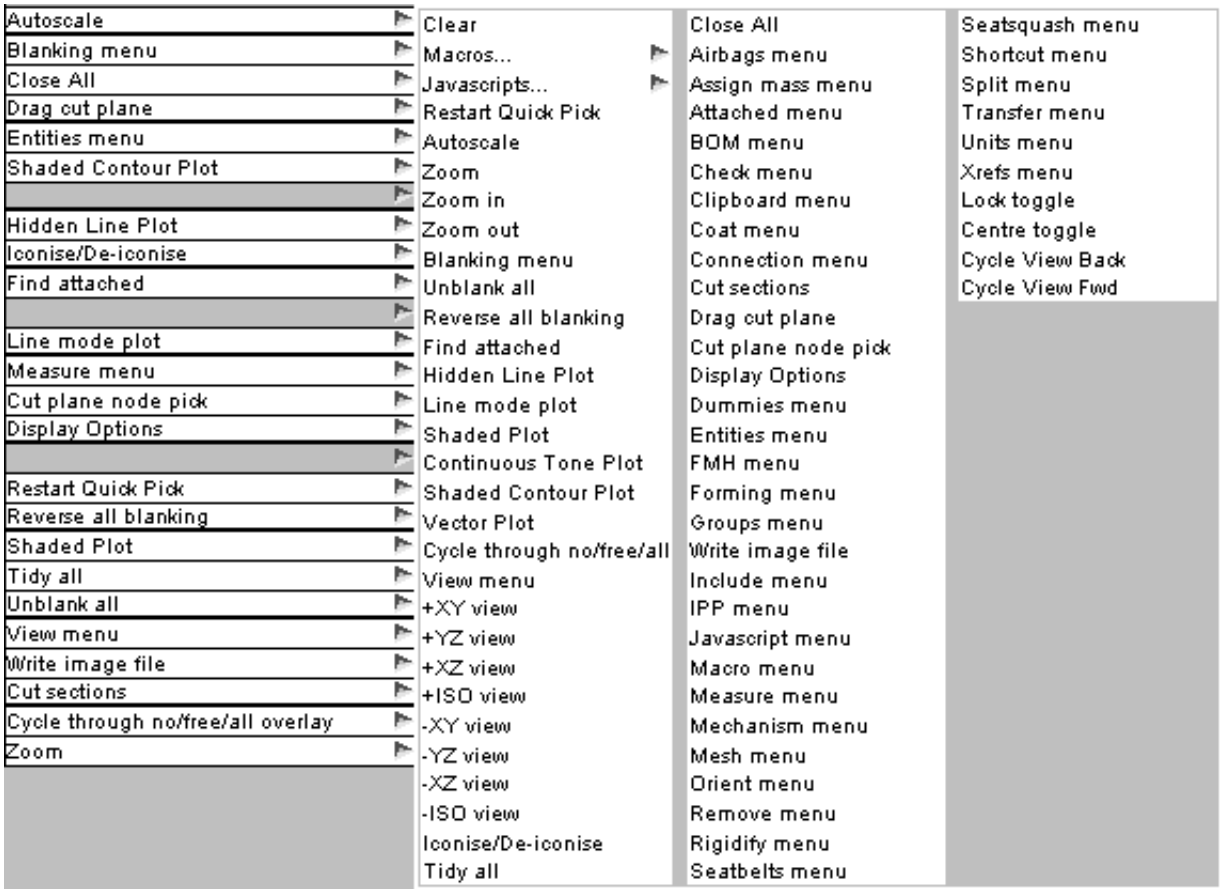


This will bring up a panel, from which you may assign the shortcuts, Javascripts and Macros/Command Files to the keys. Note that upper and lower case letters can be assigned different shortcuts.


Programmable Shortcut Keys
Restore Defaults
Save to Preferences
Dismiss
Shortcut
Javascript
Reload Preferences
Clear All
Macro
Configure 3D SpaceMouse Buttons
F1
F2
F3
F4
F5
F6
F7
F8
F9
F10
F11
F12
A Autoscale
B Blanking menu
C Quick pick close polygon
D Drag cut plane
E Entities menu
F Shaded Contour Plot
G
H Hidden Line Plot
I Iconise/De-iconise
J Find attached
K Reset Item Attributes
L Line mode plot
M Measure menu
N Cut plane node pick
O Display Options menu
P Toggle all Pred Pick
Q Quick pick selection mode
R Quick pick reject last pt
S Shaded Plot
T Tidy all
U Quick pick undo last action
V View menu
W Write image file menu
X Cut sections menu
Y Cycle no/free/all overlay
Z Zoom
0
1 +XY view
2 +YZ view
3 +XZ view
4 +ISO view
5 -XY view
6 -YZ view
7 -XZ view
8 -ISO view
9
Space
a Autoscale
b Blanking menu
c Close All
d Drag cut plane
e Entities menu
f Shaded Contour Plot
g
h Hidden Line Plot
i Iconise/De-iconise
j Find attached
k Reset Entity Visibility
l Line mode plot
m Measure menu
n Cut plane node pick
o Display Options menu
p Toggle current Pred Pick
q Quick Pick Toggle
r Reverse all blanking
s Shaded Plot
t Tidy all
u Unblank all
v View menu
w Write image file menu
x Cut sections menu
y Cycle no/free/all overlay
z Zoom
!
"

$
%
&
'
(
)
*
+ Zoom in
,
- Zoom out
.
/ Shortcut menu
:
;
<
= Zoom in
>
? Shortcut menu
@
[
\
]
^
_ Zoom out
`
{
|
}
~

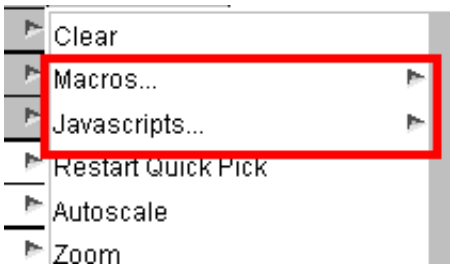
The image above shows the default key assignations, but each key may be programmed to use any of the predefined functions as shown in this screen-grab:



In addition the **Macro >** and **Javascript >** options permit user-defined scripts to be assigned to any key, so that users can create and assign their own functions.

Macros are described in section 6

Javscripts are described in section 10



Shortcut keys are effective when the mouse is in any PRIMER window ***except*** the dialogue box, in the latter they are interpreted as normal text input to the command-line interpreter.

From v11 onwards, there is an option to configure 3D mouse buttons through the shortcuts panel. Click on the **Configure 3D SpaceMouse buttons** to assign functions, macros and javascripts to buttons on a 3DConnexion 3D mouse. See section 9.4.4 for more information on 3D mice.

2.3 Dialogue input in the screen menu interface

A limited command-line capability (see Appendix X11) is preserved when PRIMER is running in screen-menu mode, and you are free to mix command-line and mouse-driven input at will. These may be typed into the dialogue box:

The dialogue box is also used for listing messages, warnings and errors to the screen. It can be scrolled back and forth (its buffer is 200 lines long) to review earlier messages. The following colours are used:

- Normal messages and prompts Yellow
- Text typed in by you White
- Warning messages and Error messages Red

2.4 Window management in the screen interface

Menus in PRIMER are either "docked" (appear in a fixed size and position in the Current Menu Panel) or "floating" (can be moved and resized).

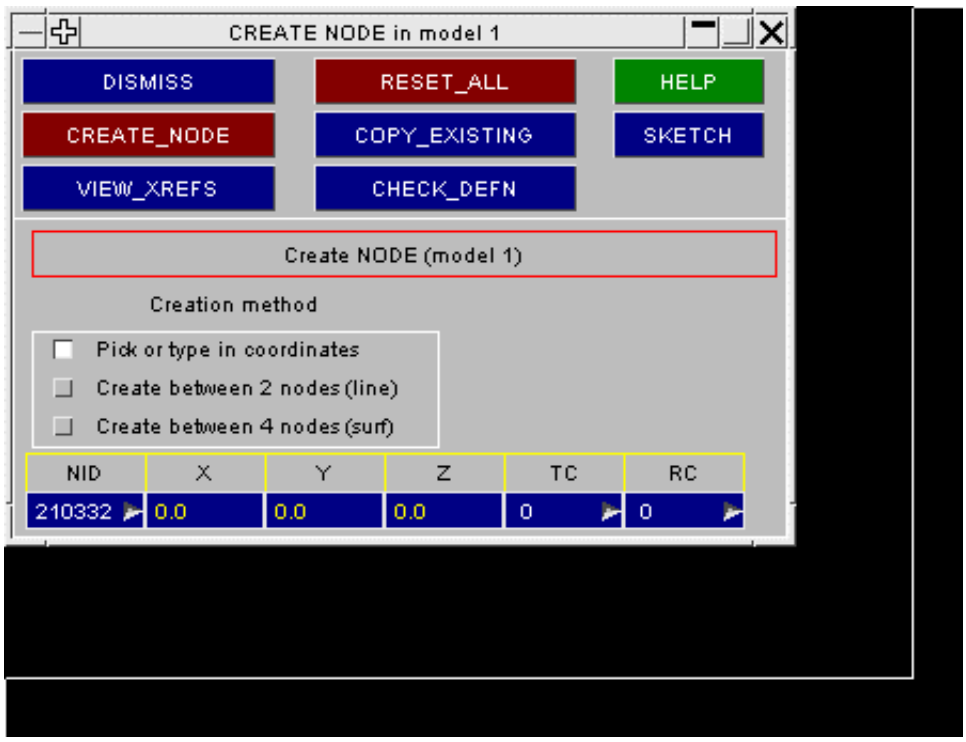
Moving, resizing and scrolling of windows is based on the conventions used in the Motif Window Manager.

To move a window (floating menus only): Click down on its title bar, then drag the window to where you want it to be. A "rubber-band" outline moves to show the window's current position.

To resize a window (floating menus only): Click on a border bar to move just that side, or on a corner bar to move both sides attached to that corner. Again, a rubber-band outline shows you the new shape. You can maximise a window using the square button at its top right, and iconise it using the minimise button next to this.

To scroll a window: If a window has become too small for its contents then horizontal and/or vertical scrollbars will appear. Click on a scrollbar slider and move it to the desired position, the window contents will scroll as you do so. Alternatively click on the arrows at either end of the scrollbar for timed motion in that direction.

This example shows a sub-window being resized:



The user has chosen to drag the bottom right corner out.

The "title bar" is the area where, in this example, it shows the name of the sub-window:

CREATE NODE in model 1

To dismiss a window, Either press **DISMISS**, or click the x in the top-right of the window or press ESC on the keyboard.

2.4.1 Popup menus for window management:

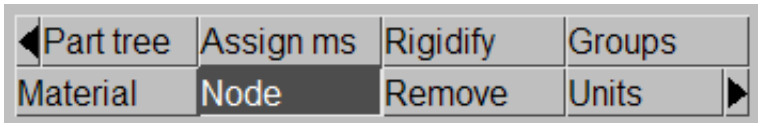
Clicking on the [-] button at the top left of a window invokes the popup menu for window management:

MAXIMISE	expands the window to its full size (in the case of the dialogue and graphic areas this is taken as the entire PRIMER window, for other sub-windows the minimum size such that no scroll bars are required).
MINIMISE	collapses the window to a bar. This will be positioned where the top right -hand corner of the window was. If a window has already been maximised the option to do so will be replaced by **RESTORE** or if minimised by **EXPAND**. These will undo the effect of maximisation and minimisation respectively.
RAISE	raises the window to the front of the "stacking order", obscuring any others.
LOWER	lowers the window to the bottom of the stacking order, allowing other to obscure it.
SAVE->BITMAP	saves this window (and its borders) as a "bitmap" (.bmp) file.
Copy->Clipboard	For "text" windows, ie dialogue input box and listing boxes, the complete text in the window is copied to the clipboard On Windows platforms only other windows are saved as a bitmap image on the clipboard. (This is not feasible on X11-absed window managers, typically Unix and Linux, because there is no common protocol for exchange of images.)

2.4.2 Use of Menu Tabs

Docked menus appear in the Current Menu Panel, and hence do not obscure the graphics area. Any number of such menus can be present concurrently; these are positioned on top of one another. Each menu has a corresponding tab in the Menu Tabs area, and can be bought to the fore by clicking on its tab.

The Model menu and Part Tree are always present in the tab list; other menus are invoked by the user. When there are more than eight menus open, the user can scroll through the open tabs using the left and right scroll tab buttons.



2.4.3 Iconisation of Menus

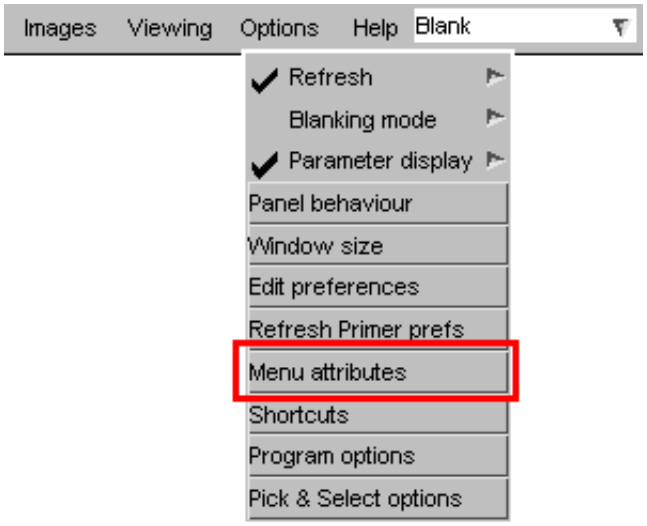
Menus can be iconised by clicking . Click to restore them.
Alternatively a list of options is produced by clicking on the button in the top-left corner.
Pressing **I** will iconise all windows or if restore them all if they are all already iconised
It is also possible from **Options** > **Auto Minimise** > to enable the auto minimise function. If this is set to on then whenever a screen picking option(other than Quick Pick is selected and the cursor is in the graphics area then all floating windows will automatically iconise.

In the Viewing and Drawing Commands box the Tidy menu (invoked by right-clicking)presents several options for handling menus. **Tidy All** iconises all floating menus and positions them in the top left of the graphics area (left clicking on **Tidy** invokes this function). **Minimise All** iconises the menus but does not move them.
Restore All and **Close All** restore and close all floating menus respectively.

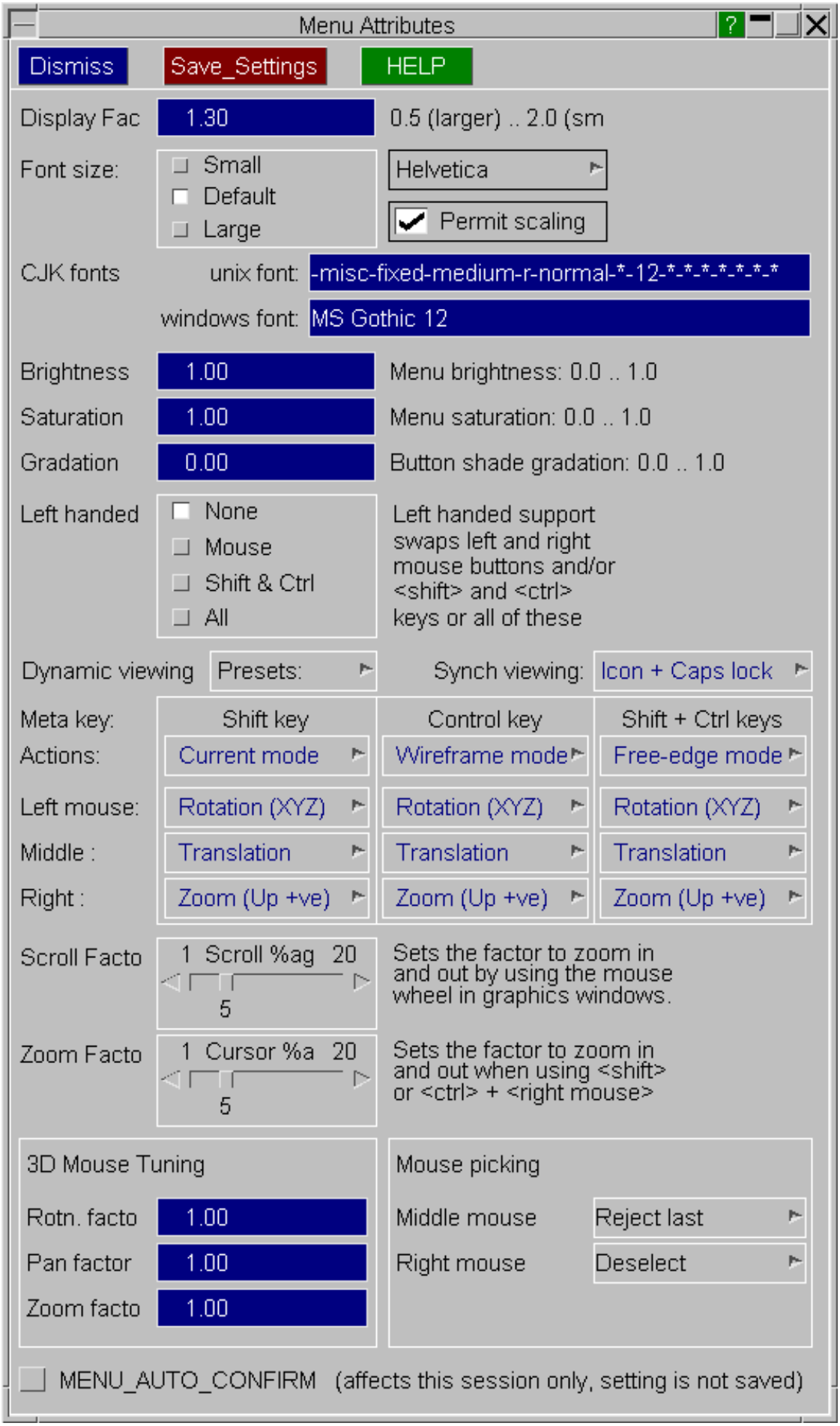
2.4.4 Customising the User Interface

2.4.4.1 **Menu Attributes**: Customising Menu size, fonts, dynamic viewing and handedness

As described in section 1.2 the scale of the menu interface, the font typeface and size, and also the left-handedness of the menu interface may be customised interactively using **Options > Menu Attributes.**



Gives the menu attributes panel:


Menu Attributes
Dismiss
Save_Settings
HELP
Display Fac
1.30
0.5 (larger) .. 2.0 (sm
Font size:
Small
Default
Large
Helvetica
Permit scaling
CJK fonts
unix font:
-misc-fixed-medium-r-normal-*-12-*-*-*-*-*-*-*
windows font:
MS Gothic 12
Brightness
1.00
Menu brightness: 0.0 .. 1.0
Saturation
1.00
Menu saturation: 0.0 .. 1.0
Gradation
0.00
Button shade gradation: 0.0 .. 1.0
Left handed
None
Mouse
Shift & Ctrl
All
Left handed support swaps left and right mouse buttons and/or <shift> and <ctrl> keys or all of these
Dynamic viewing
Presets:
Synch viewing:
Icon + Caps lock
Meta key:
Shift key
Control key
Shift + Ctrl keys
Actions:
Current mode
Wireframe mode
Free-edge mode
Left mouse:
Rotation (XYZ)
Rotation (XYZ)
Rotation (XYZ)
Middle :
Translation
Translation
Translation
Right :
Zoom (Up +ve)
Zoom (Up +ve)
Zoom (Up +ve)
Scroll Facto
1 Scroll %ag 20
5
Sets the factor to zoom in and out by using the mouse wheel in graphics windows.
Zoom Facto
1 Cursor %a 20
5
Sets the factor to zoom in and out when using <shift> or <ctrl> + <right mouse>
3D Mouse Tuning
Rotn. facto
1.00
Pan factor
1.00
Zoom facto
1.00
Mouse picking
Middle mouse
Reject last
Right mouse
Deselect
MENU_AUTO_CONFIRM (affects this session only, setting is not saved)

Display Factor	Is a factor on the overall scale of the display, lying in the range 0.5 to 2.0, default 1.0. Larger values make the display seem bigger to the software, resulting in smaller menu panels and fonts. Smaller values increase the size of menu panels, buttons and fonts, and can be useful for the visually impaired. This factor can be especially useful on "wide screen" displays with very asymmetric horizontal and vertical resolutions. The operating system *should* determine the physical size of the display correctly. However we have observed a few instances where this does not happen, the symptoms being that fonts and menus appear either far too big or too small and cannot be corrected by using **Display Factor**. In this situation you may need to tell PRIMER the physical dimensions of your display, and this process is described under "Setting the correct physical resolution for your display" in section 3.2 of the extra section on graphics.
Font size	Controls the size of fonts used in the menu interface (but not for graphics). This works independently of the Display Factor, allowing further fine-tuning of the appearance of the user interface.
Font Typeface	For most applications the default Helvetica (Arial on Windows) will suffice. But you can also choose Times or Courier, and Bold variants of all of these.
Font scaling	By default text in menu interface buttons can be scaled downwards to a smaller font size (if one exists) if it is too long for the button. This shows more characters, but it can look messy when the user interface has a mixture of font sizes. Turning font scaling off prevents this happening, giving a more consistent appearance. (However it is generally better to ajust the Display Factor in order to find a menu scale that gives consistent font sizes.)
`CJK fonts`	These are the ***C***hinese, ***J***apanese and ***K***orean unicode fonts used for extended typeface support in Javascript widgets. Separate descriptors are required on Unix/Linux and Windows because of the differences in the ways that fonts are handled on the two systems.
Brightness Saturation	These affect the overall brightness and also the colour saturation of the user interface. They both lie in the range 0.0 to 1.0, default 1.0.
Left-Handed support	By default PRIMER is set up for right-handed usage, which has influence on both mouse buttons and the keyboard "meta" keys: <shift> and <ctrl>. (The left and right meta keys have different functions during dynamic viewing: see section 9.4) You can swap the handedness of mouse and/or meta keys, which will reverse them in the left <=> right sense. **Note**: This swapping is local to PRIMER, and is applied after any system user interface configuration. So if you configure your computer to swap mouse buttons globally, then swap them here, the net effect will be to have unswapped buttons again!

Dynamic viewing

By default PRIMER uses the following dynamic viewing keyboard + mouse key actions:

Keyboard meta key	Viewing mode				Mouse button	Viewing action
<shift>	Normal	}		{	Left	Rotate in XY or Z
<ctrl>	Wireframe	}	+	{	Middle	Translate
<shift + ctrl>	Free edge	}		{	Right	Zoom (+ve upwards)

However different users have different tastes, and users who swap between different applications find it easier if they behave in similar ways. Therefore the following [permutations are available:

Viewing mode, may be assigned to keyboard meta-key(s) (ie <shift>, <ctrl> or <shift + ctrl>

Normal will use the current display mode

Wireframe only the line vectors in the current display mode

Free-edge special "free edge lines only" display mode

Dynamic rotation options, assigned to mouse buttons

Rotate XYZ traditional D3PLOT behaviour, rotates in XY if cursor's initial position is in centre 2/3rd of screen, otherwise about Z

Rotate XY rotates about screen XY only, regardless of where the cursor's initial position

Rotate Z rotates about screen Z only, regardless of cursor initial position

Rotate Sphere free rotation about any of XYZ, like grabbing a point in a virtual sphere and dragging it

Dynamic translation options, assigned to mouse buttons

Translate model follows cursor movement in screen XY plane

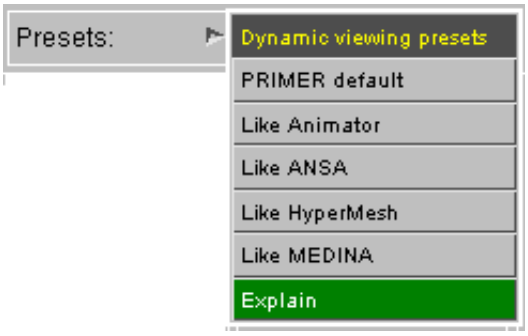
Zoom options, assigned to mouse buttons

Zoom (up +ve) up and to the right enlarge, down and to left reduce

Zoom (down +ve) down and to the right enlarge, up and to left reduce

Presets

These preset options configure PRIMER's dynamic viewing controls to operate in a similar way to those of the listed programmes. The descriptions "Like (program name)" are given only for ease of reference to certain combinations of key and mouse buttons used for dynamic viewing control.



ANIMATOR is a product of GNS mbH
ANSA is a trademark of BETA CAE systems SA
HYPERMESH is a registered trademark of Altair Engineering, Inc.
MEDINA is a registered trademark of T-Systems GmbH

The configurations these produce may not match exactly the actions in the given application, but they are the best that can be achieved at the present time with the options available.

Scroll factor

Determines the rate at which using the mouse scroll wheel to zoom in/out changes the image magnification factor. Smaller values will act more slowly, and larger ones more quickly - it is best set by experiment.

Zoom factor	Determines how rapidly the <meta key + mouse key> dynamic zoom operations above work. Again this is best set by trial and error.
3D Mouse tuning	Factors that are applied to translations/rotations when using a 3D mouse produced by 3DConnexion.
Mouse Picking	Setting for the middle and right mouse button action during picking. Options available are: Apply Selection - Apply the current selection (for example when picking nodes to create a CONSTRAINED_NODAL_RIGID_BODY, this option will create the entity (this can speed up the process of creation of many entities through picking). Reject Last - Reject the most recent selection (middle mouse button default). Deselect - Deselect items selected, either individually or by area (right mouse button default).
MENU_ AUTO_ CONFIRM	This is a special setting designed mainly for "batch" style usage, and it controls how "popup" windows that normally wait for acknowledgement from the user should respond. If it is switched on then these windows will assume that the user has clicked the default action (usually "OK") and continue operation without waiting. This can be useful when replaying scripts, but it is not recommended for normal interactive usage.

Saving Menu Attributes settings

The attributes above may be saved in the "oa_pref" file by using **Save_Settings**. Subsequent sessions of Primer will pick these up and re-apply them.

For backwards compatibility these attributes may also be set using environment variables as described in Appendix XIII. Where conflicting settings exist those in the "oa_pref" file generated by the panel above (or by hand) will "win".

Note: Oasys Ltd. LS-DYNA environment potentially reads four "oa_pref" files when an application starts, in the following order:

(1) The OA_ADMIN directory (if present)
(2) The OA_INSTALL directory (where the software is installed)
(3) Your OA_HOME directory (by default $HOME on Unix/Linux, and %USERPROFILE%, typically C:\Documents and Settings*user_id*, on Windows)
(4) The current working directory (typically "Start in" directory on Windows)

Save_Settings in this panel updates the file (#3) above in OA_HOME, on the principle that you will have write permission there and - usually - it will not affect other users. However all "oa_pref" file settings are applied on the "last found wins" basis, so if you have file in your current directory with different settings these, being the last to be found, will "win".

Full details of all "oa_pref" file options and environment variables are given in Appendix XIII

2.4.4.2 Refresh: Controlling backing store redraws

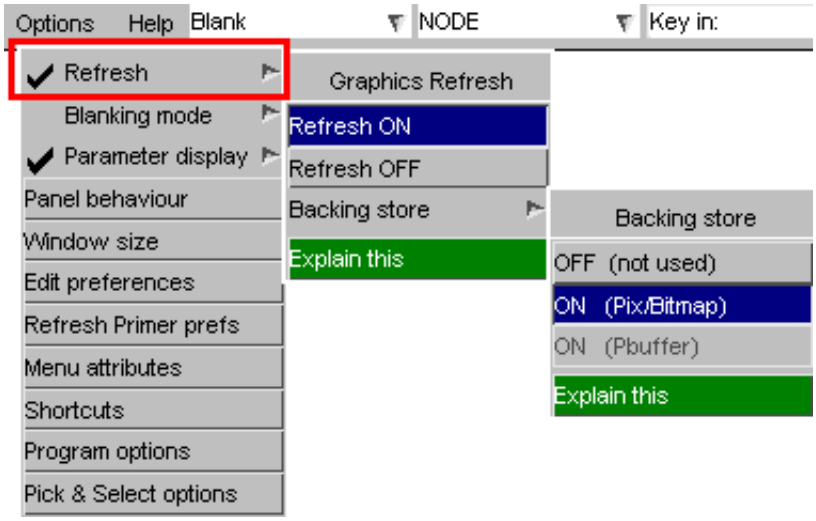
Graphics images in PRIMER may become quite complex, and therefore slow to redraw. On copmuters that do not support overlay planes in their graphics hardware (typically PCs), any window placed in front of the graphics window and then removed will leave a black hole behind, requiring the graphics window to be recomputed.

To get round the "slow redraw" problem on these machines PRIMER maintains a copy of the current graphics window in an off-screen buffer, and swaps this to the screen whenever a redraw is required: an operation that normally takes only milliseconds ... when it works. Sadly the support for off-screen graphics is not 100% reliable on all combinations of platform, operating system and graphics driver, and it is possible that the default settings will not work properly on your machine (especially, it would seem, on older laptops and under Linux). Typical symptoms when things go wrong are:

- Image does not get redrawn properly, or even at all.
- Image is redrawn, but it is unacceptably slow.
- Image is redrawn, but lighting goes wrong.

Hopefully you will not experience this, and it will simply work. But if any of the above do occur you should first of all make sure that your graphics driver is up to date. You can download new drivers from the website of the graphics card supplier, for example http://www.nvidia.com for NVidia cards, http://www.ati.com for ATI cards, and so on.

However that still may not cure the problem, in which case you will need to adjust the backing store refresh strategy using **Options > Refresh:**

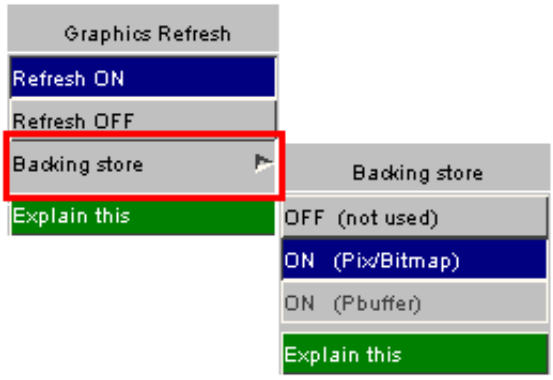


At the simplest level you can turn expose event refreshes **OFF**, using the second level menu shown here.

However that will leave your graphics image full of "holes", requiring you to give an explicit redraw command to repaint it, which is not a satisfactory solution. Nevertheless on very old, low-powered machines it may be the only thing that works.

Hopefully that will not be necessary, and one of the solutions below will be effective.

A better solution is usually to adjust the backing store display method using the **Backing store >** sub-menu



Two methods of providing backing store are available:

(1) **Bitmap** (Windows) or Pixmap (Linux/Unix).

This is usually available on all machines, uses main memory, and is reasonably quick.

(2) **PBuffer** (not Windows)

This is available on newer machines, and uses memory on the graphics card itself so - if it works - it is effectively instant.

If your machine is currently using the **PBuffer** method then try switching it to **Bit/Pixmap** to see if it improves. This seems to be particularly effective on Linux platforms, where OpenGL graphics drivers are notoriously bug-ridden.

If that doesn't work, or you are already using **Pix/Bitmap** mode (as in the example here), then try turning backing store **OFF**. This will still refresh the window, but by drawing directly to the display so there will be a visible pause and flicker while this happens. On a quick machine this may be acceptable, but on very slow machines with big models the time taken may be too long and the only solution will be to turn graphics refreshes off altogether.

Saving Backing Store Redraw Settings

Once you have found a solution that works for your machine you need to save it for future reference.

The backing store method is controlled by the oa_pref option:

**`primer*backing_store: off	on	pixmap	pbuffer`**	(The default is "on" which will choose the best method for your machine)

Graphics refreshes themselves may be turned on or off by:

`primer*graphics_refresh: off | on`

(For backwards compatibility you can also control the backing store method using the environment variables:

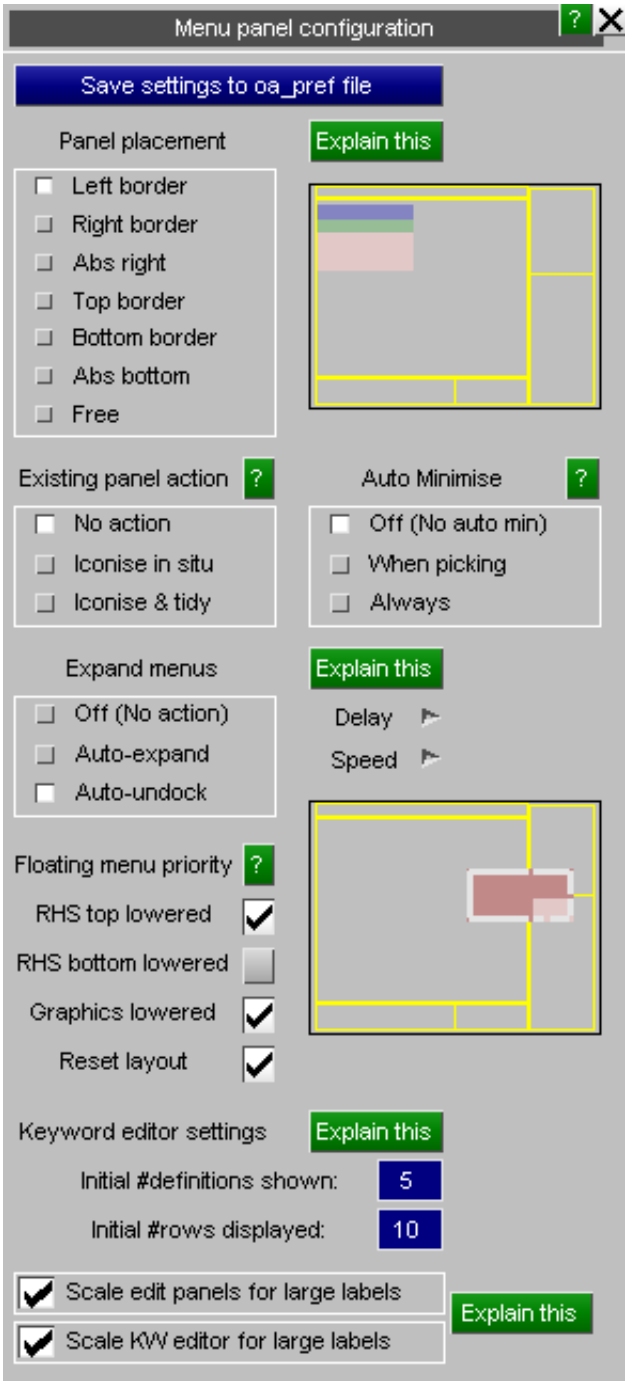
`PRIMER_NO_PBUFFER true` *which will turn off PBuffer usage*

and

`PRIMER_NO_PIXMAP true` *which will turn off all forms of backing store.)*

All these options are listed in Appendix XIII

2.4.4.3 **Panel Behaviour**: Controlling panel placement, menu expansion and action when picking.



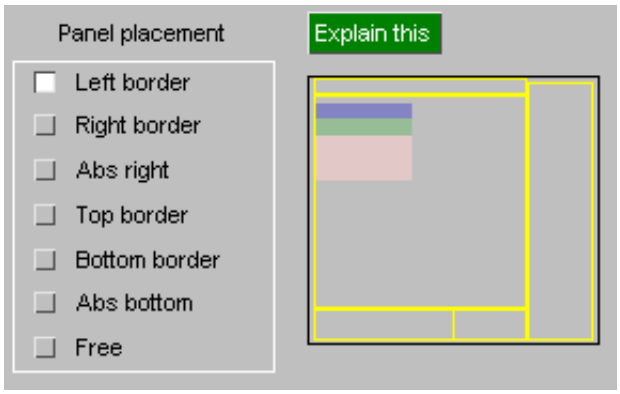
Selecting **[Options] Panel Behaviour** maps the **`Menu Panel Configuration`** panel, which controls the following:

Panel Placement	The placement of "floating" menu boxes on the display
Existing Panel Action	The action to be taken for existing floating menus when a new one is mapped
Auto Minimise	Whether to minimise floating panels when a picking operation is in force
Expand Menus	Whether to expand menu lists, the delay before doing this and their expansion speed
Floating menu priority	Whether floating menus are kept in front of the graphics window and/or the docked right hand side area.
Keyword Editor settings	The initial state of keyword editor panels when first mapped
Scaling editor panels	Automatic horizontal scaling of editor panels when large labels are used.

These options are described in more detail below.

All these settings can be saved for future PRIMER sessions in your home oa_pref file by using **Save settings to oa_pref file**.

Panel Placement



Controlling the placement of "floating" menu boxes on the display

By default "floating" menu panels, such as those which edit items (eg [Keyword], Part, Modify), will be placed somewhere in the middle of the graphics window in a location chosen automatically by PRIMER, referred to as "free" placement. Although new panels will be shifted to try to ensure that they don't overlay existing ones the default placement strategy can be annoying because it tends to put panels in front of the current graphics.

If you wish you can locate panels in a more convenient position that suits your screen size and method of working by choosing one of the Left, Right, etc options above. To see where panels will be placed click on the options in the radio button set, and the display on the right will change to show where new panels will be created.

You may also need to experiment a bit to see what method suits you best.

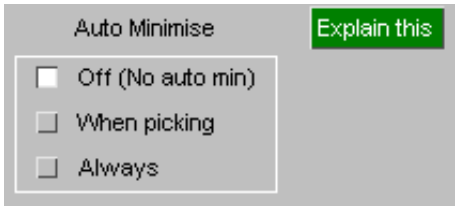
Existing panel action

What, if anything, to do with existing floating menu panels when new ones are mapped.

There are three options

No action (default)	By default existing floating panels are left as they are when new panels are mapped, and the new panel is positioned so that it overlaps existing ones in a sensible way
Iconise in situ	Existing floating panels are iconised in their existing locations, the equivalent of clicking on their top right [-] button, and the new panel is positioned to just below or alongside the icon.
Iconise & Tidy	Existing floating panels are iconised and "tidied" to a neat stack at the top left of the display, then the new panel is mapped in the appropriate location.

Auto-minimise



Whether, and in what circumstances, to minimise floating panels automatically.

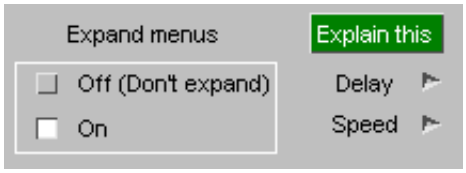
There are three options

Off (no auto min) (default) — Auto-minimisation is not active.

When picking — When cursor picking (other than the global "quick pick" operation) is active then a panel will automatically minimise itself when you move the cursor out of it into the graphics window. The panel will be restored automatically if you move the mouse back over its icon, or when the picking operation has been completed.

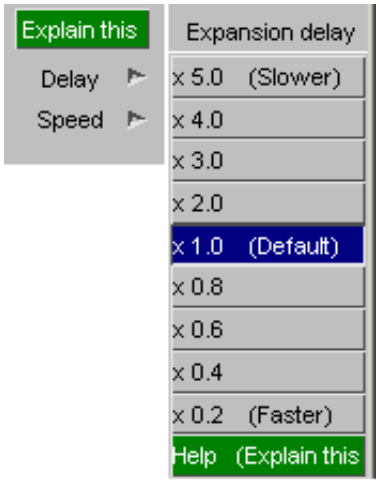
Always — Floating panels are always iconised when the cursor moves into the graphics window, whether picking is active or not.

Expand menus



Whether or not to expand lists of items in menus automatically, and parameters for this.

Many of the menus in PRIMER are too narrow when first mapped to show all the columns of their data, so by default "auto expansion" is enabled. This causes the menus to widen themselves, typically to 90% of the enclosing width available, when you move the cursor into them. They will revert to their original width when the cursor moves out of them again. This behaviour can be controlled by turning **`Expand menus`** **Off** or **On**.



You can also control:

DELAY the time interval between the mouse entering a window, and the window starting to expand.

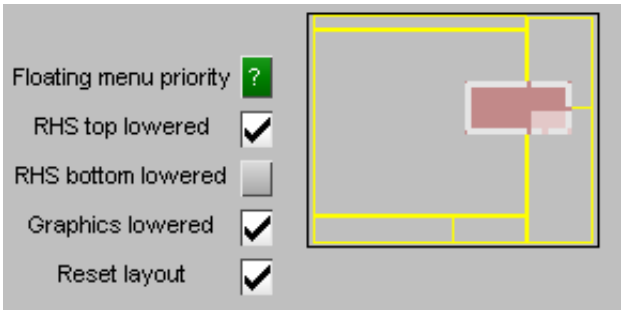
The delay time is controlled as a factor on the default behaviour.

The actual delay time will vary from system to system depending upon the Window system and underlying speed, but a typical delay will be approximately 0.5 seconds.

SPEED (Not shown here) is the rate at which the menu expands and contracts.

As above it is controlled as a factor on the default speed.

Floating menu priority



"Floating" panels are the editor and other panels that are free to "float" in the PRIMER window, or the desktop, and which can moved, resized raised and lowered by the user.

These windows can sometimes get "lost" when they are moved behind (underneath) the graphics window, or behind the docked right hand side (RHS) area, and restoring them can be difficult.

RHS top lowered	Will keep the top half of the docked RHS area, containing Tools and Keywords, at the bottom of the window stacking order. The default is to keep this top half lowered.
RHS bottom lowered	Will keep the bottom half of the docked RHS area, containing "tabbed" panels, lowered. Using this can have side-effects. When a new Tools or Keyword operation is started the bottom RHS area will not be brought to the front, and as a result any floating panels currently lying over the RHS area may obscure the new panel. It is easy enough to move any such floating panels out of the way, but if you find this a nuisance you may wish not to use this setting. For this reason the default is not to keep this bottom half lowered.
Graphics lowered	Will keep the graphics window at the bottom of the window stacking order. This is not unconditional. The user may wish to bring the graphics window to the front in order to obtain an unimpeded view of the model, and this can still be done by clicking on its top bar, or by maximising it with the top right [] button. It will remain in front during any dyanmic viewing operation using shift/control + mouse. However if **Graphics Lowered** is active the graphics window will be lowered again as soon as any button clicks take place anywhere in the user interface.
Reset Layout	When the master PRIMER window is resized, or goes through an Iconise / Restore cycle, then by default the window layout is reset to the default. This means that the graphics window is resized to fit into its proper "slot" on the screen, the RHS docked area is reset to its default position, and the dialogue box is also returned to its standard size and shape. If this option is deselected then the layout of all of the above will be unaffected by master window size and shape changes.

The default is for both of these options to be set but, as with the other settings on this panel, you can capture your current settings in the oa_pref file for use in future PRIMER sessions.

Note: From V11 onwards there is an alternative, more direct way of restoring "lost" floating windows using the"lower" button at the top right of the graphics box. Regardless of the two settings above this will forcibly lower both graphics window and docked RHS area to the bottom of the window stacking order, and any floating panels "lost" behind them will become visible again.

Keyword editor settings

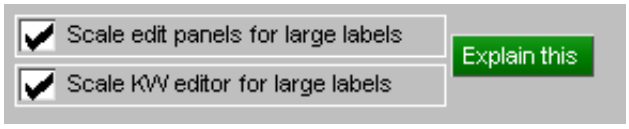
Controlling the initial appearance of the generic keyword editor.

When the Keyword editor is first mapped you can control the following attributes of its panel to limit its initial size:

`Initial #definitions shown`	The maximum number of actual items that will be displayed.
`Initial #rows displayed`	The maximum number of rows of data that will be displayed. If the items being shown span several rows of data then this may limit the number of items actually shown. However there will never be less than one item shown, regardless of how many rows of data it has.

Practical considerations may also limit the size of the panel: if there is not enough space available on the screen to display the requested data then the number of rows and/or items may be limited further.

Scaling editor panels for large labels



Controlling whether or not editor panels are scaled horizontally to show "large" labels.

I order to minimise the screen space used by editing panels the buttons are sized to show "small" labels in the range 1 to 99,999,999. When "large" labels are used this can result in the labels being truncated and difficult to read, so PRIMER can scale these panels horizontally in order to allow the full labels to be shown. The default is for panels to be scaled.

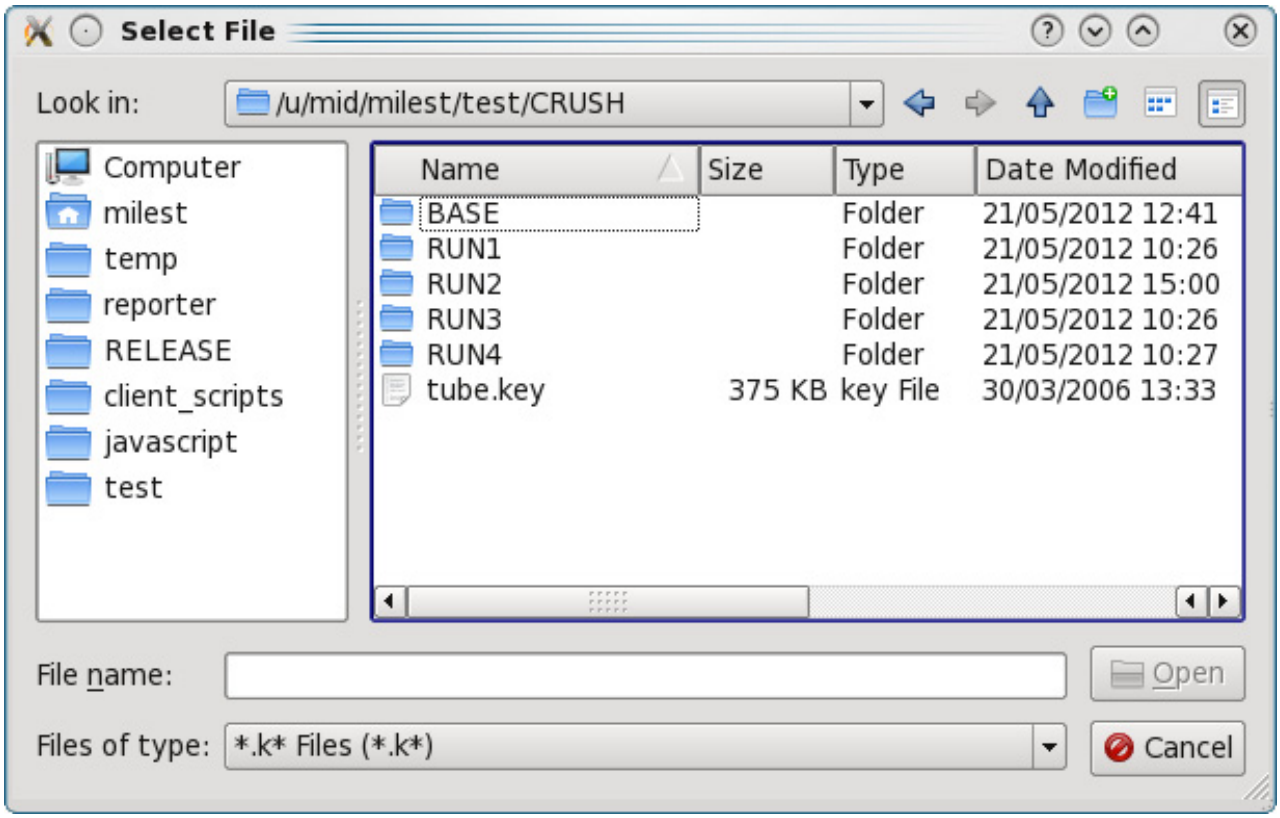
This process is robust but quite crude: PRIMER finds the largest label of any item in the model, computes its width, and then scales the panel to accomodate this. If the panel does not in fact include any items with large labels this can result in quite a lot of wasted space, so this behaviour is switchable, and can also be saved using the preferences below.

Option	Description	Preference
Scale edit panels for large labels	Controls whether or not "scalar" editing panels for single items are scaled.	**`primer*scale_edit_panels: true | false`**
Scale KW editor for large labels	Controls whether or not Keyword Editor panels are scaled.	**`primer*scale_kw_editor: true | false`**

2.5 Using standard "file filter" boxes.

Wherever PRIMER requires you to enter a filename you will be presented with a text box into which to type it. However, to the right of this text box you will also see a button with an image of a yellow folder, which may be used to invoke a standard file filter box. The appearance of this is operating system dependent.

2.5.1 Standard linux file filter box



The files can be filtered according to file types by using the **Files of type** popup, in this case the pathname is **`/u/mid/milest/test/CRUSH/`** and the pattern is `*.k*`.

The main window show a list of the directories within the present one and a list of files that match the "Files of type" selection.

To go back up the directory tree you need to select the button, or you can click on the **Look in** popup to select any of the parent directories.

The **File name** box shows the current selection.

The **Open** button closes the file filter box and opens the selected file

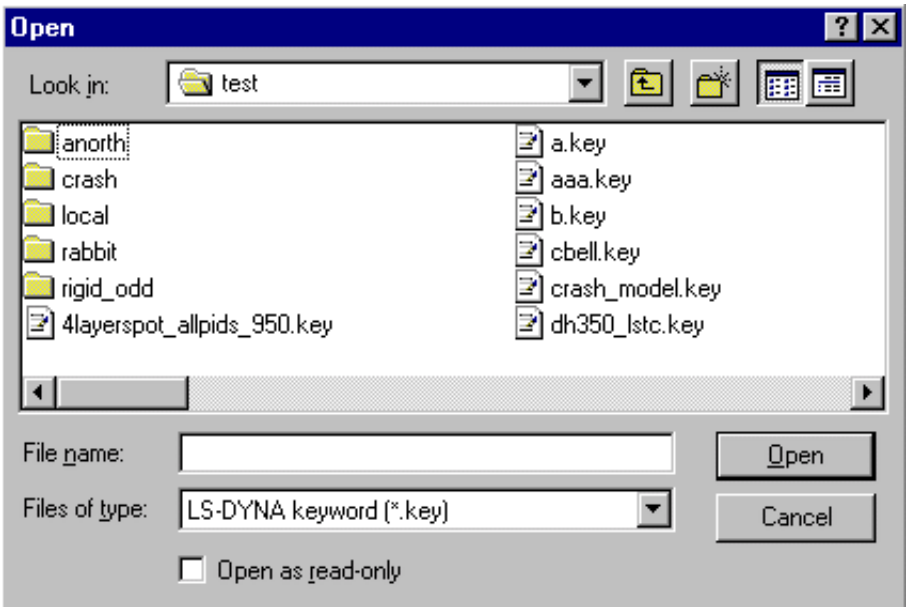
The **Cancel** button closes the file filter box without opening any files

You select a directory by clicking on it.. This updates the "Files" box accordingly. You then select a file by clicking on its name in the main area, and finally on **Open** to make it your choice and return.

As an alternative to selecting a file and pressing **Open** you can double-click (quickly) on the file to make your selection.

The left hand area of the menu shows commonly used directories. In this case **`temp`**, **`reporter`**, **`RELEASE`** etc. You can add directories to the list by dragging them from the main area and dropping them. Clicking on one of these directories updates the main area to that directory.

2.5.2 Standard "Windows" file filter box



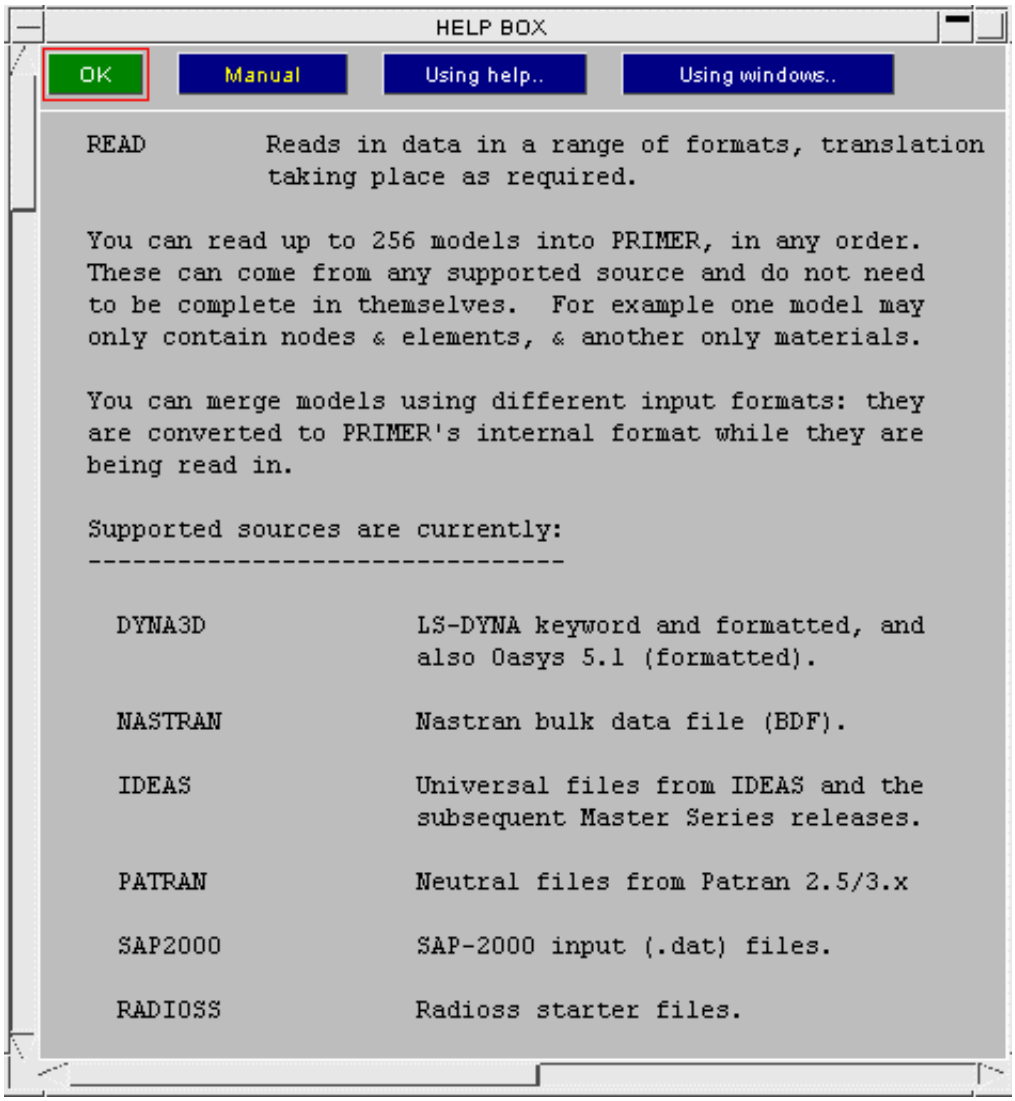
Double-click on the directory required, then on the filename you wish to open.

To open files that do not have the (**`*.key`**) extension you will need to select:

All files (*.*) from the **Files of type** pull -down menu.

2.6 Obtaining Help and advice.

PRIMER has extensive on-line help available. In any context you will find either a **HELP** button or a [**?**] (on a green background) that will give access to help. Generally speaking it will map a "Help Box", as shown in the example below, and input will be locked into that box until you click on **OK** (or hit `<return>` in that box). The **Manual** button links to the appropriate page of the on-line version of this manual.



2.6.1 Keyword manual shortcut

On some panels (Current Menu panel, Keyword Editor and Create/Edit panels, see section 5.1 for more details) a [**K**] button (on a red background) will give access to the LS-DYNA keyword user's manual. It will open in a default pdf viewer at the volume and page corresponding the current panel. Note that the default pdf viewer and the command used to open at a specific page can be controlled via preferences, see Appendix XIII.

The LS-DYNA keyword user's manuals and the look up file used to determine the correct page to open at will be added to the appropriate directories upon installation. If, however, the keyword manual shortcut button does not appear please contact your local distributor or Oasys Ltd. for support.

2.7 Error and Warning messages

Occasionally you will get error or warning messages. These are written to the dialogue box in red, prefaced by **%%% ERROR** or **%%% WARNING** respectively.

Internal errors (let us hope you never see any) are also copied to standard output, ie the terminal window from which PRIMER has been invoked. If you get any of these please make a copy of them and inform Oasys Ltd.

2.8 Dealing with crashes

In an ideal world PRIMER would never crash, but sadly this is not the case so it has two methods for dealing with crashes:

1. It has a "crash handler" which intervenes when the operating system detects an illegal instruction that would cause a crash, and gives you some recovery options. These options include saving all models in the database in emergency keyout files, and also - depending on the mode chosen - attempting to continue execution. It is also possible to write trace-backs and "mini dump" files which can be sent to Oasys ltd to help us to determine what went wrong.

2. In addition it automatically saves every command and screen button click performed by the user in a "checkpoint" file. If it crashes this file can be replayed to repeat exactly what happened previously, and because it does not replay the final command - which caused the crash - it should return you to the point just before things went wrong.

Neither method is perfect: checkpoint files can be cumbersome to replay if it was a long session, attempting to continue execution does not always work if memory has become horribly corrupted, and not all aspects of work in progress are saved in emergency dump files. However these methods are better than nothing!

2.8.1 The crash handler

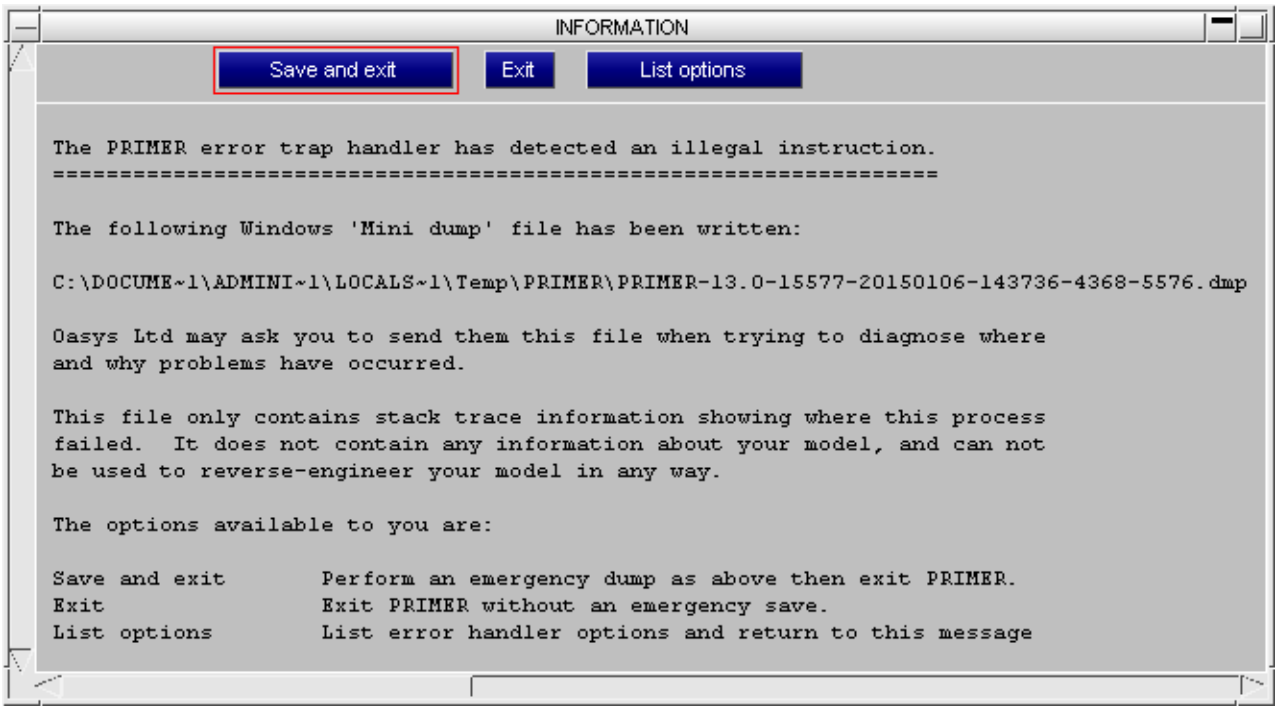
Normally when a piece of software crashes the operating system terminates the process there and then leaving no means of recovering data or determining what went wrong. This doesn't help you, the user, since you lose all your work; it doesn't help us (Oasys Ltd) either since we don't have any evidence with which to debug the problem.

In an ideal world the following would happen when a programme detects an impending crash:
- It would save all your data.
- It would determine where the crash occurred and save this information for debugging.
- It would then allow you to continue execution as if nothing had gone wrong.

Sadly it is not possible to do all the above, but it ***is*** possible to go some way towards it, and this is what the crash handler does. It works as follows:
- It starts PRIMER in a special mode which tells the operating system "if you detect a crash then don't just kill me, instead tell me what happened and let me deal with it".

- Control is passed to the "crash handler" which tries to deal with the problem as gracefully as possible.

Here is a typical crash handler panel as captured on a Windows platform:



You can control how the crash handler works using the preference **`primer*error_handler:`** ***`option`***

Option	What it does

mini_dump (*only available on Windows*)	Writes a traceback if possible Gives you the option of saving all models in keyout files Writes a special "mini dump" file that can be used by Oasys Ltd for debugging Terminates execution
trap_continue	Writes a traceback if possible Gives you the option of saving all models in keyout files Gives you the option of continuing, which may not always work Alternatively you can terminate execution
trace_exit	Writes a traceback if possible Terminates execution
no_action	The normal "immediate exit" action of the operating system

The default settings are:

Windows	**mini_dump**
Linux	**trap_continue**

The summary below gives more information about crash handling. It is a bit technical and you don't have to understand it, so if you need more help please contact Oasys Ltd.

What is a "traceback"?

This is a summary listing of the "stack frame" of the programme. It gives a "tree" of the functions currently being called, and sometimes also line numbers and values passed to those functions.

This helps Oasys Ltd to debug the programme since while it may not tell us exactly what went wrong it can at least tell tell us more or less where, and sometimes also gives clues about why as well. Here is a typical example taken from a Linux machine:

```
Here is the stack trace (16 entries):

0: /home/dyna71/rhe5_12/primer14_64.exe [0x78081a]
1: /lib64/libc.so.6 [0x3fba230280]
2: /home/dyna71/rhe5_12/primer14_64.exe [0xc44eb9]
3: /home/dyna71/rhe5_12/primer14_64.exe(build_off_screen_image_og_+0x5af)
4: /home/dyna71/rhe5_12/primer14_64.exe(generate_image_+0x384) [0x584914]
5: /home/dyna71/rhe5_12/primer14_64.exe(laser_user_+0xc51) [0x7e7ec1]
6: /home/dyna71/rhe5_12/primer14_64.exe(process_sm_+0xb9e) [0x6a5d4e]
7: /home/dyna71/rhe5_12/primer14_64.exe(us_input_+0x1c0) [0xf1b9c0]
8: /home/dyna71/rhe5_12/primer14_64.exe(getstr_+0x26f) [0xf197d1]
9: /home/dyna71/rhe5_12/primer14_64.exe(comand_+0x11) [0xf17e81]
10: /home/dyna71/rhe5_12/primer14_64.exe(manage_+0x4ea) [0x57ddda]
11: /home/dyna71/rhe5_12/primer14_64.exe [0x780a07]
12: /home/dyna71/rhe5_12/primer14_64.exe(MAIN__+0x142) [0x4e7872]
13: /home/dyna71/rhe5_12/primer14_64.exe(main+0x46) [0x4e7716]
14: /lib64/libc.so.6(__libc_start_main+0xf4) [0x3fba21d974]
15: /home/dyna71/rhe5_12/primer14_64.exe [0x4e7619]
```

You will see that this tells us what functions were called, and in this case it indicates that problems arose when building an off-screen image while capturing a plot for a laser printer.

What is a "mini-dump" file?

This is a special file, generated on Windows platforms only, that can be used to debug the process. It is typically about 50kBytes long and it contains enough information for code developers to be able to observe the full stack frame, much like the traceback above, but in rather more detail.

If you have ever encountered the normal Windows behaviour when a crash occurs that asks whether you want to send debugging information to Microsoft, and you have said "yes", then this mini-dump file is what you have sent.

This file will be placed in your home directory, typically under
C:\users*yourname***\local\appdata\temp\primer** and it will have the syntax

primer-*version-svn_rev-date-time-pid-thread*.**dmp.**

What are the security implications of sending tracebacks or mini-dump files to Oasys Ltd?

Oasys Ltd may ask you to send us tracebacks or mini-dump files to help us to work out what went wrong.

If you are working on a model that is confidential you may be concerned that by sending this information to Oasys Ltd you may be giving us confidential information that you are not allowed to divulge for commercial reasons.

Quite apart from the fact that we always treat client data as being confidential, you should not be concerned that we can reverse-engineer your model in any way since these files do not contain enough information to do this.

Tracebacks	Contain no model-specific information at all. As you will observe from the example above all they contain is a listing of the functions in which the crash occurred. There is absolutely no data from your model, and they are 100% secure in that respect.
Mini-dump files	May contain a tiny amount of information, typically about quantities of data. The data these contain is a "frozen" slice of what you were doing at the time, and this may include some scalar information. Typical examples might be: • The number of nodes or elements in your model. • The coordinates of the node you were dealing with at the time • The value of a variable It must be stressed that no lists of data will ever be available in these files, for example while we may get the coordinates of the single node being processed when the crash occurred we will ***not*** be able to look at the coordinates of any other nodes. Nor can we recover what was on the screen or the contents of any windows in the user interface. The size of these files, typically 50kBytes, is an indication that they simply cannot contain much information!

How can execution continue after a crash in "trap_continue" mode?

It sounds paradoxical that it is possible to continue execution after a fatal error has occurred, but this is possible if the following method is used:

- When PRIMER first starts make a copy of this initial "clean" stack frame.
- Create an error handler that intervenes during a fatal crash
- Allow this error handler to throw away the existing "corrupt" stack frame and revert to that initial "clean" one.

In non-technical terms you can think of this as having a spare set of clothes available to change into if you have fallen into the mud and made your existing clothes all dirty.

However there are some limitations to this process. If your database has become corrupt then attempting to continue will fail very quickly once again, and this time you will not be able to continue. To use the simple analogy: if you are stuck in the middle of a muddy field then putting on your spare clothes won’t help much since as soon as you take a step you will get them muddy as well, and now you have no more spare clothes to change into.

Normally it is best to do the absolute minimum possible required to save your work when continuing execution, then to exit PRIMER and to start again with a fresh session.

Why can I not continue execution after writing a mini-dump file?

It would be wonderful if "mini_dump" mode, the default on Windows, allowed you to continue execution once you had written the file, in the way that "trap_continue" mode does.

Unfortunately because of the way Windows works this is not possible: if the ability to write a mini-dump file is enabled then, once it has been written, execution has to terminate. This is because control is passed to a special Windows handler and cannot be "retrieved" once it has entered this.

You can set "trap_continue" mode to be your default on Windows, but the disadvantages of this are two-fold:

1. The stack trace, if it available, provides far less debugging information than a mini-dump file, so Oasys Ltd may not be able to work out from it what went wrong.
2. Continuing execution may often not work for the reasons given above.

Therefore we would encourage you to use the default "mini_dump" mode on Windows, mainly because it makes it much easier for us to find the sources of crashes ... and therefore to fix them.

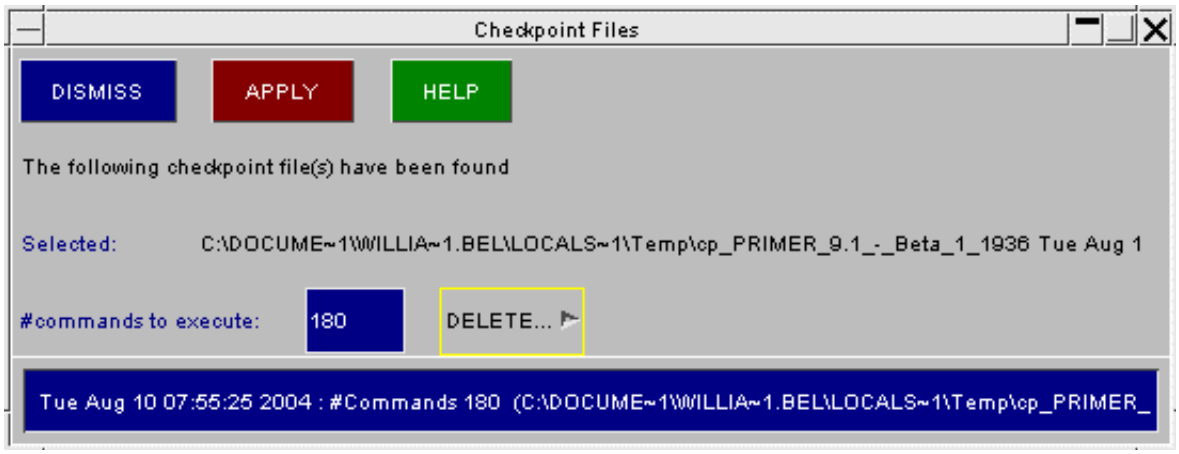
2.8.2 Checkpoint/Recovery files

All commands of your primer session are recorded in a binary (non-editable) checkpoint or recovery file (CP_PRIMER_14.0_xxx). If the session terminates normally the file is automatically deleted. If the software crashes, the file will be left behind.

When you next start primer, you will be offered the option of rerunning any existing checkpoint files. Do not forget to **remove the last command**, by decrementing the *#commands to execute* counter or the crash will simply repeat.

Note that if you have overwritten your original file during the session, rerunning the command file will not be helpful.

In some cases, sending the checkpoint file and the input files to Oasys Ltd will assist in debugging the software, although "mini dump" files and/or tracebacks as described in 2.8.1 above are normally more useful for debugging.

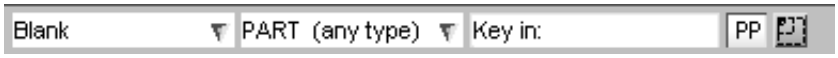


If you rerun someone else's checkpoint file on a different computer you may find that it fails for either or both of the following two reasons:

Problem	**Solution**	**Explanation**
Any files read or written have an incompatible pathname embedded, meaning that they cannot be found.	Set the environment variable **`CP_FILE_FILTER true`**	Means that whenever a file filter was used in the original run PRIMER will map a file filter during checkpoint replay, and wait for you to select the file manually.
The screen window comes up the wrong size or shape, meaning that screen-picking operations do not always select the correct items	Set the environment variable **`CP_REFORMAT true`**	Will attempt to reformat the PRIMER window's resolution to that of the display on which the checkpoint file was captured. This may not work if the resolutions of the two devices are wildly different

Note: Checkpoint files should be cross-platform, ie a file generated on machine A should replay on machine B; however they are ***not*** cross-version, and will only work with exactly the same version of the software.

2.9 Quick Pick Function



This function allows a range of operations to be applied through Screen Picking. The function has two menus to make selections from, located above the graphics area; defining the action applied to selected entities, and the entity type to be selected.

Quick Pick has three possible selection modes, toggled through either by clicking on the button to the right of **[PP]** or via the upper case "**Q**" short cut key. The modes are:

1	Pick + rectangular drag	Pick a single item, or drag out a rectangular area to select all items within it.
2	Polygon pick	Define a complex polygon with between 3 and 100 vertices, selecting all items within it.
3	Pick + circular drag	Pick a single item, or drag out a circular area to select all items within it.
4	Feat Line Pick	This mode applies only to the picking of 2D and 3D elements, and of nodes. Similar to Feature Line Picking

The default "Pick + Rectangular drag" mode

Picking works as follows:

- Left-clicking on an item selects just that item and applies the current function.
- Left click and dragging out an area selects the items in that area and applies the current function.

- Right click on an item selects it and maps the popup of possible functions to be applied.
- Right click and drag applies selects the items in the area and maps the popup menu of possible functions.

- Middle click means "undo"the most recent quick pick function. In most cases you can undo all the way back to the initial quick pick; however deleting a model will have the effect of deleting operations upon that model from the undo stack.

- You can also type an item label into **Key in** box, which is one way to locate items by label.

- [PP] Refers to the current Predictive Picking status, and can be used to toggle it on/off temporarily.

"Polygonal area" mode

Picking works as follows:

- Left mouse clicks are used to define a series of points that will form the vertices of a polygon. This can be any shape, with concave and crossed edges. It must have a minimum of three points and may contain up to 100 points.

- Once it has, or will have, three points the polygon can be closed by any of the following methods:

    - Clicking on (or very close to) the first point, to signify "close the polygon by joining last and first points".
    - Double-clicking when selecting the next point, to signify "create this point and close polygon".
    - Short cut key upper case "**C**" to signify "close the polygon by joining last and first points".
    - Clicking on the **[C]** button on the top right (as in the image to the right here)

- You can reject the most recent point by:

    - A middle mouse click.
    - Using the upper case "**R**" keyboard short cut.
    - Clicking on the **[R]** button at the top right.

    Rejection of points can be repeated to delete successive points backwards towards the 1st until none remains.

- You can reject the complete polygon and start afresh by:

    - Using the "**escape key**" keyboard short cut.
    - Clicking on the **[X]** button at the top right.

- You can also use right mouse clicks to define the polygon, which work in exactly the same way as above except that when the polygon is closed you will be offered the same "right click options menu" of options you would get when using a right click for selection in the default "Pick + Rectangular drag" mode.

"Pick + Circle drag" mode

This works in exactly the same way as "Pick + rectangular drag", except that the area dragged out is a circle rather than a rectangle.

"Feature Line Pick " mode

This works in exactly the same way as Feature Line Picking.

The purpose of the right click functionality is to permit any of the functions listed below to be applied to selection without having to change the master quick pick mode.

The first menu allows selection of what function is to be applied to left click operations. A summary of these is provided here:

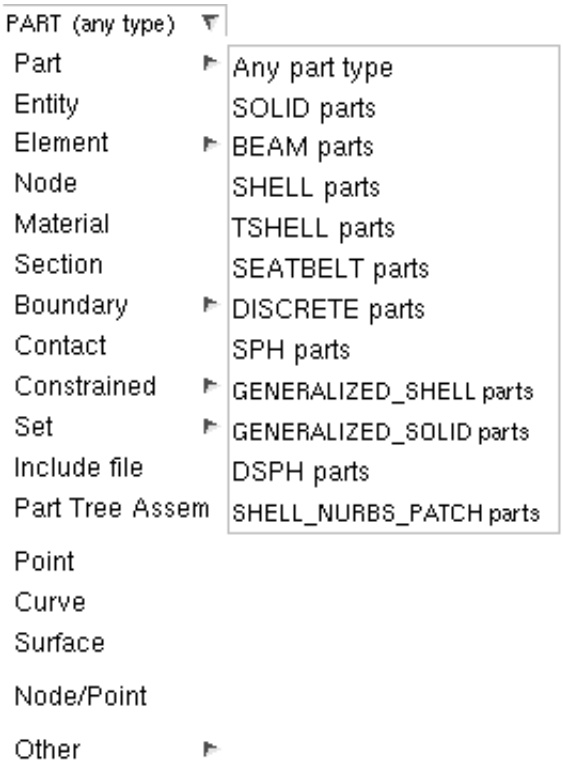
The exact options in this popup menu will vary according to the entity type selected for operations.

The example here is for the commonest case of **PART**

Blank	Blanks the selected item(s). For more information on Blanking see section 4.5
Unblank	Unblanks the whole of the selected item(s) (only available for certain types)
Only	Blanks everything except the selected item(s)
Delete	Delete the selected item(s). There are two options. "Confirm" means you will be asked to confirm the deletion of the entity(s) and associated entity(s), "No confirm" means the deletion will occur without confirmation.
Information	Provides a list of the item's properties. (If multiple items are selected only the first is shown.)
Label	Labels the item on the screen dynamically, with a choice of attributes selected in the sub-menu
Edit	Maps the standard editing panel for the selected item(s). (To a maximum of 20 panels)
Keyword	Maps the standard Keyword Editor (see section 5.1.3), showing the selected item(s) only.
Colour	Sets the colour of the selected item(s) to the one set in the pop-up menu (accessed by >)
Transparency	Sets the transparency of the selected item(s) to the value set in the pop-up menu (accessed by >)
Plotting Mode	Sets the plotting mode (Shaded, Wireframe, etc) of the selected item(s) to that set in the pop-up menu (accessed by >)
Locate in Tree	Highlights the selected part(s) in the part tree. Add selects in addition to any currently selected in the part tree, only selects instead of any existing selection. See section 7.17 for more detail on the Part Tree.
Part Table	Produces a Part Table for the selected part. See section 7.16 for more detail.
Sketch	Sketches the selected item and locates cross-hairs at its centre (generally used via **Key in** <label>)
Find	Envoke Find feature for selected item. For more information on Find, see section 6.16.
Xrefs	Envoke the Xrefs panel for selected item. For more information on Xrefs, see section 6.36.
Set current layer	Set the current include layer to the include file which the picked item is in.

<item> **Details** Opens the detailed information panel for that item.

There are a number of different types of item to which Quick Pick can be applied. The item type to be selected is chosen from the second menu, displayed here below. This selected choice here will affect the options available from the first menu. For example, Part Table is only available when Part is selected as the item type and <item> Details is only available for elements and Nodes.



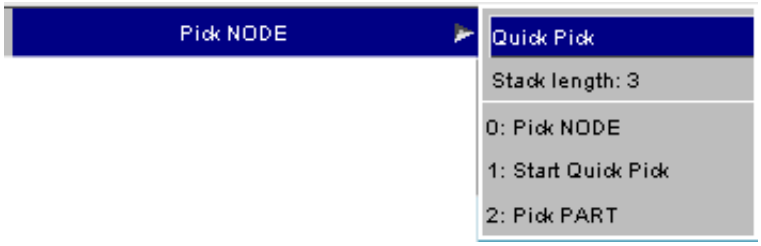
This shows the master popup menu of all possible types, and the second level menus under Element, Boundary, etc permit more detailed selection of type.

A special case is Part picking, which can be too crude at times, especially when attempting to select a beam part from beams in front of 2D or 3D elements, as the latter will always be favoured.

Therefore it is possible to restrict PART picking to a specific element sub-type by selecting from the 2nd level popup **Part >** as shown here.

To revert to general part picking use **Any part type**.

Whenever screen picking can be applied for a menu other than Quick Pick (see section 2.13) the menus in the top option box will be replaced by a box indicating what can be currently screen picked. Quick Pick control can be restored either by clicking the white cross in the top left of the graphics area, or from the drop-down in the Quick Pick control.



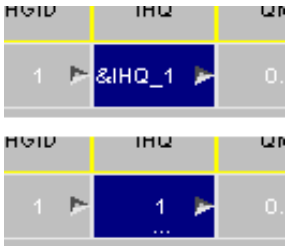
2.10 Using Parameters in Edit Panels.

From PRIMER 9.3RC2 onwards Parameters, as in the LS-DYNA **`*PARAMETER`** keyword, are fully supported in interactive editing panels.

- **Wherever Parameters have been used in the input deck these will be displayed in edit panels.**

  Parameters can be displayed either as they would appear in the keyword file, ie **`&NAME.`** (Here **`&IHQ_1`**)

  Or their numeric values can be shown, underlined with dots to show that the field is parameterised.



- **Parameters may be typed into any editing panel data field.**

  In exactly the same way that you can type in numbers you can now also type in parameters using ***&NAME*** syntax.

  If <**NAME**> is an existing parameter its value will be used.
  If it is a new parameter you will be invited to provide its value.

  This behaviour is triggered by typing the initial ampersand "**&**" into the data field. A list of all parameters will be mapped, and as you type more letters the narrows down to show only those which match.

  It is also possible to use wildcard syntax containing * and ?. In that case the popup will show all parameters matching this pattern when * is replaced with any character string and ? with any single character.

- **Hovering the cursor over a parameterised field gives further options.**

  If you hover the cursor over a field containing a parameter a popup box giving more details about its attributes will be mapped. You will also be able to **EDIT** the parameter by using the appropriate button in that box.

- **Parameters may be created, edited and deleted just like any other keyword item.**

  Parameters can now be processed just like any other keyword item using the PARAMETER keyword tool.

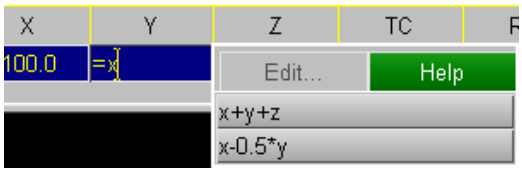
- **The `*PARAMETER_EXPRESSION` keyword is now fully supported.**

  The `_EXPRESSION` variant of parameters, in which a parameter may be defined using an arbitrary mathematical expression that can reference other parameters, is now supported.

  These are evaluated on initial keyword input, and the correct value is used in the data field.
  They may also be created and edited interactively.

Full details of the processing and display of Parameters may be found in Section 5.

2.11 Formulae in Edit panels



From PRIMER 13.0 onwards it is possible to specify the value of a field on an edit panel as a formula in terms of the values of other fields. The fields are referenced by the corresponding acronym on the header of the edit panel. For example, the input `=(x+y)/2` or just `(x+y)/2` to the field for the Y coordinate will set the Y coordinate to the sum of the X coordinate and the previous Y coordinate divided by 2.

The acronyms to reference the fields are almost always the name of the field in the LS-DYNA keyword manual, which is also written above the field in the edit panel. To reference an acronym whose name contains an arithmetical operator, e.g. `r/lx`, the acronym should be typed `"r/lx"` (in quotes) to distinguish it from `r` divided by `lx`, which may or may not exist. Note that acronyms are not case-sensitive.

Similarly to the behaviour for parameters above, just typing "`=`" maps a list of all formulae previously used in the model. When typing more characters, the list will be refined accordingly.

Once the formula is evaluated, Primer does not store which data field you used it for. If you change the X coordinate after the formula `=(x+y)/2` has been evaluated, the field where you used that formula does not change its value unless you edit it again. If the formula references a parameter or a field containing a parameter, the evaluation will be done just using its value.

Unlike for parameters, integer by integer divisions in formulae are carried out using floating point arithmetic. For instance, when a formula contains the calculation `7/4`, the result will be 1.75 when assigned to a floating point field.

In every panel in Primer, not just in keyword editing panels, it is possible to use a formula for an arbitrary numeric input field as long as it only uses numeric values but no acronyms. To replace the value of a field by twice its old value, you can just click at the end of the field without deleting the existing character string and append `*2`. Primer will carry out the calculation.

More detail about how to use formulae in the generic keyword editing panel (the Keyword editor) can be found in Section 5.1.3

2.12 Operational Hierarchy

Operations in PRIMER act internally using a "hierarchy" of entity types, and it is important that you appreciate how this is applied. A cut down summary of this internal hierarchy is:

Highest level		===>		Lowest level
	DUMMIES	CONTACTS	ELEMENTS	L.CURVES
MODELS	SETS	PARTS	RIG WALLS	NODES
	MATERIALS	GEN STIFF	BND CONDS	LOADS
	SECTIONS	AIRBAGS		

When you select an object for operations it implicitly selects all entities below its level, but not those at or above its own level.

For example selecting a model has the effect of operating on everything in that model, while selecting a **MATERIAL** in that model acts as follows:

- The **MATERIAL** selects **PARTs**, **ELEMENTs**, etc that refer to it;
- The **ELEMENTs** select **NODEs** on them;
- The **BOUNDARY CONDITIONs** might select **L.CURVES**, etc.

But note that selecting a **PART** does not (directly) affect a contact surface that references that part; although operating on its nodes might affect the contact geometry. Likewise selecting an **ELEMENT** will affect its nodes, but not boundary conditions applying to those nodes.

2.13 Selecting Entities for Operations

In many contexts within PRIMER (blanking, orienting, deletion, ...) it is necessary to select one or more entities for the current operation. Selection takes place using a system of cascading "object menus", combined with screen-picking, area selection and keyed in data.

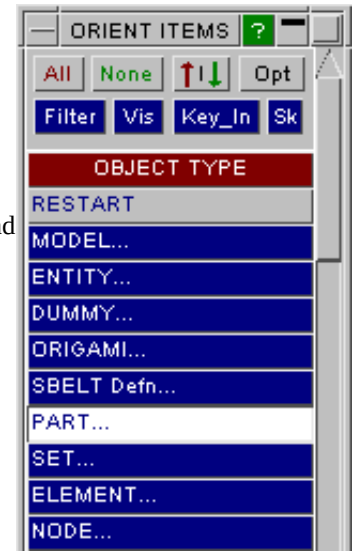
Primary Selection
- Selection from menu list
- "Filtering" selections

Screen and area picking
- Rejecting screen picks

"Keying in" selections
- "Key in"syntax
  Combining methods, Restarting, Other selection methods.

When performing operations you have to select those entities upon which to operate. This is done via a standard selection menu hierarchy as follows:

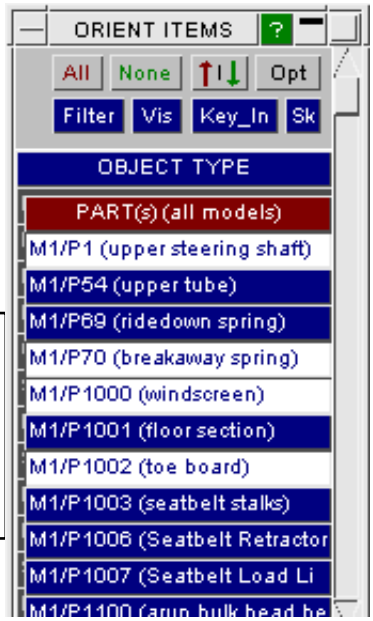
Primary selection of object type



In whatever context you are operating, here **ORIENT**, you will be presented with the primary menu of object types to operate upon.

In this example the user has chosen **PARTS** from the range of possible categories. The list of categories available will depend on the operation being carried out, its context and model contents.

Selection of objects from the menu list



Once an object category has been selected you are presented with a list of possible choices. In this case there are two models, each with several parts, and the user has selected one part from model 1 and two from model 2.

Objects can be selected or deselected (by clicking on them again) at will.

You can also use

All	To select all eligible items	Note that **All**, **None** and **I**(*nvert*) ***only operate upon what is shown in the menu*** for reasons that will become apparent below.
None	To deselect all eligible items	
I(*invert*)	To invert the current selection	

Of the other buttons at the top of this panel:

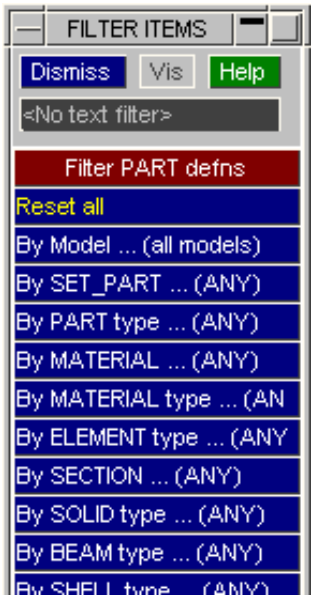
Opt(ions)	Further options (refresh, clipboard, blanking)
Filter	Applies "filtering" to what is shown
Vis	Maps a panel showing further "visible" picking options
Key_In	Maps a panel allowing you to key in label ranges directly
Sk(etch)	Sketches what is currently selected.

"Hover over": showing what will be selected.

By default hovering the cursor over a row in an object menu will also highlight and label the item on the screen, helping to identify where in the model the item is.

This process is described in section 4.7, which covers Predictive picking, since the two functions are closely related. Details of how "hover over" works, and how to control it are given in section 4.7.4

Using **Filter** to limit what appears in the menu

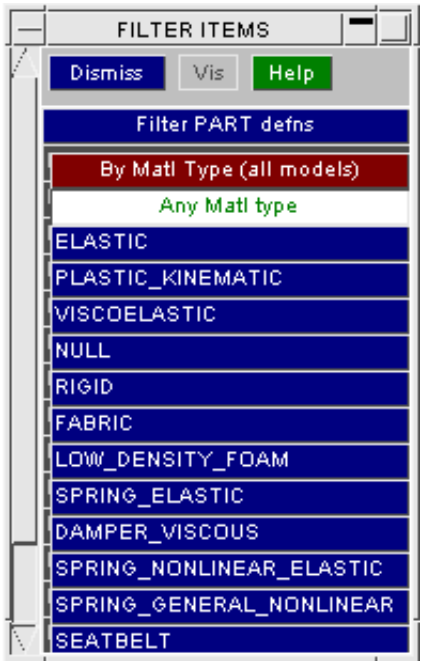


Where the list of objects is short this method of selection presents no problems. But in some cases the list may be hundreds or even thousands of items long, and a method of cutting down what is displayed is required. This is provided by the **FILTER** button at the top of this box.

This allows you to control what is displayed in the selection menu by providing a series of tests against objects are compared before they are included. The tests vary by object type, those for PARTS are shown here. By default all tests are unset (**ANY**), but you can set any combination: multiple ones combine in effect.

In addition to the specific types in the menu rows you can search the menu object menu list by Text filter.

Example of setting a **Filter**

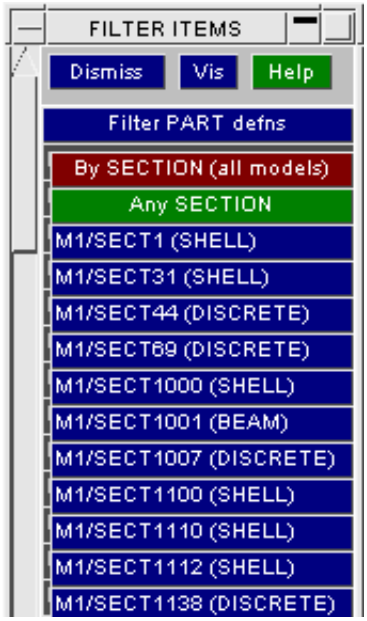


Here the **MATL TYPE ...** option for PARTS has been selected.

You are presented with all possible material types in the model(s), and can choose one. The **ANY** default may be chosen, as here, to revert to no filtering by this category.

An **<undefined>** category is sometimes included. This is because models may contain (say) PARTs referencing MATERIALs that have not been defined yet will have a <null> material type entry.

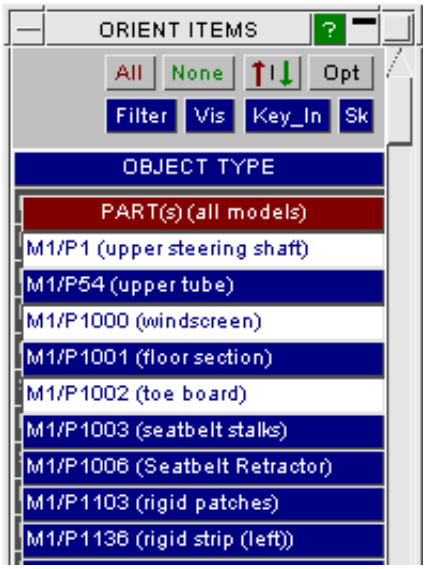
Another FILTER example, this time a pickable one.



Here the user is filtering by **`SECTION`**, and because these are screen-pickable the **Vis** button in the filter menu is now active.

In this context you can choose either to select an explicit row, as above, or to use **Vis** and to screen-pick a section from the current image.

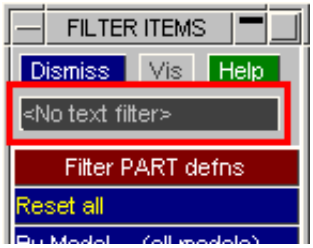
The influence of selecting a FILTER option



In the example shown here the user has selected ***MAT_RIGID** as the material filter. This causes the selection menu to be updated immediately to show only those PARTs which reference rigid materials.

Note that the **ALL** and **NONE** options ***will now only operate on the six parts shown here.*** They will not affect the selection status of anything picked previously that does not now appear in the menu list.

Using **Text Filter** to search by text string



By typing something into the text filter box you will limit what is displayed in the main object menu to items that match that string.

To cancel text filtering simply delete the contents of the box, and it will revert to showing <No text filter> as here.

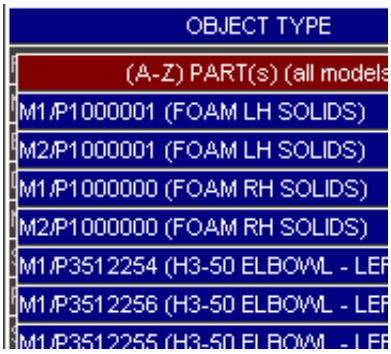
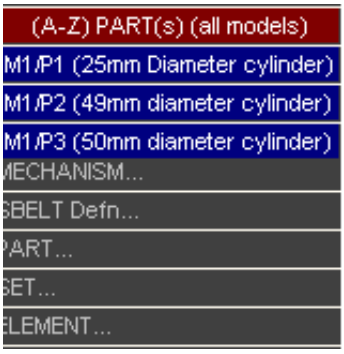
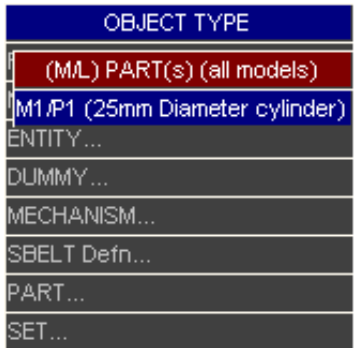
Text matching is not case sensitive, and leading and trailing white space characters are ignored. However embedded white space in a string is considered when pattern matching. In addition filtering supports the following "wildcard" characters:

- **?** means match any single character
- ***** means match any number of characters.

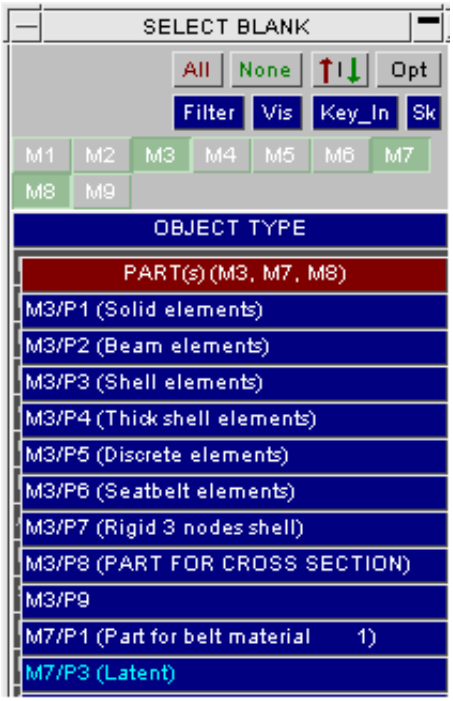
These may be used any number of times in a string, for example "***quick*fox***" will match the string "**The quick brown fox jumped**".

In addition the string has a "virtual" ***** at its beginning and end, so typing in **seat** is exactly equivalent to typing in ***seat***.

Here are some examples showing how filtering works, demonstrating the use of wildcards.

Here is the unfiltered table	Here "**diam**" has been typed into the filter box, finding 3 entries	Here the filter box has been changed to "**2*diam**", giving just one entry.
		

Filtering by ***Mnn*** Model "tabs"

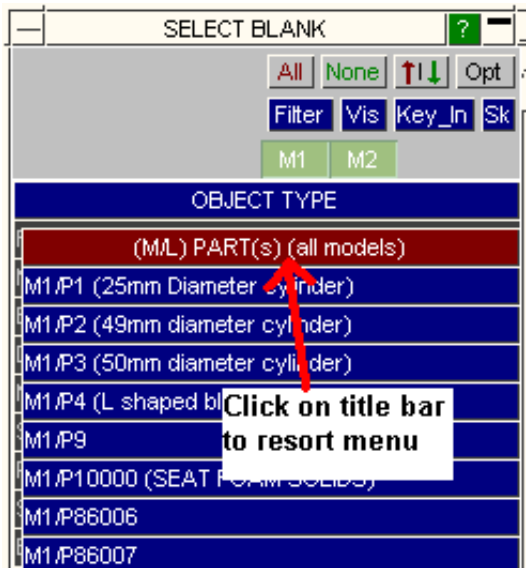


If your database contains more than one model then ***Mnn*** "tabs" will automatically be shown at the top of all menus where selection across multiple models would be legal: blanking in this example.

Using these tabs is identical in effect to filtering "By model", and will limit what is shown in the menu below. In this example only parts from models 3, 7 and 8 will be shown.

If models are made inactive using **Model > List** (see section 3.0.1) then their ***Mnn*** tabs will automatically be unset in all menus.

Sorting object menu contents



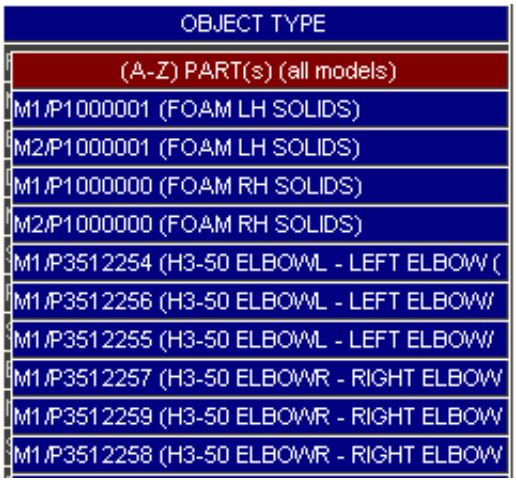
By default object menus are present in **M**odel/**L**abel order, referred to as M/L, meaning

- All items in Model #1, sorted into ascending order by item label
- All items in Model #2, sorted by ditto
  etc

However it is possible to resort object menus dynamically by clicking on the menu title bar, which will cycle the sort process through:

- A-Z (alphabetic sorting by item title)
- Z-A (reverse alphabetic sorting)
- 0-9 (ascending numeric sorting by item label)
- 9-0 (reverse numeric sorting)
- M/L (default **M**odel/**L**abel sorting)

Further clicks will repeat the cycle above.



For example in this case sorting by **A-Z** gives the following.

Note that alphabetic sorting has ignored both model id and item label, and considered only titles. Since there are two (very similar) models in this example this has resulted in rows from M1 and M2 being interleaved.

Note also that the menu title row has changed to show the current sort method, here showing:

```
(A-Z) PART(s) (all models)
```

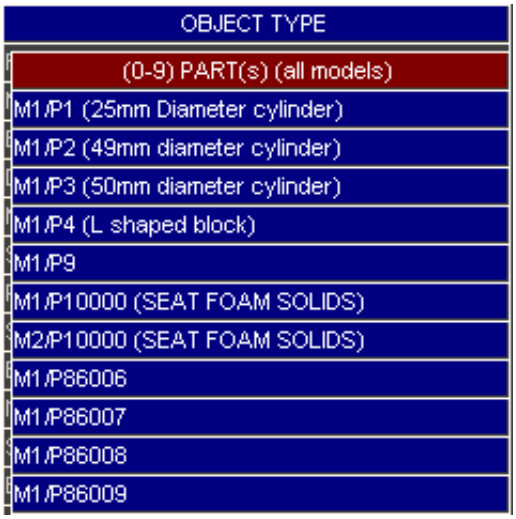
In this example two further clicks have given `0-9` sorting

Note again that the model id has been ignored, and sorting is by strict label id regardless of model.

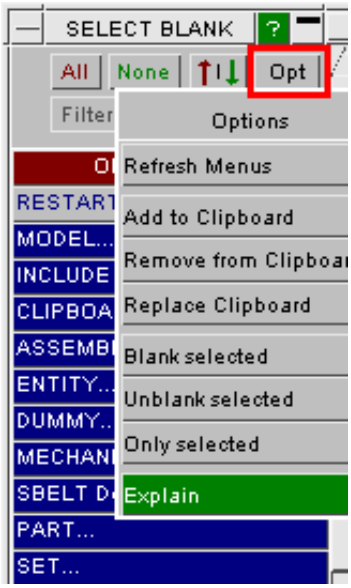
Where items in 2 or more models share the same label, for example here P10000, they are shown in ascending model order, but otherwise the model id is ignored.

As before the top row of the menu shows the current sort method, now:

`(0-9) PART(s) (all models)`



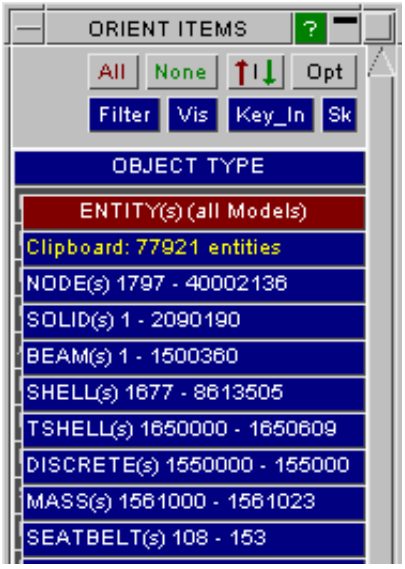
The **Options** popup menu



`Refresh Menus`	Refreshes the current menu, updating it to reflect changes to things such as titles and set contents which may not have triggered an automatic menu refresh
`Clipboard Add`	Adds the current selection to the Clipboard. Existing clipboard contents remain, and only new items are added.
`Clipboard Remove`	Removes the current selection from the Clipboard. If the selected items are not already in the clipboard then no change takes place.
`Clipboard Replace`	Replaces the clipboard contents with the currently selected items. Any existing clipboard contents are lost.

Clipboards are model-specific, and replacement only takes place in the models active for this object menu. For example a selection made to change set contents, implicitly for a single model, will only replace the clipboard contents for that model; whereas selection for Blank, which is multi-model, will replace the clipboard contents in all active models.

The clipboard may be used in a range of different ways in PRIMER, see section 6.8 **CLIPBOARD**, but in the context of object menus it may be used to save and reuse the current selection.



Once the clipboard in a model contains something then in any Object Menu context where:

- Multiple selections are legal
  and
- The clipboard contains one or more items of the specified type

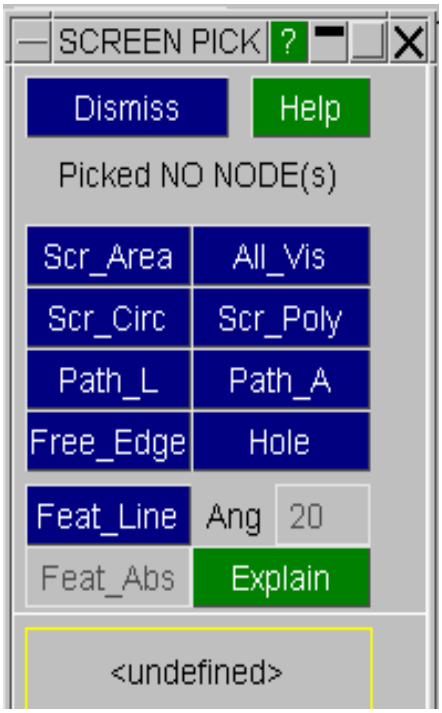
Then a "**Clipboard: nnn** <*item type*>" row will appear at the top of the menu.

Selecting this row is a "one click" way of selecting all items on the clipboard which match the current type(s). This can mean multiple types, as in the example here using **`ENTITY`**, where the clipboard contains a mixture of entity types.

`Blank selected`	Blanks the selected items
`Unblank selected`	Unblanks the selected items, turning on their entity visibility switches if necessary in order to make them visible
`Only selected`	Blanks everything except the selected items, again turning on their entity visibility switches if required.

When "**`Only`**" is used all other items in all models will be blanked, regardless of whether this object menu refers to a single model or multiple ones. This is necessary if "only" the selected items are to be visible.

Using **Vis**(ible) screen-picking to select items



In addition to Quick Pick operations, Primer allows multiple menus to be active concurrently, each of which may require a picking operation (e.g. Modify Part, Delete Element, etc).

As well as selecting items from a menu you may pick them from the screen using the mouse. Any permutation of explicit selection from the menu and screen-picking may be used, they are simply different ways of performing the same task.

When such a menu is invoked or bought to the front using the menu tabs, it automatically takes control of interpretation of screen-picking; this is indicated in the Quick Pick control. The user can cause any capable menu to control screen picking by clicking the white cross in the top-left of the menu. Quick Pick control can be restored either by clicking the white cross in the top left of the graphics area, or from the drop-down in the Quick Pick control.

The mouse button during picking used is significant:

- LEFT mouse button selects
- MIDDLE button rejects the most recent selection
- RIGHT mouse button deselects (the picked items are removed from the list of currently selected items)

Mouse button usage is described in more detail below.

Screen-picking is always "live" in the graphics window once you have selected an object category that is capable of being picked, it is not necessary to select **Vis** explicitly, and it can be accomplished in a range of ways:

"Scalar" picking of single items	Just click on the approximate centre of the item to select it.

"Rectangular Area" picking of a range of items	Click and drag out a rectangular area. Everything within the area is selected.
Within the **Vis** panel there are the following further screen-picking options:	
Scr_Area	Is an alternative way of defining a rectangular area by picking two points at opposite corners. Eligible items within the rectangle are selected.
Scr_Circ	Selects within a circular area. Click on the centre of the circle, drag out to define its radius and release to select eligible items within the circle.
Scr_Poly	Selects within an arbitrarily shaped polygon. Select three or more points (up to a limit of 100) to define the polygon, and close it when complete. All eligible items within the polygon will be selected. While the polygon may be any shape, and include concave sections, it should not be excessively complex; and it is also recommended that it should not have crossed edges since while these will work the algorithm used to distinguish "inside" from "outside" may become confused by them.
Path_L	This mode applies only to the picking of Shell Nodes or Shell Elements. Pick any number of Nodes/Shells in a Shell part. User selections are connected via a shortest path of adjacent entities with all path entities in between also selected.
Path_A	This mode applies only to the picking of Shell Nodes or Shell Elements. Pick any number of Nodes/Shells in a Shell part. User selections are connected via a shortest path of adjacent entities with all path entities inbetween also selected. User selections can be then used to create an enclosed polygon. The final selection will also contain entities within/outside that polygon.
Free_Edge	This mode applies only to the picking of nodes in a Shell Part. Pick two nodes on a free edge. All the nodes situated in the edge and between the two nodes selected will be also be selected.
Hole	This mode applies only to the picking of nodes in a Shell Part. Pick one node on the edge of a hole formed by shell elements. All the nodes on the border of the hole will be selected.
All_Vis	Will select automatically all "visible" items that are eligible. Note that "visible" in this context means what is displayed on the screen, but not necessarily what you can see. Items hidden behind other items are still "visible", as are items off the border of the current window. A more precise definition of "All Visible" would be "things which would be visible in a wireframe plot autoscaled to fit in the current window".
Feat_Line and associated angle	This mode applies only to the picking of 2D and 3D elements, and of nodes. • A single node or element is picked and its outward normal vector is computed. • The pick is then propagated across the mesh of that element type so long as the difference in angle at a common edge between the outward normal of an element and its neighbour is not greater than the Feature Angle value. This has the effect of propagating a pick across a flat or (typically) gently curved surface, selecting all nodes or elements on that surface.

Path Line picking

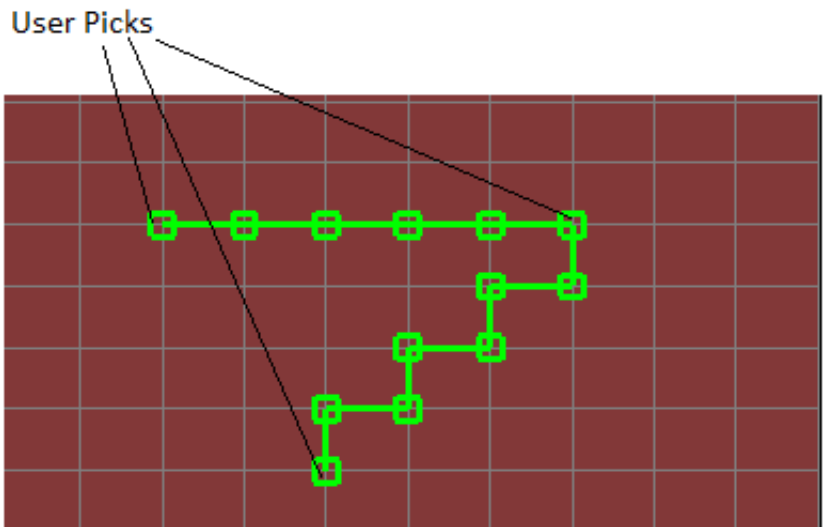
This is a special mode in which user can keep picking Shell Nodes or Shell Elements and PRIMER will keep creating the shortest path between consecutive user picked entities with all path entities in between also selected .

Panel for Path Line Object Menu Selections

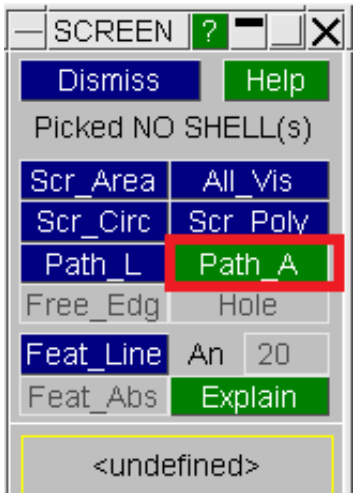
Add/Remove Pick List	Adds/Remove all the highlighted entities to the Pick List.
Reject Last	Rejects last selected entity and loses the last created path.
Reject All	Rejects all selections in the current path.
Done Picking	Closes the Path Line panel.
Abort Picking	Rejects all highlighted entities and closes the panel.
Help	Shows the help for this Path Line panel.

Path Line Example

PRIMER highlights entities joining the consecutive user picks via shortest path



Path Area picking



This is a special mode in which user can keep picking Shell Nodes or Shell Elements and PRIMER will keep creating the shortest path between consecutive user picked entities with all path entities in between also selected .

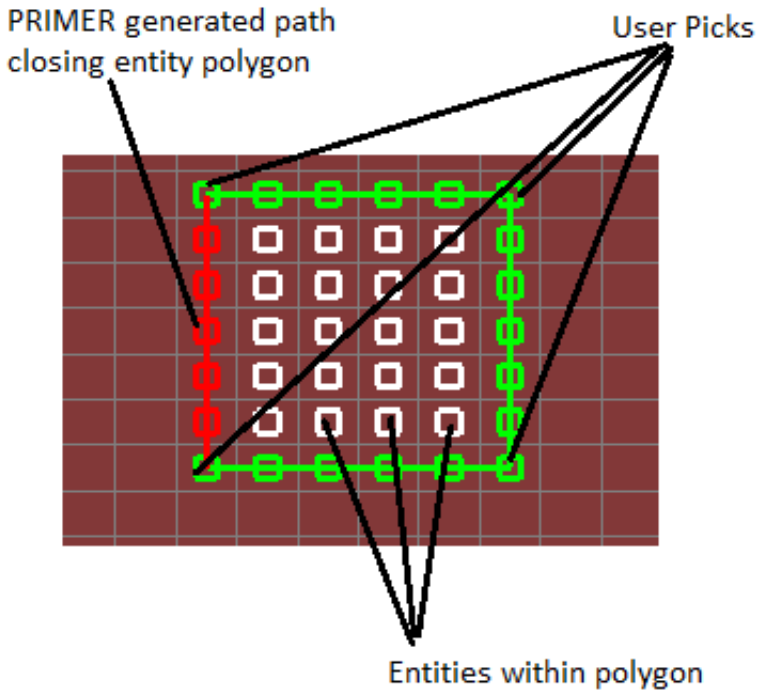
User selections can be then used to create an enclosed polygon. The final selection will also contain entities within/outside that polygon.

Panel for Path Area Object Menu Selections

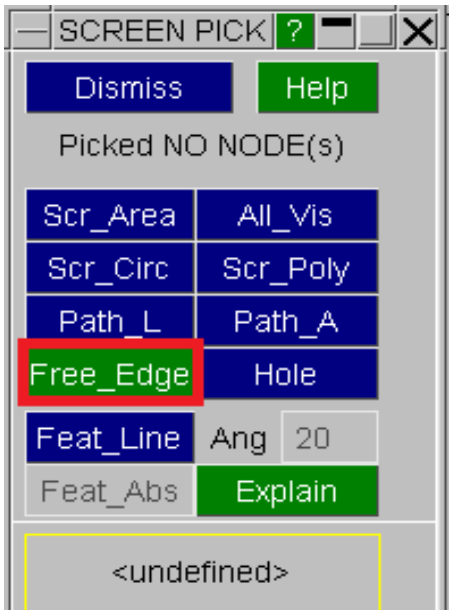
Add Pick List/ Remove Pick List	Adds/Remove all the highlighted entities to the Pick List.
Cycle Area	PRIMER cycles through and highlights all the potential areas of entities that are either enclosed within the boundaries of polygon or are enclosed by boundary of polygon and free edges of Model.
Reject Last	Rejects last selected entity and loses the last created path.
Reject All	Rejects all selections in the current path.
Done Picking	Closes the Path Line panel.
Abort Picking	Rejects all highlighted entities and closes the panel.
Help	Shows the help for this Path Line panel.

Path Area Example

PRIMER highlights entities joining the consecutive user picks via shortest path. Then closes polygon by calculating the shortest path between first and last user picks. And also highlights the entities within the polygon.

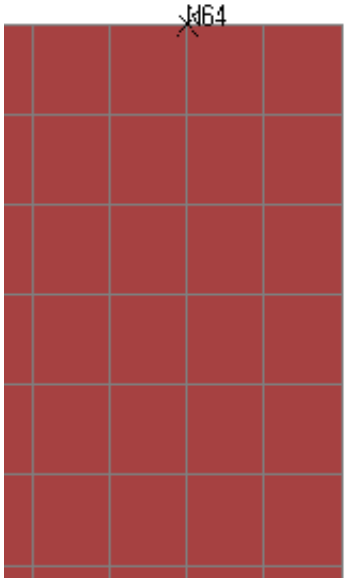


Free edge picking

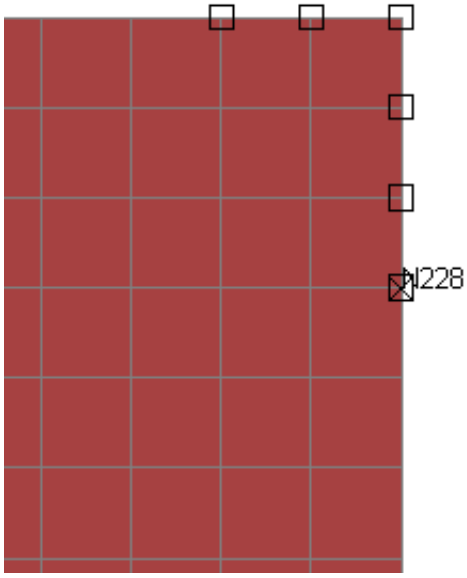


This is a special mode in which picking two nodes of an edge will result in the selection of all the nodes between them.

Selection of N64 as Node 1



Selection of N228 as Node 2

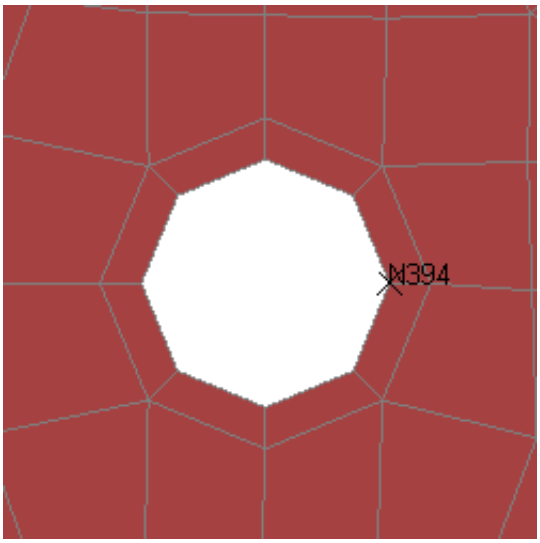


All the nodes on the free edge between N64 and N228 (shortest distance) are also selected.

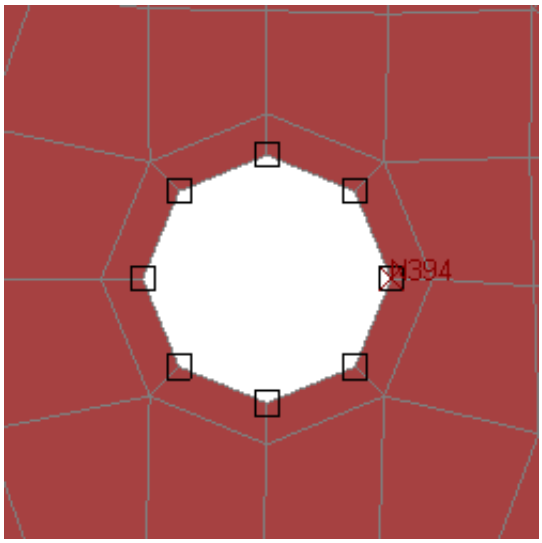
Hole picking

This is a special mode in which picking a node on the edge of a hole will result in the selection of all the nodes around the edge of the hole.

Selection of N394 on the edge of the hole

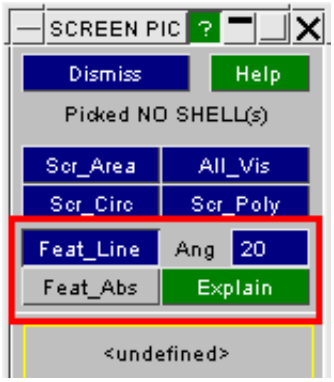


All nodes round hole selected



All the nodes on the edge of the hole are selected.

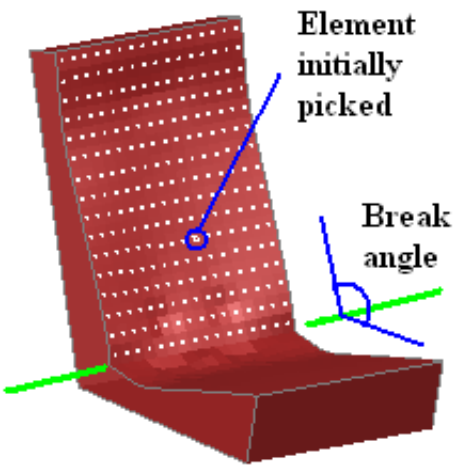
Feature Line picking



This is a special mode in which a pick on a single element or node is propagated across a connected surface mesh, and all such elements or nodes on that surface are selected. The edges of the surface are defined by "feature lines", which are edges at which the difference in angle between adjacent elements is greater than the stipulated Feature Angle.

It is only applicable to meshes of 2D and 3D elements (ie Shells, Solids, Thick shells and Segments) and to Nodes on such a mesh.

In this example using a crudely meshed seat, made from a single Part, the user has selected a single Shell element in the middle of the backrest with the feature angle set to 20 degrees.



Selection has propagated to the top and side edges, but has stopped where the backrest meets the bottom because the "break angle" in the mesh at this point exceeds 20 degrees.

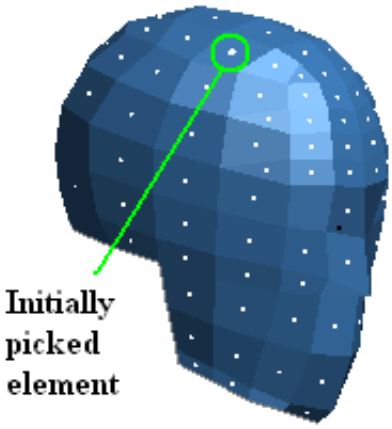
The following rules apply to feature line picking:

- Feature line propagation only applies to single picks on eligible elements or nodes. It will be ignored if an area pick of any type is used.
- For 3D elements (solids and thick shells) propagation will be from the selected element face only, and will not "track" across multiple faces of this or subsequent elements. This has been found to give a more natural determination of a "surface".
- For 2D elements (shells and segments) the winding order of the nodes in elements meeting at an edge are compared, and the computed normal of an element with nodes numbered in the opposite direction will be reversed before angular comparison. This means that adjacent elements can be "upside down" and still pass the angle test.
- For nodes the surface outward normal is determined from the average of the elements meeting at the node initially picked, and then extrapolation across the surface proceeds as for elements above with the nodes on eligible elements being selected. The results may be unpredictable if the initial node is badly chosen, for example a node on an edge or corner. If a node is connected to both 2D and 3D elements the 2D elements only will be used for normal determination and subsequent propagation.

Feat_Abs : Optional "absolute" feature angle

An option in Feature Line picking above is the definition of an "Absolute" angle. This applies exactly the same logic as described above with the additional restriction that:

- the angular difference between the outward normal of the original node/element and this one must also not exceed the feature angle.

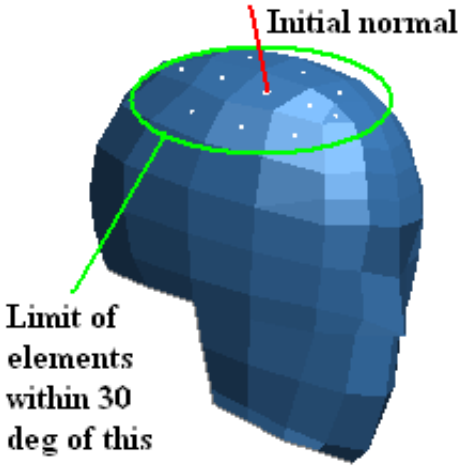


This has the effect of limiting propagation over curved surfaces where the angular difference between adjacent elements may be small, but the overall surface curvature exceeds the specified angle.

In the first image here, using a feature angle of 30 degrees, no "absolute" angle has been specified. A pick on the top of the head has propagated down to all the elements visible here since the angular difference between adjacent elements is < 30 degrees even though that between the top of the head and the chin is closer to 90 degrees.

The second image is exactly the same, except that the "absolute" angle option has been set, meaning that only elements with normals within 30 degrees of the original element's normal (shown approximately in red here) have been selected.

As this example shows propagation across the curved surface has been limited, which can be useful for selecting a geometrical subset of elements on a curved surface.



Rules applying to screen-picking

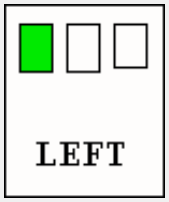
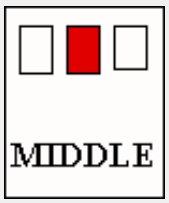
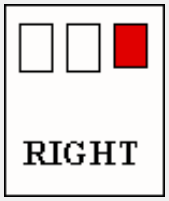
Screen-picked entries (by any method) go into the cursor list, of which the 10 most recent entries are shown in this box.

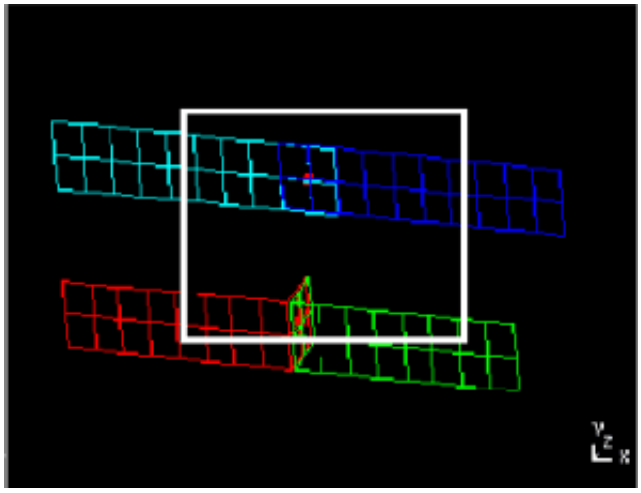
Note that:

- The current **FILTER** setting also applies to screen-picking: ***you will not be able to pick an item that has been filtered out.***
- Multiple (Area, Circle or Polygon) picking is only available in contexts where it makes sense. If, for example, you are picking a single node for an element you will not be permitted to drag out an area. The cursor symbol gives a prompt: a "cross" permits only scalar picks, a "hand" permits multiple picks.
- When 3D elements are picked by area or polygon the treatment of elements inside a mesh, which are not drawn because all their faces are "internal", depends on the **AREA_PICK** setting below.
- Screen-picked items can be rejected in a range of ways - see below

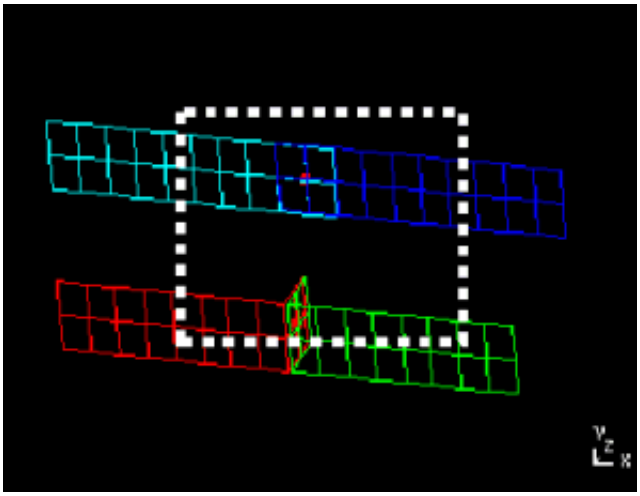
Rejecting/Applying items that have been screen-picked.

Picked items can be rejected (deselected) in exactly the same way that they were selected by using the middle and right mouse buttons. You can also "Apply" the selection. The defaults are shown below, however these are configurable in the menu attributes panel:

Mouse Button	Default Function
	***Selects*** items: by singler pick, rectangular area, circular area or by arbitrary polygon as described above. A thin, solid white line is used to define areas and polygons.
	***Rejects*** the most recent selection: "last in, first out". The picking stack remembers all picks in the current operation, and repeated middle mouse clicks will back-track up it until it is empty. Area (of any type) picks are rejected en-bloc, ie items selected within a single area pick are also rejected via a single middle mouse click.
	***Rejects***: • What is explicitly selected (scalar pick) • All items in the area (multiple pick) A thick, broken white line is used to define areas and polygons.
	Apply Selection: • Apply the current selection (for example when picking nodes to create a CONSTRAINED_NODAL_RIGID_BODY, this option will create the entity (this can speed up the process of creation of many entities through picking). Note this is not assigned to a mouse button by default, but can be assigned to the middle or right mouse button via the menu attributes panel.



Left Mouse button ***Selects***:
Solid borders for areas and polygons.



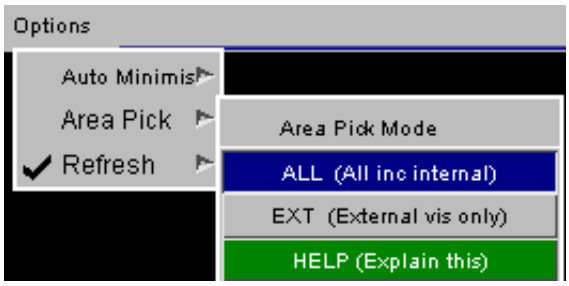
Right mouse button ***Rejects***:
Broken thick borders for areas and polygons.

Area_Pick: What is "visible" when area or polygon picking

For anything other than 3D elements the test is simple: if it has been drawn, even if it is obscured by something else, it is "visible".

For 3D elements, solids and thick shells, the question arises of how to treat elements that are interior to a solid block of mesh. These are not actually drawn since internal face culling removes them from the graphics pipeline, so are they "visible" or not?

This is determined by the setting of the (cursor) **AREA_PICK** parameter in the options popup menu.



ALL	Selects all 3D elements through the thickness, regardless of whether of not they have been culled due to internal face removal. This has the effect of punching a hole completely through a 3D mesh. From Primer release 8.2 this is the default behaviour. In earlier releases no option was given, and the behaviour was implicitly **EXT**ernal as defined below.	
EXT	Selects only those 3D elements which have actually been drawn, ie those which are "EXT"ernal. This tends to have the effect of "peeling the outer layer of the onion": only the outer layer is selected, and successive picks are required to make a hole right through the mesh.	

Using KEY_IN to type in selections

It is also possible to type in selection labels by invoking the **KEY_IN** box. Valid syntax is:

- Single labels: 1 101 27 93
- <start> to <end>: 1 to 21 99 : 1000

Or any combination of these. (Note that either "to" or ":" may be used to denote a range.)

"Key in" syntax when model and/or type codes must be defined for labels.

In the example above the model id and type code (Part) were both known, so simple numbers were adequate.

However in some situations multiple types may be possible (for example "element" permits "solid, shell, beam, ...") and the type code acronym must prefix the labels.

For example to select:	**Solid 27** ***and*** **Shells 1 to 20** ***and*** **Beam 99**	You must define	**H27 S1:S20 B99**

It is also possible that selection across multiple models will be permissible.

For example to select:	**Model 1: Solid 27 and Model 2: Shells 1 to 20 and Model 3: Beam 99**	You must define	**M1/H27 M2/S1:M2/S20 M3/B99**

How explicit menu selection, screen-picking and keying-in work together

These three methods of selection co-exist with cumulative effect: they are simply alternative ways of selecting objects for processing and are designed to be used together.

Selecting something by screen-picking or typing in its label will automatically depress the appropriate menu row, likewise deselecting the menu row of an item that has been screen-picked acts like rejecting a pick, and removes it from the cursor list. You can use any combination of methods in any order to select items. (Screen picking, or keying in the label of, an item that has already been selected manually from the object menu is legal, but has no effect.)

Selections are *not* cumulative across different item types

If, for example, you select the type PARTs and then swap to ELEMENTs, the PARTs will no longer be selected - only the selected elements will be remembered. If you wanted to select all elements from one PART, and then a subset of elements from a second PART, you could do it as follows:

- Select object type ELEMENT;
- Set filter option "by PART" and select the first part;
- Select "ALL" elements (implicitly only in that PART);
- Unset the filter and select the required further individual ELEMENTs explicitly.

The selection list will contain the results of both categories of selection.

Selections persist following most operations (except BLANK and DELETE)

When you have made your selections, and carried out the relevant operation, the selections remain in memory if this makes sense in that context. Thus you could carry out some other operation on the same list, add to or subtract from them prior to a further operation, and so on.

However exiting from that operation (for example leaving the **ORIENT** menu) will destroy any current selections. For this reason it may be better to iconise a window to get it temporarily out of the way, rather than to **DISMISS** it. The former will not affect its selection status, whereas the latter will destroy it!

Exceptions are:

BLANKING	once operated on the items chosen are deselected.
DELETE	they will no longer exist!

To delete all current selections and start again

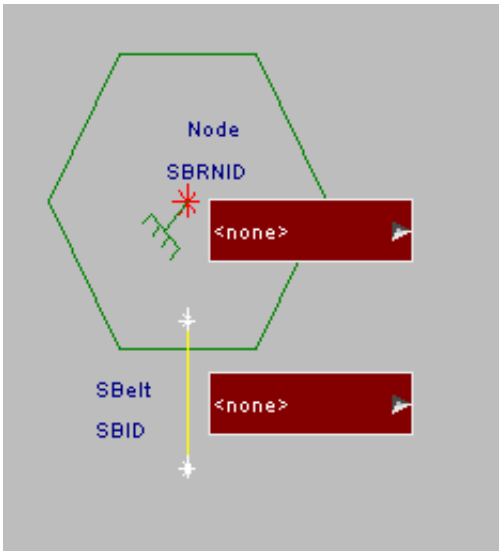
Using the **RESTART** row at the top of the object menu has the effect of canceling all current selections, unsetting the current object category, and resetting the selection process to its initial state.

Exiting from the current operation menu also clears any current selections as described above.

Other selection methods

In many contexts where an individual item (as opposed to a list) is required PRIMER will use "popup" menus to select things.

For example the creation of retractor elements requires, among other things, the definition of a central node and a seatbelt element.



As shown here you can type a label into the relevant text entry box, or use the right mouse button to invoke a selection menu which will have the standard options:

These standard options allow you to (screen-)**PICK** the item directly, or **SELECT** it from a standard selection as described here.

Picking and sketching will only be available for viewable items (for example you can’t pick a loadcurve).

The **CREATE** and **EDIT** functions will only be available for those items which PRIMER currently has the ability to

create/edit.

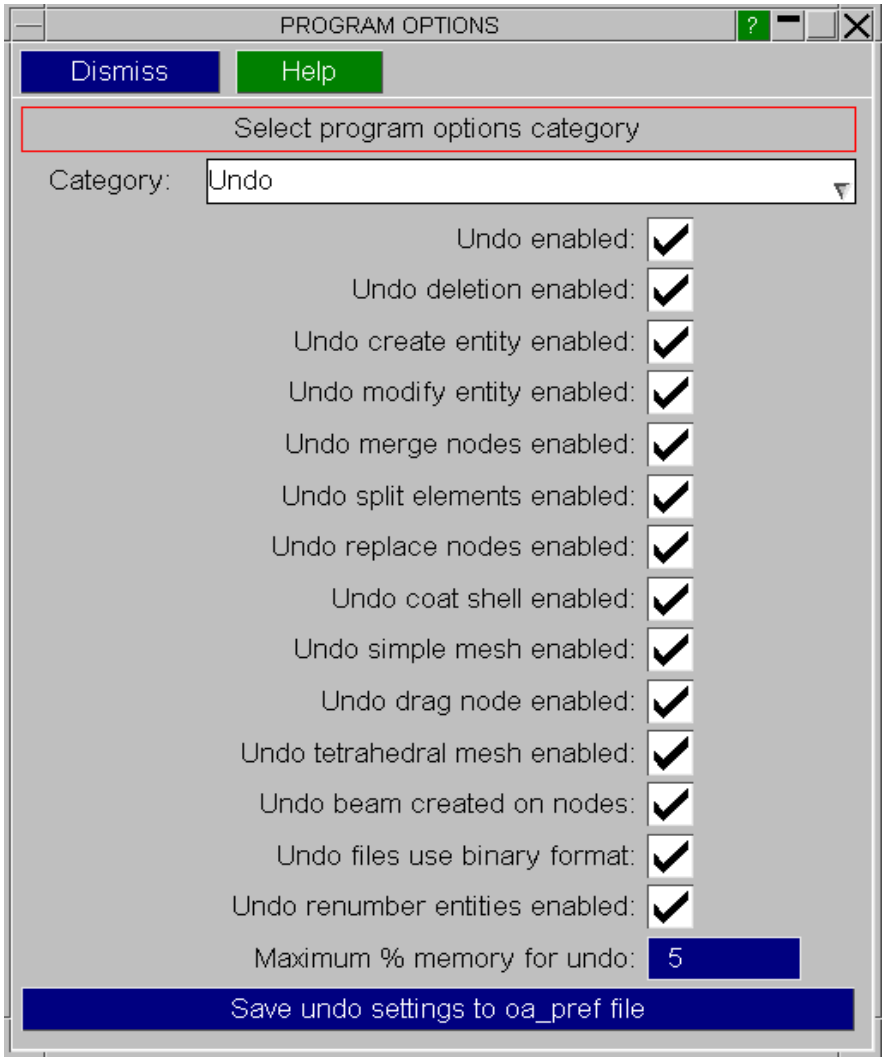
Given a little thought these concepts are straightforward, but you need to take a little care. For example selecting a **PART** is not the same as selecting all the **ELEMENTS** in that part:

- Deleting by ELEMENT will delete the elements but leave their PART definition intact (if redundant).
- Deleting by PART will delete both elements and the PART definition itself.

(The interaction of hierarchies and deletion is explained further under section 6.36**REMOVE**.

2.14 Undo

There is a generic **UNDO** button at on the top menu bar in PRIMER. This allows you to UNDO recent operations such as entity creation, modification or deletion. To the right of the UNDO button there is a text indication what will be undone when you click on the button. The UNDO operation involves writing temporary files to disk, so at times can be slow. Because of this the various types of UNDO can be turned on/off in the program options panel:

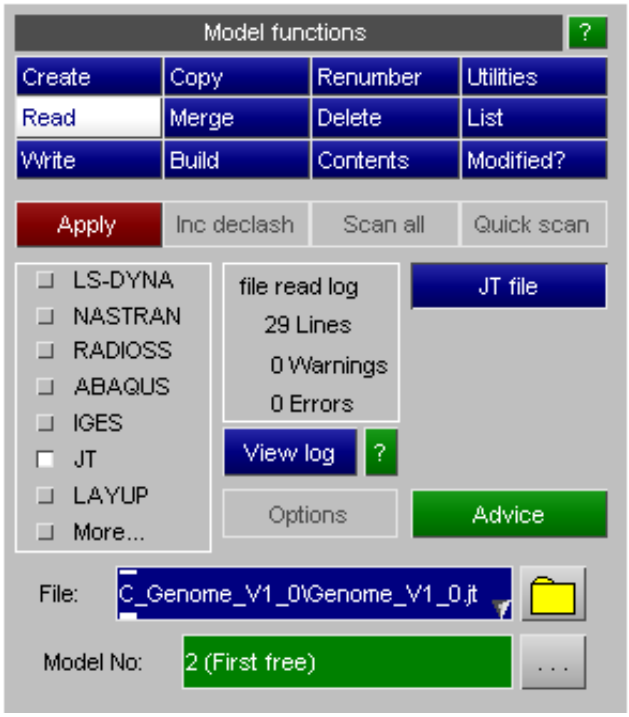


3 Model manipulation

Quick links to sections:

3.0 How PRIMER treats models
3.1 Creating a new model
3.2 Reading in models
3.3 Writing out models
3.4 Merging Models
3.5 Copying Models
3.6 Deleting Models
3.7 Renumbering Models
3.8 Model Contents
3.9 Checking the correctness of Models
3.10 Operations on Models
3.11 Viewing Models
3.12 Memory Management and Usage
3.13 Include Files
3.14 Include Transform
3.15 Model Database

The **MODEL** menu (here showing the **Read** sub-panel)

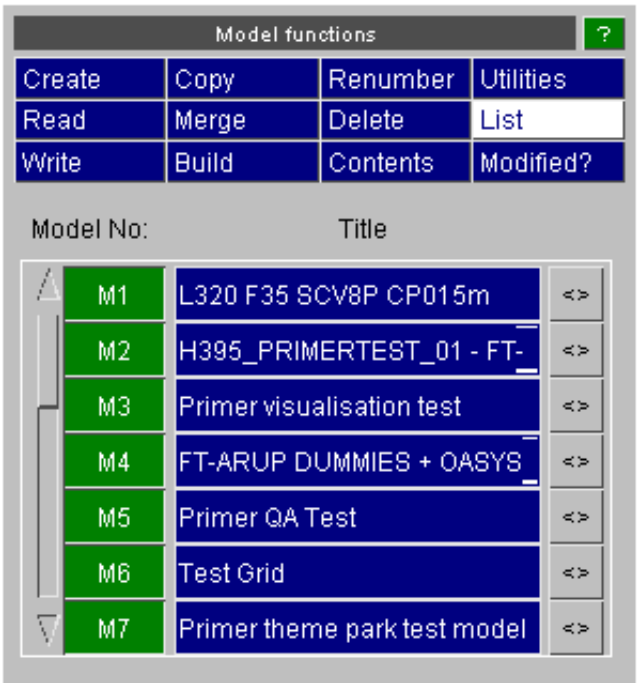


The model menu can always be accessed via the **MODEL** tab in the window control area.

This section describes how to read model data from disk, manipulate it within PRIMER, and write it out again.

3.0 How PRIMER treats "models"

PRIMER is unusual in that it permits you to work with concurrent multiple models.



A "model" within PRIMER is a self-contained set of data, derived from any source. It need not be a complete input deck, and might indeed only have a few nodes and elements or even be empty. Up to 255 models may be stored and processed simultaneously, and each is kept totally separate until the user takes some action which merges two or more of them.

The purpose of this approach is to permit output models to be built up from sub-assemblies of other input models. For example to build a house you might have three input models: "walls", "door" and "window". The output model ("house") could be assembled from "walls" merged with "door", and possibly five copies of "window" located in different places.

In the example above there are currently 9 models in memory, but only 7 at a time can be displayed in this panel, so a scroll-bar has been added.

3.0.1 **Model > List**: Listing models and setting their "active" status.

The "Model No:" column:

Model numbers are arbitrary in the range 1 to 255, and may be changed at will. Model #0 is reserved for internal use.

- Shows the ids of each model (**M***nnn*). Numbers are assigned automatically to models in PRIMER in ascending sequential order from #1 when they are read in, you can change these numbers at any time using **Model > Renumber** (see section 3.7).

- If the "Model No:" entry button is selected (as they all are here, shown by the green colour) then that model is available for display. If de-selected (coloured red) then that model will not be drawn. This is the highest level of display control, and provides a quick and easy method of un-cluttering the display.

Deselection via **Model > List** is the recommended method for suppressing models that are to remain in the database, but are not currently being worked on since it not only stops them being drawn, but also:

- Automatically deselects their **M***n* "tabs" in selection menus (see section 2.13), meaning that their contents are not shown by default.

- If only one model is active (green) then PRIMER is able to assume that this is the one you want to work on, and it is able to eliminate a layer of "which model do you want?" questions in many selection contexts.

The "Title" column:

- Shows the title of all models, as read from their **`*TITLE`** cards in the input deck.

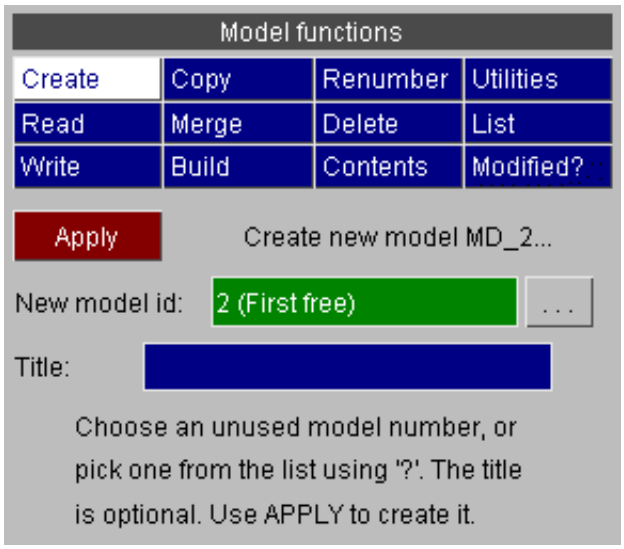
- A model's title may be changed by typing a new string into its "title" button. (It can also be changed in the **Keyword > Control** editing panel)

- The **[<>]** button toggles between display of model title and model filename. (It has no effect on the actual title of the model written out after the ***TITLE** keyword.)

3.1 MODEL > CREATE

Creating a new model - an empty model is created.

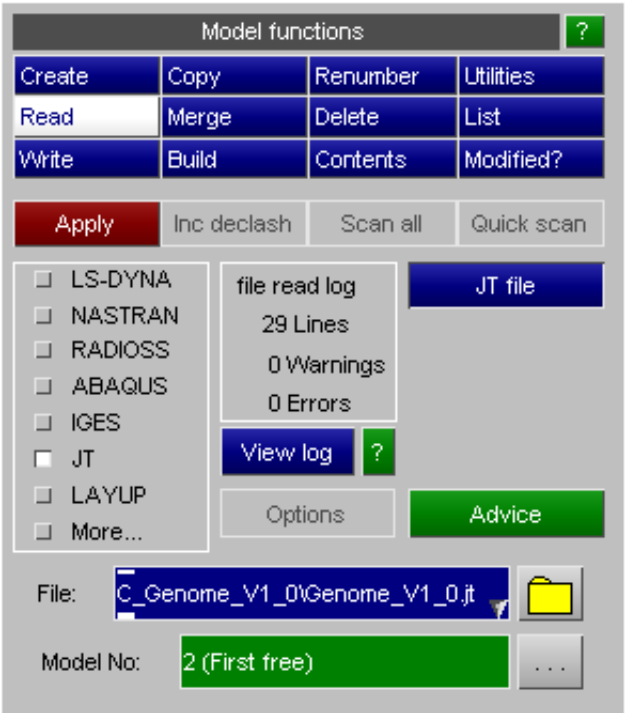
The **CREATE** command allows you to generate a new model without reading anything from disk. You must define its internal number, (which must not already be in use), and optionally give a title.



(Note that any model's title can be modified at any time by simply over-typing it in the "Title" column of the **List** menu.)

It is not necessary to create a model before reading data into it or using it as the output of some other **MODEL** >command: all the model-based commands described below will generate a new output model automatically if required, and indeed the default mode of most of these operations is to do so.

3.2 MODEL > READ



Reading in models from disk.

PRIMER's native internal data structures are based on LS-DYNA keyword format, with some additions for specialist purposes.

Files may be read in a variety of formats, some of which require a considerable degree of translation to convert them to PRIMER internal data.

Section 3.2.2 below summarises what is read and how it is converted: more details of format conversions are given in Appendix VI.

In the example a DYNA3D keyword file is to be read into model #2 (which is the first free model number).

Compressed and binary LS-DYNA input files

From PRIMER 15 onwards LS-DYNA files that are compressed in .gz and .zip format may be read and written directly. In addition a new proprietary binary format that is both more compact and gives faster i/o may be used, see Appendix VI for more details. PRIMER 16 onwards will read .zip archives that are encrypted.

3.2.1 Options for LS-DYNA Keyword files only

Apply Reads the file in the nomal way, and stores it in the database

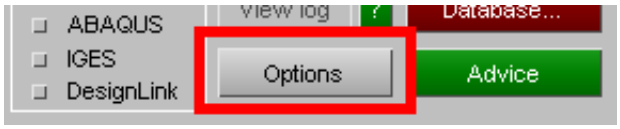
Inc Declash Reads a master keyword file followed by any number of include files, and increments labels of items in include files if necessary so that the resulting model does not contain any labels clashes. It attempts to preserve existing labels where possible, but if clashes occur it is inevitable that some relabelling will take place.

Scan all Scans the file looking for include files, including looking for "nested" include files (ie include file referring to child include file. An include file tree is built, and the Include panel is built: see section 3.13.

Quick scan Scans the master file only looking for include files, nested include files are ignored, and builds an Include file panel as above. (See section 3.13)

Options Maps a sub-menu of options to control keyword input file behaviour:

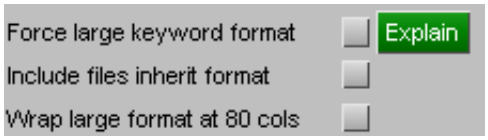
Options: Controls many aspects of reading LS-DYNA files



The defaults for reading LS-DYNA files are chosen to give behaviour that should be appropriate for the vast majority of input decks, and in most cases you will not need to visit this panel. However problem decks that contain errors, or keywords that contain undocumented extra data fields, may become readable if the relevant options are used.

It is also possible to tune the hardware settings used for file reading and writing. This sometimes helps with access speed when files are on a remote network disk.

Options: Force large keyword format



Large ("long") keyword format expands all data fields in an input deck to 20 columns wide, permitting larger labels to be used. See "Getting started", "General card format" in Volume I of the LS-DYNA manual for more information.

Normally the format of an LS-DYNA input deck is determined automatically from a "**`long=...`**" field on the **`*KEYWORD`** card, or by trailing "**`+`**" or "**`-`**" suffices after other keyword headers. However it is possible to omit these yet to force LS-DYNA to read a file in long keyword format by adding "**`long=y`**" to the execution line, therefore PRIMER also permits the format to be stipulated in this way using this option.

If this option is set, but a keyword input deck includes conflicting directives such as "**`long=s`**" on the *KEYWORD card then these directives will override this option.

Options: Include files inherit format

Normally each new include file is read (from the point of view of large/small format) as if it were a free-standing file, meaning that if it does not contain an explicit "**`long=...`**" field it is treated as being in short format. In this way a model can contain a series of include files using different formats.

However it is possible that some pre-processors may write include files in long format without inserting the "**`long=y`**" field, relying on the fact that the software reading the file will "know" that it is in large format. To enable PRIMER to read these decks turn on the **Include files inherits format** option, which will apply the format of the parent file to each include file it references.

This setting can be made permanent via the following preferemce:

`primer*inherit_file_format: true | false` (default **`false`**)

Options: Wrap Large format at 80 cols

The syntax of "long" format in LS_DYNA changed in LS-DYNA 7.1, November 2013.

- Prior to this date all lines in long format were limited to 80 columns, and a new line was started if the card contained more than 4 data fields resulting in 2 or sometimes 3 physical lines per "Logical" data card line.
- From LS-DYNA 7.1, Nov 2013, onwards "long" format lines are permitted to be up to 240 columns long, meaning that each logical line in the keyword manual also occupies a physical line in the file.

The newer (7.1) format is now definitive, and the older format is obsolete. Since large format was not really used prior to early 2014 all large decks are expected to use the later syntax, however there may be a limited need to convert "old" decks to the "new" format, and the option to **Wrap Large format at 80 cols** permits this by making the PRIMER keyword reader obey the old rules.

(Note that when writing large format PRIMER uses the "new" syntax by default, but it is possible to write decks using the older syntax.)

Options: Pre-read **`*PARAMETER`** cards

Generally PRIMER does a good job of reading decks that use **`*PARAMETER`** cards, even when the definition of a parameter only appears after it has been used. In order to avoid the time delay involved in performing two read passes it handles this "used before defined" problem as follows:

- If this is not an item's label field the "unknown parameter value" of zero is inserted
- If this ***is*** a label field then a special non-zero internal label value is inserted (zero won't work for labels)
- Once the complete deck has been read and parameter values are known it revisits these cards ad substitutes the correct values.

However this approach can fail in some cases. In some situations the temporary substitution of zero in a data field causes an error in the keyword. Another example is where the format of a data card depends on the value of fields within it, and while substituting zero may be legal it means that the meanings of other data fields on the card are interpreted wrongly. Yet another situation that may cause problems is very complicated usage of **`*INCLUDE_TRANSFORM`** that depends on parameters.

In these situations the solution is to make sure that the value and type of all parameters are known before their names are referenced, as in this way the correct values can be used ab initio with no need to go back and reformat a card. Selecting this pre-read option achieves this by making keyword reading a two phase operation:

1. The complete input file, and any include files, are scanned for *PARAMETER cards only so that their information is known.
2. The whole input deck is then reread, using the now known parameter values.

Although the scan phase (1) is much faster than a normal keyword read (2) it will still take a significant amount of time, and it is recommended that you perform the following reorganisation of your input deck to avoid the problem in the first place. Your goal is to make sure that all parameter definitions are "known" before they are "used", and you can achieve this as follows:

Suggested methods of resolving "out of order" parameter problems in input decks.	
Best method:	Define all *PARAMETER cards explicitly at the top of the master input file, before any other keywords and in particular before any *INCLUDE files in which they might be used.
Also good:	Define all *PARAMETER definitions in one or more *INCLUDE files dedicated to this purpose, and put these at the top of the input deck, before any other keywords. You need to be careful that the order of include files does not get changed, but this is simple enough.
Acceptable:	Define *PARAMETER definitions used in a given include file at the top of that file, before any other cards that might use them. There are two weaknesses to this method: firstly parameters defined in include file A may end up being referred to in include file B, which may go wrong if B is read before A; secondly there is a danger of parameter name clashes if the same parameter is defined multiple times in different files. If you adopt this approach it may be worthwhile to copy all parameter definitions in the top of all include files, using the _LOCAL suffix or the *PARAMETER_DUPLICATION card to avoid problems with clashes. This has the merit that it makes include files "free-standing" as far as parameter usage goes, but it may be complex to maintain.
Bad!!	*Placing *PARAMETER definitions at the end of the master file, after any *INCLUDE definitions that may refer to them.*

Warning: PRIMER can handle "out of order" parameters, but LS-DYNA may not be able to do so. In particular LS-DYNA explicitly forbids a *PARAMETER_EXPRESSION to contain a reference to a parameter that has not yet been defined.

There are options inside PRIMER to reorder parameters to fix this problem, see "The order in which parameters are defined" under [keyword] PARAMETERS. However it is best to avoid this situation by paying attention to how input decks using parameters are organised.

Options: Convert <*expression*...> parameters to values

LS-DYNA permits "implicit" parameter expressions to be created in any data field by enclosing an expression in <...>. For example:

```
*PARAMETER
         R     scale      3.25
*NODE
   10,   <2.0*scale>,  2.0,  3.0
```

In this fragment **SCALE** is a conventional parameter of value **3.25**, and the X coordinate of node 10 will be **2.0 * scale = 6.50** (Using this syntax requires comma-separated format.)

PRIMER handles this by creating an "implicit" ***PARAMETER_EXPRESSION** associated with that data field, allowing it to be edited and "remembered" for subsequent keyword output.

However in a model with many <*expression*> definitions this results in many parameters being created, and this can make PRIMER very slow and unwieldy because parameters are expensive to store and process. Therefore an alternative of converting these <*expressions*> to their equivalent numeric values during keyword input is available, which works as follows:

- The "explicit" parameters defined using ***PARAMETER** cards (eg **scale** in the example above) are read and stored in the normal way.
- When an <*expression*> is found it is evaluated, and replaced with the numeric value. So in the example above **<2.0 * scale>** is replaced by **6.50**.

Clearly this means that the <*expression*> is lost, the field cannot be edited as an expression and it no longer "knows" that its value depends on parameter **scale**. If the deck is written out this loss becomes permanent and the values are "frozen".

So there are significant disadvantages to using this option and it needs to be used with some care and forethought, but it does solve the problem of processing files with a large number of distinct <*expression*> definitions.

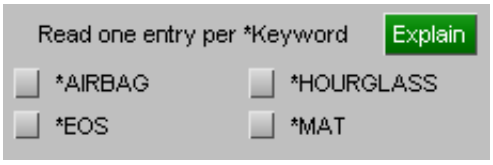
The default for this option is off, but this can be controlled by the preference

primer*convert_implicit_parameter: true | false

During keyword input PRIMER keeps track of how many <...> expressions have been encountered and issues a warning if this exceeds a threshold, offering to turn on this option for future expressions. By default this threshold is 100, but you can alter this by using the preference

primer*warn_num_implicit_parameters: <value> Where **<value>** is a +ve integer, default 100, or zero to turn the warning off altogether.

Options: Read one entry per *keyword:

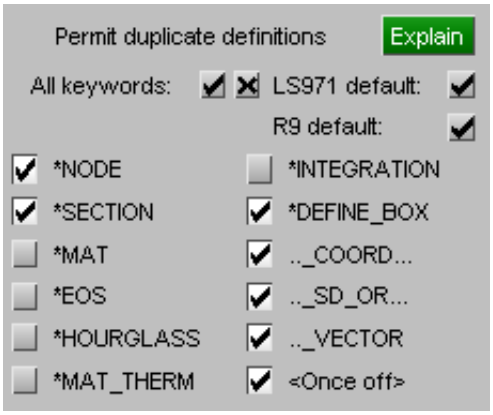


Possible way of reading newer input decks in an format that is not yet understood by PRIMER.

When an input deck for a version of LS-DYNA newer than the current version of PRIMER "understands" is read, keywords sometimes gain extra lines of data. This confuses the keyword reader, making it think that the input deck is invalid, and it rejects the definition - even if there is only one definition per *keyword header.

If this option is selected then PRIMER will read up to the number of lines it expects for the selected keywords (***AIRBAG, *EOS**, etc) and ignore any unexpected trailing ones by skipping to the next *keyword header, which usually makes these input decks readable.

Options: Permit duplicate definitions



LS-DYNA permits some keywords to be read multiple times. This applies both to:

- The special case of coincident **`*NODE`** definitions.
- Some other labelled keywords, eg **`*MAT`**
- Some "once only" keywords, eg **`*CONTROL_xxx`**

`*NODE` is a special case. Duplicate definitions with the same label in different include files will be merged if:

They have the same restraint codes (**`TC`** and **`RC`**)
AND
The distance between the nodes is less than 1e-8 times the average coordinate of the nodes. If a **`*NODE_MERGE_TOLERANCE`** card is defined then the nodes are also merged if the distance between them is less than this tolerance.

Duplicate nodes not meeting these criteria for coincidence are treated as an error during keyword input.

Coincident nodes are often used to "stitch together" models from include files, where multiple definitions of nodes are merged into a single entity. PRIMER emulates this by merging nodes using the same criteria so long as the **`*NODE`** entry in this panel is selected, which it is by default.

`*CONTACT` is also a special case.

LS-DYNA will permit duplicate labelled contact surface definitions, and from PRIMER V13 there is special logic to handle this.

See Section 5, Contact, Duplicates during input for details.

The treatment of duplicate nodes (and other item types) has changed in PRIMER V12 with the introduction of "cloning":

Prior to V12	Versions of PRIMER prior to V12 will "lose" the 2nd and subsequent definitions of duplicate nodes, meaning that if a model and include files containing these definitions is read in and then written out these nodes will be omitted.
From V12 onwards	From PRIMER V12 onwards these duplicate nodes will be remembered as "clone" definitions. This is implemented in the following way: • Only a single, unique, version of a **`*NODE`** is stored in the database. So it can only have a single coordinate and other properties, and all references to it point to this single unique definition. • Any number of "clone" definitions of this node can exist subject to the following rules: - Each such definition must be in a different include file. It is illegal to have two items of the same label in the same file. - A clone definition is simply a "placeholder" recording the fact that the node exists in some other include file. - It will appear in cross reference listings, but it will not be editable. • On keyword output an identical keyword for each duplicate node will be output in each include file where it was originally found. In this way duplicate nodes are no longer "lost", and will be rewritten in all include files in which they originally appeared. See [Tools] Clones for more information on how clones work, and how to process and visualise them.

A similar (and also undocumented) feature in LS-DYNA is that the following keywords may be multiply defined with the same label. In LS-DYNA it is likely, but not certain, that the last definition found will be used and the earlier ones discarded.

`*SECTION`	**`*MAT`** (Not in R9)	**`*DEFINE_BOX`**
`*DEFINE_COORDINATE_...`	**`*DEFINE_SD_ORIENTATION`**	**`*DEFINE_VECTOR`**

It seems probable that the following keywords either are, or will, also be permitted to occur multiple times in LS-DYNA:

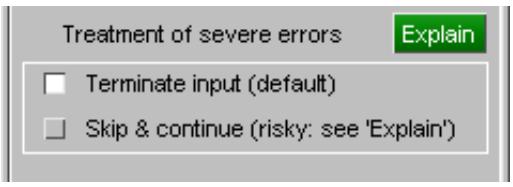
`*EOS`	**`*HOURGLASS`**	**`*MAT_THERMAL`**

Therefore PRIMER contains similar logic that will permit these cards to be defined multiple times if selected here.

Prior to V12	Warnings will be issued if such duplicates are found, and only the first such definition will be used with the remainder being discarded. If the model is written out the 2nd and subsequent duplicate definitions will be lost.
From V12 onwards	These definitions will be "cloned"during keyword input in much the same way as duplicate nodes, however unlike nodes there are no rules with LS-DYNA to permit duplicate definitions to be merged using some tolerance. In addition it is not always clear whether LS-DYNA will take the first definition encountered, discarding the rest; or will always use the last, discarding all earlier ones. Therefore PRIMER applies the following rules: • When keyword input is complete any duplicate definitions of a given keyword are checked for differences. • If all definitions are found to be identical then PRIMER silently discards all but one, making "clones" of the rest, and input processing continues without intervention from the user. • If differences are found these are listed, and the user is required to choose what action to take. Various options including "abort", "take first", "take last", "take definition in master file", etc are available. See [Tools] Clones, What happens if duplicate definitions contain differences? for more information about these options. It is ***strongly*** recommended that if differences are found the keyword input process should be aborted and the problem investigated manually. The reason being that LS-DYNA will not generally warn you about this problem, and if supposedly identical definitions in different include files are not in fact the same the results you get from an analysis may depend on the order in which the include files are read. (Special rules apply to **`*NODE`** cards, as described above. It is an unconditional error if duplicate but non-coincident nodes are found during input.) Assuming that duplicate definitions are found to be identical PRIMER will maintain a single unique definition of each definition, plus some number of clones. On keyword output an identical definition will be output in each include file in which it originally appeared.

A further, and once again undocumented, feature of LS-DYNA is that "once only" keywords souch as **`*CONTROL`** cards will also be accepted if defined multiple times. As with the keywords above versions of PRIMER before V12 would "lose" the 2nd and subsequent definitions; from V12 onwards these too tested for differences, and then "cloned", and identical definitions will be written to every include file in which they were originally found.

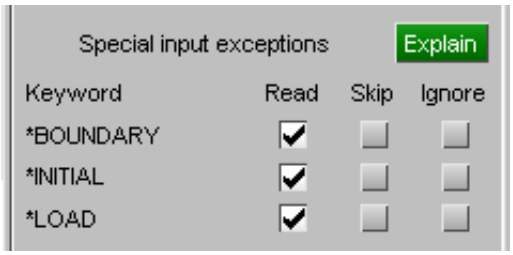
Options: Treatment of severe errors



PRIMER is quite strict about errors when reading input decks, since invalid data fields can result in a corrupt database. This can be a problem when it rejects what it believes is a corrupt deck when you, the user, know that it will in fact be OK to continue.

If you choose **Skip & Continue** these errors are downgraded to warnings, the offending data cards are skipped, and input continues. ***You do this at your own risk, and you must deal with the consequences of any resulting database inconsistencies.***

Options: Special input exceptions



Sometimes an input deck will contain a very large number of **`*BOUNDARY`**, **`*INITIAL`** or **`*LOAD`** cards, and this can be a nuisance if you only want to look at geometry as it makes the model slow to read and will also consume a lot of memory. You can choose to:

- **Read** these cards. This is the default behaviour, and they will be read normally.
- **Skip** these cards. They will not be read, but will be placed in a "skip" file. If the model is written out they will be copied verbatim from the "skip" file into the output deck.
- **Ignore** these cards. They will not be read or saved in any way. If the model is written out they will be lost.

Clearly the **Ignore** option is only suitable for read-only inspection of the deck, and the **Skip** option may give problems if nodes or elements are deleted or renumbered.

Options: Save embedded comments

Comment lines are often placed among keywords, and PRIMER can remember these and write them back out in the same locations. PRIMER regards comments as being "embedded" if they are between a *Keyword header and data

```
$ This comment is "free standing" since it is not between a *keyword and data
lines
$
*NODE
$ This comment is "embedded": it is between the keyword header (*NODE) and
following data
         1         1.0e3            0.0              0.0
$ This comment is "embedded" because it is between data rows
2         2.0e3            0.0              0.0
$
$This comment is "free standing" because it is not between a *keyword and data
lines.
$
*ELEMENT_SHELL
```

In the example above the comments in italics are "free standing" and will not be saved, whereas those in normal text are "embedded" and will be saved.

A special exception is comment lines starting "**$:**",. PRIMER does not save these lines, regardless of their location. Such lines are used when PRIMER writes things like data field headers, and it stops these lines being saved and rewritten multiple times. For example:

```
*PART
$ User-defined comment
$:     PID     SECID       MID     EOSID      HGID      GRAV    ADPOPT
TMID
         1        10         2         0         0         0         0
0
```

The "user defined comment" line starting with plain **$** is embedded, and will be saved; but the field header line starting **$:** is ***not*** saved. It will be recreated on output if field header display is turned on.

More details about the processing of embedded comments may be found in section 5.1.10 "Embedded" keyword comments.

Options: Find data during scan

When using **Scan** or **Quick scan** to read a subset of include files from your model problems can arise if ***PARAMETER** or ***DEFINE_TRNSFORMATION** cards are ***referred to*** by the include files read, but are ***defined*** in other include files that have not been read.

If this option is selected then PRIMER will search other include files for the missing definitions and read them in so that the files that have been read are correct.

Options: Zero field spillover.

PRIMER is strict about formatting and grammar errors when reading data, and this can result in an input deck being rejected due to minor editing errors. A particular example of this is when a data field has been hand-edited and given too many trailing zeros so that it spills over into the next data field. Consider the following example:

```
$    FIELD1     FIELD2     FIELD3     FIELD4     FIELD5
     200000   100.0000          1       30.00         1
```

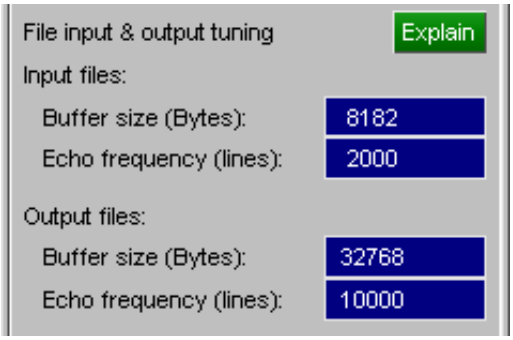
In this example it is clear that the trailing zero on 30.00 (in red) is an accidental spill-over into the next data field, and can safely ignored. By default PRIMER will reject this line as being wrongly formatted, but permitting "Zero field spillover" will accept it instead.

WARNING: permitting this can cause errors. The example above showed harmless trailing zeros on a floating point number, but consider the following:

```
$    FIELD1     FIELD2     FIELD3     FIELD4     FIELD5
     200000   100.0000          1       30000         1
```

Field #4 is now an integer, and by ignoring the trailing zero its value has been changed from 30000 to 3000, which would (probably) be an error. Use this facility with caution, and only after you have inspected the offending line to make sure that it is safe.

Options: File input & output tuning.



We have received reports of slow keyword file read and write behaviour on some platforms when the files in questions are on a remote networked disk.

PRIMER uses standard ANSI C buffered i/o routines when reading and writing files and in most cases the default system settings are satisfactory. However users with "problem" remote files have reported improvements when changing the default settings, so the following may be altered:

Input and output Buffer size

These are the sizes in bytes of the memory buffers used by the system for reading and writing files.

ANSI C documentation states that they must be a multiple of 2 bytes, although it does not stipulate a size; most systems default to 4096 bytes. Experimentation by Oasys Ltd suggests that:

- Values less than 512 bytes or greater than 131072 bytes are likely to reduce speed.
- Values in the range 4096 to 32768 bytes are likely to give the best results.
- Values less than 4096 bytes should be a multiple of 512.
- Values greater than 4096 bytes should be a multiple of 4096.

However you may find that different values work better on your system: you can experiment by setting these values manually and measuring the time taken to read and write files. If you find values that work better than the defaults then you can set them in the "oa_pref" file using:

```
primer*input_buffer_size: nnnnnn
primer*output_buffer_size: nnnnnn
```

Where ***nnnnnn*** is the size in bytes.

For most users disk read/write speed is not a problem, and it is suggested that the default values are used unless they are demonstrably inefficient.

Echo frequency (lines)

During file input and output PRIMER reports its progress via an "echo" of the current line to the dialogue box, by default every 1000 lines. It also checks the user interface at this frequency to see if the user has made any inputs, for example using the **Stop** button to halt i/o.

Users working on remote displays, where updates of the user interface have to travel over the network, may find that there is a speed advantage in reducing this echo frequency. You will need to experiment, but values of 10,000 or greater may help in these circumstances. To store revised settings in the "oa_pref" file use:

```
primer*input_echo_frequency: nnnnnn
primer*output_echo_frequency: nnnnnn
```

Where **nnnnnn** is the number of lines.

Users working on a local display will almost certainly find no benefit is gained from increasing these values.

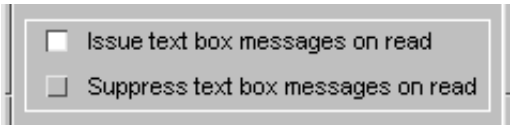
Options: Comment reading options.

Read HM comments - Turn ON/OFF reading of HM comments in the keyword file.

Read ANSA comments - Turn ON/OFF reading of ANSA comments in the keyword file.

Copy HM titles - When ON, Primer will set a material or section title (if one is available from an HM comment), and if the item does not already have a title. If OFF, The title will not be set to any available HM comment title, but the comment will be retained for keyout.

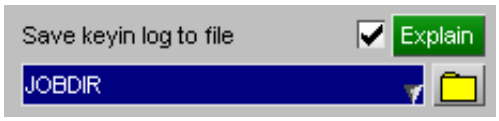
Options: Text box messages during Keyword read



Some errors and warnings during Keyword input generate a "text box" panel that halts processing until the user clicks OK (or something else) to acknowledge the message.

If the "suppress" option is used then the messages will still be output, but only in the dialogue box, so the user does not have to acknowledge them and processing is not halted.

Options: Save Keyin log to file



During keyword input all messages written to the dialogue box are also saved to file, and can be viewed in an external editor using the **View log** button. However this log file is overwritten each time a new keyword file is read, and it is also lost at the end of the PRIMER session unless saved manually from an external edit session.

It can be useful to save this information, especially when running PRIMER in batch mode (no graphical user interface), and this option permits all messages to the dialogue box to be saved in file **primer_readlog.txt** in the directory of your choice.

The directory may be specified in two ways:

- By giving an explicit pathname, for example **C:\users\myself\readlogs**
- By using the special name **JOBDIR**, which means "use the directory from which the master keyword file was read". It is legal to append relative paths to this, for example JOBDIR/../logs will select the directory above, then down into sub-directory logs. This can be useful if your system has read-only directories.

If the file **primer_readlog.txt** already exists its name will be amended to **primer_readlog_*n*.txt** where **_*n*** is the smallest integer required to make the filename unique. In this way existing files will not be overwritten.

Writing of this log file can also be turned on in two other ways:

1. By using the command-line option '**-rlog_dir=**pathname'
2. By setting the preference '**primer*save_read_log_dir:** pathname'

It doesn't matter which method is used, all create the same file. The order in which these are read are (1) preference, (2) command-line, (3) this options panel. The most recently read definition will control the pathname that is used.

If this facility is turned on then two extra blocks of information are written:

(1) Merged *NODE data	LS-DYNA permits multiple definitions of a ***NODE** with the same label to exist in an input deck so long as the bopundary conditions (**TC** and **RC**) of the two definitions are the same, and their coordinates are coincident to within a small tolerance. This tolerance is defined by: **(xdist2 / xdist1) < 1.0e-8** where • **xdist2** is the vector distance between the coordinate of the two nodes • **xdist1** is **max(1.0e-16, vector distance of node from origin)** In addition if a ***NODE_MERGE_TOLERANCE** card is defined then the nodes must be further apart than this tolerance to be considered "not coincident". PRIMER uses the same logic as LS-DYNA, and automatically merges coincident nodes that obey these rules. It "remembers" the duplicate definitions as "clones" - see **Permit Duplicate Definitions** above for more details about this process. Normally this merging process is carried out silently, but if this log file is being written then an extra report of all merged nodes is written to the file. An example of this output is: **Node 1 merge SUCCEEDED** **First definition at 1.00110E+03 1.00000E+02 4.23501E+03, TC = 0, RC = 0, in incl_1.key** **Duplicate definition at 1.00110E+03 1.00000E+02 4.23501E+03, TC = 0, RC = 0, in Master file** **Distance to origin (xdist1) = 1.0172e+03, separation distance (xdist2) = 0.0, (xdist2/xdist1) = 0.0** It will be seen that the coordinates and also the include file containing each definition is reported. Merged nodes can be visualised, see **[NODE] Duplicates** for more information.

(2) Duplicate *PARAMETER data	LS-DYNA permits the **`*PARAMETER`** keyword to use suffices **`_LOCAL`** and **`_MUTABLE`**, which make it legal for a parameter name to be used more than once, possibly with different values. In addition the **`*PARAMETER_DUPLICATION`** card controls how potentially illegal duplicate parameter definitions will be processed, and controls which of multiple definitions will be the true value.
	
	These options may be flexible, but they can also lead to complexity if input decks with many include files contain multiple different definitions of a parameter, making it hard to tell which value is used in which context.
	
	PRIMER tolerates this, obeying the same rules as LS-DYNA to determine how multiple definitions should be processed. It also tolerates and "remembers" illegal duplicate definitions, both so that they can be written out again and also to cope with changes to the various keywords that could alter their legality. This processing is normally carried out silently, and it is necessary to go to the [keyword] **PARAMETER, MODIFY** panel to view the status, value and legality of all parameters.
	
	However if keyin logging is in force then the status of all duplicate parameter names is written to this file, giving a permanent record of their status when read. An example of this output is:
	
	`Parameter name: X701`
	
	`Instance 1: *PARAMETER (Real, value = 701.1) in master file`
	`OK-USED: global definition`
	
	`Instance 2: *PARAMETER_LOCAL (Real, value = 801.1) in incl_2.key`
	`OK-USED: locally because of _LOCAL suffix`
	
	`Instance 3: *PARAMETER (Real, value = 901.1) in incl_3.key`
	`ERROR-IGNORED: illegal definition is saved but ignored in PRIMER`
	
	In this example parameter **`X701`** is defined three times:
	1. In the master deck as plain **`*PARAMETER`**, with the value 701.1
	2. In include file incl_2.key as **`*PARAMETER_LOCAL`**, with the value 801.1
	3. In include file incl_3.key as plain **`*PARAMETER`**, with the value 901.1
	
	Usages #1 and #2 above are legal, since #2 (being **`_LOCAL`**) will only apply in include file incl_2.key. So both are marked as **`OK-USED`**
	
	Usage #3 is illegal since a plain **`*PARAMETER`** with no suffix is "global", and this conflicts with the original definition in the master file. So this is marked as **`ERROR-IGNORED`**.
	
	You can find out more about how PRIMER handles parameters under the **PARAMETER** keyword documentation.

Processing of LS-DYNA include files.

By default any **`*INCLUDE`** files referenced in an input deck will be read, but you can choose to read the master file only, ignoring include files, by deselecting this option.

The treatment of **`*INCLUDE`** files on output is handled separately, see below in section 3.3.

Reading of files using compression (.gz and .zip) and binary (.kby) formats

PRIMER can both write and read the following variants on normal LS-DYNA keyword syntax
- "Raw" ASCII files, typically (but not necessarily) "name.key" or "name.k"
- Binary format files using PRIMER's special format, "name.kby"
- Compressed files using "gzip", "name.key.gz"
- Compressed files using "zip", "name.zip" or "name.key.zip"
- Permutations of these, for example a gzip compressed binary file would be "name.kby.gz"

More information about writing these formats can be found in Section 3.3: see Format and Include files and compression

This introduces some potential problems with file names during keyin, especially include files. Consider the following example:

The master keyword file contains the lines

```
*INCLUDE
child.key
```

But depending on the output settings this might actually be written to disk as any of

Gzip compression active	child.key.gz
Zip compression active	**child.zip**
Binary output active	**child.kby**
Binary and gzip compression combined	**child.kby.gz**
... and it is possible to contemplate further permutations.	

A further problem is that the user might edit that file inside PRIMER, then write it out in a different format to the same directory. So now we might have both:

Original file child.key
and
Edited file child.kby

To solve this problem PRIMER uses a sort of "fuzzy logic" when opening include files, testing many different variants of the raw name. If a single unique match is found then this is opened directly, but if multiple possible candidates are found a warning message is written and the more recently modified file is used.

For master files PRIMER presumes that the given filename is "correct", and assuming it is found on disk it is opened without further searching. However if the given filename is not found then it applies the same search logic as above.

This behaviour can be modified by the preference

primer*keyin_filename_syntax: newest | verbatim

The default is **newest**, but if **verbatim** is chosen then the exact (raw) filename is used without further searching.

View log: Viewing the input log

During keyword input any warnings and errors are sent to the Dialogue box, but if there are many of them they can fly by too quickly to be read, and get lost if they exceed the number of lines in the scroll buffer.

From Version 9.3 PRIMER summarises these on the **Model > Read** panel, and stores them in a temporary log file. **View log** will view this file in the default editor on your system

3.2.2 List of file formats read by PRIMER

Format	Description	What is read
LS-DYNA	LS-DYNA "Keyword" format	Complete input deck is supported. (No conversion required.)
	"Long" (large) LS-DYNA format	PRIMER V12 will read both traditional ("small") LS-DYNA keyword format, and also "long" (wide) format in which all data fields are expanded to 20 columns width. Long format permits item labels to be up to 18 characters long, and PRIMER V12 support for this is described in section 5.1.7 Long keyword format and large labels. Detection of large format is automatic from the file syntax and no user intervention is normally required, however long format can be forced on the **Options** panel. This may be required if the input file syntax is deficient and the "long=y" instruction has been added to the LS-DYNA execution line. See "General Card format" under the "Getting started" section at the beginning of Volume I of the LS-DYNA keyword manual for more information about long format files. Versions of PRIMER before V12 will only read traditional format, and will only process labels up to the 32 bit integer limit of ~2e9.
	Compressed files	PRIMER 15 will read files compressed to .gz and .zip format. Compression may be by PRIMER itself (see the section on Output Compression) or carried out independently by the gzip and winzip utilities.
	Binary files	PRIMER 15 will read binary format files that it has previously written (see the section on Binary output), both in raw form (.kby) and compressed to .gz or .zip formats.
NASTRAN	NASTRAN "Bulk Data" (.bdf) format	Nodes, elements, loads, properties, materials, SPCs, + others: see Appendix VI (1)
IDEAS	Master Series and IDEAS "Universal" (.unv) file format	See Appendix VI for a list of supported modules.
PATRAN	MSC Patran level 2.5 "Neutral" (.neu) file format.	Nodes and elements only are read.
SAP2000	SAP 2000 input deck	Most items are read, and some interpretation takes place - see Appendix VI (2)
RADIOSS	Mecalog RADIOSS "Starter" and "engine" files, fixed format v4.1	Most items are read, and some interpretation takes place - see Appendix VI (3)
ABAQUS	ABAQUS input deck	Nodes and some element types. Also, basic section, set and simple material data. See Appendix VI for details.
IGES	Geometry data in IGES 5.3 format	Most points, curves and surfaces. See Appendix VI for details.
JT	Siemens JT file	Only tessellation in the JT file is read for visualization. No geometry (NURBS surfaces etc) is read from the file, even if it is present.
DesignLink	xml format	Elements, Nodes, Matls, Sect, Set, nodal Load & SPC from DesignLink xml format

To read a particular format:

- Select that format in the "Input file formats" area;
- Select any sub-type (here **KEYWORD** has been chosen);
- Select a "target" model id. Here the default of the next free model (#9) has been chosen, but you are free to use

any model in the range 1 to 255.

It is ***strongly*** recommended that you don't read external data into an existing model since, if any clashes (in labels) are found the read operation will be aborted with both incoming data ***and the existing model*** lost. To be safe **COPY** the existing model first! However you are free to do this if you are certain that there are no clashes between existing and incoming labels, and this will effectively merge the two models. An option can be set inthe read options panel so that any model read into an existing model is placed into the current include layer. By default, the information read in will be plaved into the master layer.

- Choose an input filename. If you know it type it into the "File" text box, otherwise use the button to obtain the file selector box.

Once these steps are complete the **APPLY** button will be enabled, and you can press it to read the file. Assuming that it reads in successfully it will be added to the list of current models, and drawn in the graphics window in the current drawing mode.

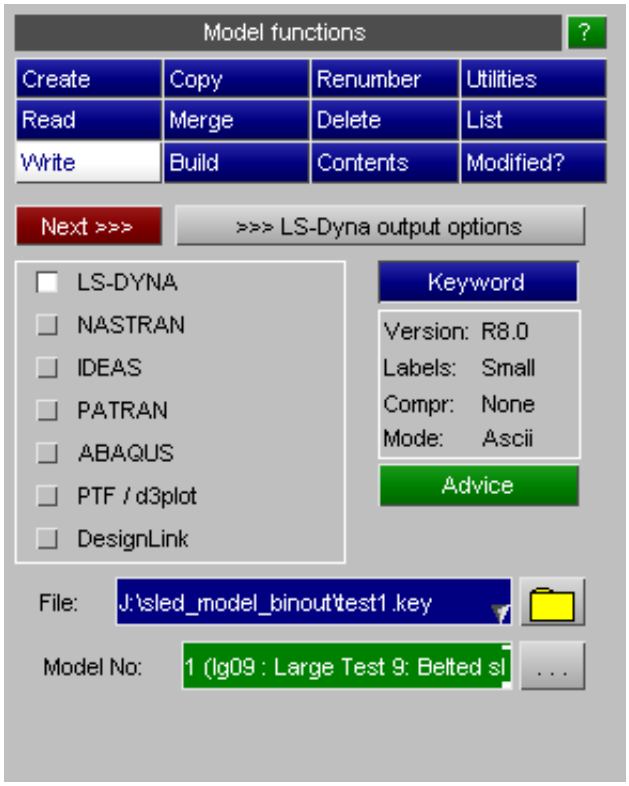
You will be warned if any errors are found, the action taken depending upon context:

- Where the error is not fatal that keyword or section will be skipped, and input will continue.
- If the error cannot be recovered then input will terminate, and any data read so far will be destroyed. Destruction is necessary to prevent any internal inconsistencies arising from the errors corrupting the database.

Data formats which require interpretation may request further information about translation defaults. These are:

- **SAP2000** Described in Appendix VI (2)
- **RADIOSS** Described in Appendix VI (3)

3.3 MODEL > WRITE



Writing out models to disk.

- Select a file format; (details in table 3.2 below)
- Select an output filename;
- Select the model to write out.
- Click on **Next >>>**

If the file already exists you will be given the choice of overwriting it or giving a new filename.

For LS-DYNA output only the **>>> LS-DYNA output** options button maps the pre-output panel shown below, allowing you to change output options without actually having to pretend to write something. (The current LS-DYNA keyword format selected can affect model checking.)

The following table summarises the formats written and what conversions and limitations apply: see Appendix VII for more details.

Format	Description	What is written
LS-DYNA	LS-DYNA "Keyword" format using LS-DYNA 9xxx syntax	Everything in the file is supported, no translation required. See **Version** below for specific formats supported.
NASTRAN	Nastran "Bulk Data" (.bdf) format.	Conversion to Nastran bulk data on output matches approximately that applied during input. See Appendix VII (3).
IDEAS	Master Series and IDEAS "universal" (.unv) file formats	Large number of items in an Oasys Ltd N/CODE compatible form. See Appendix VII (1)
PATRAN	MSC Patran level 2.5 "Neutral" (.neu) file format	Nodes, elements, materials and properties only. See Appendix VII (2)
ABAQUS	ABAQUS "Input" (.inp) file format	Conversion to Abaqus data on output matches approximately that applied during input. See Appendix VII (4).
PTF / d3plot	PTF / d3plot file format	Conversion of Elements, Nodes and Geometry surfaces to PTF / d3plot format. See Appendix VII (5)
DesignLink	xml format	Conversion of Elements, Nodes,Matls,Sect,Set, nodal Load & SPC to DesignLink xml format for generic handling

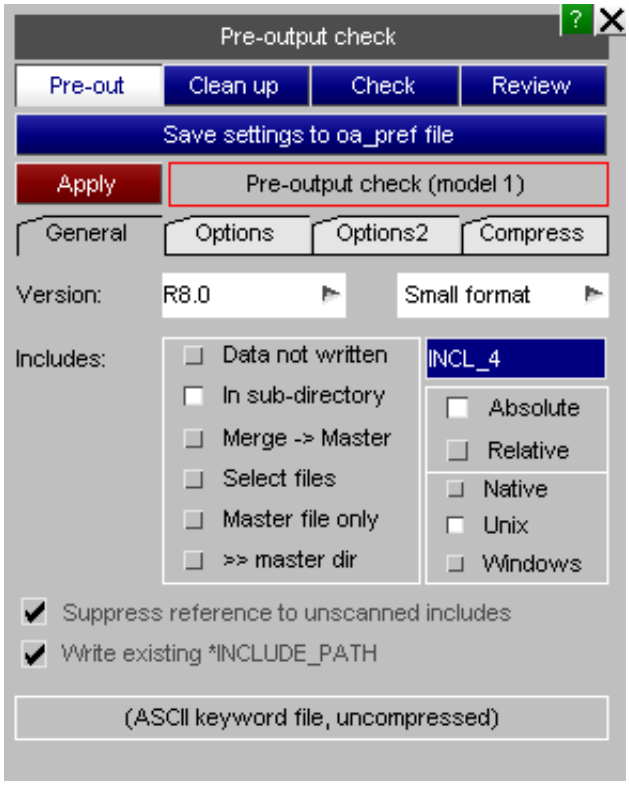
Table 3.2: External data formats written by PRIMER

LS-DYNA output: Pre-output checks and output options

Before writing out the model in LS-DYNA keyword format the user is given a number of review functions.

Pre-out	Shows the standard output options, described in detail below under • General Tab (Most commonly used settings) • Options Tab (Further options) • Options2 Tab (Yet more options) • Compress tab (Compression and binary output options)
Clean up	Runs the standard "Model clean up" operation to get rid of redundant items (see section 6.36.2)
Check	Runs the standard "Model Check" operation which checks all keywords. (see section 3.9)
Review	Gives a summary of typical model control options, principally `*CONTROL` and `*DATABASE` options.

The **Save settings to oa_pref file** button saves the various **Pre-out** options to the preferences.

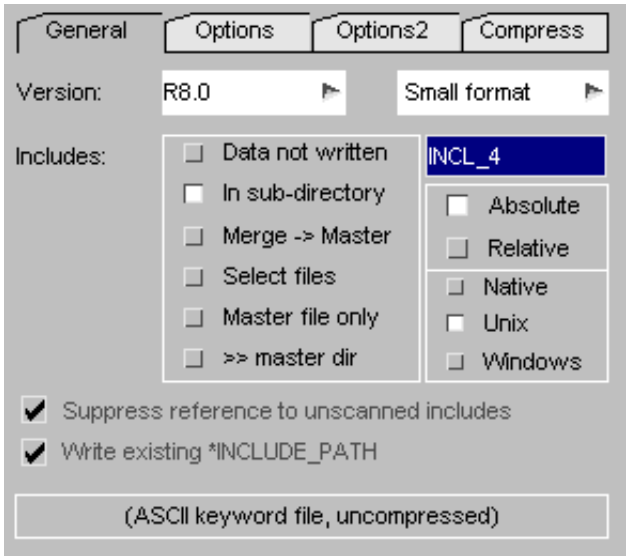


Pre-out: General tab.

The following table summaries the options available. Click on the link in the first column for a more detailed description of each one.

Version	Defines the LS-DYNA version used for output.
(Format)	Whether output is "small" (traditional), "long" for large labels, binary, etc.
Includes	How include files are written.
(Machine)	Native / Unix / Windows determines the syntax used for include file pathnames, also how line endings in text files are written.

The summary at the bottom, here [`(ASCII keyword file, uncompressed)`] shows the current output file type that will be written. The Compress options may change this (eg compressed, binary, etc).



Version: tailoring output to a particular LS-DYNA version.

Version allows you to modify or suppress keywords to be compatible with different versions of LS-DYNA. PRIMER release 14.0 supports the following LS-DYNA formats:

- **`R11.0 (development)`**
- **`R10.0`**
- **`R9.0`** (This is the default output in PRIMER release 14.0)
- **`LS971R5`**
- **`R8.0`**

- **`R7.0`** (previously referred to as LS980 by LSTC)
  PRIMER releases before V12 support only the "mechanical" keywords in volumes I and II of the R7.0 user manual, support for the volume III multi-physics keywords has been added from V12 onwards.
- **`LS971R6.1`**
- **`LS971R6`**
- **`LS971R4`**
- **`LS971R3`**
- **`LS971R2(7600)`**

- **`LS970 v3858`**, **`v5434`** and **`v6763`** formats are fully supported.
- **`LS960+`** contains a limited subset of the **`LS970`** keyword format.
- **`LS940`**, **`950`** and **`960`** "legacy" forma

Setting a different default output version

Each release of PRIMER is assigned a default output format that is normally set to "one before the current" LS-DYNA release. You can change the default output version via the preference:

`primer*dyna_output_version:` *version number* (eg **`971R6.1`**)

Conversion between the formats of different LS-DYNA versions

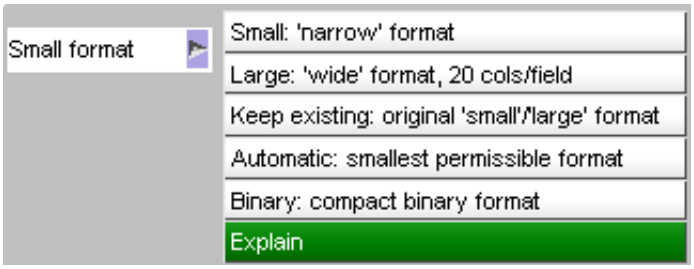
Writing out "higher order" decks in "lower order" format, for example a **`LS960`** deck in **`LS950`** format is legal, and has the following consequences:

- Where higher order data can be converted to lower order without loss of information this is done silently.
- Where no lower order version exists the data (fields or whole cards) are omitted, and a warning notice is printed.

As a general rule writing out a higher order deck from a lower order file (eg read LS960, write LS971) works without losing information, but there are a few cases where keywords have changed during LS-DYNA development, meaning that the result may not be functionally identical.

> *While we have made every effort during **VERSION** conversion to detect and process changes between the different LS-DYNA file formats, we cannot guarantee that we have found every one. Moreover running the same analysis in different versions of LS-DYNA may give different answers due to changed parameters within theLS-DYNA code.*
> ***It is your responsibility to ensure that your analysis is correct.***

Format: Setting the file format



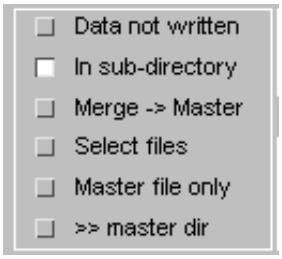
By default LS-DYNA output is written in "small" format ASCII files where lines are limited to 80 characters and typical field widths are 8 or 10 columns (although selected fields are wider). However PRIMER supports the following alternative options:

Small format	The default LS-DYNA output format
Large format	Specified by adding "**`long=y`**" to the **`*KEYWORD`** card • Data field <20 wide become 20 wide • Lines may be up to 240 characters long
Keep existing	The format of output files is the same as that when they were read in.
Automatic	Files default to small format if possible, but use large format if labels are too wide to fit.
Binary	Proprietary binary format, new in PRIMER 15. About 1/3 of the size of the equivalent ascii file, and much faster to write. In PRIMER 16 these files are given the extension ".kby". This format is described in detail under the Compression tab section below.

If a model contains "large" labels, which is defined as labels > 99,999,999 then normally it is necessary to use Large format to write it out, and PRIMER will default to this. However if small format is explicitly chosen in this situation then it will be used until a card containing "long" labels is encountered, in which case a new keyword header using the "+" suffix will be written and the rest of that particular keyword will be written in long format.

Details of the way that PRIMER handles large labels may be found in section 5.1.7.

Includes: How includes are written



An input model may have any number of **`*INCLUDE`** files, which may be (although they usually are not) scattered widely around a disk system. PRIMER reads these and "remembers" what was in each file.

On file output you can choose to process data read from **`*INCLUDE`** files as follows:

Data not written	Include files are not written at all.
In sub-directory	Include files are written to the specified sub-directory. This is the default.
Merge->master	All include files are merged into the master file, meaning that the include structure and membership information is not written (it won't be lost from model in memory)
Select files	User selects files he wants to write, choosing overwrite/new-file/into sub-directory for each **`*INCLUDE`** file. The details of this procedure are shown below under **Select Files** output method.
Master file only	Only master file written. It will contain references to include files, but the includes files themselves are not output.
>>master dir	All include files are written to the same directory as the master file.

Where includes are written

*Include files may be read from anywhere on disk, including "read-only" file systems and directories. Therefore it may not be practical or sensible to try to restore these files to the directories from which they came, and the following approach is adopted:

- A new sub-directory called **INCL** is created.
- If this directory already exists suffices "**_1**", "**_2**", etc are added so that the directory name is always unique.
- All *include files have their incoming paths stripped, leaving only the filename, and are written to this sub-directory.
- The references to them in the master file thus becomes **INCL**/<include_filename>.
- If necessary suffices "**_1**" etc will also be added to filenames in this directory to make them unique.

You may replace the default name **`INCL`** with a name of your own choice.

Include filename syntax

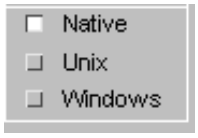
You need to choose whether include files use **Absolute** or **Relative** pathname syntax. THe differences are

Syntax method	Typical result on Windows	Typical result on Linux
Absolute, prefix uses explicit pathname	C:\users\my_model\INCL\a.key	/home/users/my_model/INCL/a.key
Relative, prefix is relative to master file	.\INCL\a.key	./INCL/a.key

How master file references include files:

LS-DYNA will accept include files referenced with an absolute path or a path relative the master file, which may be preferred as Dyna is currently limited to a string length of 80 characters. However, for include files within include files it is recommended to always use the **absolute** path.

Machine: operating system dependent syntax



The filename syntax used can be one of **Native**, **Unix** or **Windows**:

Filename syntax	Typical result
Unix (Linux)	/home/users/my_model/INCL/a.key
Windows	C:\users\my_model\INCL\a.key
Native	Detects the type of this machine and uses the relevant option above

Users running in a cross-platform environment, typically Windows on the desktop and Linux on a remote cluster, may also wish to use PRIMER's drive mapping options. These allow specific Windows drive letters A to Z to be mapped onto Unix-style pathnames, permitting "mixed" syntax to be used on both input and output.

See the oa_pref options

`primer*drive_a` etc
`primer*output_os`

in Appendix XIII

For example in the oa_pref file you might map the windows "**`S`**" drive to correspond to the unix directory "**`/data`**" thus: **primer*drive_s: /data**

Thus an include file "**`/data/includes_1/inc.key`**" (read in unix) will be referenced under the **`*INCLUDE`** as "**`S:\includes_1\inc.key`**" if the model in written out in **`WINDOWS`** format. Likewise files read in Windows versions may be converted to Unix format, the conversion is bi-directional.
The default setting **`NATIVE`** writes in the format appropriate to the machine.

Windows vs Unix/Linux line endings

There is a subtle difference between Windows and Unix/Linux text files that - extraordinarily - goes back to the days of teletype machines. Teletypes, being mechanical, treated [carriage return] and [line feed] characters as separate operations, since a return without a new line allowed lines to be over-typed.

Therefore the standard ascii text (sometimes referred to as MS-DOS) output on Windows operating systems terminates each line with the pair of characters [carriage return] [line feed], and PRIMER adheres to this convention when files are opened in Windows mode.

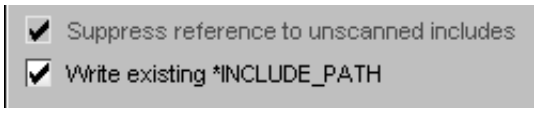
However on Unix/Linux systems text files end a line with just [line feed], implicitly assuming that each new line starts at the left hand margin and, again, PRIMER adheres to this convention.

For the most part this doesn't matter as PRIMER, and most other applications, will read files with either line ending syntax quite happily; moreover if you are working on a single operating system this is unlikely to trouble you. However if you are working in a cross-platform environment you may occasionally find that this affects you, for example:

- Text files written on Windows may be slightly larger than the supposedly identical file written on Linux because of the extra [carriage return] character at the end of each line.
- Some older Windows applications, eg Notepad, will not add the implicit [carriage return] when reading Unix/Linux format files. (However most others, eg Wordpad, read these files quite happily.)
- Some older Unix/Linux applications, eg vi (as opposed to vim), may show the superfluous (to them) [carriage return] character as a ^M symbol at the end of lines.

It is unlikely that you will encounter these problems, but if you do hopefully the explanation above will help. If you need more help please contact Oasys Ltd for advice.

Output of *INCLUDE_PATH cards



If the model has one or more **`*INCLUDE_PATH`** cards tick the **Write existing *INCLUDE_PATH button** if you wish PRIMER to write out the include files with the **`*INCLUDE_PATH`** keywords (these are stored on read-in of the model).

This option is only available if **Relative** path syntax is chosen.

("Suppress reference to unscanned includes" is a specialist option that is used only if the model was originally "scanned" rather than "read", and limits output only to the include files actually processes)

Select Files output method

If no includes are modified, it is recommended that you select **Master file only** under the **`*INCLUDE`** heading. If any renumbering has occurred or you made any changes specific to the content of any individual Include file it is recommended that you select **Select files** under the *INCLUDE heading. This will allow you to save both the Master file and the modified Include files without wasting time writing includes that have not changed.

When you have completed all pre-output checks, press **APPLY** to map the write selected panel.

FIND MODIFIED will run the model modified function and red-light any modified includes. These will also be selected for write along with the master file. If the modified include is child of another, the parent, grandparent,etc include will also be selected.

You may manually Select/deselect files by clicking on the grey box to their left. A tick will appear in the selection box and the options **SUB-DIR** and **RENAME** will be offered.

If you select **SUB-DIR** the modified file will be written to a newly created sub-directory (INCL, INCL_1, etc.) in the directory where the Master model is saved.

If you select **RENAME** you have the opportunity to specify the name and/or path of the selected file. The text box will always display the full path of the file, but if you may have selected the *Relative option* from the previous panel this will be used. If the background of the text box is red, it indicates that a file overwrite will be incurred. If the background is bright orange it indicates that you do not have permission to write the file (either the directory or the filename is protected) and the "APPLY" button will be greyed until you deselect the include.

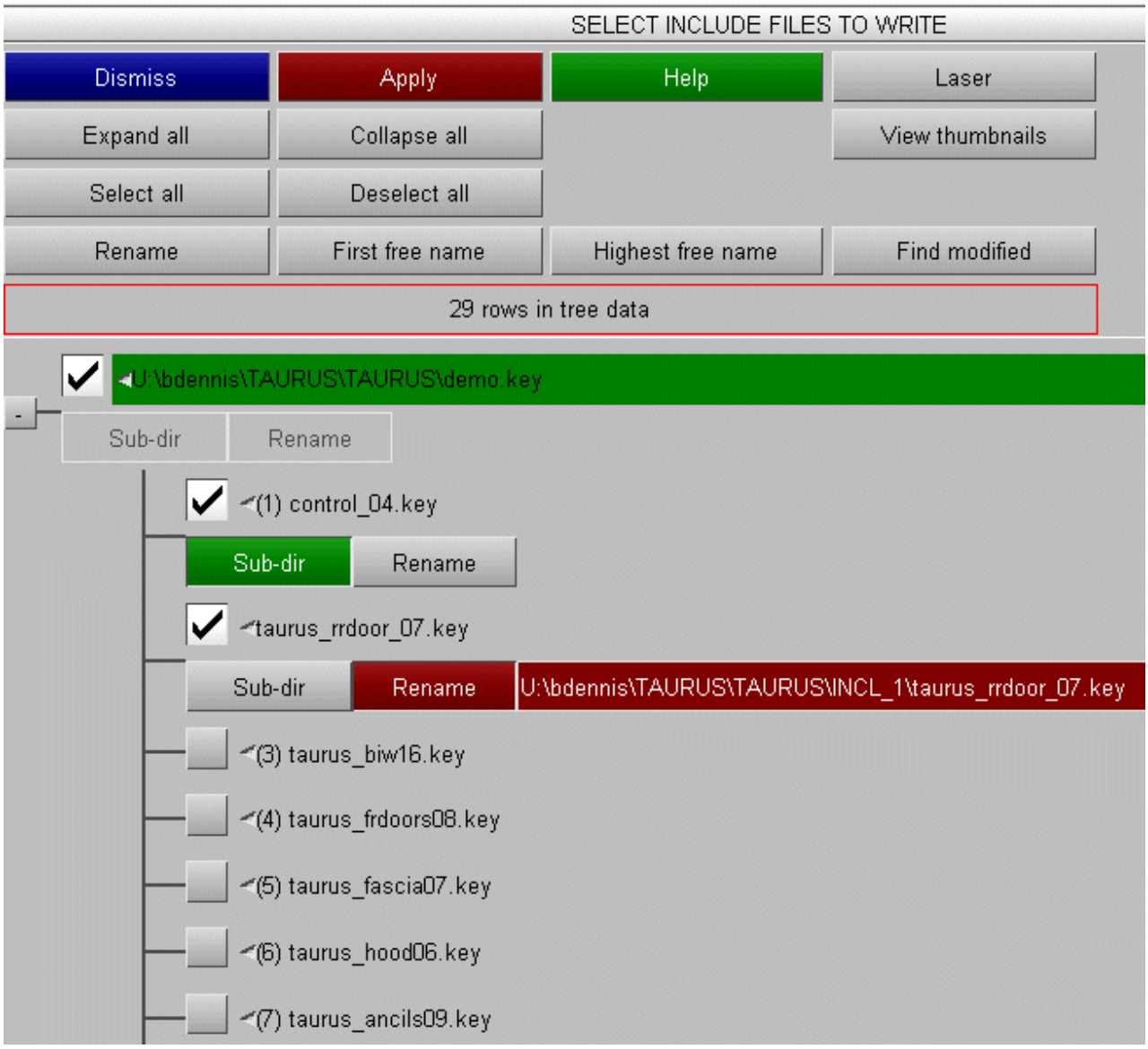
Some tools exist to help renaming of multiple files.

First free name will increment the name of each file that is selected for overwrite until a free name is found, e.g. file_aaa will become file_aaa_001, file_bbb_002 will become file_bbb_003. This may leave filenames at different indices.

Highest free name will determine the highest index of selected files, increment until an index is found at which all files have free names and set them, e.g all files will assume index _003. This may be preferred if you want to keep files at same index.

Note the original file names can be restored by simply pressing **RENAME**

Once you have selected all files to write, press **APPLY** to start the write process.



Include files and Compression.

Several issues arise when combining compressed output with include files.

Using **Select Files** with .zip package compression

If compressed output (eg .gz or .zip) has been selected, then individually written include files will be written in compressed form, however a problem arises if:

- Compressed output with all files written to a single .zip archive is selected
  and
- The master file is not selected for output in the include tree (ie "**demo.key**" in the example above is not selected).

In this situation what should the .zip file package be called? To solve this problem PRIMER does the following:

- The master archive file *name*.zip (in the example above "**demo.zip**") is still created.
- The master keyword file name.key ("**demo.key**" in the example above) is written to that file, however it will only contain *INCLUDE keywords.
- The selected include files are added to the .zip archive as normal.

Does compression have to apply to all include files?

No. When ***reading*** a file PRIMER uses the "fuzzy logic" described in Reading of files using compression (.gz and .zip) and binary (.kby) formats above which permutes the various possible filename extensions to find one that is present. This might be the raw filename, or it might be a .gz, .zip, .kby or other variant.

When ***writing*** a file you have the following options (see Compress files below):

1. No compression, so all files are uncompressed (raw)
2. Keep original, so each file is written in the format in which it was read.
3. Compress all files to individual .gz format
4. Compress all files to individual .zip format
5. Compress all files into a single .zip package

With the exception of option #2, Keep original, the original input format is not considered. Therefore any file can be read in any format, and written in any format.

How should compressed files be named in *INCLUDE keywords?

They should always use their uncompressed (raw) syntax in the keyword file. For example all of the include files **include.key**, **include.key.gz** and **include.zip** should be referred to by the keyword

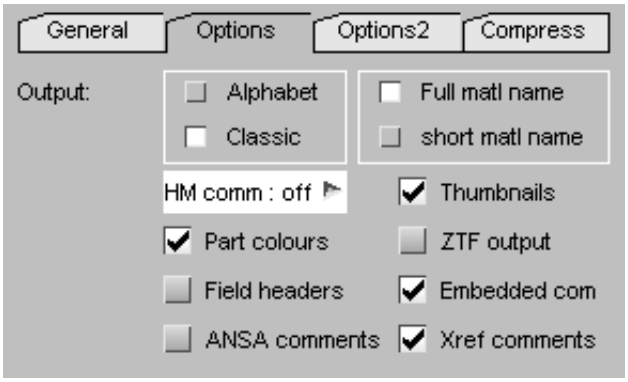
```
*INCLUDE
include.key
```

PRIMER will automatically try the various alternatives (.key, .key.gz, .zip) when searching on disk for the file, see Reading of files using compression (.gz and .zip) and binary (.kby) formats above.

In the case of the master file being a packaged .zip file (ie reading **master.zip**) it will first look for the file using its raw name inside /INCL sub-directory in the package, and if it is not found it will look on disk as above.

Pre-out: Options tab

Options controlling the layout and other attributes of the output file.



The **OUTPUT** option gives a choice between writing keywords in **alphabetical** order or a more intuitive **classic** order e.g. sets are written together with their referencing objects.

The **Short matl name** option (available from LS 970 onwards) will write all material cards in the form **`*MAT_NNN`**(_option) rather than the full name.

The **HM comments** option writes out part, section and material titles as comments for use with the Hypermesh pre-processor.

The **Thumbnails** option will write out any existing include file thumbnail images at the end of each keyword file.

The **Part Colour** option will add a PRIMER-readable comment line to all *PART cards that contains their colour, display mode and transparency settings, meaning that these will be restored when the file is reread into PRIMER

The **ZTF output** option will write a **`<name>.ztf`** file, which is an extra binary data file readable by D3PLOT, making it possible to visualise extra information when post-processing.

The **Field Headers** option adds a comment line above each row of keyword output containing the data field acronyms (eg PID, SECID, EOSID, etc)

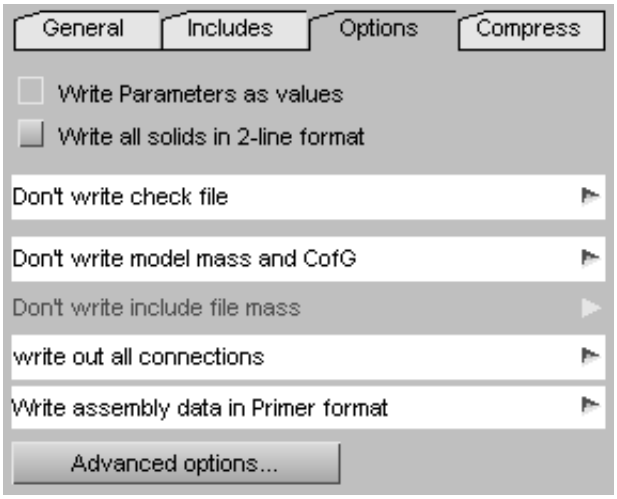
The **Write comments** option determines whether or not embedded keyword comments read from the input deck or added in this session are written out in the output deck.

The **ANSA comments** option allows users to decide whether or not to write ANSA comments in the current model. It is not possible to select the option "Write assembly data in ANSA format" if the check button "ANSA comments" is switched on. In that case, PRIMER will use the default option "Write assembly data in PRIMER format".

The **Xref comments** option controls whether cross-references to items are written as comment lines in the output file.

Pre-out: Options2 tab.

This panel contains further miscellaneous options which influence output.



Write parameters as values applies only to input decks that contain **`*PARAMETER`** cards. If selected then instead of writing out the parameter names (**`&name`**) the actual numeric values will be written instead. This can be useful when writing LS-DYNA keyword decks for import into 3rd party software that cannot handle parameters.

Write all solids in 2-line format means that all element solids will be written in the newer 2 line format (EID and PID on the first line, up to 10 nodes on the second line).When this option is NOT set, Primer will write out solid elements in the older one line format if the solids have 8 nodes or less.

Check files: listing any errors and warnings, has the options:

Write to main file	Lists errors and warnings at the top of the master file
Write to <fname>.check	Writes errors and warnings to a separate .check
Don't write check file	No errors or warnings are written

Model mass and C of G output has the options:

Write to main file	The overall model mass and Centre of Gravity are written to the master file
Don't write	No mass and C of G output is written

Include file mass output has the following options:

Write to each include file	The mass of the items in each include file are written to that file
Don't write	No include file mass is output

Connections output has the options:

Write all connections	All connections in the model are output.
Suppress all connections	No connections are output
Suppress created by Model > Check	Only connections read in or created by the user are written. Those created during a Model Checking operation are omitted.

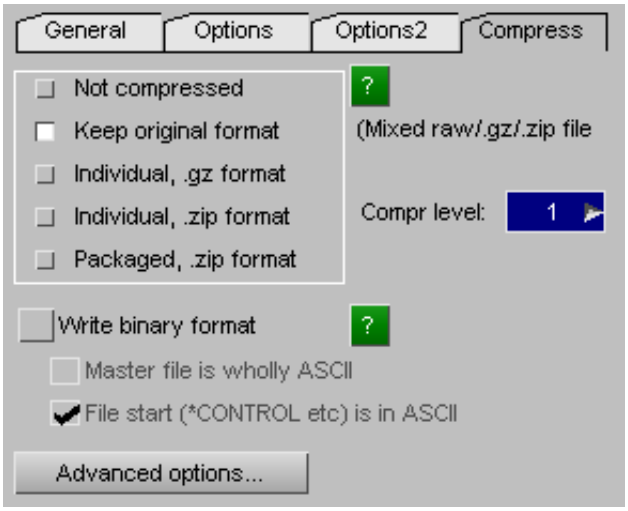
Connections are an innovation in PRIMER 9.3 which provide a common way to handle the processing of spotweld, bolts, adhesives and other connection methods. They are described in section 6.12

Assembly output has the options:

Write assembly data in Primer format	Write assembly hierarchy information in Primer format
Write assembly data in HM format	Write assembly hierarchy information in Hypermesh comment format
Write assembly data in ANSA format	Write assembly hierarchy information in ANSA comment format

Advanced options is covered under the Compress section below

Pre-out: Compress tab



Compression and binary output format are new in PRIMER 15, and provide a way of both speeding up output and also reducing file sizes.

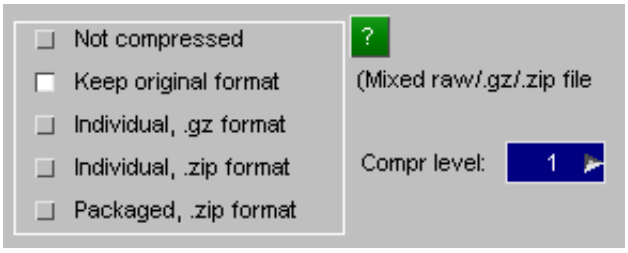
- Compression uses the industry standard "ZLIB" library to compress files into .gz and .zip formats.
- Binary files are proprietary to PRIMER and provide a way of storing data more efficiently, they also improve reading and writing speed

The defaults in PRIMER 15 are:

- The compression status of input files is "remembered" and re-used during output, the "keep" option.
- Binary output is ***not*** used by default, even if the input file contained binary data.

These defaults can be changed via preferences, and using the **Save settings to oa_pref file** button will update these preferences automatically to the current settings..

Compress files



If this option is used then the current output files (of whatever format: small, large, binary) are compressed using the Huffman coding in ZLIB into one of the following formats:

".gz" format	Industry standard "gzip" format as handled by g(un)zip etc
".zip" format	Industry standard "zip" format as handled by Winzip etc

The .gz and .zip files produced by PRIMER can be read by the standard Winzip, gzip, etc utilities; and PRIMER can read files created by those utilities provided that they use the default Huffman compression method.

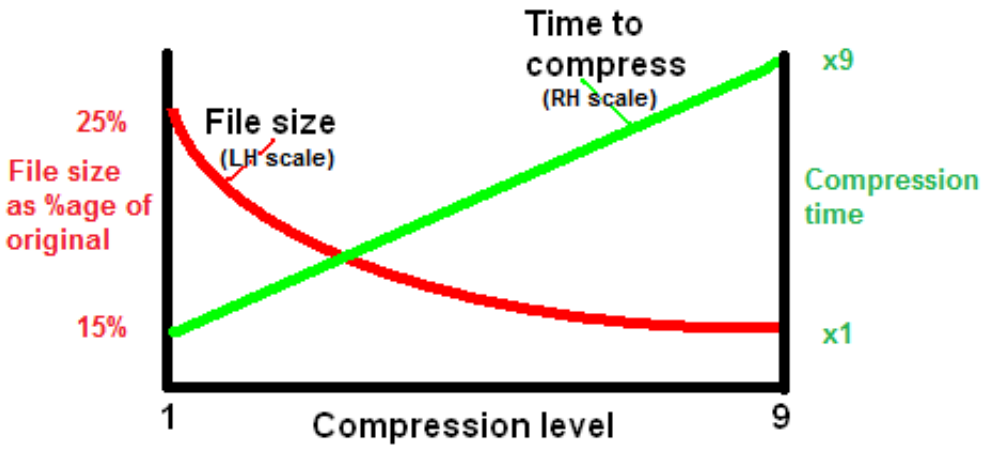
Compressed format	**Typical output**	**What is done.**
Not compressed	All output is uncompressed	Regardless of the input compression status all master and include files are written in uncompressed form. Filnames will be adjusted if necessary, for example if compressed file **model.key.gz** was read it will be written as **model.key**.
Keep original format	Output compression matches input.	For each file individually, ie master file and any include files, the input compression status is "remembered" and re-used during output.
Individual .gz files	master.key => master.key.gz child.key => child.key.gz	Regardless of input compression each file is separately compressed from its original form into .gz format, producing multiple .gz files. Files will be renamed as necessary, for example **include.key** will become **include.key.gz**
Individual.zip files	master.key => master.zip child.key => child.zip	Regardless of input compression each file is separately compressed from its original form into .zip format, producing multiple .zip files. Files will be renamed as necessary, for example **include.key** will become **include.zip**
Packaged .zip format	master.key } child1.key } => master.zip child2.key }	Regardless of input compression all files, that is master file and all include files, are compressed into a single master .zip file. The directory structure **INCL**/*child* is preserved within the zip package, and the master file will be renamed as necessary, for example **master.key** will become **master.zip**.

The treatment of include files is covered in Include files and Compression above.

Level: the degree of compression.

The Huffman coding used by ZLIB works by looking for repetition of similar blocks of data in a file, and replacing the 2nd and subsequent references to a (long) block of data with a (short) reference to that block. Clearly the greater the distance within the file a search is made the more likely a match is to be found, but equally this search will take longer to perform, so there is a trade-off between compression time and file size.

Normal implementations of gzip and winzip use "default" compression, which is actually 6 in a range of 1 (least) to 9 (most), and while the time taken to compress a file rises linearly with "level" the amount of compression obtained tends to be some power less than one of the "level value. We could draw a graph something like this (values are notional):



File size reduction

Most users of PRIMER will be working interactively and are likely to want the fastest output rather than the smallest file size, and experience shows that setting the lowest level of compression gives an acceptable result without costing too much time to compress during file output, therefore the default is 1 which tends to reduce typical small format ascii files to about 20 - 25% of their original size. However users who wish to prioritise small file size may wish to set a higher value.

Note: the compression level will not affect the ability of the standard gunzip and winzip utilities to read these files, they will read files written with any level of compression in the range 1 to 9.

Speed of output of compressed ascii files.

At the lowest level of compression speed-up of writing ranges from about 1.25x faster to a fast local disk, and up to 4.0x faster to a remote disk on a slow network.

Reading of compressed ascii files by PRIMER

PRIMER 15 can read compressed files directly, there is no need to convert them externally back to ascii format. When given a file with ".gz" or ".zip" extension it will automatically decompress the contents.

Speed of reading compressed ascii files.

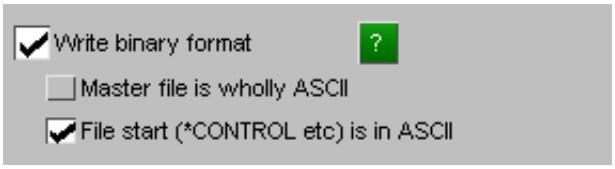
Decompression is threaded (parallelised) so decompression does not have much impact on the speed of reading. From a fast local disk there is little difference, from a remote disk on a slow network reading can be up to 2.5x faster.

Reading encrypted .zip files

PRIMER 16 will read encrypted .zip files, detecting encryption automatically and prompting you for a password. (Earlier **PRIMER** versions will treat encrypted .zip files as corrupt and fail to read them.)

Write Binary Format

Binary format is new in PRIMER 15, it provides a way of both reducing file sizes and speeding up input and output.



Description of binary format files

Each line of a normal keyword file is made up of numbers and text, and these are stored in a way that is readable by humans using ascii text. For example consider the format of a *NODE card in LS-DYNA which is:

Field name	**NID**	**X**	**Y**	**Z**	**TC**	**RC**
Format width	I8	F16	F16	F16	I8	I8
Internal #bytes	4	4	4	4	1	1

Therefore this card requires (8 + 16 + 16 + 16 + 8 + 8) = 72 ascii characters for output, yet only (4 + 4 + 4 + 4 + 1 + 1) = 18 bytes for internal storage, which is 25% of the space.

Moreover when this file is written out in ascii format considerable cpu time has to be spent converting the internal binary format to the external ascii format, in fact for many models this conversion process is the most "expensive" part of the output process.

So writing this card in its internal binary format is not only faster, but also results in substantially smaller files, and this is what PRIMER’s binary format does. The file also encodes the format of each card, so a binary file can be converted back to a normal ascii file without any loss of information.

Format of the binary file.

A binary file starts off in ascii just like a normal keyword output file, then swaps to binary when the keyword

`*START_BINARY`

is encountered. Thereafter the file is binary and not human readable. The details of the binary format are proprietary.

This is not a standard LS-DYNA keyword, and LS-DYNA will be unable to read the file unless it is converted back to ascii.

Formatting options:

Master file is wholly ascii	If this option is selected then the master file of a model will not be converted to binary format. This can be useful when the master file just contains *INCLUDE, *CONTROL and *PARAMETER cards and the bulk of the model's data is in include files, since it means that the top level file can be hand-edited in its entirety.
File start (*CONTROL etc) is in ascii	If this option is selected then the start of every file (master and include) will be written in ascii format. "Start" in this context means initial title and comments, plus any *CONTROL, *PARAMETER and *INCLUDE cards. If *INCLUDE_TRANSFORM are present then any *DEFINE_TRANSFORMATION cards associated with them will also be written in ascii. This means that the start of the file remains human readable, and even though a text editor will be bamboozled by the binary data (so don't try to edit it!!) it is at least possible to inspect the top of the file in a read-only fashion.

File size reduction

Because information about the format as well as the contents of the line have to be stored, the actual file size tends to be about 30% of the original (small format) ascii file. When writing large (wide) format files the reduction in file size is substantially greater since the external ascii format is so much more verbose, but the cost of encoding this formatting is no greater, so files may be as little as 15% of the original size.

Speed of output of binary files

Speed-up of file writing tends to be in the range 6.5x on a fast local disk, to better than 18x on a remote disk on a slow network.

Reading of binary files by PRIMER

PRIMER 15 will read binary files directly. No special action is required on the part of the user as the file is opened in the normal way and PRIMER then detects the **`*START_BINARY`** keyword and interprets the binary format directly.

Speed of reading binary files

From a fast local disk input tends to be about 1.5x faster, rising to 3.0x times faster on a slow remote disk. (Binary input means that the speed bottleneck moves from file read to internal database construction.)

Converting binary files back to ascii format

PRIMER users will be able to "read binary, write ascii" if they wish.

For users who do not have access to PRIMER Oasys Ltd provide a free conversion utility that will turn binary format files back into ascii format.

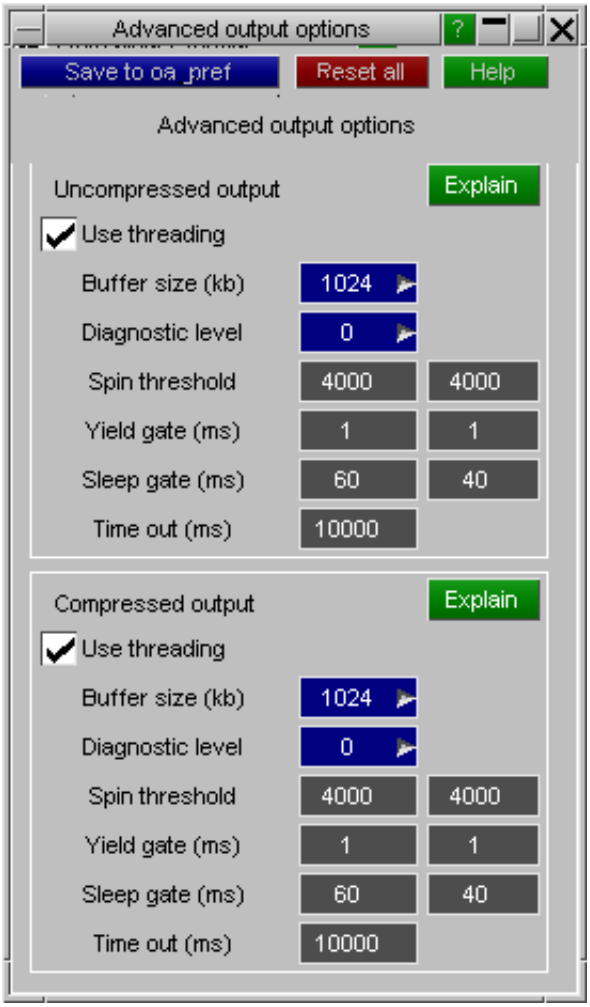
Compression of binary format files

Compression and binary format may be used together, ie a binary file can be further compressed to .gz or .zip format.

Experience shows that at the standard level of compression a typical model which compresses to 20 - 25% of its original size in ascii format will compress to 15 - 20% in binary format, so there is some gain to be had.

Compression at the default level of 1 increases the time taken to write and read binary format files by a small amount, usually less than 5%. On a very slow network there may be an improvement in speed if the benefit of transmitting the smaller file size outweighs the cost of compression.

Advanced options



Specialised tuning of compressed and uncompressed i/o.

You don't need to understand this, in fact it is usually best left well alone!

By default PRIMER 15 now threads (parallelises) both reading and writing of keyword files by using two separate threads:

- The master thread is the main process which handles the database.
- The child thread handles the disk i/o operation, including (de)compression if that is being used.

This allows the child thread to be busy handling disk i/o while the master process is concurrently busy dealing with the data to be written and read. There are obvious advantages to this when (de)compressing files, but a more subtle advantage also arises when dealing with a slow disk as the latency of the i/o process (ie its stop/start nature) is less likely to hold up the master thread.

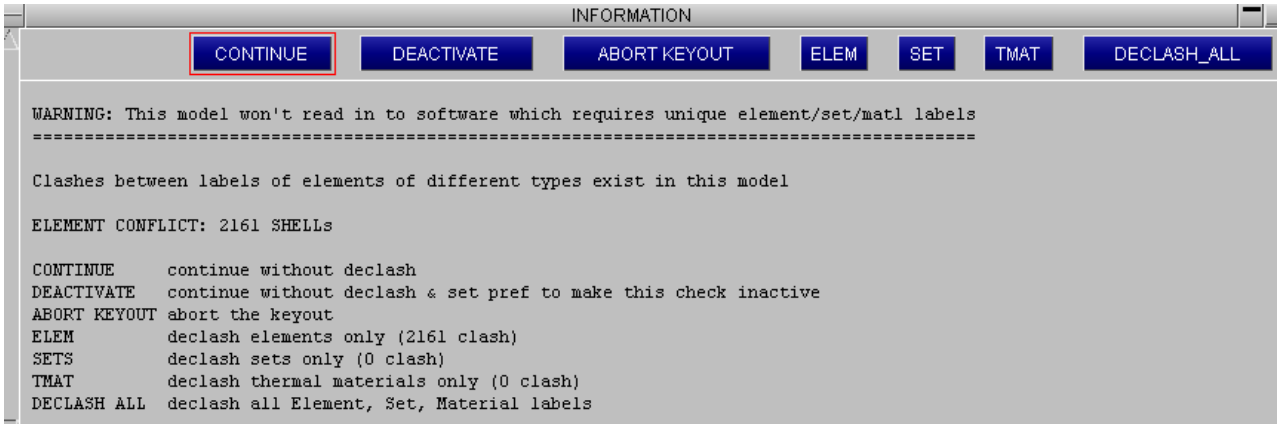
Normally this "just works", but when synchronising slave and master processes on a time-sharing operating system there are of necessity various compromises that have to be made, and tuning settings that have to be arrived at by trial and error. The settings here are quite complex, and changing them adversely could absolutely cripple i/o performance so it is strongly recommended that you leave them alone unless you encounter problems.

If you do encounter i/o problems please contact Oasys Ltd. In the first instance we will ask you to turn on the diagnostics, then depending on what that reveals we may ask you to change some of the tuning settings here. These settings can be saved in your oa_pref file so hopefully if changes are required it will be possible to save the optimum values for your system once these have been determined.

Declash on writing a model

Some pre-processing software require that element, set and material cards do not clash for the different types (e.g. shell 1 and solid 1 are not permitted), although this is NOT a requirement for LS-Dyna or Primer.

On keyout of a model which contains such clashes, the user will be prompted by a clash information panel and given the option to fix all clashes or those on selected types. If the user selects DEACTIVATE the pref setting <check_for_clashing_element_and_set_labels_on_keyout> will be set to FALSE and the panel will not appear again.



When writing out a model may change its representation in memory

Writing out a model does not usually affect its contents in memory (*) in any way, the exception being when label resequencing is required to prevent clashes in the external format. This usually arises because LS-DYNA permits different classes of item to have overlapping labels, whereas the external data format does not. For example IDEAS universal file and PATRAN neutral file formats do not permit overlapping element numbers.

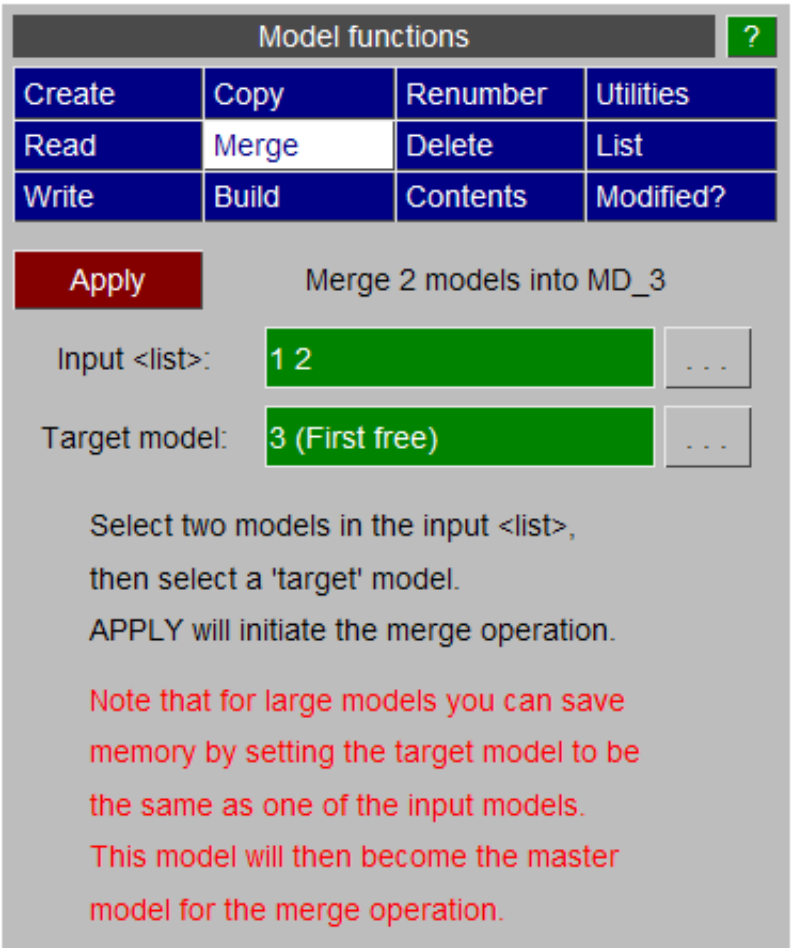
In this situation the labels in the internal model are permanently modified as required. If this is not acceptable **COPY** the model before you write it out, then **DELETE** the model that has been modified during the write operation.

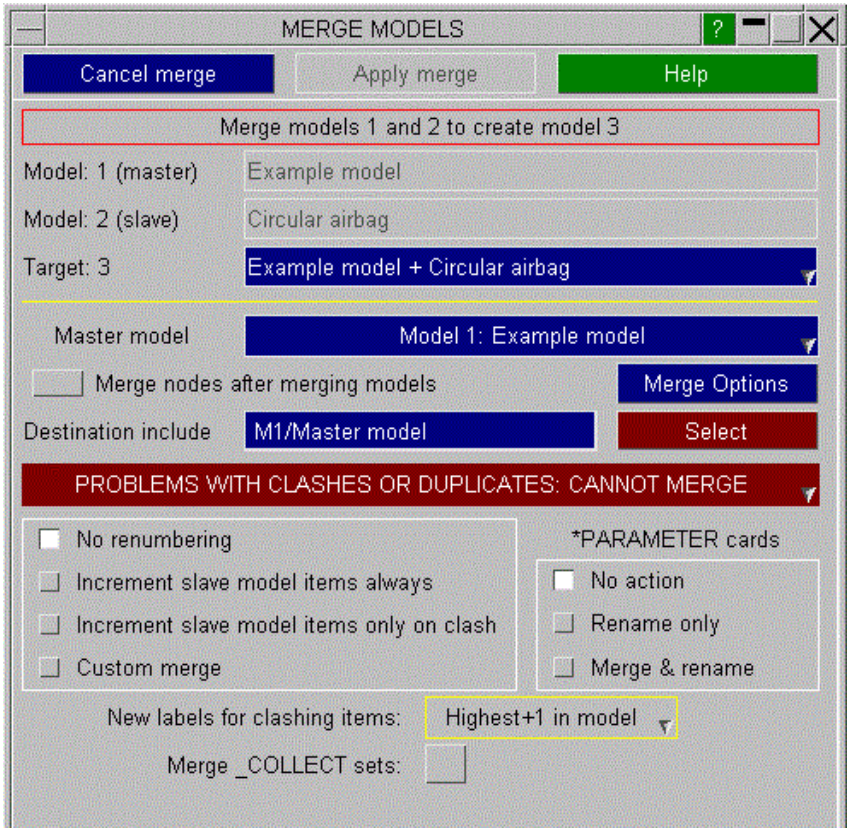
(*) There are some pathological cases where this may not be true. For example, <INITITER> on *CONTACT_..._MPP where a blank entry means use option 2 not (as one would expect) option 0 (!) In this case, Primer does not unconditionally accept an as read value of 0 as correct and keyout (at the user’s discretion) may modify it.

3.4 MODEL > MERGE

"Merging" combines two input models into a single output one.

- Define an input <list> of two models to be merged. In this example the user has selected models 1 and 2.
- Define a "target" model, which can exist, but generally will be an unused model.
- Click on **Apply** to initiate the merge.





If the output model is different to the input models then the window on the right is shown.

The two models which are selected for merging are shown in the top of the window. One of the models is designated as the master model and one is designated as the slave model. This can be changed by using the **Master model** popup.

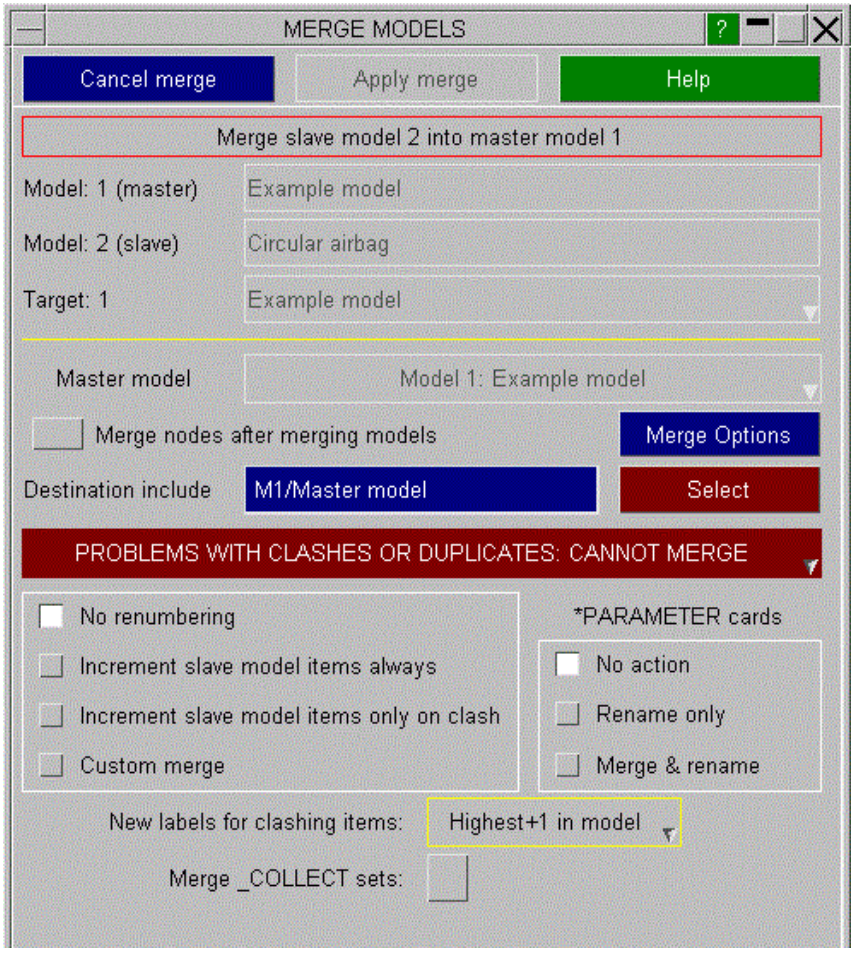
A title for the target model will automatically be created by using the titles from the master and slave models (in this example 'Example model + Circular airbag'). Alternatively the popup can be used to select from predefined titles or a new title can be typed in. This title will be used when the new model is created.

A destination include file can be specified if you want the slave model entities to end up in a particular include file in the master model.

In this example there are potential problems with merging the two files together. This is shown by the red button **PROBLEMS WITH CLASHES OR DUPLICATES: CANNOT MERGE**.

If there are no potential problems with merging the two files together the button would be coloured green and labelled **NO PROBLEMS WITH CLASHES OR DUPLICATES: OK TO MERGE**. Additionally the **APPLY_MERGE** button will be ungreyed to allow the merge to proceed.

If you want to perform a "merge nodes" operation when merging models (e.g. if you reflected a half vehicle model and are merging the 2 halves together then select the **Merge nodes after merging model** checkbox. For more details see section 3.4.3.



If one (or both) of your input models is very large then copying the model data into a new model when merging could mean that you run out of memory on your machine (as you approximately double the memory used when merging).

In this case you can set the target model to be the same as one of the input models. As you are not creating a new model no more memory will be required. However, as you are changing one of the input models make sure that your original model is saved before you start the merge.

If the target model is the same as one of the input models then that model is automatically chosen as the master model. The options that allow you to change the master model will be unavailable.

Additionally some of the options for fixing clashes in the custom merge case will not be available as they are not possible.

3.4.1 Options to fix clashes

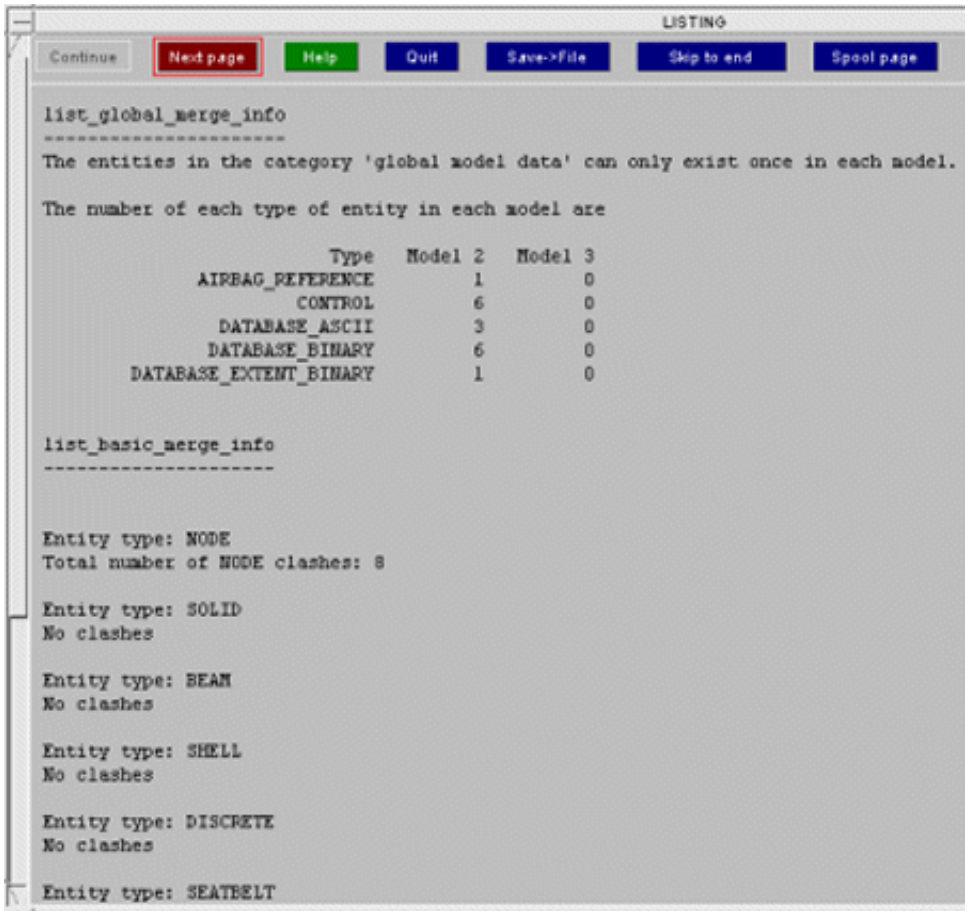
- **No renumbering**. This option is only possible if there are no clashes or duplicates in the two source models.
- **Increment slave model items always**. This option will increment the labels of items in the slave model by the maximum value in the master model. For example, if nodes in the master model ranged from 1 to 100, and from 25 to 40 in the slave model, the nodes in the slave model would be incremented to be in the range 125 to 140. This will occur even if there are no clashes between labels.
- **Increment slave model items only on clash**. This option is similar to option 2 except that only items which actually clash between the two source models will be incremented.
- **Custom merge**. This option gives much greater control on how the models are merged together. It is only recommended for experienced users.

When there are parameters with the same name in both master and slave models, this is can also be a clash why merging the models is not possible. There are the following options to deal with this:

- **No action**. With this option the models can be merged only if there are no parameter name clashes.
- **Rename only**. When there are parameters with clashing names, Primer automatically generates new names by appending _1, _2 etc. to resolve these clashes.
- **Merge & rename**. Here Primer automatically renames clashing parameters when necessary, but merges parameter definitions with the same name and value from both source models.

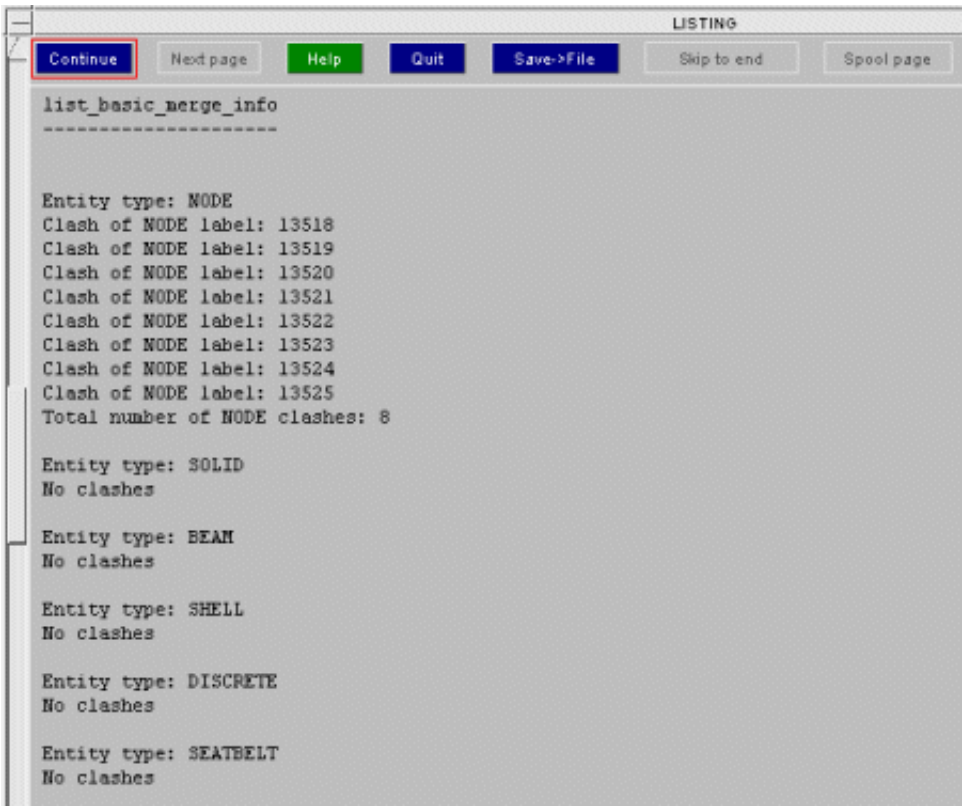
Determining the nature of clashes

The two main problems with merging models together are items which have potential label clashes such as nodes, elements, parts etc. and items which are only allowed once in a model such as control and database cards, airbag reference geometry etc.



You can get a summary (above) or a detailed list (below) of the potential merge problems by using the popup on the **PROBLEMS WITH CLASHES....** button.

For example in the summary above you can see that there are clashes of nodes. The detailed list (below) tells you which node labels are clashing as well as the total number.



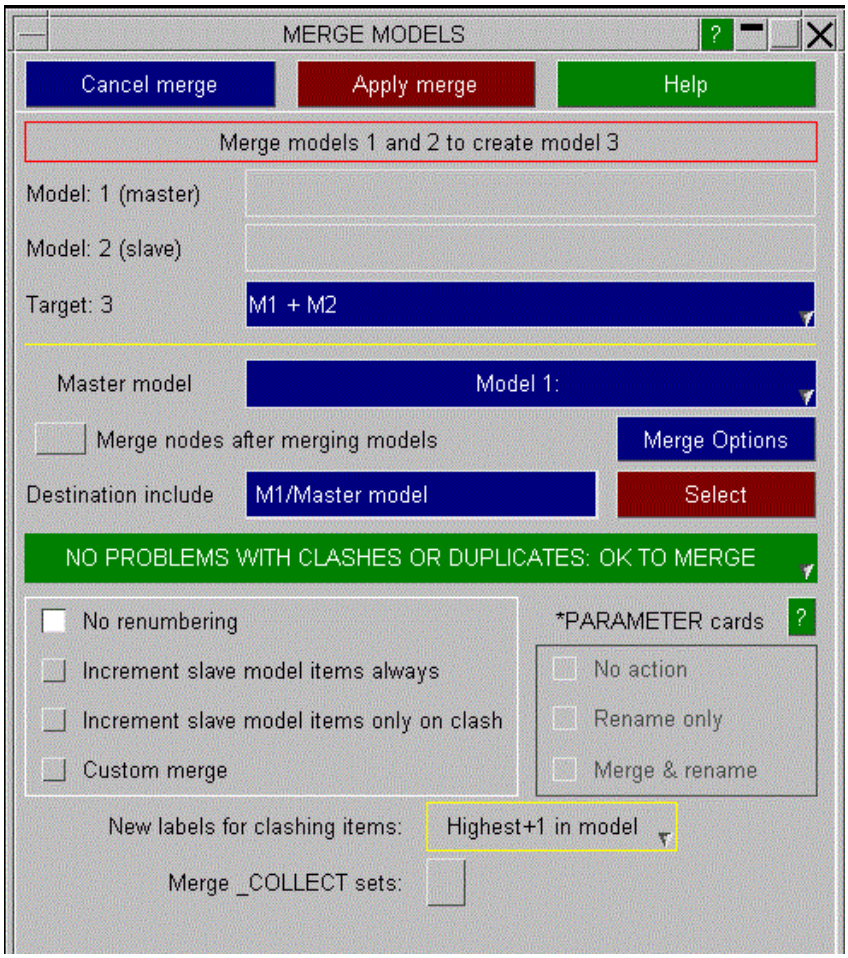
The problem with label clashes is easily solved by using either option 2 or option 3 above (or custom merge for greater control)

The problem with duplicate items such as control cards etc is solved by the following method in options 1, 2 and 3.

- If the item exists in the master model but not in the slave model it is taken from the master model.
- If the item exists in the slave model but not in the master model it is taken from the slave model.
- If it exists in the slave and master model it is taken from the master model.

Other items in the model which do not have labels such as constrained cards and boundary cards cannot clash so there is no problem with merging these entities. All of the cards from both models are used in the merge process.

Once you have chosen a method for merging the two models together which fixes any potential problems the **PROBLEMS WITH MERGE...** button in the main merge window will change message and turn green. The **APPLY MERGE** button will be ungreyed and the merge can proceed.



This method is the easiest way to merge two models together. The original models will not be deleted after merging so if the outcome of the merge is not what you wanted you do not lose your original models. However, it is good practice to save your models before attempting to merge them together.

For much greater control on how models are merged together the **Custom Merge** option can be used but this is only recommended for experienced users as there are potential problems. This is described in section 3.4.2.

New labels for clashing items

By default the labels for clashing items will be changed to be greater than the highest label in the master and slave model. e.g. if the labels for nodes in the slave model were in the range 100 - 1000 and in the range 200 - 2000 in the master model the clashing node labels would be changed to be 2001, 2002 etc. This can be changed with the **New labels for clashing items** popup. The default value of **Highest+1 in model** is the above behaviour. Instead a start label can be given for new labels by changing this to **Start at label** and giving a starting label in the text box. For example, with the above labels you may want clashing labels to start at 100000.

Options for *PARAMETER cards

When there are parameters with the same name in both source models, the models cannot be merged without further action. The following options are available:

- No action: In this case the merge operation may not be possible.
- Rename only: Whenever there are name clashes between parameters without _LOCAL suffix, Primer automatically renames the parameters in one of the source models by appending _1, _2 to their names to avoid these clashes.
- Merge & rename: Here Primer merges parameters with the same name, data type and value from both source models into one parameter definition for the target model. If the data type or value does not match, then parameters are renamed as in the previous option. Currently Primer does not merge _EXPRESSION and _MUTABLE parameters, but always renames them on clash.

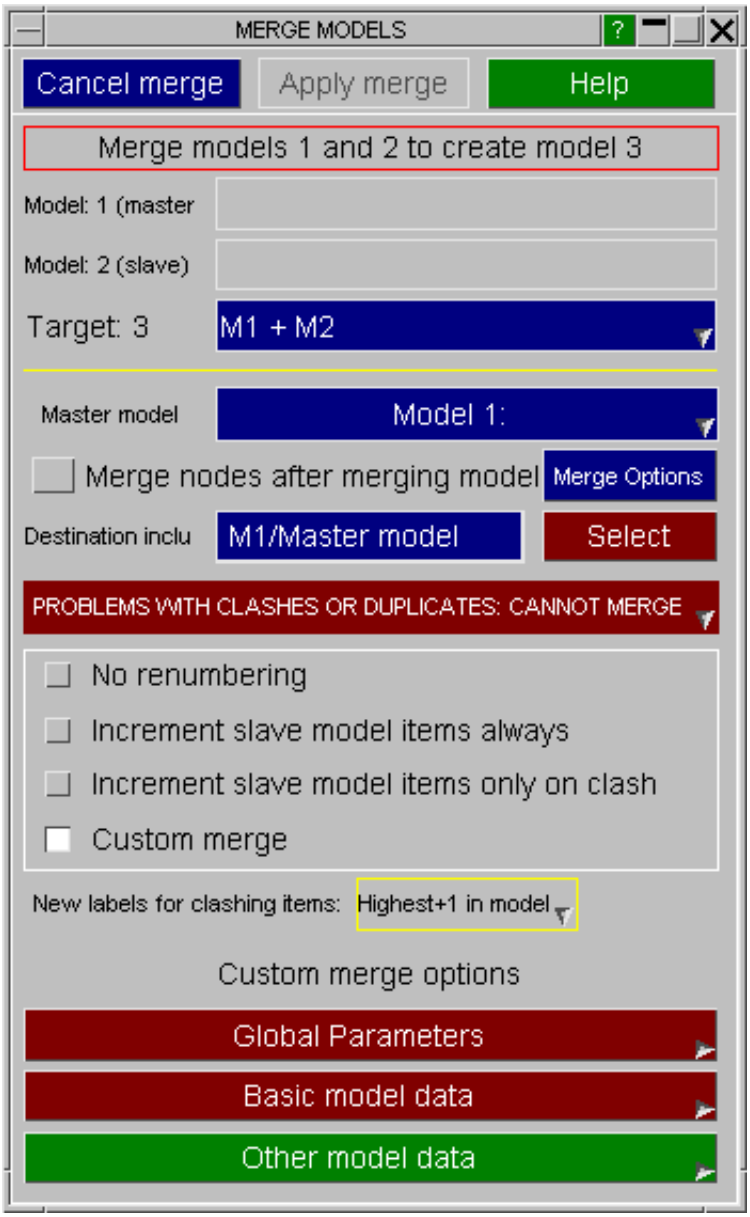
3.4.2 Custom Merge

More control over merging process

The custom merge option gives control of how each type of entity is merged from the two source files. For example, shells can be merged together in a different way to nodes and parts.

If custom merge is selected, three extra buttons are shown at the bottom of the window:

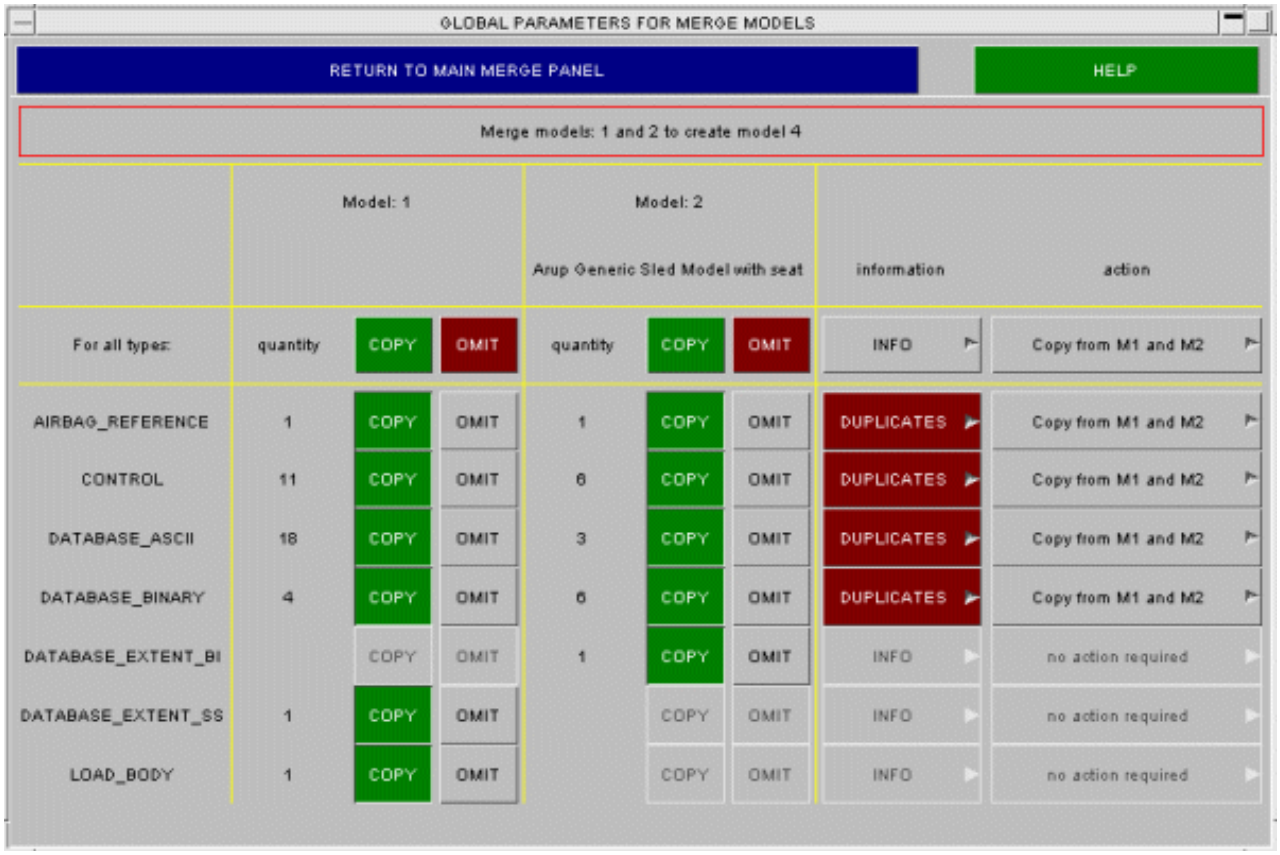
- **GLOBAL PARAMETERS** This is used for things that can only exist once in the model such as control and database cards, airbag reference geometry etc.
- **BASIC MODEL DATA**. The basic model data comprises entities which have labels and so can clash. E.g. shells, solids, nodes, loadcurves etc.
- **OTHER MODEL DATA**. The other model data comprises entities which do not have labels and so cannot clash. E.g. boundary cards, constrained cards.



If a button is red then there are problems with merging that type of data. For example in the figure above there are problems with the basic model data. The "other model data" button will always be green because there cannot be clashes as the entities have no labels. A summary or a detailed list of the problems for each category is available by using the popup on each button.

Clicking on each of the custom merge buttons will bring up a new window showing all the entity types in that category and the associated problems. Using these windows the problems can be solved and then the models can be merged together.

Custom Merging Global Parameters

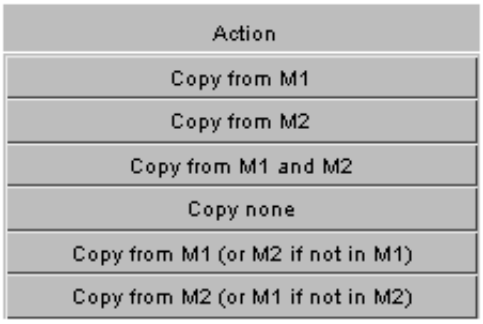


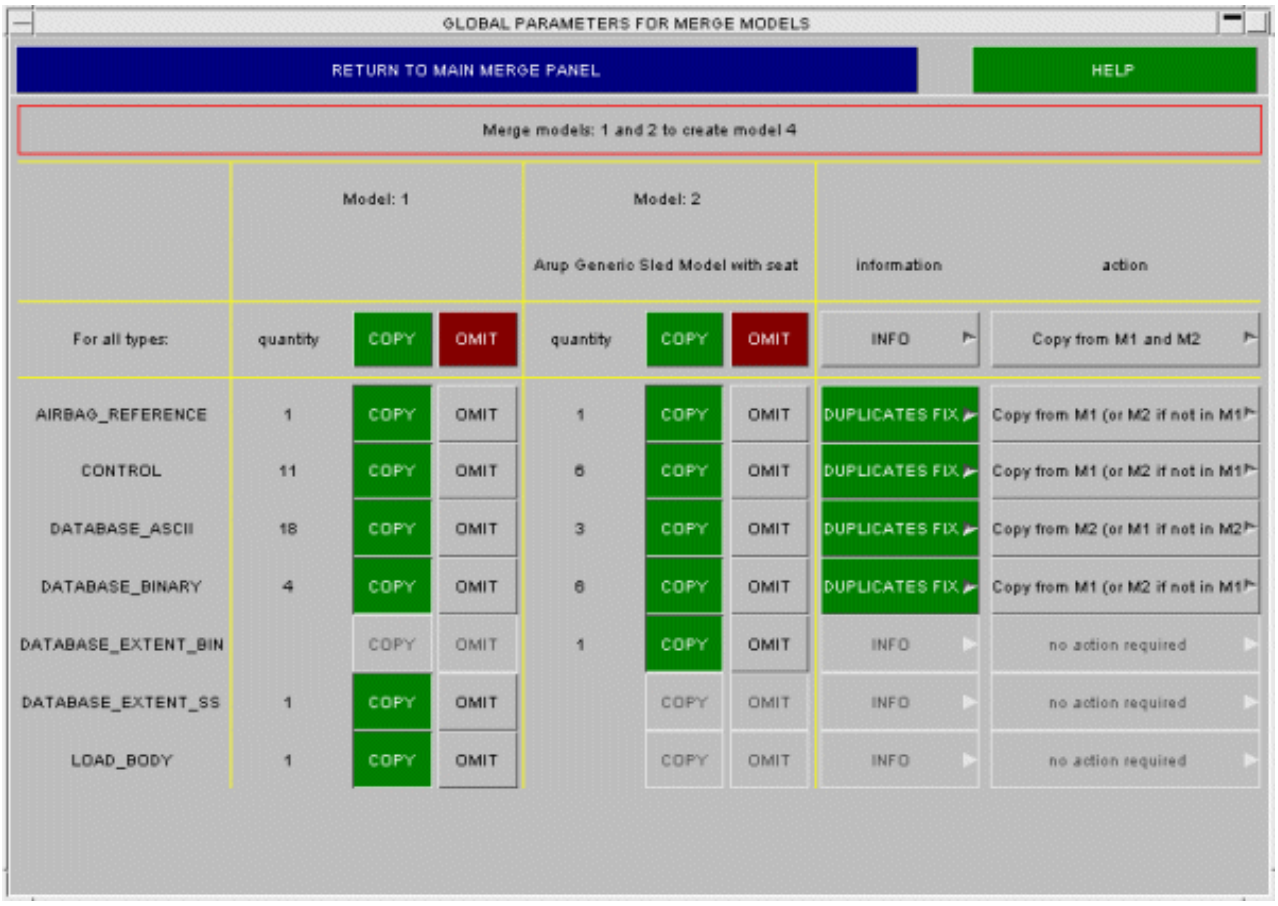
This figure shows the global parameters window for the example above. You can see that:

DATABASE_EXTENT_BINARY **DATABASE_EXTENT_SSSTAT, LOAD_BODY**	Cards only exist in one of the 2 models so there is no problem.
AIRBAG_REFERENCE_GEOMETRY, **CONTROL,** **DATABASE_ASCII** **DATABASE_BINARY**	Cards exist in both models and so there are duplicate cards which is causing a problem.

Resolving clashes using the global "Action" popup menu

The problems arise because the default action for all the types is to copy from both models (**Copy from M1 and M2**). This can be changed for all types, or for each individual type by using the action popup. The actions are self explanatory except for the last two. The **Copy from M1 (or M2 if not in M1)** card is only in the first model or is in both models. If will take a card from the first model if the the card is only in the second model it will be taken from the second model. This is useful if you want to make sure that all control cards are copied from both models. They will be taken from both but the control cards in the first model will take precedence over the control cards in the second model.

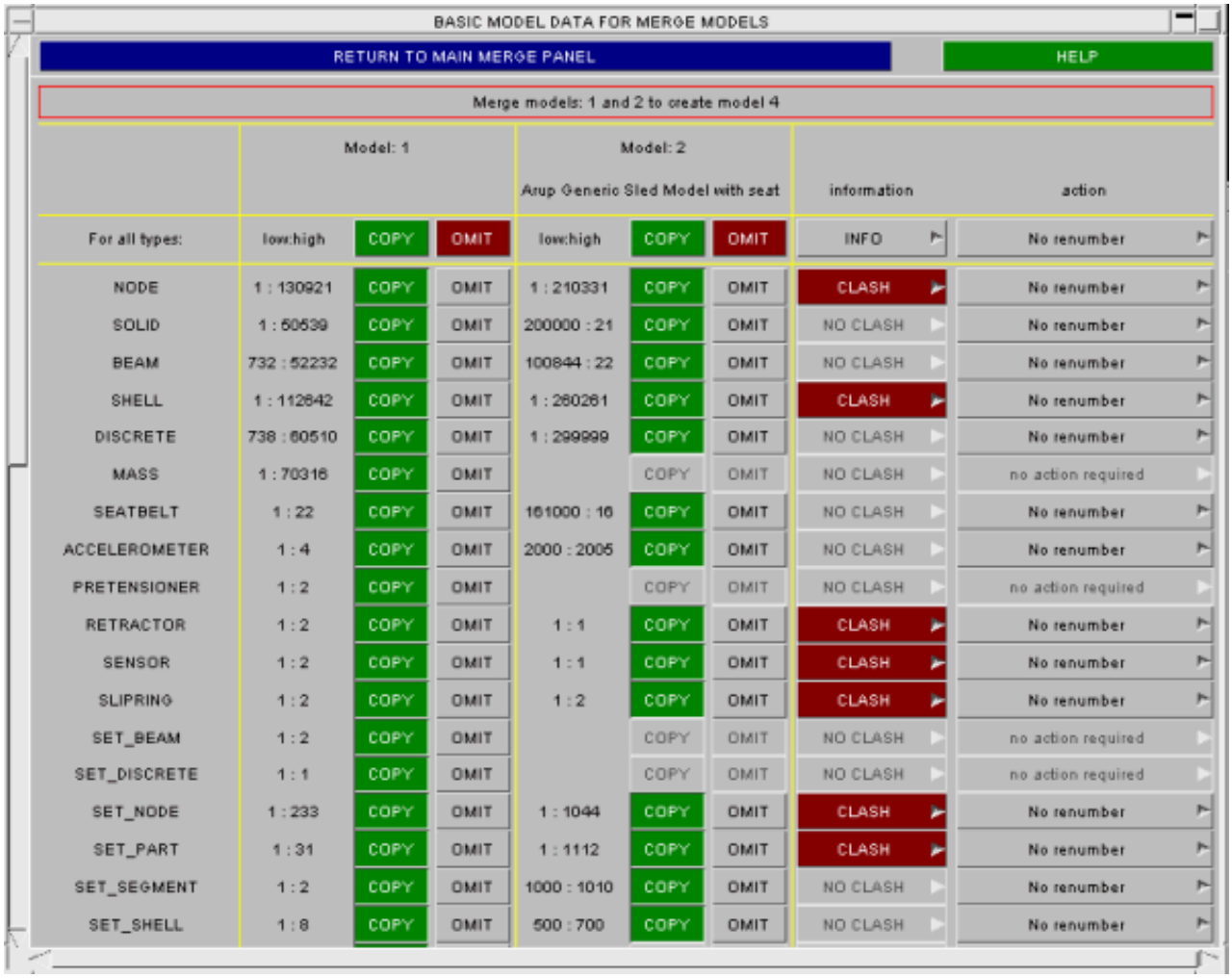




As appropriate actions are chosen for each type (or all types) the **DUPLICATES** will be replaced by **DUPLICATES_FIXED**. The figure above shows the same model after actions have been chosen to fix problems.

Summary or detailed information on a problem with each type is available by using the information popups.

Custom Merging Basic Model Data

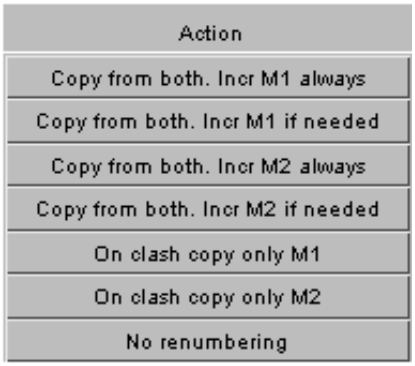


	Model: 1			Model: 2 Arup Generic Sled Model with seat			information	action
For all types:	low:high	COPY	OMIT	low:high	COPY	OMIT	INFO	No renumber
NODE	1 : 130921	COPY	OMIT	1 : 210331	COPY	OMIT	CLASH	No renumber
SOLID	1 : 60539	COPY	OMIT	200000 : 21	COPY	OMIT	NO CLASH	No renumber
BEAM	732 : 52232	COPY	OMIT	100844 : 22	COPY	OMIT	NO CLASH	No renumber
SHELL	1 : 112842	COPY	OMIT	1 : 260261	COPY	OMIT	CLASH	No renumber
DISCRETE	738 : 60510	COPY	OMIT	1 : 299999	COPY	OMIT	NO CLASH	No renumber
MASS	1 : 70316	COPY	OMIT		COPY	OMIT	NO CLASH	no action required
SEATBELT	1 : 22	COPY	OMIT	161000 : 16	COPY	OMIT	NO CLASH	No renumber
ACCELEROMETER	1 : 4	COPY	OMIT	2000 : 2005	COPY	OMIT	NO CLASH	No renumber
PRETENSIONER	1 : 2	COPY	OMIT		COPY	OMIT	NO CLASH	no action required
RETRACTOR	1 : 2	COPY	OMIT	1 : 1	COPY	OMIT	CLASH	No renumber
SENSOR	1 : 2	COPY	OMIT	1 : 1	COPY	OMIT	CLASH	No renumber
SLIPRING	1 : 2	COPY	OMIT	1 : 2	COPY	OMIT	CLASH	No renumber
SET_BEAM	1 : 2	COPY	OMIT		COPY	OMIT	NO CLASH	no action required
SET_DISCRETE	1 : 1	COPY	OMIT		COPY	OMIT	NO CLASH	no action required
SET_NODE	1 : 233	COPY	OMIT	1 : 1044	COPY	OMIT	CLASH	No renumber
SET_PART	1 : 31	COPY	OMIT	1 : 1112	COPY	OMIT	CLASH	No renumber
SET_SEGMENT	1 : 2	COPY	OMIT	1000 : 1010	COPY	OMIT	NO CLASH	No renumber
SET_SHELL	1 : 8	COPY	OMIT	500 : 700	COPY	OMIT	NO CLASH	No renumber

This figure shows the basic model data window. In this example there are problems with clashes in the types **`SHELL, SET_PART, PART, MATERIAL`** and **`CONTACT`**.

Resolving clashes using the basic "Action" popup menu

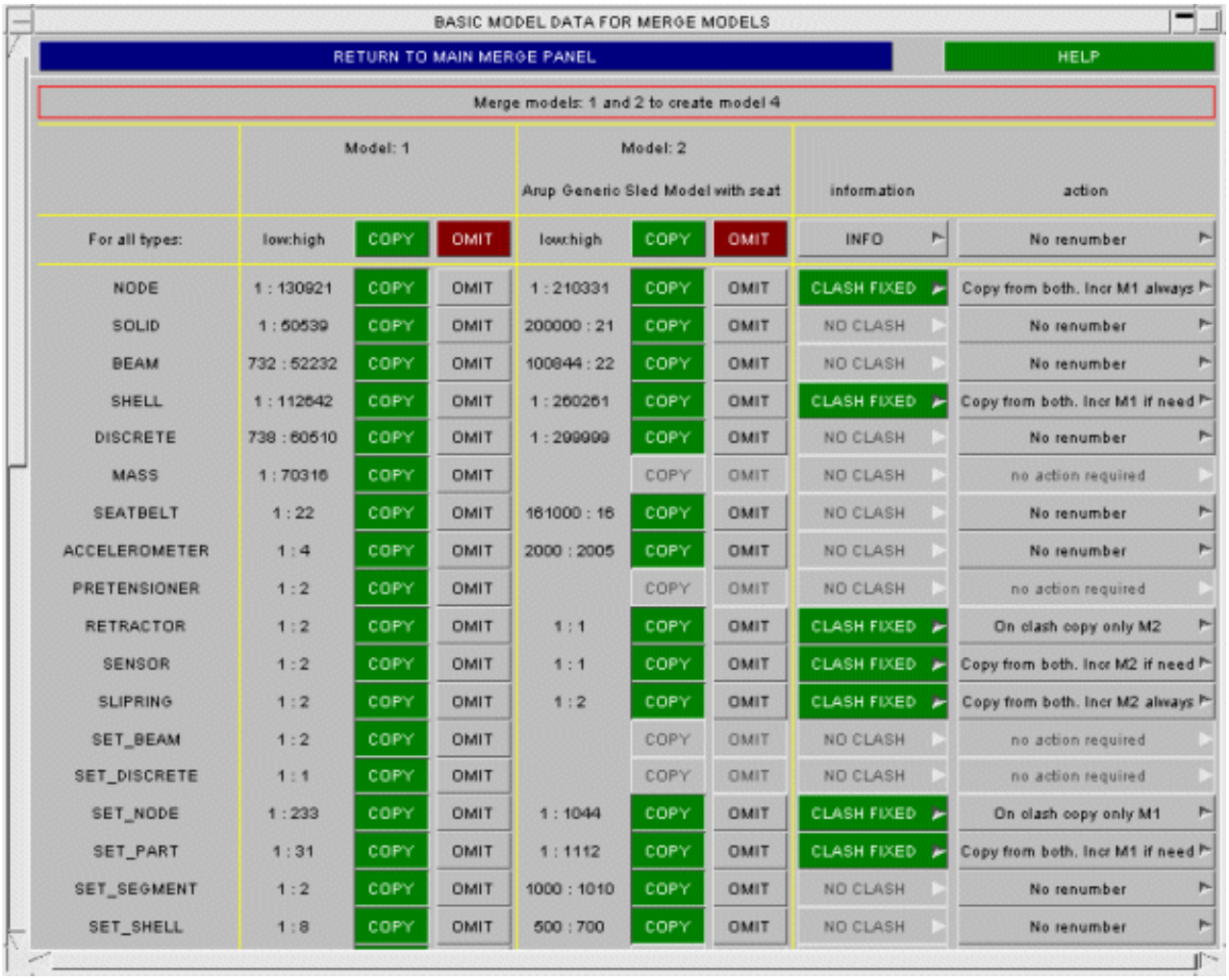
Just as in the global parameters window, actions can be used to solve the clash problems.



The actions available to resolve these, in the **CLASH** > popup menu, are, in more detail:

1 **Copy from both. Inc M1 always**. The items will be taken from both models. The labels of all items in M1 (of this type) are renumbered to be above the item labels in M2

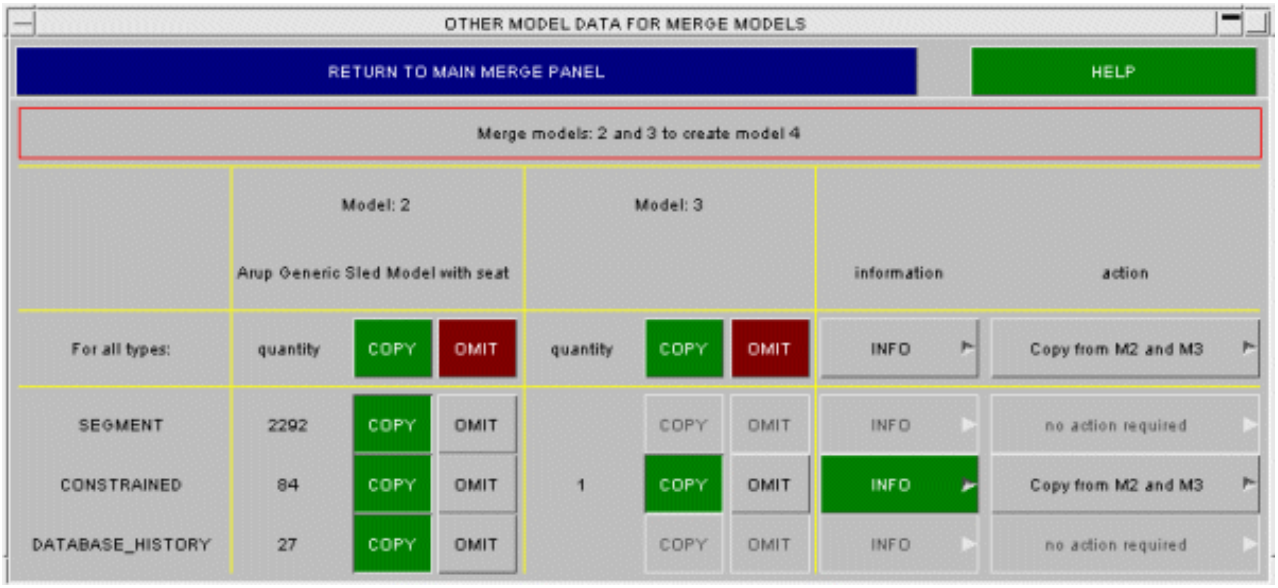
2	**Copy from both. Inc M1 if needed**	The items will be taken from both models. The labels of items in M1 (of this type) will be renumbered to be above the item labels in M2 only if there is a clash. If there is no clash the original label will be used
3	**Copy from both. Inc M2 always**.	As 1. except models swapped.
4	**Copy from both. Inc M2 if needed**.	As 3. except models swapped.
5	**On clash copy only M1**	The items will be taken from both models except when there is a label clash. When this occurs only the item from M1 will be taken
6	**On clash copy only M2**.	As 5. Except item will be taken from M2
7	**No renumbering**.	Nothing will be renumbered. This is only possible if there are no clashes.



As appropriate actions are chosen for each type (or all types) the **CLASH** will be replaced by **CLASH_FIXED**. This figure shows the same model after actions have been chosen to fix problems. The **action** column shows what has been done in each case.

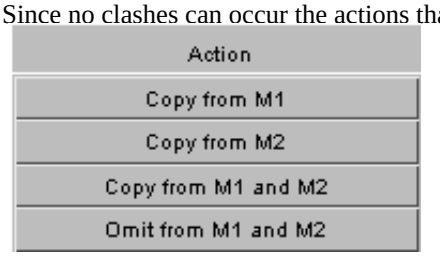
Custom Merging "Other" Model Data

"Other" means items that don’t have labels, so which can never clash, but which may require manual control.



In this example there are *AIRBAG_INTERACTION, *BOUNDARY and cards.

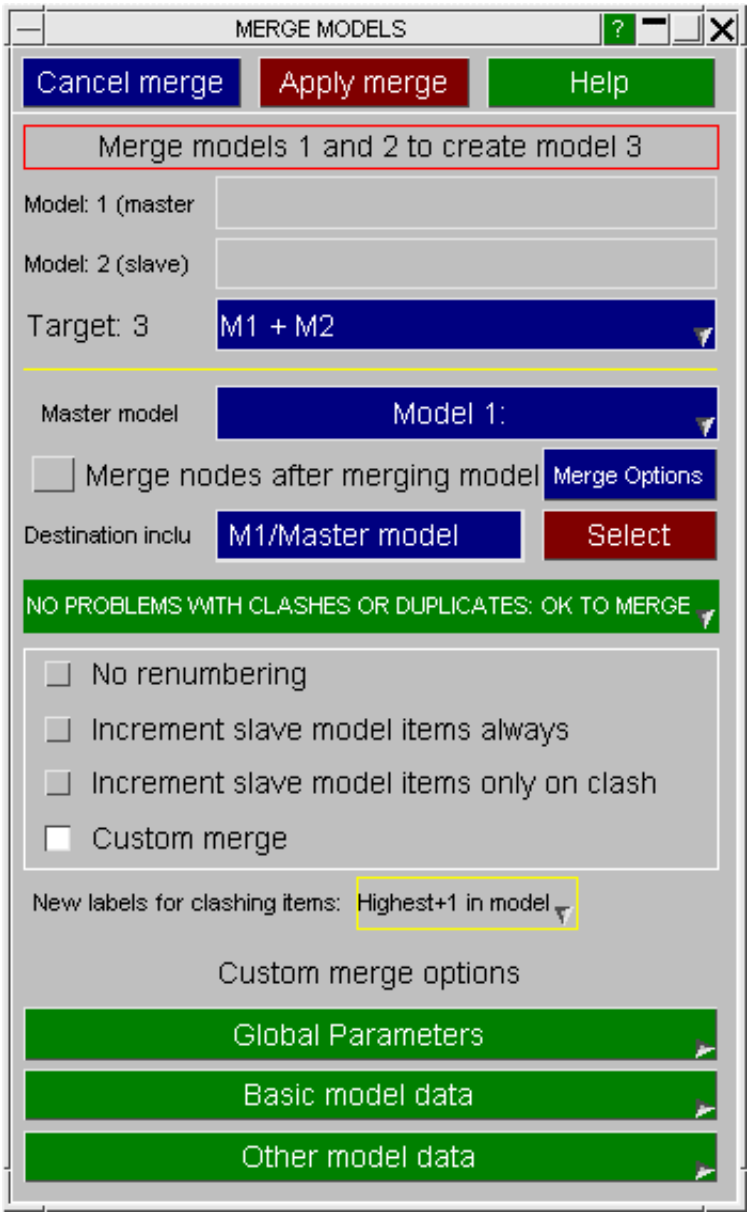
Since no clashes can occur the actions that can be taken are simply to copy or to leave each category.



Other Issues in Custom Merge

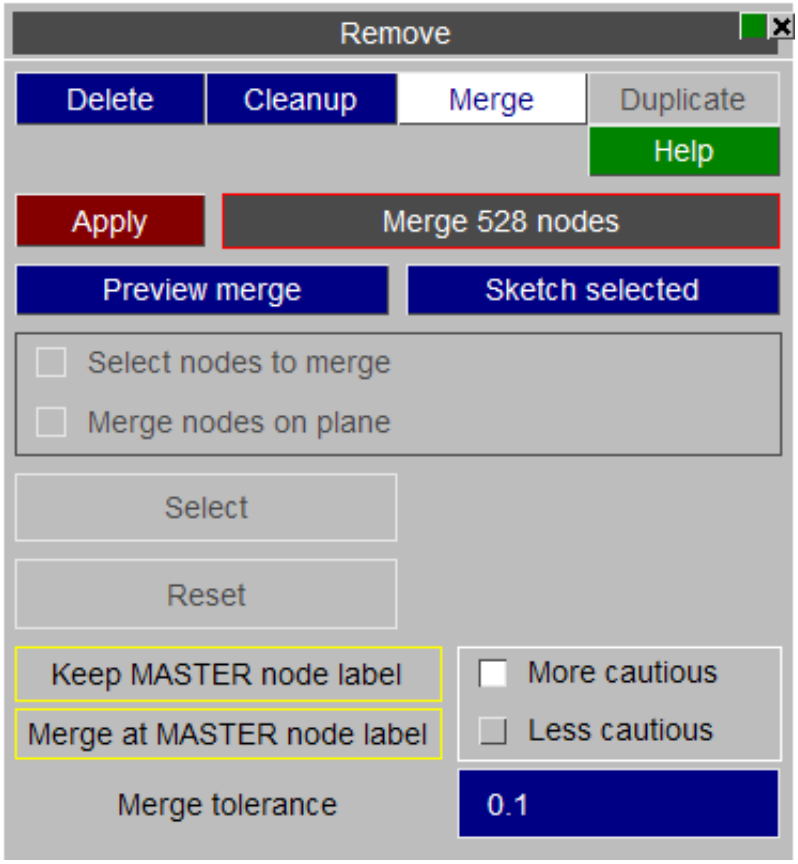
As the problems in the **GLOBAL PARAMETERS**, **BASIC MODEL DATA** and **OTHER MODEL DATA** are fixed the main merge window will be updated. When all the problems are fixed the merge can be done. The original models will not be deleted after merging.

Great care must be taken when using the custom merge options, especially when omitting some entity types from either model.



As an example imagine merging 2 models which both have a rivet from node 1 to node 2. When doing a custom merge you will be warned that there is a clash of nodes between the 2 models. If you choose an action to renumber the nodes everything will be OK. If instead you only take the nodes from one of the models then there is no clash as the rivets have no labels, but the merged model will have 2 rivets from node 1 to node 2. There are lots of similar situations which may occur which do not cause errors in the merging process but may give an unexpected result when the models are merged.

3.4.3 Merging nodes during model merge



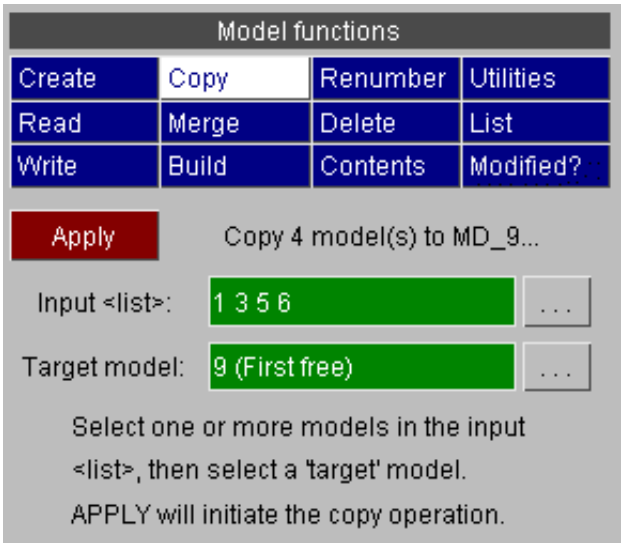
If the **Merge nodes after merging model** checkbox on the main model merge panel is selected then PRIMER will perform a node merge on the target model after the models are merged together.

The merge nodes panel is started in a special mode where only nodes from the slave model can be replaced by nodes from the master model. In this mode:

- The node label from the master model is always retained.
- The position of the merged node will always be the position of the node in the master model.

For more details on merging nodes see the merge nodes section of the manual.

3.5 MODEL > COPY Copying models internally.



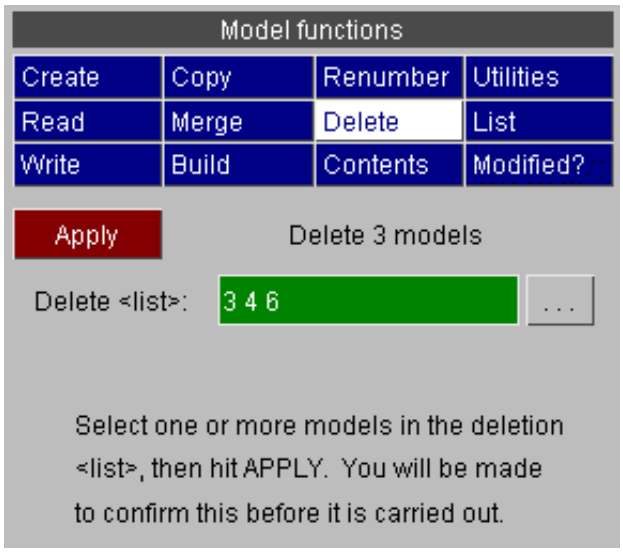
You can copy a <list> of ***n*** existing models to ***n*** new models starting at model ***i***.

The process is simple, as shown in this figure:

- Select a list of 1 or more input models (which must all exist).
- Select the first target model (which must not exist).
- Press **APPLY** to start the copy operation

The input models are copied in the order defined to new models starting at the target model id. New models are created in a contiguous sequence of free models: any existing ones are skipped over, not deleted.
Copying a model duplicates all the internal data; and the new model(s) created are totally separate from their originals. (Internally the model is effectively written out and read back into the new model, although this is carried out in memory and no disk i/o is performed.) For this reason a **COPY** operation may take a little time, although it is usually still much faster than re-reading from disk file.

3.6 MODEL > DELETE Deleting internal models



Entire models may be deleted from memory with the **DELETE** command (Models are only deleted from PRIMER, not from disk).

Deletion is carried out as shown in this figure:

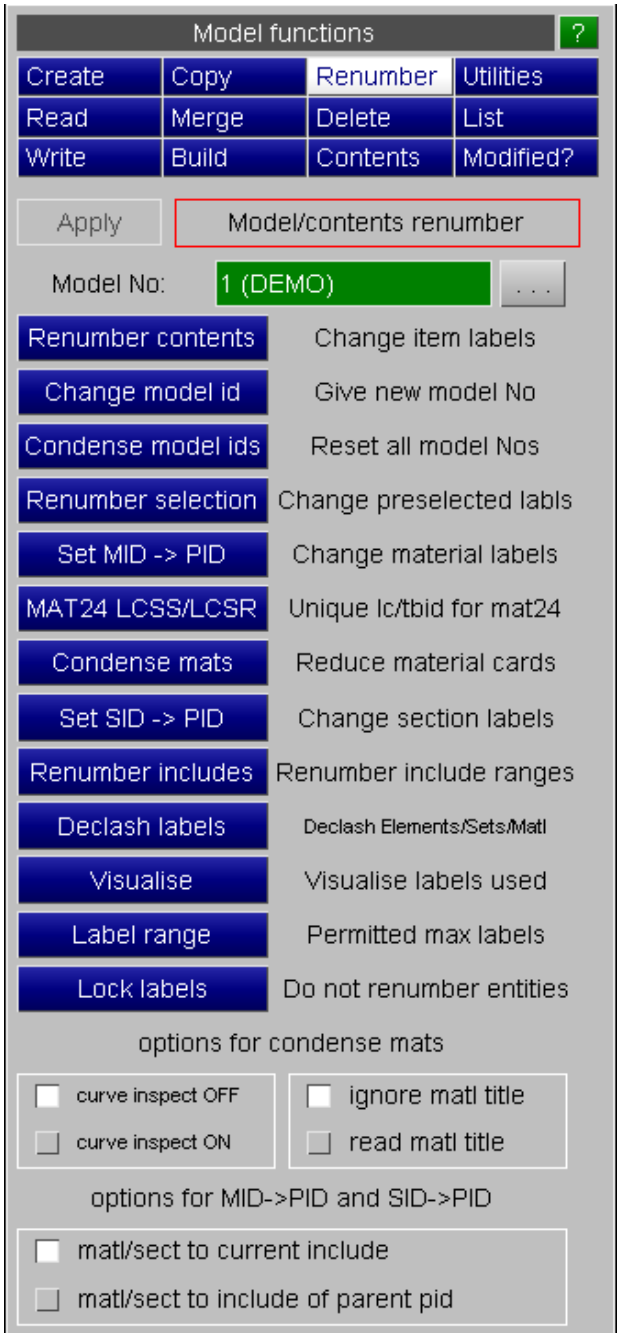
- Select a list of 1 or more existing models;
- Hit **APPLY**.
- You will be forced to confirm that you want to do this before they are actually deleted.

Once confirmed the complete model(s) will be removed from memory. **Deletion is irrevocable**: once it has been deleted a model cannot be restored.

The **MODEL** > **DELETE** command should only be used when you want to remove a ***complete model*** from memory. To delete a ***subset*** of a model (eg individual parts, elements, nodes, etc) you should use one of the following options:

Function **REMOVE**	**DELETE UNWANTED** **CLEANUP UNUSED**	Lets you select items of any type for deletion. "Cleans up" redundant items from a model.
Most **KEYWORD >** panels	**DELETE**	Lets you select items of that type for deletion
Generic Keyword Editor	**Row index > Delete**	Deletes item(s) on selected row(s)

3.7 MODEL > RENUMBER Renumbering models and/or their contents



This command is useful for renumbering models, or whole categories within models and, while you can use it to renumber individual items, it may be easier to edit them directly.

A brief description of each option is given below, follow the hyper-links for more information.

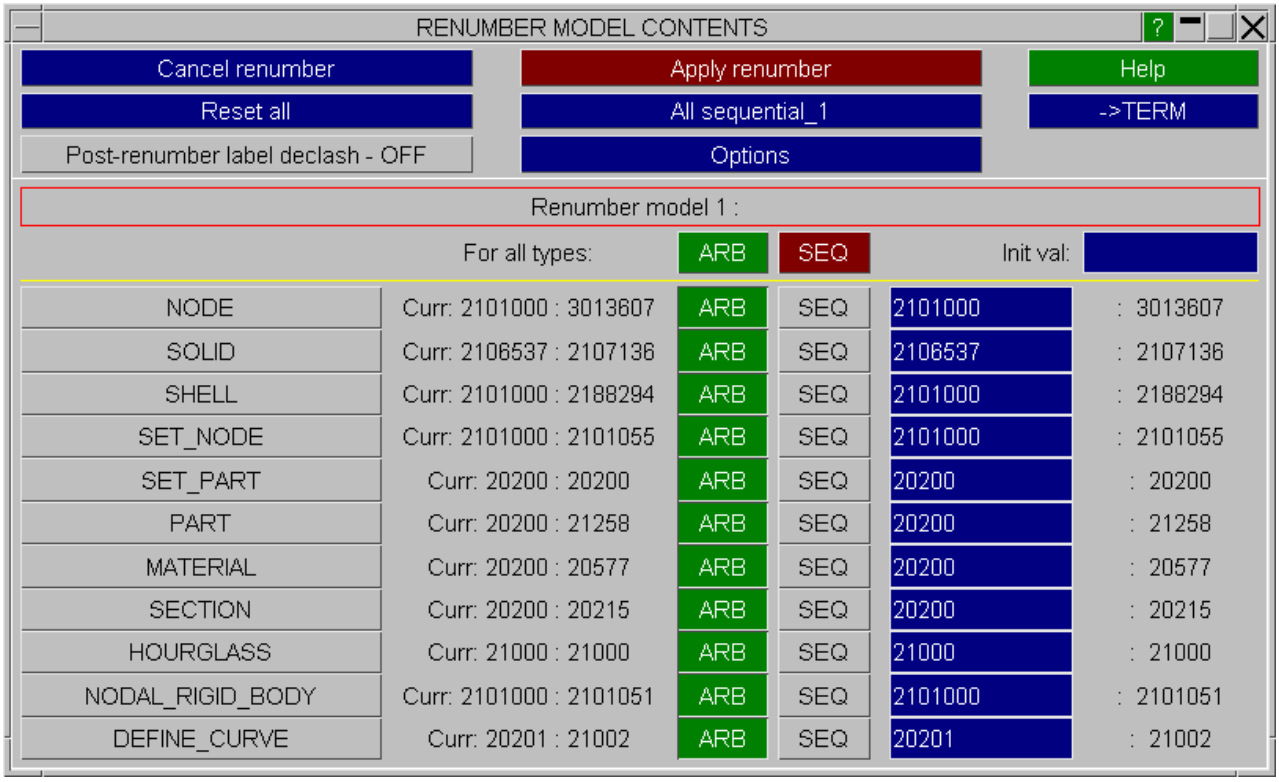
- **Renumber contents** lets you renumber the contents of a model.
- **Change model id** lets you renumber the label of a model itself (not its contents).
- **Condense model ids** renumbers <n> models from 1 to n. (Only the model ids themselves are renumbered, their contents are unchanged)
- **Renumber selection** renumber items selected via an object menu
- **Set MID->PID** establishes a `*MAT` card for every part using the same label. If more than one part uses the same material, then a copy of the material is created. This does not apply for `*PART_COMPOSITE` parts that can refer to > 1 material.
- **MAT24 LCSS/LCSR** sets a unique load curve or table id on the material (MAT24) cards that use the same curve or table. Copies are made of the curve / table in order to achieve this.
- **Condense mats** reverses the effect of **set MID->PID** by removing duplicated material cards. Material titles are ignored by default but the option may be switched to consider. By default curve labels are compared, but an option may be set to inspect the x y data instead of the label
- **Set SID->PID** establishes a `*SECTION` card for every part using same label. Again, if more than one part uses the same section, then a copy of the section card is created
- **Renumber includes** lets the user renumber ranges for general types and for types with explicitly relevant labels for the master file and for one or more include files.
- **Declash labels** will offer the user the option to declash all element, set and material labels
- **Visualise** displays the distribution of labels in a model by type in diagram form.
- **Label range** sets the limits on labels used when checking a model. Labels can lie in the traditional 8 digit "small" format range of 1 - 99,999,999; or can use 15 or 18 digit labels in "large" format output. See section 5.1.7 "Wide" keyword format and "large" labels for more infomation.
- **Lock label ranges** provides a means to lock a range of labels for one or more entity types lying in one or more includes against renumbering.

The **Options for MID->PID and SID->PID** determine where the newly created **`*MAT`** and **`*SECTION`** cards are placed if the model contains include files:

Matl/Sect to current include	Places all newly created definitions in the current include file, regardless of where their referring **`*PART`** cards occur.
Matl/Sect to include of parent PID	Places each newly created **`*MAT`** and **`*SECTION`** card in the same include file as its referring **`*PART`** card

3.7.1 RENUMBER CONTENTS Renumbering the item labels within a model

- Select a model.
- Press **RENUMBER CONTENTS** to get the renumbering panel



This figure shows a typical panel, but the actual appearance will depend upon the contents of your model.

An individual category (eg **NODE**) can be renumbered selectively by clicking on its category name button.

The model renumbering table has the following columns for each item category:

KEYWORD	Current range	ARB	SEQ	<Initial>	Highest
Each item category in your model, defined by its LS-DYNA keyword.	The unmodified lowest : highest labels for this category	**ARB**itrary or **SEQ**uential labels for this category		The first (lowest) label of the category	What the highest label will become.

To renumber all the items in a category (eg all NODES):

You can control two aspects of labelling for any item category:

1: The spacing between item labels: which may be **ARB**itrary or **SEQ**uential.

ARBitrary	Starts at the given initial value, but preserves the gaps between successive items. This is the default.
SEQuential	Starts at the given initial value, and numbers items sequentially upwards from that with no gaps

2: The initial value.

The default is whatever the input model contained, but you may change this to any positive integer. Successive values will be adjusted in an "arbitrary" or "sequential" fashion from this value. Notes that latent items will not be renumbered in this panel.

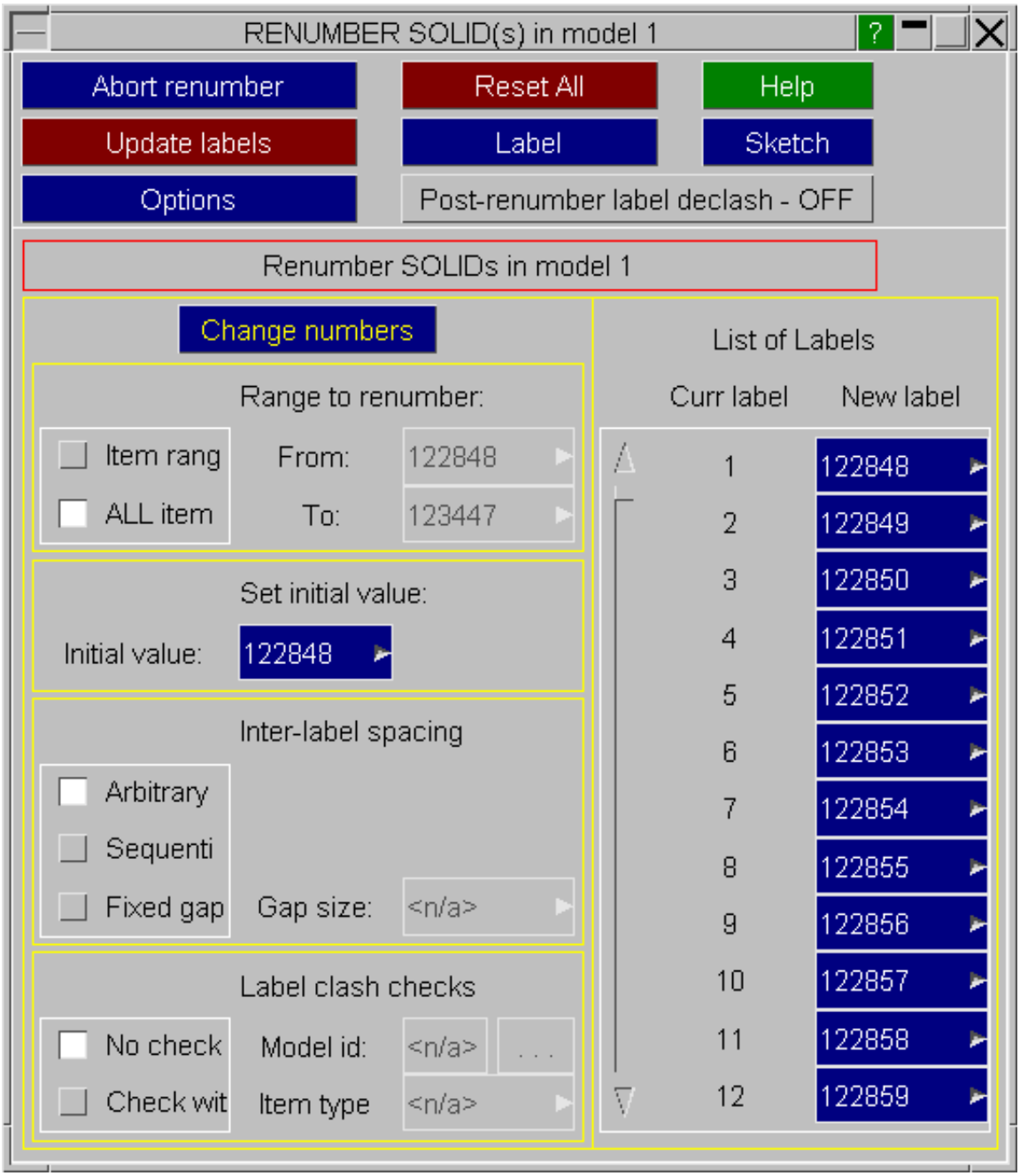
Each item category may be changed individually, or a complete column can be changed by using the "**For all types**" boxes at the head of the list.

Also there are some commonly used global options:

ALL_SEQUENTIAL_1	Renumbers everything sequentially starting from 1.
CANCEL_RENUMBERING	Exits renumbering without making any changes to the model.
RESET_ALL	Sets all the values of all boxes back to their original values.
POST-RENUMBER LABEL DECLASH	Post renumber declash of certain entity types. See section 3.7.9 - Label declash option for more information.

The changes made in this box are volatile.
They are only permanently saved in this model when **APPLY_RENUMBERING** is used.

To renumber an individual category selectively



By clicking on a keyword button in the left hand column of the renumber contents panel, eg the **SOLID** button, you can invoke the standard item renumbering panel for that category, as shown in the adjacent figure.

Range to renumber: Select the range of items to be processed.

Set initial value: Choose the initial value for this range

Inter-label spacing: Set the gaps between labels

Label clash checks: Check for and eliminate clashes between categories

Post-renumber label declash: Post renumber declash of certain entity types. See section 3.7.9 - Label declash option for more information.

This panel is designed to let you change the labels of individual items, or a range of items, in this category (here solid elements have been chosen).

The left half of the panel allows you to select a range, and update any or all of:

- Its initial value. Default is the current start of the range;
- The gaps between adjacent labels. The default is the current ("arbitrary") gaps.
- Any clashes between these and other items. You can choose both the item type and the model id to check against, for example this user might check against solids in another model.

Clash checking against the following generic categories is also available:

- **ELEMENT** Any element type. Useful where no clashes are permitted between element numbers of any type.
- **SET** Any set type. Useful where no clashes are permitted between sets of different types.

When you have set up those changes you wish to make **APPLY_CHANGES** to see their effect. You may alter settings and repeat this operation as often as you like since you are only operating on a "scratch" definition.

The slider box on the right side of the panel both shows the current status and also allows you to renumber items individually: just type in a new label, or use the popup options.

Note that latent items are displayed in the list, but will not be renumbered.

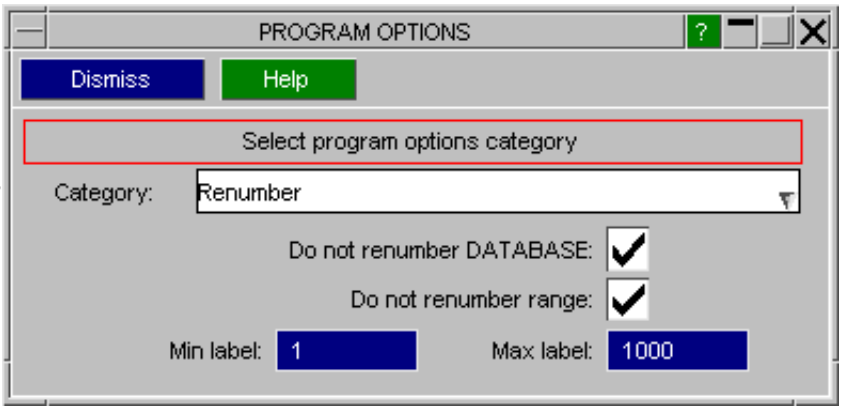
All operations within this panel operate on "scratch" labels.

To make these changes take effect in the permanent database you must use **UPDATE_LABELS**

Freezing entity labels during renumbering

Entity labels that lie within a user-specified range can be 'locked' during renumbering. This can be done by selecting the appropriate options in the Renumbering tab in the Program Options panel. The Renumbering options panel can be reached by clicking on the Options button either in the generic renumbering panel or in the category renumbering panel.

Likewise entity labels that are used by DATABASE_HISTORY cards can be 'locked' during renumbering.

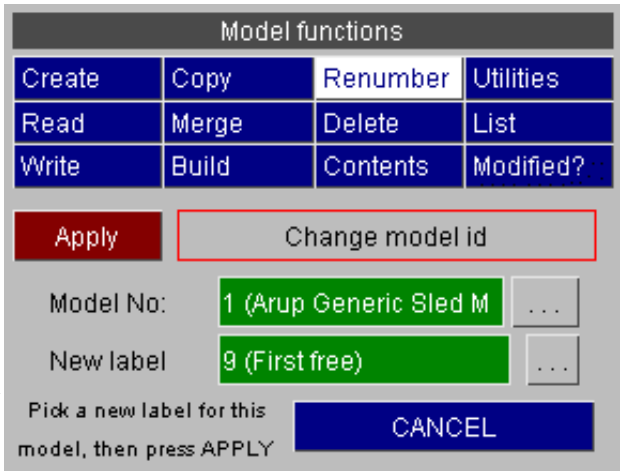


3.7.2 CHANGE MODEL ID

- Select the model number to change
- Select a new (unused) model label

Changing a model's id (number) does not affect its contents at all: its "id" is purely the number by which it is identified within PRIMER.

It is not possible to change a model id to that of another model which already exists, in order to achieve that you would first have to renumber or delete the target model.



3.7.3 CONDENSE MODEL IDS

Renumbers all current <n> model ids from M1 to Mn. (In effect from "arbitrary" model labelling to "sequential starting at 1".)

Condensing model ids has no effect on model contents.

3.7.4 RENUMBER SELECTION

Renumbering will only be applied to those items pre-selected through the object menu panel. Provisionally, the select items are to be renumbered **sequentially** starting at the defined start label ("START AT" option - the default) or to be **offse**t by a defined value ("OFFSET" option).

If, however, such renumbering would cause a clash of labels, some additional action must be taken.

Two options are offered for modifying labels of items **other** than the select items, to make sequential labels available for the select items:

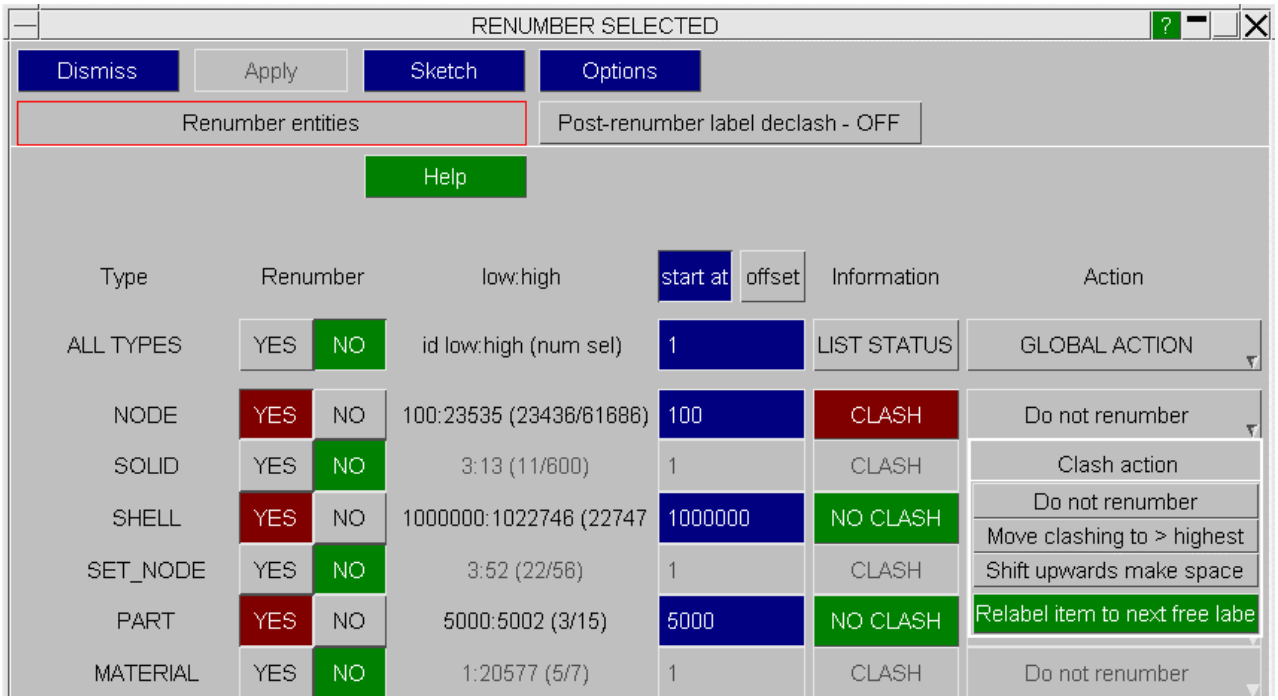
- Move any clashing labels to above the highest label for that type in the model
- apply an offset to all labels of the type, to shift all the labels clear of the sequential renumbering range

Alternately, rather than the select items being renumbered sequentially, one may:

- Renumber select items into the next available free label

In the case of "OFFSET" the only available clash fix is to move the offending labels to above the highest type label.

Latent items: Items which are referenced by a keyword but do not actually exist in a model are called latent. The renumber-selection function has been designed to avoid renumbering the labels of latent items. These labels are therefore reserved, and renumbering/clash fixing will always work around them. In the "OFFSET" case, if the required label for an item (current+offset) already belongs to a latent item, the item will not be renumbered.



The **SKETCH** function will sketch only those items that have been assigned for renumbering by activating "**YES**" button.

The **Post-renumber label declash** option allows the declash of certain entity types. See section 3.7.9 - Label declash option for more information.

3.7.5 CONDENSE MATS

Models that contain multiple definitions of the same material, for example a material card for each part cards, may be tidied up with the **CONDENSE MATS** function.

Duplicate materials are detected by matching type, title(if any) and then comparing each entry (using a test of 5 significant figures for floating point values other than zero). If they are found the reference PID->MAT is adjusted to make them redundant. The user is then prompted to apply the deletion function to remove them.

The following **Options for Condense Mats** apply.

Curve inspection ***Applies to fields LCSS and LCSR on MAT24 and MAT123 only***	**OFF**	Materials are only condensed together if the curves they refer to have the same labels.
	ON	If all other fields match, but the curve labels referred to are different, then the curves themselves are inspected. If the curve data points match, despite having different labels, then the materials are condensed.
Material titles *(Applies to all material types)*	**Ignored**	Materials are condensed regardless of any mismatch between their title lines.
	Read	Materials are only condensed if their titles match.

3.7.6 RENUMBER INCLUDES

- Select a model.
- Press **Renumber Includes** to get to the renumbering panel

New ranges can be specified for the master file, or for one or more include files using the appropriate text boxes. Ranges - for both general types and for nodes, elements, node sets and constrained nodal rigid bodies - can be generated for these files using the appropriate **Auto** button. An **Auto All** button is also available.

Upon selecting the **Apply** button, Primer evaluates the specified ranges to check whether renumbering would be necessary. If a given type has labels outside the specified range, Primer attempts to renumber those labels. Primer computes the number of labels of a particular type that exist outside the user-specified range. This is then compared

with the number of unused labels available in the range (including the range labels). Users are warned if the specified range is not large enough to accomodate all labels. In that case, Primer renumbers as many labels as it can within the specified range. It then renumbers the remaining labels starting from the highest ID for that particular type. A warning is also issued if user-specified ranges for two or more files overlap. Nodes, elements, node sets, and constrained nodal rigid bodies are renumbered, as are general types that always carry a label and any entities specified in the Detailed entity ranges panel (see below). General types that support an optional ID are only renumbered if they carry an explicit label.

Additional information about overlapping ranges and about out-of-range items can be obtained using popups attached to two sets of status buttons (**Range unique?** and **In range?**). The popup on **In range?** can also be used to renumber individual include files into range. The list of include files can be sorted using a popup that is available at the top of the window. There is also a provision to copy node/element/nset/nrbc ranges into general type ranges using the **Copy ranges** button.

The **Read csv** button can be used to import user-defined ranges in the form of a .csv file. Likewise, current ranges can be exported to a .csv file using the **Write csv** button.

The **Range unique** button permits specification of generic renumbering options.

The **Post-renumber label declash** option allows the declash of certain entity types. See section 3.7.9 - Label declash option for more information.

More control over the label ranges for specific entity types is available via the **Detailed** button. This button is coloured as follows:

- grey if no label ranges for specific entity types have been specified for that include/master file;
- green if label ranges have been specified for specific entity types and there are no entities out of that range for that include/master file;
- red if label ranges have been specified for specific entity types and there are entities out of that range for that include/master file.

Clicking the **Detailed** button opens the Detailed entity ranges panel for the corresponding include file (or master file):

Detailed entity ranges for suspension2.key (model 1: Car.key)

Dismiss Save and close

Entity name	In range?	Start	End
AIRBAG	INACTIVE	5000	5999
ALE	INACTIVE	5000	5999
BOUNDARY	INACTIVE	5000	5999
NODAL_RIGID_BODY	INACTIVE	5000000	5999999
SPOTWELD	INACTIVE	5000000	5999999
CONSTRAINED	INACTIVE	5000	5999
CONTACT	INACTIVE	5000	5999
DATABASE_CROSS_SECTION	YES	5000	5999
DEFINE_HEX_SPOTWELD_ASSEMBLY	INACTIVE	5000000	5999999
DEFINE	INACTIVE	5000	5999
BEAM	YES	5000000	5249999
SHELL	NO	5250000	5499999
SOLID	NO	5500000	5749999
TSHELL	YES	5750000	5999999
EOS	INACTIVE	5000	5999
HOURGLASS	INACTIVE	5000	5999
INITIAL	INACTIVE	5000	5999
INTEGRATION	INACTIVE	5000	5999
INTERFACE	INACTIVE	5000	5999
LOAD	INACTIVE	5000	5999
MATERIAL	INACTIVE	5000	5999
NODE	YES	5000000	5999999
PART	INACTIVE	5000	5999
RAIL	INACTIVE	5000	5999
RIGIDWALL	INACTIVE	5000	5999
SECTION	INACTIVE	5000	5999
SENSOR	INACTIVE	5000	5999
SET_BEAM	INACTIVE	5000	5999
SET_SHELL	INACTIVE	5000	5999
SET_SOLID	INACTIVE	5000	5999
SET_TSHELL	INACTIVE	5000	5999
SET_NODE	INACTIVE	5000000	5999999
SET_PART	INACTIVE	5000	5999

Label ranges for specific entity types can be specified by clicking the entity name to enable entry of start and end labels. As for the general renumbering panel, the popup on **In range?** can also be used to renumber entities in the selected include file into range. On clicking **Apply** conflicts between any defined entity ranges and existing 'nodes/elements/node sets/constrained nodal rigid bodies' label ranges and 'general' label ranges are detected. If there are conflicts an option is given to either adjust the 'nodes/elements/node sets/constrained nodal rigid bodies' and 'general' ranges or modify the detailed entity ranges.

This detailed renumbering of entities per include file is also available via the include tree, see Include Files.

3.7.6.1 Detailed renumbering of rigid patches

It is now possible to renumber rigid patches. A rigid patch is defined as a rigid part with is smaller than some reference length. PRIMER at the moment classifies rigid patches according to the diagonal of the part which is smaller than some reference length. This length can be set in Options -> Program Options -> Renumber -> Reference size for small rigid patch.

Thus, it is clear that rigid patches are a subset of parts. Thus, when the rigid patch option is active, the detailed entity panel makes sure that the entities that can be renumbered through the PART tab, excludes any entities that are classified as rigid patches. PRIMER reverts to normal functioning (all parts can be renumbered through the PART tab) when the RIGID_PATCH option is inactive.

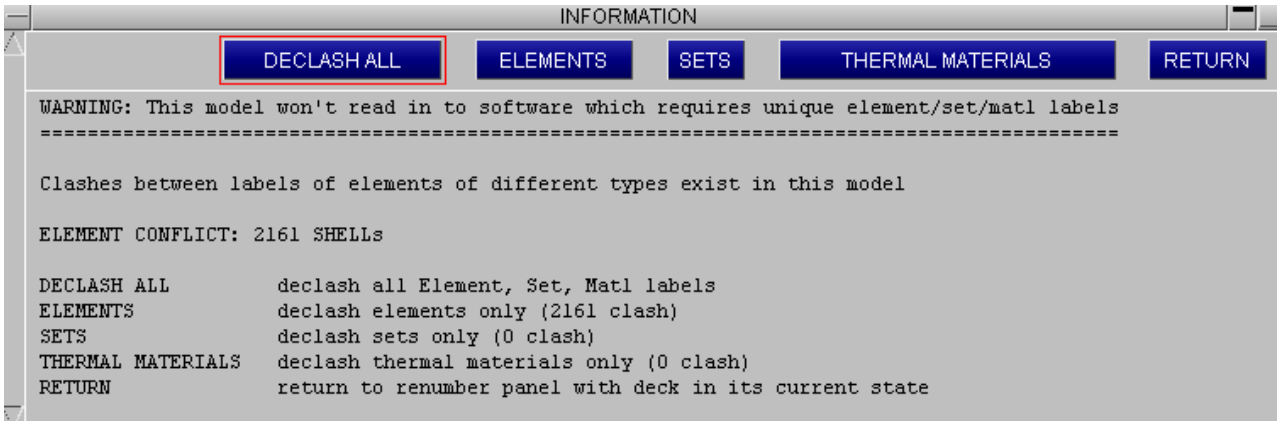
3.7.7 MAT24 LCSS/LCSR

This function will ensure that each material card of type 24/123 will have a unique table with unique curves. This will allow stochastic variation of material properties.

3.7.8 Declash Labels

This function allows declash of element labels (so shells don't clash with solids, etc), set labels (so node sets don't clash with shell sets, etc) and material labels (so structural materials don't clash with thermal materials). You may want to do this to avoid problems when reading your model into other pre-processors that do not allow label clashes across element types/set types/material types.

If any clashes are found the fixing panel will be displayed. The user may select to **DECLASH ALL** or to address the individual types. Note this also considers clashes between elements and entity types other pre-processors may consider to be elements (an example of this is *CONSTRAINED_NODAL_RIGID_BODY).



3.7.9 Label declash option

The **Post-renumber label declash** option on renumbering panels should be turned on should you wish to avoid label clashes that could cause problems when reading the model into other pre-processors. With this option on, after renumbering PRIMER will declash element labels (so shells don't clash with solids, etc), set labels (so node sets don't clash with shell sets, etc) and material labels (so structural materials don't clash with thermal materials). Note this also considers clashes between elements and entity types other pre-processors may consider to be elements (an example of this is *CONSTRAINED_NODAL_RIGID_BODY). When turning this option on/off for the first time you will get the option to set a preference to always have this option on. The option can also be turned on through **Options->Program Options->Renumber->Avoid label clash**. Note that this option applies in contexts other than renumbering. Any action where elements are created and assigned labels will avoid clashes with labels of elements

of different types (the same applies for labels between different *SET types and different *MAT types). So, by turning the option on in this panel, or the options panel, the option will be on for all renumbering panels and will be used throughout PRIMER when creating entities.

3.7.10 Visualise

The **Visualise** feature allows you to view label distribution within your model in a graphical form. It allows you to identify ranges of labels that are currently used, and also ranges of labels that are free. The panel that opens when clicking on **Visualise** will look like this:

The entity types currently in the model are displayed as rows. Labels are shown along the top of the graphical area. The black lines/blocks represent label ranges that are currently used in the model. The white areas represent label ranges that are currently free. When moving the cursor over the graphical area, the black/white areas will be highlighted with a red border. The feedback section at the top of the panel will give you information about the highlighted area. In the following example, and area on the SOLID entity row highlighted shows that labels between 1329133 & 2000999 are not currently used for solid elements.

There are various options/actions that can be carried out on this panel:

Changing the label range shown

By default, the range of labels shown spans from the minimum label of all types in the model until the maximum labels of all types in the model. This can be changed by typing in a new range in the input boxes in the top left of the panel.

Alternatively it is possible to zoom by holding the SHIFT or CTRL keyboard key and right mouse button hold and drag over the graphical area. Similarly, it is possible to pan the display by holding the SHIFT or CTRL keyboard key and middle mouse button hold and drag over the graphical area. To return to the default view of seeing the whole range click on the **AC** button on the panel, or click the keyboard shortcut "a" while your cursor is over the panel.

Writing information from the panel

The label range information on the panel can be written to a text file in CSV format by clicking on the **Write** button. The format is as follows:

entity type 1, start of used range 1, end of used range 1, start of used range 2, end of used range 2, ...
entity type 2, start of used range 1, end of used range 1, start of used range 2, end of used range 2, ...

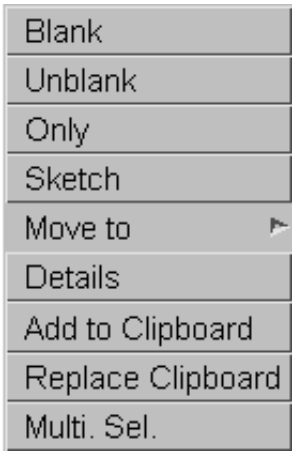
So a file may look something like this:

All, 1, 1000, 2000, 2346, 5000, 5678, 8001, 8790, ...
NODE, 1, 700, 2345, 5050, 5106, 8100, 8102, ...
SOLID, 1, 60, 2070, 2074
...

Interactive options on currently used label ranges

Various interactive operations can be performed on currently used label ranges (black blocks) on the panel. A left mouse click hold and drag operation over a black block will give you the option to drag to renumber the entities represented by the block. When renumbering in this way a feedback display will appear telling you what the outcome will be. If the feedback display is green, the new location for the block does not clash with any other labels of that type in the model. If there is a clash, the feedback will be red. When you release the left mouse button with the block on the new location PRIMER will ask you to confirm the renumbering.

A right mouse click on a black (used) block will give the following options:



Blank - Blank the entities represented in the used block.
Unblank - Unblank the entities represented in the used block.
Only - Only the entities represented in the used block.
Sketch - Sketch the entities represented in the used block.

Note for the above the visual panel is quite big and may cover a large percentage of the graphics window, so it may be difficult to see the result of the above action. In these situations it is useful to use the keyboard shortcut "i" (iconise) which will iconise open panels allowing you to see the graphics window. Pressing "i" again will reopen the panels.

Move to - Type in a starting label to renumber the entities in the used block to.
Details - Gives some details on the contents of the used block.
Add to Clipboard - Add the entities represented by the used block to the clipboard.
Replace Clipboard - Replace the entities currently on the clipboard with the entities represented by the used block.
Multi. Sel. - Set the panel in multiple select mode which allows you to select multiple blocks of used labels for an operation. While in this mode, you will be locked into operating on one row. To select multiple blocks, drag a selection box around the appropriate used blocks using the left mouse button:

In the above image the selected area is highlighted as a blue box around label blocks on the SOLID row. A left mouse click drag renumber or a right mouse click to open the above options are now available to be applied to multiple selected blocks rather than just one. To quit out of multiple selection mode, click on **Cancel Multi** at the top of the panel.

Interactive options on currently free label ranges

When right mouse clicking on a currently free label range (white) there is an option to **Renumber to here**. When selecting this option, you can select entities of of the appropriate type to renumber into the selected range.

Entity mode and Include mode

By default the panel will open up in **Entity** mode, which means all the entity types currently in the model will each have a row in the table. The display can be changed to **Include** mode in the top left hand corner. In **Include** mode instead of one row per entity type you get one row per include:

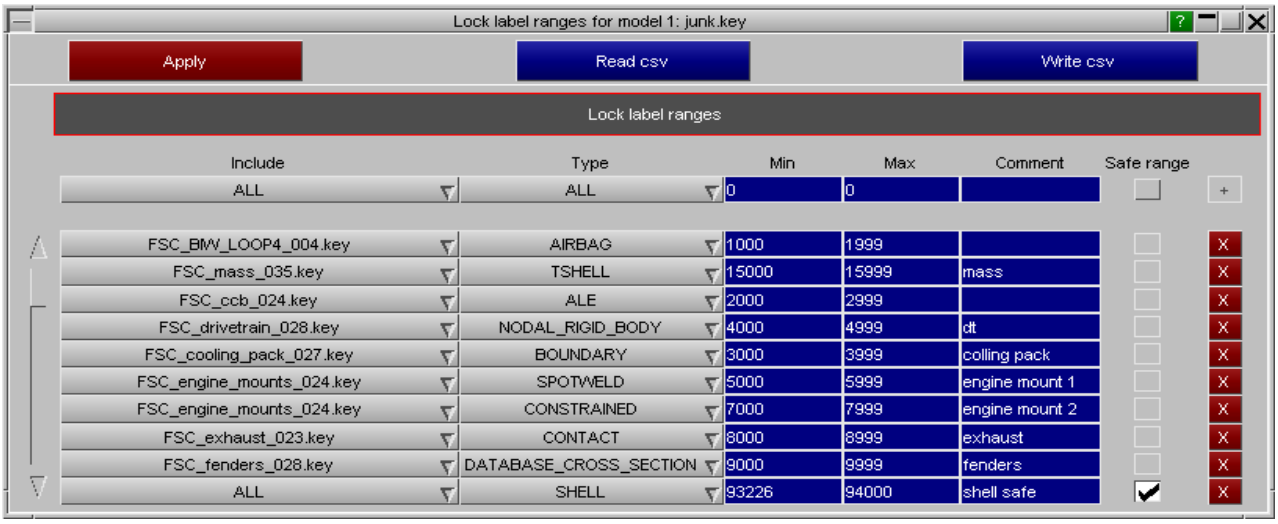
The used blocks (black blocks) in **Include** mode represent used labels of any entity type within the include file specified on the row. All the same operations that are available in **Entity** mode are also available in **Include** mode (drag renumbering, right click **Blank** etc.).

There are some extra features available in **Include** mode. If you have include numbering ranges set, they are displayed on the panel. For more information on include ranges, see section 3.7.6. Two include ranges can be set for each include file, one for entities where you may many of them in the model (nodes, elements, nodal rigid bodies etc) and one for all

other entities. The node/elem range is shown in transparent blue and the general range is shown in transparent green. Either set of ranges can be turned on/off in the top left of the panel. It is possible to modify the include range within the panel by right mouse click dragging when hovering over the start or end of the range block. When you hover over the ends the range is highlighted purple which will indicate you are dragging the range rather than dragging the labels themselves.

3.7.11 Lock label ranges

The **'Lock label ranges'** feature allows you to lock a range of labels for one or all entity types in one or all include files against renumbering. In other words, attempts to renumber locked labels will be blocked. Those locked ranges that apply to all includes may also be designated as **Safe ranges**. Safe ranges are protected ranges that other entities may not be renumbered into.



Information regarding these locked ranges are stored as special comments in includes and the master model. They will, therefore, survive model keyout/keyin. These ranges may also be read or written from/to csv files to facilitate easy transfer between models.

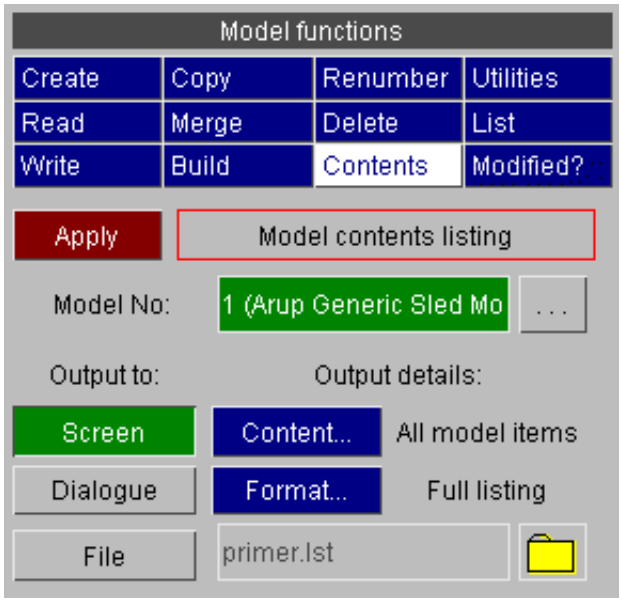
3.8 Model Contents

There are two ways of listing the contents of a model:

MODEL > CONTENTS	General listing of all model contents to screen and/or dialogue area and/or file
UTILITIES > WRITE_SUMMARY_FILE	More specialised list of Part, Material, Contact and Element properties

3.8.1 MODEL > Contents

It is possible to summarise the contents of a model to the screen and/or to file with this option.



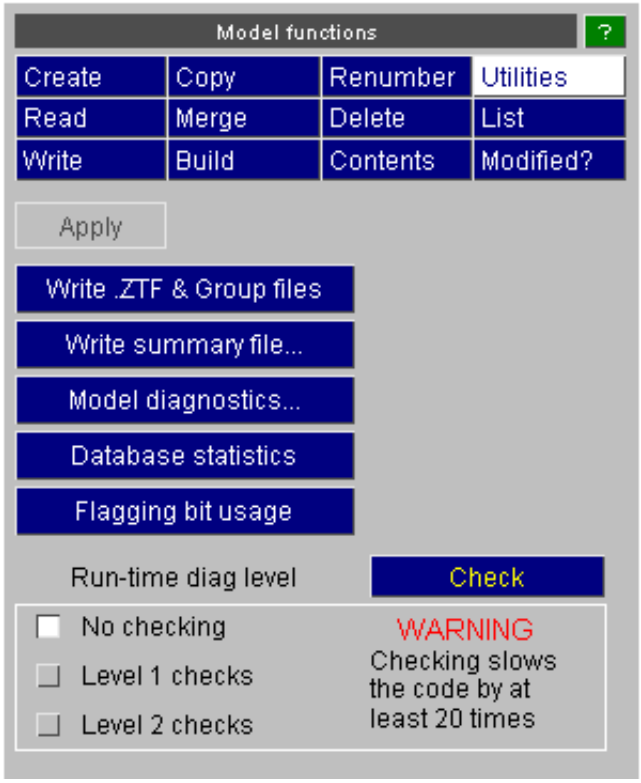
You can select any or all of the following "**Output to:**" locations:

- **SCREEN** A paged screen window.
- **DIALOGUE** The screen dialogue box
- **FILE** A disk file.

Output details: control what is written:

- **CONTENT...** May be all items (default), or just a listing of parts and contacts.
- **FORMAT...** May be a full listing, or a summary of the number of items of each type

3.8.2 UTILITIES



The **UTILITIES** menu allows you to:

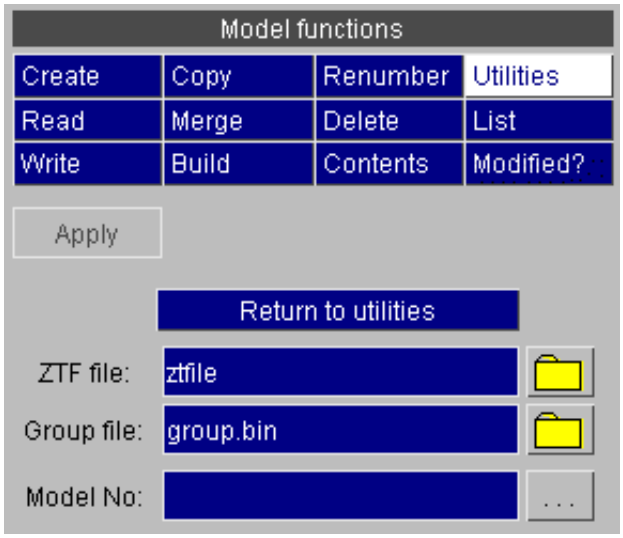
Write a summary file of the model

Write ZTF and group files

List flagging bit usage

The diagnostics and statistics options are described briefly below, but they are intended for use by Oasys Ltd developers only.

Writing ZTF and group files



The ZTF and group files are used by D3PLOT in order to visualise during post-processing information that is not available in the normal LS-DYNA results files. Only PRIMER can write these files.

The **ZTF file** contains information about nodal restraints, spotwelds, nodes on contacts, parts, sections, etc; which enables D3PLOT to draw and process extra information. This information is not available in the ptf, ctf or xtf files.

The binary **Group file** (.bin) is now superseded in D3PLOT, although it can still read it. Its output is preserved here for compatibility with older versions.

Writing a ZTF file

This may be carried out in any of three ways in PRIMER:

1. As shown in the panel here: fill in a file name (`<jobname>.ztf` is preferred) and **Apply** it.

2. If ZTF Output is selected in the **Model > Write** panel file `<jobname>.ztf` will be generated at the same time as the output .key file is written. See section 3.3.

3. If PRIMER is run in batch mode with the following command line arguments: (see Appendix XIII)

   `-d=batch -ztf=<input filename>`

   . This method is used by the Shell to generate ZTF files automatically after an analysis has been run.

Writing a binary Group file (*deprecated*).

This file type was read by D3PLOT 9.1 and earlier, providing a method of exporting groups in PRIMER for use in

post-processing. It had the disadvantage that, being a binary file, it was not only large, opaque and uneditable, but it was also "tightly" locked to the given analysis and could not be used with a different one.

Therefore from release 9.2 onwards an alternative way of exporting group information as an ASCII file via the **GROUPS->EXPORT** function was introduced: (see section 6.20GROUPS)

- This is simply a copy of the **`*GROUP`** keywords (after **`*END`**) used by PRIMER to encode group information in the keyword output file.
- It format is compact, being typically less than 100 lines line.
- It is in a "human friendly" format and, being an ASCII file, it is manually editable if required.
- It is also flexible: any mis-match between the group file contents and the file being processed is simply ignored, making it usable over a wide range of broadly similar analyses.

ASCII groups file may also be read into PRIMER using **GROUPS->IMPORT** .

Although D3PLOT releases from 9.2 onwards will continue to read the binary groups file it is strongly recommended that you use the ASCII version instead. This feature may be withdrawm in future releases of PRIMER

WRITE_SUMMARY_FILE...

This command may also be used to generate a summary of the model contents giving more detail than **MODEL** > **LIST**.

It writes a file containing a part summary, material summary, contact summary, element summary and element quality summary. The default file name is **model_summary** but this can be changed at the prompt before being written out.

The part summary lists model contents by:
- Parts, in order of part ID, giving a break-down of material ID, material type, section type, gauge (if shell elements), part mass, timestep added mass and title for each part and part inertia.

The mass of each part is calculated as it by LS-Dyna on model initialisation. If nodes on deformable elements are attached to rigid parts, mass is lost by the deformable and gained by the rigid (or discarded if rigid is part-inertia). Lumped masses on rigid bodies (non part-inertia) are included in their part mass, whereas those on deformable bodies are added to the lumped mass total. If a rigid part is merged onto another, the slave loses its mass and this is added to the master (or discarded if master is part-inertia).

The mass contributions from parts, lumped masses on deformable, part inertias and nodal rigid bodies, the total model mass, and (if any) the timestep mass scaling as %age of model mass are summarised at the end of the part listing. Model mass is also written to the dialogue box.

The material summary contains a listing of:
- The commonly used materials in each model with a few useful details about each material e.g. E, and for each *MAT_ELASTIC.
- Note that not all material parameters are given for each material and not all material models are supported by the summary file. A list of materials which exist in the model for which no details are given is included at the end of the material summary.

The contact summary contains a list of:
- title
- contact type
- slave and master type

The element summary lists:
- parts by element type referenced, giving part ID, material ID, section ID, hourglass ID, thermal material ID and title.

The element quality summary lists:
- part ID
- percentage of elements failing one or more quality criteria
- minimum element length - worst value and the percentage of elements failing this criterion
- aspect ratio - worst value and the percentage of elements failing this criterion
- skew - worst value and the percentage of elements failing this criterion
- warpage - worst value and the percentage of elements failing this criterion
- minimum angle - worst value and the percentage of elements failing this criterion
- maximum angle - worst value and the percentage of elements failing this criterion

Model Diagnostics

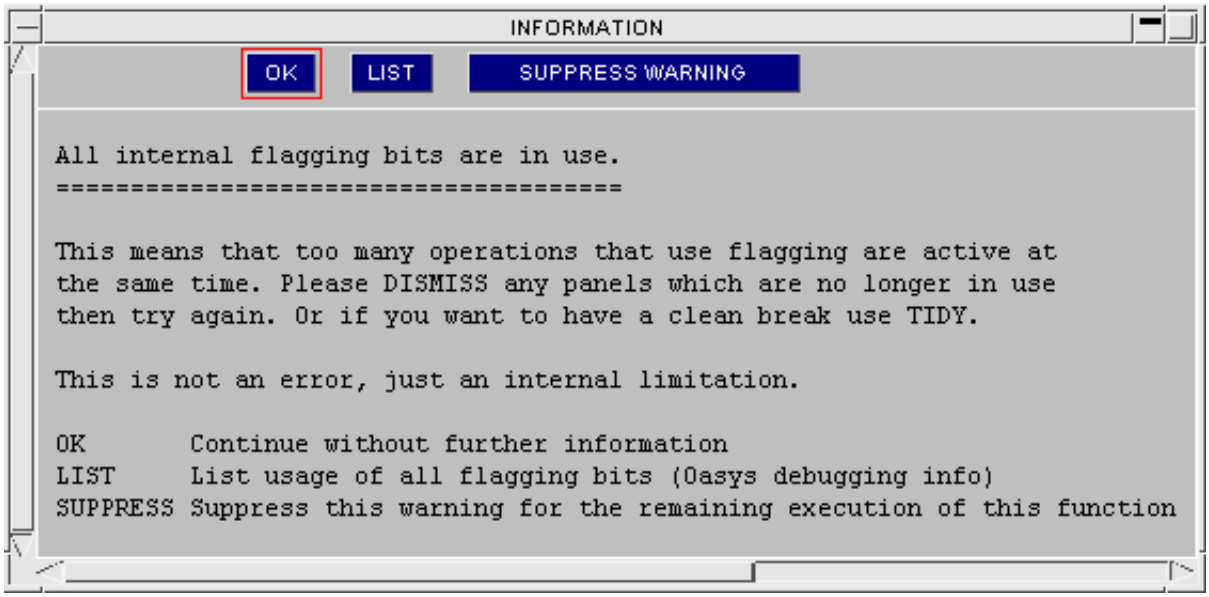
This option lists a summary of the main internal entity tables to the controlling terminal <stdout>. It is intended to help the developers and will not be of any help to ordinary users.

Database Statistics

This option is also intended for the developers. It writes to the controlling terminal <stdout> a list of memory allocation and usage by each internal data type, together with a summary of total usage. It may, with some guidance from Oasys Ltd, help with the debugging of memory-related problems.

Flagging bit usage

PRIMER makes heavy internal use of "flagging bits". These are literally bits in an internal word used when selecting items for processing and, unfortunately, they are a finite resource which can at times become exhausted. If you try to perform too many functions at the same time you may see the following message:



LIST will give a table of flagging bit usage by model, allowing you to work out what you need to close down in order to free some flagging bits for the wanted operation.

In order to avoid having to provoke this message in order to find out which bits are being used where, this command will list them on demand.

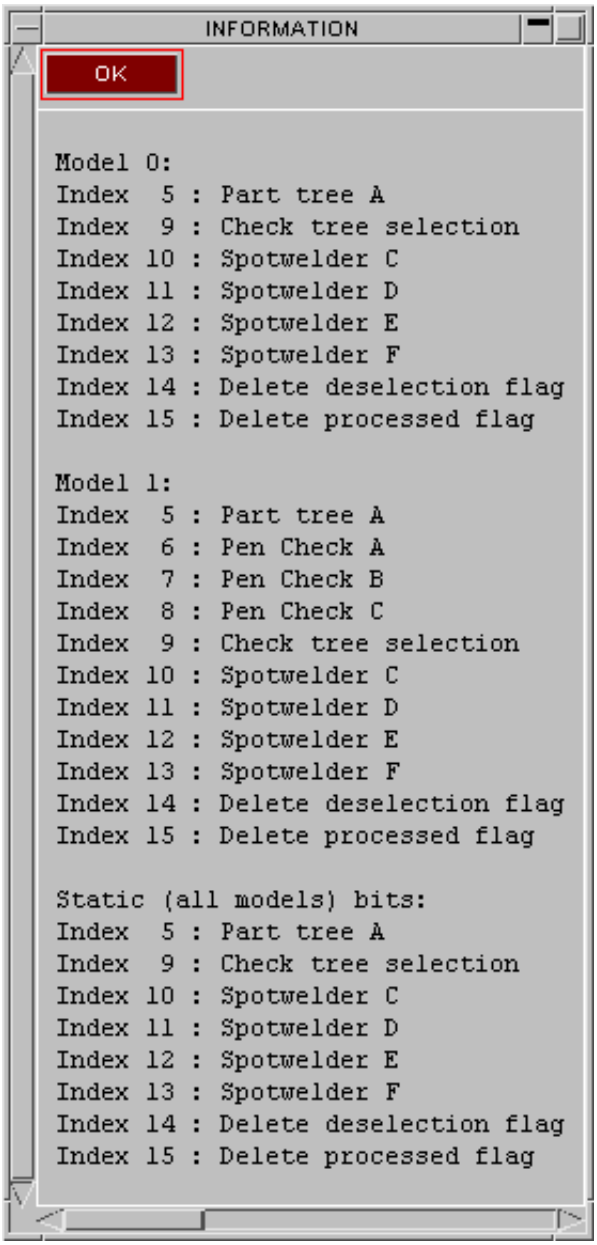
The example listing here shows that:

- The Part tree is using one bit. This is always the case and cannot be changed.
- The Contact penetration checker is using 3 bits in model #1.
- The Model Check panel is using 1 bit in model 1.
- The spotwelder in the Connections panel is using 4 bits in all models.
- The deletion panel is using 2 bits in all models.

PRIMER release 9.3 has a total of 19 dynamically allocatable flagging bits, numbered 5 to 23. As this example demonstrates some operations (eg DELETE) allocate bits in all models, whereas others (eg contact penetration check) are specific to a particular model.

Bit exhaustion can occur even if not all bits in a particular model are in use, since "all model" operations require that bits have the same number in each model.

Therefore when shutting down panels to free flagging bits it is usually most effective to close those which allocate "all model" bits.



Run-time Diagnostics.

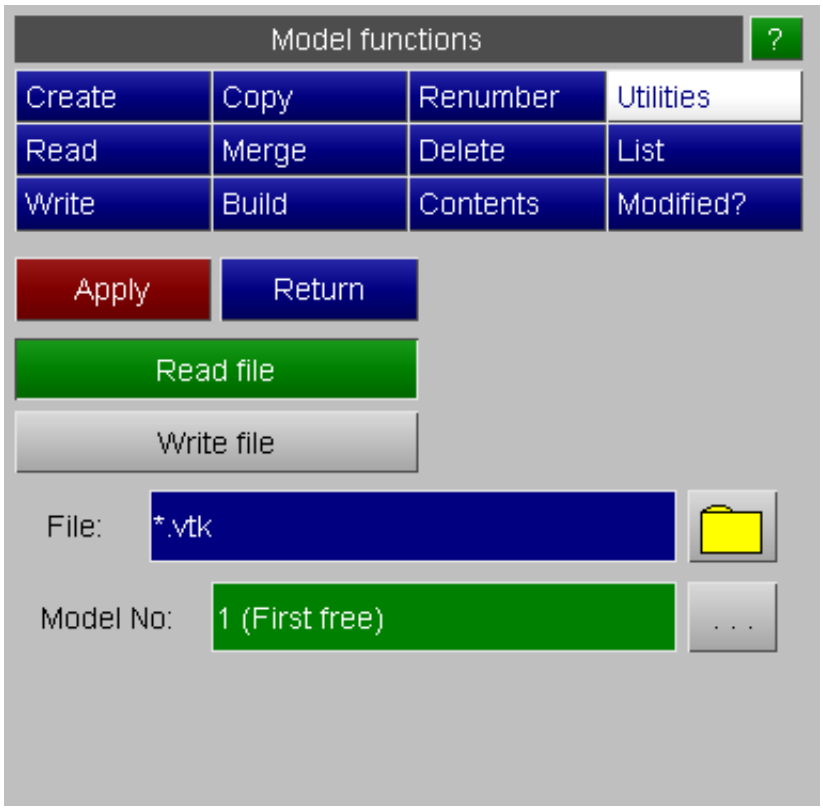
As with the other diagnostic options above the run-time diagnostic level is intended for use by the developers. It monitors internal memory allocation to detect "memory leaks". This would never normally be switched on by a user.

The **Check** option lists to the controlling terminal a snapshot of pointer allocation and integrity.

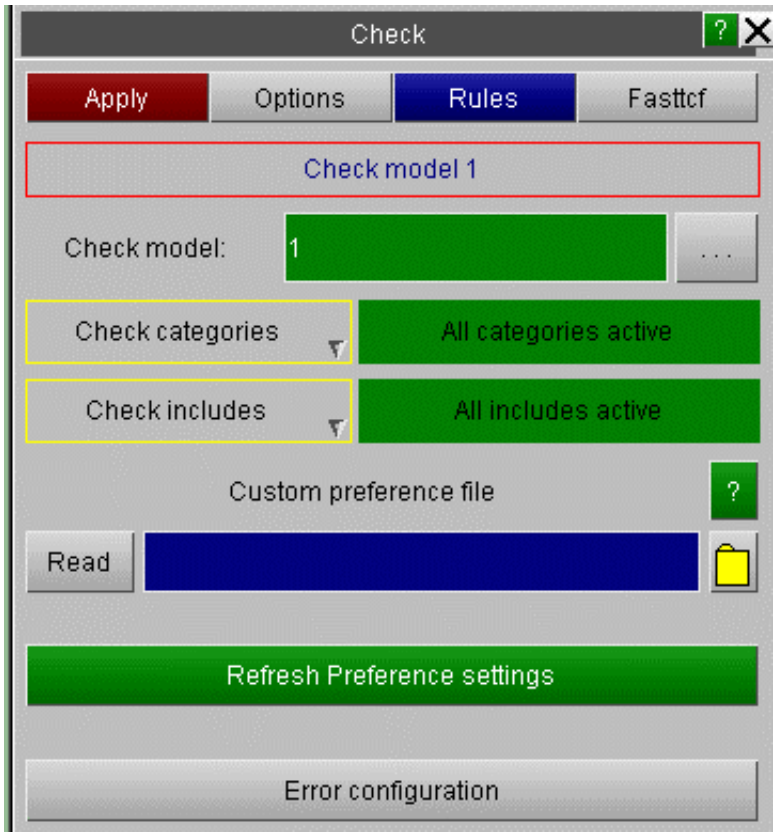
Writing a vtk file.

A Visualisation Toolkit or vtk file can be read or written via the Utilities menu. Enter a file name and choose either **Read file** or **Write file** follwed by **Apply**.

These files contain a list of node data and topology describing how the nodes are connected. Currently only SOLID, BEAM, SHELL and THICK SHELL entity types are supported. Vtk files offer an alternative method of viewing models and can be read by other programs such as ParaView.


Model functions
?
Create
Copy
Renumber
Utilities
Read
Merge
Delete
List
Write
Build
Contents
Modified?
Apply
Return
Read file
Write file
File:
*.vtk
Model No:
1 (First free)
. . .

3.9 MODEL > CHECK



PRIMER offers a powerful checking feature to validate models before the user submits the model in LS-DYNA. Over 3000 individual checks are perfomed to check a whole model for grammatical, contextual and other errors. In addition to flagging fatal errors (invalid fields, cards and so on), many of these checks will also improve the quality of the model analysis. PRIMER separates the check results into 2 types: errors and warnings. **Errors** usually indicate a model that will not initialise within LS-DYNA or will be of poor quality, **warnings** are indications of bad modelling practise and/or integrity. The results of the check can be viewed in a table format or navigated using an intuitive tree structured viewer similar to the part table.

In addition, a FASTTCF file can also be checked (for details on this type of file see the T/HIS manual - section 7). The **CHECK** option can be found in the tools menu.

Checking a Model

- Set up any checking **OPTIONS...** required
- Select a model
- Press **APPLY** to run the checker (more than 3000 checks are applied).
- Press **RULES** to run a set of user defined or custom checks (below) .

The whole model is checked, which may take a little time, and a summary of any errors found is given, together with any that can be "Auto-Fixed". A summary table and error tree viewer can then be utilised to navigate through the error categories and sub-categories.

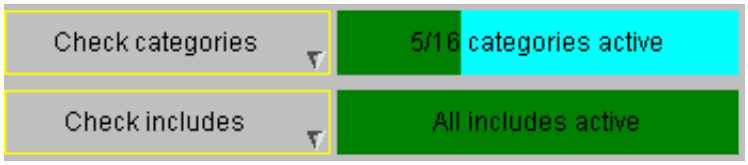
A combination of **CHECK** and **REMOVE** > **CLEANUP_UNUSED** will go a long way to eliminating most errors in models.

Most **KEYWORD >** panels and **CREATE/EDIT** panels include a **CHECK** option for checking specific categories or items, which is the way to check individual items. The checking routines are the same in all cases.

FASTTCF CHECK:

- Press **CHECK FASTTCF FILE** to continue to the FASTTCF check menu.

Reduced Checking: categories and includes



Primer's default is to apply checking to every item in the model which can be checked. For very large models, one has the ability to limit checking to a subset of categories (for example, excluding any checks on CONTACTs and CONNECTIONs) and/or to limit the include files that are checked (for example, exclude the one that contains all material definitions). The selection is made via the drop-down and the fact that reduced checking is being applied will be highlighted by the appearance of cyan backround on the button.

The effect of excluding a sub-set from checking may not have the expected consequences for complex checks where different categories interact e.g. PART and MATERIAL. Similarly exclusion of an include file means, on the face of it, that no item in that include will be checked. However, if a check on an item (e.g. PART) refers to an sub-item (e.g.MATL) it will make no difference if the include of the secondary item is excluded as the filter is only applied at the top level (in this case, before applying check of a particular PART).

Saving the on/off status of Model Check categories

From V12 onwards the on/off status of checking of each model category can be saved in the oa_pref file. The syntax of the preference is

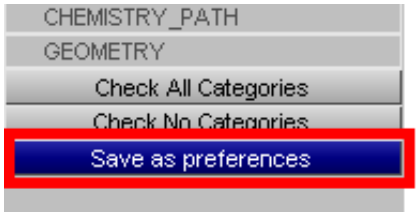
`primer*model_check_`*`category`***`: on | off`**

for example:

`primer*model_check_airbag: on` Airbags will be checked

`primer*model_check_dummy: off` Dummies will not be checked

To make it easier to save this status there is a **Save as preferences** button in the category check popup which will automatically save this preference for all categories present in your model.

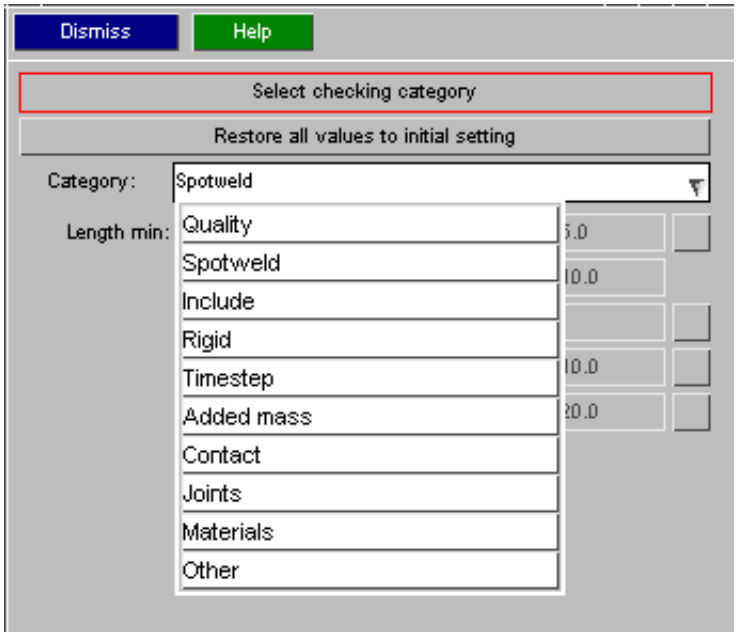


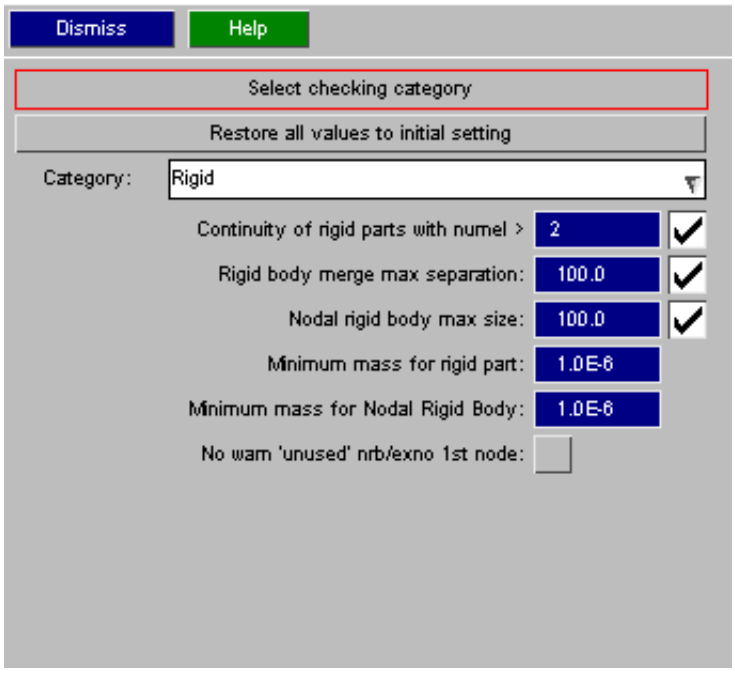
Users are recommended to always apply a complete model check with no catagories/includes excluded before completion of their work.

3.9.1 OPTIONS... Setting model check options

The check option panel is now divided into pages for ease of use.

Use the popup to go to the page you want.

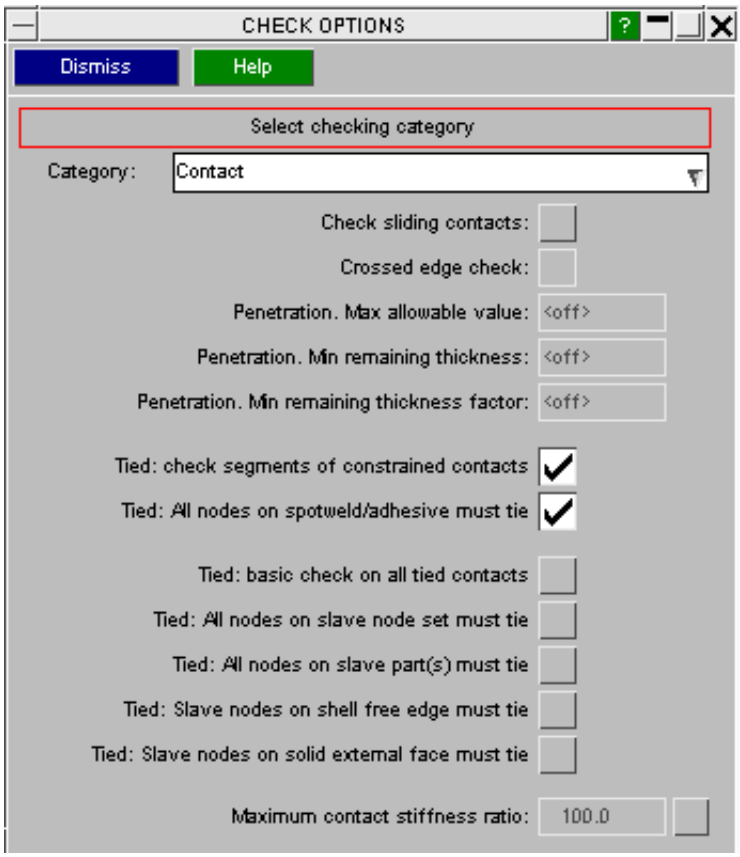




These are the options that control checks related to rigid entities

Options for which zero is a valid entry will have an ON/OFF tick box to control whether the option is active.

For others entry of zero (or blank) de-activates the option.



These are the options that control checks related to contacts

Sliding contact - by default the checks are **off**. If active, crossed edge checking is done. Additionally, penetrations may be reported if they fail one of 3 thresh-hold criteria.

- the max allowable value of penetration (typically 0.5mm)
- the minimum remaining unpenetrated segment thickness (typically 0.5mm)
- criterion B expressed as ratio of average segment thickness (typically 0.7)

Tied contact - by default constrained contacts will be checked for clashes with one another and connection contacts

will be checked for any untied nodes. In the general case, there is no rule for which of slave nodes are supposed to tie so a variety of options (all off by default) have been added.

- basic check - will give contact error if no node is tied
- slave node set - error if any contact with slave side defined by node set does not tie all the nodes
- slave part -error if contact with slave set defined by part/part-set or shell-set does not tie all the nodes
- free shell edge - error if slave nodes on free edges of shell do not tie
- solid face - error if any external nodes on solid do not tie

The majority of PRIMER's checks are automatic and built into the software. These flag up fatal errors that would stop the model initialising in LS-DYNA and errors which could mean that the model is unreliable or may fail to run to completion. However, PRIMER enables the user to turn off/on many of its model integrity checks to allow the user to set a custom level of model warnings. The options available to control the level of checking have expanded considerably in recent versions of PRIMER. Most of these options can have an initial state set using the **oa_pref** file. For example:

Contact Penetration Checks

Controls whether contacts are checked for penetrations and crossed edges.

Set the following line in the oa_pref file:

```
primer*contact_penetration_checks: ON (default OFF)
```

The following is a summary of the check options in PRIMER. Most options can also be set as user preferences.

LS-DYNA version number (LS970, etc)

- Because earlier versions of LS-DYNA imposed some restrictions on the valid label range for certain items (for example a maximum of 99999 load curves) this version number can sometimes be significant when checking.
- The version number is the same as that used in the keyword output section.

Warnings about output fields suppressed due to the current LS-DYNA version number.

- From release 15 onwards PRIMER will add to the Model Check output the warning messages that would be generated during keyout were the current LS-DYNA version to be used. This means data fields on cards that are not valid for the current LS-DYNA version, and also whole keywords that are not supported.
- By default these messages are delivered as warnings, but they can be changed to become errors under the Options, Other category. The check can also be turned off altogether.
- These warnings are only available within Model Check, and not as part of "free standing" check operations elsewhere in PRIMER. This ie because generating them requires a special (silent) peudo-keyout operation from which the warnings are collected, and this is only performed within Model Check.

Element quality checks (ON/OFF/PREF) - by default each check set to PREF value (or OFF if this is unset)

- A master control for whether element quality checks are performed
- A single check may be set on or of by setting a switch and giving a value (if needed), for example: **primer*element_length_check**: ON and **primer*element_min_length**: 6.0
- by default element checks are at their individual settings. However, **primer*element_quality_checks**: ON (or OFF) activates/de-activates all checks and will over-ride any individual settings

Spotweld checks - by default these are ON

- The Length min/max refer to lengths of individual beams within a spotweld
- The minimum distance between welds and maximum number of panels to attempt to weld can also be set here

The other checks are self-explanatory, providing the user with several mass checks and model quality checks. A few worth noting are:

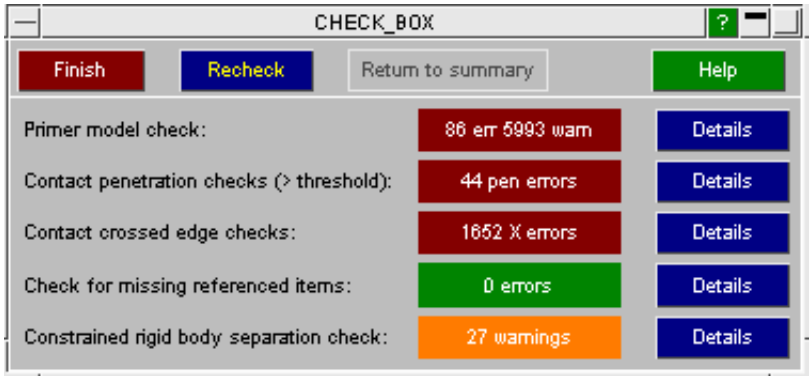
- **max separation for rigid body merge** checks the distance between merges to detect a merge that has accidentally been made across the model
- **nodal rigid body max size** warns if nodal rigid bodies as measured by the diagonal of an enclosing box exceed defined size
- **check mat24 curves to strain (FAIL=0)** it is an error for MAT24 load curves to cross the x axis before failure strain is reached, similarly the curves in a table should not cross over one another below such strain. By default (FAIL=0) the failure strain is very large (1e21) so the curve will get extrapolated and two curves may converge in theory. In reality extrapolating strain to this amount is unrealistic, users may prefer to limit the check to a realistic strain value (e.g. 100.0).
- **check for duplicated elems in same part** - elements which share the same nodes & part, also called overlap check
- **part quality** will warn if more thah a user defined %age of elements of a part have failed the user defined quality checks
- **minimum mass for rigid part** the default is <auto> which means Primer attempts to calculate the minimum mass of the rigid part required to keep the adjoining deformable elements stable under conditions where model has added mass. Such mass is effectively lost at the nodes which join the rigid part, hence its mass must be sufficient to compensate. Users may set their own minimum mass value.

The current element quality check settings apply to both standard checking and APPLY_RULES checking. To invoke

the latter the rules checks must be activated.

Note: some checks may require intensive operations, e.g. Model %age added mass requires calculation of model mass. On very large models, this may make the checking process appear slow.

3.9.1.1 Rules check



The **CHECK** >**RULES** function applies a set of custom checks which can be controlled through the oa_pref file or by the **OPTIONS** panel.

The standard model check and element quality checks are as described above. The model checking is always run without contact checks in this mode.

Contact check - if these are active, for example by the oa_pref settings
- **primer*contact_penetration_rule**: ON
- **primer*contact_penetration_max_allowable_value**: 0.2

Rules check will run the contact checker directly and report penetration (above thresh-hold) and crossed edge count. In this case **Details** button will access Primer's penetration checker via an intermediate panel.

Rigid-body separation check - measures the distance between the centroids of merged rigid bodies and reports those that exceed the given value.

This check is now done as part of model checking if the option is active. For historic reasons, it is available as separate feature in rules check. **Details** will start a bespoke visualization panel. The oa_pref settings are:
- **primer*rigid_body_merge_check:** ON
- **primer*rigid_body_merge_max_separation:** 200.0

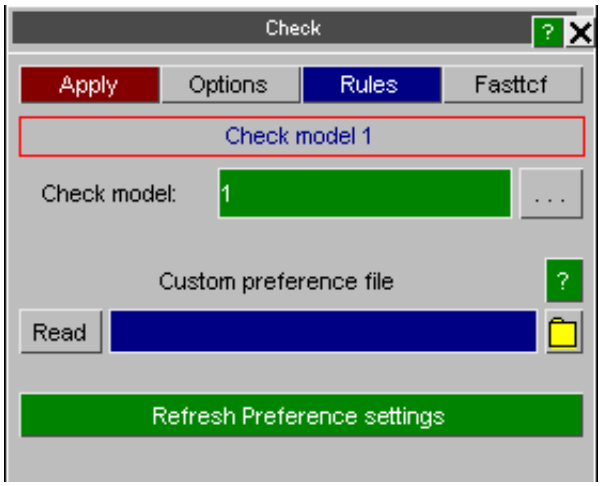
Missing item check is always done. If, for example, an element references a part, but the part card is absent from the model, the missing item check will detect it.

For model check, the **Details** button will list take the user to the standard checking panel.

Whenever a custom check is made a summary is dumped to the text file **apply_rules.txt** written in cwd.

3.9.1.2 Custom oa_pref file

On start up Primer reads the system oa_pref file (in the $OA_INSTALL directory) and then the home oa_pref file (in the home area) and finally any oa_pref file in the current working directory (which is an uncertain concept on windows machines). The last preference file may be considered to be a custom pref file. Primer now has a more effective way of handling these which disposes of the need to put the file into cwd.



Under **CHECK > OPTIONS** there is a function for selecting and reading the custom pref file. If you have changed any check options previously in this session, it is advisable to run **Refresh Preference settings** to restore the settings to their state at start up. Note that reading a preference file will set to ON or OFF any preferences that are present in the file, but omitting a preference **does not turn it OFF**. Therefore, you m ust ensure that your any settings which are activated by home or system preference file are actively turned OFF in the custom pref file if you do not want them.

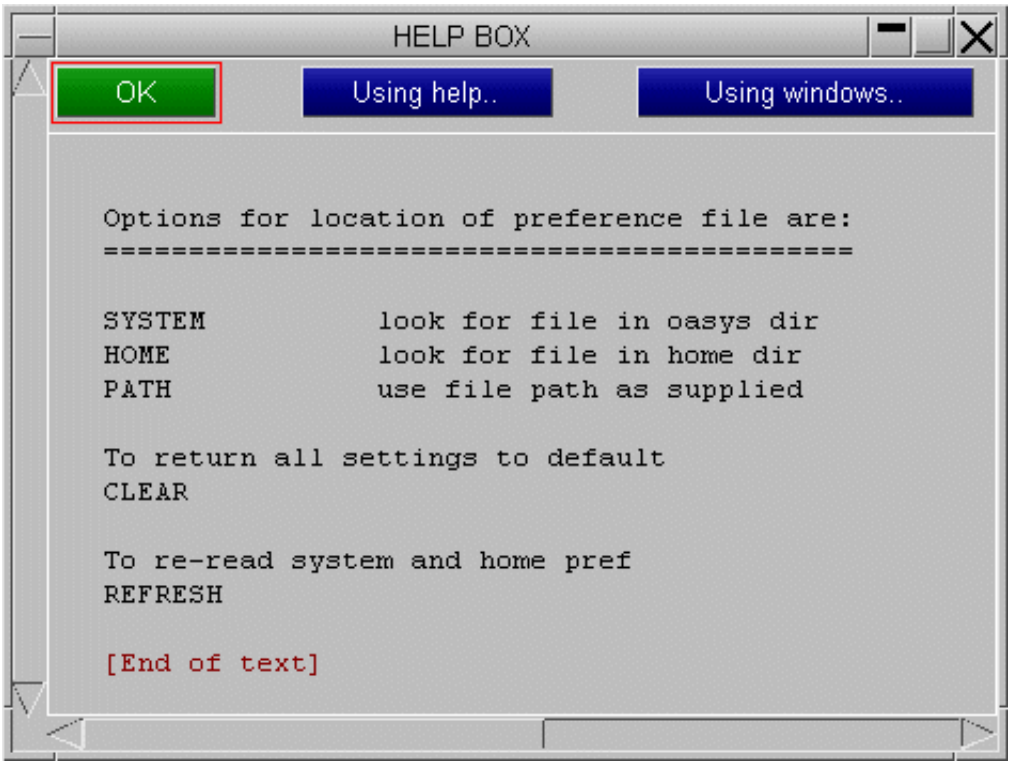
The custom pref file may be specified as a command line argument "-pref=xxx", see command line arguments.

A custom file may be applied for model checking by using batch file. For example, the following command file will read the custom oa_pref file "pr90.pref" from the system area

/pref system pr90.pref

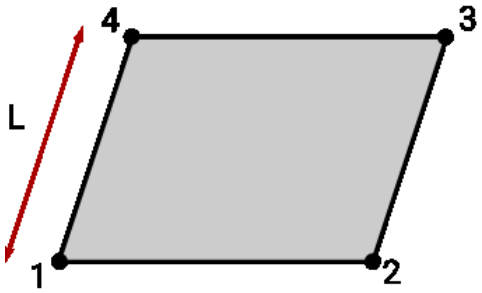
/read dk pr90.key 1

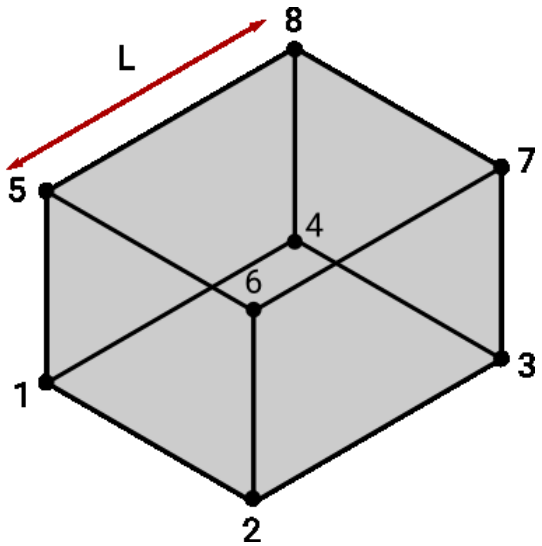
/check full list model 1 apply



3.9.1.3 Min Length

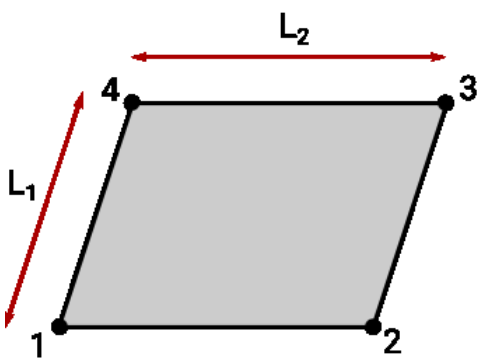
The length of each edge of an element is calculated, the smallest of which is then compared with the specified minimum length criteria.

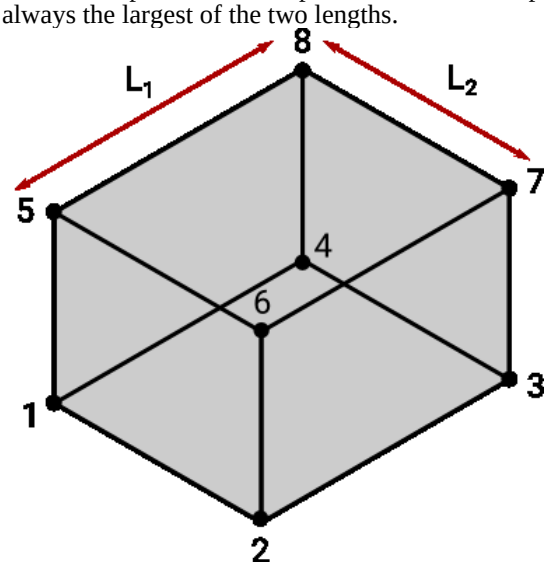




3.9.1.4 Max Aspect Ratio

For each face of an element, the length of each edge is calculated. The ratio of the lengths of each adjacent edge is then calculated. The largest (in the element) of these ratios is then compared with the specified maximum aspect ratio criteria. When calculating the ratios, the numerator is always the largest of the two lengths.



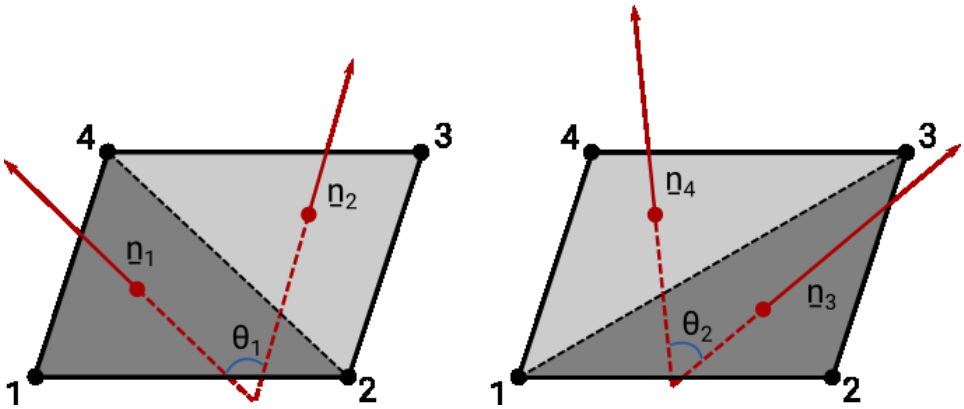


$$ratio = \left(\frac{L_i}{L_{j \neq i}}\right)_{max}, \qquad L_i > L_j$$

3.9.1.5 Max Warpage

The warpage for each face of an element is calculated, and the maximum of each of these is then compared to the maximum warpage criteria. Any triangular faces are given a warpage of 0.

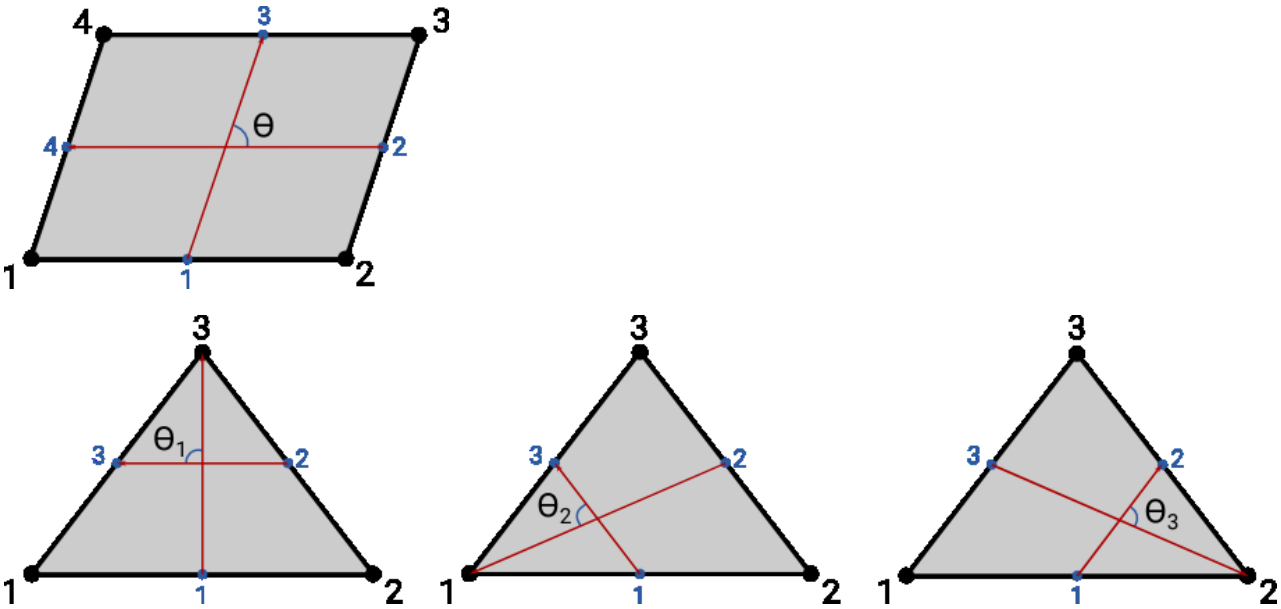
To determine the face warpage the face is first split along a diagonal (as below), such that the quad is then made up of two triangles. The angle (in degrees) between the unit normals of these triangles is then calculated. The same is done for the 'alternative' split, and the maximum of the two resulting angles is taken to be the warpage for that face.



3.9.1.6 Max Skew

The skew is calculated for each face of an element, the largest of which is then compared to the maximum skew criteria. For quadrilateral faces, the midpoint of each edge is found. Vectors are then drawn between opposite midpoints (as shown below). The skew for that face is then taken to be the acute angle (in degrees) between these two vectors.

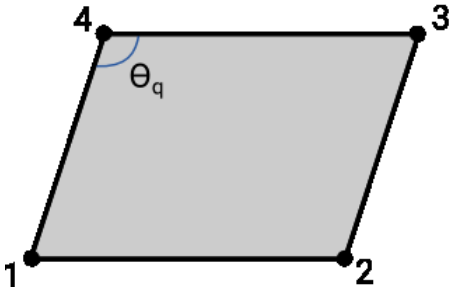
For triangular faces, the midpoint of each edge is found. One vector is drawn between one midpoint and the opposite node, while the other vector is drawn between the two remaining midpoints. The acute angle between these two vectors is then found. This is repeated until each node has been 'used' to define a vector (see below). The largest these angles is taken to be the skew for that face.

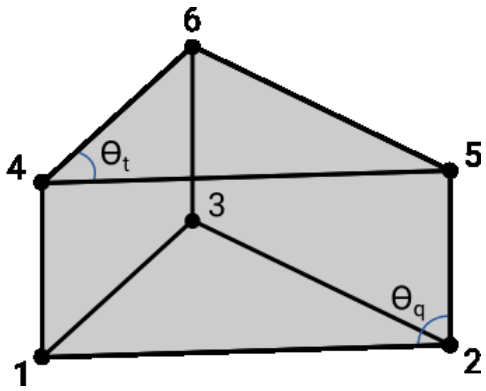


3.9.1.7 Min/Max Angle

The internal angle of each vertex of each face, within an element, is calculated, with the largest and smallest then being compared to the maximum and minimum internal angle criteria respectively

Checking the internal angles of triangular and quadrilateral faces can be turned on separately. For SOLID elements that have both triangular and quadrilateral faces both checks must be turned on to test all faces (otherwise only the triangular or quadrilateral faces will be tested, depending on which check is turned on and which is turned off).





3.9.1.8 Max Jacobian

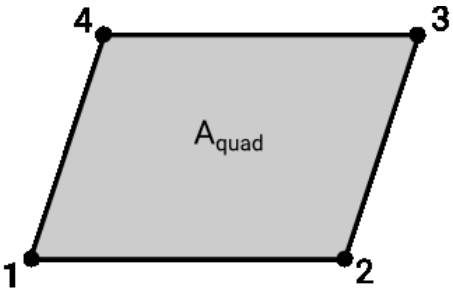
Jacobian checks determine how much an element differs from its ideal shape. In PRIMER, these checks can be made using either the element's nodal points or Gauss points. Triangle SHELLs and tetrahdral SOLIDs are given perfect values of 1.0.

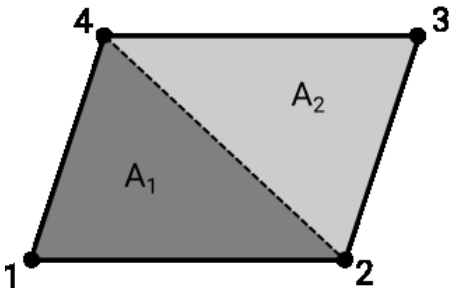
3.9.1.9 Max Taper

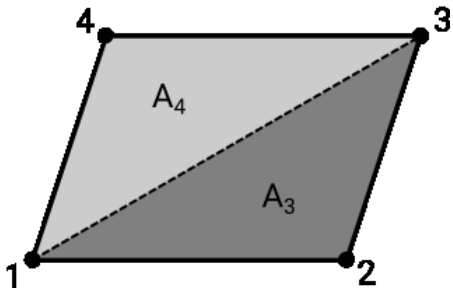
Taper checks are only calculated for quadrilateral SHELL elements. Triangular SHELL elements are given a taper value of 0 so as to not flag a warning in the Error tree, though this means that they will be drawn in a taper CT/SI plot.

The quad is first split along one of its diagonals such that it is made up of two triangular areas (as shown below). The area of each triangle is then divided by half of the area of the quad. This is repeated for the 'alternative' split (see below). The smallest of these fractions is then taken away from 1, the result of which is the element's taper value.

Due to the way in which the area of the quad is defined, the taper value can become negative for warped SHELLs. This area definition may also cause discrepencies between taper values provided by other pre-processor software.





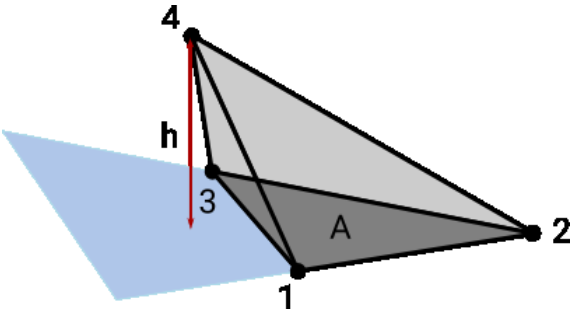


$$taper = 1 - \left(\frac{2A_i}{A_{quad}}\right)_{max}$$

3.9.1.10 Min Tet Collapse

Tet collapse checks are only calculated for tetrahedral SOLID elements. Other SOLID elements are given a tet collapse value of 1.0 so as to not flag a warning in the Error tree, though this means that they will be drawn in a tet collapse CT/SI plot.
For each node in a tetrahedron, the node's height above the base (the opposite face) is divided by the square root of the base's area. The smallest of these fractions is then divided by 1.24, the result of which is taken as the element's tet collapse value.

(The height of the node is defined as the shortest distance between that node and the plane described by the base.)



$$tet\ collapse = \left(\frac{h_i}{1.24\sqrt{A_i}}\right)_{min}$$

3.9.2 CHECK output

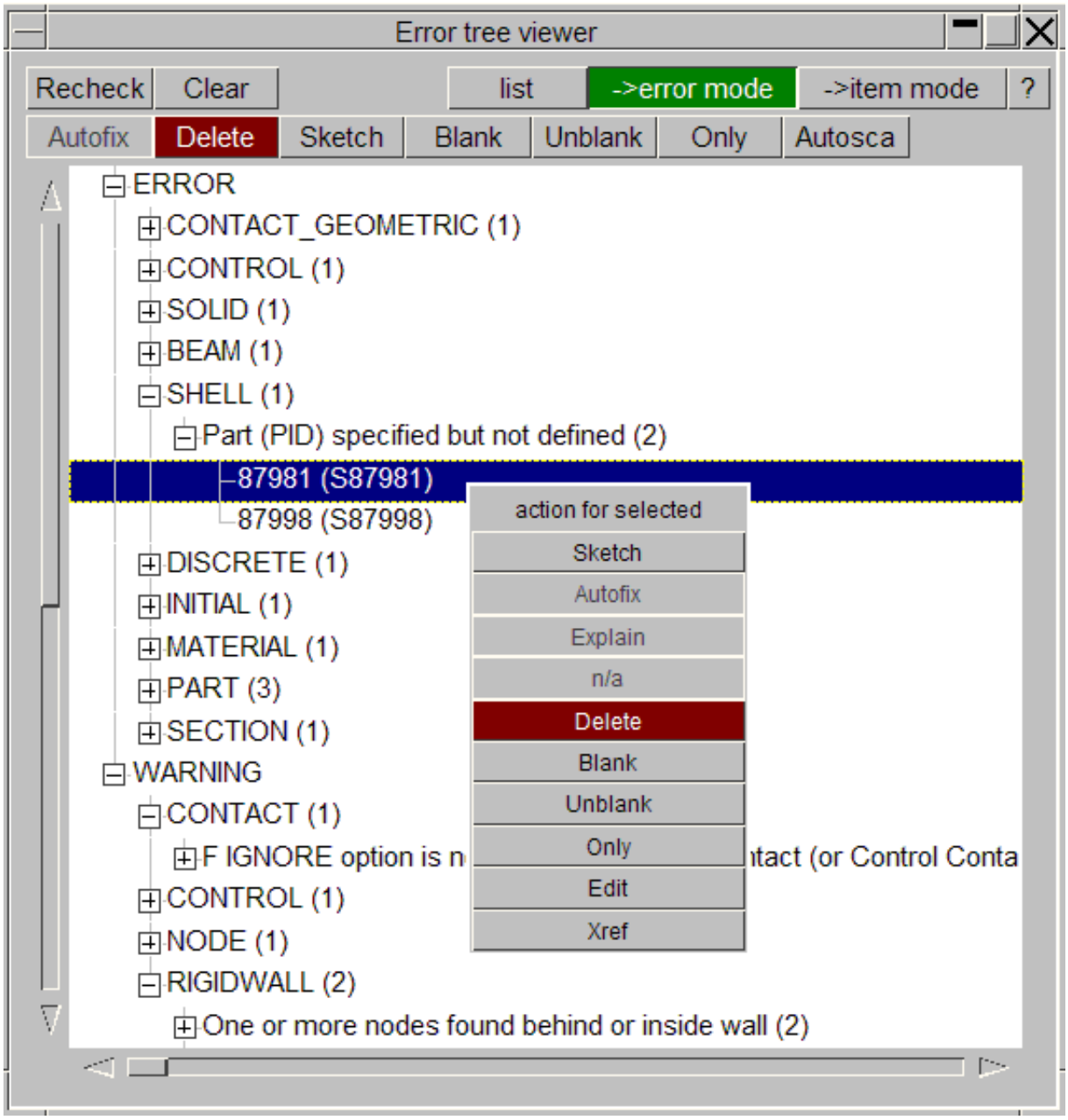
Two panels will appear during the model checking:

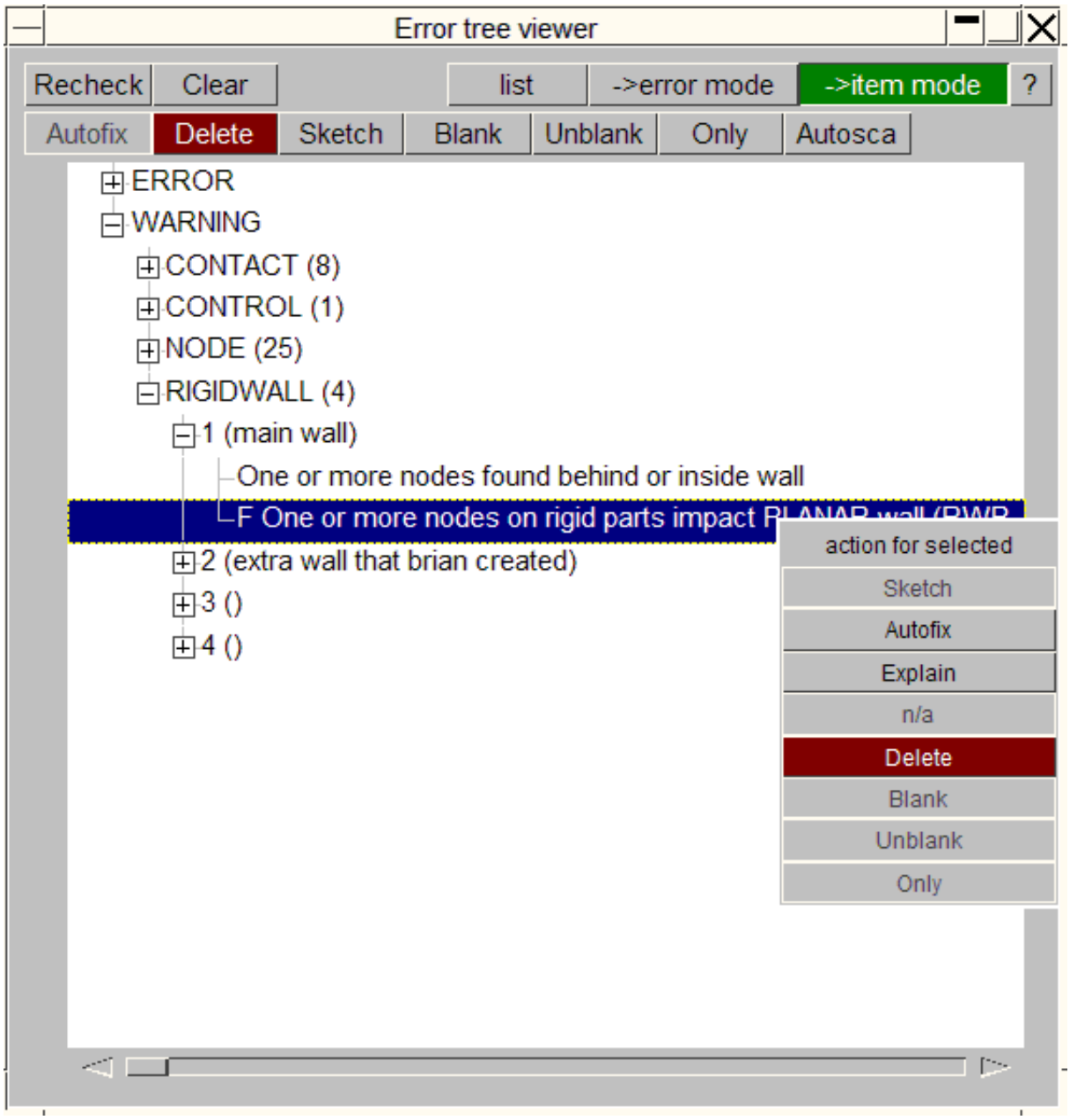
1. **Error tree viewer** this groups the errors and warnings into categories and sub-categories. Intuitive navigation through the checks is possible via a tree system
2. **Check model summary panel** this gives a compact summary of the errors and warnings found in a table format, with a popup from each entity type

3.9.2.1 Error tree viewer

When PRIMER has performed the model check, the results are grouped into errors and warnings and then sub-divided into entity categories. Each category has further divisions to group the checks together, depending on the mode selected:

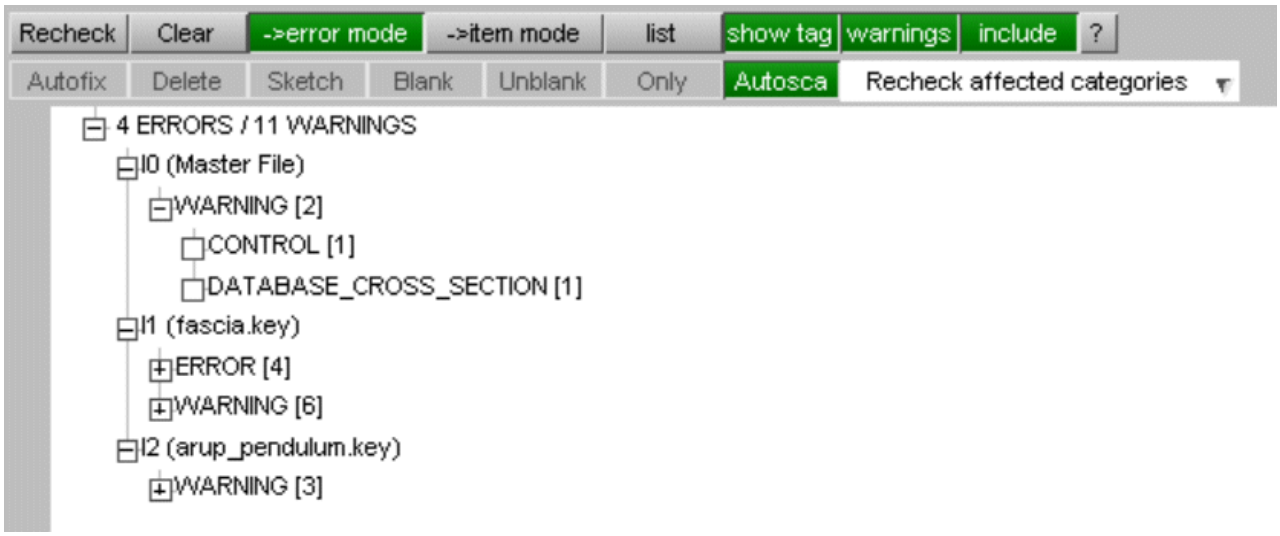
1. **>error mode** lists under each error code the entities which exhibit the error. This is the default mode.
2. **>item mode** lists under each entity which has been found to have an error, the error or errors which pertain to it


Error tree viewer
Recheck
Clear
list
->error mode
->item mode
?
Autofix
Delete
Sketch
Blank
Unblank
Only
Autosca
ERROR
CONTACT_GEOMETRIC (1)
CONTROL (1)
SOLID (1)
BEAM (1)
SHELL (1)
Part (PID) specified but not defined (2)
87981 (S87981)
87998 (S87998)
DISCRETE (1)
INITIAL (1)
MATERIAL (1)
PART (3)
SECTION (1)
WARNING
CONTACT (1)
F IGNORE option is n
ntact (or Control Conta
CONTROL (1)
NODE (1)
RIGIDWALL (2)
One or more nodes found behind or inside wall (2)
action for selected
Sketch
Autofix
Explain
n/a
Delete
Blank
Unblank
Only
Edit
Xref



Navigation of the errors and warnings is very simple due to the hierarchy within the error tree. Categories can be expanded or contracted using the small box to the left of the headings. Once the categories have been expanded a number appears to the right of the heading - showing the number of sub-categories directly underneath this level.

INCLUDE button will add an extra layer to tree so the errors are diplayed per include file. **WARNINGS** button will toogle these on or off.



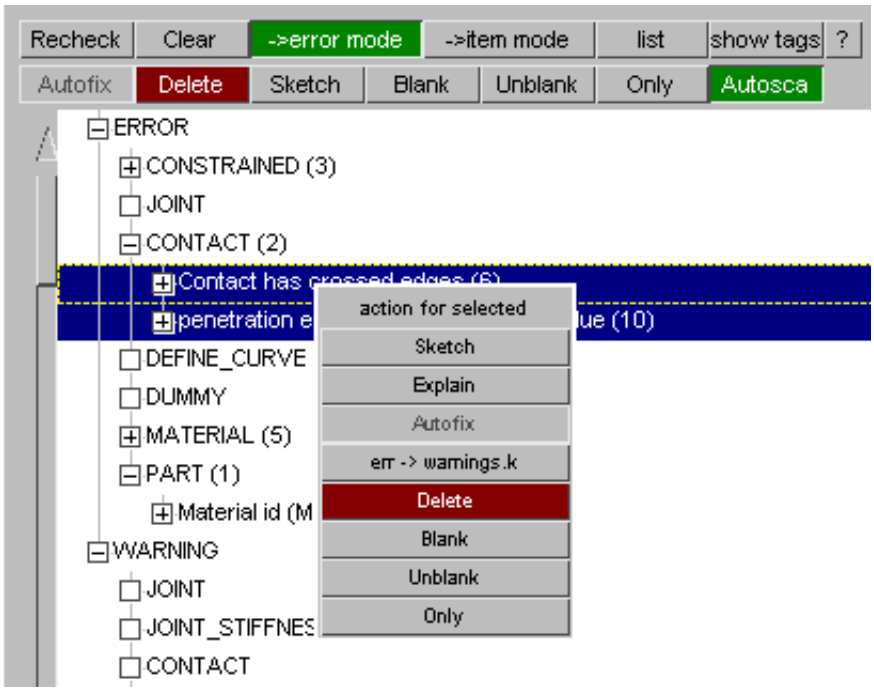
Various operations are available once an error(s) has been selected. Multiple selections of errors are warnings are possible using the shift and control keys (similar to the part tree behaviour). Viewing operations can be utilised through the buttons at the top of the panel to **sketch**, **unblank**, and manipulate the model display to help identify the problem.

List will give a concise summary of the errors and warnings in a text box. If **INCLUDE** is active, the listing will be by include file.

Right-clicking on an selection brings up a popup (the figure shown above). Here, if possible the selection can be **sketched**, **autofixed** (this is explained below), **explained** (extra details if available), **deleted**, and viewed in different ways. The user can also **edit** the entity (if a single entity has been picked) or look at its **xrefs** within the model using the cross-reference viewer in a separate panel.

On completion of an autofix or deletion of item in error, in consideration of speed with larger models the recheck function will be run only on the type directly affected. In most cases this action will be suffiecient. However, sometimes fixing one error will fix other errors implicitly and occasionally it may cause new errors to appear. It is recommended, therefore, that you do a full recheck when you have completed you fixes by pressing **Recheck.**

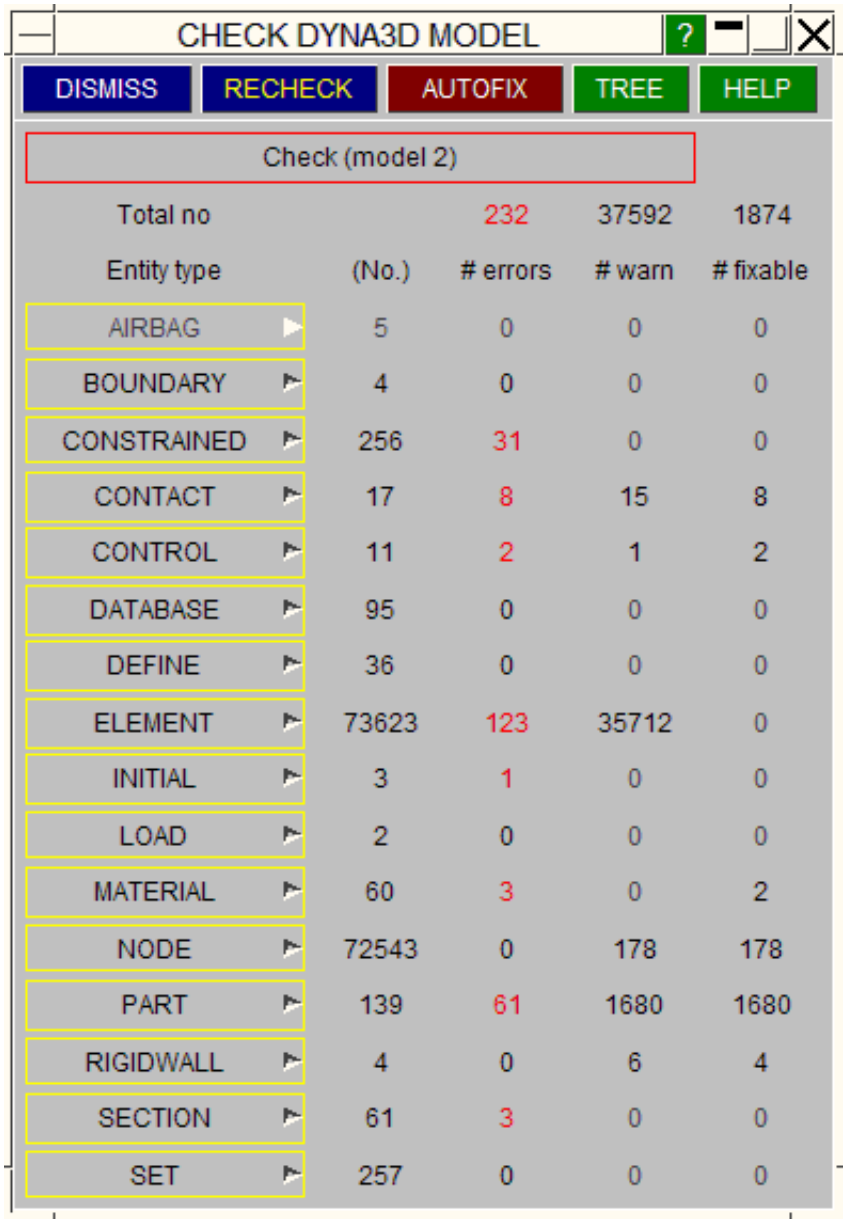
The drop-down from the check tree allows Nodes and Elements in error to be written to a set, parts to be put onto the table, and contact penetration and tie errors to be written to sets in a separate include (warnings.k) file. Pressing **err->warnings.k** will run the contact checker and generate node and segment sets each with a title corresponding to the error detected.


Recheck
Clear
->error mode
->item mode
list
show tags
?
Autofix
Delete
Sketch
Blank
Unblank
Only
Autosca
ERROR
CONSTRAINED (3)
JOINT
CONTACT (2)
Contact has crossed edges (6)
penetration e
ue (10)
DEFINE_CURVE
DUMMY
MATERIAL (5)
PART (1)
Material id (M
WARNING
JOINT
JOINT_STIFFNES
CONTACT
action for selected
Sketch
Explain
Autofix
err -> warnings.k
Delete
Blank
Unblank
Only

3.9.2.2 Summary table panel

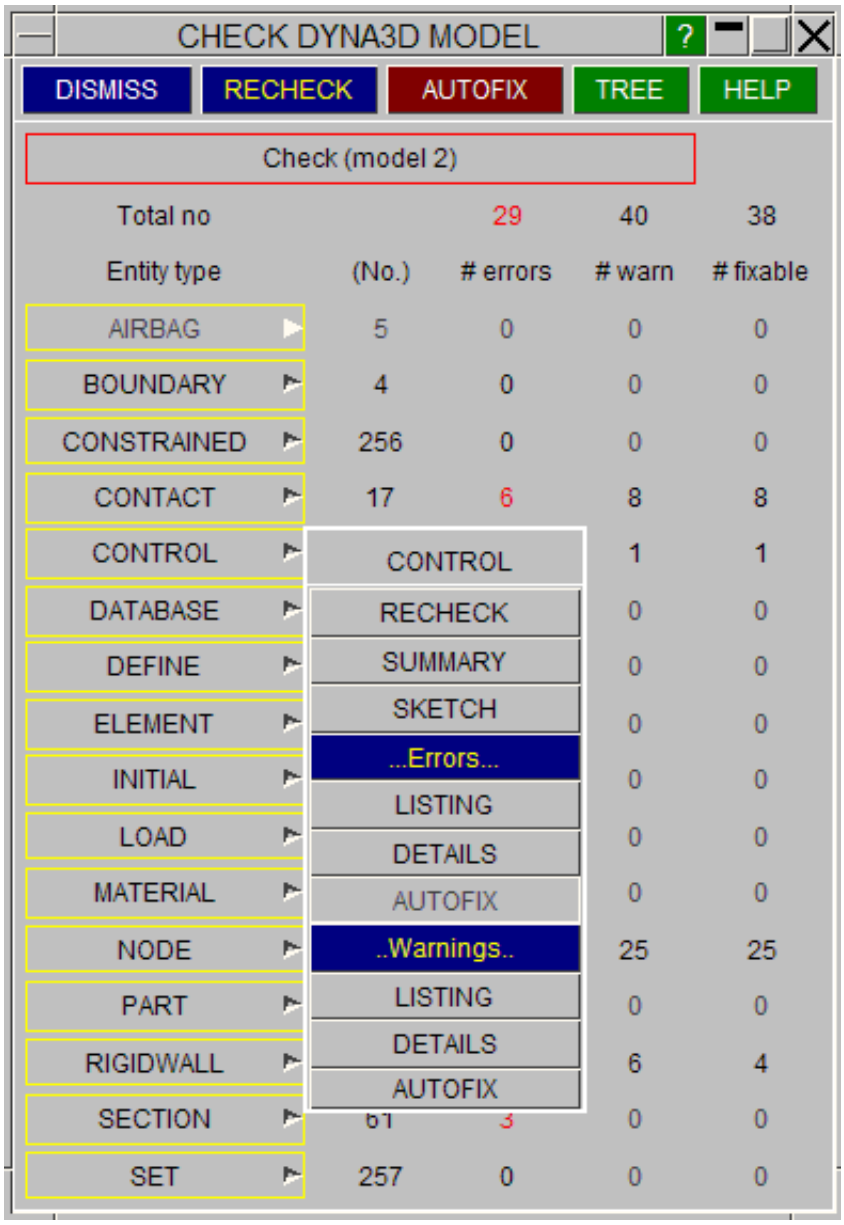
The second check panel gives a concise summary of every master keyword category in the model, with a listing of:

- **(No.)** The total number of items of that keyword
- **#errors** The number of errors found
- **#warnings** The number of warnings found.
- **#fixable** The number of these errors/warnings that can be "Auto fixed"



Total no		232	37592	1874
Entity type	(No.)	# errors	# warn	# fixable
AIRBAG	5	0	0	0
BOUNDARY	4	0	0	0
CONSTRAINED	256	31	0	0
CONTACT	17	8	15	8
CONTROL	11	2	1	2
DATABASE	95	0	0	0
DEFINE	36	0	0	0
ELEMENT	73623	123	35712	0
INITIAL	3	1	0	0
LOAD	2	0	0	0
MATERIAL	60	3	0	2
NODE	72543	0	178	178
PART	139	61	1680	1680
RIGIDWALL	4	0	6	4
SECTION	61	3	0	0
SET	257	0	0	0

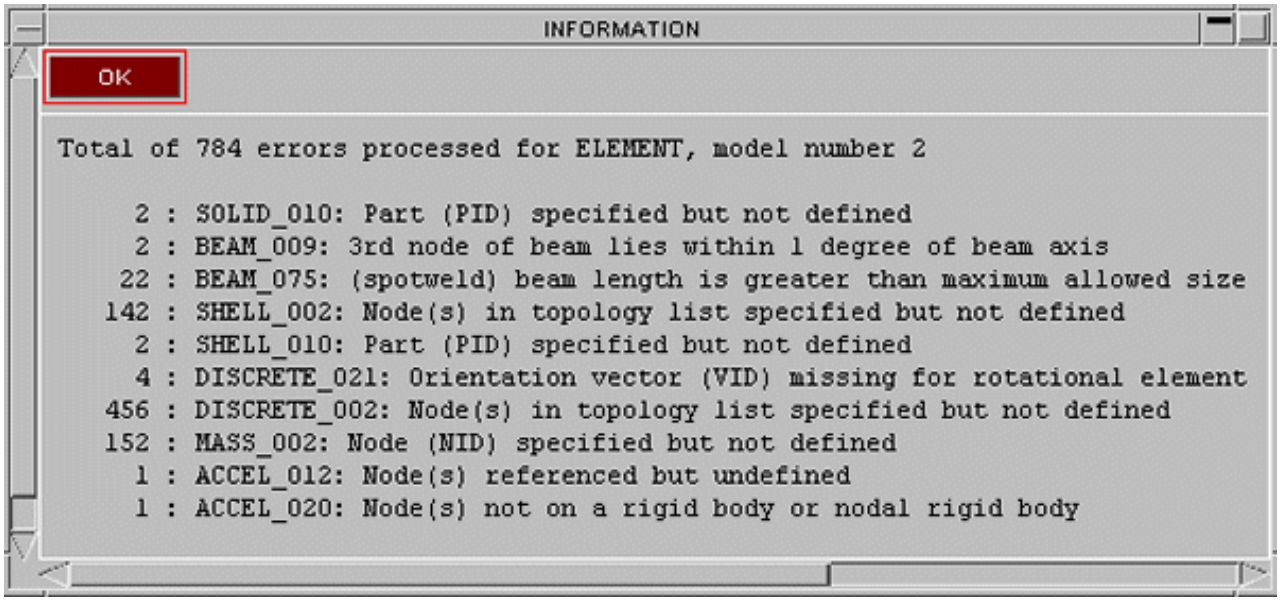
You can then use the popup menu of each category to examine problems in more detail. In this figure the user has used the popup menu for category **CONTROL**. The popup displays warnings and errors in separate areas.



Click on the links for descriptions of:

SUMMARY
LISTING
DETAILS
AUTOFIX
SKETCH

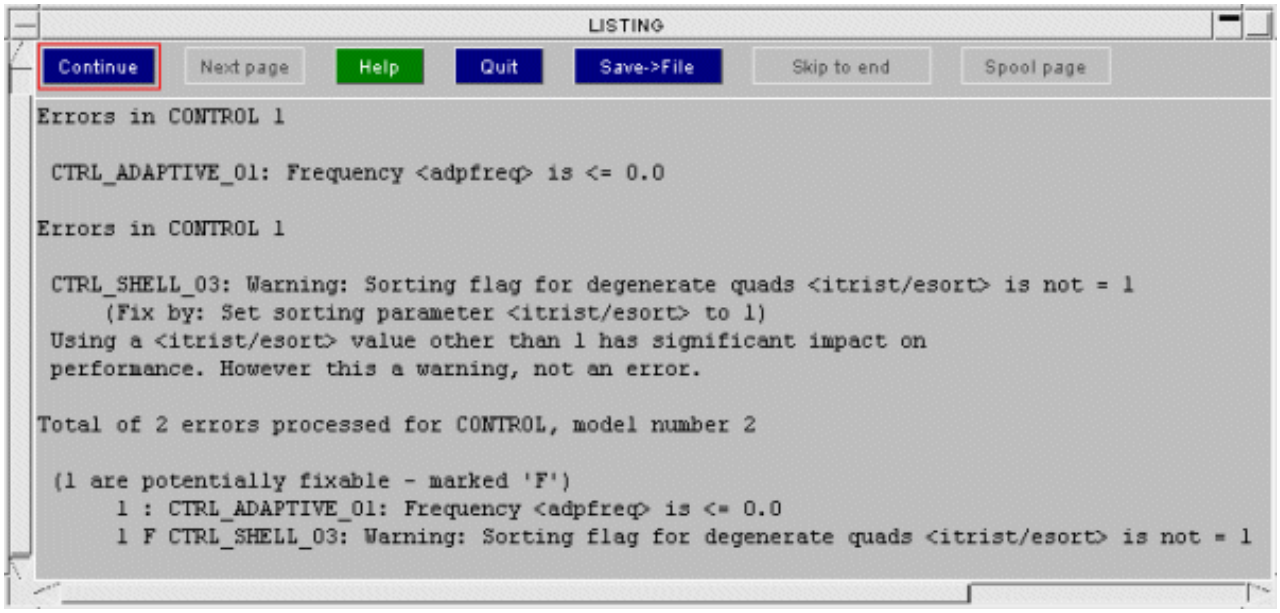
3.9.2.2.1 > SUMMARY



Lists only the number of occurrences of each class of error for this category.

Here ELEMENTS are listed.

3.9.2.2.2 > LISTING

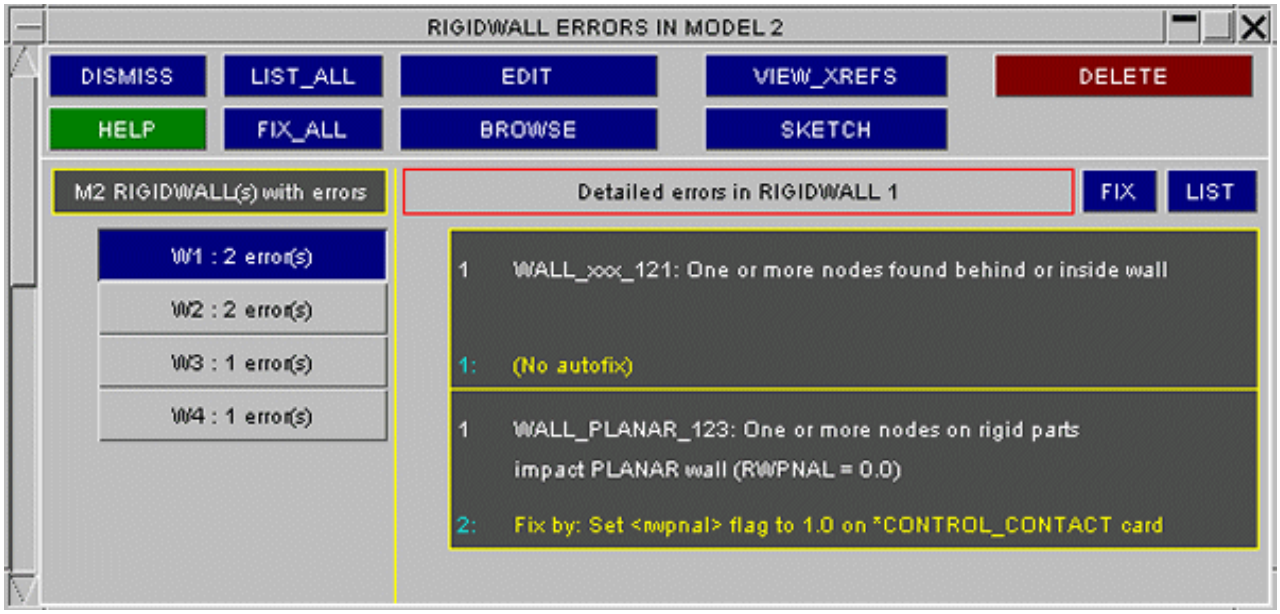


For each item with errors lists the details of the error and any extra information available.

If it can be "Auto-fixed" a description of how is also given (as here).

This can be a long listing, so it is paged and can be saved to disk file with **SAVE->FILE**.

3.9.2.2.3 > DETAILS

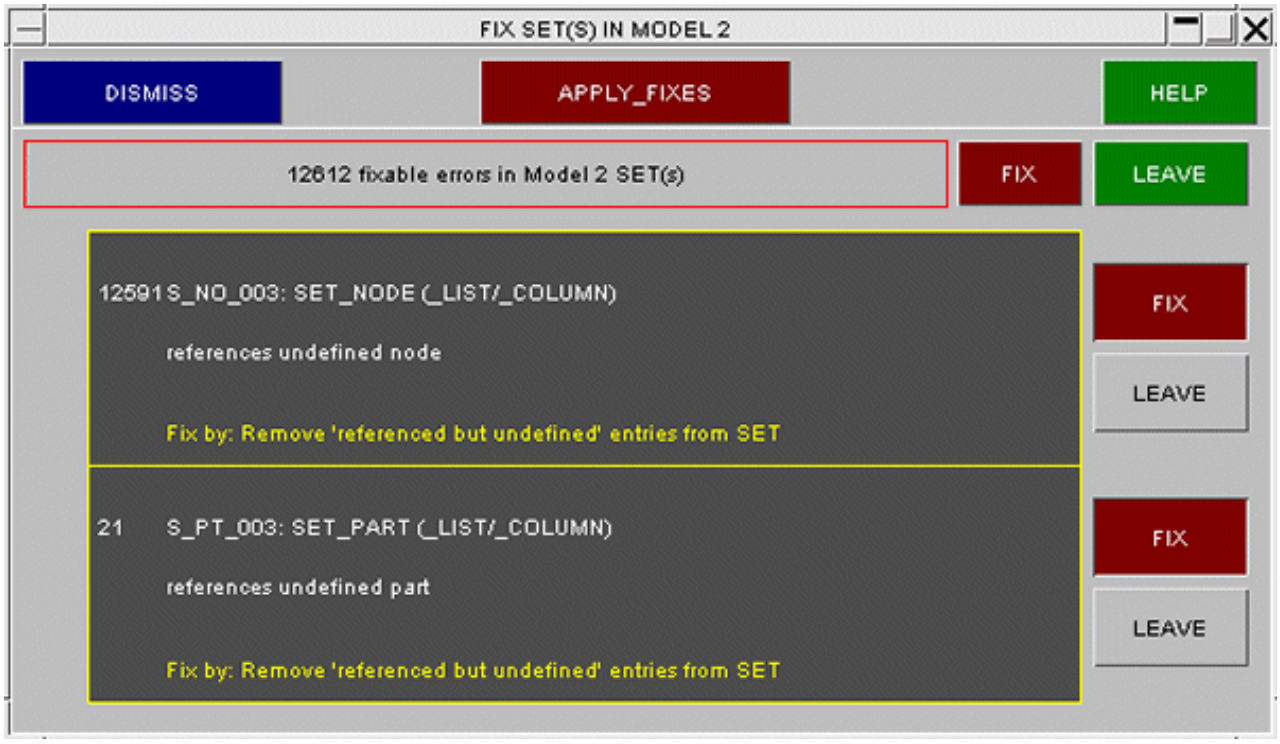


For each category builds a panel containing a list of the items with errors down the left hand side.

These can be selected in turn to view their specific errors, and to fix them (or not) by whatever means you wish.

Auto-fixes may be applied on a case-by-case basis.

3.9.2.2.4 > AUTOFIX



Builds a panel containing all fixable errors for this category.

You can choose to **FIX** or **LEAVE** each error.

APPLY_FIXES will then correct what you have selected.

In this way Auto-fixes may be applied to all errors in this category.

3.9.2.2.5 > SKETCH

The items in the category which contains errors may be sketched on the current image.

3.9.2.3 Auto-Fixing errors.

"Auto-fixing" is where PRIMER offers to correct errors it has detected for you. Generally PRIMER will only offer to make fixes where it can do no harm, so typical fixable errors would be:

- "Out of range" values, which would be reset to defaults.
- Duplicate entries in sets or time-history blocks, which would be eliminated.
- Reversing element topology to make area or volume +ve.
- Eliminating segments that don't lie on element faces.

. Auto-fixes can be applied:

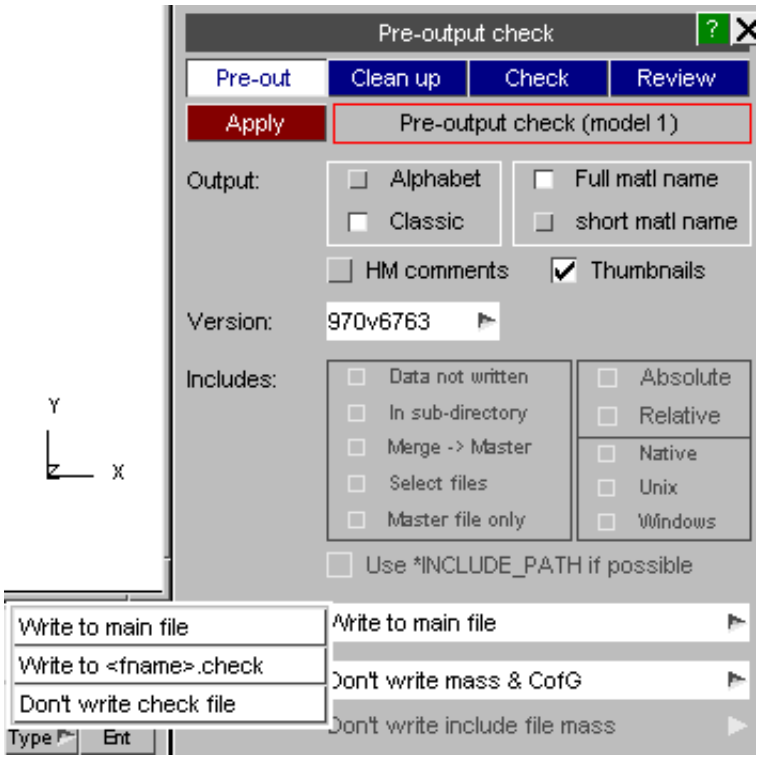
- On an entity by entity basis via the **DETAILS** popup in the check summary panel;
- On a whole category basis via the **AUTOFIX** popup in the check summary panel;
- Using the autofix option when right-clicking in the error tree viewer.
- To the whole model via the **AUTOFIX** button at the top of the check summary panel.

3.9.2.4 Notes on checking:

- All keyword categories present in the model are shown, but those for which checking functions are not available are greyed out. These will follow in the fullness of time.
- That no errors have been detected is no guarantee that there are none! The checking algorithms in PRIMER are not based on those in LS-DYNA, and they pick up some that DYNA misses, and vice-versa.
- Checking may be done at any time, but is good practice to check models prior to output, and in fact this is the default behaviour in the model **WRITE** command.
- A check that produces many errors is sometimes due to the consequences of deleting items leaving an "untidy" model. Try a **REMOVE**, **CLEANUP_UNUSED** operation before trying more detailed corrections, since it may sort out most of the problems.

3.9.2.5 Keyout error check

On keyout a listing of the results of model checking can be written either to the top of the keyword file or to a new file which will have the same name as the keyword file but with a **.check** extension. This may be set up on the pre-output check panel or by making appropriate **oa_pref** settings.



3.9.2.6 Batch error check and autofix

Checking may be called from the command line using the syntax: **CHECK MODEL <n> CHECKFILE <filename> APPLY** or **CHECK MODEL <n> APPLY.**

You may include the option **FULL_LIST** if you want a detailed listing of every item label in the file, otherwise labels will not be printed for the more populous items types (such as nodes and elements).

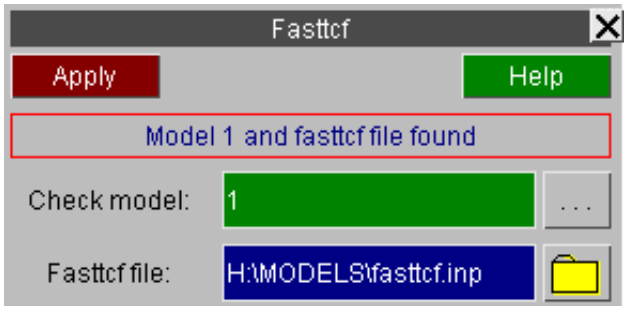
Similarly the Autofix all function may be called in command line using the syntax **AUTOFIX MODEL <n> APPLY**.

3.9.3 FASTTCF CHECK

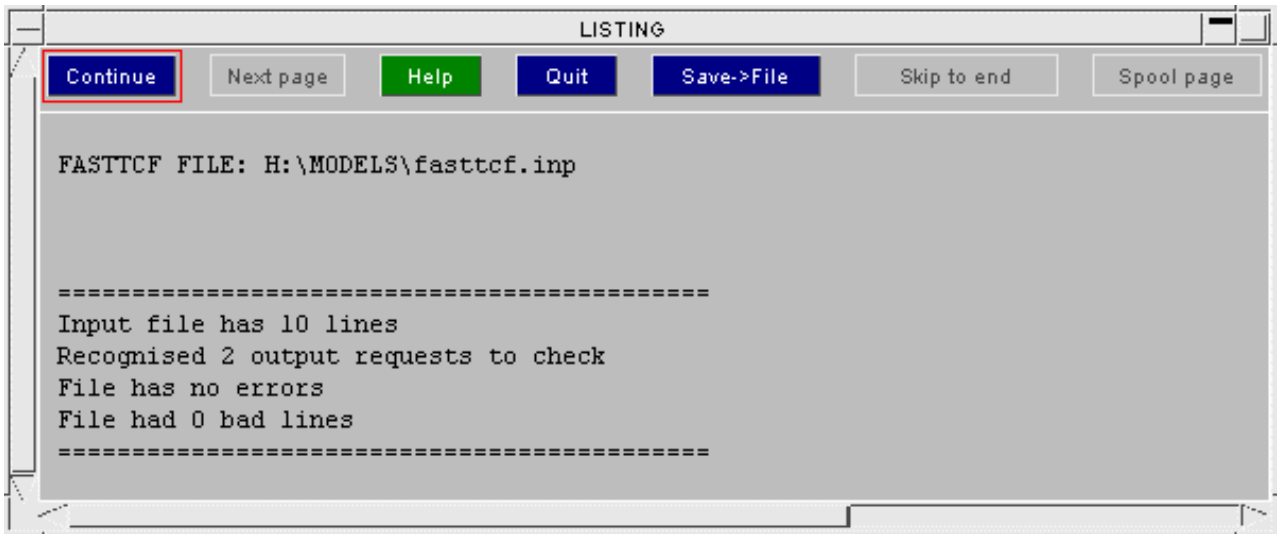
This section analyses a FASTTCF file and checks any data extraction requests for errors. The main checks are as follows:

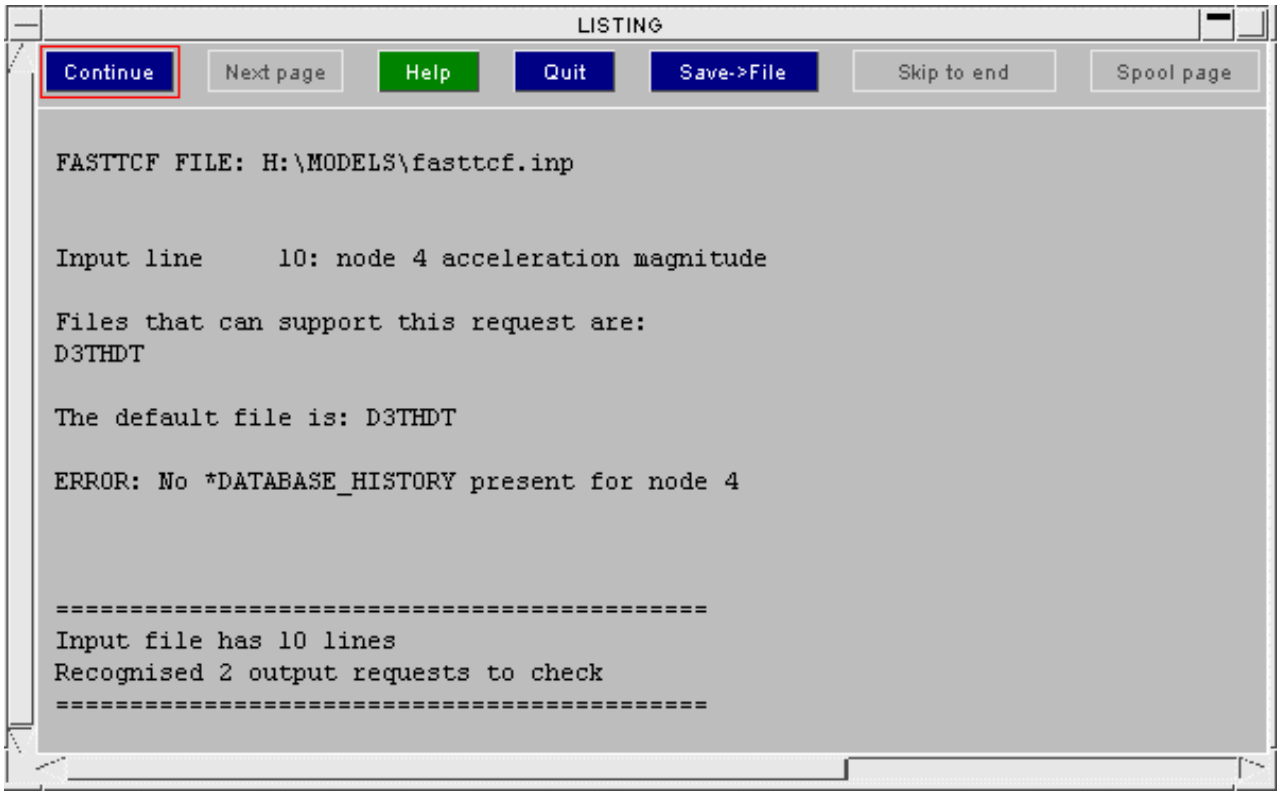
1. The request is supported by FASTTCF
2. The file requested for the extraction is being outputted from DYNA
3. The data extraction request is included in the *DATABASE_HISTORY output if it needs to be
4. If the entity i.d. exists or if it is latent

Choose the fasttcf file to check, and the model to check against. Then press the **APPLY** button.



Once the file has been checked a report text box appears to highlight any errors found. The following two images are examples of the output text. The first has no errors reported and the second has an error regarding the database history output. To solve the error in the second example the *DATABASE_HISTORY_NODE should have node i.d. 4 added.





3.9.4 Error vs Warning: User configuration

By default Primer offers two levels of configuration - error and warning. We have assigned "error" status to, cases where the error will prevent a model initializing, is likely to result in a model failing to run to completion or to give sensible results and those where a user defined criterion is infringed.

We have assigned the less severe "warning" status to cases which will not stop a model running but could have an adverse affect on the result.

This is not an exact science. So users now have been provided with the ability to configure errors for themselves. Within Primer the any existing error can be given user status which will over-ride the default. Additionally a string may be attached to the error which can be identified in the output.

The implementation is as follows:

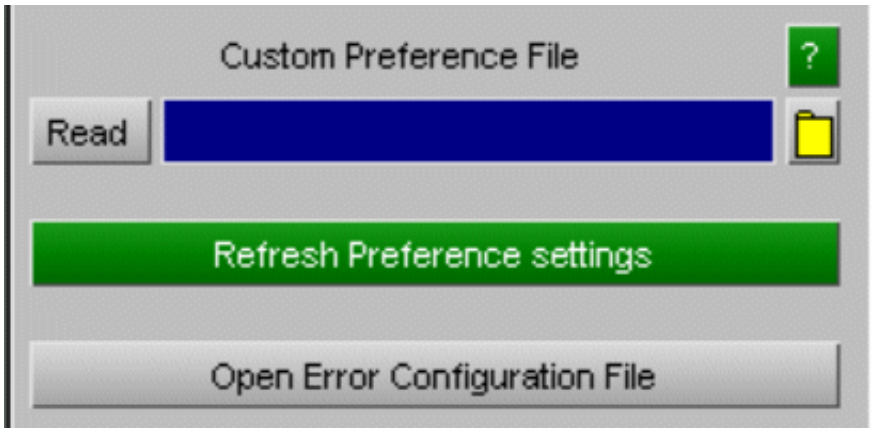
Set up a comma separated error config file which consists of one line per error in the format - error tag, user config, optional extra comment, e.g.

JNTC_13, ERROR, FATAL ERROR
JNTC_14, ERROR, FATAL ERROR
JNTC_15, ERROR, FATAL ERROR
JNTC_16, IGNORE
The **error tag** is the unique code which identifies the error and can be displayed on the error tree (you need to have a model in which the error exists and to activate **show tags** on the error tree). The **user config** may be ERROR, WARNING or IGNORE which will determine how the error will be displayed in Primer (if at all). The **optional extra comment** will be printed in the error summary output file and may be detected by your own process

Define the oa_pref setting: **primer*error_configuration_file: /path/filename.**

If the error file exists Primer will read it on start up and report this in the dialogue box.

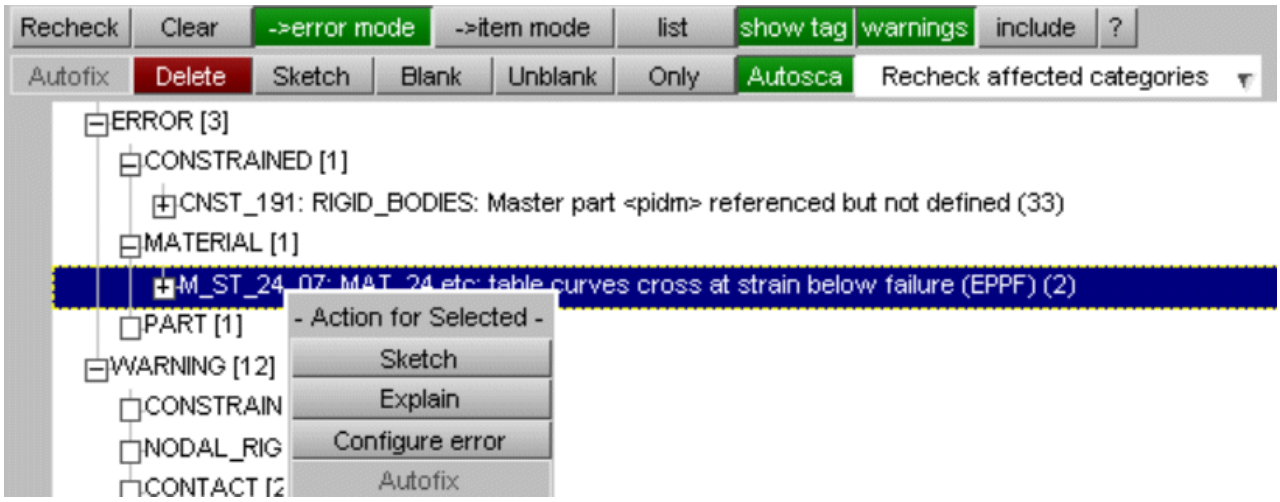
The error config file may be edited in a text editor by pressing **Open Error configuration File**.

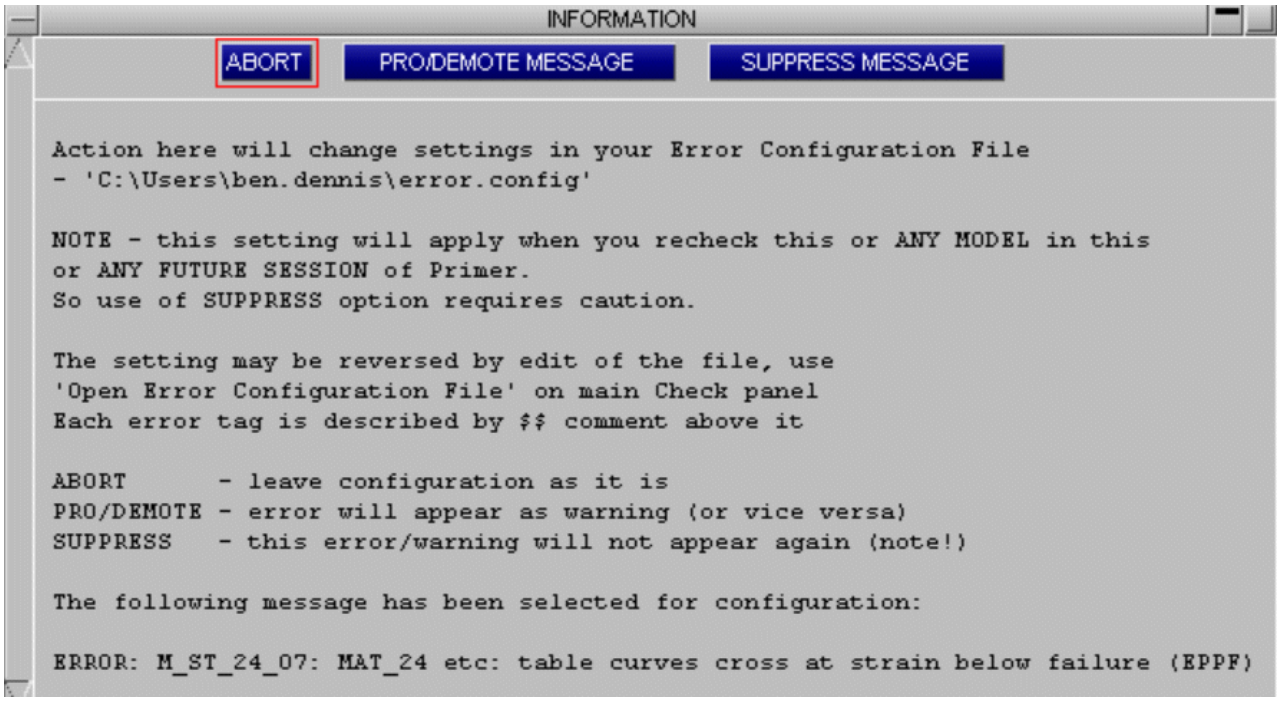


Creation/edit of this file may now be managed using **Configure error** option on the error tree available when you click on an error message.

If you don't have a configuration file one will be created in your home area called "error.config". All subsequent edit will apply to this file.

A message may be promoted from warning to error, demoted from error to warning or suppressed altogether by selecting **PRO/DEMOTE** or **SUPPRESS**





When you write an error file (using **list** function at top of error tree or otherwise), the extra message is included and can be detected by the controller.

The check listing can be written by using using a command file and running Primer in batch mode.

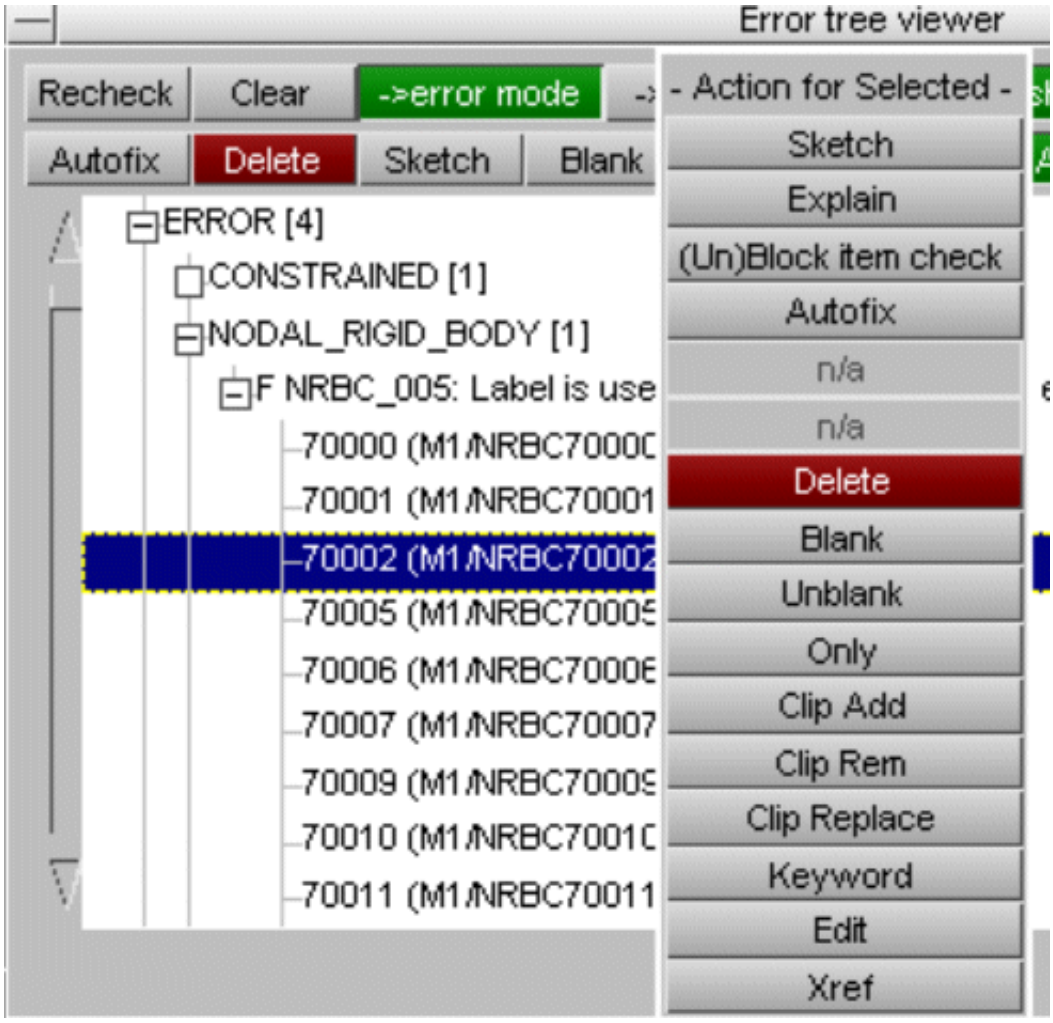
Users can then use their own process to detect the presence of significant strings (such as "FATAL ERROR" in this example) and take the appropriate action.

3.9.5 Blocking checks on individual items

Clicking on an item (as opposed to an error message) enables the ability to block checking of an individual item by pressing **(Un)Block item check** button.

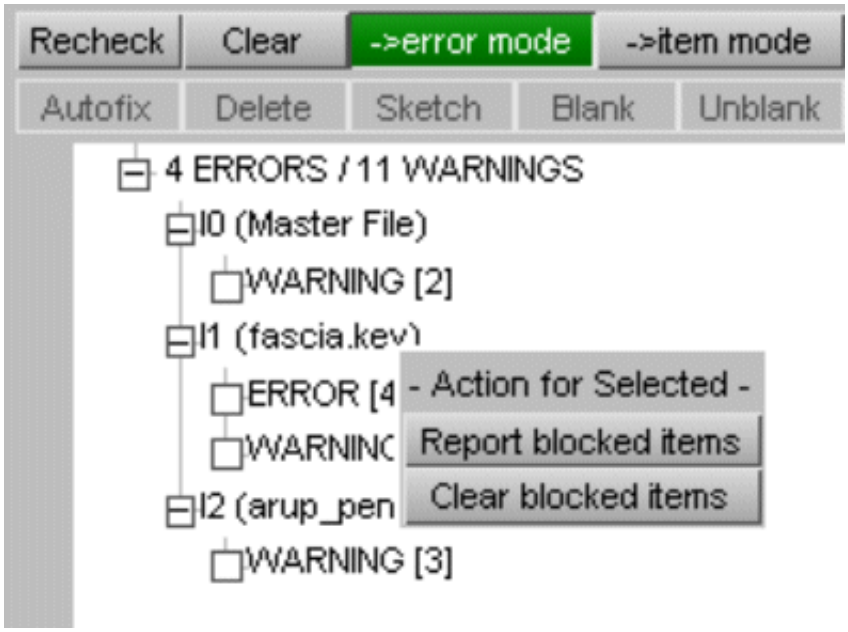
The same button will remove the block so long as you have not rechecked the model - then the entry will be gone.

This process writes a special keyword comment ($PR_suppress_item_check) to the item.



A report of blocked items for the whole model or in each include may be obtained by **Report blocked items** off the ERROR/WARNING header or off the include file.

There is also an option to **Clear blocked items** for whole model or per include. The user may also remove the comment using **Text edit** on the edit panel.



3.9.6 User defined checks using JavaScript

New user defined checks can be added by writing a check in JavaScript. Two different types of checks can be added.

- An item check that will be run for every item of a single keyword (e.g. every part in a model). This check should be in a file called **`<classname>`**`.js` in a subdirectory called `checks` in the PRIMER script directory. e.g. to write a part check the script should be called `Part.js` (case is important). Only one script file for each class is allowed **in each script directory** but the file can contain multiple checks. These checks can only be written if there is a class in JavaScript for the item type.
- A custom check that will be run for the whole model. This check should be in a file called custom.js in a subdirectory called `checks` in the PRIMER script directory.

PRIMER will automatically search for these scripts when doing a check and if found they will be run. The default script directories are:

- $OA_ADMIN/primer_library/scripts (if $OA_ADMIN is defined)
- $OA_INSTALL/primer_library/scripts
- $OA_HOME/primer_library/scripts

but these can be changed if required. See section 10.1.5 for more details).

So for example to add a part check that is accessible for all users you should create a file
`$OA_INSTALL/primer_library/scripts/checks/Part.js`

You could also write your own part check by adding another file
`$OA_HOME/primer_library/scripts/checks/Part.js`

Both scripts would be run.

User defined item checks

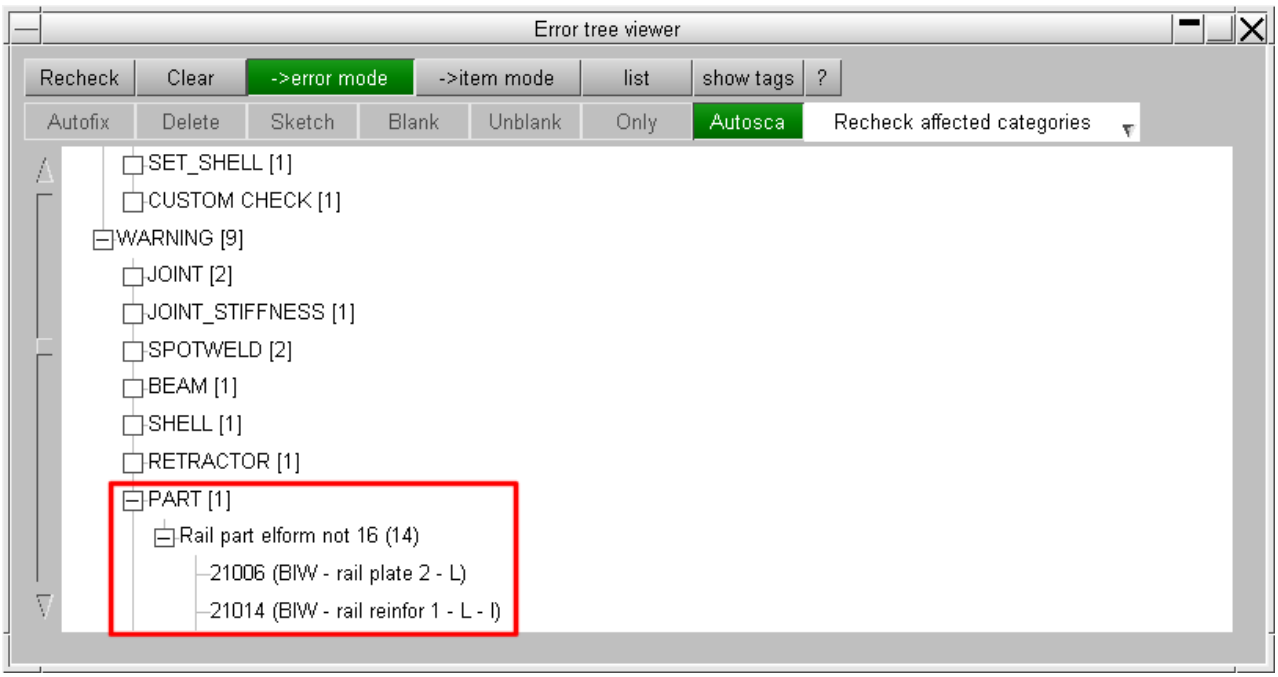
An item check will have 3 arguments passed to it automatically in the arguments array. The first argument is the script name, the second argument is the Model object, and the third argument is the object for the JavaScript item.

For example, we may want to check that any shell parts in the model that have 'BIW' and 'rail' in their title use element formulation 16. This could be done by writing the following script in Part.js.

```
// arguments[0] is name of script
var m = arguments[1];   // arguments[1] is model
var p = arguments[2];   // arguments[2] is part
if (p.exists && p.secid && p.heading.match(/BIW/) && p.heading.match(/rail/))
{
    var s = Section.GetFromID(m, p.secid);
    if (s.exists && s.type == Section.SHELL && s.elform != 16)
        p.Warning("BIW Rail part elform not 16");
}
```

You can trigger a warning or an error by using the `Warning()` or `Error()` methods in the item class. In the above example we triggered a Warning. You can trigger multiple warnings and/or errors for an item by calling the methods more than once with different strings. Warnings and/or errors with the same string will be grouped together (shown as a branch below) in the error tree.

These warnings and/or errors are shown in the tree just like normal warnings/errors. For example the output from the above check is shown below.

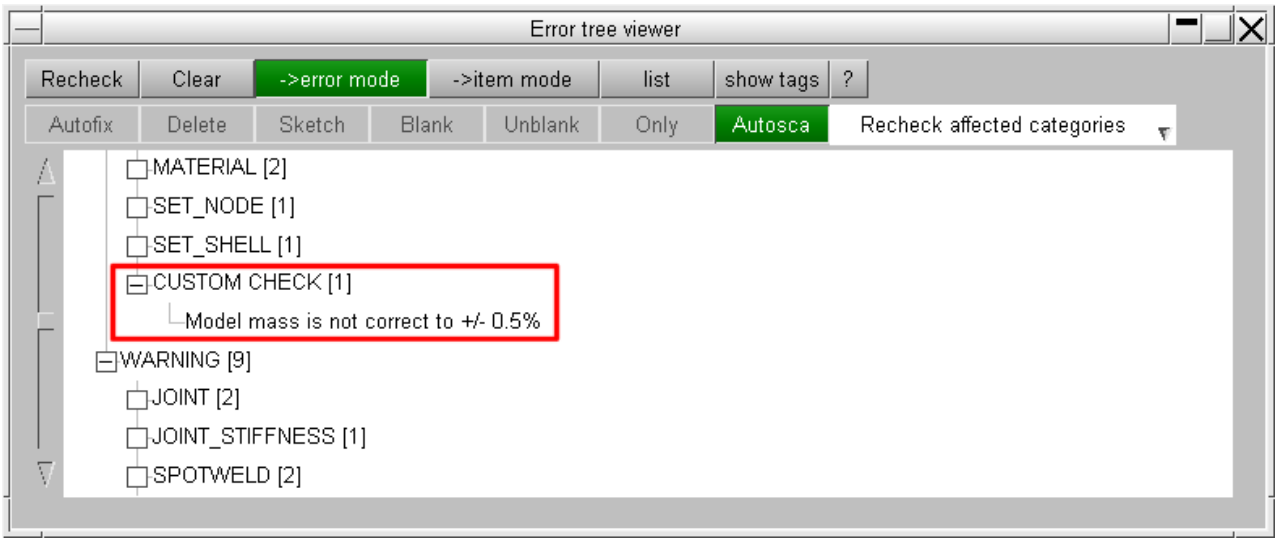


User defined custom check

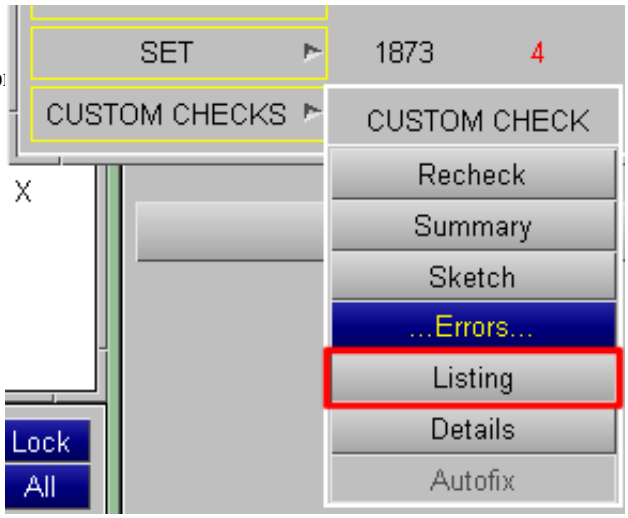
A custom check will have 2 arguments passed to it automatically in the arguments array. The first argument is the script name and the second argument is the Model object. For example we may want to check that the mass of the model is 1500kg +/- 0.5%. We could do this by writing the following in a script `custom.js`.

```
// arguments[0] is name of script
var m = arguments[1];   // arguments[1] is model
mass = m.Mass();
if ( Math.abs((mass-1500)/1500) > 0.005)
    Check.Error("Model mass is not correct to +/- 0.5%", "Mass is "+mass+" but
should be 1500kg");
```

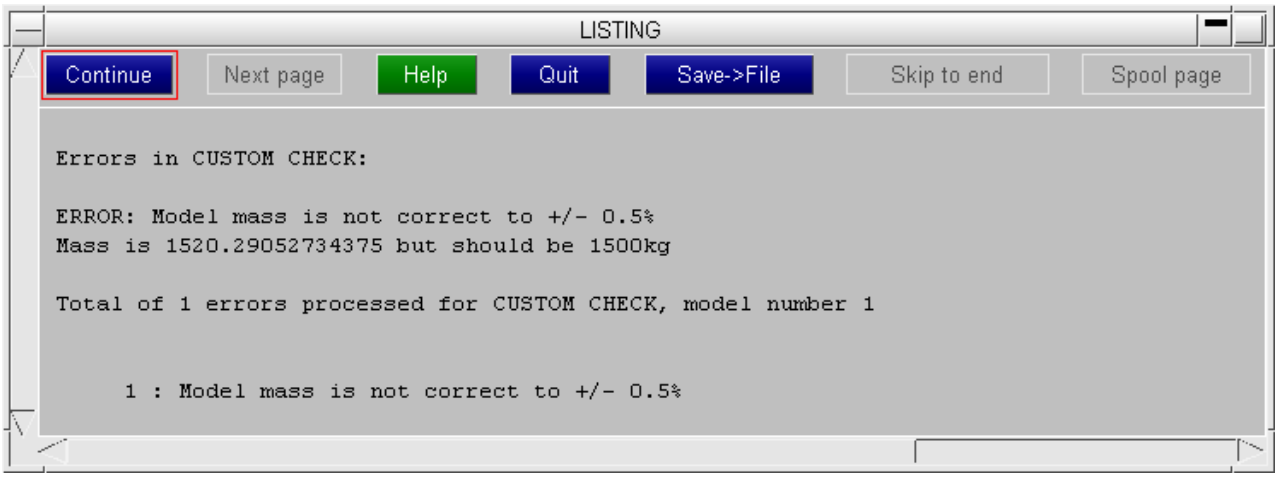
You can trigger a warnings and/or an error by using the static methods in the `Check` class `Check.Error()` and `Check.Warning()`. These errors are shown in a new branch of the error tree called **CUSTOM_CHECK**. For example, the output from the above check is shown below.



The above example also shows that you can pass a detailed message as a second argument to the `Warning()` or `Error()` methods. Currently this is not available in the error tree. To see it you have to use the popup from the other check window and select **Listing**.

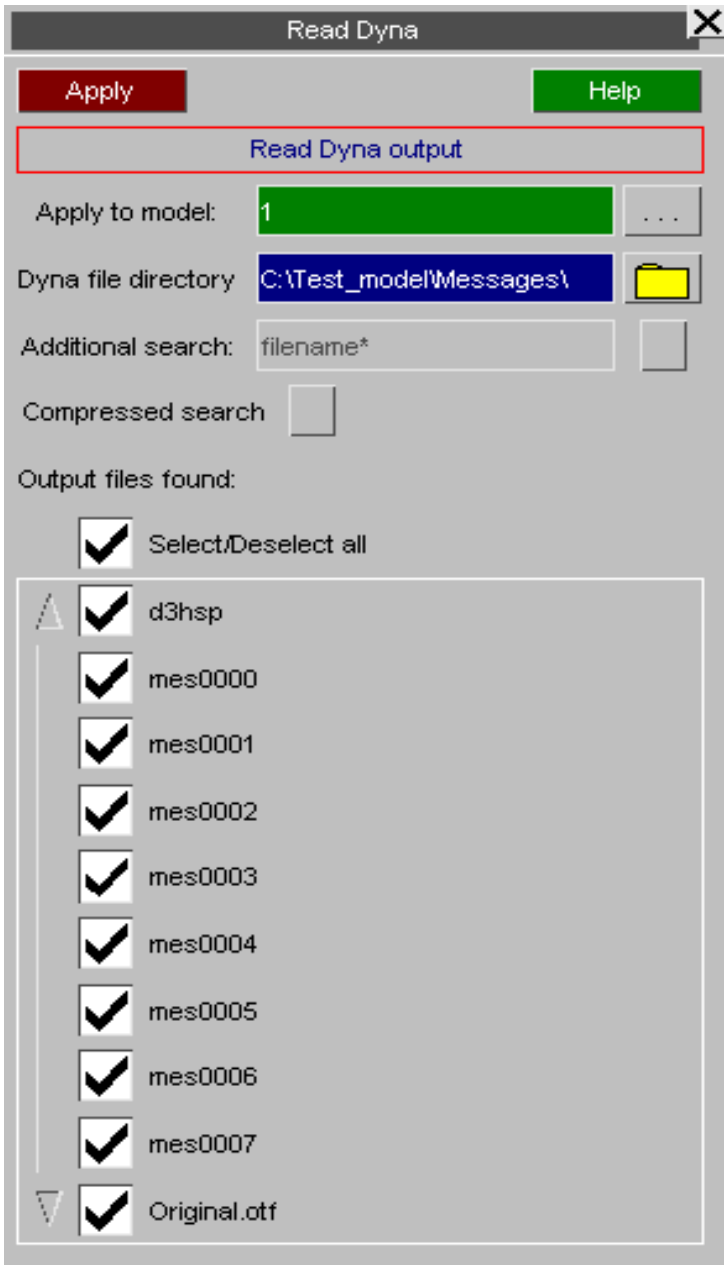


The detailed message is then shown in the listing window.



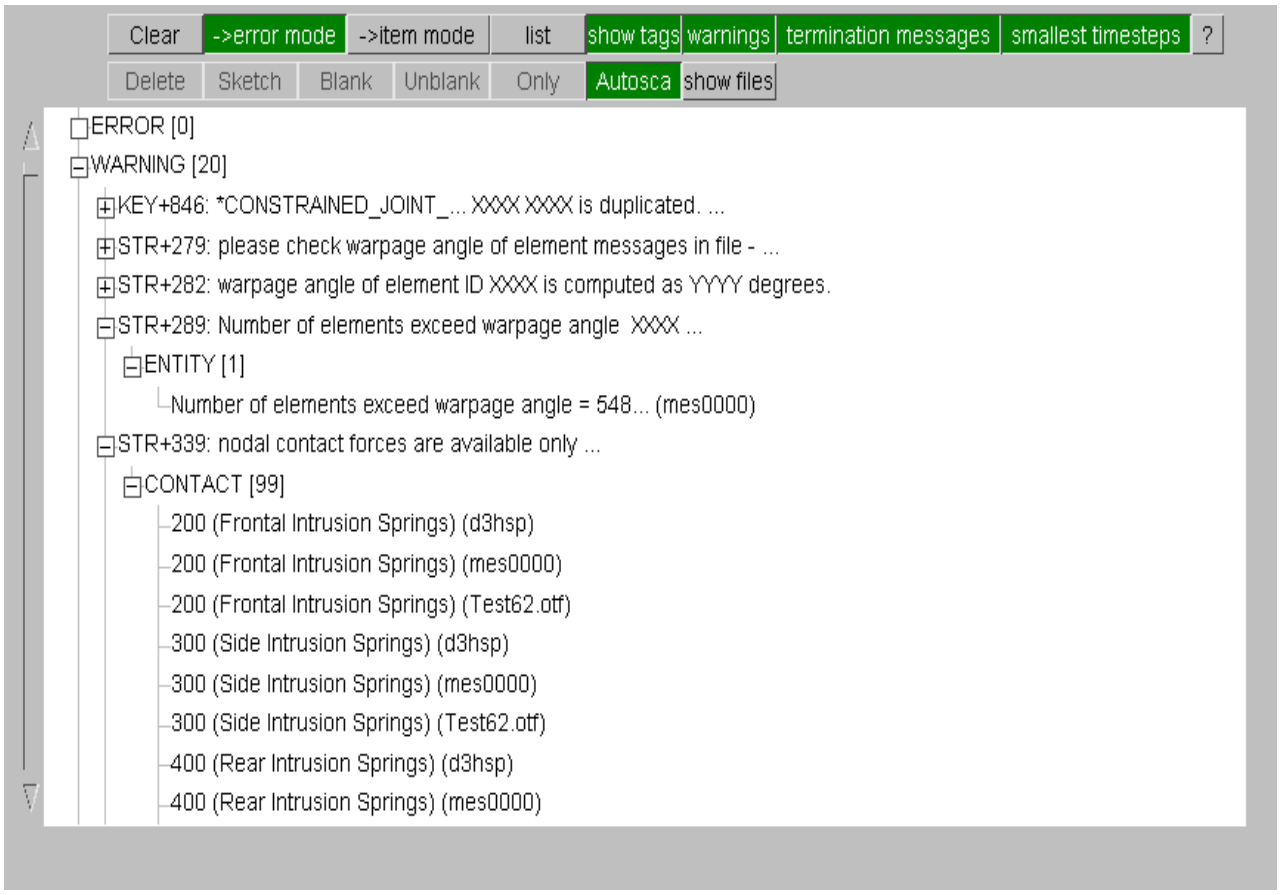
3.9.7 Reading LS-Dyna output error files

It is possible to interrogate the output files produced when LS-Dyna is run for errors and warnings and display the associated entities within PRIMER. This functionality is accessed by clicking the **DYNA Output** button which displays the following window:



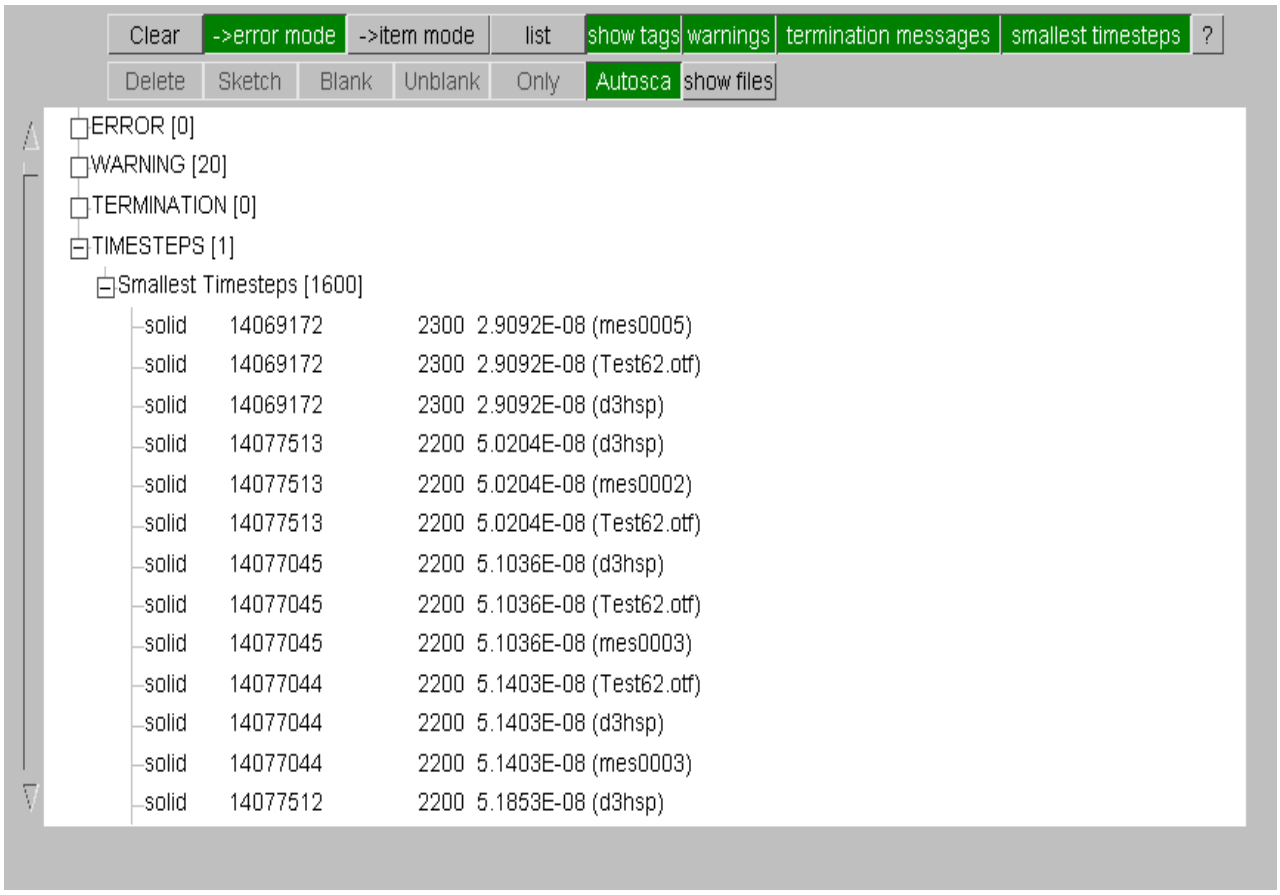
By default the Dyna file directory field is populated with the directory containing the model read in to PRIMER. If more than one model is loaded, use the file selector to browse to the directory in which the LS-Dyna output files are located, alternatively type the path in the Dyna file directory box. On specifying an output directory, any output files with a filename of the format *.otf, *.prt, d3hsp, *.d3hsp, mes*, or *.mes* will be listed and automatically selected for viewing in the tree view. Different filename formats can be found via the Additional search, which can include wildcards ('?' represents one character, '*' represents any number of characters). This additional search can be set via the text box and turned on or off via the adjacent tick box on the panel; or with the preferences 'additional_dyna_output' and 'additional_dyna_output_search', respectively. LS-Dyna output files that have been compressed as .zip or .gz files can also be found and read. Compressed files are only found when the Compressed search tick box is on (also set via the preference 'compressed_dyna_output_search'), which searches for filenames of the format *.zip and *.gz. Reading LS-Dyna output error files is most useful when the corresponding model is loaded so the entities associated with the errors/warnings/termination messages can be loacted and manipulated. The corresponding model should be entered in the optional Apply to model text box either directly or using the model selector button. If Apply to model is left blank only output file text will be displayed in the tree view.

If at least one output file is ticked the **Apply** button is enabled. When clicked, provided the Dyna output reader XML file is correctly found, a tree view of Errors, Warnings, and Termination Messages will be displayed.



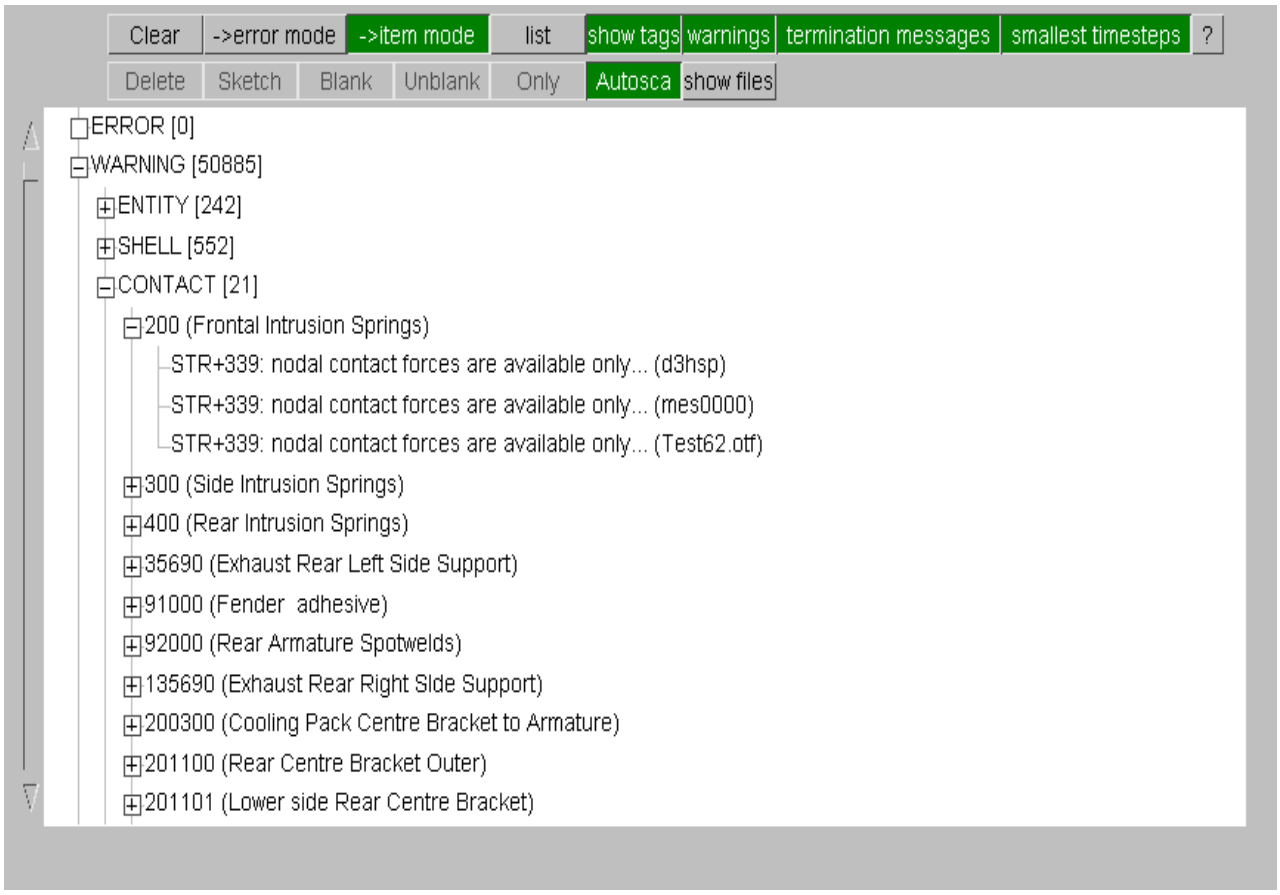
As for the check tree, categories can be expanded or contracted using the small box to the left of the headings. Once the categories have been expanded a number appears to the right of the heading showing the number of sub-categories directly underneath this level. The **show files** button (off by default) will add an extra layer to tree so the errors/warnings/termination messages are displayed by output file. The **warnings**, **termination messages** , and **smallest timesteps** buttons toggles the warnings, termination messages, and timesteps branches, respectively, on or off.

By default the Dyna output tree viewer opens in **->error mode** (as shown above). In this mode the tree branches can be expanded to view at the first level, a generalised version of the error/warning; at the second level the entity type(s) to which the error/warning pertains (if known/relevant, otherwise 'ENTITY' is shown); at the third level a list of entity labels referred to by the error/warning (if found) or a shortened form of each specific error/warning message. Hovering over any shortened messages will display the full message text.



For the timesteps branch, the **->error mode** appears slightly differently (as shown above). This branch can only be expanded as far as the second level, at which the entity specifc message is shown. Hovering over the second branch will give column titles for the data displayed in the message. Additional operations and right-clicking (as described below) can be performed on this second level.

In **->item mode** (shown below) the tree branches can be expanded to view at the first level the entity types that have errors/warnings associated with them; at the second level a list of entity labels that are referred to by one or more errors/warnings; at the third level a shortened form of each specific error/warning pertaining to that particular entity.



Various operations are available once an entity label or type has been selected. Multiple selections are possible using the shift and control keys (similar to the part and check tree behaviour). Viewing operations can be utilised through the buttons at the top of the panel to **Sketch**, **Unblank**, and manipulate the model display. **Autosca** will autoscale the model display after an **Only** operation. **List** will give a concise summary of the errors, warnings and termination messages in a text box. The **show tags** button toggles on and off the LS-Dyna error/warning tags, e.g. STR+339.

Right-clicking on an selection brings up a popup (the figure shown above). Here, if possible the selection can be **sketched**, **deleted**, and viewed in different ways. The user can also **edit** the entity (if a single entity has been picked) via an edit panel or a **keyword** editor or look at its **xrefs** within the model using the cross-reference viewer in a separate panel. Clipboard functions are also available.

3.9.7.1 The Dyna output reader XML file

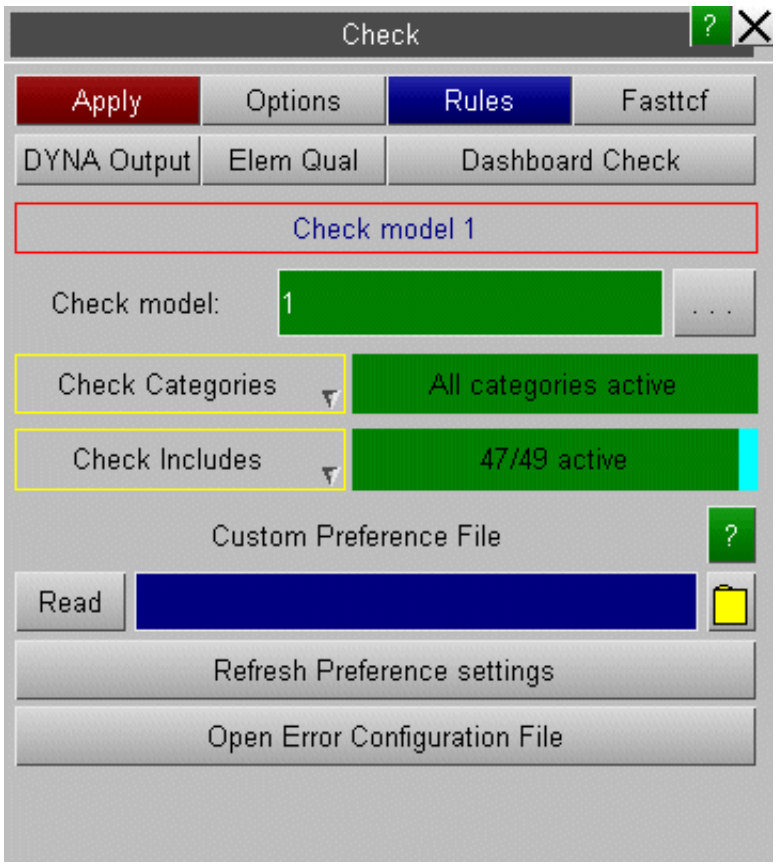
The Dyna output reader functionality relies on an XML file to interpret the errors, warnings, termination messages and other items printed in LS-Dyna produced output files. This XML file, DynaOutputReader.xml, should be located in the primer_library\DynaOutputReader\ folder in either the OA_ADMIN, OA_INSTALL or HOME directory. It contains a list of all the errors, warnings and termination messages produced by LS-Dyna together with the corresponding PRIMER entity (or entities) to which they refer. If an error, warning or termination message is found which does not have an entity associated with it yet, the XML file can be edited in a text editor accordingly. Please contact Oasys Ltd. support if such a situation occurs so the XML file can be updated. Since the XML file is not bound to any particular version of PRIMER, new, improved versions can be distributed between software releases.

3.9.7.2 Batch Dyna output check

The dyna output checks can be performed from the command line and in batch mode by using the syntax: : **DYNA_OUTPUT MODEL <n> DYN_OUT_FILE <filename> APPLY or DYNA_OUTPUT MODEL <n> APPLY.**

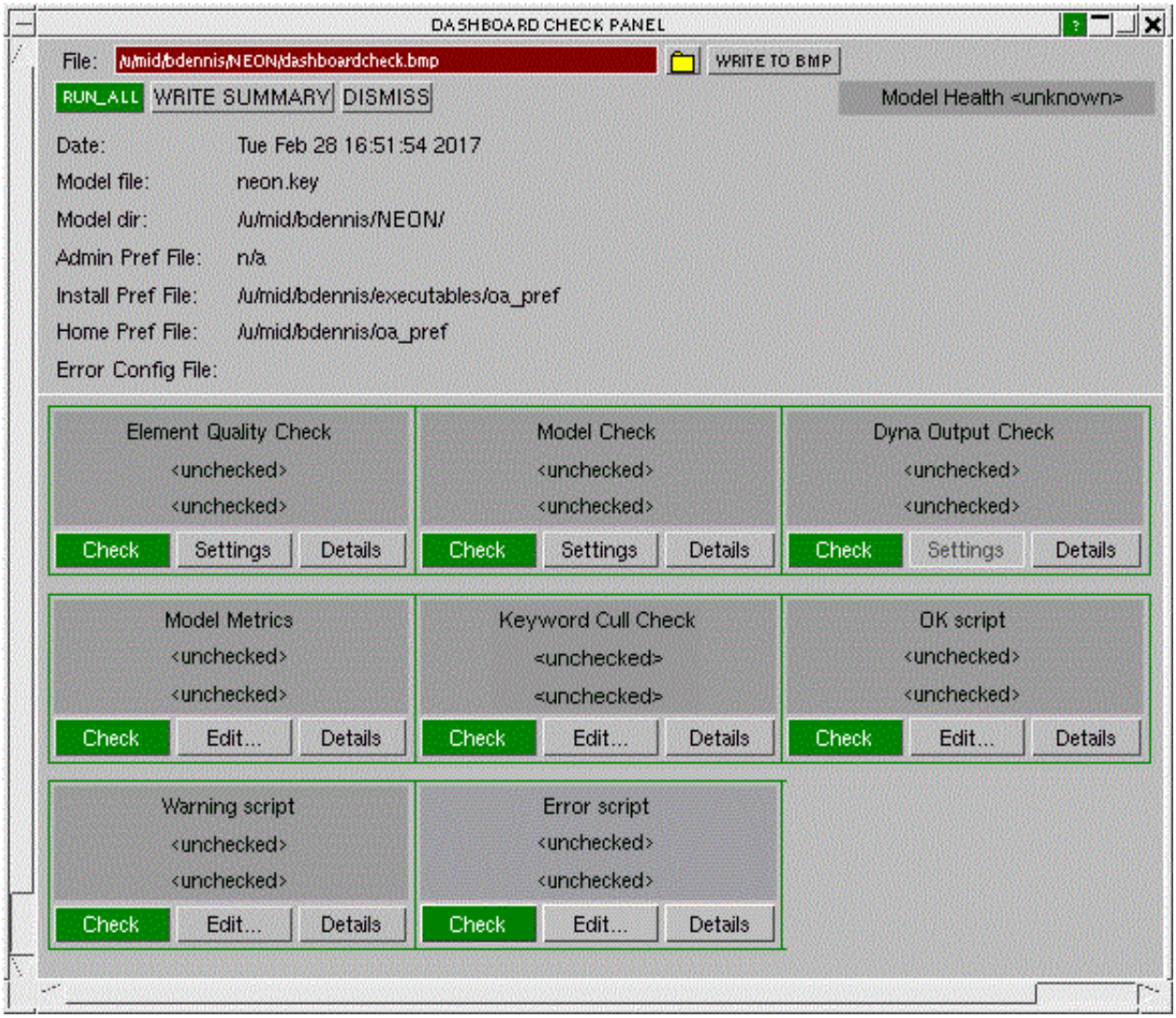
This command will the dyna output check and write out a summary file..

3.9.8 Dashboard Check

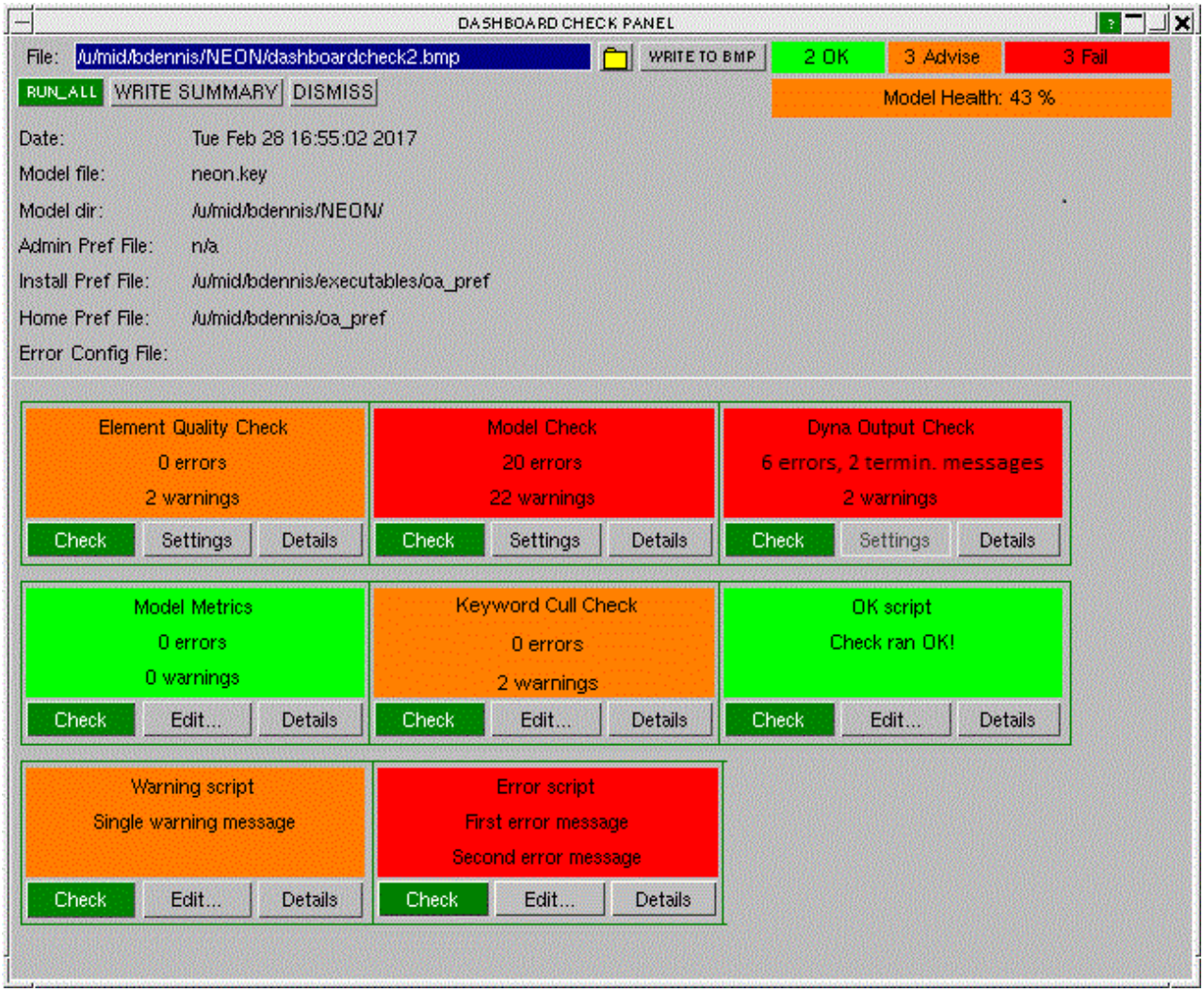


This function is accessed by the **Dashboard Check button** under the check tool. This will display the dashboard check panel which consists of a set of different check tools.

- element quality check - model check but only running the active quality checks (see check > options) on shells, solids and tshells. These include timestep and added mass as well as geometric tests
- model check - all Primer model checks but without quality checks
- Dyna output check - finds otf and prt files for this model (or d3hsp) and any mess000 files and reports errors, warnings and termination messages
- model metrics - some bespoke model checks which are are defined in configuration file /primer_library/scripts/dashboard_checks/model_metrics.config
- Keyword Cull Check - Lists all the keywords/fields which will be culled by writing out the model in the current LS-DYNA output version.
- javascript checks - an unlimited number of checking scripts which reside in /primer_library/scripts/dashboard_checks



The required checks can be activated by pressing **Check** or to do the lot **RUN_ALL**



Settings - will access the check options panel

Details - will give listing boxes of detailed information about checking

Edit - will access text editor for the model metric configuration file or the javascripts for review and modification

3.9.8.1 Model metrics config file

These are located in /primer_library/scripts/dashboard_checks which may be located in the system or home area or cwd.

The **model_metrics.config** file controls the model metrics check. If the file is blank no checks are made. If no file exists, primer will create a template file for you. Each tag uses MIN or MAX to specify and upper or lower bound and ERR or WARN and a given value.

CHECK_MASS
- MODEL_MASS_MIN_ERR, <value> - error if model mass less than given value
- MODEL_MASS_MAX_WARN, <value> - warn if model mass exceeds given value
- MODEL_COFGX_MIN_WARN, <value> - warn if CofG X less than given value
- MODEL_COFGZ_MAX_ERR, <value> - error if CofG Z exceeds given value

CHECK_ADDED_MASS
- MODEL_ADDED_MASS_MAX_ERR, <value> - error if added mass exceeds value
- MODEL_ADDED_MASS_PERCENT_MAX_ERR, <value> - error if percentage added mass exceeds value

CHECK_TIMESTEP
- MODEL_TIMESTEP_MIN_ERR, <value> - error if model timestep less than value
- ELEMENT_TIMESTEP_MIN_ERR, <value> - error if smallest element timestep less than value

3.9.8.2 Javascript checks

Some simple example javascripts are supplied in /primer_library/scripts/dashboard_checks.

These show how the Javascript Check Class (see Javascript manual) can be used to set the check status, configure the headings that appear and add extra messages.

- Check.SetDashboardStatus(Check.ERROR)
- Check.SetDashboardStatus(Check.WARNING)
- Check.SetDashboardMessage("mess1", "mess2")
- Check.AddDashboardComment("message")

The logic which determines how to set these is in the hands of the scriptor.

3.9.8.3 Model Health

It is now possible to compute the overall health of the model using results from all the other dashboard checks. These results can be accessed from a special script called model_health.config.js. This script should be placed along with the other user defined JavaScript files.

User defined model health will be computed using the Javascript and will appear on the dashboard panel below the overall results buttons.

There are special Javascript functions which can be accessed only from this file. These are listed below :

- Check.GetAllDashboards();
- Check.AddDashboardHealth(Message, text colour, button colour);

For more details please look at the Javascrip manual and the example script which has been provided.

3.9.8.4 Batch dashboard checks

Dashboard checks can be performed from the command line and in batch mode by using the syntax: : **DASHBOARD MODEL <n> DASHFILE <filename> APPLY or DASHBOARD MODEL <n> APPLY.**

This command will run all the dashboard checks and write out the summary of all the dashboard checks.

3.10 Operations on models

Once in memory models can be drawn; they can also be translated, rotated, reflected and scaled via the **ORIENT** command. Models may also be deleted, and their contents edited in a variety of ways. These and other operations are described elsewhere in the manual, in particular:

Section 5: **Keywords** Describes how to create, edit and process *keywords

Section 6: **Tools** Describes how to use the various "tools" in PRIMER

3.11 Viewing models

Models become visible as soon as they have been read in.

The default action when a model is input is to calculate its max/min dimensions, then to display it, autoscaled if necessary, in the current plotting mode: the default mode can be set in the oa_pref file (see appendix XIII for details).

There are a range of commands which affect what is visible, how it is drawn and what labelling takes place. These are described in section 4, but a summary is:

- Model visibility is controlled globally via the "**Mnnn**" buttons under **MODEL** > **LIST** (see section 3.0.1). When depressed (green) a model is potentially available for viewing, when up (red) the model is removed from the view list.
- Classes of entity (eg Shells, Nodes, Constraints, etc) may be made visible and, optionally, labelled using the Entity Visibility controls (see section 4.4). These flag an entity class for display and/or labelling across all models. This panel can be accessed by the shortcut key E, the top bar menu **DISPLAY** > **ENTITIES** or the button **ENT** from the viewing drawing window.

- Any item, or range of items, can be made visible or invisible using the **BLANK** command. "Blanking" may be applied in a hierarchical fashion to models, or subsets thereof, down to individual items. See Section 4.5 for more information on Blanking.

Basic drawing itself takes place in one of three modes: (see section 4.1)

- **LI**ne "Wireframe", with no hidden-surface removal.
- **HI**dden Also wireframe, but with hidden surface removal applied.
- **SH**aded 2D and 3D items are drawn shaded and lit, with 1D and other items superimposed in hidden mode.

In addition items may be **SKETCH**ed on top of the current image. "Sketching" superimposes a wireframe (unhidden) sketch of the relevant items on top of the current image, in an alternate colour and without clearing the current image. Sketching is not affected by the entity switches or blanking settings.

Data-bearing items may also be contoured or otherwise displayed using: (see section 4.2)

- Vector plots (**VECT PLOT**). Arrows or similar symbols, for example of initial velocity.
- Continuous Tone (**CT**) or Shaded Image (**SI**) contour plots, for example of timestep, shell thickness, etc.

3.12 Memory management and usage.

PRIMER stores all data in memory, therefore it must manage memory efficiently despite reads, deletes, merges, copies etc. This is done by allocating chunks of memory by data category as they are required, and returning these chunks to the relevant free list when that data is deleted. For example when a node is deleted the space required to store its data is returned to the "node data" free list for re-use the next time a node needs to be stored, and so on for all internal categories.

However there is some overhead associated with this and, like middle-aged spread, memory consumption tends to grow as more operations are carried out. In addition many create/delete operations will lead to greater memory fragmentation, and thus more page-faults, so the performance of the programme will degrade.

This is not usually a problem with small models, but when you start to manipulate larger models you may experience some performance degradation as you approach the memory limit of your computer. There are some things you can do to alleviate this:

- Don’t have more models in memory than you require at any one time, and perhaps consider writings models temporarily out to disk (and then deleting them from memory!) before reading in new models.
- Try to avoid unnecessary read/delete/read/delete cycles. This will cause a steady build-up of memory consumption, and also increased fragmentation.

- If merging a succession of large models consider doing the job in stages: merge (say) 2 or 3 models, write out the result, then exit PRIMER and re-enter it to do the remainder. This will lead to memory being more organised, and hence give a faster response.
- Try not to run other memory-hungry processes on your computer at the same time as a large PRIMER session.
- There is a small saving to be made by using X-Windows graphics rather than a 3-D protocol, and also by trying to avoid drawing large images with lots of added labels, symbols, shading, etc.

Tests on this release of PRIMER suggest that just to read in and display a typical model each 1,000,000 nodes and elements requires approximately 750 MBytes (figures for 64 bit version).

Actually working with models, and in particular performing memory-hungry operations such a merging, contact penetration checking and spotwelding can push this requirement up to 1.5 GBytes per million nodes and elements; and this figure should be used when estimating the memory required for a workstation.

In practice, the memory needed for a workstation or desktop is more likely to be controlled by post-processing needs than by PRIMER.

The **UTILITIES** > **DATABASE_STATISTICS** button can be used to provide a summary of current memory usage. This is intended primarily for the programmer to use during debugging, but it provides a useful general guide to how efficiently PRIMER is managing its space. The total consumption it reports will always be an underestimate since it only lists memory used for data storage.

To see actual memory usage use:

Under Unix / Linux	The "ps" command. eg "ps -ealf | grep primer", or the "top" command
Under Windows	The Task Manager, Processes tab, Peak Mem Usage and VM size columns

3.13 Include Files

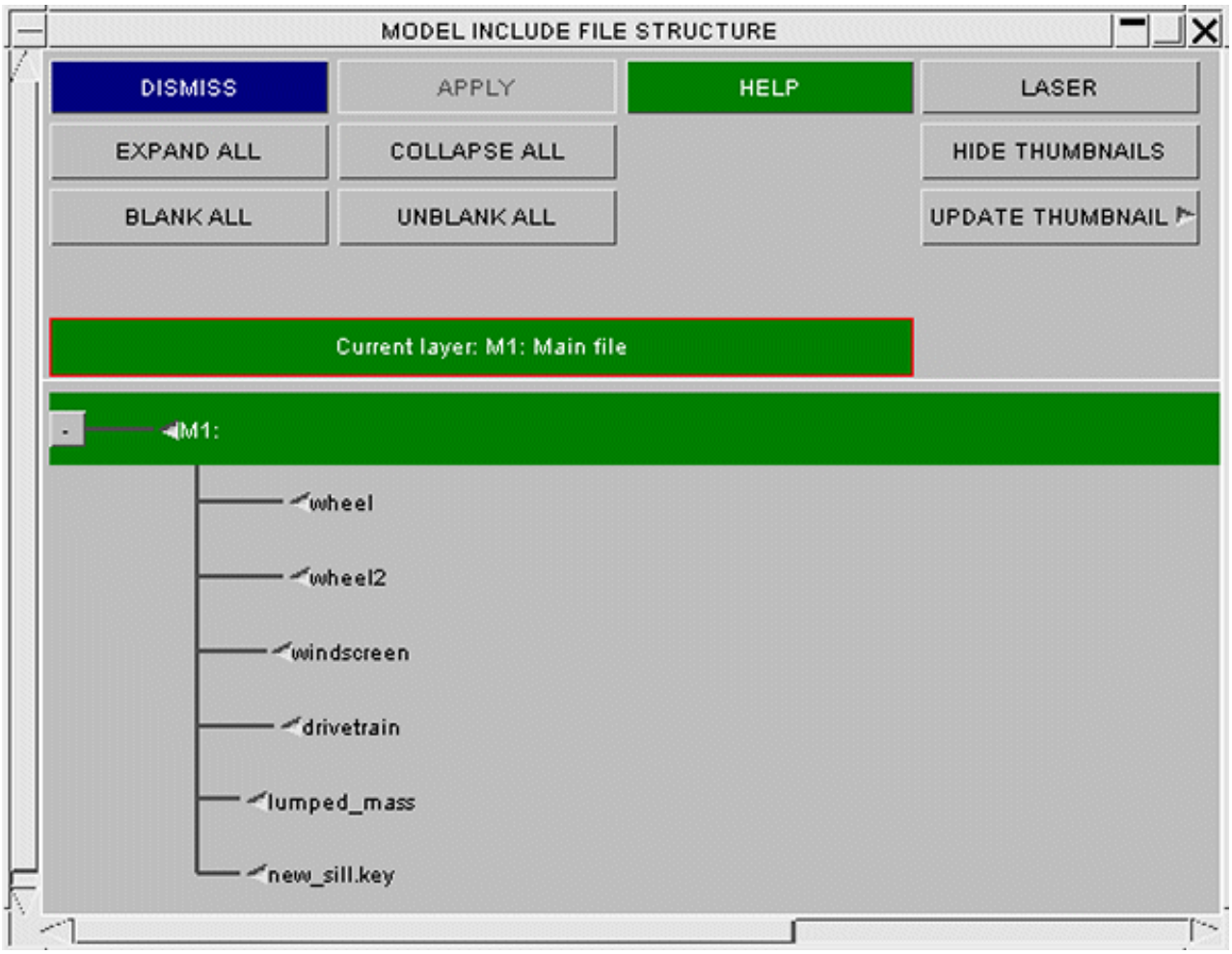
- Include file structure
- Reading in selected include files
- Viewing and managing include files
- Editing comments in include files
- Reading include files after the *END keyword
- Writing out specific include files
- Browse for missing include files

Include files provide a simple way to break up large analyses into smaller, more easily managed files. LS-DYNA supports the following include file types:

Keyword	Description
`*INCLUDE`	A simple include file with no additional attributes
`*INCLUDE_AUTO_OFFSET` (introduced in LS 980)	As above, but labels of nodes and elements in the include file are allowed to overlap with labels of the same items elsewhere in the model. This is intended for wholly self-contained model sub-assemblies for which labels are not important, typically the tolling of dies and formers in metal-forming analyses. During keyword input LS-DYNA will automatically increment the labels in this file to make them unique. Similarly PRIMER will automatically add a label increment where required to make these labels unique during the PRIMER session, then remove them again during keyword output so that the file is unchanged. PRIMER's label increments are very unlikely to be the same as those used by LS-DYNA.
`*INCLUDE_TRANSFORM`	An include file in which labels of items may be incremented by explicit offsets on this card, and which may also have its position in space transformed by a **`*DEFINE_TRANSFORMATION`**.card.
`*INCLUDE_UNITCELL` (introduced in LS 980)	Either defines a "unit cell" mesh in an include file, or specifies the dimensions and mesh parameters of such a mesh that LS-DYNA will create and - optionally - write to a conventional include file.

There are further *INCLUDE_xxx keywords, but these do not contain sub-files of keyword data that are read into PRIMER, and so are not considered here. They may be processed in PRIMER via the **INCLUDE v** keyword popup.

3.13.1 Include File structure



The "**INCLUDE**" button in the tools panel will invoke a tree diagram (shown above in **EXPAND ALL** mode) describing the include structure of the models in memory. Models built of include files are referred to as "**Master models**", the include files themselves are generally "**Component models**" (which will usually be valid LS-dyna models in their own right) and "**connection files**".

Include files offer an easy, robust method to organise your analyses into a file structure, represented above as a tree, of smaller (component) files which can be edited individually.

The "Master keyword file" or "Root file" (at the top of the tree) references all the include files which reside at the first layer of depth. These files themselves may then contain include files at second layer of depth, and so on.

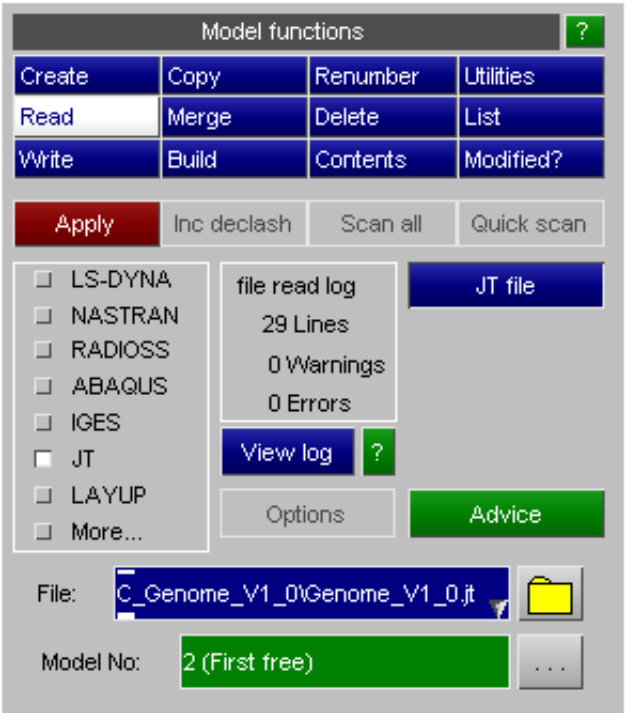
The Include files can exist in different directories to the Master file allowing the user to organise the them with flexibility (e.g. under different directories in a database). On keying out the model each include file is referenced after its *INCLUDE statement, either with the **full path** or with the **path relative to the master file**. See **MODEL > WRITE**.

3.13.2 Reading in selected Include Files

If Master Model already exists

If Master Model is to be created

3.13.2.1 Master Model already exists



From the Menu, select **Model**, **Read** and use:

- **Scan all** to look for all include files, including those "nested" as include files within include files.
- **Quick scan** to look only for include files in the master file.

This scans the input deck. "Scan" in this context means look only for include file information, but don't actually import any normal keyword data into a model.

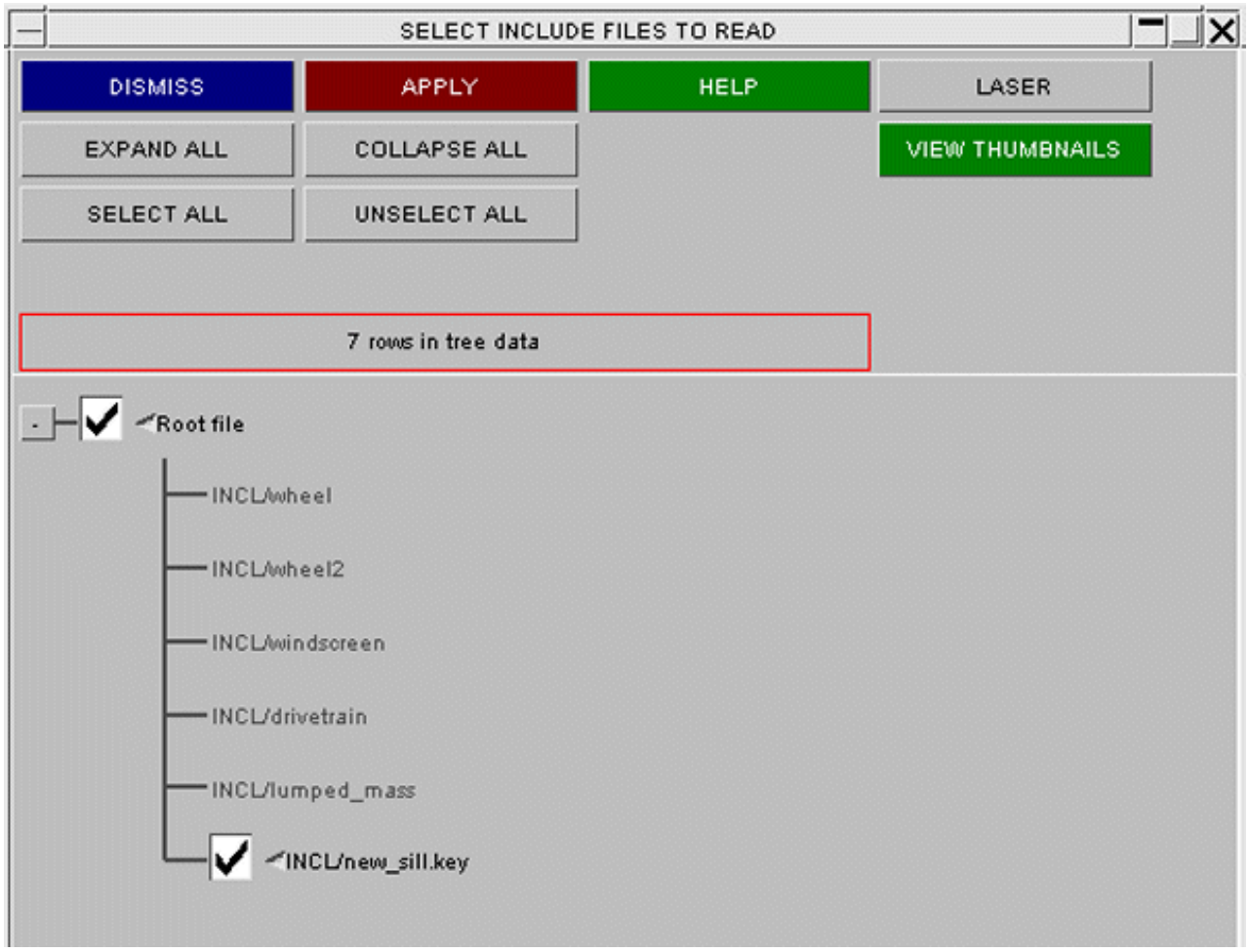
Scanning is much faster than reading since no internal computation or storage allocation is required; normally it is limited only by the disk or network speed of the machine.

Once complete this operation maps the panel below which allows you to control what is to be read in.

In order to view all the Include files present in the Master Model, Select **EXPAND ALL**

Select the files you wish to read in (more than one include file can be read in at once) and press **APPLY**. Select options are invoked by a right mouse click on the popup.

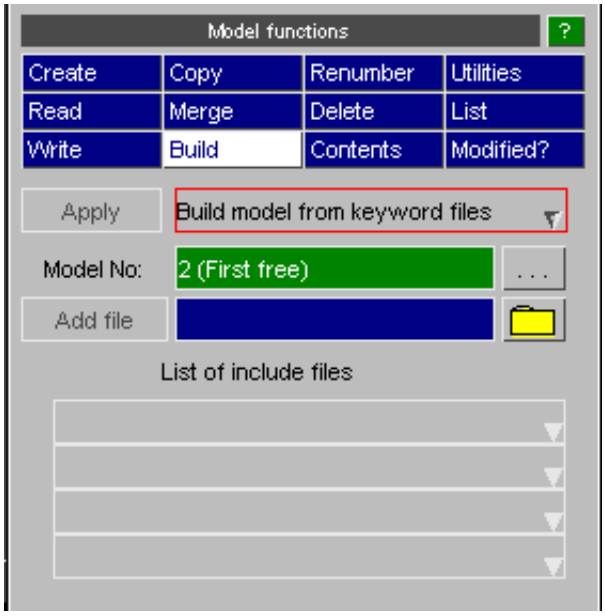
NOTE: When reading Include files into PRIMER it is important to ensure that you read them into the same Model in order to allow all the references across include files to operate successfully



3.13.2.2 Master Model is to be created

From the Main Menu, select **MODEL**->**BUILD**

Simple build from keyword files



If there are **no label clashes** between the include files, a simple build can be used.

The Include files you wish to read in are selected one by one, by inputting the name into the text box and pressing **ADD FILE**. The files will appear in the list.

Once you have selected all the Include files you wish to read in, press **APPLY** and the model will be built with the added files at the first layer of depth. These files may themselves contain include files, thus the method may be used to create a multi-layered include file structure.

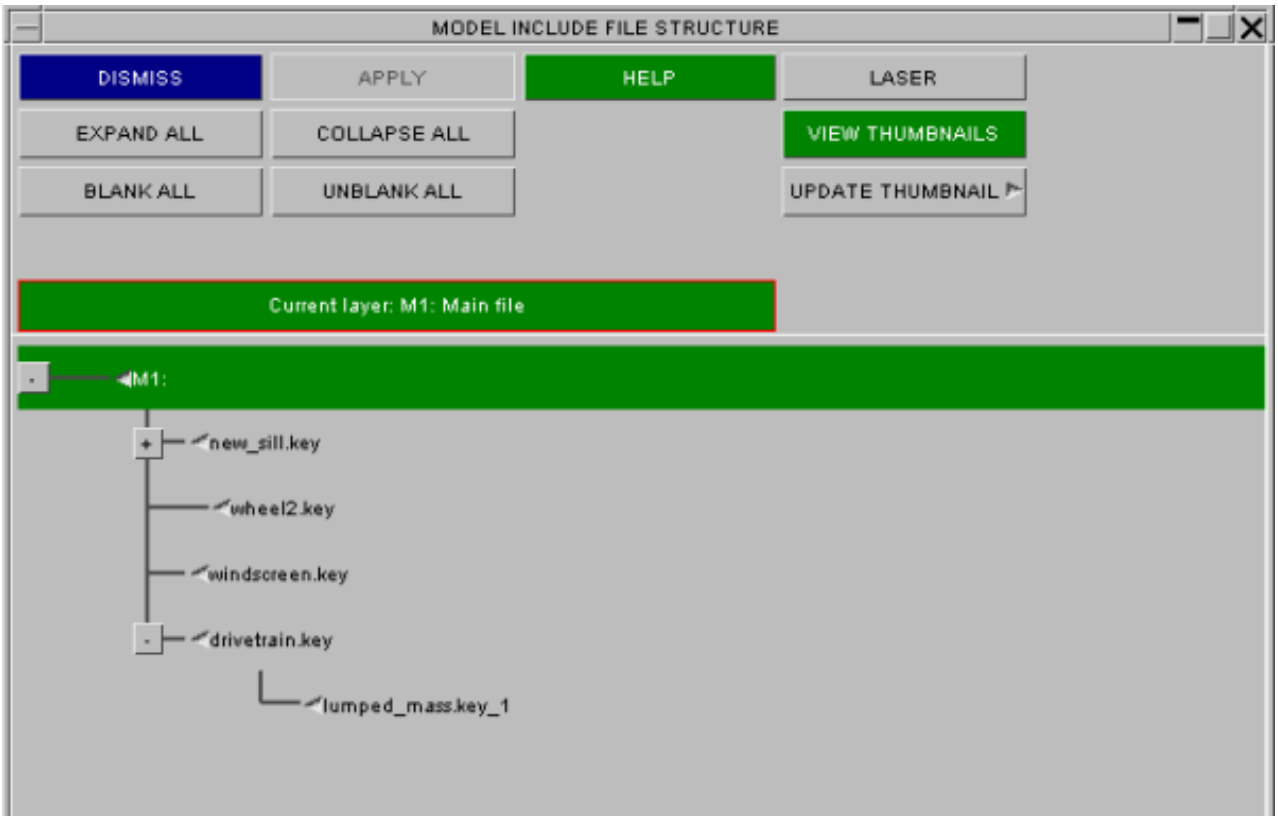
The more complex method of building a model from a database is described in section on Model database.

3.13.3 Viewing and managing the Include file structure

From the Keyword Menu, select **INCLUDE** to access a diagram illustrating the tree structure of the current model.

- Viewing the contents of the Master Model
- Thumbnails
- Adding Include files to a model
- Removing include files.
- Replacing include files.
- Controlling Include file location for newly created entities

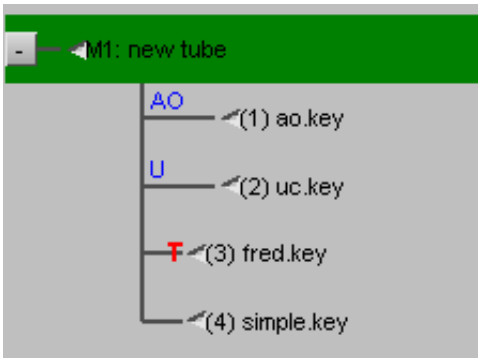
Viewing the contents of the Master Model



In order to display the names of the Include files in all layers of the file, click on the Expand all tab. In order to condense the window in order to just show the Master model, click on the Collapse all tab. In order to expand an individual Include file to see the Include files contained within, click on the grey square to the left of the Include file you wish to investigate

How types of include file are displayed in the tree

The include file tree includes symbols to show the types of individual include files:

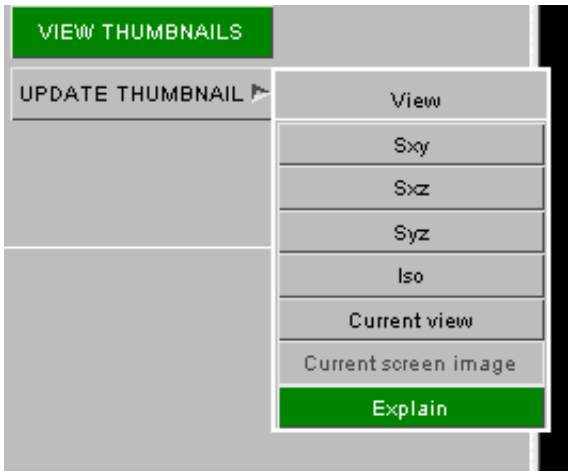


AO	_AUTO_OFFSET
U	_UNITCELL
T	_TRANSFORM
<no symbol>	plain *INCLUDE

Transforms can be suspended within PRIMER, in which case the **T** symbol will have a cross **X** through it, thus:

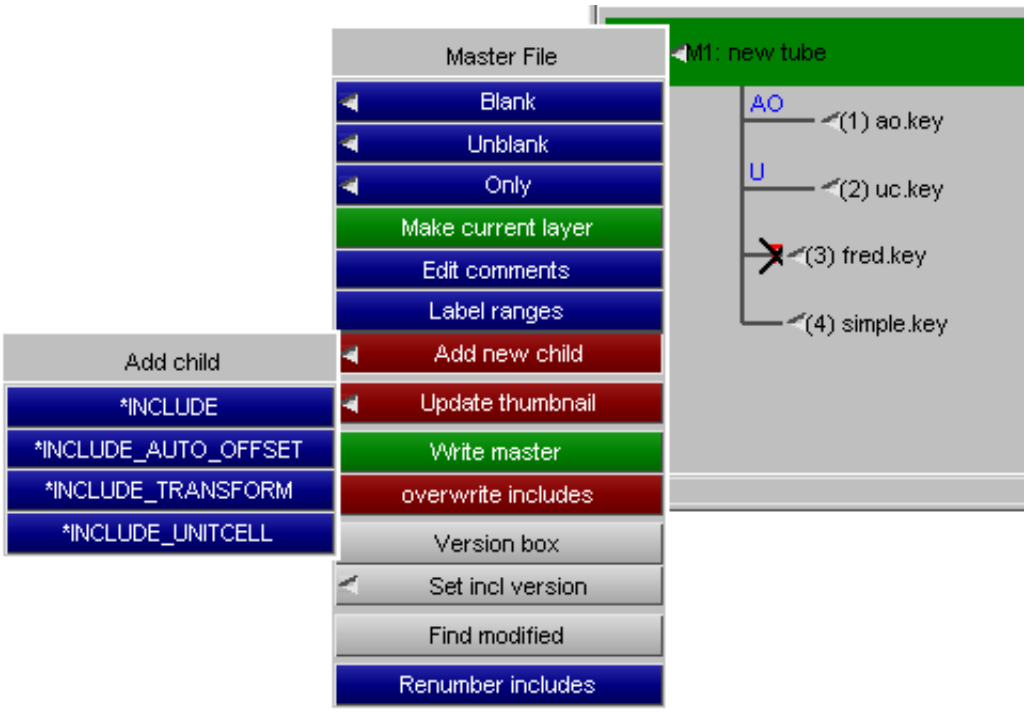
Thumbnails

Thumbnails can be inserted next to the include file names to provide a graphical illustration of the contents of individual include files. To access this capability, right click on the Update Thumbnails tab at the top of the window and a pop-up will come up asking you to select a view. The current option will display the contents of the include file as they currently appear on the screen. Individual thumbnails can be updated by right clicking on the name of the keyword file and selecting the Update Thumbnails tab in the pop-up window. If generated, thumbnails can be turned on and off by toggling the **VIEW THUMBNAILS** button



On keyout, unless **WRITE ANY THUMBNAILS** is clicked off, any thumbnails generated will be written below *END of their keyword file.

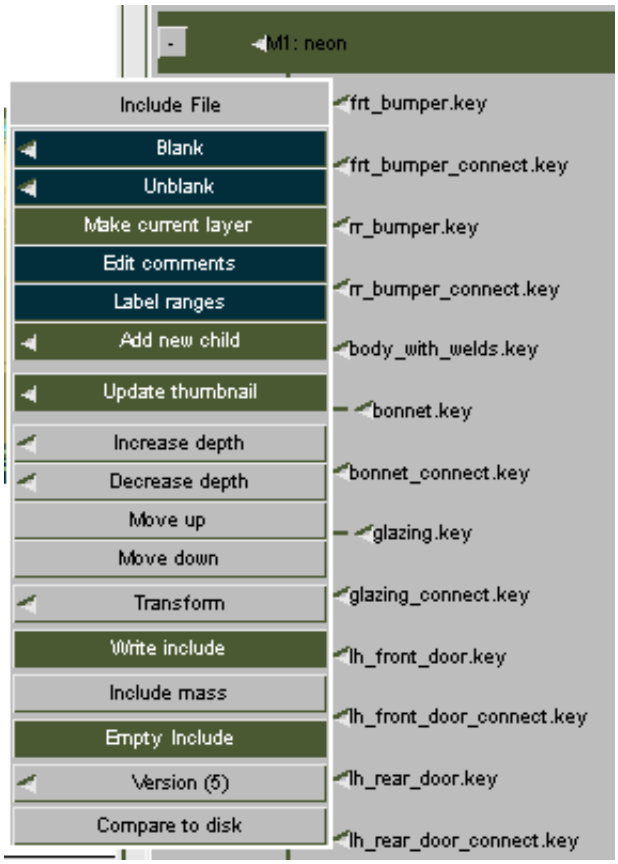
Adding Include files to a model



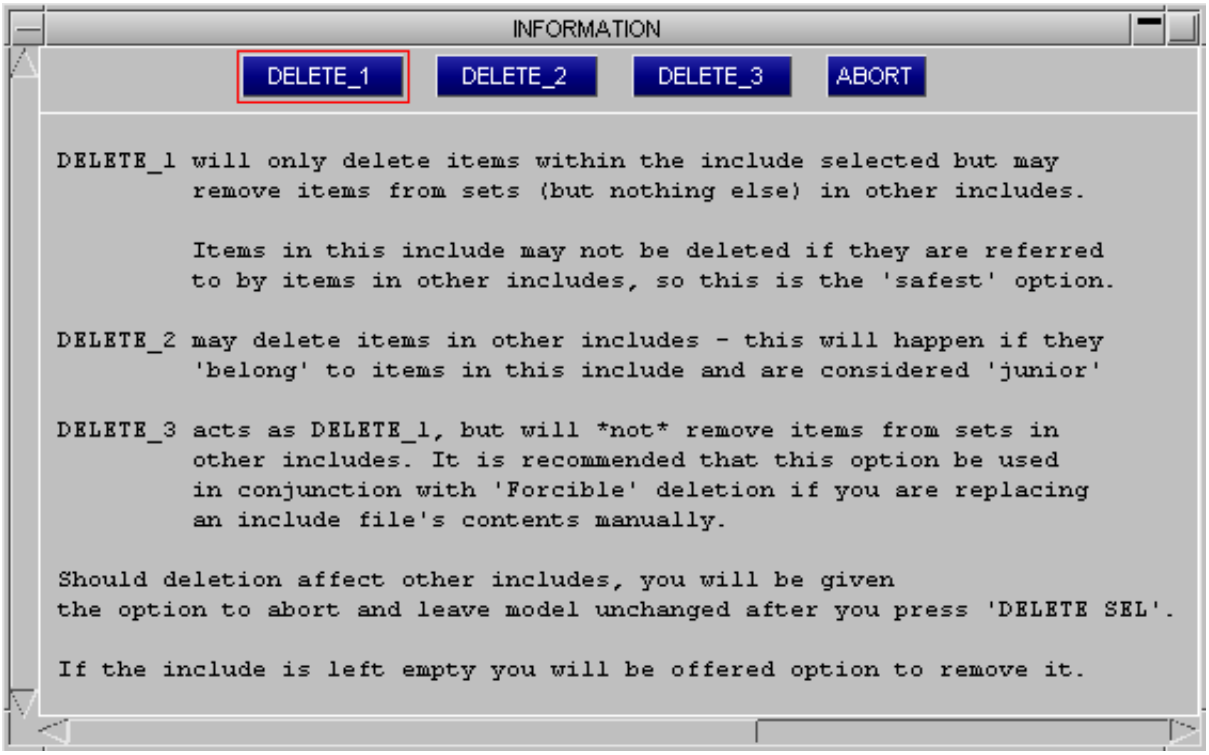
Include files can be added to a model by right clicking on the keyword file you wish to reference the new Include file in and selecting the **ADD NEW CHILD** option in the pop-up. Select the file you wish to add by either inputting the name and path into the box, using the search option by clicking on the question mark, or by accessing the model DATABASE by clicking on the Database tab. If you want to create a new Include file within the model, then input the name and path of the new file you wish to create.

Removing Include file

Off **TOOLS** > **INCLUDE** tree there is an option on the include file dropdown to **Empty Include**.

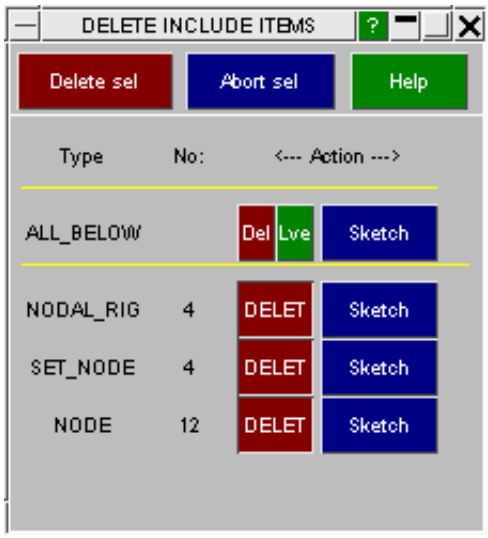


Three deletion methods are available.

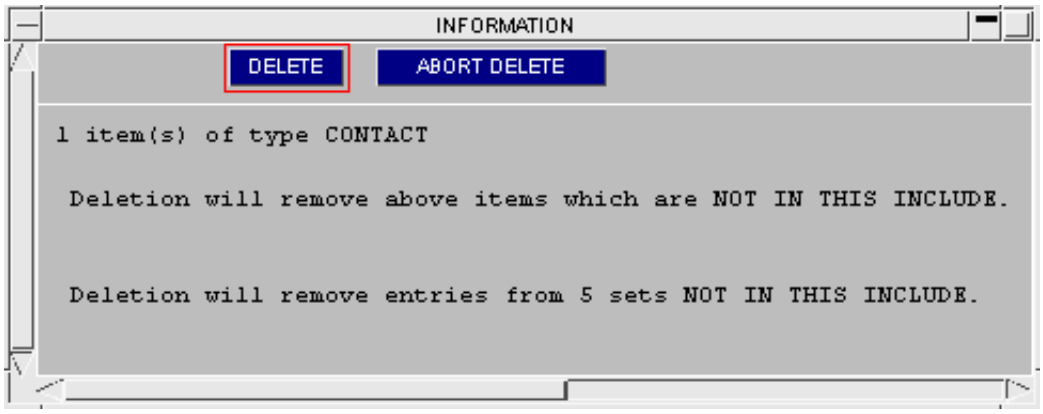


DELETE_1	Is strict deletion which will only try to delete items in the include file, therefore it is the "safest" option. If items in this include file are referred to by items in another file then the items in this include file will be locked against deletion. Examples that might cause this locking are elements in this file belong to a *PART in another file, or a *PART being in a Constrained Rigid Body Merge with a *PART in another file, or a *NODE in this file being part of the topology of an item defined in another file. Many other situations exist, and if file content is locked against deletion you will need to use the green [?] button in the post-deletion summary panel to find out why items have not been deleted. However, if the **Remove from sets** option is active (by default it will be), then this option ***will*** remove of items from *SETs in other include files, and membership of those sets will not lock items in this file against deletion.
DELETE_2	Is "propagate" deletion. This is analagous to selecting the contents of the include, putting them on the clipboard and applying delete. Any items which are junior to those selected (e.g. the nodes of an element) will be flagged for deletion without regard to their include file. This mode is more likely to want to delete items outside the include file
DELETE_3	Is like **DELETE_1**, but items in *SET definitions in other files are ***not*** removed. This option is intended for use with "forcible" deletion, and is recommended if you are replacing include file contents manually.

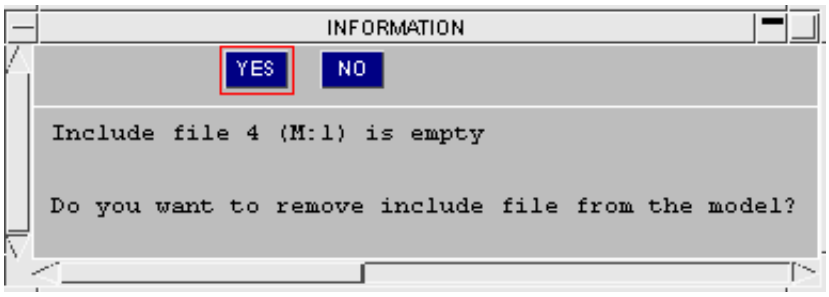
In all modes, the interactive deletion panel will then be activated. Items may be de-selected by using **Leave**. Pressing **Delete Sel** will initiate the deletion.



Before deletion is actually carried out, Primer will detect if any items to go are outside the include file. These will be reported and user given the option to abort the deletion operation. The model will be unchanged.



If deletion is applied, Primer will then check to see if the include file is empty. If so, you will be given the option to remove the include file from the model structure. Alternately you can leave it as an empty include, presumably for later population.



Replacing an include file

You may also replace an include file with a different one. Typical uses are to replace dummy A (say 50th percentile) with dummy B (say 95th percentile).

Internally the steps that are gone through are:
1. Delete existing include file
2. Read in new file
3. Merge into model

This sounds like a statement of the obvious, but consider the case where some content in the include file is referred to by something else in the model. For example say a contact surface defines one side by a part that is in the include file to be replaced: what happens to this cross-reference, and why does "reference to item in include file" not lock the contents of that file against deletion, as would normally be the case when you try to delete an item that is referenced by something else?

Special logic applies when replacing an include file so that:

During step 1 above, "Delete existing include file", PRIMER uses forcible deletion logic, which means that:
- Where something is not referred to it is totally removed.
- Where something ***is*** referred to it is deleted, and replaced by a latent definition of the same label.

During step 2, "read in new file", the file is actually read into a temporary separate model.

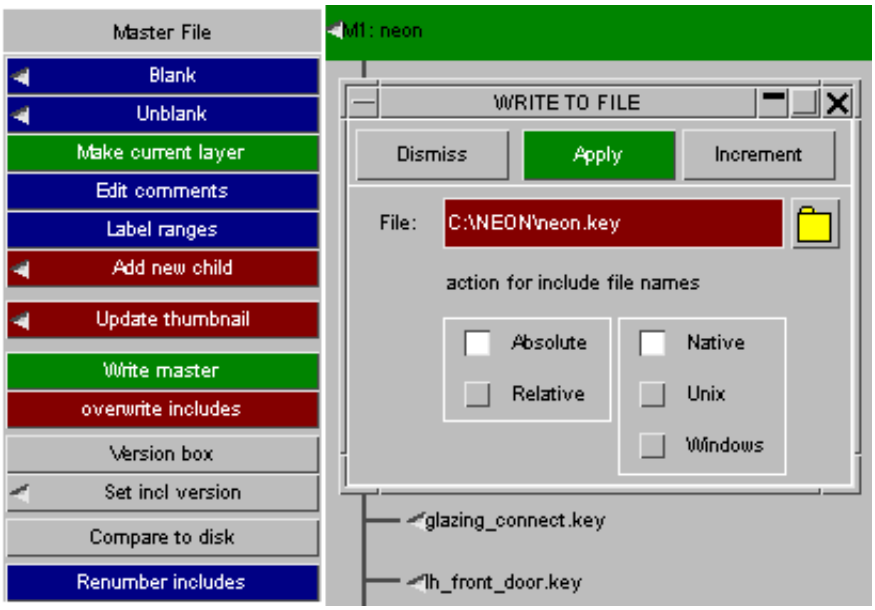
During step 3, "merge into model", the contents of this temporary model are checked for label clashes against items in the target model, and:
- Where no clash occurs the new item is simply copied verbatim.
- Where an existing latent item is found, left "hanging" from step 1 above, the new definition populates this and makes it installed.
- Where a genuine label clash occurs you are offered automatic relabelling options to resolve the clash. Additionally, you can choose to manually renumber items in the include file after the automatic clash fix.

So "replace include" is intrinsically a safe operation, moreover it is designed to handle the problem that the new include will contain a direct replacement for some or all of the items in the old include file. It is the recommended way of swapping components into and out of a model.

Writing Master file

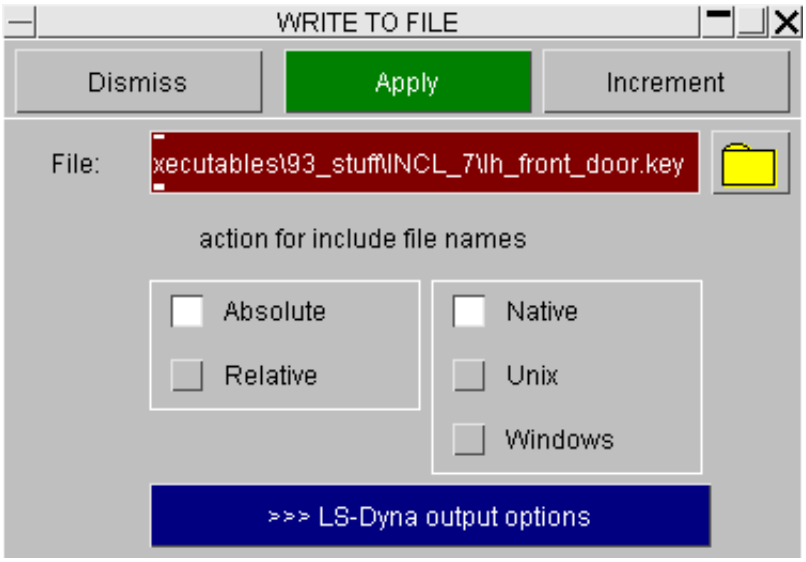
The master file can easily be written from the the include tree using **Write Master**. The panel allows you to set the options for include files.

Increment changes the file name from fred.key -> fred_001.key -> fred_002.key or from fred_1.key -> fred_2.key, etc.

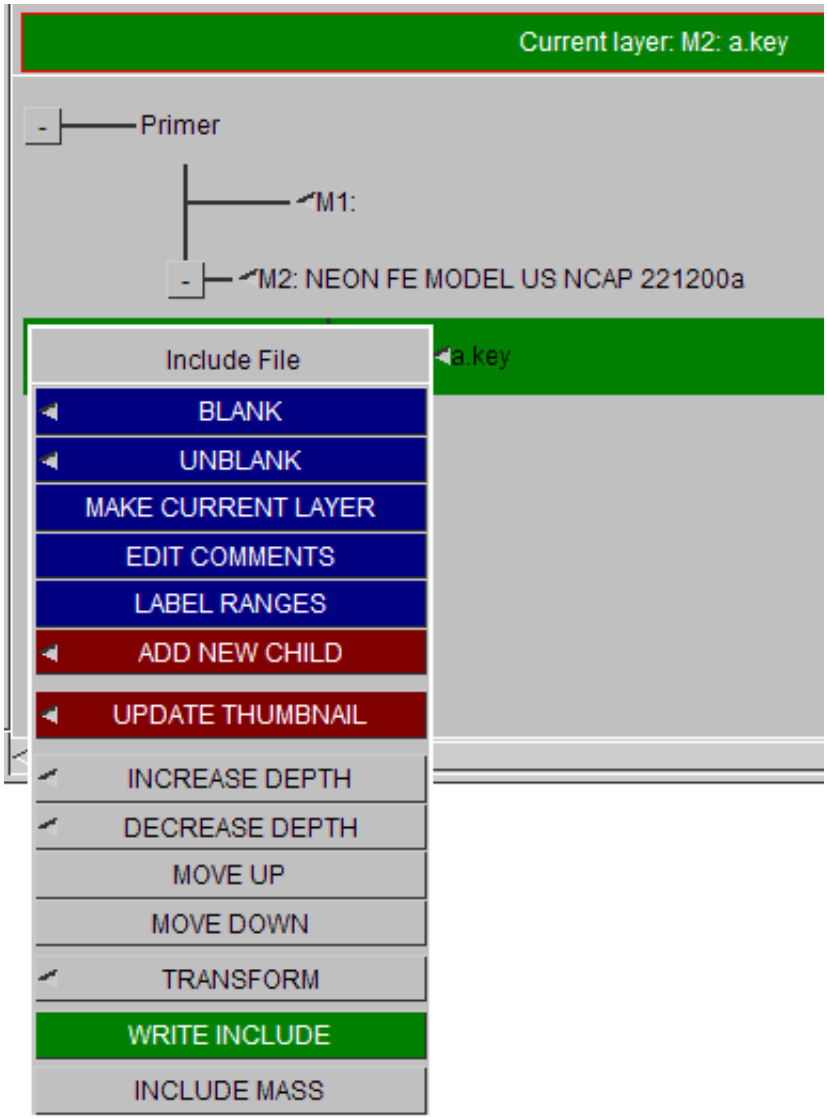


Writing Include file

An easy way to write out a single include file is provided by the **WRITE INCLUDE** option accessed from the include file tree. This initiates a WRITE TO FILE box ready to overwrite the existing file, shown by the red background. No further warning will be given if the user presses **APPLY**. Modifying the file name will set the background to green. Some output options are given on the WRITE TO FILE box. For all the output options, click on **>>> LS-Dyna output options**.



Controlling Include file location for newly created entities



A Green band highlights the **Current working layer** or **current include**.

Any new items made in the model will by default belong to this layer and will be written with this INCLUDE file. Items which are merely modified should always stay in their original layer, unless the user specifies a change of include layer through an edit panel. In order to change the current working layer, right click on the layer you wish to make current and select the **MAKE CURRENT LAYER** option in the pop-up.

BLANK/UNBLANK by include file (or use of the ONLY function in the Part tree) is a useful check for which items are in which layer.

Renaming an include file

To rename an existing include file right click on an include file and select **Rename Include** from the popup. This maps a window allowing you to choose a new name for the include file.

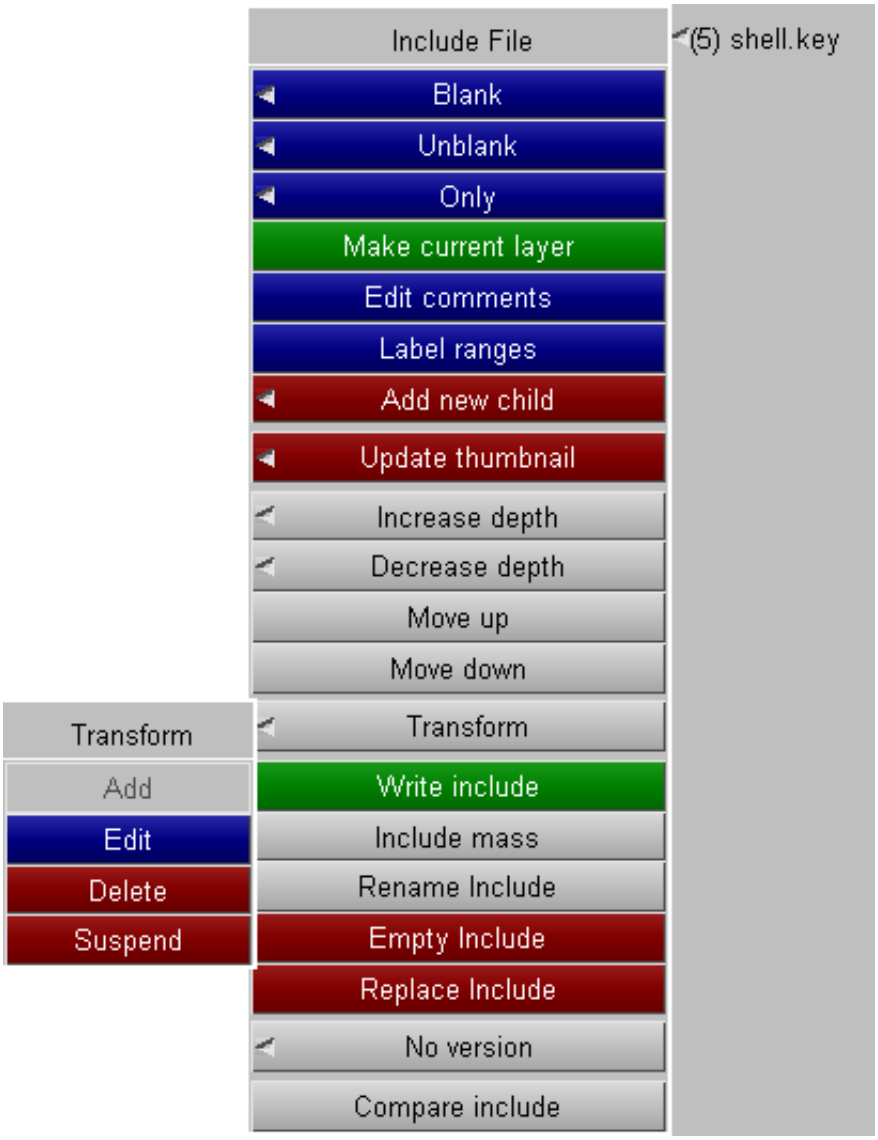
Setting label ranges for an include file

To set label ranges in an existing include file right click on an include file and select **Label ranges** from the popup. This maps a window allowing you to set label ranges for 'general' entity types, for 'nodes, elements, node sets, and constrained nodal rigid bodies' and entities specified in the Detailed entity ranges panel. For further information see the renumber includes section.

Temporarily suspending the transformation on an include transform

Sometimes it may be useful to temporarily suspend the transformation (translation, rotation and scaling) on an include transform. This can be done by using **Suspend** in the **Transform** popup. If an include transform has previously been suspended there will be a **Reinstate** option instead which will reinstate the transformation data.

This can also be done from the part tree.



3.13.4 Editing comments in include files

The **Edit comments** option allows you to change comments that are stored at the top of include files (or the main model). Using this option will start an external editor that allows you to change the comments.

On windows the editor defaults to notepad. On unix the default editor is vi (opened in a new xterm window). The editor that is used can be changed by either:

1. Setting an oa_pref option **`primer*text_editor`** to the editor you want to use.
2. Setting an environment variable **`EDITOR`** to the editor you want to use.

The oa_pref option is checked before the environment variable. Note that the oa_pref option/EDITOR variable **should be set to a filename** and the editor should always start another window.On windows this will always be the case. However, on unix if you just set the editor to **`/usr/bin/vi`** then this would try to open vi in the command/xterm window you started Primer (or the shell) from. This can cause problems (and it would definitely cause a problem if you had started primer by clicking on an icon as there is no xterm window!).

To get round this you can start vi in an xterm window by using a wrapper c-shell script and setting the oa_pref option/environment variable to this. e.g. the following c-shell script will start vi in a new sterm window.

```
#!/bin/csh -f
xterm -title "Edit Primer comments" -e vi $1
```

This technique can also be used if you need to pass other command line options to your editor.

Editors such as dtpad (/usr/dt/bin/dtpad) open a new window so that is OK.

Previous versions of PRIMER would not allow you to do other operations while editing a file. In version 10.0 this restriction has been removed.

3.13.5 Reading include files after the *END keyword

From V11 onwards PRIMER will read include files in the "post END" section of the input deck after *END, for example:

```
*END
$
*INCLUDE
/pathname/filename.key
$
```

Include files read in this way, and their contents, are entirely "normal" and behave just like items read from the main body of the input deck, but this process is subject to some limitations:

- Post-END include files cannot be "nested", thus such a file cannot contain a **`*INCLUDE`** keyword.
- The **`*CASE`** keyword is not supported within these files.
- Any data after **`*END`** in these files will be ignored.

This capability may be generally useful, but it has been introduced specifically to help with the processing of partially encrypted input decks. A typical situation would be that a 3rd party vendor provides a model, typically a crash dummy, in the form of an input deck that is wholly or partially encrypted. PRIMER does not have the key required to decrypt these decks, so their contents are unknown, and it is sometimes the case that such decks will generate errors in PRIMER because key components - typically (but not exclusively) materials and loadcurves - are missing.

Being aware of the problem the vendor will often provide a simplified version of the encrypted data in a separate include file for pre-processing only, and including this file in the input deck resolves the missing items and gets rid of the errors. However when it comes to analysis the input deck will contain both encrypted data and "in clear" simplified data, and the latter must be removed otherwise clashes will occur, and this creates a work-flow problem.

By including the simplified "in clear" input deck after **`*END`** the problem is solved because, without having to edit the input deck:

- During pre-processing PRIMER will "see" the encrypted data but will not decode it, rather placing its data to one side for subsequent output. It will also "see" the simplified data and use this to resolve any missing definitions.

- During analysis LS-DYNA will "see" and decrypt the encrypted data, but because it does not read beyond *END it will not "see" the simplified data, thus clashes are avoided.

However there are some disadvantages to this method. Data inside encrypted blocks is effectively "frozen", and in particular any labels buried in these blocks must not be changed. In addition there may be references from within these encrypted blocks to items elsewhere in the input deck that PRIMER cannot know about, possibly leaving these items vulnerable to being deleted during a "cleanup" operation. Similarly items within the simplified input file may themselves be vulnerable to being deleted. PRIMER does not "know" about these problems and thus cannot protect the user from accidental deletion or renumbering that may result in the input deck becoming invalid during the input phase.

Alternative ways of dealing with encrypted data

PRIMER has some alternative solutions for dealing with partially or wholly encrypted data blocks within a file which may be superior to the above. In brief:

- Special syntax rules permit the initial line of **`*MAT, *DEFINE_CURVE`** and **`*DEFINE_TABLE`** definitions to be in clear while their body is encrypted. This means that their labels are known, and this is often enough to overcome errors due to missing definitions.

- Where **`*MAT, *DEFINE_CURVE`** and **`*DEFINE_TABLE`** definitions are wholly encrypted (including their labels) it is possible to provide alternative in-clear versions after **`*END`** in a special **`*ENCRYPTED`** data block.

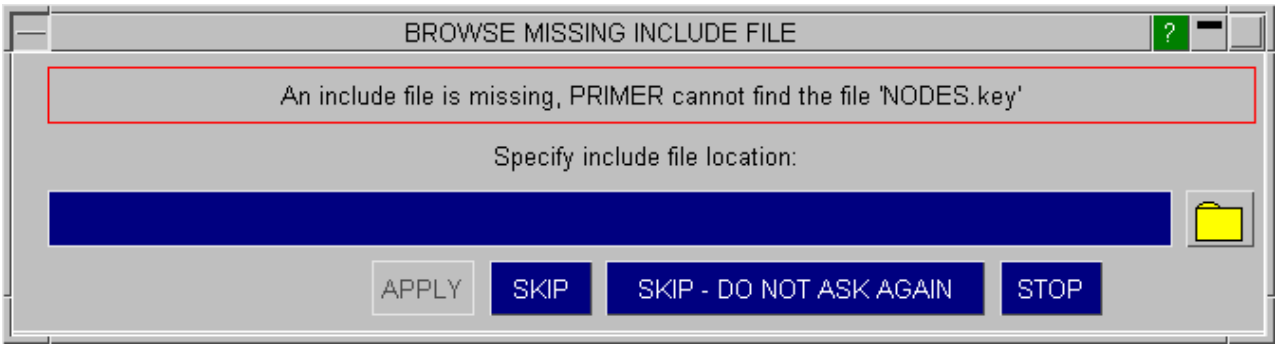
- From PRIMER V13 onwards it is also possible to define **`*PARAMETER`** cards in this post **`*END`** encrypted block to provide alternative definitions for actual parameter values that are embedded inside a PGP encrypted data block.

In both of these situations PRIMER "knows" that the definitions must not be changed and has interlocks to prevent them being deleted or relabelled. So despite being less flexible these solutions may be preferable to using post-END include files.

More details can be found in section 5.1.8: PGP Encrypted Data

3.13.6 Browse for missing include files

If PRIMER fails to locate an include file during a **MODEl >READ** operation, it will generate a popup panel that will permit users to manually locate the missing files.



The following options are available on the popup panel:

APPLY	By default **APPLY** button is greyed out and it will be ungreyed only if a file is specified. The user can specify the file location in blue textbox or browse for the file using file selector icon.
SKIP	If the **SKIP** button is pressed, Primer will continue to read the model and will skip the missing include.
SKIP - DO NOT ASK AGAIN	If the **SKIP - DO NOT ASK AGAIN** is pressed it will not popup this panel again if any missing include files are encountered later when reading.
STOP	If the **STOP** button is pressed, Primer will stop reading the model.

To turn off this feature set "**read_missing_include_file**" preference to FALSE in "**oa_pref**".

3.14 INCLUDE transform

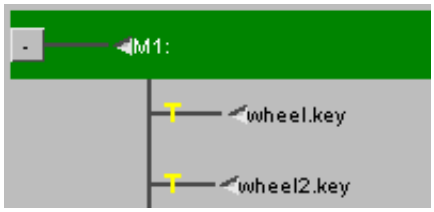
- Display of Include Transform
- Edit of include transform
- Edit of define transform
- converting include files
- keying out include transform

*INCLUDE_TRANSFORM in LS-Dyna provides a means of setting label offsets and applying units conversion to items of an include file.

Additionally the transform *may* reference a Define Transformation statement which will apply a geometric transformation to the include file.

3.14.1 Display of transform

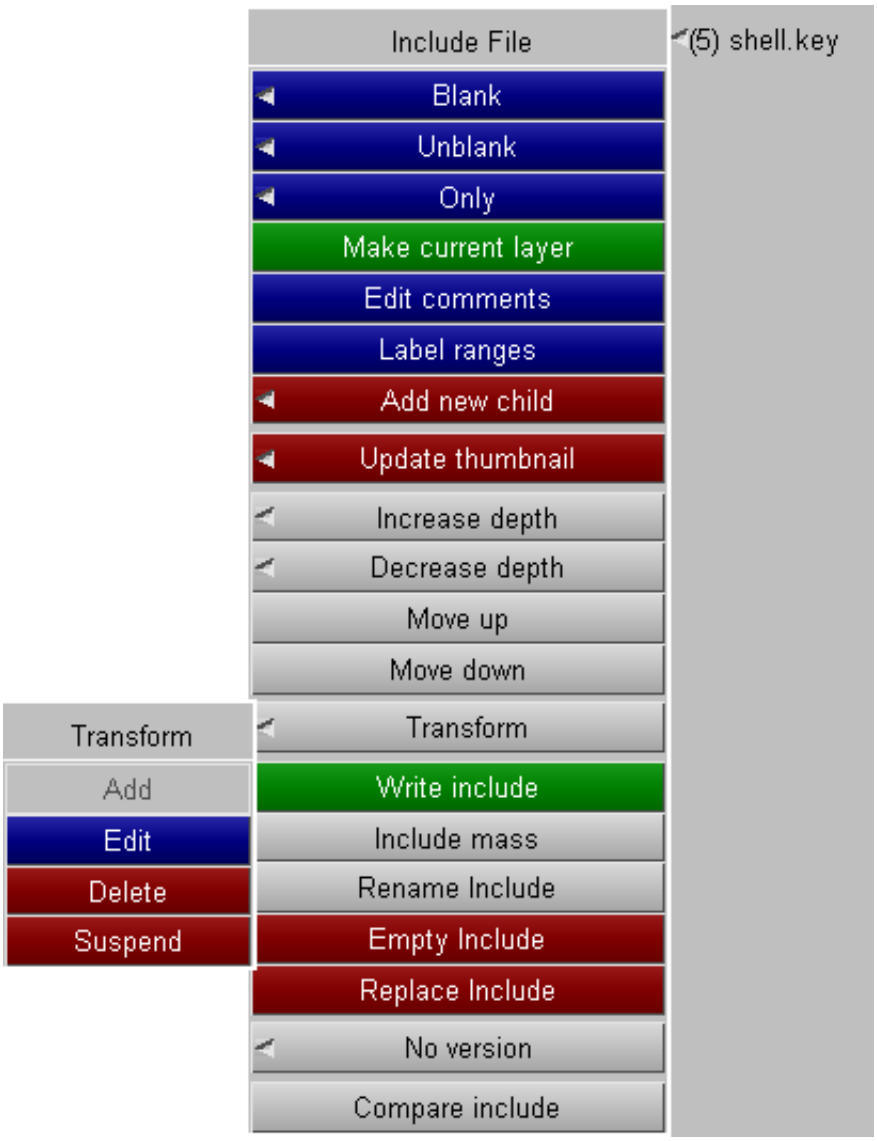
Include files which belong to include Transform statements will be displayed in the include file tree with a prominent "**T**" on their branch.



PRIMER applies the transformations to the model on read-in. Therefore, on write-out the transformations are usually reversed and the transformation definition data restored (but the user may prevent this).In the graphics window, the items of the include file will be displayed *in their transformed state.* That is with their labels offset and their geometry changed as necessary.

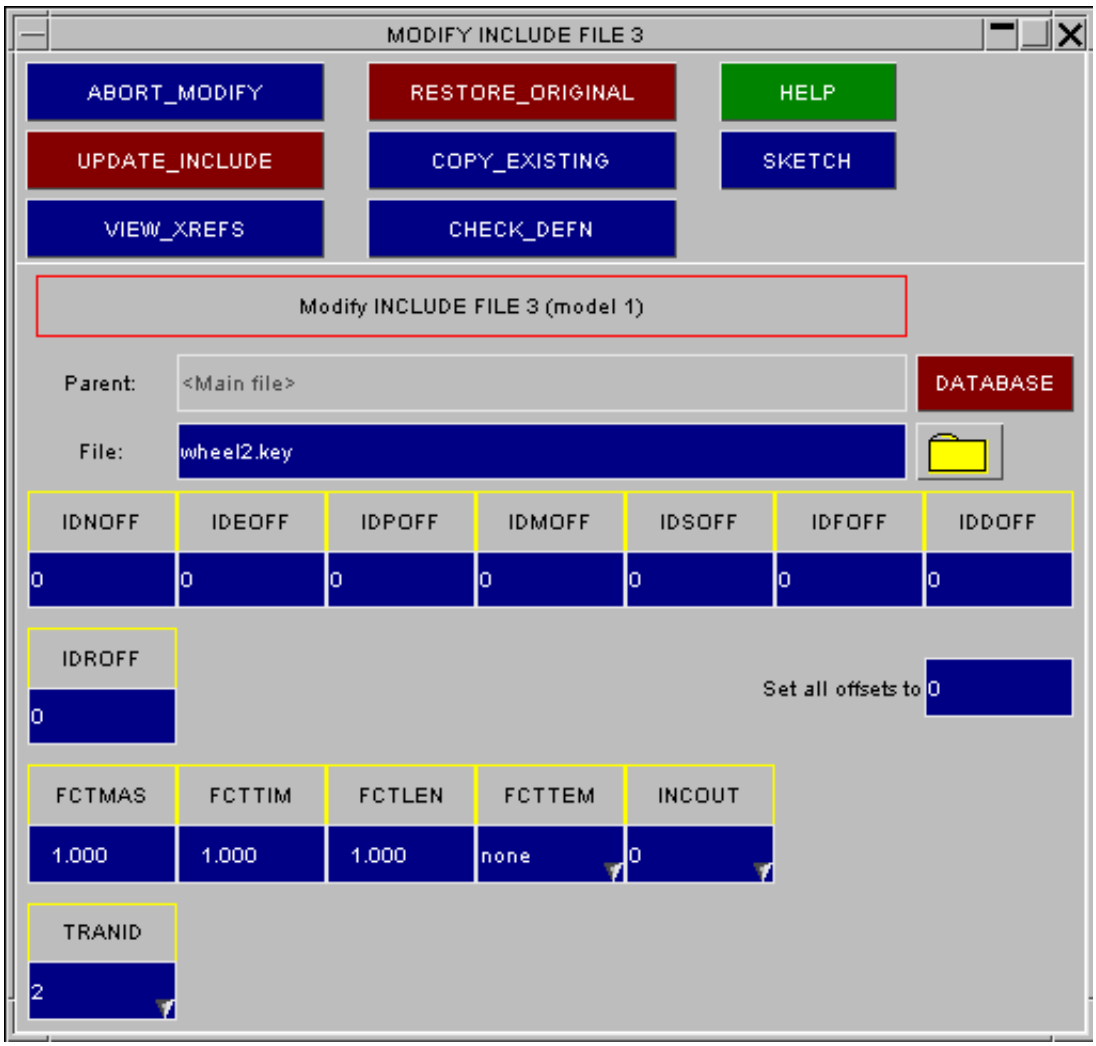
Sometimes it may be useful to temporarily suspend the transformation (translation, rotation and scaling) on an include transform. This can be done by using **Suspend** in the **Transform** popup. If an include transform has previously been suspended there will be a **Reinstate** option instead which will reinstate the transformation data.

This can also be done from the part tree.



3.14.2. Edit of include transform

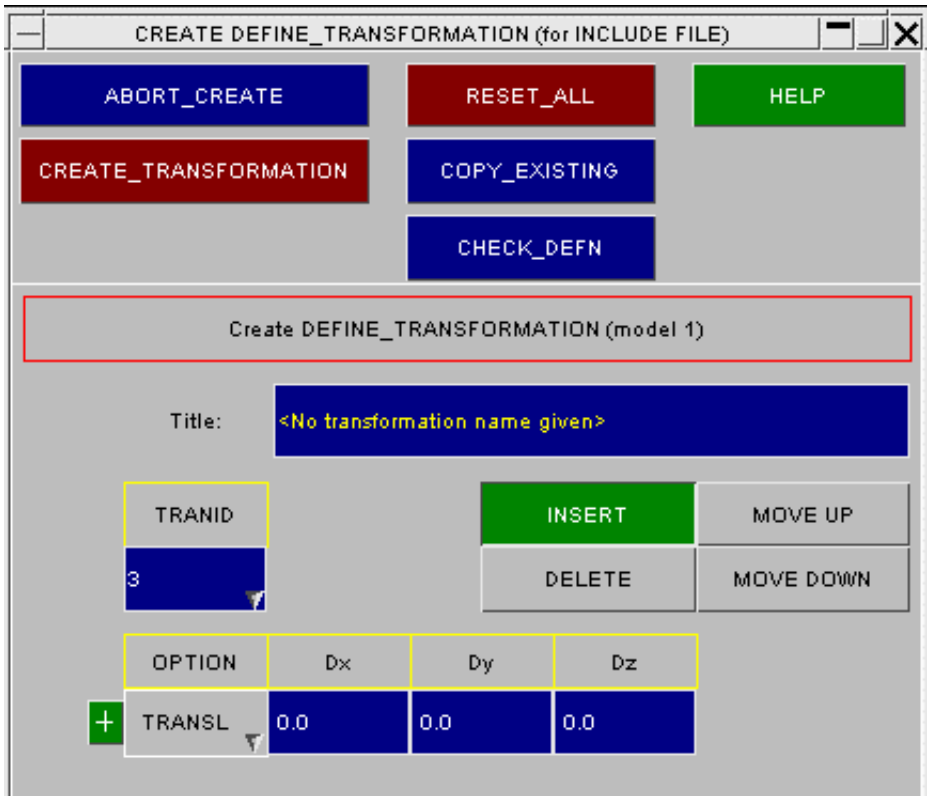
Using the popup from the tree diagram, select **TRANSFORM**->**EDIT** to invoke the edit panel.



This menu allows definition of label offsets and unit conversion to be applied on reading the model, and refers also to a geometric transformation (TRANID)

3.14.3. Edit of define transform

Using the **TRANID** popup, you may create or edit the Define Transformation statement. Any newly created will be be placed at the top of the include file.



Each definition can contain multiple transformations (the options available for which are **TRANSLATE, ROTATE** and **SCALE**. Whilst **INSERT** mode is selected clicking the green [+] button will add another in the row below. **MOVE UP** and **MOVE DOWN** allow the altering of the order in which the transformations are applied. A label in the TRANID filed must be defined before the **CREATE_TRANSFORMATION** button will become active.

3.14.4. Converting include files

The **ADD NEW CHILD** function on the include file tree may be used to add an Include Transform as well as an ordinary include file.

An ordinary include may be converted to transform type (**TRANSFORM**->**ADD**) and a transform type to an ordinary (**TRANSFORM**->**DELETE**).

In the latter case the option is offered of leaving the data in its transformed or untransformed condition .

3.14.5. Keying out include transform

The method of keyout will affect how the data transformation is handled.

- merge->master - the data is transformed
- in sub-directory - as all the transformation calls will be present the data is written in its native state
- select files - the user must select the mode as "**NO-CHANGE**" (leave the data in its native state) or "**MOVE**"(change data to its transformed state, i.e. as if for use with an ordinary include file).

When using the select file mode, it is unclear whether the applicable DEFINE_TRANSFORMATIONS are included or not, as they exist entirely separately from the INCLUDE_TRANSFORM statements. The user must decide.

3.15 MODEL > BUILD

- Building a model using Model Database
- Creating and managing a Model Database
- Reading files using a Model Database
- Managing Templates
- Build of multiple models
- Build from csv targeting files

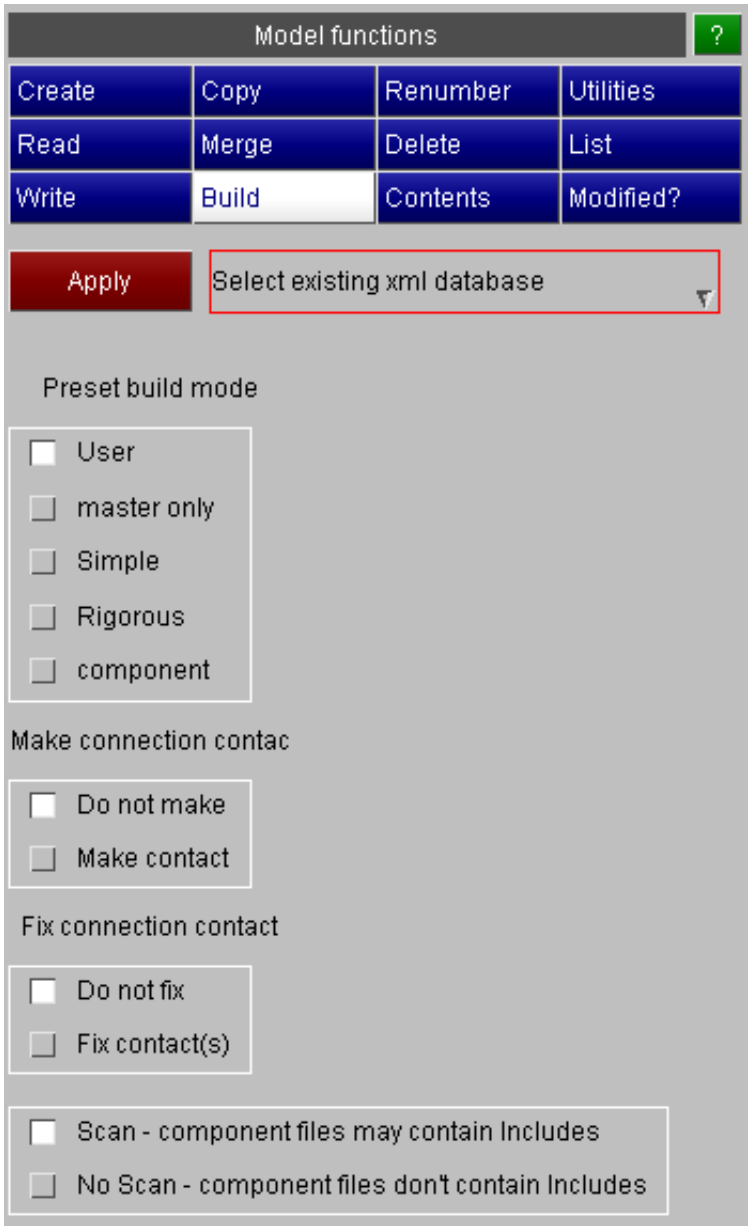
A Model Database is an xml listing of include files and other information that enables you to easily assemble a model from smaller files, typically component and connection files.

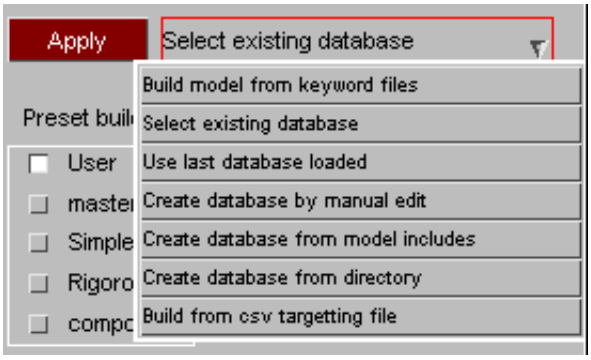
3.15.1 Building a model using Model Database

- Selecting/Creating a Database for model build
- Viewing the Model Database
- Selecting Include files to read
- Building the Model
- Orienting the include files
- Post model build panels
- Writing the Model

3.15.1.1 Selecting/Creating a Database for model build

In the menu, select **MODEL**->**BUILD.** The default is **use last database loaded** which will on its first call revert to **select existing database**.





NOTE: this procedure is to be distinguished from **MODEL**>**READ**>**DATABASE** (see section **MODEL**>**READ**) in which each selected model is read into a separate file.

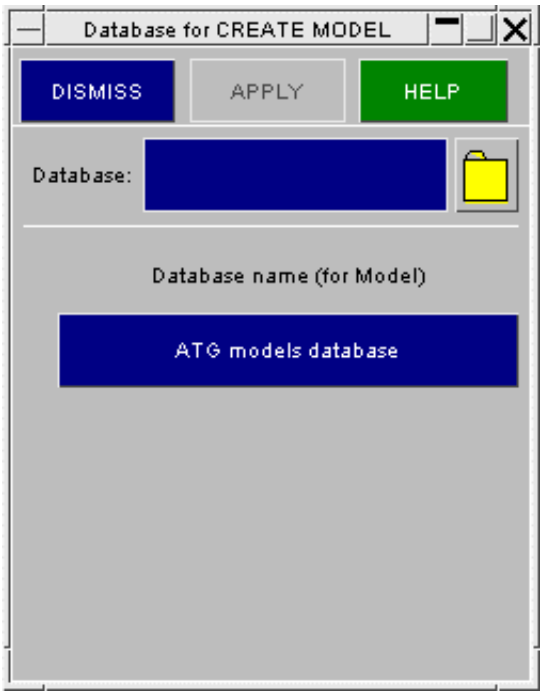
In order to select an existing database, select the **SELECT EXISTING DATABASE** (the default) option and press **APPLY**.

Connection contacts? The main panel allows you to configure the build mode to simple, rigorous, etc and to activate the post build **connection contact** options either to make new contacts for unconnected welds/adhesives or to fix existing contacts by using the penalty copy contact method (or setting IPBACK flag if keyout version is R7.0 or higher).

Scan component files? The use of include files with component files is blocked for rigorous build. However, the simple build process does permit it. This requires a pre-read scan of component files to determine how many includes they contain which is time consuming. If the user knows that there are no includes in any of the selected component files, the process can be speeded up considerably by inhibiting the scan - the **No Scan** option.

Build log file. All information written to the dialogue box may be saved to a session file on exit of Primer. By default this is not done unless the pref setting *save_dialogue_dir* has been set - then you will find files *pr_dialog_nnn* in the designated directory. However, in the absence of this setting. if a model build has been performed, the information will be written to a *pr_build_info_nnn* file in the directory of the database. If something goes wrong with the build process this file will aid the debugging of the problem as it contains many relevant print messages.

To create a new Database select the appropriate option and press **APPLY**. See **CREATE NEW DATABASE.**

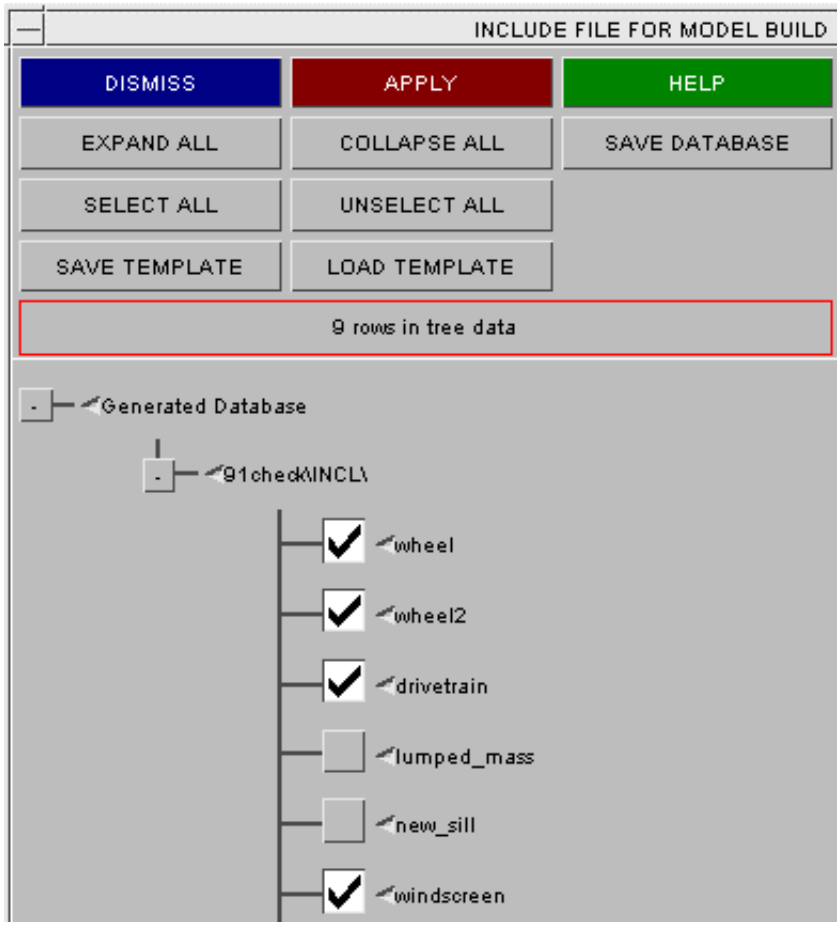


Select an existing database by inputting the name and path in the input box or using the search facility or by selecting one of the databases listed in the Database Name list.

Press **APPLY** to load the database. Once this is read in the following section applies.

3.15.1.2 Viewing the Model Database

In order to see descriptions of all the files available to read, click on the **EXPAND ALL** tab. In order to display just the top layer of the database select the **COLLAPSE ALL** tab. To see the thumbnails linked to each file, to give an idea of its contents, select the **VIEW THUMBNAILS** tab. To hide the thumbnails click on the **HIDE THUMBNAILS** tab.



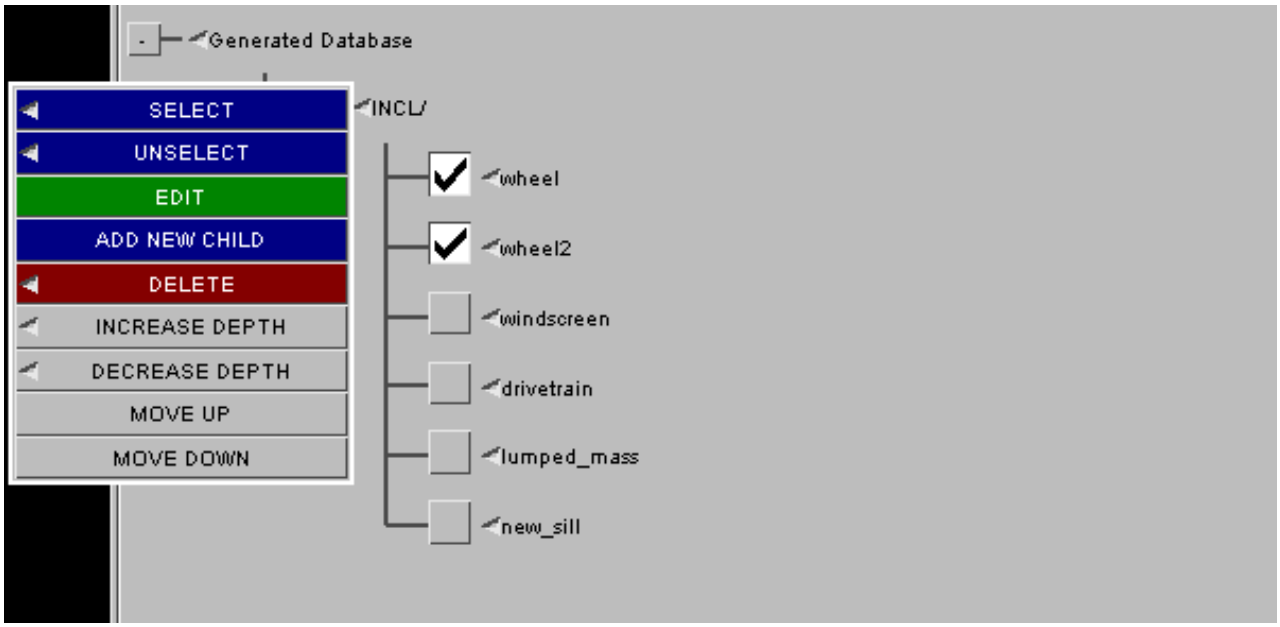
3.15.1.3 Selecting include files to read

In order to select a standard file from the database to read into PRIMER, select the file by clicking on the grey box to its left. The box will display a tick when selected to show which files are to be read in.

Changing the keyword file associated with an entry: To select a file not currently in the database you can select any row, right click on the small arrow to the left of the option and select **EDIT** on the pop-up window that appears. The file which is read in when this option is selected can now be changed by changing the name of the keyword file in the Keyword file input box.

Adding a new entry: Alternatively you can add a new entry into the database by right clicking any option and selecting the **ADD NEW CHILD** tab. You can then enter the file for your new entry.

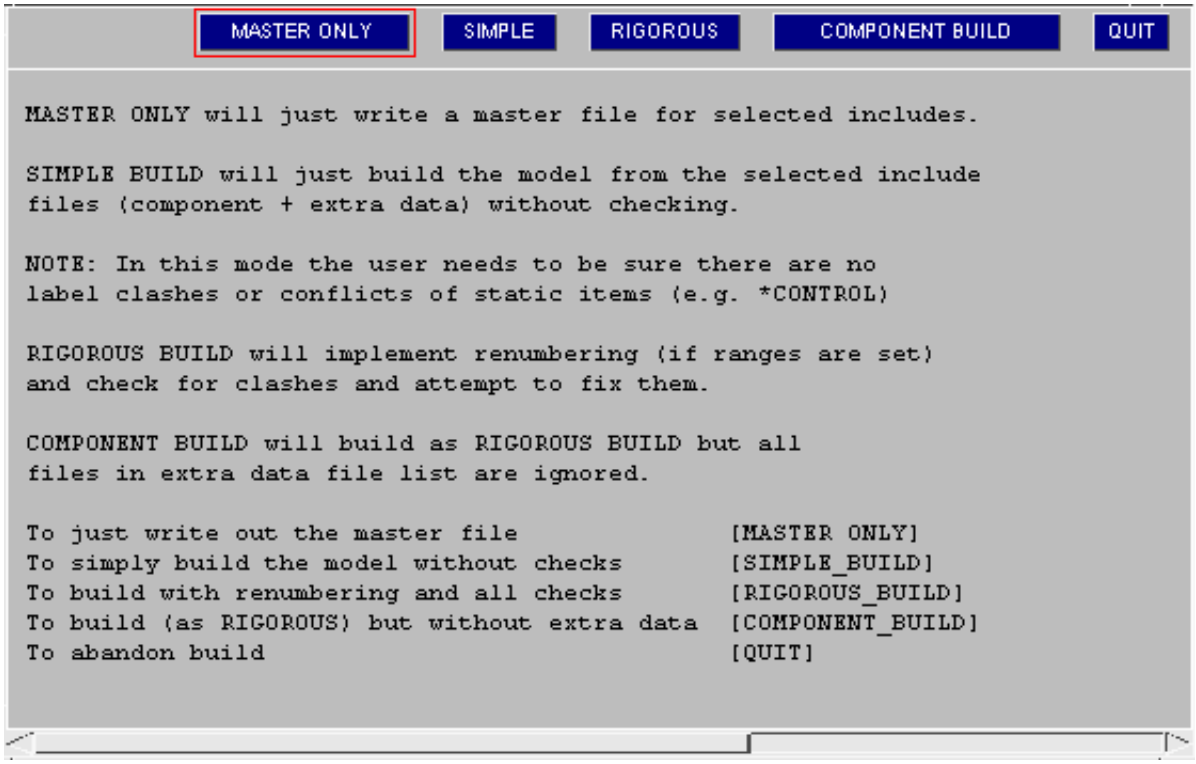
Using a template: If there is a particular combination of Include files you frequently wish to read in together, you can save a template listing these include files (described later in this section). Click on **LOAD TEMPLATE** in order to open an existing template and the include files listed in the template will be automatically selected.



3.15.1.4 Applying the Build

Once you have selected all the files you wish to read in, press the **APPLY** button. You will now select one of 4 build modes. **MASTER ONLY** will create a master file for the selected include files. **SIMPLE BUILD** is suitable when there is no renumbering scheme and no label clashes amongst the include files. If either of these two conditions is not applicable a **RIGOROUS BUILD** is required. This is considerably slower as each include file has to be read into a scratch model, checked against the existing model, perhaps re-labelled and then read into the existing model. **COMPONENT BUILD** will ignore all connection files during the build, which allows user to check that all component files are self contained (no references to items which don't exist) and numbered correctly within their ranges.

You can skip this panel by pre-setting the build mode to **SIMPLE BUILD** or **RIGOROUS BUILD** by setting the radio button option.



PRIMER will now take you through a number of stages to help ensure the model is built correctly.

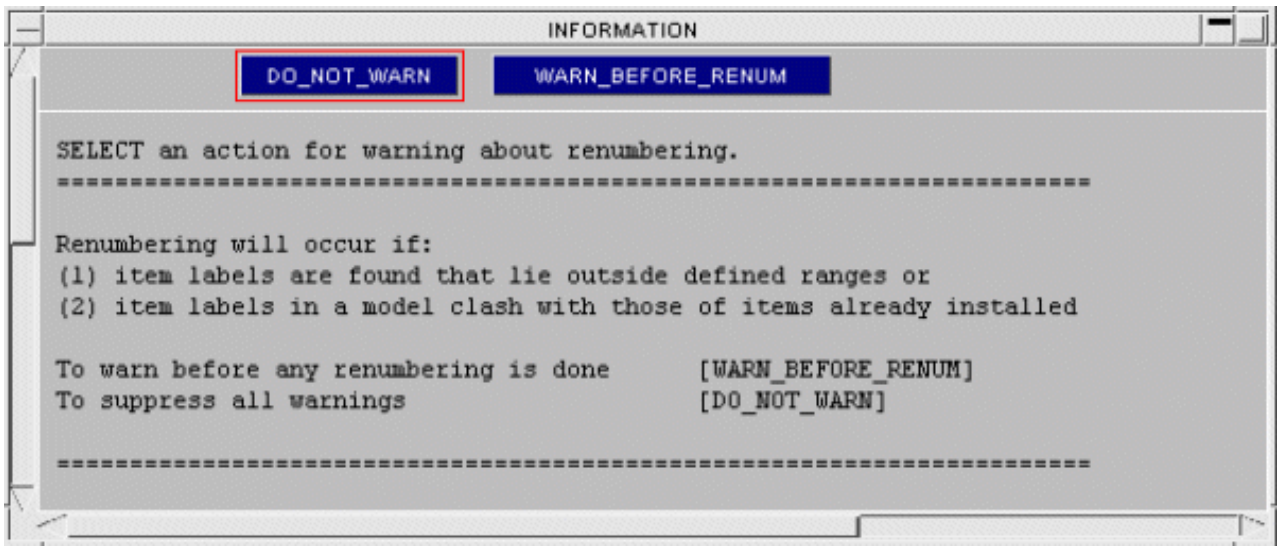
The next choice PRIMER will ask you to make concerns warning over model renumbering. Model renumbering may occur because either certain model labels lie outside the ranges specified in the database or because item labels in the include file are already taken up in the model.

The first pass involves setting item labels to meet the renumbering ranges. If the "**general type id**" range is set, part/section/material (see note below) labels must already be within this range. Other items will get renumbered into it. An **optional second range** may also be set for the more populous items (nodes, elements, nodal rigid bodies and node sets). Thus a node in the first or the second range will not have its label changed, but a node lying outside both will be renumbered into the second range. This node may subsequently be re-labelled for clash fixing. There is also a **FROZEN** range, intended for DATABASE_HISTORY items, the labels of which will never be changed.

Note: renumbering is NOT applied to latent items, so Materials that are referenced by a part but exist elsewhere (typically in a material database file) will not be affected.

The second pass involves checking the (perhaps re-labelled) items of the include file against those of the model and, if possible, re-labelling to avoid clashes. If this proves impossible, e.g. because a label range is exhausted or frozen items lie out of range, the read operation will trip an error.

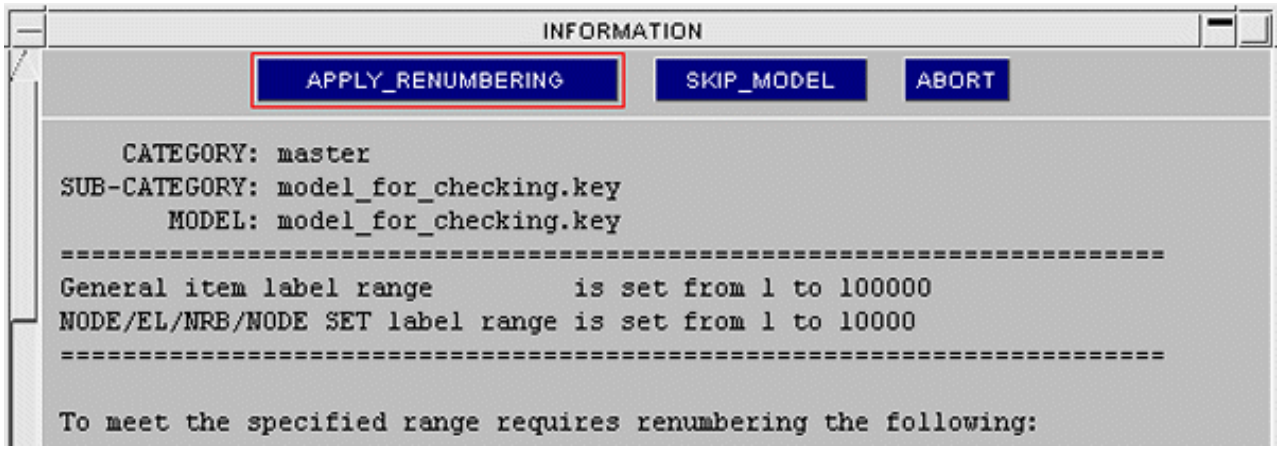
If you wish to be warned of any renumbering before it occurs, press the **WARN BEFORE RENUMBERING** button. If you wish PRIMER to renumber without notifying, press the **DO NOT WARN** tab. Then dialogue box will only be invoked in the event of a failure to renumber.



Renumbering into Range

If you select the option **WARN BEFORE RENUMBERING**, a window will pop up in PRIMER before any renumbering takes place providing a description of the renumbering necessary and a list of options.

If renumbering is achievable the options are **APPLY RENUMBERING**, **SKIP_MODEL** or **ABORT**. Otherwise, they are: **CONTINUE**, **SKIP_MODEL** or **ABORT**.

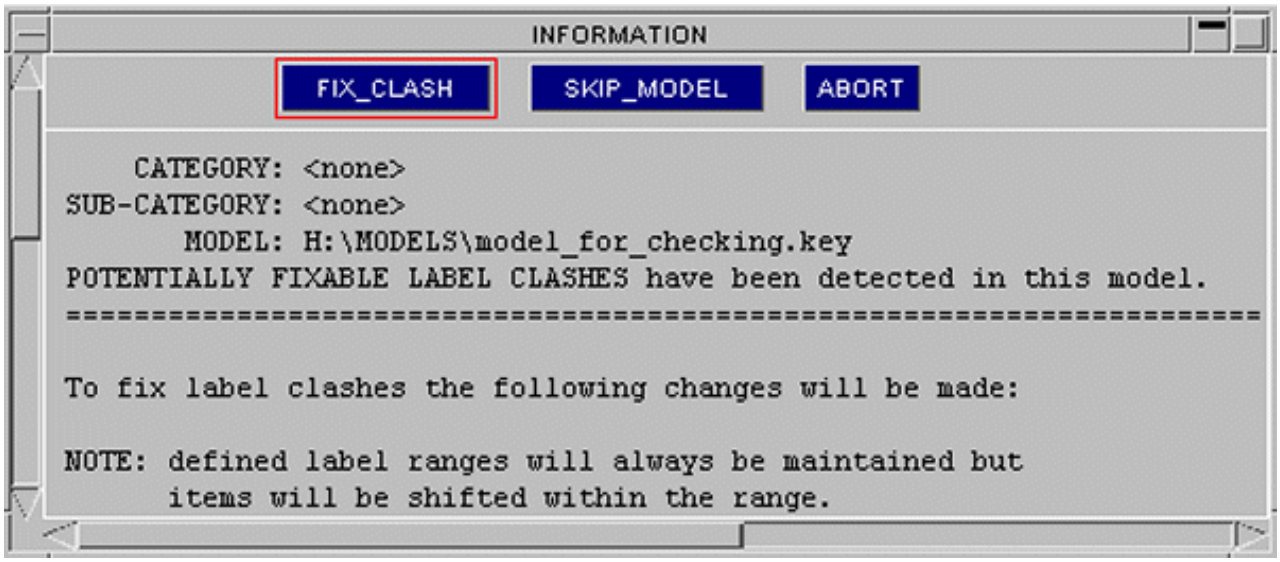


If you select **CONTINUE** the operation will proceed as usual. If you select **SKIP_MODEL**, the file that required renumbering will be skipped and PRIMER will continue to read all further files. If you select **ABORT**, PRIMER will stop the operation entirely.

Renumbering to fix clashes

Similarly, If renumbering is required because of a clash of labels, PRIMER will detail the fixing procedure automatically. Three options will be available: **FIX_CLASH**, **SKIP_MODEL** or **ABORT**.

If you select **FIX_CLASH** PRIMER will fix the numbering problem as detailed in the pop-up window. If you select **SKIP_MODEL**, the file that required renumbering will be skipped and PRIMER will continue to read all further files. If you select **ABORT**, PRIMER will stop the operation entirely.



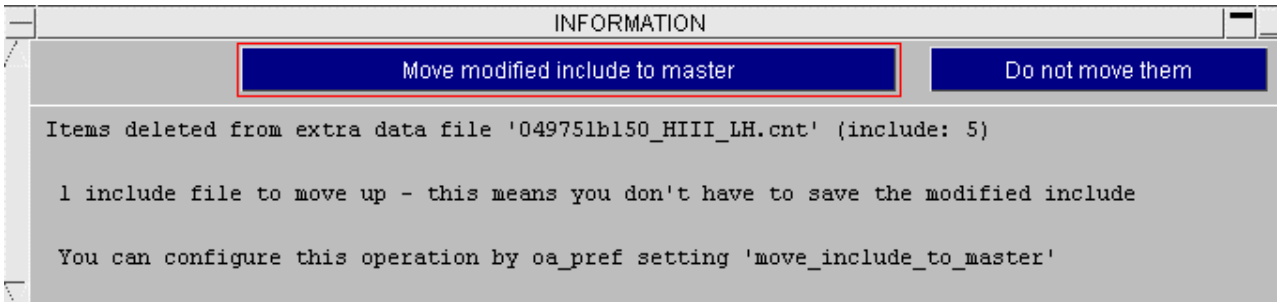
Once the model build has been completed, PRIMER will search the model for references to missing items. These will arise typically, where a connection file spans several components, but a choice was made to build the model with a subset of those components. For example, a connection file containing a contact which includes a vehicle dummy, an airbag and a steering wheel, but with only the dummy and the steering wheel read in as components. The airbag parts are now latent items in the connection file and must be removed before the job can run in LS-Dyna.

A window will pop up mentioning the number of missing items and offering option to **FIX LATENT.**

If you select **CONTINUE**, the missing items will remain in the model and it will require fixing before it can be run in LS-Dyna

If you select **FIX LATENT** Primer will run an auto-fix procedure to delete the offending items.

As the option 'move_include_to_master' is set to ASK, you will get a 2nd information panel.



The deletion of latent items has modified the extra data file. If the option **move modified include to master** is taken, all the items of the file are moved up into the master file, and in the master file the keyout of the include file is itself suppressed. Thus the keyout of nthe master file alone is sufficient to represent the load case.

Fix for Missing Items

For more information on model cleanup look at section 6.4.2 Cleanup unused.

If you apply the fix Primer will also remove any unused items in the model (with the exception of parts and part sets which we deliberately keep) as this is implicit in the cleanup unused procedure.

The removal of missing items may incur further deletion operations such as the removal of contacts with empty sets. For this reason the cleanup unused procedure is necessary to ensure that the model will run in LS-Dyna.

Primer will now summarize the build procedure in the Summary box and present the result of checks made on the model in the Check box. Once you have studied these boxes and wish to continue, click on the **FINISH** tab to complete the build procedure. The model Database will reappear. The information contained within the Summary and Check boxes will be written to a file called **build_status.txt.**

3.15.1.5 Post model build panels

The following panels appear if the **RIGOROUS** or **COMPONENT** build has been selected.

Summary Box

Finish | Help | Condense | Apply checks | List missing | Hide panel

Master file title: Model built by primer

Category:	Sub-category:	Model file:	Owner:	Installed?	Version	x-refs?	Renum'd?	Extra data
MASTER FILE								
Control Cards	Term. time 140ms	contr_term_140ms.k		OK	default	OK	NO	
Control Cards	General	contr_general.k		OK	default	OK	NO	
General	General	datab_general.k		OK	default	OK	NO	
Dummies 049751	LH 50%ile HIII	049751b154_HIII_LH.k		OK	default	OK	NO	RENUM
Dummies 049751	RH 50%ile HIII	049751b503_HIII_RH.k		OK	default	OK	NO	RENUM
IP & Dash 049752	LHD IP Fascia	049752b054_IP_fascia.k	M Buckley	OK	default	OK	NO	RENUM
IP & Dash 049752	LH Driver EA Bracket	049752b151_LHDriver_EA.k		OK	default	OK	NO	OK
IP & Dash 049752	LH Steering Column	049752b401_LH_Strg_Col.k	M Buckley	OK	default	OK	NO	^MOVED^
IP & Dash 049752	LH Steering Wheel	049752b451_LH_Strg_Wheel.k		OK	default	OK	RENUM	
IP & Dash 049752	LH Steering Column Shroud	Strg_Col_Shroud.k		OK	default	OK	NO	
Chassis 049754	LHD LWR Column	049754b050_Lwr_Strg_Col.k	D Ashby	OK	default	OK	RENUM	
Chassis 049754	Pedal Box / Booster	049754b701_Pedal_box.k	M Buckley	OK	default	OK	RENUM	
Powertrain 049755	HVAC	049755b725_HVAC.k	M Buckley	OK	default	OK	NO	RENUM
Seats 049756	208 50th M (front) LH	049756b250_LH_FEDNCAP_front_50th.k		OK	default	OK	RENUM	
Seats 049756	208 50th M (front) RH	049756b550_RH_FEDNCAP_front_50th.k		OK	default	OK	NO	

Listed down the left hand side of the box are the names of all the keyword files that you asked to be read in. In order to show only those keyword files that have been renumbered (orange) or that show an error (red), click on the **CONDENSE** tab.

Up to 7 pieces of information are available on each file under the headings:

- Owner?
- Installed?
- Standard?
- x-refs?
- Renum'd?
- Extra data?

A green box illustrates that no problems were encountered under the corresponding heading in the building procedure. An Orange box illustrates a Warning - items in the file were renumbered or a connection file was shifted into the master file (see below). A Red box illustrates a fatal problem indicating that the file was skipped or contains missing cross-references.

In order to list any missing parts that have been read into the model, press the **LIST_MISSING** button..

In order to display the model Database in its tree format, press the **SHOW_TREE** button.

To access the Check box if it isn't already present, press the **APPLY_CHECKS** button

If the check box is displayed, you may temporarily hide the summary box by pressing the **HIDE_PANEL** button. It will be restored by **RETURN_TO_SUMMARY**.

Owner

This category states who the owner of the corresponding keyword file is.

Installed

This category declares whether the keyword file was installed successfully (**OK**) or failed to install (**FAILED**).

Standard

This category reveals whether the installed keyword file was a standard file (**YES**) contained in the loaded database or a non-standard file (**N/S**), resulting from an edit to the the database.

X-refs

The **BAD XREF** error indicates that the include file contains references to items which do not exist in the model. It should only arise if the Deletion and Cleanup functions have been aborted during the model build.

Renum'd

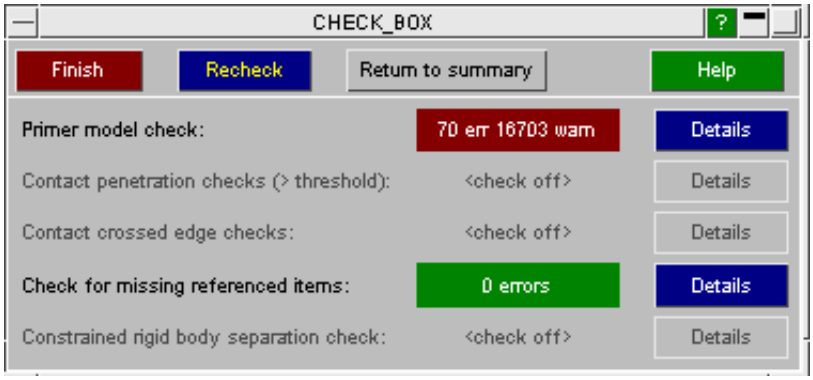
This category specifies whether or not any renumbering of the contents of the keyword file occurred in the building process.

Extra data

This category specifies whether the extra data files linked to the file were read in successfully (**OK**) or reports a warning, such as **FAILED** or **BAD XREF**.

If, during the build process, latent items of an extra data file have been deleted (see **APPLY FIX**), the file contents will have been shifted to the master file and the include file itself suppressed. Such files will bear the warning "^**MOVED**^".

Check Box



This window contains the result of a number of predefined checks specified as **CHECK>OPTIONS.**

In order to detail the results of any check that generated errors, press on the **DETAILS** button.

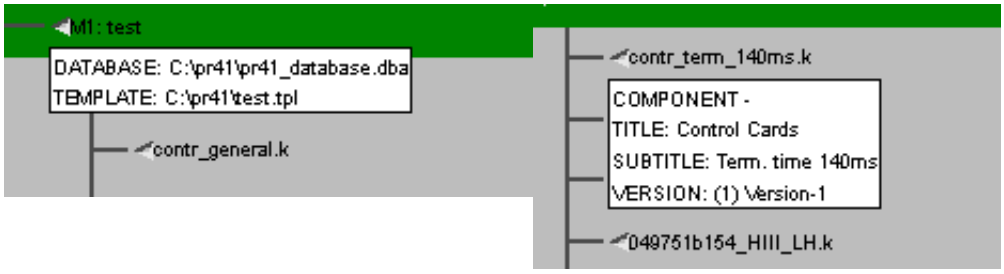
To close this window and return to the summary box, click on the **RETURN_TO_SUMMARY** tab. In order to rerun the checks, for example, after modifying the **CHECK**->**OPTIONS** settings, click on the **RECHECK** tab.

NOTE: The same check panel can be activated for any existing PRIMER model by the route **CHECK**->**APPLY**->**APPLY_RULES**.

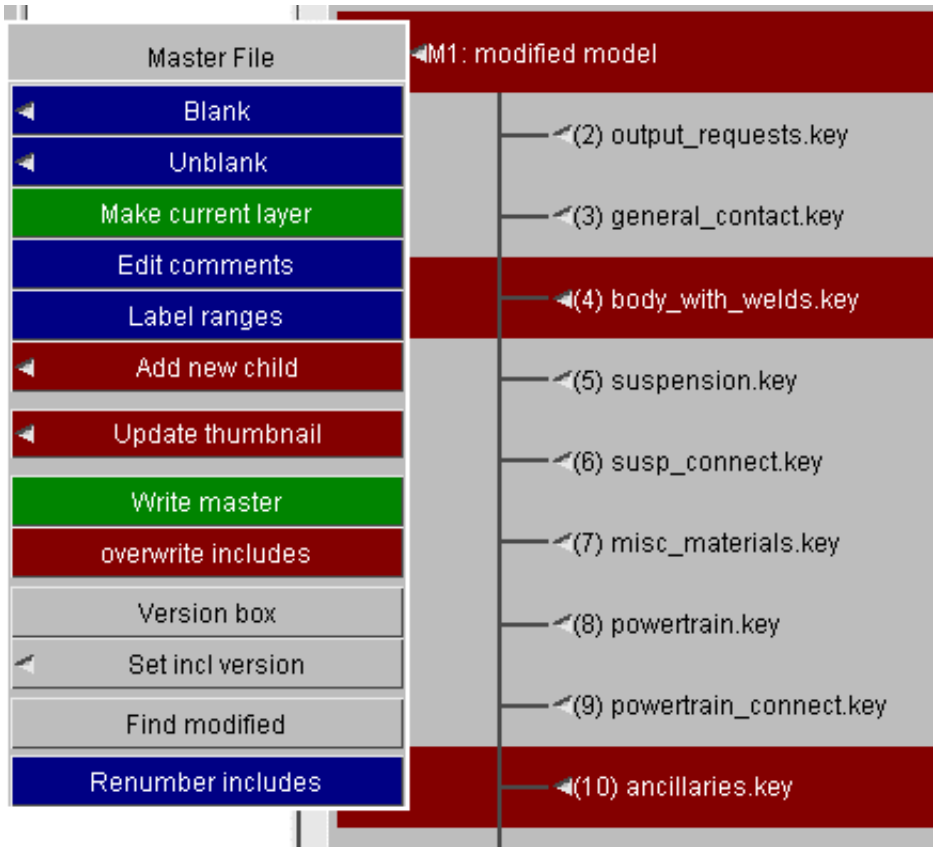
3.15.1.6 Modifying the built model

The built model now exists in Primer and may require modification, such as fix by auto-fix or otherwise of model errors, modification of properties, modification of boundary conditions, etc. On completion of the changes the updated include file must be saved. If it is overwritten the database requires no change, however, if it is written to a new filename the database entry requires update.

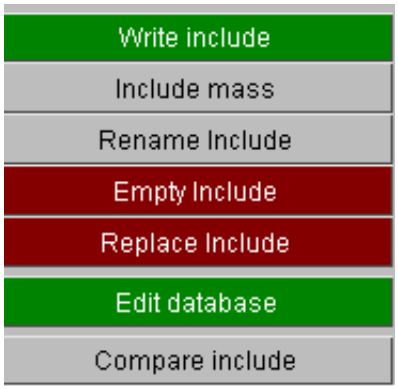
The include tree (TOOLS > INCLUDE) is now aware of the provenance of the model and its includes. This is displayed in hover text.



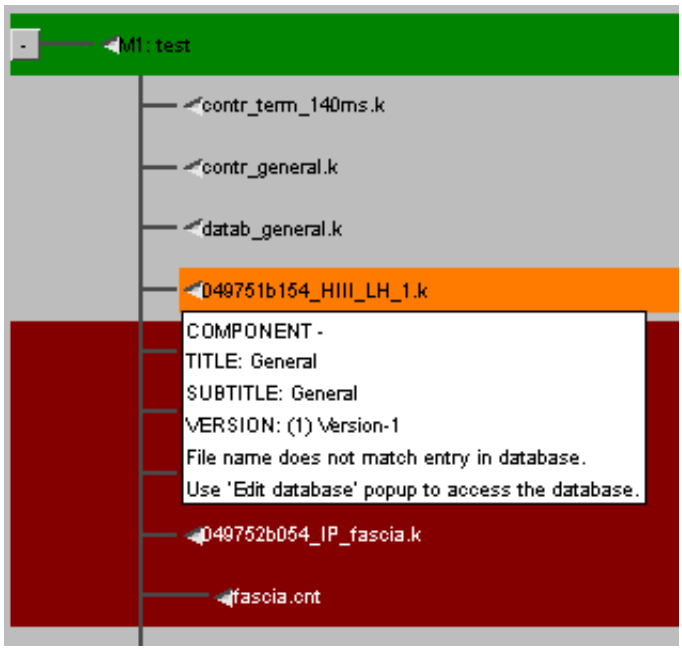
The Find modified function may be run on all the includes if accessed off the model popup. This will write the current include to a model and read the original include file into a model. These two models are compared and any differences reported.



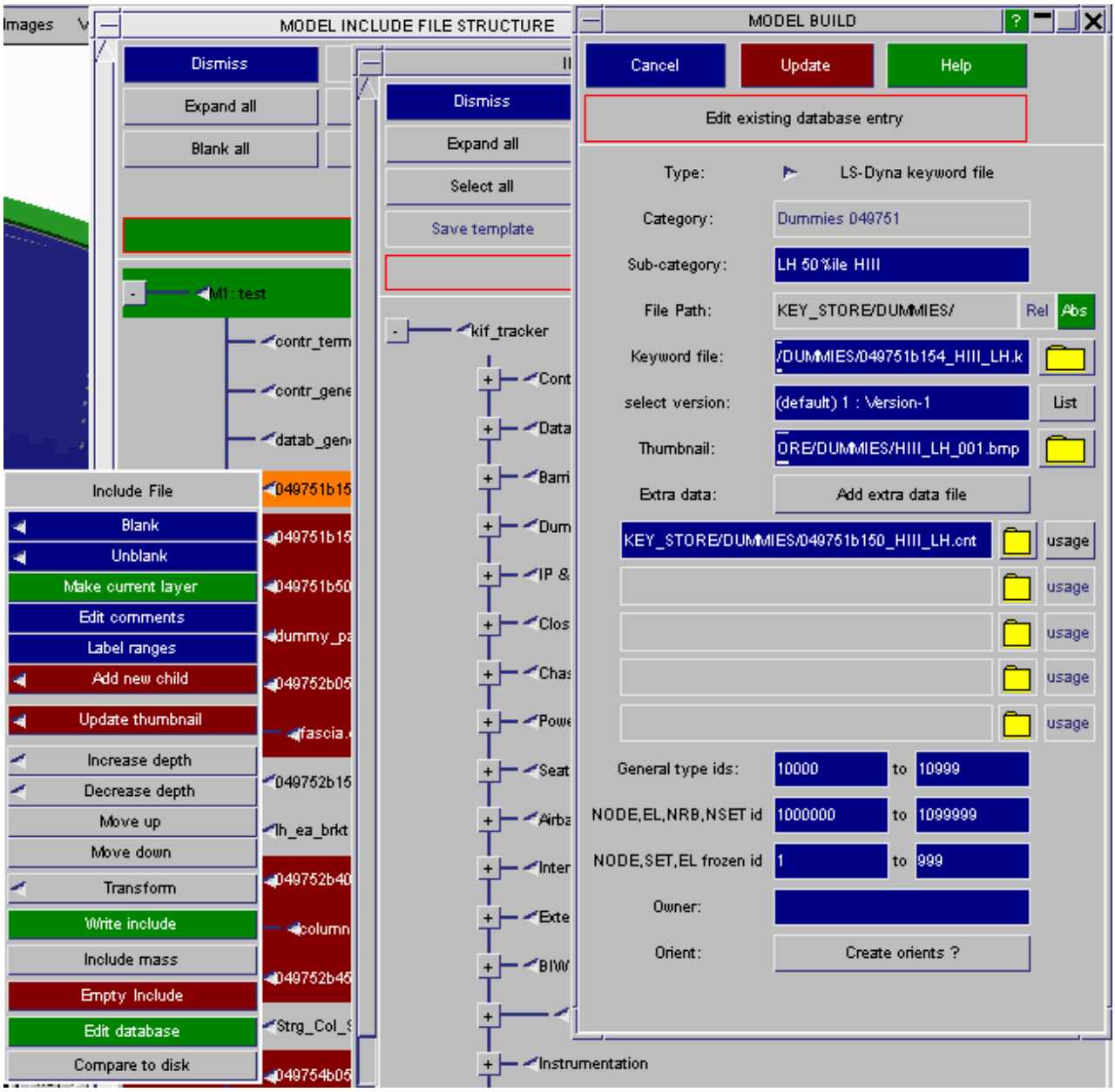
On completion of the function the modified includes will be marked by a red background as shown above. The details of the differences for an individual include may be reviewed by running compare include off the individual include popup.



Write include available off the include popup can then be used to save the modified include to a new name. The entry then becomes orange to warn the user that the database entry is out of date. The hover text is also modified.



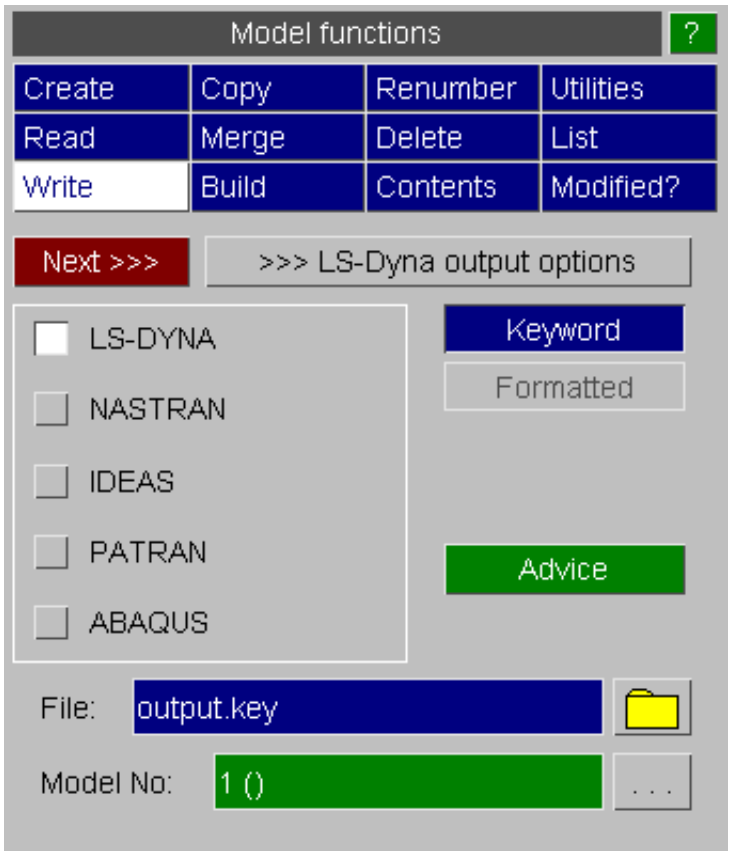
Edit database available off the include popup can be used to take you directly to the entry on the database. Here the keyword file (or extra data file) entry can be updated (at a newly created version number if you are using version tracking), the database entry updated and the database re-saved. The orange entry will then have its grey background restored, as the model and database are again consistent.



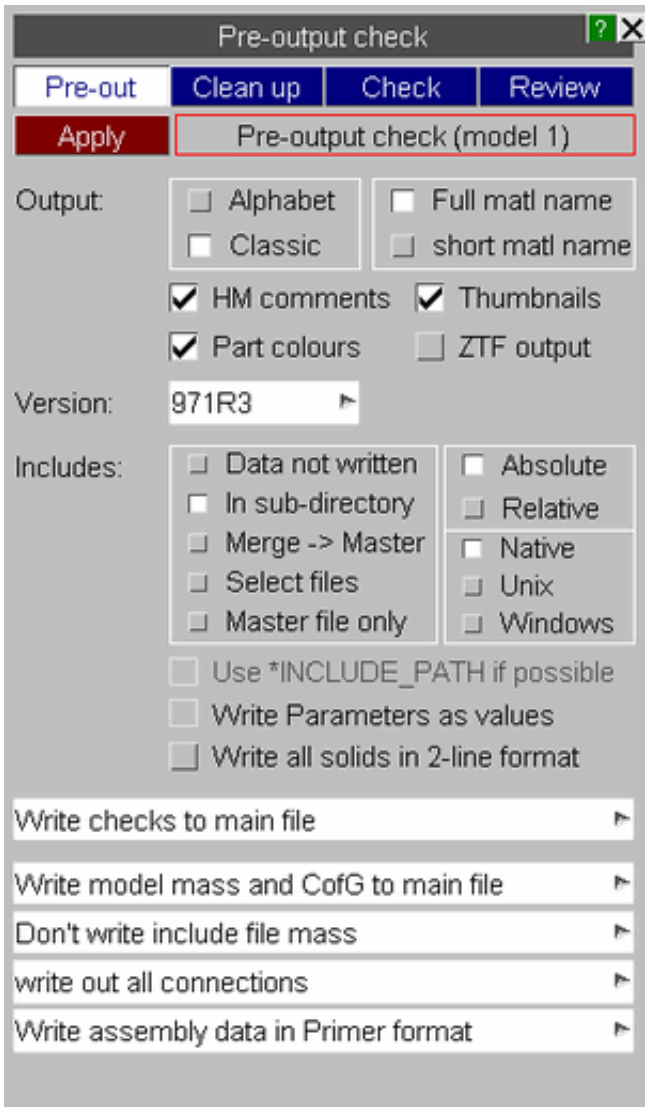
3.15.1.7 Writing the Model using select files

To save the model once it has been built, in the main menu right click on Model and select write in the pop-up.

In the Write to file window input the name you wish to save the master file under in the input box:



Select appropriate options in the Pre-Output check window:



If you select **Master File only** mode and any include files have been renumbered during the build process Primer will prompt you to change to **Select files** mode, where the renumbered includes will be preselected, putting up the following panel.

SELECT_FILES CONTINUE

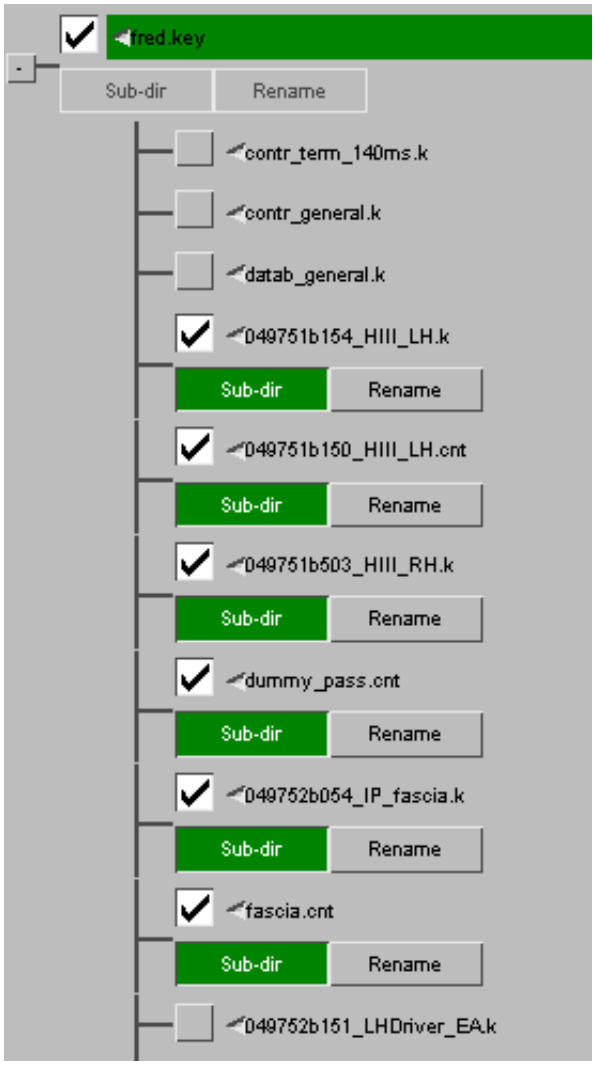
```
25 include file(s) were renumbered during model build
=====================================================

C:\pr4l\KEY_STORE\DUMMIES\049751b154_HIII_LH.k
C:\pr4l\KEY_STORE\DUMMIES\049751b150_HIII_LH.cnt
C:\pr4l\KEY_STORE\DUMMIES\049751b503_HIII_RH.k
C:\pr4l\KEY_STORE\DUMMIES\dummy_pass.cnt
C:\pr4l\KEY_STORE\IP_AND_DASH\049752b054_IP_fascia.k
C:\pr4l\KEY_STORE\IP_AND_DASH\fascia.cnt
C:\pr4l\KEY_STORE\IP_AND_DASH\049752b401_LH_Strg_Col.k
C:\pr4l\KEY_STORE\IP_AND_DASH\049752b451_LH_Strg_Wheel.k
C:\pr4l\KEY_STORE\CHASSIS\049754b050_Lwr_Strg_Col.k
C:\pr4l\KEY_STORE\CHASSIS\049754b701_Pedal_box.k
C:\pr4l\KEY_STORE\POWERTRAIN\049755b725_HVAC.k
C:\pr4l\KEY_STORE\POWERTRAIN\hvac.cnt
C:\pr4l\KEY_STORE\SEATS\049756b250_LH_FEDNCAP_front_50th.k
C:\pr4l\KEY_STORE\AIRB_AND_SEATB\driver_airb.cnt
C:\pr4l\KEY_STORE\AIRB_AND_SEATB\Passenger_airbag.k
C:\pr4l\KEY_STORE\AIRB_AND_SEATB\pass_airbag.cnt
C:\pr4l\KEY_STORE\AIRB_AND_SEATB\seatb_frnt_lh.cnt
C:\pr4l\KEY_STORE\AIRB_AND_SEATB\049757b501_Belt_RH_50th.k
C:\pr4l\KEY_STORE\AIRB_AND_SEATB\seatb_frnt_rh.cnt
C:\pr4l\KEY_STORE\INTERIOR_TRIM\049758b001_LH_door_trim.k
C:\pr4l\KEY_STORE\INTERIOR_TRIM\int_trim_a_pillar.cnt
C:\pr4l\KEY_STORE\INTERIOR_TRIM\049758b400_Header.k
C:\pr4l\KEY_STORE\INTERIOR_TRIM\int_trim_header.cnt
C:\pr4l\KEY_STORE\INTERIOR_TRIM\049758b450_LH_Carpet.k
C:\pr4l\KEY_STORE\BIW\049760b500_vs2sled_vl.k


SELECT_FILES will switch to <select files mode> and preselect the
renumbered files.

To select renumbered files        [SELECT_FILES]
To continue in master only mode   [CONTINUE]
```

SELECT_FILES will then take you to the keyout selected panel.



Note - Primer stores a marker that the component has been renumbered, this is **not** a rigorous check for modified includes. To achieve that you need to use the compare to disk function described above.

3.15.1.8 Linking model to database

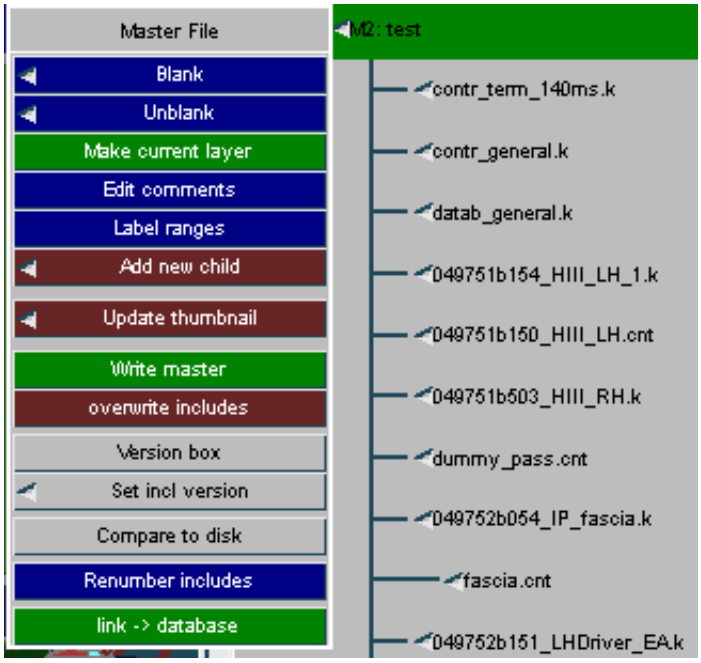
When a model has just been built, the include tree is linked implicitly to the database from which the model has been built as described above.

However, you may wish to write out the built model (probably just the master file) and come back to work on it later. This can be done in Primer by re-linking the read model to its database.

When you read in a model that has been built using the database/template method, a special comment in the master file enables Primer to recognize the origin of the model. Hover text on the model popup will show the database and template. By default the model is **not linked** to the database and may be treated as an independent entity.

By activating the **link -> database** option off the model popup, the include tree will behave as if the model has been built. It does not matter if the contents of model have been changed before linking. However, it is assumed that the include file structure has not been changed, i.e. includes have not been added/deleted or written to different names.

After linking the hover text that shows the provenance of each include will become active and **Edit database** will be available. Writing out includes (to different file names) is now handled as described above.



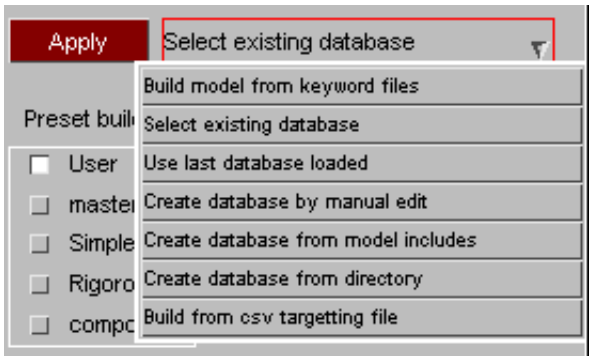
3.15.2 Creating and managing a model Database

- Creating a new model database
- Editing the model database
- Creating and editing database entries
- Templates

A Model Database consists of an hierarchical list of Include files and information regarding those Include files from which an individual can select a number of Include files to build a model.

3.15.2.1 Creating a new Model Database

Access the model Database creation options by clicking **BUILD** under the MODEL tab. Then select the appropriate build option.



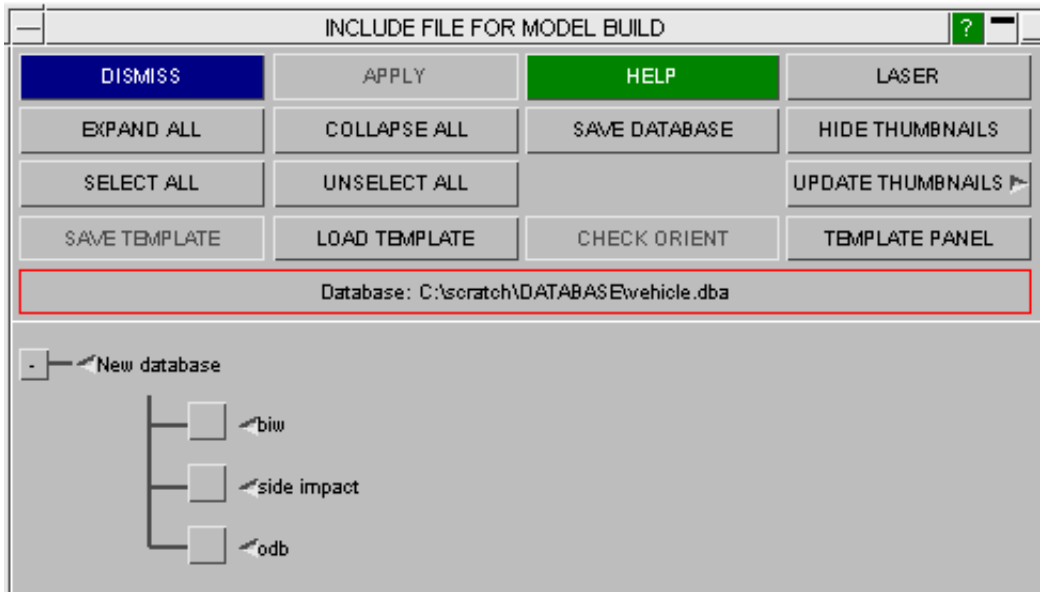
Create database from directory. If your files have been grouped under meaningful directory names (DUMMIES, BARRIERS, etc.) you can easily create a database with this option. You need only specify the start directory and Primer will locate all the ".key" files in sub-directories and create the database structure.

Create database from model includes. If you have a model in memory which contains an include file structure, you may create a database directly from the model. The database will group the keyword files for each directory, under a category named after the directory itself.

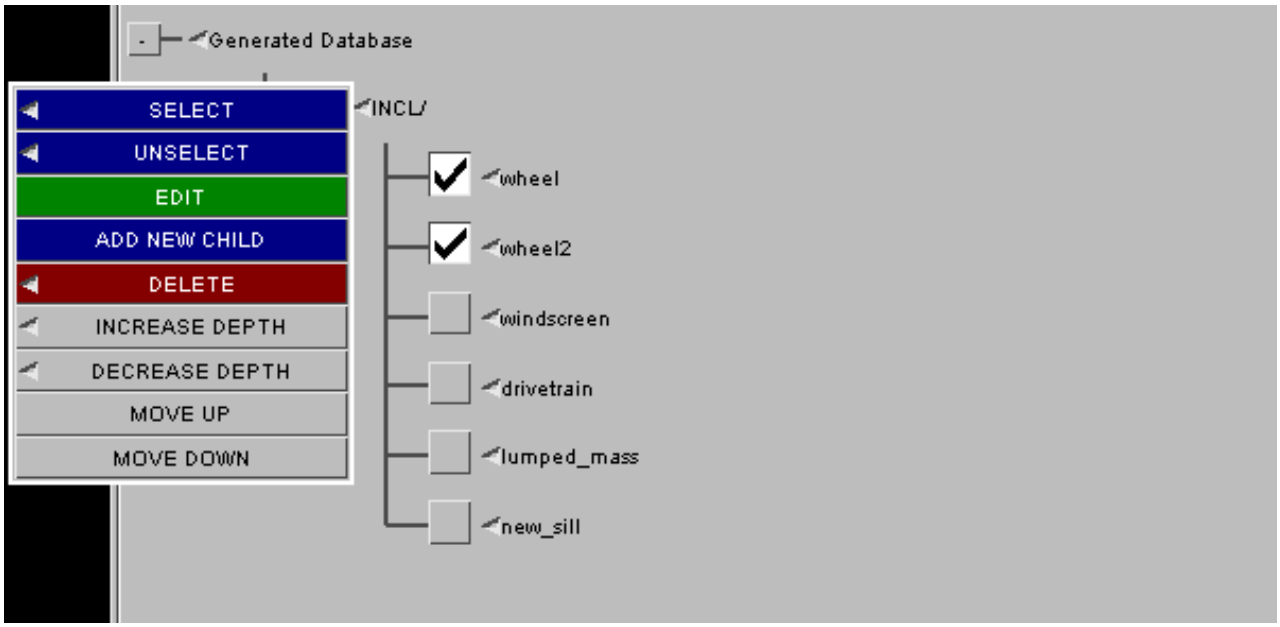
In both the above cases the new database should then be loaded by reverting to **select existing database** option.

Create database by manual edit. Apply will create a starter database with a single row. In order to add entries into the database, right click on the present entry and select **ADD NEW CHILD** in the popup menu and a window will appear asking you to provide information about the new entry. See the creating and editing database entries section. To save the database you have created, select the **SAVE_DATABASE** tab and input the path and name you wish to give to the database into the input box.

In order to write out a postscript file to provide a print out of the Database, select the **LASER** tab and fill in the required categories.



3.15.2.2 Editing a Model Database



In order to add entries into the database, right click on an entry in the layer above where you wish to create the new entry that would serve as a appropriate category for the entry and select **ADD NEW CHILD** in the popup menu. A window will appear asking you to provide information about the new entry. See the creating and editing database entries section for information about the contents of this window.
Current entries can be edited by right clicking on the entry and selecting the appropriate option.

- Edit
- Delete
- Increase Depth
- Decrease Depth
- Move up
- Move down

Edit

This option allows you to alter the information about the file inputted when first created. For more information see the creating and editing database entries section.

Delete

This option brings up a second pop up menu when you place your cursor over it. Selecting THIS will delete the entry selected, this option cannot be selected if the specified entry has any children. Selecting CHILDREN will remove any lower level files under this entry. Selecting THIS AND CHILDREN will remove both the selected file and any lower level files underneath it.

Increase Depth

This option brings up a second pop up menu when you place your cursor over it. Selecting THIS will move only the selected entry up a level in the database, this option cannot be selected if the specified entry has any children. Selecting THIS AND CHILDREN will move both the selected entry and all entries in lower levels up a level.

Decrease Depth

This option brings up a second pop up menu when you place your cursor over it. Selecting THIS will move only the selected entry down a level in the database, this option cannot be selected if the specified entry has any children. Selecting THIS AND CHILDREN will move both the selected entry and all entries in lower levels down a level.

Move up

Selecting this option will move the selected entry down an entry in their current level.

Move down

Selecting this option will move the selected entry down an entry in their current level.

3.15.2.3 Creating and editing database entries

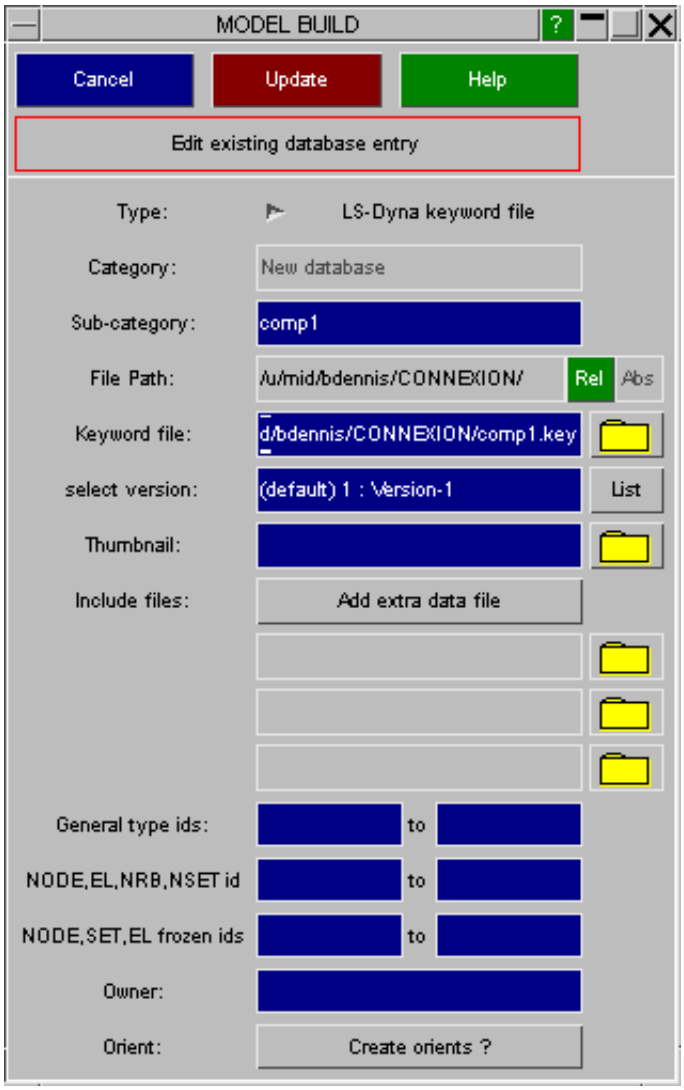
When adding a new child or editing an existing entry in a database the following information can be given;

- Category
- Sub-Category
- Model Path
- Keyword file
- Thumbnail
- Extra data files
- General type ids
- NODE,EL,NRB,NSET ids
- NODE,SET,EL frozen ids
- Owner
- Orientation data

A category must always be specified if the entry is to exist in the Database. If the entry references a keyword file then a Sub-Category must also be present.

The combination of category and sub-category must be unique. This is how a database entry is referenced by a template, and is deliberately independent of the keyword file name. The user who is building the model is selecting the item category, not the keyword file directly. The person responsible for maintaining the database must ensure that the end user’s selection gets the most up to date version of the component file.

Primer now allows you to store multiple keyword files for each component. The current version will determine which keyword file is actually used. See Version Control.

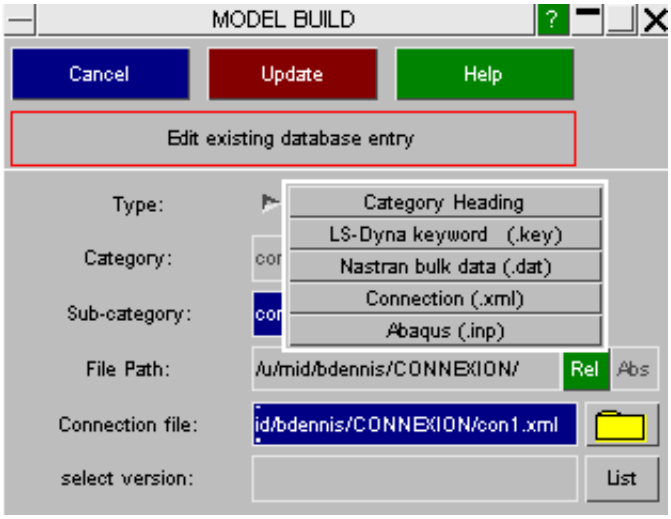


Component type

In addition to an LS-Dyna keyword file, the component may be described by a Nastran or Abaqus file (for which Primer supports a subset of keywords).

Also an xml connection file may be loaded as a component of the build. These may be managed using the same version control as keyword files.

On completion of the build a connection panel will be invoked to process the connection file(s). See build with connections.



Category

This option allows you to give the database entry a title by which you may identify it in the future. If no sub-category is given then the category will be displayed as the name of the entry.

Sub-category

The Sub-category acts as a second name to identify your the entry. If specified it will also be displayed as the name of the entry.

Model Path

This contains the path to the keyword file and is automatically generated from the full path when a file is selected from the selector box. If you type in the filename directly without a path, the file will be expected to be in the same directory as the database.

Usefully, the path may be set to Absolute or Relative using the **Abs** or **Rel** button.

Keyword File

If you wish the entry to contain a keyword file, enter the name of the file here.

Thumbnail

This option enables you to insert a bitmap image to act as a graphical representation of the contents of the entry. Enter the name and path of the bitmap file here. The image will be displayed alongside the name of the entry in the Database.

Extra data files

Press **ADD EXTRA DATA FILE** to specify the name(s) of any extra data file(s) which are to be associated with this component.

These will always be read as includes immediately after the main component file has been installed. Although the files may contain any LS-Dyna keywords, they would normally be expected to contain items such as contacts, rigid body merge connection, nodal rigid bodies, etc. which connect component files together. Therefore, unlike the component files, these files are not "stand alone" and will contain references to items in component files. If a referring component is omitted, this risks leaving the extra data file with reference to a missing item (e.g. part-set of contact refers to missing part). To handle this, Primer on completion of build will find all latent items and offer to delete them using FIX LATENT function. On completion of deletion, you will be given the option to move the contents of the modified extra data file up into the master file and suppress the keyout of the include file itself. If you take this option, you will not need to save the modified include.

General type ids

This option allows you to specify a number range into which PRIMER will renumber general items (excludes nodes, elements, nrbs, node sets).

Any items labelled outside the range specified will be renumbered to fit in the range. If the range is not set no renumbering of these items will occur.

Input the lowest boundary of the range to the first input box and the highest boundary of the range to the second input box.

If any parts, sections or materials lie outside the specified range an error will be reported. These will never be renumbered.

NODE,EL,NRB,NSET ids

As nodes, elements, node sets and nodal rigid bodies often require a larger range than other items, they have their own item range.

All nodes, element, node sets and nodal rigid bodies lying outside the node/el range will be renumbered into this range.

If the range is not set no renumbering of these items will occur.

Note on latent items: If an item in an include file is effectively latent, e.g. a material card that is referred to but not actually in the file, it will **not be renumbered**.

NODE,SET,EL frozen ids

This option allows you to specify a range of node, element, node set and element set numbers that will never be renumbered in the build process.

This can be useful if you wish to preserve your time history items or if you wish to protect items (other than parts) used for connection.

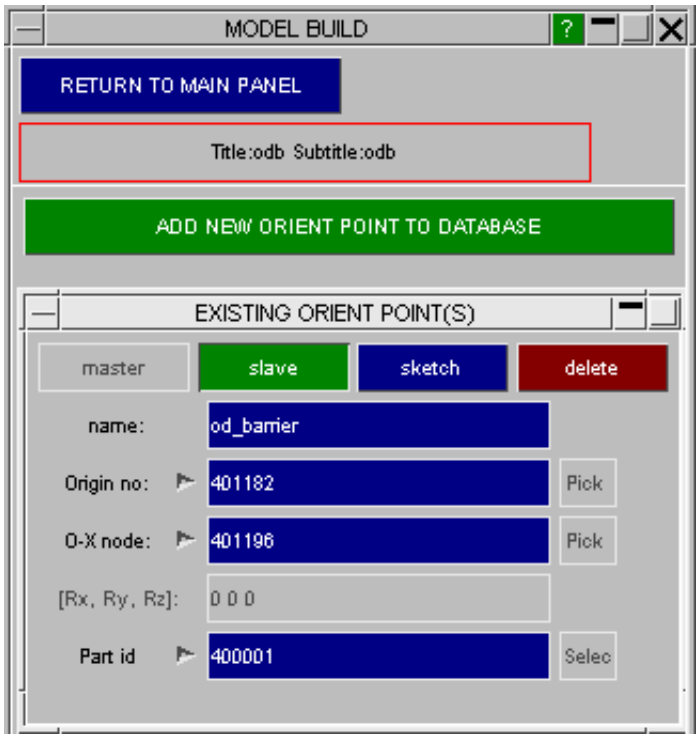
Owner

Enter the owner of the file here.

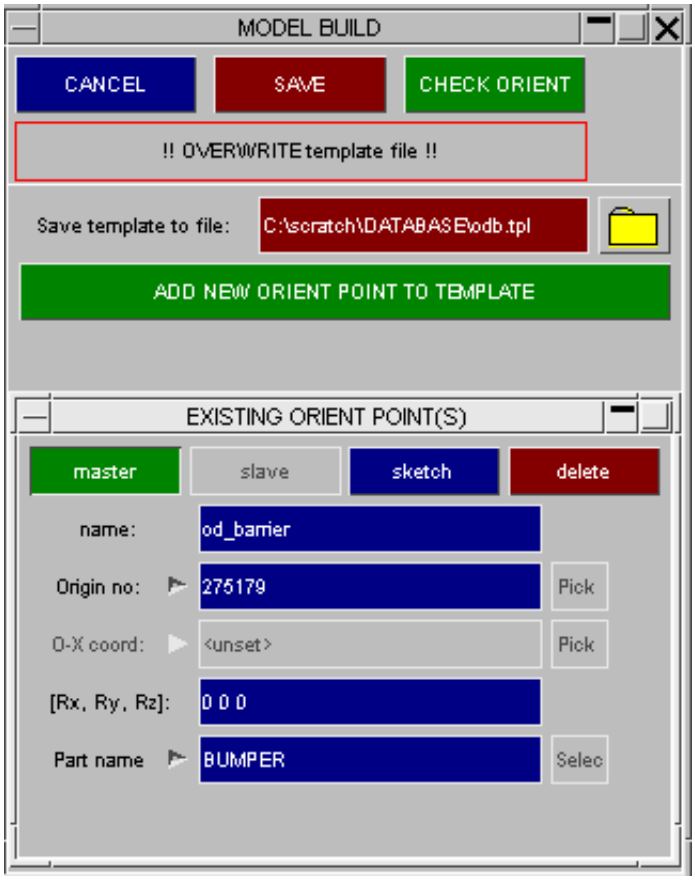
Orienting the include files

It is possible to orient include files during the model build process. This is achieved by generating *INCLUDE_TRANSFORM rather than plain *INCLUDE.

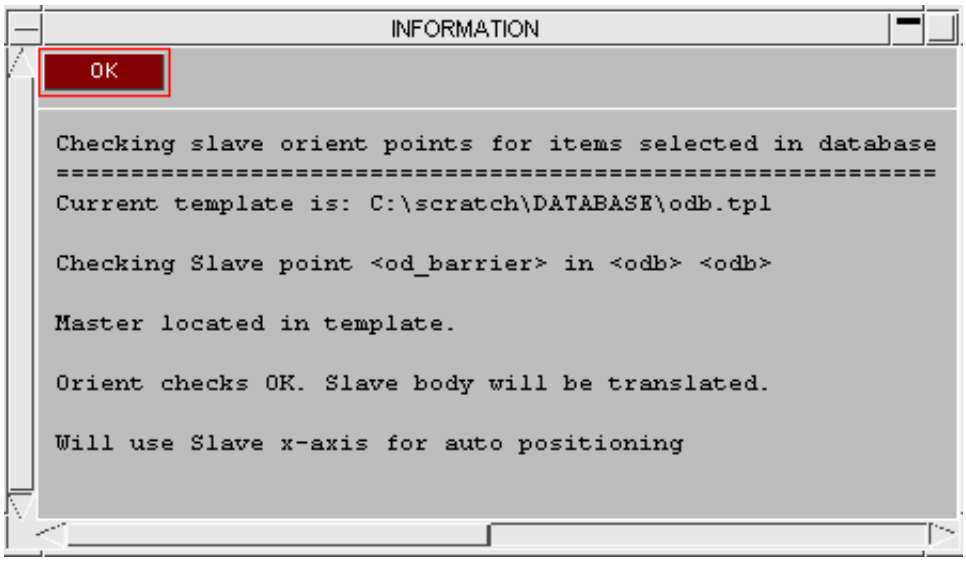
The Orient create/edit feature is accessed through the **Create/Edit slave/master Orients** button on the category edit panel.



The user creates each orient by adding a master and a slave point of matching name. The slave points will always reside in the database, under the include file to which the orient is to be applied. The master points may be stored either in the database or in the template as they apply for a particular load case.



After you have created/edited orients you need to save the database or template. It is recommended that you then run the **CHECK ORIENT** function, which will sketch the orient as well as report its status.



In the simplest orient case you need to define a master point and its co-ordinate (or node id) and a slave point of the same name and its co-ordinate. The build process will detect matching master & slave orient points, resolve any node ids into co-ordinates and calculate the necessary transform to bring the slave point to the master. A*INCLUDE_TRANSFORM will then be applied to the include file associated with the slave orient.

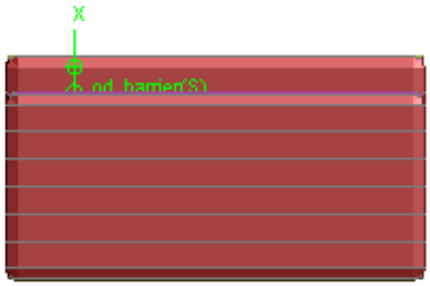
Rotation of component model: A master orient point may additionally include global rotations [Rx, Ry, Rz about the master point] which will be applied to the slave include file.

Moving into position: On the slave orient a second point (O-X) may be defined (co-ordinate or node id) which defines a "line of flight" vector. This defines the direction in which the impactor will be moved if it is *not penetrating* and against which it will move if it *is penetrating*. In the normal case where the impactor is initially positioned away from the vehicle the vector should point toward the vehicle (as shown below).

Additionally information is required so that a contact can be created and the slave body will be either depenetrated or advanced along the "line of flight" vector until it is on the point of contact. Thus an odb barrier can be set up so that it will be optimally positioned for different bumper designs with minimum wasted cpu time before the onset of impact.

Setting up the contact. The orient point may reference a contact that exists in the model directly. If so, this may be located on the master orient point or the slave point. In the event of both being defined, the one on the master orient will be used. Alternately, a part or part-set may be defined for both the master and slave orient points, these will be used directly to define a surface-surface contact. Alternately, a single part set may be defined for the slave side only. In this case, Primer will split the set up to form a surface-surface contact between the impactor and the vehicle. For part/part-set method the contact is disposable and will not appear in the model.

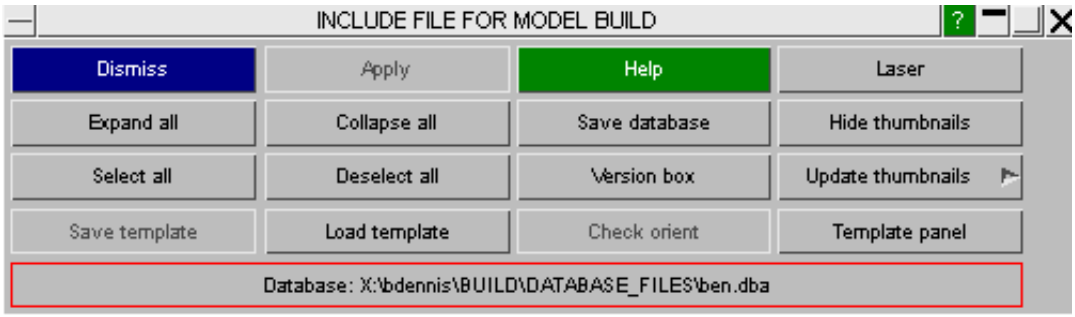
Definition of orient points. These may be defined as node id, node name (if *DATABASE_HISTORY_NODE_ID) or a co-ordinate. The node method has the advantage that if the component files get moved the orient points will still be in the correct position. If the node is defined by ID it must not be renumbered. This may required the node to be included in the frozen range if renumbering during model build is active. If using the co-ordinate method the orient data must be updated if component files are moved.



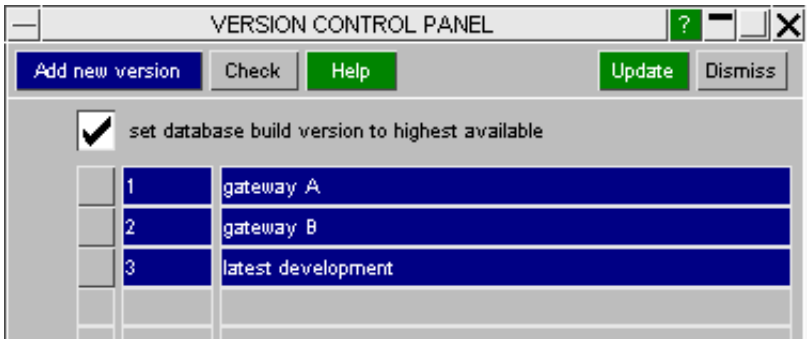
3.15.3 Version Control

Version control provides a powerful method of keeping track of the development of component models during a project. It also allows you easily to recreate a previous version of your model should you need to do so. The information is stored both in the database and a corresponding .history file which Primer updates.

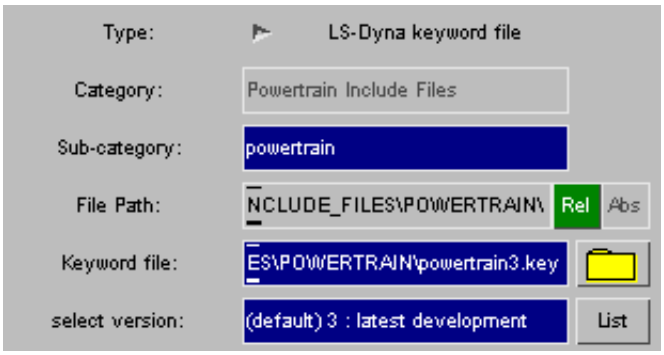
One way of managing version control would be this. At suitable point you will add a new version to the database using the "**Version Box**" function, e.g. version 3 for "latest development" build. You will freeze all the component files and copy them, e.g. copying the current development *powertrain.key* component file to *powertrain3.key*.



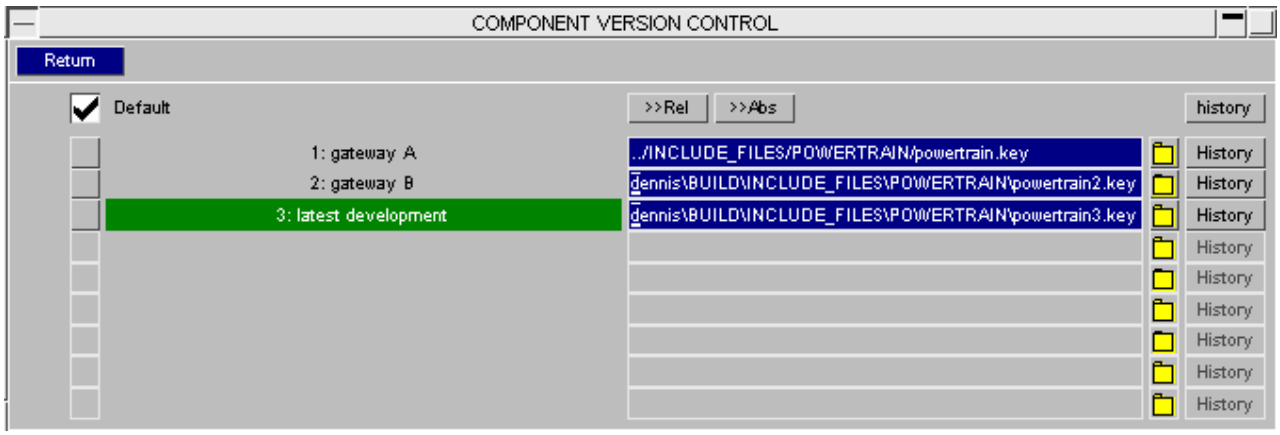
The version control panel by default will be set to use the latest version (v3 in this case).



Using the edit function on the build tree, select the component of interest and, instead of selecting a keyword file directly, press **LIST** to access the multiple keyword panel.



Add in the new file as the version 3 component. Return and update the component.



3.15.3.1 User defined history

By pressing the **HISTORY** button you can bring up a text editor to add history comments to the describe the version.

pr_tmp29 - WordPad

File Edit View Insert Format Help

```
Do not edit, remove or add any lines starting with '$'
User comments go below the $ FILE line
$ TITLE   : Powertrain Include Files
$ SUBTITLE: powertrain
$ VERSION : 1 (gateway A)
$ FILE    : ../INCLUDE_FILES/POWERTRAIN/powertrain.key
comments about gateway A
$ VERSION : 2 (gateway B)
$ FILE    : X:\bdennis\BUILD\INCLUDE_FILES\POWERTRAIN\powertrain2.key
.....user comments here.....
$ VERSION : 3 (latest development)
$ FILE    : X:\bdennis\BUILD\INCLUDE_FILES\POWERTRAIN\powertrain3.key
comment about what has changed in the latest version
```

Build of the model will now use version 3 if available for all components. If the version is not available it will use the highest previous version.

Version information and the relevant history comment will be written out with the master file.

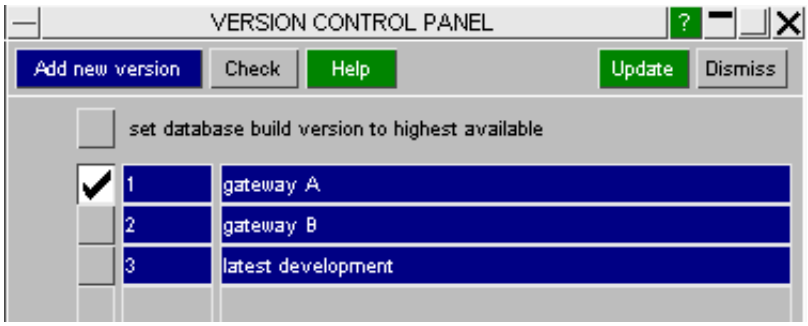
a.key (X:\bdennis\BUILD) - GVIM

File Edit Tools Syntax Buffers Window Help

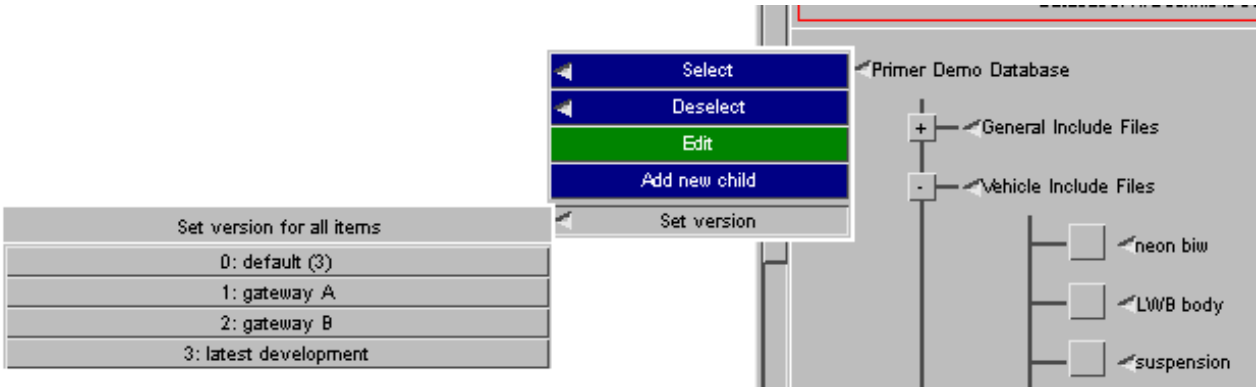
```
$
$ =============
$ INCLUDE cards
$ =============
$
*INCLUDE
$==============================================================================
$PR_COMPONENT_TITLE   : Suspension Include Files
$PR_COMPONENT_SUBTITLE: suspension
$PR_INCLUDE_VERSION   : 1
$==============================================================================
X:\bdennis\BUILD\INCLUDE_FILES\SUSPENSION\suspension.key
$
*INCLUDE
$==============================================================================
$PR_COMPONENT_TITLE   : Powertrain Include Files
$PR_COMPONENT_SUBTITLE: powertrain
$PR_INCLUDE_VERSION   : 3
$ comment about what has changed in the latest version
$==============================================================================
X:\bdennis\BUILD\INCLUDE_FILES\POWERTRAIN\powertrain3.key
$
$
```

3.15.3.2 Setting the applied version

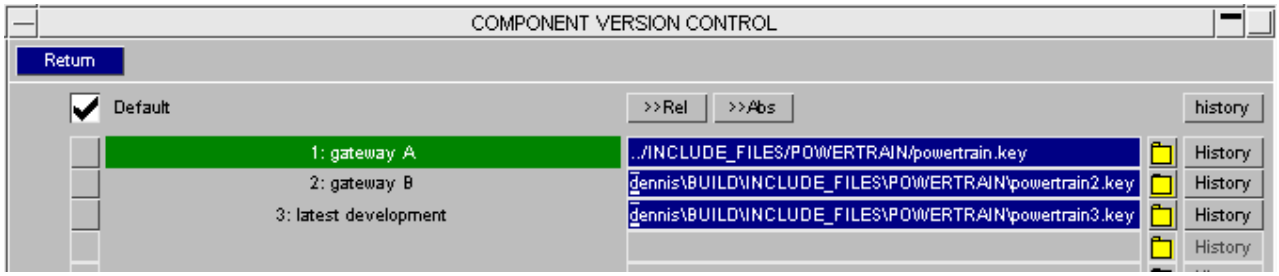
To build a previous version of the model, use the version box to select the one you want, e.g. gateway A (version 1)



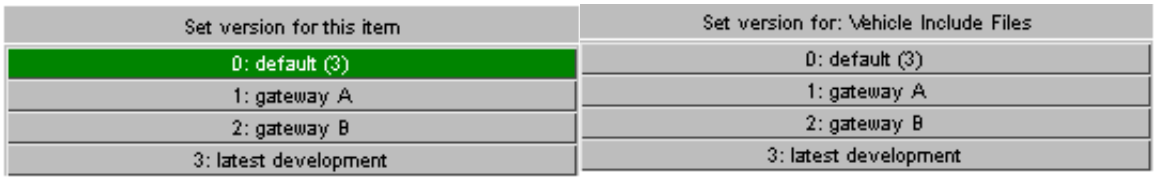
Use the popup of the head node of the build tree to set all component files to use default version (this is their default setting).



All component files will now use version 1 as shown by inspecting the edit panel.



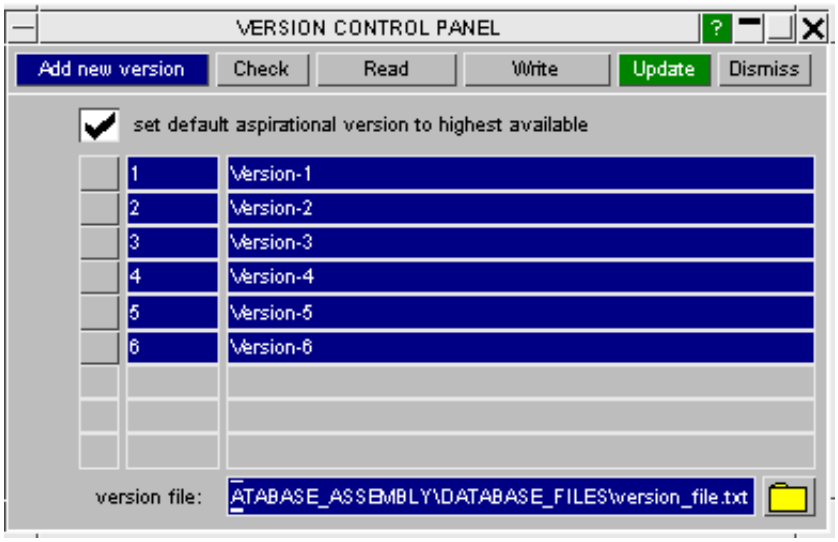
The version may be set on individual component files, or for a whole branch of the build tree, using the version popup.



3.15.3.3 Recording a snapshot of versions

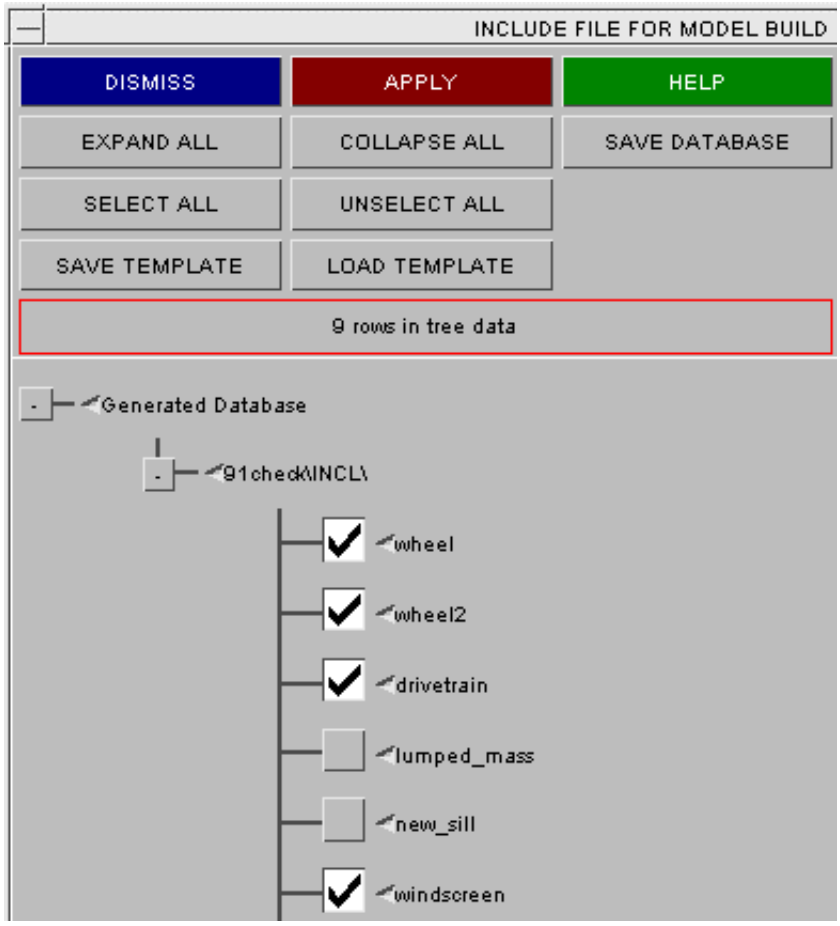
The method described above works on the assumption that the development of component files of the project marches progresses in a monotonic manner. Some users have requested a more exact control of which version applies to which component.

This can be achieved by the writing/reading a version control file. **Write** will dump a text file with the current version of each component file in the database. **Read** will apply the specified versions in the text file to each component. Thus a particular build can be constructed using the version setting tools described above and the file can be written to capture these. At a later stage of the project, the snapshot may be recaptured by loading the database and template and applying **Read** of the version file.



3.15.4 Templates

The template provides an easy way to select a set of include files with which to build a model.



Templates provide a way of saving particular combinations of include files in order to allow you to easily read in a particular, frequently used, pattern of files without having to select each file from the database every time you build a model. Generally, there will be one database for a vehicle programme, and one template for each load case or variant.

In order to save a particular combination of include files, select the desired combination in the Model Database window and press **SAVE TEMPLATE**.

When a standard keyword file is present, the template will reference *only the category and sub-category* of the

mentioned file, hence if the keyword files of the database are externally updated, the model read in from the template will automatically be the latest one.

If a template is saved that lists a non-standard include file (i.e. the user has modified the original database entry - it shows in red), the name and path of the include file will be specified in the template. When this template is read in, a warning will be given that non-standard keyword files are being used.

In order to read a template file click on the **LOAD TEMPLATE** tab and the selection of files in the template will be selected in the model database.

3.15.5 Editing multiple templates

On vehicle programs there will be many variant load-cases to analyze and consequently many templates to handle. The TEMPLATE CONTROL PANEL accessed from the **TEMPLATE PANEL** button will display contents and allow modification of multiple templates.

Primer will locate all the templates that exist in the search directory, applying the filtering string if it is set.

Reread All will discard any current edits and reread templates from disk.

Add new tpl will create a new blank template which can be populated and saved.

Increment all will modify all loaded template names, such that *fred.tpl -> fred_001.tpl* or *fred_001.tpl -> fred_002.tpl*. This allows easy version control for templates. The renamed templates must then be saved. Special logic has been added so that *fred_1.tpl* will increment to *fred_2.tpl* (not *fred_002.tpl*).

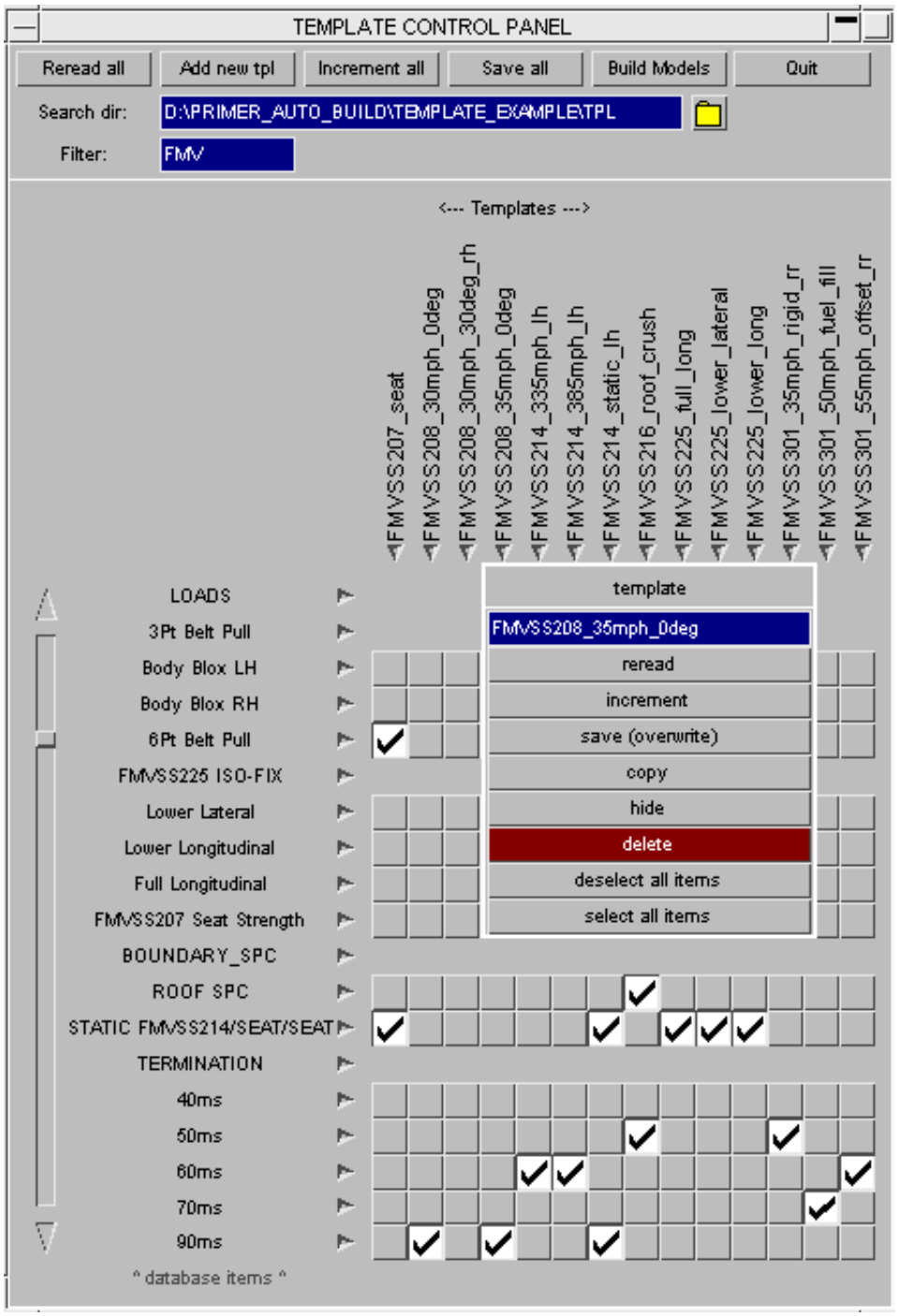
Save all will save all the loaded templates to disk in the search directory, overwriting (without warning) if necessary.

Build Models starts multiple model build panel, see below.

Quit returns to the database panel.

The above functions may be activated for a single template by using the drop down (as shown below). Additionally, this allows user to **Copy** an existing template.

The Database item popups allow selection of an item across all templates. They also access the same category edit panel that is available from the database panel.



3.15.6 Multiple Build from template panel

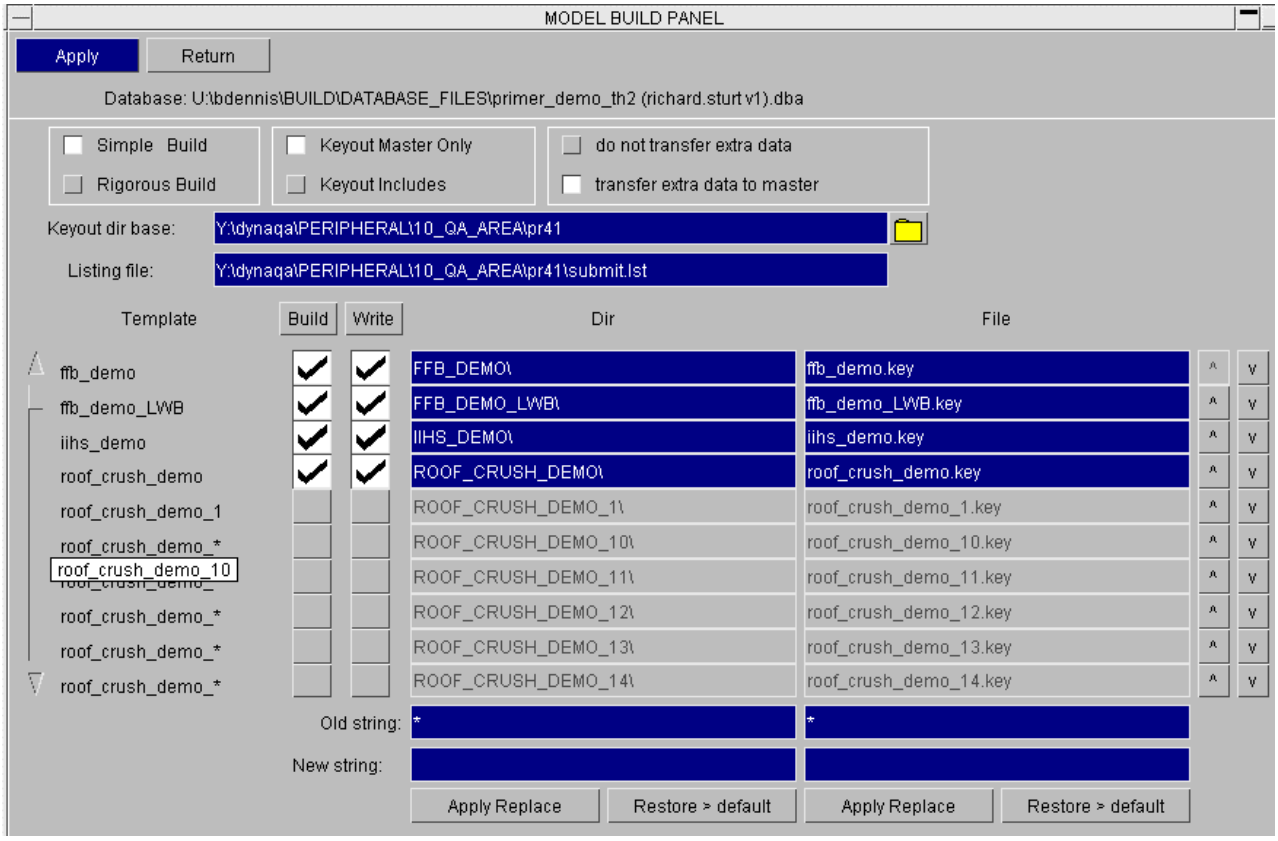
The **Build Models** button will take you to a panel which enables the build process for all the active templates. As this process reads files from disk, it is essential that the database and all templates have been saved.

Build. Activating this will mean that the model will be built and retained as a model in memory in Primer.

Write. Activating this without build will mean that the model will be built (if necessary), keyed out and then deleted from memory. If build is of simple mode and contains no orientation, it is not necessary for primer to build the model. The filename and directory are automatically generated, based on the template name, the directory being appended to the keyout directory base. A listing file will also be written which can be read by the Shell to submit a set of LS-Dyna jobs.

Keyout Master Only. This is the default for build from templates. In this mode, the implemention of bolt connections (when an xml connection file is included as part of the build recipe) is run without applying the setting *use_parent_layer_for_bolt* (irrespective of how this is defined) with the consequence that all created bolt FE will

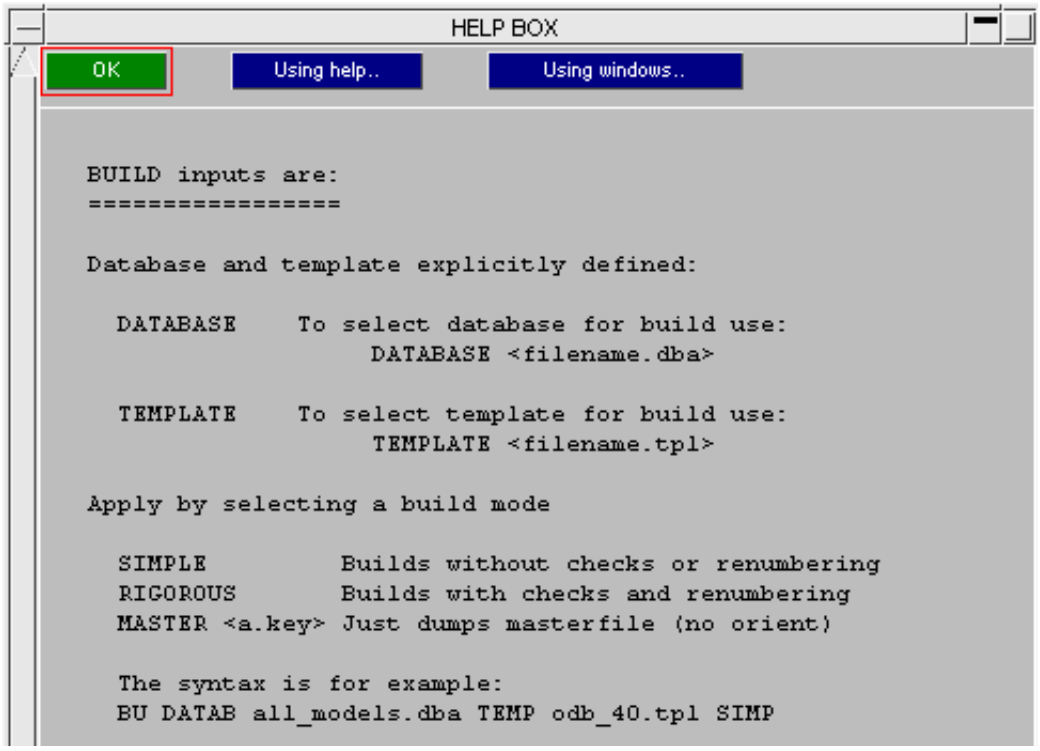
appear in the master file itself, thus obviating the requirement to additionally save includes.



3.15.7 Single Build from command line

The database/template build may be run in simple, rigorous or master only mode from the command line or in batch mode using a command file.

Type **BUILD** on the command line to set the mode. Then **HELP** to get a description of the syntax.

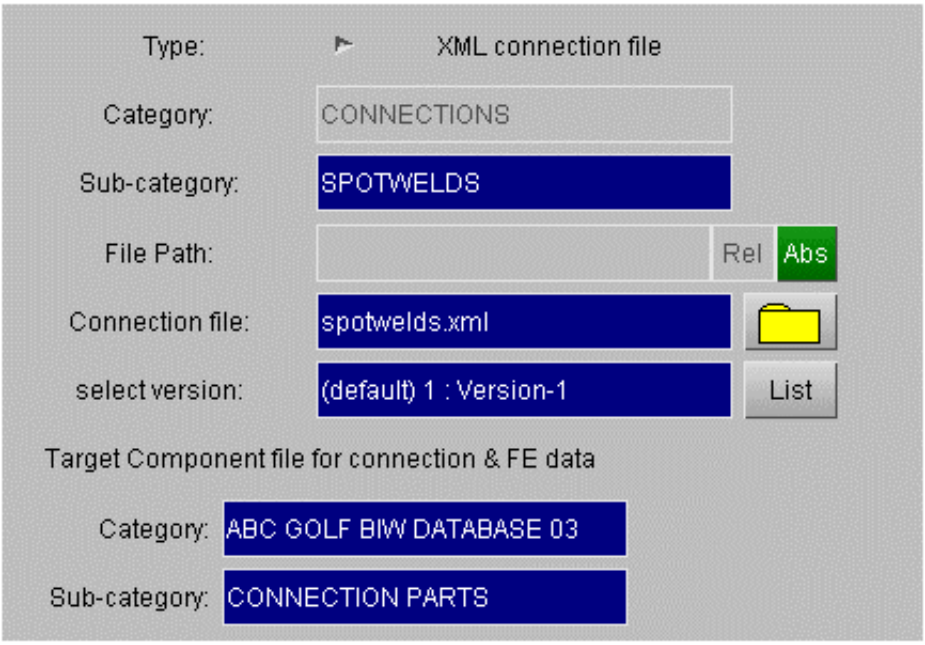


3.15.8 Connection file as component of build

Edit of component that is an xml connection file will bring up the panel below.

The data supplied in the xml file is expected to contain a complete description of what is required to generate the connection FE. Defaults are not applicable..
Target component file. On completion of build the connection FE will be created in the include file which matches the category/sub-category set by the user "component file for connection" or, failing that, in the master file. Note that if build has been performed off the Template > Build panel and the mode is keyout master only, any target setting here will be ignored.

If you are using this method to make connections each time a model is built, you should **not** have the same connections stored as post-end data, although other connections may be.



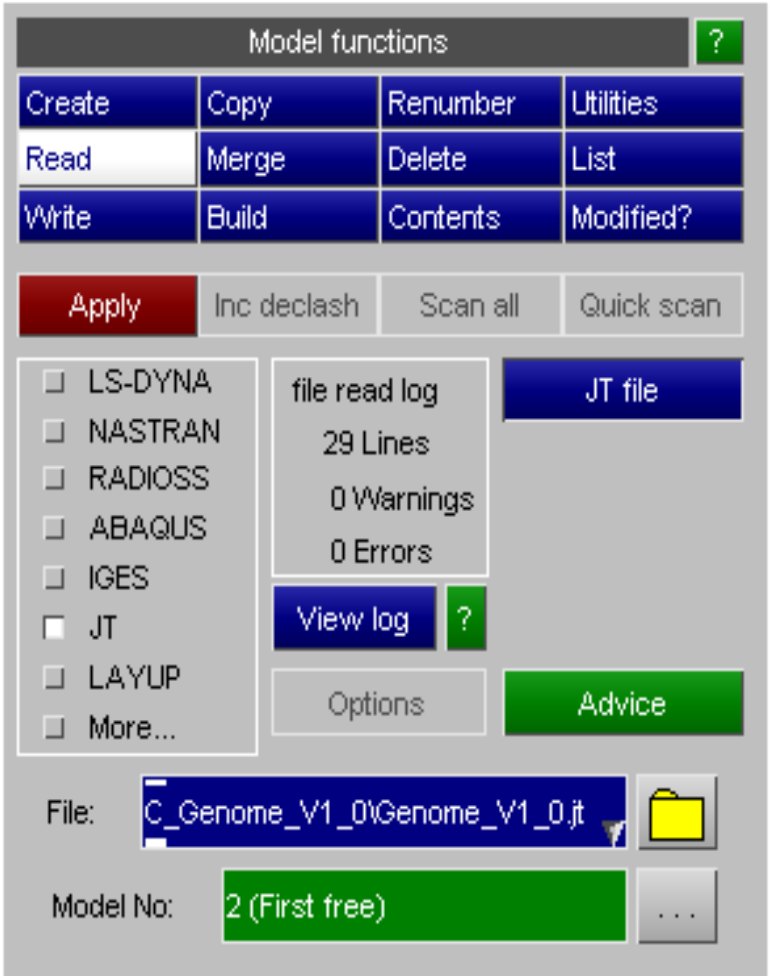
3.15.9 Reading files using a Model Database

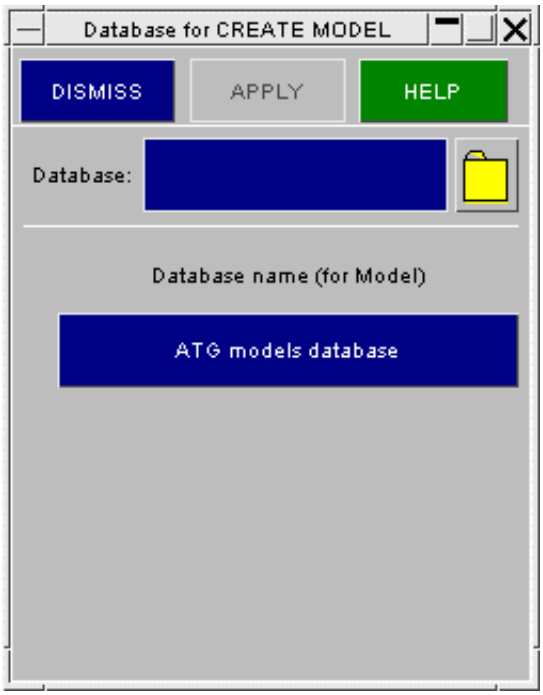
In the menu, click on **Model** and select **Read.**

In the **READ** menu press the **DATABASE** tab.

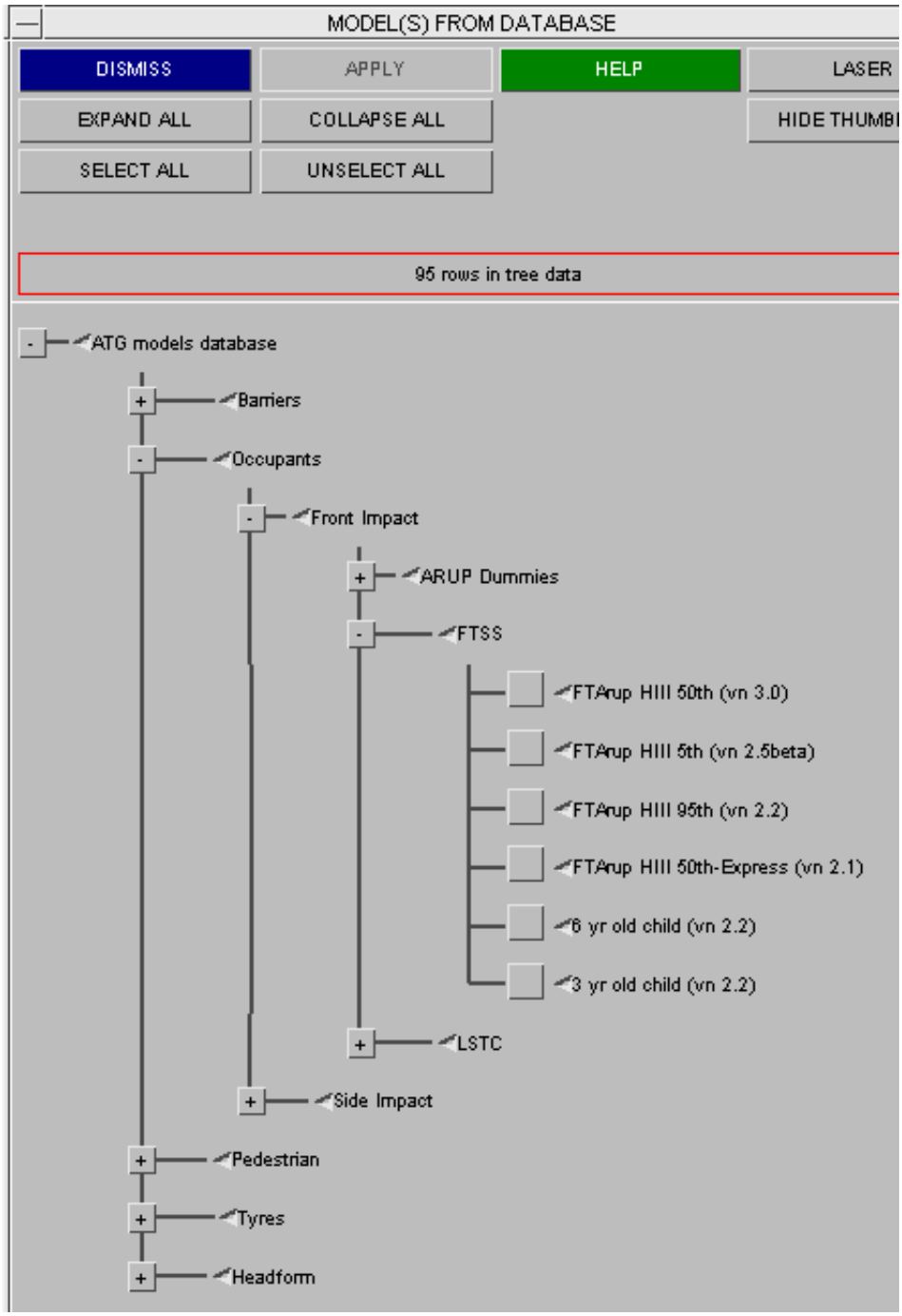
Select the database you wish to read in by inputting the name and path in the input box or using the search facility or by selecting one of the databases listed in the Database list.

Select **APPLY** to load the specified database



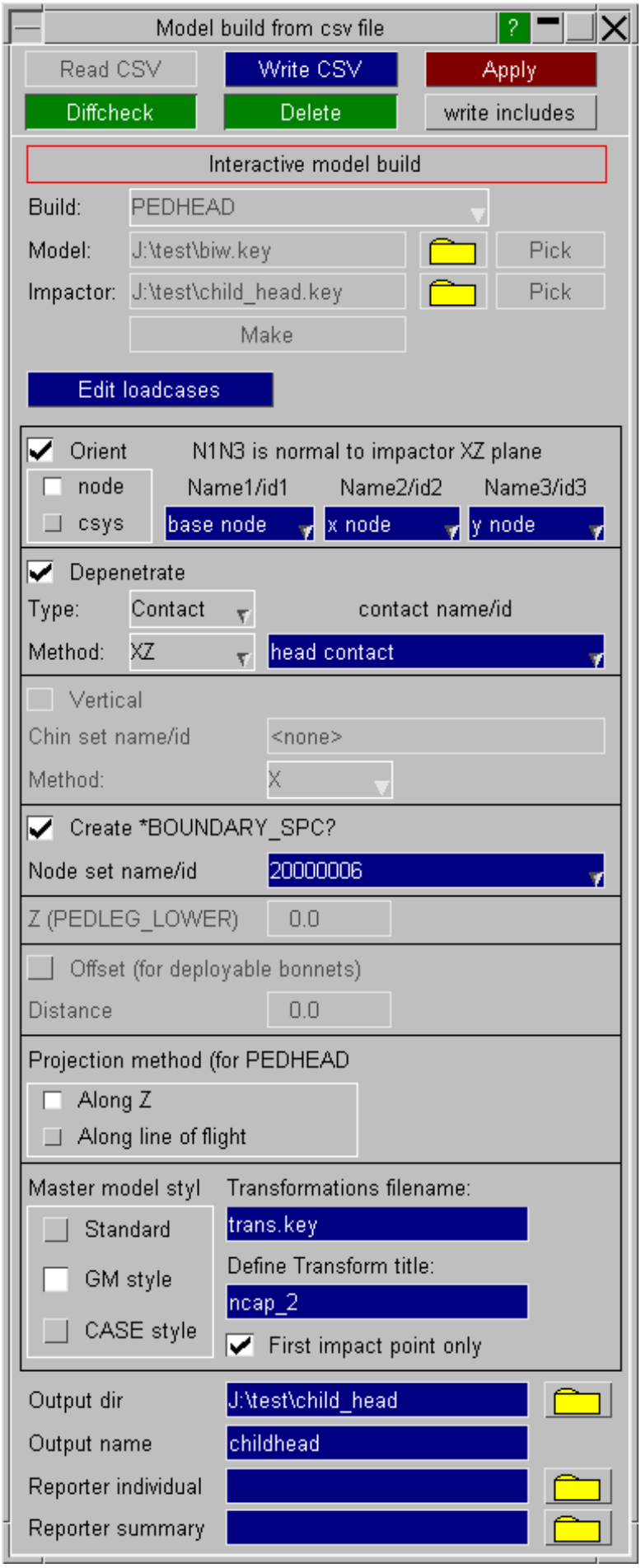


Select the file/files you wish to read in the model database window and select **APPLY**. The selected files will all be read in to **separate Models** in PRIMER. The files in this database may be compressed, as PRIMER will search for a compressed version of the file, if it fails to find the uncompressed one.



3.15.10 Building using csv targeting file - IHI, PDH build

Select **MODEL**->**BUILD**. Choose the **Build from csv targeting file** option.


Model build from csv file
Read CSV
Write CSV
Apply
Diffcheck
Delete
write includes
Interactive model build
Build:
PEDHEAD
Model:
J:\test\biw.key
Pick
Impactor:
J:\test\child_head.key
Pick
Make
Edit loadcases
Orient
N1N3 is normal to impactor XZ plane
node
csys
Name1/id1
Name2/id2
Name3/id3
base node
x node
y node
Depenetrate
Type:
Contact
contact name/id
Method:
XZ
head contact
Vertical
Chin set name/id
<none>
Method:
X
Create *BOUNDARY_SPC?
Node set name/id
20000006
Z (PEDLEG_LOWER)
0.0
Offset (for deployable bonnets)
Distance
0.0
Projection method (for PEDHEAD
Along Z
Along line of flight
Master model styl
Transformations filename:
Standard
trans.key
GM style
Define Transform title:
ncap_2
CASE style
First impact point only
Output dir
J:\test\child_head
Output name
childhead
Reporter individual
Reporter summary

A model and an impactor can be read using the appropriate text boxes or file selectors. Alternatively, an existing CSV file can be read in. A model and an impactor are automatically located. Selecting the **Make** button will merge these into the active model.

The following templates are available, of which one can be chosen:

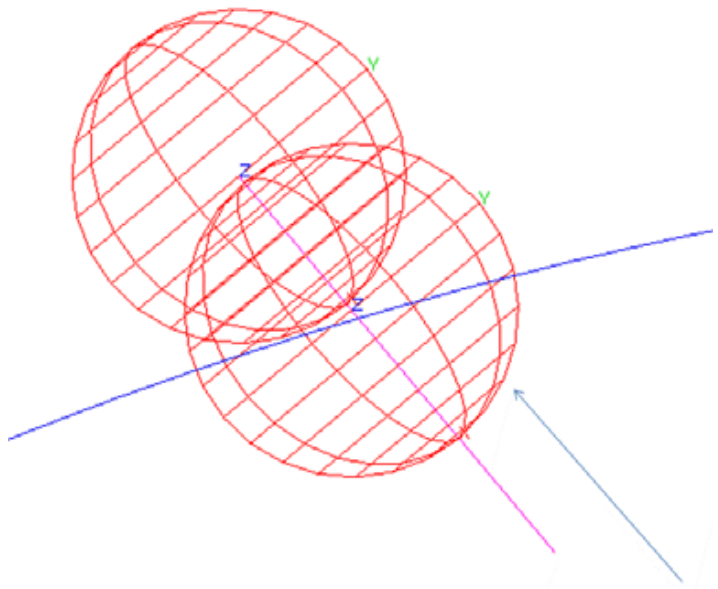
- IHI
- PEDHEAD
- PEDHEAD_ANGLE
- PEDLEG_LOWER
- PEDLEG_UPPER
- PEDLEG_UPPER_2
- GENERAL_TRANSLATE
- GENERAL_TRANSLATE_ROTATE
- GENERAL_TRANSLATE_TRIAD
- GENERAL_TRANSLATE_VECTOR

Refer to Appendix XIV for more information about these templates.

Orientation and depenetration options, root directory, output file name, reporter individual template, and reporter summary template can also be specified using appropriate text-boxes/selectors.

Note on IHI positioning. Rather than just positioning to a set vertical angle, Primer can now automatically position the IHI headform to the maximum vertical angle. The process is positioning the headform at zero vertical angle, rolling the headform down until the chin touches the trim, then rotating the headform back by a set back angle. The user needs to specify a shell set that represents the chin of the headform. The user can choose the method of head depenetration when rotating.

With the default depenetration method ’X’, the headform will roll off the target point as it would in reality.



Using the ’XZ’ or ’XYZ’ setting, Primer will attempt to move the headform back towards the target point after each rotation iteration.

The back angle (Bangle) is set on the loadcase panel for IHI (see below). On this panel the user must also specify that the loadcase uses the auto-vertical method. When the auto-vertical method is used, the vertical angle specified (Vangle) is the maximum angle the headform will rotate to when carrying out the automatic process.

An additional offset can be specified for the impactor for ’PEDHEAD’ and ’PEDHEAD_ANGLE’ types. The impactor will be moved back along the line of flight by the user-specified distance once the build operation is complete.

The Z-coordinate of the impactor is fixed for the PEDLEG_LOWER type. As a result, Z is specified globally and not as a part of the loadcases.

Two possible projection methods are available for the PEDHEAD build type. The default method (Along Z) will generate target Z by projecting the picked point along global Z. The second option will generate target Z by projecting the picked point along the reversed line of flight.

A root directory, output file name, reporter individual template, and reporter reporter summary template can also be specified using textboxes/selectors.

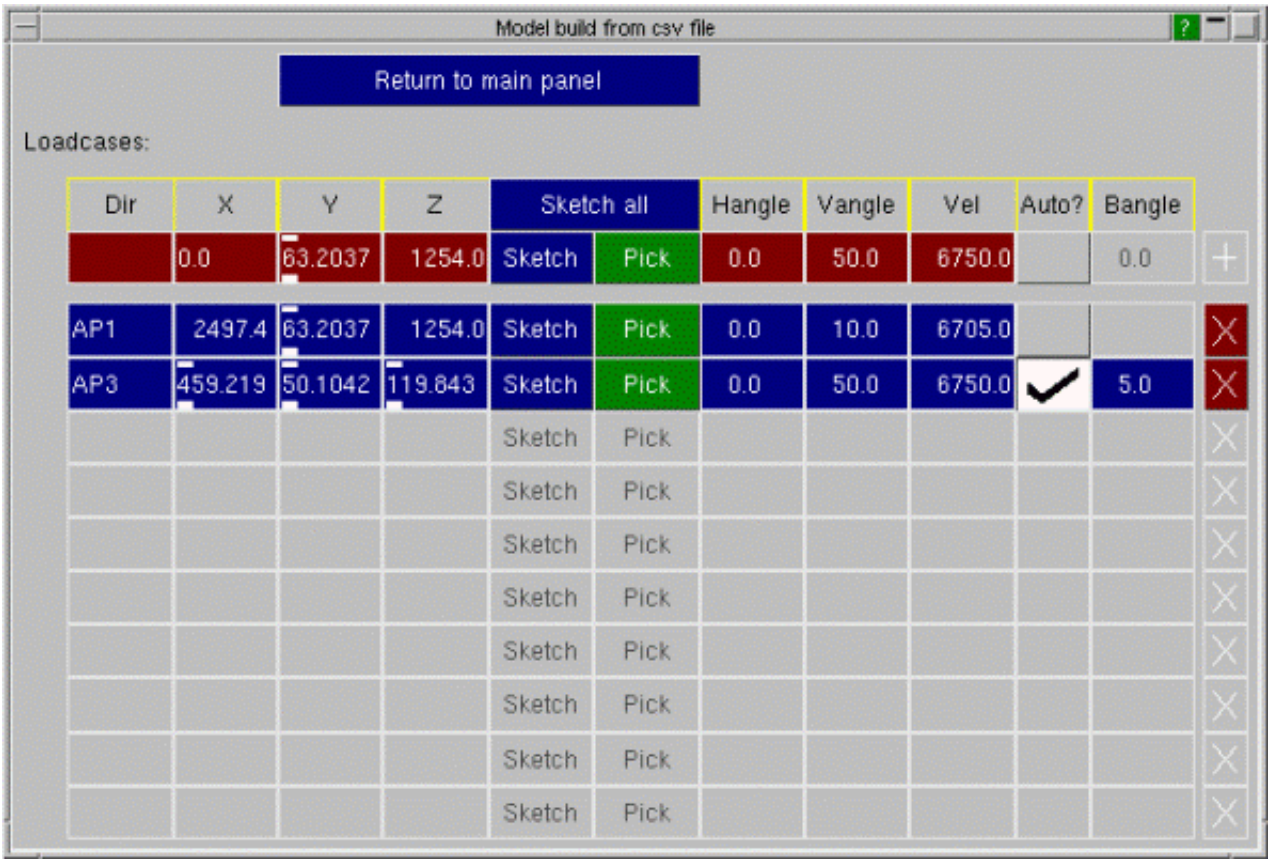
One of three master model styles can be chosen:

- Standard: Each *DEFINE_TRANSFORMATION definition is written to the corresponding master model.
- GM style: All *DEFINE_TRANSFORMATION definitions are written to a common user-defined file. Each *DEFINE_TRANSFORMATION definition is given a unique label which is equivalent to the directory name if valid. Each master model then refers to the appropriate *DEFINE_TRANSFORMATION by its label. Also, a couple of extra transformations are added such that the last translation represents the aim point. Additionally, a title string may be specified. The tile for each *DEFINE_TRANSFORMATION card would be a combination of the label, the aforementioned string, and the depenetration type.
- CASE style: A single master model is written. Each *DEFINE_TRANSFORMATION and the corresponding *INCLUDE_TRANSFORM definition is specified using a *CASE definition.

The 'First impact point only' option is specific to 'GM style'. It will generate *DEFINE_TRANSFORMATION information for each loadcase (single file as above) but will only write out the first master model.

*BOUNDARY_SPC cards can now be generated on the fly. These are created using a user-defined node set. All degrees of freedom will be restrained in the resultant card.

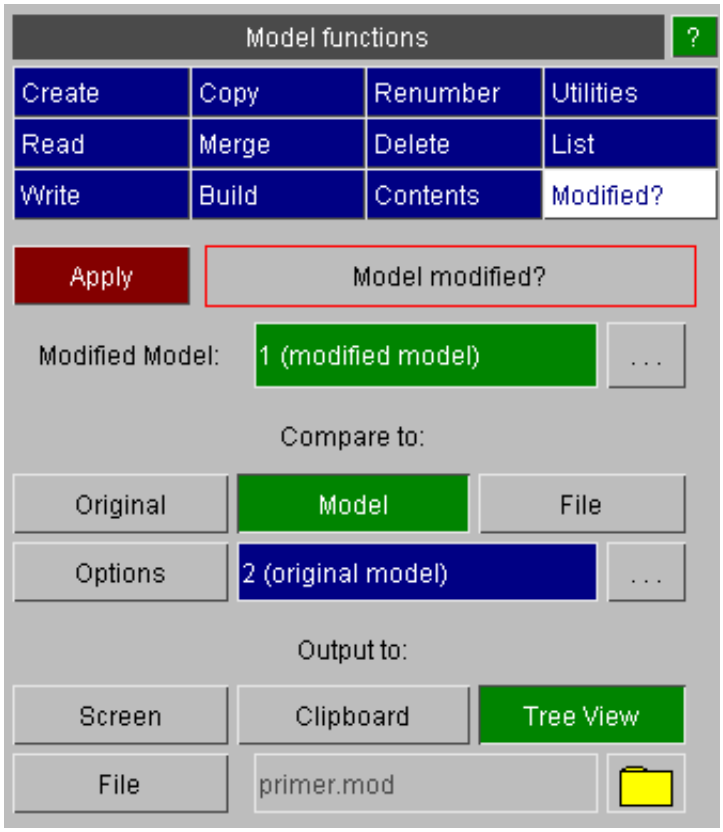
Load-cases can be specified by selecting the **Edit Load-case** button



It is recommended that users write the CSV file out before proceeding with the model build. A model save operation might also be necessary in certain cases.

3.16 Model Modified

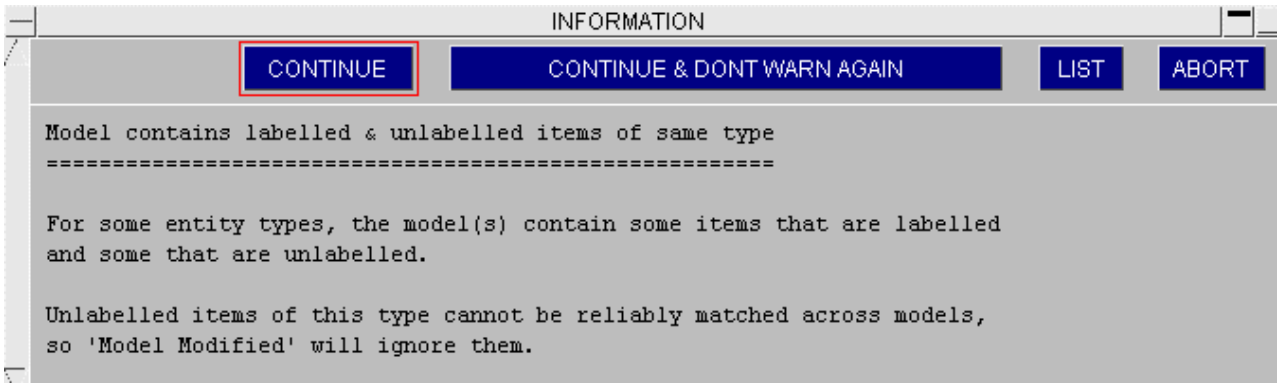
The model modified function allows you to:
- see if a model has changed (compare to original)
- compare two models (modified model vs original model)
- compare a model to a file



By default the output is displayed in a **Tree View** as described below, but it can also be sent in the form of listing to the **Screen** or to a **File**.

Items which have been changed or created in the modified file can be put on to the **Clipboard** so you can view/modify them as required.

Primer will report items that have been created in the modified model (**only in modified model**), items that have been deleted from the original model (**only in original model**) and items that have been matched across models which have been modified (**differ**). For labelled items match across models is trivial. For unlabelled items this is done by trying to match the data on the cards. It is not always possible to tell whether an unlabelled item has been modified or created/deleted. There is a particular difficulty with types which admit of both labelled and unlabelled items (e.g. CONTACT). If all are labelled there is no problem. If they are all unlabelled, they will be treated the same as an unlabelled type. If some are labelled and some unlabelled, Primer will give a warning message and decline to treat the unlabelled items.

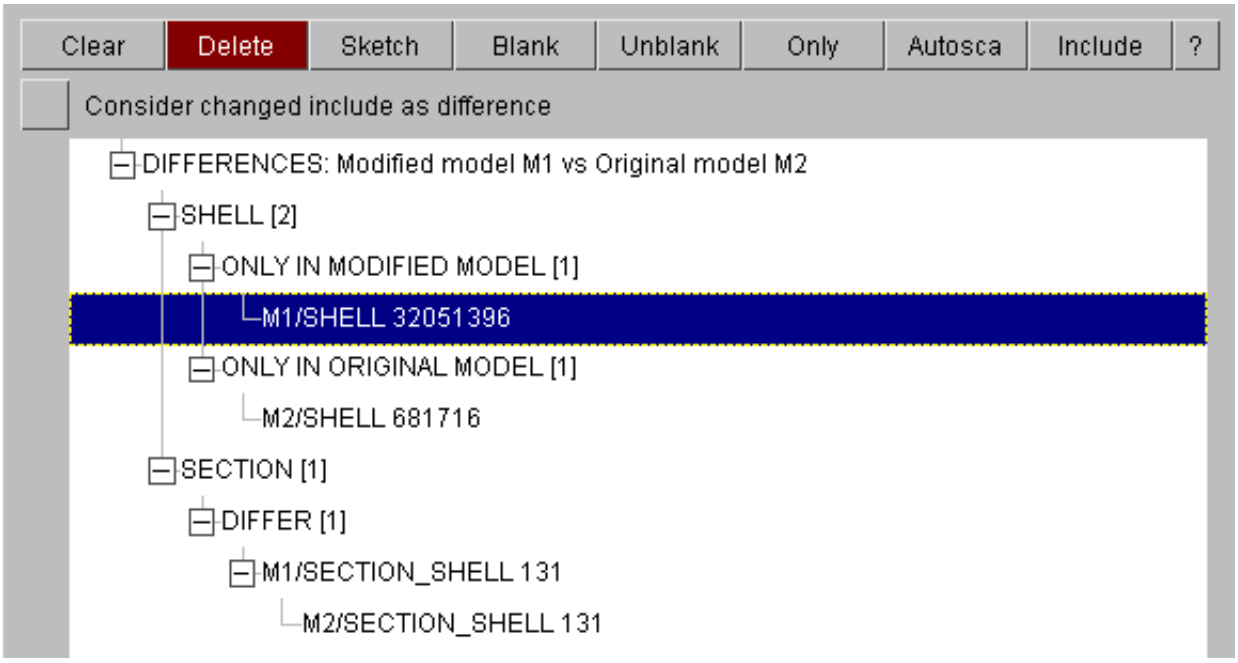


This function (in compare to original mode) is also available from the **Find modified** button in the **Include** tree (if >1model in memory use the drop-down off the model tag) or when selecting include files to write keyword files. Any include files which have changed are highlighted so they can be written out.

3.16.1 Comparing one model to another model

Comparing two models in memory is the preferred method as all items concerned (including those deleted from the modified model) are available for interrogation (and possible manipulation) on the modified tree.

In this example a shell has been created, a shell has been deleted and a section card has been changed.

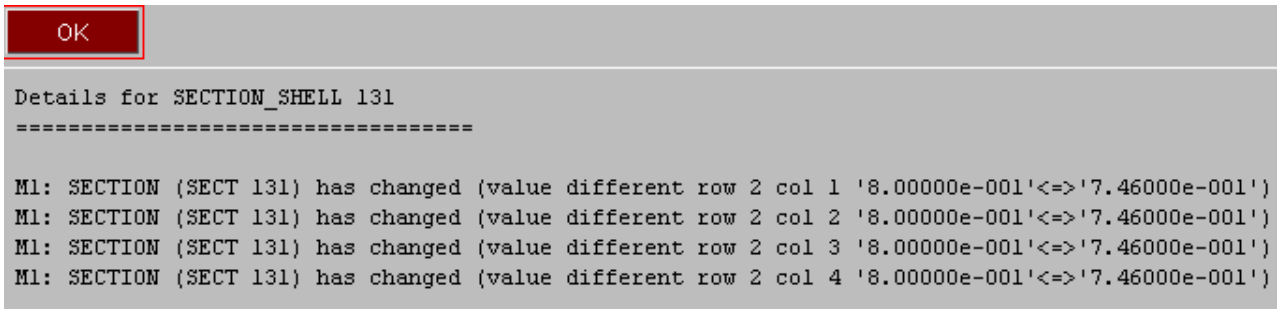


Options for sketch, edit, blank, etc. are available from the drop-down.

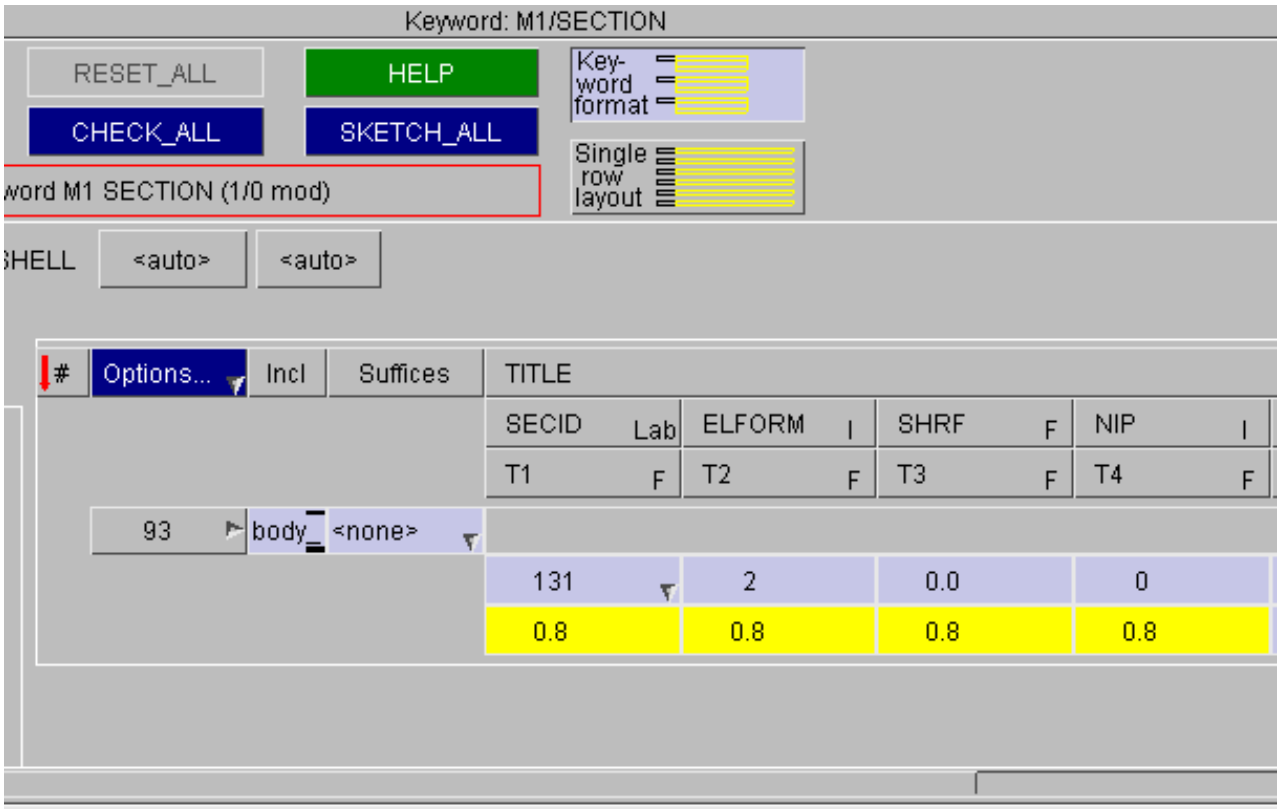
The drop-down from the created shell(s) in the modified model gives the option to **Delete** or put the shells into a set - **make_SET_SHELL**

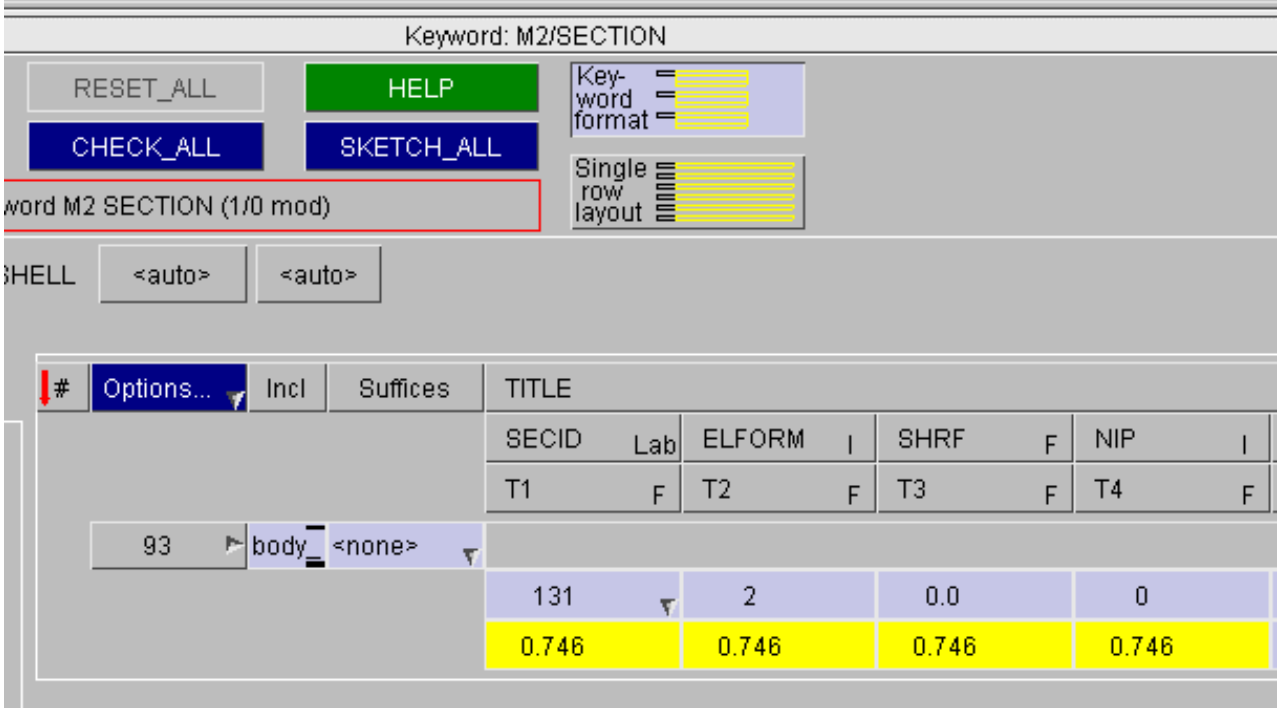
The drop-down from the shell(s) in the original model, deleted from the modified model gives the option to copy them back in **Copy_M2->M1**

For the modified section card, the **Details** drop-down shows what has changed.

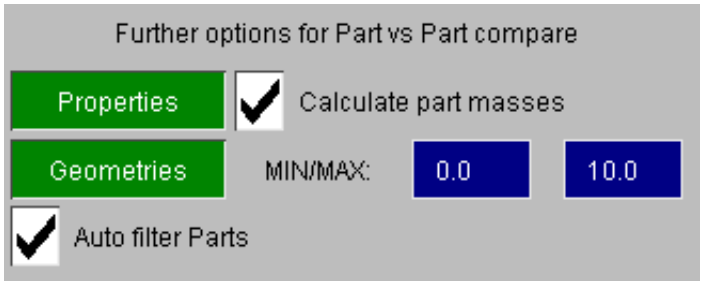


The keyword editor may also be invoked in a special mode which will highlight the changes.





Additional options for parts

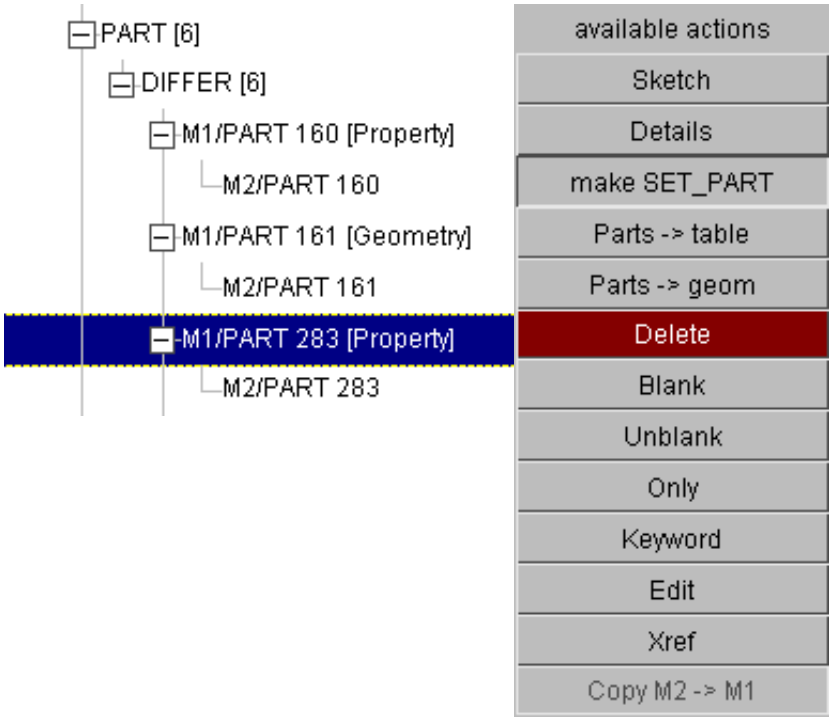


Additionally with this method part properties and geometries may be compared by using the part compare function, see section 7.3.2

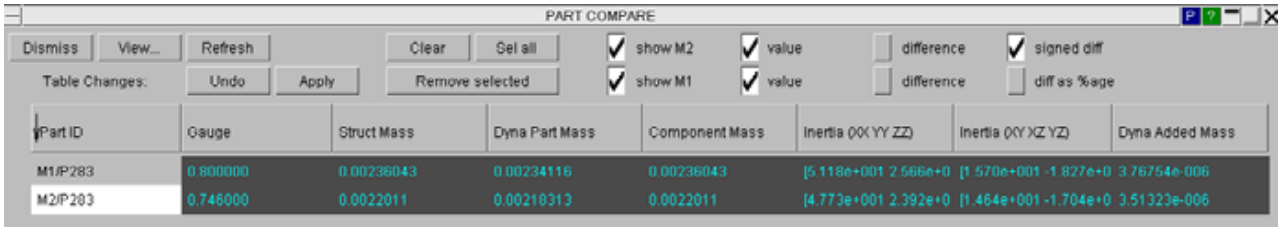
Properties If active, all properties available on part table (mass properties can be switched off) will be calculated for each pair of matched parts. Any parts for which properties differ will be reported on the tree.

Geometries This function will run a contact type check to detect gaps (using defined min/max values) between matched parts (of type shell only). The option **Auto filter parts** is recommended to block the test (which can take a few secs) for part pairs which are unlikely to be geometrically different (same element count, same geometric CofG and same surface area).

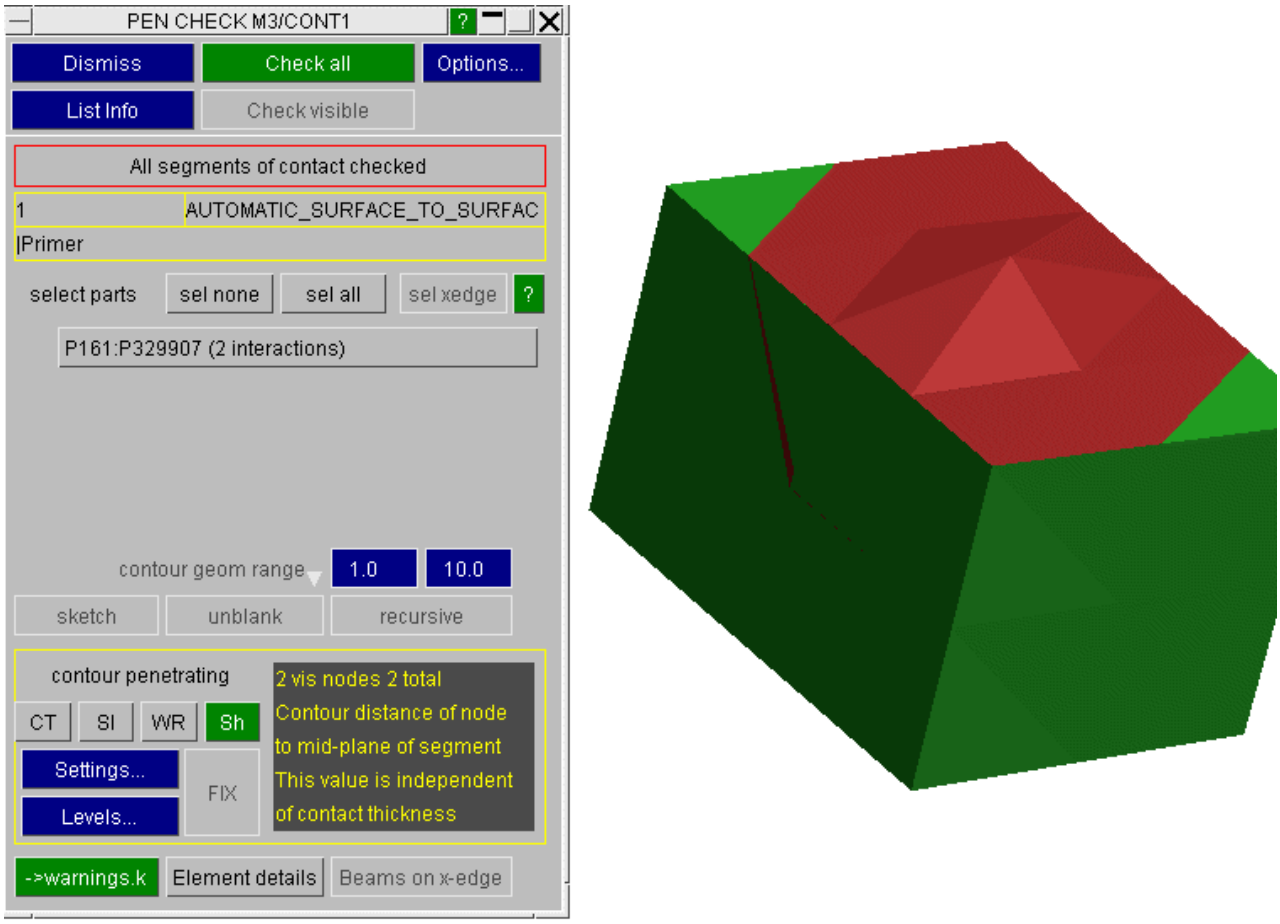
These options enable the user to readily identify parts which have been changed as result of change to another keyword, such as *SECTION or *NODE.



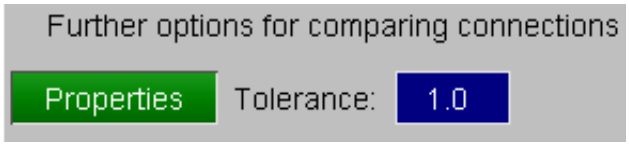
For parts with property differences, the **Parts -> table** function will give a detailed description.



For parts with geometric differences, the **Parts -> geom** function will invoke a display where the difference can be contoured. See section 7.3.3. This must be dismissed to return to the tree viewer.

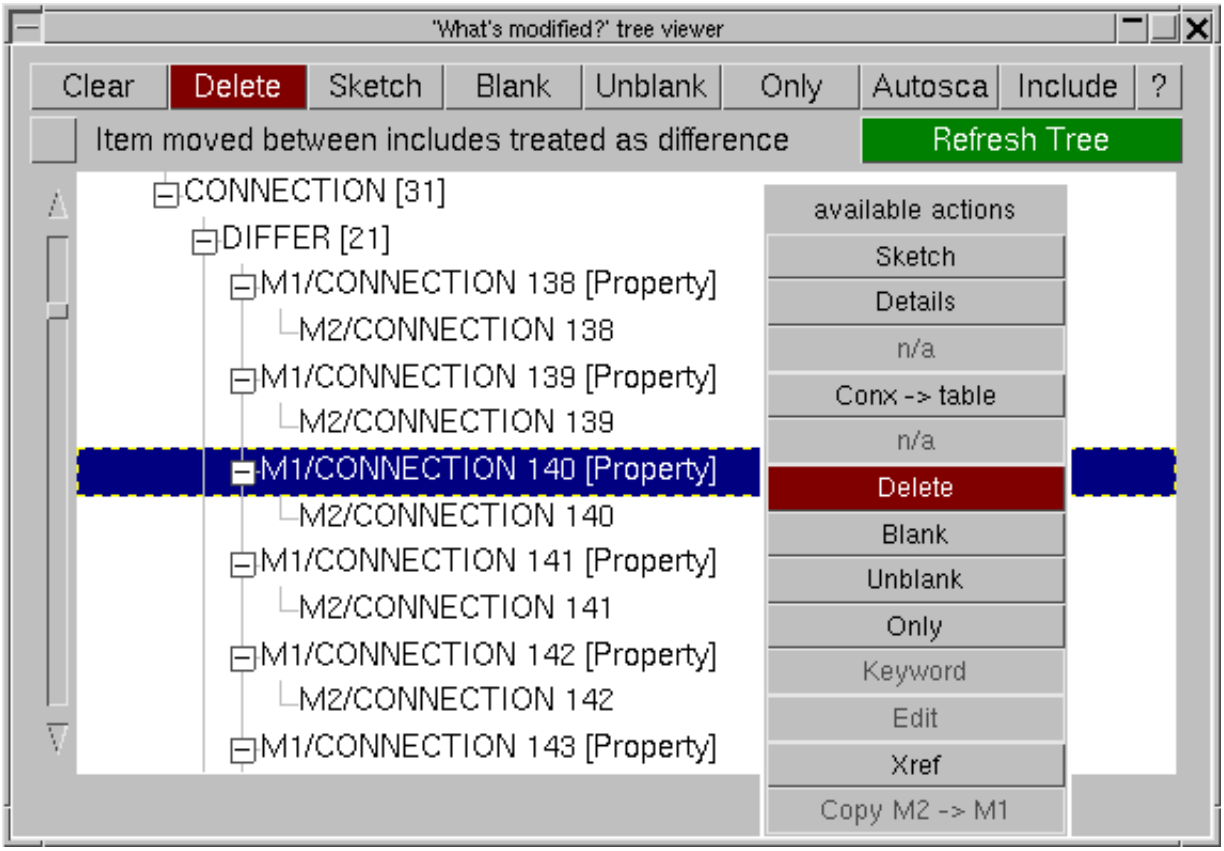


Additional options for connections

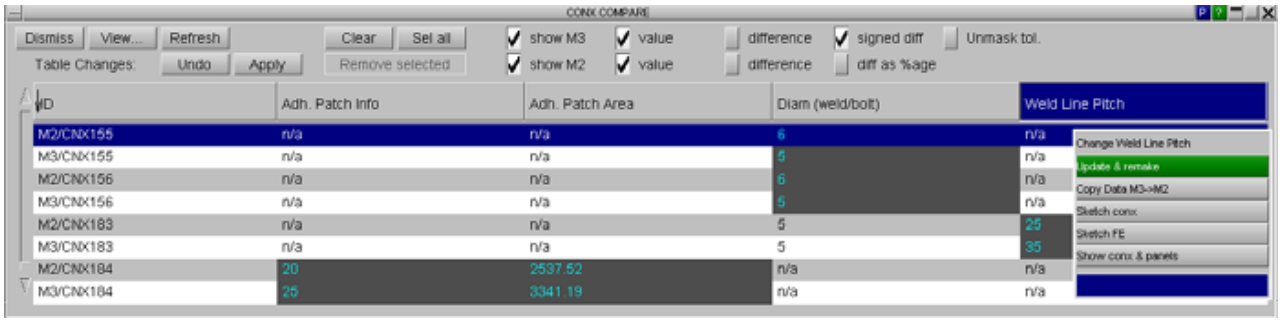


Properties If active, all properties available on connection table will be calculated for each pair of matched connections. Any connections for which properties differ will be reported on the tree.

Tolerance is the spatial tolerance to be used while matching connections



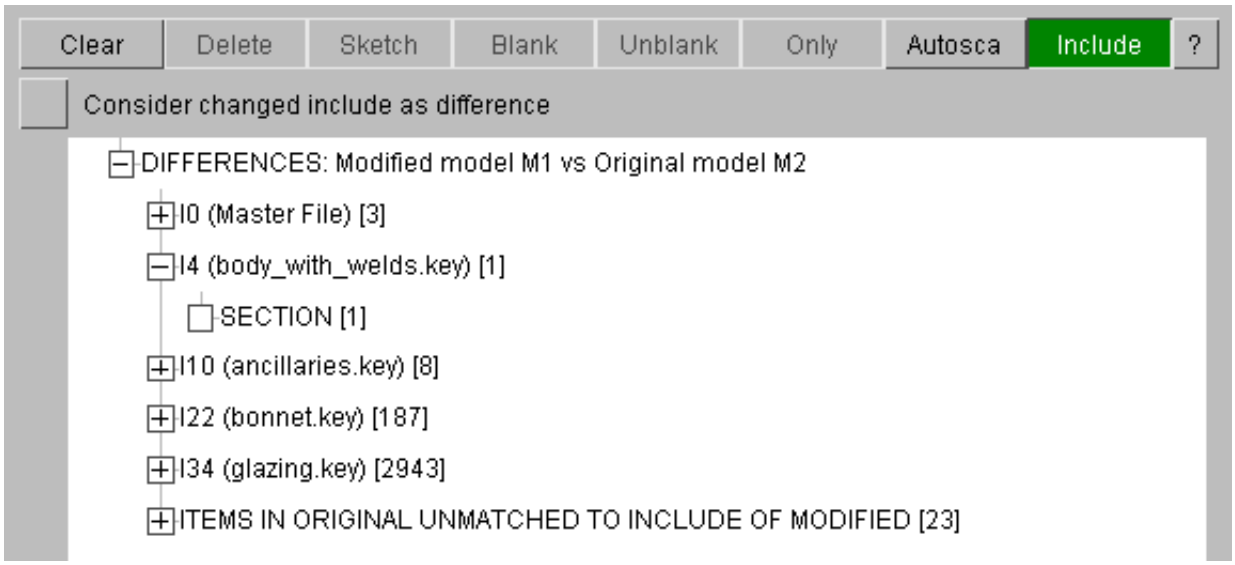
For connections with property differences, the **Conx -> table** function will give a detailed description.



For detailed use of connection compare table, see section 6.12.15

Displaying modified items by include

The modified items may be displayed under their include file if the **include** button is activated. This will be the list of includes derived from the modified file, so creation of include files presents no difficulty. If, however, an include has been deleted (or renamed) items will appear under the heading **ITEMS IN ORIGINAL MODEL UNMATCHED TO INCLUDE OF MODIFIED**.



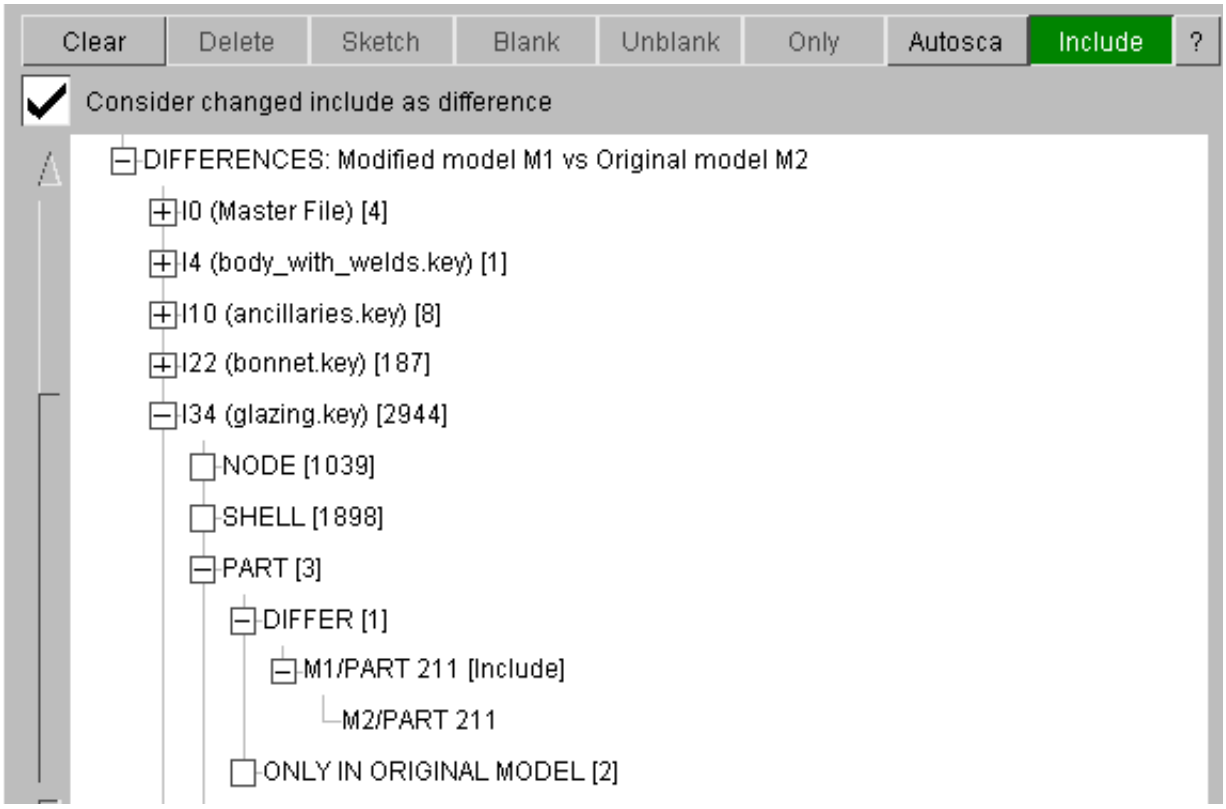
Item has changed include

By default Primer does not treat a change of include as a difference, so items which are the same but have simply been moved to another include will not appear on the tree.

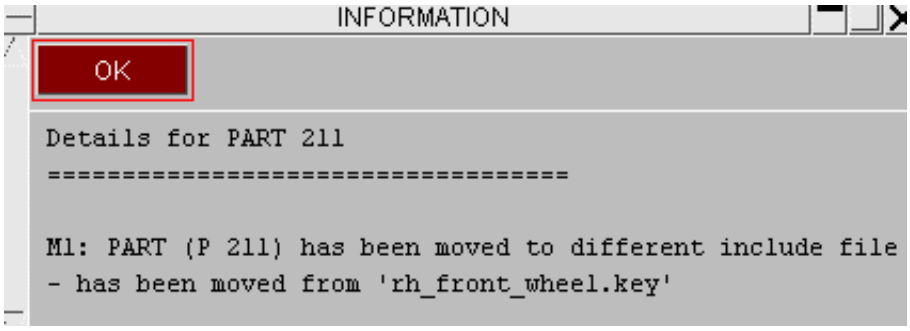
Thus a model with no includes may be reported as identical to one split into many includes.

By activating the option Consider changed include as difference, the tree view will show them (in addition to the changed items).

Primer matches includes across models by using the name, the order in which they have been read is irrelevant. Only if a model contains multiple includes of the same name (these will be *INCLUDE_TRANSFORM) will the order be significant and a change of order between the models may give rise to spurious reports of include difference.



The Details drop-down will give the include from which the item has been moved.



3.16.2 Comparing a model to original or to a file

To compare a model to the original version of the file (i.e. when you read the file into PRIMER) select **Original** or **File**.

PRIMER will reread the original model/selected file into the next free model and then compare the 2 models.

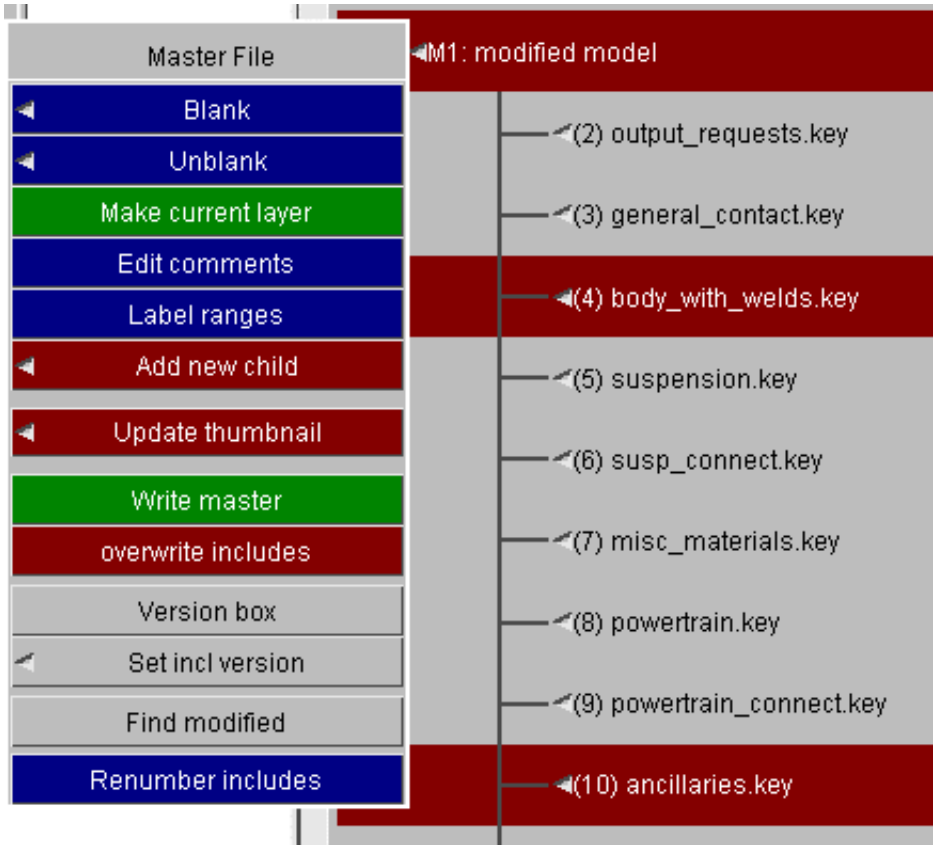
As in compare to file, the 2nd model is not plotted and deleted as soon as the comparison has completed. This means that the further options for part comparison are not available and deleted items can only be reported (as opposed to interrogated) on the modified tree. Otherwise the process is much the same as model vs model comparison.

3.16.3 Comparing individual include files

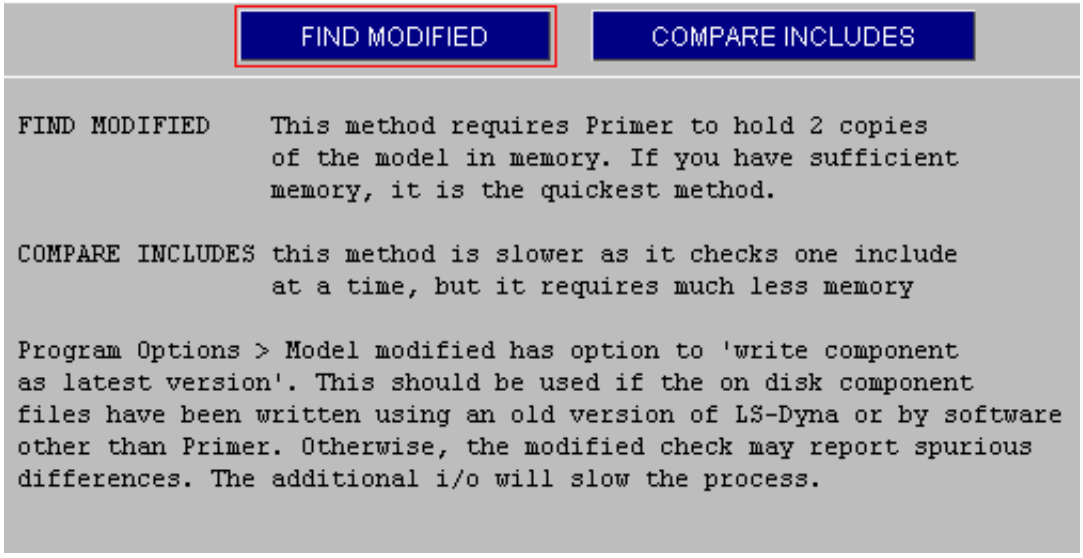
Find modified described above works by reading the original file and comparing it to the current model in memory. This methodology cannot be applied to models which have been **built in Primer**. We would need to record how each model is built so we could repeat the process to construct the original - this is not practical.

Find modified also requires that we hold 2 copies of the model in memory which may not be possible for very large models on machines with limited amount of memory.

Instead, we have an alternate function available off the model drop-down on the include tree. Note - the **Find modified** button always runs the normal function.



You will be given the option of running the normal **Find Modified** or **Compare Include**.

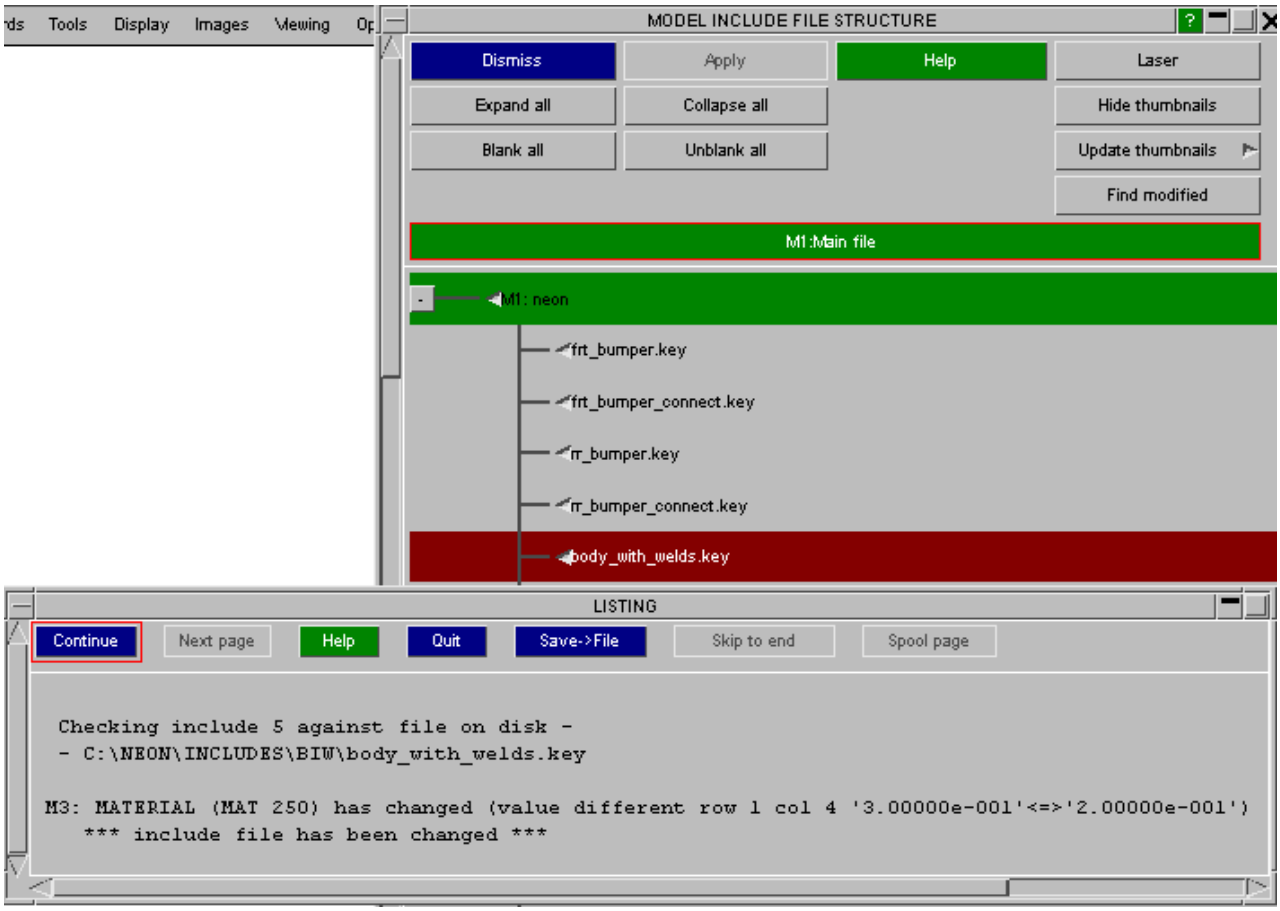


Additionally, **Compare include** may be run for an individual include (off the include drop-down).

Compare Include works by writing the current include to scratch area and reads it back in to form model A and then reads the original include file to form model B. Models A and B are then compared using the model modified function which effectively performs a dxdiff between them and reports any differences.

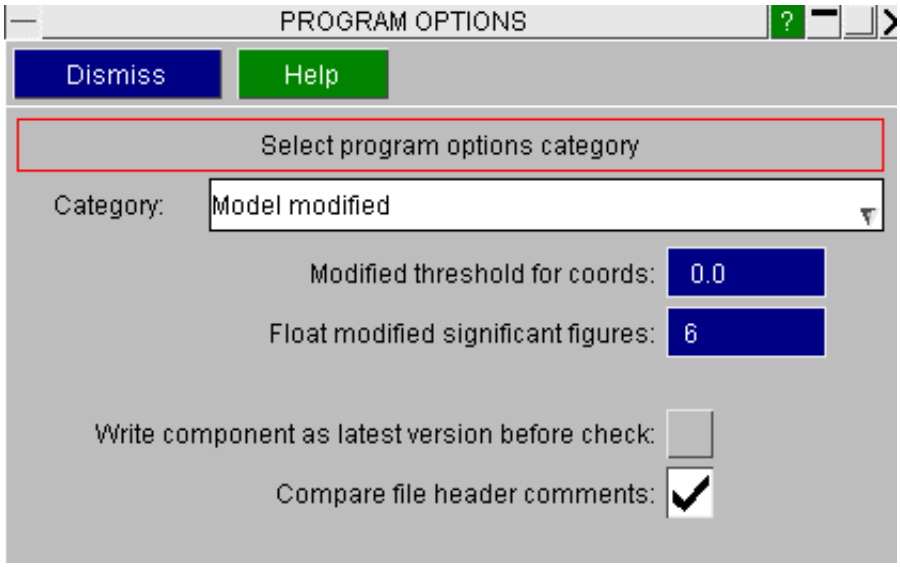
The function involves intensive disk I/O and so if run on all the includes of a large model may take a while to complete. The original model modified function will achieve the same result much more quickly (when it can be applied) so is recommended.

For all models this function is very useful for interrogating an individual include to see the details of what has changed. With a non-built model you can use **Compare include** to detect which include files have changed, and then extract the details for the include of interest using compare to disk. Although they work in different ways, both processes should always report the same differences.



3.16.4 Options for comparing models

The Options button on the modified panel will give direct access to the appropriate program options panel.



Comparing floating point values

By default, floating point numbers are compared to 6 sig fig. This can be reduced at the user's discretion and will affect the comparison of all floating point numbers.

Coordinates (on nodes, connections & airbag reference geometry) are special cases which admit of an absolute difference threshold which can be set by the user. This is particularly useful to remove the spurious differences when comparing models with *INCLUDE_TRANSFORM. These arise due to rounding when the transforms are applied and unapplied.

Avoiding spurious differences

Write component as latest version before check This option applies only when we are comparing include files. If this option is set Primer will read the component file, write it using the latest output version and then re-read it. The I/O overhead is considerable. If the component files are rather old or have been written by software other than Primer this may be worth doing to avoid spurious difference reports. If the component file has been written from Primer relatively recently there is no need to do this, hence the default if off.

File header comments

By default Primer will compare file header comments for the master file and the includes at the beginning of the model modified process and warn if these appear to differ. This includes a check on include file label range definitions. You may switch this off.

3.17 Model POST

Version 15 of PRIMER can link with the D3PLOT and T/HIS post processors via shared memory, making it possible to exchange commands and data between the programmes.

This makes "Pre" functionality available in post-processors, for example editing the original keyword definitions; and "Post" functionality becomes available in PRIMER, for example extracting deformed coordinates.

In the case of D3PLOT graphics can be synchronised between the two codes, so that dynamic viewing, blanking, colours, cut-sections, etc are updated simultaneously at both ends.

By default no link takes place, but it can be opened in any of the following ways:

- A running PRIMER session starts a new D3PLOT and/or T/HIS session using the stipulated model
- A running D3PLOT or T/HIS session starts a new PRIMER session using the stipulated model.
  and
- Once a link is established, in either of the modes above, further models can be opened and linked at will.

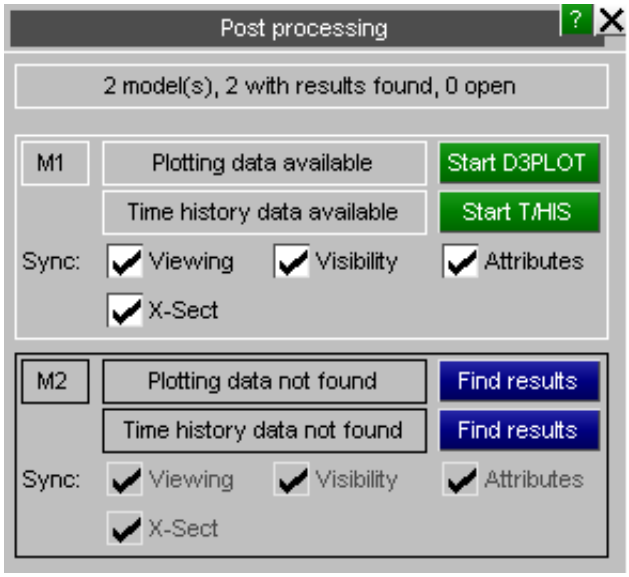
The link is symmetrical and bi-directional, with no concept of parent or child, and it can be closed at any time leaving both codes running autonomously. What you can’t do at present is to link an already running, PRIMER, D3PLOT or T/HIS session with another autonomous session unless it was opened by the other code first.

Terminal output from the newly opened session will be sent to the following locations:

On Windows	Each session will always open its own, separate MS-DOS terminal (console) window, which will receive all "Terminal" output.
On Linux	If the parent session was started directly from a terminal window then all output from child processes will be sent to that window. If the parent was started via a separate "xterm" window, the default when using the "Oasys shell", then each child process will also open its own, separate xterm window to act as a terminal (console) window.

3.17.1 The **POST** panel:

Initial status



For each model currently in the database PRIMER scans the directory containing the top (master) keyword file looking for results based on input filename, then for each model the initial status will be one of:

Option	Status of model	Action performed
Start D3PLOT Start T/HIS	Found graphical post files Found time-history files	Launch D3PLOT with these results Launch T/HIS with these results
Find results	No sesults files found	You must browse for results before a post-processor can be launched.

In this example results have been found for the first model M1, but not for the second M2. No linked post-processor has been opened yet.

(There is a corresponding Pre panel in D3PLOT and T/HIS, with similar layout and functionality.)

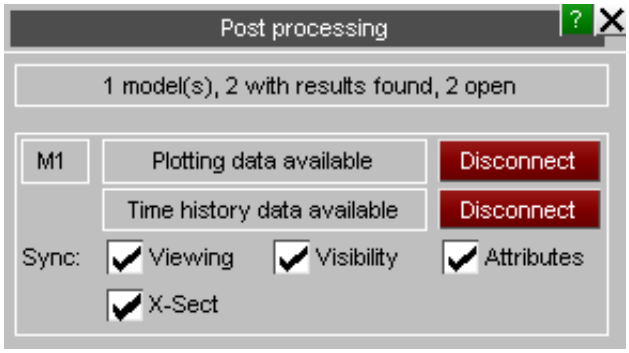
Filename search logic

The logic that PRIMER uses to search for files is based on input master keyword file ***path/name***.k*. In directory ***path*** it searches for:

For D3PLOT it searches for	***name***.ptf d3plot
For T/HIS it searches for	***name***.thf, ***name***.xtf d3thdt binout* Any ascii database names, eg abstat, glstat, etc.

The first match, in the order above, in any category is treated as "results found".

Status once a post-processor is opened



Once a child D3PLOT and/or T/HIS process has been started there is a shared memory link between those codes and this PRIMER session.

Each process runs autonomously, and if you **Disconnect** D3PLOT or T/HIS they will continue to run in the normal way. Similarly if you disconnect or terminate those codes locally PRIMER will detect this, clean up the shared memory link and continue to run normally.

Effects of linking and unlinking models:

In all cases:

- Linking or disconnecting a model does not affect that model's status in either programme, both D3PLOT and/or T/HIS and PRIMER will continue to run normally.
- Models may be disconnected and reconnected at will.
- When a model is deleted in PRIMER it is implicitly disconnected in D3PLOT and/or T/HIS, but will not be deleted from those codes. Similarly if a model is deleted in D3PLOT or T/HIS is will be disconnected from PRIMER, but not deleted.
- The link logic attempts to keep model numbers the same in PRIMER, D3PLOT and T/HIS, however it is possible to defeat this by opening additional models in one programme but not the other. Doing so may cause the linkage to fail in some respects (so don't do it!) .
- The POST panel can be opened or closed at will without affecting the status of linked models, it simply provides feedback about the current status and attributes of linked models.

3.17.2 Synchronising attributes.

It is possible to synchronise the following attributes across the link:

Attribute	What it does
Viewing	The current view: scale, orientation, position on screen, perspective settings. Includes the effect of dynamic viewing.
Visibility	Blanking and entity visibility settings
Attributes	Item colour, transparency and drawing mode (current, shaded, etc)
Xsec	Cut-sections: location, orientation, setting. Includes the effect of dragging the section.

Symmetry:

All the above attributes are symmetrical. For example if viewing is synchronised then a view change in D3PLOT will affect PRIMER, and one in PRIMER will effect D3PLOT.

Switching on/off

Each attribute type can be turned on/off independently for each model. The switches themselves are symmetrical: changing a setting in the Pre panel of D3PLOT will update the same setting in the Post panel in PRIMER.

Effects of multiple models

D3PLOT may put multiple models in different windows or the same window, but PRIMER places all models in the single window. This can lead to slightly strange behaviour since rotating only a single model (of several) in its own window in D3PLOT will affect all models in PRIMER, whereas rotating a model in PRIMER will affect all windows containing linked model in D3PLOT.

3.17.3 Synchronised Operations

Both commands and data can be exchanged across the link using the following methods:

Edit Load Curve in T-HIS

Load curves in PRIMER can be sent to the linked session of T-HIS, see LOAD CURVES for more details.

4 Model visualisation

4.1 Basic Drawing commands

4.2 Data Plotting commands

4.3 Controlling model visibility

4.4 Controlling Entity Visibility and Labelling

4.5 Blanking

4.6 Dynamic labelling

4.0 Visualisation and labelling.

This section describes how to draw models, control what is drawn, and also add labels and associated data to plots. Viewing control is covered in section 9.

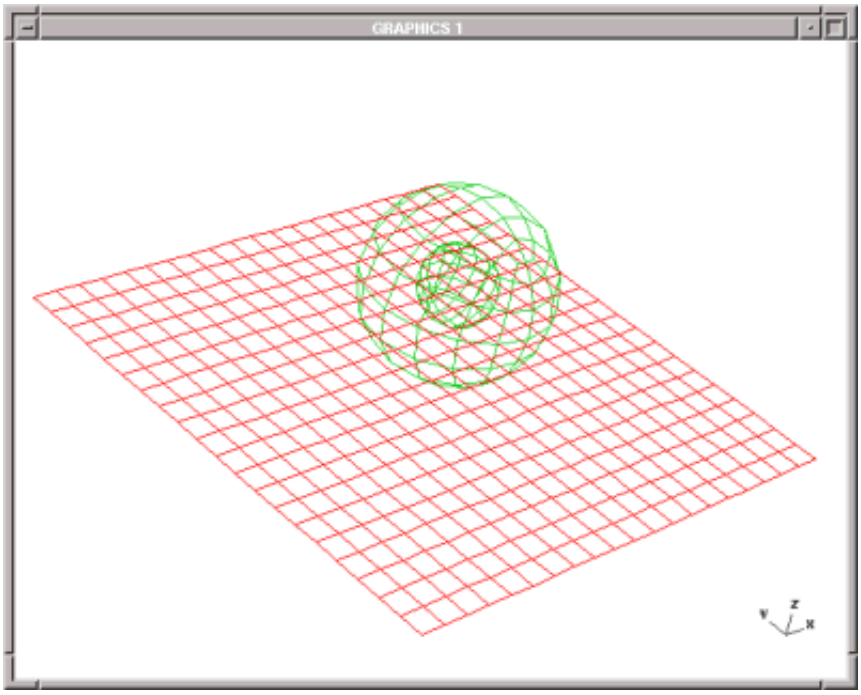
4.1 Basic drawing commands: LI(ne), HI(dden line), SH(aded image)

PRIMER is capable of drawing basic model geometry in three modes: "**LI**ne", "**HI**dden-line" and "**SH**aded".

"LINE" mode (**LI**) draws all element borders with no hidden surface removal. However the back faces of 3D elements are removed when graphics are in 2D mode (but not in 3D mode).

This figure shows an example of a line mode plot of a ball (made of solids) above a flat plate.

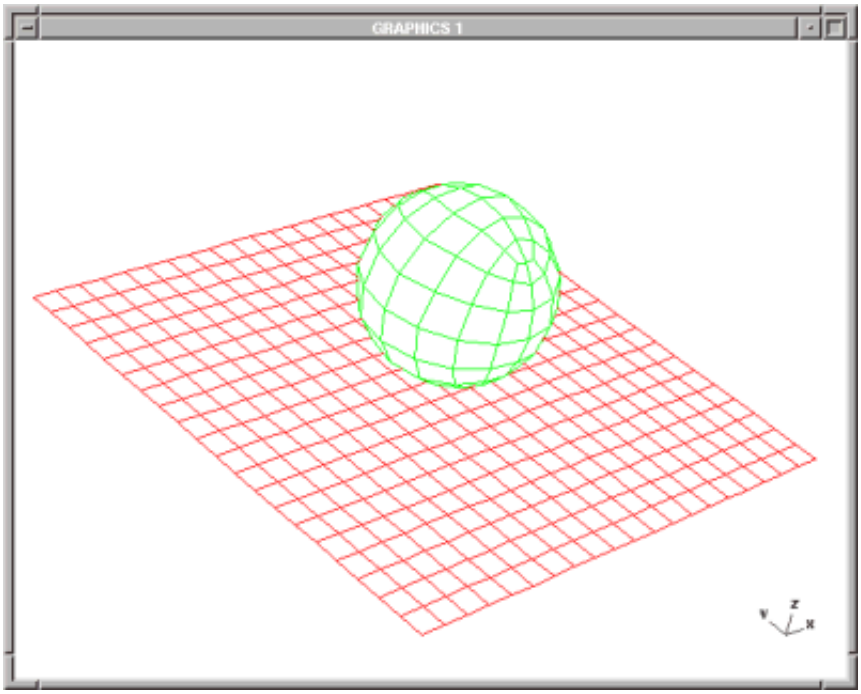
Note that no hidden surface removal has been carried out.



"Hidden-line" mode (**HI**) also draws element borders, but this time with hidden surfaces removed. (Back face removal is implicit in this.)
This figure shows an example of a hidden-line plot, with the same model

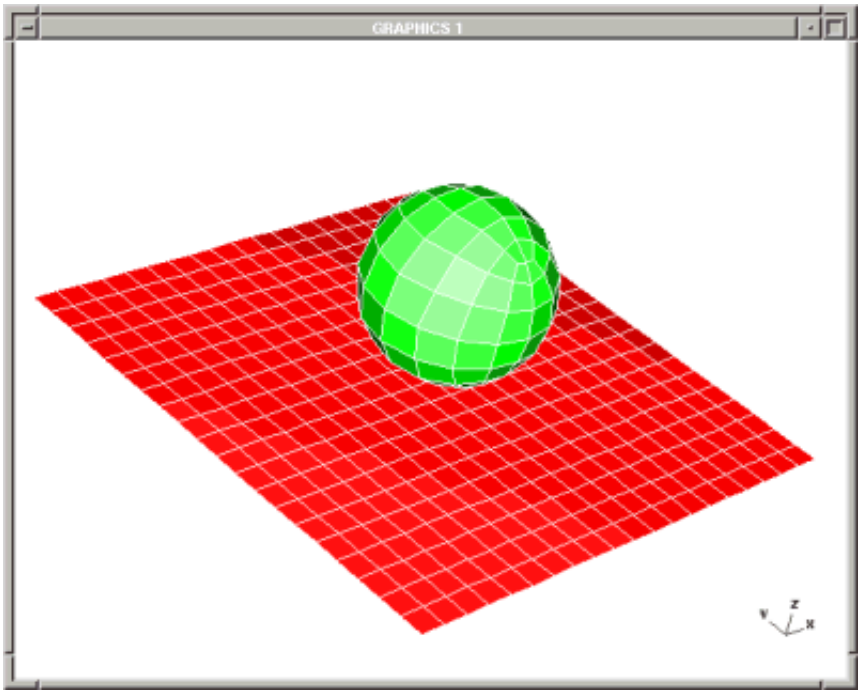
as above.
It is now obvious that the hidden surfaces and lines have been removed, and it is easy to tell that the ball lies above the plane.

Hidden surface removal requires more computation than a simple line mode plot, so it will be slower to generate. Most displays with 3D graphics protocols have a hardware "Z-buffer" which makes this process faster, but even so complex images may take an appreciable time to draw. For this reason the dynamic viewing modes, which permit real-time manipulation of the view, have a facility to drop back to line mode (or even "free-edge" mode) while the image is being moved, reverting to the normal display mode when the motion is complete. See section 9.4 for a description of how to do this.



"Shaded" (**SH**) mode also performs hidden-surface removal, but this time the surfaces of 2D and 3D elements are shaded and lit in the appropriate colours.

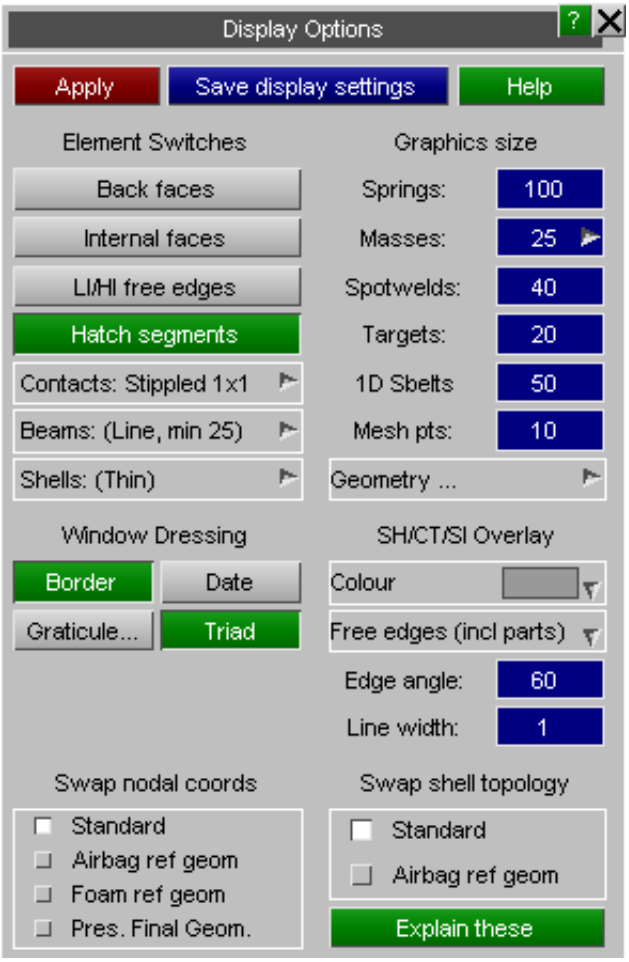
This figure shows an example of a shaded-image plot. Lighting and hidden-surface removal have both been applied, and the element borders have been overlaid on the resulting plot.



As with hidden-line plots, 3D devices with hardware assistance will generally produce these images much faster than the software alone (2D) method, but (in either mode) computation time will be longer and the ability to drop back to line or free-edge mode during dynamic viewing also applies.

Shaded, Line and Hidden plots may also be invoked with the shortcut keys S,L and H.

4.1.1 DISPLAY OPTIONS... Controlling plot parameters.



"Options" gives user control over a number of graphical features.

Most of these can be preset in the "oa_pref" file: see section 4.1.5 below. The button **Save display settings** saves the settings directly to the preference file.

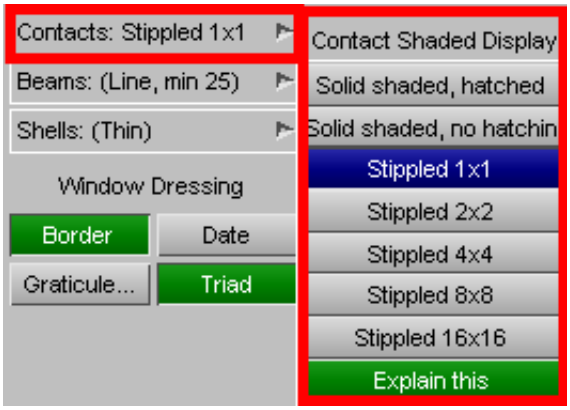
Back faces Determines whether or not the back (ie facing away from you) faces of 3D (solid and thick shell) elements are drawn.

Internal faces Determines whether or not the internal faces of 3D elements are drawn. You should only use this if you need to see them, as it slows down drawing by a large factor.

LI/HI free edges Determines whether or not the internal faces of 3D elements are drawn on **LI**ne and **HI**dden-line plots. You should only use this if you need to see them, as it slows down drawing by a large factor.

The overlay on **SH**aded and data bearing plots is controlled in **SH/CT/SI Overlay**

Hatch segments Controls how segment sets are displayed. The default is to draw a "hatched" wireframe overlay on them in order to distinguish them from ordinary 2D elements, but this can slow down graphics considerably on some platforms so it is switchable.



Contacts Historically PRIMER drew contacts using a hatched wireframe overlay but, as with segments above, this could be slow. Therefore the default in V9.3 has been changed to "**Stippled 1x1**" which is fast, and distinguishes them visually from shells by giving them a semi-transparent appearance since every other pixel is omitted.

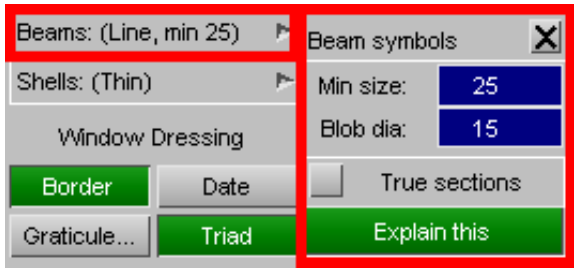
This is much faster than hatching, but may not be to the taste of all users, so a range of options is given as shown here. **Solid shaded, hatched** will give the original (slow) appearance, and **Solid shaded, no hatching** will give an opaque result visually indistinguishable from shells.

The default appearance can be changed in the "oa_pref" file using

`primer*contact_shaded_display:` *<option>*

Beams

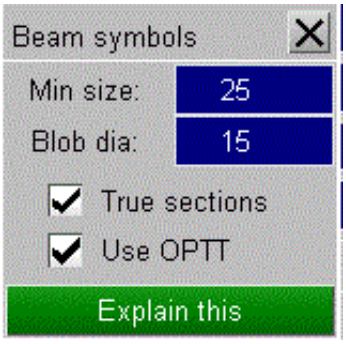
PRIMER draws ordinary (not spotweld) beams as simple lines between the 2 nodes, but when these nodes get very close together the result can be a very small point of a single pixel which is almost impossible to see.



Therefore when the distance between the 2 nodes as drawn on the screen is less than **Min size** the symbol is changed to "blobs" on each node of **Blob dia** to make them easier to see. (Both these dimensions are in screen space units.)

With **True Sections** switched on beams will be drawn with their explicit sections dimensions and orientation.

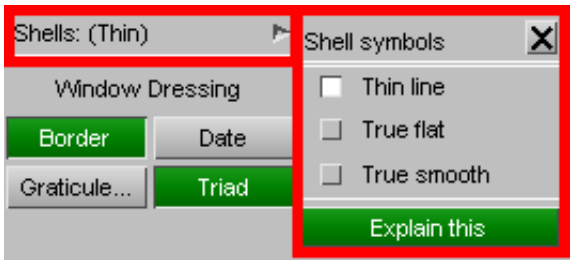
For beams where only Area, Ixx and Iyy properties are available then a thin-walled rectangular section that matches these properties is synthesised. This should be approximately correct, but obviously it cannot represent I beams or rectangular sections with varying wall thicknesses, but it should give a reasonable representation of beam dimensions. If you use inconsistent or impossible properties you may get some strange looking sections!



Use OPTT is now available with **True Sections** and will plot the beam as circular using the value of OPTT on *PART_CONTACT for diameter rather than the physical section

Shells

PRIMER normally draws thin shells as lines in wireframe mode, or as zero-thickness facets in shaded and contoured modes, neither of which give any indication of shell thickness.

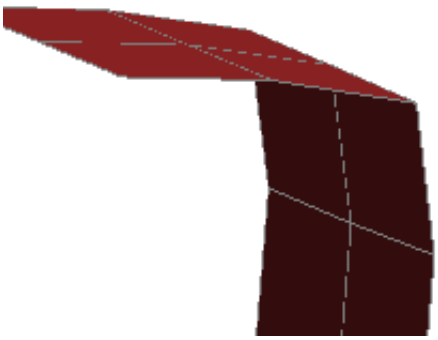


The true thickness of shells can be displayed in two ways as shown below:

Thin draws a shell as a single infinitely thin facet.

This is fast to draw and minimises graphics memory usage, so it is the default mode.

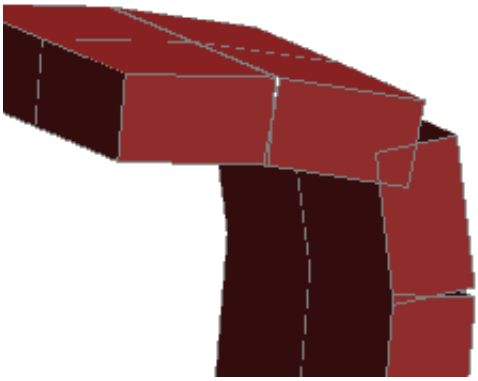
The two images below show exactly the same elements drawn using the two "thick" true section variants.



True flat "extrudes" each element individually, making no attempt to form a continuous surface between adjacent shells sharing a common edge.

This can be useful when trying to determine exactly what the thickness of each shell element is, as shown in this example.

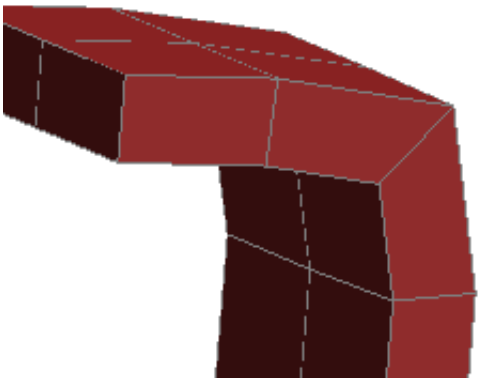
However it doesn't look very nice!

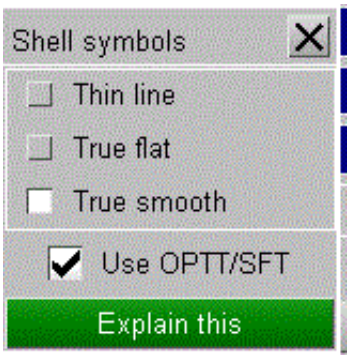


True smooth extrudes as above, but attempts to join up shells at common edges to form a continuous surface.

This image shows the same elements as above, but demonstrates how edges are now joined up. It looks much better, but it is not so easy to determine the exact shape and extent of each shell.

This mode is better for presentation purposes.





Use OPTT/SFT In True flat/True Smooth this option will derive the thickness from OPTT or SFT on the PART_CONTACT card rather than the physical shell thickness.

The "true" shell shapes shown above take into account any variation in element thickness (eg fields **`T1`** to **`T4`** on **`*SECTION_SHELL`**), and also any offsets of the neutral axis from the plane of the nodes as defined by **NLOC** (eg on **`*PART_COMPOSITE`**) or by **`*ELEMENT_SHELL_OFFSET`**.

This is not the case in "thin" mode, which is always drawn on the shell's nodal plane, ignoring any offsets.

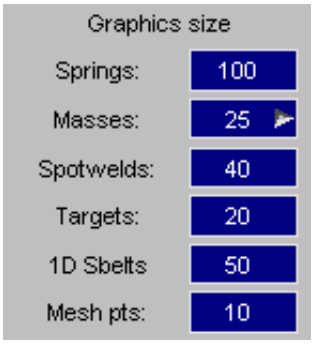
In addition if the shell is a composite then in "true" mode lines will be drawn on the sides showing the individual layer thicknesses.

The default shell display mode can be changed using the preference:

`primer*shell_graphics_mode: thin` *or* **`flat`** *or* **`smooth`**

Window Dressing

Controls the display of the plot border, and display of the current date.
The "**GRATICULE**" is tick marks around the edge of the plot which show the current window dimensions: useful for estimating distances on the screen (although **MEASURE** provides a more accurate method). For more information on the graticule see section 4.1.4.

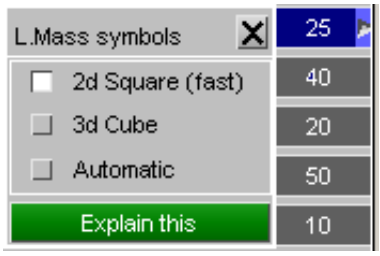


Graphics size

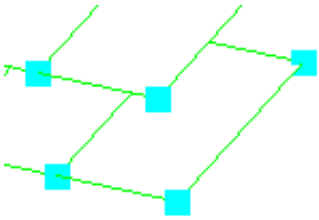
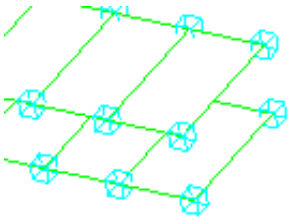
Sets the display size of certain dimensionless symbols. (Springs get a "small spiral" symbol when their size gets too small to visualise as their "normal" symbol.)

These sizes can be controlled via the preferences:

`primer*lumped_mass_size:` *<size>*
`primer*spring_size:` *<size>*
etc



Lumped mass symbols can be displayed in two different ways, controlled via the popup on the **`Masses`** size button.

2d square	2d square symbol. Not very attractive, but very fast to draw	
3d cube	3d cube symbol. Attractive and intuitive, but much slower to draw. Models with masses at many nodes become unacceptably slow to respond to dynamic viewing commands if this display method is used.	
Automatic	Uses "3d cube" symbols if there are < 10,000 mass elements in the model, otherwise "2d square". This is the default since it gives a reasonable trade-off between image quality and rendering speed.	

The default lumped mass symbol display mode can be controlled by the preference

`primer*lumped_mass_symbol: square | cube | automatic`

SH/CT/SI Overlay The element border overlay for **SH**(aded) plots, and also the contoured **CT** (continuous tone) and **SI** (shaded image) is separately controllable.

Colour	Is one of the standard PRIMER colours selected from the popup menu
Overlay edging mode	Is one of **No overlay**, **Free edges**, **Feature lines** or **All edges**. **Free edges** are defined where an element edge is not connected to any other element of the same type, or where the part ids of the elements at an edge differ. Therefore a topological plot of the boundaries of mesh zones is produced. **Feature lines** are a superset of "free edge" mode in which the angle between adjacent elements is considered. Where this angle is greater than the "Edge angle" defined below then a feature line edge is defined, and this is added (logically ORed) to the free edges. The effect is to give a better idea of the shape of the mesh than is available from free edges alone.
Edge angle	Sets the angle (in degrees) between adjacent element faces at which a "feature line" edge will appear.

Swap nodal coords allows you to swap the nodal coordinates used throughout PRIMER with:

Standard	Reverts to using the normal coordinates defined under the **`*NODE`** card.
Airbag ref geom	The coordinates of nodes defined under **`*AIRBAG_REFERENCE_GEOMETRY`**
Foam ref geom	The coordinates of nodes defined under **`*INITIAL_FOAM_REFERENCE_GEOMETRY`**

This is a straight swap: the values of the nodal coordinates used for all internal PRIMER operations are swapped over, and there are no interlocks or warnings to prevent you misusing this.

If you use this option it is your responsibility to manage it in the appropriate context(s) and to unset it when finished.

However, note that if this option is set when reading a model, PRIMER will automatically unset it first before reading the model to ensure that node data does not get corrupted.

Swap shell topology allows you to swap the shell topologies used throughout PRIMER with:

Standard	Reverts to using the shell topology defined under the **`*SHELL`** card.
Airbag ref geom	The topology of shells defined under **`*AIRBAG_SHELL_REFERENCE_GEOMETRY`**

This is a straight swap: the values of the shell topologies used for all internal PRIMER operations are swapped over, and there are no interlocks or warnings to prevent you misusing this.

If you use this option it is your responsibility to manage it in the appropriate context(s) and to unset it when finished.

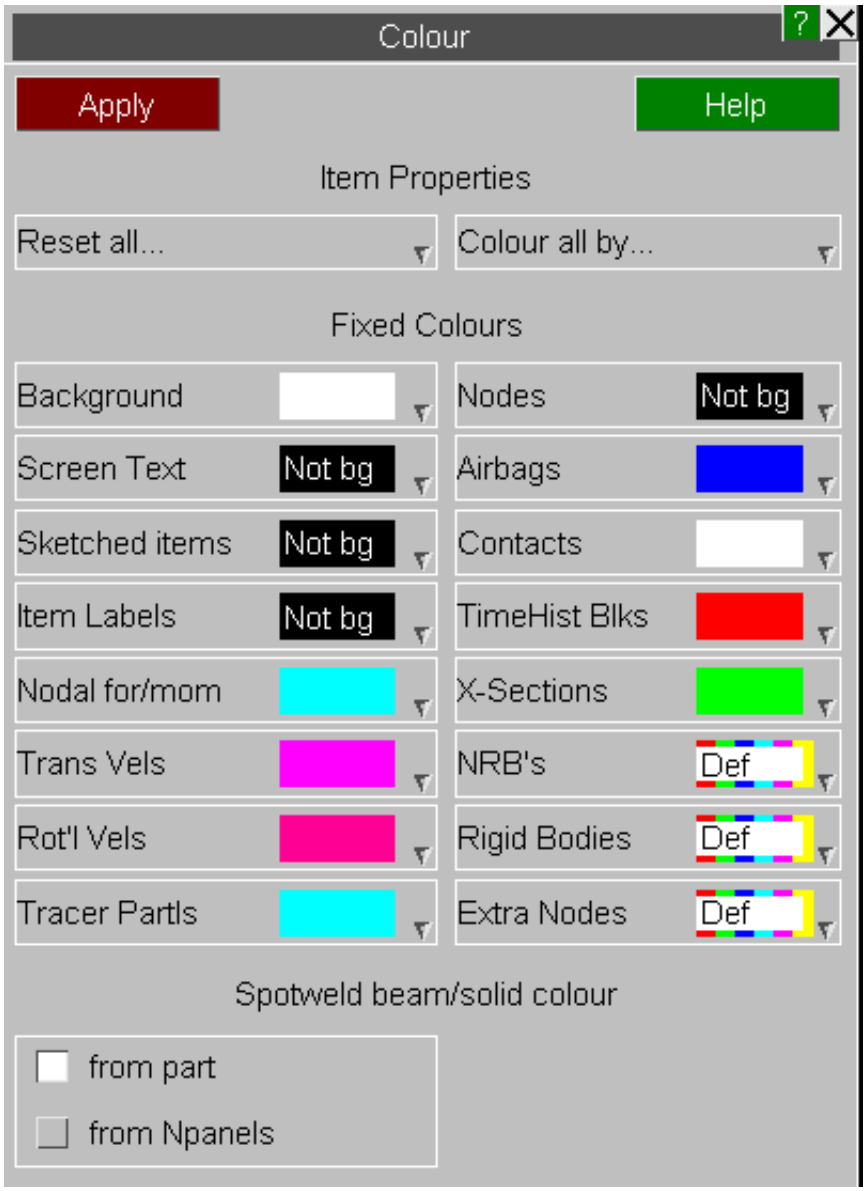
However, note that if this option is set when reading a model, PRIMER will automatically unset it first before reading the model to ensure that shell data does not get corrupted.

4.1.2 COLOUR... Setting item colours in plots.

All options in this panel have popup menus giving a range of colours, with the current selection being shown.

The special colour **Not bg** means "not the background". This is a colour guaranteed to show up well against the current graphics window background, and is the default for text, labels and sketched items.

This colour will change automatically as required if the background colour changes.



Background Sets the background colour of the graphics window. Default: black. This can be configured in the oa_pref file using:

```
primer*background_colour: <colour>
```

Screen Text Sets the colour for title, date, contour bar values, etc. Default: **Not background**. This can be configured in the oa_pref file using:

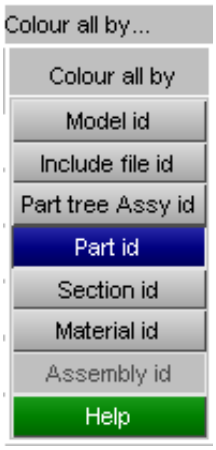
```
primer*text_colour: <colour>
```

Sketched Items Sets the colour for anything sketched in any context. By default it is set to "**Not background**", the logical opposite of the current background setting so as to establish good contrast. It can also to be set to a fixed colour or **Use Text Colour** which will use the same setting as **Screen Text** is set to.

Item Labels Sets the colour for item (eg node) labels. Default: white.

Colours can also be set for several other types of item in PRIMER using the other popups.

Colour all by gives options of how to colour model items.



- MODEL id sets colour by model number.
- INCLUDE file id sets colour by which include file items belong to.
- PART TREE ASSEMBLY bases colour on part tree assembly ids
- PART id sets part-based element colours by part number
- SECTION id sets part-based element colours by section
- MATERIAL id sets part-based element colours by material.
- ASSEMBLY id uses colours based on the selected mechanism or dummy assembly ids.

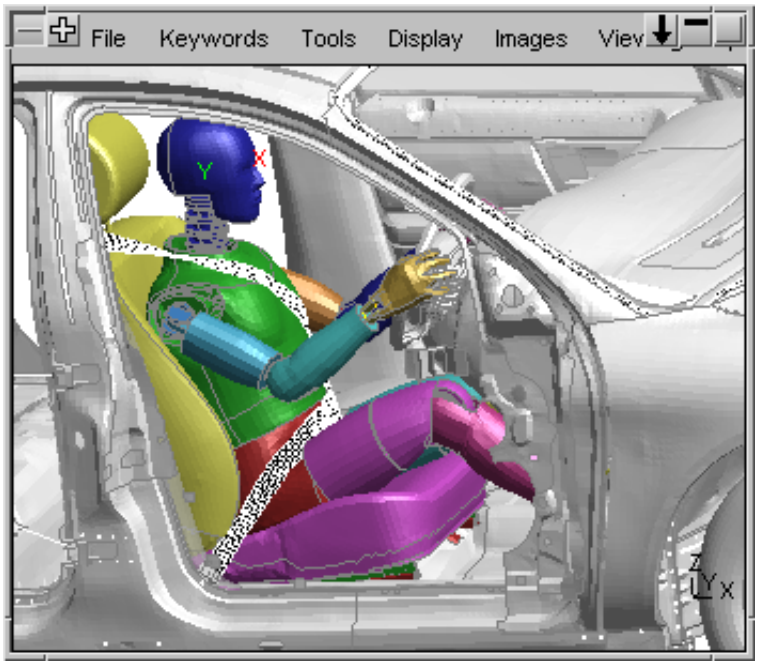
Using colour based on PART, SECTION or MATERIAL id:

For element types that use parts (solid, shell, beam, tk shell, discrete, seat-belt, sph) the colour may be based on one of these properties. The label of the property is used, for example all elements of part 1 will be the same colour. Where a property is undefined, for example no material defined on a *PART card, grey is used.

The default is for all such elements to be drawn by **PART** colour.

Setting user-defined colours for individual parts or groups of parts can be achieved using Quick Pick, Part Table or Part Tree. User -defined colours may also be defined for materials, elements and some other entity types using Quick Pick.

Using colour based on Mechanism or Dummy Assembly id:



This is a special case in which the subset of parts in the model which belong to the selected mechanism or dummy receive colours based on the assembly ids in that mechanism / dummy, and the rest of the model is coloured white. This is the default behaviour when positioning mechanisms, but when used in this context it will persist during normal operations. In the example here the seat and dummy form a mechanism.

There are some limitations to this display mode. It is computed by assigning an assembly id to each part to give a colour, with parts not in an assembly (or in a different model) receiving an id of zero, which is rendered as white. This assembly id is stored in a special internal storage "slot" on the part, and the contents of this slot can get changed in the following ways:

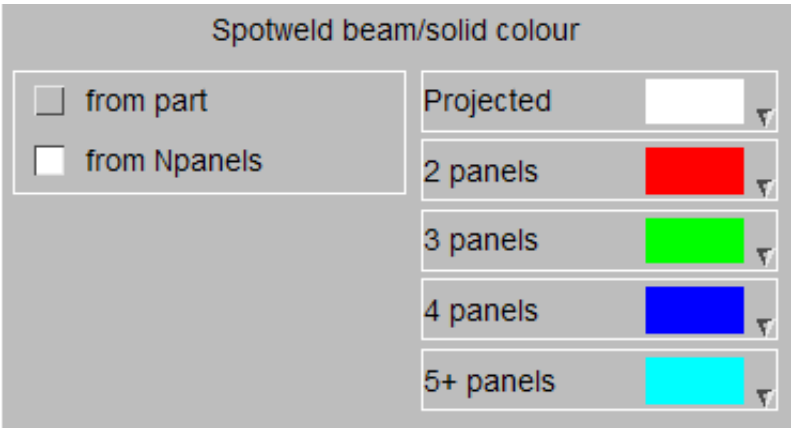
- Using the dummy or mechanism positioner updates these storage slots, so if this display method is in force and you start positioning a different dummy / mechanism then the colours in a given model will change to those of what is being positioned.

- Editing or created a dummy / mechanism also updates these storage slots, so the colours will change in a given model to represent the most recently edited definition.

- Checking (eg by model check) a dummy / mechanism also updates these storage slots, so that too may change them to show the most recently checked definition.

In addition checking and positioning a dummy does not consider null parts when assigning assembly ids to parts, so if an assembly is coated with null parts these may end up being drawn in white. If this happens the solution is to return to this panel and to redefine the dummy / mechanism which is to be used, which will restore its colours - including those for null parts.

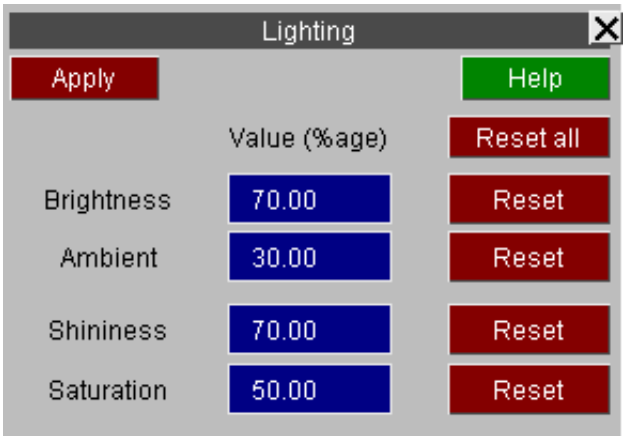
Spotweld beam/solid colour allows you to change the colours used when drawing spotweld beams or solids. The default is **from part** in which case the normal colour for the element is used. If **from Npanels** is chosen then the colour of the element will change depending on how many panels the spotweld connects. Popups allow you to change the colours as necessary.



4.1.3 Lighting

The light source position in PRIMER is fixed at approximately the viewer's right shoulder.

Only the attributes shown here can be changed.



Brightness There is a light source located approximately at the observer's right shoulder (this cannot be altered). This field controls how bright this source is. Facets normal to the observer reflect the maximum amount of light from this.

Ambient This control the level of "black-body radiation" which illuminates all facets equally regardless of orientation.

Shininess Low values will give a matt (dull) appearance, high values a shiny one.

Saturation This controls the 'depth' of colour in the range pure colour to grey.

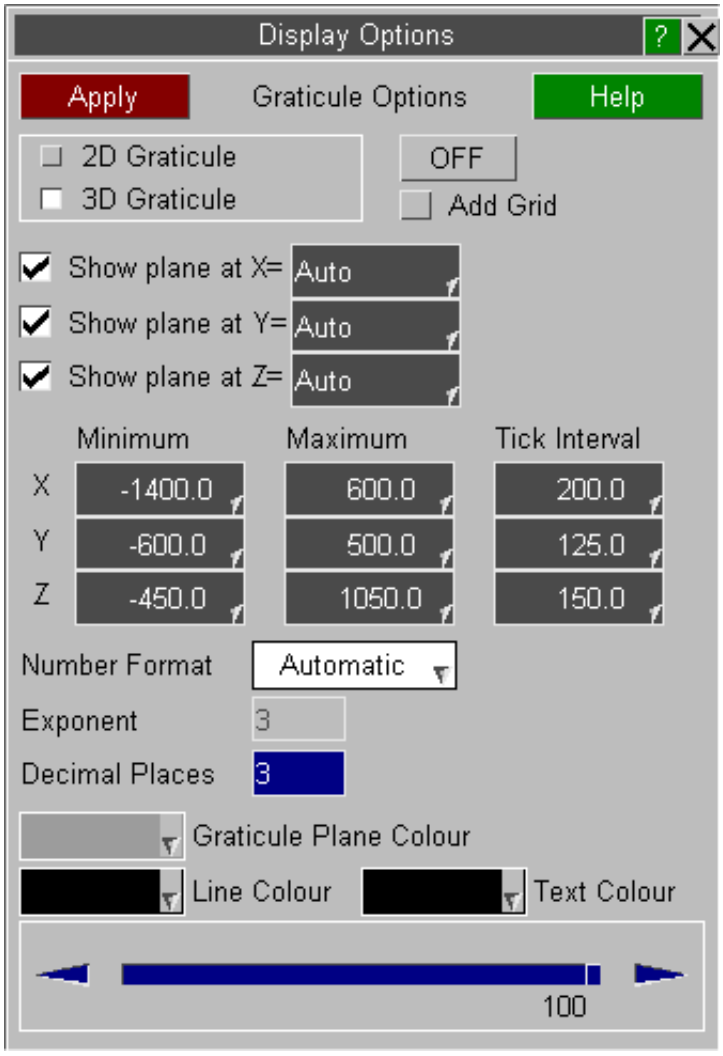
4.1.4 Graticule

This can be used to display the current model dimensions.

The graticule can be drawn in either 2D or 3D.

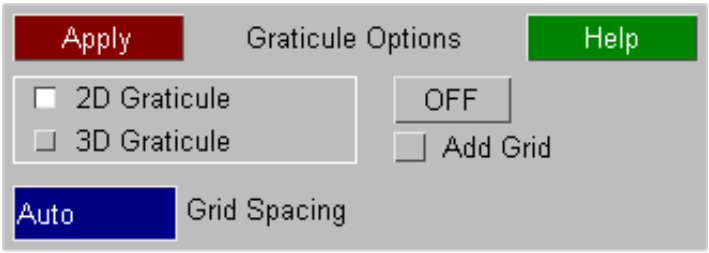
The format of the numbers on the graticule can be set automatically by PRIMER or you can manually select the number of decimal places and the exponent value to display.

The line and text colours can be modified if necessary.



4.1.4.1 2D Graticule

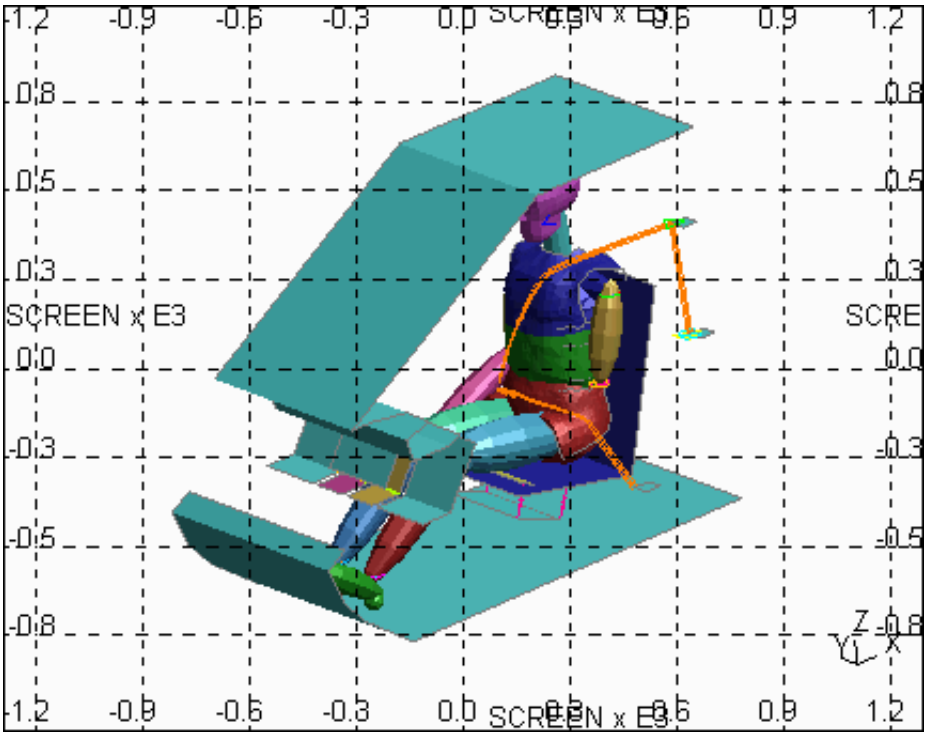
If the grid spacing can is set to 'Auto' PRIMER will calculate a sensible value. If you want you can input a value manually.



With the 2D graticule the space system used to display the model dimension depends on the current view.

Model space Is used if the view is orthogonal down one of the screen X, Y or Z axes. The appropriate XY, YZ or XZ coordinates are shown, and these move as the model moves (try dynamic translation and you'll see).

Screen space Is used if the view is not orthogonal. This just shows the current window dimensions (X = 0 - 4095, Y = 0 - 3129). It is only useful for setting up volume clipping using screen space orientation.



If a **GRID** is added it draws a grid on the screen at the current tick mark interval.

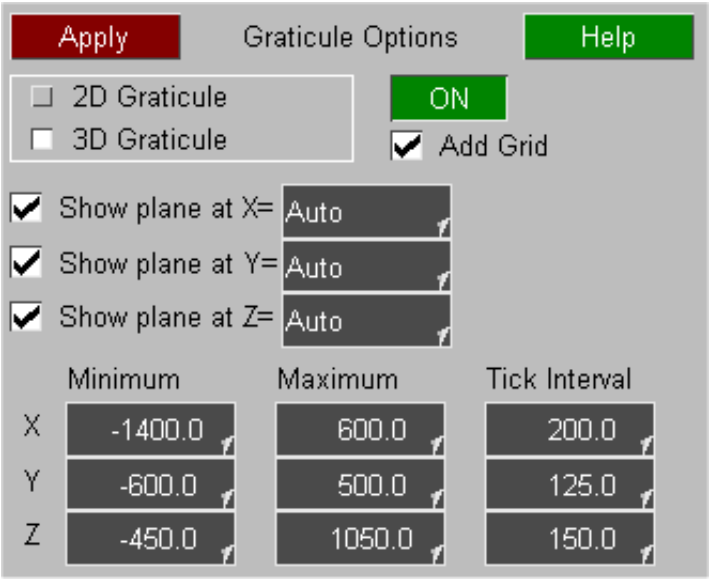
4.1.4.2 3D Graticule

The 3D graticule option will produce 3 planes aligned with the global x, y and z axis which show the model bounding box.

The display of each of the 3 plane can be turned on and off separately as required.

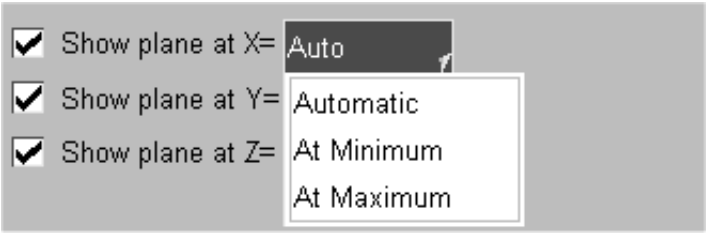
As well as specifying the minimum and maximum dimensions for each plane the location of each plane can also be specified along with the grid interval.

By default PRIMER will automatically calculate all the graticule plane values. If the user modifes any of the values then the text box colours will change to WHITE text on a DARK BLUE.



By default PRIMER will automtically calculate the location of the 3 graticule planes based on the model dimension. The location of each plane can be changed by entering the new location in the text box.

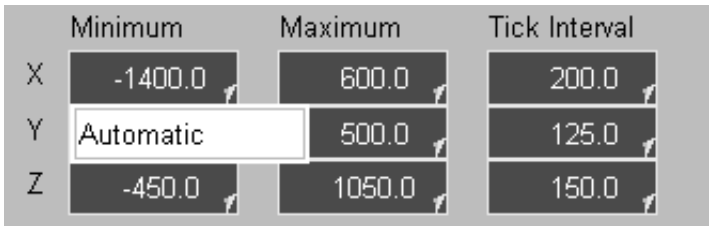
Alternatively 3 pre-set locations can be selected.



Automatic This is the default option. PRIMER will automatically locate the plane at either the minium or maximum value so that it is positioned behind the model from the users view point. As the model is rotated PRIMER will adjust the plane location as required.

At Minimum The plane will automatically be located at the minimum value for the axis. If the axis minimum is modfied by the user the plane location will automatically update.

At Maximum The plane will automatically be located at the maximum value for the axis. If the axis maximum is modfied by the user the plane location will automatically update.

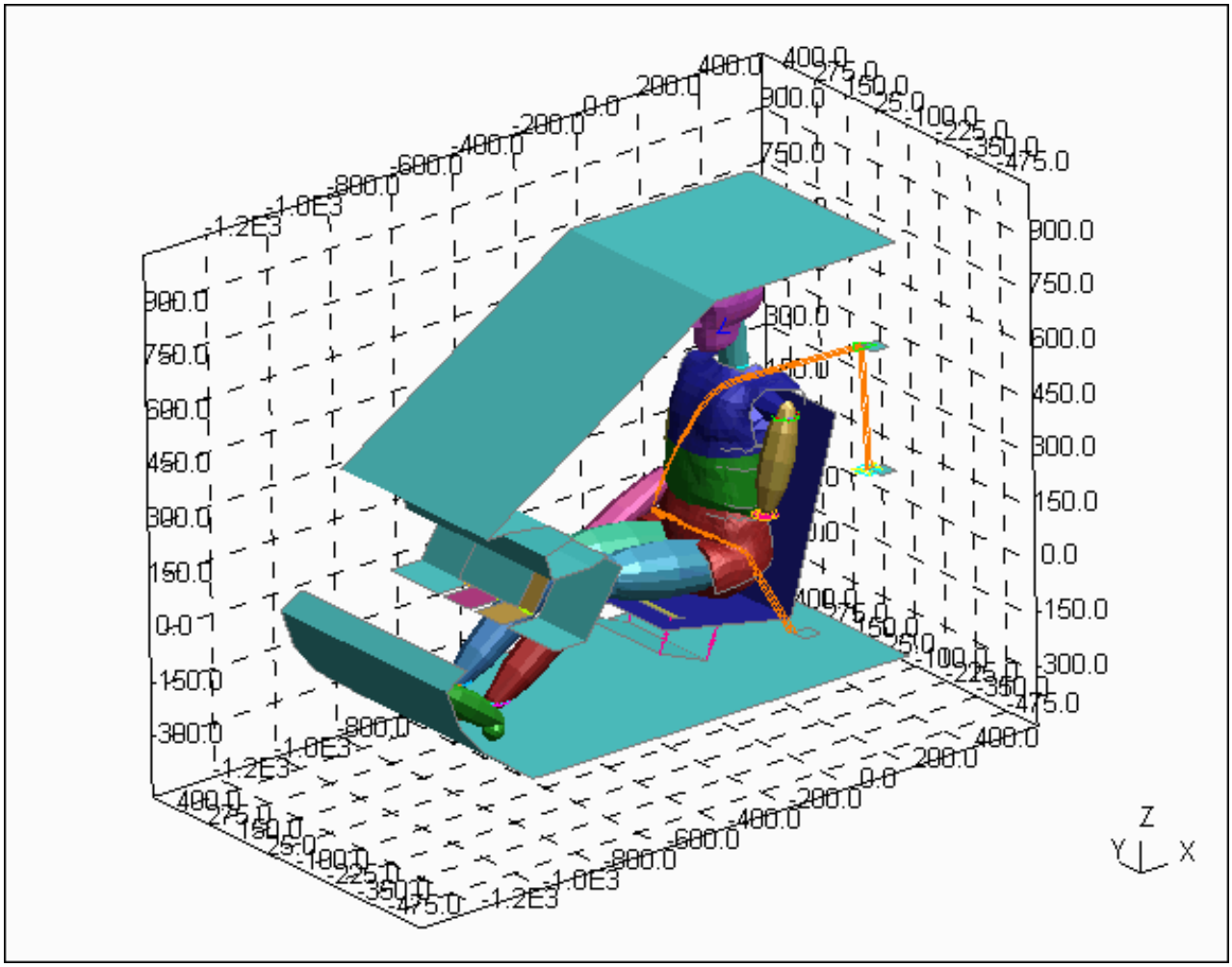


By default PRMER will automtically calculate the minimum and maximum values used to display each plane along with the interval between the values displayed.

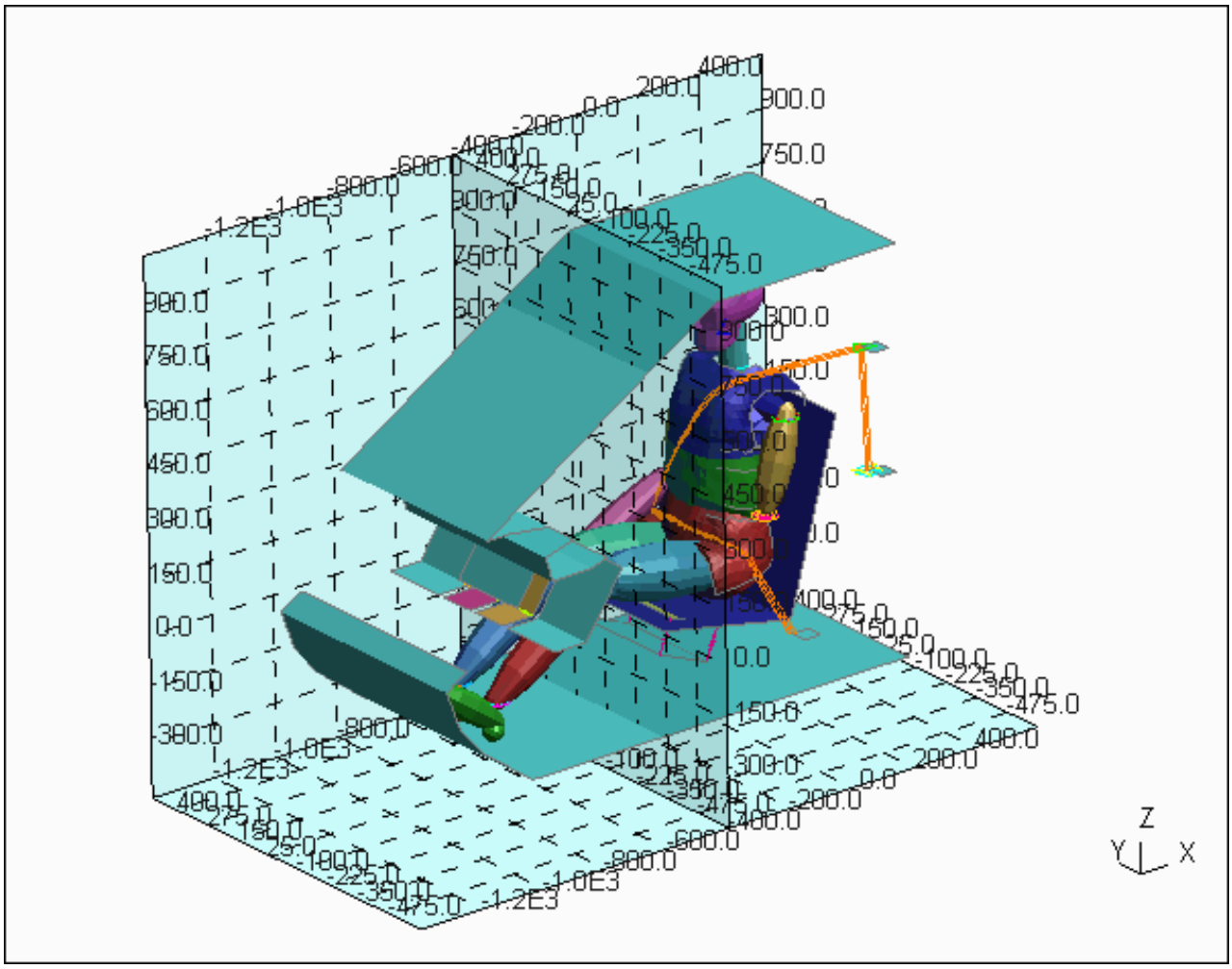
The minimum and maximum values along with the tick intreval can be changed using the text boxes. If any of the values are changed then the text box colours will change to WHITE text on a DARK BLUE.

All of the values can be reset to **Automatic** using the popup menu.

If the Tick Interval is set to **Automatic** D3PLOT will adjust the tick spacing if required as you zoom in and out.



If necessary a transparency value and colour can also be set for the 3 plane



4.1.5 Graphics setup via the "oa_pref" file.

The following parameters can be preset via the preferences file, either by manual editing or using the Preferences Editor. Syntax being:

```
primer*<keyword>: <argument>
```

for example:

```
primer*initial_plot_model: SHAD
```

Keyword	Possible arguments	Default value
plot_border	**ON** or **OFF**	**ON**
overlay_mode	**OFF** or **FREE** or **ALL**	**ALL**
overlay_colour	**WHITE, GREY, BLACK, RED, GREEN, BLUE, CYAN, MAGENTA, YELLOW, ELEMENT**	**WHITE**
background_colour	**WHITE, GREY, BLACK, RED, GREEN, BLUE, CYAN, MAGENTA, YELLOW**	**BLACK**
initial_plot_mode	**LINE** or **HIDDen** or **SHADed**	**LINE**
contour_levels	**1** to **13**	**6**

Further "oa_pref" file options, and details of the interactive preferences editor, are given in Appendix XIII.

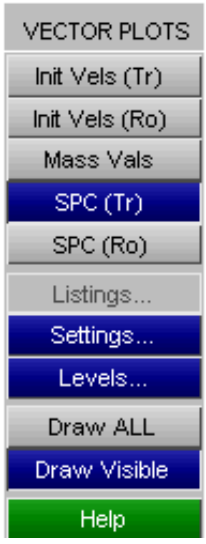
4.2 Data Plotting Commands:

VEC(tor)
CT (continuous tone)
SI (shaded image)

Each command has a popup menu that gives some or all of the following options:

Data component:	Timestep, Shell thickness, etc.
Listings...	Written output of displayed data. For example lists of element timesteps sorted into ascending order.
Settings...	Unique panels for each data component that control what is drawn and how it is displayed.
Levels...	Control and display of the number of contour levels.

4.2.1 Vector plots.

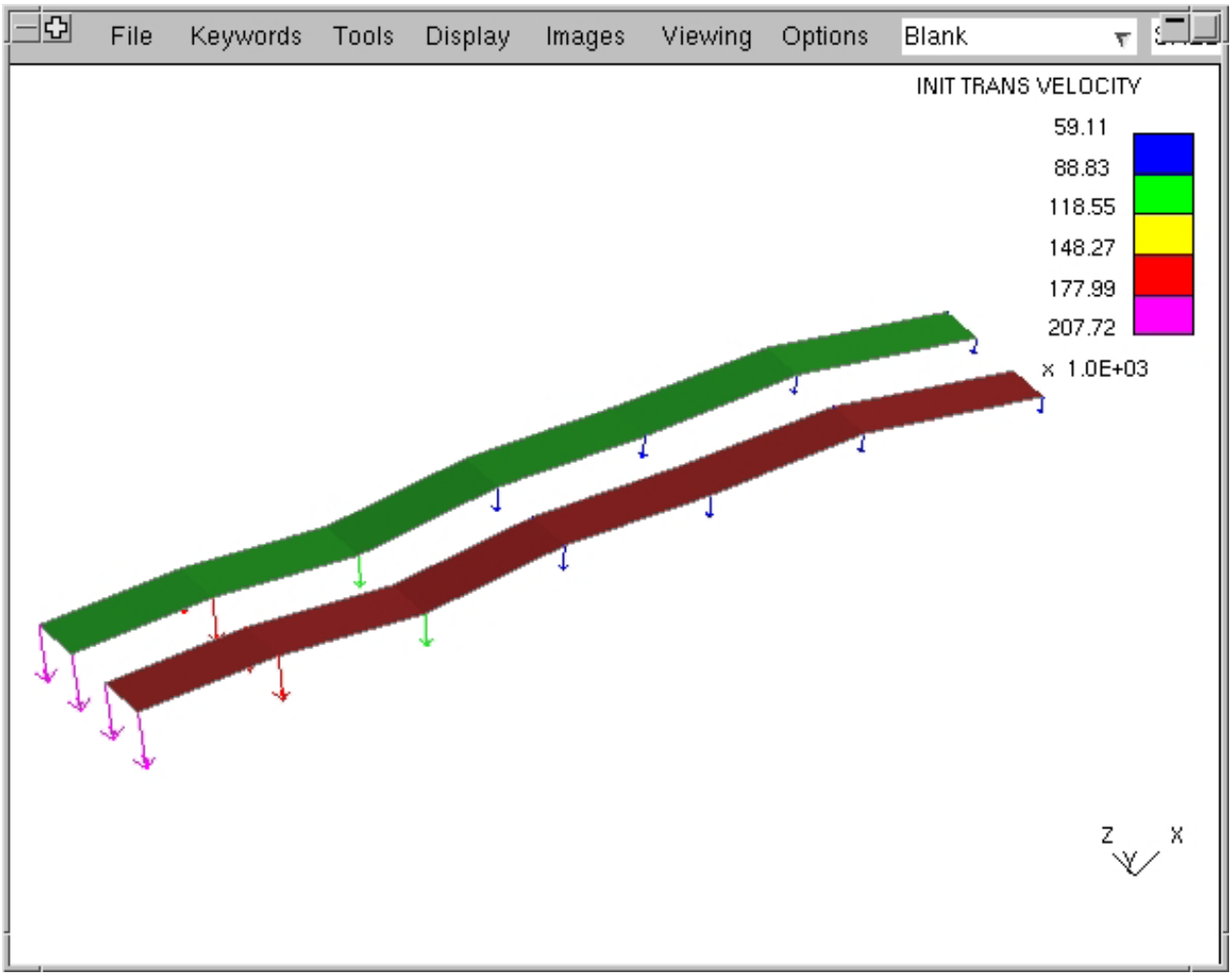


Vector plots superimpose nodal data on the current image display mode (**LI, HI** or **SH**). To display these data on a different mode draw it first, then repeat the **Vect plot** command.

At present vector plotting is only available for:

Init Vels (Tr)	Vectors of initial translational velocity
Init Vels (Ro)	Vectors of initial rotational velocity
Mass Vals	Lumped masses plotted by mass
SPC (Tr)	Translational SPC's are plottred for Nodes, Rigid Parts and Nodal Rigid Bodies
SPC (Ro)	Rotational SPC's are plottred for Nodes, Rigid Parts and Nodal Rigid Bodies

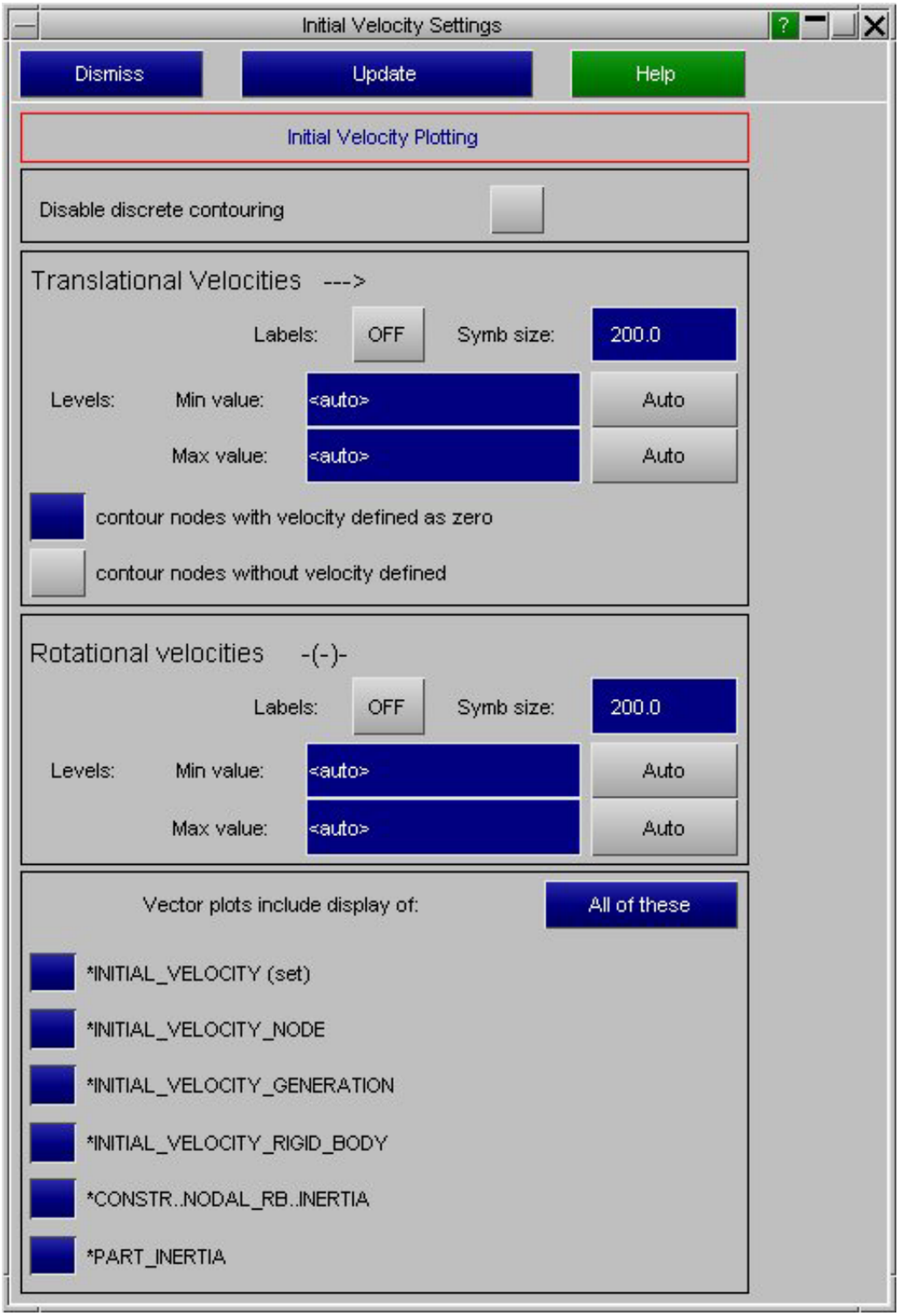
Vector plots of Initial Velocities.



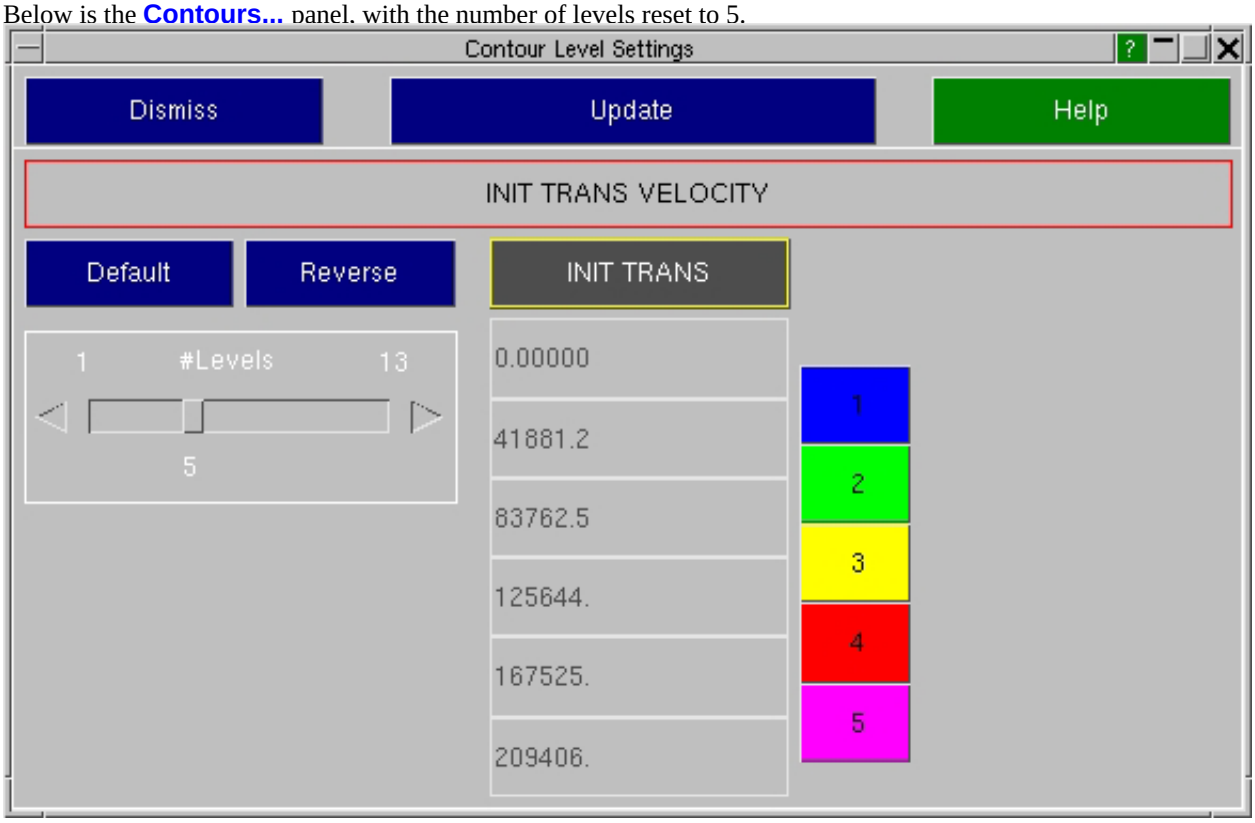
The figure above shows a typical plot of initial velocities for a simple structure. To the right is the **Settings...** panel for this plotting mode.

Note that:

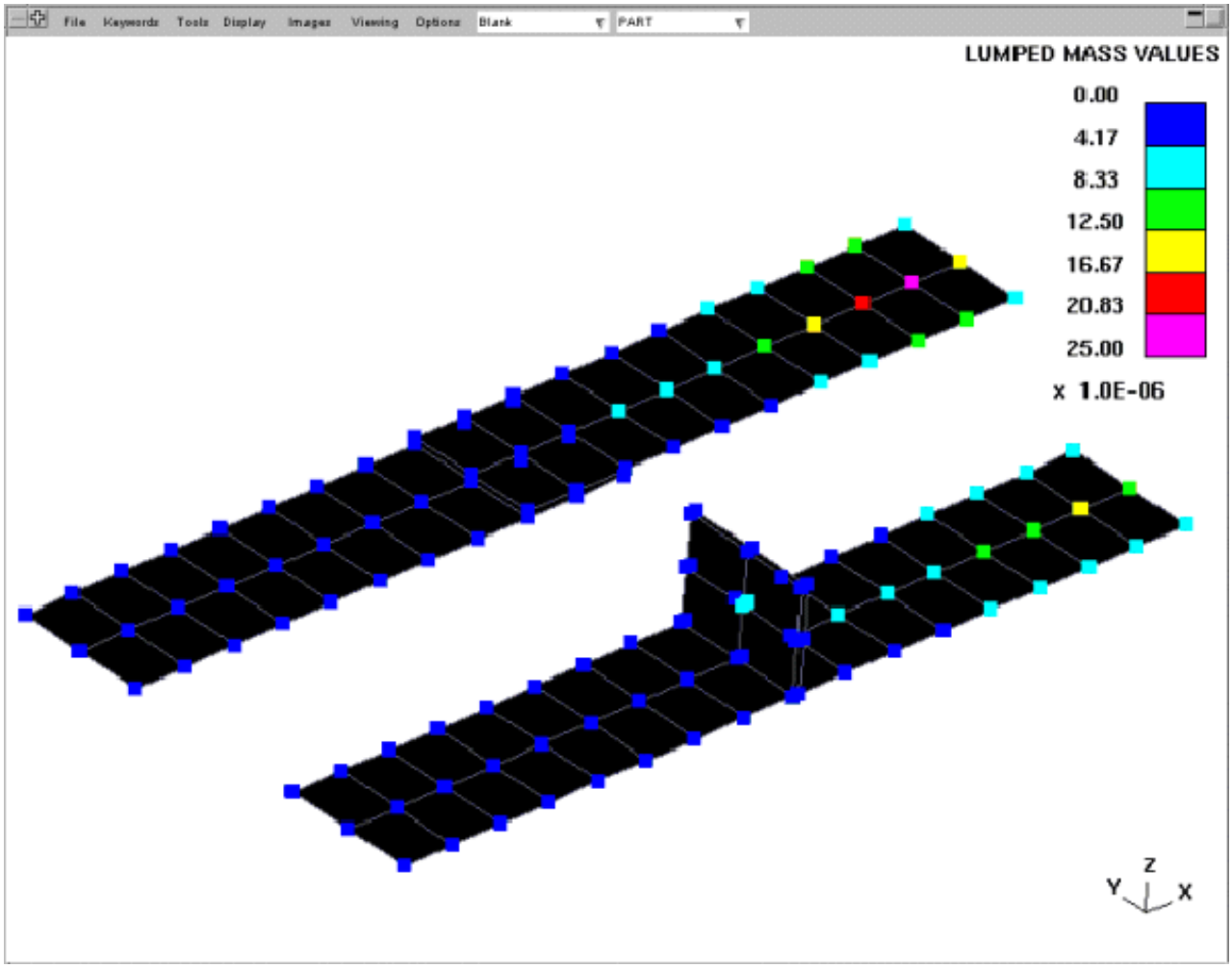
- Initial velocities in LS-DYNA can arise from five different definition methods. Display of each of these is independently switchable.
- Both translational and rotational initial velocities can be plotted, but as separate plots.
- The default contour bounds are automatic, but you may set any range you wish.


Initial Velocity Settings
Dismiss
Update
Help
Initial Velocity Plotting
Disable discrete contouring
Translational Velocities --->
Labels:
OFF
Symb size:
200.0
Levels:
Min value:
<auto>
Auto
Max value:
<auto>
Auto
contour nodes with velocity defined as zero
contour nodes without velocity defined
Rotational velocities -(-)-
Labels:
OFF
Symb size:
200.0
Levels:
Min value:
<auto>
Auto
Max value:
<auto>
Auto
Vector plots include display of:
All of these
*INITIAL_VELOCITY (set)
*INITIAL_VELOCITY_NODE
*INITIAL_VELOCITY_GENERATION
*INITIAL_VELOCITY_RIGID_BODY
*CONSTR..NODAL_RB..INERTIA
*PART_INERTIA

Below is the **Contours...** panel, with the number of levels reset to 5.

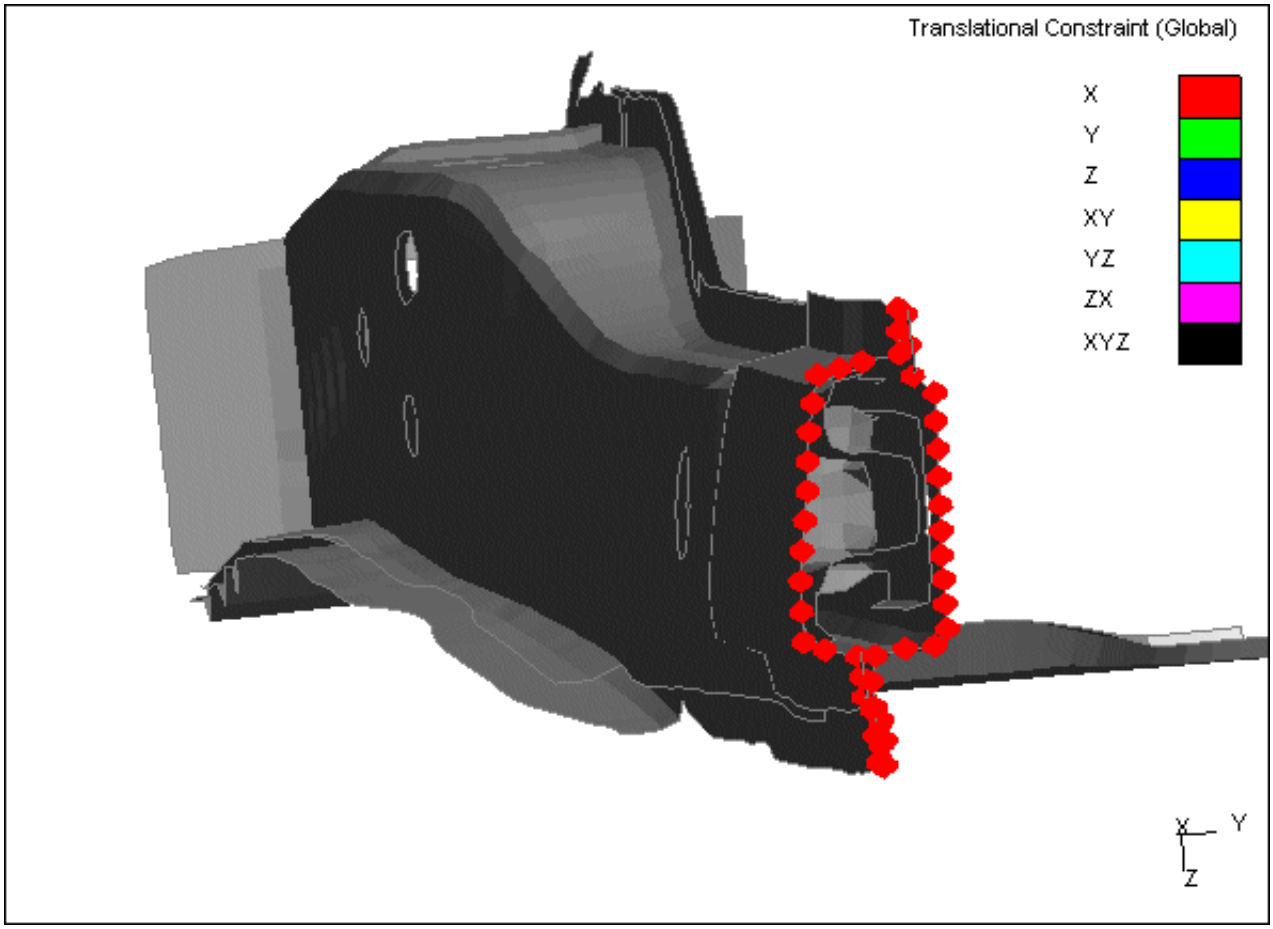


Vector plots of Lumped Mass values



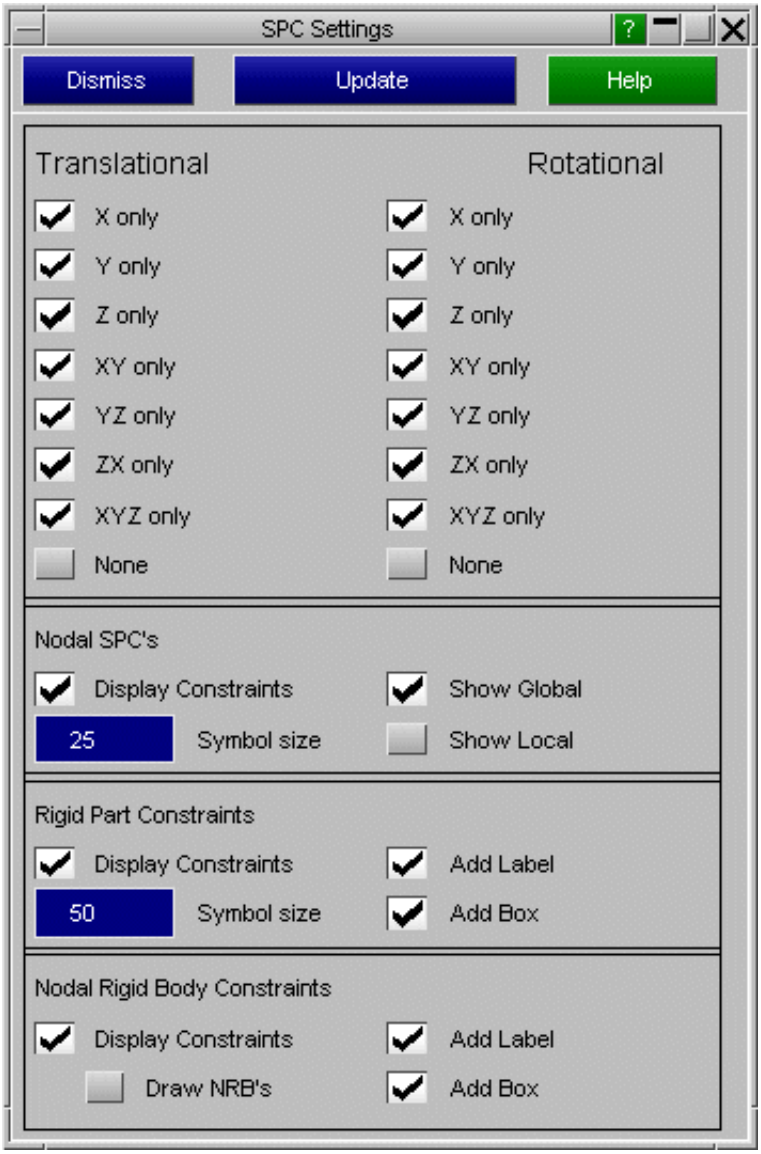
This figure shows a "Vector" plot of lumped mass values for the structure above. (Following an "Assign Mass" operation to shift the centre of gravity in the +ve X direction, hence the concentration of mass towards the right.)

SPC Translational and Rotational plots



This figure shows a "Vector" plot of the translational SPC's on the structure.

The settings panel for SPC's can be used to control which SPC's are plotted.



4.2.2 CT and SI plots.

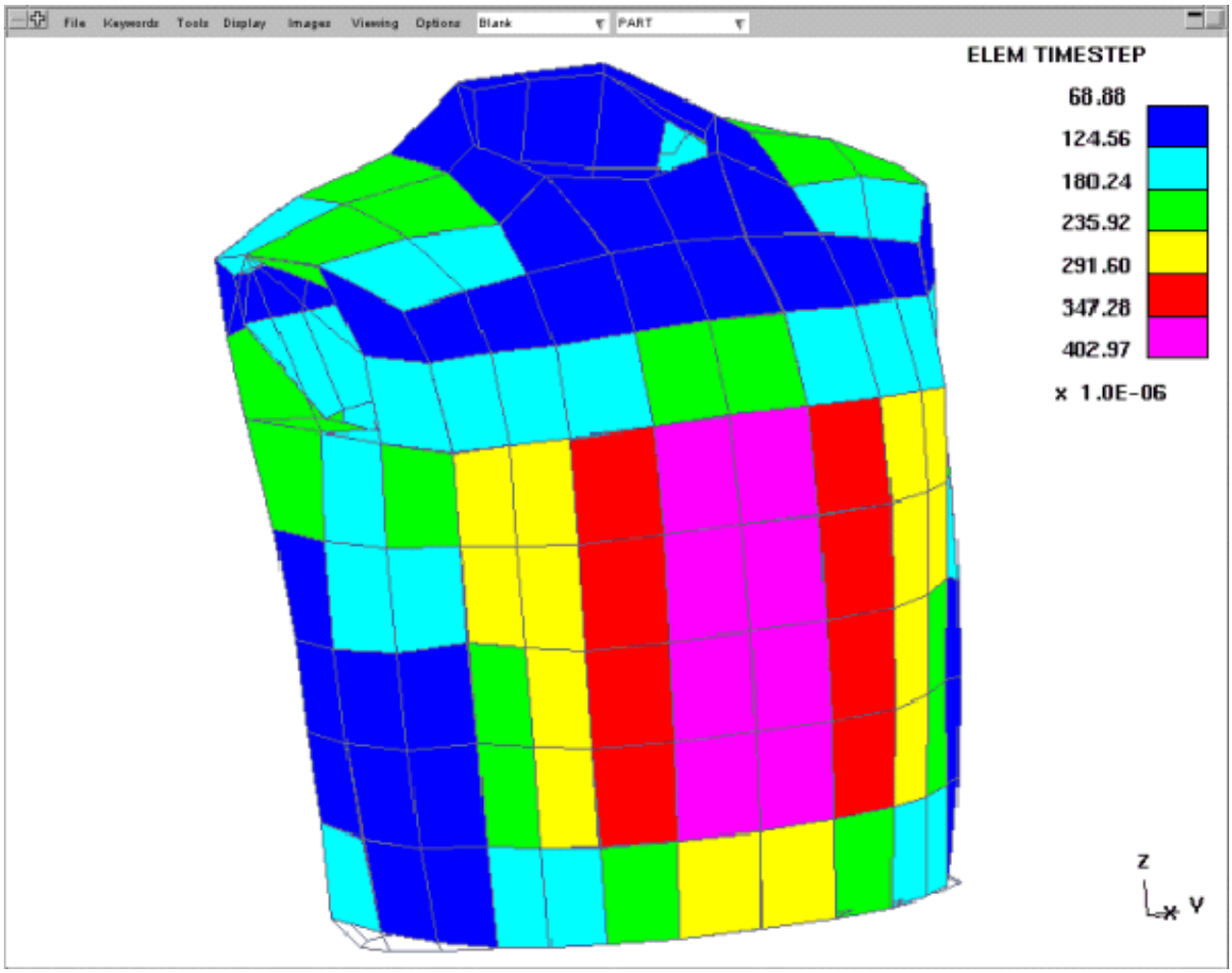
CT (continuous tone) and **SI** (shaded image) plotting modes both display the same data, but the former is unlit whereas the latter is shaded.

Both modes are used primarily to display data for 2D and 3D elements, so the underlying plotting mode is always "hidden surface with fill".

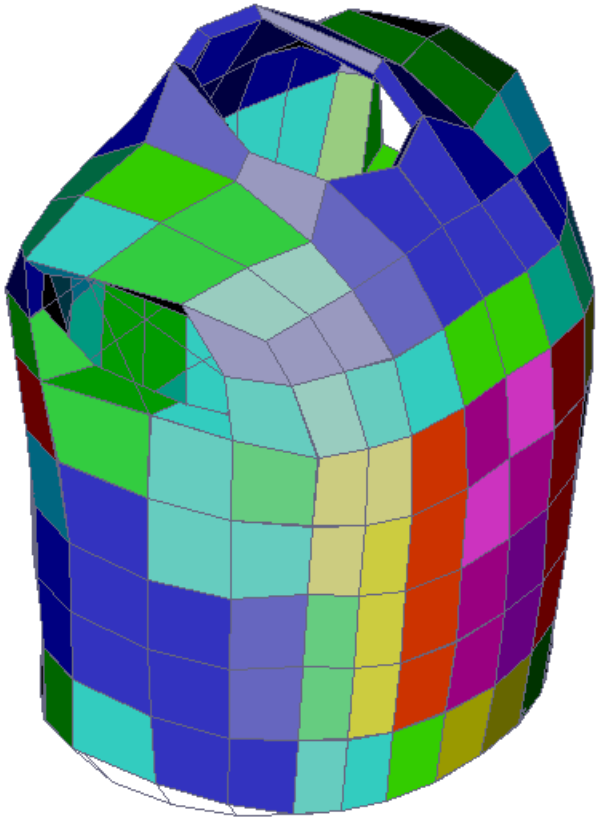
Current data components available are:

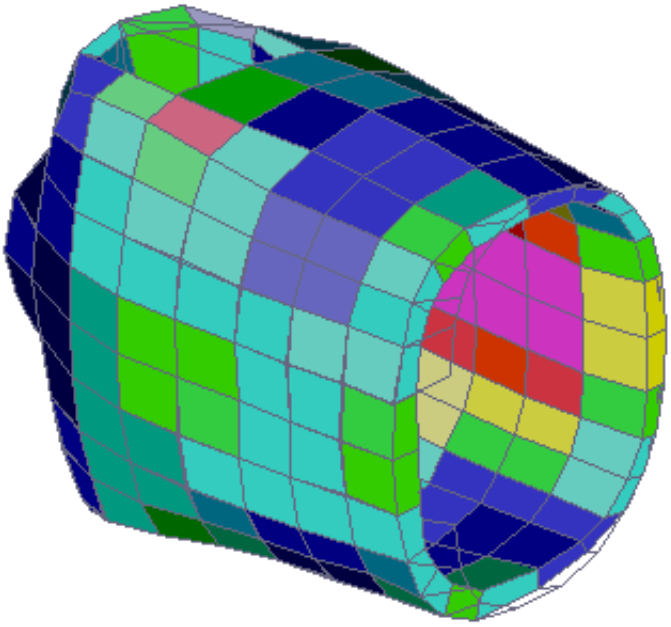
Timestep	Contours of timestep size in elements.
Shell Thk	Contours of (thin) shell thickness, and % thinning and % remaining of those shells after a forming analysis.
Load Shell Dir	Contours of Load Shell Direction (AWAY, TOWARDS, BOTH, NONE), representing the direction of the loading applied to the shell
Mass Scale	Contours of mass added during mass scaling, both by elements and by parts.
Matl Props	Contours of Density, Yield stress, Poisson's ratio & Young's modulus. Also material number (eg 20 for MAT_RIGID)
Shell Normals	Contours of Shell normals (AWAY or TOWARDS) or Arrows representing the direction of shell normals.
Elem Props	Formulation, #Int points and plastic strain.
Elem Qual	Contour of element quality.
Init Vels	Contours of initial velocity components and resultant initial velocity.

CT plot of element timesteps.

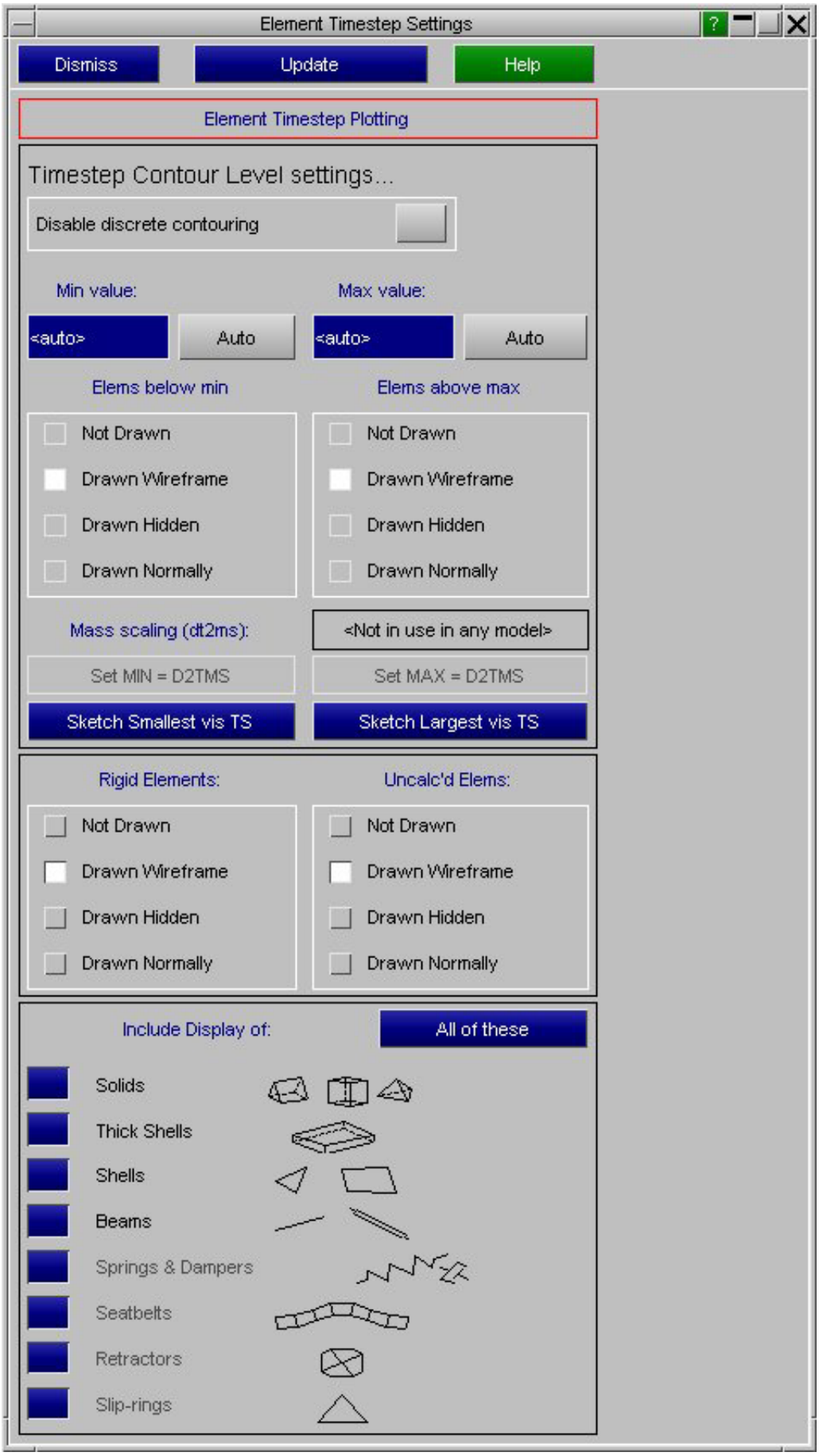


Here is the same image displayed from 2 different angles in **SI** mode, to show how lighting can be integrated with contouring to give a better idea of shape. This is a solid mesh of the torso section from a deformable crash dummy.





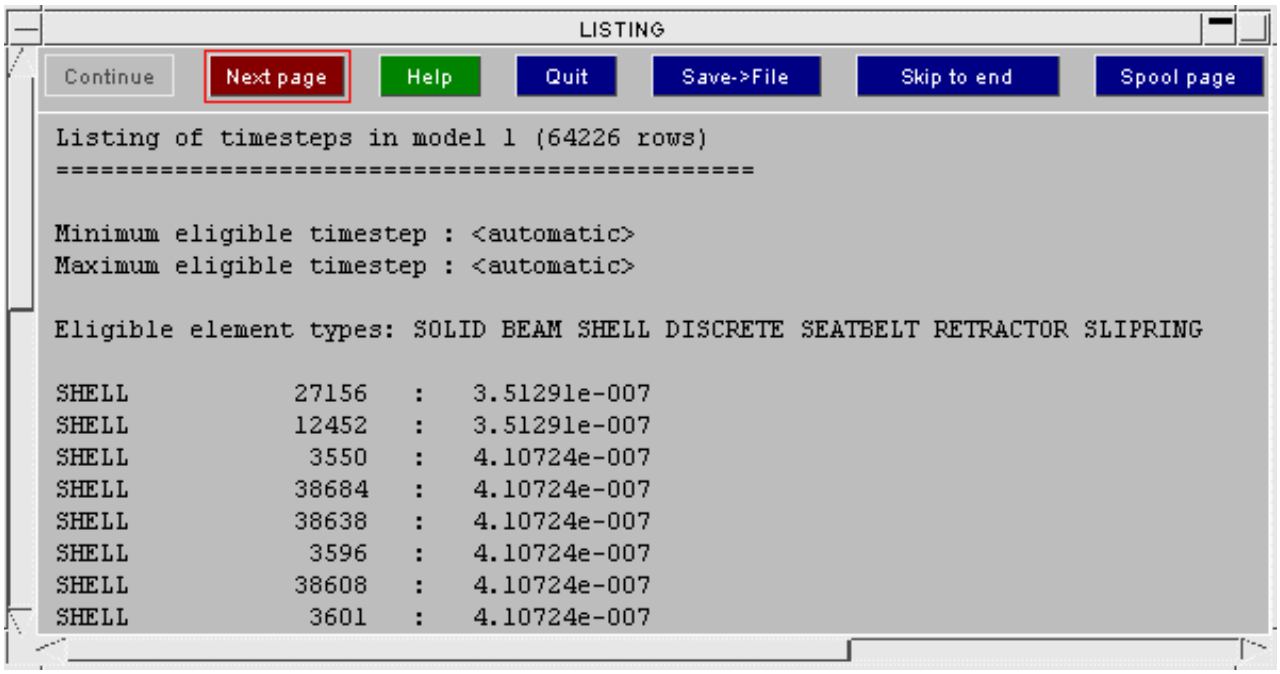
Here is the **Settings...** panel for element timestep plotting:



Note that:

- Timesteps are not computed for rigid elements, so their display is separately controllable.
- Timesteps are also not computed for elements that use more obscure material types. (Most commonly used materials are supported.) The display of uncomputed elements is also separately controllable.
- Timesteps are also currently not computed for discrete elements, or seatbelt types.

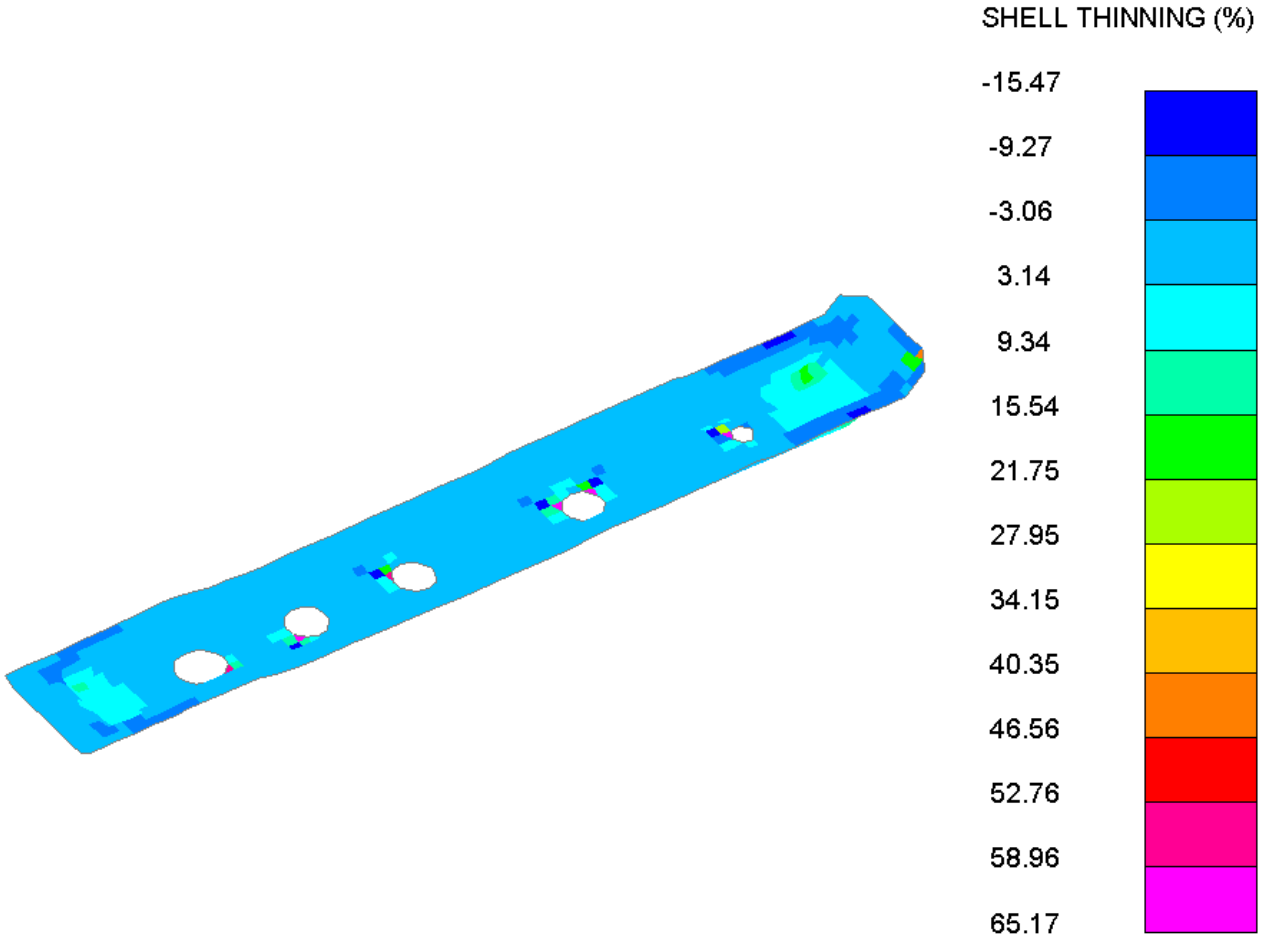
A **Listing...** of the smallest timesteps is shown below.



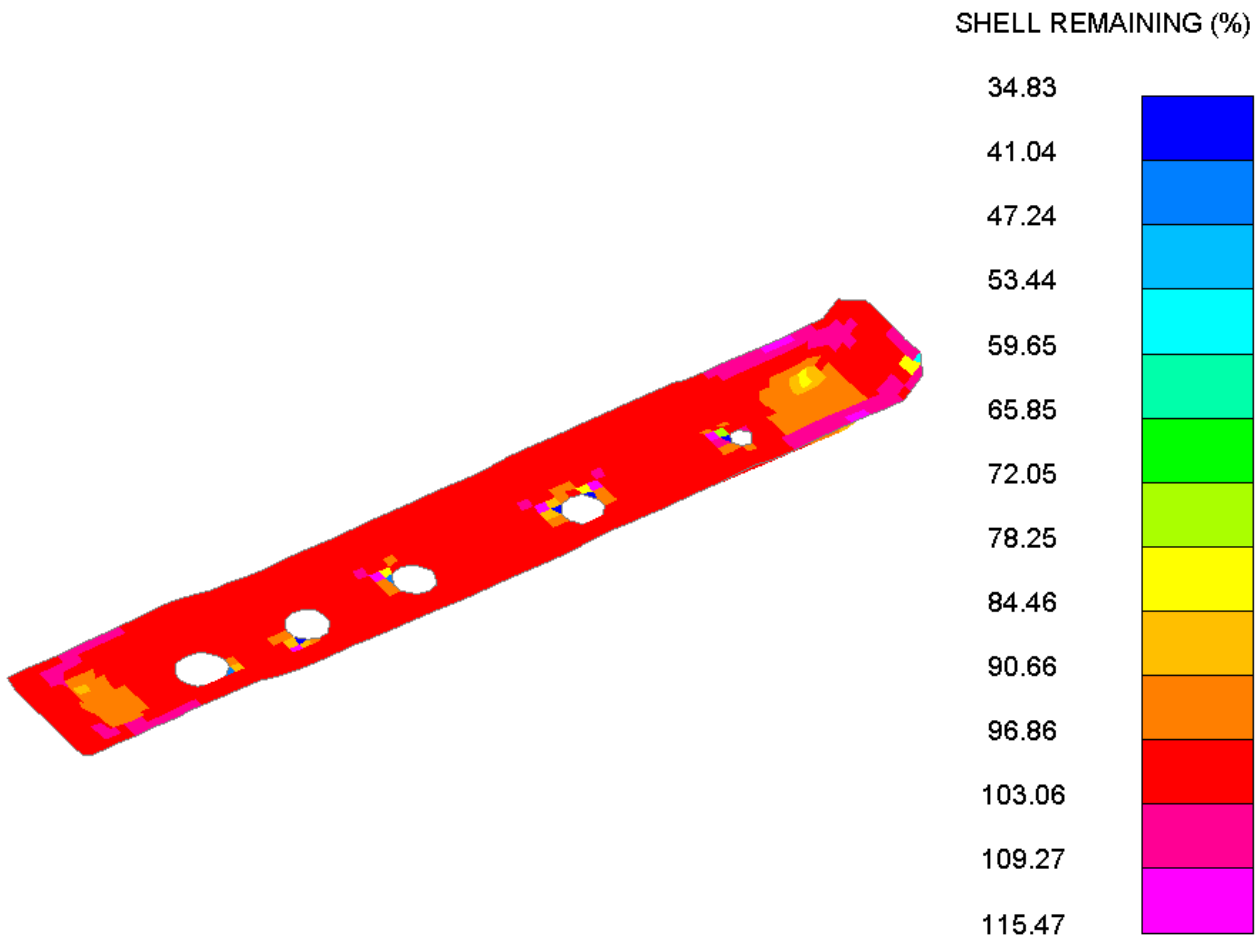
The remaining components that can be contoured in **CT** or **SI** mode are processed in a similar way, although the **Settings...** panel for each varies according to its context.

CT plot of% shell thinning.

This is the image of a plot of % shell thinning after a forming analysis.

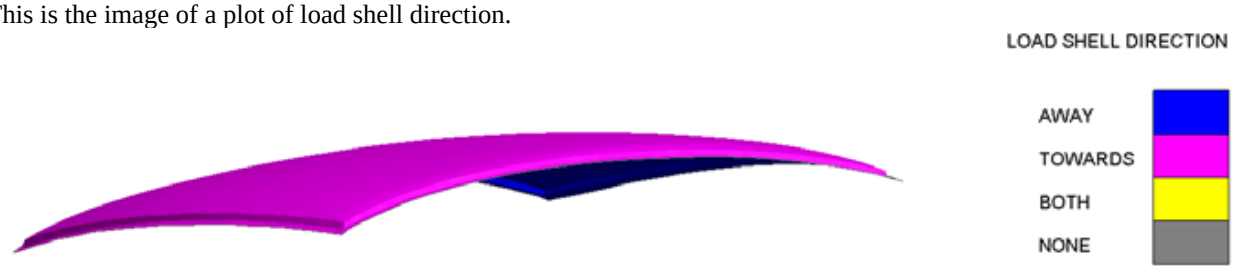


And this is a plot of % shell remaining. For every shell the sum of the % thinning and % remaining is always 100% (e.g. if the thinning of a shell is 20%, its remaining is 80%)



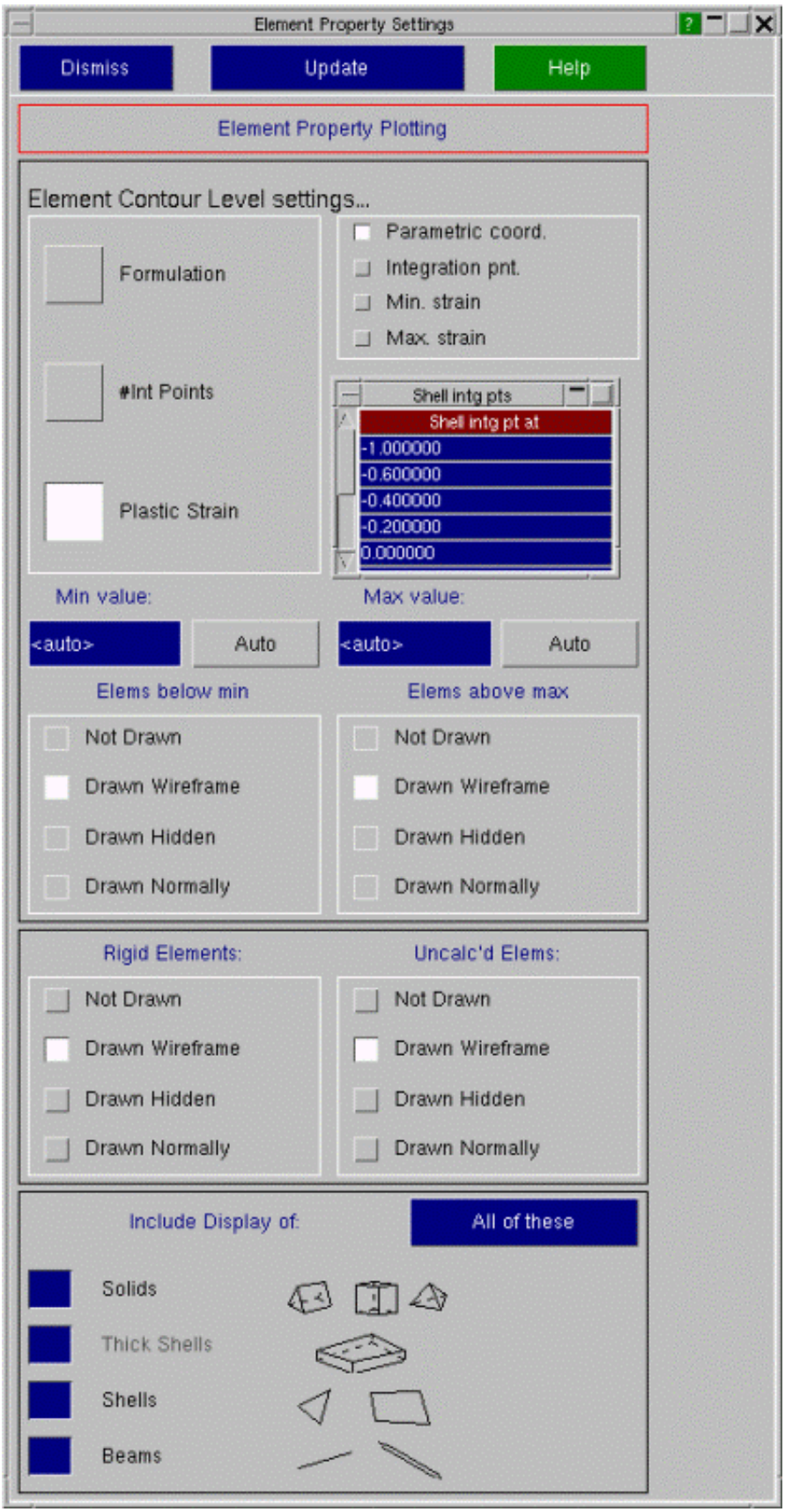
CT plot of load shell direction.

This is the image of a plot of load shell direction.



CT plot of plastic strains.

The **Settings...** panel for **Elem Props > Plastic Strain** is shown below:

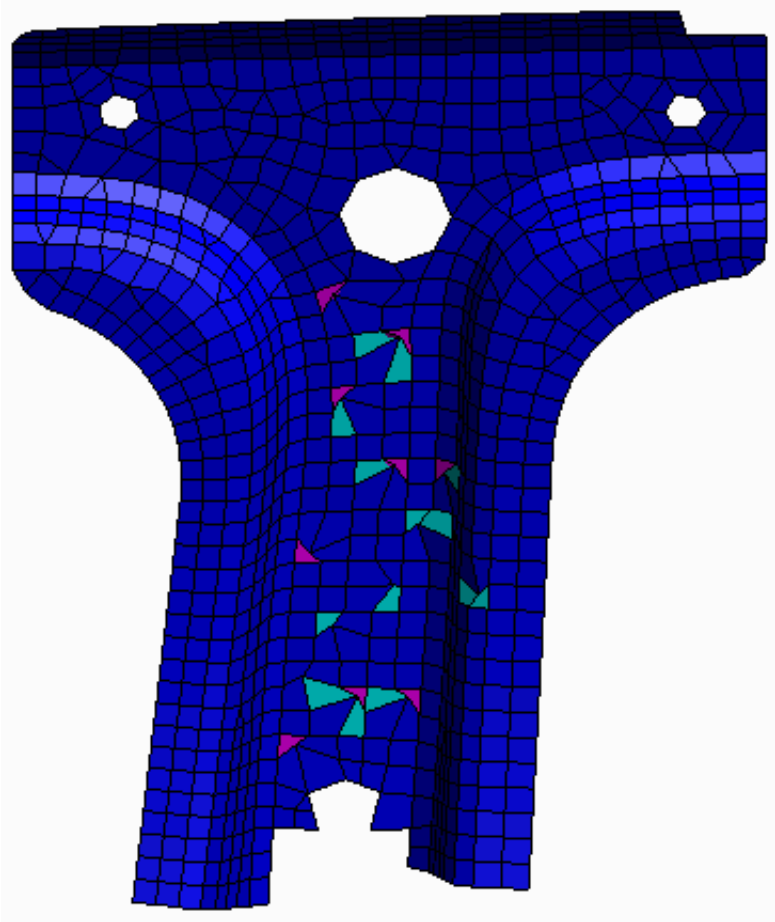


You are able to plot effective plastic strain values on initial stress cards in a number of ways. For parametric coordinates, the panel will contain a list of parametric coordinates of through-thickness shell integration points (-1 to 1 inclusive) sorted in ascending order. For integration points, the panel will contain a list of integration point numbers through the thickness. Finally, you can plot the maximum or minimum strain values for each shell. If the model(s) currently loaded in Primer does not contain effective plastic strain data, the list **"Shell intg pts"** shown above is blank.

The effective plastic strains at any one of the listed locations can be plotted for all elements in the model(s) by selecting an item from the list and clicking **'Update'**.

CT plot of element quality.

A contour of Quality Imperfection is shown below:



The **Settings...** panel for **Elem Qual** is shown below:

Individual quality metrics such as aspect ratio can be contoured using appropriate radio buttons.

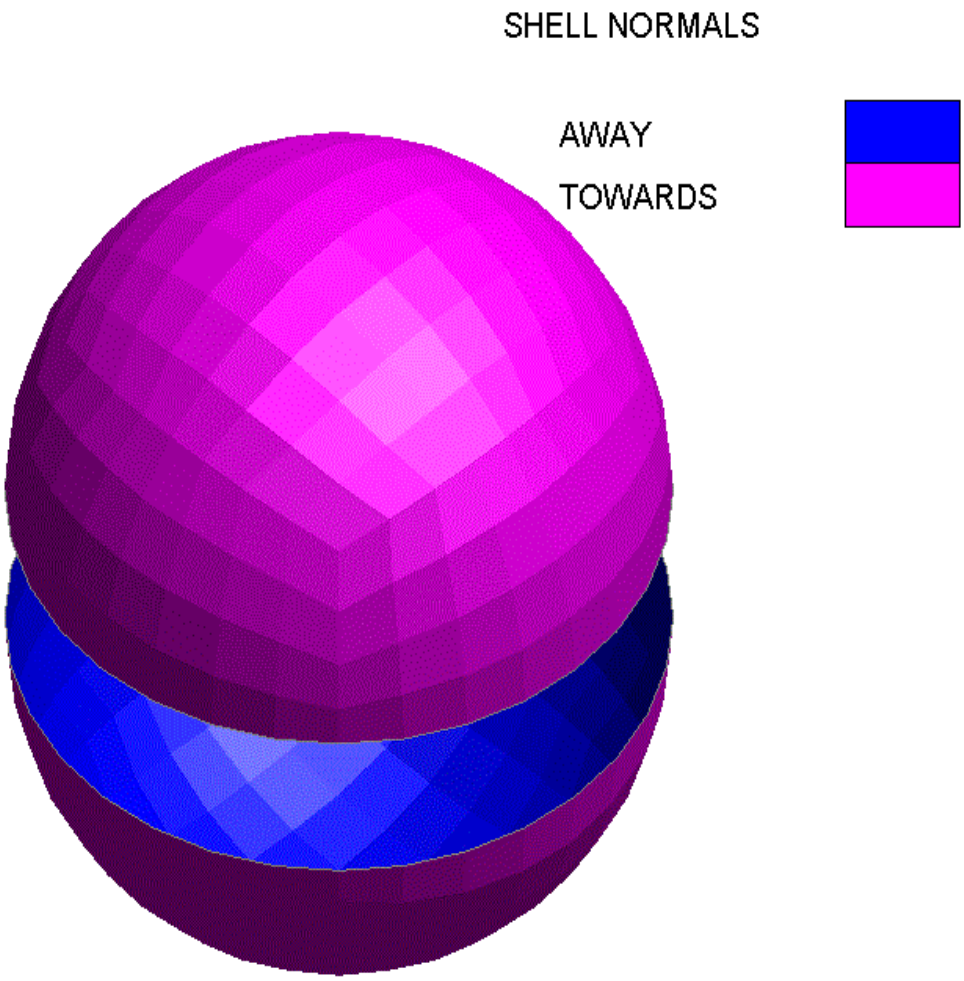
An overall penalty value can be contoured using the **Quality Imperfection** option. User can specify weighting factors used for this computation in the appropriate text boxes.

Elements failing one or more criteria can also be contoured using the **Failed Criteria** option. One or more metrics can be turned on or off using the appropriate tick buttons. Priority is given to that metric which has a higher overall penalty when an element fails multiple criteria.

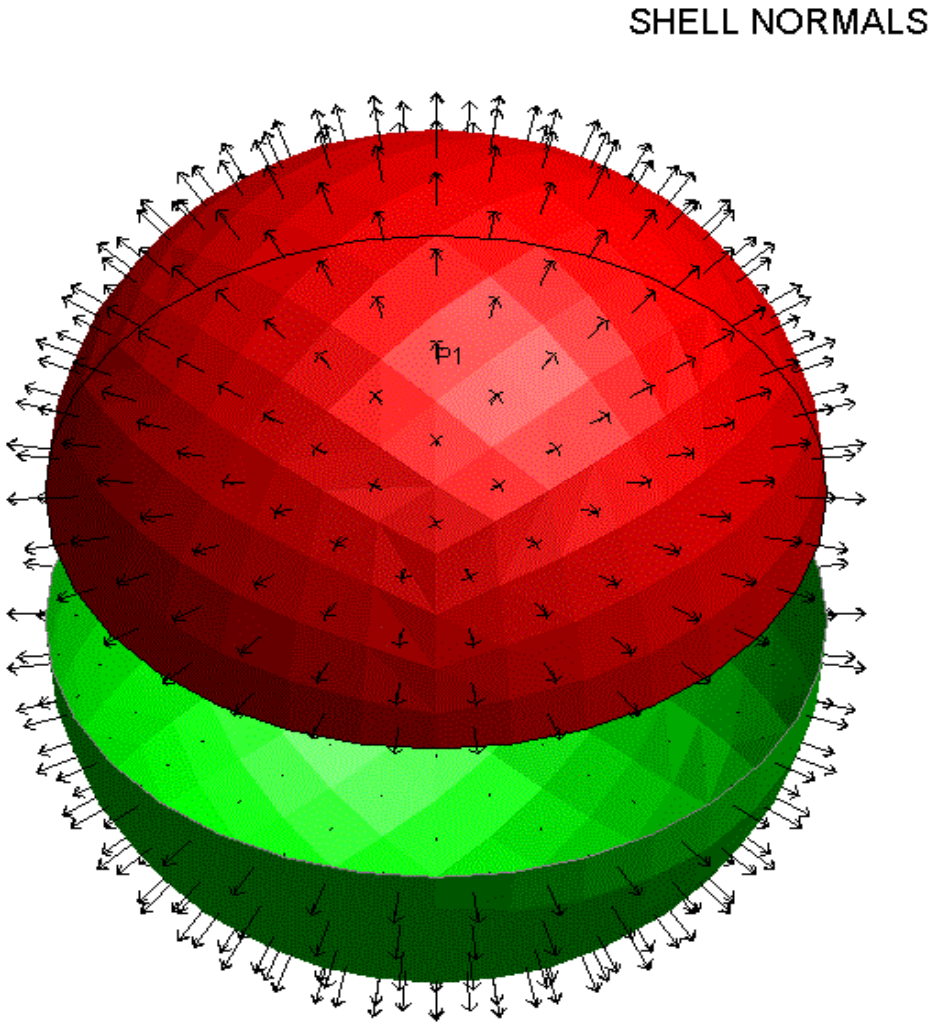
CT plot of shell normals.

There are 2 sub options **Contour** and **Vector** available for shell normals **CT/SI** plot.

The **Contour** sub option colors shells pink or blue depending on which direction the normal points to(Towards or Away from the user). The **Vector** sub option draws as arrow on the shell centroid pointing in the same direction as the shell normal.

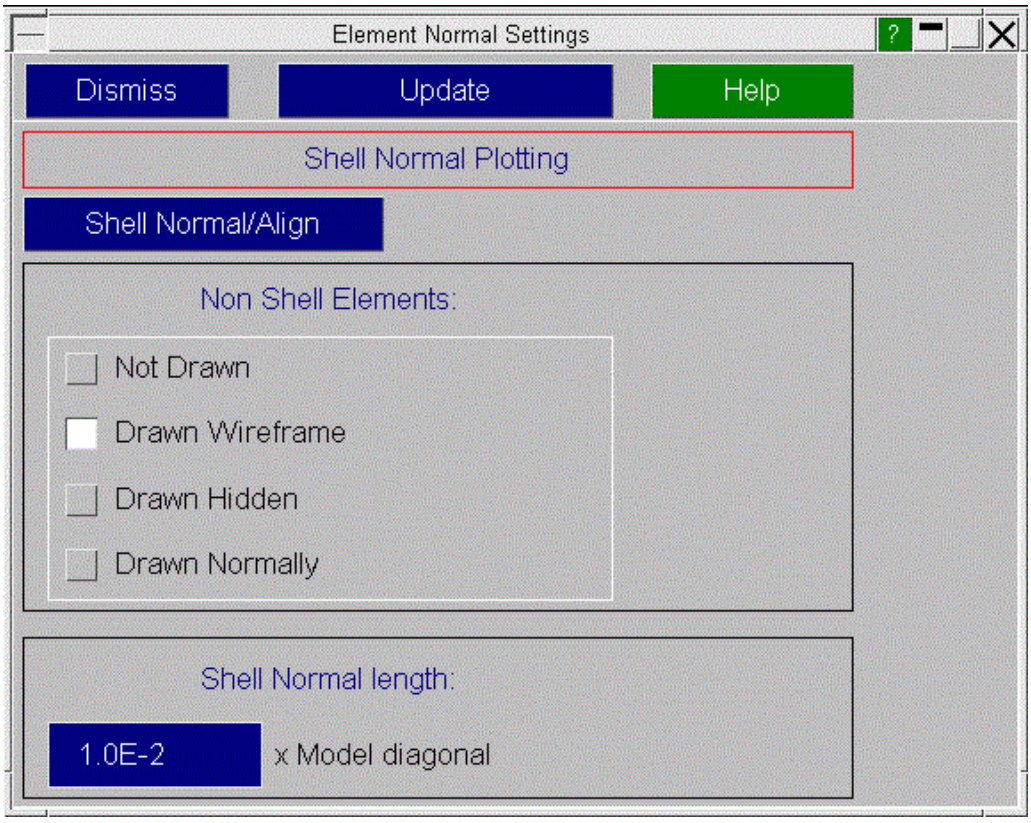


Plot showing shell normal **Vector** sub option.



Plot showing shell normal **Contour** sub option.

The **Settings** pop up for shell normals is shown below:



The Shell **Normal/Align** panel can be opened from **Settings** panel. In the **Settings** panel the user can change the mode in which non shell elements are drawn. The length of the shell normal arrows can also be changed as a multiple of Model diagonal length.

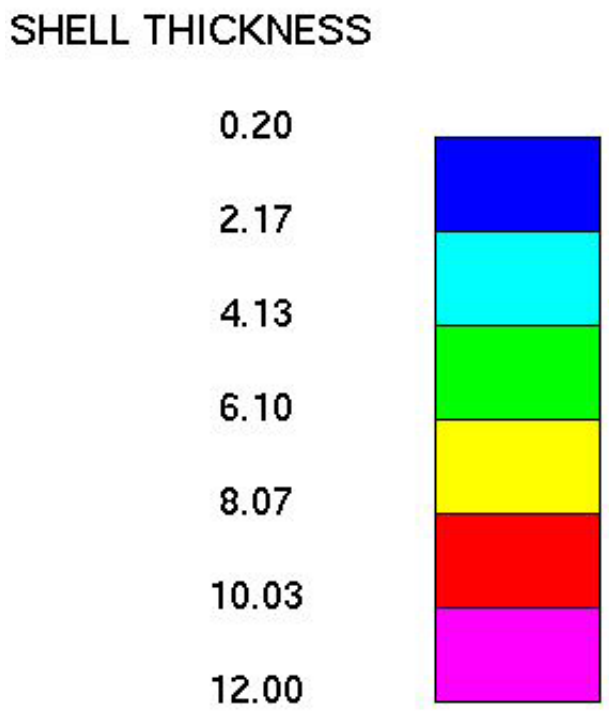
4.2.3 Contour levels on the contour ramp.

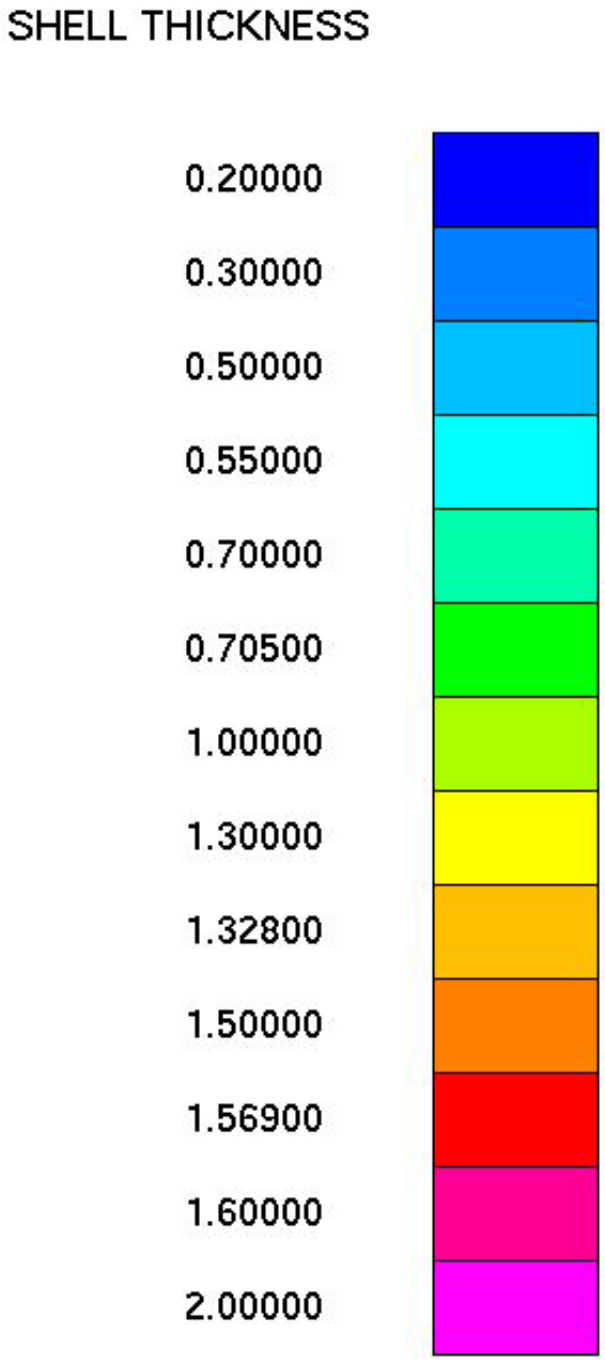
For all relevant Vector, CT and SI plots, the number of contour levels on the contour ramp can be set to any number between one and thirteen via the **Levels...** panel. When the number of distinct values being contoured is in excess of thirteen, the user-defined number of contour levels are displayed on the contour ramp with each colour representing a **range** of values being contoured.

However, if the number of distinct values being contoured in the visible model(s) is thirteen or less, each distinct value being contoured is allocated its own distinct colour on the contour ramp, and all values are automatically represented in it. Hence in such cases, the number of contour levels specified in the **Levels...** panel is ignored.

The following table illustrates the two different types of contour ramps just described. The first ramp is of a model containing shell elements with more than thirteen different thickness values. In this case, each colour represents a range of shell thicknesses, and the contour ramp contains six levels as specified via the **Levels...** panel. The second contour ramp is of a model containing shell elements with exactly thirteen distinct shell thickness values. In this case, each shell thickness is assigned its own colour in the contour ramp, thereby overriding the number of contour levels specified by the user.

This behaviour can be modified by enabling the ’Disable discrete contouring’ option from any of the ’Settings’ panels. If this option is specified, thirteen contour bands will be displayed even if the model contains fewer than thirteen distinct values unless a different number is specified in the ’Levels’ panel.

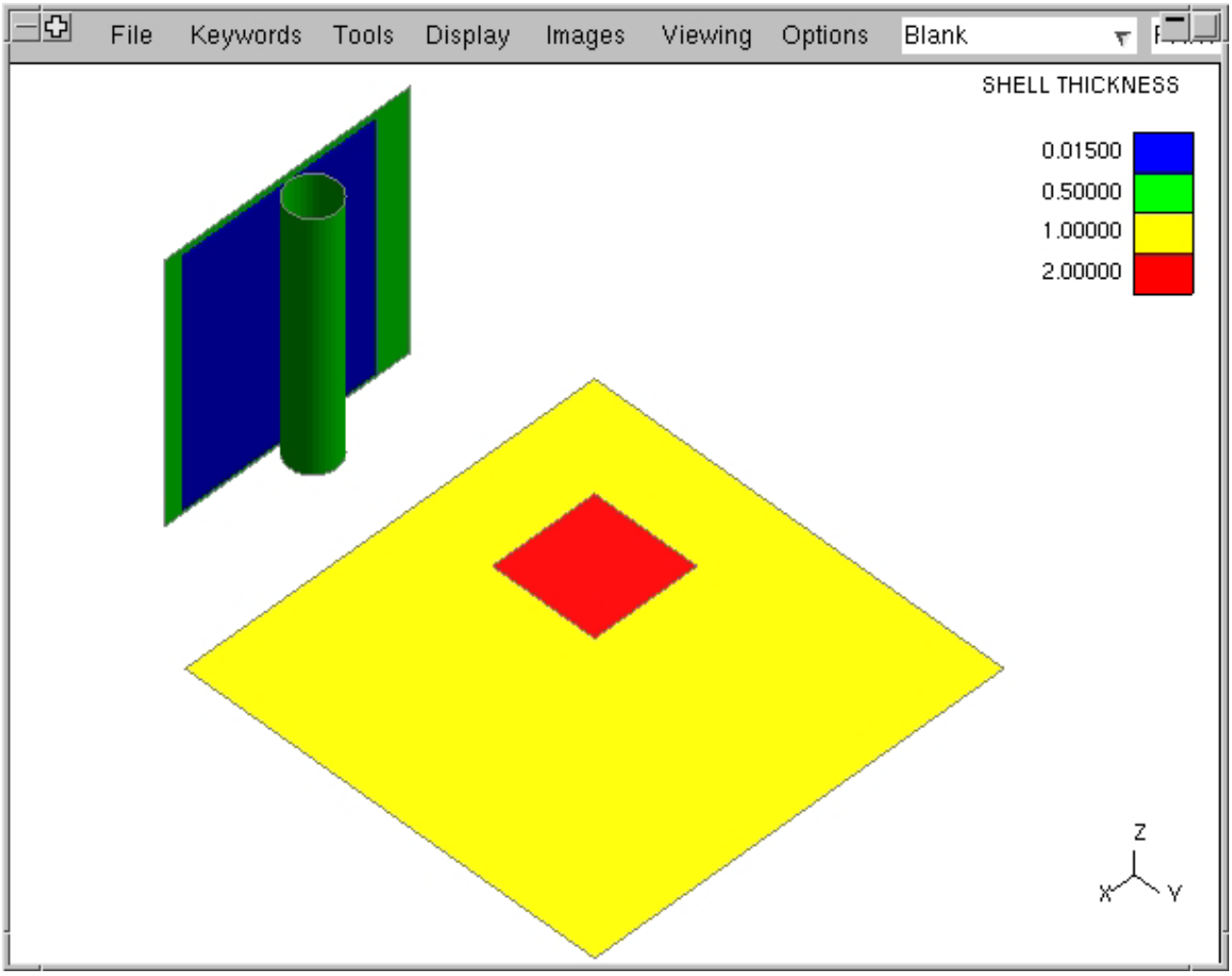




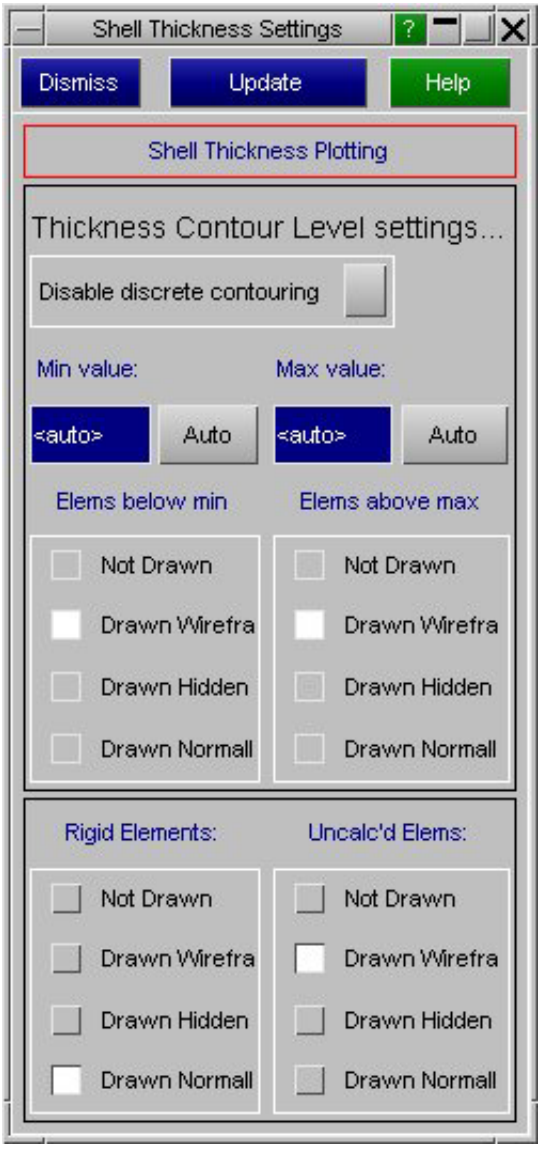
Contouring specific values in plots.

If required, a specific value of an entity can be contoured by specifying a narrow range of values in the **Settings...** panel.

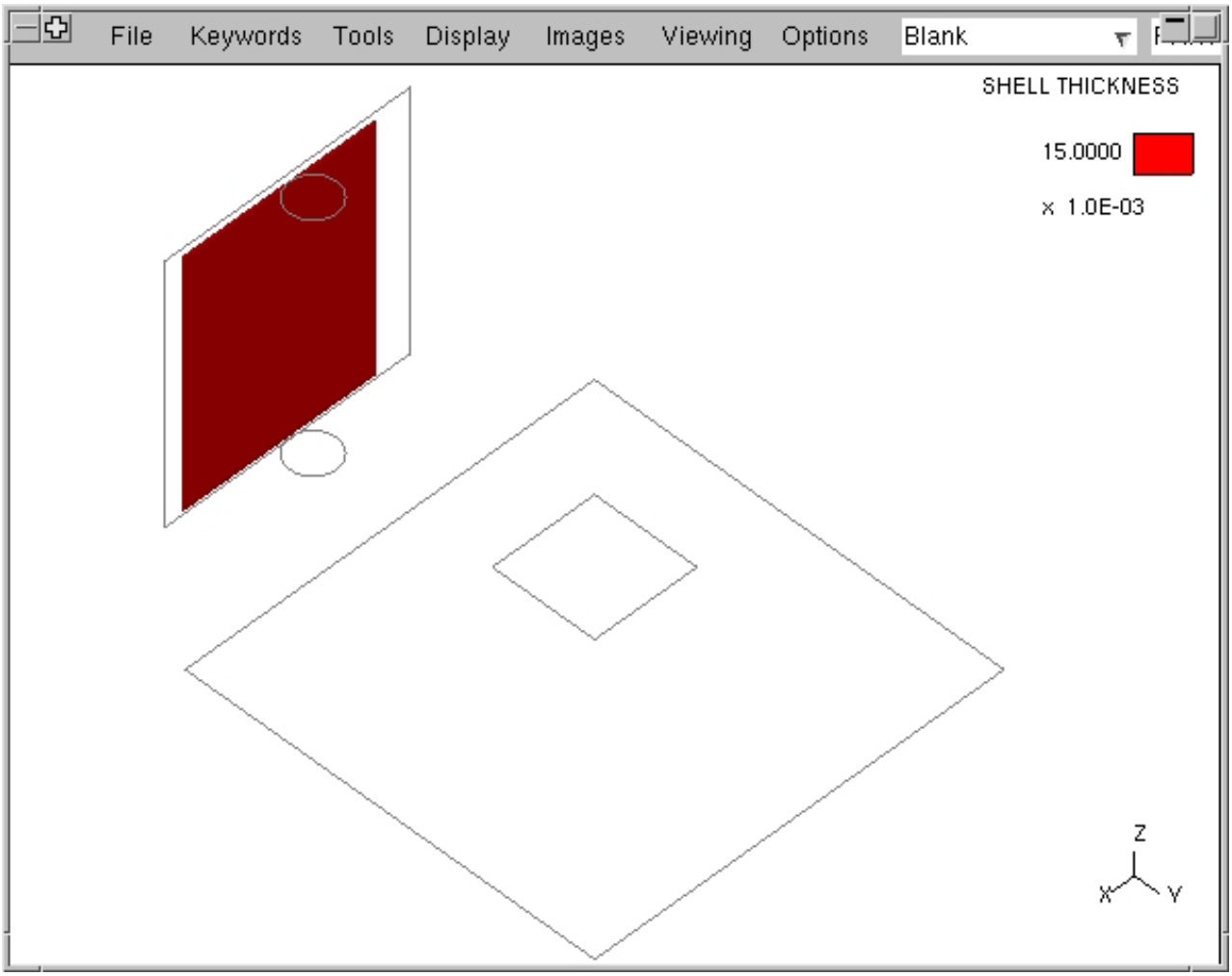
As an example, consider the following model containing shell elements of four distinct thickness values. The contour ramp thus contains four colours, each representing a distinct shell thickness value.



In order to visualize only those shell elements which are 0.015 units thick, a narrow range of values encompassing the desired value to be contoured is specified in the "**Min value**" and "**Max value**" boxes of the **Settings...** panel as shown below.


Shell Thickness Settings
Dismiss
Update
Help
Shell Thickness Plotting
Thickness Contour Level settings...
Disable discrete contouring
Min value:
Max value:
<auto>
Auto
<auto>
Auto
Elems below min
Elems above max
Not Drawn
Drawn Wirefra
Drawn Hidden
Drawn Normall
Not Drawn
Drawn Wirefra
Drawn Hidden
Drawn Normall
Rigid Elements:
Uncalc'd Elems:
Not Drawn
Drawn Wirefra
Drawn Hidden
Drawn Normall
Not Drawn
Drawn Wirefra
Drawn Hidden
Drawn Normall

Clicking the **UPDATE** button produces the following plot.



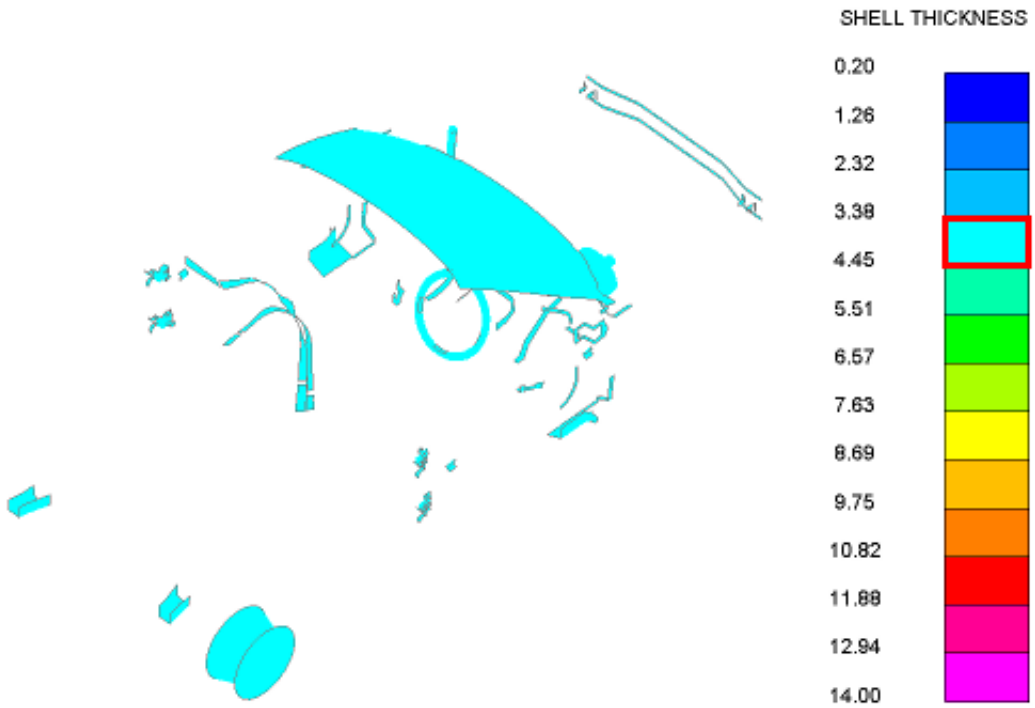
In the updated plot, shell elements with a thickness of 0.015 only are contoured as desired, while the remaining elements are drawn in the **wireframe** mode as per the options set in the **Settings...** panel.

Note that it is necessary to specify a range of values as opposed to the specific value to be plotted in the **Settings...** panel. If the exact value to be plotted is specified in both the "**Min value**" and "**Max value**" boxes, rounding errors that occur during computation might prevent the desired plot from being generated properly.

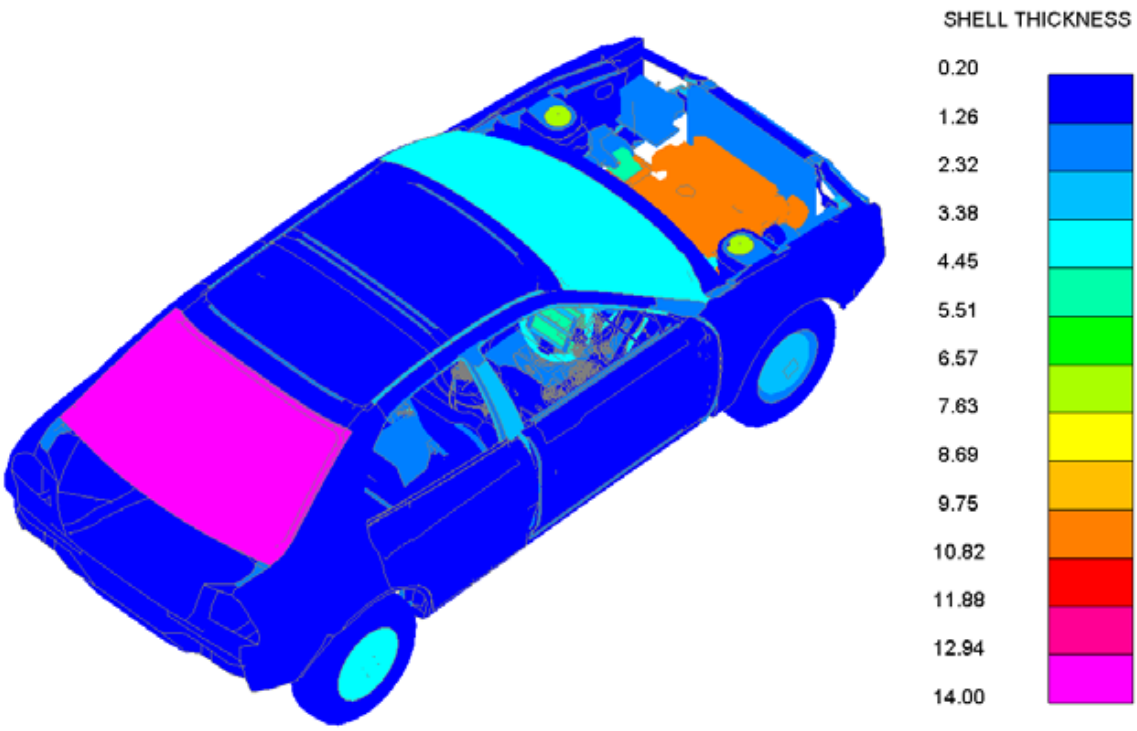
4.2.4 Contour refinement using the ramp.

For all relevant CT and SI plots, bands on the contour ramp now respond to cursor-driven actions.

Left clicking on a particular band restricts the contour to entities corresponding to the current band. It is possible to quickly interrogate contours corresponding to different bands by left-clicking on various bands successively.



A mid-mouse click anywhere on the ramp resets the contour to automatic mode.



Right clicking maps a popups that includes the following sub-options:

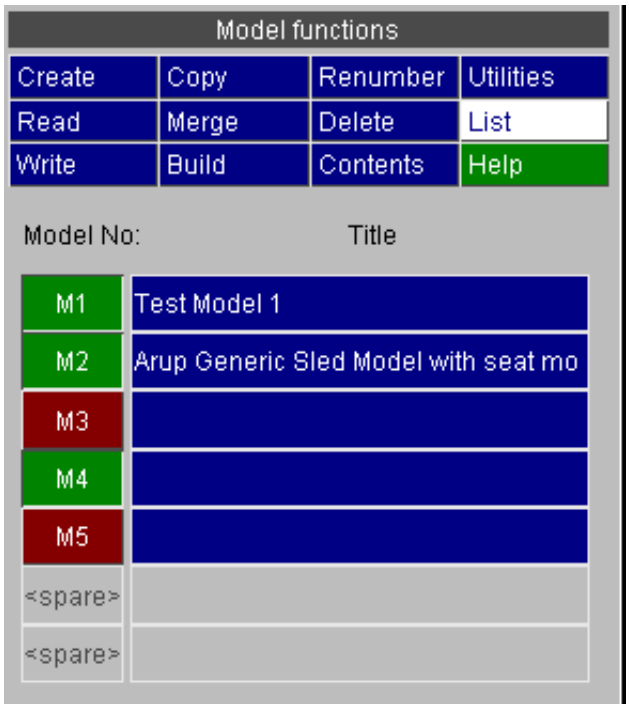
- **Only items in band**: As for left mouse above
- **Only + Reset contours**: As for left mouse click but contour bands are also reset to max and min using the current band as limits

- **Reset to auto**: As for mid mouse above
- **Settings**: Maps the settings panel for the current contour component

4.3 Controlling Model Visibility

Models can be enabled or disabled for display at will. This is carried out by setting them to "hidden" or "viewable": hidden models will not be drawn by any drawing command. By default a model is viewable when it is first read in, but thereafter its visibility is controlled by the user. Changing its status only takes effect the next time a drawing command is given.

Manipulating a model's status is simple:

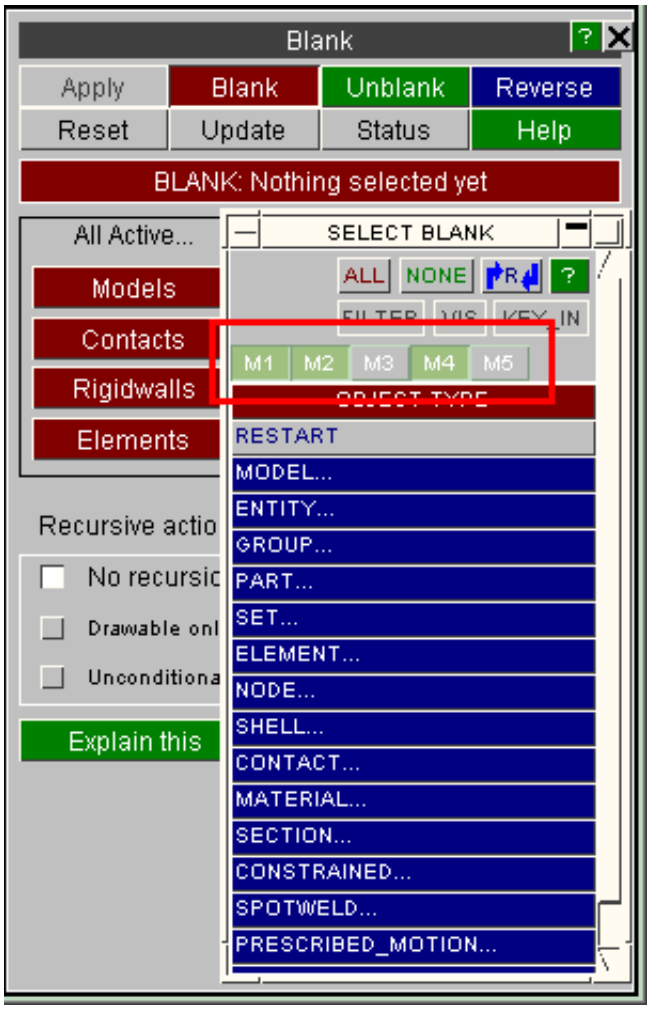


Under **MODEL** > **LIST** click on the **Mnnn** buttons in for the relevant models. A depressed button (green) is viewable, undepressed (red) is hidden.

In this example models 1,2 and 4 are viewable.

Setting a model's visibility in this way has the highest priority when determining whether something should or should not be drawn. If the model is not viewable none of its contents will be, regardless of Entity switches or Blanking. However, making a model viewable does not cause its contents to be displayed if the entity types are not visible (section 4.4) or if the entities are blanked (section 4.5).

In addition turning off a model in the **MODEL** > **LIST** menu has the effect of turning off its "**Mn**" tab in all selection menus throughout the code. For example given the case above of five models, with M3 and M5 deselected, the **BLANK** panel will start off looking like this:



Note that the M3 and M5 tabs are deselected. You can still turn them on manually if you wish.

In other contexts, for example when creating items, if you only have one model "live" in the **MODEL** > **LIST** menu the question "which model do you want to create it?" will be omitted, saving one mouse click.

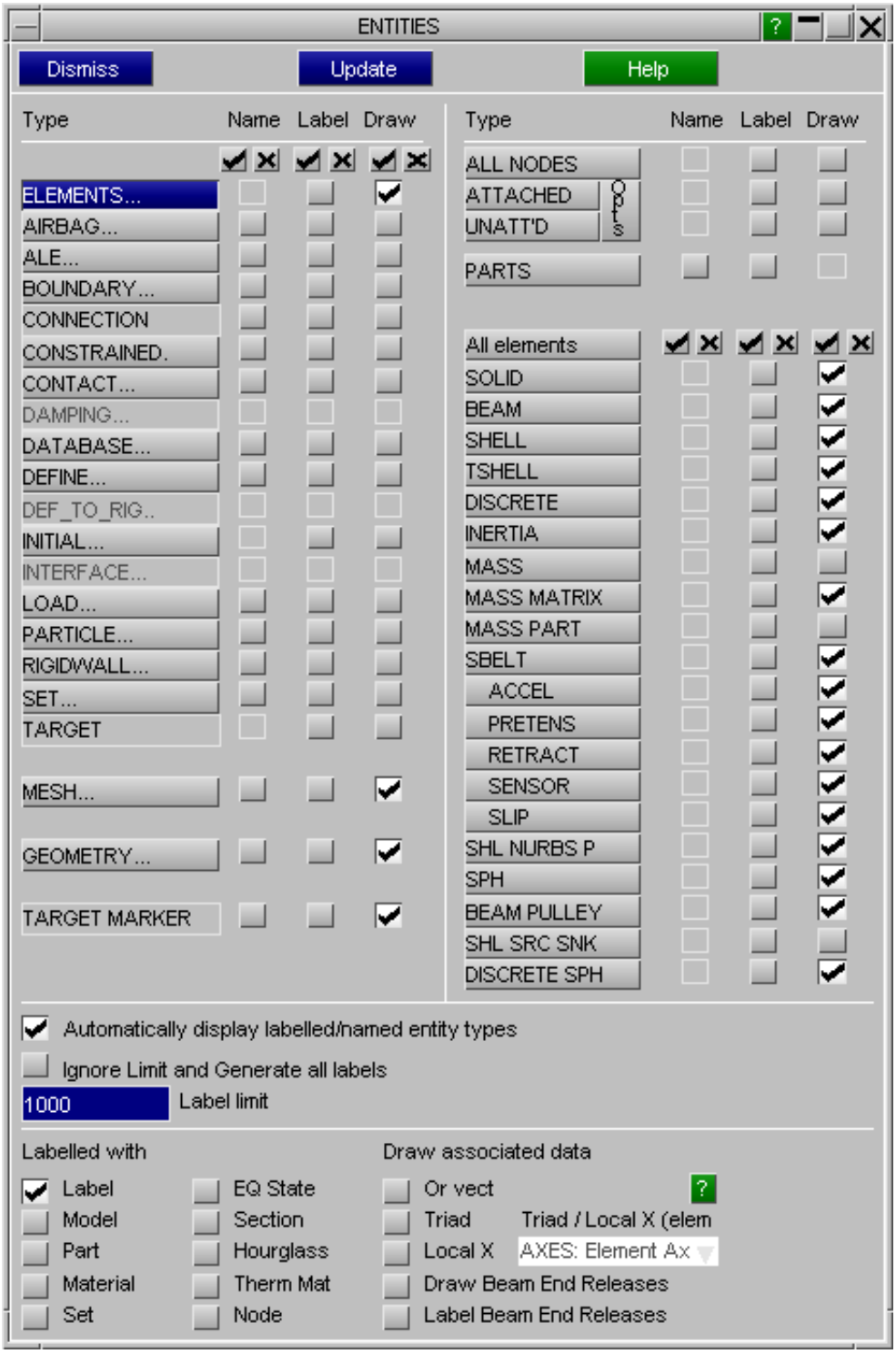
4.4 Controlling Entity visibility and labelling

By default only the elements in a model are drawn, with no labels, node symbols or other information appended to them.

You can add extra information to plots, control the display of classes of information and label items dynamically on the screen using the **ENT**ity Viewing panel. This can be accessed in 3 ways:

1. The keyboard shortcut key **E**.
2. The top bar menu **DISPLAY** > **ENTITIES**
3. The **ENT** from the viewing and drawing window.

This panel controls the display of elements and nodes, (ie basic "structural" items); also their symbols, labels and local direction triads as well as the display of "other" items, such as constraints, contacts, rigidwalls, etc; and also their labels, symbols and other related displayable data.



It must be stressed that these commands only permit or deny the display of *classes* of information, they do not control the visibility of individual items or models. However they do provide one means of accessing the "dynamic" labelling of items: see section 4.6.

For example they might be used to enable the display of nodes and of contact surfaces. This would permit nodes and contacts in any models to be displayed provided they were not made invisible by some other command.

The left hand column of the panel dictates the display of the right hand column. At any one time a "master" category will be selected from the left-hand column (in this example **Elements** is selected). The "master" categories each contain further "child" categories below them. The right hand column displays the appropriate "child" categories for the selected"master". The **Label** columns control whether or not the items will be labelled (with the information selected under **labelled with.** The **Drawn** columns control whether or not the items will be drawn. "Child"categories can be controlled individually (in the example shown the display of beams has been turned off), or all the child categories may be switched on/off together by switching on/off the master category or the ALL_<*category*> row.

Labelled with determines what is actually drawn as a "label" when labelling is selected for an element or node class.

Selecting multiple labelling categories will lead to compound labels being generated (eg **M1/H1001/P12/MAT12**) and plots will become very cluttered if too much information is displayed.

4.4.1 Elements and nodes (Structural Items).

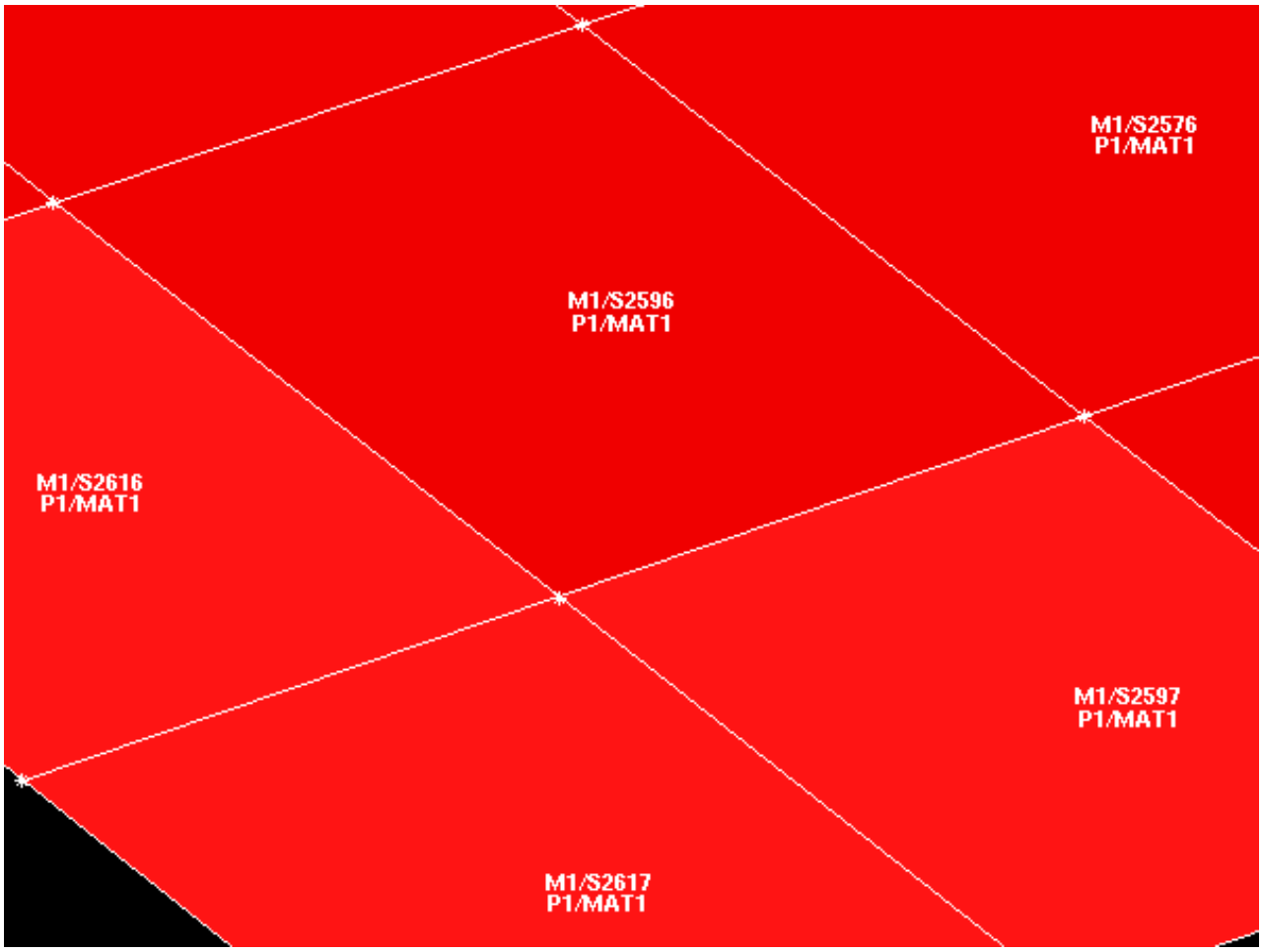
Nodes are treated as a special case:

- **ALL NODES** draws all nodes, regardless of attachments.
- **ATTACHED** draws only nodes attached to some other items currently displayed.
- **UNATTACHED** draws only nodes that are not attached to anything (visible or not).

Associated data: Local direction **TRIADS** are drawn for element types with coordinate systems, and **OR**(ientation)_**VECT**(or)**S** for springs and dampers.

4.4.2 How labelling on plots is handled for nodes and elements

The default label is a node or element number, but a variable amount of information can be generated to form a "label" which can run to multiple lines, as this example shows:



This figure shows an example of shells which have been labelled with:

MODEL	**Mnnn**	for ***M***odel number <nnn>
LABEL	**Snnn**	for ***S***hell <nnn>.
PART	**Pnnn**	for ***P***art <nnn>.
MATERIAL	**MATnnn**	for ***MAT***erial <nnn>

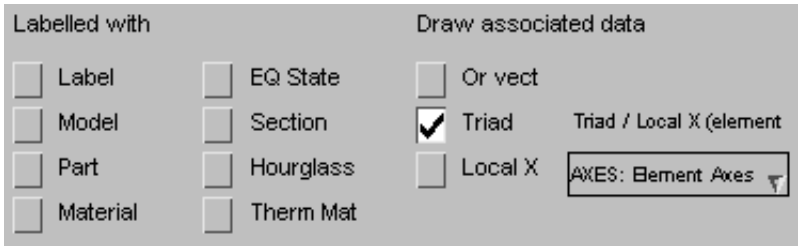
PRIMER attempts to group labels logically and to locate them so that they don’t overlap, but if you try to add too much information you will end up with a total mess on the page. This example, with four categories of data labelled on elements, is the sensible maximum; and even it starts to get messy when label numbers get large (> 5 digits).

Labelling uses the standard acronyms for entities, these are listed in Appendix 1.

The "attached" nodes in this figure have also been switched on: these are drawn as asterisks (*) at the relevant element vertices.

4.4.3 Triads (elements)

It is possible to draw triads on elements that would depict the local material orientation. Alternatively, the local X direction can be drawn by toggling the appropriate button "On".

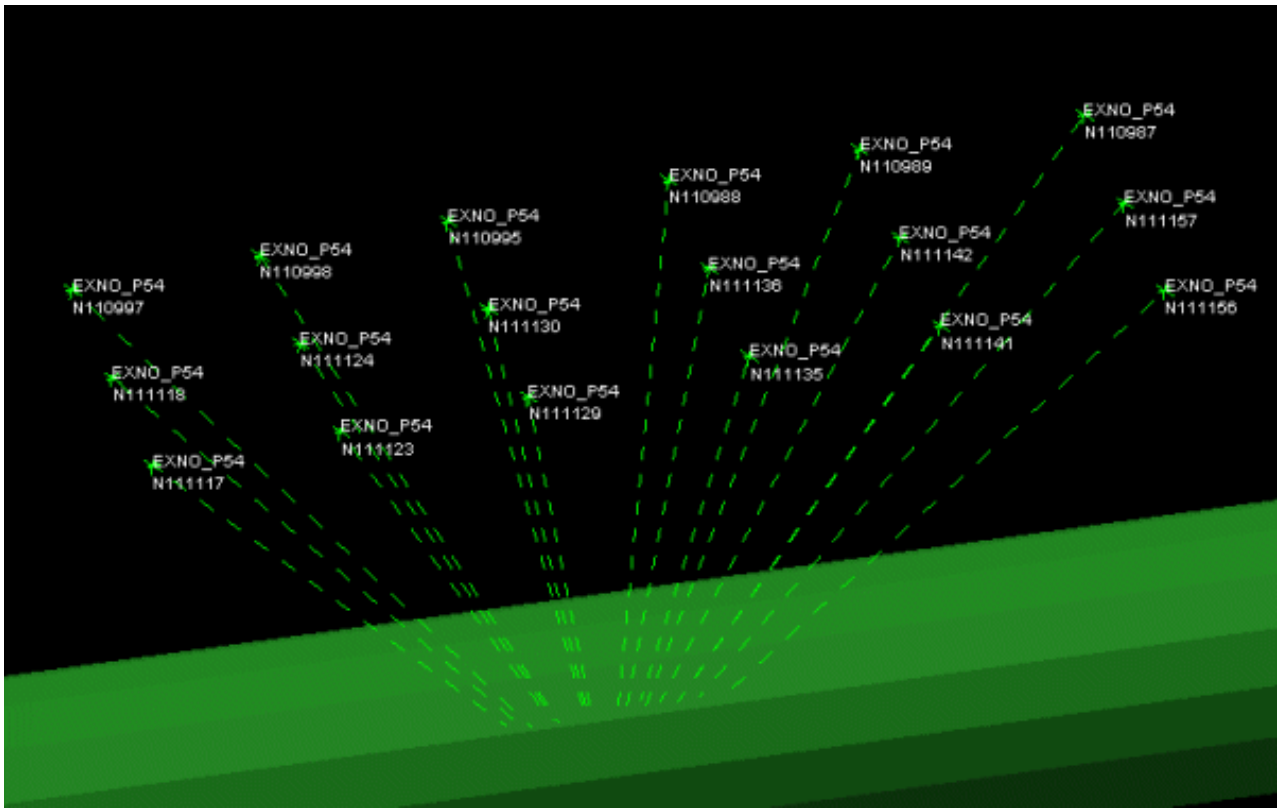


The following options are available for drawing element triads/local X direction and can be chosen using the popup:

Element Axes	This is the default option. Element orientation, as defined by its topology is drawn. Local angle specifications are disregarded.
Material Axes	Local angles as defined by MAT, ELEMENT_SHELL_BETA, ELEMENT_SHELL_MCID, ELEMENT_SOLID_ORTHO are computed. A suitable triad/local X is drawn on each element. However, layer-specific angles are not evaluated.
All layers	This option is only applicable to shells. Local angle calculation is carried out as in the "Material Axes" case. In addition, local direction specification is considered for each integration point. This can be defined using PART_COMPOSITE cards or using SECTION_SHELL cards in conjunction with INTEGRATION_SHELL or Gaussian or Lobatto integration rules. A triad/local X is drawn for each layer.
Top, bottom, middle layers	Local angle computation is carried out as in the "All layers" case. However, only the top, bottom, and middle layers are sketched. A sensible middle integration point cannot be identified if an INTEGRATION_SHELL is defined or if the element has even number of integration points. In such cases, only the top and bottom integration points are drawn.
Intg pt <n>	Local angle computation is identical to the "All layers" case. However, the triad/local X is drawn only for the specified integration point <n>.

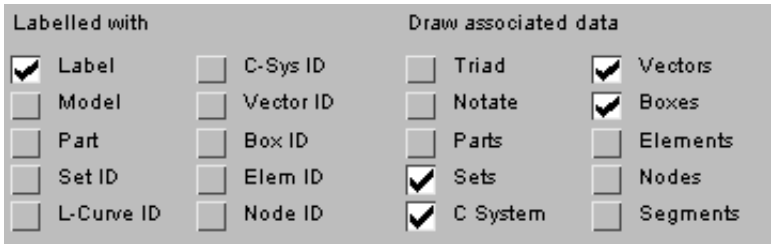
4.4.4 "Non-element" items

Most of the viewable entity types are shown by special symbols. For example, *CONSTRAINED_EXTRA_NODES are illustrated below.



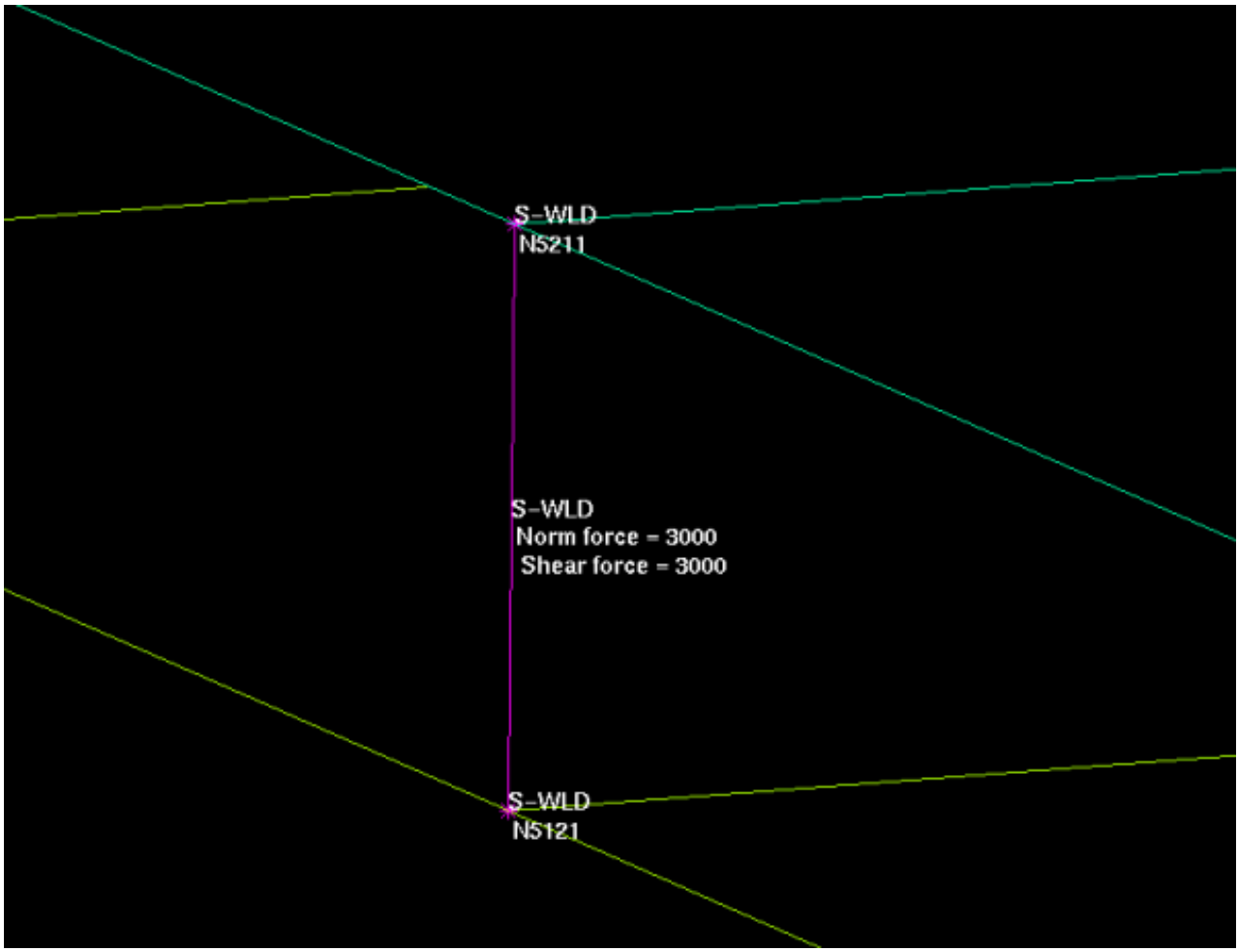
4.4.5 How "associated data" is drawn for non-element items

The addition of extra "associated" data to symbols is controlled by the lower half of its box:.



An example might be a contact surface, which is defined by sets of parts, sets of nodes, bounding boxes and which references a load-curve. It is possible to draw and label all these items (although the screen might get a bit cluttered!)

Therefore it is necessary to get used to the idea of primitive objects (nodes, elems, boxes, ...) being drawn because they make up part of some other higher order entity, and being labelled with that entity.



In this example, which shows a ***CONSTRAINED_SPOTWELD**, both the weld itself and the nodes at its ends have been drawn.

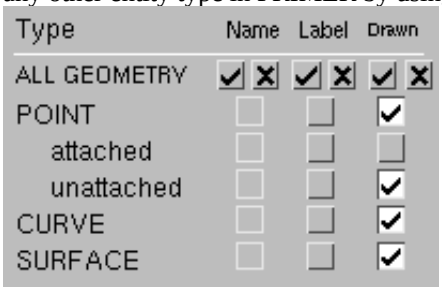
The labelling of associated nodes has been turned on, but note that the primary label on the nodes associates them with the spotweld since it is their "parent" entity in this context.

This example also demonstrates the use of the **NOTATE** function. This is a global function which adds "useful" data where possible to some visible items, in this example the normal and shear force values of the spotweld.

Other examples are the stiffness and damping factors of joints; motion and orthotropic data on rigidwalls; stiffness and stop angle data on generalised stiffnesses; and so on. Generally speaking **NOTATE** adds extra data to items where this can be expressed concisely enough to fit onto the screen.

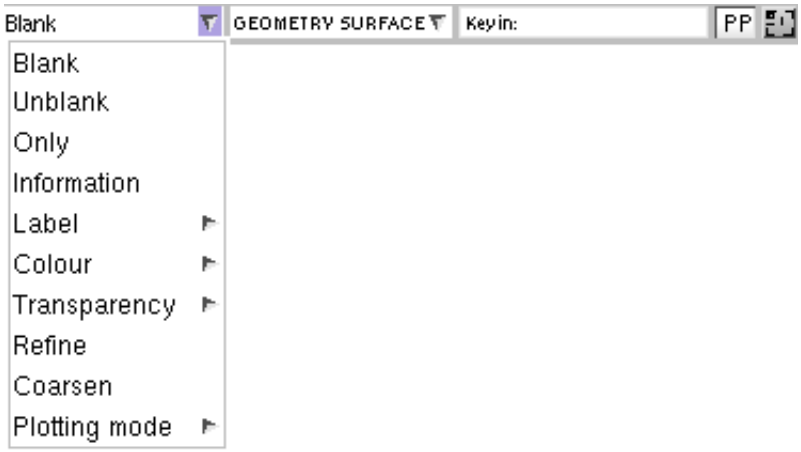
4.4.6 Geometry entities

From version 10.0 PRIMER can display basic geometry. Points, Curves (and lines) and Surfaces can be displayed. By default curves, surfaces and any points that are not attached to a curve or surface are drawn. This can be changed like any other entity type in PRIMER by using the **GEOMETRY...** button.



For better visualization of surfaces, **Refine** & **Coarsen** features have been added to the "Quick Pick" menu. **Refine** operation improves the smoothness of rendered surface(s) and vice-versa for **Coarsen** operation. They can be

accessed from the "Quick Pick" menu by selecting the entity as **GEOMETRY SURFACE**. For details on "Quick Pick" functionality refer section 2.9.



Note that the geometry engine in PRIMER is still in its infancy. There are currently no facilities to create or modify geometry. Additionally rendering of many surfaces will make PRIMER use considerably more memory and will make drawing slower. It is expected that the geometry engine will be developed over time. Please contact Oasys Ltd if you encounter any problems.

4.5 BLANKING Controlling entity visibility

Blanking allows the user to cut down what is displayed by controlling whether individual items are marked as drawable or not.

For an item in PRIMER to be drawn it must pass the following three tests:

Is the model visible?	=>	Is the entity type drawable?	=>	Is the entity unblanked?
(See section 4.3)		(See section 4.4)		(This section 4.5)

These represent increasingly more detailed levels of testing and the last of these checks, blanking, is performed on a per entity basis. Every drawable entity in PRIMER may be flagged as

- either "***blanked***" (not eligible for drawing)
- or "***unblanked***" (eligible for drawing)

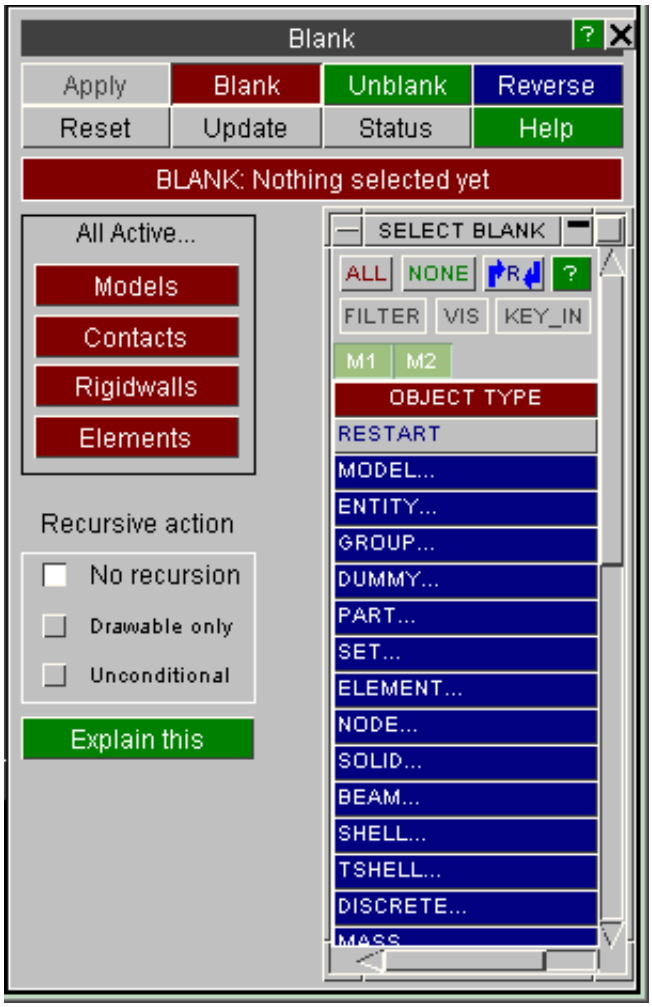
The default being ***unblanked***. Control of the blanking status can be exercised in the standard hierarchical fashion of models, sets, parts and finally down to individual items; thus it may be used to control exactly what is seen on the screen. The way that blanking selection propagates down through the model can be controlled by the **Recursive Action** setting - see section 4.5.1 below.

As well as the main **BLANK** menu described in this section, Blanking may be activated by:

- Quick Pick blanking (section 4.5.2 below)
- The Part Tree (section 4.5.3 below)
- Keyboard short-cut keys (section 4.5.4 below)
- Special keys in the View panel (section 4.5.5 below) which include "locking".

4.5.0 The BLANK menu

This figure shows the main **BLANKING** Menu:



It has three colour-coded states:

BLANK	(Red) makes the selected entities invisible.
UNBLANK	(Green) makes the selected entities visible again.
REVERSE	(Blue) inverts the status of the selected entities.

To use it: select a list of items, choose one of the three states above, and press **APPLY**.

The effect will be seen the next time the image is drawn, or when **UPDATE** is used.

The **ALL_xx** commands are to provide short cuts for commonly issued commands:

ALL_MODELS	Means ***everything***! All the contents of all models currently in memory will be operated on.
ALL_CONTACTS	Means all contact surfaces in all models.
ALL_RIGIDWALLS	Means all rigid walls in all models.
ALL_ELEMENTS	Means all elements (of all types) in all models.

These short cut commands will operate faster than the equivalent commands from the **SELECT** menu since they don't have to perform the hierarchy propagation checks implicit in using the menu.

RESET	Resets to null the contents of the **SELECT** menu. This may be used to delete any current selection and start again.
UPDATE	Redraws the current image following a blanking change. This is necessary to see the effect of any changes (unless the Update Level in the View Control box has been set to "frequent", in which case changes take effect immediately).

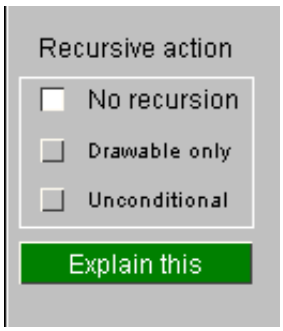
4.5.1 Recursive Blanking

Blanking in PRIMER has always been a contentious issue because of the way that selection propagates down through the hierarchy of items in a model. The increasing complexity of ls-dyna models has exacerbated this problem.

In earlier versions, where blanking propagation was unconditional, a user could accidentally blank a "junior" object (typically nodes) through propagation without being aware that this was happening. This would then give rise to "why won't it draw xxxx?" questions, which could only be solved by unblanking things which the user didn't think he had blanked in the first place!

In PRIMER V9.1 an on/off switch for blanking propagation was provided.

In PRIMER V9.2 yet more control has been provided over how blanking propagates through the structure by allowing the user to set the **Recursive Action** value, which controls how blanking (but not any other form of selection) is propagated down a model hierarchy.



"**No recursion**" means that only the selected items are blanked, with no propagation.

"**Drawable only**" means that blanking propagates downwards, but only affects items that are currently drawable (ie their Entity switch is on).

"**Unconditional**" propagates blanking unconditionally down through the model.

The "**No recursion**" case has some exceptions built into it as follows:

- Blanking an item that is not itself drawable, but which is drawn via its underlying items (eg MATERIAL, PART, SECTION, SET, etc), causes limited propagation downwards to the drawable items. Thus, even with "**No recursion**" set, blanking a MATERIAL will propagate downwards to blank the elements of that material, but not any further (ie not to the nodes on the elements).

- SETs are another special case when "**No recursion**" is selected:

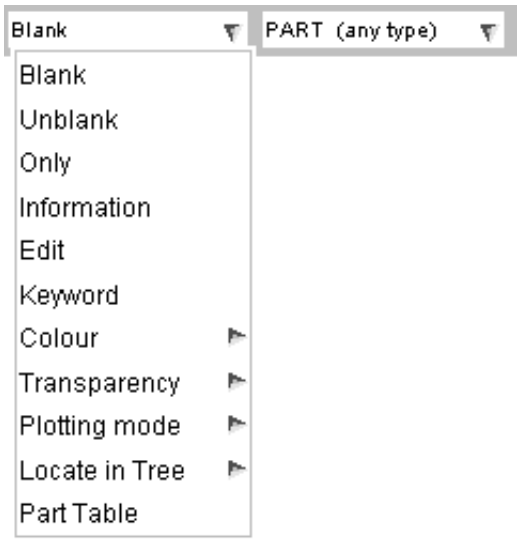
  - Normally SETs are ***not*** drawn explicitly, and if a SET is blanked then the blanking propagates down as above to the drawable items in the set.

  - However if SETs ***are*** drawn then blanking them stops them being drawn, but does ***not*** affect the visibility of the underlying items, meaning that the SETs will no longer be superimposed on the image.

The **Recursive Action** flag also affects how "Quick Pick" and Part Tree blanking propagate (internally BLANK, "Quick Pick" and Part Tree blanking are the same, simply using different selection methods.)

4.5.2 "Quick Pick" Blanking.

The "Quick Pick" functionality has been described in section 2.9. However it is so central to PRIMER usage that it merits a brief repeat here.



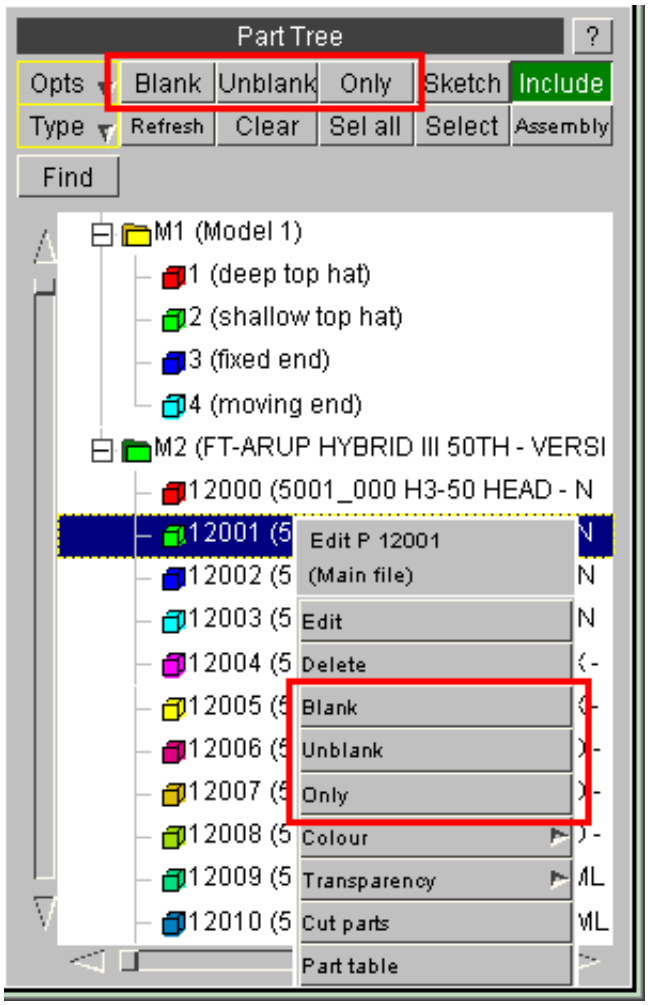
Blank Blanks the selected items.

Unblank unblanks them

Only draws only the selected items, effectively blanking everything else.

4.5.3 Part Tree Blanking

Blanking may also be performed from the Part Tree (see section 6.17).



The current operation of the Part Tree can be set to **Blank**, **Unblank** or **Only**.

Similarly a right-click popup on a given row also contains those options.

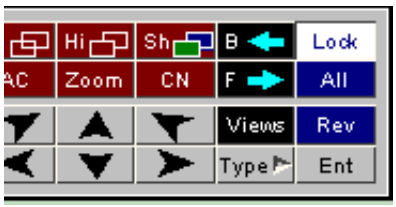
4.5.4 Blanking control using keyboard shortcut keys

The following keyboard "short cut" keys influence blanking:

U (**nblank all**)	Unblanks everything, ***unconditionally***, subject to "locking" (see below)
R (**everse all**)	Reverses the blanking status of everything, using the current **Recursive Action** logic.

4.5.5 "Locking" blanking in the "View" panel.

Blanking may also be "locked" to its current status via the following buttons in the View panel:



Lock "locks" the current blanking status so that keyboard short cut **U**(**nblank all**) returns to the "locked" visibility status. The Lock button toggles on / off.

All is exactly the same as keyboard shortcut **U**(**nblank all**) above

Rev is exactly the same as keyboard shortcut **R**(**everse all**) above

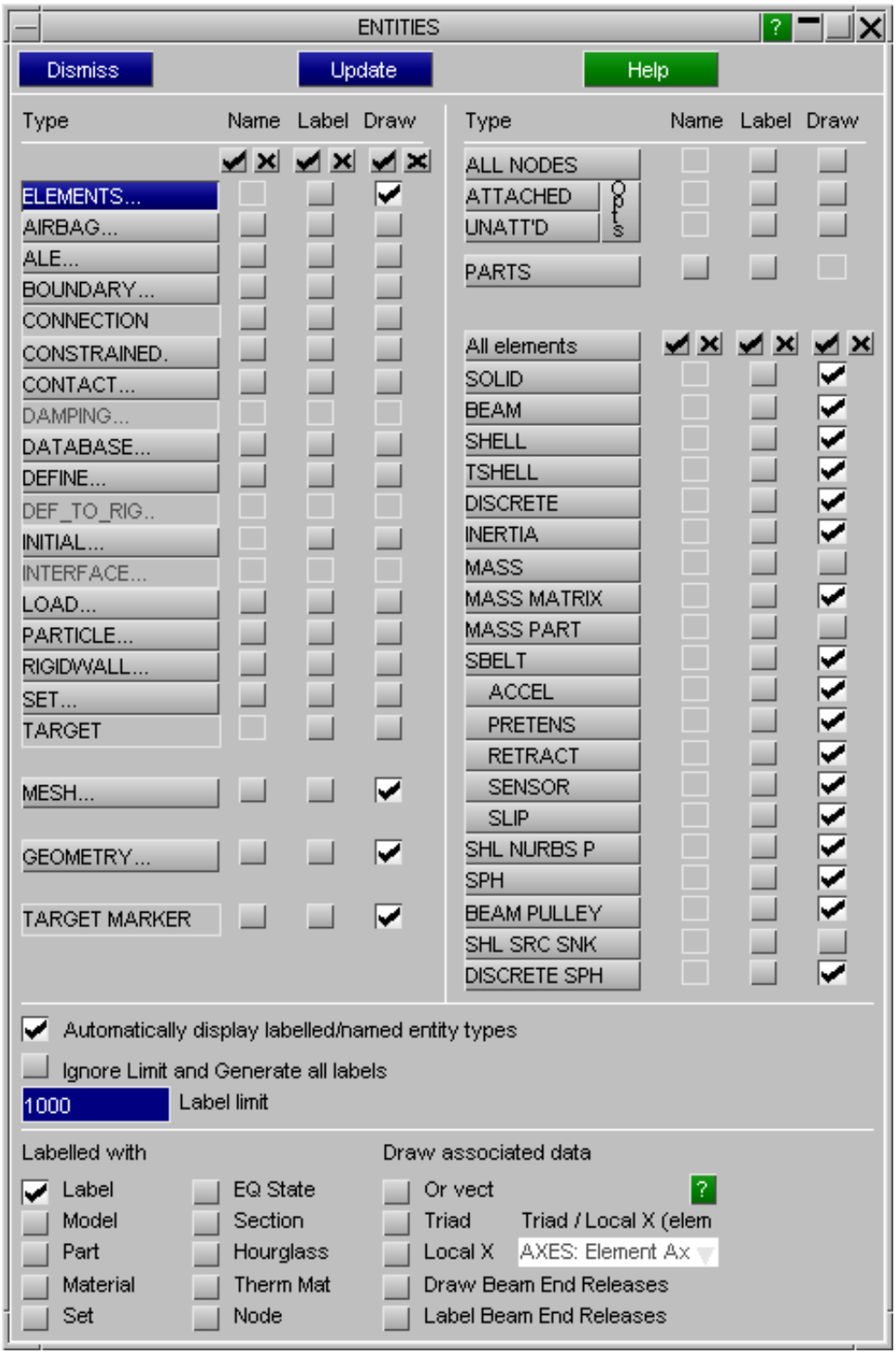
The purpose of "locked" blanking is to allow the user to return easily to a previous image.

"Locking" does ***not*** affect blanking carried out by the main Blank Panel, Quick Pick blanking, or blanking from the Part Tree.

4.6 Dynamic Labelling

Sketching labels and associated information on the existing plot.

4.6.1 Using the "type" Element and Node buttons in the Entities panel

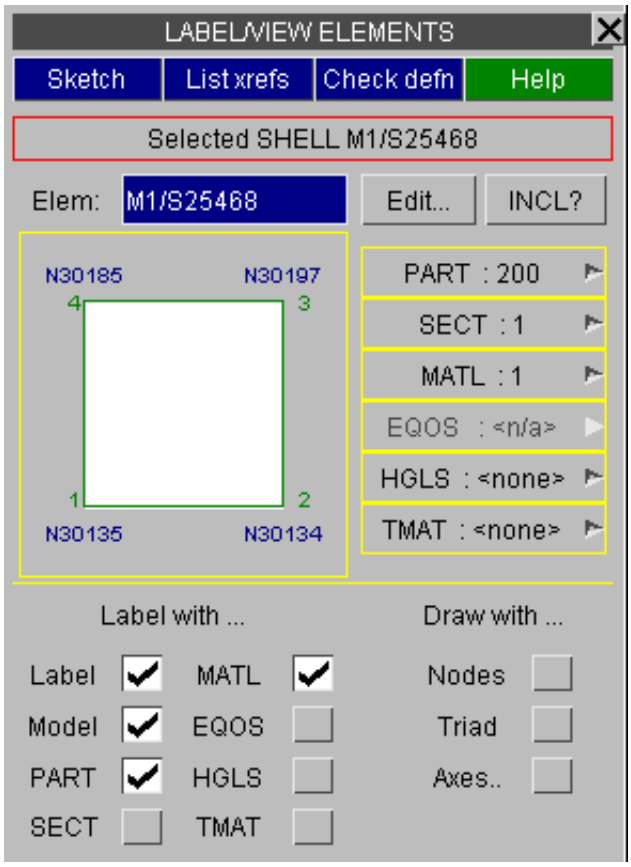


In section 4.4 above the use of the ENTity Viewing panel to control labelling on plots was described.

It is also possible to label nodes and elements "dynamically", which means "instantly on the existing image".

- Select**ELEMENTS** from the left hand "type" column
- Select a category from the right hand "type" column .
- This maps the labelling panel for that item type (below).
- Click on items of that type to label them immediately.

Alternatively, use Quick Pick, set the entity type to SOLID, SHELL, etc and the action to "Element Details".



This figure shows a typical dynamic labelling box for shell elements.

It is updated automatically as you click on elements, or you can type a new element number into the **Elem:** box.

More than one model is current in this example, so typed in elements must be prefixed with their model id. In this example shell element 25468 in model #3 (**M3/S25468**) has been selected.

The **EDIT** button invokes the detailed editing panel for this element. The **INCL?** button lists the elements position in the model's include file structure. **List Xrefs** invokes the cross-reference viewer for this element (see section 6.15 for more details).

Not only is the element in question labelled on the screen, but its major attributes are presented in this panel:
- The nodes on the element are drawn schematically. (Note that the schematic shape is idealised, here as a square, not the true shape of the element.)
- Its Part, Section, Material and other attributes are given.

By using the popup menus against the **PART, SECT**, etc boxes it is possible to view the details of these in their respective edit/browse panels.

The "**Label with ...**" buttons control how the selected items are labelled on the screen. The categories are the same as those in the main **ENT**ity Viewing panel, but apply only to these "dynamically" labelled items.

The "**Draw with ...**" buttons control what extra information is added to the selected items:

Nodes Adds the labels of nodes connected to this element

Triad Draws the local axes as a triad (if relevant for this class of element)

Axes Draws other local axes where relevant: orientation vectors, etc

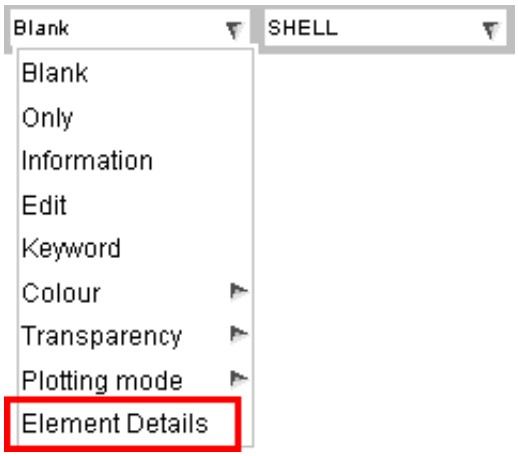
Selection will be limited to the class of item selected in the **ENT**ity Viewing panel. However selecting class

ALL_ELEMENTS permits any class of element to be selected for labelling.

The details of this **LABEL/VIEW** panel will vary with the class of object being shown: for example the panel for nodes doesn't show a diagram but rather lists coordinates, restraints, rigid connectivity (if any), etc for the node.

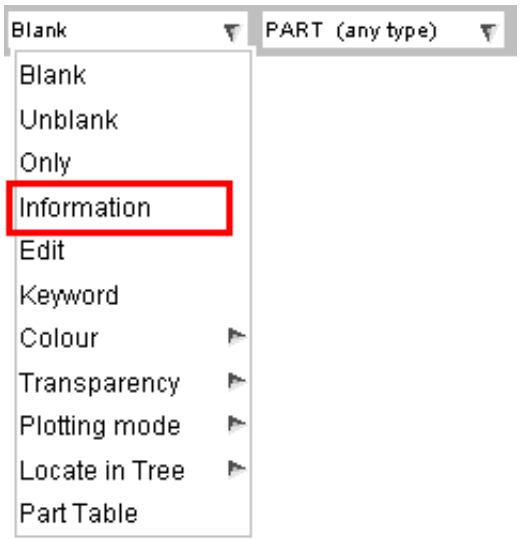
This panel can also be invoked anywhere in PRIMER from popup windows offering the **LABEL/VIEW** option

4.6.2 Using the "Quick Pick" Element and Node Details option



The same panel may also be invoked using the Quick Pick **Details** option for nodes and elements only.

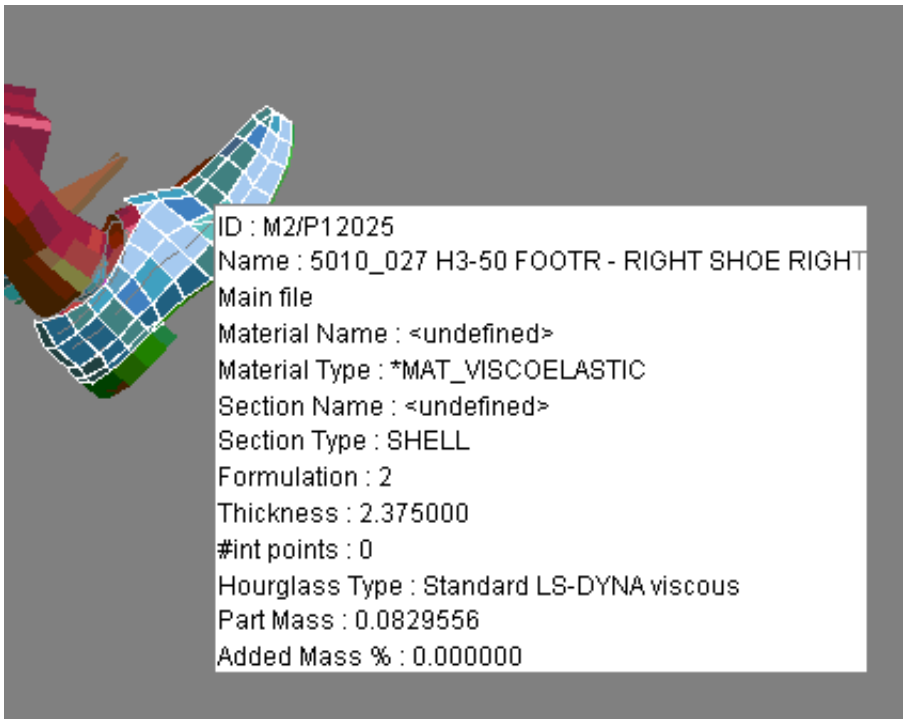
4.6.3 Using the "Quick Pick" Information option.



The "information" option in the quick pick menu gives type-specific data about any item picked.

This includes the label, but also a host of other information.

In the example below the user has clicked on the foot of a dummy to receive information:


ID : M2/P12025
Name : 5010_027 H3-50 FOOTR - RIGHT SHOE RIGHT
Main file
Material Name : <undefined>
Material Type : *MAT_VISCOELASTIC
Section Name : <undefined>
Section Type : SHELL
Formulation : 2
Thickness : 2.375000
#int points : 0
Hourglass Type : Standard LS-DYNA viscous
Part Mass : 0.0829556
Added Mass % : 0.000000

4.7 Predictive Picking and Menu "Hover Over"

"Predictive picking" highlights what would be picked were you to left-click with the mouse.

"Menu Hover Over" highlights items in menu lists, helping you to identify what they are.

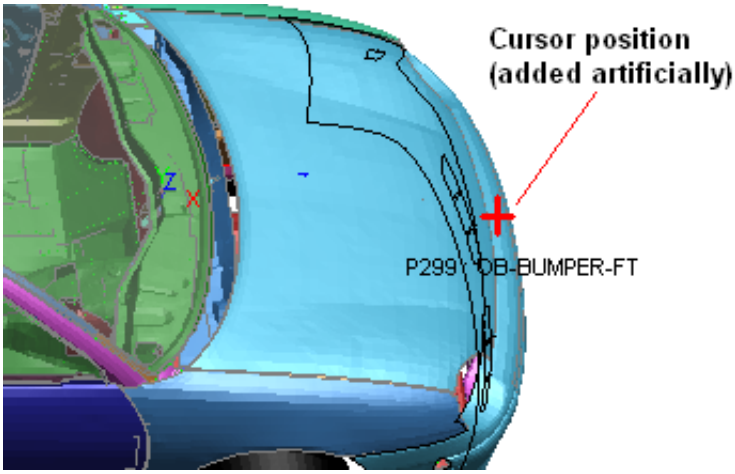
4.7.1 Description of Predictive picking.

From PRIMER 10.0 onwards all screen-picking operations have "predictive picking" enabled by default. This means that when you move the cursor into the graphics window and position it over something pickable in the current context, the item in question will be highlighted by sketching and labelling it, identifying what would be selected were you to perform a left mouse click at that position.

In this example the cursor (red cross added artificially here) has been hovered over the front bumper of a vehicle model.

The current mode is the default "Quick pick by part", so the part making up the bumper has been sketched in free edge mode, and labelled with its id and title, here "P299 OB-BUMPER-FT".

The sketching used to highlight items is transient: it will disappear as you move the cursor away from the object in question, and there is no need to refresh the graphics window to get rid of it.



In the example here the current pick mode was "Quick pick by part". Predictive picking is always associated with the current picking operation, so for example if you chose **[Keyword] Element Shell, Modify** then the current picking mode would be to select a shell, and predictive picking would change to highlighting shells under the cursor.

4.7.2 Controlling Predictive Picking

Most of the time Predictive Picking is helpful, but there are two situations in which you might want to turn it off:

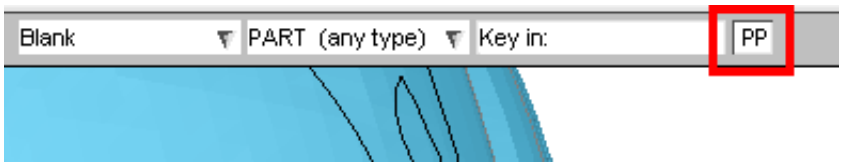
1. If your computer is very slow, or you are displaying graphics over a network, you may find that the need to keep updating the display as the cursor position moves makes the response sluggish.

2. If your image is very complex, and you are picking items which generate a lot of extra graphics when they are highlighted (typically sets, or contacts defined by set) you may find that predictive pick highlighting becomes a nuisance.

In the first situation you might want to turn it off for all picking operations; but in the second you may just want to suppress it for the duration of the current pick operation, turning it back on when you revert to picking items that are less visually complex. Therefore two levels of control are provided:

Switching on/off temporarily for this picking operation only.

The **[PP]** button to the right of the "Quick Pick" selection buttons can be used to toggle predictive picking on/off ***for the current picking operation only.***



As an alternative you can use the "**p**" (note lower case) keyboard short-cut to have exactly the same effect.

This only affects the current picking operation, and the setting is "forgotten" once that operation ends.

Special case of Predictive Picking and contact surfaces

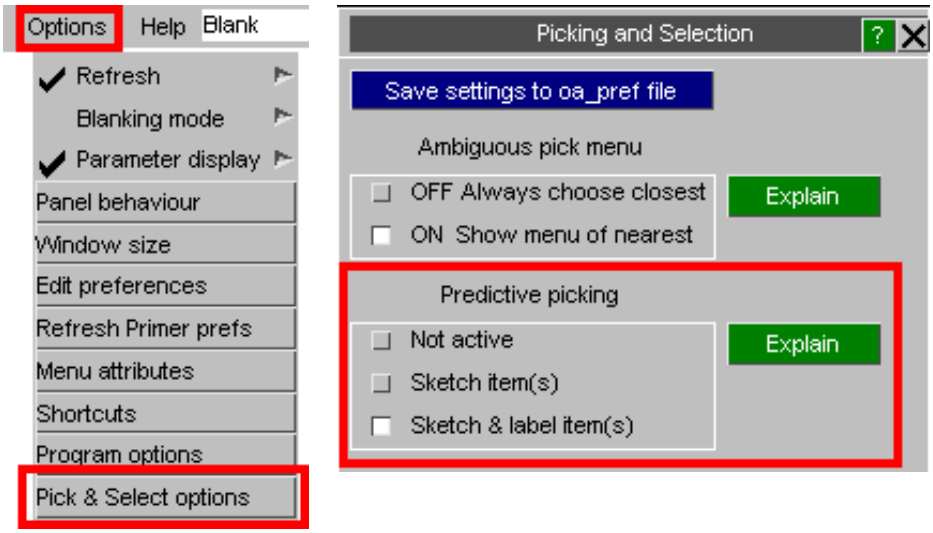
Experience has shown that the combination of Predictive Picking and contact surfaces is not helpful. Most contact surfaces are defined by sets, often sets of many parts, and it is sometimes the case that the whole model will have been placed in a global contact.

As a result Predictive picking of contacts tends to select many items, and if the whole model is in a contact it will always highlight the whole model - which is a hindrance and not a help!

As a consequence there is a special exception in the case of the [Keyword] **Contact** panel, where predictive picking is off by default. This is equivalent to disabling it temporarily by the **[PP]** button or the "**p**" shortcut whenever this panel is entered, and it can be re-enabled by either of these means if desired.

Switching on/off globally, and controlling what is displayed.

[Options]> Pick & Select options maps the **`Picking and selection`** panel:



Here you can choose from three possible modes for Predictive picking:

Not active	Turns Predictive Picking off globally for all picking operations.
Sketch item(s)	The item(s) in question are sketched; usually in free edge mode, but the exact sketching method depends upon what is being displayed.
Sketch and label item(s)	The item(s) are labelled as well as being sketched. Labelling is generally at the item's visual centre. (In this context "visual centre" means its average coordinate, which may not be its true centre of gravity.) Note that you can't have "label only", ie label without also sketching.

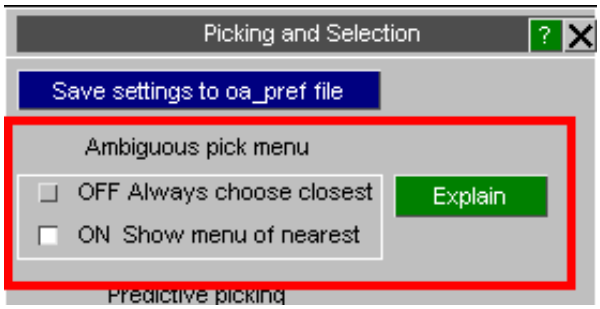
The current setting (along with all others in this panel) can be saved to the oa_pref file using **Save settings to oa_pref file** so that it is remembered for future sessions of PRIMER.

It is also possible to toggle programme-wide predictive picking on/off using the "**P**" (note upper case) keyboard short cut. This is equivalent to selecting Not active, or reverting to the current setting, in the panel above. However it is not "remembered" in any away, so a future PRIMER session will revert to the default behaviour as (possibly) modified in the oa_pref file.

4.7.3 Ambiguity and Predictive Picking

PRIMER has two possible ways of handling ambiguity during screen-picking operations. In the **`Picking and Selection`** panel shown above you can choose whether or not to map the Ambiguous selection menu using the options:

OFF: always choose closest	Picks always select the closest item without any further intervention from the user.
ON: Show menu of nearest	A list of possible candidates sorted by distance from the pick point is shown, and the user is invited to choose which is to be used.



Predictive Picking works with this setting as follows:

- When the ambiguous menu is turned **OFF** Predictive picking will only ever show the closest item.
- When the ambiguous menu is turned **ON** Predictive picking will show all possible candidates if the current cursor position would result in an ambiguous selection.

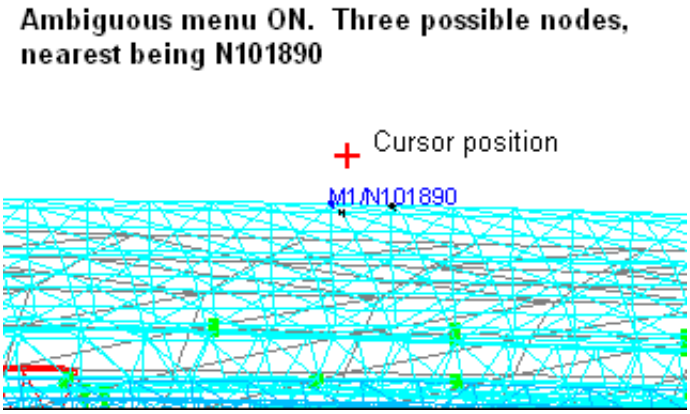
  In this second case only the closest item is labelled (assuming that labelling is active), and it is drawn in colour (yellow or blue depending on the background colour). All other potential candidates are only drawn, and in the current sketch colour (black or white depending on background colour).

This is illustrated in this example.

Here the display mode is LINE (ie no hidden surface removal), we are currently picking Nodes, and the cursor has been positioned just outside the mesh.

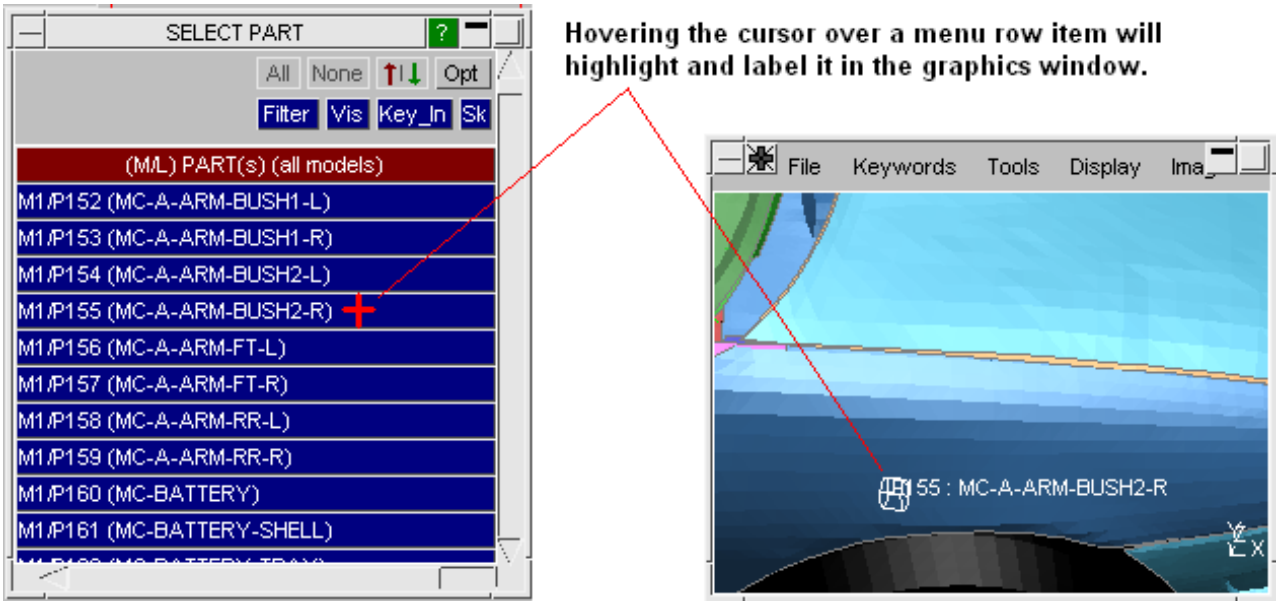
Three possible candidate nodes have been identified and highlighed, but only the nearest (M1/N101890) is labelled.

If the ambiguous menu was OFF then only N101890 would have been sketched and labelled.



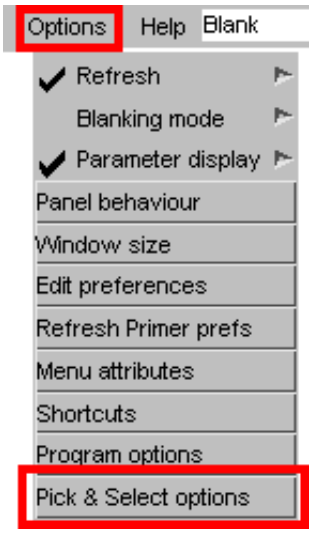
4.7.4 Description of Menu "hover over" highlighting

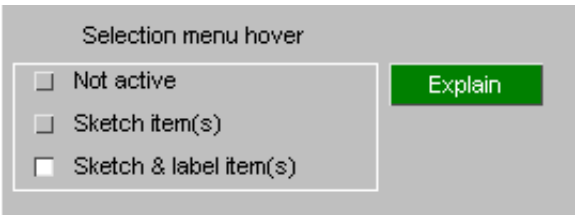
Menu "Hover over" highlighting is very similar to Predictive picking. Whenever PRIMER builds an "object menu" showing a list of items then hovering the cursor over a menu row will highlight and label that item on the screen.



Controlling menu "hover over" highlighting

[Options]> Pick & Select options maps the `Picking and selection` panel:





Here you can choose from three possible modes for menu hover over:

Not active	Turns hover over off
Sketch item(s)	The item(s) in question are sketched; usually in free edge mode, but the exact sketching method depends upon what is being displayed.
Sketch and label item(s)	The item(s) are labelled as well as being sketched. Labelling is generally at the item's visual centre. (In this context "visual centre" means its average coordinate, which may not be its true centre of gravity.) Note that you can't have "label only", ie label without also sketching.

The current setting (along with all others in this panel) can be saved to the oa_pref file using **Save settings to oa_pref file** so that it is remembered for future sessions of PRIMER.

5 Keywords

5.0 Index to keywords

Volumes I and II keywords

*AIRBAG	Airbags (CVs) and interactions
*ALE	Arbitrary Lagrangian Eulerian
*BOUNDARY	Boundary conditions
*CASE	Case membership
*COMMENT	Explicit comments
*CONSTRAINED	Constraints
*CONTACT	Contacts (Sliding Interfaces)
*CONTROL	Control card processing
*DAMPING	Damping cards
*DATABASE	Database output control
*DEFINE	Define cards
*DEFORMABLE_TO_RIGID	Part state switching
*ELEMENT	All element types
*EOS	Equations of State
*HOURGLASS	Hourglass control cards
*INCLUDE	Include files (in section 3.13)
*INITIAL	Initial conditions
*INTEGRATION	Integration rules
*INTERFACE	Interface cards
*LOAD	Applied loading
*MATERIAL	Structural & thermal materials
*NODE	Nodes (grid points)
*NODE_TRANSFORMATION	Node transformations
*PARAMETER	Parameters
*PART	Parts
*PERTURBATION	Perturbation
*RAIL	Rail cards
*RIGIDWALL	Rigid ("stone") walls
*SECTION	Section processing
*SENSOR	Sensor cards
*SET	Set processing of Beam, etc types
*TERMINATION	Termination settings

Volume III keywords

Support for Volume III keywords is available in version 12 of PRIMER. Currently PRIMER can read, copy, renumber and write the Volume III keywords but support for creating and editing them is limited. This will be improved over future releases as the keywords become more widely used.

***CESE**	Conservation Element/Solution Element (CESE) compressible fluid solver
***CHEMISTRY**	Chemistry databases
***EM**	Electromagnetism
***ICFD**	Incompressible fluid flow solver
***MESH**	Mesh generation of Vol III solvers
***STOCHASTIC**	Particles and numerical details for solving a set of stochastic PDEs
***LSO**	Data output for LSDA files

5.1 Keywords

PRIMER allows you to create, modify, list and delete the constituent parts of an input deck, and this is done by object category, ie "Keyword".

Keywords are chosen from the **Keywords** menu which contains all major keywords.
Most keywords have sub-categories, the indicator that a popup menu can be invoked is:

Once an item has been selected the top-level menu panel for that item will be invoked: see section 5.1.1 below.

There is no limit to the number of different keyword manipulation windows that can be current at any time: for example you can edit concurrently as many PARTs as you like.

Keywords present in the LS-DYNA manual but not in PRIMER's Keyword panel cannot be viewed or edited in PRIMER. However, they are read in, stored within PRIMER and written out, so no valid LS-DYNA data is lost. Further, the implications of actions such as renumbering or deleting parts of a model will be correctly applied to those "hidden" keywords.

5.1.0 Volume III keywords

By default PRIMER shows the keywords from Volumes I and II of the keyword manual (as shown in the above image). You can toggle between viewing the **Volumes I & II** keywords and the **Volume III** keywords by pressing the buttons above the list of keywords.

The volume III keywords are grouped together by solver. For example, there are keywords **`*CESE_CONTROL`**, **`*EM_CONTROL`** and **`*ICFD_CONTROL`**.
When showing Volume III keywords PRIMER will show the keywords for a single "main" keyword category. For example, the image on the right shows the keywords for **`*CESE`** (**`*CESE_BOUNDARY`**, **`*CESE_CONTROL`**, **`*CESE_EOS`** etc).

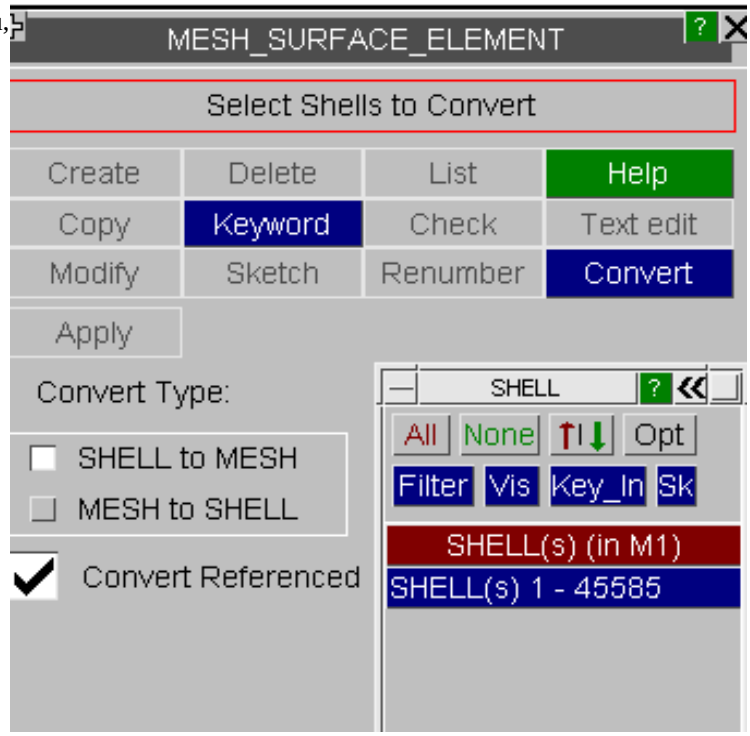
To show a different Volume III keyword press the appropriate keyword button. For example if **ICFD** is pressed PRIMER then shows the keywords for *ICFD (**`*ICFD_BOUNDARY`**, **`*ICFD_CONTROL`**, **`*ICFD_DATABASE`** etc).

The other keywords are available at any time by using the popups. For example the **`*EM`** keywords are still available in the image on the right by right clicking on **EM**.

Inside MESH_SURFACE_ELEMENT dock menu, specialized **CONVERT** Button is provided to Convert ELEMENT_SHELL to MESH_SURFACE_ELEMENT and vice-versa.

The Function also converts the referenced PARTs and NODEs to MESH_PARTs and MESH_NODEs and vice-versa.

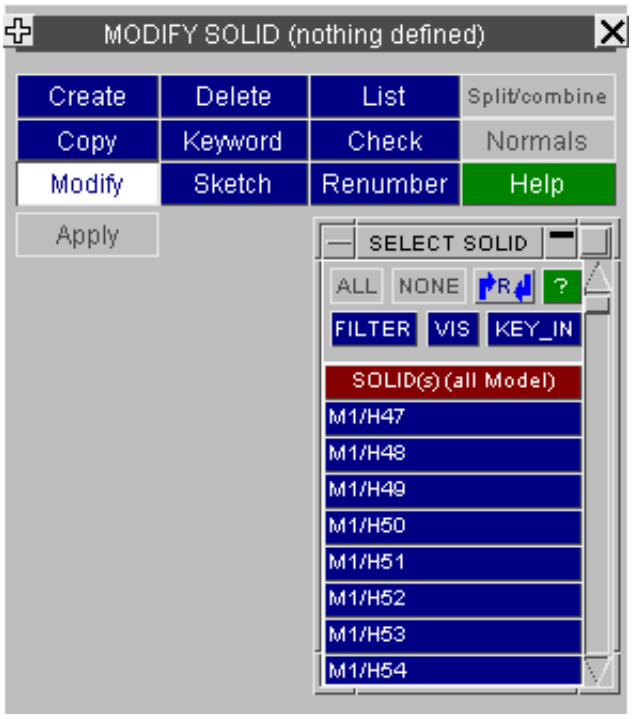
The converted Elements/Mesh Elements and referenced Parts/Mesh Parts and Nodes/Mesh Nodes can be deleted via the Delete box.



5.1.1 Standard Keyword top level menu options.

Most of the **KEYWORD** options have a set of standard options. The exact contents may vary slightly with context, but the basic functionality is the same in all cases.

This figure shows a standard display, here for **ELEMENT_SOLID** keywords, but it is the same for any item type. Currently Modify is the selected option. Selecting any of the other buttons will switch to that option.
These "standard operations" options are:



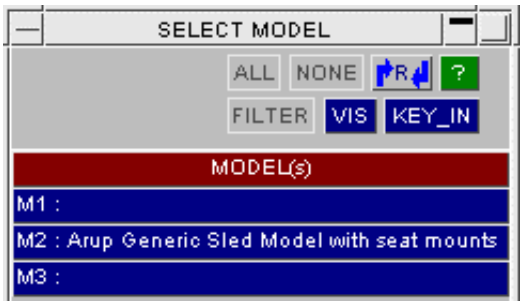
Create	Create a new item
Copy	Copy existing items
Modify	Edit (modify) an existing item.
Delete	Delete one or more existing items.
Keyword	Invoke the generic keyword editor
Sketch	Sketch existing items on top of the current image.
List	List a summary of the contents of existing items.
Check	Check items for errors.
Renumber	Change the labels of items.

Not all options will be available in all contexts, for example **Create** and **Modify** will not be available where specific editing/creation functions do not yet exist in PRIMER, and not all types have a **Keyword** editor.

Operations requiring an explicit "Parent" model id.
If you have more than one model in memory, and the operation in question requires an explicit "parent" model id, then you will be forced to select a model prior to the operation taking place.

For example when you **CREATE** an element it must exist in one model only so, in the multiple model case, you will be asked to select which model to use.

If only one model is current it will be used automatically and this selection stage will be skipped

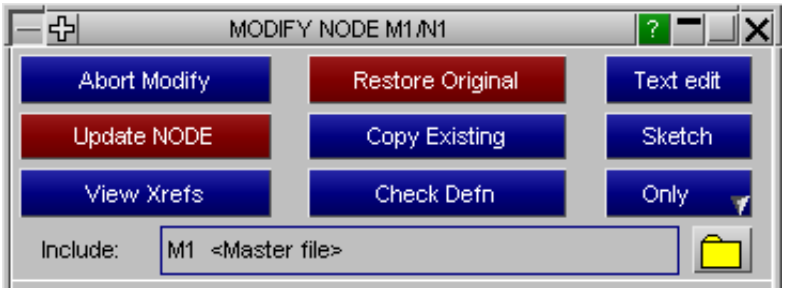


.

5.1.2 Standard "static" header for CREATE and EDIT functions

The Create/Edit panels for Keywords differ in detail, but share a common layout of the "static" functions at their top.

This example is taken from a Part editing panel, but the top buttons are the same in all cases.

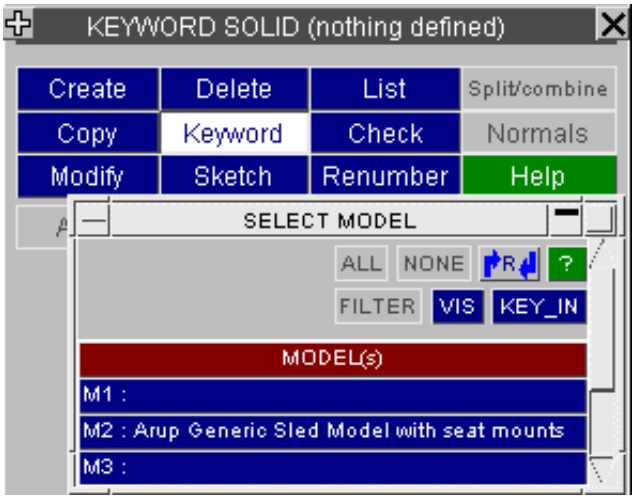


The standard buttons act as follows. Note that some have vary between "Create" and "Edit" modes, whereas others are common to "Both":

Abort_item	(Both)	Terminates the current operation, leaving the permanent definition unchanged (edit case) or undefined (create case).
Reset All **Restore Original**	(Create) (Edit)	Resets the definition to null, canceling any entries made so far. Restores the original, unedited definition (copied from the permanent definition), overwriting any changes made so far.
Text edit	(Both)	Writes a "mini keyword output" file of just this keyword, and opens the system standard text editor on this file. See Text Edit below for more information. This will also also display any "keyword comments" associated with this keyword as hover text when the mouse is hovered over the button. You can tell if any comments are present since the **Text Edit** button will be drawn in White on Light blue instead of the standard white on dark blue. Text edit Edit keyword in external text editor. Comments on this keyword (can also be edited via text editor): $ Added initial comment line <= Keyword comment
Create item **Update item**	(Create) (Edit)	Creates a permanent entry from the scratch definition. Overwrites the "old" permanent definition with the revised entries from the scratch definition.
Copy existing	(Both)	Copies entries from an existing definition into this one, superseding any entries or changes made so far.
Sketch	(Both)	Sketches the currently defined scratch definition on top of the current graphics image.
View Xrefs	(Both)	Invokes cross-reference viewer.
Check Defn	(Both)	Checks the current scratch definition for errors
Find/Only/ Blank/Unblank	(Both)	Changeable between these four options by drop-down. In the Only case the view will display only the entity whose editing panel we are looking at. In the Find case it brings up the Find panel as in Tools -> Find.

Remember: All Create and Edit functions ***always*** take place on a "scratch" copy of the current definition (if any). No changes are made to the permanent database until the **CREATE** or **UPDATE** buttons are used.

5.1.3 The generic KEYWORD editing panel.



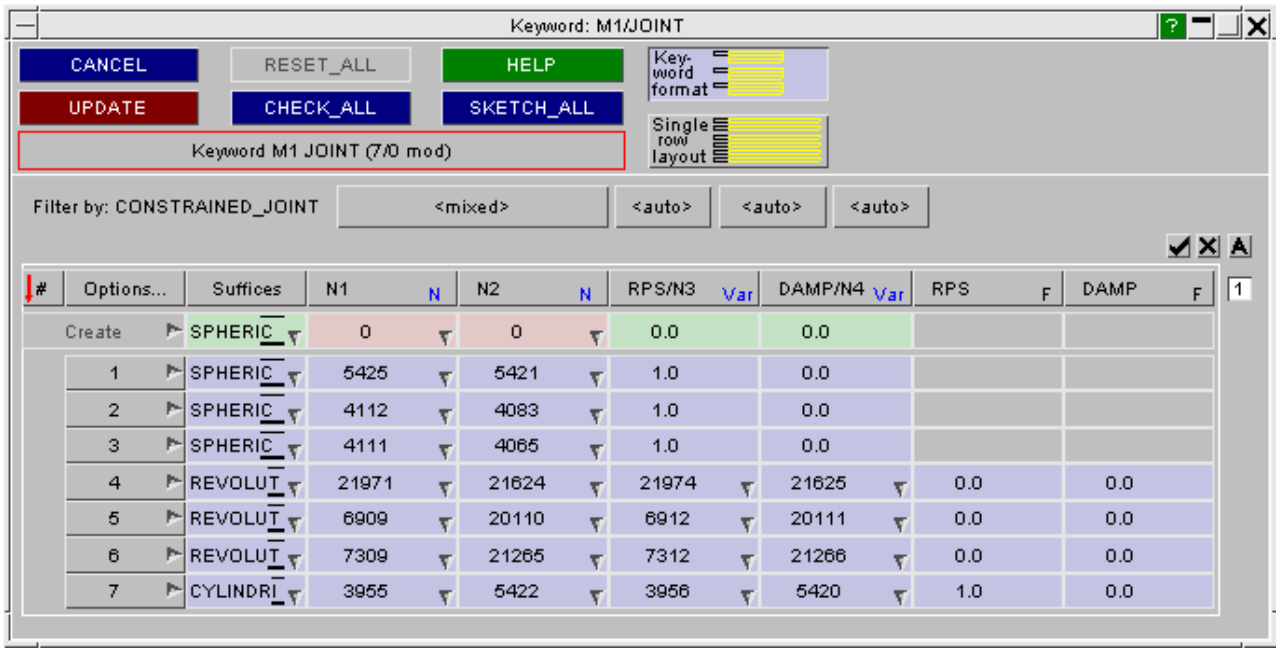
There are many places in PRIMER where an explicit create/edit panel is not necessary, and a generic "keyword editor" will suffice. The "Keyword editor" has the additional advantage that it lists all items of a particular type, allowing multiple edits to be carried out with a single command.

This is invoked by the **KEYWORD** tab, as shown here, from the standard options available once a selections has been made from the Keywords panel.

The Keyword Editor has been completely rewritten for PRIMER release 9.3 to improve its functionality and make it easier to use. In particular:

- The separate **`Edit`**, **`Insert`** and **`Delete`** modes of the original version have been superseded, with all this functionality now available in a single operating mode.
- The limitation that only a single keyword sub-type could be displayed at a time has gone: it is now possible to display entities of all sub-types of a master keyword together, combining any permutation of sub-keyword suffices.
- Following from this it is now possible to insert, remove or change the sub-keyword suffices for the entities shown.
- The display and functionality of the editor have both been improved, including sorting by column and "intelligent" editing of data fields over multiple entries.

The following examples use the **`*CONSTRAINED_JOINT`** keyword to illustrate the new editor. In this model there are a variety of different joint types, and the figure below shows **`_SPHERICAL`**, **`_REVOLUTE`** and **`_CYLINDRICAL`** types simultaneously.



Terminology

Various areas of this panel are referred to below:

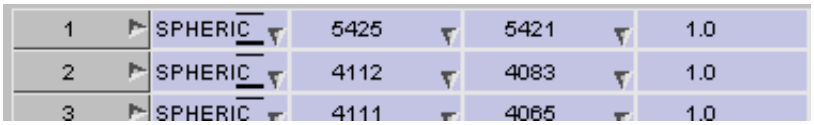
The Acronym row: One or more rows of headers showing the acronyms for each field.

Hovering the mouse over a button will give more information about the data. Clicking on a button (other than **Options...**) will sort the data rows by that column.

The Entry row: One or more rows on a green background to enter new data.

This row will initially be blank: you must type in or select data to populate it, then use **Create** to create the definition and store it in the database.

The Data rows: Rows of existing data on a blue background.



Entries in these fields can be edited by over-typing them or by selecting new values from popup selection menus.

Multiple rows may be edited simultaneously by selecting the rows and then changing the required data field on any row to propagate its change to all other selected rows.

Use of button background colours

The keyword editor uses a variation on the standard button colour scheme used elsewhere in PRIMER in order to distinguish between inactive and active / selected.

In all cases the colours are dark text on a light background when either the data field or the whole row are not active, and invert to become light text on a dark background (the standard PRIMER colours) when they become active.

Green is used on the **Entry row** to denote data fields that are grammatically correct in their current state	Inactive	Active data entry
Blue is used on the **Data rows** to denote data fields that are grammatically correct in their current state	Inactive	Active data entry
Red is used on all rows to denote a field that is either invalid, or empty and requires population with data	Inactive	Active data entry
Cyan is used on all rows to denote references to latent (referenced but undefined) items	Inactive	Active data entry

Filter by: controlling what is shown in the panel.

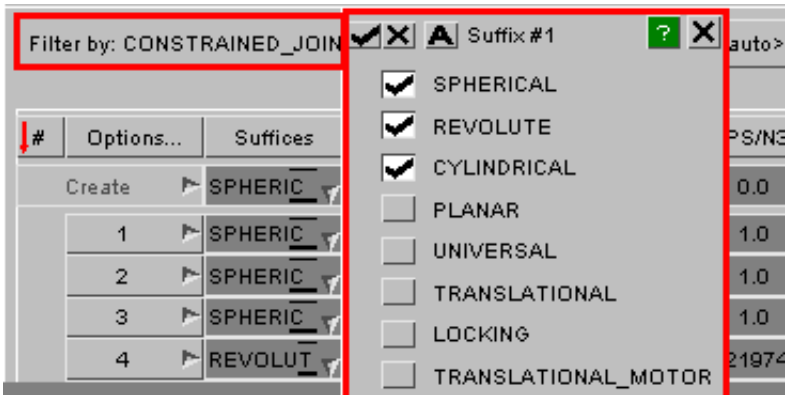
`*CONSTRAINED_JOINT` has four possible sub-keywords:

1: Joint type	2: Force output option	3: Failure option	4: Label & title option
`_SPHERICAL` **`_REVOLUTE`** etc	*<none>* **`_LOCAL`**	*<none>* **`_FAILURE`**	*<none>* **`_ID`**

To see the keyword suffices for each of the buttons above hover the mouse over them, and a popup window will display the relevant options.

To control what is actually displayed in the editor click on the appropriate column and select the suffices to be shown.

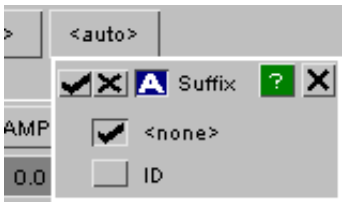
By default the **`[A]`** option, for "Automatic" will be selected for all suffices, causing all sub-types of this keyword in the model to be selected automatically for display.



This model contains the first three joint types, and also some **`_TRANSLATIONAL`** ones; but the last of these has been deselected meaning that these joints are not shown in the image above.

Many sub-keywords are optional, for example **`_ID`** in this context, and the alternative is for that sub-keyword to be omitted altogether.

In this situation you will be given the choice of that keyword or **`<none>`** as shown in the popup here for the **`_ID`** column.



This process of selection may be carried out for all sub-keyword columns, and what is shown in the editor rows below is the logical AND of the selected keyword suffices.

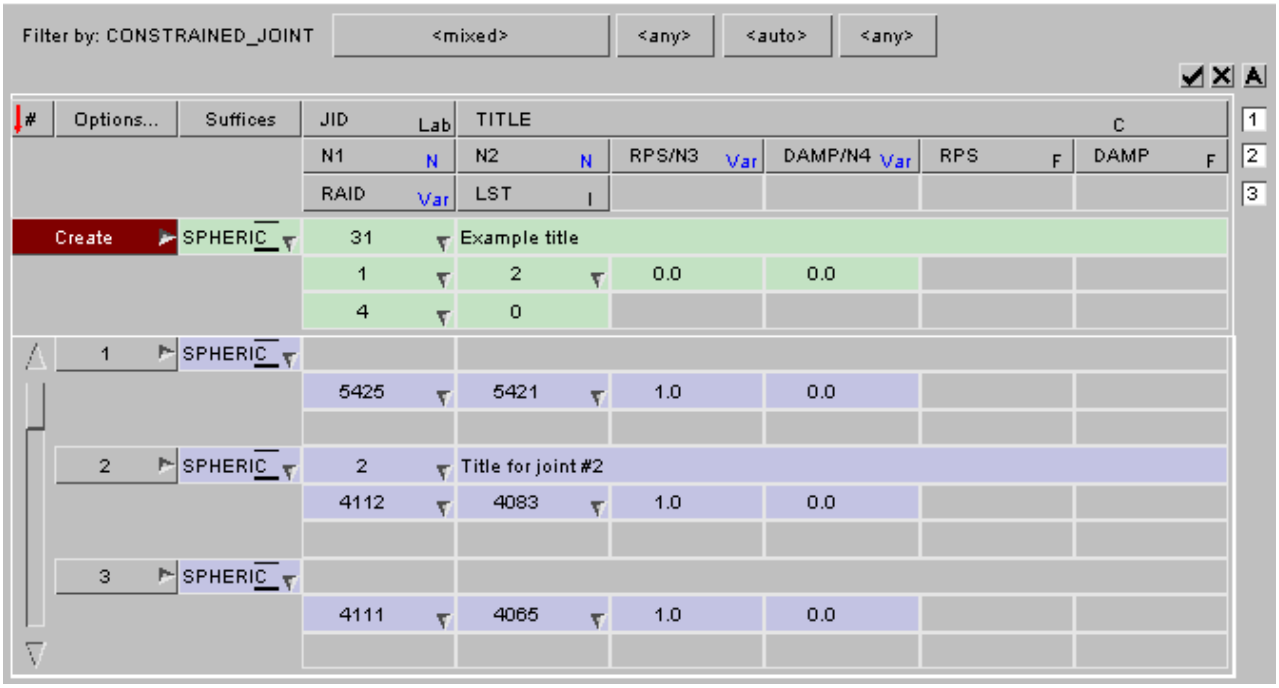
You can change what is shown at any time, and the effect is only to change what is shown in the editor rows below. No change is made to the actual keyword definitions themselves.

Displaying data for different sub-types

The display options above make it possible to display entries requiring different numbers of rows, and for row/column fields to have different contents and data types - or even to be absent.

The example above has been modified so that:

- Both **_LOCAL** and **_ID** options are displayed
- The top entry (green) row has been given notional values
- The second existing joint (blue row 2) has been given the **_ID** suffix and an explicit label and title



This presents several display problems:

- The **_ID** row (**JID** and **TITLE**) row is undefined for existing joints #1 and #3
- The **_LOCAL** row (**RAID** and **LST**) is undefined for all existing joints.

It can be seen from the figure above that this problem is solved by simply "greying out" the data fields that are not relevant.

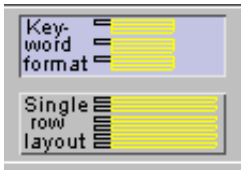
Changing the display layout

As the example demonstrates shows turning on lots of suffix options can results in the table rows expanding and containing a lot of empty grey rows as a consequence. It is often the case that the data fields of interest are on the early rows of a keyword, for example Joint nodes appear on the first line, and while it may be necessary to display lots of suffices in order to see all variants one is only interested in a few rows of data.

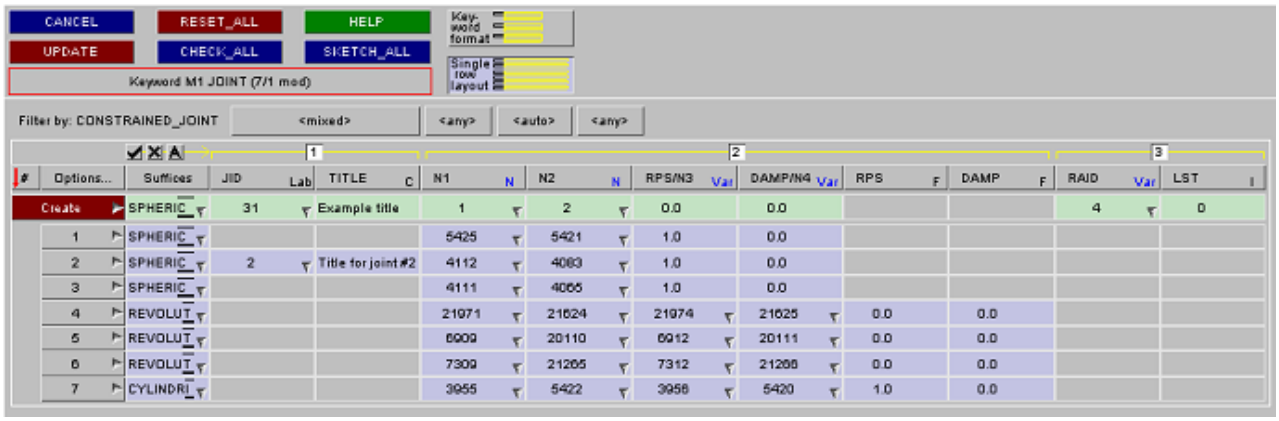
One solution to the problem is to change the layout of the editor rows:

Keyword Format is the default, shown in the figures above, that mimics the LS-DYNA keyword row and column layout.

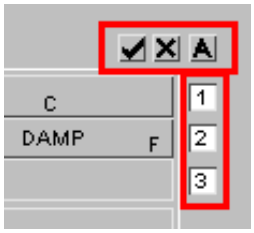
Single Row Layout is the alternative which condenses each item onto a single line by concatenating the rows.



The image here, reduced in size to fit the page, shows how the three rows have been concatenated so that each item only uses a single line. Empty fields are still greyed out, but they no longer affect what is displayed so severely. Note also that "wide" fields such as **`TITLE`** (column #2) are compressed into a standard with column in this display mode.

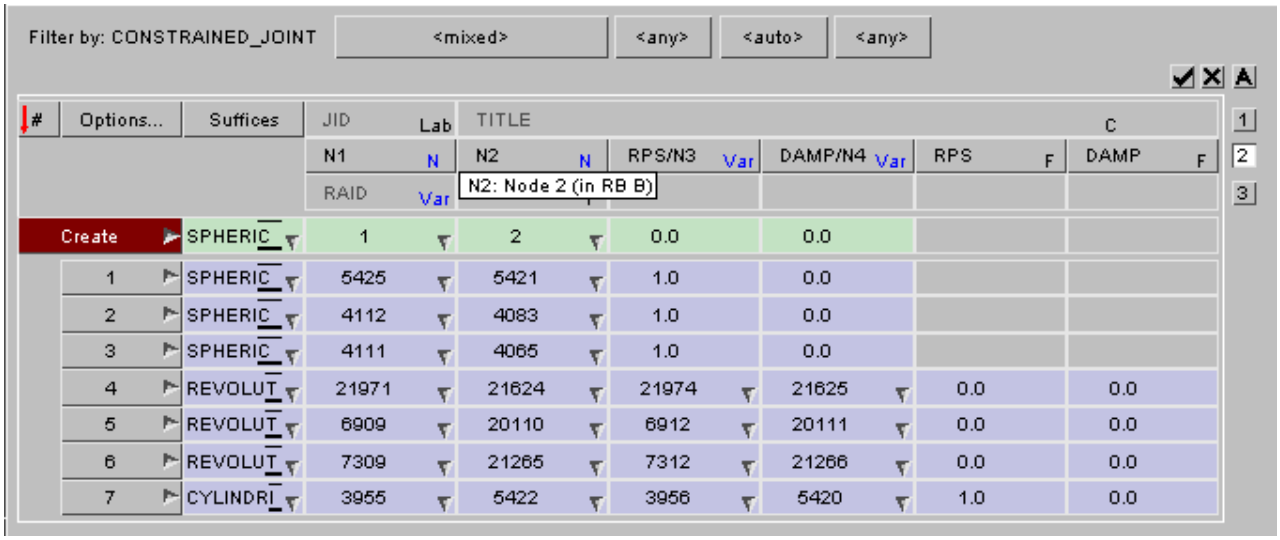


Limiting the rows displayed



An alternative solution to the "too much empty space" problem is to turn off the display of selected rows. In Keyword Format there is a numbered button to the right of each acronym row at the top of the panel, and this may be used to de-select a given row for display in the table below.

In the example below rows #1 (**`ID`** and **`TITLE`**) and #3 (**`RAID`** and **`LST`**) have been turned off leaving only row #2 containing nodes etc.



This demonstrates that the "acronym" rows are still present, albeit greyed out, but that only the selected data rows are shown thus condensing the display.

The **[A]** button, "Automatic", will turn back on all rows containing active data.

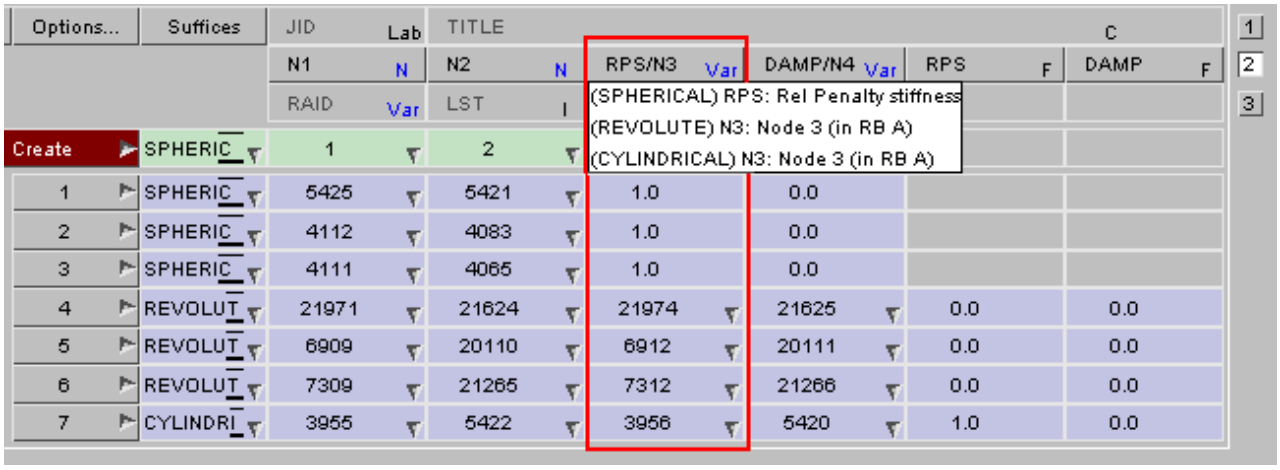
These row selection buttons are also available in **Single Row Layout** format where they work in the same way by eliminating the columns of the selected rows.

Mismatched data in row/column fields.

In the examples above we have:

Entity type	1	2	3	4	5	6
SPHERICAL joints, data fields	**N1**	**N2**	**RPS**	**DAMP**		
REVOLUTE joints, data fields	**N1**	**N2**	**N3**	**N4**	**RPS**	**DAMP**
CYLINDRICAL joints, data fields	**N1**	**N2**	**N3**	**N4**	**RPS**	**DAMP**

Therefore column #3 may contain **RPS** or **N3**, and column #4 may contain **DAMP** or **N4**. This is handled as follows:

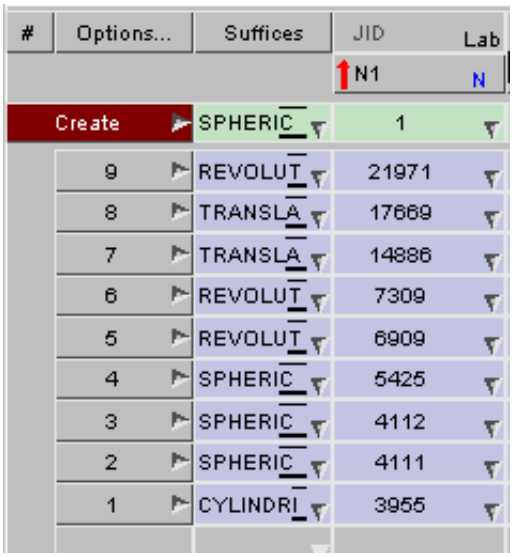


The acronym header button shows the various entries, here **RPS / N3**. In addition hovering the mouse over that button, as shown here, displays the details of that row/column contents by suffix type.

The data rows contain the relevant data. Here the first three entries for **SPHERICAL** joints contain an **RPS** value of 1.0, and the remaining rows show **N3** node values.

The keyword editor always "knows" the type of the data in a given field, and processes it accordingly. This is significant when multiple rows are edited as described below.

Sorting rows by data field

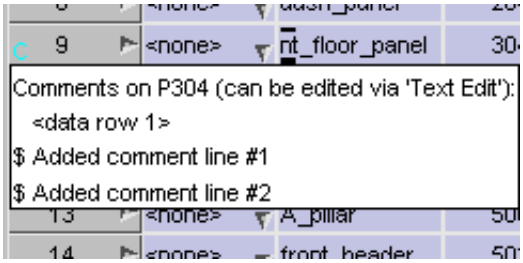


By default editor rows are sorted in ascending order by index (column # at the top left) but it is possible to sort using any data field by clicking on its acronym button. The first click on a button will sort by ascending order and subsequent clicks will reverse the sort order, the current direction being shown by a small red arrow on the button.

In this example rows have been sorted in descending order by node **N1**.

To revert to the conventional "sort by index" click on the **[#]** button at the top left.

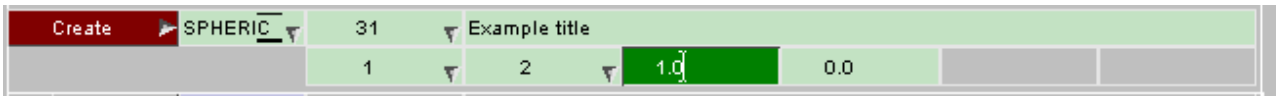
Comments on keywords



If a keyword contains embedded comments a light blue C will be shown on its row button, and hovering the cursor over that button will list the comments on that keyword.

Comments can be added, modified and deleted using the **Text edit** capability.

Creating a new definition in the Entry row

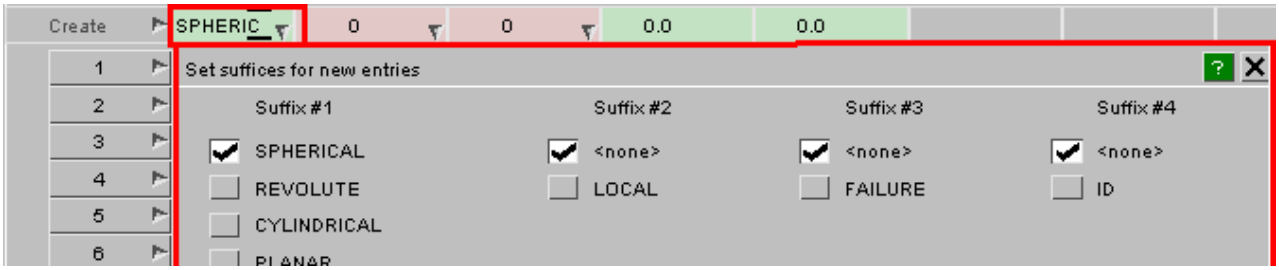


The green row at the top of the list of existing items is where data is inserted to create a new definition.

- Type in, or select from popup menus, sufficient data fields to populate the definition.
- Use **Create** to make the definition. This installs it in the database, and it will also appear in the **Data rows** below.

Changing the Suffix of the Entry row definition.

To change one or more keyword suffices right click on the field in the **Suffix** column (here `[SPHERIC v]`), and choose the revised suffices. (The image below has been truncated vertically.)

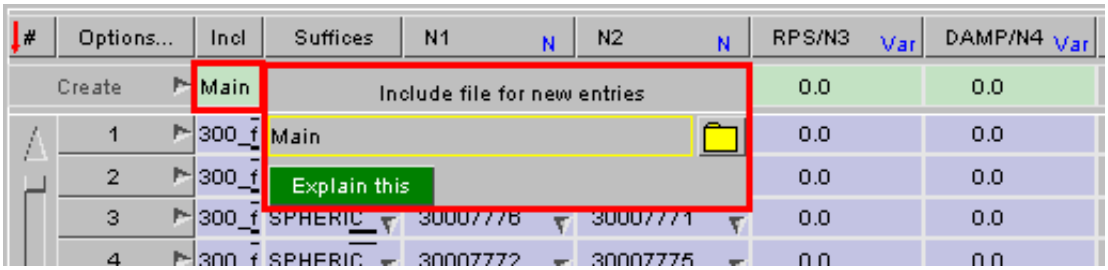


In this example there are four columns of suffices to choose from, other keywords will be different. You can only tick one entry in each column since, obviously, entries within a column are mutually exclusive.

In most cases column suffices are independent but there are a few cases where changing one suffix may affect what is legal in other columns. One example is `*RIGIDWALL` where further suffices on the `_PLANAR` suffix are not compatible with the `_GEOMETRIC` suffix. In these situations any illegal combinations will be removed, and the relevant buttons greyed out.

Setting the Include file for the new entry

If include files are present in the model then there will be an extra **Incl** column between **Options...** and **Suffices**, and each entry in the table will show its include file name.



By clicking on the entry (here the `Main` file) a small sub-panel for selection of an alternative include file will be mapped.

The Keyword editor will always initialise itself to create new entries in the current include file for this model, but if you change this then subsequent new entries will be in the selected file. Include files in PRIMER are described in more detail in section 3.13

Using **Create** to make the entry.

The **Create** button will be one of three colours:

Greyed out	This means that the row fails the "grammatical" check and the entry cannot be created. One or more the data fields will be red and must be populated or corrected.
Red	This means that the row passes the "grammatical" check, ie no red data fields, but that the standard "Check" function has found one or more errors. The **Check** option on its popup menu can be used to list these errors. The entry can still be created, but you will be warned about the errors and may have to correct them later.
Green	The row passes both grammar and contents check, and can be created with no error or warning messages.

The popup options on the **Create** button

These options may be use to manipulate the **Entry row** as follows:

Create...	Maps the standard Create/Edit panel for this item. When you exit from this the saved definition will be used to populate the **Entry row**. (This option will be greyed out if a create/edit function has not been written for the current data type.)
Check	Runs the standard check function on this definition and reports any errors.
Sketch	Sketches the definition in its current form on the model
Reset	Resets the **Entry row** to its default (empty) state.

Changing an existing definition in the Data rows

1	SPHERIC	5425	5421	1.0	0.0		
2	SPHERIC	4112	4083	1.0	0.0		
3	SPHERIC	4111	4065	1.0	0.0		
4	REVOLUT	21971	21624	21974	21625	0.0	0.0
5	REVOLUT	6909	20110	6912	20111	0.0	0.0

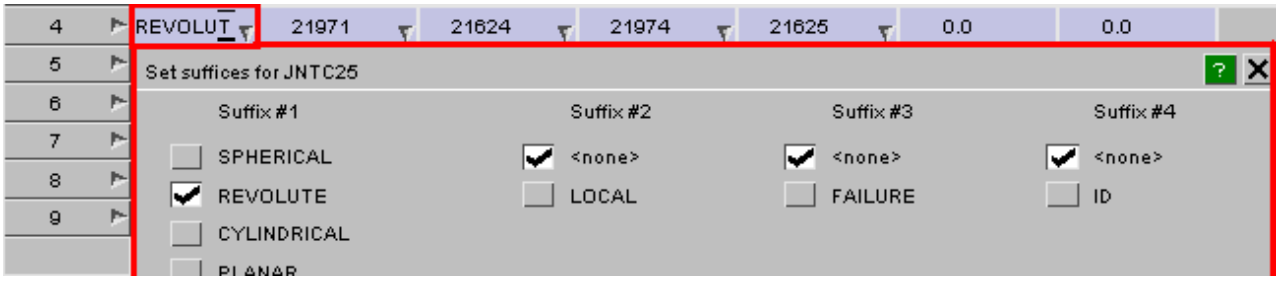
Simply overtype the existing entry, or select a new value from the standard popup selection menu.

Changing a definition in the **Data rows** does two things:

- A backup of the existing definition (before the change) is made, so that the original definition can be restored if required.
- The current definition in the database is changed immediately

Changing the Suffix of a Data row definition.

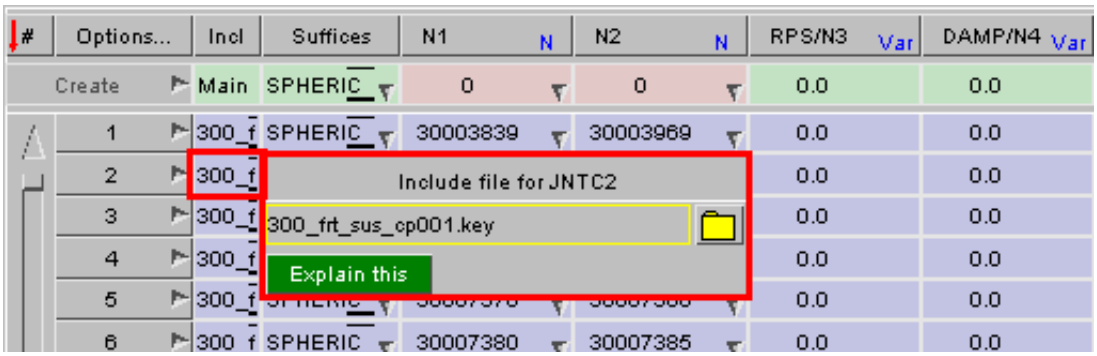
To change one or more keyword suffices right click on the field in the **Suffix** column (here `[REVOLUT v]`), and choose the revised suffices. (The image below has been truncated vertically.)



This popup and its usage are identical to that described under the **Entry Row** above.

Changing the Include file of a Data row definition

If include files are present in the model then an extra **Incl** column will be shown, and the name of each entry's include file will be listed. This is truncated to a narrow column width to save space, but if you hover the mouse over an entry the full include file details will be given.



To change an entry's include file click on its data field, and the selection popup shown above will be mapped allowing you to select a different include file.

Warning: Include file changes are ***not*** undone by a **Reset** command

Because of the way include file membership is handled in PRIMER moving a **Data row** entry to a new include file is not reversible by a **Reset** command. The only way to revert to its original include file is to reset it explicitly using the <click> + <select new> process above.

The popup options on Data row Index buttons.

Each **Data row** has the following popup menu options:

Edit... Maps the standard Create/Update panel for the current definition. When the edits are saved the Data row will be updated.

Check Runs the standard check function on this Data row

Reset Resets this Data row back to its original condition (before any edits, not just the most recent one.)

Xrefs Maps the standard cross-reference viewer panel for this item.

Sketch Sketches this item on the current model.

Manipulating blanking using these options

These three commands act immediately, there is no need to update the display to see the changes.

Blank Blanks this item from the current display.

Unblank Unblanks this item in the current display.

Only Makes this the only item visible in the current display

Text edit: editing definitions in an external editor

The external text editor works in exactly the same as for scalar editing panels (see description above) in that it performs "mini keyword output" operation to write a keyword file containing data for the selected row(s), and then performs a "mini keyword input" to read the file back in again and to update the model.

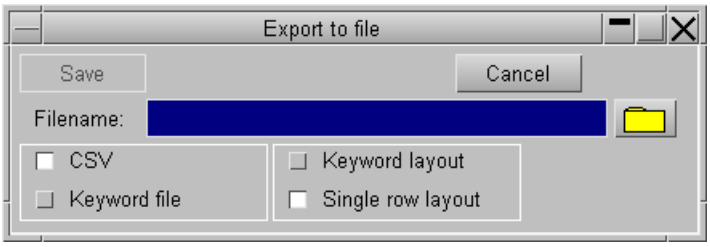
Because the keyword editor permits operations on multiple items Text edit in this context does the same, reading in all definitions found in the edited file that match the current keyword type. Definitions with the same label as existing items replace these, and items with new labels are added to the model.

Using **Text Edit** to import new keywords

From V10.1 onwards **Text Edit** in the context of the keyword editor may also be used to import new definitions of the type being edited, as well as modifying existing ones. Therefore it is legal to edit a given row (or rows), and within the external editor to create or import new definitions of this keyword type. On exit from the external editor the new items will be imported into PRIMER exactly as if they had been read by conventional keyword input, and added to the current model using the current include file. The new items are added to the permanent model database, and will appear in the current keyword editor panel immediately, so long as they are not filtered out by current (sub-)keyword selection.

The addition of new keywords in this way is not reversible: a **Reset** in the keyword editor, or an **Abort** of the editor panel, will ***not*** remove these items; so if you change your mind about importing them you will need to **Delete** them explicitly.

Export file: writing keyword definitions as CSV or keyword file



The button **Export file** will open a menu to select a file which the data is written to in either CSV format or keyword format. In the CSV mode you can select the layout how the data is written. This is similar to the layout of the main editor: In keyword layout line breaks are written in the same way as in keyword files. When in single row layout, every keyword definition is written on just one line.

Delete: Deleting the current definition

Using **Delete** will map a cut-down deletion confirmation menu for this item. If you choose to **Delete Items** the standard PRIMER deletion confirmation dialogue will be mapped and the item will be deleted.

Deletion is not reversible - once an item has been deleted it cannot be recovered.

Working with multiple rows

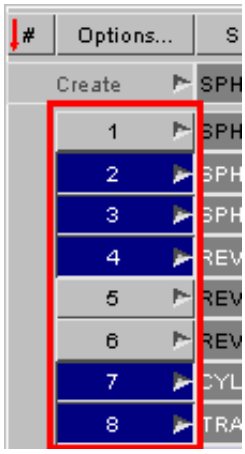
It is possible to select a range of **Data rows** and to change their properties, or their suffices, or to delete them in a single operation as described above for a single row.

Rows that have been selected invert their colour to a dark background, and become active for "multiple" operations. In the figure below rows 2 to 4, and 7 to 8 have been selected.

#	Options...	Suffices	N1 N	N2 N	RPS/N3 Var	DAMP/N4 Var	RPS/N5 Var	DAMP/N6 Var
	Create	SPHERIC	0	0	0.0	0.0		
	1	SPHERIC	5425	5421	1.0	0.0		
	2	SPHERIC	4112	4083	1.0	0.0		
	3	SPHERIC	4111	4065	1.0	0.0		
	4	REVOLUT	21971	21624	21974	21625	0.0	0.0
	5	REVOLUT	6909	20110	6912	20111	0.0	0.0
	6	REVOLUT	7309	21265	7312	21266	0.0	0.0
	7	CYLINDRI	3955	5422	3956	5420	1.0	0.0
	8	TRANSLA	14886	14889	14887	14890	14888	14891

Selecting a range of rows

Rows may be selected by a range of methods, which may be combined in any order:



By clicking on the row index buttons:

`<Click>`	Selects that row only, deselecting any others.
`<Shift + click>`	Selects all rows between the most recently clicked on and the current row.
`<Control + click>`	Inverts the selection status of the current row, leaving other rows' selection status unchanged.

From the Popup menus on the **Options ...** button

SEL_ALL	Selects all rows
UNSEL_ALL	Deselects all rows
Select ...	Maps the standard PRIMER object menu allowing you to select items in the normal way.

Actions on selected rows

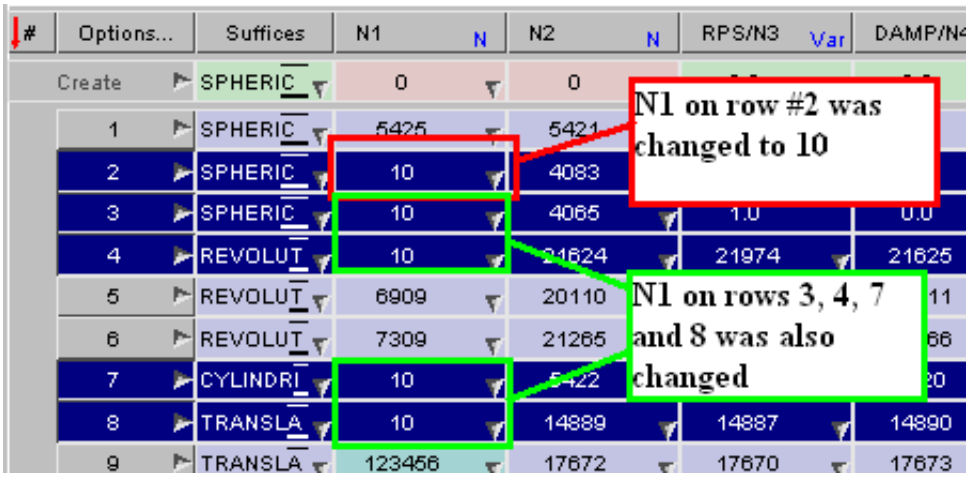
Show_All	Shows all Data rows. Needed if only a subset has been displayed using the options below.
Only_Sel	Shows only those **Data rows** which have been selected. This can be useful if you have selected a small and diverse subset of a large number of items
Hide_Sel	The opposite of the above: shows only those **Data rows** which have ***not*** been selected.
Sketch_Sel	Sketches the currently selected **Data rows** on the current model
Reset_Sel	Peforms a **RESET** of all selected **Data rows**, restoring them to their original unedited state.
Delete_sel	Deletes the selected **Data rows**, going through the same selection and confirmation procedures described above for deleting a single row.

Editing entries on multiple rows.

When multiple rows have been selected then editing any field on any selected row will result in the same field on all other selected rows, if compatible, being changed to the same value.

For example taking the image above, if field N1 on row 2 is changed to 10 (ie node 10), then N1 on rows 3, 4, 7 and 8 will also be changed.

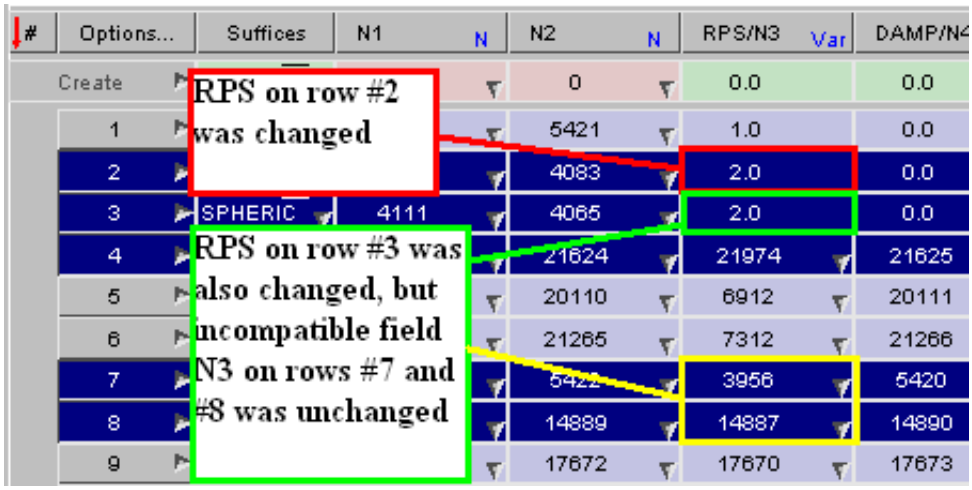
When you type in a formula as explained in Section 2.11, then the formula will be evaluated on each row separately. Typing =n2-1 into the field for N1 on an arbitrary selected row as depicted on the right will set N1 to 4082 on row 2, to 4064 on row 3, to 24623 on row 4 etc.



Only "compatible" data are changed

If the data in the field that is changed does not match that in the same field on another selected row, then the latter is unchanged.

In this example RPS on row #2 matches RPS on row #3, but not N3 on rows #7 and #8, so only row #3 is changed.



Popup menu actions on multiple rows.

When multiple rows are selected the popup menu on any index button works in exactly the same way as for a single row, except as described below.

The following two options act on **this row only**

Edit... Maps the Create/Update panel for the selected item only.

From V12 onwards any edits made will be listed when the edit finishes, and you will be given the opportunity to choose whether or not to propagate them to other highlighted rows.

Xrefs Maps the standard cross-reference viewer for the selected item only.

The remaining actions operate **on all selected rows**

Check Runs the standard check function

Reset Resets to original unedited state

Sketch Sketches on the current model.

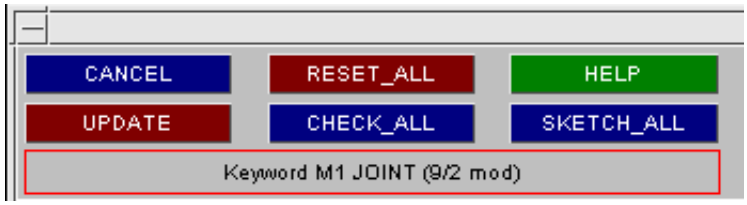
Blank Blanks them

Unblank Unblanks them

Only Draws only them

Delete Deletes them

Saving and discarding changes



Changes made to **Data rows** in the keyword editor update the current database definitions immediately, but these changes only become permanent if and when you **UPDATE** to end an editing session. However any new entries that you have **CREATEd** will remain in the database regardless of how you exit the editor.

In more detail:

CANCEL	Undoes all edits, and exits the editor leaving all original definitions unchanged
UPDATE	Exits the editor making all changes permanent
RESET_ALL	Undoes all edits (equivalent to a **Reset** on every modified **Data Row**), returning all rows to their original state.
CHECK_ALL	Runs the standard checking function on all **Data rows** and reports the results.
HELP	Provides a text summary of how the keyword editor works.
SKETCH_ALL	Sketches all **Data rows** on the current model

General rules in the Keyword editor:

- Only one instance of a Keyword editor may be active at any one time on a given Model/Keyword combination. This is because changes made act upon the true definition in the database, not a scratch copy, therefore multiple instances would permit changes to conflict with one another.

  A Keyword editor may be used as the output of another command, for example to list results of a **Check**, or from the **Quick Pick** selection menu. Such usage counts as an "instance", and will also prevent a second panel being mapped.

- If you open a separate editing panel on an item in the **Data rows** outside the keyword editor (eg by **Keyword, *<item type>*, Modify**), make changes in that panel and then save it, the current row in the editor will only be updated the next time it is drawn. There is no interlock between these two methods of editing, and in particular the keyword editor does not "know" that one of its **Data rows** may have been changed externally, so it will not make a backup definition if one does not already exist.

  This method of working will not produce conflicts within PRIMER, but it does have the potential to cause confusion for you, the user. If you want to invoke a standard editing panel on an item in the **Data rows** it is better to use the **Edit...** option on the popup menu attached to its row index button. This will map exactly the same editing panel, and if the definition is changed it will also update the relevant **Data row** and create a backup, making it possible to undo changes.

- There are a few specific instances where a keyword editor panels retains something of the "old" layout which restricted display to certain keyword suffices. These have been retained for ergonomic reasons, and are:
  - **`*ELEMENT_SOLID`** There are options to restrict display by number of nodes on element (eg tetrahedra, wedges, hexahedra) since experience has shown that it makes sense to restrict display in this way.
  - **`*ELEMENT_SHELL`** There are similar options to restrict display, and a further option to segregate 4 noded **`SEATBELT`** elements, since these are really shells in disguise.
  - **`*INITIAL`**_xxx_**`STRESS/STRAIN`** There are options to limit display to specified numbers of integration points, both through the thickness and on plan.
  - **`*SECTION_xxx`** is split up into the different section types (**`_BEAM`**, **`_SHELL`**, etc) since it would create a total mess trying to display all section types in a single panel.

- The **`*MAT`**erial keyword editor limits the keyword suffices you can choose to only those material types currently in the model. If you want to create a material of a new type you won't be able to select its keyword suffix from the sub-keyword popup, instead you will have to use [Right click on row index button] **Create** (to make a new material) or Edit (to change an existing one) and use the standard scalar editing panel to do this. Once you update from that panel the keyword editor will include the new material type in the list of keyword suffices in the keyword editor. A moment's thought explains this limitation: with over 150 material models to choose from showing them all in the sub-keyword suffix popup menu would be totally impractical.

5.1.4 "Daisy-chain" functionality via Popup menus

The previous sections describe how to invoke and process data via Keywords.

This constitutes "ab initio" invocation of entity creation, modification, etc; where no particular context is implied. For example creating a new material in isolation (without any external references to it) means that the material may be of any type, and is not limited to types suitable for a particular class of element.

However the "Create", "Edit" and "Browse" PRIMER capabilities for all the following keywords may also be called from a given context. For example:

```
ELEMENT --(edit)---+
                   |
                PART -- (edit) ---------+
                                        |
                                        +-- MATERIAL
                                        +-- SECTION
                                        +-- EQUATION OF STATE
```

What this implies that while processing an element you might edit its Part definition and, through that, its Material, Section and Equation of State.

In this situation, referred to as "Daisy-Chaining", those items in the chain below the element have an implied context. For example the Material and Section definitions will be constrained to the element type of the Part, and you will not be permitted to add other types of element to the part.

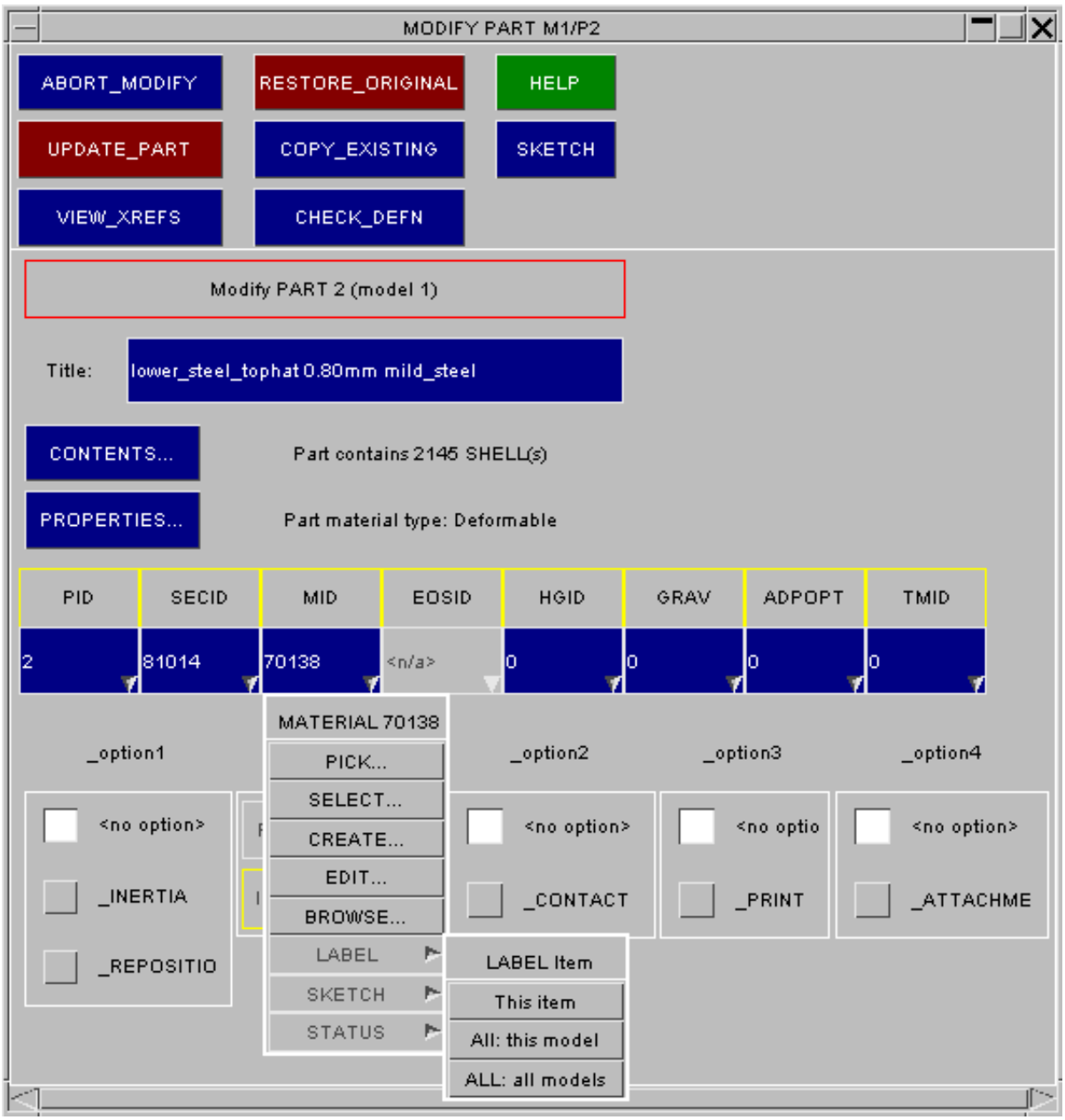
This process is invoked by "popup" menus, as shown in this example here.

The user is editing a Part, and wants to Label everything that uses this Part's material.

Thus he has used:

[Popup]-->**Label-->Sketch**

The **Edit** and other daisy-chain functions are also available. These are described in the table below.



"Daisy-Chain" function	Description
PICK...	Select an item for this location by picking directly from the screen with the mouse.
SELECT...	Select an item for this location via the standard selection menu, which offers filtering, screen-picks by various methods, typing in of labels, and so on.
CREATE...	Create a new item for this location. This maps the standard creation panels as described in the following sections.
EDIT...	Edit the current selection, using the standard Create/Edit panel as above. Only available if the selection actually exists already.
BROWSE...	Browse the current selection. This lets you look at it using the standard Create/Edit panel, but any options that would allow changes are disabled. Therefore this is a "safe" way of viewing data.
LABEL > **or** **VIEW/LABEL >**	The "Label" function allows you to label either the single item, or the whole range of these, on the current image. The "View" option is only available for those classes of item for which a viewing function (see section 4.5) exists.
SKETCH >	Lets you sketch this item, or the whole range of them, on the screen. (Sketching is always uses "line" mode, in white, to superimpose the items on the current image.)
STATUS >	Provides status listings at different degrees of complexity.

Daisy-chained panels may have "children" and "grand-children" and so on to any level of complexity, but all are owned by the original "parent". Dismissing the parent panel will destroy all its child panels, to the <nth> generation, equivalent to dismissing each one individually: they do not exist autonomously.

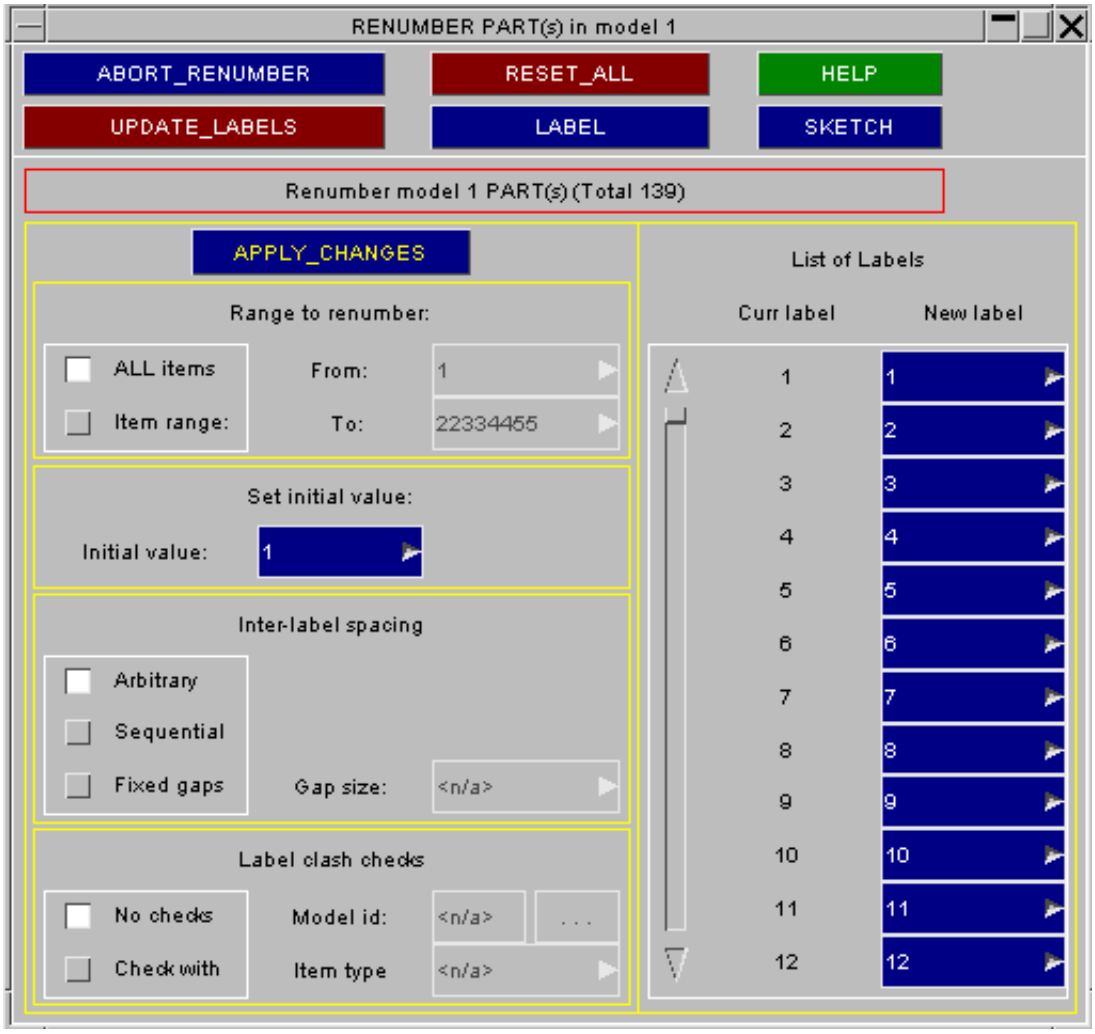
This makes is possible to be performing the same function on an item in two or more places simultaneously, for example you might be editing a part in two separate panels. There is no conflict, since all Create and Edit operations always take place on a "scratch" copy of the original item, and the original is only modified when the user explicitly requests this by using the **UPDATE/CREATE** button.

Therefore the permanent definition of the part in this example is always defined by whichever panel most recently updated it.

Clearly multiple Browse operations, being "read-only" in nature, never constitute a threat to the original definition since they will never update it, which is why the facility is provided.

5.1.5 Standard category renumbering panel.

Wherever a **RENUMBER** option is available for an item category this will invoke the standard renumbering panel for that item



This panel is the same as that described in section 3.7.1: refer there for usage details.

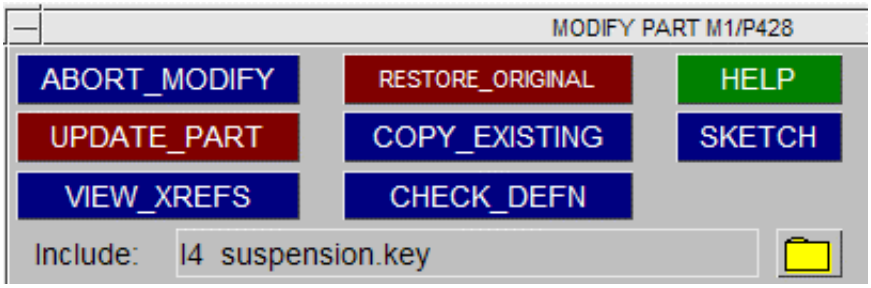
As with editing functions it is possible to have two renumbering panels live for a given model/item combination: one opened from its create/edit panel, and one from the main Model>Renumber command.

This is not a good idea, but should it happen the one from which the most recent **UPDATE_LABELS** command is issued will "win" in determining the final labels for that item category.

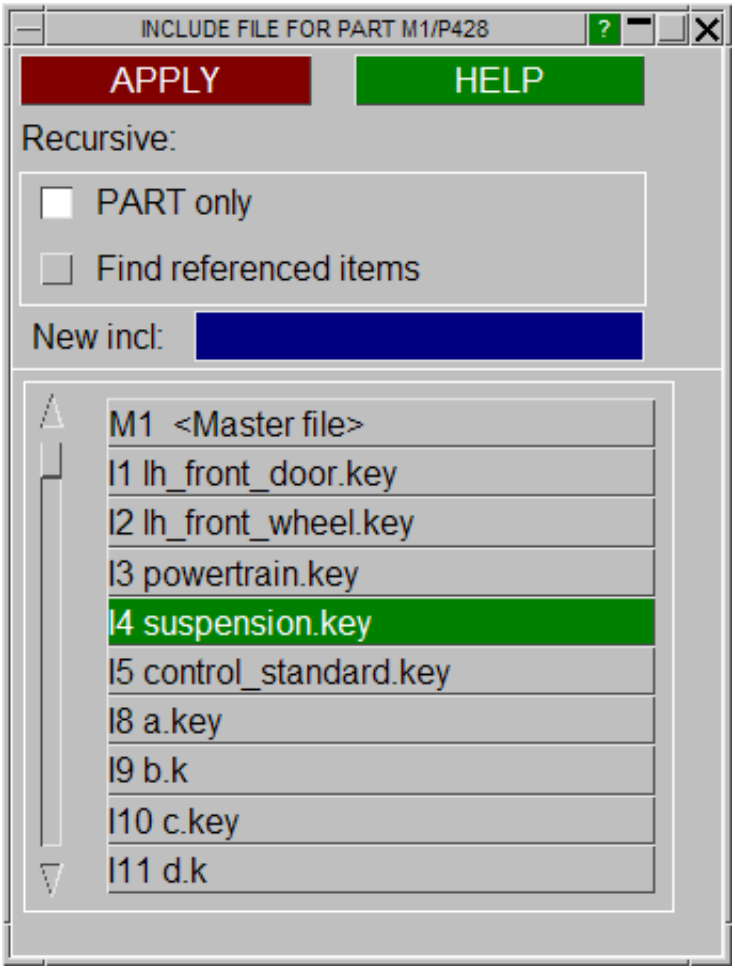
Renumbering is normally carried out using the **RENUMBER** option from the **MODEL** menu (see section 3.7). This section describes the standard category renumbering panel available for specific keywords.

5.1.6 INCLUDE file selection on edit panels

Edit panels now display which INCLUDE file the entity is located in. The INCLUDE file can also be modified in the edit panel when creating or modifying an entity.



The INCLUDE file is displayed in the box below the main edit panel buttons. If the INCLUDE file has an **INCLUDE_TRANSFORM** applied to it, **(T)** will be displayed next to the INCLUDE name. By clicking on the folder icon next to the include name the include selection panel opens up:



At the bottom of the panel all the INCLUDE files in the model are listed. The INCLUDE file that the entity being modified/created in is highlighted in green. To change the INCLUDE file, click on another file in the list and click **APPLY**. New INCLUDE files can also be created in the include selection panel. To do this, enter the name of a new INCLUDE next in the blue box next to **New incl**. The INCLUDE file will then be created. By default, only the calling entity will be moved to a newly selected INCLUDE file. By selecting **Find referenced items** a new panel is opened when clicking **APPLY** which allows you to choose any referenced items you may wish to move as well.

5.1.7 "Long" keyword format and "large" labels.

Historically the LS-DYNA keyword input deck has allocated field widths of either 8 or 10 columns for more items, resulting in a de facto maximum label of 99,999,999. The use of an intermediate "structured format" deck further limits some field widths, with a few items being limited to 5 columns, ie 99,999. In the discussion below this "small" format will be referred to as "regular".

To address this problem LS-DYNA (release 971 R7.1 onwards) also supports a "long" input format in which all data fields become 20 characters wide, permitting larger labels up to approximately 9e15 (in practice up to 999,999,999,999,999) to be used.

PRIMER has been developed to support these features, and the current status is:

PRIMER up to and including V11.1	Only supports the original regular format input deck, and labels up to ~2e9.
PRIMER V12 onwards	Supports both "long" keyword format and "large" labels up to ~9e15 Also supports "extended" label ranges up to ~9e18.

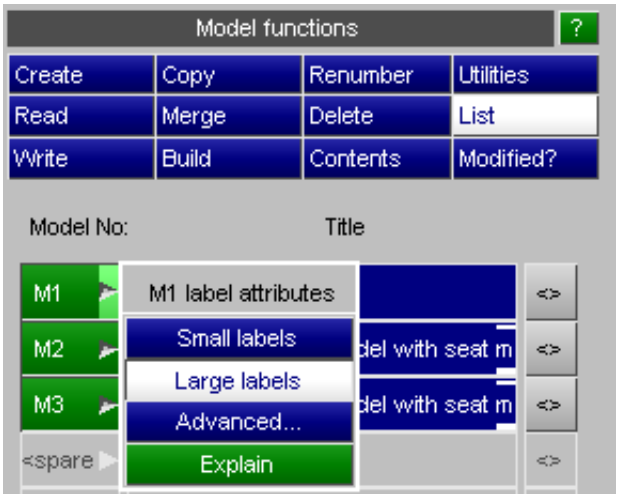
Setting the current label limits in PRIMER

PRIMER V12 detects the current label limit range from the input format and the largest label in the model, and selects the appropriate output format and label checking limits automatically. Historically input decks have been in regular (small) format with labels up to 99,999,999, and since this gives smaller keyword files than large format PRIMER will keep to these limits if possible, and only set large format if required.

Label format is a "per model" attribute, and each model in PRIMER can have different settings which can be changed in the following ways:

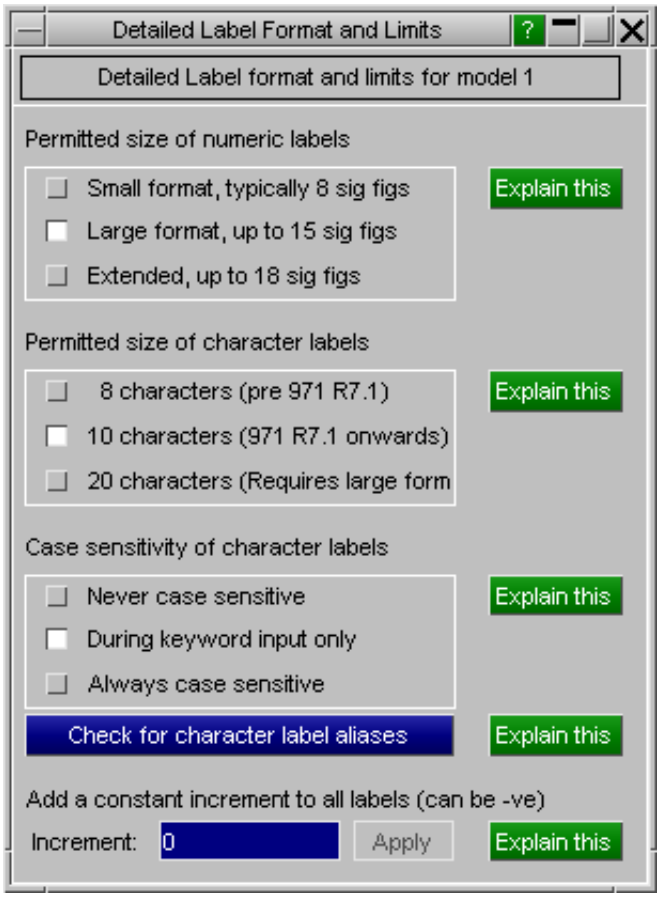
Simple method: using the **Model List** panel

You can swap a model between "small" and "large" labels formats using the popup attached to the ***Mn*** buttons thus:

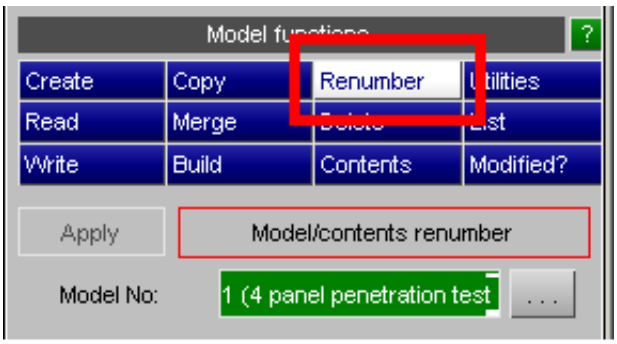


Meanings of label attributes options	
Small labels	Numeric labels: 1 to 99,999,999 Character labels: 10 characters
Large labels	Numeric labels: 1 to ~9e15 Character labels: 20 characters
Advanced...	Maps the detailed panel below.

Advanced... method: using the Detailed label format panel.



The detailed label panel can also be mapped via **Model > Renumber, Label range**



As explained above PRIMER works internally using 64 bit integers for labels, permitting the full ~9e18 range in all internal contexts, however it is useful to impose "soft" limits on labels if you are planning to use normal (small) output format. The **Label Range** panel allows you to set one of three modes on a per-model basis:

Label format chosen	Typical limit sig figs	Comments
Small format (8 sig figs)	8	Limits most labels to 8 "9"s Limits character labels to 8 chars Fully supported by Javascript
Large format (15 sig figs)	15	Limits labels to approx 15 "9"s Limits character labels to 20 chars ***Suitable*** if you plan to use LS-DYNA or Javascript
Extended format	18	Limits labels to approx 18 "9"s Limits character labels to 20 chars ***NOT suitable*** if you plan to use LS-DYNA or Javascript

The sections below explain in more details what these limits mean and why they exist.

(The ability to add a constant increment to all labels in a model was added for testing purposes when developing the ability to process 64 bit labels. It is an entirely "dumb" capability with no checking for consequences, and if you use it you do so at your own risk. More sophisticated renumbering capabilities are available under Model, Renumber.)

More about "large" labels

In "long" LS-DYNA format all data fields become 20 characters, and the differences between this and normal format are given in the following table:

Label type	**Limit in regular keyword format**		**Limit in long keyword format, all fields 20 wide**	
Most items, eg *NODE, defined via 8 wide fields	99,999,999	8 sig figs	~9e15	15 sig figs
Materials etc defined by 10 wide fields	~2e9	9 sig figs		
Items constrained by 5 wide fields	99,999	5 sig figs		
Character labels	8 characters, or 10 chars from 971R7.1 onwards		20 characters	
Parameter names	9 characters		9 characters (as of July 2013)	

Note that

- LS-DYNA limits labels in long format to 15 significant figures and that scripting in Javascript imposes the same limit, both dictated by the mantissa precision of the 64 bit IEEE double length floating data type.
- PRIMER is capable of handling labels up to the 18 significant figure limit imposed by the 64 bit IEEE "long long" integer type.

  "Large" label format in PRIMER imposes the same 15 digit limit as LS-DYNA in order to ensure compatibility "Extended" label format in PRIMER permits the full 18 digit range, but is not the default because it is incompatible with bot LS-DYNA and Javascript.

These limits are explained in more detail below.

The following discussion is an advanced topic and you don't need to understand it fully. If you need further help please contact Oasys Ltd.

Numerical label limits dictated by the hardware

In order to support "large" labels it has been necessary to move from using 32 bit (4 byte) integers to 64 bit (8 byte) ones to store them. The maximum values that can be represented by these two data types are:

Label limits dictated by 32 bit and 64 bit integer data types			**#useful decimal digits**
32 bit signed integer	Limit is 2**31 - 1	+/-2,147,483,641 (~2e9)	9
64 bit signed integer	Limit is 2**63 - 1	+/-9,223,372,036,854,775,807 (~9e18)	18

PRIMER V12 has been modified to use 64 bit integers for all labels, therefore it is capable of dealing with the full range of ~9e18.

Numerical label limits dictated by scripting and LS-DYNA

There are further limits imposed by both LS-DYNA and the Javascript language, so if you use either this will limit the maximum size of label you can use.

Javascript stores all numbers in a 64 bit "double" variable, and LS-DYNA also uses this type in some contexts. This occupies the same space in the hardware as a 64 bit integer, but it is a floating point value with different internal organisation, and this means that it only has 53 bits of mantissa precision available since the remaining 11 bits store the sign and exponent. The maximum value that can represented accurately in decimal format is:

Label limits dictated by 64 bit long double data type used by Javascript			#useful decimal digits
64 bit double	Limit is 2**53 - 1	+/-9,007,199,254,740,989 (~9e15)	15

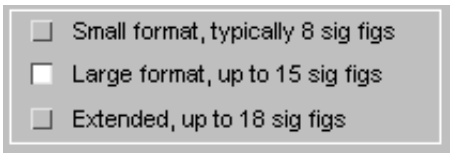
Numerical label limits dictated by common sense

It will be clear from the discussion above that the absolute maximum usable numbers are not very human-friendly powers of 2, and moreover attempts to exploit every last piece of precision could result in numerical overflow. For example 2,200,000,000 cannot be represented by a 32 bit signed integer.

Therefore it is ***strongly*** recommended that you stick to human-friendly numbers of decimal digits, giving limits expressed by some number of 9s, and the following table gives suggested sensible limits. It is further recommended that when using large format you limit label usage to 15 decimal digits, as numbers larger than this will not be accessible in Javascript or read by LS-DYNA.

Input deck format	Max recommended label value	#digits	Comments
Regular (small) input format	99,999,999	8	Limited by typical field width of 8
Large (long) input format	999,999,999,999,999	15	Limited by the use of 64 bit double. **Permits use of LS-DYNA & Javascript**
Extended format	999,999,999,999,999,999	18	Limit dictated by use of 64 bit integer. **Prohibits use of LS-DYNA & Javascript**

The table above explains where the settings on the **Label Range** panel come from, and it is recommended that users stick either to the 8 or the 15 significant figure modes.



5.1.8 Character labels

Prior to LS-DYNA 971 R7.1 character labels may be up to 8 characters wide, and can be used for the following keywords

- EOS
- HOURGLASS
- MATERIAL (structural and thermal)
- SECTION

From LS-DYNA R7.1 onwards character labels can be as wide as the data field permits, typically 10 characters in regular format, and 20 characters in long format.

Also the usage of character labels has been extended to keyword

- PART

PRIMER fully supports the above:

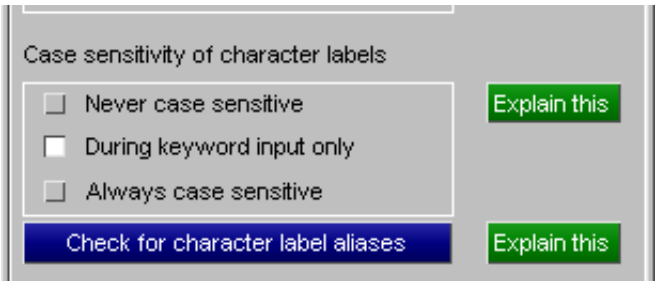
- Internal storage of character labels is always 20 characters wide, but "soft" limits on character label width can be set.
- In the editing panels for the above keywords character labels can be used instead of integer labels. Additionally the character labels will be shown in object menus and any other context where item labels are displayed.
- Various strategies for dealing with mismatches between character label usage and chosen output format are provided.

Case Sensitivity of Character Labels

Testing reveals that LS-DYNA may treat character labels in a case-sensitive fashion, for example labels "**`Body`**" and "**`BODY`**" may be considered to be different to one another. This may change in the future, but it appears to be the behaviour of LS971 R7.1 (tested February 2014).

Therefore PRIMER adopts the following strategy regarding the case sensitivity of character labels:

- By default it does ***not*** handle character labels in a case-sensitive way: normally PRIMER will consider **`Body`** and **`BODY`** to be the same label.
- In this default mode the label will be stored exactly as it is first encountered. So if the first occurrence is **`Body`** then this is how PRIMER will "remember" it and write it out subsequently, regardless of what other permutations of upper and lower case text is found in later occurrences of the word, and all such permutations of the word will map onto this single definition.
- It is possible to change this default behaviour by selecting the "**Case Sensitivity**" option on the Advanced labelling panel.



This option, which is programme-wide and not just for this model, may also be made the default by using the preference

`primer*clabel_case_sensitive`: **`keyin only`** *or* **`always`** *or* **`never`**

The meanings of these options are explained below.

Case sensitivity is **`keyin_only`**

This is the default mode and works as follows:

- During normal operation PRIMER runs in case-Insensitive mode, so **`Body`** and **`BODY`** are considered to be the same label.
- During keyword input only PRIMER reads character labels from the keyword file as if they are case-Sensitive.
- At the end of keyword input it searches for any character labels that might be accidental case mis-matches, such as **`Body`** and **`BODY.`**
- If any are found they are listed and the user is asked to determine what action to take. There are three possible options:

Merge cases	Will merge all case-dependent matches into a single definition. Thus **`Body`**, **`BODY`** and **`body`** will all be merged into a single definition. The rules used when merging definitions are described below under "The rules used when merging "alias" character label names".
Switch to case-sensitive	Switches PRIMER's mode of operation to Case Sensitive **`always`** and leaves the various permutations of labels unchanged.
No action	Does nothing, leaving the current mode as Case INsensitive, but also not doing anything with these separate labels.

- When keyword input is complete PRIMER reverts to running in case-Insensitive mode unless **Switch to case-sensitive** was chosen.

The reason for this is that it is not possible to tell whether the use of case-sensitive labels in the input deck was deliberate, exploiting LS-DYNA's behaviour, or a mistake - possibly from merging files from different sources. By treating the problem this way the user can either choose to continue, or use PRIMER to resolve differences and make the model consistent.

If there are no potential matches this post-keyin check is silent, so no message means no problem.

Case sensitivity is **always**

In this mode PRIMER will always treat character labels in a case-Sensitive fashion and will consider **Body** and **BODY** to be totally separate labels in all contexts.

No post-keyin checking for character label case-sensitivity will be performed since all character labels are treated verbatim.

Case sensitivity is **never**

In this mode PRIMER will always treat character labels in a case-Insensitive fashion and will consider **Body** and **BODY** to be teh same labels in all contexts.

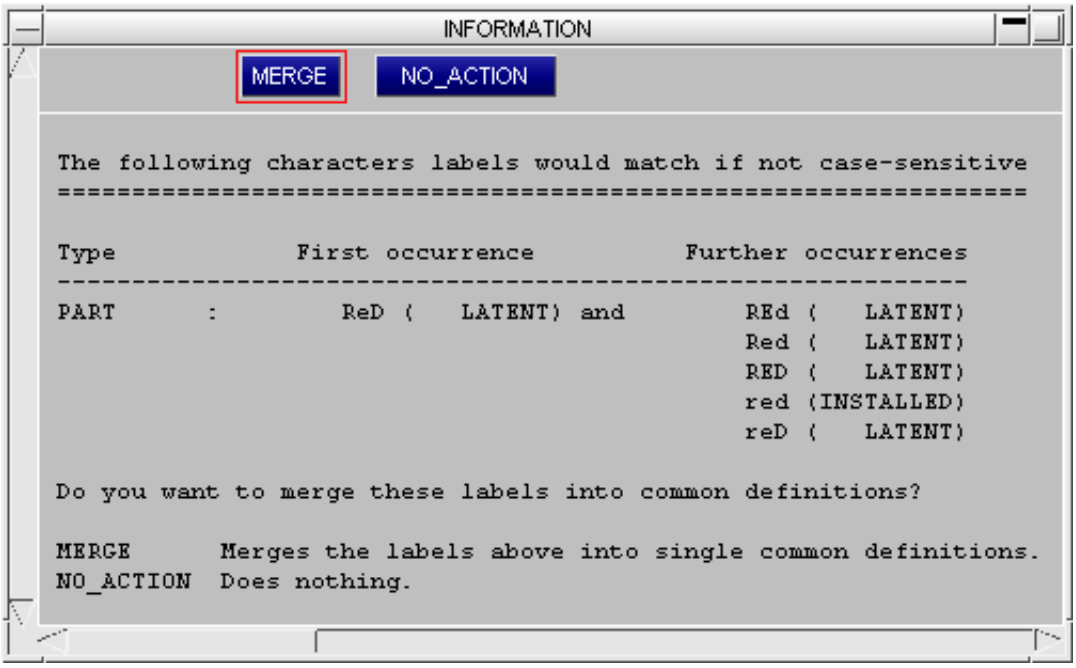
This remains the case during keyword input, where no post-keyin checking is required because all variations on a name will have been merged into a single version (the first that was encountered).

Check for character label aliases

This command will run manually the check that is run automatically after keyword input, searching for any character labels that would be "aliases", ie the same word, if compared in a case-insensitive way.

Here is an example of the result in a model in which the spellings **ReD**, **REd**, **Red**, **RED**, **red** and **reD** of the name "Red" have been used for ***PART**.

Using **MERGE** will "collapse" all these definitions onto a single definition.



The rules used when merging "alias" character label names

When merging character labels PRIMER will always prefer to use the spelling of the "installed" version of the name, which means the name used when the item was actually defined.

This is best illustrated by example. Here are the keywords from the file that produced the warbing message above:

```
*PART
Red panel
       red          2          1          0          0          0          0
0
```

The lines above are the "true" definition of the part, and thus this definition is "installed" using the

```
*ELEMENT_SHELL
       1      RED       26       27       14       15
       2      Red       27       12       13       14
       3      REd        9       10       27       26
       4      ReD       10       11       12       27
       5      reD       15       16       28       26
       6      red       26       28        8        9
```

The above references to the **`*PART`** on **`*ELEMENT`** cards make a series of "latent" definitions using the various spellings of "red". These alternative spellings are latent because they are "referenced but not defined".

So if **MERGE** is used in this example the spelling of the "true" installed definition, **`red`**, will be used.

If there is not a single "installed" definition to use then the rules are:

All candidate definitions are "latent"	The last encountered spelling will be used.
More than one candidate is "installed"	The last encountered installed spelling will be used. ***WARNING***: when there are are multiple "installed" definitions using different upper/lower case variations of the same character label then a **MERGE** will collapse them all onto the last such definition, leaving a single definition and ***deleting all prior ones***.

It is strongly recommended that users do NOT rely on the case-sensitive behaviour of LS-DYNA to make character labels distinct.

Not only may this behaviour change in future LS-DYNA versions, but it is a "hostage to fortune" that is likely to give rise to errors.

Permitted Character Label syntax

Testing of LS-DYNA 971 R7.1 (Feb 2014) suggests that the permitted syntax for a character can be described as "anything that is not a number". For example it appears that the following are all legal:

Characteristic	For example
Embedded white space	"**`Wheel hub`**"
Starting with a digit 0 - 9	"**`123 Pelham`**"
Contains non-alphanumeric characters	"**`That's OK!!`**"

PRIMER 12 will support this, but in order to do so a substantial amount of grammatical checking of input, both interactively and from keyword file, has to be turned off which means that grammatical errors or mis-typed labels may be interpreted as character labels. This behaviour can be controlled by the following preference

primer*clabel_syntax:	**very_loose** (default)	Anything which is not a "number" is treated as a potential character label. The label may start with any character, including **0-9** and **+** or **-**. Embedded white space is permitted, and only the characters listed below are forbidden.
	loose	As "very loose" except that the label may not start with **0-9** or **+** or **-**.
	rational	The label may not start with 0-9 or + or -, and also must not contain any embedded white space.

In addition there are two further limitations in PRIMER that apply to all the settings above:
- A character label may not start "**&**" or "**-&**". This is to avoid confusion with parameters.
- A character label may not contain the double quotes character **"**. This is to avoid confusion in properties files.

It is not known whether LS-DYNA will also impose these two limitations, but PRIMER requires them.

This section is correct at the time of writing, March 2014, but it is possible that the permitted syntax in LS-DYNA will change during the life of this release of PRIMER making it necessary to change from the default "very loose" syntax rules.

It is strongly recommended that users do NOT rely on the "very loose" character label syntax permitted by LS-DYNA.

Not only may this behaviour change in future LS-DYNA versions, but it is likely that third party software may have problems dealing with a totally arbitrary syntax for character labels and PRIMER may trip up too.

At the very least "loose" syntax, preferably starting names with A-Z is recommended, and to minimise the likelihood of errors when processing input decks through other software it is recommended that "rational" syntax is used, obeying the following rules:
- Character label names should start with the letters A-Z
- They should not contain any white space, use the underscore character to separate words. For example **bearing_assembly**
- Avoid using exotic characters in names: it is best to restrict usage to to A-Z, 0-9 and underscore. In particular avoid +, -, ", ’, &, :, and so on.

Merging models containing clashing character labels

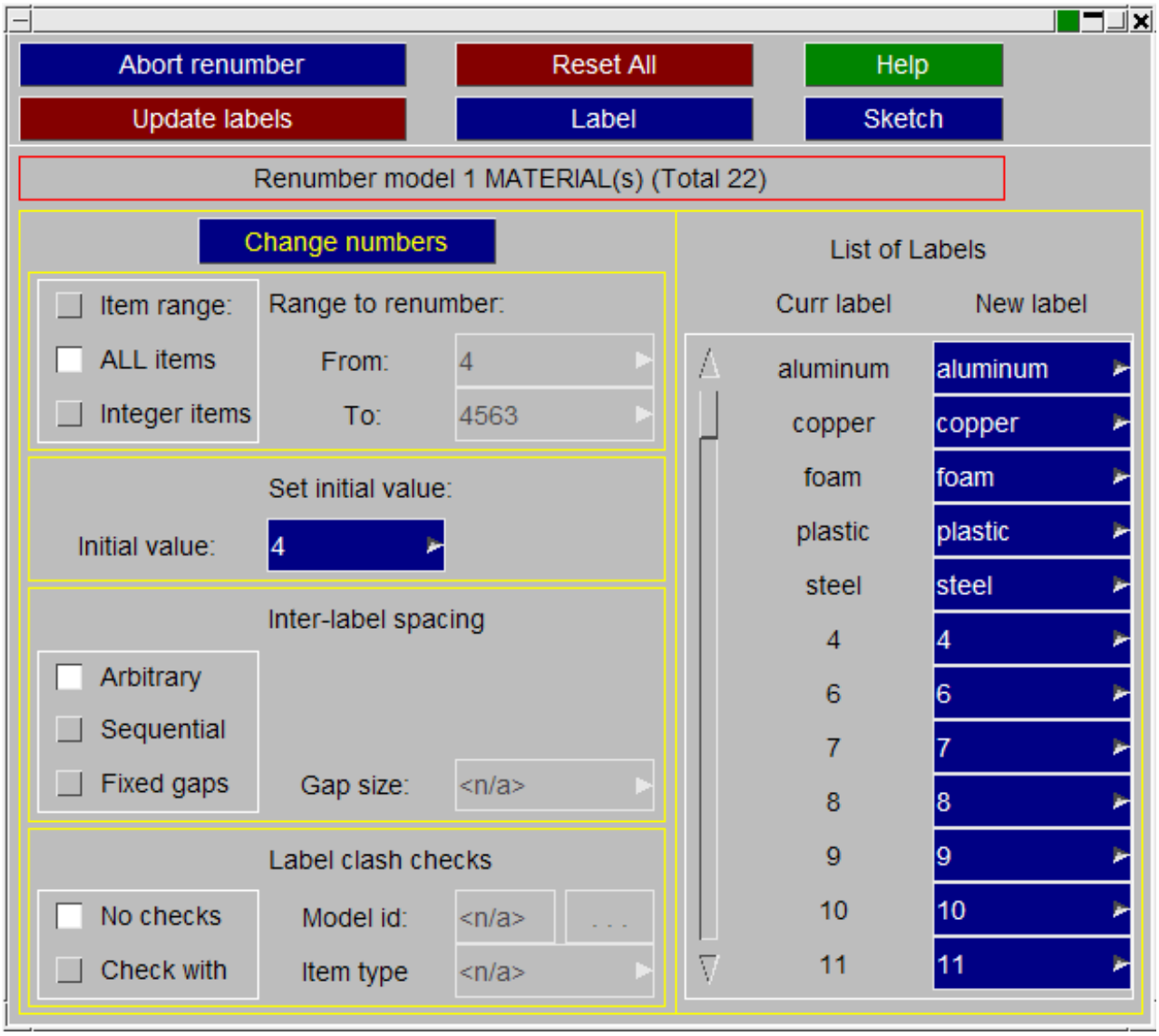
If clashing character labels are found when merging models PRIMER will try to fix the clash by:
- if the name ends with ’_<number>’ then read the number, increment by one and replace ’_<number> ’with ’_<number+1>’.
- otherwise append ’_1’ onto the name
- repeat until a non clashing name is found
- if the name is longer than 8 characters, strip off the last character in the name (after stripping off any ’_<number>’) and repeat the process.

More details on model merge are available in section 3.4.

Renumbering models containing character labels

If a model contains character labels then they will not be included when renumbering using the main Renumber contents panel.

However they can be viewed changed/renumbered etc using the individual category panel. e.g. the following figure shows the renumbering panel for a model that contains materials with character labels.

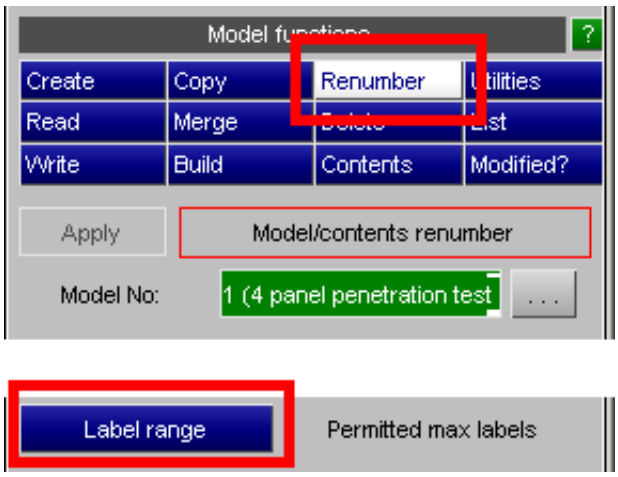


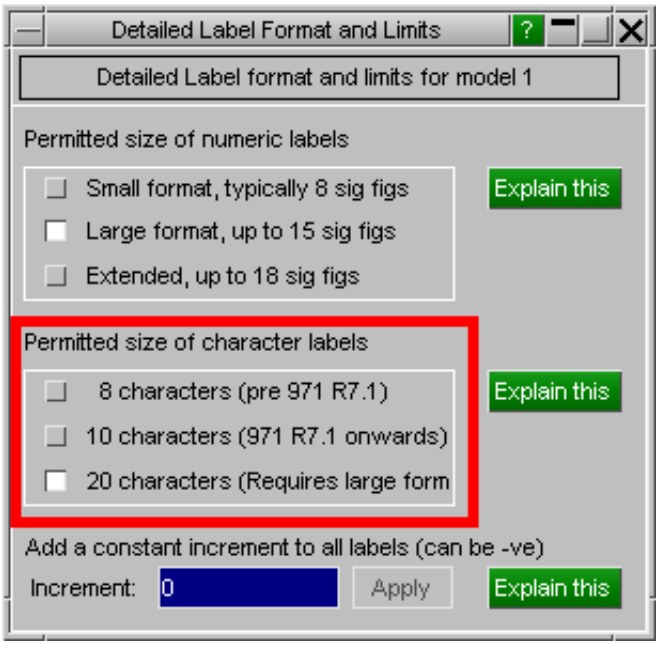
If the category contains character labels then the **Range to renumber** section will have an additional option, **Integer items**. If **ALL items** is selected then the character labels will be included in the renumbering. If **Integer items** is selected then they will not be included.

Setting "soft" limits on character label size

As explained above the limits on character label width from LS971 R7.1 onwards are 10 characters in regular format, and 20 characters in long format. However if you are planning to generate keyword decks for older versions of LS-DYNA you may wish to enforce the older 8 character limit in order to ensure compatibility.

Model, Renumber, Label Range will map the label control panel





The "permitted size of character labels" is a per-model setting that is initialised automatically following inspection of an input deck. As with numeric labels PRIMER will prefer the regular file format of 10 characters width, but if the file is detected to be in long format then 20 will be set instead.

This size is used purely for checking, meaning:

- Whenever a "check" function is run on a keyword the length of any character label will be tested against this value.
- Whenever a character label is created or modified you will not be permitted to define a label longer than this value.

You can change this limit at any time, it will not affect the current contents of the model at all.

Dealing with character labels in older output formats

Various problems can arise when writing keyword decks for versions of LS-DYNA prior to LS971 R7.1

Problem	**Solution in PRIMER**
Character labels have been used for *PART, but this is not supported in the chosen format.	Each such part is automatically given a new and unique numeric label in the keyword output deck so that this is readable. The character label of the part inside PRIMER is not changed.
Character labels >8 characters wide have been used for *PART, and referred to by *ELEMENT cards, but use of regular format means that the data fields on the *ELEMENT cards are only 8 columns wide.	From 971 R7.1 onwards the relevant *ELEMENT cards will be written in long output format, designated by appending a "+" to the end of the keyword header line for the elements. Output formats prior to 971 R7.1 will adopt the "replace character label with unique numeric label" strategy as described above.
Character labels >10 characters wide have been used, but the chosen output format is regular, meaning that only 10 column wide data fields are available.	This will be detected by the normal pre-output model checks, and the user will be warned and presented with a range of options. The correct solution is to swap to long format, making 20 wide data fields available; alternatively the character labels can be edited manually to reduce their width to 10 characters or less. However if the user continues with regular format output then any character labels wider than the field width available will be replaced by a unique new label as described above.

5.1.9 PGP Encrypted keyword data

LS-DYNA release 971 onwards permits data encrypted with "Pretty Good Privacy" (PGP) to be embedded anywhere in an input deck using the sequece:

```
-----BEGIN PGP MESSAGE-----
[Any number of lines of encrypted data]
-----END PGP MESSAGE-----
```

Typically this used to encode **`*MAT`** and **`*DEFINE_CURVE`** data in order to protect the intellectual copyright inherent in the values used, but it can be used for any keyword in a completely arbitrary fashion.

PGP uses "public/private key" encryption, in which anyone with the public key can encrypt data, but only those in possession of private key can decrypt it. LS-DYNA has this private key, but PRIMER does not, which means that the information in the encrypted sections is completely opaque to PRIMER. Therefore blocks of encrypted data within an input deck are handled as follows:

Free-standing PGP encrypted data blocks.

When a block of PGP encrypted data is encountered ***outside*** the specific cards of a keyword it is referred to here as "free-standing", meaning independent of any keyword. PRIMER handles such blocks as follows:

- They are copied verbatim to a "saved pgp data" file.
- Separate files are used for the master keyword file and any include files.
- When the model is written back out any saved PGP data blocks are written out in the following locations:

Where free-standing PGP data blocks are written in the keyword deck

Up to and including PRIMER V12 any free-standing PGP data blocks found in a file were always written out immediately before *END during keyout, and in most cases this was fine because the position of a keyword in a deck does not matter.

However if the PGP data contained **`*PARAMETER`** definitions which were referred to elsewhere in the deck this caused LS-DYNA to reject file because these parameters were referred to before they were defined. A similar problem could arise if the encrypted block referred to parameters previously defined, and the position at which it was output preceded these definitions.

Therefore more flexibility about the location in the deck at which free-standing PGP is written has been added to PRIMER V12.1 by introducing the preference:

`primer*freestanding_pgp_output_location:`

The possible values for this preference and the corresponding locations for writing free-standing PGP data are given in the table below.

Preference value	Location for free-standing PGP data in file
not defined	Immediately before *END (existing behaviour)
`top_of_file`	At top of deck
`after_parameters`	After *PARAMETER cards
`after_includes`	After *INCLUDE cards
`before_end`	Immediately before *END

A further preference can be used to define how free-standing PGP data is written:

`primer*freestanding_pgp_output_method:`

Preference value	Location for free-standing PGP data in file

not defined	Embedded in the file itself (existing behaviour)
`embedded`	Embedded in the file itself
`as_include`	As include file "**`fs_pgp_data.key`**"

There are some qualifications to the **`as_include`** setting:

- It only applies to free-standing PGP data in the master file. Such data in include files is always left "embedded"

  except that:

- If include files are merged into the master file for output then all free-standing PGP data in both master and any include files are collected together and written to this single include file.

Free-standing PGP data blocks in include files

If the order or position of the encrypted data in your deck ***are*** important then it is recommended that you place them in an include file, since PRIMER will treat such a file verbatim and it does at least reduce the problem of positioning the data to the insertion of a single **`*INCLUDE`** statement at the desired position in the master input deck.

From V11 onwards a special case is made for Material and Loadcurve data defined entirely within free-standing PGP encrypted blocks, which may have alternate (simple) definitions defined for them in a special post *END encrypted section. From V13 onwards it is also possible to define parameters in this way. See Providing Alternate Definitions... below.

Problems using merged keyword output when include files contain encrypted *END

It is sometimes the case that proprietary data is supplied as an include file, and the entire contents of that include file are encrypted. This is not a problem in itself, but it can cause problems if the encrypted block in the include file contains a **`*END`** keyword and the option to write a single "merged" output deck is used. Consider the following:

Original input deck with separate include file	What happens when LS-DYNA reads the file

`Master file`
`  |`
`  *INCLUDE`
`  |`
`  +-- Include file`
`  :  |`
`  :  PGP Encrypted data that`
`  :  contains encrypted *END`
`  |`
`  *other keywords`
`  :`
`  *END`
	Master file is opened
	
	Include file is read and decrypted line by line
	
	When (encrypted) **`*END`** is encountered the include file is closed and reading returns to the master file.
	
	Further keywords are read
	
	Input terminates successfully

If this original deck is written out as a single "Merged" file, and the option to embed PGP encrypted data in the master file is used:	
Merged output deck with embedded PGP data	**What happens when LS-DYNA reads the file**

`Master file`
`  |`
`  --- Begin PGP data---`
`  PGP Encrypted data that`
`  contains encrypted *END`
`  --- End PGP data---`
`  |`
`  *other keywords`
`  :`
`  *END`
	Master file is opened, and any cards prior to the embedded PGP data are read successfully
	
	The PGP encrypted data is read until **`*END`** is encountered, ***whereupon keyword reading stops.***
	
	Any further cards in the input deck are ignored and keyword input terminates prematurely with errors.

Therefore when generating include files that are fully encrypted the following practice should be adopted:

- Put the **`*END`** card in clear text at the end of the file, not in the encrypted section.

  or

- Omit the **`*END`** card altogether. It is not strictly necessary as LS-DYNA will recognise the physical end of the file for itself, although this practice is not recommended.

PGP data within a *MAT definition.

A special case is that PGP encrypted data with a material card is treated "intelligently" if the following format and rules are applied:

`*MAT_xxx_TITLE`	Where *xxx* is the material type. A new *MAT header is required for each material
`<title>`	• V10: Must be (I10,A70) format, with the material label in the first 10 columns • V11: Permits a more sophisticated treatment that also allows density, young's modulus and poisson's ratio to be read. This is described in more detail below
`-----BEGIN PGP MESSAGE-----`	The encrypted data must be the whole of the rest of this material's data only.
[Encrypted data]	
`-----END PGP MESSAGE-----`	

The title parsing rules used in PRIMER V11 onwards for the title line in a partially PGP encrypted material definition			
Mandatory:	The first token **`nnnn`** on the line that can be parsed as a label or The first token starting **`MATLx_nnnn`** on the line	Will be interpreted as material label **`nnnn`**	
Optional	A subsequent string "**`RO=xxxxx`**" (or the lower case "**`ro=xxxxx`**")	Will be interpreted as density **`xxxxx`** (There must not be any spaces in the string **`RO=xxxx`**) This will be applied to all material types for which the first data field is density (RO). It will not be applied to other material types, for example spring materials where the first field is stiffness.	
Optional	A subsequent string "**`YM=xxxxx`**" (or the lower case "**`ym=xxxxx`**")	Will be interpreted as Young's Modulus (E)	These values are currently only applied to material types 1 (elastic), 9 (null) and 20 (rigid).
Optional	A subsequent string "**`NU=xxxxx`**" (or the lower case "**`nu=xxxxx`**")	Will be interpreted as Poisson's Ratio (ν)	

If these rules are followed then PRIMER "knows" what the material label and type are, even if it doesn't know anything about the details of the data.

Given this information it can deal with the material in a reasonably intelligent fashion, for example if it is rigid then it can apply the normal rules about rigid bodies. However:

- If RO, YM or NU are not defined PRIMER cannot "know" what the material's yield stress or other properties (perhaps used in timestep calculations) are, so:

  A nominal density or stiffness is inserted as appropriate
  A minimal nominal set of remaining required data is also inserted.

- PRIMER also cannot "know" about other items, typically loadcurves, that this material might reference.

  When a deck also contains encrypted loadcurves PRIMER assumes that all encrypted loadcurves are referenced by all encrypted materials, thus locking the loadcurves against unintended deletion.

- The labels of encrypted materials are locked against being changed in most (but not all) contexts. This is because the encrypted material definition has a label id buried inside it, and this must be kept unchanged.

It is not possible to prevent a determined user from relabelling such a material if he really tries, for example via external Text Edit, hence the proviso above. It is assumed that users will be sensible about labels of encrypted items!

Here is an example of a partially encrypted material definition (*the encrypted section is fabricated*)

```
$
*MAT_PIECEWISE_LINEAR_PLASTICITY_TITLE
       102 RO=7.85E-9 Normal material title
-----BEGIN PGP MESSAGE-----
aksjdfhqwhjhfvkjqhwdqef/qwdfqefqw/fdqwfqwoqijwfqwfnbqkwjhvjqljwodfkjvi
HFYDHjghgkjhYfiyIHGLOHihougfDtygPhuyfyFyuf68587JLkndvnbGpwwqlehgvhsyc0
-----END PGP MESSAGE-----
$
```

In this example:

- The material label is 102 (red text). This has been right justified to occupy the first 10 columns in order to ensure backwards compatibility with PRIMER versions prior to 11.
- The material density is 7.85e-9 (blue text).
- The complete title line will appear verbatim in PRIMER. If you edit it be careful not to change the two fields above.

Note also the from V11 onwards PRIMER also supports alternate definitions for wholly encrypted materials and loadcurves in a post *END "encrypted" block - see below.

PGP data within a *DEFINE_CURVE definition.

Another special case occurs when the (x,y) points data for a loadcurve are encrypted using the following rules and format:

***DEFINE_CURVE**	A new *DEFINE_CURVE header is required for each curve
<lcid> <sidr> ... <dattyp>	This line must be "in clear"
-----BEGIN PGP MESSAGE-----	The encrypted data must be all the (x,y) curve points for this curve only
[Encrypted data]	
-----END PGP MESSAGE-----	

If these rules are followed PRIMER "knows" the material label, and therefore can deal with references to this curve from other items. However:

- PRIMER cannot "know" what the (x,y) data points of this curve are, so a nominal pair of points (0,0) (1,1) is inserted.

Here is an example of a partially encrypted loadcurve definition. (*The encrypted data is fabricated*)

```
$
*DEFINE_CURVE
        11         0       0.0       0.0       0.0       0.0         0
-----BEGIN PGP MESSAGE-----
nciqwgfgv7qwbfv0qw37bf0v93b4v0934bvbskldf745bvnv0923tygvb0c9034bv034hg
vnoiquefd734v982bf0jnvb9034bfbv09bu40t90bw3enf2n398b2c0984gbb094299944
-----END PGP MESSAGE-----
$
```

In this example:

- The curve label is 11
- Other factors on the label line, sidr etc, have default values of zero. They could equally well have "real" values.

Note also the from V11 onwards PRIMER also supports alternate definitions for wholly encrypted materials and loadcurves in a post *END "encrypted" block - see below.

Providing alternative definitions for wholly encrypted *MAT, *DEFINE_CURVE and *PARAMETER definitions.

It is sometimes the case the proprietary input decks will contain material and loadcurve definitions that are totally

encrypted, missing even the keyword headers and title lines. In this case PRIMER cannot "know" that these definitions exist, and their absence can generate all sorts of problems and error messages during model check. To deal with this problem from V11 onwards PRIMER allows you to provide alternative definitions for the missing items in a special "encrypted" section after *END.

We have also encountered situations where parameters have been defined within an encrypted block, but referred to elsewhere (in clear), so from V13 onwards it is also possible to add parameter definitions to this post *END encrypted block.

The assumption is that these definitions will be simplified non-proprietary variations of the encrypted data that provide enough information for PRIMER to work normally.

The syntax of this block is:

***ENCRYPTED_START**	Denotes the start of the special "alternatibe definitions for encrypted data" block
***MAT_type(_*TITLE*)** *Title line if _TITLE suffix is used* <**label**> <**density**> (*<further data fields optional>)* (*further lines of data optional*)	Any number of material definitions, in any order, may be provided. The first two data fields on line 1 of the definition, usually <label> <density> ***must*** be provided. Further data fields may be provided, but will be set to zero if omitted.
***DEFINE_CURVE(_*TITLE*)** *Title line if _TITLE suffix is used* <**label**> (*<further data fields optional>)* (*further lines of data optional*)	Any number of loadcurve definitions, in any order, may be provided. The first field on line 1, <label>, must be provided. All further data fields are optional, and will be filled with zero if omitted. If no curve points are provided the two points (0,0) (1,1) will be inserted
***DEFINE_TABLE(_*TITLE*)** *Title line if _TITLE suffix is used* <**label**>	Any number of table definitions, in any order, may be provided. Nothing is assumed about the table contents, and by default it will not contain any loadcurves. This may generate warnings, so on the whole it is better to use curves to satisfy missing curve / table labels since, for the purposes of pre-processing, it won't matter which is used.
***PARAMETER** ***PARAMETER_EXPRESSION**	Any number of parameter definitions, in any order, provided that parameters referred to in an expression are defined before the expression itself. (This capability is from V13 onwards.)
***ENCRYPTED_END**	Denotes the end of this block

Here is an example of such a block:

```
*END
$
$ ===============
$ ENCRYPTED cards (simplistic info for encrypted items in main deck)
$ ===============
$
*ENCRYPTED_START
$
*MAT_ELASTIC_TITLE
This is a fabricated material definition
        22     7.85e3
$
*MAT_RIGID
        23       10.0
$
$
*DEFINE_CURVE_TITLE
This is a fabricated curve
        99
$
*DEFINE_TABLE_TITLE
This is a fabricated table
       199
$
$
*PARAMETER
```

```
R ENDTIM       750.0
$
*ENCRYPTED_END
```

The ***MAT** and ***DEFINE_CURVE** keywords are read in exactly the same way as they would be in the main part of the deck, and they are treated inside PRIMER in exactly the same way with the following exceptions:

- Material definitions are not checked during model check. This because populating them with zero values might be invalid, and it would be silly to generate errors because of this.
- Items defined in this way are "locked" against deletion. This is because they are presumed to be present inside opaque PGP encrypted blocks, and deleting them might lead to an inconsistent model, or alternatively lead to their labels being re-used resulting in a clash.
- Their labels are locked against being changed. Again, their labels are fixed inside opaque PGP encrypted blocks, so changing them could lead to conflicts. (This lock is not absolute: a determined user could change them by various means, for example by using Text Edit; however it is assumed that users will be sensible!)
- During keyword output they will not be written in the main deck, but rather after ***END** in this ***ENCRYPTED** section. If extra data fields beyond the minimum required were provided these will be written out, but otherwise only the bare minimum (as in the example above) will be written.

The ***PARAMETER** definition is read and processed in exactly the same way as ***PARAMETER** cards in the main body of the deck. However bear in mind that if a ***PARAMETER_EXPRESSION** refers to a parameter in an encrypted block, and you define the missing parameter after ***END** in the same file, then you will get an "out of sequence" error during keyin since the expression will not initially be able to be resolved. PRIMER will sort this out later when the deck is complete, but to avoid this problem you should consider carefully the order in which files are read.

Further considerations affecting encrypted *MAT and *DEFINE_CURVE definitions

Locking encrypted loadcurves against deletion.

It is usually the case that when encrypted materials and loadcurves occur in the same model that the materials refer to the curves. Unfortunately it is not possible to tell which, if any, materials do actually refer to which loadcurves, and if no action is taken it is likely that some or all loadcurves would appear to be unreferenced, and hence unused, and would be deleted by the standard PRIMER "cleanup" operation.

To prevent this PRIMER automatically assumes during deletion or cleanup operations that:

- For the partially encrypted definitions, where labels are provided in clear, all encrypted loadcurves are referred to by all encrypted materials, which has the effect of locking all such curves against deletion so long as there are any encrypted materials in the model.
- For the wholly encrypted case, where alternate definitions are provided after ***END**, both materials and loadcurves are locked unconditionally against deletion.

These locks can be a nuisance, but it is hard to see how else the problem can be dealt with and, in practice, most encrypted models will be effectively "read only" anyway.

Controlling material and curve labels

Although partially encrypted material definitions have their labels defined in their title line their "true" label is fixed inside their encrypted section and cannot be changed, so PRIMER must not change it either.

Partially encrypted loadcurves have their label field in clear, so at first sight it would seem that this label could be changed with impunity. However a moment’s consideration reveals that these curves are likely to be referred to from deep within the encrypted part of material definitions, so they too cannot have their labels changed.

Finally both materials and loadcurves defined in the alternate post-END definition represent items that have their labels fixed within their wholly encrypted sections in the main body of the deck, so they too must not have their labels changed.

Therefore PRIMER tries to lock these labels against change. However this lock is not absolute, and a sufficiently determined user will manage to change their labels, for example via the Text Edit function. Therefore take great care when changing labels in encrypted decks, since PRIMER cannot protect you from all potential errors. The following strategies are recommended:

The vendor of the encrypted deck may provide some mechanism for adjusting labels. Sometimes this is a simplifed (in clear) include file that can be used during model assembly, with the proper (encrypted) include file being substituted for the analysis.

Another alternative is to include encrypted include files within an **`*INCLUDE_TRANSFORM`** definition, and to use this to offset labels. This is usually a better solution since it effectively moves the point at which the actual renumbering occurs to somewhere inside the LS-DYNA keyword reader, where the encrypted data is then available in clear.

Editing encrypted definitions

The normal editing panels will work for encrypted curves and materials, and they will show that a minimal primitive amount of data have been inserted.

- If you edit partially encrypted definitions the edits will be accepted and remembered, but when the deck is written out the original encrypted data block will be re-inserted and any edited data ignored.
- If you edit alternate definitions in the post *END section these edits will be remembered and written out in that section.

Dealing with encrypted keywords other than *MAT, *DEFINE_CURVE/TABLE and *PARAMETER

The solutions described above work reasonably well, but they are limited to materials, loadcurves and tables and some encrypted input decks require other keywords to be provided in clear in order to function correctly.

From V11 onwards PRIMER supports the use of **`*INCLUDE`** files after the **`*END`** keyword, making it possible to include a simplified definition of ***any*** keyword for pre-processing purposes without this conflicting with the "true" encrypted data during analysis. There are problems with this approach: PRIMER cannot "know" that items in the file must not have their labels changed, nor can it know that it must lock such items against deletion.

However the flexibility that this method provides may outweigh these disadvantages. It is described in more detail in section 3.13.5: Reading include files after the *END keyword.

5.1.10 "Embedded" Keyword Comments

From release 10.0 onwards PRIMER reads, stores, edits and writes out comment lines embedded within keyword data. The following rules are used to define what is meant by "embedded", and also to limit and control which comments are stored. (Note that these are not the same as comments after a **`*COMMENT`** keyword.)

Rules for comment storage

PRIMER faces the problem that while it stores keyword data, and knows which include file this occupies, it does not "remember" where (at which line) a particular keyword occurred. In fact it may often re-order the data in a keyword deck when it writes it out. Therefore in order to be stored comments need to be associated with a keyword, and PRIMER takes this a stage further by remembering exactly which line in a keyword each comment is associated with.

Consider the following fragment of a keyword deck, which has been artificially separated into sections:

These comments are "free-standing", coming after the previous keyword.	`$`
	`$ The following parts refer to the seat.`
	`$ These were added on 11th May 2011`
	`$`

The comments here are "embedded" within the *PART definitions	`*PART`
	`sill_swan_neck`
	`$:  pid  secid  mid  eosid  hgid`
	`$ This part used to be steel, but is now aluminium`
	`$ Thas change was made on 2nd January 2010`
	`  5  3  5  0  0`
	`$ This part uses the original steel material`
	`  6  3  6  0  0`

These comments are "free-standing"	`$`
	`$ Ten further parts were moved to include file "floor.key"`
	`$`

The comment here is embedded within the *SECTION keyword	`*SECTION_SHELL`
	`$ This section has been increased from 1.0 to 1.2mm thickness`
	`  1  2  0.0  0`
	`  1.2  1.2  1.2  1.2`

There are three sorts of comment lines here:

"**Embedded**"
 	
Shown in **red**
 	
Saved by PRIMER
	These comments lie between the *Keyword header and a data line, or between sucessive data lines.
	
	PRIMER associates these with the line of data that ***immediately follows*** the comment, and saves them in that location on that keyword.
	
	For example the comments
	
	`"$ This part used to be steel ..."`
	and
	`"$ This change was made ..."`
	
	Are both associated with line 1 of Part 5. And the comment
	
	`"$ This part uses the original ..."`
	
	is associated with line 1 of Part 6. Any number of comments may be associated with any data lines, and they are stored as text strings in the order in which they appear.
	
	Comments stay with their definitions so if, for example, Part 5 in this deck was moved to a different include file it would take its embedded comments with it, and they would appear above the relevant data line.
	
	If the keyword data is modified so that its data lines are reformatted then comments still associate themselves with line N of the definition, regardless of what that might contain. In other words association with a given line of a definition is "dumb" and is not related in any way to the data that line contains.
	
	These comments can be visualised in PRIMER, and modified using the Text edit capability.

"**Special**" Shown in **blue** Ignored by PRIMER	If a comment line starts with a dollar followed immediately by a colon, "**`$:`**", then PRIMER treats this as a special comment that does not need to be remembered. For example the following line in the above is a special comment: `"$:  pid  secid  mid  eosid  hgid"` This syntax is used for data field headers and comment lines such as those listing the items referencing a loadcurve, since these will be regenerated automatically (and may change) each time a new keyword output file is created, so it does not make sense to store them. LS-PREPOST starts "special" comments with the alternative "**`$#`**" syntax. By default PRIMER will also treat these lines as special comments, ignoring them in the same way as lines starting "**`$:`**". To change this behaviour set the preference **`primer*ignore_lspp_comment`** to **`false`**.
"**Free standing**" Shown in **dark green** Ignored by PRIMER	These comments lie after the last data line of the previous keyword, and before the *Keyword header of the next one. PRIMER ignores these comments since they have no fixed association with a particular keyword. (An exception is made for comments at the top of a keyword deck / include file, which are stored as a property of that file and may be edited.)

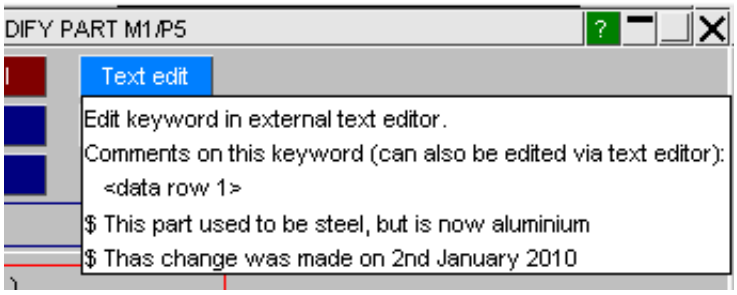
Visualising embedded comments

PRIMER allows you to visualise and edit comments in two ways:

On a scalar editing panel

The **Text edit** button will be shown in light blue, and hovering the cursor over that button will list comments for that keyword.

This example shows the result for Part 5 above.



On a Keyword editor row

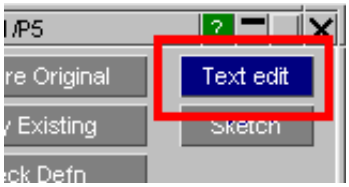
If the definition on a row contains comments then a light blue C will be shown on its row button, and hovering the cursor over that button will list them.

In both cases you can add, remove and edit comments with the external text editor by using the **Text edit** button. (In the keyword editor case this is one of the options in the right-click popup menu associated with the row button

5.1.11 **Text Edit** Editing keyword data externally.

You can export data to file in keyword form and then use a system text editor to view, modify and update one or more keywords. This also gives the ability to import data from other decks by "cut and paste" into the text editor session and then import back into PRIMER. Embedded keyword comments can also be visualised and edited using this capability.

Editing a single item



In any scalar (single item) editing panel you can use the **Text Edit** button to generate a mini-keyword deck containing just this keyword. It is generated from the scratch data currently populating the editing panel, and will contain any comments embedded in the card. (It is also possible to edit multiple items using Text edit on the keyword editor, this is described below.)

Note also that there is a general **Text Edit** button in the **Tools** panel that can be used to edit any keyword, not just one in an editing panel or the keyword editor. This is described in section 6.43**Text Edit**

An example is given below for a PART, and the default output is split into three sections as annotated here

Section	File Contents (continuous text in an actual file)

(1) Header	`$ Edit/view data and comments.`
	`$ ----------`
	`$`
	`$ Only lines after *keyword below will be reread by PRIMER. See RULES`
	`$ at the end of this file for details on data and comment editing.`
	`$`

(2) Data	`*PART`
	`sill_swan_neck`
	`$:  pid  secid  mid  eosid  hgid  grav  adpopt`
	`  5  3  5  0  0  0  0`
	`$`

(3) Rules	`$ RULES for editing data and comments in this file.`
	`$ ----------`
	`$`
	`$ Make any changes you wish to the data and/or comments and then save`
	`$ the file under its existing name to update the PRIMER definition.`
	`$`
	`$ Rules when saving changes:`
	`$`
	`$ If you simply 'quit' from this edit this file will be deleted and`
	`$ no changes will be imported back into PRIMER.`
	`$`
	`$ If you make changes to either data or comments then save this file`
	`$ under its current name your changes will be imported back into the`
	`$ current edit panel in PRIMER, and the file will then be deleted.`
	`$`
	`$ If you save this file under a different name, with or without any`
	`$ edits, that new file will remain on disk. Any changes will only be`
	`$ imported back into PRIMER if you also saved them to the original`
	`$ filename, and that original file will be deleted as above.`
	`$`
	`$`
	`$ Rules when editing comments for import back into PRIMER:`
	`$`
	`$ Only comments inserted between the *keyword header and data rows,`
	`$ or in between multiple data rows, will be 'remembered'. Comments`
	`$ before *keyword, and 'trailing' comments below data rows (such as`
	`$ these rules) will be ignored.`
	`$`
	`$ Comment lines (anywhere) starting '$:' will be ignored.`
	`$`
	`$`
	`$ To suppress the initial explanation and these RULES set the preference`
	`$`
	`$ primer*text_edit_show_rules: FALSE`
	`$`
	`$ To turn off data field headers in the file to be edited (setting does`
	`$ not affect the output of headers in normal keyout files) use`
	`$`
	`$ primer*text_edit_show_names: FALSE`

Controlling the output in the file

This default output is verbose and is intended for a 1st-time user. You can control the output using the following preferences:

`primer*text_edit_show_rules:`	**`true`** or **`false`**	Whether or not to show sections (1) and (3) above. Most users who have become accustomed to the rules explaining how this works will wish to use this preference to cut output down to just section (2) only.
`primer*text_edit_show_names:`	**`true`** or **`false`**	Whether or not the field headers (here `pid`, `secid`, etc) are shown above each row of data.

Controlling which text editor is used.

By default PRIMER will attempt to use the default system editor.

- On a Unix/Linux system this may be an editor built into the Window manager system, or if the environment variable $EDITOR is set then this will be used.
- On a Windows system PRIMER will try Notepad or Wordpad if no explicit editor is defined.

On any operating system you can override these defaults by setting the following preference:

`primer*text_editor:`	*<pathname of editor>* for example **`C:\Program Files\Vim\Vim70\gvim.exe`**

Rules for using the text editor

The text editor operates as follows:

- A temporary filename containing the keyword output is generated in the directory containing the model. If this cannot be opened, for example because that directory is read-only, then the file is created in a temporary directory instead.
- The relevant editor is launched to read this file. This is done in a separate thread so that the editor process is autonomous and does not "lock" the PRIMER session in any way.
- You are free to modify the temporary file in any way you please, or to read new content into it to replace the existing. You can also save copies of the file under different names, and generally do anything you would normally do in an editing session, including aborting the session without changing anything.
- When you finally exit from the editor the following happens:
  - Any files other than the temporary file that you might have created (by "Save as...") are left unchanged on disk, and are not considered or read in any way.
  - If the original temporary file is unchanged - specifically if its "last modified" time has not changed since its creation - then no further action is taken and it is deleted.
  - If the original temporary file has changed then it is read back into PRIMER via a "mini keyword read" process, and the contents of the file will overwrite the scratch definition in the editing panel. The temporary file is then deleted.

    Inside PRIMER this is done by reading all the file contents into a scratch model, then extracting the first definition of the relevant type (here *PART) from that scratch model and using it to replace the definition being edited. So if you import more than a single definition, or perhaps load different keywords from some other file, all except the first definition encountered of the relevant keyword will be ignored. If there are no keywords of the relevant type (here *PART) in the file, then PRIMER will issue a warning message, the file contents will be discarded and no changes will be made to the definition currently being edited.

    Therefore you can use this method as an alternative way to update keywords, perhaps by cutting and pasting text in the editor from other files. However it is restricted to changing only the current keyword definition, and cannot be used as a generalised method of importing unrelated keywords into the model.

    If you want to read "fragments" of some other deck into this model it is better to use **Model Read** to read these into a separate model, then **Model Merge** to merge them into this model. This will permit input of any keywords, and the merge operation will protect you against label clashes.

Editing multiple items using the Keyword Editor

The example above describes how a single definition can be edited from a scalar PRIMER editing panel. It is also possible to **Text edit** multiple items by using the same function from the generic Keyword Editor described above.

This works in exactly the same way, and is subject to the same rules and limitations below, except that

- Multiple items can be exported to the external keyword file by selecting several rows for editing. In this example rows 2 to 14 inclusive will be exported for editing.

- Multiple items can be re-imported back into the model. Where labels in the external file match existing items in the PRIMER database then those items' definitions will be replaced, where external labels do not match an existing definition then a new definition will be created. The number of items read in does not have to match the number exported.

  Therefore it is possible to import multiple new entries into the model by this method, however the limitation that only those items of the current *Keyword will be considered still stands so, once again, this is not a generalised way of importing new model data.

Consequences of Text Edit being an autonomous process

Because the text editing session is an autonomous process in a separate thread it does not "lock" (freeze) the PRIMER session in any way: in computer-speak the PRIMER process and its child text editing session are "asynchronous", and this has some consequences:

- If you open a PRIMER editing panel, launch the text editor, and then close the editing panel while the editor is still running there is no longer a "scratch" definition to be updated when you finally close the editor. PRIMER detects this situation, warns you and simply throws away any changes in the edited file without updating the model in any way.

- If you record a Macro that includes Text Editing sessions the macro will not "remember" when a particular session ended, and in fact it has no way of knowing what - if anything - happened in the text editor, or when. This means that when the macro is played back there is no way of synchronising text editor sessions with operations inside PRIMER, so if a macro implicitly contains the sequence:

  - User starts editor panel in PRIMER
  - User launches text editor to modify the scratch definition
  - User exits text editor and changes are imported back into PRIMER

  The macro will not work as expected, since the 3rd step of editing text editor and importing changes will not take place (because the macro cannot replace the user and drive the external editor).

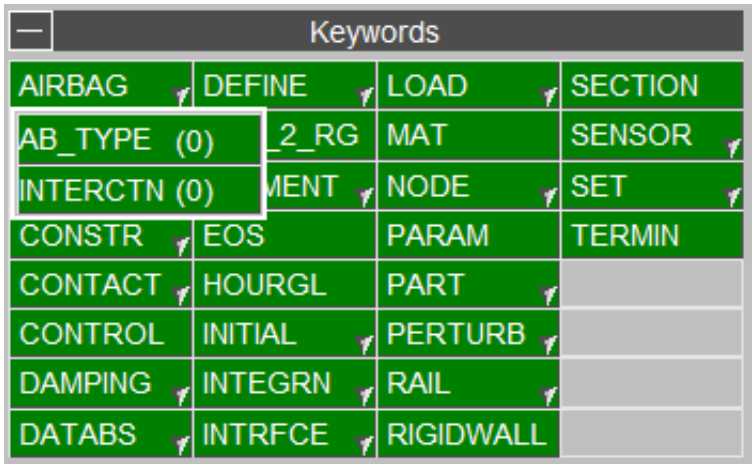
  For this reason it is recommended that you do not embed text editing sessions in macros

AIRBAG: Airbag (Control Volume) definitions.

- **AIRBAG_<type>** sub level menu
- Creating a new definition
- Copying a definition
- Editing an existing definition
- Deleting airbag definitions
- Other operations
- Visualising airbags.
- **<_INTERACTION>** menu

"Airbags" are definitions which determine the gas pressure inside an enclosed volume by applying one of a range of gas expansion laws. This pressure then acts upon the elements that form the volume, generating pressure loads. The gas pressure is a function of incoming and outgoing gas flow, its thermodynamics and the size of the volume; the pressure is updated during the analysis as these parameters change and interact.

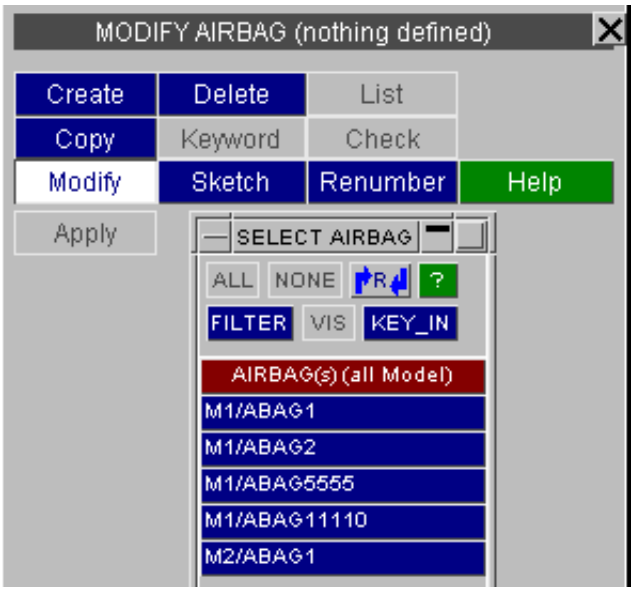
The name "airbag" is really a misnomer as they don't have to be used solely for vehicle airbags (which don't contain "air" anyway). They can, for example, be used to model the pressure inside tyres or indeed any structure where changes in volume may affect internal pressure. The term "control volume" is better, and is in fact used in formatted LS-DYNA input.



Select the sub menu desired using the AIRBAG popups in the KEYWORD menu.

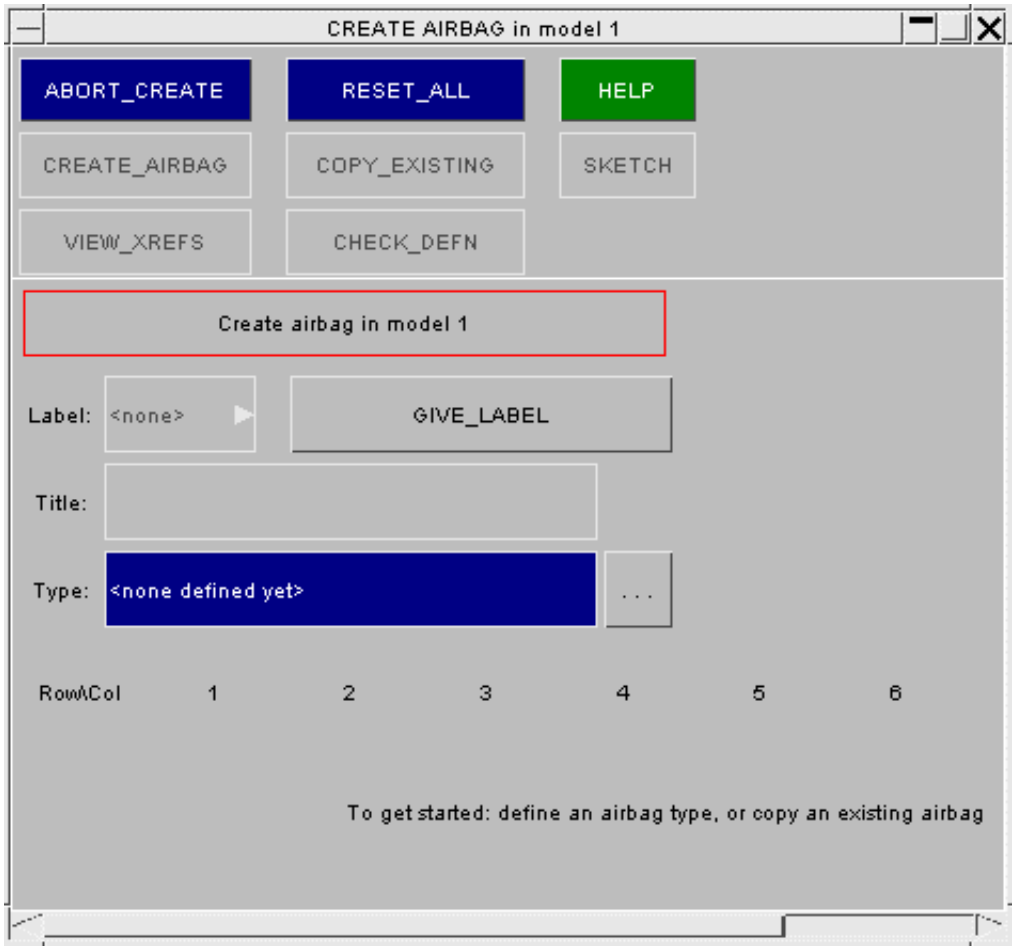
AIRBAG

This figure shows the main **AIRBAG** menu as selected from the Keywords panel. The functions currently available have their standard meanings (see section 5.1.1). Greyed out functions are not currently available:



CREATE Making a new airbag definition.

This figure shows the standard **CREATE**/**EDIT** panel for airbags. Here **CREATE** has been used, so a blank airbag creation panel is displayed.



The static buttons in the top section of the panel have functions which are common to the other editing panels within PRIMER.

Only once sufficient data has been input by the user will the **CREATE_AIRBAG** button become active. Until that time it will remain greyed out.

The data on the panel is as follows:

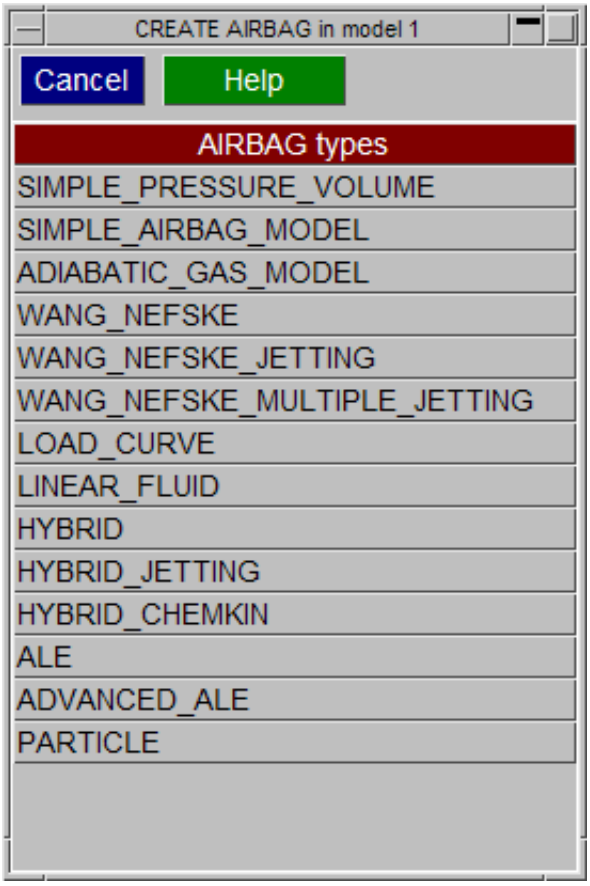
LABEL This is an internal label used by PRIMER and is not written out in the keyword deck. It is set to the first free airbag label within the model as a default. This value can be changed if necessary though existing labels cannot be overwritten. The popup window allows the **FIRST-FREE** label or the **HIGHEST + 1** label to be automatically selected.

GIVE_LABEL This button allows the optional numeric ID to be used. (Airbag definitions in LS-DYNA do not have to have an explicit number, although PRIMER requires this.) If an internal **LABEL** is already set then this is used by default. A new **LABEL** is selected as described above.

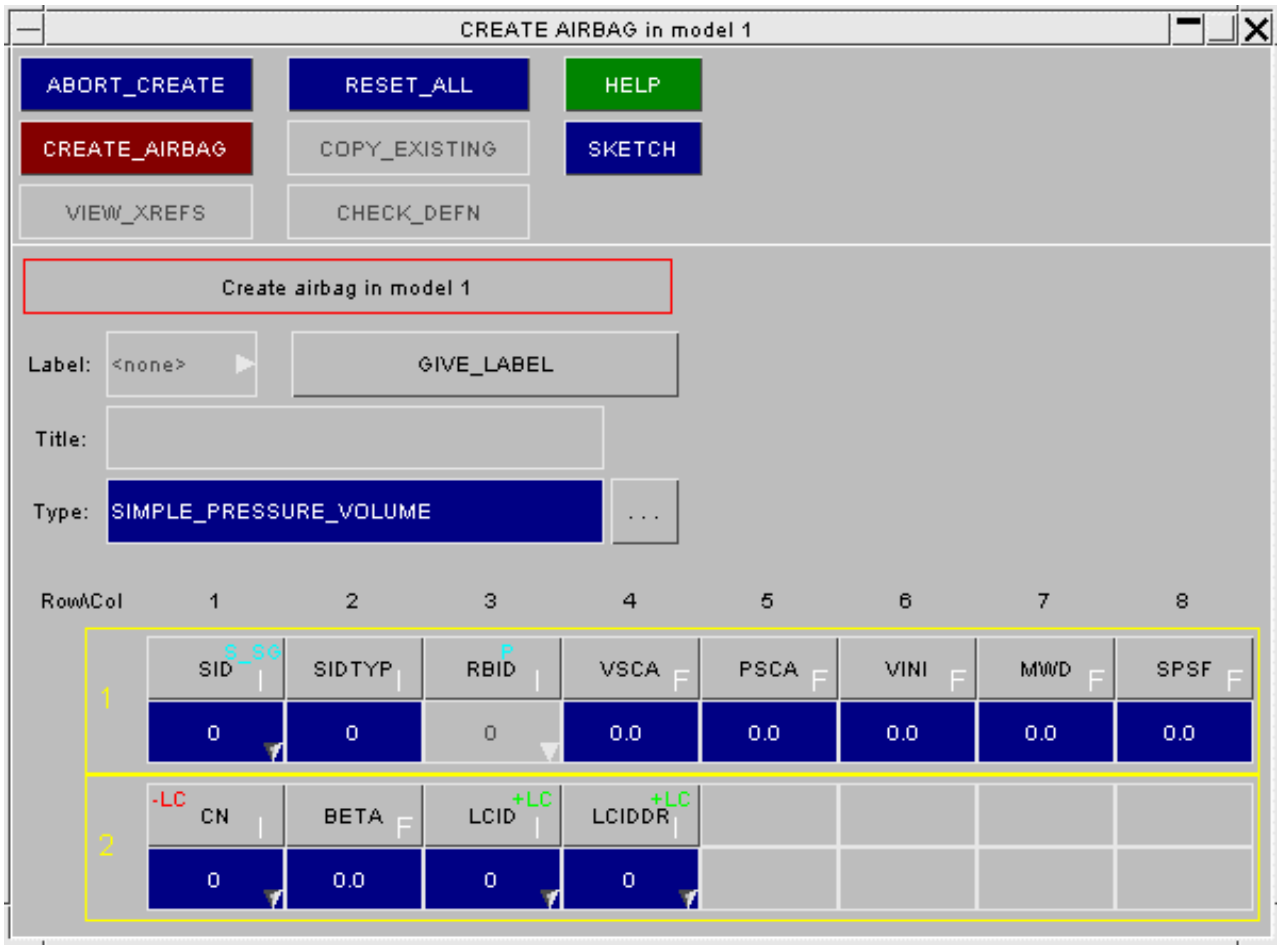
TYPE The airbag type is defined using this button. The [...] Shortcut button can be used to browse through a list of airbag types. The browse panel is shown in figure below.

ROW/COL The data relevant to each airbag type is displayed in row and column format identical to that of DYNA keywords

To start creating an airbag you must first define the type. You can type in a standard keyword if known, or invoke the selection menu in this figure with the [...] button.



Once the material type has been defined the keyword data will be displayed on the panel, as shown in figure below. Initially all values will be set to zero.



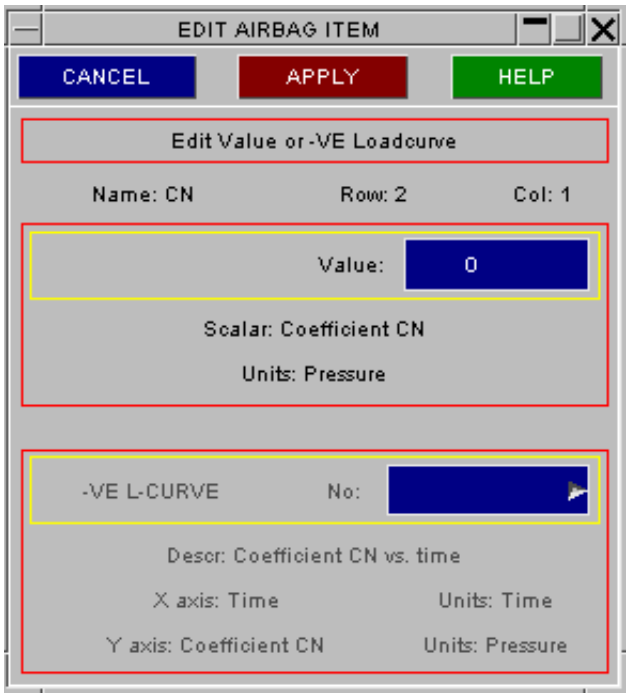
The airbag data can then be typed into the relevant boxes. The expected data type is indicated on the grey button, which also shows the acronym for that data value:

F	White **F** floating value
I	White **I** integer value
+LC	Green **LC** +ve loadcurve
-LC	Red **LC** -ve loadcurve

(Note that the component **'RBID'** is not available for editing. This is required for user defined activation subroutines which are not yet enabled from this panel.)
Information about each individual data component can be requested by pressing the grey data component button. For

example; to request information about data component '**CN**' (2nd row, 1st column) press the grey button with **CN**. This will create a new sub-window with detailed information about the data component showing:

- A one-line description of it;
- Its current units type
- Its current value.



Once all of the data has been input on the airbag card, **CREATE_AIRBAG** installs the airbag permanently in the model.

COPY Copy existing airbag(s) to make a new airbag(s).

COPY makes new airbags, in the same model(s), that are identical to their originals apart from their labels. By default only the airbag definitions themselves are duplicated.

Where **RECURSIVE COPY** is requested, all items associated with that airbag (i.e. elements, parts, etc.) are also copied.

MODIFY Modifying the attributes of an existing airbag.

MODIFY functions in the same way as **CREATE**. Obviously, the airbag type will already have been selected so the panel will resemble that shown above.

Any modifications made to the airbag definition will not be made permanent until the **APPLY_MODIFY** button is pressed. At this point a the local copy which has been updated is used to overwrite the version in the model.

DELETE Delete existing airbag definitions.

The **DELETE** function deletes the selected airbag. Two switches may be used:

DELETE_RECURSIVE	Also elects for deletion any segment sets, their segments and associated nodes that are used uniquely to define this airbag. Where airbags are defined by Part or other non-unique "structural" methods these items are not selected for deletion.
REMOVE_FROM_SETS	Is needed if segments are to be deleted as it stops their definition in sets "locking" them against deletion.

SKETCH Sketch elements used by an airbag on the current image.

SKETCH sketches on top of the current image the parts and elements that are referenced by the selected airbags.

LIST Not yet active

CHECK Not yet active

RENUMBER Renumbering airbag labels.

RENUMBER lets you change any or all airbag labels within a given model using the standard renumbering panel.

To change the label of an individual airbag it may be simpler just to **MODIFY** it.

Visualising Airbags

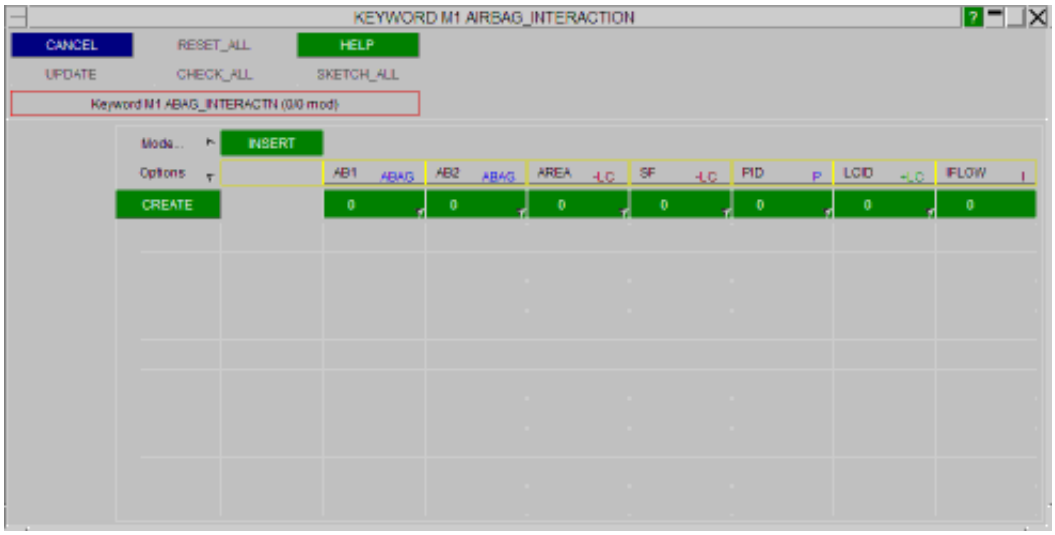
Airbags can be drawn (activate in the **ENT**ity Viewing panel). Alternatively,

- **SKETCH** it, or
- **MODIFY** it and sketch it.

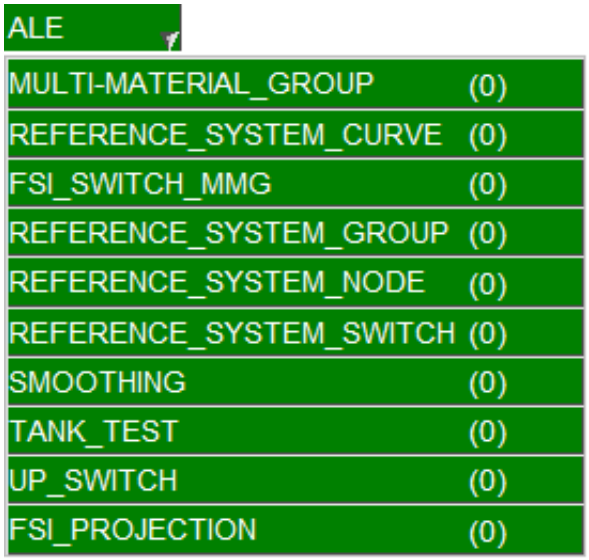
It will be drawn in terms of the parts and elements, or segments if defined by **SET_SEGMENT**, that defines it.

*AIRBAG_INTERACTION

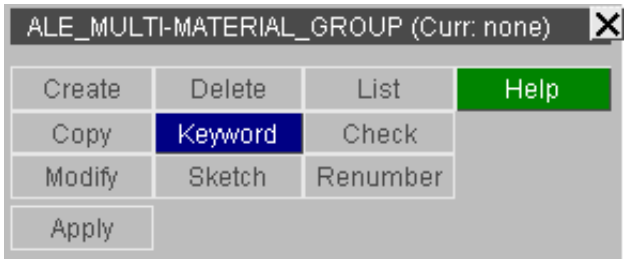
This is available through the standard keyword editing panel in section_5a.



ALE



There are currently seven sub-keywords available as shown on the left. Currently these can be edited through the generic Keyword Editor.



Sanity checks for *ALE_MULTI-MATERIAL_GROUP labels

This keyword is unusual in LS-DYNA in that each row of **`<sid> <idtype>`** forms a new definition, and this definition has an internal label that is its order of occurrence in the input deck. Therefore when a model contains N definitions these will always have internal labels 1 to N. Here is an excerpt from the LS-DYNA user manual for this keyword that illustrates this:

```
*ALE_MULTI-MATERIAL_GROUP
         1         0
         2         0
         3         0
         4         0
```

$$\leftarrow 1^{st}\ line = 1^{st}\ AMMG \Rightarrow AMMGID = 1$$
$$\leftarrow 2^{nd}\ line = 2^{nd}\ AMMG \Rightarrow AMMGID = 2$$
$$\leftarrow 3^{rd}\ line = 3^{rd}\ AMMG \Rightarrow AMMGID = 3$$
$$\leftarrow 4^{th}\ line = 4^{th}\ AMMG \Rightarrow AMMGID = 4$$

These labels, **`AMMGID`** in the lines above, matter because they are referred to on other **`*ALE`** cards, **`*DATABASE_ALE`** and **`*SET_MULTI`** so PRIMER has to manage them meaning that this is a "labelled" keyword. However it is important that these labels are not changed during renumbering and remain in sequential order with no gaps.

This means that PRIMER must assume that if **`*ALE_MULTI-MATERIAL_GROUP`** N exists then definitions 1 to N-1 also exist or will be created, so if asked to process definition label N PRIMER will generate any missing definitions in the sequence 1 to N-1 as latent items. This is exceptional behaviour, normal "arbitrarily labelled" keywords permit gaps in their label sequence, it is only this particular keyword that presents problems.

Clearly this could consume large amounts of memory if the user accidentally types in a large label, as the following example shows:

- Model contains two **`*ALE_MULTI-MATERIAL_GROUP`**s, labels 1 and 2
- User is editing **`*SET_MULTI`** and types in 1003 instead of 3.
- PRIMER will then generate latent definitions 3 to 1002 before creating the specified definition 1003

In this example the problem is not too serious, as ~1000 latent definitions do not consume too much memory, but for larger labels this has the potential to consume all the memory on the machine and to "hang" the PRIMER process.

It would be an extremely unusual model that contained more than a handful of these cards, so the following sanity checks have been implemented when labels for **`*ALE_MULTI-MATERIAL_GROUP`** are processed.

Purpose	default	Environment variable to change it	Applies to
Trigger warning message	100	**`PRIMER_AMMG_WARN`**	Interactive and batch input, plus keyword read.
Trigger input error	1000	**`PRIMER_AMMG_MAX`**	Batch input and keyword read only.

During interactive input a label of any size will be accepted as input, but a label greater than the "warning" value (default 100) will trigger a warning message requiring you to confirm this.

During batch input, or when reading a keyword file, a label greater than the "warning" value but less than the "maximum" value (default 1000) will trigger a warning message, and a label greater than the "maximum" value will be rejected as an error. Explanatory messages are printed in both cases.

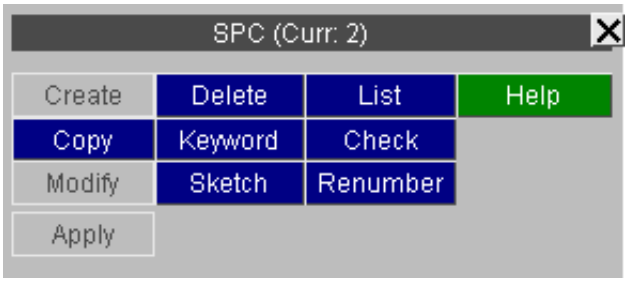
Both of these limits can be changed by setting the appropriate environment variable in the table above to some other value, however note that an environment variable will only take effect in future PRIMER sessions, it will have no effect on the current session.

BOUNDARY: Defining boundary conditions.

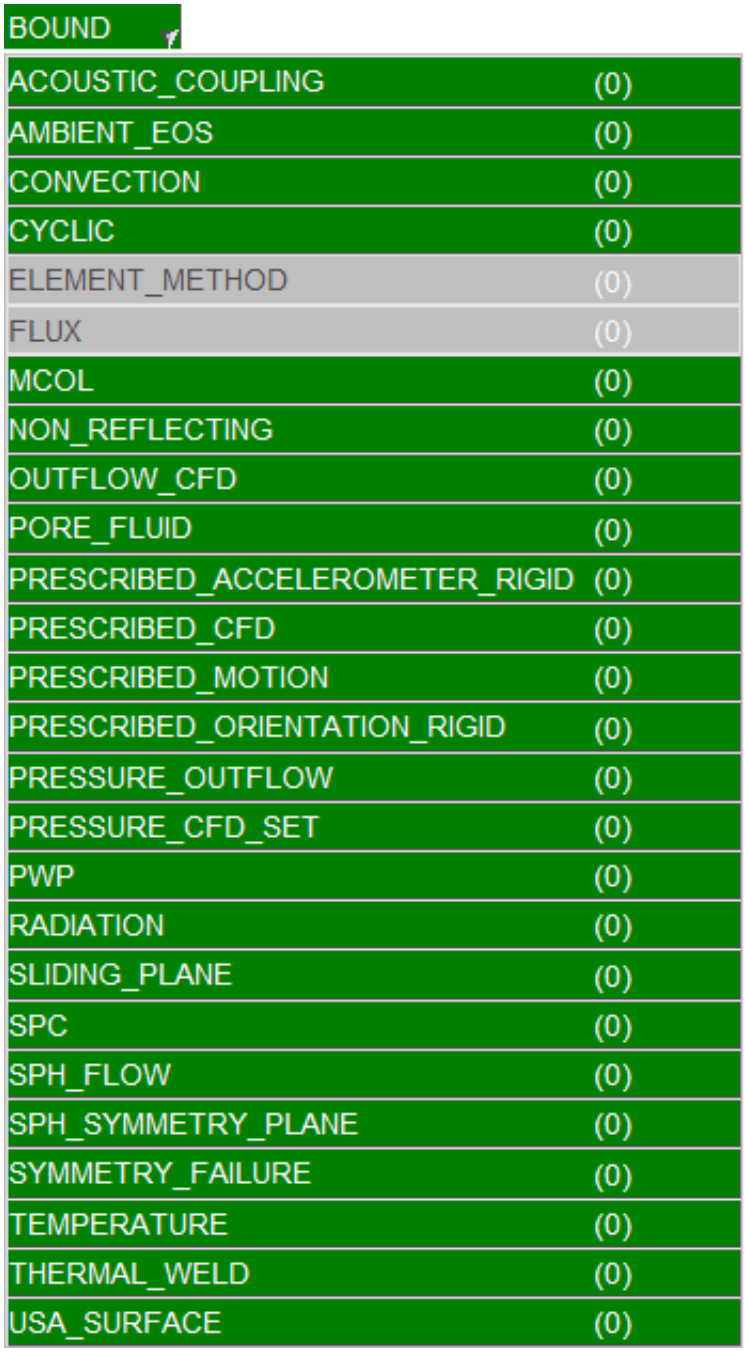
- Selecting the *BOUNDARY sub-keyword
- Explicitly drawn sub-types
- Drawing of other sub-types
- Labelling of *BOUNDARY items

Boundary conditions within LS-DYNA apply a range of restraints and other imposed conditions to models.
All *BOUNDARY sub-keywords except *BOUNDARY_ELEMENT are editable within PRIMER. (Boundary elements do not logically belong here: really they merit their own section since they imply a totally different type of analysis.)

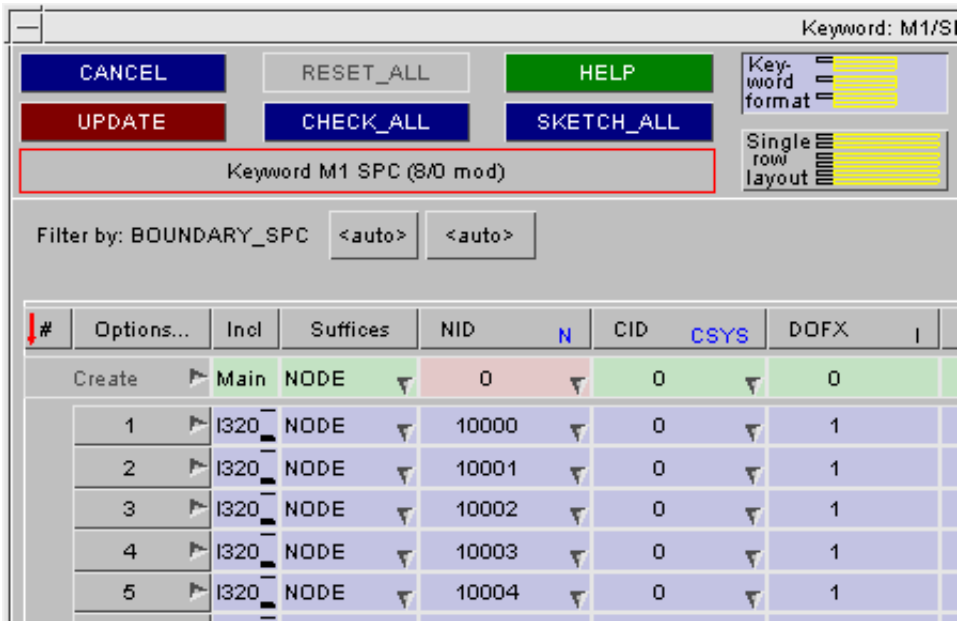
***BOUNDARY** cards can at present be edited only with the generic "Keyword" editor: no specific Create/Edit panels have been written yet.
All ***BOUNDARY** keywords except **_ELEMENT_METHOD** may be edited in this way.



The other commands (**COPY, DELETE,** ...) function in the standard manner described in section 5.1.1.



All of the boundary keywords are selected from the pop-up menu produced after Boundary is selected in the Keywords panel.



This shows an example of the Keyword editor for ***BOUNDARY_SPC**.
There are two sub-keywords: **_NODE** and **_SET**, with different formats.

The **_SET** variant is being edited here.

Drawing *BOUNDARY items.

These definitions can be viewed using the **ENT**ity viewing > **BOUNDARY** options. At present only the following keywords are fully visualised:

`*BOUNDARY_SPC` Restraints ("single point constraints") at nodes.

Restraint codes are drawn as vectors in the relevant X, Y or Z directions, using the colour scheme:
X Red

Y Green

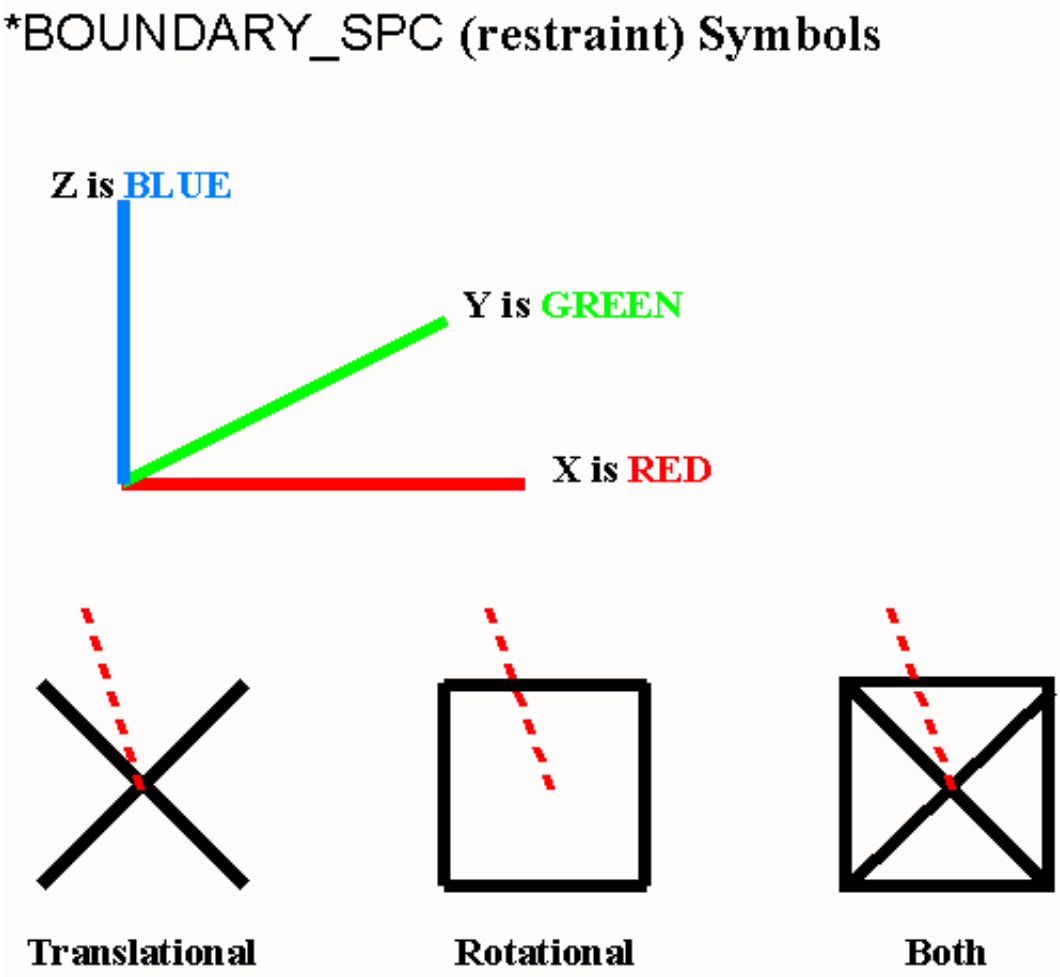
Z Blue

The symbols used at vector ends are:

Trans: Cross

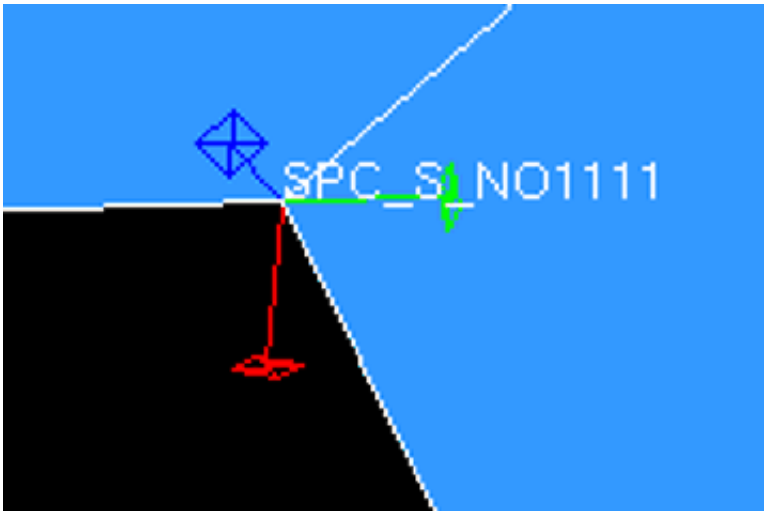
Rot'l: Square

Both: Cross + Square



Symbols are drawn at every restrained node.

This example shows a node which has been fully restrained in all of X, Y and Z, both in translation and rotation.
Labels have been turned on for this, and they show that this node is restrained via node set 1111.

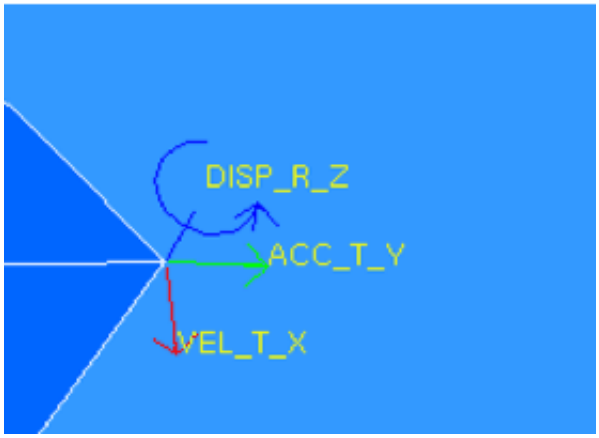


*BOUNDARY_PRESCRIBED_MOTION

This is visualised as

- An arrow in the relevant direction, coloured (X=red, Y=green, Z=blue). For rotational motion an arrow circling the relevant vector is used.
- A description at the arrow head, eg "**VEL_T_Z**" for Z translational velocity.

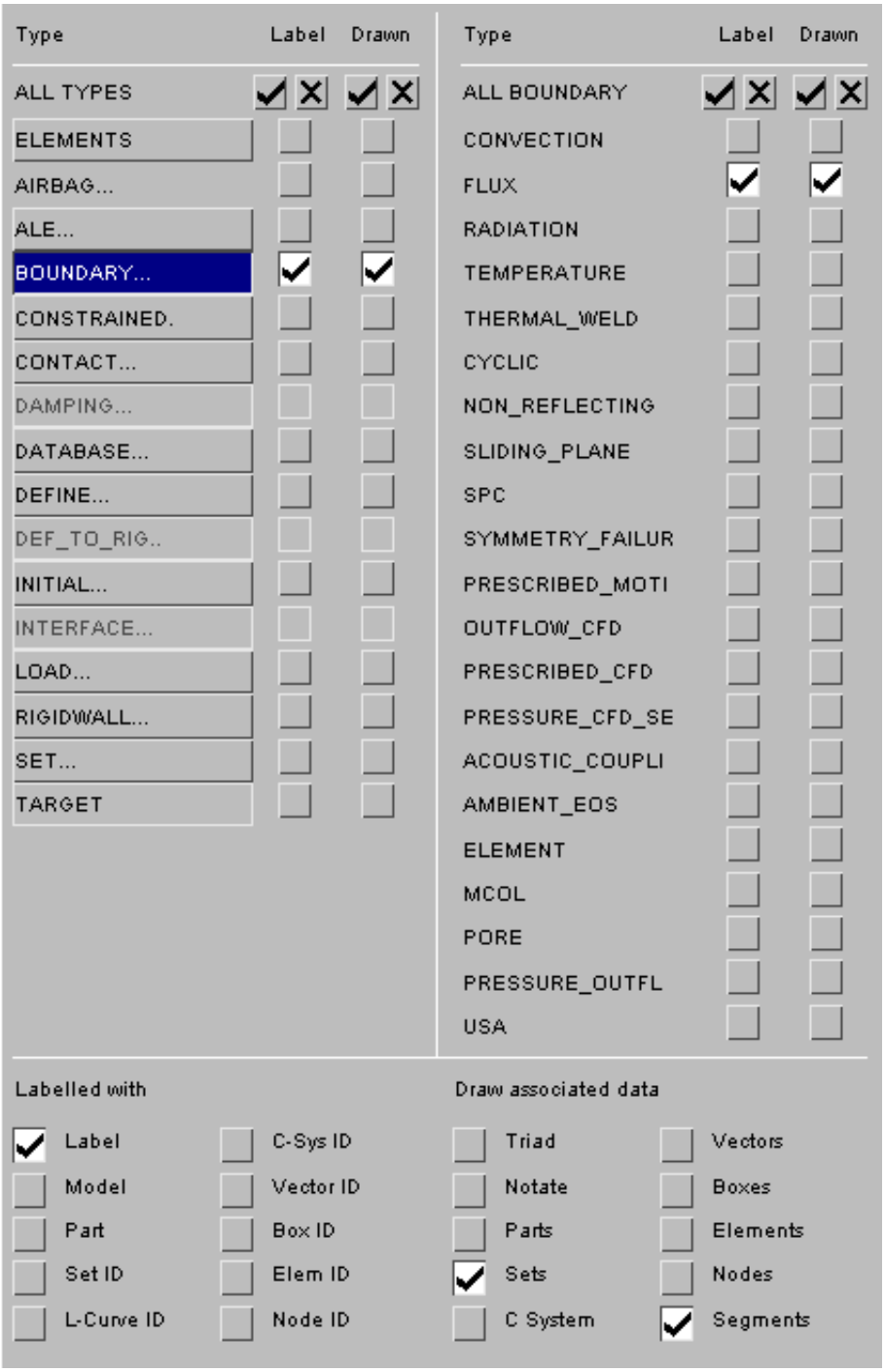
This example shows:
- Translational Acceleration in Y
- Translational Velocity in X
- Rotational Displacement in Z



All other ***BOUNDARY** sub-keywords:

Are visualised only in terms of the components that they reference: sets, elements, nodes, etc. Turn on the relevant items in **ENT**ity viewing to see these.

Turning on the relevant *BOUNDARY sub-keyword labels will annotate them correctly.

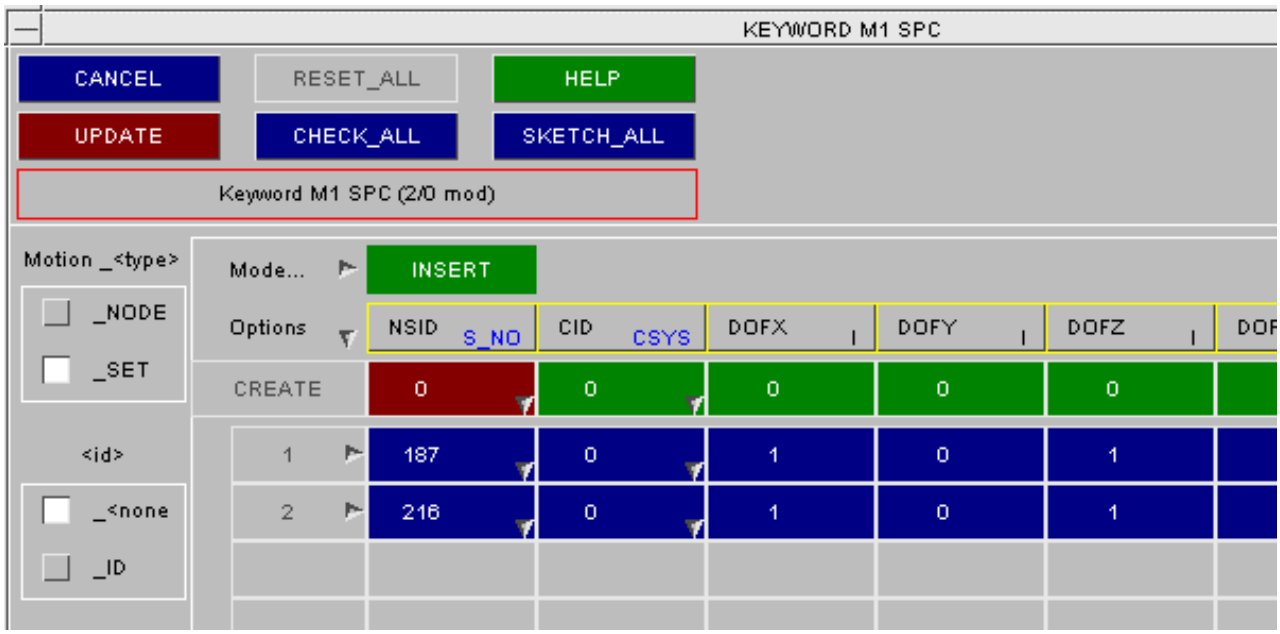


Labelling of ***BOUNDARY** items within PRIMER.

LS-Dyna keyword input has optional labels for *BOUNDARY items: the conversion from "keyword" to "formatted" input that precedes every LS-Dyna analysis converts them from discrete definitions to attributes applied to other items.

For internal consistency, for items not already labelled in the input model, and for items that cannot be labelled in LS_DYNA, PRIMER assigns new labels to everything that can be defined "once or many times", so *BOUNDARY definitions are given labels based on their order of appearance in the keyword input file.

Where LS_DYNA offers optional labels, (e.g. *BOUNDARY_SPC_ID versus *BOUNDARY_SPC), the labelling option is invoked in the keyword editor by selecting option "ID" (see below).

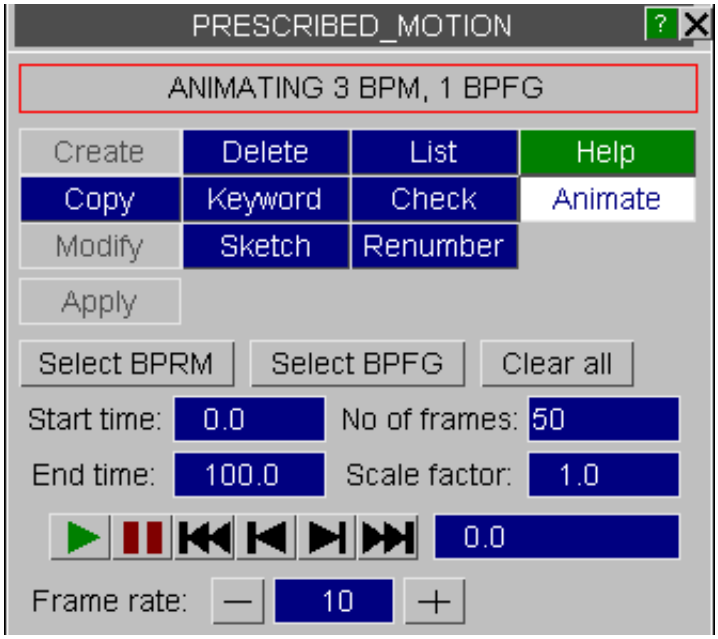


PRIMER's new labels; where generated:

- May safely be ignored - you don't have to worry about them if you don't want to!
- Are treated sequentially, starting at 1. (Thus **BNDY_1, BNDY_2, ... BNDY_n**)
- Are not grouped by sub-type: **BNDY_1** might be an **SPC, BNDY_2** a prescribed motion - they are based solely on the order in which they appear in the input deck. Each *BOUNDARY definition encountered gets the next label in the sequence.
- Are used in selection menus (eg for blanking, deletion, etc). Are also used in the output deck when defining what is referenced by what.

Animation of `BOUNDARY_PRESCRIBED_MOTION` and `BOUNDARY_PRESCRIBED_FINAL_GEOMETRY`.

Primer can animate prescribed motion as defined by *BOUNDARY_PRESCRIBED_MOTION and *BOUNDARY_PRESCRIBED_FINAL_GEOMETRY cards. Any number of the aforementioned cards may be selected for animation.



The following *BOUNDARY_PRESCRIBED_MOTION options are currently supported:

- Keyword options _NODE, _SET, _RIGID, _RIGID_LOCAL
- DOF options 1, 2, 3, 4, -4, 5, 6, 7, 8, -8
- VAD options 0 (velocity), 1 (accleration), 2 (displacement)

BOUNDARY_PRESCRIBED_MOTION and BOUNDARY_PRESCRIBED_FINAL_GEOMETRY_CARDS may be selected using appropriate buttons in the 'Animate' panel. Users may then play the animation in a continuous loop using the 'Play' button or may choose to navigate through specific frames or time instances manually.

Additional factors such as start and end times, frame rate and number of frames can be controlled using appropriate text boxes.

CASE

Mini tutorial on *CASE in LS-DYNA and PRIMER

Creating and editing a master *CASE
Creating and editing child (sub) cases

The master *CASE summary panel (choosing which master *CASE is current, and other operations)

Other operations affected by *CASE

Limitations

Version history

PRIMER 11	Support for the **`*CASE`** keyword was added but with some limitations, notably that it could not be used in conjunction with **`*INCLUDE_TRANSFORM`** or **`*DEFINE_TRANSFORMATION`** if encountered inside **`*CASE`**. In addition the generic keyword editor did not show, or permit changes to, the current case membership of an item.
PRIMER 12	These limitations have been removed. **`*CASE`** can be used with any keyword type, and there is now a "Case" column in the keyword editor if cases are present in the model.

Maximum numbers of *CASE definitions in PRIMER:

- There is no limit to the number of Master **`*CASE`** cards.
- The following limits apply to child cases:

  - The maximum number of child cases in a model is 4095 (but their labels can have any value)
  - The maximum number of unique combinations of child cases is 1023
  - The maximum number of child cases in a combination is 1023

  (The concept of a "**combination**" of child cases is explained below)

Using *CASE inside include files

When the **`*CASE`** keyword first appeared in LS-DYNA no mention was made in the keyword manual about limitations upon where it could be used, and in particular no restriction upon its use in include files was mentioned. Therefore PRIMER 11 permitted its use in both master deck and any include files.

However the LS-DYNA 971 R7 manual (2013) states that the **`*CASE`** keyword cannot be used inside include files. This date coincides with the development of PRIMER 12, so rules forbidding its use inside include files have been added to that version, and using **`*CASE`** in any file other than the master deck will generate an error. This is a "soft" restriction, so if LS-DYNA ever lifts this limit in the future a simple switch in PRIMER will also permit it.

A mini-tutorial on *CASE

This section gives a summary of how the **`*CASE`** keyword is used in LS-DYNA, and how it is implemented in PRIMER. It also describes the terminology and colour scheme used in this section and in the online help texts inside PRIMER.

Please read this before using **`*CASE`** in PRIMER as it explains some important concepts that you will need to understand.

How *CASE is used in LS-DYNA

The **`*CASE`** keyword is used in two related but distinct ways in LS-DYNA input decks:

`(1) *CASE` (with no further suffixes)	Referred to as a "**Master *CASE**" in PRIMER
This can be thought of as a "collector" keyword which owns one or more sub-cases (or "**Child cases**" in PRIMER) • It is a free-standing keyword, just like any other in the deck. • It has label which must be unique to this master **`*CASE`** • It contains one or more child cases • It may not itself be contained inside a **`*CASE`** keyword • It may only be used in the master keyword deck, and not in include files.	

`(2) *CASE_BEGIN_`*nnn ... other keywords **`*CASE_END_`*nnn**	Referred to as a "**Child case**" in PRIMER
These keywords define what is contained within child (sub-) cases. They always come in "begin ... end" pairs that surround other keywords, suffix ***_nnn*** giving their child case number ***nnn***. • These are not free-standing keywords, and they have no autonomous existence in the input deck. • The label ***nnn*** defines this child case uniquely. • Child case labels are totally independent of Master **`*CASE`** labels, thus it is legal to have Master **`*CASE`** 1 and also child case 1. • The child case begin ... end pair of keywords may occur any number of times within a deck • Child cases keywords may be disjoint or nested. • They may only be used in the master keyword deck, and not in include files.	

Therefore a keyword deck might contain 10 child cases labelled (say) 101 to 110, and these can be used in arbitrary combinations within Master **`*CASE`**s. For example:

- Master **`*CASE`** 1 might contain child cases 101, 102, 103
- Master **`*CASE`** 2 might contain only child case 110
- Master **`*CASE`** 3 might contain child cases 101, 105, 108
  ... and so on.

When analysing an input deck containing **`*CASE`** LS-DYNA does the following:

- Splits the input deck into N separate sets of files, where N is the number of Master **`*CASE`** cards.

- Each file contains:
  - the keywords that are not in any case, referred to as being "**in the main model**" in PRIMER
  - plus those defined within the child cases contained with the relevant Master **`*CASE`** definition.

- Then N separate analyses producing N separate sets of results are run.

More information can be obtained from the **`*CASE`** section of volume I of the LS-DYNA keyword manual.

How *CASE is implemented in PRIMER

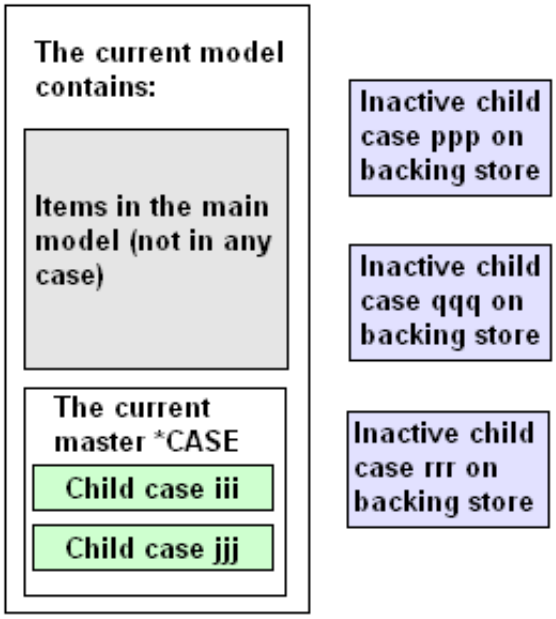
PRIMER handles this problem by splitting items up into three categories:

(1) Items in the main model	These are not inside any **`*CASE_BEGIN ... *CASE_END`** keywords, and occur exactly once.
(2) Items in the currently active Master **`*CASE`**	Items within the child **`*CASE_BEGIN ... *CASE_END`** keywords of all child cases in the currently active Master **`*CASE`**
(3) Items that are inactive and in backing store	Items within child **`*CASE_BEGIN .. *CASE_END`** keywords of child cases that are not referred to in the currently active Master **`*CASE`**

When PRIMER is dealing with models that contain one or more **`*CASE`** cards it requires that a Master **`*CASE`** is current at all times, and only items in this master **`*CASE`** and in the main model are visible and available for editing. In order to work with items in backing store you must first make the master **`*CASE`** containing them current.

This organisation is expressed in the diagram on the right.

The user may make a new master **`*CASE`** current at any time, but this is quite an expensive process since it requires a lot of internal processing (see "Changing the current Master *CASE" below) so it is best done only when necessary.



The "Backing store" referred to above is still in main memory, not on disk, nevertheless changing the currently active Master **`*CASE`** can be quite a slow operation when a model is large.

How PRIMER handles multiple definitions of the same item in different cases

Consider the following two sequences of keywords, each of which define versions of *NODE label 1.

Two separate definitions

```
*CASE_BEGIN_1
*NODE
       1    1.0       2.0       3.0
*CASE_END_1
*CASE_BEGIN_2
*NODE
       1    2.0       3.0       4.0
*CASE_END_2
```

"**Aliases**" in PRIMER

Single definition in combination

```
*CASE_BEGIN_1
*CASE_BEGIN_2
*NODE
       1    1.0       2.0       3.0
*CASE_END_2
*CASE_END_1
```

"**In combination**" in PRIMER

In this left hand example we have two separate definitions of node 1 (here with different coordinates): one in child case 1 and one in child case 2. It is clear that changing one definition, for example in child case 1, should not have any effect on the definition in the other child case.

In PRIMER the two definitions are referred to as being "**aliases**" of one another.

In this right hand example we have a single definition of node 1 that is common to child cases 1 and 2. It is clear that editing the definition, say changing a coordinate, should result in a change in both child case 1 and child case 2.

In PRIMER the two definitions of Node 1 are referred to as being "**in combination**" with one another.

Clearly the ***alias*** definitions on the left above must be stored separately since they are independent, and editing one definition will not make any changes in the other.

However PRIMER also stores each ***combination*** definition separately as well. This is necessary because, despite being in combination, the two definitions of node 1 in the right hand column above might have different locations in space or boundary conditions if parameters are used to define those data fields, or perhaps if the node is moved by a transformation. However if a definition that is in ***combination*** is edited within PRIMER then all copies of it are also

edited, whether current or in backing store, to preserve their combination status.

Also note that in a ***combination*** of cases the order in which the child cases are defined is irrelevant, so the two keyword sequences below are the same in PRIMER:

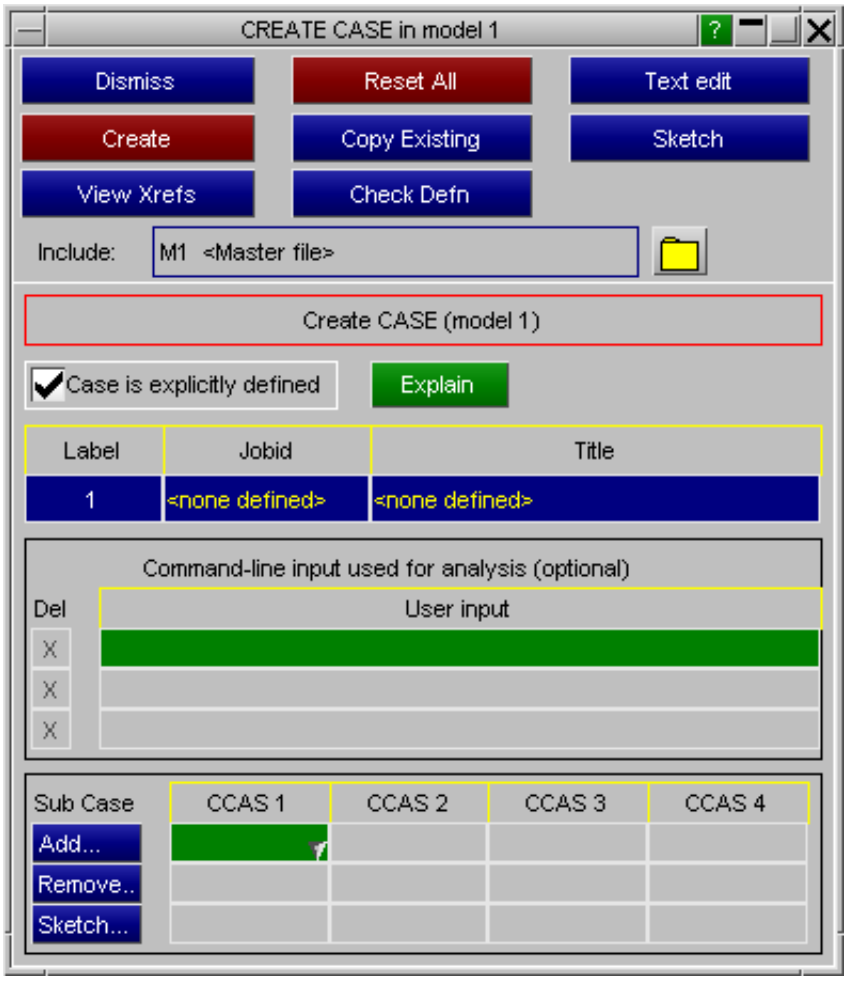
```
*CASE_BEGIN_1
*CASE_BEGIN_2
*NODE 1 1.0 2.0 3.0
*CASE_END_2
*CASE_END_1
```

```
*CASE_BEGIN_2
*CASE_BEGIN_1
*NODE 1 1.0 2.0 3.0
*CASE_END_1
*CASE_END_2
```

End of mini tutorial.

Creating and editing a master *CASE

From [Keyword] **CASE** use the **CREATE** option to map the Master **`*CASE`** creation and editing panel.



`Case is explicitly defined`	This is explained below.
`Label`	Is the unique Master **`*CASE`** label
`Jobid`	Optional string
`Title`	Optional: used only by PRIMER
`Command line input`	Optional: any number of text lines giving command-line input to be used when the job is run

Adding child (sub-)cases to this Master *CASE:

Child cases in PRIMER use the acronym CCAS (for Child CASe). You can add child cases to this master case by:

- Typing in the label of an existing child case.
- Using the right-click popup to select or create a child case
- Using Add... to select a range of existing child cases from a menu list

Removing child cases from this Master *CASE:

- Remove their labels from the list.
- Use Remove ... to select child cases to be removed.

"Case is explicitly defined"

The LS-DYNA syntax makes provision for a model in which **`*CASE_BEGIN_`*`nnn`*** **`... *CASE_END_`*`nnn`** cards

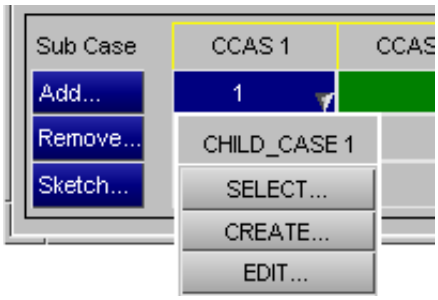
appear, but no master ***CASE** cards are defined. In this situation every child case becomes an "implicit" master ***CASE** of the same label nnn.

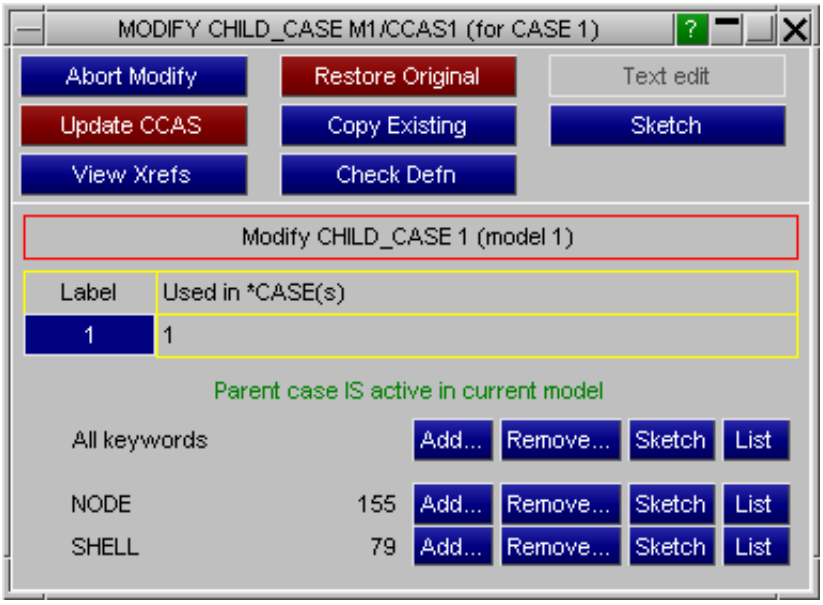
In PRIMER this is handled by automatically creating a master ***CASE** of label *nnn* for each child case when keyword input is complete. To distinguish this from an explicitly defined master ***CASE** keyword PRIMER adds an "explicit" flag which will be set only if a master ***CASE** keyword was read, or created interactively.

The only difference between an explicit and an implicit master ***CASE** in PRIMER is that implicit master cases are not written to the keyword output file. In all other respects they are the same, and can be used in the same way. You can switch cases between implicit and explicit at will, however bear in mind that the LS-DYNA syntax only "auto creates" a master case for each child if there are ***no*** explicit master ***CASE** keywords in the model. So having a mixture of implicit and explicit master cases will result in the implicit ones being "lost" when the model is written to a keyword output file.

Creating and editing a child case

Using the right-click popup from an entry in the master ***CASE** panel above you can invoke the Child Case editing panel.





This allows you to **Add...** and **Remove...** items of any type in the current child case.

"**Add**ing" an item	moves it from the main model to this child case.
"**Remov**ing" an item	moves it from this child case back to the main model.

Note: editing of a child case's contents is only possible from this panel if the child case is part of the current master ***CASE**. The child case editing panel will state whether it is or not.

If the child case is not active in the current master ***CASE** this will be stated, and the **Add...** and **Remove...** buttons will be greyed out.

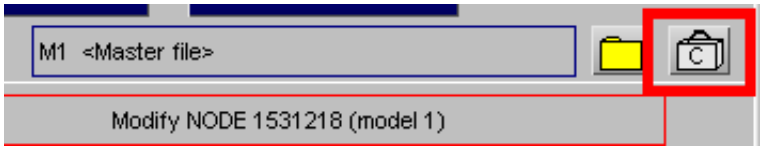
However you will still be able to sketch and list contents. The reason for this limitation is that child cases that are not current are on backing store, where their content is effectively "read only".

This panel is designed to provide a fast and simple way of moving items between the current child case and the main model, and it has the advantage that multiple items can be added / removed in a single operation. However this process has limitations when a labelled item in the current child case is also in ***combination*** with one or more other child cases that are currently on backing store and you attempt to move such an item from the child case to the main model. This would cause a clash if the other case(s) in the combination were made current because this would mean duplicate definitions in both the other case(s) and the main model.

Prior to the **Update** (to make the scratch definition permanent) PRIMER checks all items selected for addition to or removal from this child case to detect any such clashes, and gives you the following options if anty such clashes are found:

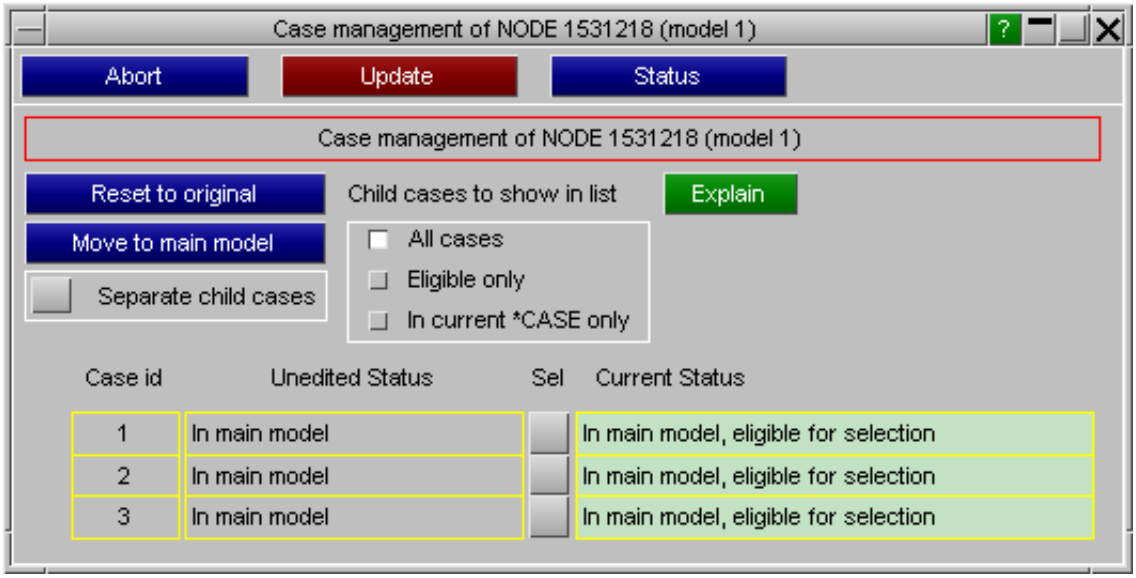
Abort the **Update** operation	Go back and correct the definition to remove the clashes.
Move only the legal items	Only the non-clashing items are moved to the main model, the rest stay in their current child case
Move and delete / make latent	The clashing items are deleted is possible, but if referenced are left as latent definitions in the main model. If one of the other child cases with which an item is in ***combination*** is made current then its definition will be merged silently with the latent definition in the main model. This is legal and will not cause conflicts.
Move everything anyway	All selected items are moved to the main model regardless of potential clashes. If you attempt to make one of the cases with which the item is in ***combination*** current PRIMER will detect an impending "duplicate item" clash and will force you to relabel things in order to avoid it. So use this option only if you know that this is not likely to be done, in other words if the other child case(s) in the combination are not referred to in any master **`*CASE`**.

Editing the child case membership of an item using the [C] button:



Another, and much more flexible, way of editing the child case membership of an item is via the **[C]** case membership editing button in the item's editing panel.

This will only appear if the model contains **`*CASE`** data, and it will always be to the right of the file filter button for include files. This maps the child case membership editing panel for the item in question (here node 1531218):



In this example the node is currently in the main model, and there are three child cases currently defined.

Master **`*CASE`** 1, the current master case, contains child cases 1 and 2.
Master **`*CASE`** 2, not currently active, contains child case 3.

At present the node is eligible for moving to any of the child cases, which can be done by ticking the relevant box in the **Sel(ect)** column.

Case id	Unedited Status	Sel	Current Status
1	In main model	✔	Selected in case 1 (current)
2	In main model		Potential clash in master *CASE 1
3	In main model		Eligible for selection in case 3

The user has now ticked the Sel box for child case 1, and the following has happened:

- The node is shown as "Selected" in child case one.
- Selection in child case 2 has been prohibited.

  Remember that the current master **`*CASE`** contains child cases 1 and 2, so if the node is present in child case 1 it cannot also be present in child case 2, since this would constitute an illegal duplicate definition of the same label.
- Selection in child case 3 is still available if the user wants to choose it, as there is no potential clash.

Case id	Unedited Status	Sel	Current Status
1	In main model	✔	Selected in case 1 (current)
2	In main model		Potential clash in master *CASE 1
3	In main model	✔	Selected in case 3 (current)

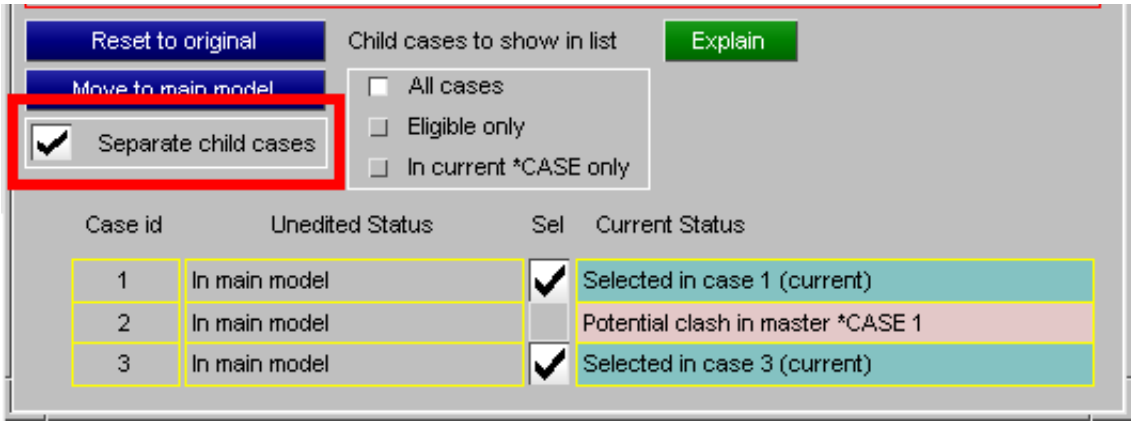
The user has now chosen to place this node in child case 3 as well.

This means that there will be a single **combination** definition of this node in child cases 1 and 3. If the keyword deck were written out at this point it would contain the lines:

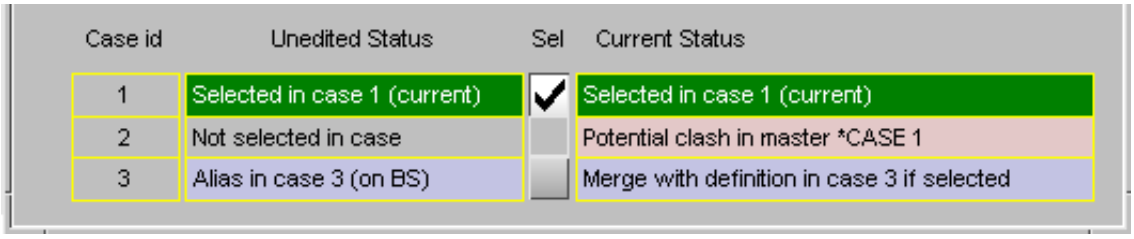
```
*CASE_BEGIN_1
*CASE_BEGIN_3
*NODE
  1531218       1.0       2.0    (coordinates etc made up here)
*CASE_END_1
*CASE_END_3
```

Separate child cases: changing the definition from a **`combination`** to an **`alias`**.

Let us suppose that the user does not want to have a single common definition, but rather two wholly separate definitions of this node in cases 1 and 3 - perhaps so that they can have different coordinates. The "separate child cases" button tells the editing panel that on update it should separate the single common definition into two.



At this point it should be made clear that all edits to a definition's child case membership made in this panel are "pending", as with all changes made in an editing panel, and will only become permanent when the parent editing panel is updated and closed. So if we go through the update cycle for this node, then reopen it for editing and use the **[C]** button again the following will be shown:



This is telling us that:

- The current definition of this node is in child case 1 only, which is current in this model.
- It is not selectable in child case 2 because that would cause a crash for the reasons given above
- There is a separate **alias** definition of this node in child case 3, currently on backing store (BS).

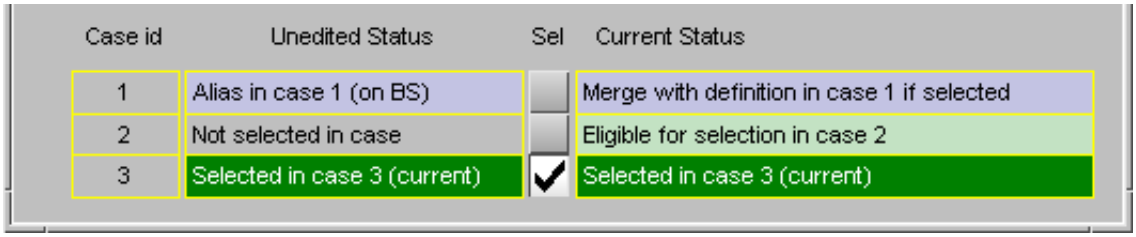
If this deck were written to a keyword output file the lines for this node would now be written out twice as two separate definitions:

```
*CASE_BEGIN_1
*NODE
  1531218 1.0     2.0 (coordinates etc made up here)
*CASE_END_1
*CASE_BEGIN_3
*NODE
  1531218 1.0     2.0 (coordinates etc made up here
*CASE_END_3
```

If we wished to merge these two **alias** definition back into a single **combination** definition, then ticking the **Sel** box on the row for child case 3 would result in that change being made when the node editing panel was updated.

To make this point clear we now:

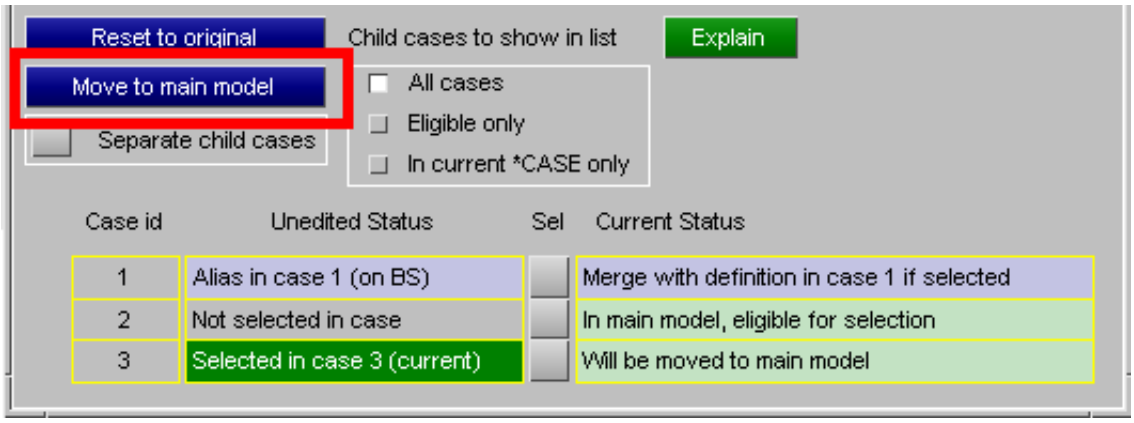
- Abort this node editing panel, leaving the definitions as **aliases**.
- Change the current master **`*CASE`** from 1 to 2 (which contains child case 3)
- Reopen the node editing panel
- Click on the **[C]** case editing button again



We are now editing the definition of the node in child case 3, and you will see that the status above has been swapped round with the **alias** definition in child case 1 now on backing store.

Move to main model: moving this definition out of the child case and back into the main model.

If we select this option then the panel (still in child case 3) above becomes:



Upon update from the parent node editing panel the following will happen:

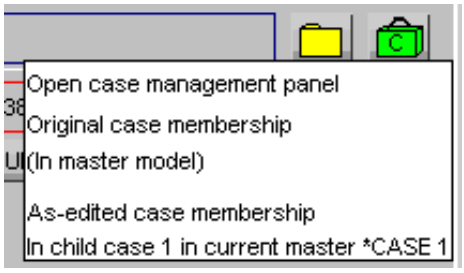
- This definition (in child case 3) will be moved to the main model.
- All other definitions of this node currently on backing store **will be deleted**.

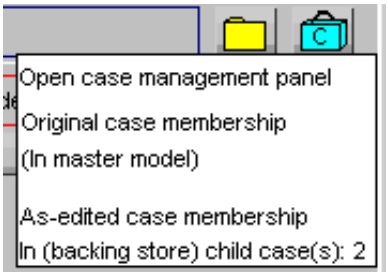
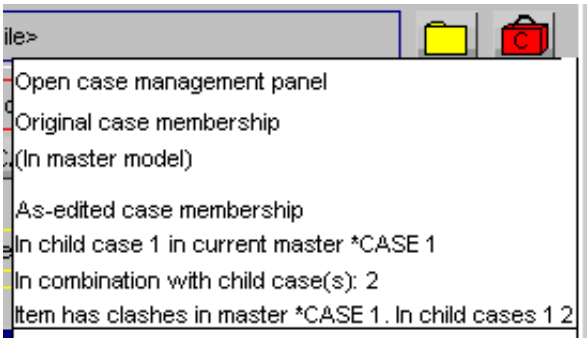
Reset to original: resets case membership to its original state before editing.

As explained above all changes made in the case membership editing panel are "Pending", and will only be made permanent when the parent editing panel (a node editing panel in the example above) is updated. Resetting case membership to its original state removes any edits and restores case membership to its state when the editing panel was first opened.

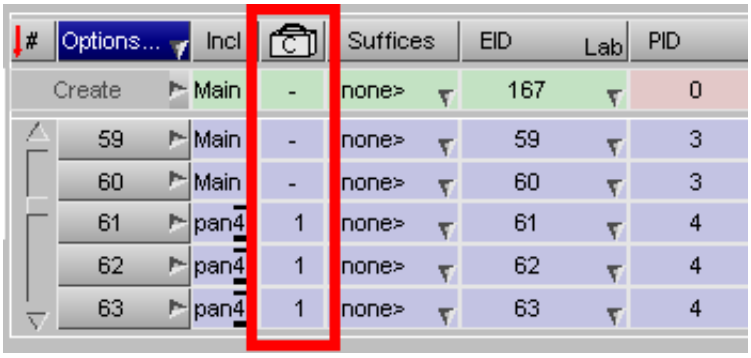
Feedback from the [C] button

The colour of this button conveys information about the case membership status of the item being edited, and hovering the mouse over it will also give a summary of the current case status if the item being edited.

Appearance (showing the status message given by hovering the mouse over the [C] button)	**Colour**	**Status**
	Grey	The item being edited is in the main model, ie not currently in any child case. This will be the normal case for most items in most models.
	Green	The item being edited is (or will be) in one or more child cases that are in the current master `*CASE`, ie currently active in the model. It may also be in **combination** with definitions on backing store, or have **aliases** on backing store. If this is the case the "hover over" message will give a summary of this.

	Cyan	The item being edited will, if the edit proceeds, be moved into a child case which is ***not*** in the current master **`*CASE`**, and thus will be moved onto backing store. This means that when the editing panel is updated the item ***will disappear from the current model*** when it is moved to backing store. If it was referenced by something else in the model it will be replaced by a latent definition of the same label in the current model.
	Red	The item being edited either exists in a clashing combination of child cases, or will do so if the edit proceeds. This is illegal since it would mean duplicate definitions of the same label in the model, and you should - hopefully! - never see this. (This example was created artificially for demonstration purposes, and if this situation ever arises you will be forced to resolve the case membership or label clash before updating the edit.)

The [C]ase button in the Keyword editor



If a model contains **`*CASE`** data then all keyword editor panels will have a [C]ase column showing the current case membership of each row.

`[ - ]`	Means that the item is in the main model
`[ n ]`	Means the item is in child case **`n`**

Clicking on the relevant row button maps the same case management panel as above, which acts in exactly the same way.

Note: Selecting multiple rows and clicking on the [C] button does ***not*** apply the case membership changes to all selected rows, but rather ***only*** to the row in question. This limitation may be removed in the future, but at the present time it is "too hard" to implement in this context.

The Master *CASE summary panel

By default PRIMER will make the first master **`*CASE`** in a model the current one, but you can change the current master **`*CASE`** at any time using **Master *CASE Summary and Set**

The Master ***CASE** summary panel allows you to do several things:

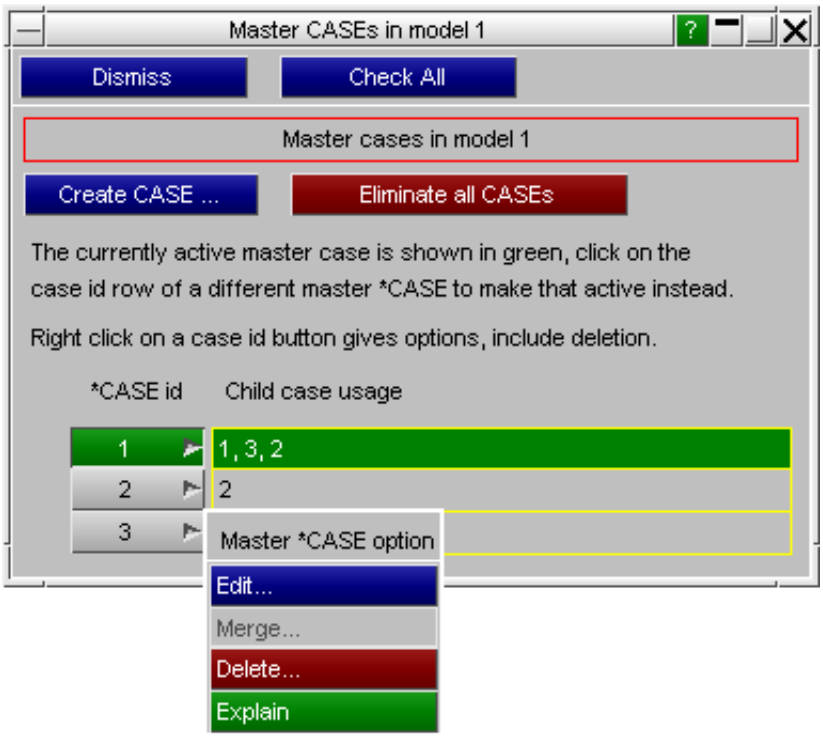
- Change the currently active master ***CASE**.

  This is done by clicking on the relevant ***CASE id** button, the currently active row always being shown in green. In this example ***CASE** 1 is current.

- Edit, merge or delete master ***CASE** definitions via the right click popup, shown here for master ***CASE** 3.

- Create a new master ***CASE** using **Create CASE ...**. This is just another way of getting to the master ***CASE** creation panel described above.

- Remove all master and child cases using **Eliminate all CASEs**, merging everything into unique definitions in the main model.



Changing the current master *CASE

When you change to a new master ***CASE** the following operations are performed:

1. All items currently in child cases are moved to Backing Store.
2. Any transforms in or affected by these cases (include_transform, node_transform, part move, etc) are undone
3. If this leaves "referenced but undefined" definitions behind, then Latent definitions are created to fulfil these references.
4. Items in the child cases referred to in the new master *CASE are moved from backing store to the main model.
5. Where these items duplicate an existing latent (referenced but undefined) definition, then the definitions are merged.
6. The whole model then performs an "internal rebuild" to ensure that references to items point to the correct - possibly new - definitions.
7. All internally cached data such as calculated masses, set contents, etc are marked as invalid and needing recalculation.
8. Any transformation in or affected by this case change are re-applied.
9. All graphics and visibility tables are recalculated, and the model is redrawn.

From reading the list above you will appreciate that this is a complicated and time-consuming process which may take quite a long time for a large model. Therefore it is best to try to organise your work so that you perform as few master ***CASE** change operations as possible.

[Popup] Edit... Editing a master *CASE

This is just another way of getting to the master ***CASE** editing panel described above. You can edit any master ***CASE**, not just the current one.

[Popup] Merge... Merging the contents of the current master *CASE into the main model

This performs the following operations:

- The contents of any child cases that are ***unique*** to this master **`*CASE`** are removed from their child case and moved back into the main model, and those child cases are deleted.
- The contents of any other child cases are left unchanged (since they are used by other master **`*CASE`**s)
- This master **`*CASE`** definition is deleted

This operation is only available for the current master **`*CASE`**. To merge contents of other master **`*CASE`**s into the main model they must first be made current.

This operation is not reversible!

[Popup] Delete ... Deleting a master *CASE

Any master **`*CASE`** apart from the current one can be deleted. This deletes the master **`*CASE`** definition itself, but ***does not delete or merge*** or affect its child case contents in any way.

This may result in child cases becoming "orphans", ie not referenced by any master **`*CASE`**. This is perfectly legal, and such definitions will still be written out in any keyword output file, but they will not be part of any subsequent LS-DYNA analysis.

Eliminate all cases Removing all *CASE definitions from the model.

This performs the following operations:

- All items in child cases in the current model have their child case status removed and become part of the main model.
- All child cases currently on backing store, and their contents, ***are deleted***.
- All master and child **`*CASE`** definitions and storage are removed.
- The model is rebuilt internally as required, recalculated and redrawn.

WARNING: the outcome of this operation will probably delete data!

You will see from the above sequence items currently on backing store are deleted, not moved into the main model. If you want to move these into the main model you will need to make a master **`*CASE`** containing them current and merge this into the master model.

Therefore this is a simple and quick way of removing all case information from a model, but if the details of what are to remain are important it may be better to perform a more selective merge process, or even to process individual items.

Other operations affected by *CASE

The following operations are influenced by **`*CASE`**, and may require consideration.

Model WRITE and *CASE

When a model containing **`*CASE`** information is written out to a keyword file PRIMER will retain the logical case organisation, although the order in which keywords appear may change.

Also PRIMER will never use the syntax

```
*KEYWORD     CID=nnn
data
```

during keyword output, always replacing it with

```
*CASE_BEGIN_nnn
*KEYWORD
data
*CASE_END_nnn
```

This gives rather more verbose output, but the functionality is identical.

Model COPY and *CASE

When a model containing **`*CASE`** is copied to another model inside PRIMER the case-specific organisation and data will be retained completely.

Model MERGE and *CASE

When two models are merged, and either of them contain **`*CASE`** data, then PRIMER retains this in the output model.

However in the situation where both models contain either master or child case definitions of the same label PRIMER does ***not merge the case definitions***, rather it applies the same "increment labels to avoid clashes" logic that it applies to other keywords, retaining separate cases in the merged model.

This is best illustrated by example. In the following example the "increment labels in slave (model 2) to avoid clashes" option has been chosen

Source model 1	Source model 2	Result in output model 3
Master *CASE 1	Master *CASE 1	Master *CASE 1 (*was 1 in model 1*) Master *CASE 2 (*was 1 in model 2*)
Child case 1 Child case 2	Child case 1 Child case 2 Child case 3	Child case 1 (*was 1 in model 1*) Child case 2 (*was 2 in model 1*) Child case 3 (*was 3 in model 2*) Child case 4 (*was 1 in model 2*) Child case 5 (*was 2 in model 2*)

If you wish to merge the contents of child cases in the output model you will have to do this manually.

*CASE and include files.

PRIMER 12 allows you to move include files into and out of child cases, and it also obeys the LS-DYNA limitation that the *CASE keyword may not be used in include files. This means that the sequence [in the master file]

```
*CASE_BEGIN_xxx
*INCLUDE
filename
*CASE_END_xxx
```

Requires that all items in include *filename*, and all its descendants, are explicitly in child case xxx. PRIMER checks for this when running the standard "check" functions, and also when editing case membership where it will warn you if you try to change the case membership of an item in an include file to something different to that of the include file itself.

*CASE plus *INCLUDE_TRANSFORM, *NODE_TRANSFORM and *PART_MOVE

PRIMER 12 allows these "transform" keywords to be used with **`*CASE`** and should deal with all permutations of them correctly.

Moreover it should be possible to have:

- These "transform" keywords in child cases or the main model
- **`*DEFINE_TRANSFORMATION`** cards referred to by these in different cases
- All of the above using **`*PARAMETER`** definitions in yet other cases.

But please note the word "should" in the sentence above. It is possible to pile complication upon transformation upon further complication using the keywords above, and while we believe that it works it is entirely possible that attempting to edit a fiendishly complex permutation of these keywords may not behave correctly. A moment’s thought about the internal house-keeping required to process them will reveal why this is the case.

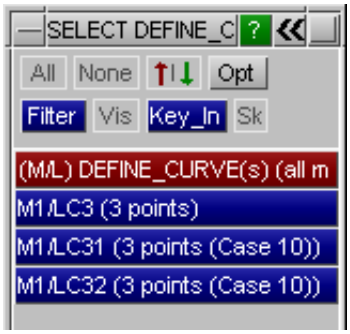
If you use these keywords in combination to create multiple input decks please try to keep usage simple!

Selection from Object menus and *CASE

Object menus provide support for items in cases in two ways:

If an item is in a child case this fact is appended to the menu listing.

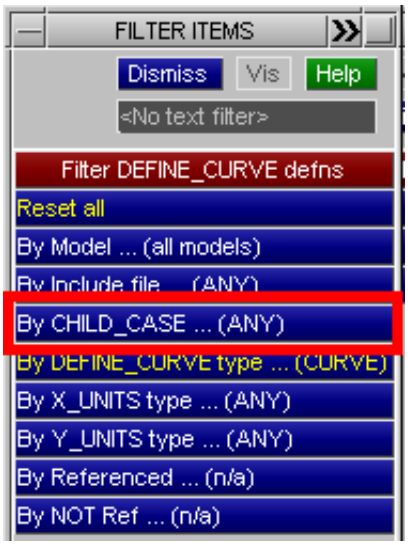
In the example here loadcurves 31 and 32 are both in child case 10

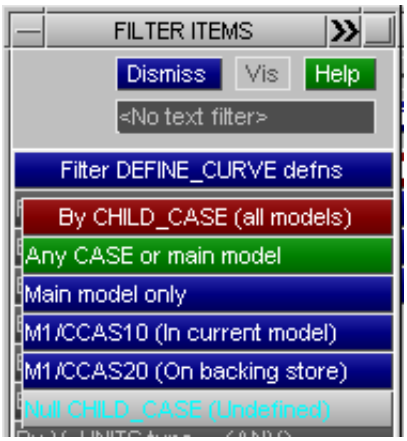


Filtering by child case is available

This can make it easy to "drill down" to the items of interest in a child case.

Note that "Main Model only" is available as an option here, which is the same as saying "not in any child case".



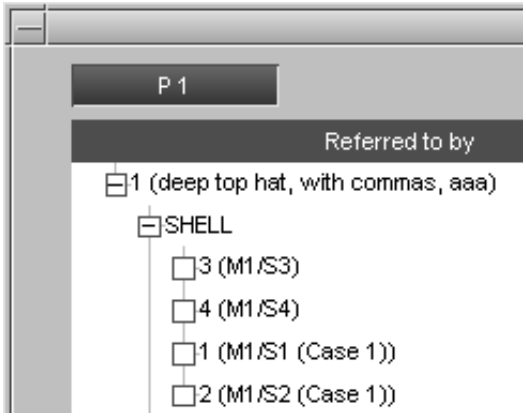


Cross-references and *CASE

Within the current model internal cross-references are created normally, regardless of whether referrer or referee are in a child case or in the main model. For example a **PART** referred to by a **SHELL** that is in a child case will show that it is referred to by that shell, and this reference will "lock" the part against deletion.

In addition the cross-reference editor will show that a reference is from an item in a child case that is in the current model.

PART 1 in this example is referred to by shells 3 and 4 in the main model, and shells 1 and 2 in child case 1.



However what happens when a **PART** in the main model is referred to by a **SHELL** that is in a child case that is not current, ie on backing store? There are three possible solutions to this problem:

1. Use normal cross-references from items in backing store.
2. Do not have any cross-references from items in backing store.
3. Use modified "special" references from items in backing store.

A brief discussion of the pros and cons of each of the options above will help to explain why option (3), modified references, has been chosen despite some minor disadvantages.

(1) Use normal cross-references from items in backing store.

This has the advantage that an item in the main model "knows" that it is needed by something in a child case that is not current, and this locks it against deletion. However it presents several problems:

- PRIMER uses cross-reference lists for many purposes such as graphics, calculation of mass, connectivity checks (eg free edge display, "find attached") and so on. Clearly we do not want to draw items in cases currently in backing store, nor do we want these items to contribute to mass or inertia calculations, etc.

  Therefore it would be necessary to make some sort of distinction between an "active" reference from an item in the current model, and a "passive" one from an item on backing store. This would be possible, but it would be difficult to maintain and apply properly in all possible contexts and would be a possible cause of hard-to-find errors.

- The user would see cross-references from multiple instances of an item. For example there might be two references to a **`PART`** from **`SHELL`** label 1: one would be current and the other on backing store.

  Again, it would be possible to show a distinction between these but they would clutter up the cross-reference lists and there would be the potential for confusion between these and multiple "clone" definitions in the current model.

Faced with the problems above the "normal cross-reference" method has been rejected as being too difficult to implement safely.

(2) Do not have any cross-references from items in backing store.

This has the merit of simplicity, and it solves the problems cited in (1) above: no duplication of mass or connectivity, and no confusing multiple references. However it presents two major problems:

- Items in the main model would not "know" that they might be referred to when an item in a child case currently in backing store is made current, and they would not be locked against deletion. Therefore a "Cleanup unused" or a selective deletion might remove things that are actually needed.

- Just as importantly a reference ***to*** something inside PRIMER is not an integer label, but rather a "pointer" to the actual address in memory used to store that item. For example when **`SHELL 1`** in backing store refers to **`PART 1`** in the main model the reference is to the address of **`PART 1`** in memory. Therefore if that definition of **`PART 1`** is deleted things go horribly wrong, which is why cross-references are so important!

Therefore having no cross-references from items in backing store is simply not possible given the internal architecture of PRIMER.

(3) Use modified "special" references from items in backing store.

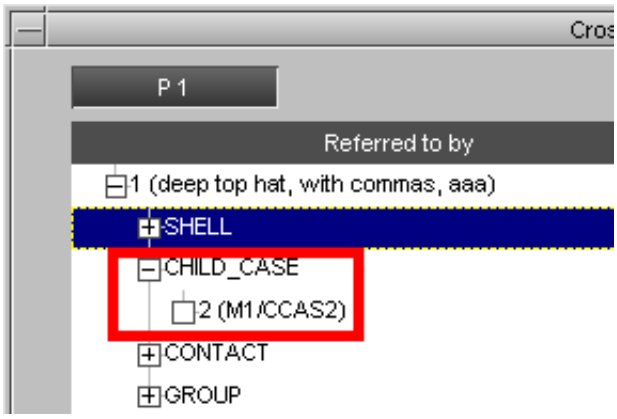
So the solution adopted is to make a cross-reference to an item in the main model from items in backing store, so that the main model item is locked against deletion, but in a way that is different to normal cross -references. This is done as follows:

- Any "normal" references from the item moved to backing store to items in the main model are removed.
- However a single generic reference from the child case is inserted instead.

This example shows how this appears in the cross-reference editor as a reference from a **`CHILD_CASE`**:

Here **`PART 1`**"knows" that items in child case 2 currently on backing store refer to it, and this locks it against deletion.

Only a single cross-reference from the child case is made, regardless of how many items within it reference the item in the main model.



This method is not perfect since it is not possible to tell exactly what in child case 2 references this part, the only way to do that would be to make a Master **`*CASE`** containing child case 2 current. However it is simple, it works and it solves the objections identified in methods (1) and (2) above.

COMMENT

PRIMER processes two sorts of comments:
1. "Embedded" comment lines, starting with a "$", as described in section 5.1.9
2. "Explicit" comments following the ***COMMENT** keyword header, as described here.

From LS971R6 onwards LS-DYNA permits a ***COMMENT** keyword followed by any number of comment lines (not starting with "$") to appear anywhere in the input deck. Any number of such keywords may exist, and while they play no part in the analysis they are echoed by LS-DYNA to "messag" and "d3hsp" output files.

Rules for locating *COMMENT definitions.

PRIMER reads, stores and writes out these cards but it has the additional problem that it has to try to "remember" their location in the input deck so that they are written out again in the correct place. Therefore the following rules are used:
- A ***COMMENT** card is assumed to be associated with the following keyword, on the assumption that descriptions usually precede what they describe.
- Therefore each ***COMMENT** definition in PRIMER has an "associated keyword", which is the (non-comment) keyword immediately following it in the input deck.
- On keyword output the ***COMMENT** card will be written out immediately before its associated keyword.

In order to make this logic work inside PRIMER a ***COMMENT** definition will make a cross-reference entry to its associated keyword, and this will show up in the cross-reference viewer. However this cross-reference will not "lock" the associated item against deletion, and if it is deleted then the ***COMMENT** definition will become "free-standing". (Note that no attempt is made to "re-associate" the ***COMMENT** with the next or previous associated keyword in this situation.)

Free-standing *COMMENT definitions

A ***COMMENT** is free-standing inside PRIMER if:
- It was never followed by a keyword. In other words it was followed by ***END** or by the "end of file".
- It was originally followed by an associated keyword, but this was subsequently deleted in the PRIMER session.

Free-standing ***COMMENT** definitions are always written out immediately before the ***END** card in their parent file.

*COMMENT and include files.

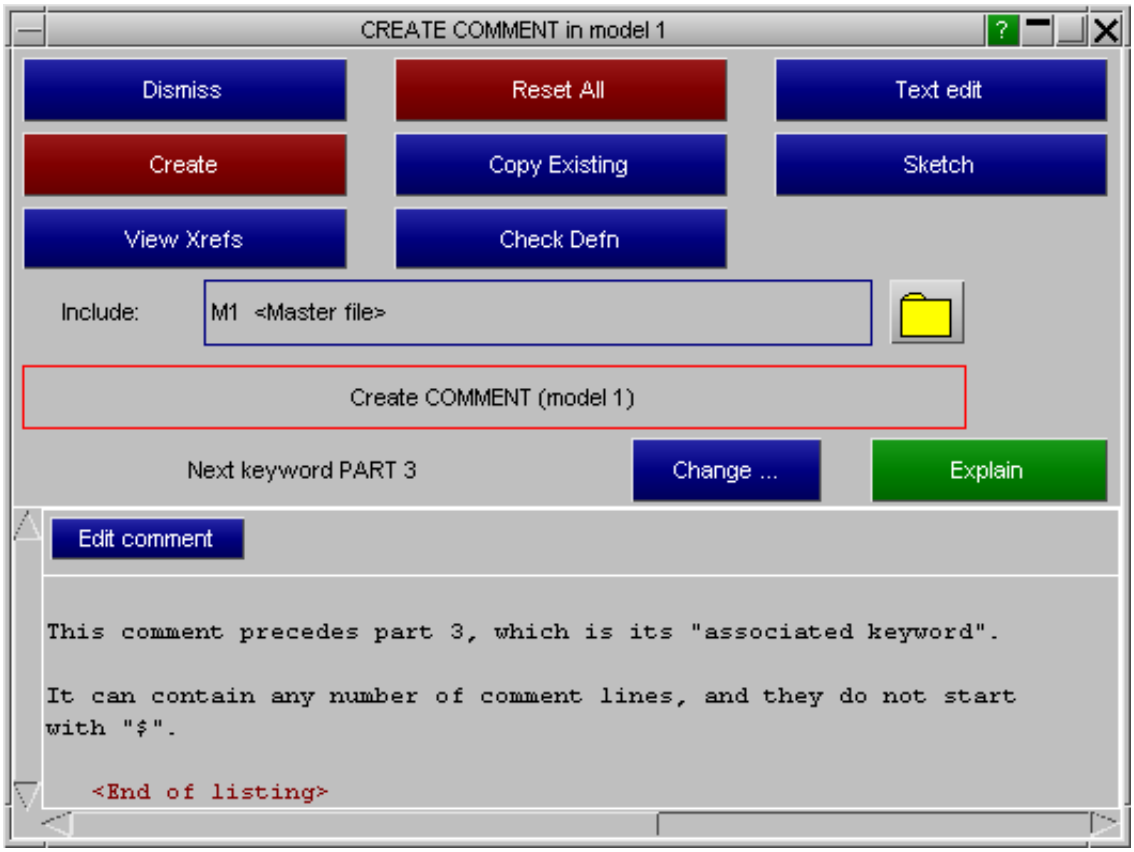
A ***COMMENT** definition will always have a "parent" include file, which is that from which it was read or that to which it is subsequently moved inside PRIMER . Normally its associated keyword will be in the same include file, but it is possible to change this inside PRIMER with the result that the ***COMMENT** definition and its associated keyword are in different include files. In this situation the following rules are used:
- On keyword output the ***COMMENT** definition will still be written out immediately before its associated keyword, in the include file of that keyword, regardless of its "parent" include file definition. Therefore the associated keywords "wins" in this sense if it is actually written out.

- The "parent" include file of the ***COMMENT** remains that from which it was read, regardless of the include file of any associated keyword. Editing panels for the ***COMMENT** definition will show this correctly.

- If the ***COMMENT** becomes free-standing, or on output its associated keyword is not written out but its parent include file is still written, then the ***COMMENT** will be written immediately before ***END** in its parent include file.

Creating and editing *COMMENTs interactively

***COMMENT** definitions can be created, edited and deleted just like any other keyword in PRIMER



The "Next (associated) keyword" here is PART 3, but a new keyword can be selected using **Change...**

The comment lines themselves can be edited using **Edit comment.** This launches the standard system editor (as in **Text edit**) that allows the comment lines to be edited as text.

There is no limit to the number of comment lines that may be given under a **`*COMMENT`** header, but it is recommended that the maximum line width be restricted to 80 characters for conformity with the standard input deck width. PRIMER will read up to 256 characters per line, and LS-DYNA ***may*** read more than this, but other software may truncate the file to 80 columns.

Sketching *COMMENT definitions

Obviously comments themselves can't be sketched! However it can be useful to know what they are associated with, so the **Sketch** functions draw their "associated keyword" item.

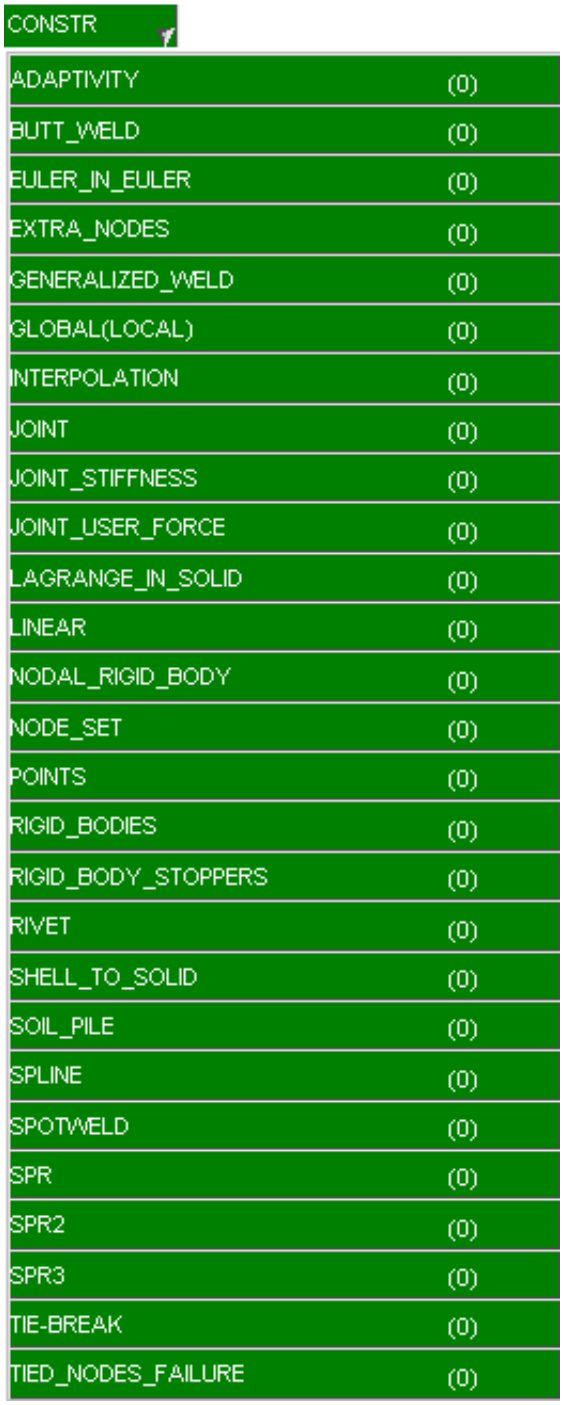
CONSTRAINED: Imposed constraints: joints, welds, etc

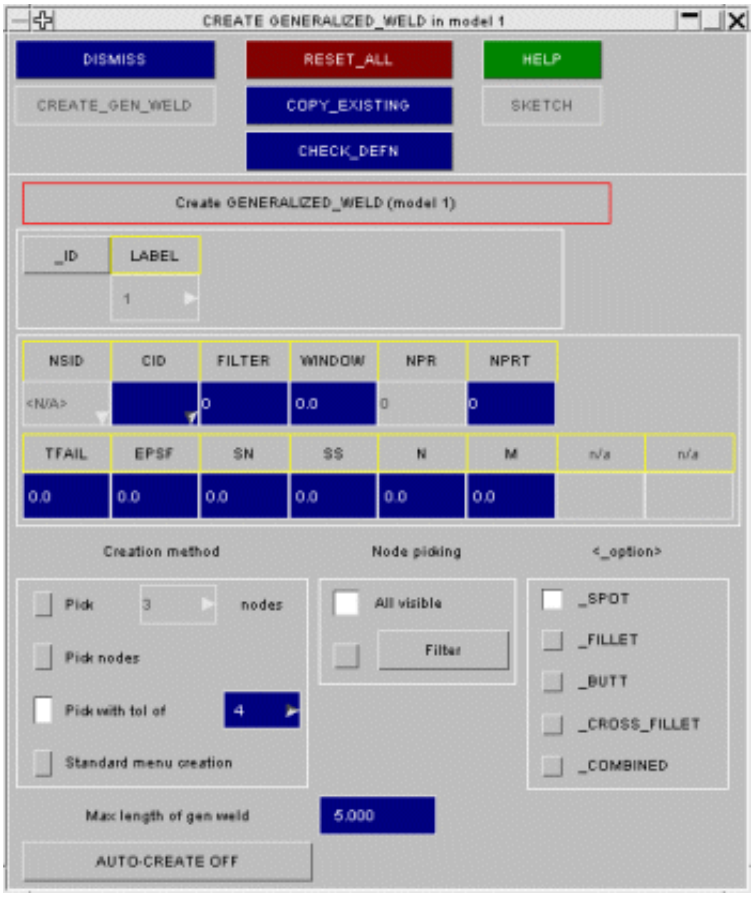
- Selecting the *CONSTRAINED sub-keyword
- "Scalar" editing panels
- "Edit range" editing panels
- Visualisation
- Labelling

Constrained conditions within LS-DYNA apply a range of constraints to models, and several ***CONSTRAINED** keywords are linked closely to rigid bodies.
All ***CONSTRAINED** sub-keywords are editable within PRIMER.

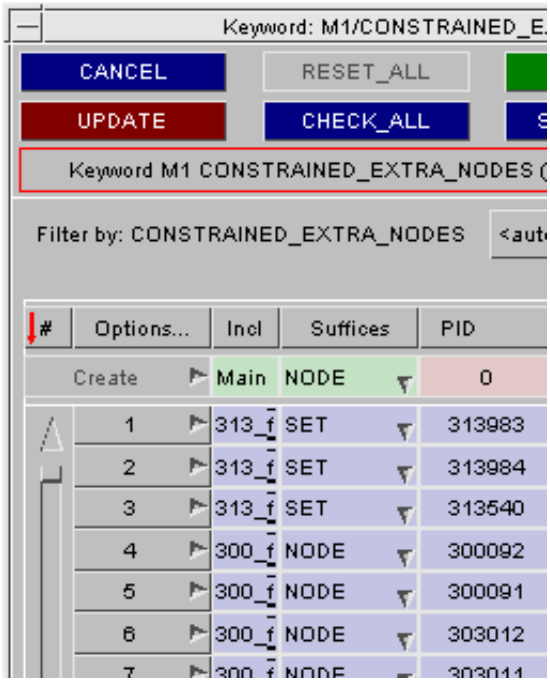
The ***CONSTRAINED** keyword has 26 sub-categories. Some may be created and modified using standard Create/Edit panels and some with the standard keyword editor. The table below defines which.

Keyword	Create/Edit panel	Keyword editor
ADAPTIVITY		✓
BUTT_WELD		✓
EULER_IN_EULER		✓
EXTRA_NODES	✓	✓
GENERALIZED_WELD	✓	✓
GLOBAL		✓
INTERPOLATION	✓	
JOINT		✓
JOINT_STIFFNESS		✓
JOINT_USER_FORCE	✓	
LAGRANGE_IN_SOLID		✓
LINEAR	✓	
NODAL_RIGID_BODY	✓	✓
NODE_SET	✓	✓
POINTS		✓
RIGID_BODIES	✓	✓
RIGID_BODY_STOPPERS		✓
RIVET	✓	✓
SHELL_TO_SOLID		✓
SPLINE	✓	
SPOTWELD	✓	✓
SPR		✓
SPR2		✓
SPR3		✓
TIE-BREAK		✓
TIED_NODES_FAILURE		✓





Example of the **CREATE/EDIT** panel (**GENERALIZED_WELD**)



Example of the keyword editor (**EXTRA_NODES_NODE**)

The methods for creating and modifying constrained entities falls into two categories.

- **EXTRA NODES, NODAL RIGID BODIES, NODE SETS** and **RIGID BODIES**
  These are "scalar" panels, in which a single definition is created or edited.

- **GENERALIZED WELDS, RIVETS** and **SPOTWELDS**
  These provide "scalar" creation and editing as above.
  Also "quick create" functionality to create a sequence of items.
  Also "edit range" functionality to permit edits to apply to a range of items.

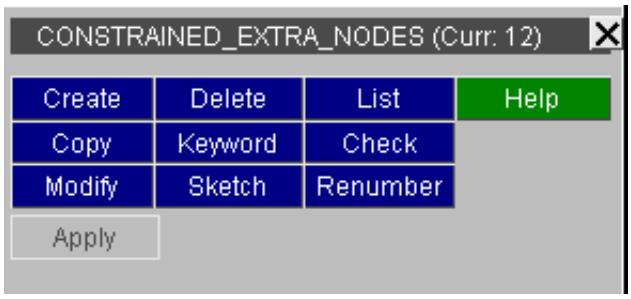
To illustrate the two categories **EXTRA_NODES** and **GENERALIZED WELDS** are presented as examples

CONSTRAINED_EXTRA_NODES Extra nodes on rigid bodies

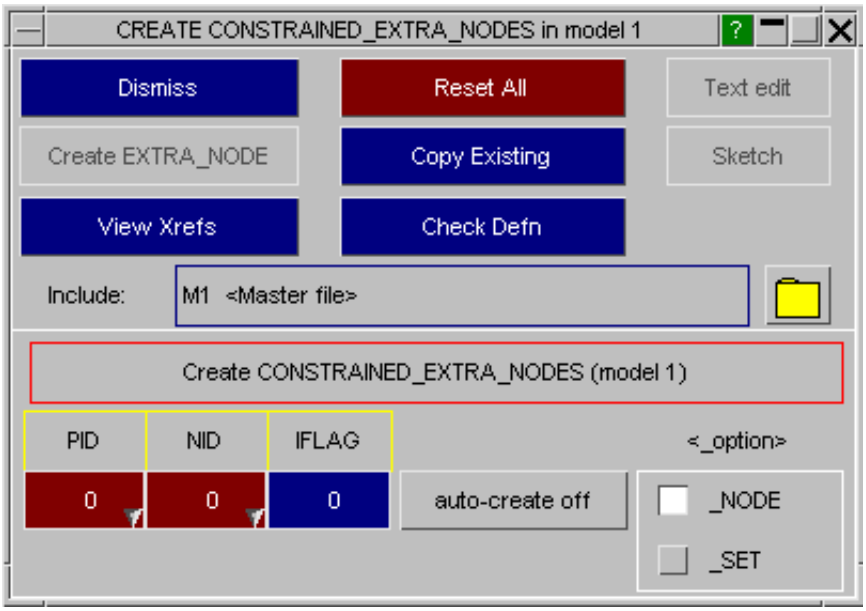
Constrained extra nodes allow a single node (**`EXTRA_NODES_NODE`**) or a group of nodes in a node set (**`EXTRA_ NODES_SET`**) to be attached to a rigid body in LS-Dyna.
This figure shows the main extra nodes menu.

The functions currently available have their standard meanings. (See 5.1.1)

CREATE and **MODIFY** apply only to single definitions ("scalar" editing)



CREATE Making a new extra node.



This figure shows the initial state of the extra nodes creation panel: no part or node has been given yet, so both are highlighted red.

The **<_option>** radio buttons can be used to change whether a **EXTRA_NODES_NODE** or a **EXTRA_NODES_SET** is created.

The part and the node (or node set) numbers can be typed directly into the text boxes. If the value is valid (for example the part must be rigid) the box will turn blue, otherwise an error message will be displayed indicating what is wrong. Alternatively, the popup menus can be used to pick a part, node or node set off the screen, or to select a part, node or node set from a list.There is an **AUTO-CREATE** button which will automatically create the new extra node once the necessary information has been given.

Once the required fields are filled in the **SKETCH** and **CREATE_EXTRA_NODE** buttons will become active.

CREATE_EXTRA_NODE saves the new definition permanently.

COPY Copying existing extra nodes(s) to make a new one(s).

You can **COPY** any number of extra node definitions, in multiple models.

For each model the <n> extra nodes chosen in that model are copied using labels <previous highest + 1> to <previous highest +n>, there is currently no control available over the new labels assigned.

MODIFY Modifying the attributes of an existing extra node.

This functions in exactly the same way as **CREATE**, using the same panels as in the figure above. The only difference is that the initial state of the panels is already set with the attributes of the extra node to be modified.

KEYWORD Invoking the standard keyword editor.

The standard keyword editing panel is set up.

DELETE Deleting existing constraints

The **DELETE** operation deletes the **`EXTRA_NODES`** definitions.

- If **DELETE_RECURSIVE** is switched on any nodes, node sets and parts, referenced by the extra nodes to be deleted are marked for deletion.
- If recursive deletion is not used only the extra node definitions themselves are removed.

Note also that the standard deletion rules described in Section 6.4.1 still apply: parts, nodes and node sets will only be deleted if nothing else (which is to remain) depends on them.

SKETCH Sketch the chosen extra node on the current image

SKETCH allows the user to select and sketch individual extra nodes on the current graphics image. Extra nodes are drawn with a dashed line from the node (or dashed lines from each node in the node set) to the centre of the rigid body.

CHECK Checking for errors

Runs the standard checking function on the selected extra nodes. Each extra node will be listed either as "OK", or a summary of the errors encountered will be printed. (This is the same as the **CHECK_DEFN** command during extra node editing.)

RENUMBER Changing labels.

Raises the standard renumbering panel for constraints in the chosen model, allowing you to renumber some or all of them.

As constraints do not have labels in LS-DYNA the usefulness of this is limited.

END_CONSTRAINED returns the user to the main **CONSTRAINED** box.

CONSTRAINED_GENERALIZED_WELD "Generalised" welds of various types.

Generalized welds in LS-Dyna are used to represent spotwelds between more than 2 nodes and fillet welds. At present only the creation and modification of spotwelds is implemented.

The main generalized weld menu has identical options to the extra nodes menu and the functions currently available have their standard meanings. (See section 5.1.1)

CREATE Making a new generalized weld.

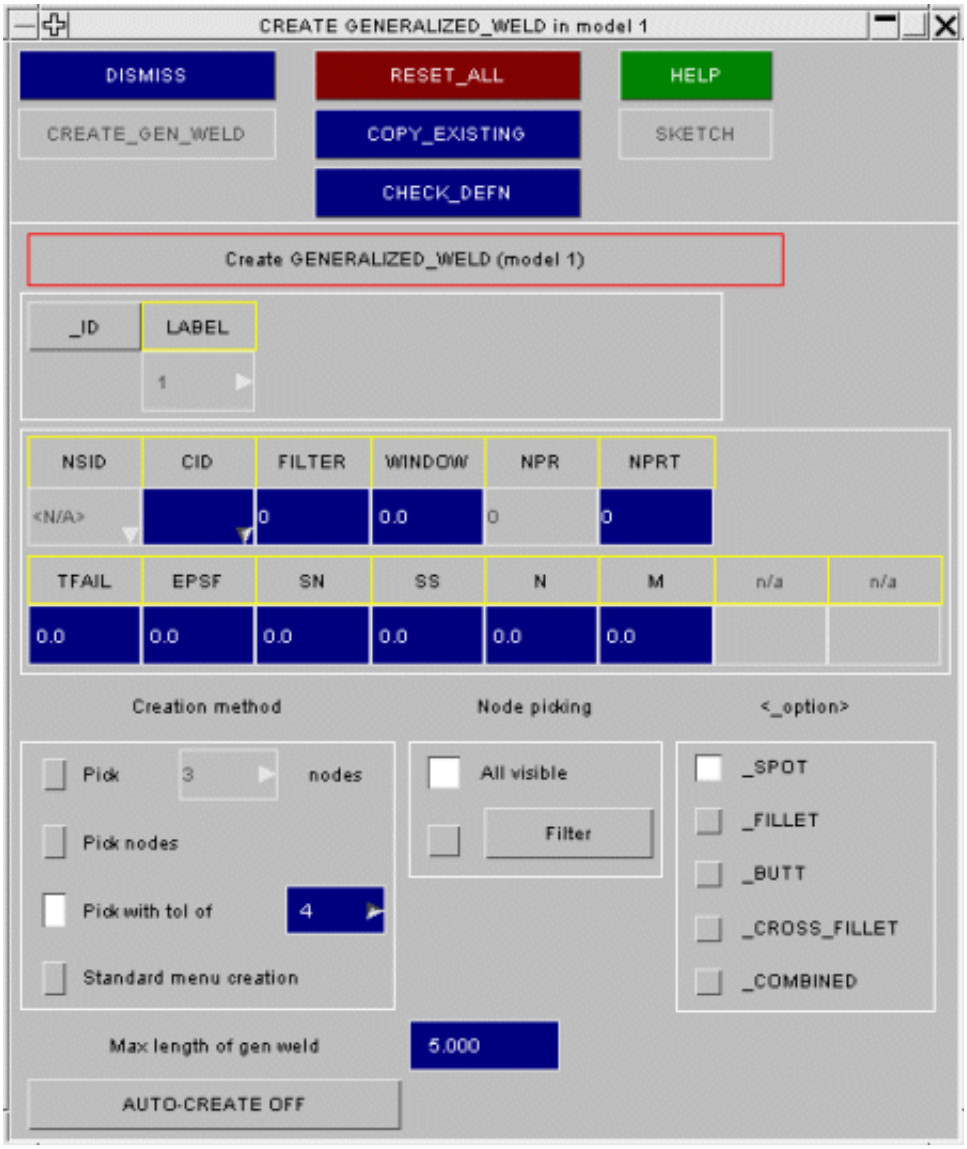
This figure shows the initial state of the generalized weld creation panel.

No node set has been given yet, so it is labelled <**N/A**>.

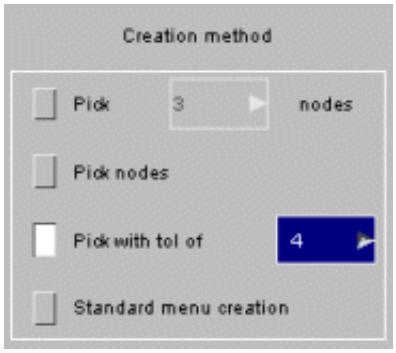
The **<_option>** radio buttons are greyed out as at present only **GENERALIZED_WELD_SPOT** can be created.

The various parameters for the generalized weld spot can be typed directly into the text boxes (eg **FILTER, SN, SS** etc). If the value is valid it will be displayed in the text box, otherwise an error message will be displayed indicating what is wrong.

Popup menus can be used to pick a coordinate system.

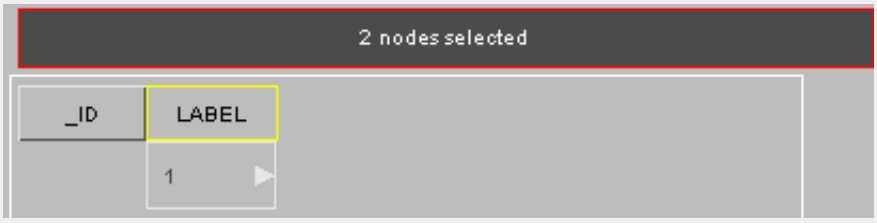


There are 4 methods available for creating the welds. For all methods except 'Pick nodes', there is an **AUTO-CREATE** button which will automatically create the generalized weld once the necessary information has been given. There is also a maximum length of generalized weld button which sets the maximum permissible length of weld. If you try to create a weld greater than this length, a warning will be given and creation stopped. If you had the **AUTO-CREATE** option on it will be turned off to give you a chance to do something about the problem such as changing the nodes or increasing the tolerance.



(1) Pick n nodes

If this option is selected you can pick nodes directly off the screen. The default number of nodes is 3 but you can easily change this by typing in a new number in the box (in the range 2 to 100) or by using the popup menu to select commonly used values. Once you have reached the number of nodes the **CREATE_GEN_WELD** and **SKETCH** buttons will be ungreyed, or if you have **AUTO-CREATE** on, the weld will automatically be created. If **AUTO-CREATE** is off and you try to pick more nodes they will be ignored and a warning given.
As you pick nodes the feedback button on the creation panel changes to indicate how many you have picked.



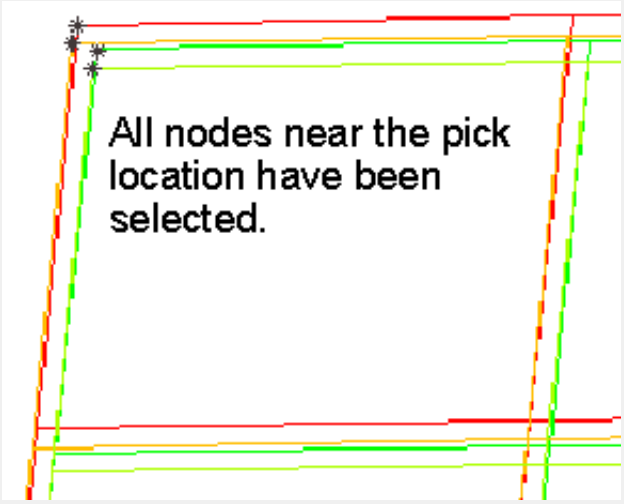
(2) Pick nodes

This is similar to method 1 but there is no limit on the number of nodes. Once you have picked 2 nodes the **CREATE_GEN_WELD** button will be ungreyed.
The **AUTO-CREATE** option cannot be used with this method. As you pick nodes the feedback button on the creation panel changes to indicate how many you have picked.

(3) Pick with tolerance of n

This method can be used to select all the nodes within a certain tolerance of a screen pick. The tolerance can be changed by typing in a number (in the range 1 to 7) or using the popup.
The nodes which you selected are sketched on the screen and the feedback button on the creation panel changes to indicate how many you have picked.

Care must be taken with this option to ensure that the weld geometry in a tightly meshed area is sensible.

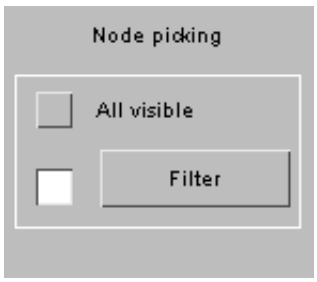


(4) Standard menu creation.

This method is the usual (scalar) method in PRIMER for creating an entity. Either type the values into the text boxes and/or use the popup menu to create or select a node set for the generalized weld.

Unlike other editing panels in PRIMER which are closed when the entity is created, the generalized weld creation panel will remain on the screen until the **DISMISS** button is pressed. Additionally all the values which you type in for the failure parameters are remembered so that when creating multiple generalized welds the information only has to be typed in once. This information is also remembered when you dismiss the window.

Node picking: filtering the nodes that are picked.
By default any visible nodes in the currently selected model can be picked for use in the generalized weld. This can be changed by using the **Node picking** option. If this is set to:



then If the filter option is chosen then

All visible Any visible nodes in the currently selected model can be picked.

Filter A sub-menu allows you to filter which nodes can be picked.
The most useful option is to select the **PART(s)** from which you want to pick the nodes. This permits you to limit selection within a dense mesh to just the panels you want to weld.

For example you could filter the nodes so that only nodes on 2 panels can be picked. All the other panels are still visible on the screen, but they will be ignored when picking nodes.

MODIFY Modifying the attributes of an existing generalized weld.

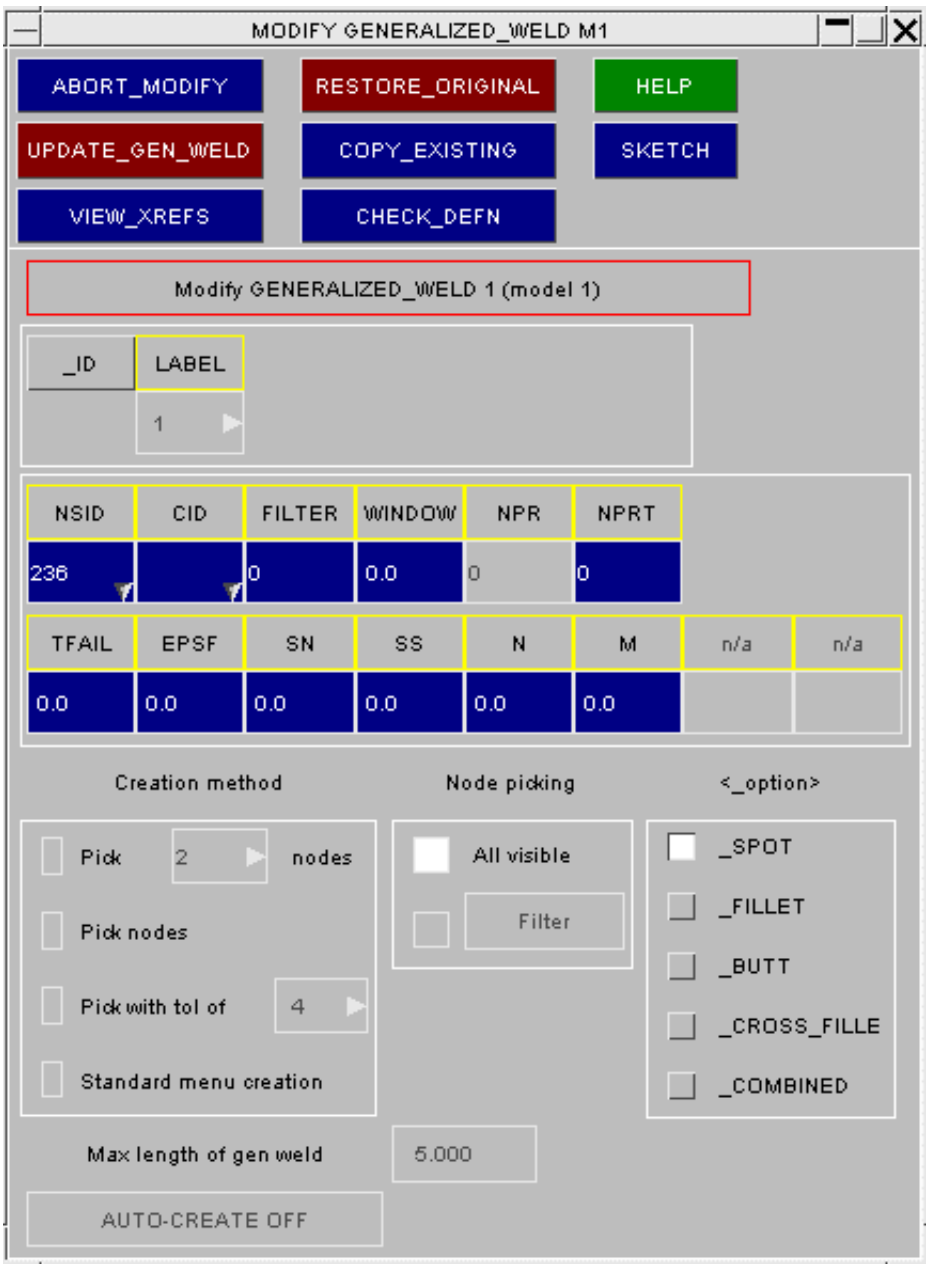
Unlike most editing panels in PRIMER it is possible to modify more than one generalized weld at a time. If only one is being modified then all attributes of the generalized weld including the failure criteria and the node set can be modified (see figure above).

None of the creation options are valid when modifying generalized welds so they are all greyed out. The node set can be changed or modified using the popup menus.

However when a range of > 1 welds has been selected then: (fig to right)

- The node set (**NSID**) is unavailable for editing.
- The default properties are taken from the first weld chosen, and will be applied to all welds, possibly modified, when you **UPDATE** the panel.
- The welds may be selected from multiple models, since only attributes, which are not model-specific, are editable.

Once all the modifications are complete the **UPDATE_GEN_WELD** button saves the new values into the database.



KEYWORD Editing welds using the generic Keyword editor.

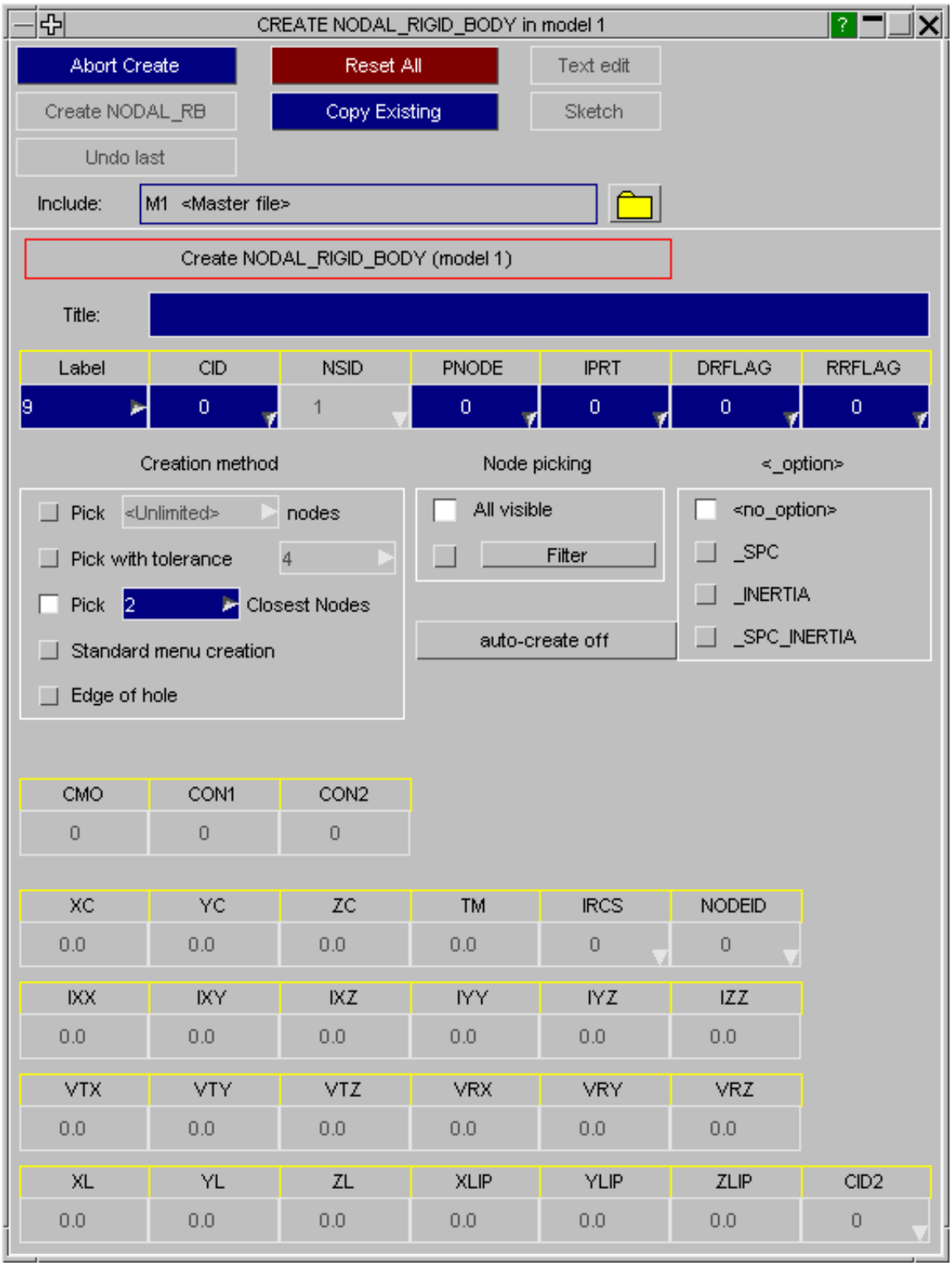
All weld types may be edited using the **KEYWORD** editor, but for the **CROSS_FILLET** and **COMBINED** types only the initial "attributes" rows are editable since the remaining rows are open-ended in length.

The keyword editor also provides a more selective way than **MODIFY** <range> above of changing properties over a range of welds

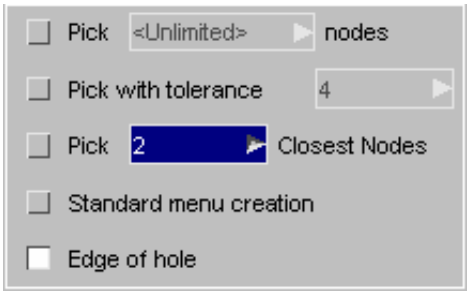
COPY	**Copying existing generalized weld(s) to make a new one(s).**	These functions work in exactly the same way as for **EXTRA_NODES**
DELETE	**Deleting existing generalized welds**	
SKETCH	**Sketching welds on the current image**	
CHECK	**Checking for errors**	
RENUMBER	**Renumbering welds**	

CREATE/EDIT panels for other constrained types

The **NODAL_RIGID_BODY** creation menu is similar to the **EXTRA_NODES** menu and uses the same principles.

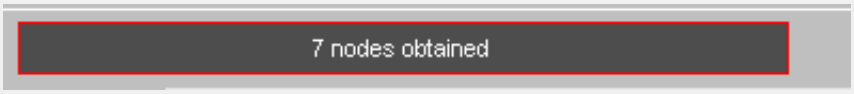


There are 5 methods available for creating a **NODAL_RIGID_BODY**. The first four methods are similar to those available for creating welds.

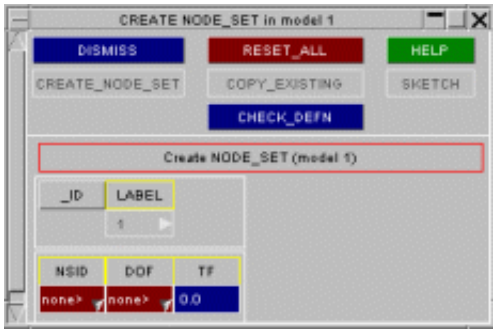


(5) Edge of hole

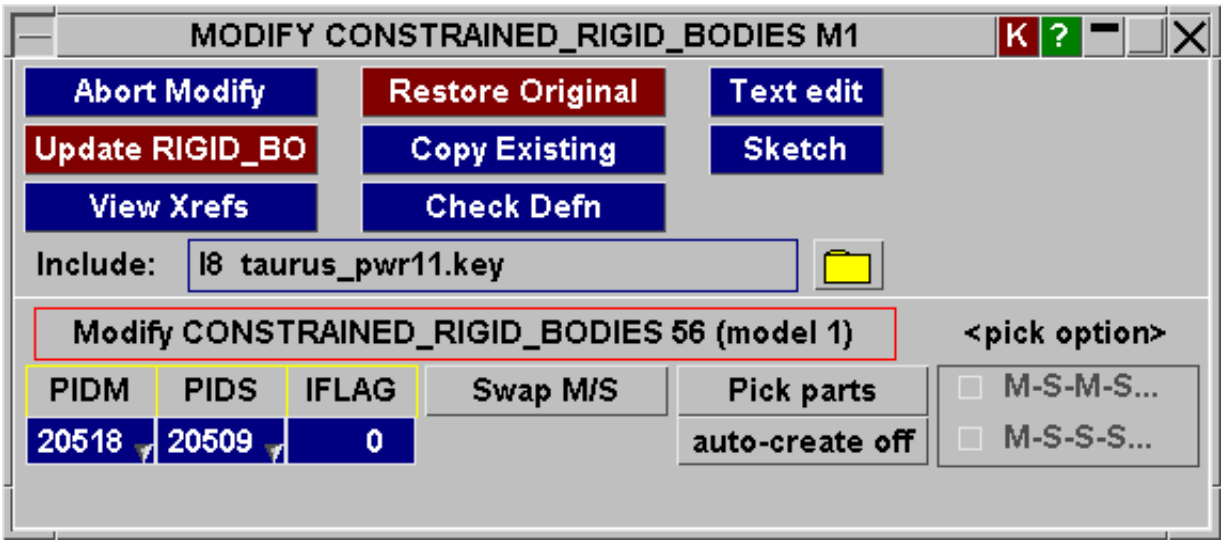
If this option is selected you can pick any one node on the edge of a hole. A node is automatically created at the centre of the hole and added to a node set. Nodes situated along the circumference are located and added to the same set. Once you have selected node on the edge of hole **Create NODAL_RB** and **SKETCH** buttons will be ungreyed, or if you have **AUTO-CREATE** on, the NRB will automatically be created.
As you pick a node on edge, the feedback button on the creation panel changes to indicate how many nodes are obtained along the circumference.



The **NODE_SET** creation menu is similar to the **EXTRA_NODES** menu and uses the same principles.



The **CONSTRAINED_RIGID_BODY** creation menu is similar to the **EXTRA_NODES** menu and uses the same principles. It has a few more specific options.



Swap M/S - Swap the master and slave entries.

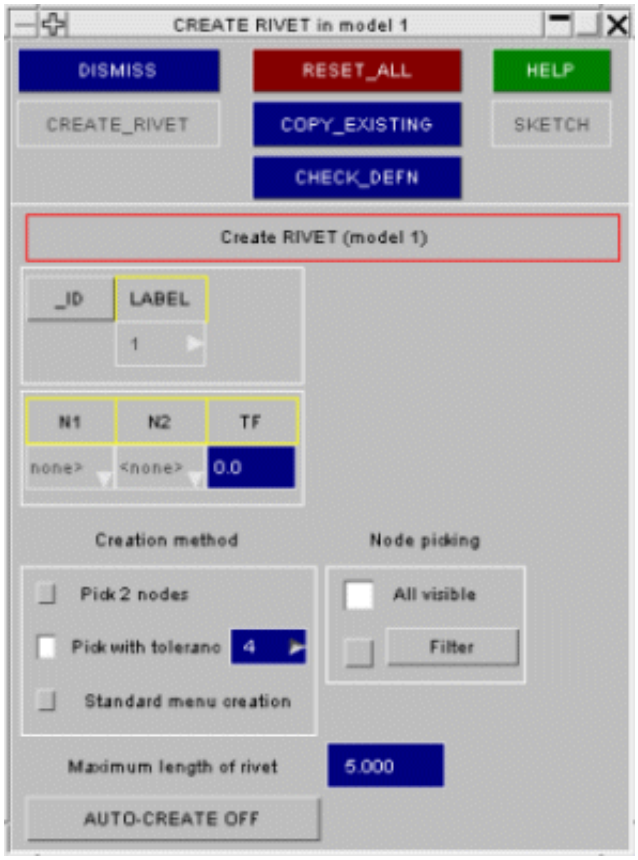
Pick Parts - Allows quick interactive picking - you do not need to instigate picking from the **PIDM** or **PIDS**

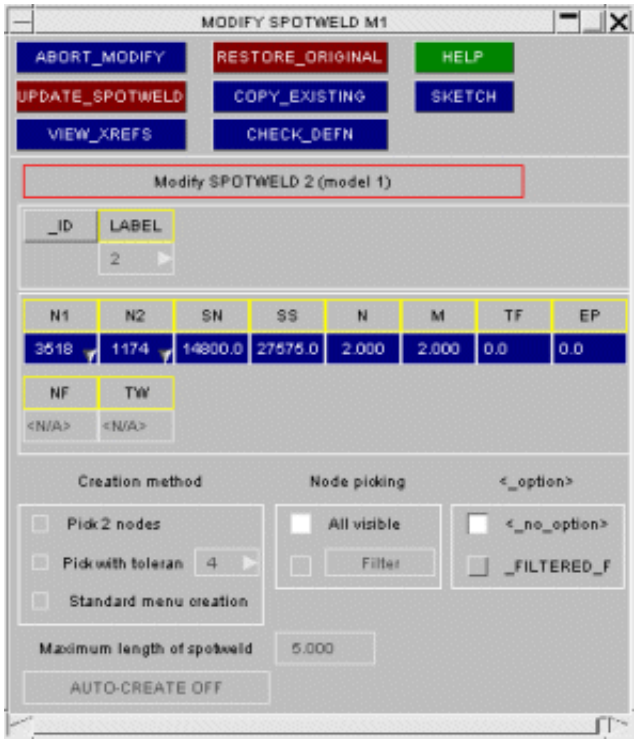
dropdowns.

Autocreate - Means a rigid body will be automatically created once the necessary information has been given (PIDM & PIDS).

Pick option - With the **Pick Parts** option on and **Autocreate** on the **Pick option** can be used to specify whether you wish to pick a new slave each time (M-S-M-S) or you wish to pick the master once followed by multiple slave to create multiple rigid body definitions with the same master (M-S-S-S).

The **RIVET** and **SPOTWELD** creation menus are similar to the **GENERALIZED_WELD** menu and use the same principles.

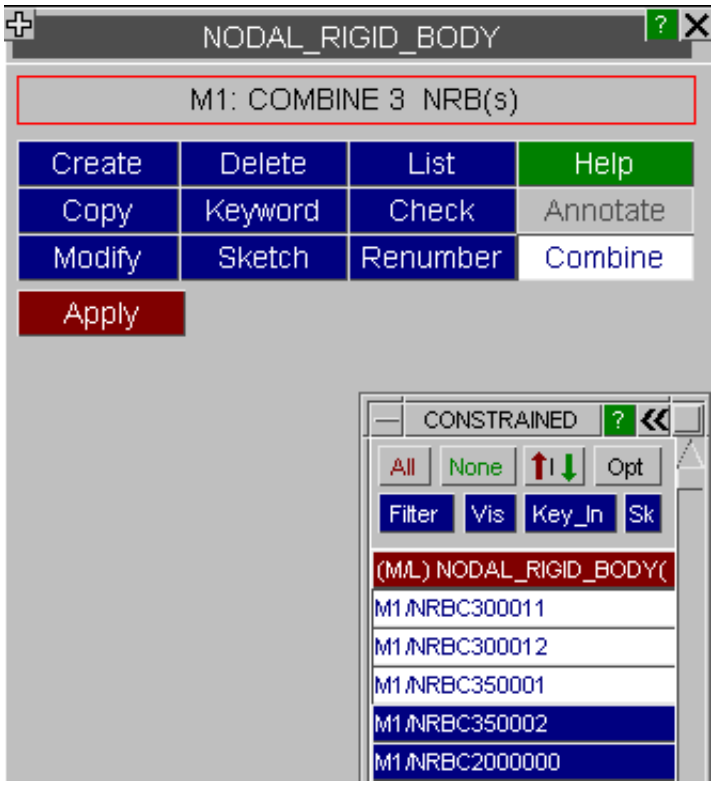




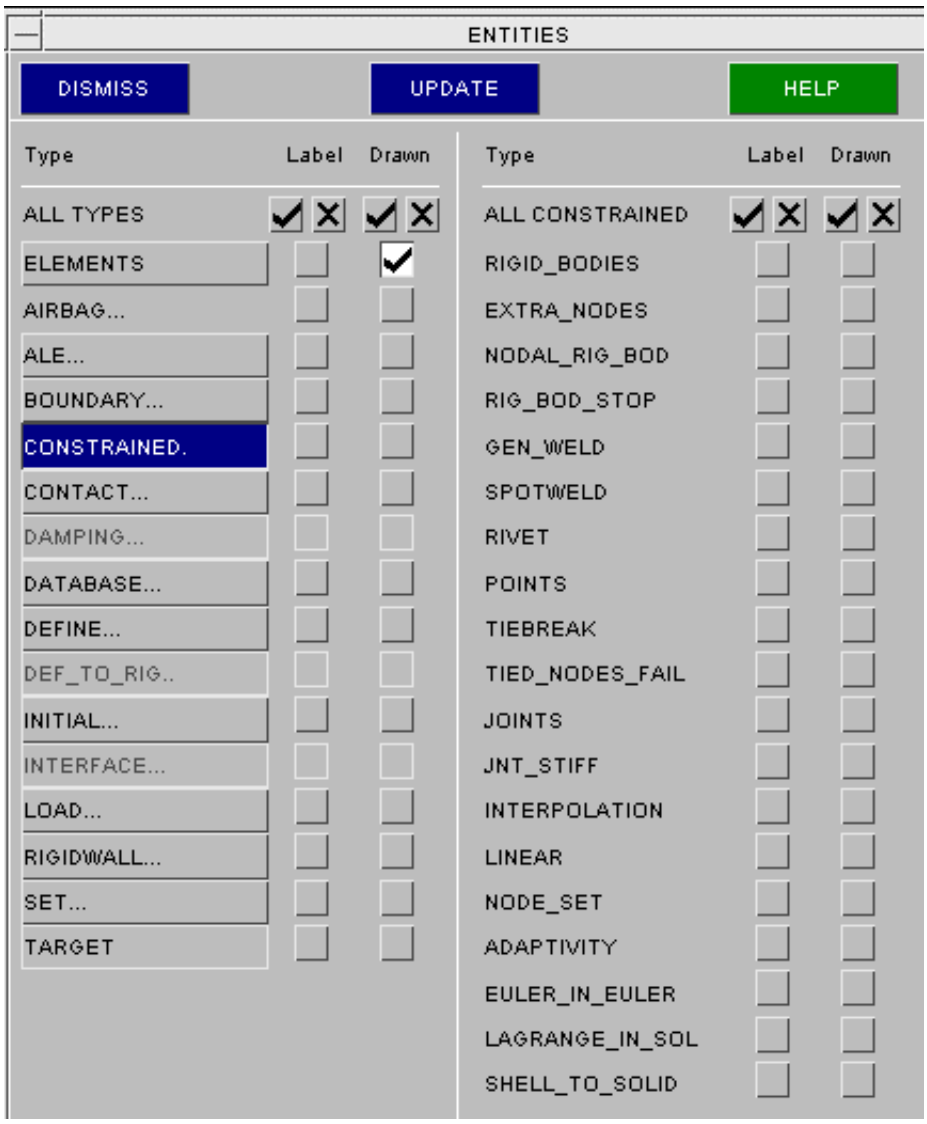
Combine `*CONSTRAINED_NODAL_RIGID_BODY`

There is a specific tool for CONSTRIANED_NODAL_RIGID_BODY (NRB) types called **Combine**. This is available through the main NODAL_RIGID_BODY panel. This tool combines multiple selected NRBs into one.

If all the selected NRBs exist in same layer (include file), new combined NRB and respective node sets will be created in the same layer, otherwise the current layer will be used.


NODAL_RIGID_BODY
M1: COMBINE 3 NRB(s)
Create
Delete
List
Help
Copy
Keyword
Check
Annotate
Modify
Sketch
Renumber
Combine
Apply
CONSTRAINED
All
None
Opt
Filter
Vis
Key_In
Sk
(M/L) NODAL_RIGID_BODY(
M1/NRBC300011
M1/NRBC300012
M1/NRBC350001
M1/NRBC350002
M1/NRBC2000000

Visualisation of ***CONSTRAINED** items



All **CONSTRAINED** items except **_LINEAR, _RIGID_BODY_STOPPERS** and **_LAGRANGE_IN_SOLID** are explicitly drawn and labelled, and all sub-types can have their constituent sets, parts or whatever displayed. Visibility is controlled by the **ENT**ity Viewing , **CONSTRAINED**panel.

Labelling of ***CONSTRAINED** items within PRIMER.

LS-Dyna has optional labels for some *CONSTRAINED items (e.g. *CONSTRAINED_NODE_SET_ID): the conversion from "keyword" to "formatted" input that precedes every LS-Dyna analysis converts them from discrete definitions to attributes applied to other items.

For internal consistency, PRIMER assigns new labels to everything that does not already have a label and that can be defined "once or many times", so *CONSTRAINED definitions are given labels based on their order of appearance in the keyword input file.

PRIMER's labels:

- May safely be ignored - you don't have to worry about them if you don't want to!
- Are treated sequentially, starting at 1. (Thus **CNST_1, CNST_2, ... CNST_n**)
- Are not grouped by sub-type: **CNST_1** might be a **NODE_SET, CNST_2** a **JOINT** - they are based solely on the order in which they appear in the input deck. Each *CONSTRAINED definition encountered gets the next label in the sequence.
- Are used in selection menus (eg for blanking, deletion, etc). Are also used in the output deck when defining what is referenced by what.

Because PRIMER groups ***CONSTRAINED** definitions by type when they are written out or copied (all **JOINT**s together, etc), and because labels are assigned in order of appearance, the labels assigned to these items may change

when decks are written out and read in again, unless the _ID option is used.

*CONSTRAINED_JOINT specific annotate tools.

There is a specific tool for joint constained types called **Annotate**. This is available through the main CONSTRAINED_JOINT panel or on individual CONSTRAINED_JOINT edit panels. This tool annotates the joints with nodal positions and rigid body information. It can be useful when creating or checking joints to ensure nodes and rigid bodies are defined in the correct order.

CONTACT: Defining Contact Surfaces.

CONTACT(_sliding)
- Top level menu
- Creating a new contact
- Keyword editor
- Copying
- Editing
- Deleting

Penetration Checking
Visualisation
Duplicates during input

The ***CONTACT** keyword has 10 sub-categories. The following can be edited in Primer:

***CONTACT**	Traditional LS-DYNA contact types.
***CONTACT_1D**	One-dimensional "slidelines" for reinforcing bars in concrete.
***CONTACT_ENTITY**	Geometric definitions of rigid bodies. Generally used when DYNA is coupled to other codes.
***CONTACT_GUIDED_CABLE**	A sliding contact that guides 1D elements.
***CONTACT_INTERIOR**	Internal contact for solid elements. Used to prevent crushable materials collapsing in on themselves and turning inside-out.
***CONTACT_RIGID_SURFACE**	A rigid surface contact definition.

The following contact sub-types cannot be edited in PRIMER.

***CONTACT_AUTO_MOVE**	Moves the master surface in acontact definition to close the initial gap between the master and slave surfaces.
***CONTACT_COUPLING**	Defines a coupling surface for MADYMO to couple LS-DYNA with deformable and rigid parts within MADYMO
***CONTACT_GEBOD**	Contact between a "Gebod" dummy and the structural FE mesh.
***CONTACT_2D**	Two dimensional contact for use with 2D and Axisymmetric elements.

This figure shows the main contact menu.

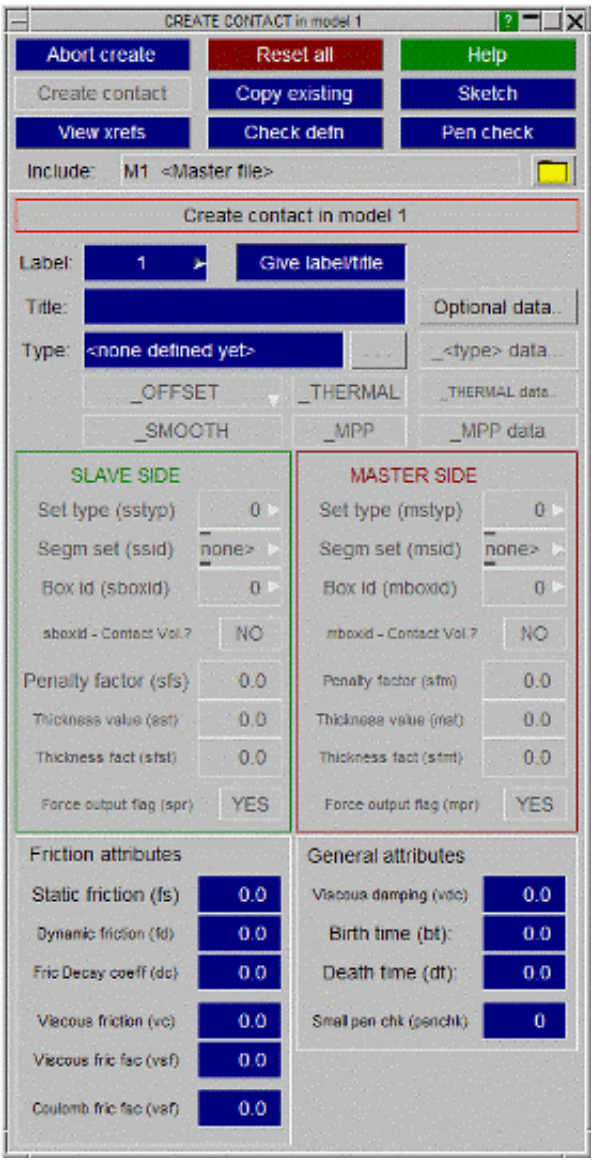
PEN_CHECK Runs the penetration checker for contacts.

> The penetration checker detects initial penetrations and crossed edges. It displays them graphically and also generates "null beams" on the crossed edges to identify them during external remeshing. Its use is summarised below.

The functions currently available have their standard meanings. (See section 5.1.1)

Only is a function for dispalying contact contents, and is described below.

CREATE Making a new contact surface.

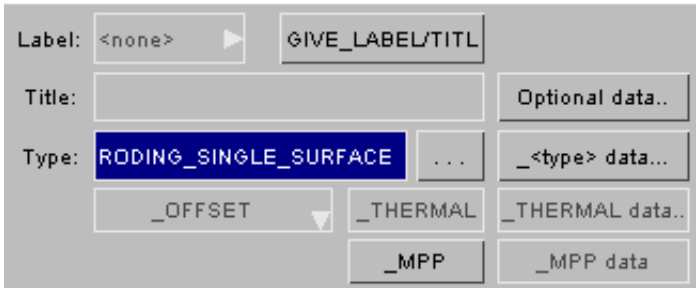


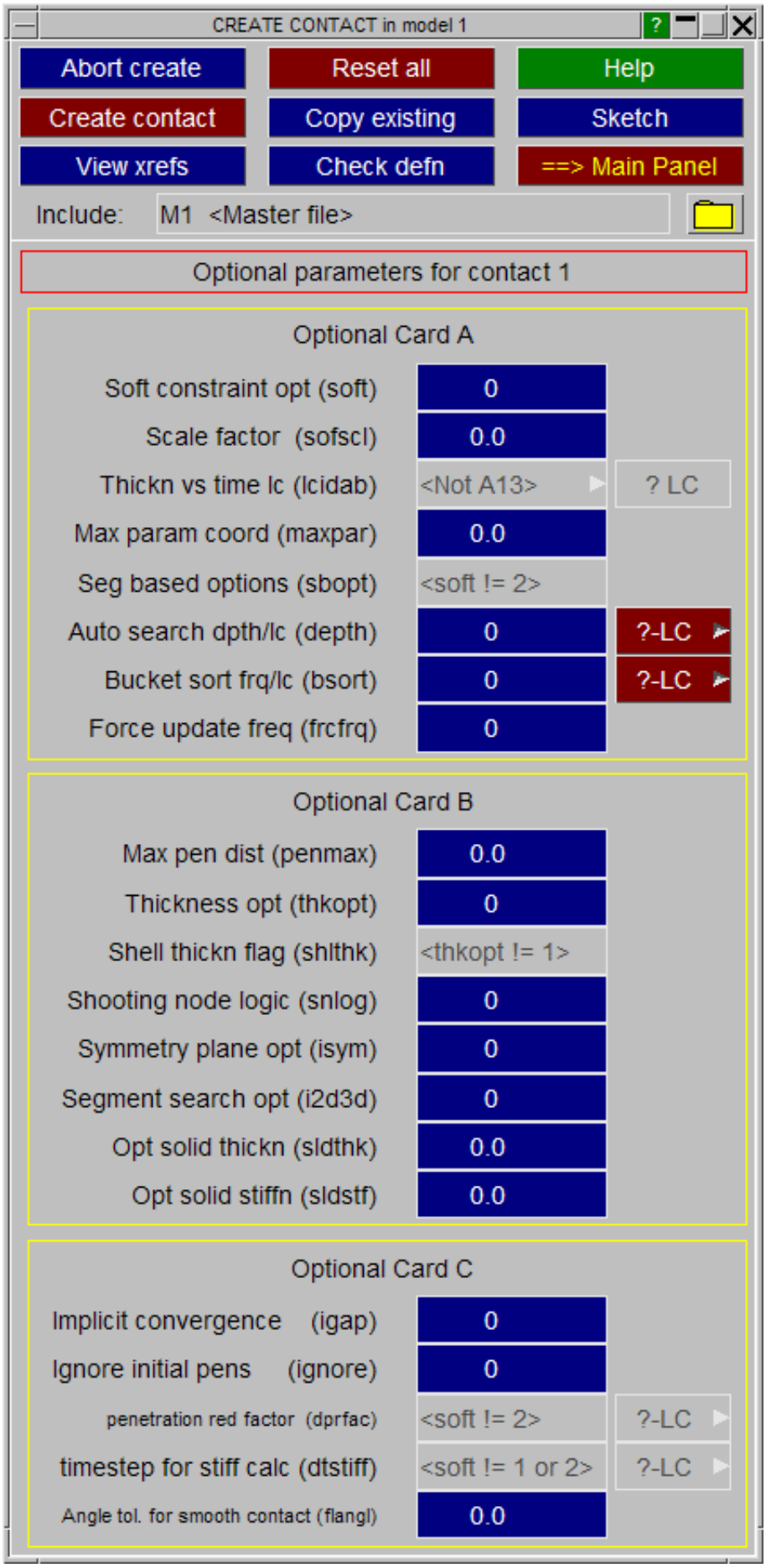
The figure on the left shows the initial state of the contact creation panel: no type has been given yet, so both master and slave side definition areas are greyed out

The figure on the right shows the same panel now that an **`ERODING_SINGLE_SURFACE`** contact type has been defined. This is a single surface contact, so only the slave side is available; had it been a two-sided contact type then both master and slave sides would have been available for input.

The data on the front panel here is required (even if zero) for all "sliding" contact types, but there are extra, optional data entries at the top of the panel that may be required:

GIVE_LABEL/TITLE Allows you to define the name and label.
LS-Dyna has an optional **_TITLE** suffix to the **`*CONTACT_<type>`** keyword that is enabled if this is defined.





Optional data...

Defining data on optional *CONTACT cards A, B, C, D & E

All contact types may have additional data for optional cards A, B, C, D & E in LS-DYNA.

The exact options available will depend upon the contact type, for example the Airbag thickness vs time loadcurve is not valid for the (eroding) contact type used in this example.

The red [**?-LC**] symbol on this panel is used to denote the fact that a loadcurve value is implied in this context if a negative number is input, whereas zero or a positive number is a simple constant. Pressing this button will give a selection menu of possible loadcurves.

(A green [**?LC**] means that a positive value implies a loadcurve.)

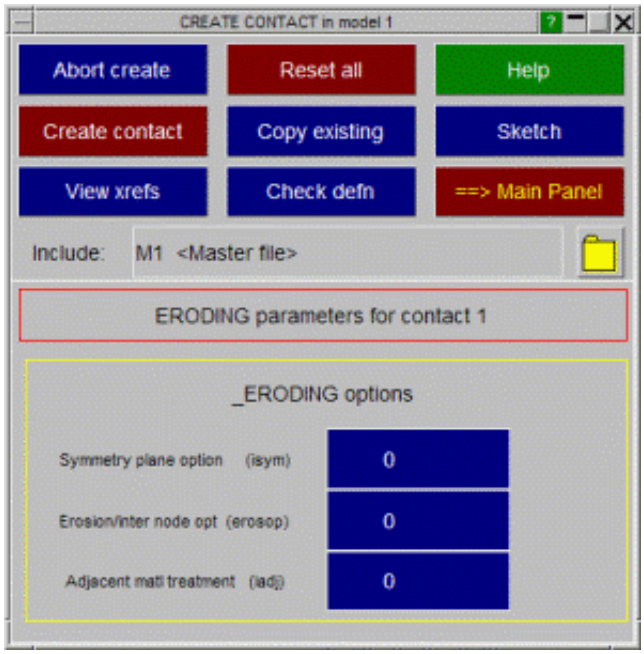
When you have completed entries on this panel **==> MAIN PANEL** returns you to the master panel shown above.

_<type> data... Special data input for this contact surface type.

A few contact surface types have mandatory <type> data cards. If this is the case the [_<type>_data...] button on the main panel will be live.

In this example the **`_ERODING_...`** contact type requires the extra data shown here; but other contact types will require different data.

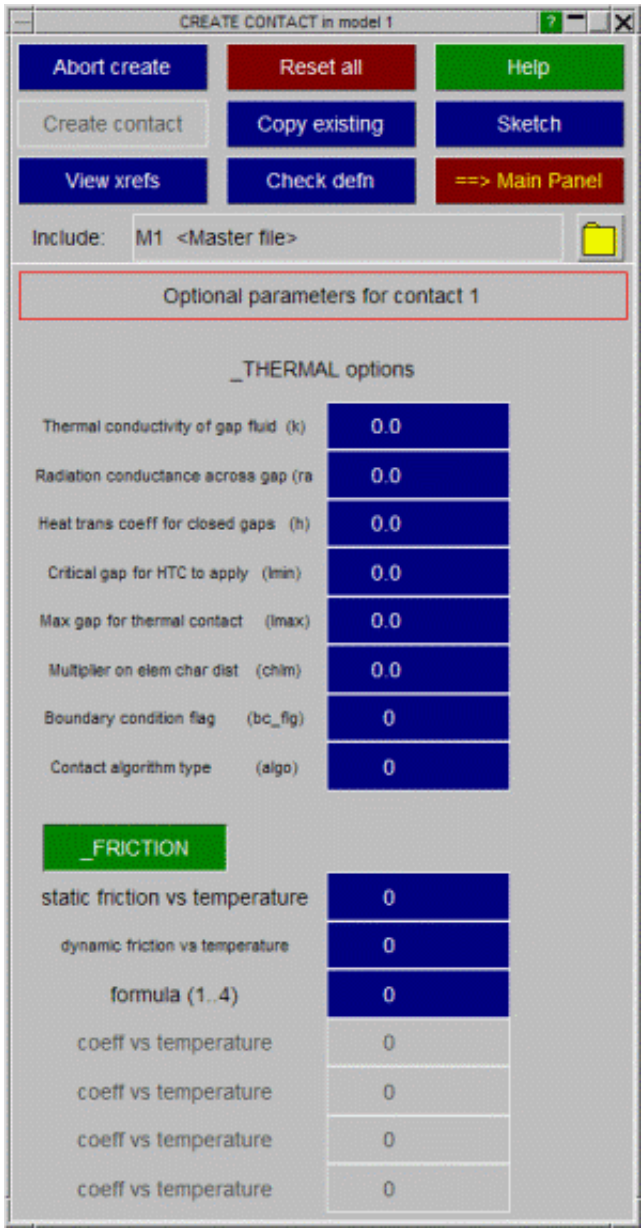
As with the optional data cards, **==> MAIN PANEL** will return you to the main contact input panel.



_THERMAL data... Special input for thermal contact types.

Where a contact surface type supports the **_THERMAL** or **_THERMAL_FRICTION** qualifier, and this is turned on, the **_THERMAL data...** button may be used to enter the thermal parameters.

As with the other optional card panels the **==>MAIN PANEL** button will return you to the master contact definition panel.



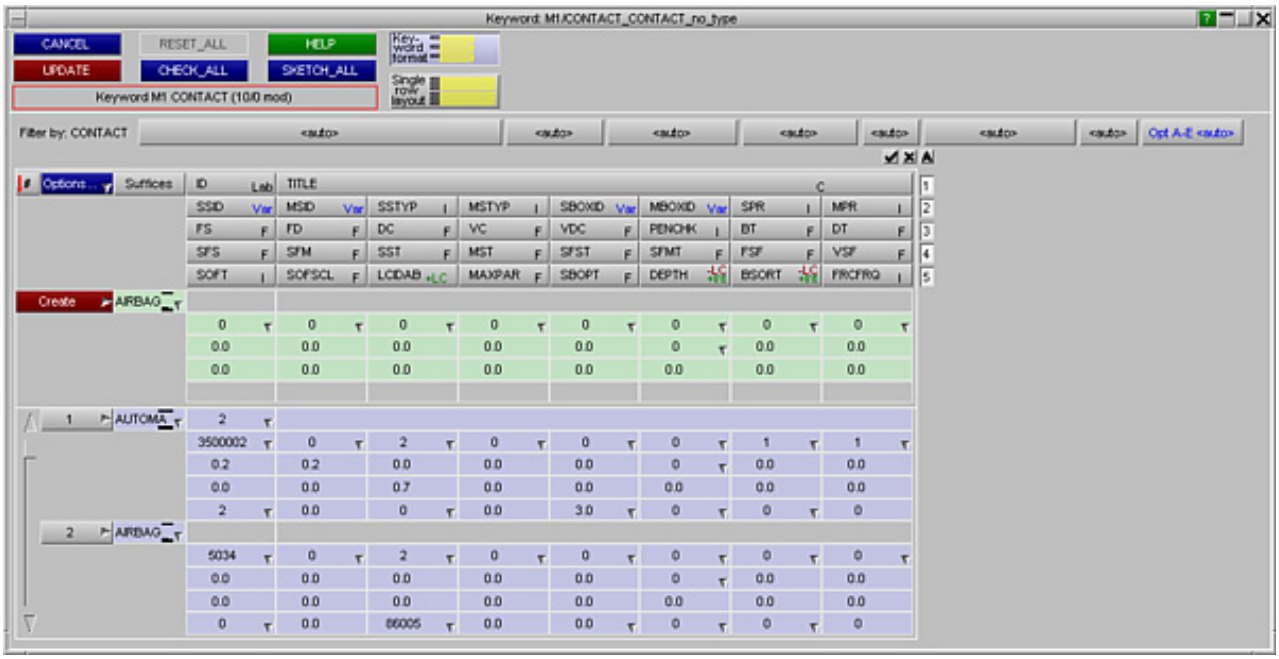
CREATE_CONTACT Saving the contact definition.

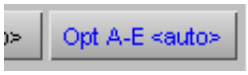
Once you have entered the minimum amount of data required to define this contact the **CREATE_CONTACT** button will be made live, and this permits you to save this definition. (If it is not live the missing fields will be highlighted in red.) The definition will be checked and any errors listed, an then it will saved permanently in this model.

Until you press this the definition remains volatile, and will be lost if you exit this panel in any other way.

ONLY will display only the contents of the contact.

KEYWORD Using the generic keyword editor to process contacts

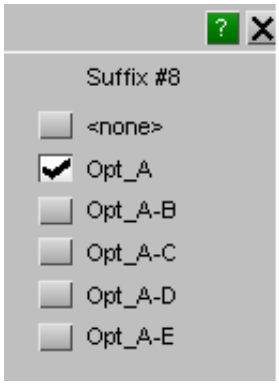




The generic keyword editor for **`*CONTACT`** works in exactly the same way as normal, with the exception that an extra "pseudo" keyword has been added to manage the display of the five optional contact cards A to E.

The reason for this is that all **`*CONTACT`** definitions occupy at least three rows of data, some use four, and if extra keyword suffices are added then more lines may be used. Adding all five optional cards makes each keyword entry very tall, and most optional fields are not used, so the keyword editor would show many lines - mostly full of zeros - for only one or two contacts if these were always shown.

Therefore, for each contact, you must choose how many optional cards are to be "live" by setting the pseudo-suffix "**Opt A-E**".



When the keyword editor is started each contact definition will have this suffix initialised to the current status, determined by scanning optional cards A to E for non-zero values or parameter usages, and will be preset accordingly.

If you want to add data to later rows, not currently active, you will first need to change this suffix to the appropriate setting, just like changing any other keyword suffix, whereupon the relevant rows will be shown in the editor.

This pseudo-suffix is purely an artifice in the keyword editor, the suffix will not be shown on the **`*CONTACT`** card when the deck is written to a keyword file.

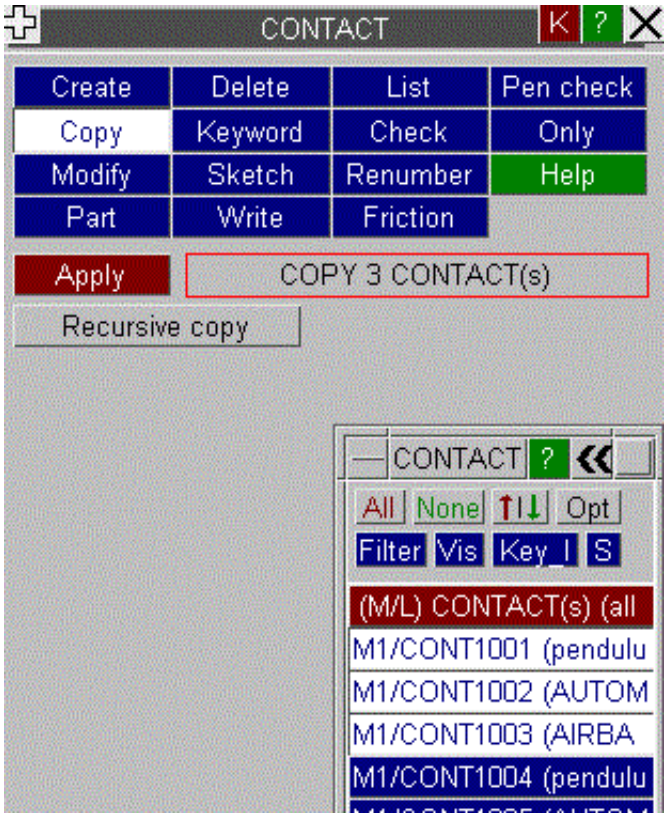
COPY Copying existing contact surface(s) to make a new one(s).

You can **COPY** any number of contacts, in multiple models.

Selection of subject models is from the menu shown in this figure.

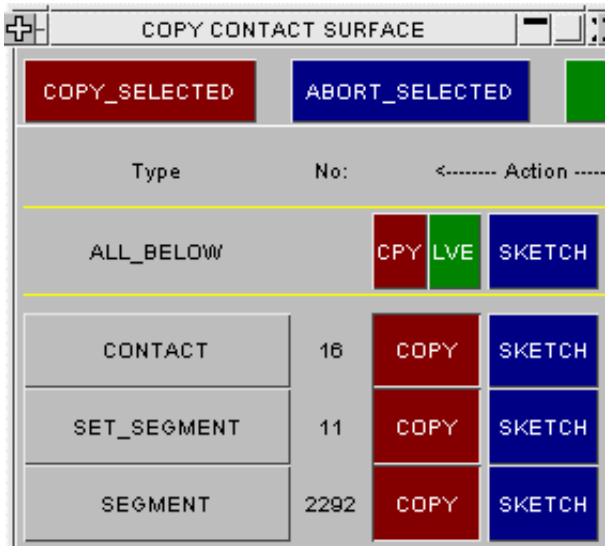
When **APPLY** is pressed you are asked to confirm what is to be copied, and then the operation is carried out.

For each model the <n> contacts chosen in that model are copied using labels <previous highest + 1> to <previous highest +n>, there is currently no control available over the new labels assigned.



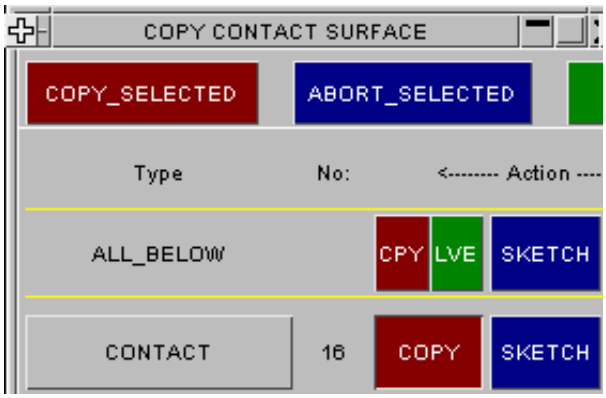
RECURSIVE_COPY Controlling the extent of a copy operation for segment based contacts.

If "recursive" copying is used then any segment sets, (and thus their associated segments), in the subject contacts are duplicated; and the newly created contacts will reference these duplicates. Without "recursive" copying the new contacts will reference the same segment sets as the old ones. This is illustrated in the examples below, using the same 3 contacts:



With **RECURSIVE_COPY** selected.

In the left box recursive copying has picked up segment sets and their constituent segment.



With no recursive copying.
In the right box, without recursive copying, only the contact definitions are copied.

Note that for contacts recursive copying only affects segment sets and their segments. A contact defined by parts, elements or nodes will not attempt to copy these in the "recursive" case.

MODIFY Modifying the attributes of an existing contact surface.

This functions in exactly the same way as **CREATE**, using the same panels as above. The only difference is that the initial state of the panels is already set with the attributes of the contact surface to be modified.

DELETE Deleting existing contact surfaces

The **DELETE** operation works exactly the same way as **COPY** described above, except that the chosen surfaces are deleted. The **DELETE_RECURSIVE** switch also operates in a similar fashion:

- If **DELETE_RECURSIVE** is switched on any segment sets, plus their segments, referenced by the contacts to be deleted are marked for deletion.
- If recursive deletion is not used only the contact definitions themselves are removed.

Note also that the standard deletion rules described in Section 6.4.1 still apply: contacts and/or segment sets will only be deleted if nothing else (which is to remain) depends on them.

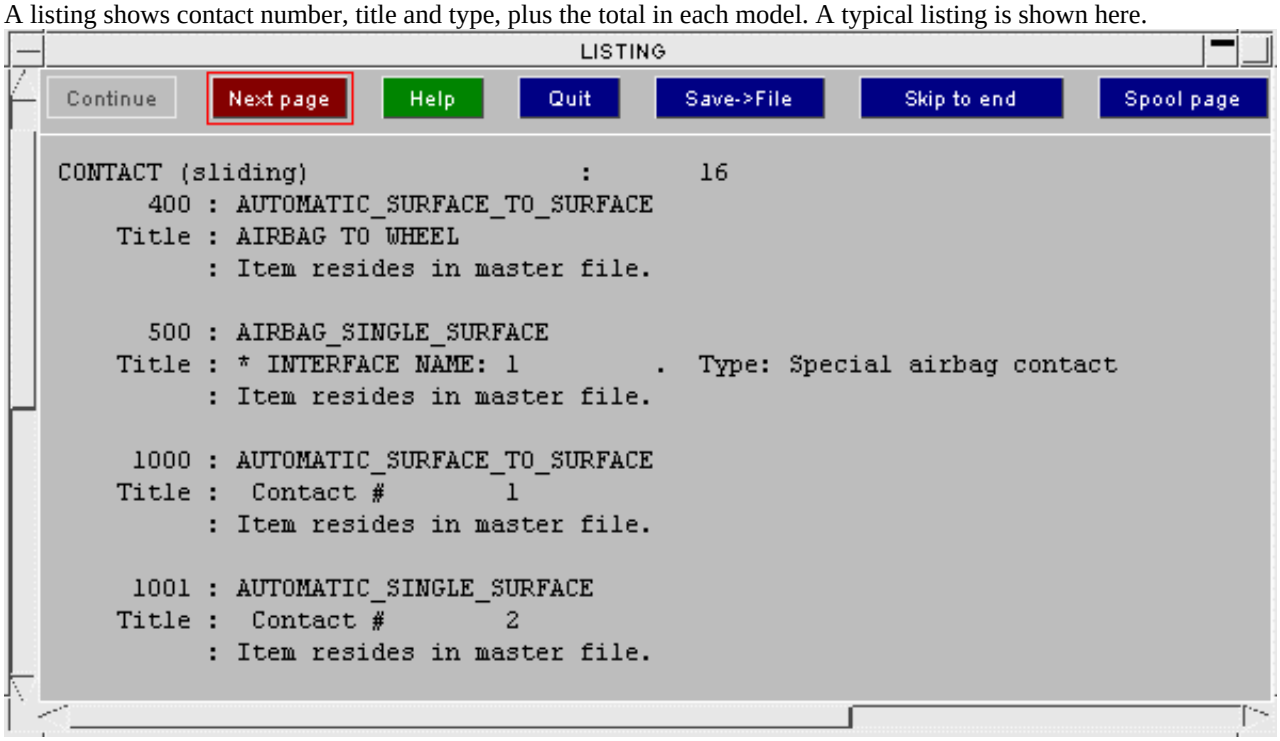
SKETCH Sketch the chosen contact surfaces on the current image

SKETCH allows the user to select and sketch individual contact surfaces on the current graphics image. Contacts are drawn in terms of the parts, elements, segments and nodes that they include.

LIST Summarise the contents of contacts

LIST allows the user to select any or all contacts and to list a summary of their attributes to the screen.

A listing shows contact number, title and type, plus the total in each model. A typical listing is shown here.



CHECK

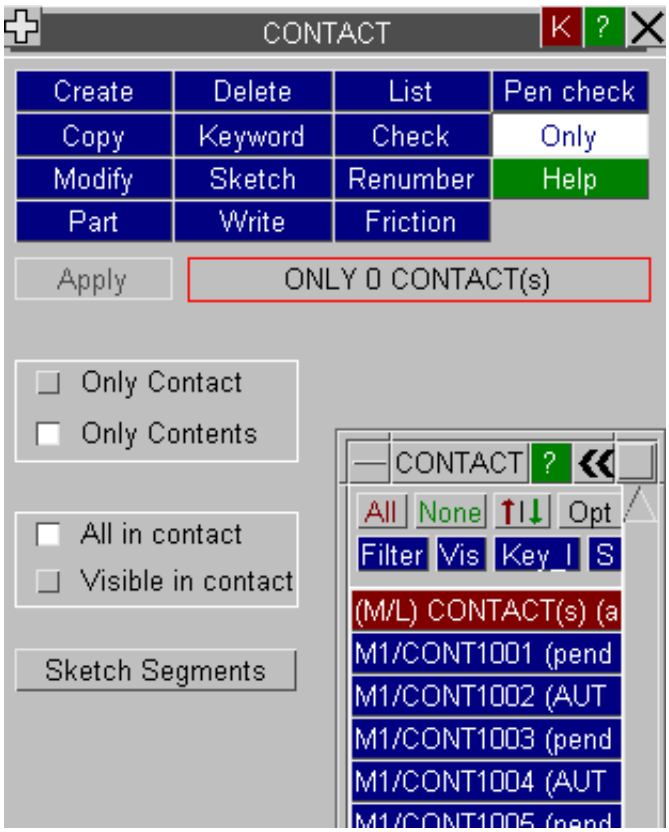
Runs the standard checking function on the selected contacts. Each contact will be listed either as "OK", or a summary of the errors encountered will be printed. (This is the same as the **CHECK_DEFN** command during contact editing.)

RENUMBER

Raises the standard renumbering panel for contacts in the chosen model, allowing you to renumber some or all of them. To renumber a single contact it may be easier to **MODIFY** it and update its label.

ONLY

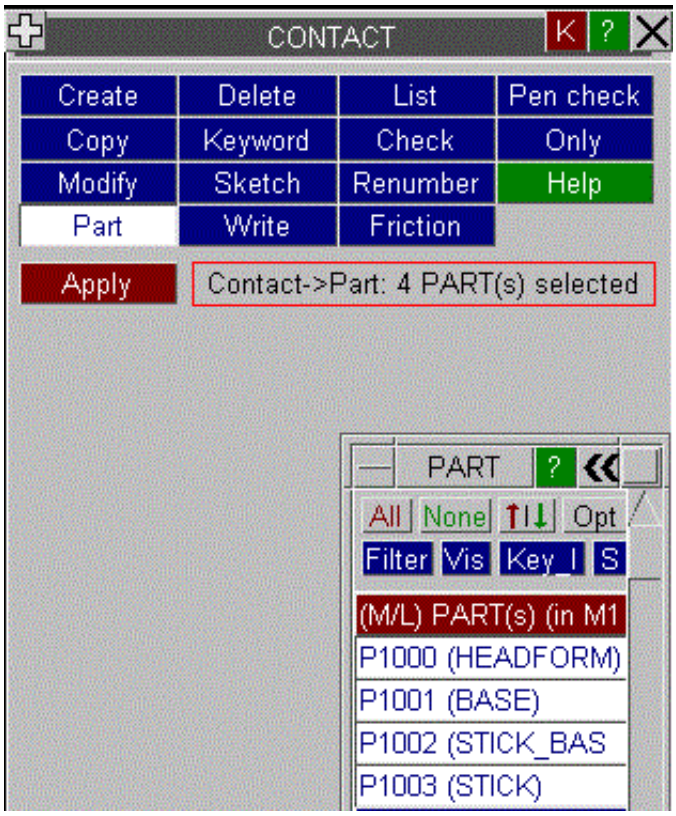
Allows you to visualize what is included in the contact.



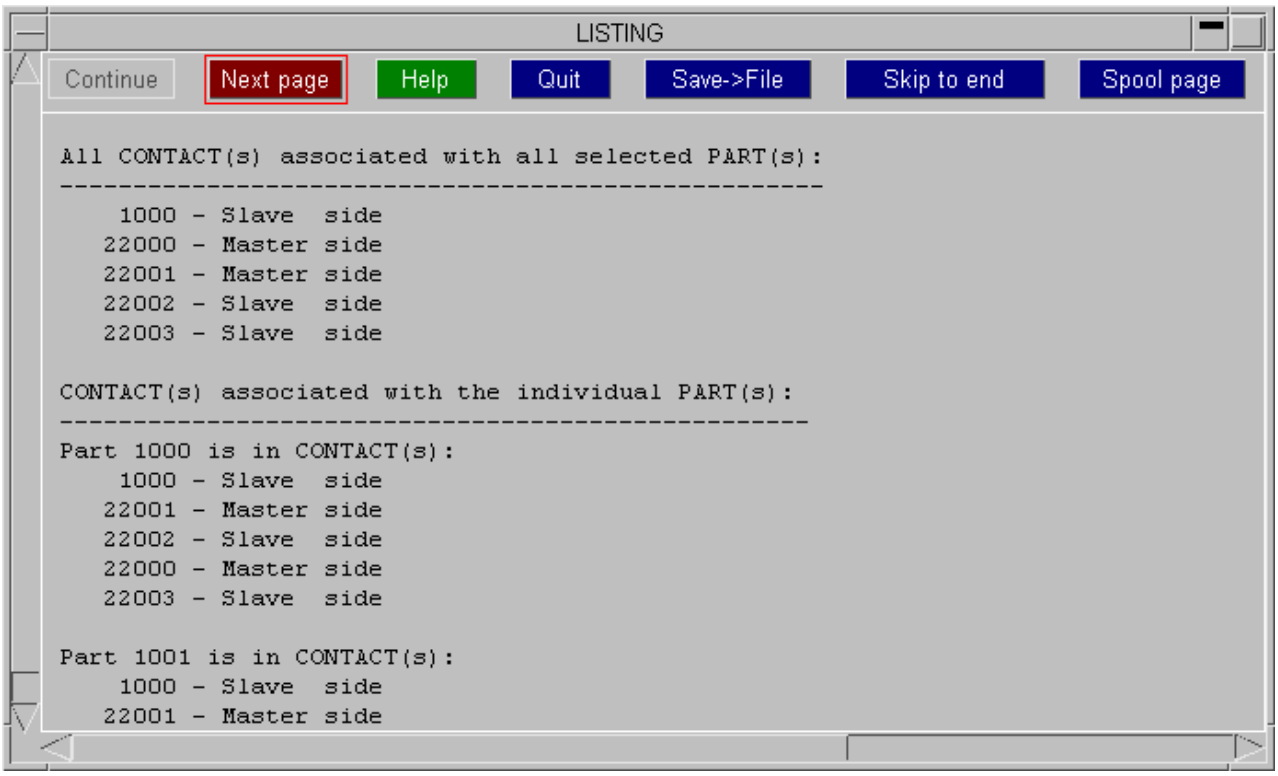
Select contacts you wish to display and click **Apply**. **Only Contact** will display the contacts selected. **Only Contents** will display the contents of the contacts selected. For the contents option, you can also optionaly choose to just display currently visible contents of the selected contacts.

PART

Allows you to get a popup listing of which contacts the selected part(s) are in.



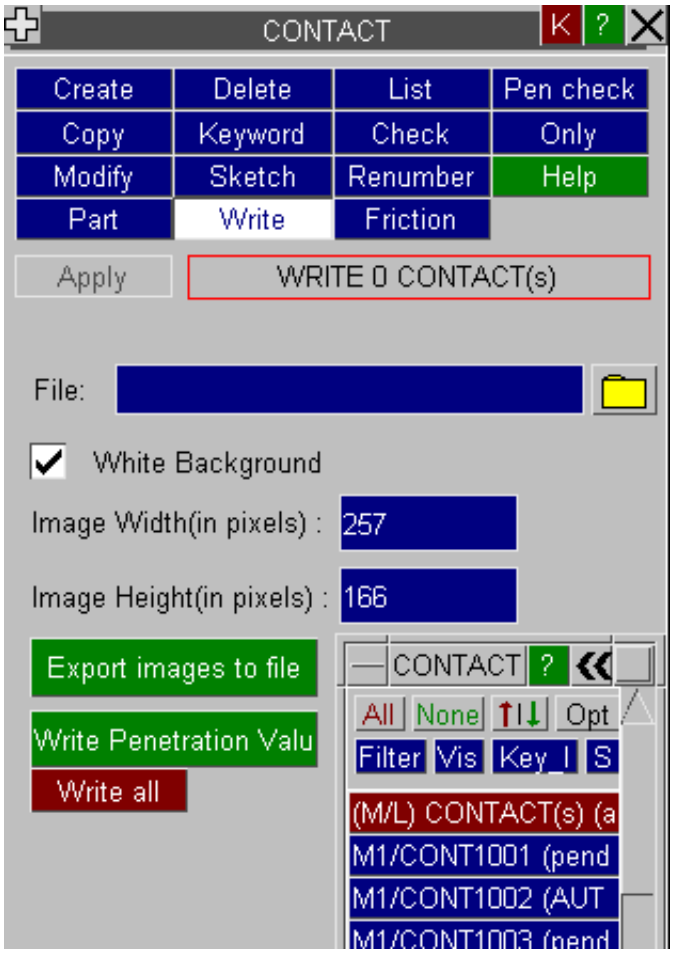
Select one or more parts and click **Apply**.



If multiple parts are selected, the output will start with a list of contacts that includes any of the selected parts and will later include details of individual parts.

WRITE

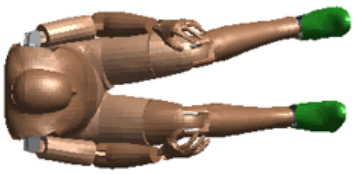
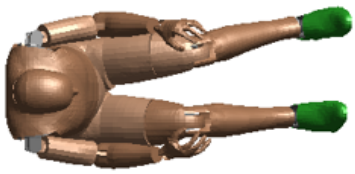
To write a Contact file, type a file-name into the text box or use the folder button to find the path where you want the file to be written.



Once a valid file-name and file-path have been selected, the user can choose to immediately write all the contacts which have been loaded by clicking on the **Write All** button.

The other option is to select the required contact from the contact list and then write the selected contacts by clicking on the **Apply** button which will become active if at least one contact is selected.

The Images are written out on a white background by default but this can be changed to the currently loaded background colour by clicking on the **White Background** toggle button. You can also change the image height and width by entering the required pixel values into the corresponding text boxes.

	Combined Image	Slave SET Type	Slave ID	Slave Image
		Part Set	1000	
		Part Set	1001	
		Part Set	1003	

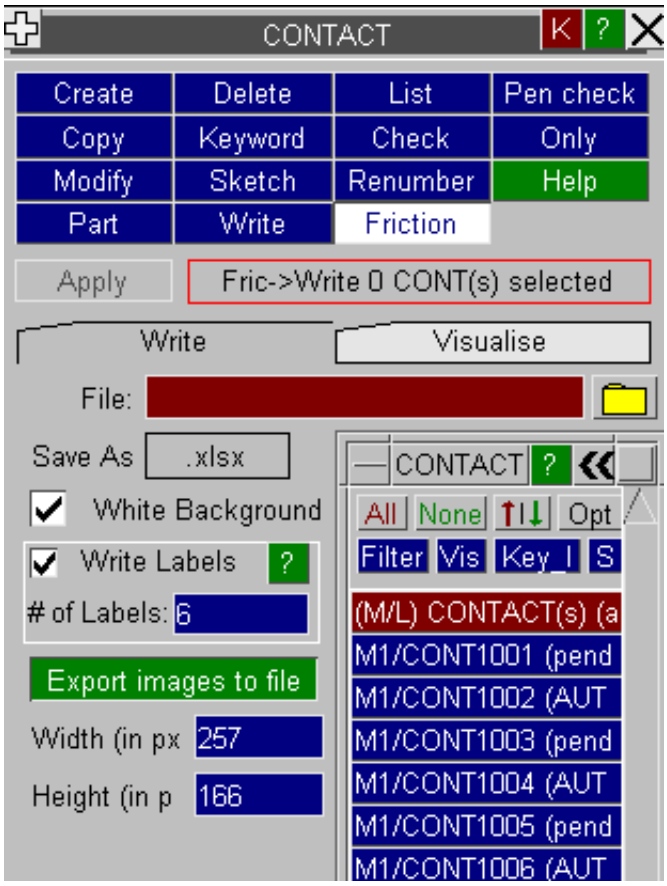
The user also has the option of deciding whether the images or the penetration data is written into the excel file. This can be done by using the **Export Images to file** and **Write Penetration Values** toggle buttons.

FRICTION

There are now a variety of ways of setting friction coefficient values used in contacts:

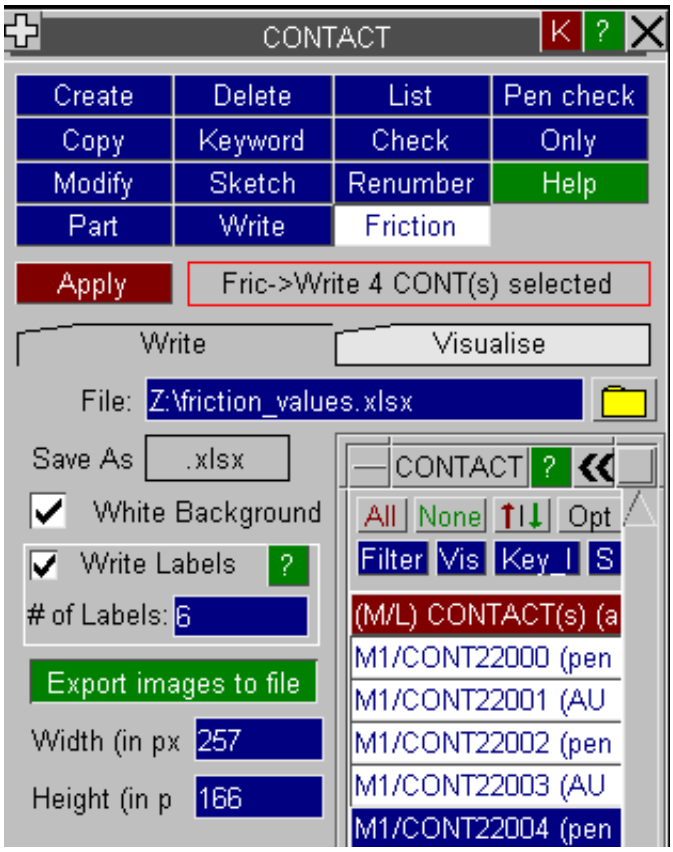
- The *COTACT card.
- The *CONTROL_CONTACT card.
- *PART_CONTACT cards.
- *DEFINE_FRICTION cards.

If you have a combination of the above methods, it can be difficult to understand which friction coefficient values are used in your model. **FRICTION** tools in PRIMER allow you to investigate these values.



Write Friction Values

Select one or more CONTACT(s) for which you wish to write friction details into an Excel or a CSV file.



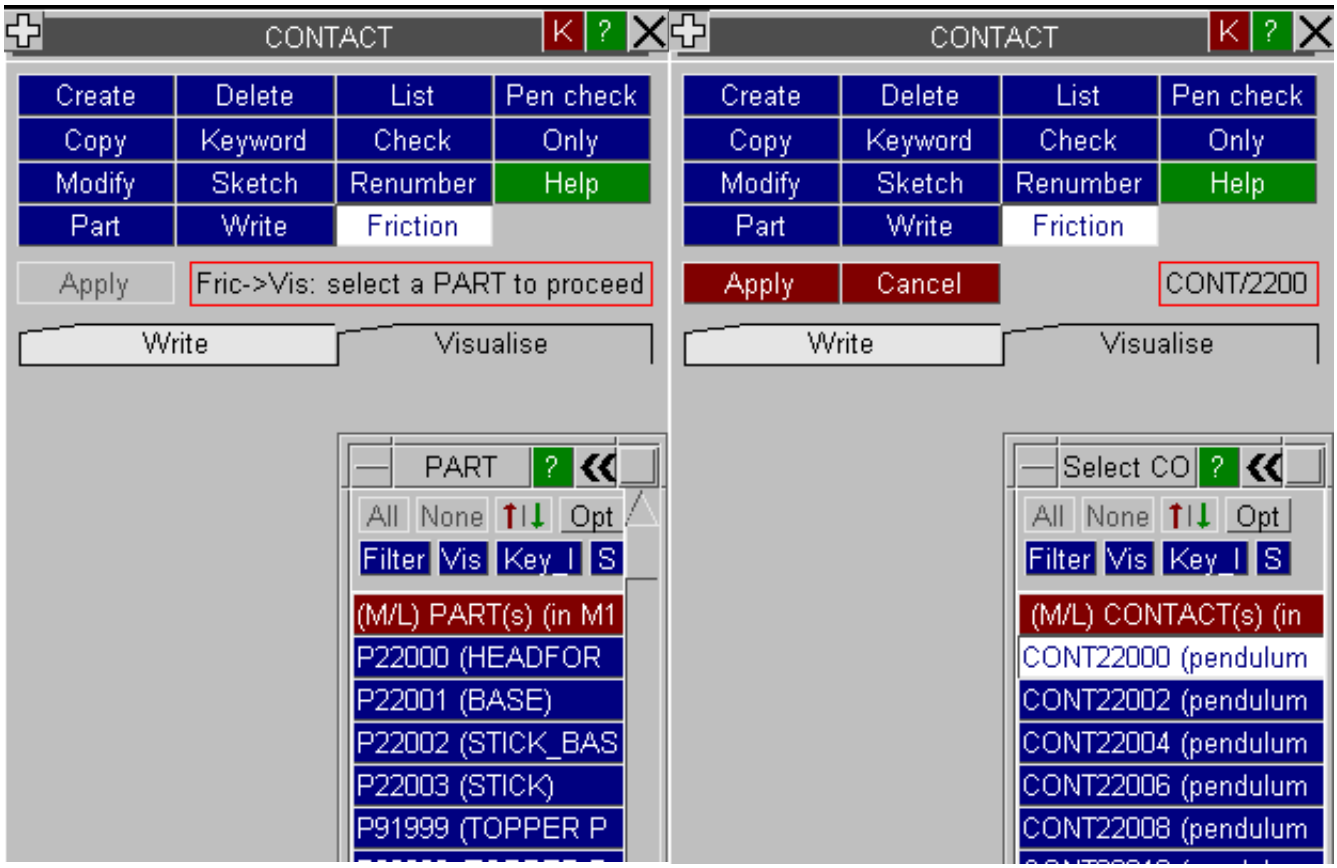
Type a file-name into the text box or use the folder button to find the path where you want the file to be written. Once a valid file-name and file-path have been selected, hit **Apply**.

The Images are written out on a white background by default but this can be changed to the currently loaded background colour by clicking on the **White Background** toggle button. You can also change the image height and width by entering the required pixel values into the corresponding text boxes.

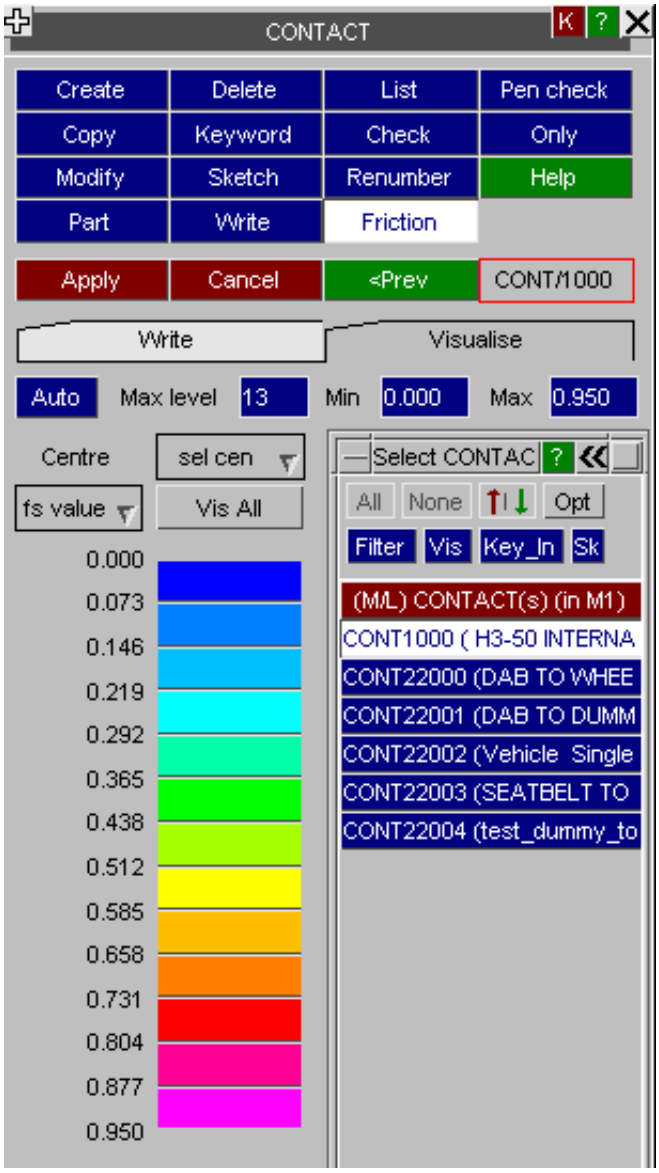
1	Contact ID	Friction Type	#Parts in Slave Side	Slave Part ID(s)	Slave Part(s) Image	#Parts in Master Side	Master Part ID(s)	Master Part(s) Image	FS	FD	DC	VC
2	22002	*DEFINE_FRICTION card	2	21204 21103					0.15	0.16	0	0
3	22002	*DEFINE_FRICTION card	2	20204 20581					0.17	0.18	0	0
4	22002	*DEFINE_FRICTION default	784	1000 1001 1002 1003 1004 1005					0.34	0.24	0	0
9	22005	*PART_FRICTION	1	21204					0.2	0.2	0	0
10	22005	*CONTROL_CONTACT				1	21150		0.124	0.345	0	0
11	22005	*PART_FRICTION				1	21199		0.4	0.5	0	0

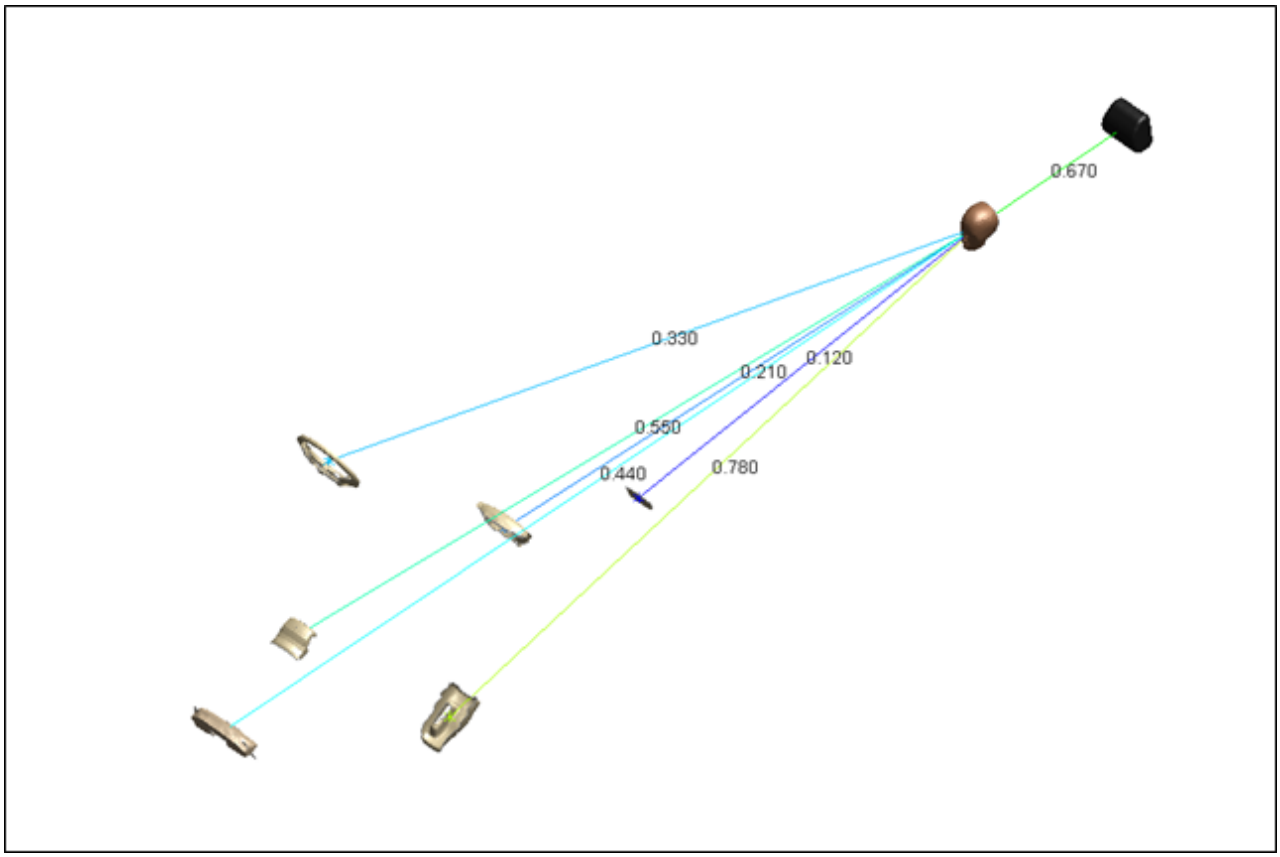
Visualise Friction Values

Once in a **Visualise** mode, select a PART to investigate contact friction values. After part selection, list of all the contact which refers to the selected part will be shown.



Select a CONTACT from the list followed by **Apply.** All the parts in the selected contact will be shown in the exploded views. Lines with friction values will be drawn between the selected part and all the exploded parts.





The number of contour levels can be between 1 and 13. You can control the number of levels used for contouring, or provide min and max value to set up contour bands automatically.

PEN_CHECK Checking initial penetrations and crossed edges

The contact penetration checker may be run from:

- The **PEN_CHECK** command in the top level menu here
- The **PEN_CHECK** command in any contact create/edit panel
- The main **MODEL > CHECK** command as part of model checking.

Calling the penetration checker from the top menu panel.

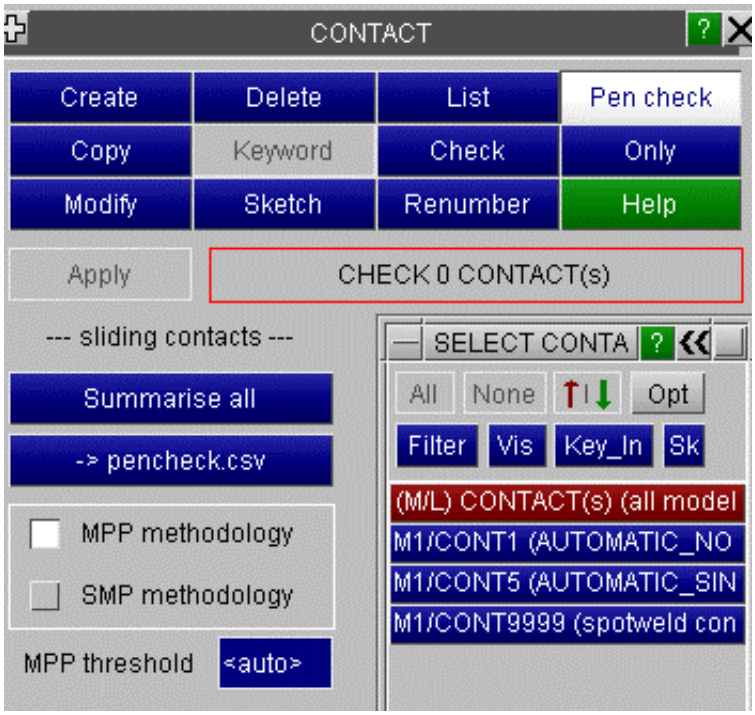
Summarise all Runs the checker on all contacts in turn in "summary" mode, producing a listing of penetrations and crossed edges for each one, but no details. Note - the number of penetrations may exceed the number of penetrating nodes (which is reported in other contexts) as there may be >1 penetration on a node.

This is useful as an initial scan to look for problems meriting further attention.

-> pencheck.csv writes all qualifying (above user threshold) part vs part interactions to a .csv file.

Explicit selection from menu

Runs the checker in detailed mode on the contact selected.



By default the contact check will be performed using MPP methods with the exception of the minimum threshold for reported penetrations. Primer uses the minimum of the SMP threshold (calculated from element sizes) and 0.001 (the hard-wired LS-Dyna threshold). This is far more sensible for models in metre units. However, the user may over-ride this and apply his own value for MPP threshold.

Calling the penetration checker from the Create/Edit panel

The **PEN_CHECK** command in this context runs the checker on the current (scratch) contact definition currently being edited.

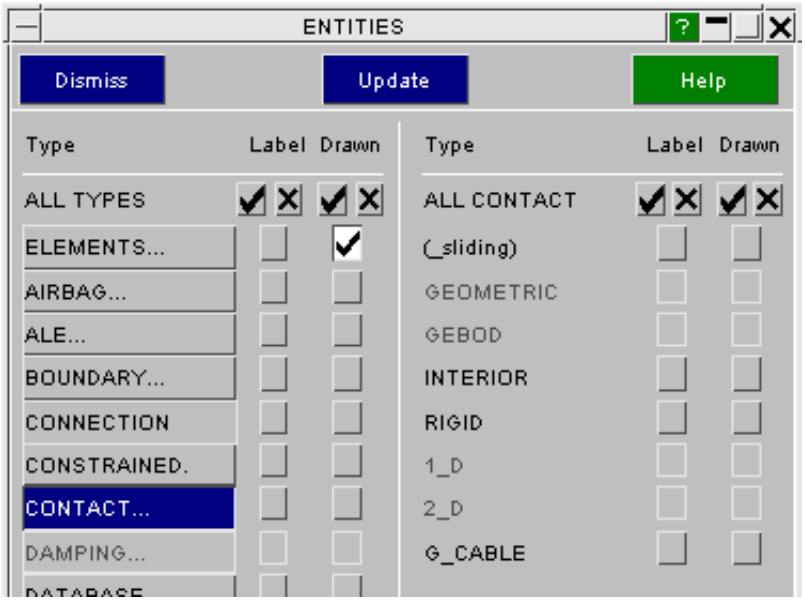
Contact penetration checking is covered in more detail in section 5.3

Visualising contact surfaces

Contacts may be selected for explicit display in the **ENT**ity Viewing panel

panel.
They are drawn in terms of the parts, elements, nodes and segments that define them. Where boxes are used to delimit contacts these are displayed if switched on in the "Associated data" section.

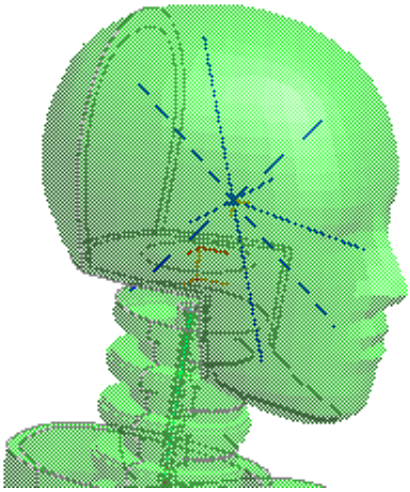
They may also be drawn via the **SKETCH** functions above.



Special display modes for contact surfaces

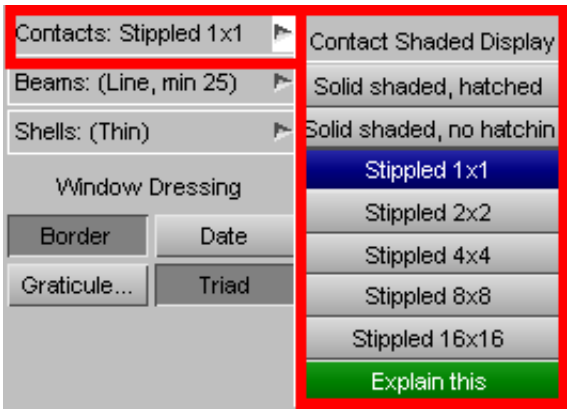
Since contacts are drawn in terms of the structural elements or segments that define them it would be difficult to distinguish a contact from normal structure if both were rendered in the same way. Therefore PRIMER use different forms of display to distinguish contacts from ordinary structure.

By default contacts are drawn using a technique called "stippling", as shown here, in which alternate pixels are omitted. This gives a distinctive slighty fuzzy appearance, but also allows graphics of other items behind the contact to be visible.



In this example a contact covers the whole of the dummy's head, but internal structure (beams and accelerometers) is visible through the stippled contact.

Other display modes for contacts are available in the **Display Options** panel.



Various degrees of stippling give different effects. It is also possible to revert to the "old" PRIMER method of drawing the contacts solid shaded with cross-hatched lines added, however this is slow to render and is not recommended.

Rather than trying to describe the effects here it is suggested that the curious try experimenting

Duplicates during input

`*CONTACT` cards can have the suffix **`_TITLE`** in which case they must also be given a specific label. If two or more such cards have the same label then normally PRIMER would treat this as an input error and reject the input deck.

However it transpires that LS-DYNA has some special logic to deal with this situation, meaning that duplicate contact definitions are (semi) legal:

- Before LS-DYNA R7: It issued a warning and continued to run with both contact definitions, leaving them both as having the same label. This could make post-processing confusing!
- From LS-DYNA R7 onwards: It also issued a warning and continued to run, but relabelled the duplicate contact to the next "highest + 1" label.

Therefore from PRIMER 13 onwards the treatment of duplicate contact cards during keyword input has been modified as follows:

- Duplicate contact cards are read silently, and the excess definitions are given temporary internal labels that prevent label clashes.
- When keyword input is complete these duplicates are listed on the screen, together with their include files and whether or not they are referenced.
- The user must then choose one of three options for dealing with them:

`Relabel`	The excess definitions are kept. but are relabelled to "highest + 1" labels, respecting any label ranges defined for their include files. This is the LS-DYNA behaviour, although the label that PRIMER assigns may not be identical to that assigned by LS-DYNA. If the deck is written out then the duplicate contact definitions will remain, but with their newly assigned labels.
`Clone`	The excess definitions are deleted, and are replaced with PRIMER "clone" definitions that refer to the first instance of the contact that was read. In PRIMER a "clone" is a reference to a keyword, and not a keyword itself. It means that only a single definition of a given labelled keyword exists, but on keyword output this definition is duplicated exactly in every include file that has a "clone" reference to the original. Therefore if all the original duplicate definitions were identical to the first one read then on output nothing will appear to have changed, however if there were any differences then these will be lost as teh 2nd and subsequent "cloned" output keywords will be identical to the first.
`Merge`	The excess definitions are deleted, leaving only the first one read. Therefore on output the 2nd and subsequent definitions encountered will disappear completely.

Warning:

Although LS-DYNA will accept this practice ***it is recommended that you do not use it***. Not only is it a "hostage to fortune" that relies on non-standard and undocumented LS-DYNA behaviour, but also the labels ultimately assigned to contact surfaces in LS-DYNA will depend on the order in which they are read, and changing include file order will affect this.

It is strongly recommended that you either eliminate duplicate definitions, or relabel them explicitly to make them unique.

Note:

This treatment of duplicate contact definitions is not standard PRIMER behaviour, and applies only to the **`*CONTACT`** keyword.

The standard treatment of duplicate labelled keyword is controlled by the **Options** panel during keyword input, and is described in section 3.2.1**Options**: Permit duplicate definitions

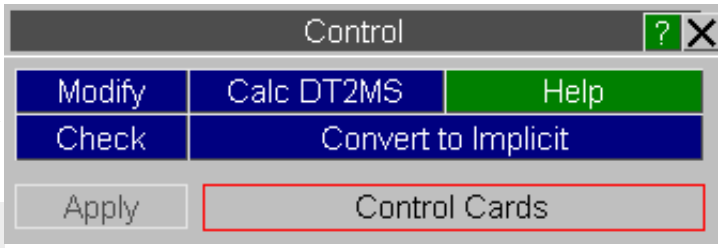
CONTROL: Defining Analysis Control Cards.

- Main CONTROL menu
- Modification
- Checking

The ***CONTROL** keyword in LS-DYNA refers to the unique keywords which control the main parameters of an analysis.
Each control card occurs either once or not at all, and none are labelled, therefore the PRIMER control and editing panels are slightly non-standard. Merging control cards presents some special problems - see the notes on this below.

Note for users of PRIMER prior to release 8.2: The layout of the control card editor has been totally revised, since the increasing number of control cards (44 in LS960) made the original panel unwieldy. In addition the distinction between "create", "modify" and "copy" modes has been removed and replaced by a single panel that performs all these functions.

The main CONTROL panel.



Since each control card can only exist once or not at all in a model the concept of separate "create", "modify", "copy", etc modes has been removed.

MODIFY	Maps the control card editing panel, in which cards can be created, modified, deleted and copied from other models.
CHECK	Runs the standard check routines on control cards.
CALC DT2MS	calculate relationship between %age added mass and timestep
CONVERT TO IMPLICIT	Make modifications to the model to convert it to an implicit analyses.

As with all PRIMER **MODIFY** functions edits and other changes only take place on a "scratch" definition, which is only made permanent when explicitly **UPDATE**d.

CONTROL_CHECK_SHELL (**CHECK SHELL**) and **CONTROL_MPP_DECOMPOSITION_TRANSFORMATION** (**MPP DECOMP**)are open ended cards with their own edit panels which can be accessed directly or via **MODIFY**

CALC DT2MS is only active if DT2MS on the *CONTROL_TIMESTEP is set to 0 or less.

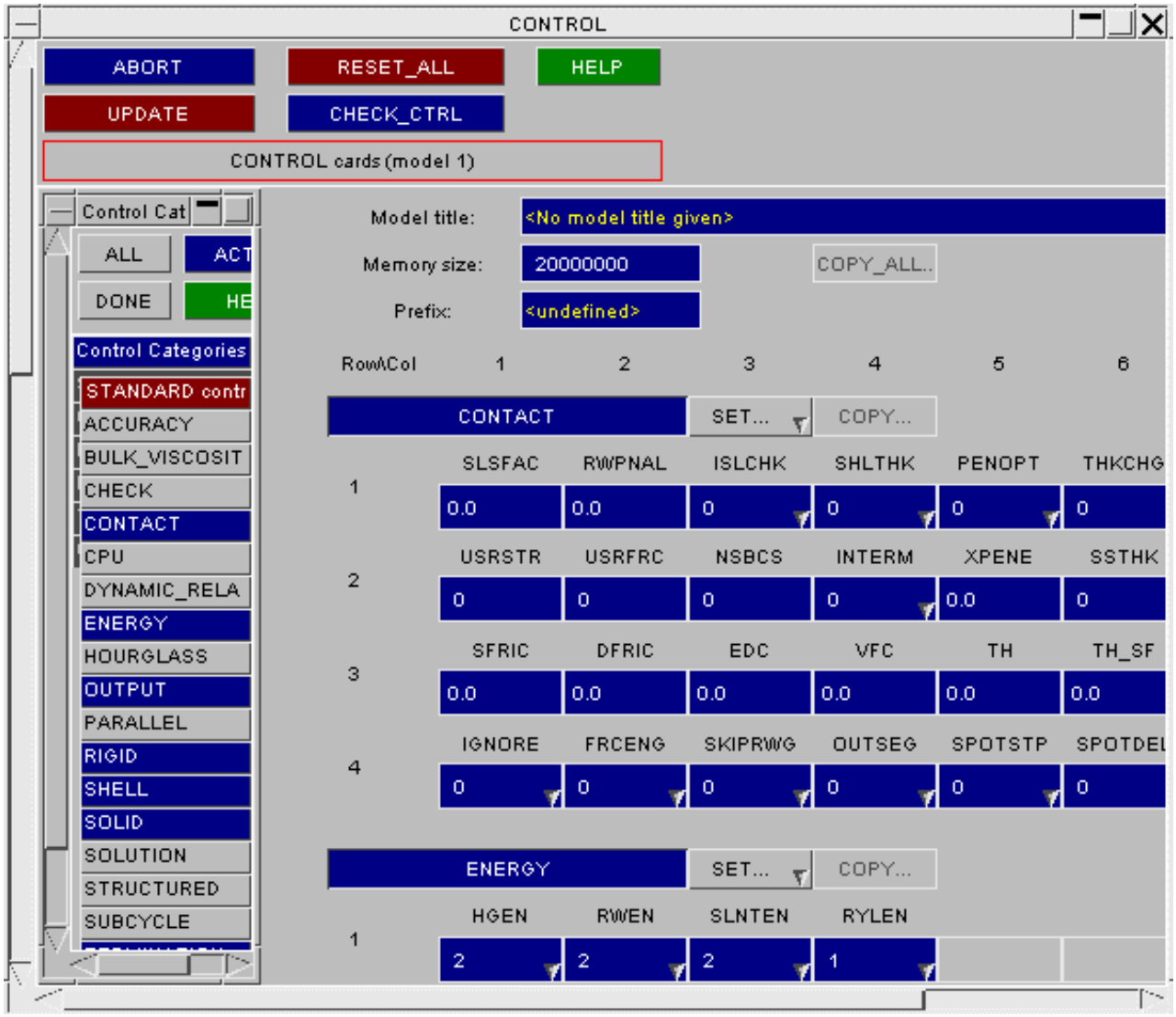
MODIFY: Creating, editing, deleting and copying control card definitions.

The single panel shown below is used to carry out all these operations.

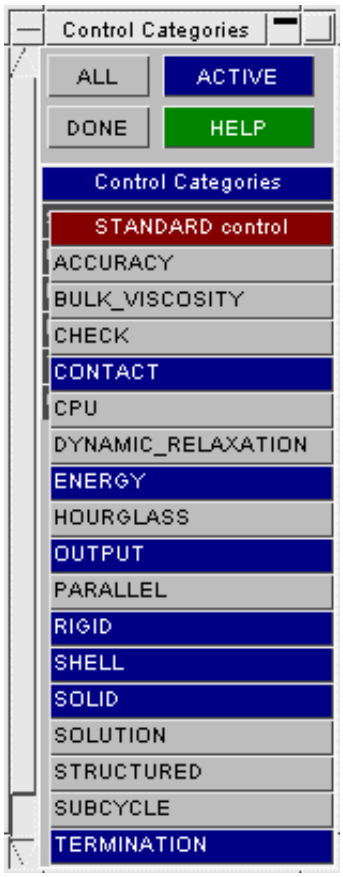
For ease of selection, Control card are now grouped into 6 categories.

ALL available Control options will be displayed (but not activated) by pressing **ALL** and all the active ones by pressing **ACTIVE**

All changes in this panel are performed on a "scratch" definition, and changes only become permanent in the database when **UPDATE** is pressed.



Selecting which control cards are displayed.



The scrolling menu on the left lists all control cards of current category (in this case STANDARD), using the following colour convention:

White on blue	Control card is present (active) in model
Black on Grey	Control card is not defined in the model (inactive)

To select a card (active or inactive) for display toggle it on/off using its row in this menu. Note: Selection makes the card active, whereas deselection just removes the card from the display panel. Thus deactivation of a card must be done explicitly.

You may also select:

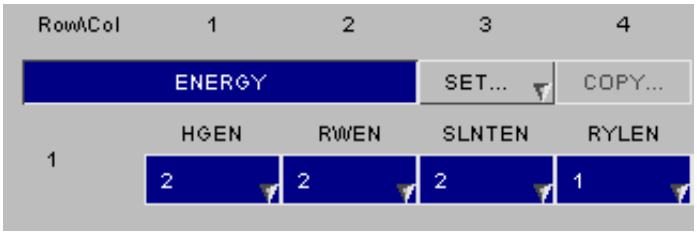
ALL All control cards, both active and inactive, are displayed

ACTIVE All active cards are displayed

DONE Return to card categories

Whenever a card is displayed and active its data fields can be updated by the normal text entry method. In addition all pre-defined lists of integers have popup menus giving the legal list of entries.

Making individual control cards active and inactive



Individual (visible) rows are toggled between active and inactive by clicking on their "name" button.

In this example:

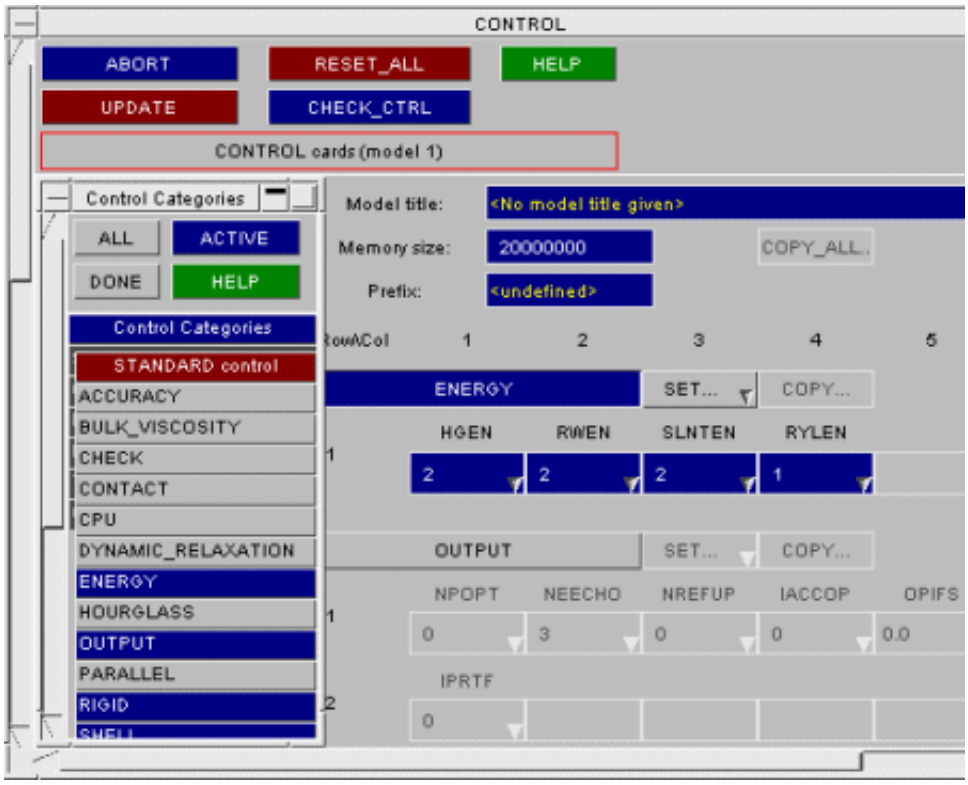
ENERGY Has been made active

OUTPUT Has been made inactive

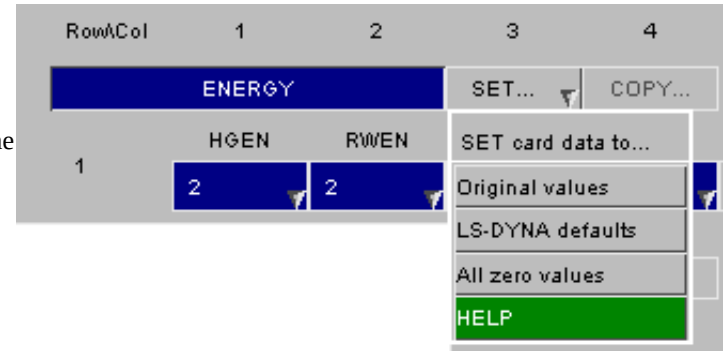
It can be seen that the (in)active status is also reflected in the selection menu on the left.

Note that inactive cards are "greyed out", and that entries cannot be made to them unless they are made active again.

Inactive cards will ***not*** be saved in the database following an **UPDATE** even if they contain (greyed out) values as here.



SET... (re)setting values for a control card.

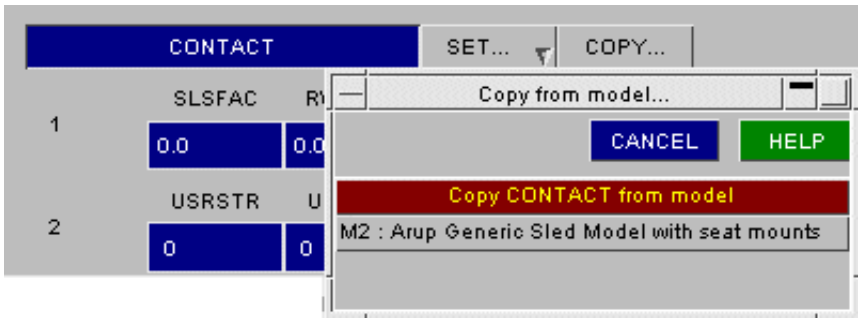


The data fields in a card may be (re)set back using the popup menu to:

Original values	The values in the current database, prior to any edits
LS-DYNA defaults	The default values quoted in the LS-DYNA keyword manual
All zero values	All fields are set to zero

Note that the local **SET...** option only affects this card. To reset ***all*** cards back to their original, unedited values use the **RESET_ALL** button at the top of the panel.

COPY... Copy data into a control card from another model

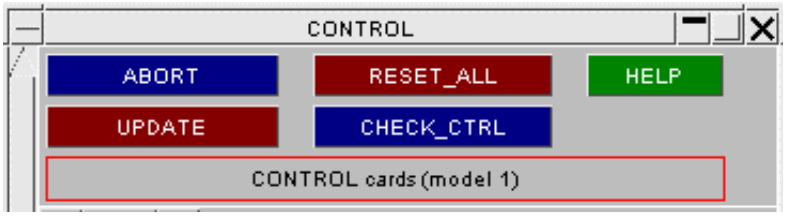


If one or more other models are present which also contain this control card then the **COPY...** button will be made live. This will give a list of possible models from which to copy this card's data. If no other models containing a definition for this card exist then the button will be greyed out.

Data copied in from another model supersedes the current data.

The **COPY...** option only affects this card. To copy data in from all cards in another model use the **COPY_ALL...** button at the top of the panel.

RESET_ALL: Restoring all control cards to their initial values.



RESET_ALL cancels the effect of all edit, copy, set and (in)activate operations by restoring all cards to their initial state as in the database.

UPDATE: Making control card edits permanent

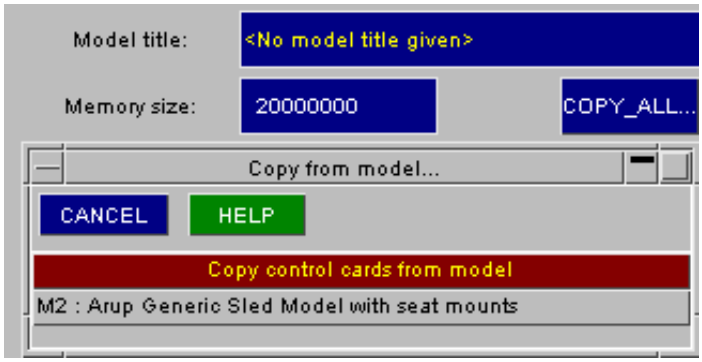
All changes above are carried out on a "scratch" definition. Changes are only saved permanently in the database when **UPDATE** is pressed.

To exit leaving the control cards unchanged use **ABORT**.

CHECK Running the standard checking function on control cards.

The **CHECK_CTRL** command runs the standard syntax and context checker. Most errors checking for control cards is based on detecting "out of range" parameters, but some interactions with data defined elsewhere in the model are also checked.

COPY_ALL: Copying all control cards from another model



COPY_ALL copies in all cards from another model, superseding any such definitions in this model.

Cards that are not active in the origin (copied from) model are not changed in the current model.

Setting model title and memory size.

In addition to editing the contents of the control cards:

- A **MODEL TITLE** of up to 80 characters can be specified within this panel.
- The **MEMORY SIZE** can also be set here. (This is optional, it is expressed in words of memory.)

CALC_DT2MS Calculates relationship of timestep to %age added mass

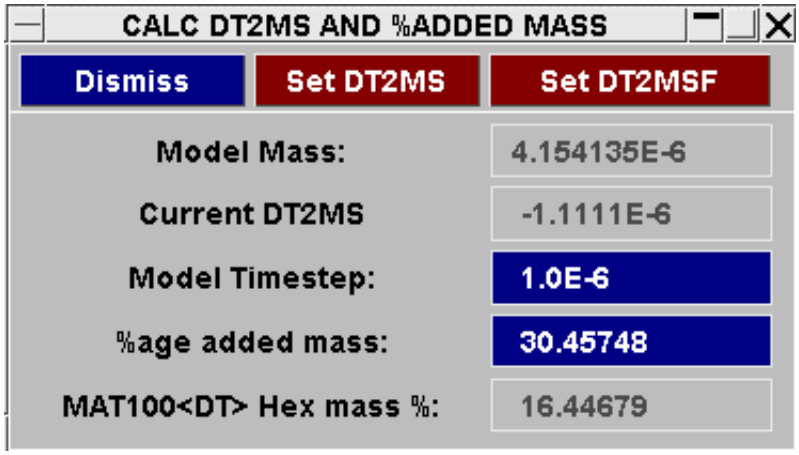
If DT2MS on the `*CONTROL_TIMESTEP` cards is set to less than zero, timestep added mass is active.

The **CALC DT2MS** function will report the %age added mass for the current timestep (expressed as DT2MS x TSSFAC).

A different value of timestep or %age added mass may be entered and the corresponding value will be calculated.

SET DT2MS and **SET DT2MSF**will update the values of DT2MS and DT2MSF on the timestep control card.

Note: TSSFAC is used for element timestep calculation and **should never exceed 0.9**. **CALC DT2MS** will never change TSSFAC.



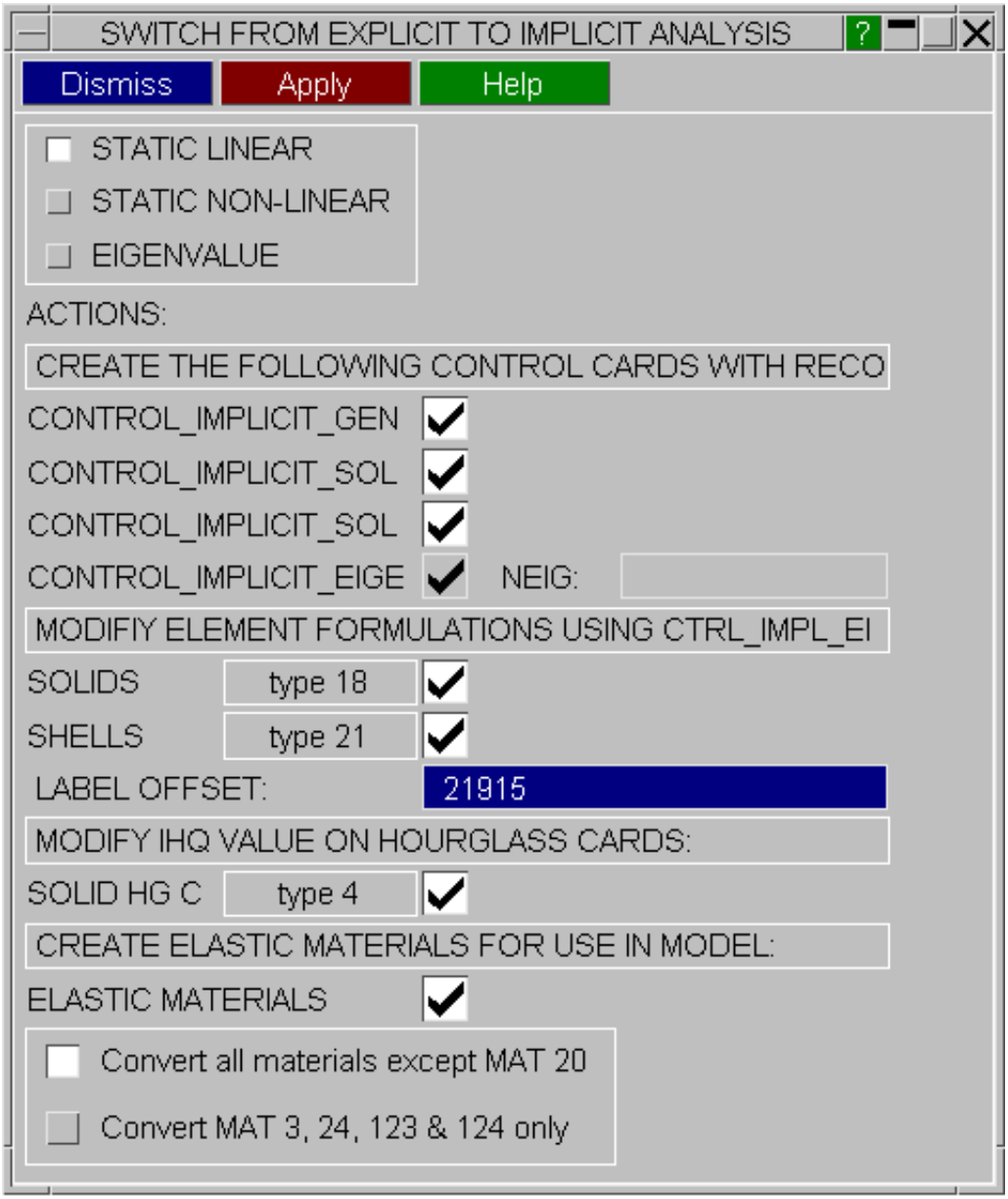
LS-Dyna has a special method of adding mass to spotweld elements using the setting **DT** on the MAT100 card.

For beam spotwelds this added mass is included in the normal total and will appear in the **%age added mass** box.

For solid spotwelds, LS-Dyna calculates the added mass differently and reports it separately. This added mass will appear in the **MAT100<DT> Hex mass %** box (it is also included in **%age added mass** given).

See note in Appendix 17 for more details.

CONVERT TO IMPICIT



The convert to implicit panel is a simple method of converting an explicit analysis to an implicit analysis by creating appropriate control cards (if not already present) and assigning default values. Also, element formulations are set to recomended defaults.

Original hourglass cards and materials cards are not modified if you select to change these. PRIMER will create duplicate cards with the relevant data copied from the original cards, and with the required changes for implicit analyses. The offset specified on this panel is applied to the original label to create the new cards. This offset must be higher than the highest material and hourglass label in the model.

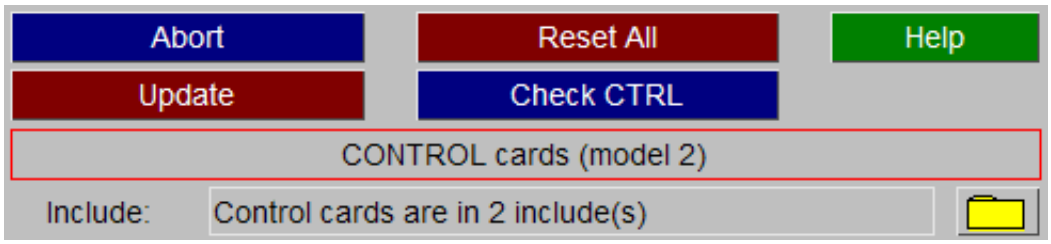
Notes on Control Cards and the model MERGE operation.

In common with other "static" (occurring either once or not at all) data in PRIMER, control cards may conflict when models are merged. The model merger allows you to select globally from source model #A or #B when cards exist in both models, but this may be too unselective for some cases.

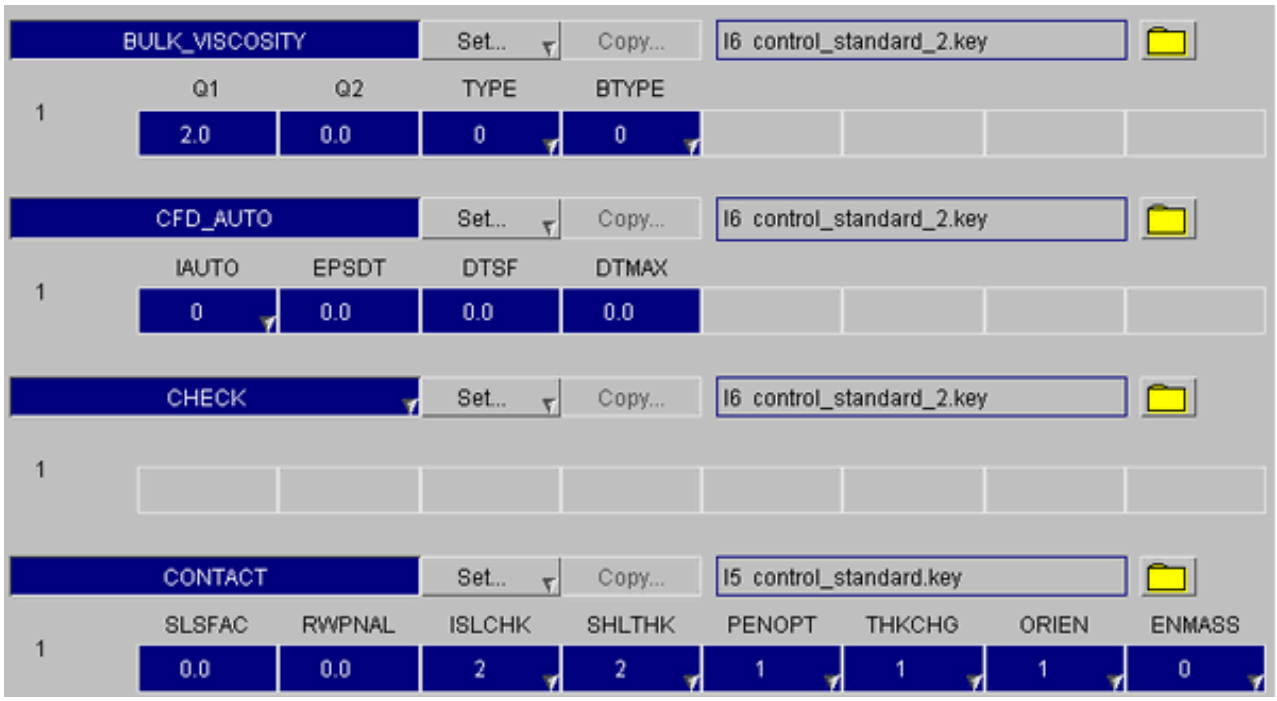
It is recommended that you review the control cards in the destination model generated by a merge operation, and make selective use of the **COPY** function above where required.

Notes on Control Cards and the include selection operation.

As in other editing panels, ***CONTROL** cards can be moved to a chosen include file using the include selection buttons at the top of the panel. All ***CONTROL** cards will be moved when this operation is carried out. When the ***CONTROL** cards are in more than one inlcude file, the include display within the editing panel will tell the user this is the case. Positioning the mouse over the include display area will print hover text to the screen listing all the include files the control cards are in.



The include file for each individual control card is displayed along side the control card information.



The include file can be modified by clicking on the folder button next to where the include name is displayed. Note the control card is only moved into the newly selected include file when **Update** is pressed on the top of the control card panel.

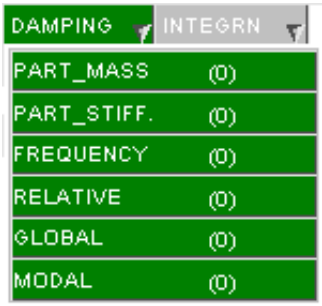
DAMPING: Defining Damping Cards.

- Main **DAMPING** menu
- Creation

LS-Dyna offers damping control via global damping, relative damping, frequency range damping, part damping by mass and part damping by stiffness. The CEAP version additionally has modal damping.

All six types may be created/edited in primer using the **DAMPING** function from the Keywords panel.

DAMPING MAIN MENU
This figure shows the main **DAMPING** menu.



The total number of damping cards for all models is reported.

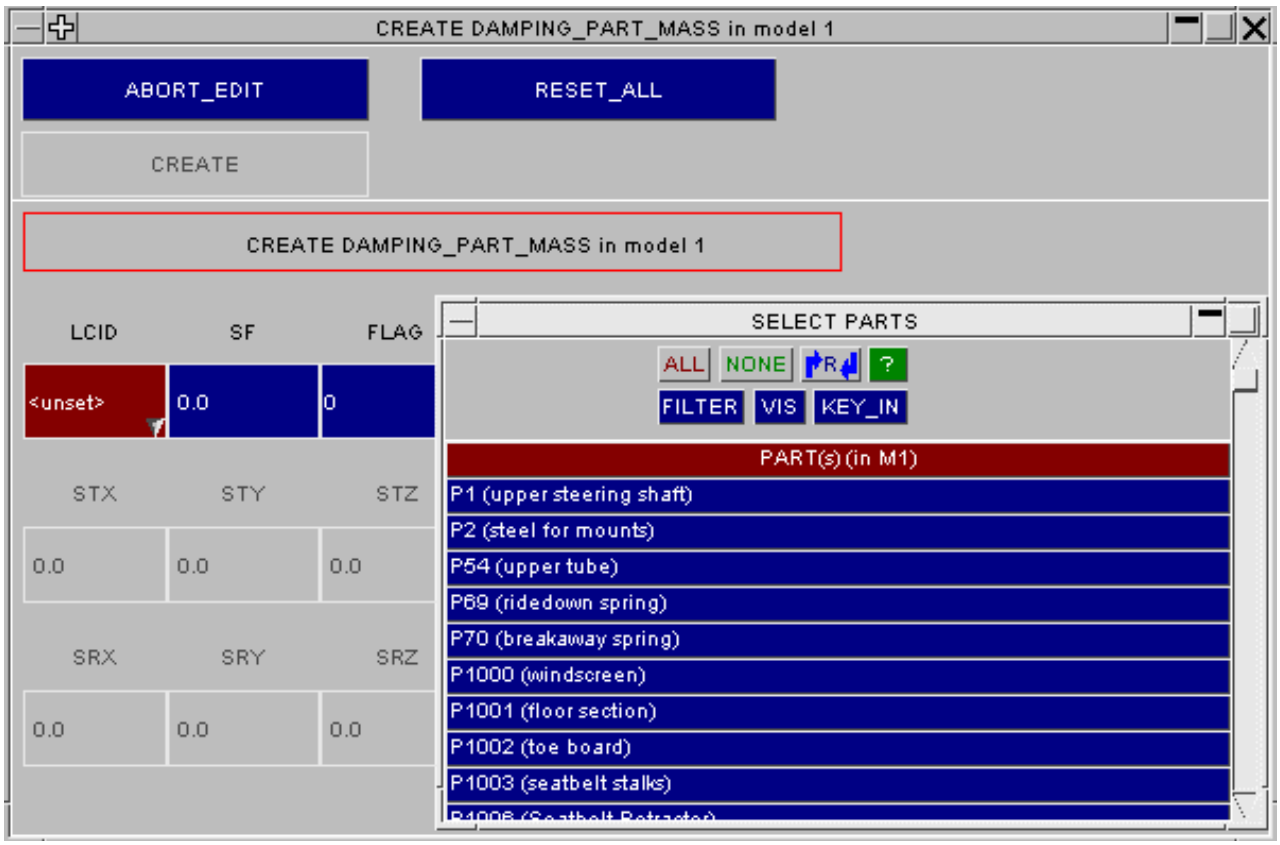
The panel will allow simultaneous editing of multiple damping cards. In the cases of Global and Modal damping there is no point in editing more than one card per model.

The static damping cards have dedicated editing panels. The others access the generic keyword editor.

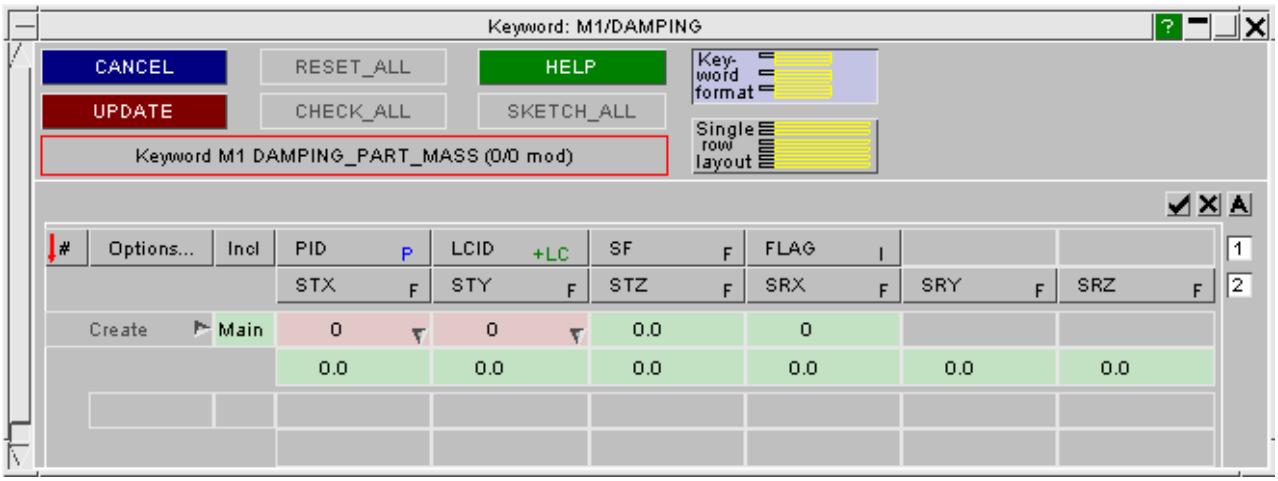
PART DAMPING

To create for a few parts, to delete and to edit, the keyword reader is perfectly sufficient. However, to enable the user to create rather larger numbers of damping cards, creation panels exist for damping part **MASS** and damping part **STIFFNESS** which allow parts to be selected from the object menu.

CREATING MULTIPLE DAMPING_PART_MASS



CREATING FROM KEYWORD READER (see section 5.0.3 for more details)



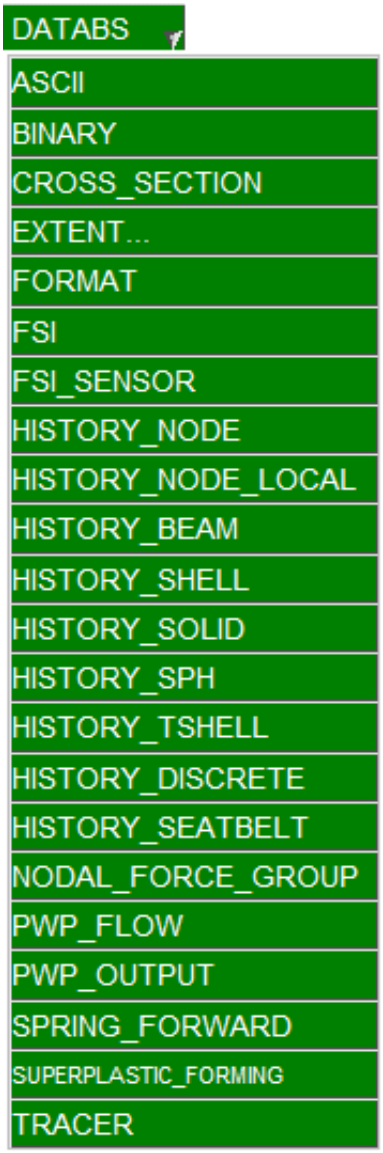
DATABASE: Defining Database Options.

- Selecting the *DATABASE sub-keyword
- Visualisation

The ***DATABASE** keyword in LS-DYNA is used to control what is written out to the BINARY and ASCII results files.
In addition to controlling what data is written out the *DATABASE cards also define how often the data is written out during the analysis and the format that is used to write the binary data files.

The main **DATABASE** pop-up menu allows any of the sub-categories to be selected. After a category has been selected a separate menu will be displayed allowing items to be created, modified and deleted.
The links below take you to the relevant sub-keyword sections:

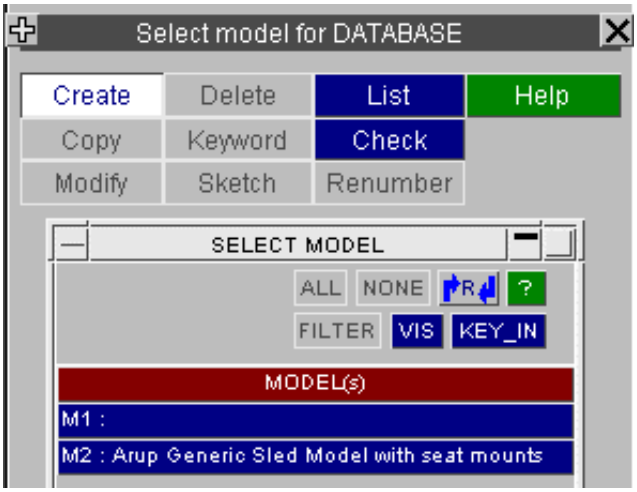
DATABASE_ASCII_xxx
DATABASE_BINARY_xxx
DATABASE_CROSS_SECTION_xxx
DATABASE_EXTENT_xxx
DATABASE_FORMAT
DATABASE_FSI
DATABASE_FSI_SENSOR
DATABASE_HISTORY_xxx **(this describes all sub-options).**
DATABASE_NODAL_FORCE_GROUP
DATABASE_SPRING_FORWARD
DATABASE_SUPERPLASTIC_FORMING
DATABASE_TRACER



If multiple models are open the main database panel will open and you will be asked to select a model. If only 1 model is open this stage will be automatically skipped and the selected sub-option window opened.

In addition the functions:

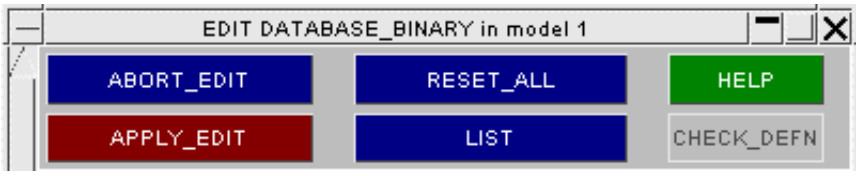
LIST	Generates a summary of all the ***DATABASE** cards in the model.
CHECK	Runs the standard check function on all ***DATABASE** definitions.



Common *DATABASE Options

Because some Keywords are "one-off" (e.g.**_BINARY**), some are "open-ended" (e.g.**_HISTORY_xxx**) and some reference normal "list" types with labels (eg**_CROSS_SECTION**) there is no standard editing panel for them.

However each of the sub-category menus contains a standard set of buttons:.



ABORT_EDIT	Returns to the main ***DATABASE** menu and discards any changes that have been made since the menu was invoked.
APPLY_EDIT	Returns to the main ***DATABASE** menu and keeps any modifications that have been made or any items that have been created. If in some of the sub-menus items have been marked for deletion this option will remove them from the model.
RESET_ALL	Resets all items back to the original values/settings they had when the menu was invoked. Any new items that have been created since the menu was invoked will be deleted.
LIST	Displays a list on the screen giving details about the status and values defined for all the items in the sub-category.
HELP	Displays a help screen giving information on what the sub-category is used for and what the different input parameters for each sub-category mean.

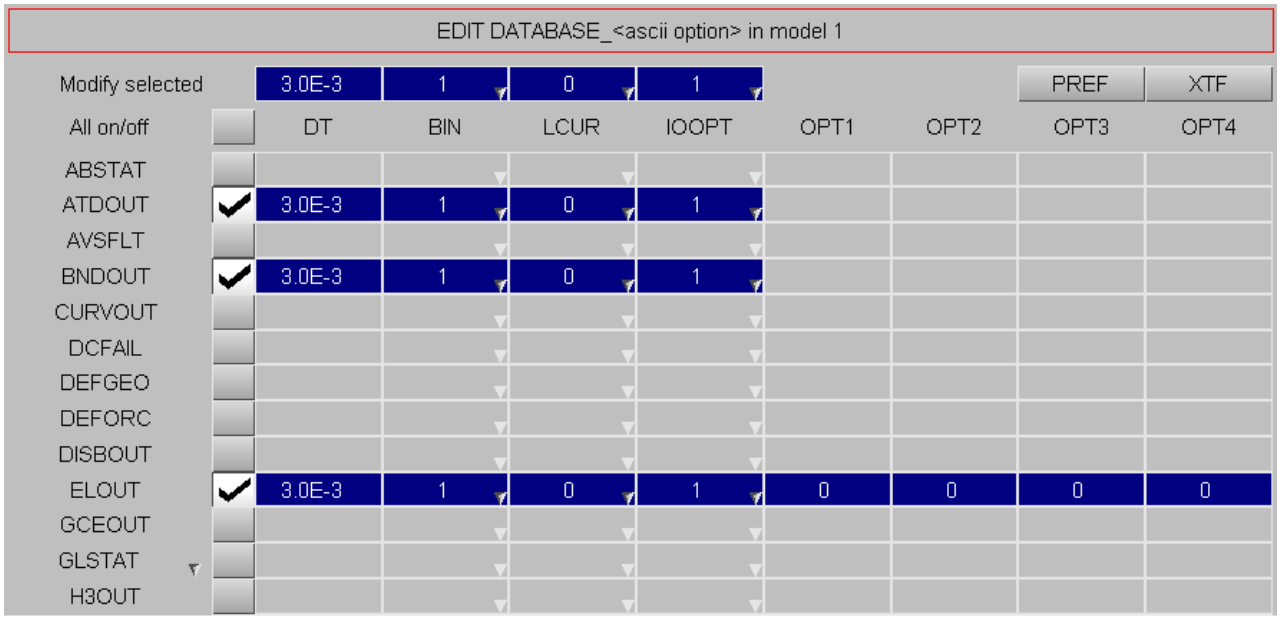
In addition to the standard set of buttons a number of the ***DATABASE** sub menus also contain the following two options.

ACTIVE / INACTIVE	Switch which can be used to include/exclude the DATABASE card in the model when it is written out. An INACTIVE card will have all parameters greyed out, whilst the ACTIVE card will allow parameters to be modified.
SET DEFAULTS	Reset all the variables within the card to the recommended LS-DYNA default values.

DATABASE_ASCII_xxx Control of "ASCII" database file output.

The **DATABASE_ASCII** menu displays a list of all the additional ASCII database files that may be created during an LS-DYNA analysis.

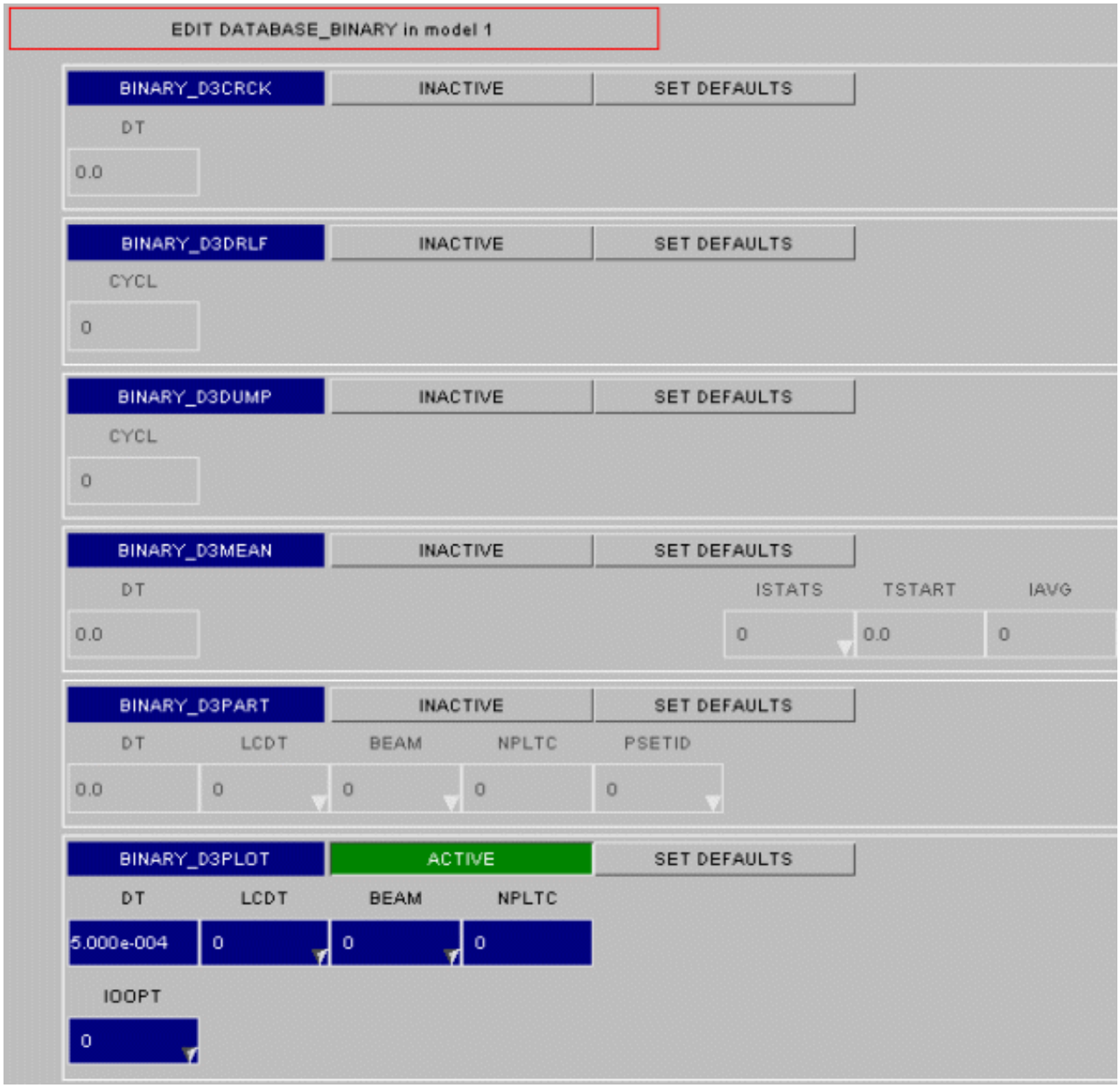
These files may now also be written in the more compact binary format.



Modify selected	Values entered in these text boxes will be automatically copied to all enabled ASCII cards. ASCII cards that are subsequently switched on/enabled will take these values by default. **DT** defines the output frequency, **BIN** defines the output mode (1=ASCII, 2=BINARY, 3=BOTH), **LCUR** defines the output frequency via a load curve ID and **IOOPT** defines the behaviour of the output frequency load curve.
All on/off	Toggles all cards on and off. Default values are taken from the **Modify selected** text boxes
ACTIVE / INACTIVE checkboxes	Switch which can be used to include/exclude the **`*DATABASE`** card in the model when it is written out. An **UNCHECKED** card will have all parameters greyed out, whilst the **CHECKED** card will allow parameters to be modified
PREF/XTF	**PREF** makes those cards active which have been set in the oa_pref file. **XTF** activates those cards that contain information written to (soon to be obsolete) xtf file.

DATABASE_BINARY_xxx Control of binary database file output

The **DATABASE_BINARY_** menu displays a list of all the binary result and restart files that may be created during an LS-DYNA analysis



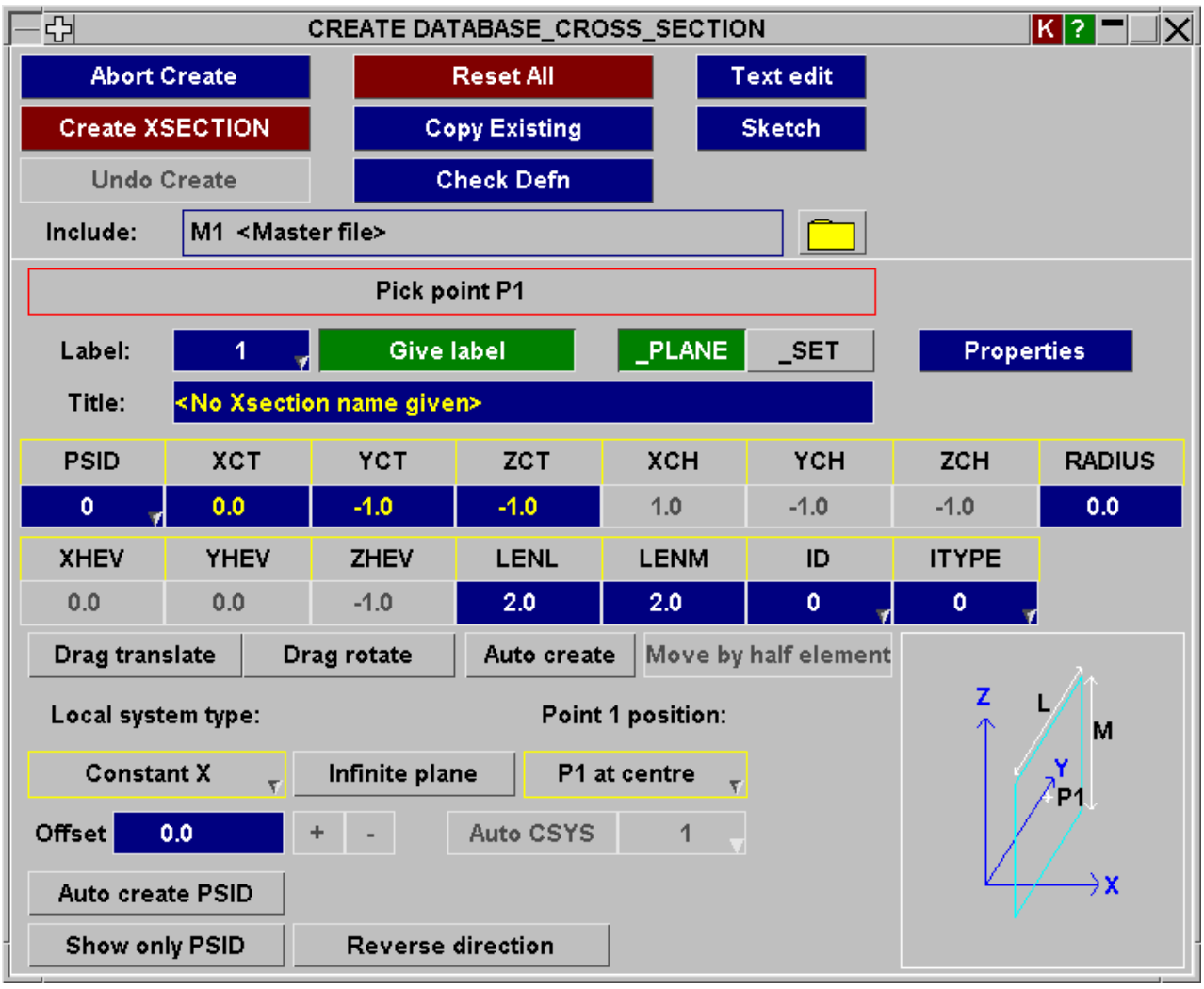
Output frequencies may be set for each database file, (note that some use time and some #cycles), and each may be made (in-)active. Some cards contain other fields which can also be set.

DATABASE_CROSS_SECTION_xxx Editing cross-sections for force extraction.

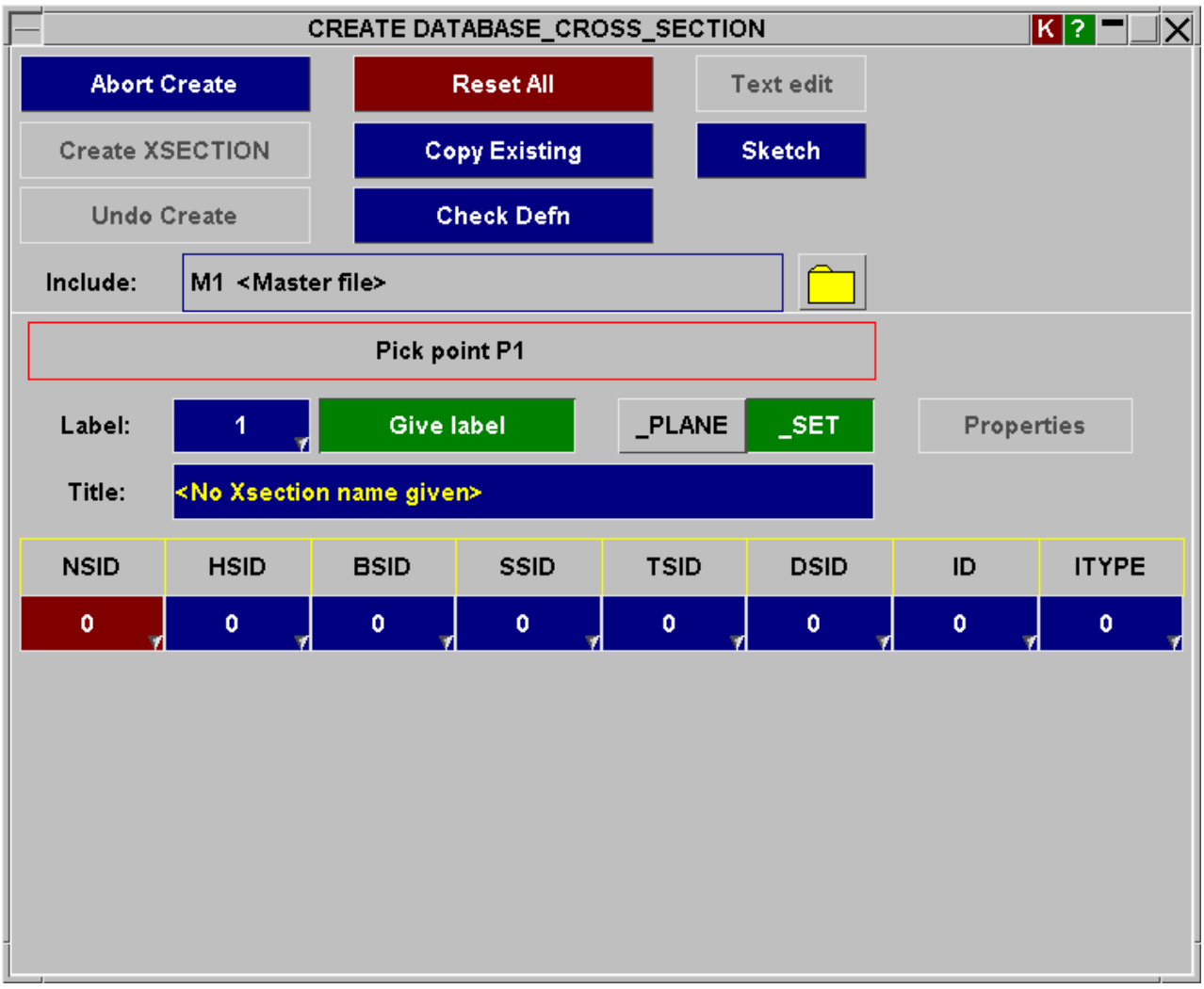
The ***DATABASE_CROSS_SECTION_PLANE** option defines a cross section for force output by defining a plane and then summing up the forces in the elements that the plane cuts through. (During the analysis the force is only summed for those elements that were cut by the plane at the start of the analysis).

The ***DATABASE_CROSS_SECTION_SET** option defines a cross section for force output by defining sets of nodes and elements for which the force should be summed.

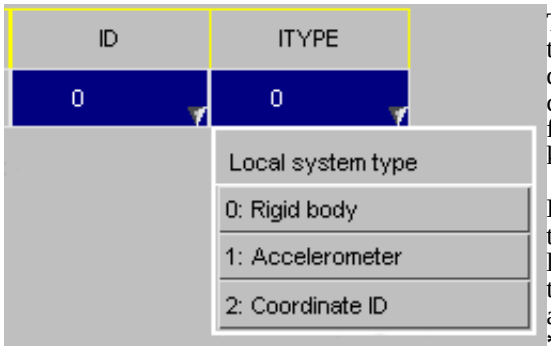
The figure below shows the **_PLANE** option.



The figure below shows the **_SET** option.



The **Give label** button allows you to explicitly define a label for the ***DATABASE_CROSS_SECTION** card being created.

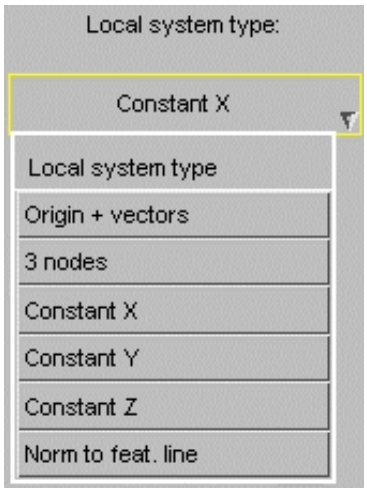


The <**itype**> data field which is common to both the **_PLANE** and the **_SET** options; controls whether the local system type <**id**> (if defined) is a rigid body or an accelerometer or an explicitly defined coordinate system. This is necessary because the type of the <**id**> field has to change, and therefore cannot remain "generic" within the keyword editor.

During the analysis, the user may want the extracted forces across the defined cross-section plane to be computed in the section plane's local coordinate system. This can be achieved quickly by means of the **Auto CSYS** button (not relevant in the _SET case) which is available when parameter <**itype**> is set to **2: Coordinate ID**. If a ***DATABASE_CROSS_SECTION_PLANE** card is created with this option switched on, a new ***DEFINE_COORDINATE_SYSTEM** card is automatically created using the **L-M-N** axes of the cross-section plane as its basis vectors. Parameter <**id**> of the created ***DATABASE_CROSS_SECTION_PLANE** card is then automatically set to be the label of this newly created coordinate system.
The user can define the label of the coordinate system to be created by entering it in the text box next to the **Auto CSYS** button.

Graphical definition of single **_PLANE** geometry. (Not relevant in the **_SET** case)

Primer allows you to quickly and easily create ***DATABASE_CROSS_SECTION_PLANE** cards in a number of ways. A particular method of plane creation can be chosen by invoking the **Local system type** pop-up shown below:

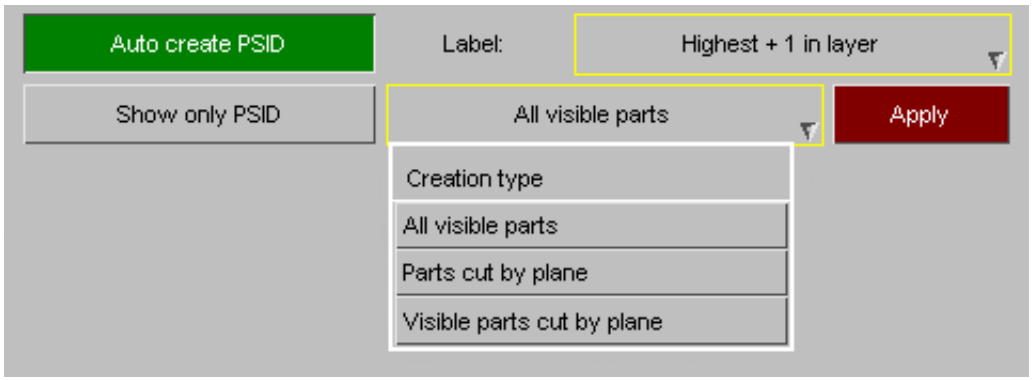


- **Origin + vectors**: Pick three nodes from the graphics window. They will define the origin, the unit normal vector **N** and the edge vector **L**, respectively, for the plane to be constructed.
- **3 nodes**: Pick three nodes from the graphics window. The first two nodes will define the origin and the edge vector **L**, respectively, for the plane to be constructed. The third lies on the **L-M** surface of the plane to be constructed, and hence defines it.
- **Constant X**: Pick one node from the graphics window. A cross-section plane passing through this node and normal to the global X-axis is computed immediately. The parameters <**lenl**> and <**lenm**> are computed automatically based on the dimensions of the model.
- **Constant Y**: Same as **Constant X**, except that the cross-section plane passes through the picked node, and is normal to the global Y-axis.
- **Constant Z**: Same as **Constant X**, except that the cross-section plane passes through the picked node, and is normal to the global Z-axis.
- **Norm. to feat. line**: Pick one node lying on a feature line or a free edge from the graphics window. A cross-section plane passing through this node and normal to the feature line is computed immediately.

The **Auto create** button (not relevant in the _SET case) automatically creates a new ***DATABASE_CROSS_SECTION_PLANE** card for every cross-section plane that is computed without the user having to explicitly click the **Create XSECTION** button. A cross section plane that is computed, but not created, is sketched in a suitable colour (depending on the background colour of your Primer's graphics window), while a cross-section plane that has been created and added to the model as a ***DATABASE_CROSS_SECTION_PLANE** card is always sketched in green.

The **Drag translate** and **Drag rotate** buttons (not relevant in the _SET case) allow the user to drag or rotate a computed cross-section plane in the graphics window about its local axes.

If a cross-section plane cuts along a mesh boundary, the results may be unreliable, as the forces from elements on both sides of the plane may be included. The **Move by half element** button (not relevant in the _SET case) attempts to remedy this problem by re-positioning the cross-section plane halfway along the edge of the element that lies on the positive side of the computed cross-section plane. If the feature line does not extend in the positive direction of the computed cross-section plane, the plane is automatically re-positioned halfway along the edge of the element that lies on the negative side of the cross-section plane.
This button is activated only when **Local system type** is set to **Norm. To feat. Line**

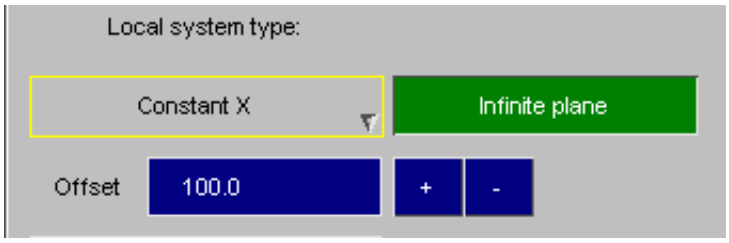


The **Auto create PSID** and related buttons (not relevant in the _SET case) shown on the left can be used to automatically create a new part set in the model and associate it with the ***DATABASE_CROSS_SECTION_PLANE** card being created (by automatically setting parameter <**psid**> on the ***DATABASE_CROSS_SECTION_PLANE** card to be the label of the newly created part set).

Once the part set is created by selecting an appropriate method from the pop-up box and clicking the **Apply** button, the parts contained in the set just created can be quickly visualized by clicking the **Show only PSID** button.

The normal direction of the ***DATABASE_CROSS_SECTION_PLANE** can easily be changed by clicking on the **Reverse Direction** button.

Graphical definition of multiple offset _**PLANE** geometry. (Not relevant in the _**SET** case)



Once a cross-section plane is computed using one of the methods described above, it can be used as a reference plane to quickly create another plane **OFFSET** units away from it. The offset plane is computed along the unit normal vector of the reference plane using the method selected in **Local system type**.

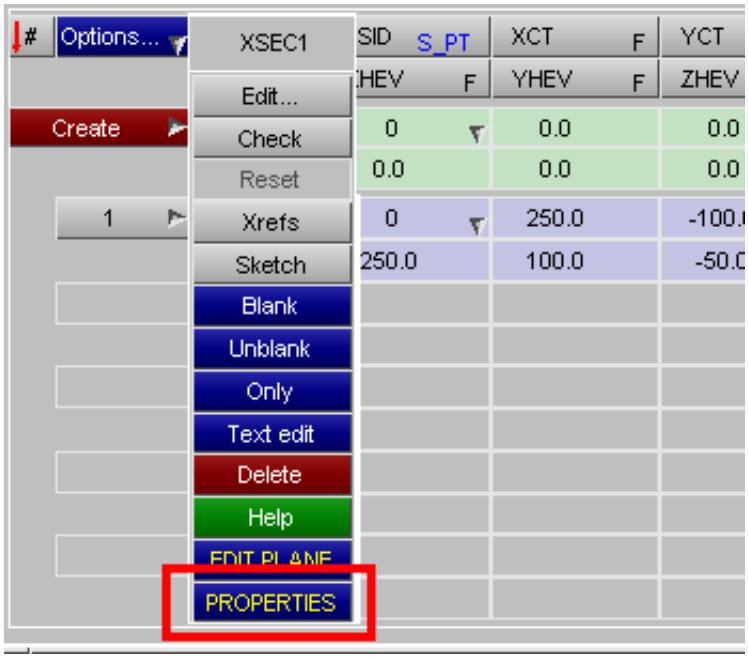
The buttons shown here are activated as soon as a reference plane is computed and a valid offset is specified in the **OFFSET** text box. Clicking the **+** button computes an offset plane along the positive direction of the reference plane's unit normal vector, while clicking the **-** button computes an offset plane along the negative direction of the reference plane's unit normal vector. The computed offset plane is then sketched using a suitable colour. The parameters of the computed plane are used to create the ***DATABASE_CROSS_SECTION_PLANE** card in the model by clicking the **Create XSECTION** button.

The **+** or **-** buttons can be clicked repeatedly to compute multiple planes each at a distance of **OFFSET** units away from the previously computed one.
If option **Auto create** is switched on, a new ***DATABASE_CROSS_SECTION_PLANE** card is automatically added to the model every time a new offset plane is computed without the user having to explicitly click the **Create XSECTION** button.

Properties: Computing the analytical properties of the cross-section.

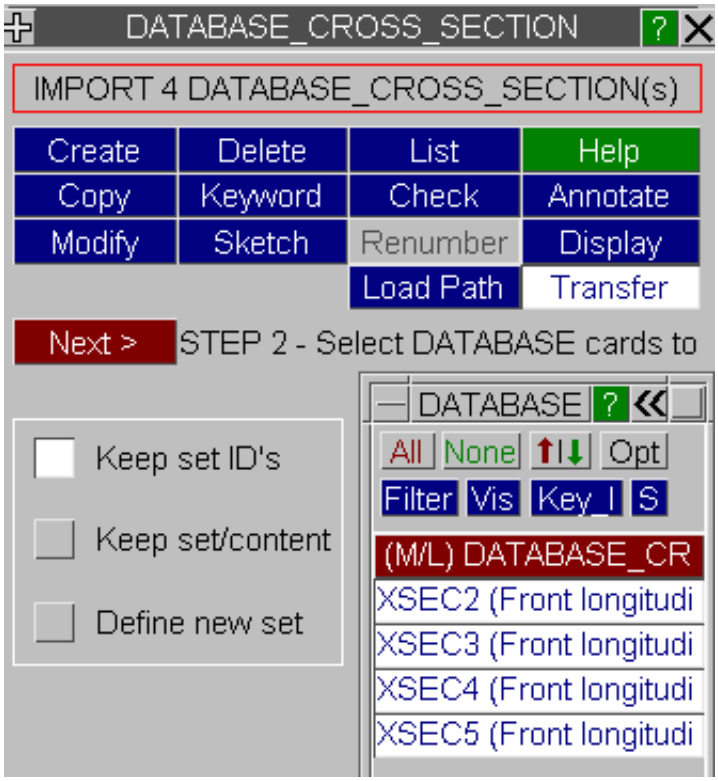
In the **_PLANE** case only, where a flat geometrical section is defined, the **Properties** button launches the cut-section properties panel described in section 6.13using the geometry of this section. This calculates and displays its elastic and plastic properties.

The same **Properties** display may also be launched from the Database Cross Section keyword editor by using tghe popup on a given section's row button.



`*DATABASE_CROSS_SECTION` specific annotate/display/transfer/reverse tools.

There are specific tools for DATABASE_CROSS_SECTION's called **Annotate, Display, Transfer** and **Reverse**. **Annotate** is available through the main DATABASE_CROSS_SECTION panel or an individual DATABASE_CROSS_SECTION edit panel. This tool annotates the cross sections with the label associated with the cross section, and if you choose to annotate a number of cross sections, they are all sketched in different colours. **Display** is available through the main DATABASE_CROSS_SECTION panel. **Reverse** is available through the main DATABASE_CROSS_SECTION panel or an individual DATABASE_CROSS_SECTION edit panel. Reverse is used to reverse the normal direction of selected DATABASE_CROSS_SECTION's. **Display** will show an individual cross section, and only the parts referenced by the cross section. **Transfer** is available through the main DATABASE_CROSS_SECTION panel. This is used for transfering DATABASE_CROSS_SECTION definitions from one model to another.



There are a few options available with this tool:

Keep set ID's - Database cards only are copied, i.e. any references to part sets are copied but the part set and contents are not copied.

Keep set/contents - Database cards are copied along with any referenced part sets. The parts in the part set are not copied, but the references to the parts are.

Define new set - Database cards are copied, and for all database cards copied the PSID field is set to a specified part set on the target model.

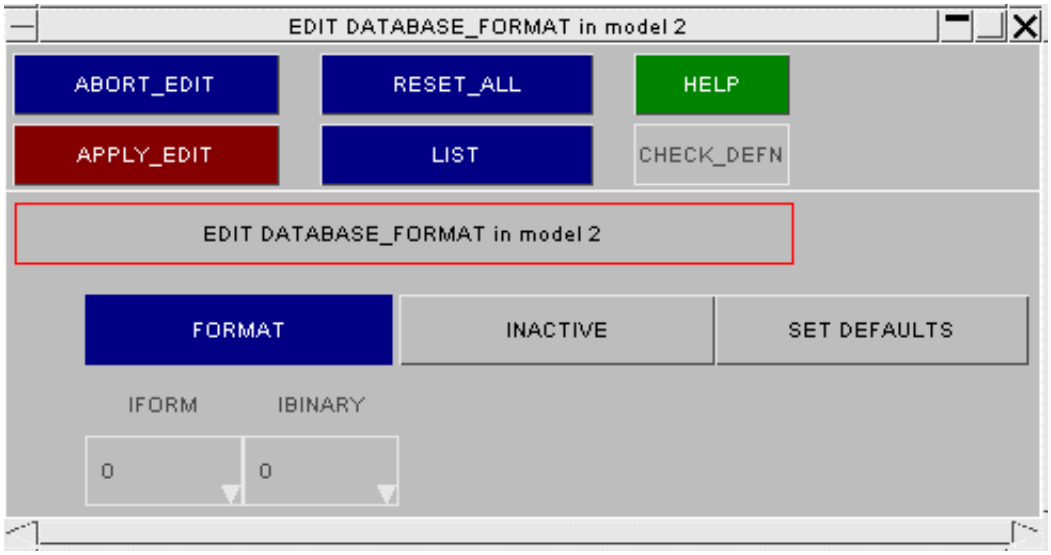
DATABASE_EXTENT_xxx

The ***DATABASE_EXTENT** menu has 2 sub options

***DATABASE_EXTENT_BINARY**	Controls a number of optional data components that can be written out to the binary results files.
***DATABASE_EXTENT_SSSTAT**	Defines a number of PART sets (see *SET) that are used to generate model sub system energies and contact forces.

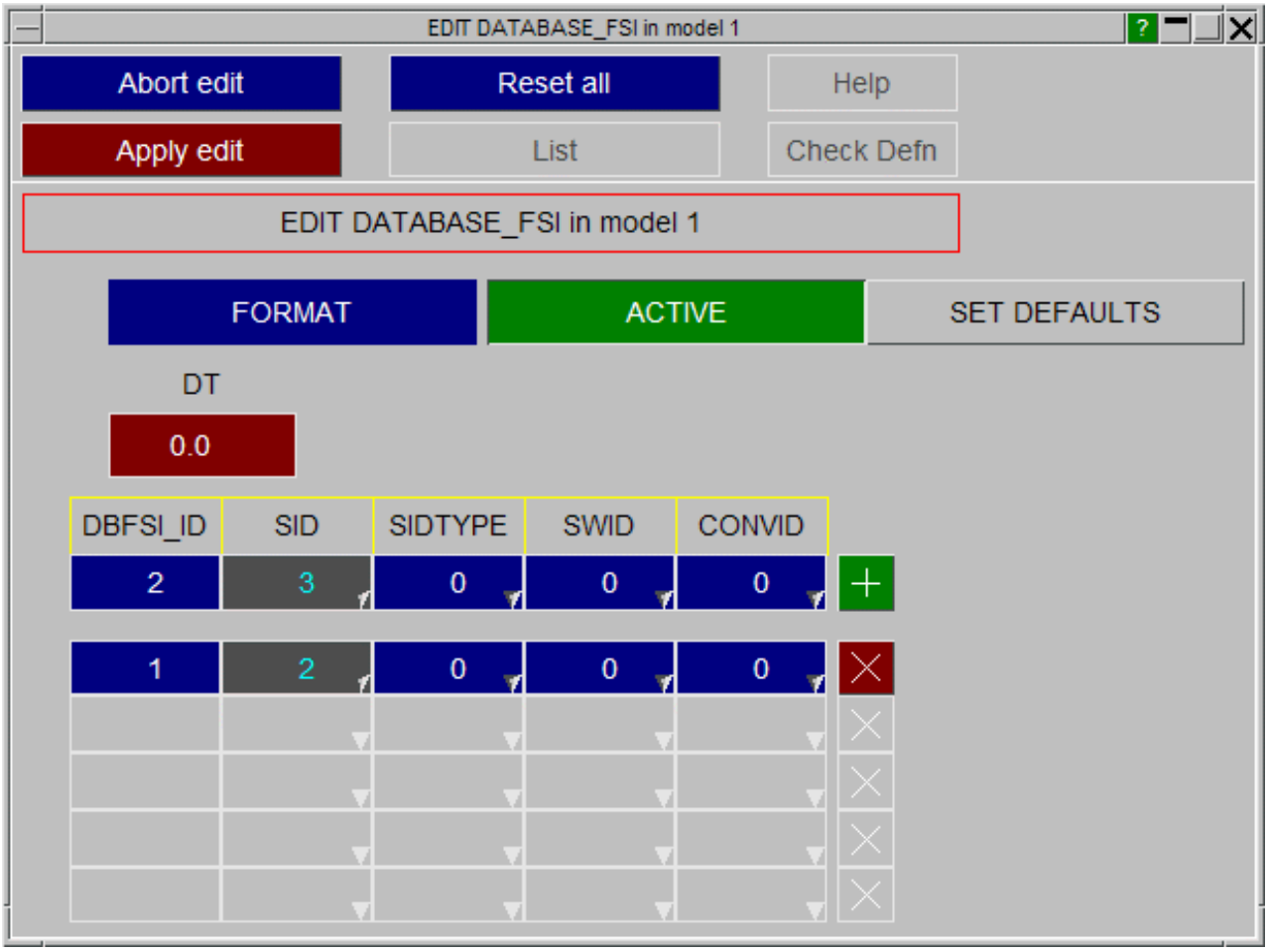
DATABASE_FORMAT Controlling binary database output format.

The `*DATABASE_FORMAT` menu controls the format the binary results files are written in.



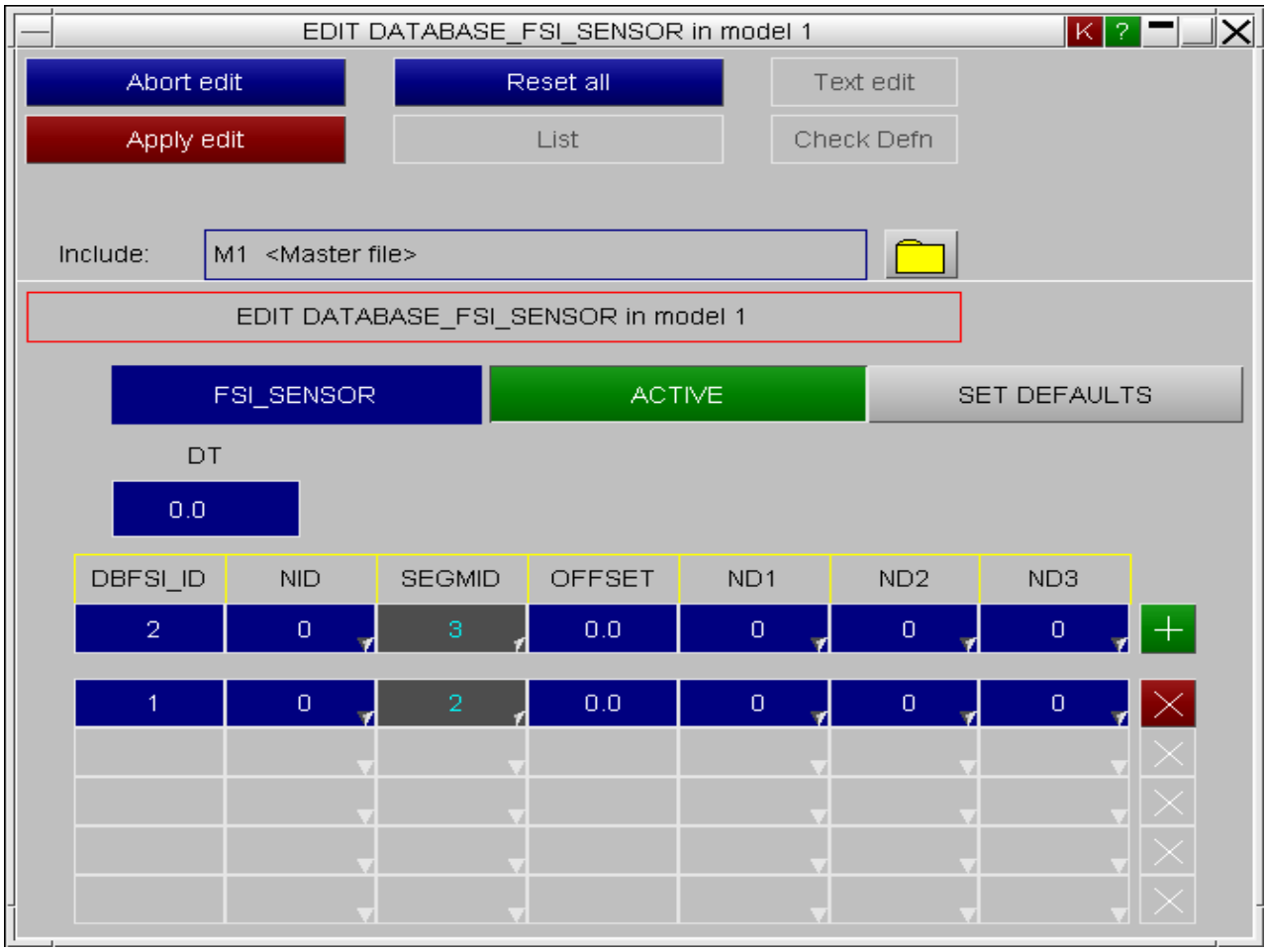
DATABASE_FSI Output information about certain Lagrangian surfaces.

The `*DATABASE_FSI` menu controls the output information about certain Lagrangian surfaces.



DATABASE_FSI_SENSOR Output of an ASCII file called "dbsensor".

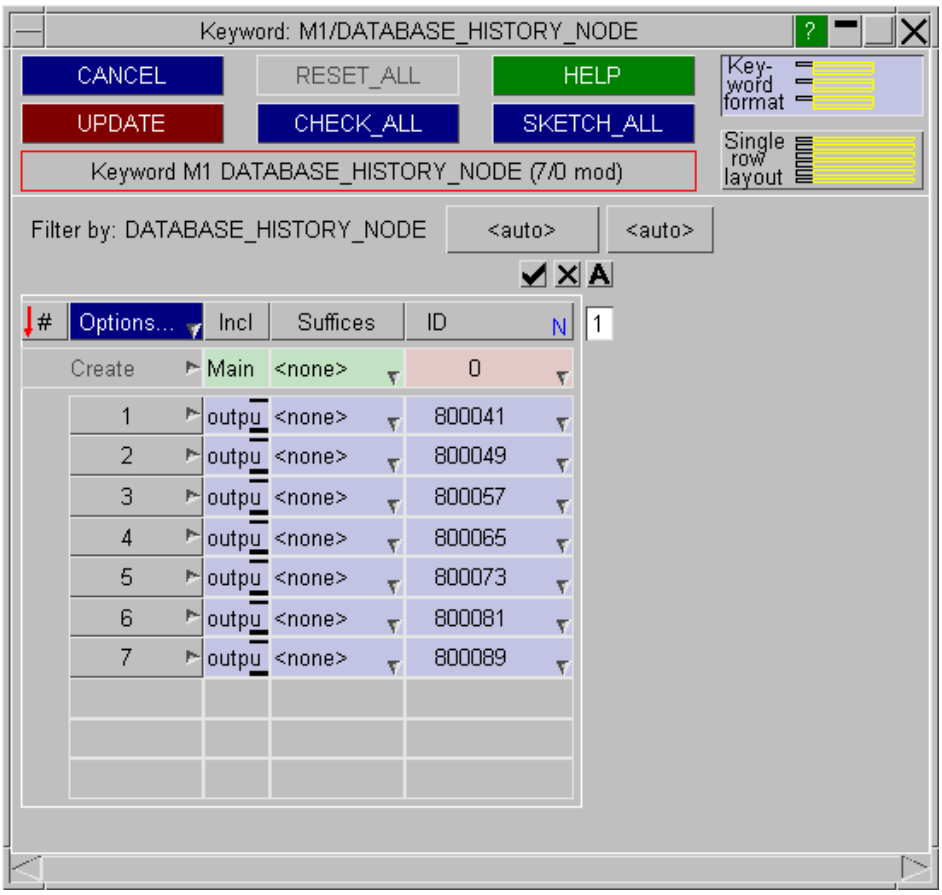
The `*DATABASE_FSI_SENSOR` menu controls the output of an ASCII file called "dbsensor". The input defines the pressure sensors' locations which follow the positions of some Lagrangian segments during the simulation.



DATABASE_HISTORY_xxx Selecting nodes and elements for "time history block" output

***DATABASE_HISTORY** contains a number of sub-options to allow nodes, solids, beams, shells, SPH elements, thick shells, discretes and seatbelt elements to be defined for output to the time history (.thf) files.

For nodes only there is an extra suffix _LOCAL that permits a coordinate system to be associated with each node/set which controls the frame of reference for the output for that item. While this used to be on a separate keyword editor panel in Primer 14, all ***DATABASE_HISTORY** definitions have now got keyword editors from Primer 15 onwards, and _NODE and _NODE_LOCAL are now on the same editor. These editors can be opened from the main ***DATABASE** pop-up.



Individual items (nodes, elements or sets depending on the option) can be added using the entry row of the keyword editor and the **Create** button. Items can be removed by right-clicking on the grey row number button on the keyword editor.

It is also possible to select multiple items at the same time by using the **MULTIPLE** option from the popup at ID and selecting them from the object menu. When clicking this, also screen picking will be active.

To remove multiple items from the keyword editor, you can right-click on **Options...** and then use **Select...** to open an object menu. Here also screen picking is a convenient option of selecting the items. To delete the selection, you should right-click on any selected row (highlighted in dark-blue) on the keyword editor and use the **Delete** button.

For every ***DATABASE_HISTORY_<type>** you can define items by:

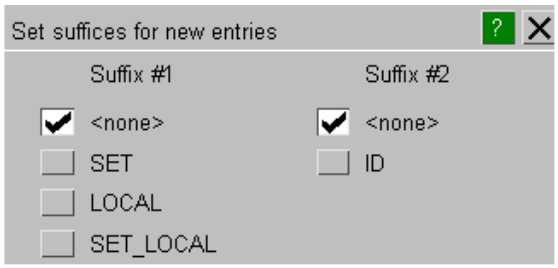
Explicit **<item type>**	Here **_NODE**. A list of explicit items is defined, in any order.
SET_<item type>	Here **SET_NODE**. A list of node sets is defined.

_ID OPTION: By activating the _ID option, one may create database history items with a title.

NOTE: The SPH option in this context uses the following:

HISTORY_SPH	The "element" number of the SPH elements (which is the same as the node ids).
HISTORY_SPH_SET	The sets are in fact node sets (*SET_NODE) since there is no *SET_SPH keyword in LS-DYNA.

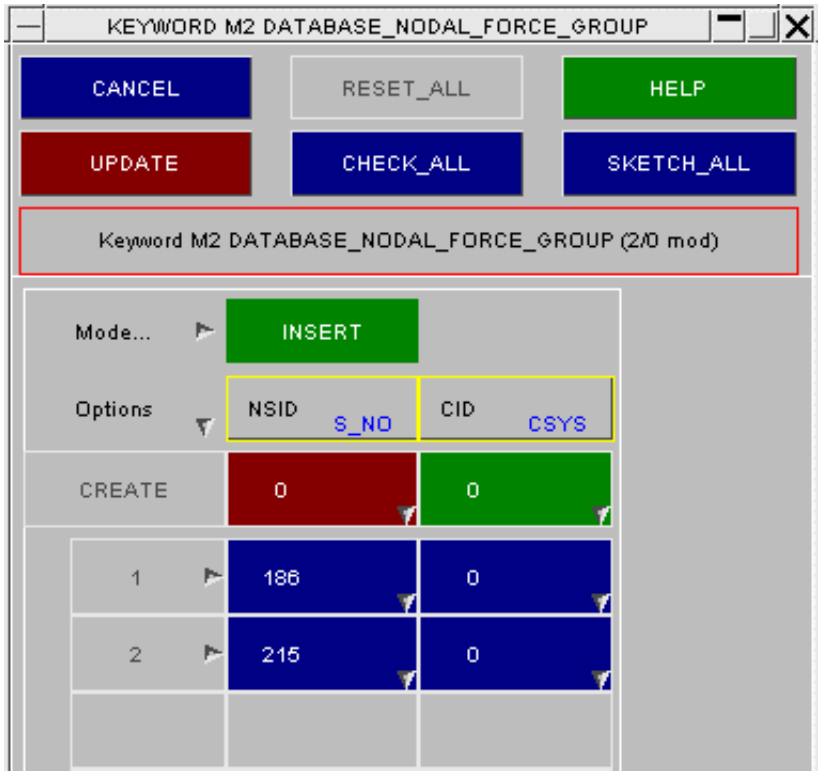
The _LOCAL suffix for ***DATABASE_HISTORY_NODE(_SET)** can also be selected from the generic keyword editor:



Nodes should not appear more than once in a time-history block, however defined. The **CHECK_ALL** function will detect duplicate definitions and, when invoked from model checking, autofix them by removing them if requested.

DATABASE_NODAL_FORCE_GROUP

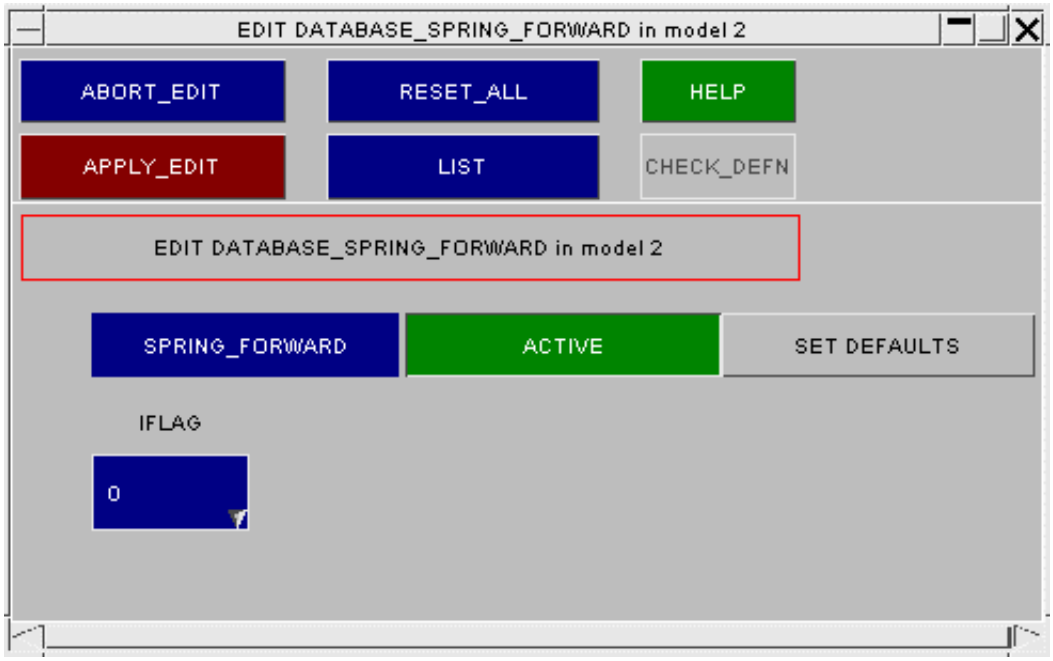
Definition of **SET_NODEs** that write summary force output.



DATABASE_SPRING_FORWARD

Output for implicit "spring forward" calculation after a metal-forming analysis

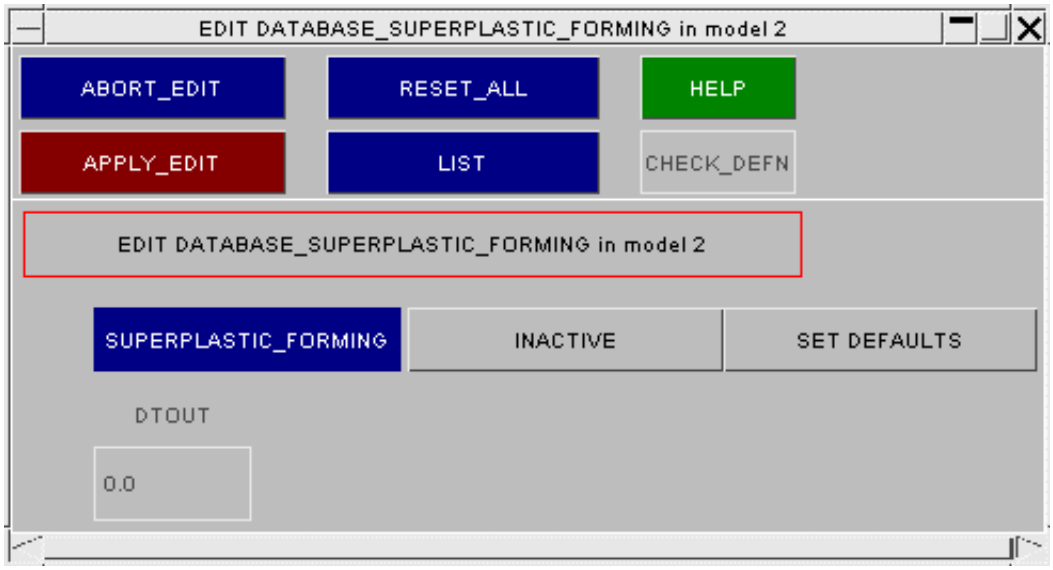
The ***DATABASE_SPRING_FORWARD** menu allows the nodal force components at the end of a metal stamping simulation to be output to an ASCII file for input to a subsequent spring forward calculation.



DATABASE_SUPERPLASTIC_FORMING

Output frequency to Super-plastic forming database files.

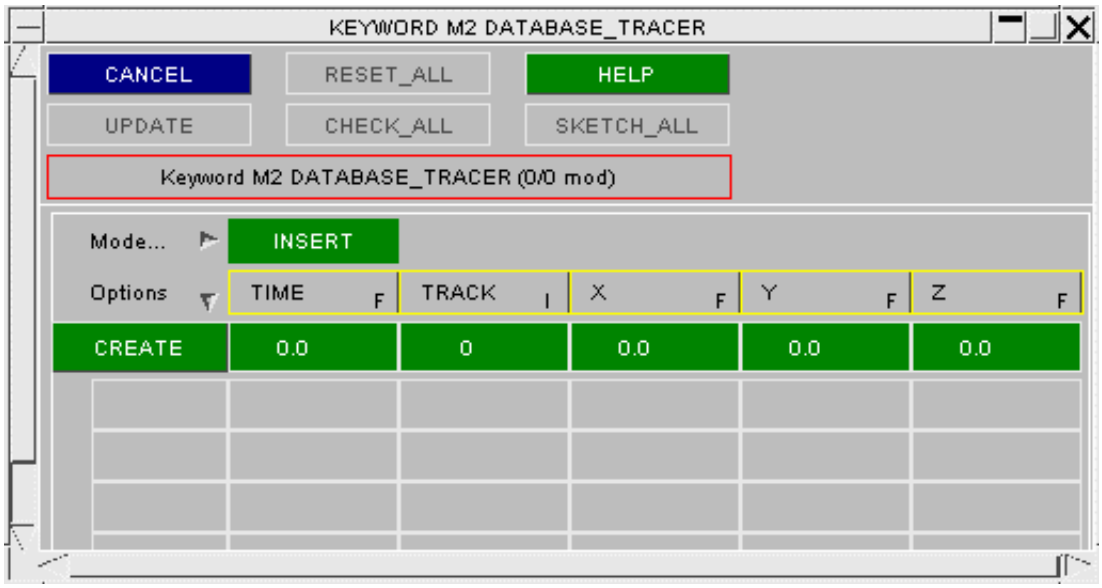
The ***DATABASE_SUPERPLASTIC_FORMING** menu allows the output frequency to be specified for output to the Superplastic Forming files.



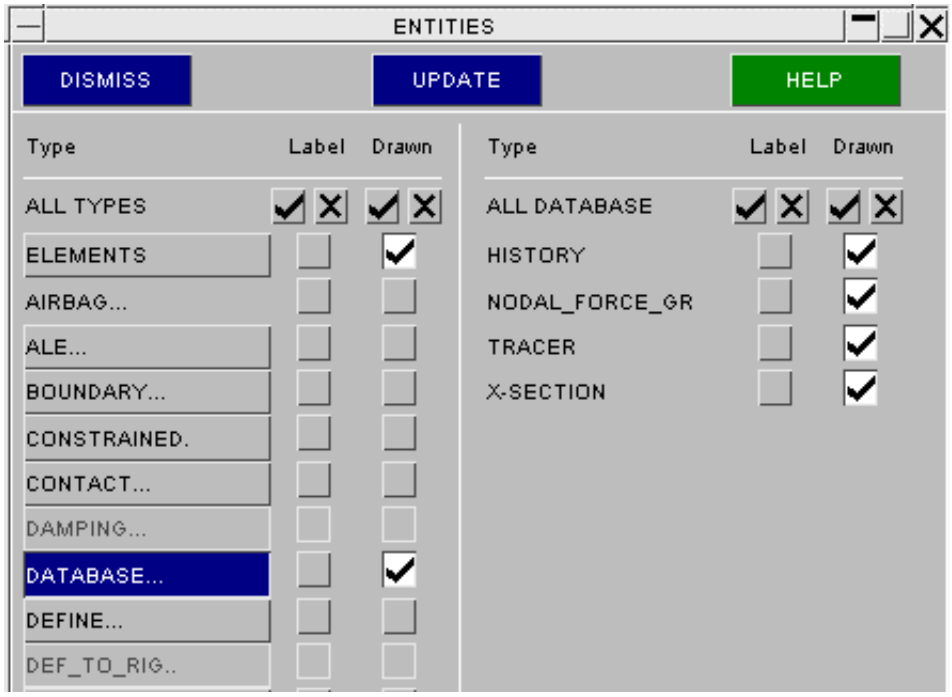
DATABASE_TRACER

Specification of "tracer" particles for Eulerian analyses

The ***DATABASE_TRACER** menu allows a series of initial points to be defined for which the position, velocity and stress components can be written out to an ASCII file.



Visualisation of *DATABASE items.



Those *`DATABASE_<type>` options which are sensibly drawable may be displayed via the **ENT**ity Viewing panel.
The **SKETCH** buttons in the relevant editing panels will also draw them.

DEFINE: Defining Define Options.

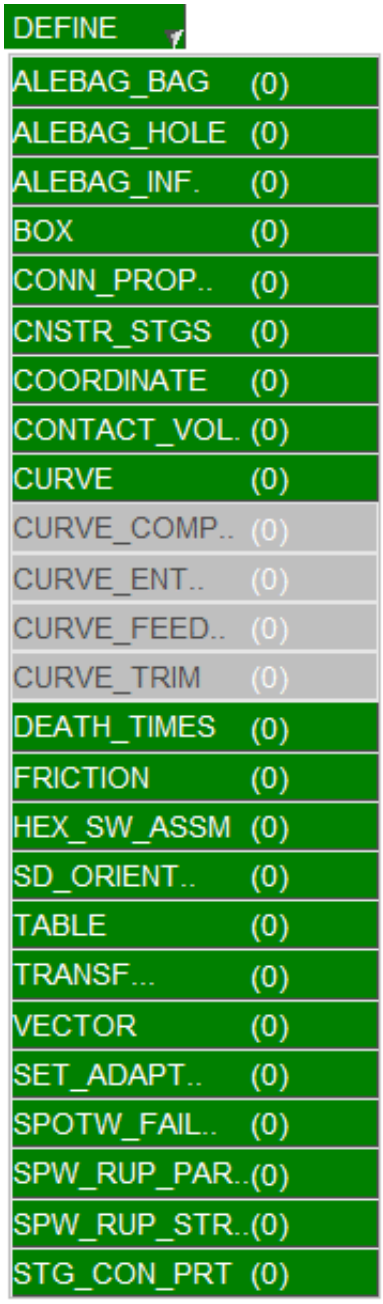
The ***DEFINE** keyword in LS-DYNA is used to define various things including boxes, load curves, tables, coordinate systems and orientation vectors.

The main **DEFINE** pop-up menu allows any of the sub-categories to be selected. After a category has been selected a separate menu will be displayed allowing items to be created, modified and deleted.
Some **DEFINE** keywords are edited in Primer using the generic Keyword editor. These are:

- **DEFINE_ALEBAG_BAG**
- **DEFINE_ALEBAG_HOLE**
- **DEFINE_CONSTRUCTION_STAGES**
- **DEFINE_SET_ADAPTIVE**
- **DEFINE_SPOTWELD_RUPTURE_PARAMETER**
- **DEFINE_STAGED_CONSTRUCTION_PART**

The other **DEFINE** keywords are edited in Primer using specific editing panels. The links below take you to the relevant sub-keyword sections:

- **DEFINE_ALEBAG_INFLATOR**
- **DEFINE_BOX**
- **DEFINE_CONNECTION_PROPERTIES**
- **DEFINE_CONTACT_VOLUME**
- **DEFINE_COORDINATE_xxx**
- **DEFINE_CURVE_xxx**
- **DEFINE_DEATH_TIMES**
- **DEFINE_FRICTION**
- **DEFINE_HEX_SPOTWELD_ASSEMBLY**
- **DEFINE_SD_ORIENTATION**
- **DEFINE_SPOTWELD_FAILURE_RESULTANTS**
- **DEFINE_SPOTWELD_RUPTURE_STRESS**
- **DEFINE_TABLE**
- **DEFINE_TRANSFORMATION**
- **DEFINE_VECTOR**

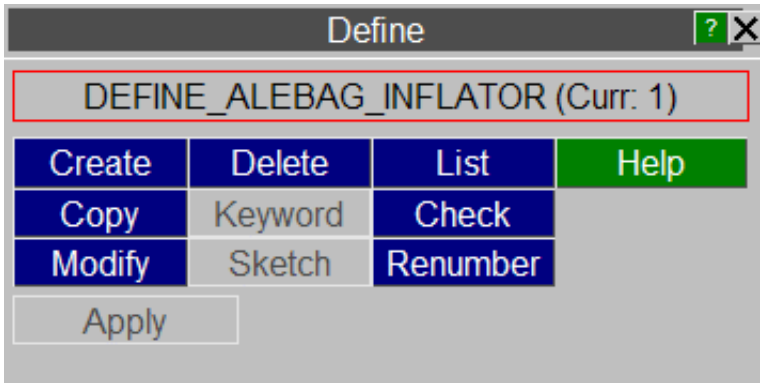


(DEFINE_) ALEBAG_INFLATOR:

These can be edited through their own specific editing panel (see below).

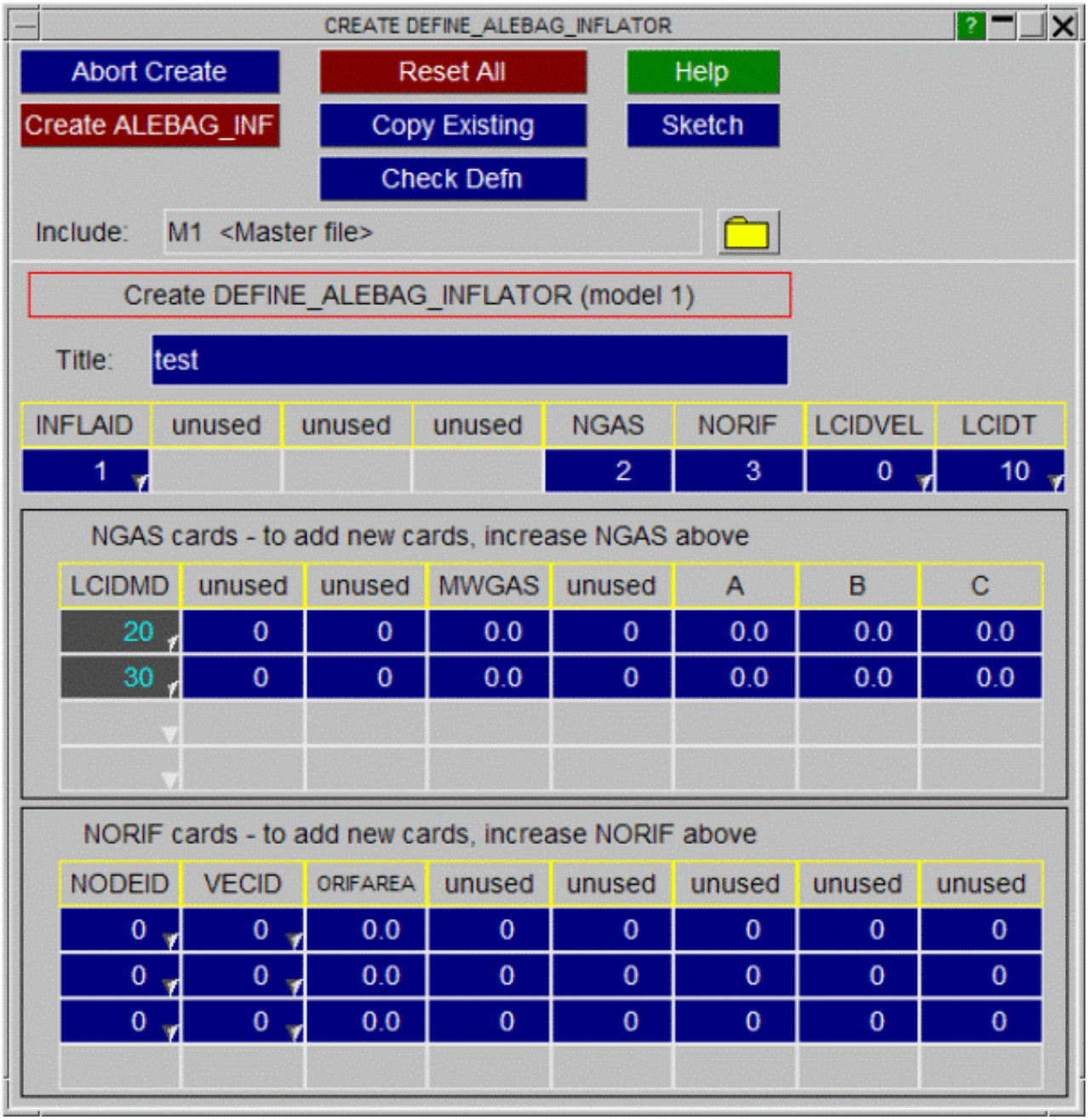
- Main Menu
- Creation
- Copying
- Editing
- Deletion
- List
- Check
- Renumber

This figure shows the main menu for the editing of alebag inflator definitions.
All functions have their standard meanings as given in section 5.1.1



CREATE Making a new alebag inflator definition.

This shows the create/edit panel for alebag inflators. New 'NGAS' or 'NORIF' rows can be added to this card by typing the required value into the **NGAS** or **NORIF** fields on the first line.



COPY Copy existing alebag inflator(s) to make a new alebag inflator(s).

The selected alebag inflators are copied. (alebag inflators do not "own" anything, so the concept of recursive copying does not apply.)

MODIFY Modifying the attributes of an existing alebag inflator.

MODIFY functions in the same way as **CREATE**, except that an initial definition will be present. Any modifications made to the alebag inflator definition will not be made permanent until the **UPDATE_ALEBAG_INF** button is pressed. At this point the local copy which has been updated is used to overwrite the version in the model.

DELETE Delete existing alebag inflator definitions.

The selected alebag inflators are deleted.
Alebag inflators do not "own" anything, so the concept of recursive deletion does not apply, however an alebag inflator that is referred to (ie "owned") by some higher order item will not be deletable unless that item is deleted too, or its reference to the alebag inflator removed.

LIST List alebag inflator summaries to screen

The selected alebag inflators are summarised on the screen.

CHECK Check alebag inflator definitions for errors

The selected alebag inflator definitions are run through the standard checking routines.

RENUMBER Change alebag inflator labels

RENUMBER lets you change any or all alebag inflator labels within a given model using the standard renumbering panel. To change the label of an individual alebag inflator it may be simpler just to **MODIFY** it.

(DEFINE_) BOX: Defining Boxes

- Main Menu
- Creation
- Copying
- Editing
- Deletion
- Visualisation

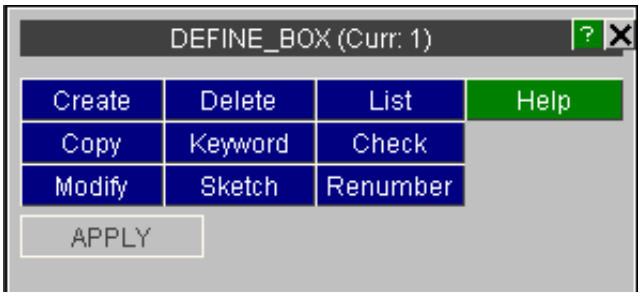
Box suffices:

_ADAPTIVE
_COARSEN
_DRAWBEAD
_SPH

The ***DEFINE_BOX** keyword is used to create "boxes". These are rectilinear volumes of space, defined by the min and max [x,y,z] coordinates, in global space units, of diagonally opposite corners. This imposes limitations upon their orientation when rotated - see below.
Boxes are used to bound the limits of other items, for example a contact may be defined by those elements which lie within a box. However they are not structural items and play no direct role during the actual analysis.

Boxes use unique labels and, although part of the ***DEFINE** keyword, their labels do not clash with other ***DEFINE_xxx** entities. For example it is legal to have **(*DEFINE_)BOX #1** and **(*DEFINE_)CURVE #1.**

This figure shows the main menu for the editing of box definitions.
All functions have their standard meanings as given in section 5.1.1



:

CREATE Making a new box definition.

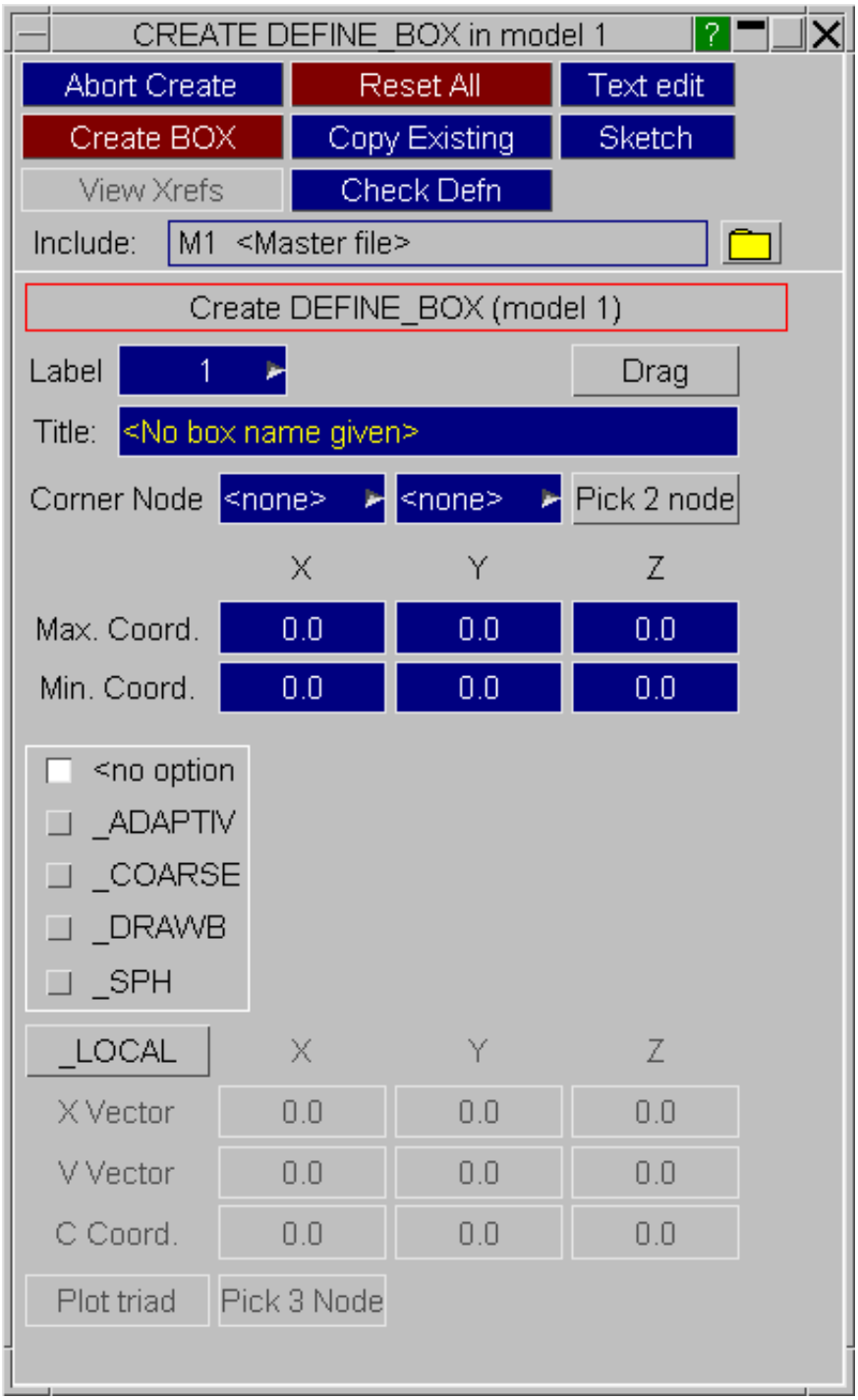
This figure shows the basic **CREATE** / **UPDATE BOX** panel.

Methods of defining box coordinates:

Corner Nodes: The box can be defined in terms of corner nodes in the current target model. In this example the box does not use corner nodes: **<none>** is displayed in the button field. A value for each node can either be typed in directly or chosen via the associated popup window (i.e. screen picking).

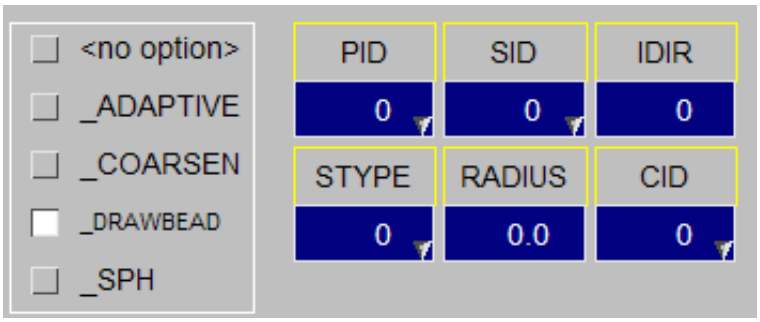
PICK 2 NODES Instead of defining each node separately, both nodes can be screen-picked together.

MIN / MAX COORDINATES The minimum and maximum X, Y and Z coordinates are displayed at the bottom of the basic editing panel. These numbers can be typed in directly if required.



Note that the coordinates of a box are independent of the methods used to define them. For example using nodes, by either method above, only extracts the coordinates of the nodes, and they do not become part of the box definition.

*DEFINE_BOX options: _ADAPTIVE, _COARSEN, _DRAWBEAD, _SPH, _LOCAL



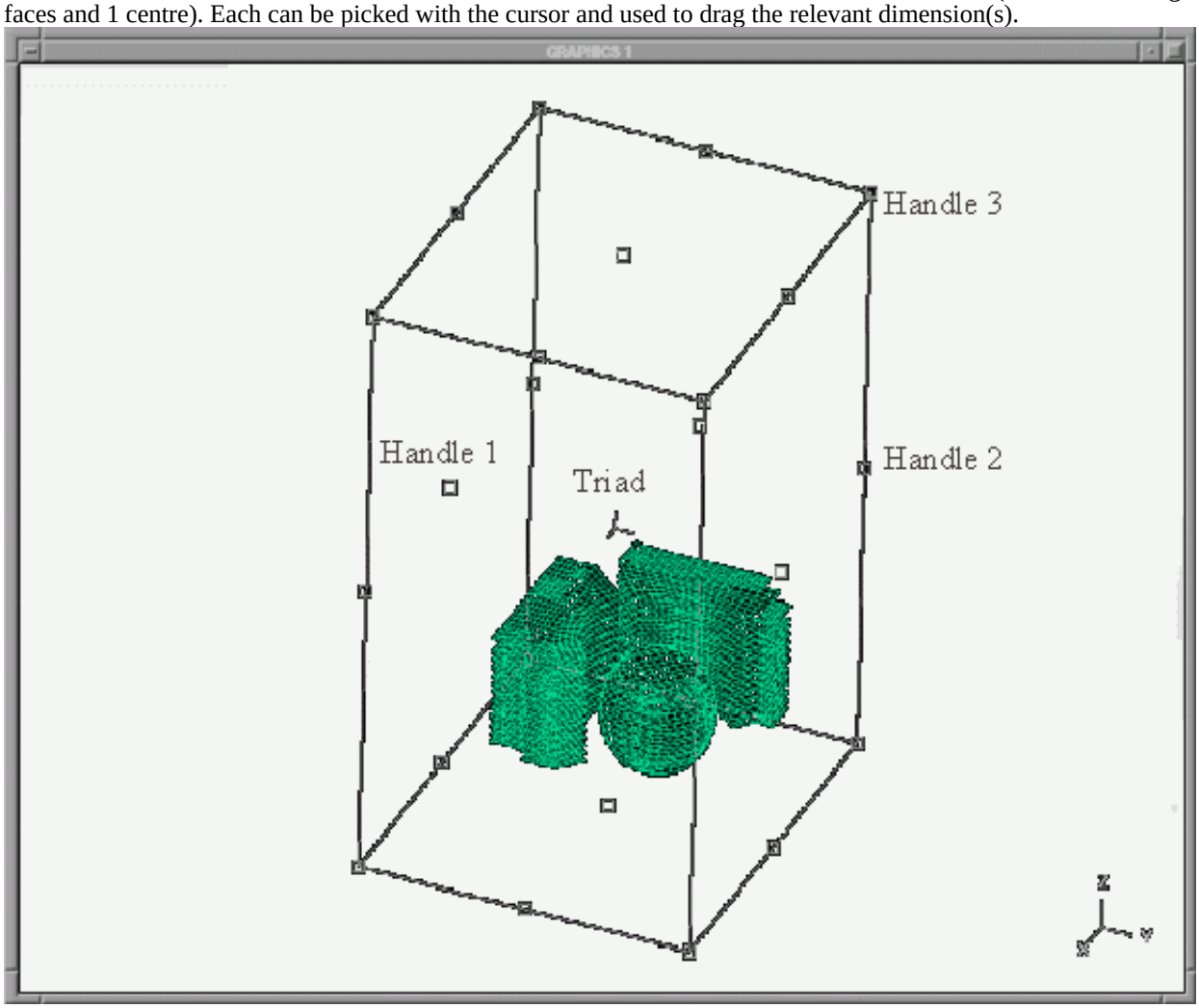
The various sub-types of box may be selected in the editor above.

The data entry rows will change accordingly. This example shows the _DRAWBEAD data.

The **_LOCAL** option allows you to specify a local coordinate system to create the **DEFINE_BOX** definition in. With this active, the max/min coord values will all apply in the local coordiante system rather than the global coordinate system. The local coordinate can be specified by typing in values defining the vectors of the local coordinate system, or by selecting 3 nodes.

DRAG "Dragging" a box size and shape interactively with the cursor.

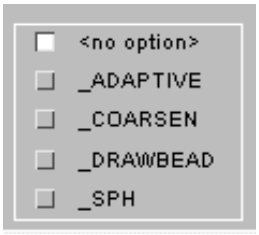
Once a box has been given some initial dimensions the cursor can be used to modify the dimensions and position of the box.
When **DRAG** is selected the current box definition is sketched, and 27 "handles" are added to it (8 corners, 12 edges, 6 faces and 1 centre). Each can be picked with the cursor and used to drag the relevant dimension(s).



The dragging "handles" are:

TRIAD (1 @ centre)	LEFT mouse button operates the X-coordinate position of the box; translating the whole box along the X-axis. Note that the X-coordinate buttons are turned green while the mouse button is depressed. MIDDLE mouse button translates the box through the Y-axis; Y-coordinate buttons are turned green. RIGHT mouse button translates the box through the Z-axis; Z-coordinate buttons are turned green.
HANDLE_1 (6: 1 @ each box face)	Any mouse button will allow the FACE of the box to translated along a vector normal to the face. Note that the coordinate buttons that are locked out turn red.
HANDLE_2 (12: 1 @ each box edge)	Any mouse button will allow an edge of the box to translated in the plane normal to the edge line. The coordinate box parallel to the edge will be locked out and turned red.
HANDLE_3 (8: 1 @ each box vertex)	Any mouse button will allow the corner of a box to be moved in any of the three axis directions

BOX_<options>
The radio buttons allow the selection the options **_ADAPTIVE, _COARSEN, _DRAWBEAD** and **_SPH**.



COPY Copy existing box(es) to make a new box(es).

The selected boxes are copied. (Boxes do not "own" anything, so the concept of recursive copying does not apply.)

MODIFY Modifying the attributes of an existing box.

MODIFY functions in the same way as **CREATE**, except that an initial definition will be present.

Any modifications made to the box definition will not be made permanent until the **UPDATE_BOX** button is pressed. At this point a the local copy which has been updated is used to overwrite the version in the model.

DELETE Delete existing box definitions.

The selected boxes are deleted.

Boxes do not "own" anything, so the concept of recursive deletion does not apply, however a box that is referred to (ie "owned") by some higher order item will not be deleteable unless that item is deleted too, or its reference to the box removed.

KEYWORD Creation / editing in the generic keyword editor

Boxes may be created, edited and deleted as a whole category in the generic keyword editor.

SKETCH Sketch box definitions on the current image.

SKETCH allows the user to select box definitions and superimpose a white sketch of them over the currently displayed image. The box and its contents (in the context in which it is being used) will be sketched if the **WITH CONTENTS** button is active.

LIST List box summaries to screen

The selected boxes are summarised on the screen.

CHECK Check box definitions for errors

The selected box definitions are run through the standard checking routines.

RENUMBER Change box labels

RENUMBER lets you change any or all box labels within a given model using the standard renumbering panel.

To change the label of an individual box it may be simpler just to **MODIFY** it.

Limitations of the box definition method that prohibit non-global orientations

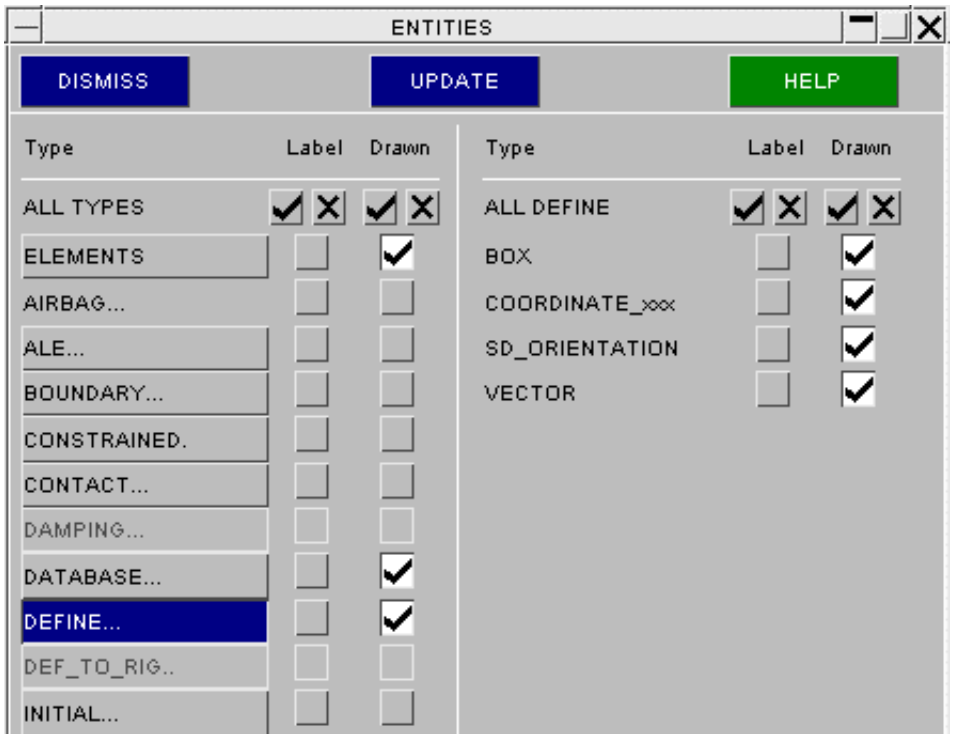
PRIMER allows you to create and edit boxes by both numerical (type in coordinates) and screen-based (pick and drag) methods. You can also transform them via the **ORIENT** function.

Note on ROTATION OF BOXES: The way boxes are specified in LS_DYNA (coordinates of two corners in global units) means that they are always aligned with the global [x,y,z] axes. If a box is rotated by any angle other than a multiple of 90 degrees it will be expanded to a global box which surrounds the actual rotated shape of the original box. Thus a box rotated by by 45 degrees will expand by a factor of 1.4. In such cases, the user will be warned and the contents of the new box sketched.

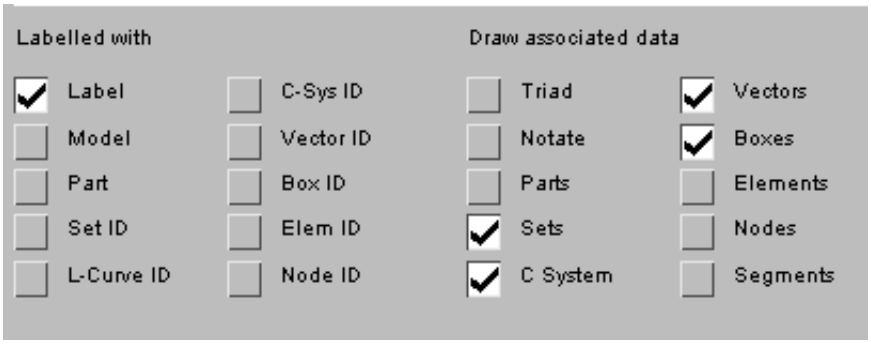
Visualising Boxes.

Boxes may be drawn by turning their display on in the **ENT**ity Viewing menu.

They can also be drawn via the **SKETCH** options above.



They may also be drawn in other contexts (for example contacts) if their display as "associated data" in the **ENT**ity Viewing menu is selected.

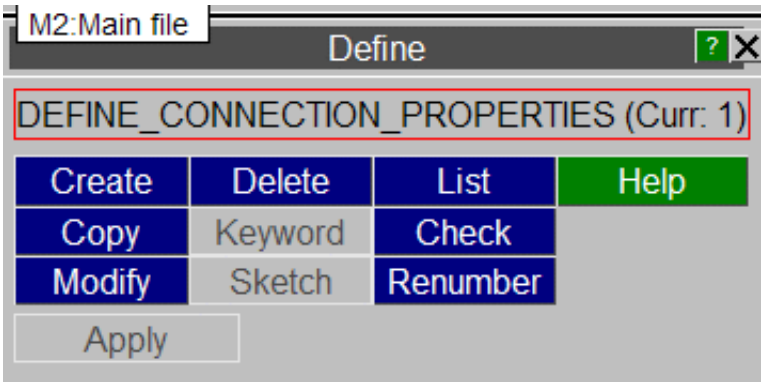


(DEFINE_) CONNECTION_PROPERTIES:

These can be edited through their own specific editing panel (see below).

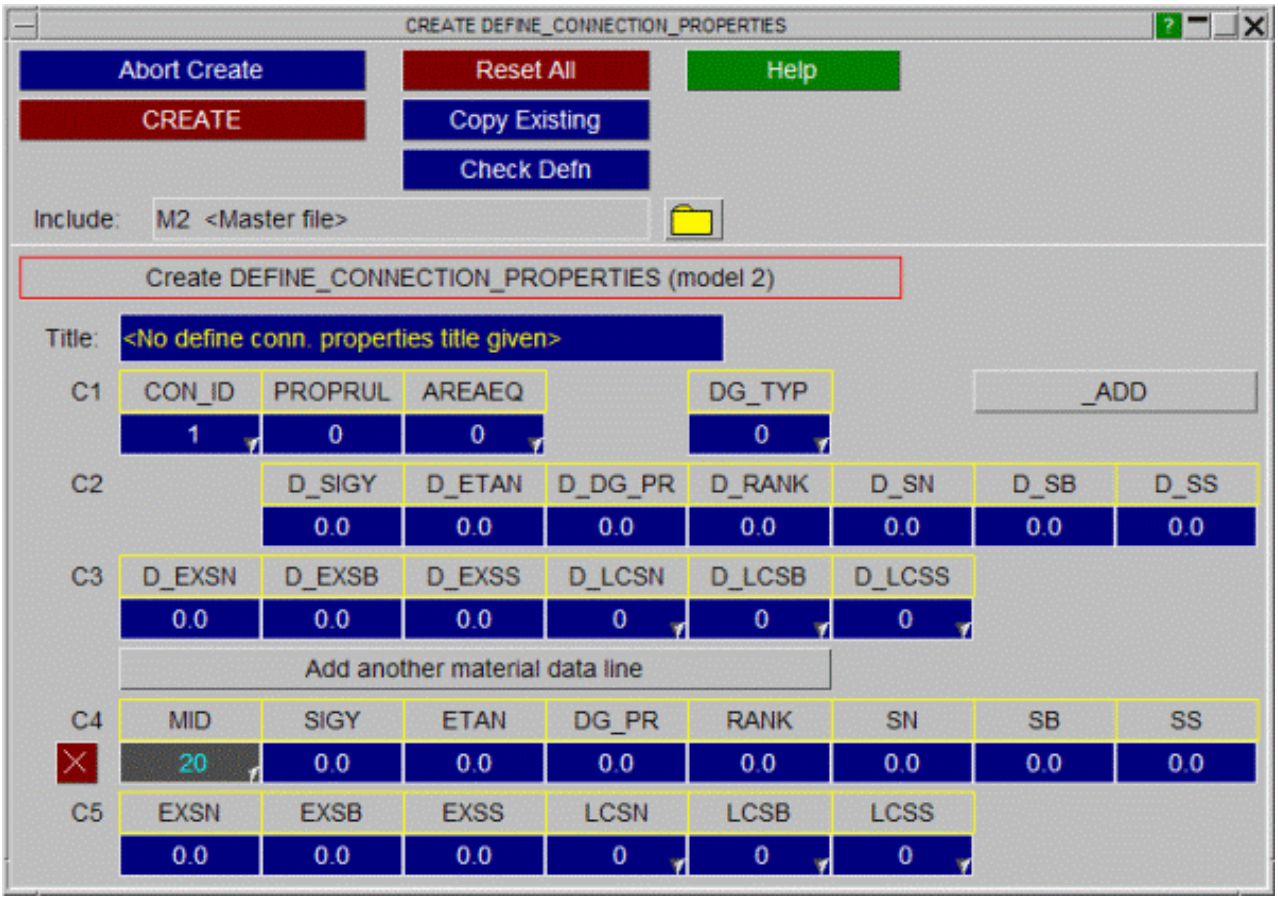
- Main Menu
- Creation
- Copying
- Editing
- Deletion
- List
- Check
- Renumber

This figure shows the main menu for the editing of connection properties definitions.
All functions have their standard meanings as given in section 5.1.1



CREATE Making a new connection properties definition.

This shows the create/edit panel for connection properties. New material data lines can be added by clicking on the **Add another material data line** button. The _ADD option can be activatied by clicking on the **_ADD** button. With the _ADD option active, cards 2 and 3 are greyed out.



COPY Copy existing connection properties(s) to make a new connection properties(s).

The selected connection properties are copied. (connection properties do not "own" anything, so the concept of recursive copying does not apply.)

MODIFY Modifying the attributes of an existing connection properties.

MODIFY functions in the same way as **CREATE**, except that an initial definition will be present. Any modifications made to the connection properties definition will not be made permanent until the **UPDATE** button is pressed. At this point the local copy which has been updated is used to overwrite the version in the model.

DELETE Delete existing connection properties definitions.

The selected connection properties are deleted.

Connection properties do not "own" anything, so the concept of recursive deletion does not apply, however a connection property that is referred to (ie "owned") by some higher order item will not be deletable unless that item is deleted too, or its reference to the connection propertie removed.

LIST List connection properties summaries to screen

The selected connection properties are summarised on the screen.

CHECK Check connection properties definitions for errors

The selected connection properties definitions are run through the standard checking routines.

RENUMBER Change connection properties labels

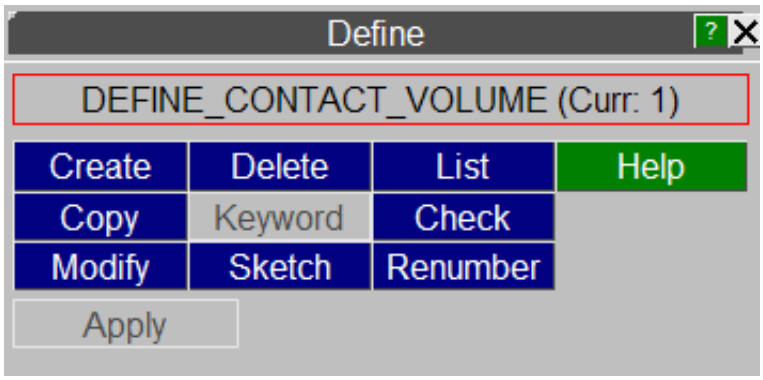
RENUMBER lets you change any or all connection properties labels within a given model using the standard renumbering panel. To change the label of an individual connection properties it may be simpler just to **MODIFY** it.

(DEFINE_) CONTACT_VOLUME:

These can be edited through their own specific editing panel (see below).

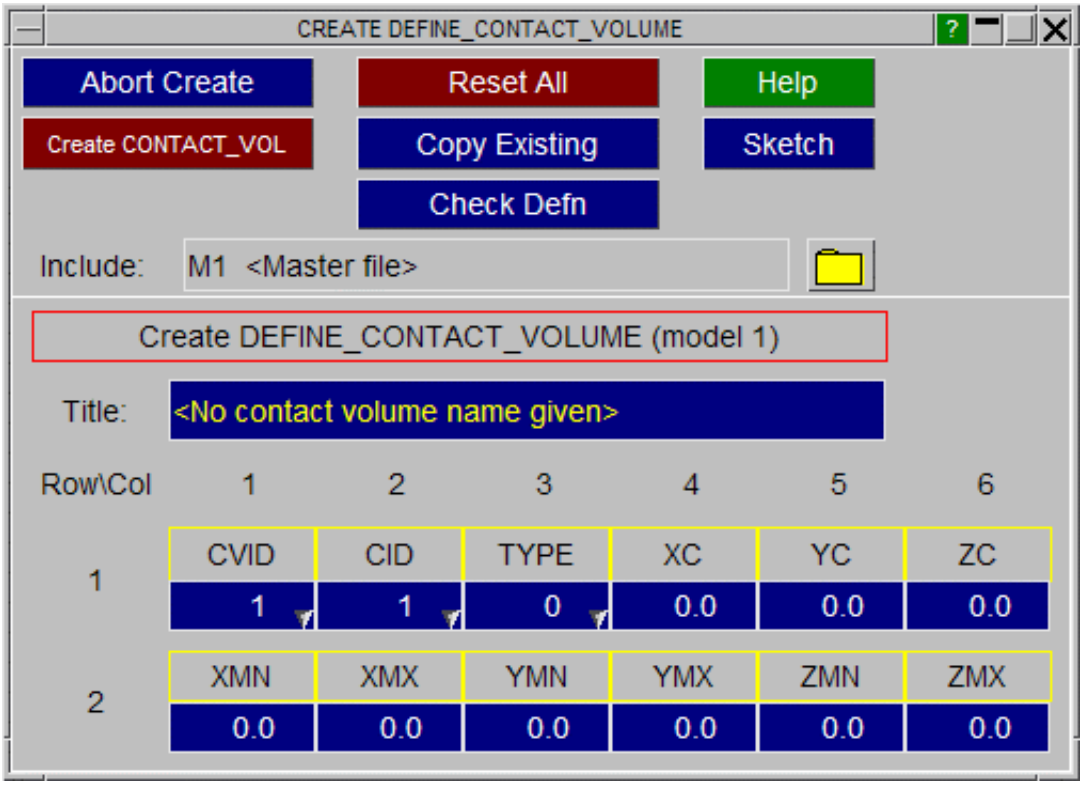
- Main Menu
- Creation
- Copying
- Editing
- Deletion
- Sketching
- List
- Check
- Renumber

This figure shows the main menu for the editing of contact volume definitions.
All functions have their standard meanings as given in section 5.1.1



CREATE Making a new contact volume definition.

This shows the create/edit panel for contact volumes. The second row of the card will change depending on the value chosen for **TYPE.**



COPY Copy existing contact volume(s) to make a new contact volume(s).

The selected contact volumes are copied. (contact volumes do not "own" anything, so the concept of recursive copying does not apply.)

MODIFY Modifying the attributes of an existing contact volume.

MODIFY functions in the same way as **CREATE**, except that an initial definition will be present. Any modifications made to the contact volume definition will not be made permanent until the **UPDATE_CONTACT_VOL** button is pressed. At this point the local copy which has been updated is used to overwrite the version in the model.

DELETE Delete existing contact volume definitions.

The selected contact volumes are deleted.

Contact volumes do not "own" anything, so the concept of recursive deletion does not apply, however a contact volume that is referred to (ie "owned") by some higher order item will not be deletable unless that item is deleted too, or its reference to the contact volume removed.

SKETCH Sketch contact volume definitions.

SKETCH draws the contact volume on top of the current graphics image.

LIST List contact volume summaries to screen

The selected contact volumes are summarised on the screen.

CHECK Check contact volume definitions for errors

The selected contact volume definitions are run through the standard checking routines.

RENUMBER Change contact volume labels

RENUMBER lets you change any or all contact volume labels within a given model using the standard renumbering panel. To change the label of an individual contact volume it may be simpler just to **MODIFY** it.

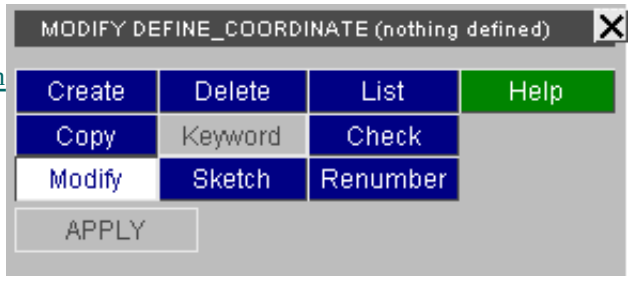
(DEFINE_) COORDINATE: Defining Coordinate Systems.

The ***DEFINE_COORDINATE** keyword is used to create local coordinate systems. Three points in space, which form two vectors are required. The coordinate system is then computed from the cross product of these vectors. They are used when a system is required that is not orthogonal to the global axes. For example boundary conditions, orthtropic materials and beam orientations.

Coordinate systems use unique labels and, although part of the ***DEFINE** keyword, their labels do not clash with other ***DEFINE_xxx** entities. For example it is legal to have (***DEFINE_)BOX #1** and (***DEFINE_)COORDINATE #1**.

- Main Menu
- Creation
- Copying
- Editing
- Deletion
- Visualisation

This figure shows the main menu for the editing of co-ordinate systems.
All functions have their standard meanings as given in section 5.1.1:



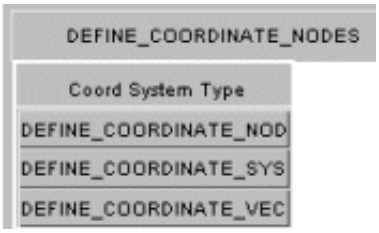
CREATE Making a coordinate definition.

This figure shows the basic **CREATE** / **UPDATE COORDINATE_SYSTEM** panel.

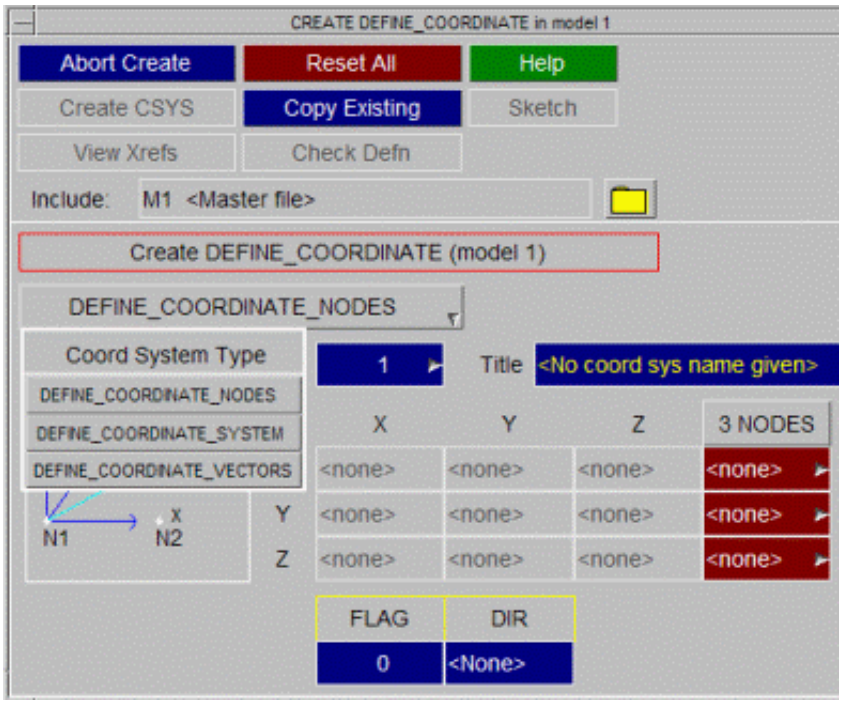
There are three ways in LS-Dyna of defining a coordinate system:

DEFINE_COORDINATE_NODES
_SYSTEM
_VECTORS

The popup menu gives these options:



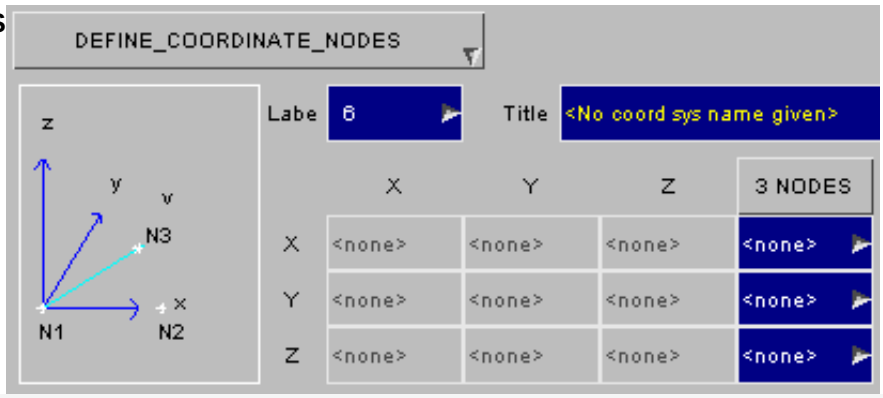
The detailed layout of the panels and definition methods vary slightly as shown below.



DEFINE_COORDINATE_NODES
Is defined by three nodes:

N1 : origin
N2 : Gives local X axis from N1N2
N3 : forms the local XY plane N1N2N3

Methods of defining the nodes:



3 NODES

Instead of defining each node separately, all nodes can be screen-picked together. Simply screen pick three nodes in the order:

- N1 (origin)
- N2 (local X vector)
- N3 (lies on local XY plane)

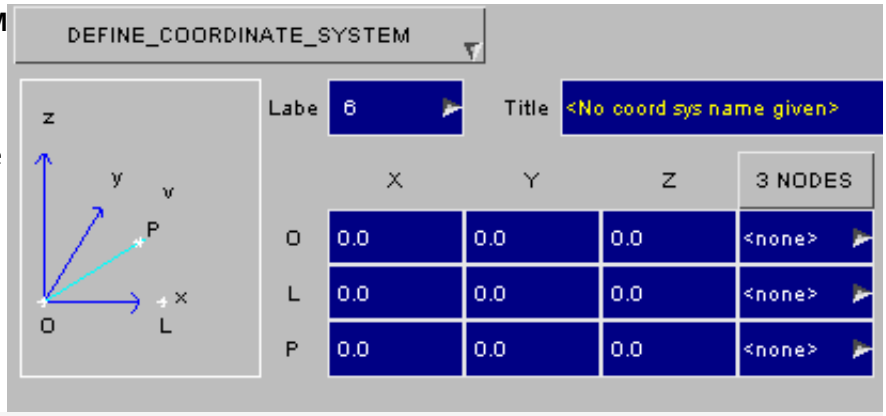
<Individually>

Alternatively use the individual popup menus to select nodes, or simply type in their labels.

DEFINE_COORDINATE_SYSTEM
Is defined by three points:

P1 : origin
P2 : Gives local X axis from P1P2
P3 : forms the local XY plane P1P2P3

Methods of defining the points:



3 NODES
(Only their coordinates are used)

Instead of defining each node separately, all nodes can be screen-picked together. Simply screen pick three nodes in the order:

- N1 (origin)
- N2 (local X vector)
- N3 (lies on local XY plane)

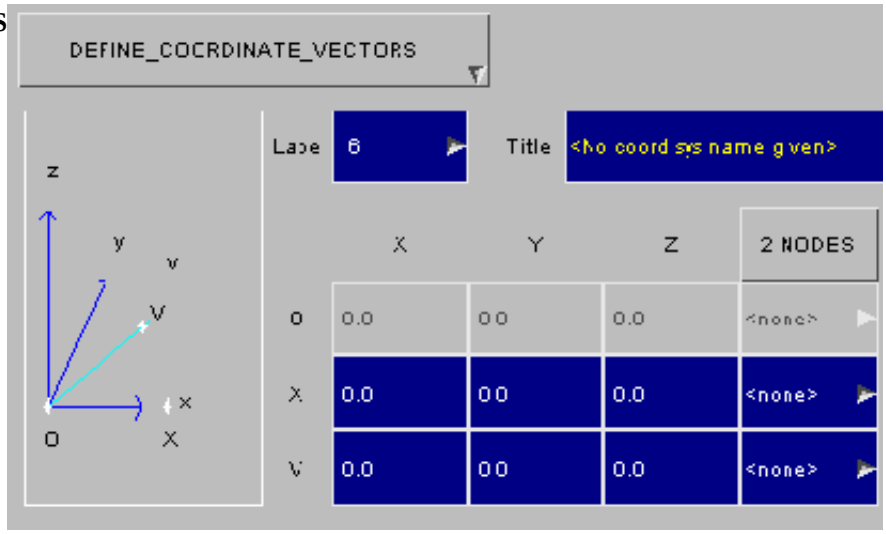
<Individually>

Alternatively use the individual popup menus to select nodes, or simply type in their labels.
Or simply type in the coordinates explicitly.

DEFINE_COORDINATE_VECTORS
Is defined by the origin and 2 points:

Or : origin
P1 : Gives local X axis from OrP1
P2 : forms the local XY plane OrP1P2

Methods of defining the points:



2 NODES (Only their coordinates are used)	Instead of defining each node separately, both nodes can be screen-picked together. Simply screen pick three nodes in the order: • N1 (local X vector) • N2 (lies on local XY plane)
<Individually>	Alternatively use the individual popup menus to select nodes, or simply type in their labels. Or simply type in the coordinates explicitly.

COPY Copy existing coordinate(s) to make a new coordinate.

The selected coordinates are copied. (Coordinates do not "own" anything, so the concept of recursive copying does not apply.)

MODIFY Modifying the attributes of an existing coordinate.

MODIFY functions in the same way as **CREATE**, except that an initial definition will be present.

Any modifications made to the section definition will not be made permanent until the **UPDATE_CSYS** button is pressed. At this point a the local copy which has been updated is used to overwrite the version in the model.

DELETE Delete existing coordinate definitions.

The selected coordinates are deleted.

Coordinate definitions do not "own" anything, so the concept of recursive deletion does not apply, however a coordinate that is referred to (ie "owned") by some higher order item will not be deletable unless that item is deleted too, or its reference to the coordinate removed.

SKETCH Sketch coordinate definitions on the current image.

Allows the user to select coordinate systems and superimpose a white sketch of them over the currently displayed image.

LIST List coordinate summaries to screen

The selected coordinate definitions are summarised on the screen.

CHECK Check coordinate definitions for errors

The selected coordinate definitions are run through the standard checking routines.

RENUMBER Change coordinate labels

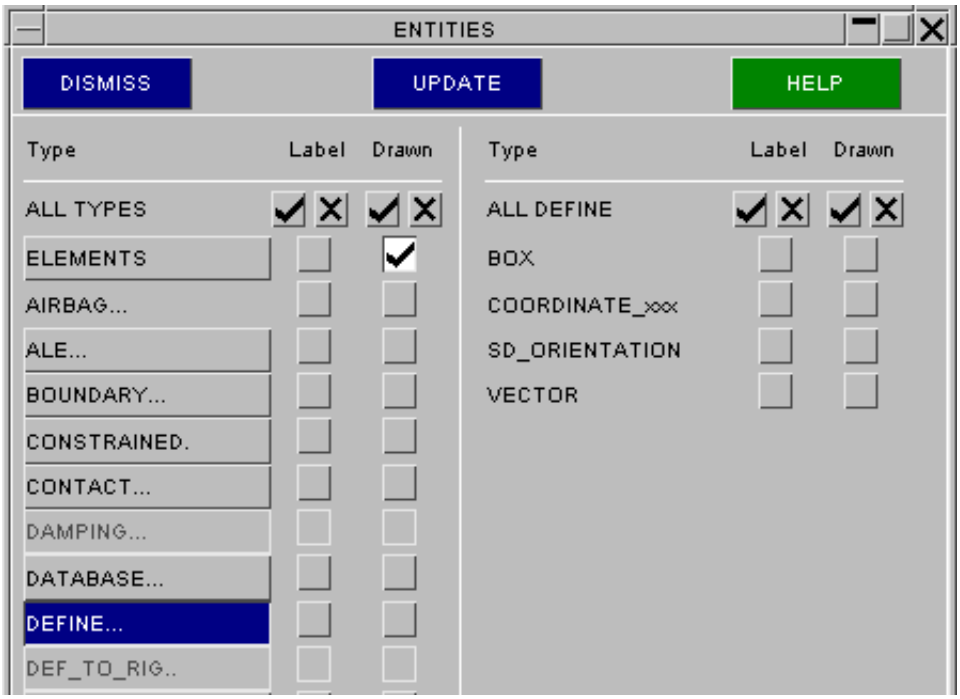
Lets you change any or all coordinate labels within a given model using the standard renumbering panel.

To change the label of an individual coordinate it may be simpler just to **MODIFY** it.

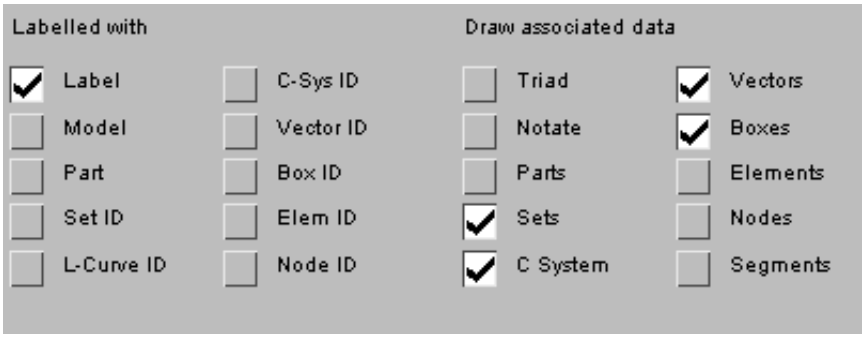
Visualising Coordinate systems.

Co-ordinate systems may be drawn by turning their display on in the **ENT**ity Viewing menu.

They can also be drawn via the **SKETCH** options above.



They may also be drawn in other contexts (for example contacts) if their display as "associated data" in the **ENT**ity Viewing box is selected.



(DEFINE_) CURVE/TABLE: Defining Load Curves.

- Main Menu
- Creation
- Copying
- Editing
- Deletion

Other curve suffices:

_COMPENSATION
_ENTITY
_FEEDBACK
_FUNCTION
_SMOOTH
_TRIM

The ***DEFINE_CURVE** keyword is used to create "loadcurves". These are lists of two or more (x, y) data points which are used extensively for defining loading (e.g. force vs. time), material properties (e.g. stress vs. strain) and other varying data in an LS-DYNA analysis.

The ***DEFINE_TABLE** keyword defines a table of loadcurves. A table is an ordered set of data pairs consisting of a value and a loadcurve id, typically a strain rate and a stress:strain characteristic. It is an unfortunate quirk of the LS-DYNA keyword input that the loadcurves belonging to a table **must follow it in sequential order**. PRIMER endeavours to maintain this ordering but care must be taken if decks are edited manually, or split into *include files, to ensure that this order is adhered to. In later versions of LS-DYNA the _2D option has been added to remove the need for this strict ordering.

Loadcurves do not have any explicit data types or units associated with them, this is implied by the items which reference them. It is legal, but generally not sensible, for any number of unrelated items to use the same loadcurve. It may cause problems for Unit change operations.

PRIMER keeps track of what references each loadcurve, and hence the implied data types and units for each axis, which makes it possible to detect and correct conflicting usage.

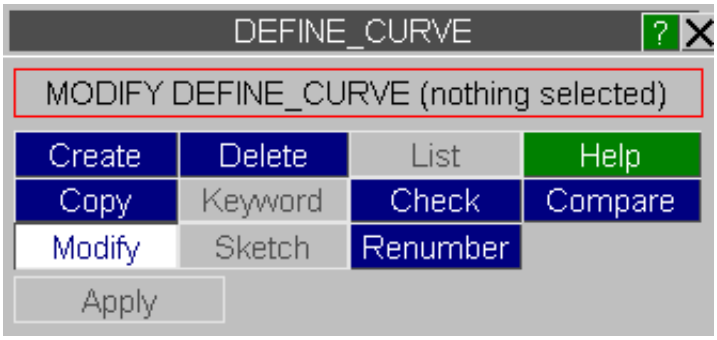
Loadcurves use unique labels and, although part of the ***DEFINE** keyword, their labels do not clash with other ***DEFINE_xxx** entities ***except tables*** (see below). For example it is legal to have **(*DEFINE_)BOX #1** and **(*DEFINE_)CURVE #1.**

LOADCURVES.

NOTE: TABLE and **CURVE** definitions occupy the same labelling space, and are interchangeable in some contexts. Thus it is ***not*** legal to have **TABLE#1** and **CURVE#1**.

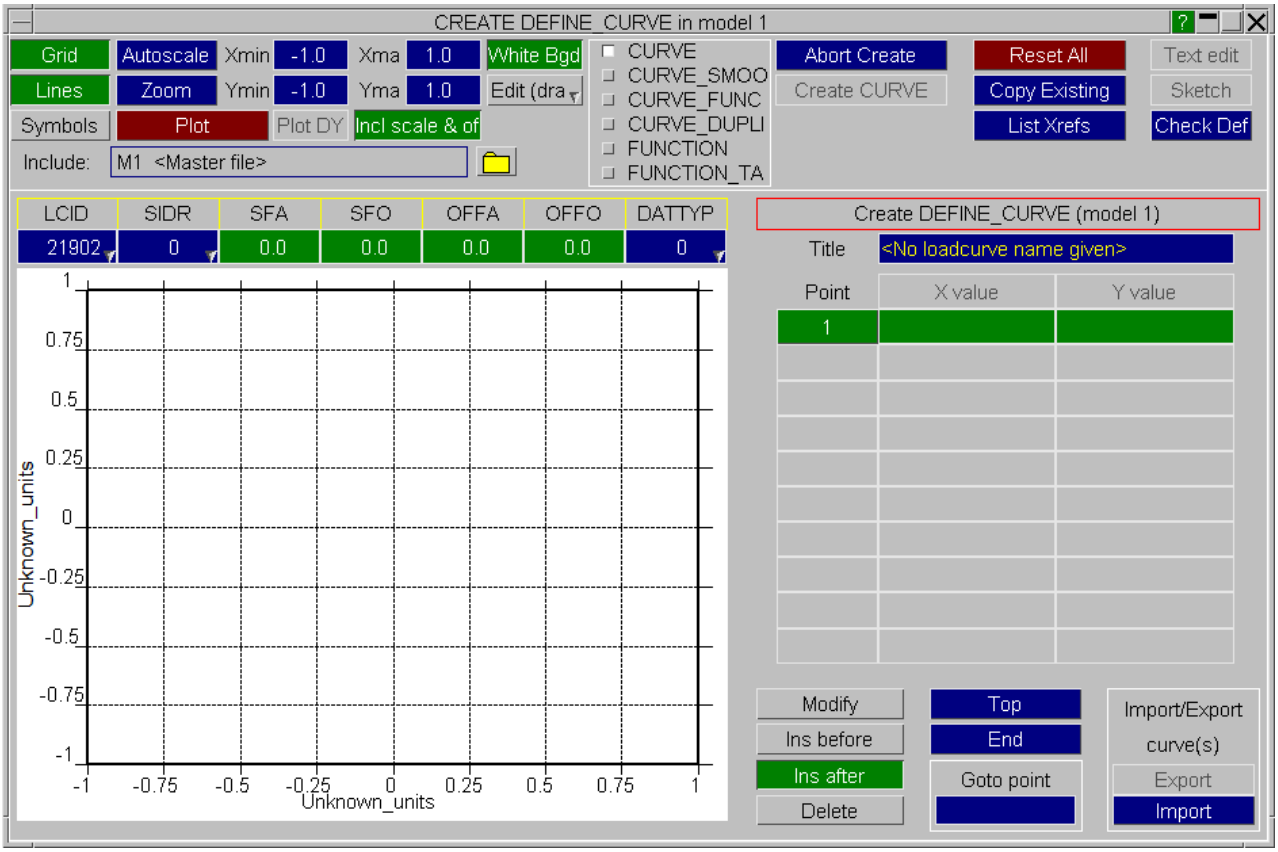
This figure shows the main menu for the editing of curves.
All functions have their standard meanings as given in section 5.1.1. The **COMPARE** option is specific to curves, and is described below.

The table and curve main menu panels are similar.



CREATE Making a new loadcurve definition.

This figure shows the basic **CREATE** / **UPDATE CURVE** panel.



The loadcurve editing panel layout is shown in this figure.
There are six main areas in the panel, each area grouping together buttons of similar function.

1. **Loadcurve display buttons**
2. ***DEFINE_CURVE options.**
3. **Loadcurve plot.**
4. **Create/abort loadcurve.**
5. **Loadcurve points.**
6. **Loadcurve modification.**

(1) Control how loadcurve is displayed on the screen.	(4) Create/abort curve. xrefs, help
(2) *DEFINE_CURVE options	
(3) Loadcurve plot	(5) Loadcurve points
	(6) Modify points in loadcurve

(1) Loadcurve display buttons

GRID, LINES & SYMBOLS

These buttons toggle whether the grid, the curve line and the curve symbols are drawn on the plot.

AUTOSCALE

Resets the scaling on the loadcurve plot so the curve just fits the screen and replots the loadcurve.

ZOOM

Two points are selected using the left mouse button. **Xmin, Xmax, Ymin** and **Ymax** are updated and the curve is replotted

PLOT

The curve is replotted as the scale currently selected by **Xmin, Xmax, Ymin** and **Ymax**.

Xmin, Xmax, Ymin & Ymax

Typing in a value changes the limits for plotting the curve.

By default when creating a curve **Xmin** and **Ymin** are -1. **Xmax** and **Ymax** are 1. When modifying an existing curve they are set so the curve just fits on the screen (equivalent to **AUTOSCALE**)

PLOT DYNA

Will display the curve points discretized according to LS-DYNA rules.

WHITE BGD

Toggle between white and black background

EDIT

When toggled on, this button can be used to drag, insert or delete points from the displayed curve(s). Right click on the button to change the option. In **drag** mode, left mouse click and hold on a point on the curve, then drag. In **insert** mode, left mouse click at a point on the curve where you wish to add a point. In **delete** mode, either left mouse click on a point on the curve to delete it, or left mouse click hold to drag a box around a number of points to delete them.

INC SCALE+OFFSET

By default when a curve is plotted on the screen the offsets (**SFA, SFO**) and scale factors (**OFFA, OFFO**) are not included. If this button is pressed then they are included in the plot. To ensure that the user is aware of this the **SFA, SFO, OFFA** and **OFFO** text boxes turn green (by default they are blue) and the curve line and symbols are plotted in green. Pressing the button again toggles the inclusion off.

Loadcurve values are scaled after the offsets are applied.

Abcissa value = SFA x *(Defined value + OFFA)*

Ordinate value = SFO x *(Defined value + OFFO)*

(2) *DEFINE_CURVE options.

LCID	SIDR	SFA	SFO	OFFA	OFFO	DATTYP
<none>	0	1.000	1.000	0.0	0.0	0

LCID

Label for loadcurve. If there is no label then the label is shown as **<none>** and the box is red rather than the default blue. A new label can be typed in the box or the right mouse button pressed to get the standard label popup box.

SIDR

Sets whether the loadcurve will be used in a transient or dynamic relaxation analysis. Either type in the value or use the right mouse button to bring up a popup menu.

Stress initialisation by dynamic relaxation
0: Used in transient analysis only
1: Used in dynamic relaxation only
2: Used in dynamic relaxation and transient

SFA, SFO, OFFA & OFFO

Scale factors and offsets for the loadcurve abcissa (x) and ordinate (y) values.

DATTYP

Sets the type of data in the loadcurve: generally this is set to zero. Either type in a value or use the right mouse button to bring up a popup menu.

Data type
0: Monotonically increasing x values
1: General xy data

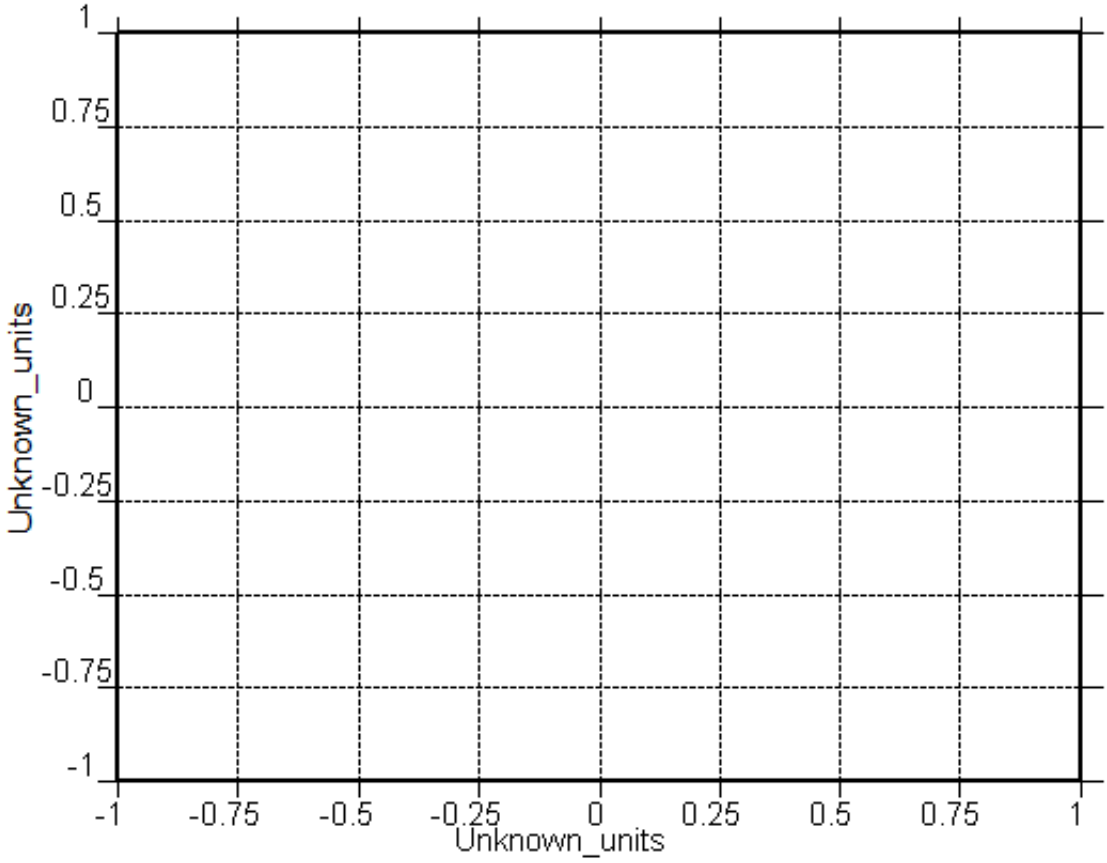
(3) Loadcurve plot.
The loadcurve currently being created or modified is plotted in the bottom left of the loadcurve panel.

If the curve has no cross references then the units for the X and Y axes are shown as Unknown units. If there are cross references, the first reference that PRIMER finds is used and the units displayed on the X and Y axes. All the cross references for the curve can be displayed with the **LIST_XREFS** button.

The visibility of the curve lines, symbols and the grid is controlled by the **GRID, LINES** and **SYMBOLS** buttons.

If the **INC_SCALE+OFFSET** button is selected, the curve is drawn in green instead of the default yellow to inform the user that the scale factors and offsets are included in the plot.

The plot can be updated at any time by pressing the **PLOT** button.



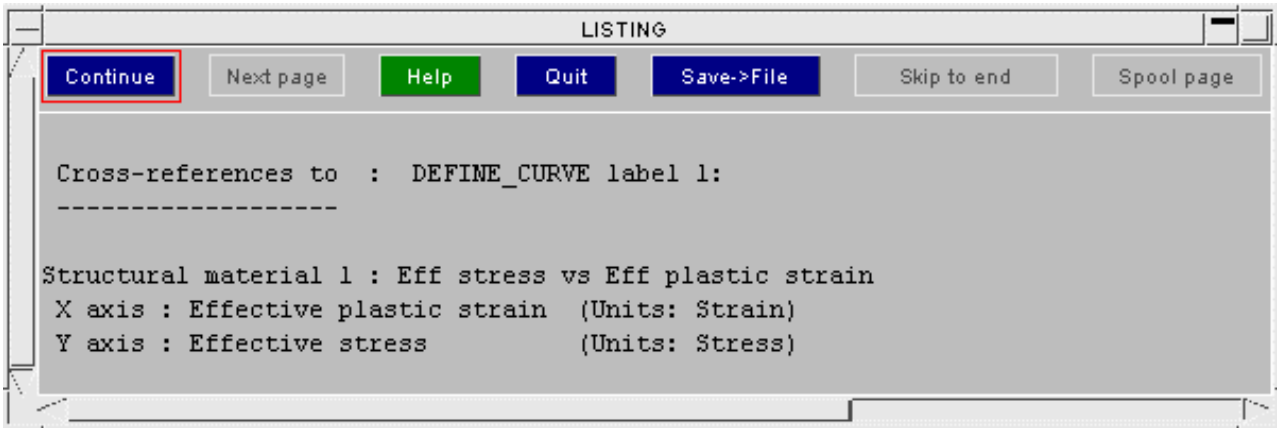
(4) Create/abort loadcurve.

ABORT_CREATE
Aborts from the current loadcurve creation without saving any of the modifications.

CREATE_CURVE
This will exit the current loadcurve creation, saving the curve in the database. This button will be inactive (greyed out) until a label (**LCID**) is given for the loadcurve and there are at least two points in the curve.

LIST_XREFS

If the loadcurve has any cross references in the database they are displayed in a dialogue window. If there are no cross references to this curve [**no cross references found**] will be displayed.



RESET_ALL

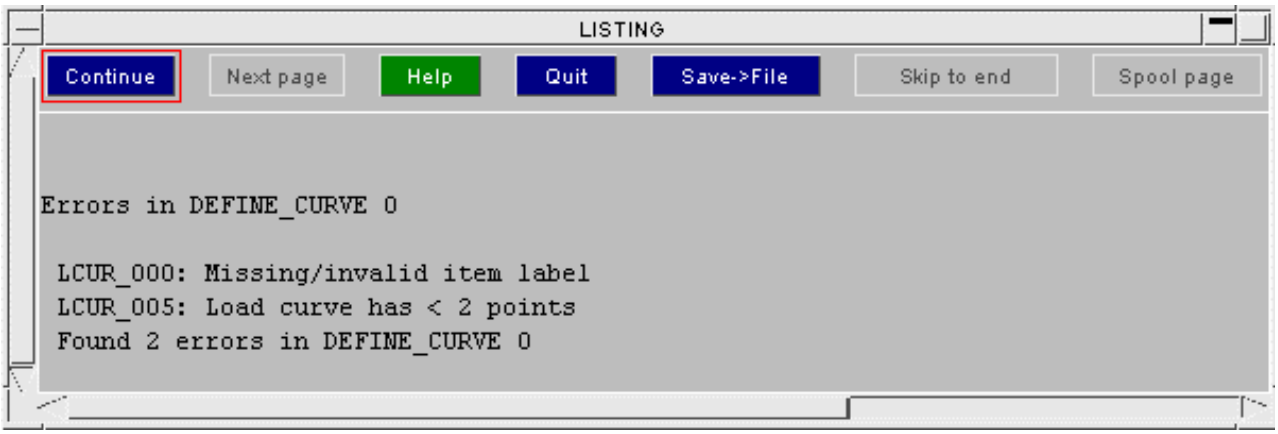
Resets the curve back to its initial state. Any points which have been added or modified are lost.

COPY_EXISTING

Copies the data from an existing loadcurve into the loadcurve currently being created. Any points which have been added since starting the create will be lost.

CHECK_DEFN

Checks the loadcurve currently being created for any errors.



HELP

Displays the help pages for loadcurves

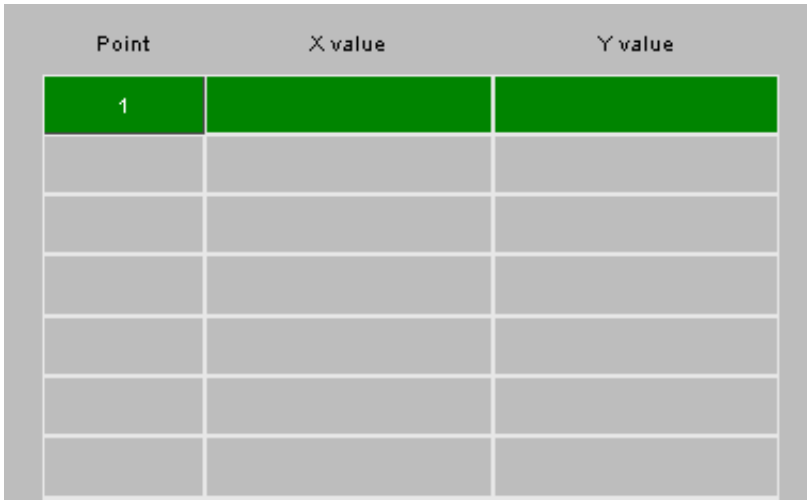
SKETCH

Sketch is currently inoperative.

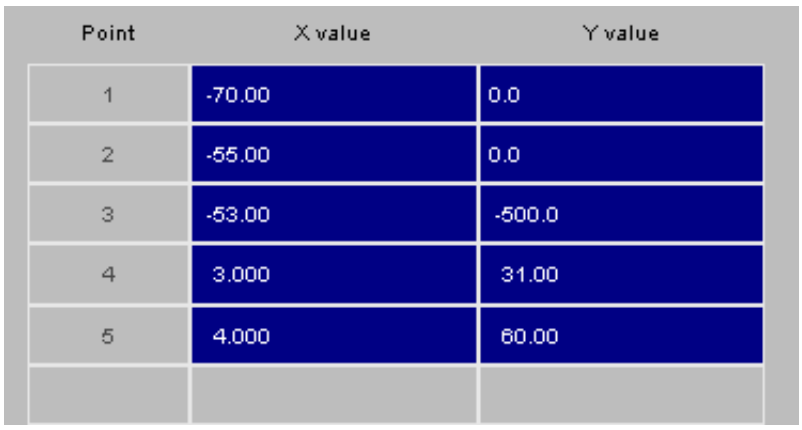
(5) Loadcurve points.
When a curve is created the user is forced into **INS_AFTER** mode until a point is created. The user will not be able to change to another mode (**MODIFY, INS_BEFORE** or **DELETE**) until this point is created.

The **X** and **Y value** boxes for this initial point are blank and coloured green to indicate that a number is required. If a number is typed into one of the boxes the box turns blue.

When numbers are present for X and Y the line for point 1 becomes blue and point 2 becomes green. As many points as necessary can be added using this method.



When there is more than one point the current mode can be changed at any time by pressing the **MODIFY, INS_BEFORE** or **DELETE** buttons.
Any incomplete points (i.e. if either the X or Y values [or both] are blank) will be deleted when changing mode. If the number of points in the loadcurve is greater than 10 a sliding bar appears by the side of the points. The mouse can be used to select which points are visible in the text box. Drag the bar up and down with the left mouse button to move between the points. Alternatively clicking on the up (or down) arrow with the left, middle or right mouse button, increases (or decreases) the points shown by 1, 10 or 100 respectively. The value of a point can be changed in any mode by clicking on the X or Y value box and typing in a number.



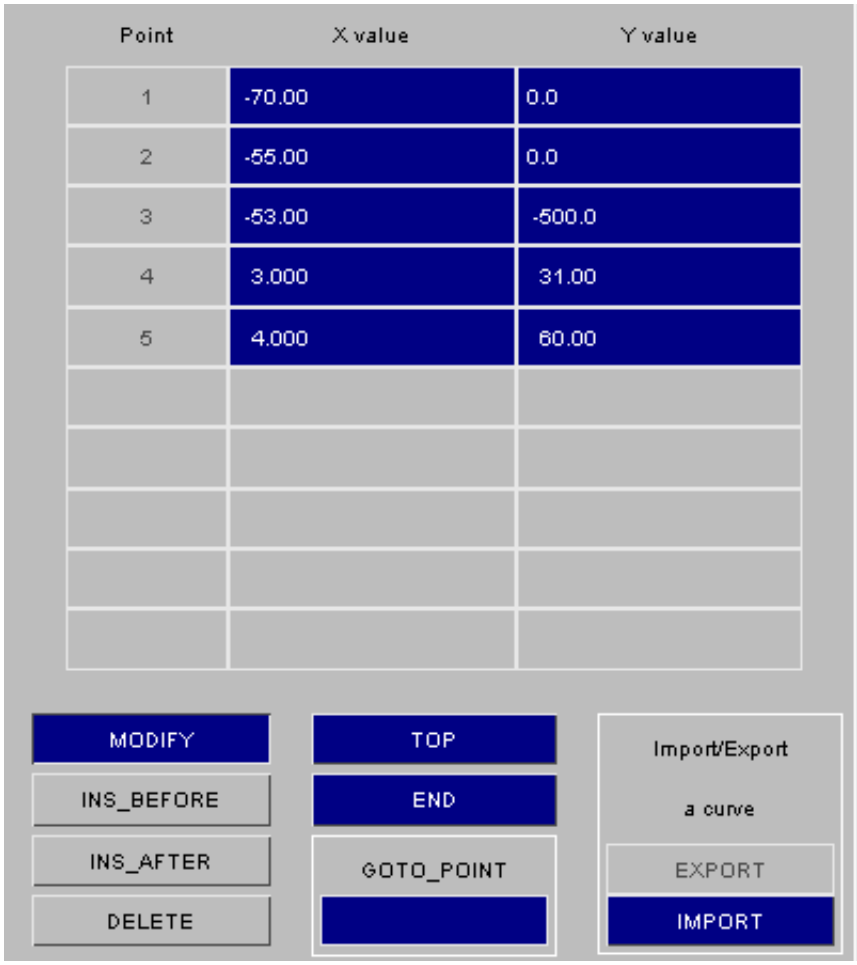
(6) Loadcurve modification.

MODIFY

Selects modify mode for loadcurve point editing. In this mode only the values of the points can be changed. No points can be added or deleted.

When in this mode the point buttons are greyed out so they cannot be selected.

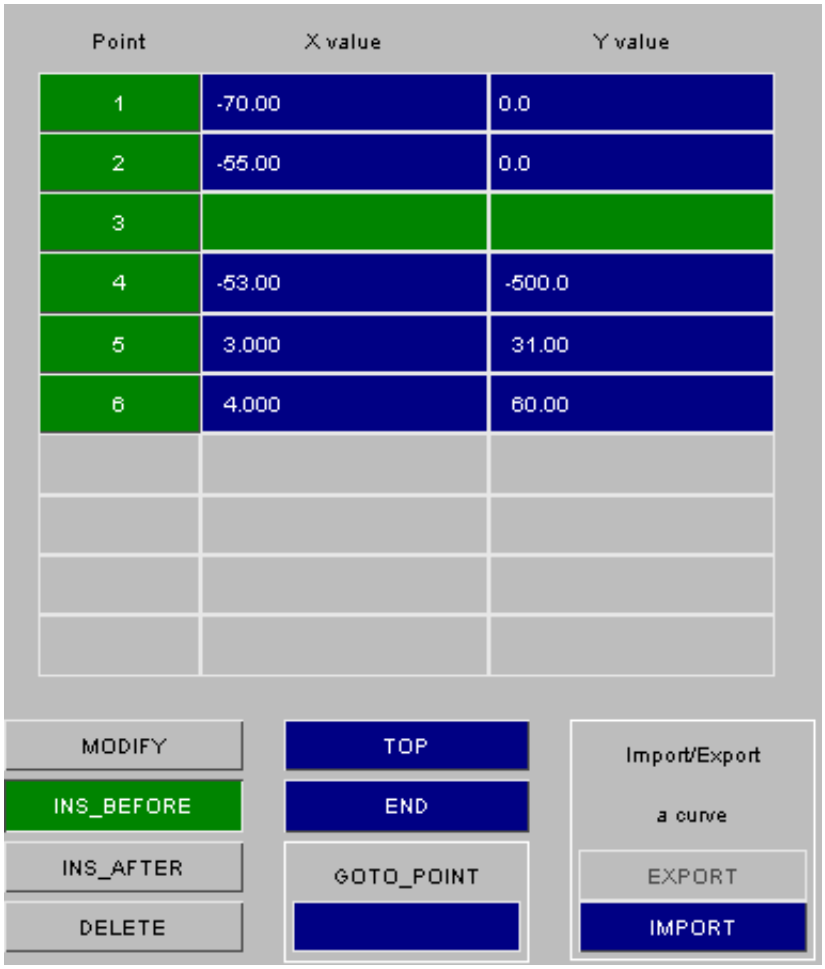
If a point is currently being edited in **INS_BEFORE** or **INS_AFTER** mode it is deleted before the modify mode is selected.



Select either **INS_BEFORE** or **INS_AFTER** mode for loadcurve point editing. This allows points to be added to the curve.

When in these modes the point buttons turn green. If a point is selected by clicking with the mouse a new point is created either before or after (depending on which mode) the selected point. If a point is currently being added in this mode and another point is selected the current point is deleted and the new point added.

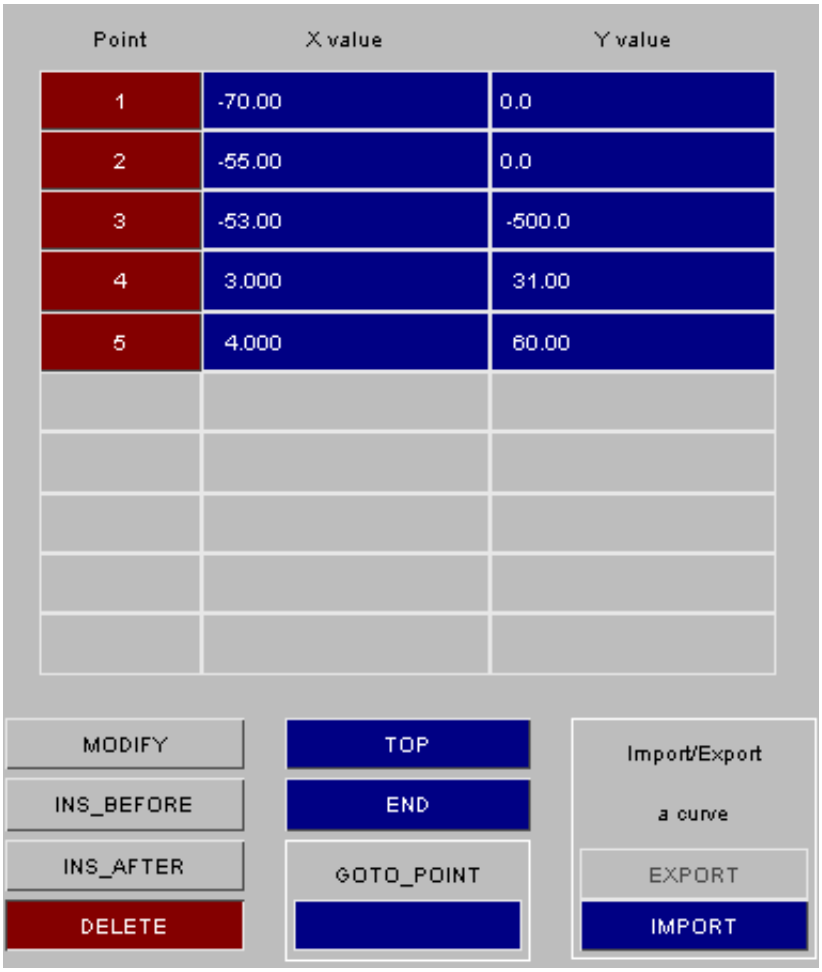
INS_BEFORE & INS_AFTER



DELETE

Selects delete mode for loadcurve point editing. In this mode points can be deleted as well as being able to change the values of the points.

When in this mode the point buttons turn red. If a point is selected by clicking with the mouse it is deleted. If a point is currently being edited in **INS_BEFORE** or **INS_AFTER** mode it is deleted before the delete mode is selected.



TOP & **END**

Moves the slider automatically to the top or end of the points for the loadcurve

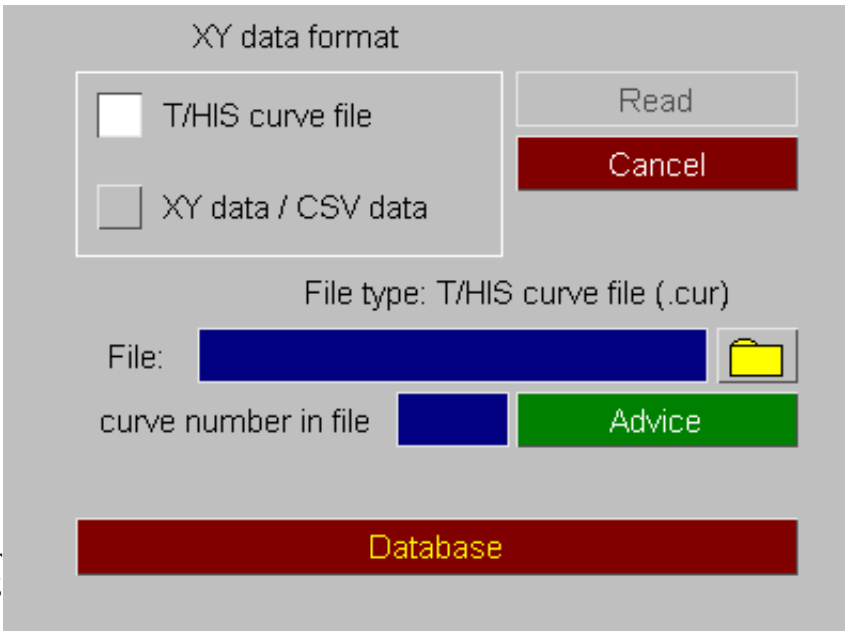
GOTO_POINT

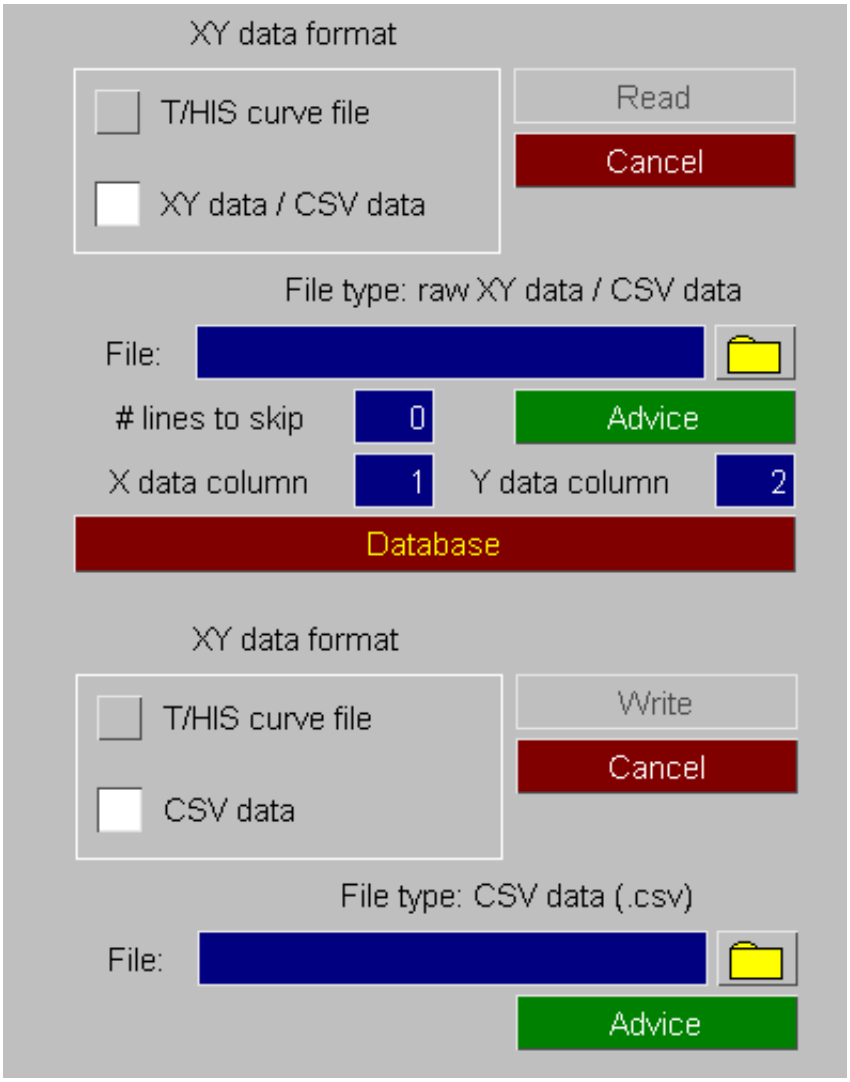
Moves the slider so that the point number which is typed in is visible.

IMPORT

Allows a loadcurve to be read from an external file or from a database in PRIMER. Pressing the **IMPORT** button brings up a new set of buttons instead of the loadcurve points.

Two types of file can be read into the loadcurve editor. T/HIS curve files and raw x,y data. The formats of these files is given in Appendix VIII. The format of the file to import is selected by using the RADIO buttons. The filename can then either be typed in the file text box or selected by browsing using the ? button. T/HIS curve files can contain multiple curves in one file. In this case the curve number in the file to read should be given. If no number is given the first curve in the file will be read. For the XY data/CSV data option you can specify the number of lines to skip at the start of the file, and also the columns for the X and Y data, should they not be in column 1 and 2.





EXPORT

Allows a loadcurve to be written to an external file from PRIMER. Pressing the **EMPORT** button brings up a new set of buttons instead of the loadcurve points.

Two types of file can be written from the loadcurve editor. T/HIS curve files and CSV x,y data. The formats of the T/HIS curve data is given in Appendix VIII. The format of the file to import is selected by using the RADIO buttons. The filename can then either be typed in the file text box or selected by browsing using the ? button.

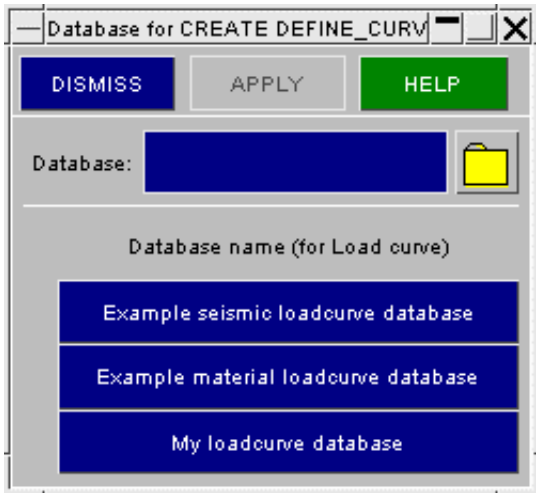
READ
Reads the selected file into the loadcurve editor and plots the curve. Any modifications to the current curve will be lost when importing a file.

CANCEL
Aborts the import and returns to the normal loadcurve editor window.

DATABASE
The **DATABASE** button starts the loadcurve database function in PRIMER.

A list of the available loadcurve databases will be shown on the screen. When one is selected a curve can be read from the database.

For further details on databases see section 5.2 and Appendix IX.



COPY Copy existing loadcurve(s) to make a new loadcurve(s).

The selected loadcurves are copied. (Loadcurves do not "own" anything, so the concept of recursive copying does not apply.)

MODIFY Modifying the attributes of an existing loadcurve.

MODIFY functions in the same way as **CREATE**, except that an initial definition will be present. Any modifications made to the loadcurve definition will not be made permanent until the **UPDATE_CURVE** button is pressed. At this point a the local copy which has been updated is used to overwrite the version in the model.

DELETE Delete existing loadcurve definitions.

The selected loadcurves are deleted.

Loadcurves do not "own" anything, so the concept of recursive deletion does not apply, however a loadcurve that is referred to (i.e. "owned") by some higher order item will not be deletable unless that item is deleted too, or its reference to the loadcurve removed.

SKETCH Sketch loadcurve definitions.

SKETCH is currently inoperative.

LIST List loadcurve summaries to screen

The selected loadcurves are summarised on the screen.

CHECK Check loadcurve definitions for errors

The selected loadcurve definitions are run through the standard checking routines.

RENUMBER Change loadcurve labels

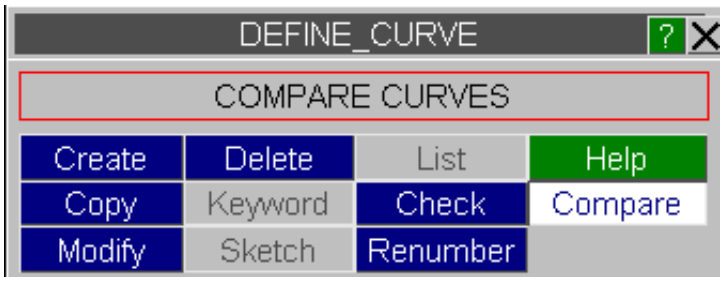
RENUMBER lets you change any or all loadcurve labels within a given model using the standard renumbering panel. To change the label of an individual loadcurve it may be simpler just to **MODIFY** it.

T/HIS Send load curve to T/HIS

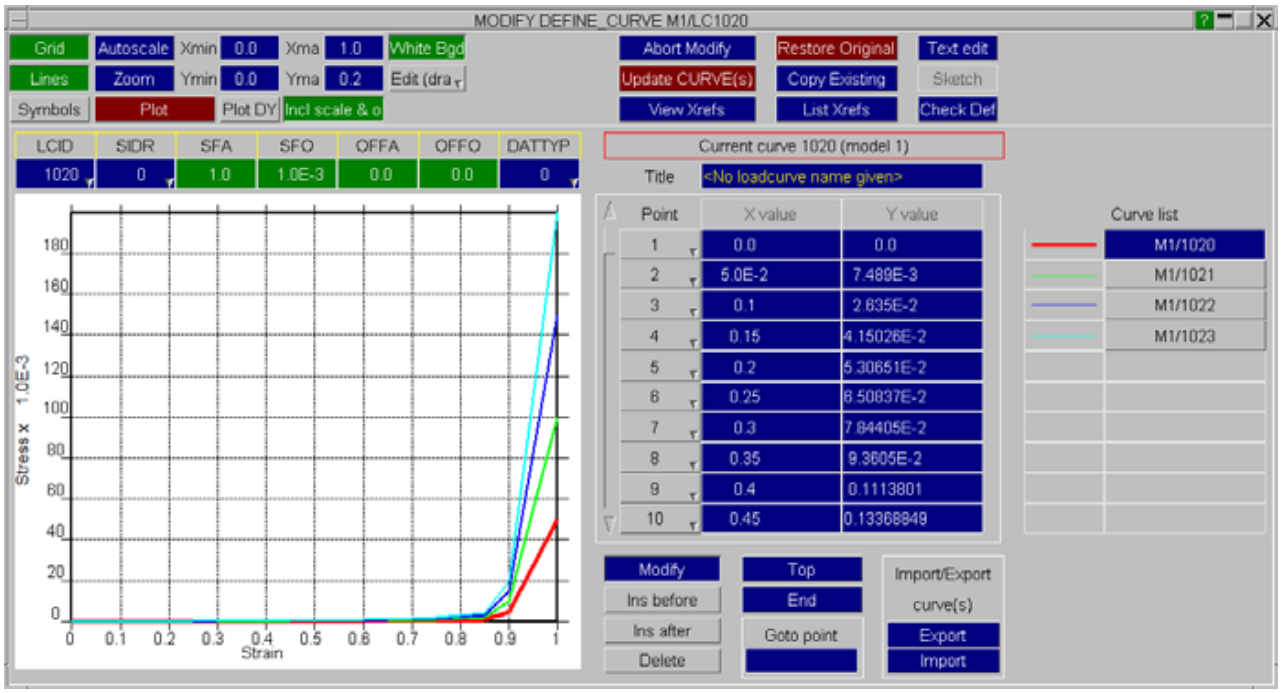
T/HIS lets you send any load curve to the linked session of T/HIS, a plotting tool which lets you perform various operation on curves and send back updated curve in PRIMER. To send a load curve, go to a load curve edit panel and hit **T/HIS**. In case there is no T/HIS session linked to PRIMER, it will launch T-HIS first and load the curve. A limit of 1200 points on the curve is set that can be sent to T/HIS without a linked T-HIS session. After an active T-HIS linked session, it should be fine to send curves more than 1200 points. See Section 3.17.1 for more details about shared memory link between T-HIS and PRIMER.

CURVE COMPARE

From the main **`*DEFINE_CURVE`** panel, there is a **COMPARE** option. This is used to visually plot and compare multiple curves on one graph.



After selecting the curves you wish to compare, the curve compare panel will open.

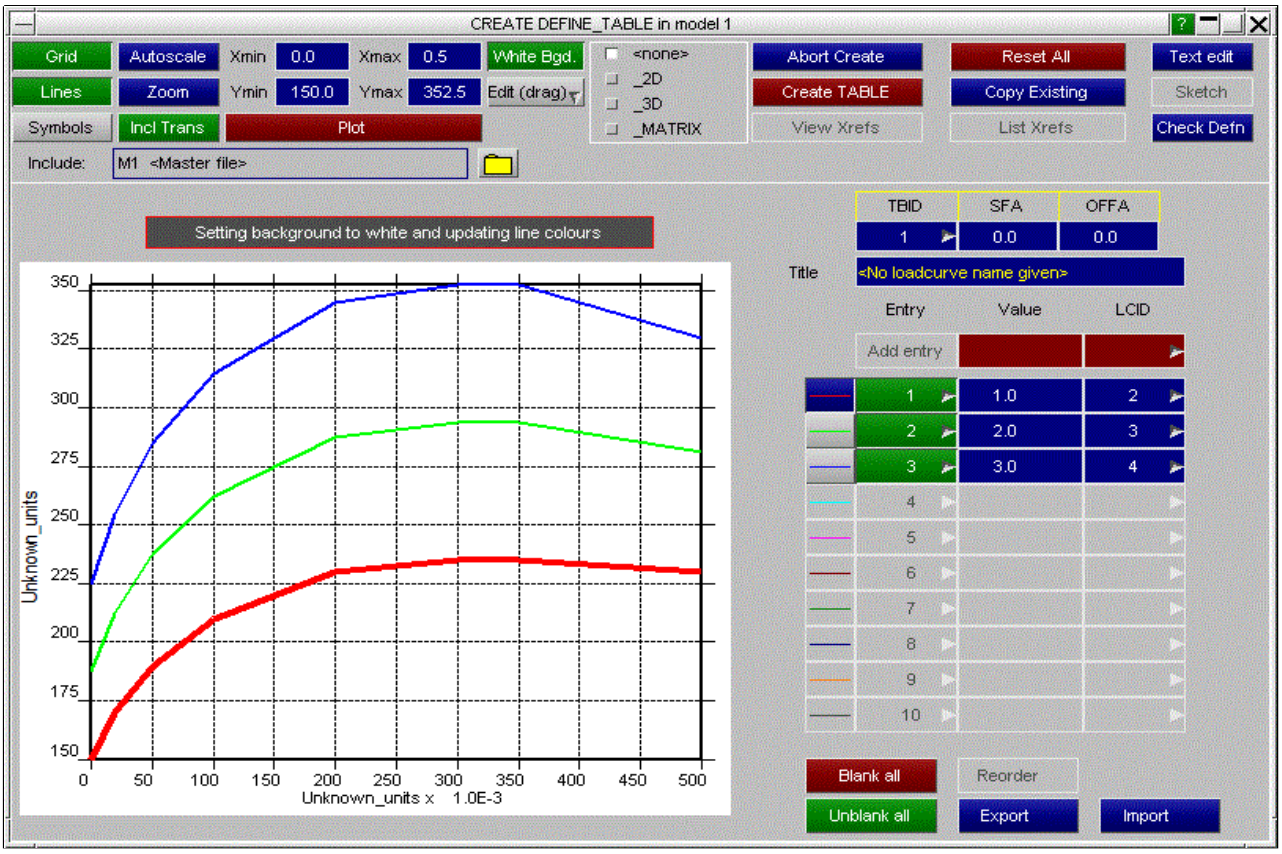


The curve compare panel is similar to the normal curve edit panel, but there is a list of the selected curves on the left hand side. Selecting different curves in this list will display the data associated with that curve in the points list. You still have all the same options for editing the curve data on this panel.

TABLES

CREATE/MODIFY Making/modifying a table definition.

This figure shows the **CREATE/UPDATE TABLE** panel.



The functionality of the table editing panel is similar to the panel for loadcurves. The following features are briefly

described.

Adding a new entry: when a value and loadcurve id (select, create, or type in) are entered into the top box **ADD_ENTRY** will become active. The new entry will automatically be added to the correct row.

Deleting an entry: a row is deleted from the table using popup of the row id button. The curve itself is NOT deleted.

Sketching an entry: the same popup may be used to sketch a single curve

Blanking/Unblanking an entry: clicking on the green (unblanked) row id button will toggle it to red (blanked)

Editing an entry: any row may be modified by typing in a new value and new loadcurve id. The latter may also be selected through the **SELECT** popup. Further, a loadcurve itself may be created/edited as the loadcurve editing panel can be accessed directly via the **CREATE** & **EDIT** popup. You are also able to interactively drag/insert/delete points from the curves on tables using the **EDIT** button.

Reordering entries: If entries become out of order the **REORDER** button will sort them.

Exporting a table: exports all the table curves to a t/his format (2e20) curve file. The table values are output as a step diagram.

Importing a table: imports all the table curves from a t/his format. Curves are created independently from the table creation. The table values are imported as a step diagram.

<none>, _3D, _3D: toggles between the various **`*DEFINE_TABLE`** options.

*DEFINE_CURVE_COMPENSATION

This definition defines a curve for local compensation..

PRIMER will read them in and write them out, but no interactive editing of them is provided

*DEFINE_CURVE_ENTITY

This definition defines a curve of straight line segments and circular arcs that defines an axisymmetric surface.

PRIMER will read them in and write them out, but no interactive editing of them is provided

*DEFINE_CURVE_FEEDBACK

These definitions do not create a loadcurve, rather they add special metal-forming attributes to an existing curve definition.

PRIMER will read them in and write them out, but no interactive editing of them is provided.

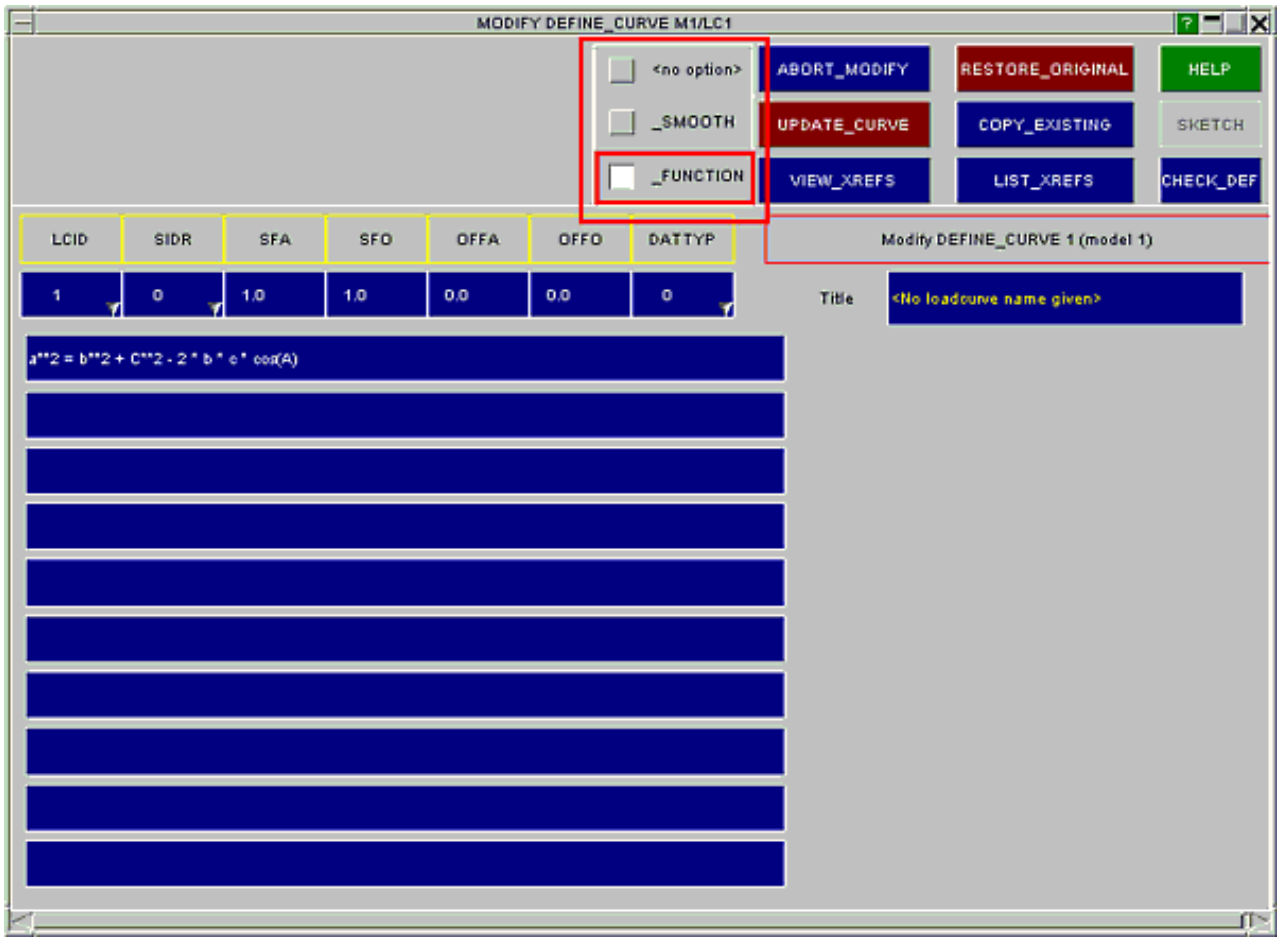
*DEFINE_CURVE_FUNCTION

These definitions are an alternative way of creating a loadcurve. Instead of explicit <x,y> data the user enters up to 10 rows of pseudo-fortran syntax, which may also contain references to functions that return the current state of LS-DYNA entities during an analysis.

PRIMER reads these in and writes them out, but does not attempt to evaluate them. As a consequence only very limited checking is provided for them.

These definitions may be edited in the normal curve editor as follows:

- Select **_FUNCTION** in the options box of the curve editor
- Enter rows of data.

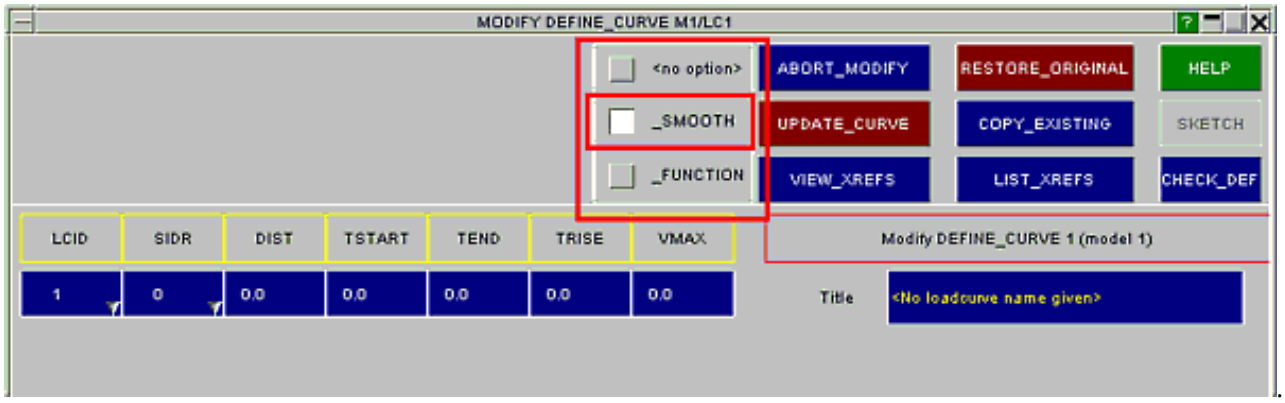


*DEFINE_CURVE_SMOOTH

This is yet another way to define a loadcurve. In this case a smoothly varying "hump" function is defined in terms of its parameters.

These definitions may be edited in the normal curve editor as follows:

- Select **_SMOOTH** in the options box of the curve editor
- Enter the relevant parameters



*DEFINE_CURVE_TRIM

These are not loadcurves at all (the name is misleading), rather they are geometrical definitions used with *ELEMENT_TRIM during springback analyses. Their label sequence is totally separate from that of conventional loadcurves, and there is no relationship between the two types.

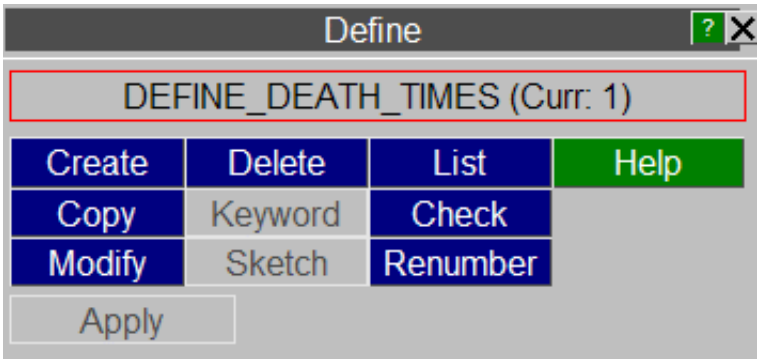
PRIMER reads in and writes out these definitions, but does not provide any interactive editing or visualisation of them.

(DEFINE_) DEATH_TIMES:

These can be edited through their own specific editing panel (see below).

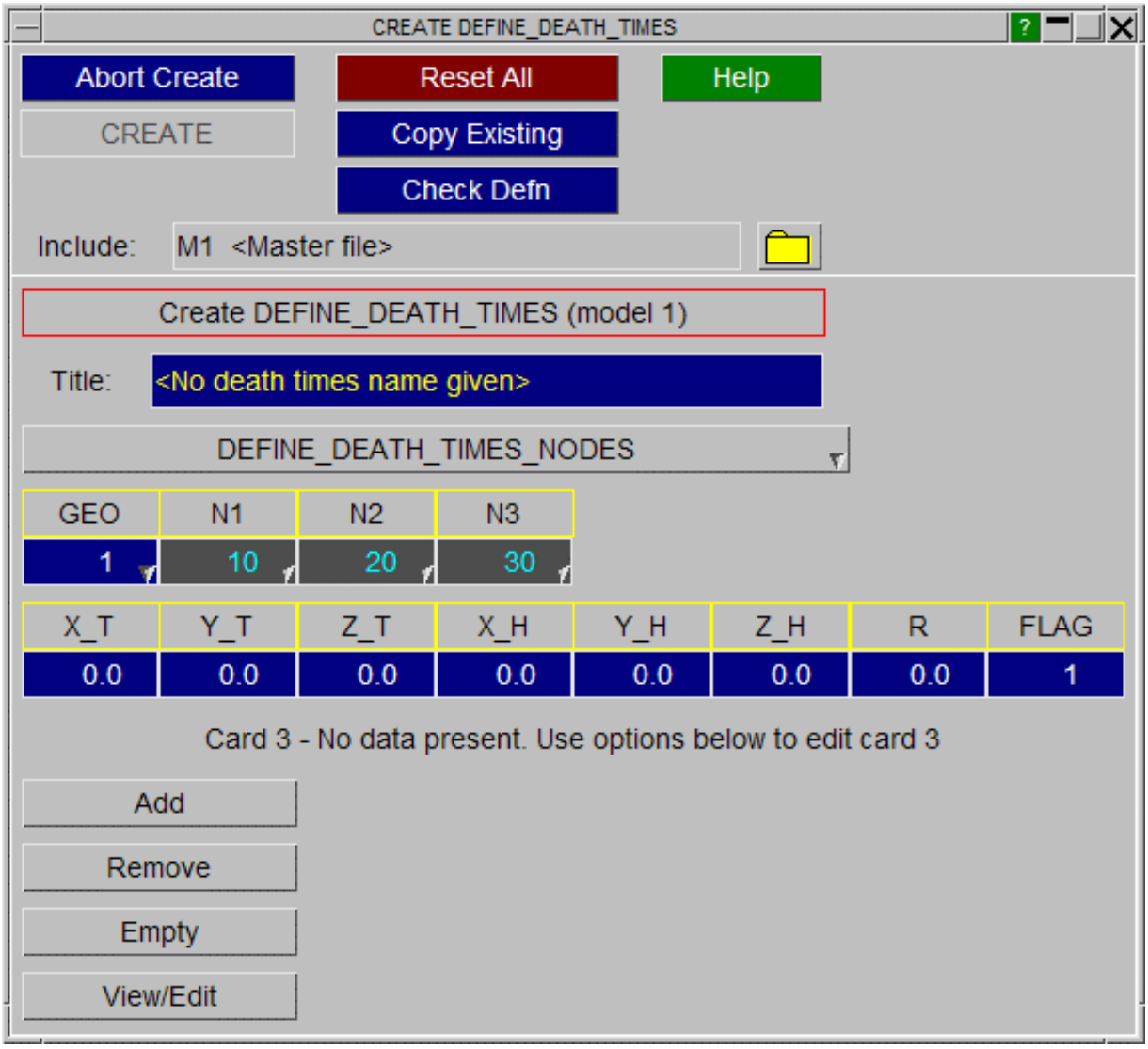
- Main Menu
- Creation
- Copying
- Editing
- Deletion
- List
- Check
- Renumber

This figure shows the main menu for the editing of death times definitions.
All functions have their standard meanings as given in section 5.1.1



CREATE Making a new death times definition.

This shows the create/edit panel for death times. Clicking on the **DEFINE_DEATH_TIMES_...** button will cycle through the _NODES, _SET and _RIGID options. Once the desired option is chosen, the information for card 3 can be added/modified using the **Add**, **Remove**, **Empty** and **View/Edit** buttons at the bottom of the panel.



COPY Copy existing death times(s) to make a new death times(s).

The selected death times are copied. (death times do not "own" anything, so the concept of recursive copying does not apply.)

MODIFY Modifying the attributes of an existing death times.

MODIFY functions in the same way as **CREATE**, except that an initial definition will be present. Any modifications made to the death times definition will not be made permanent until the **UPDATE_ALEBAG_INF** button is pressed. At this point the local copy which has been updated is used to overwrite the version in the model.

DELETE Delete existing death times definitions.

The selected death times are deleted.

Death times do not "own" anything, so the concept of recursive deletion does not apply, however a death times that is referred to (ie "owned") by some higher order item will not be deletable unless that item is deleted too, or its reference to the death times removed.

LIST List death times summaries to screen

The selected death times are summarised on the screen.

CHECK Check death times definitions for errors

The selected death times definitions are run through the standard checking routines.

RENUMBER Change death times labels

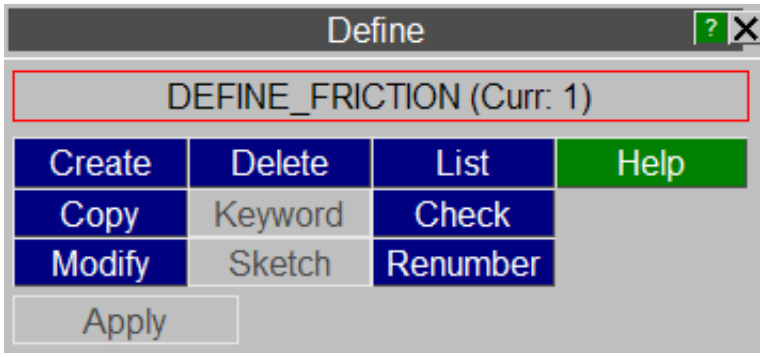
RENUMBER lets you change any or all death times labels within a given model using the standard renumbering panel. To change the label of an individual death times it may be simpler just to **MODIFY** it.

(DEFINE_) FRICTION:

These can be edited through their own specific editing panel (see below).

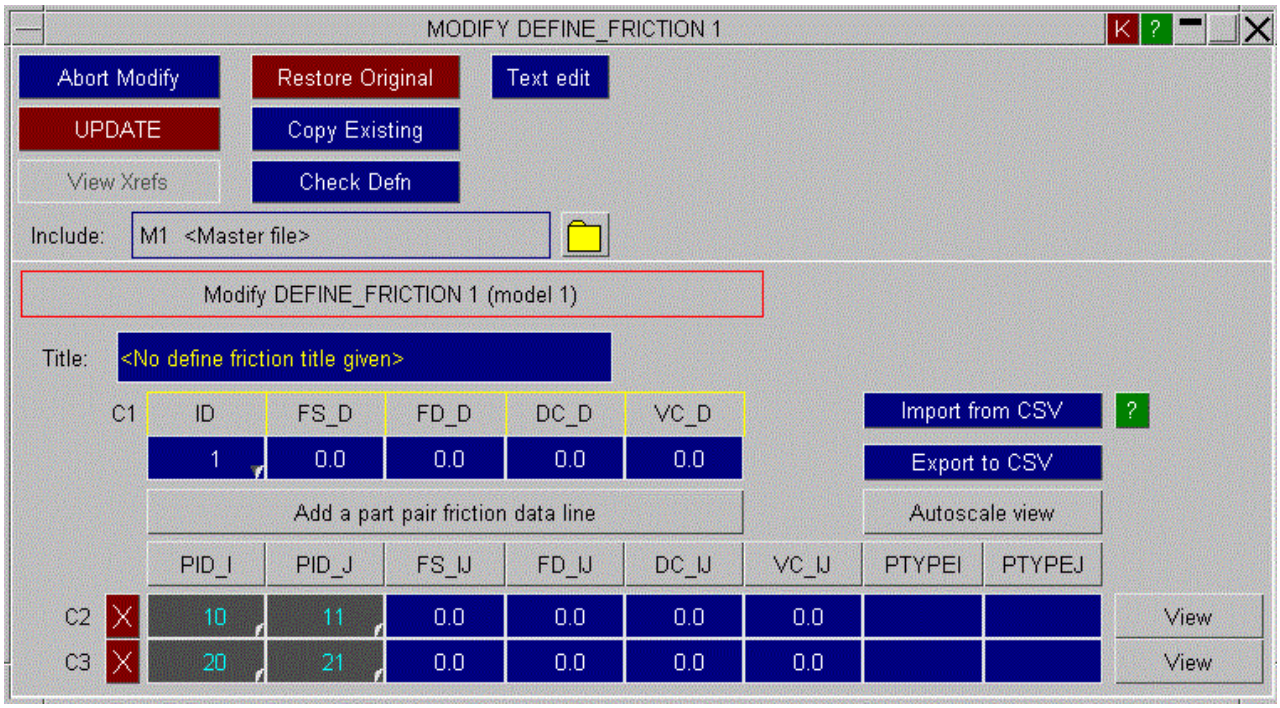
- Main Menu
- Creation
- Copying
- Editing
- Deletion
- List
- Check
- Renumber

This figure shows the main menu for the editing of friction definitions.
All functions have their standard meanings as given in section 5.1.1



CREATE Making a new friction definition.

This shows the create/edit panel for friction. New friction data lines can be added by clicking on the **Add a part pair friction data line** button.



Export to CSV lets you export part pair friction details to a CSV file which can be used later to create multiple part pair friction data lines in the card using **Import from CSV**.

COPY Copy existing define friction(s) to make a new define friction(s).

The selected define frictions are copied. (Define frictions do not "own" anything, so the concept of recursive copying does not apply.)

MODIFY Modifying the attributes of an existing friction.

MODIFY functions in the same way as **CREATE**, except that an initial definition will be present. Any modifications made to the define friction definition will not be made permanent until the **UPDATE** button is pressed. At this point the local copy which has been updated is used to overwrite the version in the model.

DELETE Delete existing define friction definitions.

The selected define frictions are deleted.

Define frictions do not "own" anything, so the concept of recursive deletion does not apply, however a connection property that is referred to (ie "owned") by some higher order item will not be deletable unless that item is deleted too, or its reference to the define friction removed.

LIST List friction summaries to screen

The selected define frictions are summarised on the screen.

CHECK Check define friction definitions for errors

The selected define friction definitions are run through the standard checking routines.

RENUMBER Change define friction labels

RENUMBER lets you change any or all define friction labels within a given model using the standard renumbering panel. To change the label of an individual define friction it may be simpler just to **MODIFY** it.

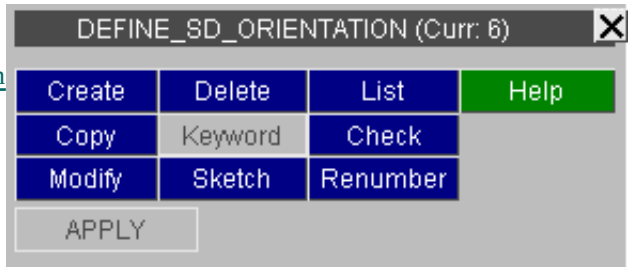
(DEFINE_) SD_ORIENTATION: Defining Spring & Damper Orientation Vectors.

- Main Menu
- Creation
- Copying
- Editing
- Deletion
- Visualisation

The ***DEFINE_SD_ORIENTATION** keyword is used to create orientation vectors for springs and dampers. These are vectors used to define the direction of the element, and are used primarily for rotational elements - which are generally of zero length.
However orientation vectors can also be used for translational springs/dampers of finite length, and they define the direction in which the elements acts, which can be different to the "natural" vector defined by its topology.

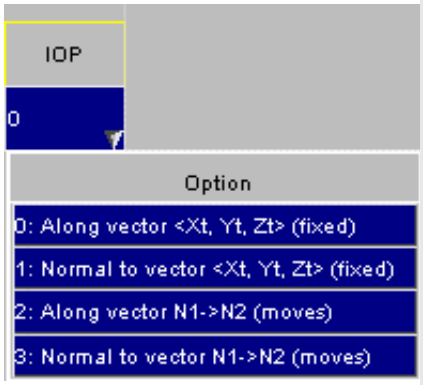
Orientation vectors use unique labels and, although part of the ***DEFINE** keyword, their labels do not clash with other ***DEFINE_xxx** entities. For example it is legal to have **(*DEFINE_)SD_ORIENTATION #1** and **(*DEFINE_)CURVE #1**.

This figure shows the main menu for the editing of orientation vector definitions.
All functions have their standard meanings as given in section 5.1.1

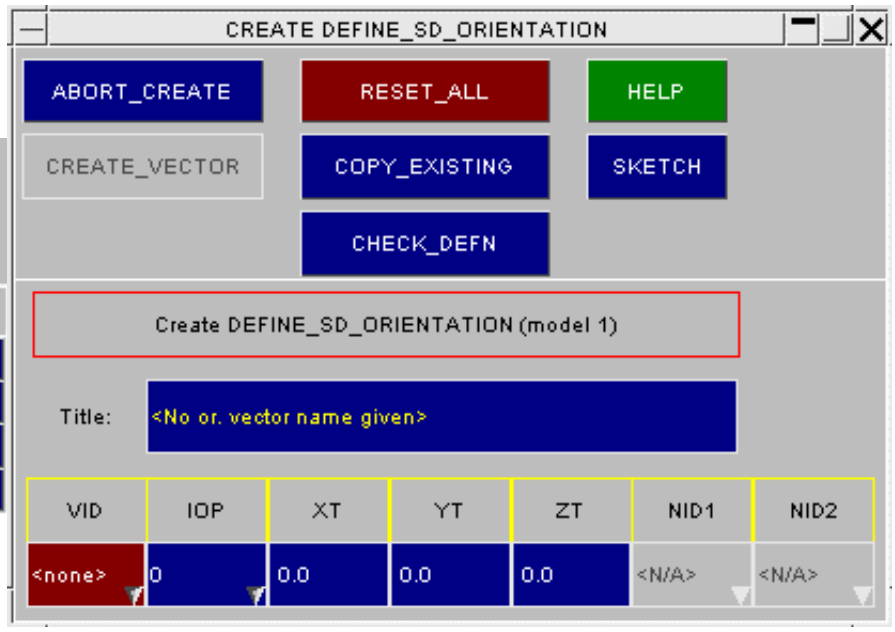


CREATE Making a new orientation vector definition.

This shows the create/edit panel for orientation vectors.
<**IOP**> defines the orientation vector definition method.



Methods #2 and #3 make the **NID1** and **NID2** boxes "live" for selection.



COPY Copy existing orientation vector(s) to make a new vector(s).

The selected orientation vectors are copied. (orientation vectors do not "own" anything, so the concept of recursive copying does not apply.)

MODIFY Modifying the attributes of an existing orientation vector.

MODIFY functions in the same way as **CREATE**, except that an initial definition will be present. Any modifications made to the orientation vector definition will not be made permanent until the **UPDATE_VECTOR** button is pressed. At this point a the local copy which has been updated is used to overwrite the version in the model.

DELETE Delete existing orientation vector definitions.

The selected orientation vectors are deleted.

Orientation vectors do not "own" anything, so the concept of recursive deletion does not apply, however a orientation vector that is referred to (ie "owned") by some higher order item will not be deletable unless that item is deleted too, or its reference to the orientation vector removed.

SKETCH Sketch orientation vector definitions.

SKETCH draws the vector on top of the current graphics image.

LIST List orientation vector summaries to screen

The selected orientation vectors are summarised on the screen.

CHECK Check orientation vector definitions for errors

The selected orientation vector definitions are run through the standard checking routines.

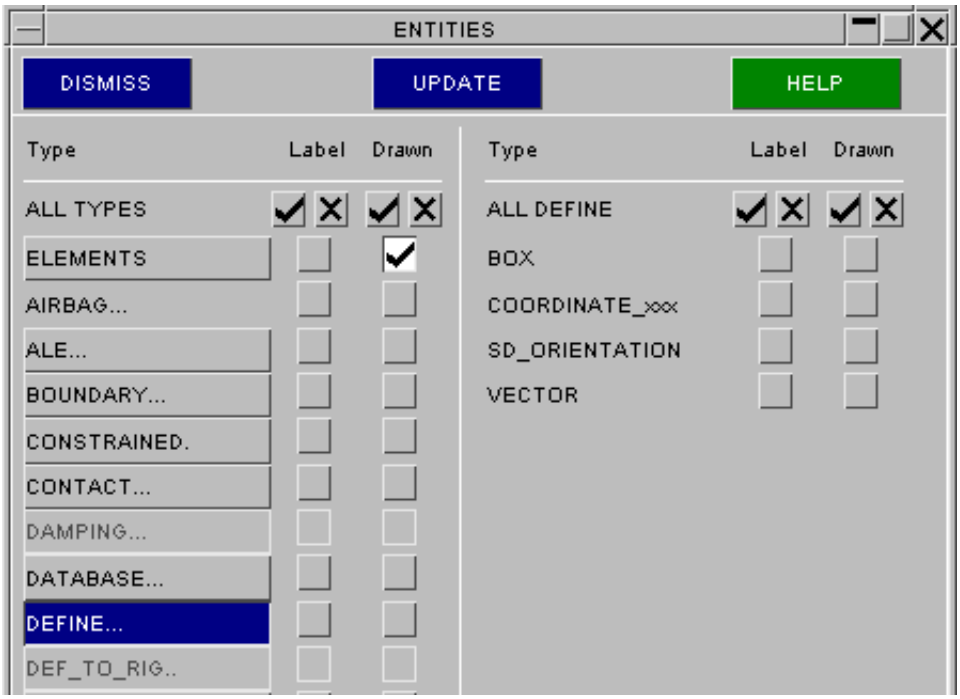
RENUMBER Change orientation vector labels

RENUMBER lets you change any or all orientation vector labels within a given model using the standard renumbering panel. To change the label of an individual orientation vector it may be simpler just to **MODIFY** it.

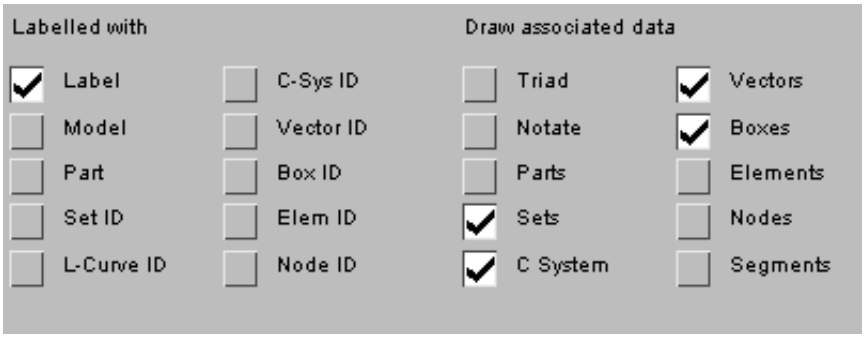
Visualising Orientation Vectors

Orientation Vectors may be drawn by turning their display on in the **ENT**ity Viewing menu.

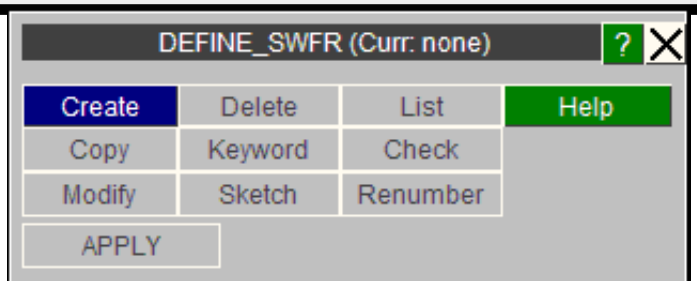
They can also be drawn via the **SKETCH** options above.



They may also be drawn in other contexts (for example contacts) if their display as "associated data" in the **ENT**ity Viewing box is selected.

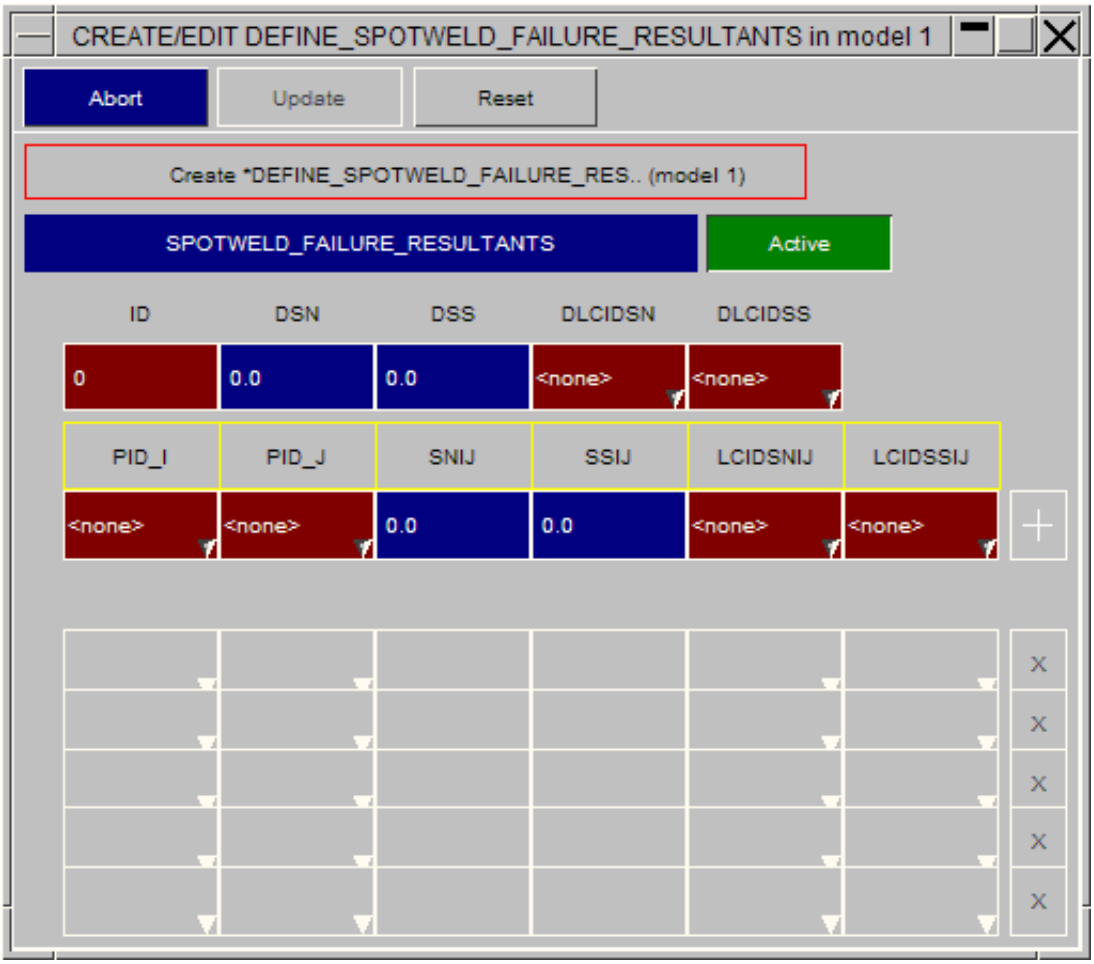


(DEFINE_) SPOTWELD_FAILURE_RESULTANTS:



This can be edited through the create and modify buttons shown above.

There is only one occurance of the DEFINE_SPOTWELD_FAILURE_RESULTANTS in each model. The following panel is used to edit the keyword:



Enter the values in the relevant boxes, and hit the "+" button to add a row, or the "X" button to delete a row.

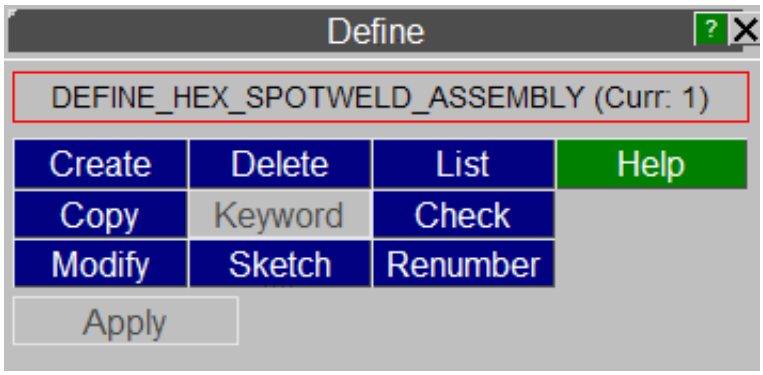
If you wish to deactivate the card from your model, press the green "Active" button to turn it grey (off).

(DEFINE_) HEX_SPOTWELD_ASSEMBLY:

These can be edited through their own specific editing panel (see below).

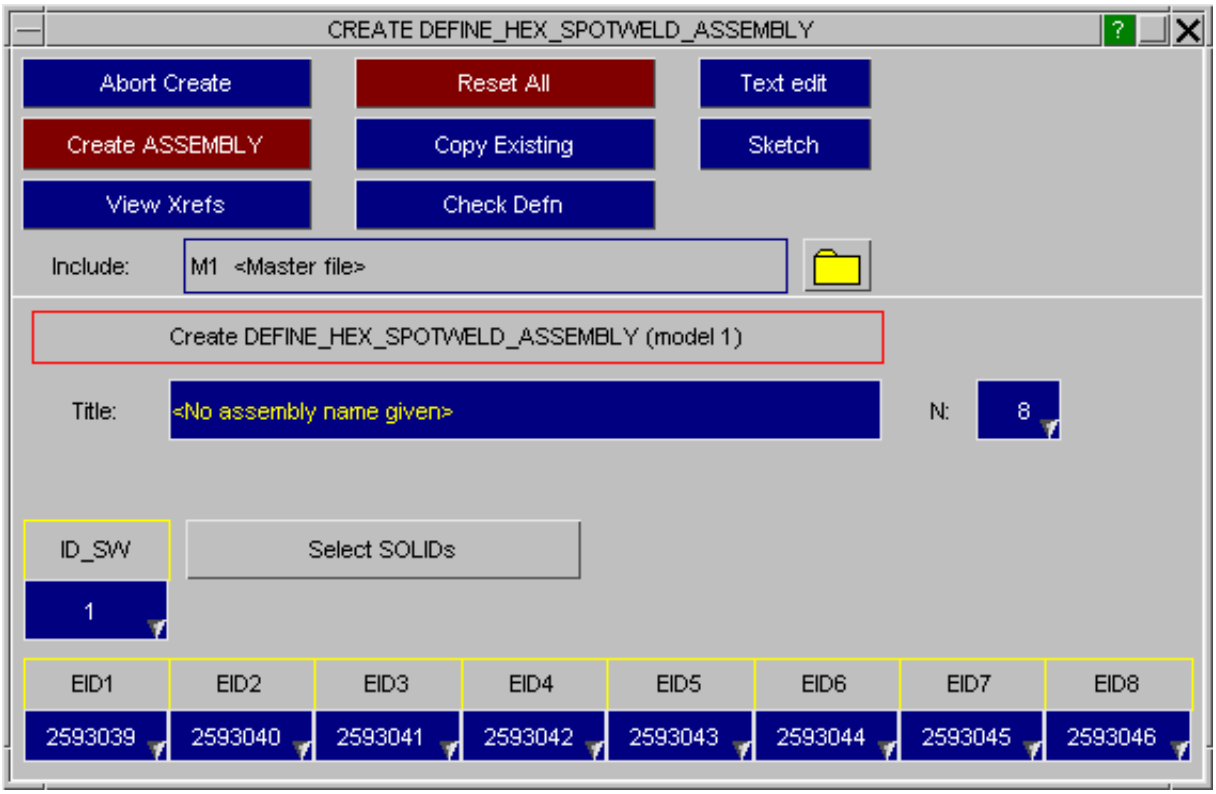
- Main Menu
- Creation
- Copying
- Editing
- Deletion
- Sketching
- List
- Check
- Renumber

This figure shows the main menu for the editing of hex spotweld assembly definitions.
All functions have their standard meanings as given in section 5.1.1



CREATE Making a new hex spotweld assembly definition.

This shows the create/edit panel for hex spotweld assemblies. The second row of the card will change depending on the value chosen for **'N'. 'N'** can be set to 4, 8 or 16 from a drop down list for that button.



The **Select SOLIDs** button can be used to select multiple solids, which will then be added to the list in ascending order.

COPY Copy existing hex spotweld assembly(s) to make a new hex spotweld assembly(s).

The selected hex spotweld assemblies are copied. (hex spotweld assemblies do not "own" anything, so the concept of recursive copying does not apply.)

MODIFY Modifying the attributes of an existing hex spotweld assembly.

MODIFY functions in the same way as **CREATE**, except that an initial definition will be present. Any modifications made to the hex spotweld assembly definition will not be made permanent until the **UPDATE** button is pressed. At this point the local copy which has been updated is used to overwrite the version in the model.

DELETE Delete existing hex spotweld assembly definitions.

The selected hex spotweld assemblies are deleted.

Hex spotweld assemblies do not "own" anything, so the concept of recursive deletion does not apply, however a hex spotweld assembly that is referred to (ie "owned") by some higher order item will not be deletable unless that item is deleted too, or its reference to the hex spotweld assembly removed.

SKETCH Sketch hex spotweld assembly definitions.

SKETCH draws the hex spotweld assembly on top of the current graphics image.

LIST List hex spotweld assembly summaries to screen

The selected hex spotweld assemblies are summarised on the screen.

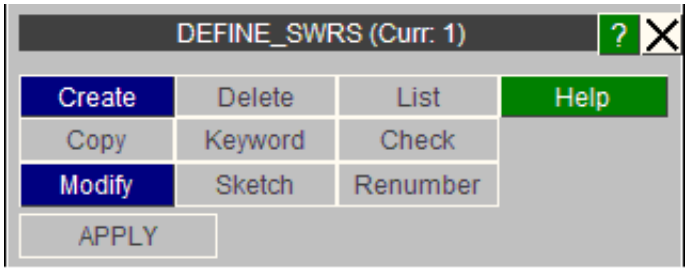
CHECK Check hex spotweld assembly definitions for errors

The selected hex spotweld assembly definitions are run through the standard checking routines.

RENUMBER Change hex spotweld assembly labels

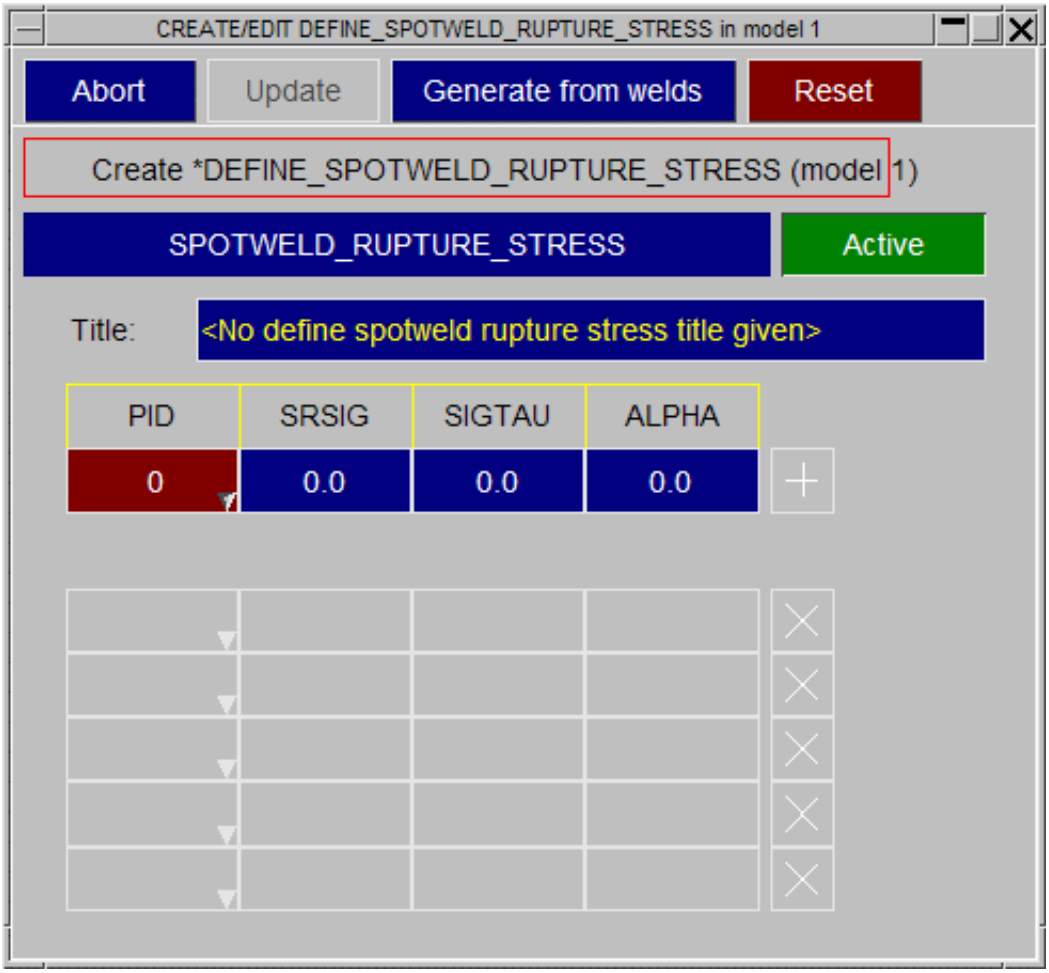
RENUMBER lets you change any or all hex spotweld assembly labels within a given model using the standard renumbering panel. To change the label of an individual hex spotweld assembly it may be simpler just to **MODIFY** it.

(DEFINE_) SPOTWELD_RUPTURE_STRESS:

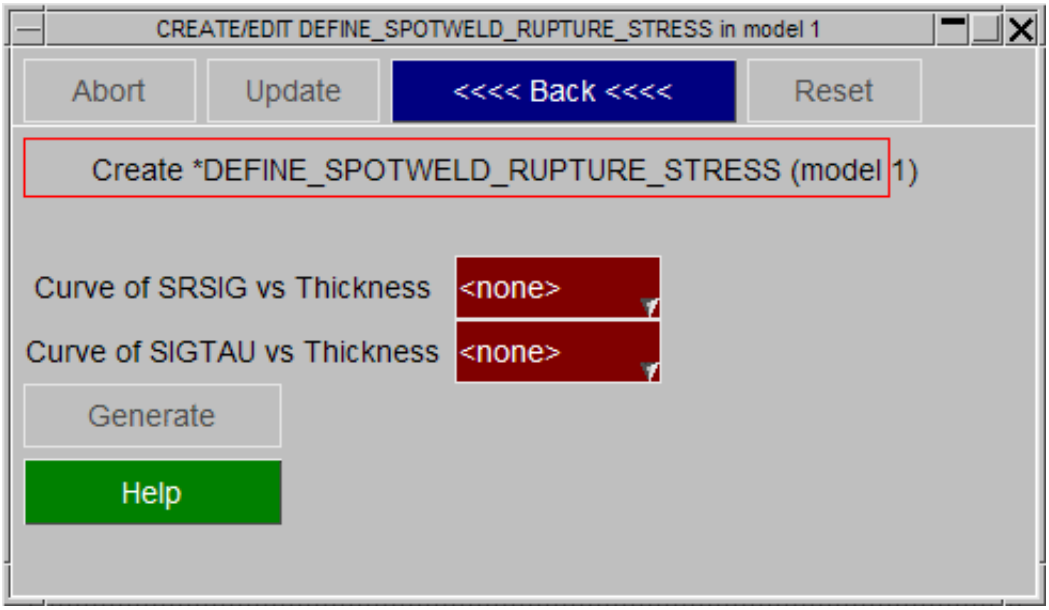


These types can be edited through the **create** and **modify** buttons shown above. There is only one occurance of the keyword in each model.

To deactivate the keyword from your model, click on the green **Active** button. Enter the relevant details in the card fields, then click on the **+** button to add a row, or the **X** button to remove a row.



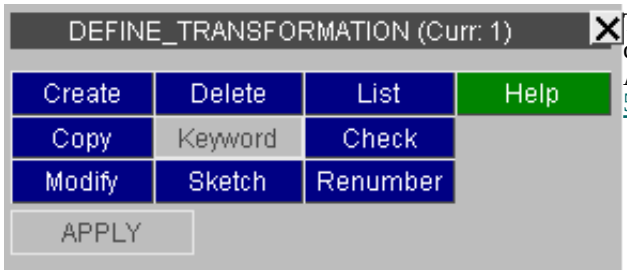
PRIMER has the ability to generate the rupture cards automatically. Click on the **Generate from welds** button and the following menu appears:



Enter a curve defining SRSIG vs thickness, and one defining SIGTAU vs thickness. PRIMER then looks at the thickness of each beam spotweld part in the model and enters the correct values in the keyword automatically.

(DEFINE_) TRANSFORM:

- Main Menu
- Creation
- Copying
- Editing
- Deletion
- List
- Check
- Renumber



This figure shows the main menu for the editing of co-ordinate systems.
All functions have their standard meanings as given in section 5.1.1:

CREATE Making a transformation definition.

See section 3.14.3. for more details about the **Create** /**Modify** panel for Transformations.

COPY Copy existing transformation(s) to make a new transformation.

The selected coordinates are copied. (Transformations do not "own" anything, so the concept of recursive copying does not apply.)

MODIFY Modifying the attributes of an existing transformation.

MODIFY functions in the same way as **CREATE**, except that an initial definition will be present.

Any modifications made to the section definition will not be made permanent until the **UPDATE_TRANSFORMATION** button is pressed. At this point a the local copy which has been updated is used to overwrite the version in the model.

DELETE Delete existing transformation definitions.

The selected transformations are deleted.

Transformation definitions do not "own" anything, so the concept of recursive deletion does not apply, however a transformation that is referred to (ie "owned") by some higher order item will not be deletable unless that item is deleted too, or its reference to the transformation removed.

SKETCH Sketch transformation summaries

The selected transformation definitions are sketched on the screen.

LIST List transformation summaries to screen

The selected transformation definitions are summarised on the screen.

CHECK Check definitions for errors

The selected transformation definitions are run through the standard checking routines.

RENUMBER Change transformation labels

Lets you change any or all transformation labels within a given model using the standard renumbering panel.

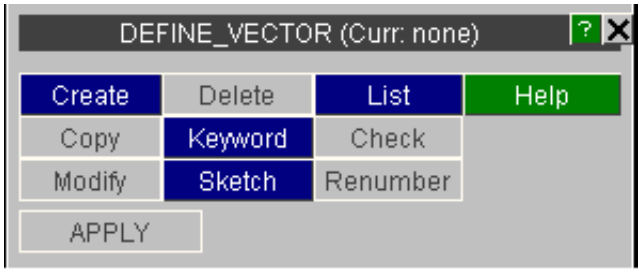
To change the label of an individual transformation may be simpler just to **MODIFY** it.

(DEFINE_) VECTOR:

These can be edited through their own specific editing panel (see below) and using the generic Keyword Editor.

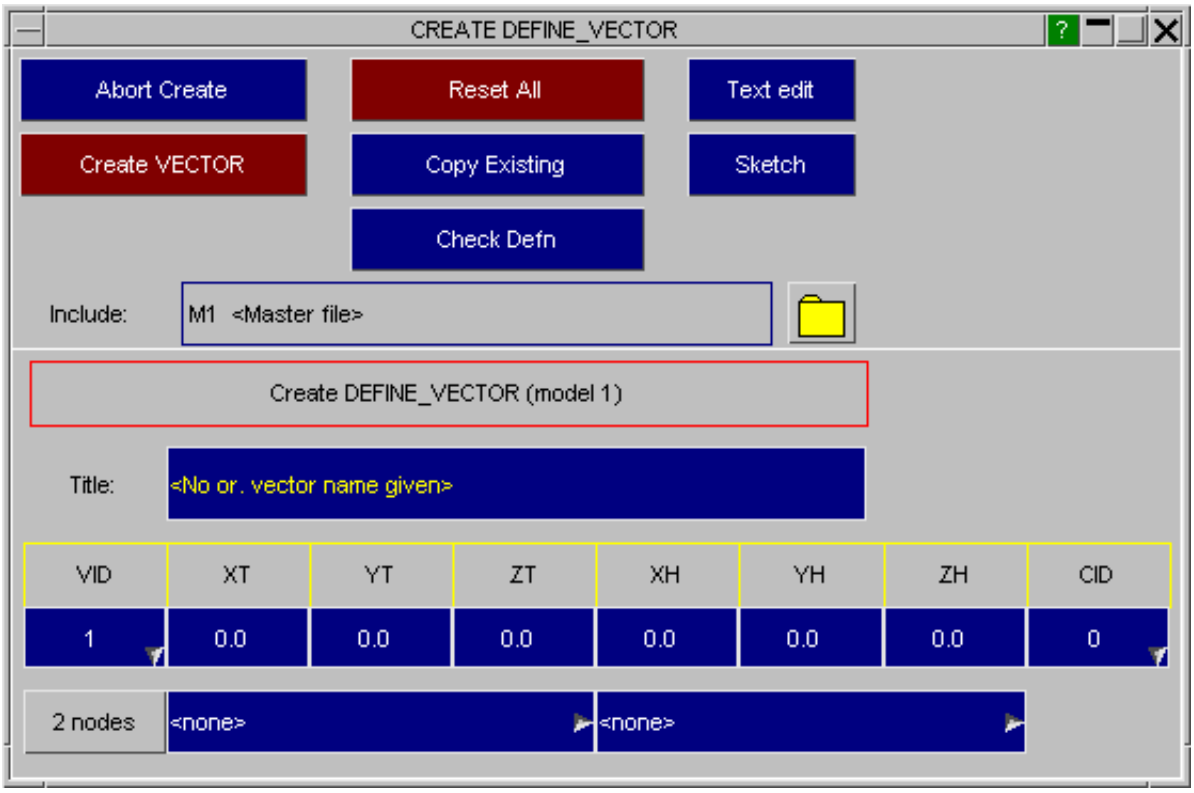
- Main Menu
- Creation
- Copying
- Editing
- Deletion

This figure shows the main menu for the editing of vector definitions.
All functions have their standard meanings as given in section 5.1.1



CREATE Making a new vector definition.

This shows the create/edit panel for vectors.



COPY Copy existing vector(s) to make a new vector(s).

The selected vectors are copied. (vectors do not "own" anything, so the concept of recursive copying does not apply.)

MODIFY Modifying the attributes of an existing vector.

MODIFY functions in the same way as **CREATE**, except that an initial definition will be present. Any modifications made to the vector definition will not be made permanent until the **UPDATE_VECTOR** button is pressed. At this point the local copy which has been updated is used to overwrite the version in the model.

DELETE Delete existing vector definitions.

The selected vectors are deleted.

Vectors do not "own" anything, so the concept of recursive deletion does not apply, however a vector that is referred to (ie "owned") by some higher order item will not be deletable unless that item is deleted too, or its reference to the vector removed.

SKETCH Sketch vector definitions.

SKETCH draws the vector on top of the current graphics image.

LIST List vector summaries to screen

The selected vectors are summarised on the screen.

CHECK Check orientation vector definitions for errors

The selected vector definitions are run through the standard checking routines.

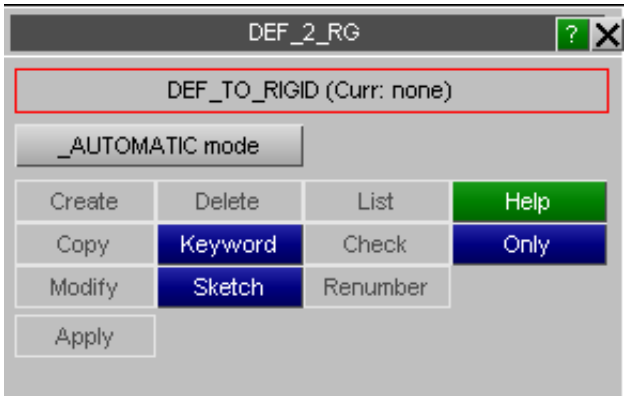
RENUMBER Change orientation vector labels

RENUMBER lets you change any or all vector labels within a given model using the standard renumbering panel. To change the label of an individual vector it may be simpler just to **MODIFY** it.

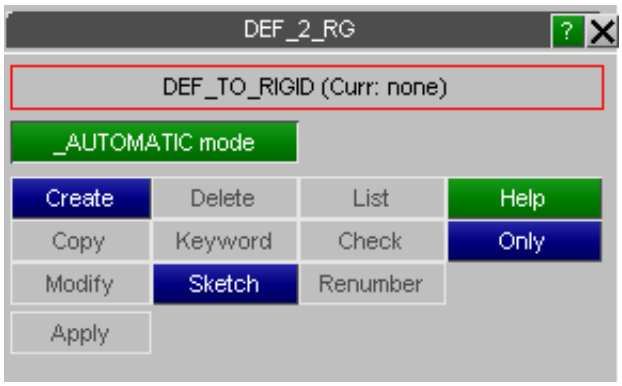
DEFORMABLE_TO_RIGID: Switching parts.

*DEFORMABLE_TO_RIGID and DEFORMABLE_TO_RIGID_INERTIA
DEFORMABLE_TO_RIGID_AUTOMATIC

This submenu panel is slightly different to the standard panels in PRIMER because there are 2 separate editing panels for DEFORMABLE_TO_RIGID - one for _AUTOMATIC, and one for _NONE and _INERTIA. To get to the _NONE and _INERTIA use the **keyword** button.

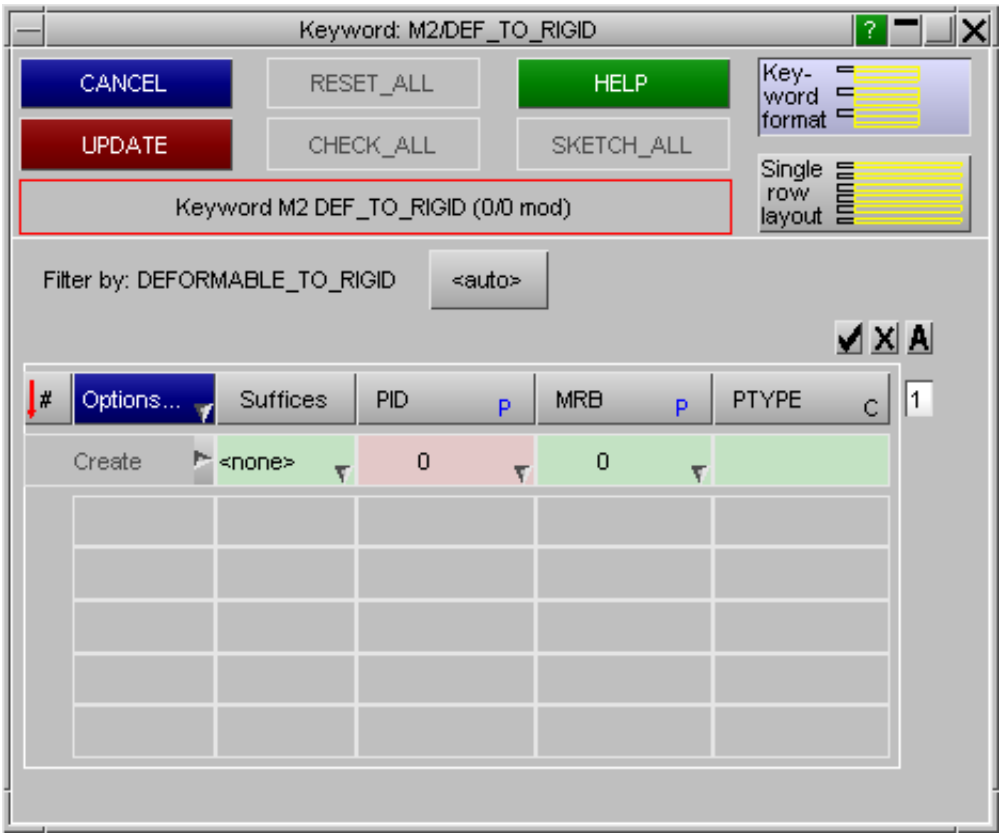


To get to the _AUTOMATIC case, press the **_AUTOMATIC mode** button and then use **Create** or **Modify** buttons as required.

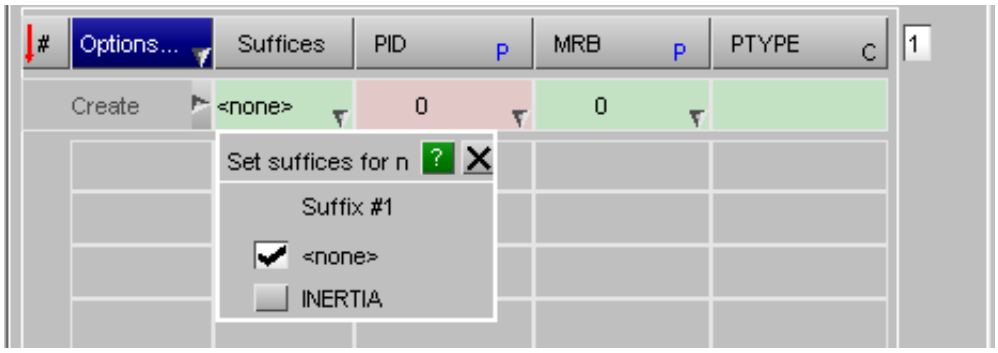


*DEFORMABLE_TO_RIGID and _INERTIA

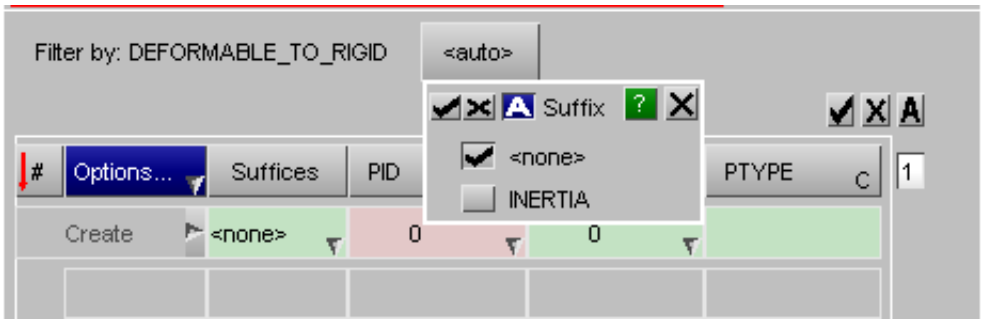
This is a standard keyword panel, described in section_5a. Use the popups from the auto button on top or from suffixes button to toggle between the two types.



Popup from suffix



Popup from auto button

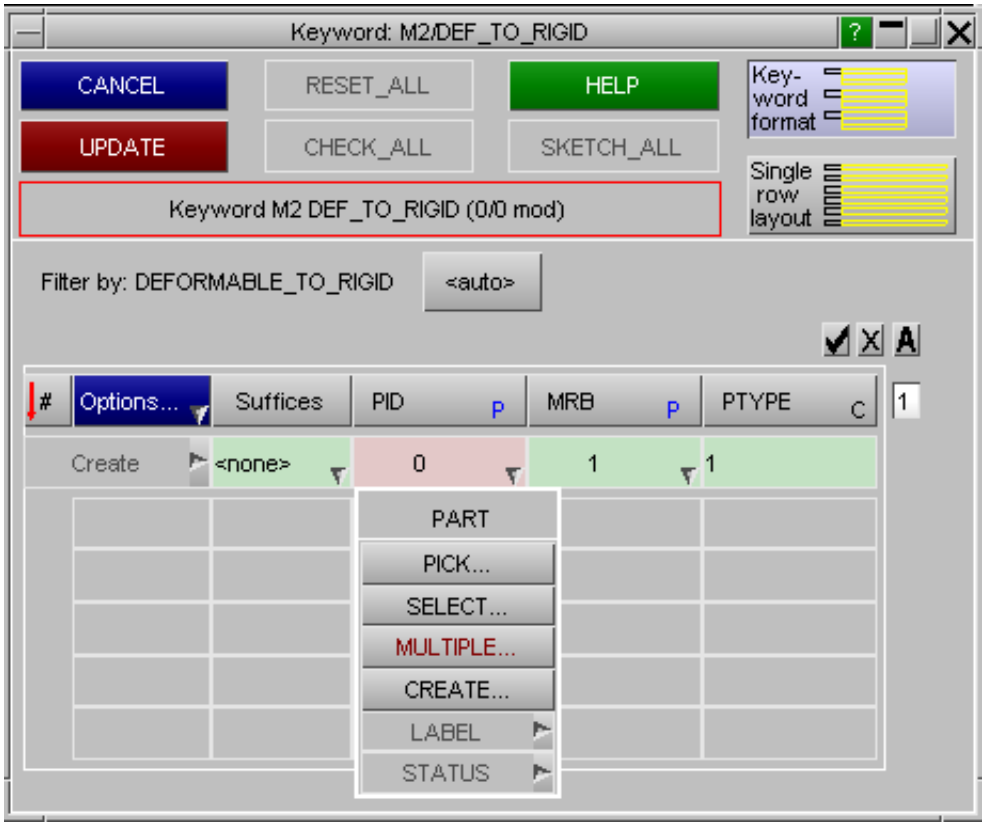


Creating multiple *DEFORMABLE_TO_RIGID and _INERTIA cards

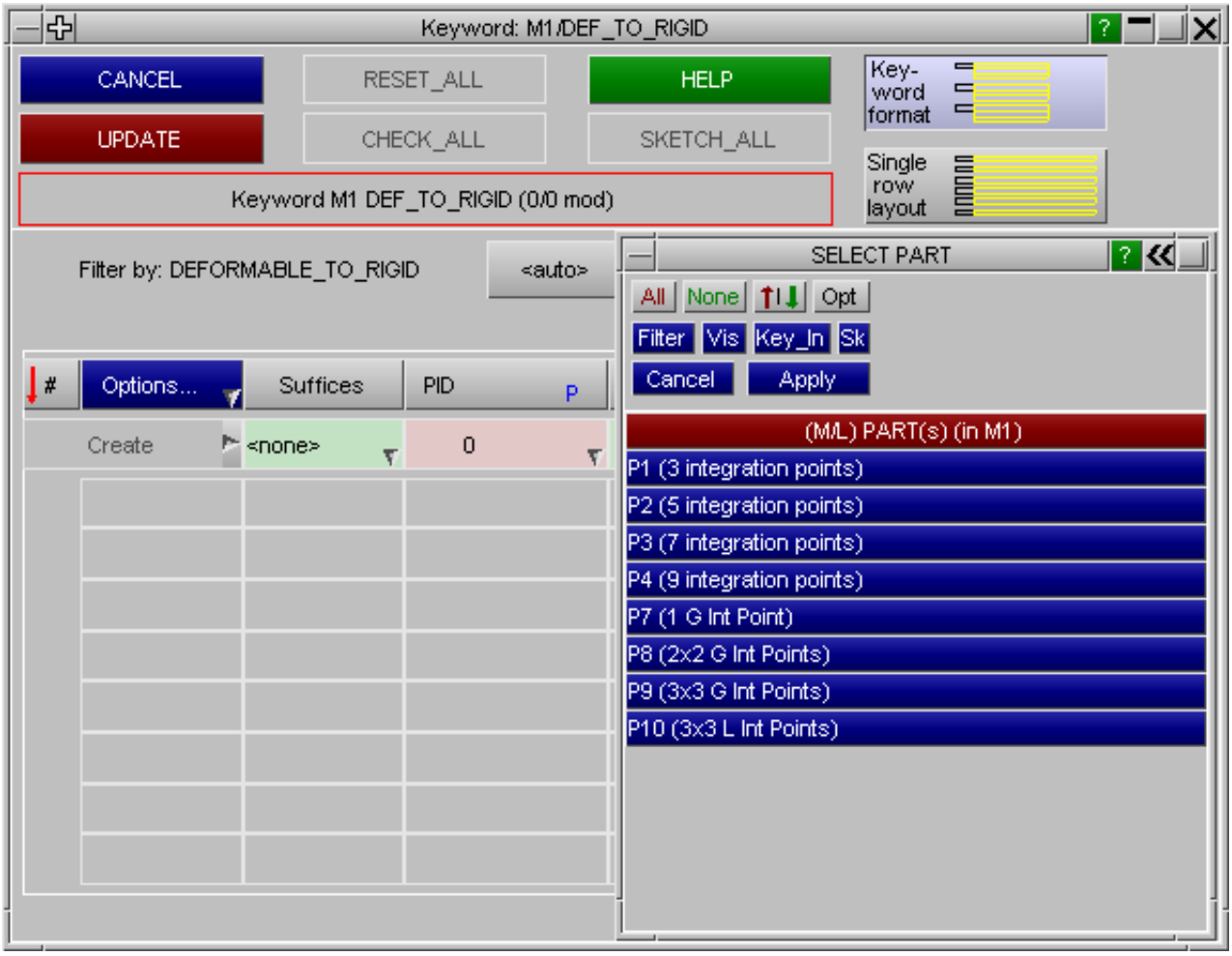
In addition to the standard creation/modification of these standard keywords, we use "Multiple" option to create cards with different "PID", keeping the remaining fields same.

Example:

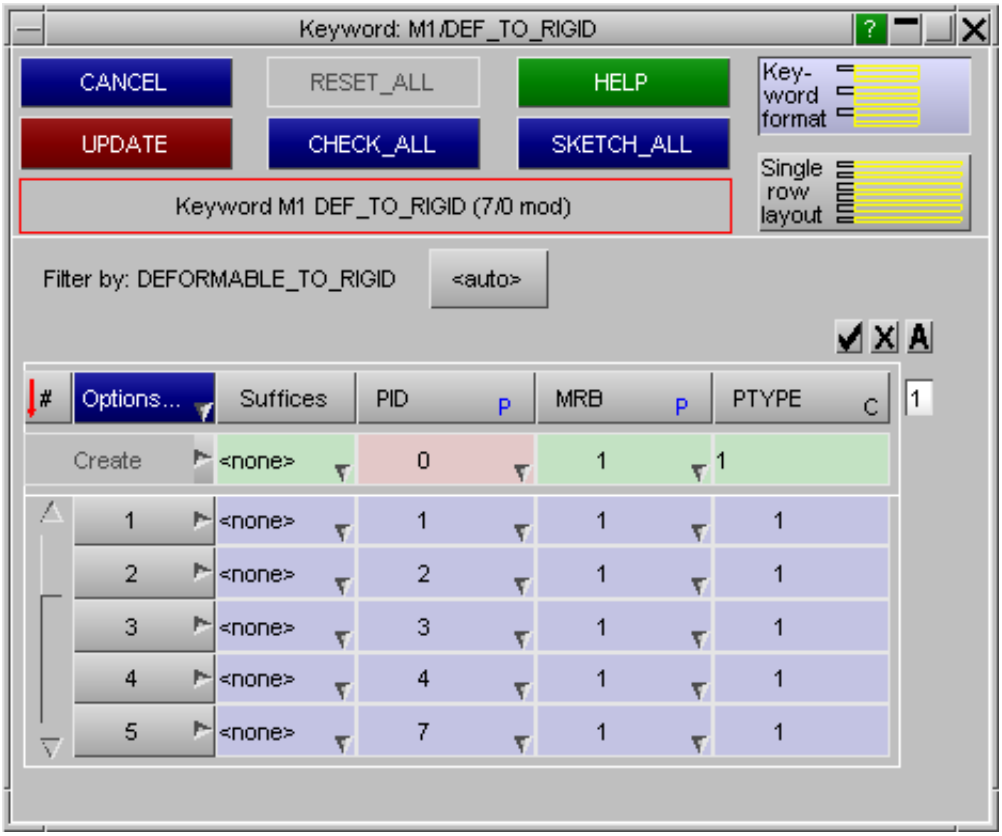
Fill all the fields expect PID with the required data. Right-click on the PID field and select MULTIPLE option from the popup.



Next, from the object menu select the required parts and hit apply

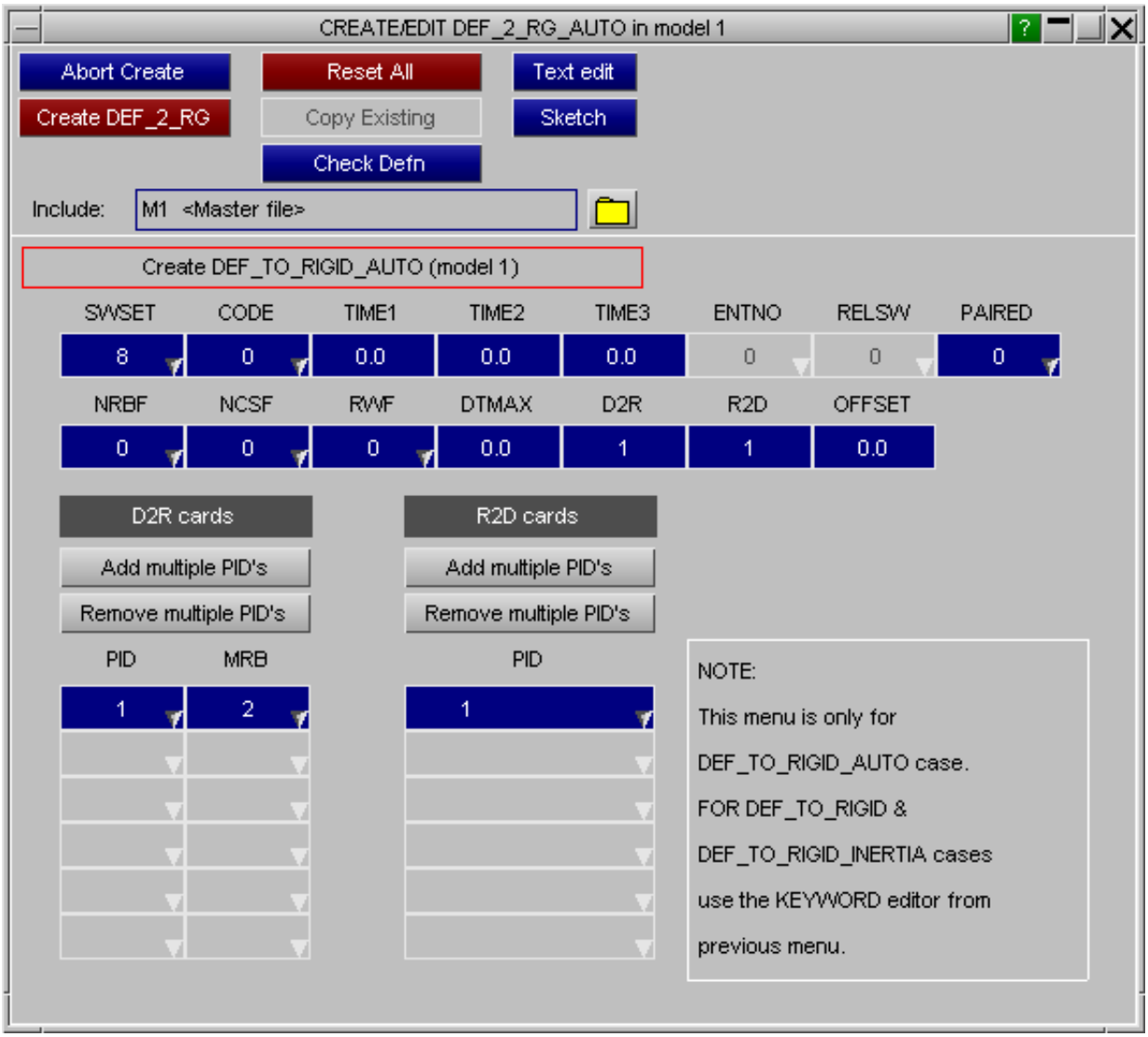


Multiple cards will be created as shown in the following figure.


Keyword: M1/DEF_TO_RIGID
CANCEL
RESET_ALL
HELP
UPDATE
CHECK_ALL
SKETCH_ALL
Key-word format
Single row layout
Keyword M1 DEF_TO_RIGID (7/0 mod)
Filter by: DEFORMABLE_TO_RIGID
<auto>
Options... Suffices PID P MRB P PTYPE C
1
Create <none> 0 1 1
1 <none> 1 1 1
2 <none> 2 1 1
3 <none> 3 1 1
4 <none> 4 1 1
5 <none> 7 1 1

*DEFORMABLE_TO_RIGID_AUTOMATIC

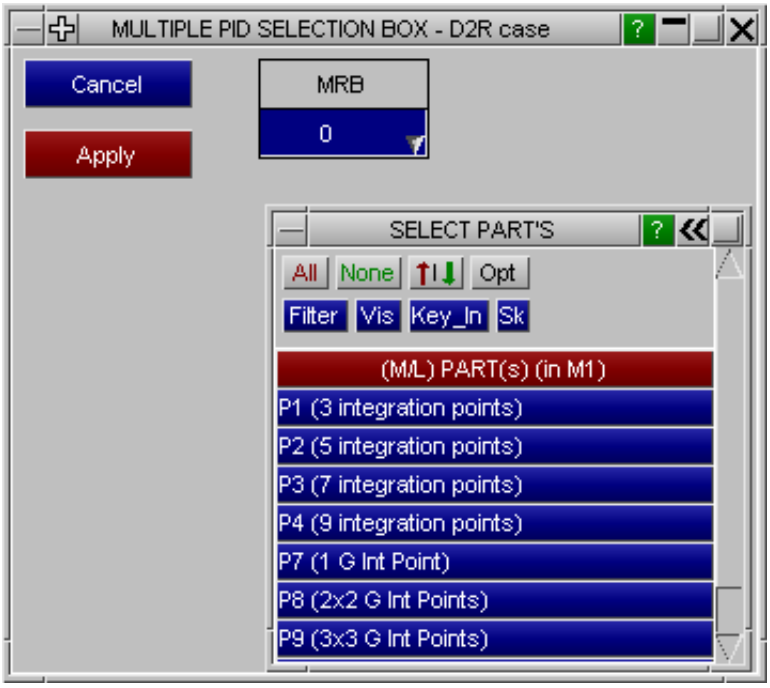
The _AUTOMATIC case has its own menu because it doesn't fit into the standard keyword layout. Define the number of D2R and R2D conversions, and the rows beneath will become active, allowing the user to enter part ids.



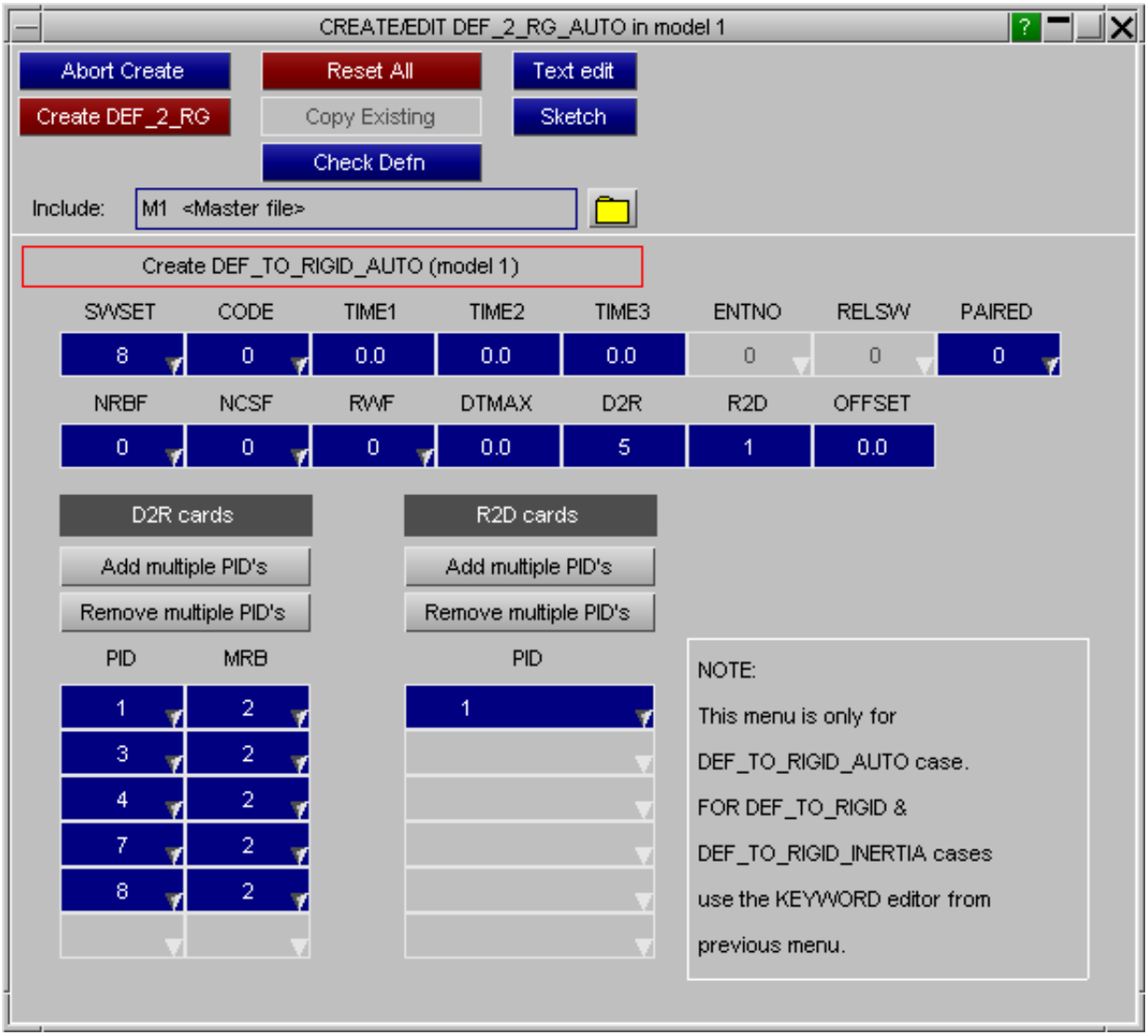
Adding/Removing multiple PID's on *DEFORMABLE_TO_RIGID_AUTOMATIC cards

Adding multiple PIDs to D2R cards

To add multiple PID's, press the **"Add multiple PID's"** button under D2R cards. A new panel appears with an object menu and "MRB" field as shown below.

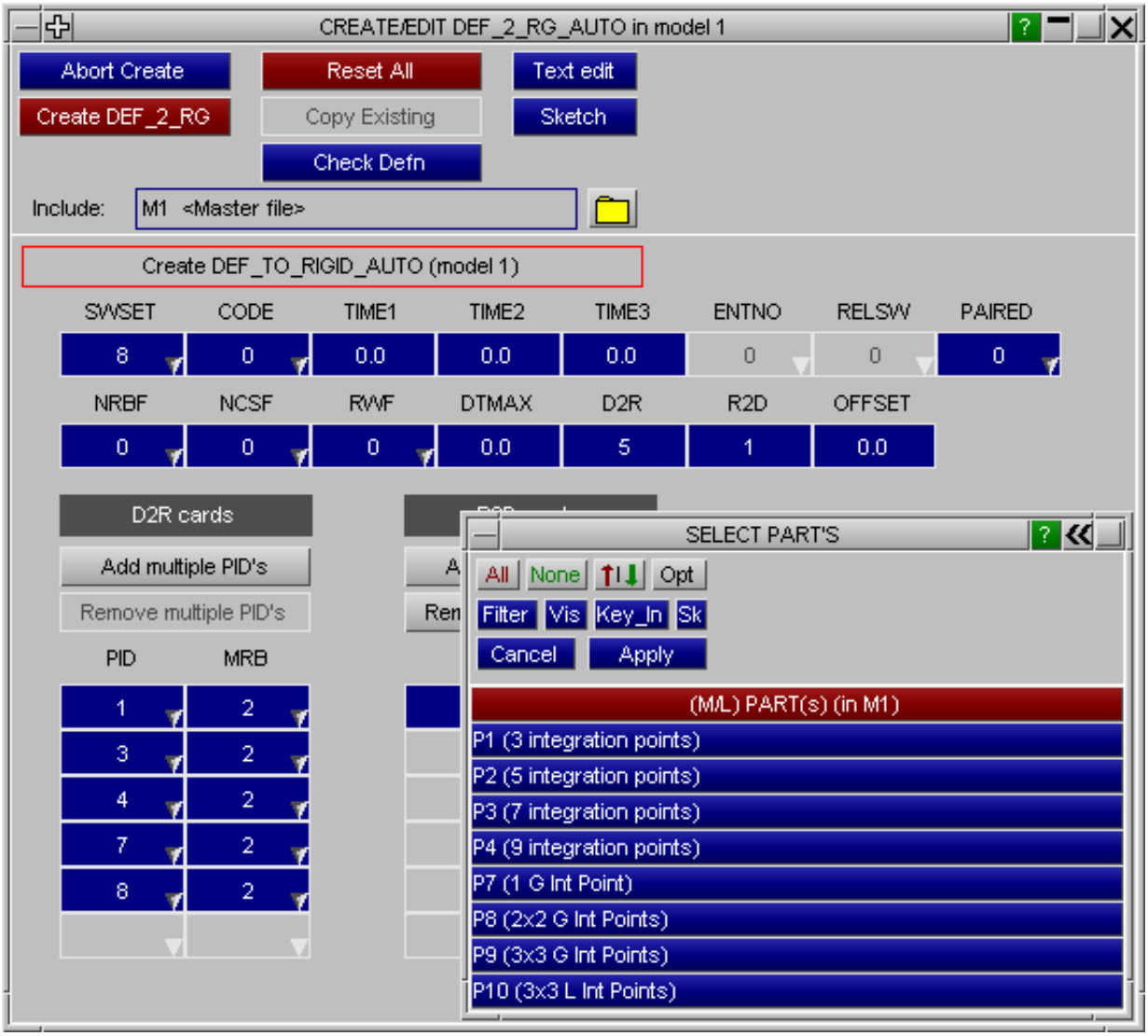


Hit **"Apply"** button after selecting the required parts and MRB. Multiple D2R cards will be created as shown in the following figure.



Removing multiple PIDs on D2R cards

To remove multiple PID's, press the **"Remove multiple PID's"** button under D2R cards. An object menu appears as shown in the following figure. Hit **"Apply"** button after selecting the required parts and the corresponding PID's will be removed form the list.

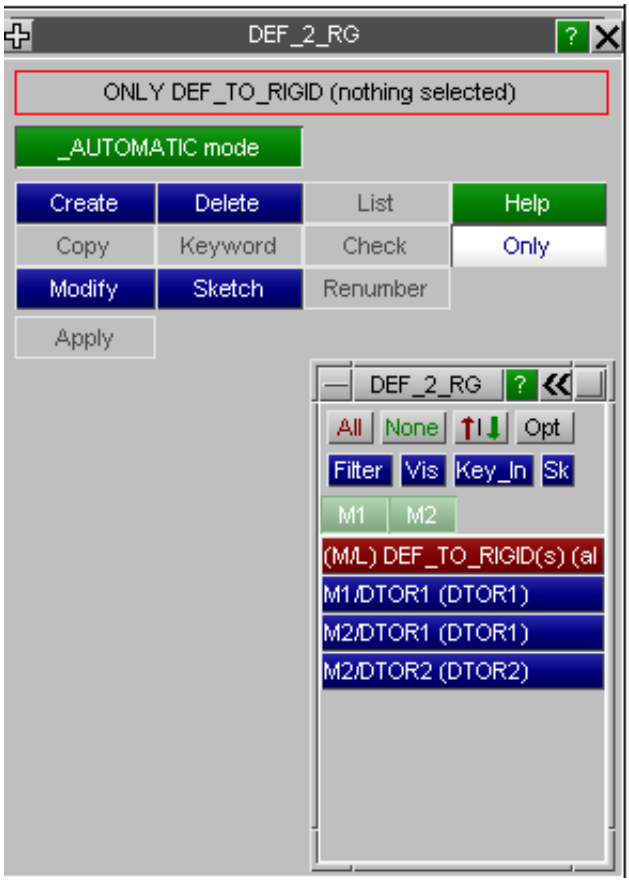


Adding/Removing multiple PIDs on R2D cards

To add/remove multiple PID's, press the **"Add multiple PID's"** or **"Remove multiple PID's"** button under R2D cards respectively (as in D2R case). Hit **"Apply"** button after selecting the required parts from the object menu and the corresponding PID's will be Added/Removed form the list.

ONLY OPTION

The **"ONLY"** button on the submenu panel is use to view/display only the selected DEF_2_RIGID cards. Select the required card(s) from the object menu and hit **"APPLY"** button to view them.



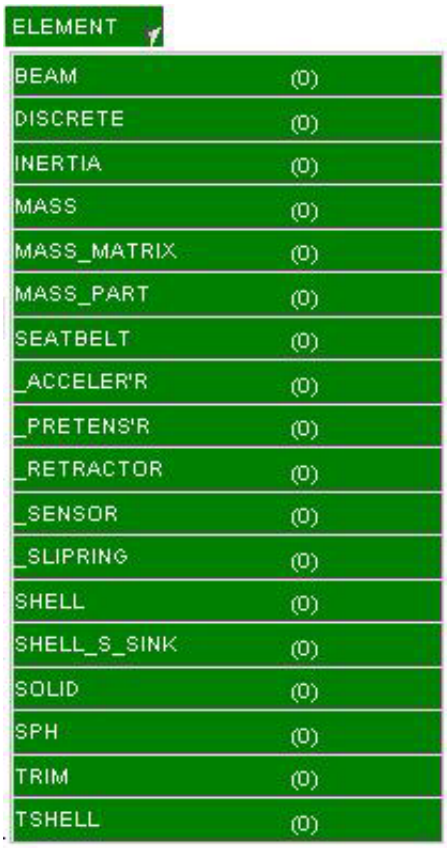
ELEMENT: Defining Structural Elements.

- *ELEMENT_<type>
- Visualisation
- Setting Colour
- Data display
- Fitting seatbelts

LS-Dyna has 15 classes of structural element types, all of which are fully editable in PRIMER in individual create/edit panels. Generic keyword editing of all element types is also provided.
All element types except *TRIM are fully drawable, and there are a range of contouring options for different types of element data.

Each class of element has its own independent label sequence, thus it is legal to have shell #1 and solid #1, etc in the same model.

The elements menu enables you to create, modify and delete all the element types that are available in LS-Dyna.
This figure shows the main element menu.



The ***ELEMENT** keyword in LS-DYNA supports the following sub-types of structural element:

- BEAM
- DISCRETE
- INERTIA
- MASS
- MASS_MATRIX
- **MASS_PART**
- SHELL
- SHELL_SOURCE_SINK
- SOLID
- SPH
- TSHELL
- TRIM
- SEATBELT
- SEATBELT_ACCELEROMETER
- SEATBELT_PRETENSIONER
- SEATBELT_RETRACTOR
- SEATBELT_SENSOR
- SEATBELT_SLIPRING

All of the element types with the exception of ***ELEMENT_TRIM** can be created by quickly picking the nodes from the screen and setting the extra data. For all element types, when an element is created, default settings are saved so that the next element will use the same defaults. For example if you create a shell and select part 1000 for the PID, then the next element you create will automatically have the PID set to 1000. Obviously, you can change the part if needed but hopefully this will speed up creation of lots of elements in the same part.

A ‘quick create’ option is also available. Once the necessary information needed for the element is defined the element will automatically be created.

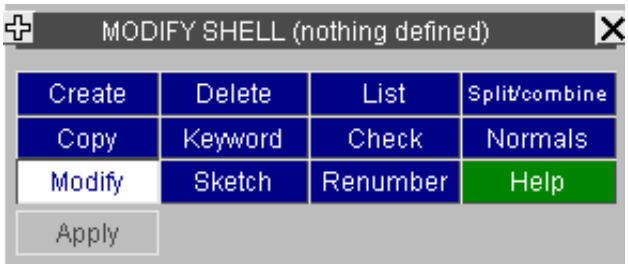
When an element is created the panel automatically remaps itself with the default values.

As the method for creating the different element types is very similar the generic method will be described in detail for shells. Any major differences in other element types will be stated

ELEMENT_SHELL

This figure shows the main element shell menu. The functions currently available have their standard meanings. (See section 5.1.1)

As with all classes of element the Generic Keyword editor may be used instead.



CREATE Making a new shell.

This figure shows the initial state of the element shell creation panel: no part has been given yet, so it is highlighted red.

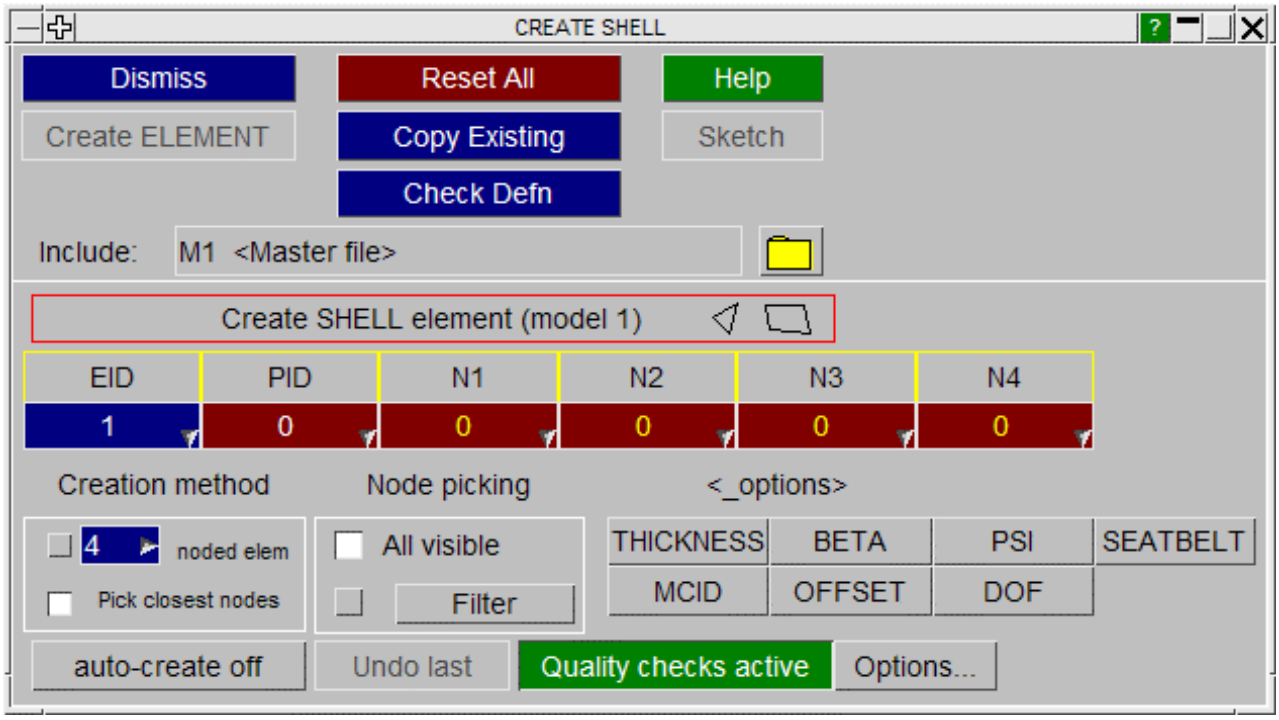
The **<_option>** buttons can be used to select among sub-keyword suffices:

ELEMENT_SHELL
ELEMENT_SHELL_THICKNESS or
ELEMENT_SHELL_BETA
etc

The **SEATBELT** option is a special case: see below.

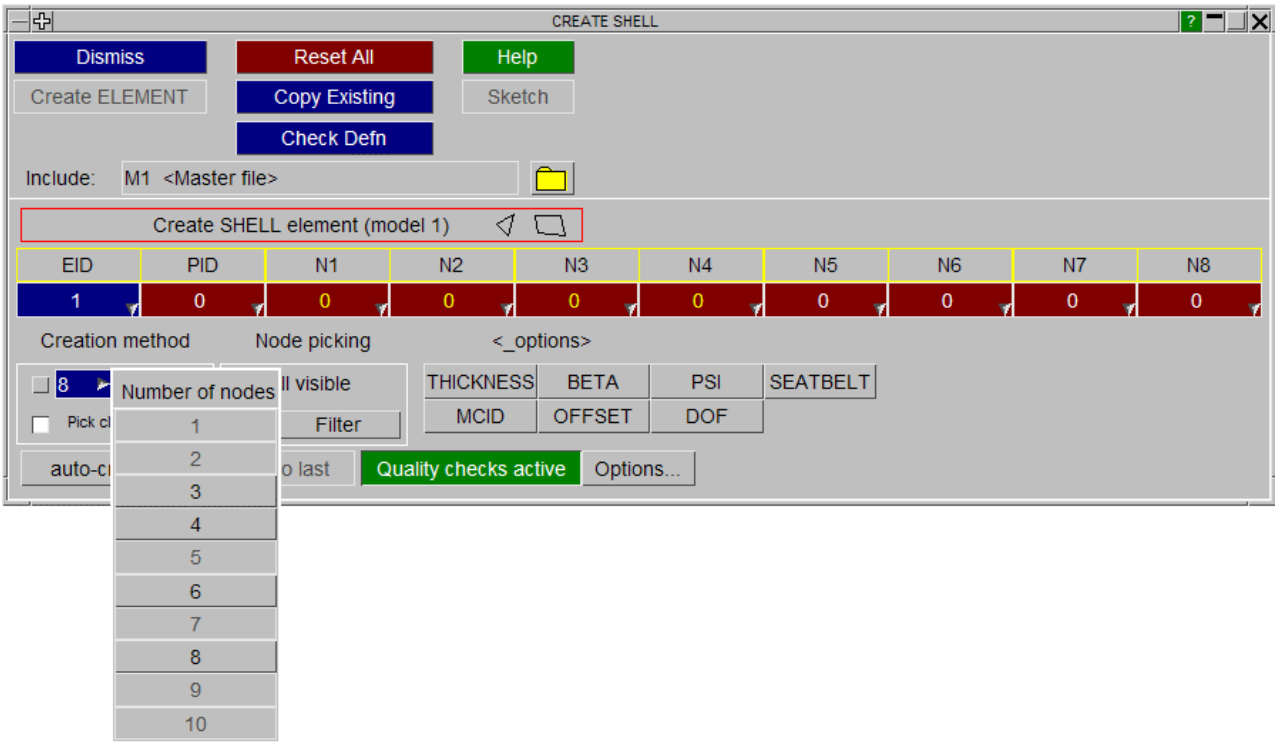
The part and the node numbers can be typed directly into the text boxes. The default element label used is the highest node label in the model + 1. This can be changed if needed. Alternatively, the popup menus can be used to pick a part, and the nodes from the screen, or to select a part, or node from a list.

To choose creation of tria or quad elements the nodes popup can be used to select the number of nodes. For a shell only 3 or 4 nodes can be chosen.



6 and 8 noded (parabolic) shell elements

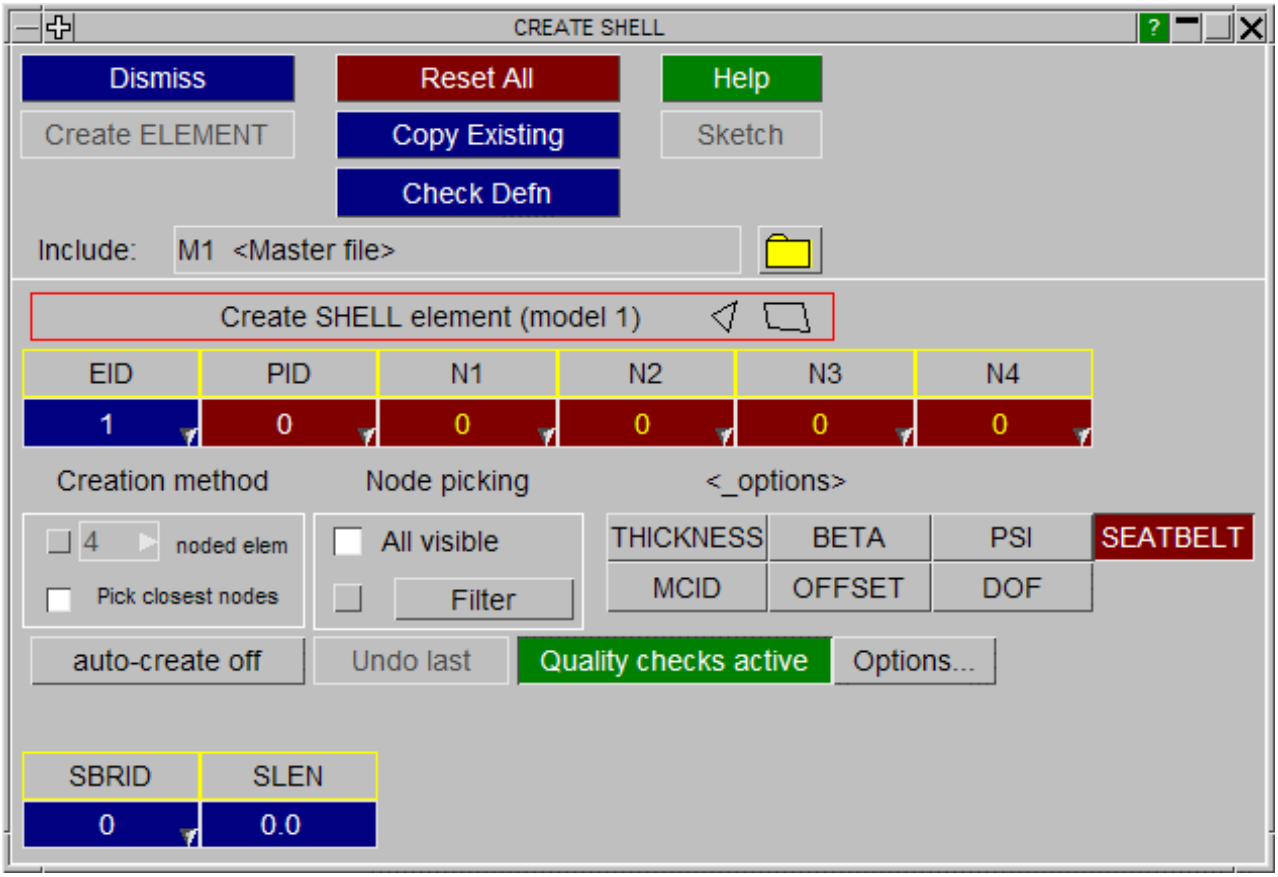
To edit parabolic shells used the "Pick >" popup to select the number of nodes on the element. The example below shows the 8 noded case:



The **SEATBELT** option: creating 4 noded seatbelt shells

In LS-DYNA release 971 4 noded "shell" seatbelt elements are introduced under the *ELEMENT_SEATBELT keyword. However these are in fact shell elements, sharing the same numbering scheme as shells, so for consistency within PRIMER they are edited under the SHELL keyword.

These elements should belong to shell parts, referencing *SECTION_SHELL cards, however they can use the *MAT_SEATBELT material definition.



The quickest method for creating a shell is to use one of the ‘quick creation’ methods:

Pick closest nodes

This is the default method for creating shells. When you pick a point on the screen the 4 (or 3 if you are creating a tria) closest nodes to the point are automatically selected. The order of the nodes will be automatically calculated for you so there is no danger of creating a shell with a negative area. The shell that will be created is sketched on the screen **but will not be created yet**. Picking a second point on the screen will update the display with the 4 closest nodes to that point. You can carry on picking a point until you have the nodes you require. When all nodes and the part are filled in the **SKETCH** and **CREATE_ELEMENT** buttons will be ungreyed.

By default all visible nodes in the model can be used for this method. The filtering option under ‘Node picking’ allows you to limit the nodes which can be used. If for example you only wanted to use nodes which are on part 1000 this option can be used.

Individual nodes can still be edited by either typing in a new value or using the popups.

Pick individual nodes

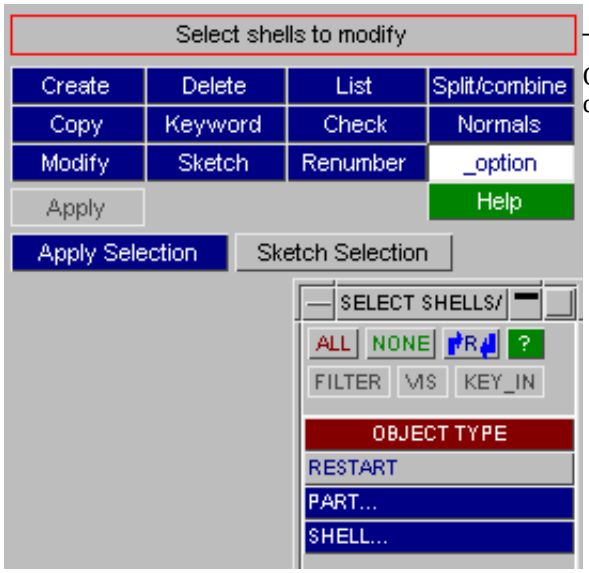
In this mode the nodes are picked from the screen in the order 1, 2, 3, 4, 1, 2… The node which will be picked is indicated by the colour of the node text in the panel. The node which will currently be picked has yellow text. All the others will have white text. You can also edit individual nodes by either typing in a new value or using the popups. With this method the filtering option for node picking is not available.

Auto create

Instead of having to press **CREATE_ELEMENT** each time you want to create a shell the **AUTO_CREATE** option can be used. When this option is set, as soon as the required data is set the element is created. Using this in conjunction with ‘pick closest nodes’ enables creation of a shell with a single click of the mouse.

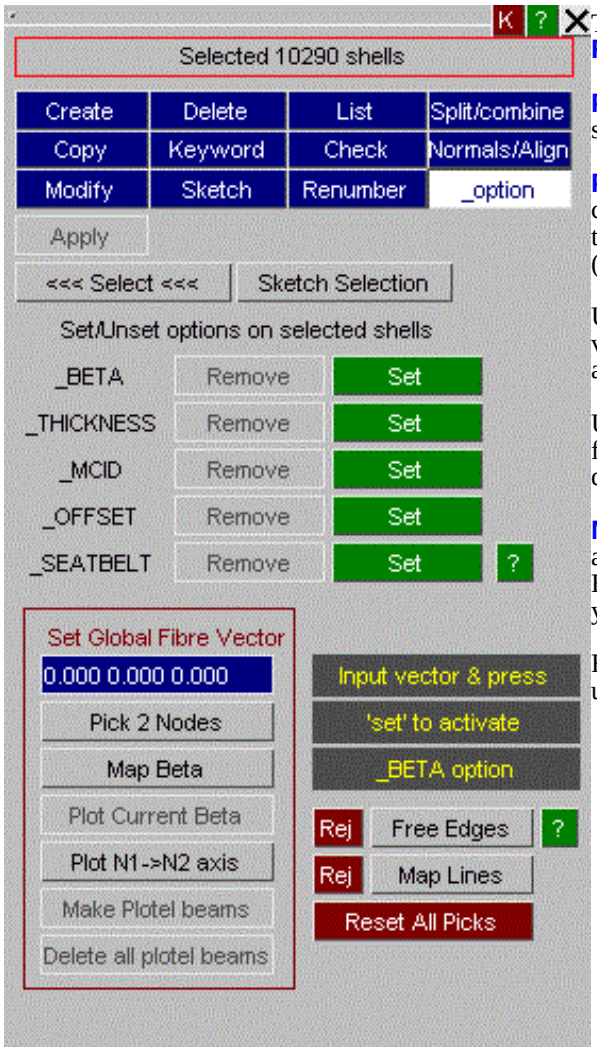
Once the element has been created the **UNDO LAST** button will be made live. This can be used to delete the element if is not what you wanted.

Shell options



_BETA _MCID _THICKNESS and **_OFFSET**

Options for *ELEMENT_SHELL can be set on shells selected directly or by part by using the **_option** function.



The option is set by pressing **Set** or removed by pressing **Remove**

Plot Current Beta shows the current beta angles of any shell_beta amongst the selection

Plot Proposed Beta shows prospective beta angles corresponding to the global input vector. Any shells onto which the vector does not project properly will be highlighted with the (dubious) angle line plotted in red and the user warned.

Use **Set** to set proposed beta angle on all selected shells. If the vector is [0,0,0] the beta angles will be set to zero. The effective angle may be viewed by pressing **Plot N1->N2 beams**.

Use **Map Beta** to orient the beta angles along the map lines and free edges. For more details on usage please refer to the documentation on **Map Fibres** on the composites page.

Make Plotel creates plotel elements to show all current beta angles. These may be removed later by using ELEMENT->BEAM->DELETE->DELETE ALL PLOTEL if you wish.

For other options, the option alone is activated and you need to use the shell Keyword editor to set the values.

Other shell creation commands.

DISMISS	Aborts the current definition and returns to the main element shell menu.
RESET_ALL	Resets all attributes to <null> for this definition: all data entered will be lost, and the panel will return to its initial default state.
COPY_EXISTING	Copies the attributes of an existing shell definition (in the current model). This may then be modified as required.
SKETCH	Sketches the current definition on top of the current image.
LIST_XREFS	Lists everything that references the current shell definition.
CHECK	Performs a check of the current definition, listing any errors

CREATE_ELEMENT Saving the element definition.

Once you have entered the minimum amount of data required to define this shell the **CREATE_ELEMENT** button will be made live, and this permits you to save this definition. (If it is not live the missing fields will be highlighted in red.) The definition will be checked and any errors listed, an then it will saved permanently in this model.

Until you press this, the definition remains volatile, and will be lost if you exit this panel in any other way.

When the shell element is created the part number and the number of nodes are saved as the defaults. When the panel refreshes for you to create another shell these defaults are automatically used to speed up element creation.

Quality checks

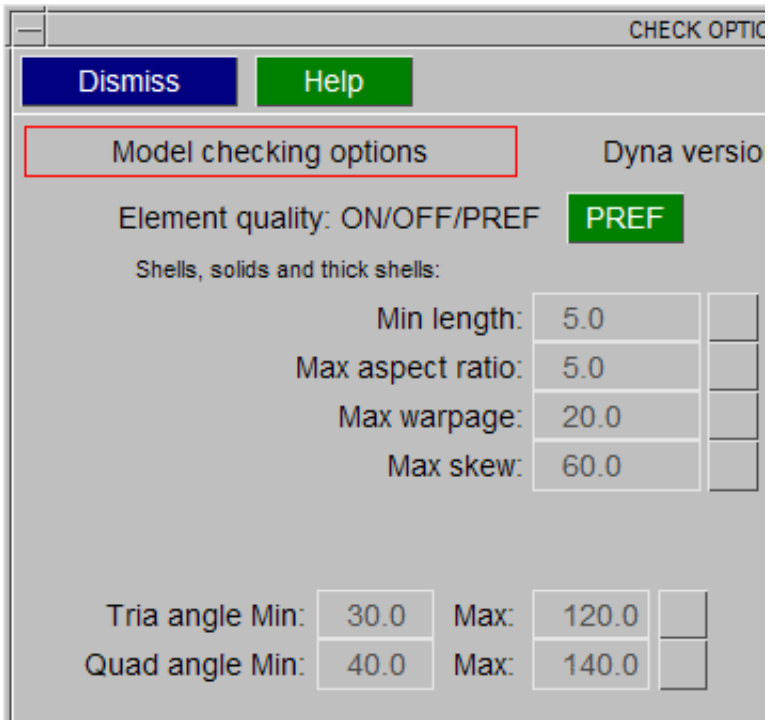
By default quality checks are done on shell elements when they are created to ensure that there are no badly defined elements. If you want to bypass the quality checks then they can be turned off by using the **QUALITY CHECKS** button. It is recommended that you keep the quality

checks on.
The values that are used for the quality checks can be changed by using the **OPTIONS** button next to quality checks. This brings up the main check options panel.

Checks are done for:

- Element length
- Warpage
- Aspect ratio
- Skew
- Minimum and maximum internal angles

The values used for checking can easily be changed.



COPY Copying existing shell(s) to make a new one(s).

You can **COPY** any number of shells, in multiple models.

When **APPLY** is pressed you are asked to confirm what is to be copied, and then the operation is carried out.

For each model the <n> shells chosen in that model are copied using labels <previous highest + 1> to <previous highest +n>, there is currently no control available over the new labels assigned.

MODIFY Modifying the attributes of an existing shell.

This functions in exactly the same way as **CREATE**, using the same panels as in figure Elem_3. The only difference is that the initial state of the panels is already set with the attributes of the shell to be modified.

DELETE Deleting existing elements

The **DELETE** operation works exactly the same way as **COPY** described above, except that the chosen elements are deleted.

- If **DELETE_RECURSIVE** is switched on any nodes, restraints and loads on the elements to be deleted are marked for deletion.
- If recursive deletion is not used only the elements themselves are removed.

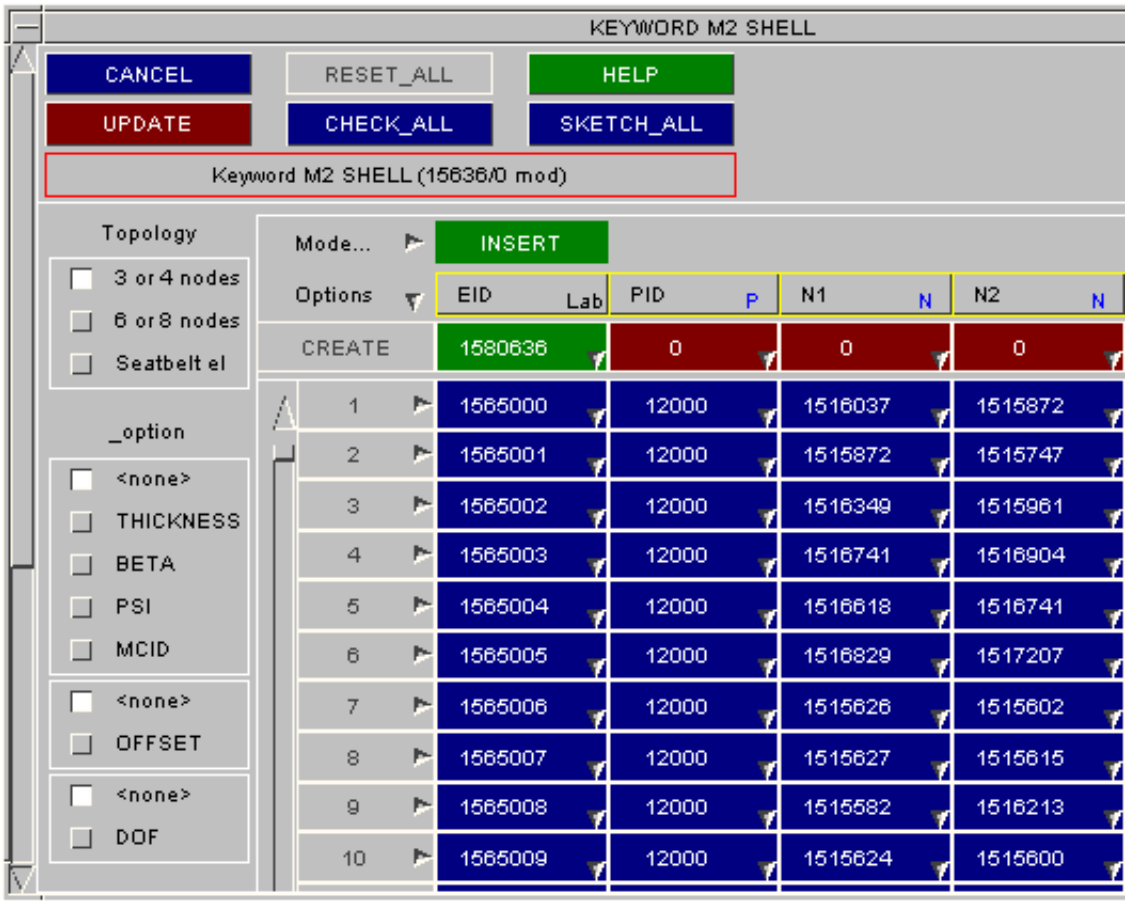
Note also that the standard deletion rules described in Section 6.4.1 still apply: nodes, loads, restraints, etc will only be deleted if nothing else (which is to remain) depends on them.

KEYWORD Generic keyword editor

KEYWORD starts the generic keyword editor which allows creation, deleting and modification of multiple shells. This is useful for modifying multiple shells in a single operation.

This example shows the **SHELL** keyword editor.

Note that the **_THICKNESS, _BETA, _PSI,** etc options require separate layouts since they have a different number of rows of data.



SKETCH Sketch the chosen shell on the current image

SKETCH allows the user to select and sketch individual shells on the current graphics image.

CHECK

Runs the standard checking function on the selected shells. Each shell will be listed either as "OK", or a summary of the errors encountered will be printed. (This is the same as the **CHECK_DEFN** command during shell editing.)

LIST

Writes a summary list of the selected shells to the screen.

RENUMBER

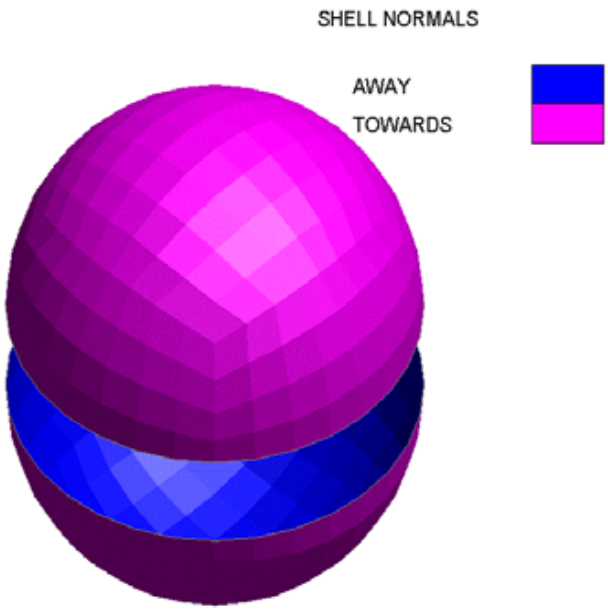
Raises the standard renumbering panel for shells in the chosen model, allowing you to renumber some or all of them.

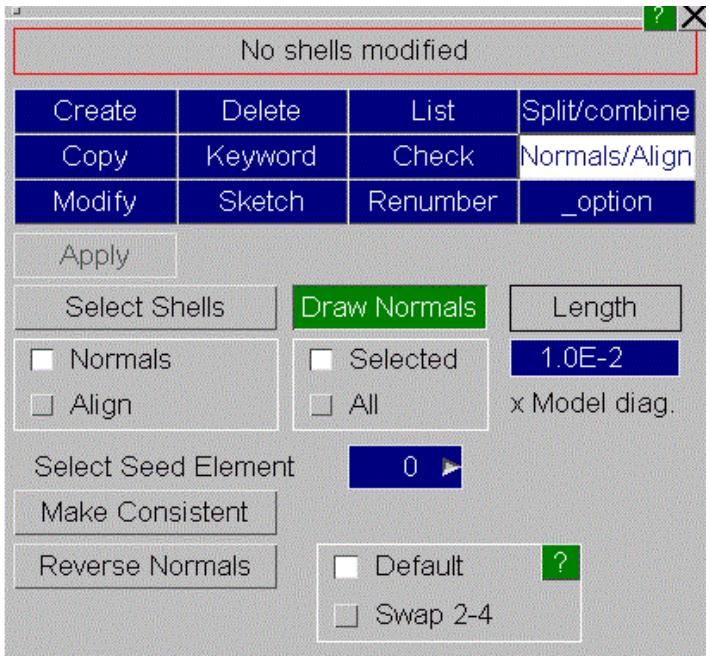
How to move shells from one part to another:

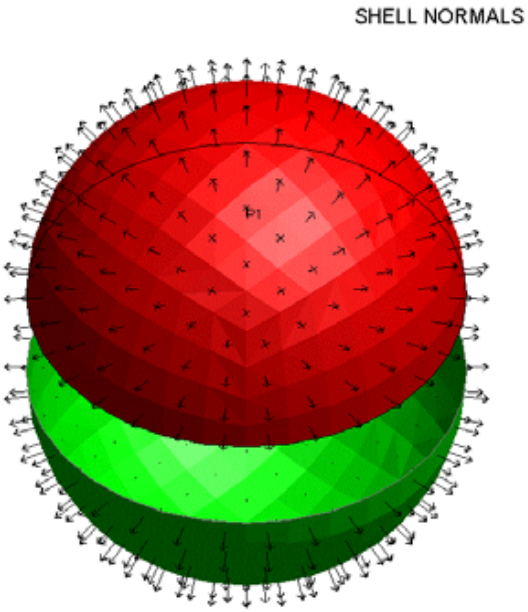
- For a single element, use the **EDIT** panel.
- Otherwise, use the Keyword editor in EDIT mode, select the shells, ill in the new part ID instead of * and press **Apply**.
- Or, select **PART** (in Keywords) > **MODIFY** (select the new part) > **CONTENTS...** > **ADD_ITEMS** and add the shells.

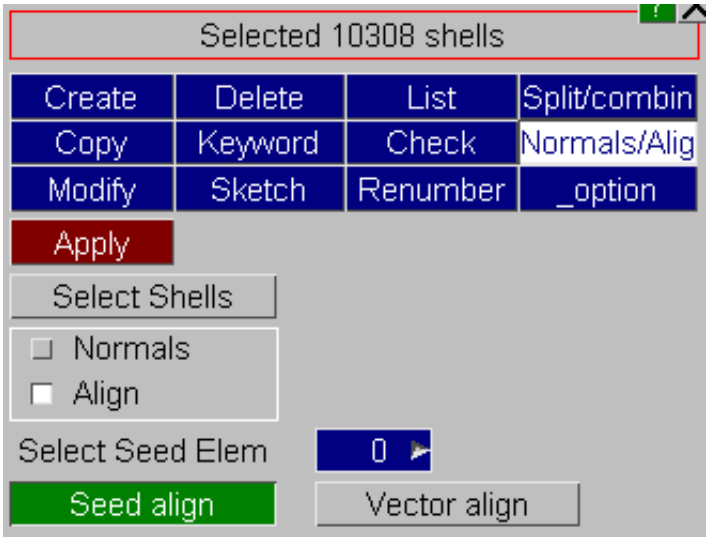
NORMALS/ALIGN

The direction of shell normals can be shown in PRIMER using contour plotting (see section 4.2).









The direction of normals is indicated by the colour of the shell plotted. Blue represents the normal heading away, pink towards. The **Vector** suboption allows the shell normals to be represented as arrows.

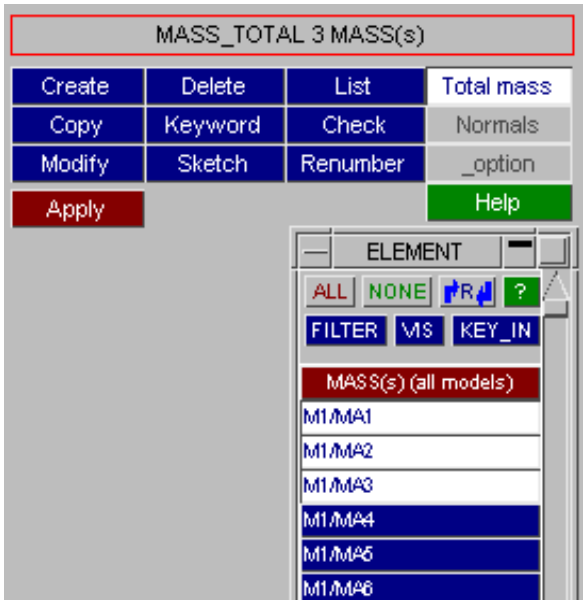
The arrows used to represent shell normals can also be turned on from **NORMALS/ALIGN** panel. The **Draw Normals** button can be used to turn this feature ON or OFF. The user can choose either **Selected** or **All** option which

will display normals to either selected or all shells respectively. The length of the shell normal arrows can also be changed as a multiple of Model diagonal length.

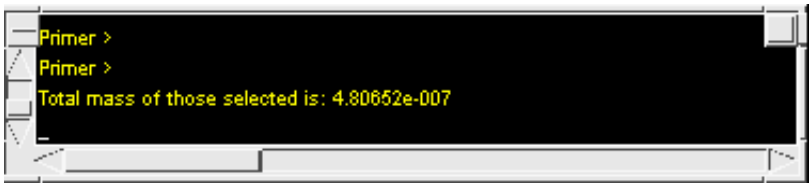
The shell normals menu is shown on the top right. **Select Shells** invokes an object menu whereby the shells can be selected. The selection can be sketched or **Apply Selection** can be used to return to the initial menu. The other options will now be "live" instead of greyed out. It is now possible to either reverse all the shell normals in the selection simply by clicking **Reverse Normals**. The other option is to make all the normals consistent with a selected one, the **Seed Element**.

The align menu can be used to align elements in Primer to a specified vector by changing the nodal order of the elements. **Seed align** should be used if you wish to align elements to the vector defined by N1->N2 of another element. **Vector align** should be used if you wish to align elements with a specified vector, either defined by an input box or by selecting 2 nodes.

SUM OF SELECTED MASSES



- The ELEMENT_MASS panel offers a function to sum the mass of mass elements selected from the object menu.



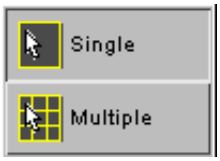
- The total mass of the selected mass elements is reported in the dialogue box when you press **APPLY**.

SPLIT/COMBINE

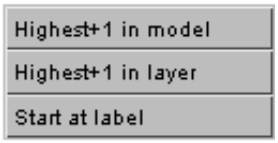
The split/combine panel allows you to manipulate shell elements. You can:

- Split shells using predefined patterns
- Split shells by drawing a line
- Find warped quads and split into 2 trias.
- Fix transitions between adjacent shells
- Detach shells from a mesh
- Combine shells together into one shell

To change the mode use the popup on the top left of the panel. The options will then change accordingly.

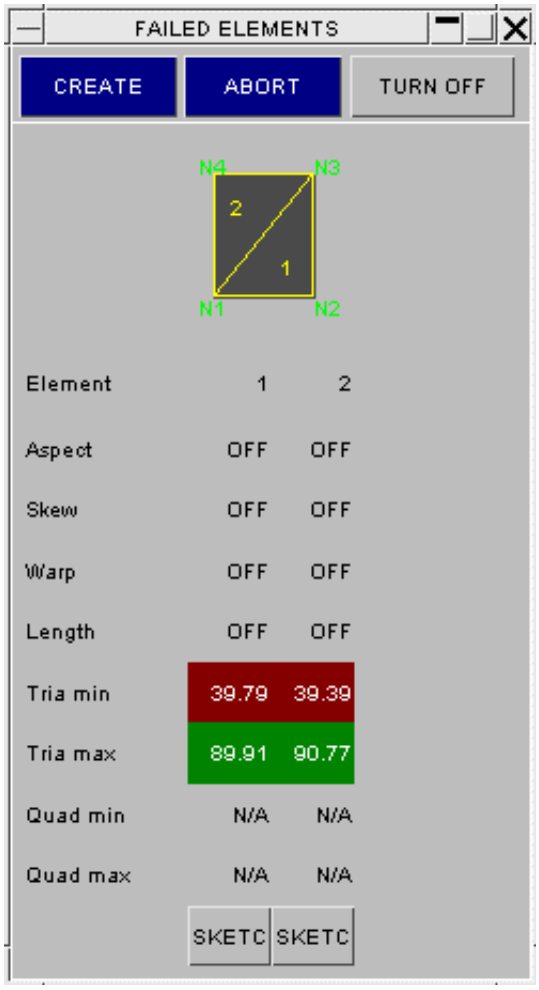


Some modes allow you to work on a single shell or multiple shells. The default, is **Single** mode. In this mode 'quick picking' is activend. Alternatively, to operate on many shells at the same time, select **Multiple** mode. The standard object menu is mapped to allow you to choose the shells you want to modify. Press **APPLY** to change them.



You can choose what labels to use for any new nodes and shells that are created. Use the popup to select which option you require.

If you choose **Start at label** then give a label number to start from. PRIMER will try to use that number. If a node or beam already exists with that label it will revert to **Highest+1 in model**.

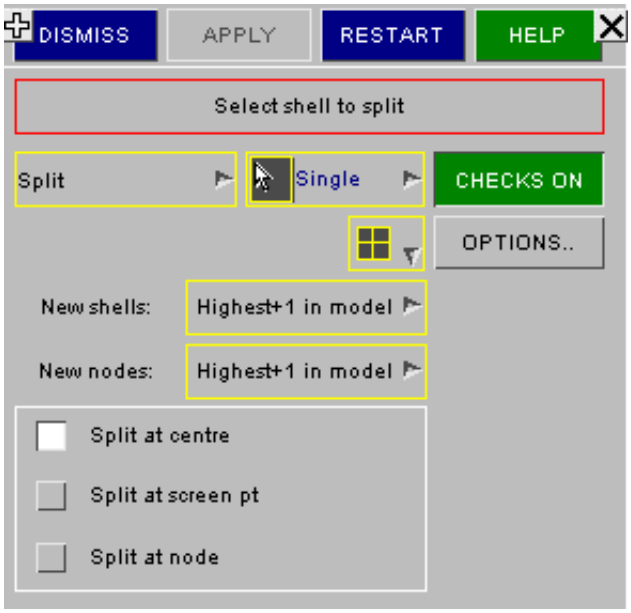


The **checks** button allows you to trap creating elements which do not pass specific quality checks. With **CHECKS ON** this checking is done. The values and types of checks done can be cahnged with the **OPTIONS...** button.

In the example on the right a quad is being split into 2 trias. However the minimum angle for the tria (38.66) is less than the allowed angle. PRIMER is warning you of this. **CREATE** will force the creation of the shell, **ABORT** will stop this operation. **TURN OFF** will turn the checks off. This is useful if you are splitting lots of shells.

Predefined split patterns

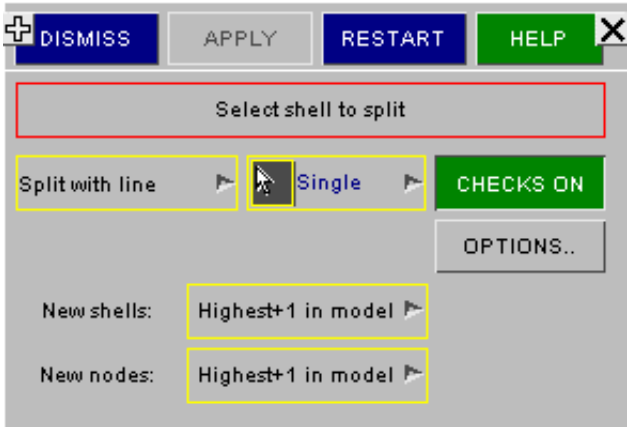
There are several predefined split patterns. To change the pattern use the popup.



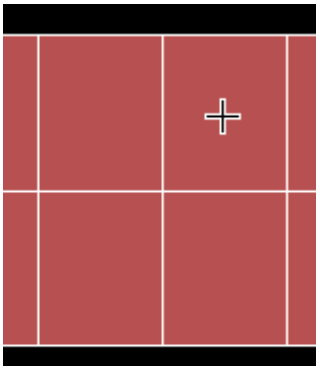
In **Single** mode just click on a shell to split it. You can split the shell at the centre of the shell, at the point you click on the screen (projected onto the shell) or at an existing node location.

In **Multiple** mode select the shells you want to split using the object menu and press **APPLY**.

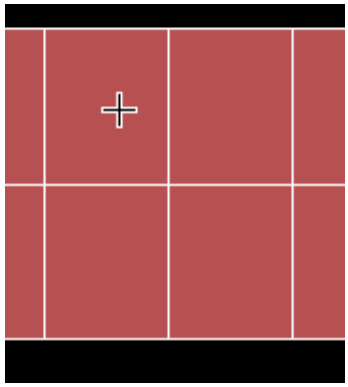
Splitting by line



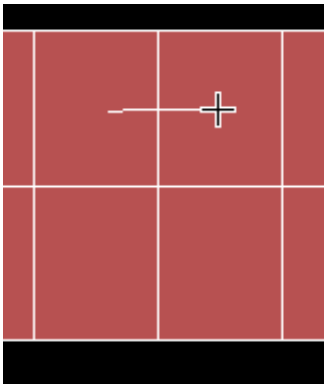
Single mode



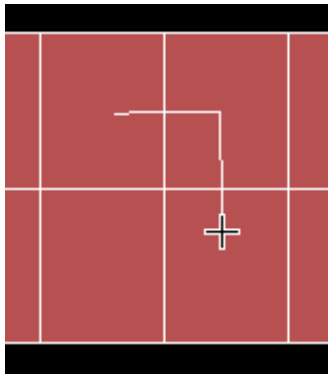
1. Select the shell to split



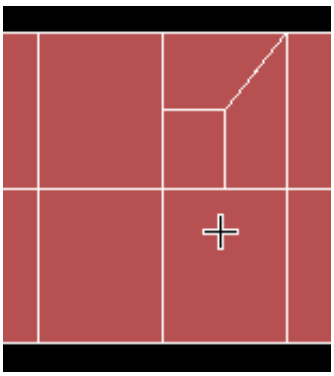
2. Click the first point on the line



3. Click the seond point on the line

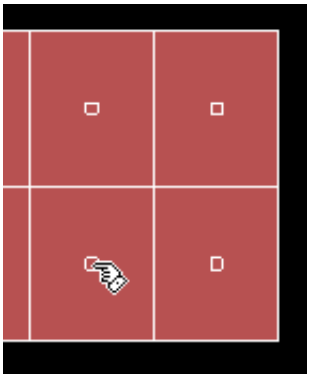


4. Click the third point on the line (if required)

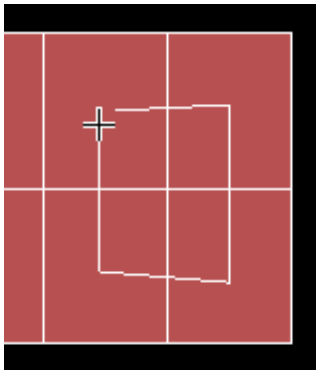


5. The shell is split

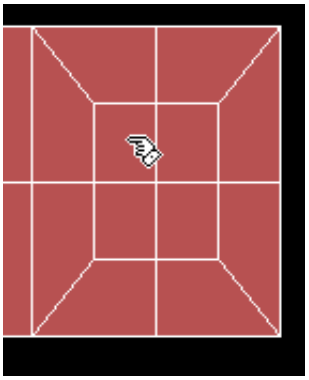
Multiple mode



1. Select the shells to split and press **DRAW LINE**.



2. Draw the line by clicking with the mouse.



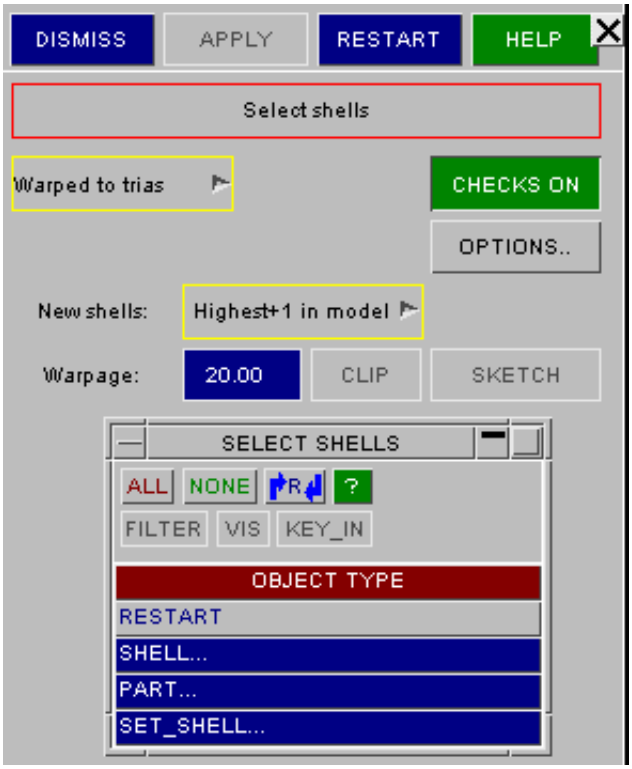
3. Press **APPLY**. The shells are split

Splitting warped quads

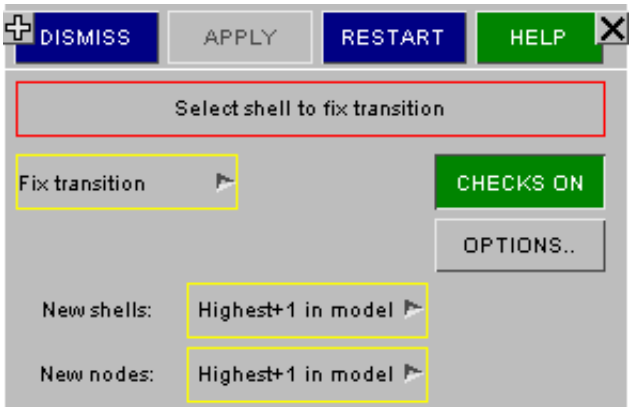
Select the shells you want to check/split by using the object menu. Give a value for the maximum warpage.

You can sketch the shells that are warped by pressing the **SKETCH** button. They can be placed on the clipboard by pressing **CLIP**.

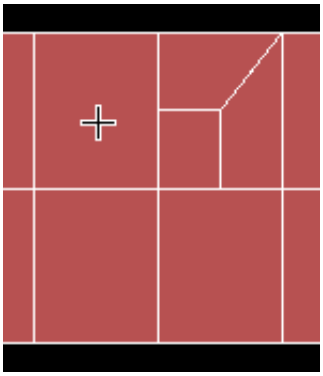
To split the shells into trias press **APPLY**.



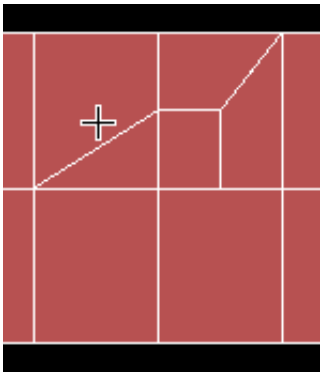
Fixing mesh-transitions



Fix transitions looks at neighbouring elements to see if the mesh is continuous. If it is not, the element is split to make it continuous.

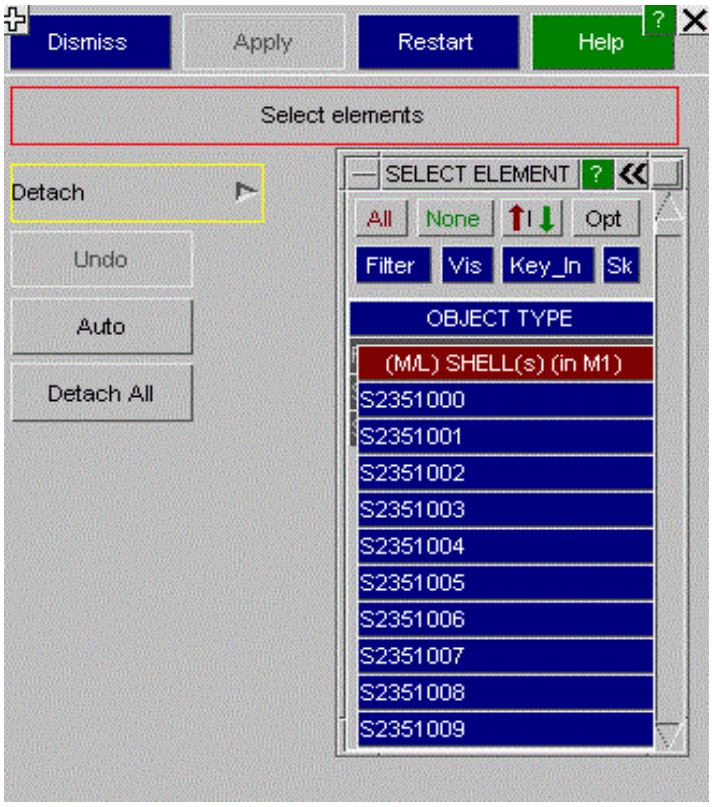


1. Click on the shell you want to split.



2. The shell is split to make a continuous mesh.

Detaching shells

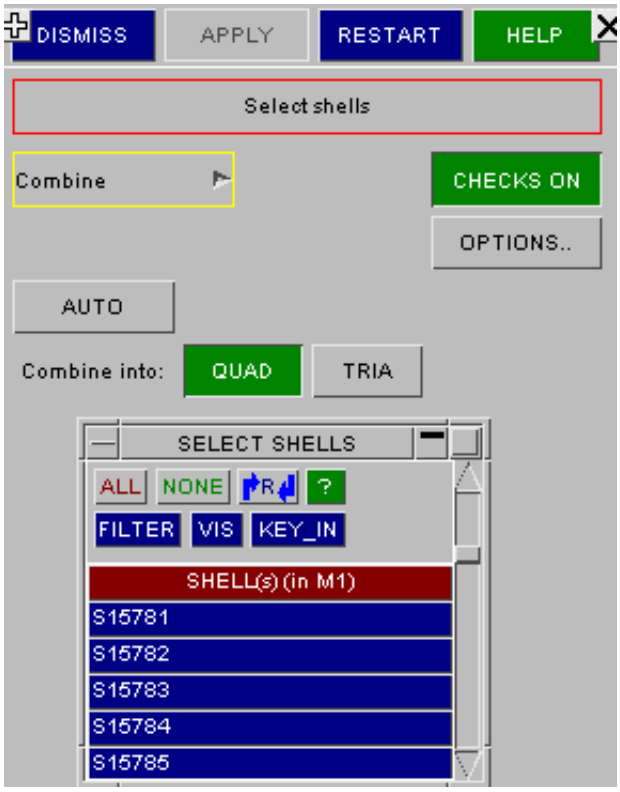


To detach one or more shells from a mesh use the **Detach** function. Select the shells you want to detach by either clicking on the screen or using the object menu.

If **AUTO** is enabled, once you have the correct number of shells they will be detached, otherwise press **APPLY** to detach them.

If **DETACH ALL** is enabled, all the selected shells will be detached from each other, otherwise only the shells which are on the boundary among all the selected ones will be detached.

Combining shells



To combine two or more shells together use the **Combine** function. Select the shells you want to combine by either clicking on the screen or using the object menu.

If **AUTO** is enabled, once you have the correct number of shells they will be combined, otherwise press **APPLY** to detach them.

The **QUAD** and **TRIA** buttons can be used to choose what to combine the shells into.

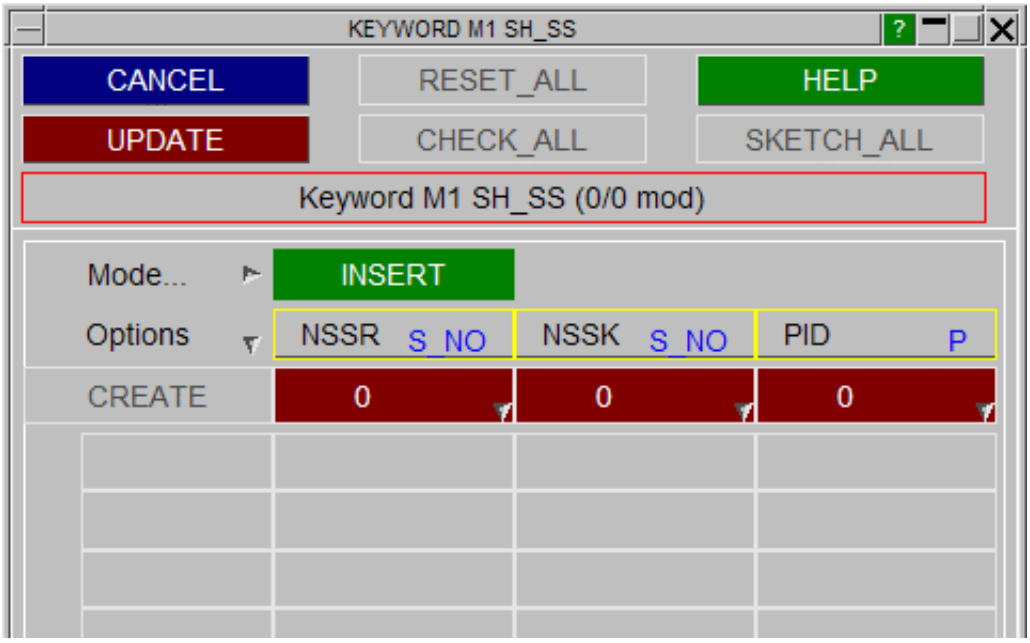
Convert Mesh Elements

The MESH_SURFACE_ELEMENTs can be converted to SHELLs along with referenced MESH_PARTs to PARTs and MESH_NODEs to NODEs.

The details can be found in Volume III Keywords section.

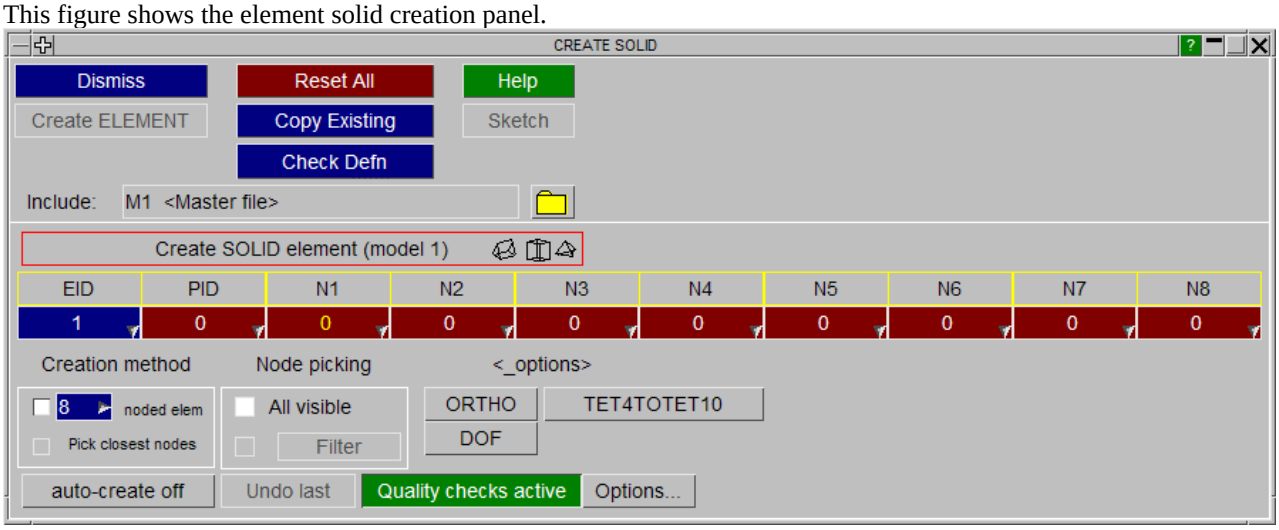
ELEMENT_SHELL_SOURCE_SINK

This figure shows the shell source sink keyword editing panel.
This keyword defines a strip of shell elements of a single part ID to simulate a continuous forming operation



ELEMENT_SOLID

This figure shows the element solid creation panel.



The **ORTHO** <_option> button can be used to change whether an **ELEMENT_SOLID** or an **`ELEMENT_SOLID_ORTHO`** is created.

Tetra, penta and hexa solid elements can be created by changing the number of nodes by typing in the number of nodes or using the popup.

Alternatively, if the number of nodes is left set at eight and the nomal LS-Dyna method for creating either penta elements (N1, N2, N3, N4, N5, N5, N6, N6) is used a penta element will be created, or tetra elements (N1, N2, N3, N4, N4, N4, N4, N4) a tetra element will be created.

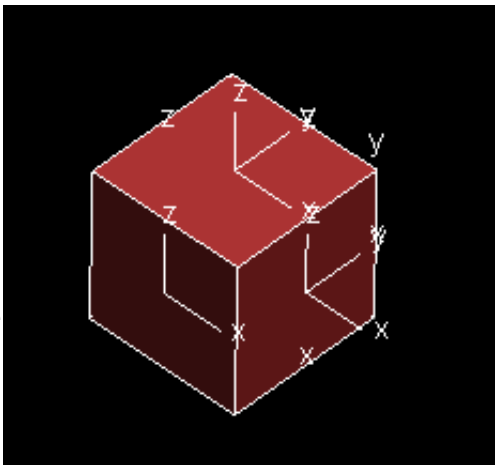
The '**Pick closest nodes**' and '**Node picking**' options are not available for solid elements.

When a solid element is created the part number and the number of nodes are remembered as defaults for the next element.

ALIGN

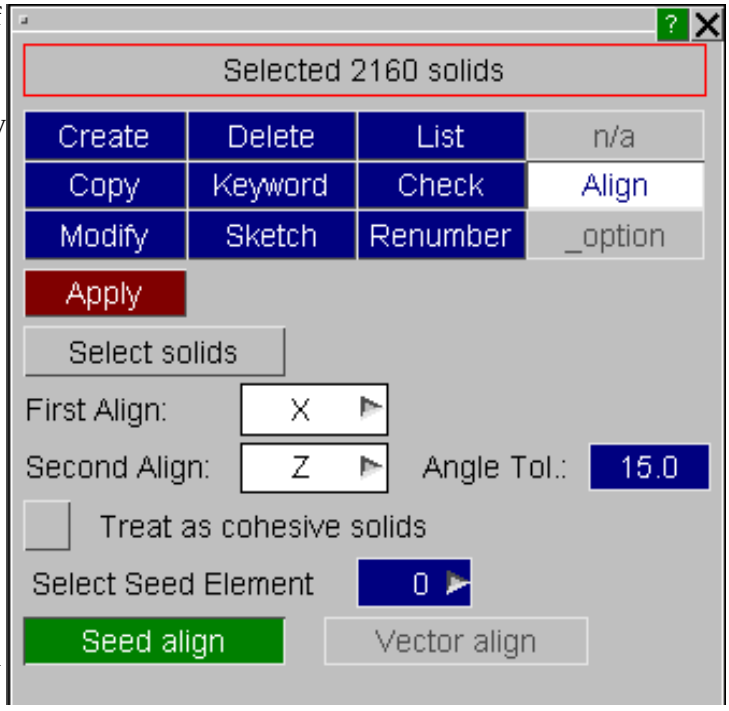
The direction axes of a solid element can be showed in PRIMER using Entities->Triad check button. (see section 4.4.3).

- X-Axis is the direction of edge connecting the first two nodes of the solid element.
- Z-Axis is the direction normal to the plane forming the first face of the solid element.
- Y-Axis is the cross product of Z and X axes directions.



The solid align menu is shown on the bottom right. **Select Solids** invokes an object menu whereby the solids can be selected. The selection can be sketched or **Apply Selection** can be used to return to the initial menu. The other options will now be "live".

The align menu can be used to align elements in PRIMER to a specified vector by changing the nodal order of the elements. **Seed align** should be used if you wish to align elements to the vector directions defined by nodes of another element. **Vector align** should be used if you wish to align elements with a specified vector, either defined by an input box or by selecting 2 nodes.



You can specify which axes to align by choosing options from **First align** and **Second align** menus.
First align is the first axes to be aligned and **Second align** is to align the second axes after alignment of the first axes. Second axis alignment is optional.

If two axes directions are chosen to be aligned, PRIMER compulsorily needs to define a seed solid. If the seed solid is still not defined, PRIMER assumes the first selected element in the list as the seed solid.

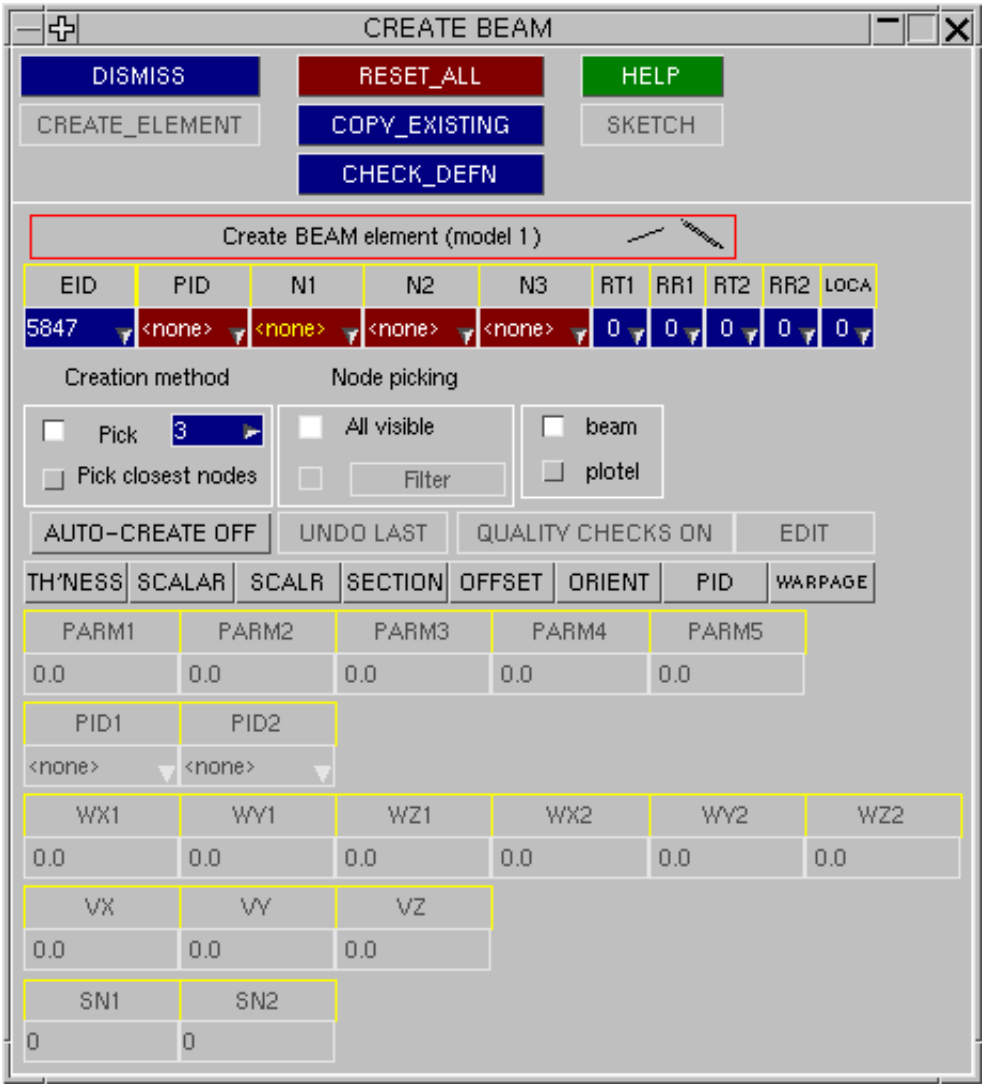
If while aligning a hexa-solid element (8-noded solids), the user does not want to change the Top/Bottom faces of the element to its side faces, **Treat as cohesive solids** option can be checked ON.

To align the second axes, the first axes alignment may have to be changed. For this purpose, the **Angle Tolerance** value can be modified. This value is the maximum angle allowed between the final first axes and the first axes direction achieved. Essentially it allows the first axes to deviate from the "best" alignment found by a specified tolerance to achieve the second axis alignment.

ELEMENT_BEAM

CREATE

This figure shows the element beam creation panel.



There is an option to create a **plotel** beam which is a 2 noded beam used for display purposes only.

The row of buttons **THICKNESS, SCALAR, SCALR, SECTION, OFFSET, ORIENT, PID, WARPAGE** can be used to select beam options, with proviso that THICKNESS and SCALAR are exclusive.

`ELEMENT_BEAM`
`ELEMENT_BEAM`_option1_option2

Three, two (discrete and spotweld), and one (spotweld) beam elements can be created by changing the number of nodes by typing in the number of nodes or using the popup.

There are no element quality checks available for beam elements at present.

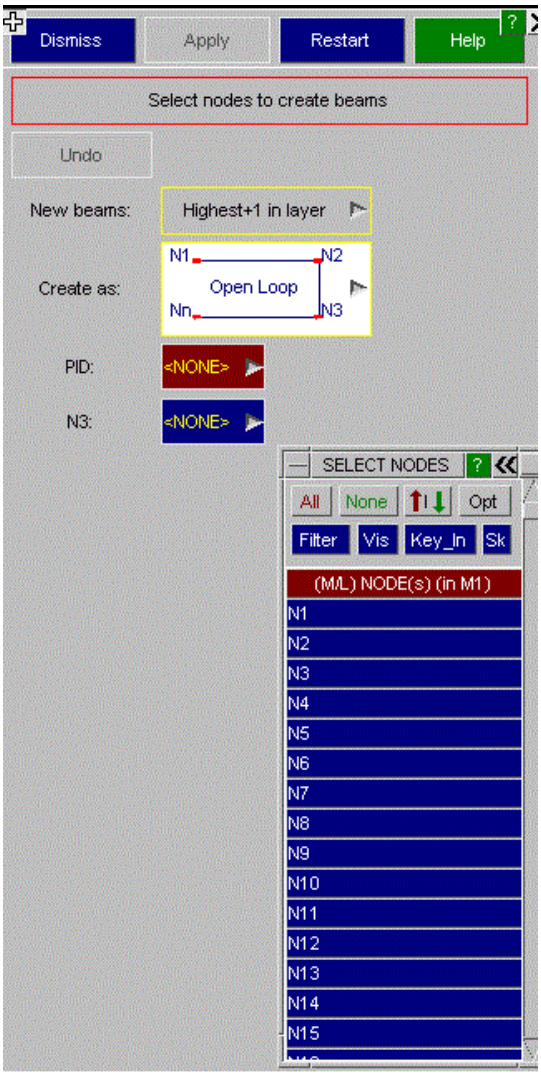
When a beam element is created the part number and the number of nodes are remembered as defaults for the next element.

Additionally if a three noded beam is created the third node is also remembered as a default.

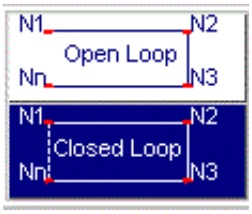
Create Multiple Beams on Nodes

The panel allows you create multiple beams connecting the selected list of nodes.

- All beams will have same part (PID) and orientation node (N3) defined in the Panel. N3 is kept optional for creation of beams.
- The beams will be created connecting nodes in exact order of selection by the user.



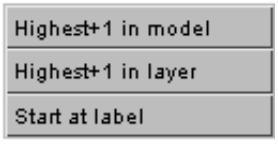
An extra beam can also be created connecting the first and the last nodes in the selection list. This is based on the user selection in 'Create as' pop-up option.



As an example, if four different nodes N1-N2-N3-N4 are selected by the user in this order:

- Open Loop' : Three beams will be created connecting the nodes in this order: N1-N2, N2-N3 and N3-N4.
- Closed Loop : An extra fourth beam will be created connecting the nodes N4-N1.

You can choose what labels to use for any new nodes and beams that are created. Use the popup to select which option you require.



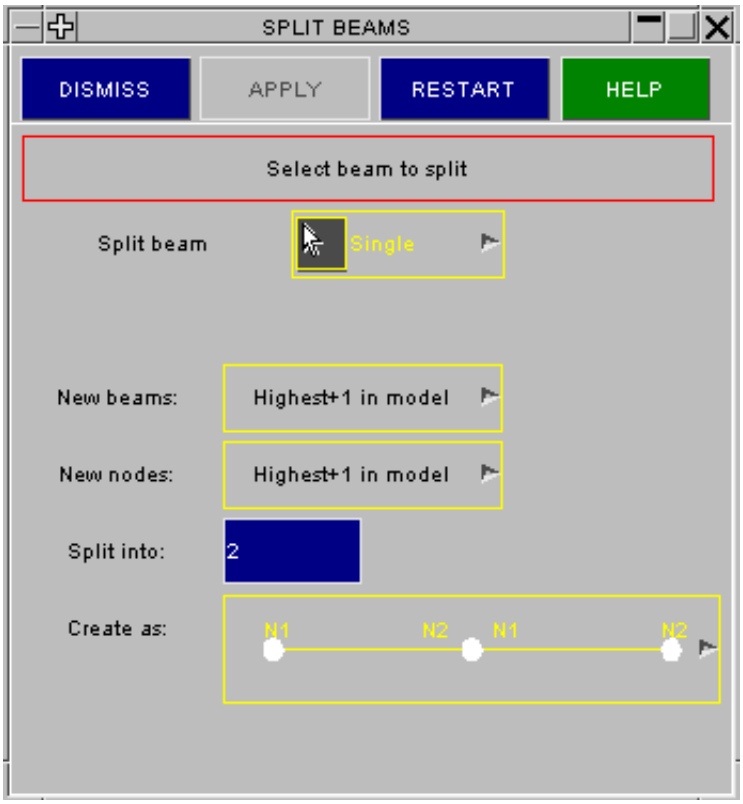
If you choose **Start at label** then give a label number to start from. PRIMER will try to use that number. If a node or beam already exists with that label it will revert to **Highest+1 in model**.

SPLIT

The SPLIT panel allows you to split beams into 2 or more beams.

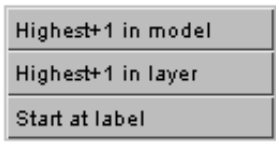
- Any beams which are created will have the same 3rd node as the original beam.
- Release conditions on the original beam nodes will be retained.
- Thickness parameters for beams will be correctly calculated (e.g. if the original beam is tapered)
- If the beam is in a set, the new beams will automatically be added to the set.

If the beam has a segment (2 noded)on it or a ***DATABASE_HISTORY_BEAM**, ***LOAD_BEAM** or ***INITIAL_STRESS_BEAM** card on it the beam cannot be split.



By default, 'quick picking' is activated in **Single** mode, and each beam you pick will be split. Alternatively, to split many beams at the same time, select **Multiple** mode. The standard object menu is mapped to allow you to choose the beams you want to split. Press **APPLY** to split them.

You can choose what labels to use for any new nodes and beams that are created. Use the popup to select which option you require.

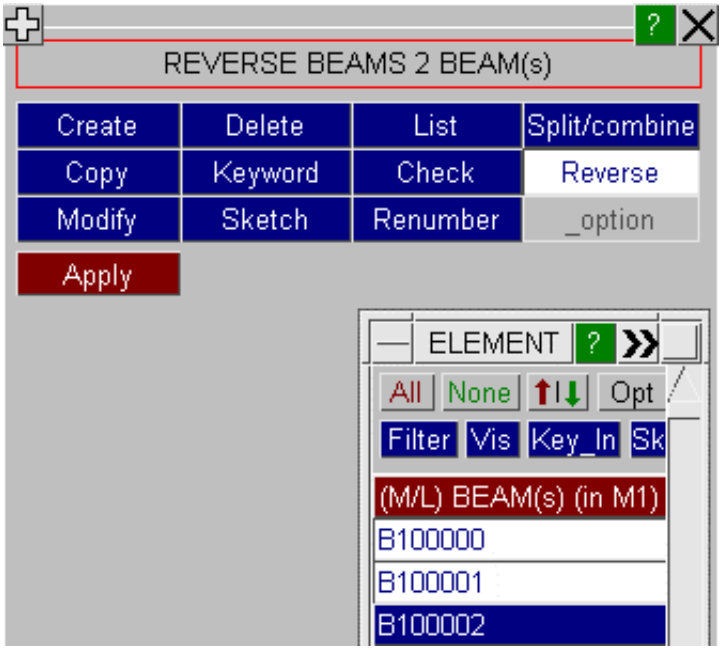


If you choose **Start at label** then give a label number to start from. PRIMER will try to use that number. If a node or beam already exists with that label it will revert to **Highest+1 in model**.

If you split a beam into two, you have the option of choosing the direction of the second beam. This can be useful if using the beam with ***MAT_SEISMIC_BEAM** as a plastic hinge can only be formed at one end of the beam. In this case N1-N2,N2-N1 should be used. For other analyses N1-N2,N1-N2 should be used so that forces and moments are plotted correctly in post-processing.

REVERSE

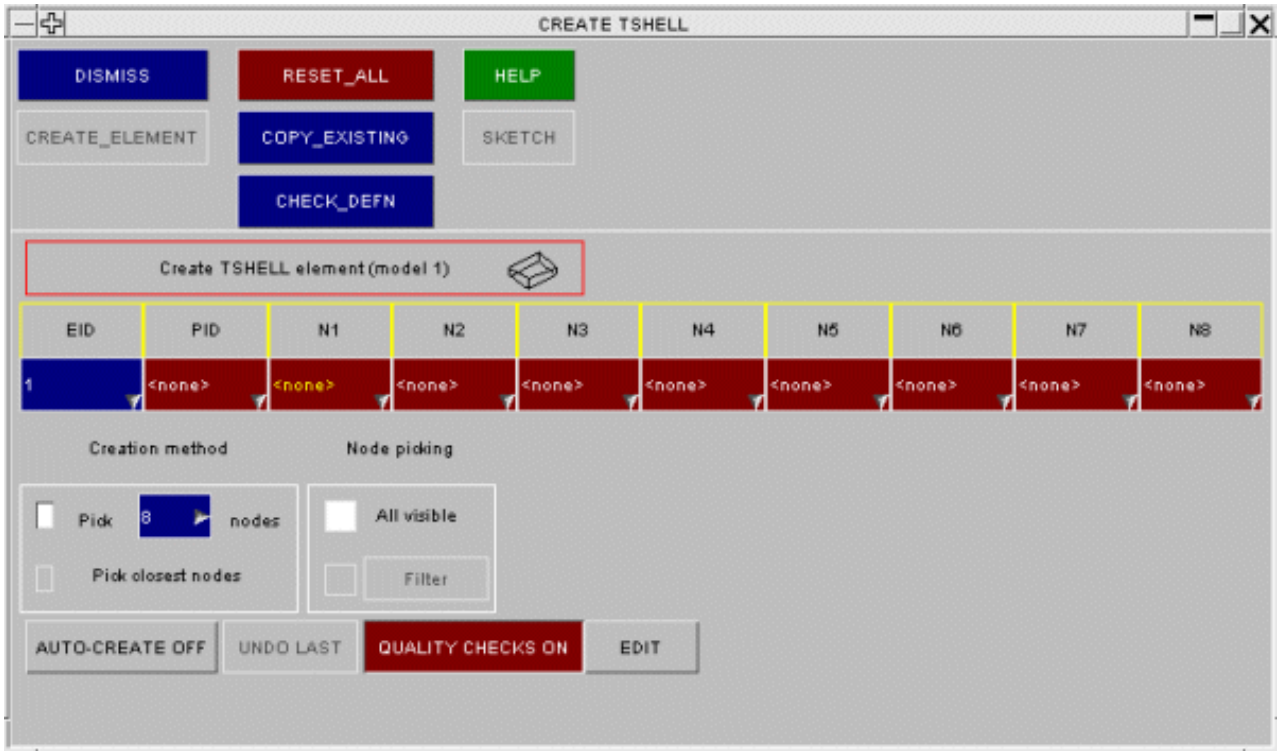
REVERSE will permit users to change beam orientation by swapping nodes N1 and N2.



ELEMENT_TSHELL

This figure shows the thick shell element creation panel.
The thick shell creation method is identical to the solid element method except that:

- Only triangular and quadrilateral elements are allowed.
- There is no **ORTHO** option



ELEMENT_DISCRETE

This figure shows the element discrete creation panel.
Two and one (grounded) discrete elements can be created by changing the number of nodes by typing in the number of nodes or using the popup.

There are no element quality checks available for discrete elements at present.

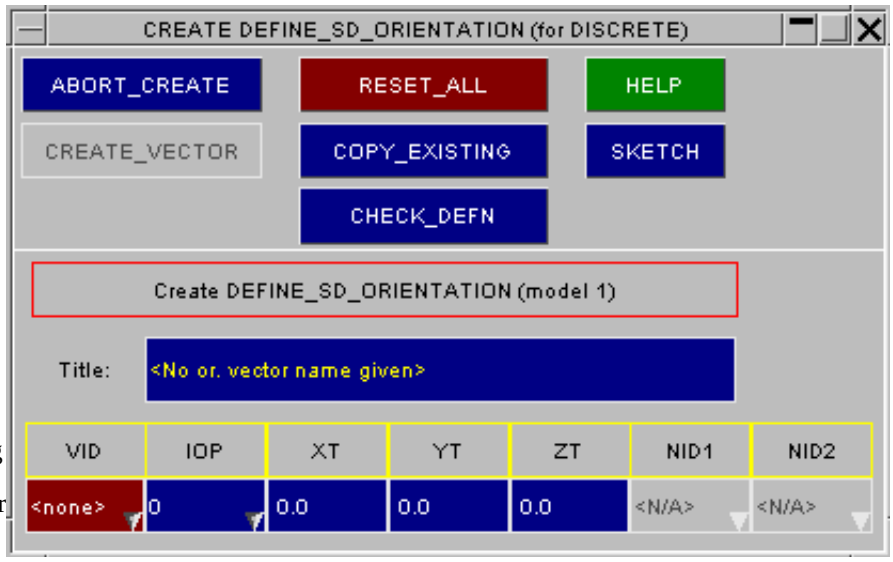
When a discrete element is created the part number and the number of nodes are remembered as defaults for the next element.

If an orientation vector (VID) is required:

If the discrete element needs to use an orientation vector then one can be created/ edited by using the **VID** popup.

The panel that the create option brings up can be seen in figure Elem_9.

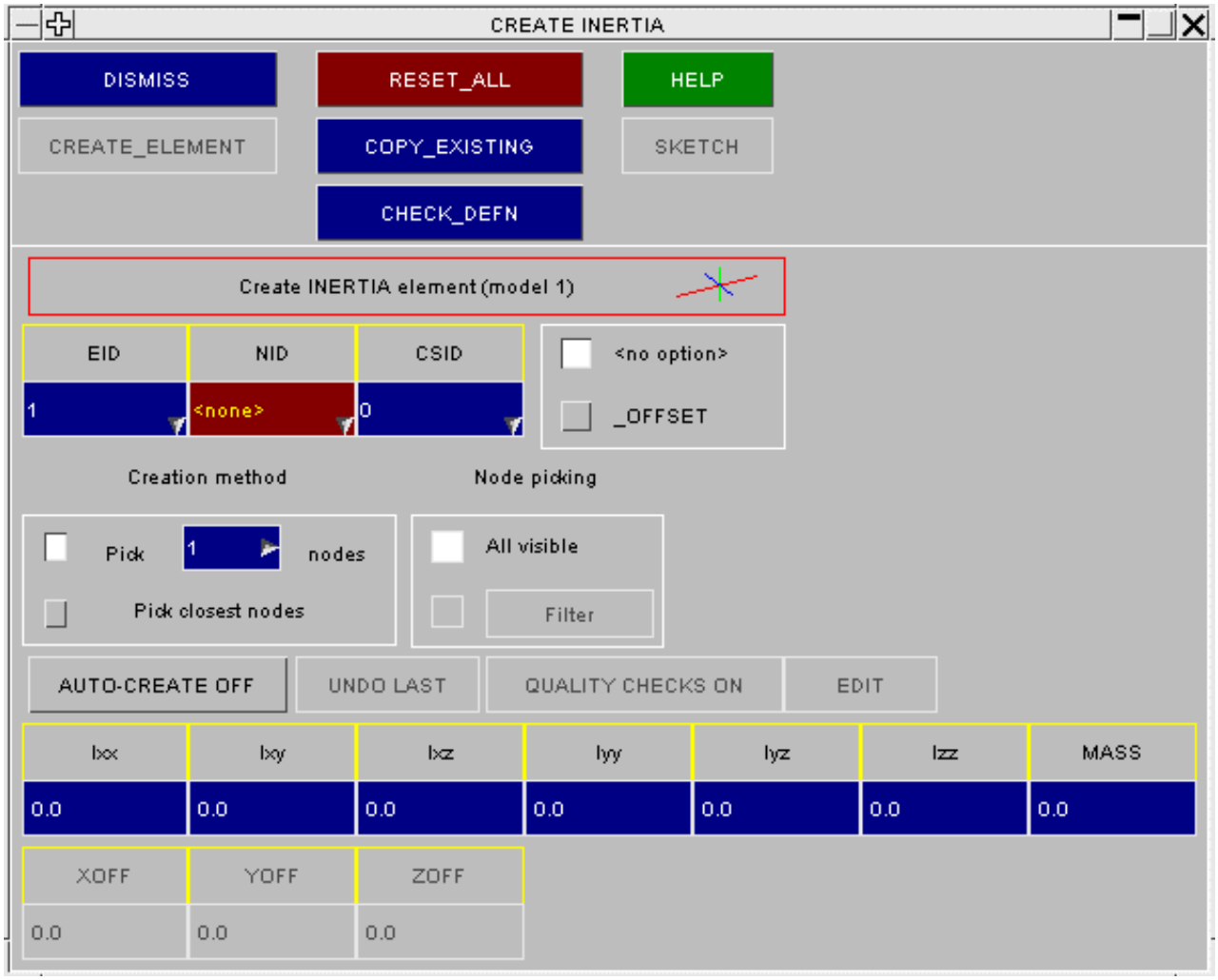
In this panel the label can be defined as usual by typing in the number or using the popup. The type of orientation vector can be set by using the **IOP** option and either the vector set using the **XT**, **YT** and **ZT** fields or the 2 nodes defined by typing in or using the popups for the **NID1** and **NID2** fields



ELEMENT_INERTIA

This figure shows the inertia element creation panel.
Only 1 node needs to be picked for the inertia element. The components of the inertia tensor and the coordinate system are saved as defaults when an element is created.

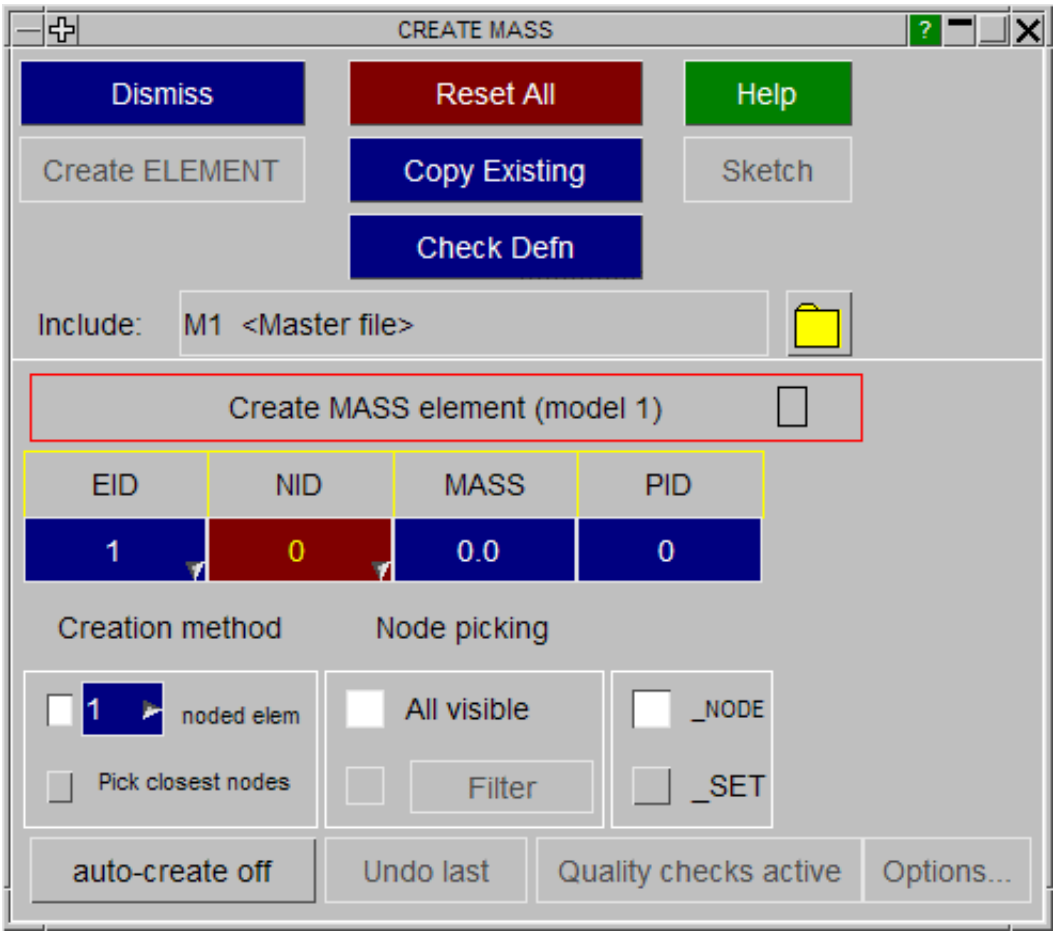
There are no element quality checks available for inertia elements at present.



ELEMENT_MASS

This figure shows the mass element creation panel.
Only 1 node needs to be picked for the mass element. The mass is saved as a default when an element is created.

There are no element quality checks available for mass elements at present.

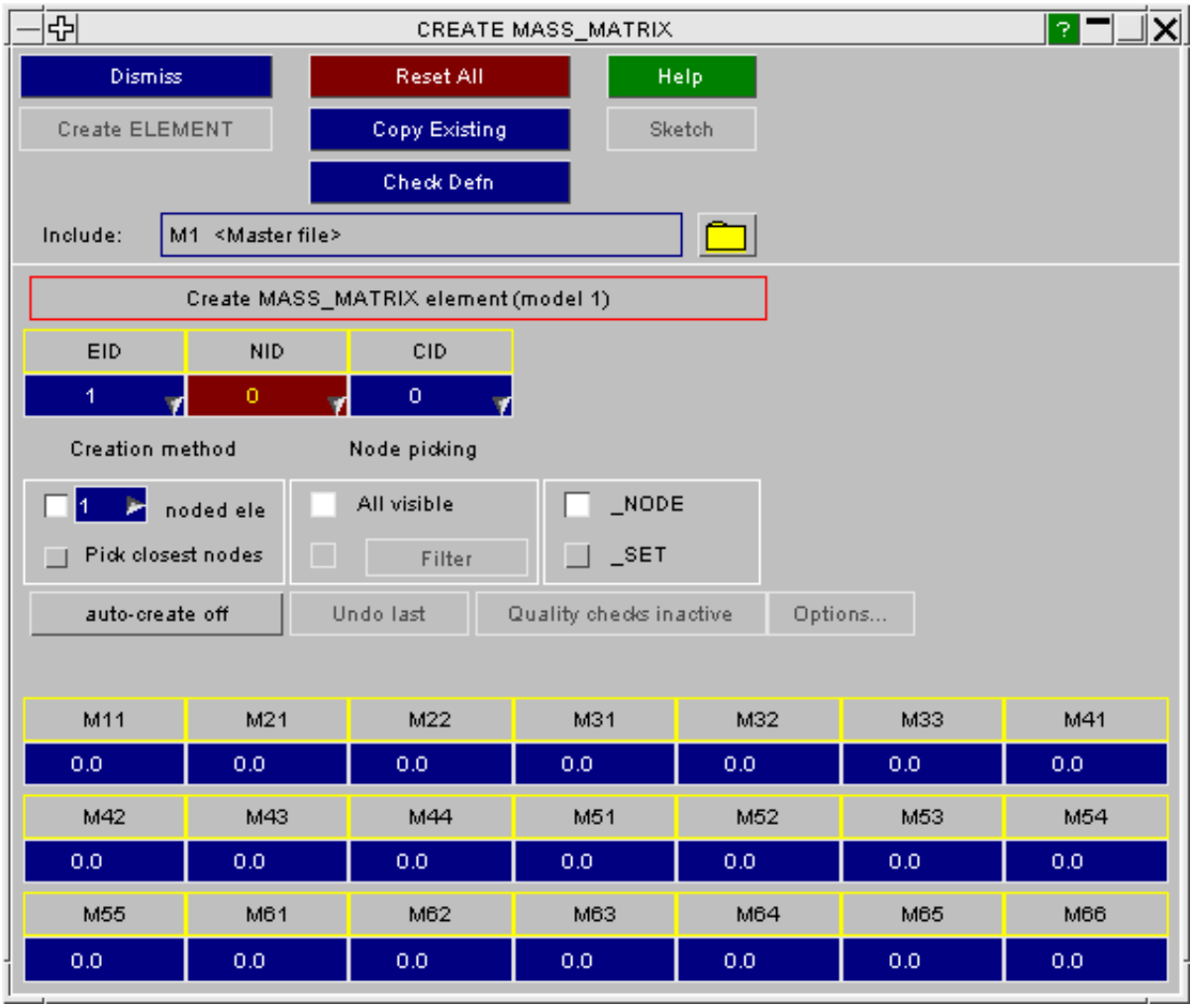


ELEMENT_MASS_MATRIX

This image shows the Create/Modify panel for *ELEMENT_MASS_MATRIX

One node, or node set, is selected at a time and a 6x6 mass matrix is defined for it. This is described in more detail below.

No quality checks are carried out for this element.



The 6x6 mass matrix [M]: what it means and how PRIMER handles it.

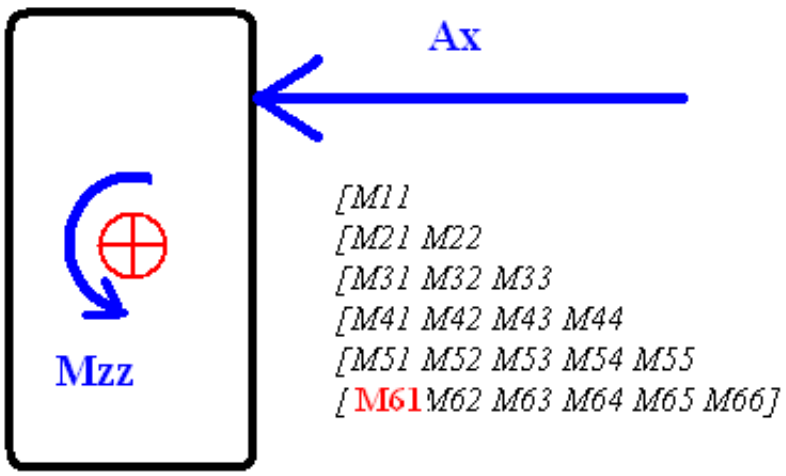
This element reads a symmetric 6x6 mass matrix which populates all possible terms of the classic **F = M.A** equation as follows:

Force	=	**Mass**							**x**	**Accel**			
Fx	[	M11	*m12*	*m13*	*m14*	*m15*	*m16*	]		Tx	Where:	Fx/y/z	are translational forces in [X,Y,Z]
Fy	[	M21	M22	*m23*	*m24*	*m25*	*m25*	]		Ty		Mxx/yy/zz	are rotational moments about [X,Y,Z]
Fz	= [	M31	M32	M33	*m34*	*m35*	*m36*	]	x	Tz		Tx/y/z	are translational accelerations
Mxx	[	M41	M42	M43	M44	m45	m46	]		Rxx		Rxx/yy/zz	are rotational accelerations
Myy	[	M51	M52	M53	M54	M55	m56	]		Ryy			
Mzz	[	M61	M62	M63	M64	M65	M66	]		Rzz			

The matrix is symmetric, with only the lower triangle defined, so upper triangle terms M12 etc (in italics above) are identically equal to M21 etc.

A good way to think of term **Mij** is that it links **force** in direction <**i**> with **acceleration** in direction <**j**>. For example:

Acceleration in X (Ax) applied away from the centroid causes moment about Z (Mzz). These two are linked by matrix term M61.



Extracting Mass and Inertia from the 6x6 matrix [M]

This raises two interesting questions:

- What is the (scalar) mass of this element?
- What is the inertia of this element?

These properties are needed when PRIMER calculates the mass and inertia of a model.

To understand how PRIMER calculates these values it is necessary to consider the full 6x6 [M] matrix above as the

following block matrix of 3x3 sub-matrices:

[	**[A]**	**|**	*[B]*	**]**	Where:	**[A]**	is a symmetric tensor describing mass
[	**----**	**+**	**----**	**]**		**[C]**	is an unsymmetric tensor describing cross-linking terms (*[B]* is the transpose of **[C]**)
[	**[C]**	**|**	**[D]**	**]**		**[D]**	is a symmetric tensor describing inertia

The "cross-linking" terms in sub-matrix [C] will be non-zero if the matrix has been set up to describe the result of accelerations not acting through the centroid, as in the image above, and - strictly - the full [M] matrix should be reworked to reduce the terms in [C] to zero before [A] can be considered to be "pure mass" and [D] "pure inertia". The physical equivalent of this reworking would be to shift the matrix so that it described accelerations acting through the element centroid.

However this is very hard to do and, given the unconstrained nature of the input, it would be easy for a user to define an unrealistic (non-physical) matrix in which this is not possible, so PRIMER adopts the following simplified approach:

MASS is taken to be the 1st invariant of sub-matrix [A]

That is the average of the leading diagonal terms: **mass = (M11 + M22 + M33) / 3.0**

INERTIA is taken to be the symmetric tensor in sub-matrix [D]:

[	**Ixx**	*Ixy*	*Ixz*	**]**		**[**	**M44**	*M45*	*M46*	**]**
[	**Iyx**	**Iyy**	*Iyz*	**]**	**=**	**[**	**M54**	**M55**	*M56*	**]**
[	**Izx**	**Izy**	**Izx**	**]**		**[**	**M64**	**M65**	**M66**	**]**

The "cross-linking" terms in sub-matrix [C] <u>are ignored</u>.

If the terms in sub-matrix [C] are large, ie the matrix [M] has been written to describe behaviour remote from the element centroid, then the Inertia (which includes "distance squared" terms) and - to a lesser extent - the Mass calculated using the methods above will be wrong.

Rotating matrix [M]

PRIMER rotates a *ELEMENT_MASS_MATRIX as follows:

The 6x6 matrix [M] is treated as a block matrix of three 3x3 matrices:

[	**[A]**	**|**	*[B]*	**]**
[	**----**	**+**	**----**	**]**
[	**[C]**	**|**	**[D]**	**]**

Each sub-matrix [A], [B], [D] is rotated independently as a 3x3 tensor.

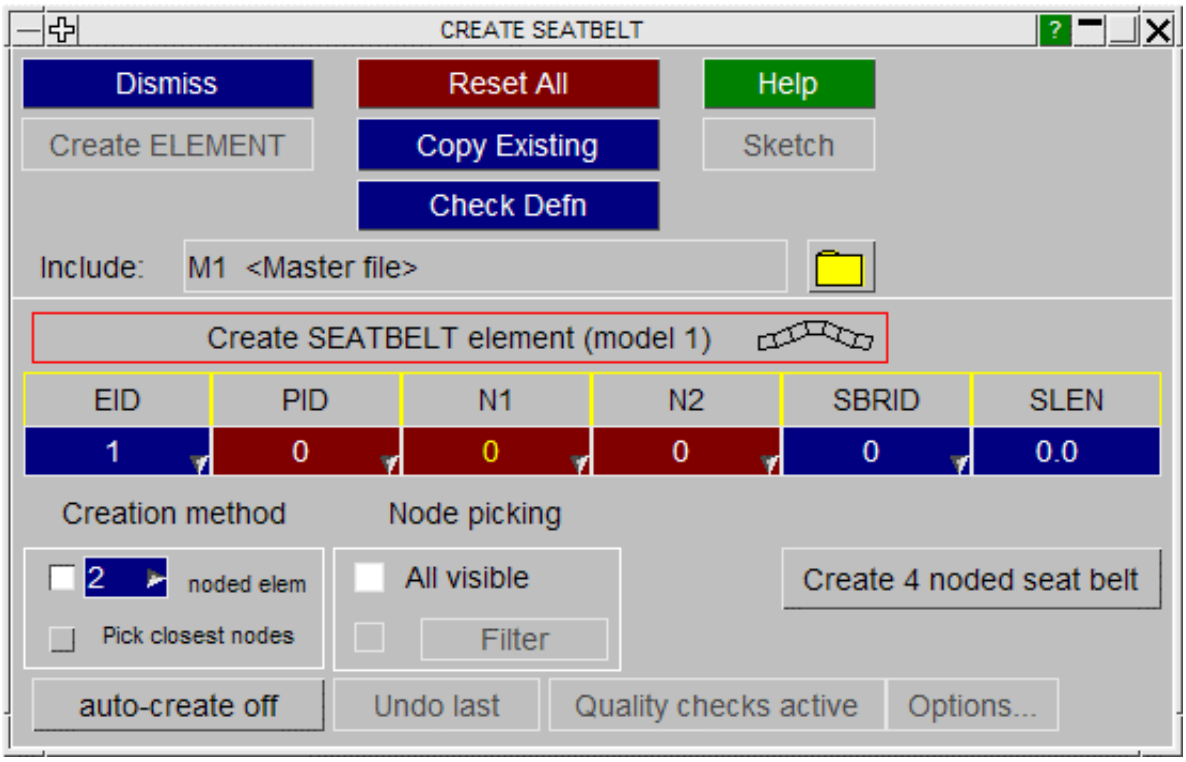
ELEMENT_SEATBELT

This figure shows the seatbelt element creation panel.
Two nodes need to be picked for the seatbelt element. The part number and slack length (**SLEN**) are saved as defaults when an element is created.

There are no element quality checks available for seatbelt elements at present.

This panel is suitable for creation and editing of individual seatbelt elements.

For generating and fitting a line of seatbelt elements to an occupant model PRIMER provides a "seatbelt fitting" capability: see section 6.



Four noded seatbelt elements (introduced in LS971)

The four noded seatbelt elements introduced in LS-DYNA release 971 are in fact shell elements, and share the same numbering sequence as conventional shells, so within PRIMER they are edited under shell elements.

On output they will be written correctly under the *ELEMENT_SEATBELT header. The user can click on **Create 4 noded seat belt** to open up a shell creation panel with the seatbelt option active.

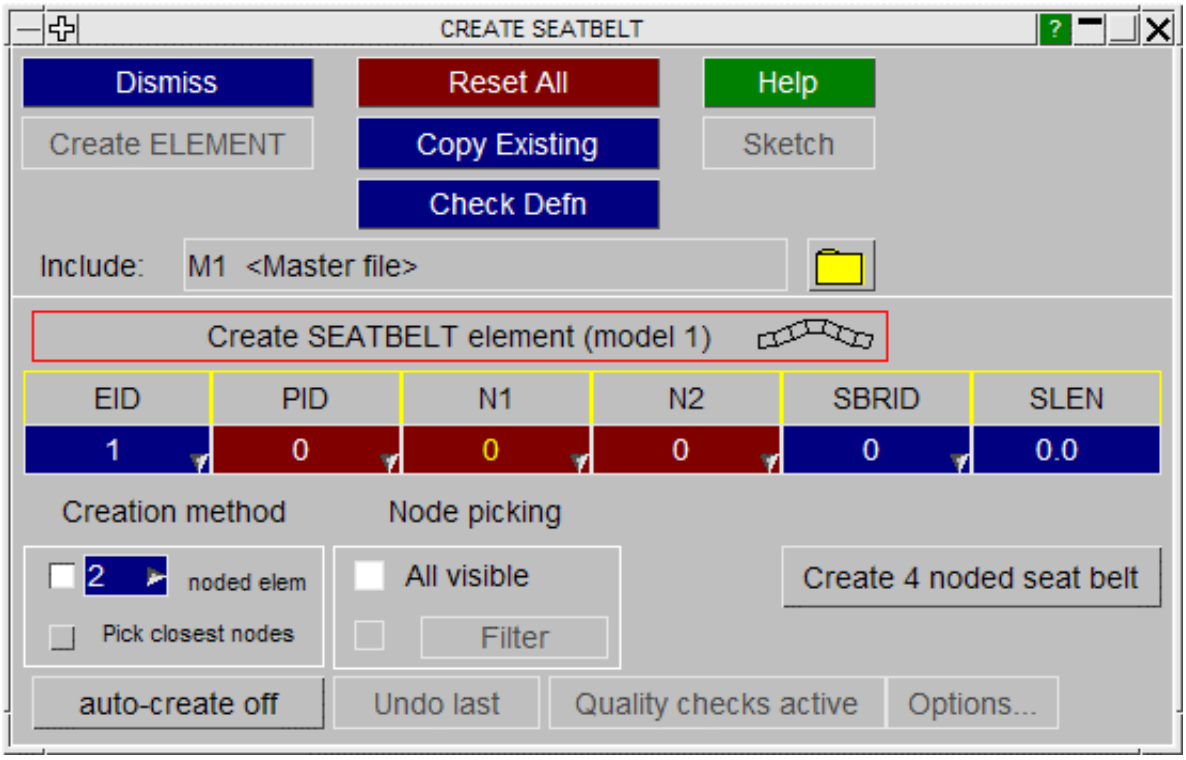
ELEMENT_SEATBELT_ACCELEROMETER

This figure shows the accelerometer create/edit panel.
Three nodes must be selected:

- N1: Origin
- N2 : Local X axis from N1N2
- N3 : Local XY plane from N1N2N3

All three nodes should be on the same rigid part.

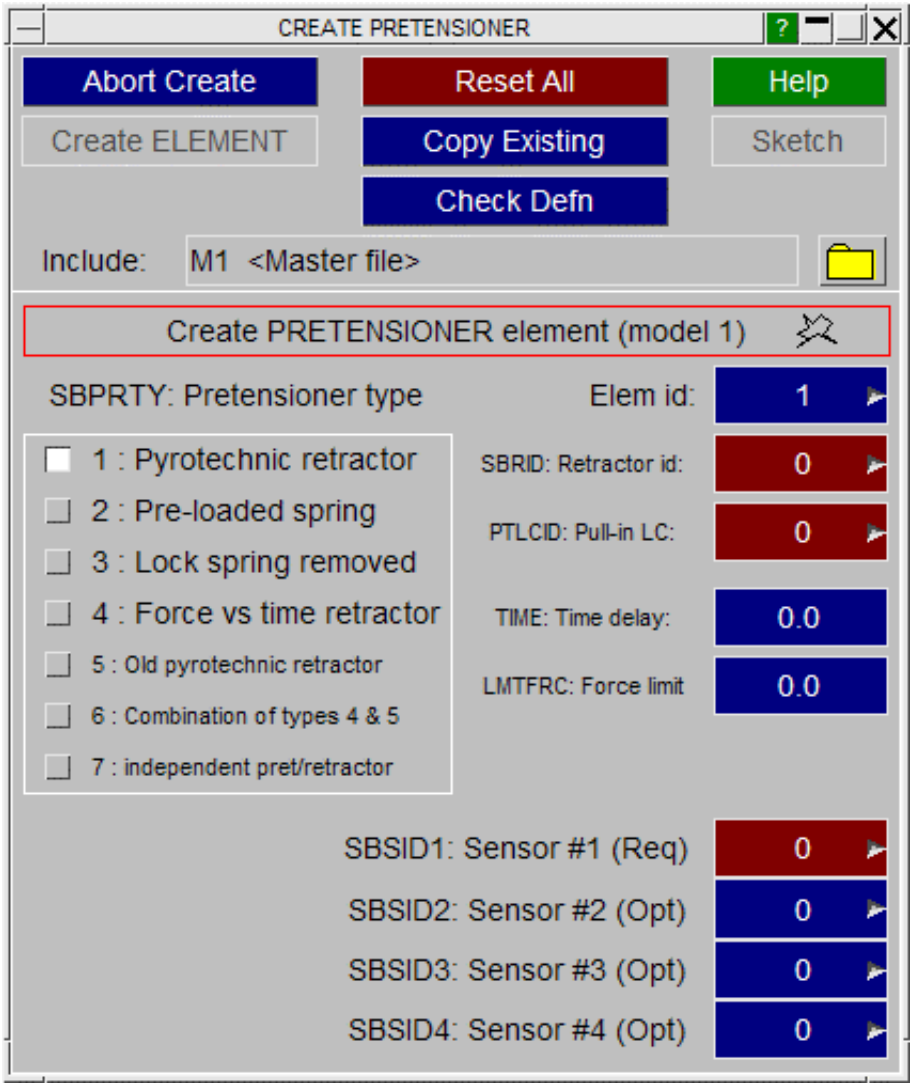
Accelerations will be output in the local coordinate system of the accelerometer.



ELEMENT_SEATBELT_PRETENSIONER

This figure shows the pretensioner create/edit panel.
Pretensioners are active devices that tighten a seatbelt in the event of a crash.

Three different types are provided, which may be triggered by up to four **SENSOR**s.

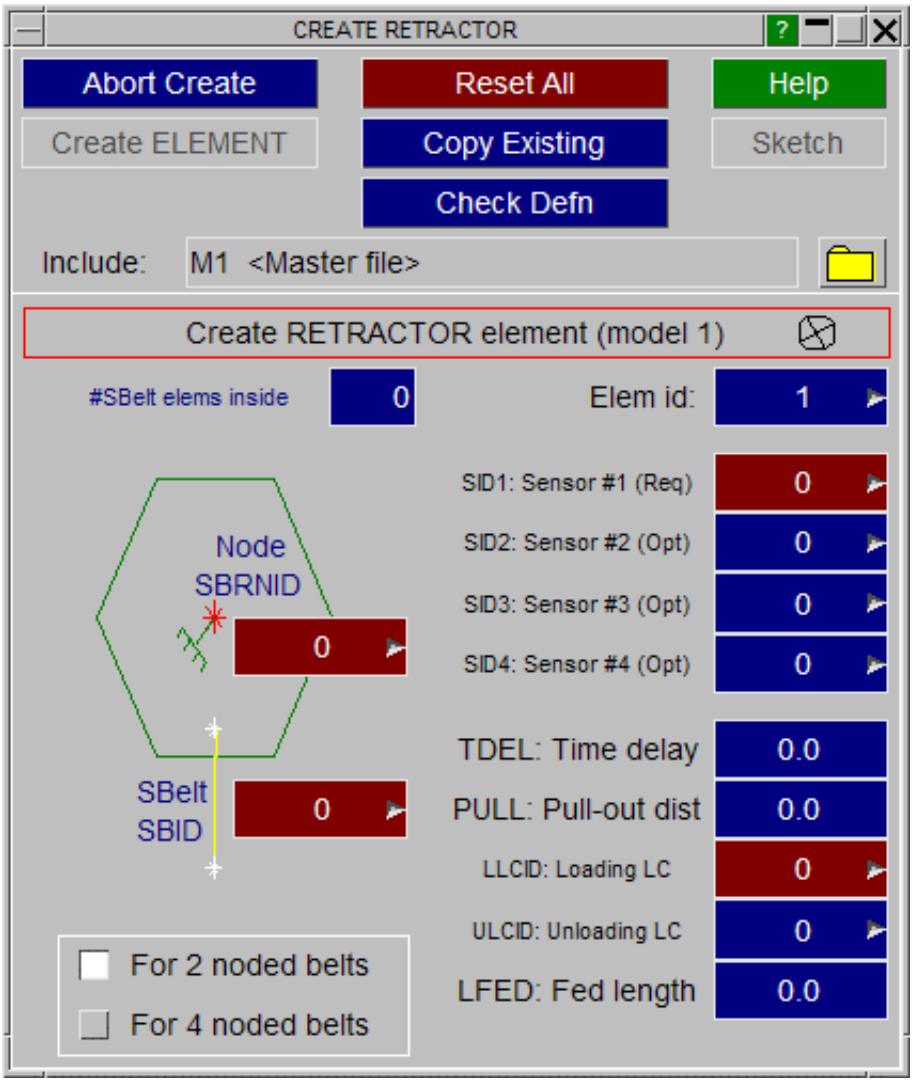


ELEMENT_SEATBELT_RETRACTOR

This figure shows the retractor create/edit panel.
Retractors are the "inertia reel" part of the seatbelt system.

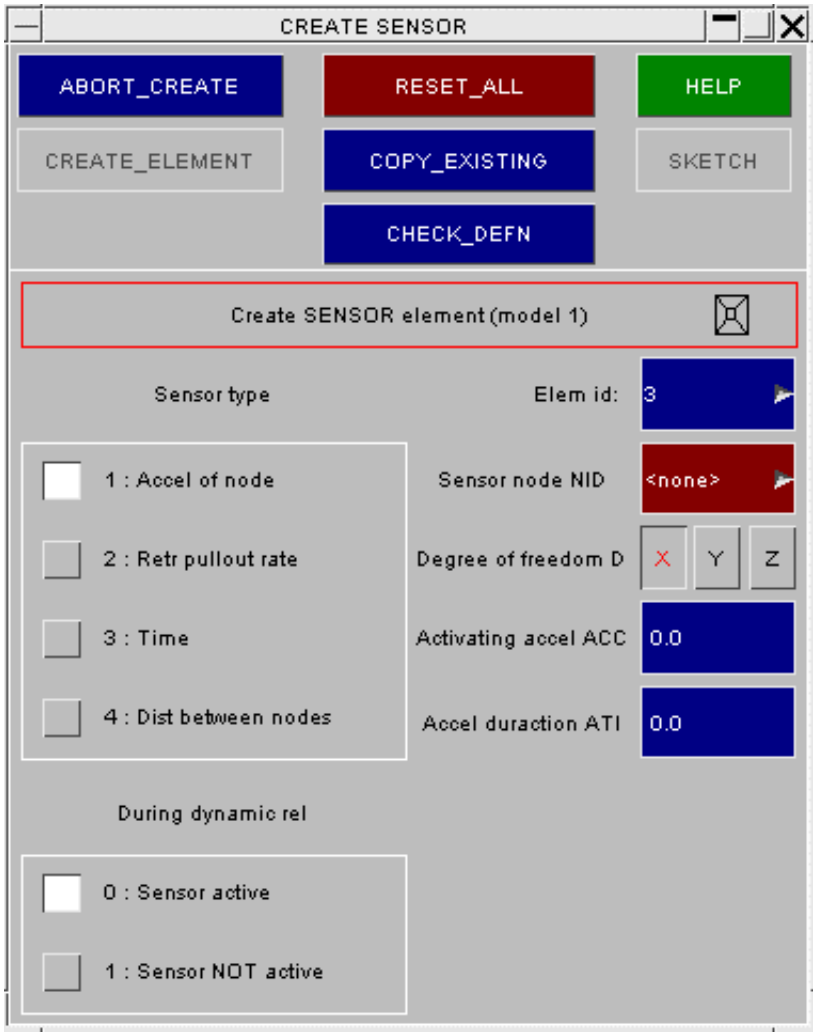
They are assumed to have some number of seatbelt elements curled up inside them, and to apply an constant tension to the belt to take up any slack.

When a collision occurs, denoted by a **SENSOR** activating, they can be programmed to "lock up".



ELEMENT_SEATBELT_SENSOR

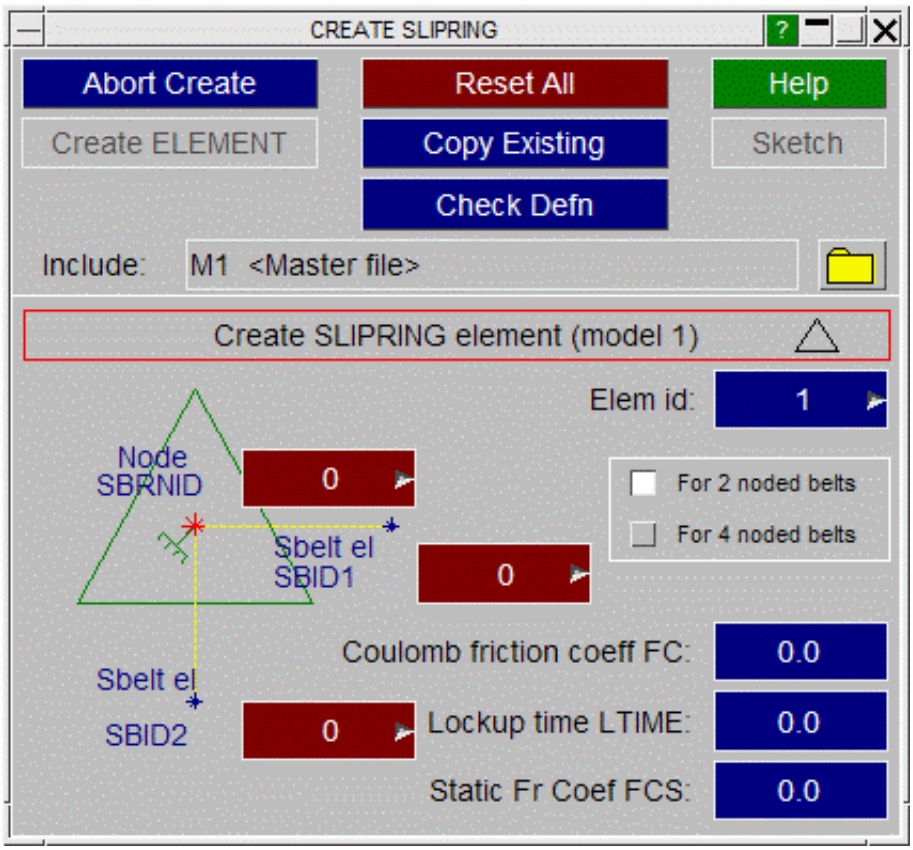
This figure shows the sensor create/edit panel.
Sensors are not really elements in the structural sense (forgive the pun!) of the word. Their role is to detect one of four types of event and then to trigger other **ELEMENT_SEATBELT_xxx** elements.



ELEMENT_SEATBELT_SLIPRING

This figure shows the slipring create/edit panel.
Sliprings permit seatbelt elements to "feed through" from one side to the other, emulating the real behaviour in a crash event.

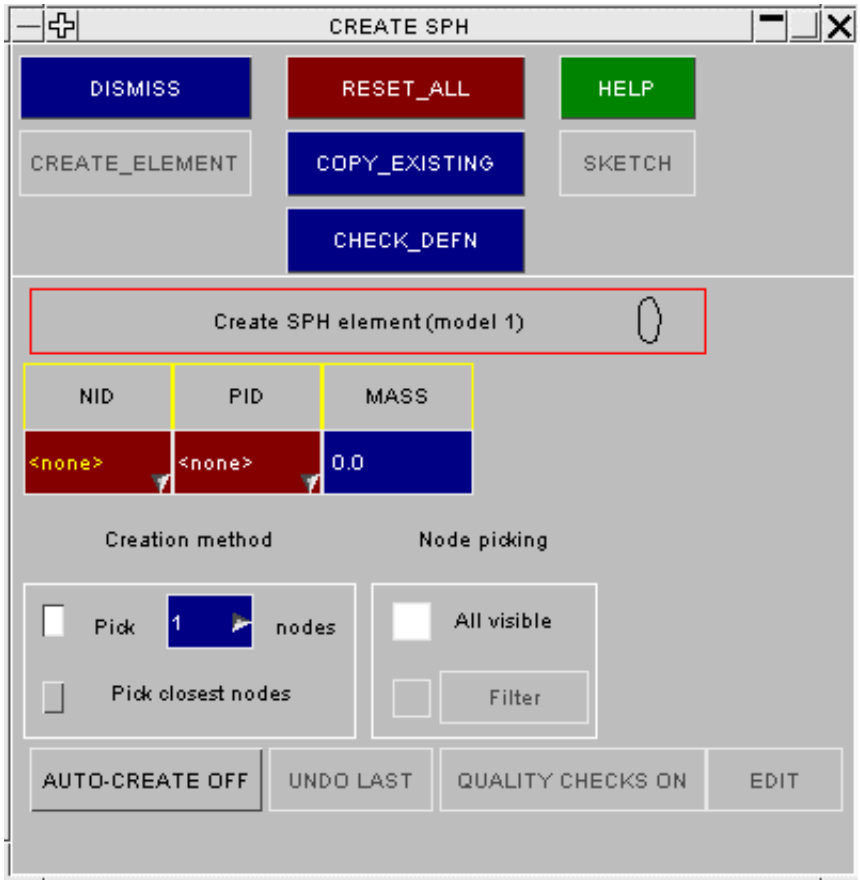
They require two contiguous seatbelt elements **SBID1** and **SBID2** to be defined, whose common node must initially be coincident with the slipring node **SBRNID.**



ELEMENT_SPH

This figure shows the sph element creation panel. This is virtually identical to the mass element except a part also needs to be chosen.
Only 1 node needs to be picked for the sph element. The mass and part number are saved as defaults when an element is created.

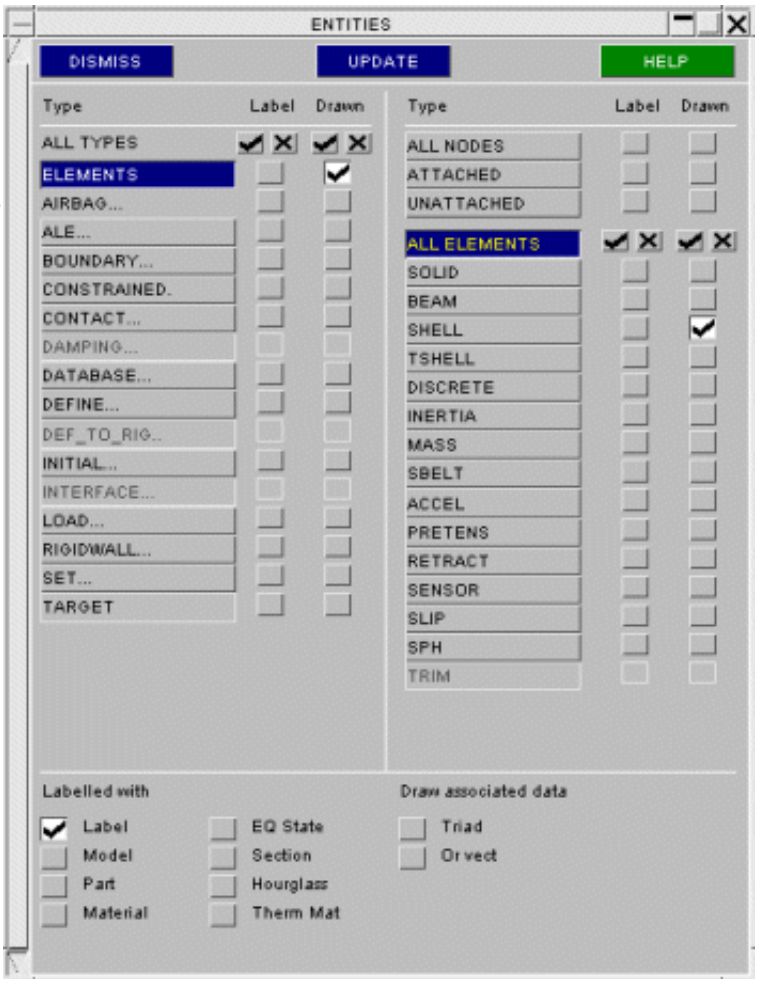
There are no element quality checks available for sph elements at present.



ELEMENT_TRIM

This figure shows the trim element creation panel. This element does not have any node picking associated with it. The only thing that needs to be defined is a part set.
Trim elements are used in conjunction with ***DEFINE_CURVE_TRIM** in metal forming analyses.

Visualisation and labelling of elements

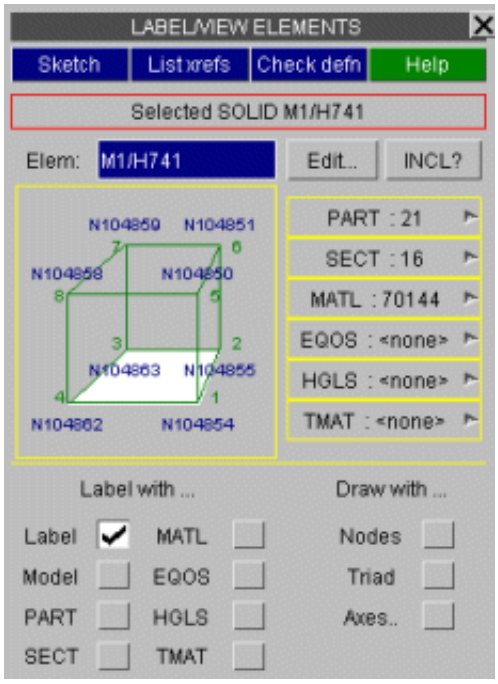


PRIMER draws and labels all element types (except TRIM), display being controlled in the

ENTity Viewing panel. (See section 4.3) Details of an individual element of a specific type can be obtained by clicking on its category in this panel (eg **SOLID**) and then screen-picking or typing in the element label.

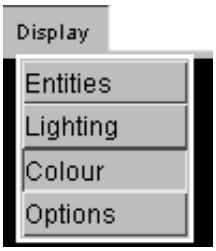
To visualise elements of any type use **ALL ELEMENTS**.

The details panel for a solid element is shown below:



This is accessed either from Quick Pick (click on the element in the graphics window) or by clicking on the word SOLID (or BEAM, SHELL, etc) in the Entities panel and typing in the label of the element.

Controlling the colour in which elements are drawn



The **COLOUR** panel menu controls the display of elements that reference ***PART** cards.

Such elements may be drawn in colours based on:

PART id (default)
MODEL id
INCLUDE file id
SECTION id
MATERIAL id

Colour can also be changed for parts and individual elements using Quick Pick.

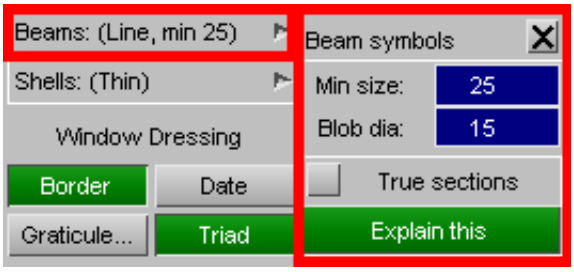
See section 4.1.2 for more detail.

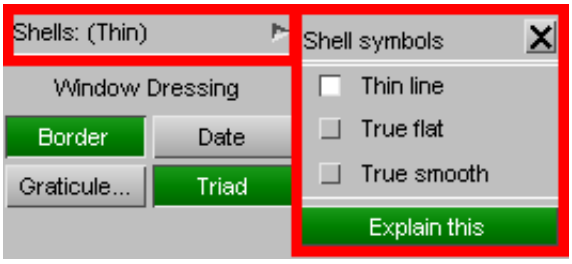
Special "true thickness" display for beams and shells

Normally beams are displayed as lines and (thin) shells as infinitely thin lines or facets, since this is the simplest and fastest way of rendering them.

However both element types have ***SECTION** cards which define their properties, and it can be useful to visualise their true shapes, including any offsets that may be defined.

The **Display Options** panel controls how beams and shells are displayed, permitting their "true" sections and offsets to be drawn.





Data display and contouring

A range of different quantities can be displayed as data contours on elements via the **CT** (Continuous Tone) and **SI** (Shaded Image) commands. The **Vect or plot** command can also display element data.
These are described in section 4.2: Data Plotting Commands.

Special capabilities for seatbelts and related element types

Seat-belt elements can be fitted to occupants using the **Safety**, **Seatbelts** menu

This involves form-finding to establish the line of the belt, a fitting process to pull it onto a dummy, meshing, element property definition and contact creation. It is described in section 6.40SEAT-BELTS Fitting seatbelts and related elements.

EQOS: Equation of State

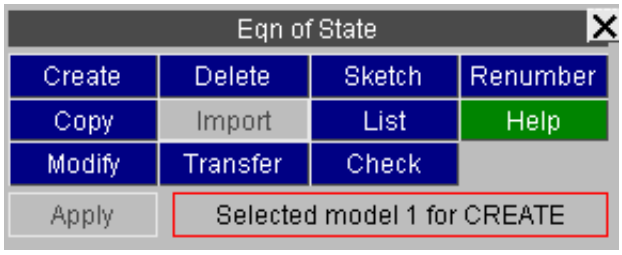
- Top level menu
- Creating
- Copying
- Editing
- Deleting
- Sketch
- List
- Check
- Renumber

Equations of State are used to define material properties for special or typical fluids or null material types. Consequently, they are controlled in a similar fashion to the ***Material** keywords.

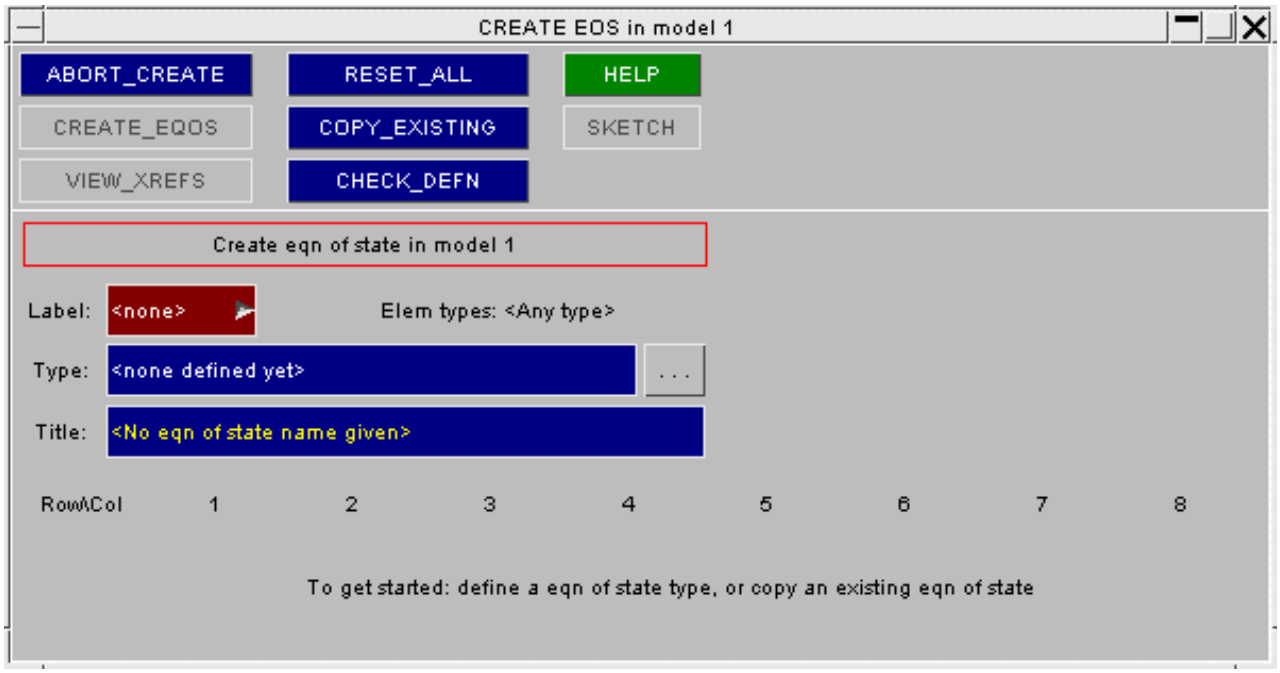
This figure shows the main equation of state editing panel.

TRANSFER opens the main window for the transfer data function (see section 6.7 for more detail)

The other functions currently available have their standard meanings. (See 5.1.1).



CREATE Making a new equation of state definition.



CREATE produces this blank equation of state creation panel, since no equation of state type has been defined yet.

Type:

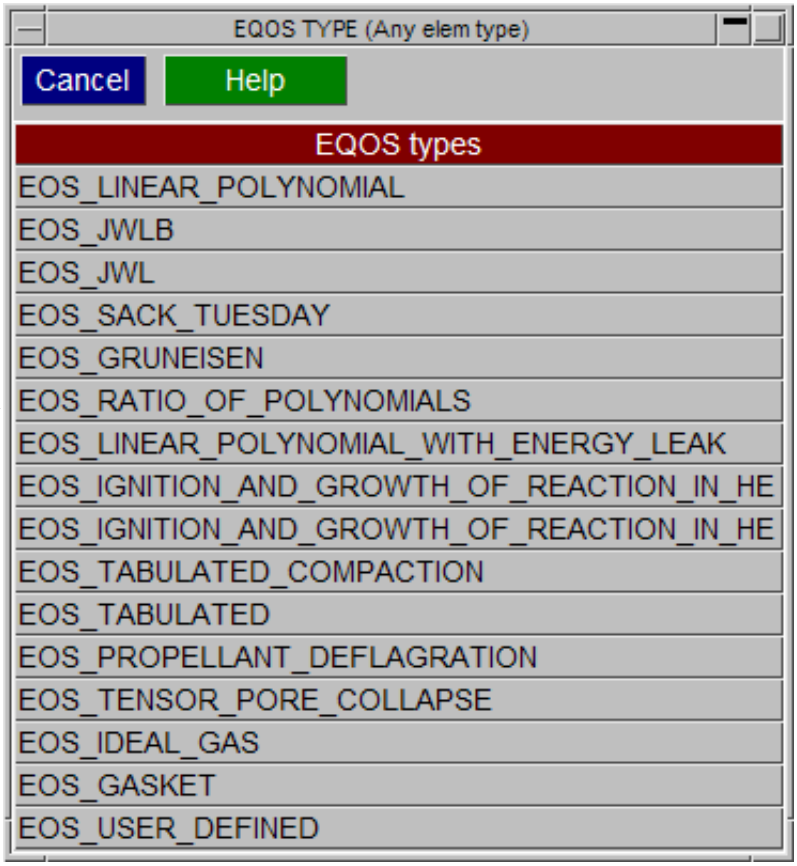
The equation of state type can be defined from this button.

The [...] Shortcut button can be used to browse through a list of equation of state types as shown here.

Note on selecting an Equation of State:

An equation of state may be selected by one of two ways:

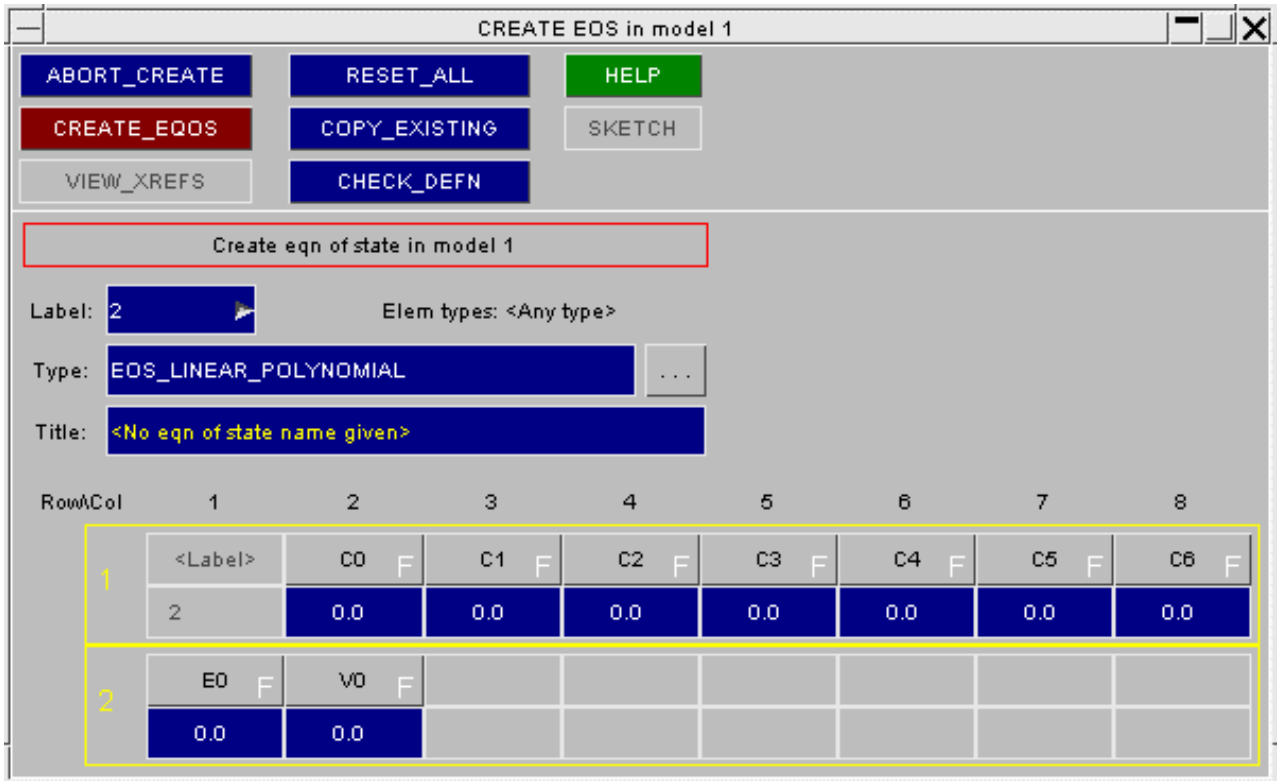
- by invoking the browse [...] button and selecting the equation of state with the mouse from the list
- by typing in the equation of state name to the "Type" box, e.g. "ideal_gas" for "*`EOS_IDEAL_GAS`"



ROW/COL

The data relevant to each equation of state type is displayed in row and column format identical to that of DYNA keyword.

Once a equation of state type has been defined the panel will become populated with that equation of state's format. For example the type ***EOS_LINEAR_POLYNOMIAL** has been chosen here:



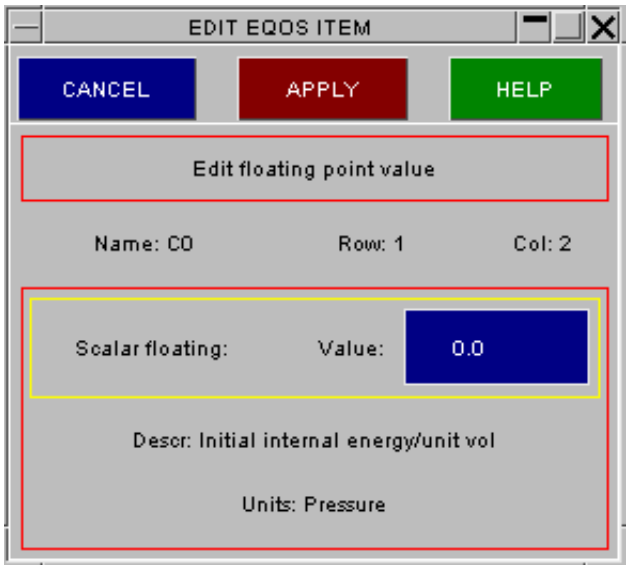
The data can then be typed into the relevant boxes. The expected data type is indicated on the grey button, which also shows the acronym for that data value:

F	White **F** floating value
I	White **I** integer value
+LC	Green **LC** +ve loadcurve
-LC	Red **LC** -ve loadcurve

Information about each individual data component can be requested by pressing the grey data component button. For example; to request information about data component '**C1**' (1st row, 3rd column) press the grey button with the C1.

This will create a new window with detailed information about that data component showing:

- A one-line description of it;
- Its current units type
- Its current value.



Once all of the data has been input , press **CREATE_EQOS** to install the equation of state permanently in the model.

As with other creation/editing functions a standard check is made of the new definition prior to saving it, and you are warned about errors found.

COPY Copy existing equation of state(s) to make a new equation of state(s).

COPY makes new equation of states, in the same model(s), that are identical to their originals apart from their labels. By default only the equation of state definitions themselves are duplicated.

Where **RECURSIVE COPY** is requested, all items associated with that equation of state (i.e. elements, parts, etc.) are also copied.

MODIFY Modifying the attributes of an existing equation of state.

MODIFY functions in the same way as **CREATE**. Obviously, the equation of state will already have been selected so the panel will resemble that shown in "populated" figure above.

If the equation of state is in use by a `PART` which has elements then the "element type" of the equation of state is locked to those elements, which will restrict the range of equation of state types you can change it to. For example a equation of state used by springs cannot be changed to a shell type.

Any modifications made to the equation of state definition will not be made permanent until the **APPLY_MODIFY** button is pressed. At this point a the local copy which has been updated is used to overwrite the version in the model.

DELETE Delete existing equation of state definitions.

The **DELETE** function deletes the selected equation of states. However you cannot delete a equation of state that is in use by a `PART` unless you remove it from the part definition, or delete that too. To help with this the following two switches may be used:

DELETE_RECURSIVE Will select for deletion the parts, associated elements, and so on that reference thisequation of state.

REMOVE_FROM_SETS Is often also needed if parts are to be deleted, as their elements, the connected nodes, and often the parts themselves may be included in sets.

A good way of getting rid of surplus (unused) equation of states is to turn these two switches off, then select all equation of states for deletion. Only those which are not used by anything will actually get deleted.

SKETCH Sketch elements using an equation of state on the current image.

SKETCH sketches on top of the current image the parts and elements that reference the selected equation of states.

LIST Summarise the attributes of selected equation of states.

LIST allows the user to individually select equation of states and display a summary listing of their attributes.

CHECK Check selected equation of states for correctness

CHECK runs the standard checking function on the selected equation of states, summarising any errors.

WARNING: Checking equations of state thoroughly is a mammoth task, which PRIMER does not at present attempt. Most equations of state are currently only checked for a positive density. Therefore that an equation of state"Checks OK" does not mean that it contains no errors!

RENUMBER Renumbering equation of state labels.

RENUMBER lets you change any or all equation of state labels within a given model using the standard renumbering panel.

To change the label of an individual equation of state it may be simpler just to **MODIFY** it.

HOURGLASS: Hourglass control cards.

- **HOURGLASS** top level menu
- Creating a new definition
- Copying a definition
- Editing an existing definition
- Deleting hourglass definitions
- Other operations

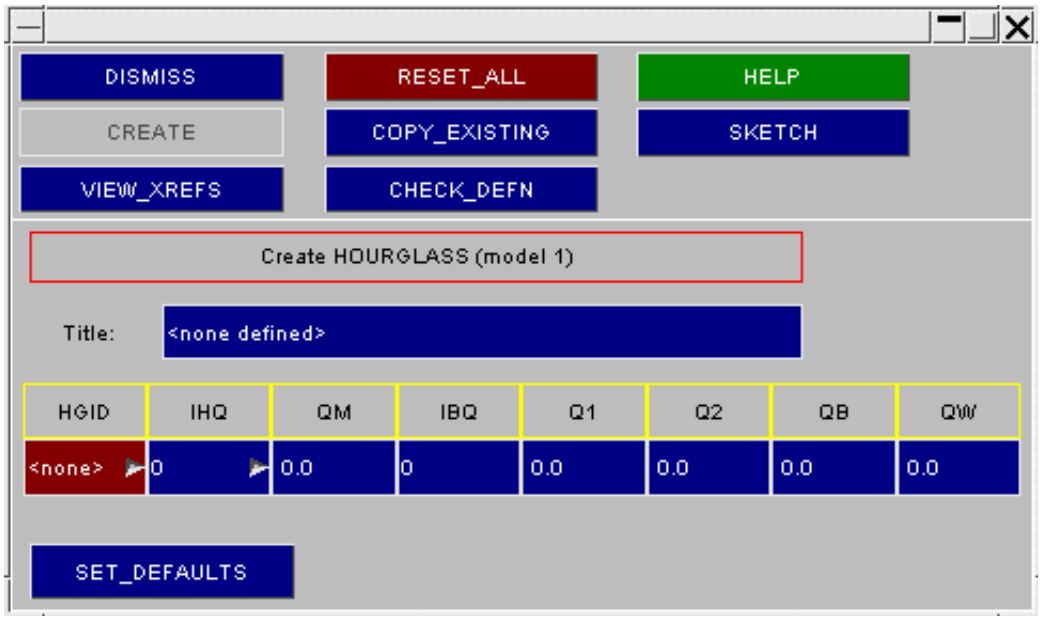
Hourglass cards are used in LS-DYNA to control the zero energy "hourglass modes" that occur with single integration point elements.
They are also used to specify bulk viscosity coefficients.

HOURGLASS MAIN MENU
This figure shows the main **HOURGLASS** menu.

The functions currently available have their standard meanings (see section 5.1.1).

Greyed out functions are not currently available:

CREATE Making a new hourglass definition.



This figure shows the standard **CREATE**/**EDIT** panel for hourglass cards. Here **CREATE** has been used, so a blank hourglass creation panel is displayed.The static buttons in the top section of the panel have functions which are common to the other editing panels within PRIMER.

Once all of the data has been input on the airbag card, **CREATE** installs the hourglass card permanently in the model.

The **SET_DEFAULTS** button will put the **current default values** into the fields. These will be taken from the `CONTROL_HOURGLASS` and the `CONTROL_BULK_VISCOSITY` settings. If the LS-DYNA default settings are active, after pressing the button the values are set thus:

HGID	IHQ	QM	IBQ	Q1	Q2	QB	QW
<none>	1	0.100	1	1.500	0.0600	0.100	0.100

COPY Copy existing hourglass card(s) to make a new card(s).

COPY makes new hourglass, in the same model(s), that are identical to their originals apart from their labels.

RECURSIVE COPY has no effect as hourglass cards do not 'own' anything else.

MODIFY Modifying the attributes of an existing hourglass card.

MODIFY functions in the same way as **CREATE**.

Any modifications made to the hourglass definition will not be made permanent until the **APPLY_MODIFY** button is pressed. At this point a the local copy which has been updated is used to overwrite the version in the model.

DELETE Delete existing hourglass definitions.

The **DELETE** function deletes the selected airbag. The **DELETE_RECURSIVE** and **REMOVE_FROM_SETS** switches have no effect here as hourglass cards do not 'own' anything and are not in sets.

SKETCH Sketch elements used by an hourglass on the current image.

SKETCH sketches on top of the current image the parts and elements that are referenced by the selected hourglass card.

LIST List hourglass cards

LIST gives a list of the selected hourglass cards showing the cross references to each card.

CHECK Check hourglass attributes

CHECK checks one or more hourglass cards for errors.

RENUMBER Renumbering hourglass labels.

RENUMBER lets you change any or all hourglass labels within a given model using the standard renumbering panel.

To change the label of an individual hourglass card it may be simpler just to **MODIFY** it.

DISMISS terminates hourglass processing.

Visualising hourglass cards

Hourglass cards are not explicitly drawn, or selectable for drawing. To view an hourglass definition:

- **SKETCH** it, or
- **MODIFY** it and sketch it.

It will be drawn in terms of the parts and elements that use it.

Note that Hourglass data can also be created or modified using PART TABLE.

INCLUDE: Include files

Include files are closely related to model handling, and are described in section 3.13 Include Files

INITIAL: Defining Initial Conditions.

Initial conditions may be defined for a range of items with LS-Dyna.

- Selecting the *INITIAL sub-keyword
- Editing panels
- Create/Edit panel for *INITIAL_VOLUME_FRACTION_GEOMETRY
- Create/Edit panel for *INITIAL_VELOCITY_GENERATION
- Visualisation

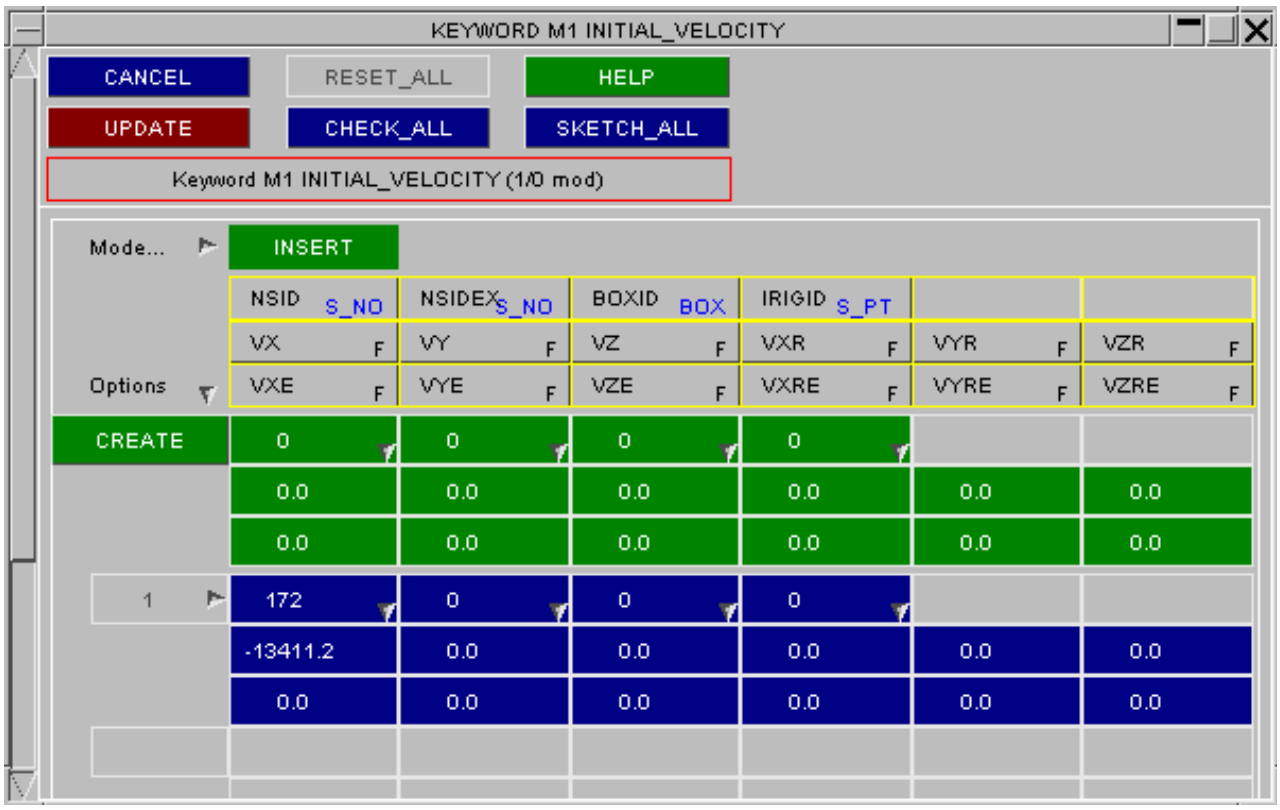
INITIAL

ALE_MAPPING	(0)
AXIAL_FORCE_BEAM	(0)
CFD	(0)
DETONATION	(0)
EXCESS_PWP	(0)
FIELD_SOLID	(0)
FOAM_REF_GEOM...	(0)
GAS_MIXTURE	(0)
INTERNAL_DOF_SOLID	(0)
MOMENTUM	(0)
PWP_DEPTH	(0)
PWP_NODAL_DATA	(0)
STRAIN_SHELL	(0)
STRAIN_SOLID	(0)
STRESS_BEAM	(0)
STRESS_DEPTH	(0)
STRESS_SECTION	(0)
STRESS_SHELL	(0)
STRESS_SOLID	(0)
STRESS_TSHELL	(0)
TEMPERATURE	(0)
TIED_CONTACT_DATA	(0)
VEHICLE_KINEMATICS	(0)
VELOCITY	(0)
VELOCITY_NODE	(0)
VELOCITY_GENERATION	(0)
VEL_GEN_START_TIME	(0)
VELOCITY_RIGID_BODY	(0)
VOID	(0)
VOLUME_FRACTION	(0)
VOL_FRAC_GEOM...	(0)

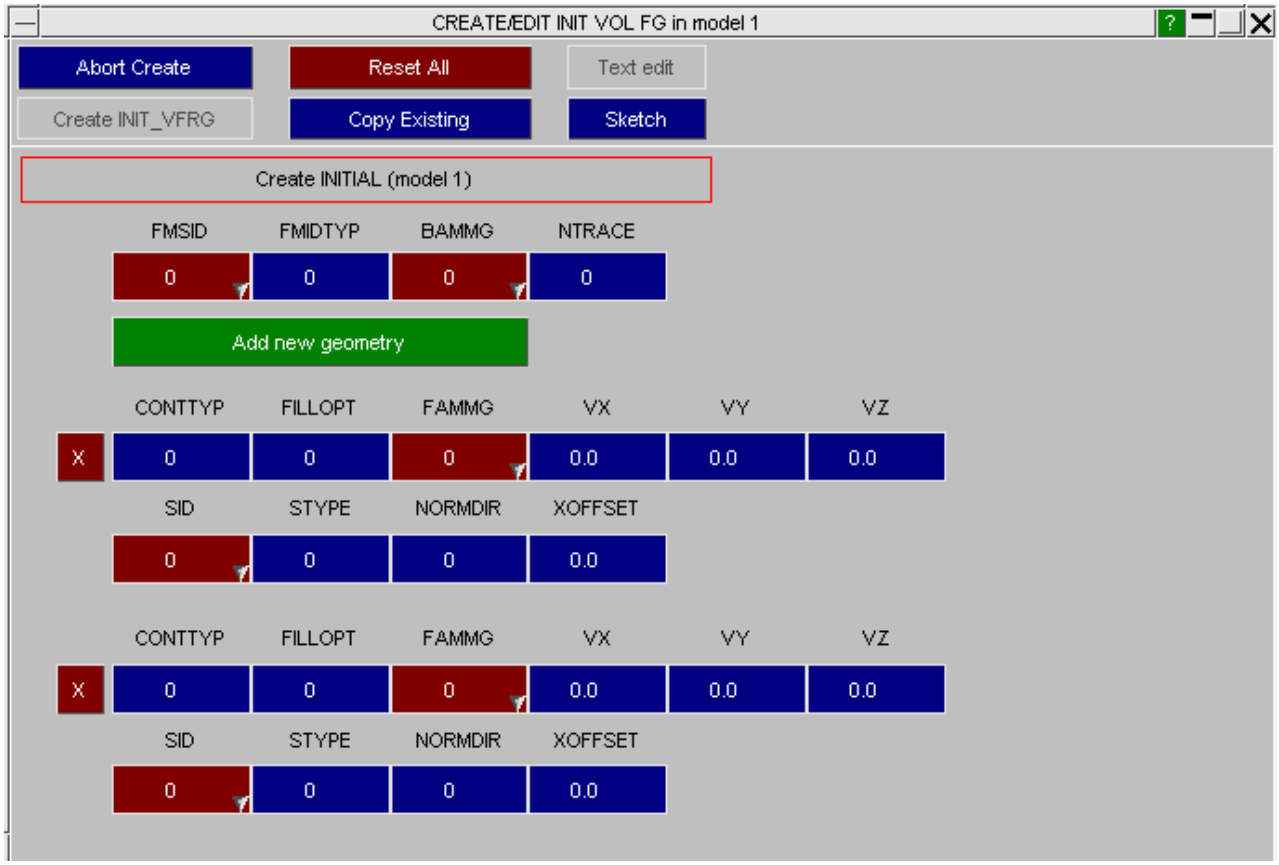
*INITIAL cards can be edited with the generic "Keyword" editor. All ***INITIAL** subtypes except **VOLUME_FRACTION_GEOMETRY** and **VELOCITY_GENERATION** (which have their own specific Create/Edit panel) can be processed in this way.

The other commands (**COPY, DELETE,** ...) function in the standard fashion as defined in section 5.1.1.

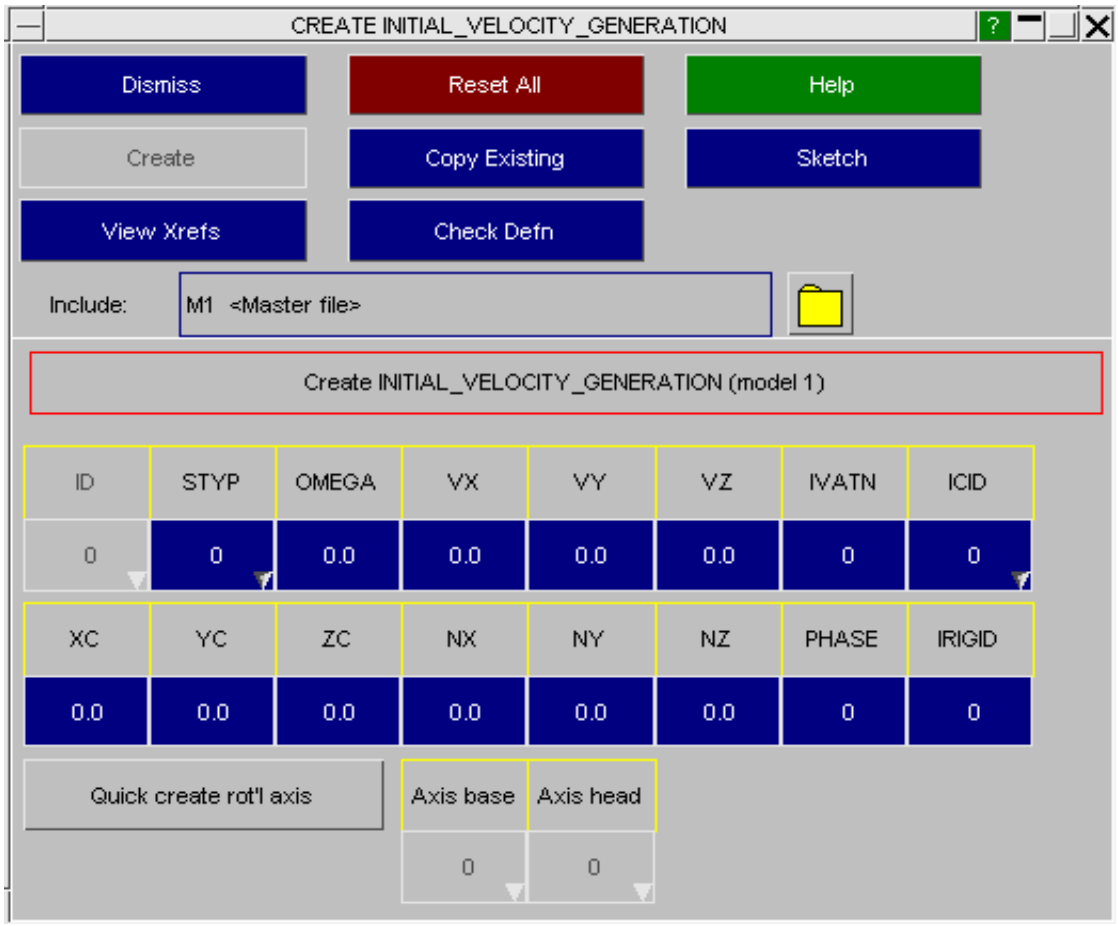
This shows an example of the Keyword editor for ***INITIAL_VELOCITY**



*`INITIAL_VOLUME_FRACTION_GEOMETRY` items are created using a specific editing panel. New geometries can be added by clicking on the **Add new geometry** button.



*`INITIAL_VELOCITY_GENERATION` items are created using a specific editing panel.

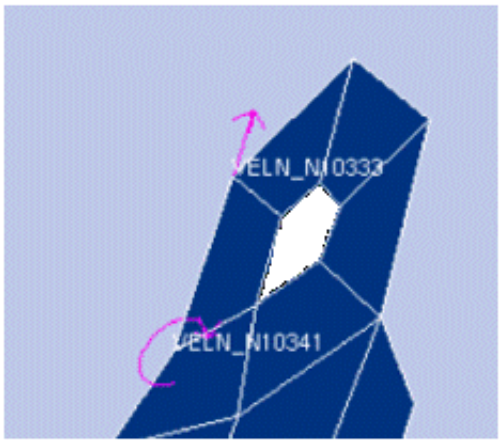

CREATE INITIAL_VELOCITY_GENERATION
Dismiss
Reset All
Help
Create
Copy Existing
Sketch
View Xrefs
Check Defn
Include:
M1 <Master file>
Create INITIAL_VELOCITY_GENERATION (model 1)
ID
STYP
OMEGA
VX
VY
VZ
IVATN
ICID
0
0
0.0
0.0
0.0
0.0
0
0
XC
YC
ZC
NX
NY
NZ
PHASE
IRIGID
0.0
0.0
0.0
0.0
0.0
0.0
0
0
Quick create rot'l axis
Axis base
Axis head
0
0

Visualising *INITIAL items

*`INITIAL` items are not displayed by default, but can be selected for display and labelling in the **ENT**ity Viewing panel.
At present only *`INITIAL_VELOCITY`(_<type>) cards are drawn fully.

These are shown as arrows in the direction of the velocity. In this example there is a translational velocity at node 10333, and a rotational one at 10341.

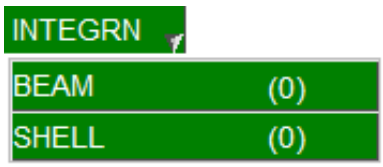
If multiple initial velocities are defined at a node (an error) then all will be drawn separately.



Using the VEC plotting mode to display initial velocities
Initial velocities from any source (not just *`INITIAL` cards) can also be visualised in "vector contour" form via the **Vect plot** command.

Other *`INITIAL` items are not drawn explicitly, but can still be visualised by turning on (in **ENT**ity Viewing) the display of the items they reference: sets, nodes, elements, etc. Labels referring to the initial velocity type are generated correctly.

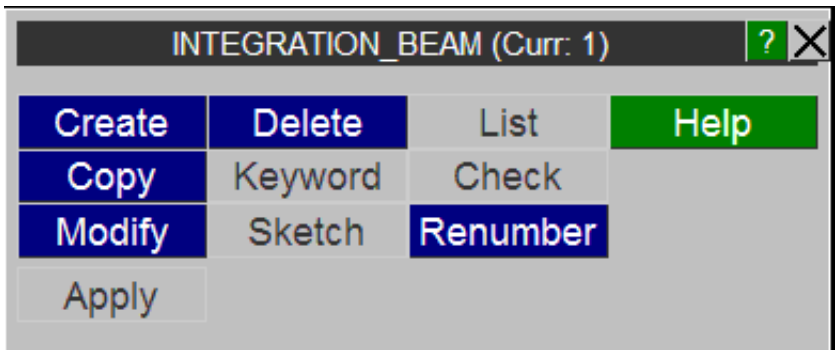
INTEGRATION: Defining Integration rules.



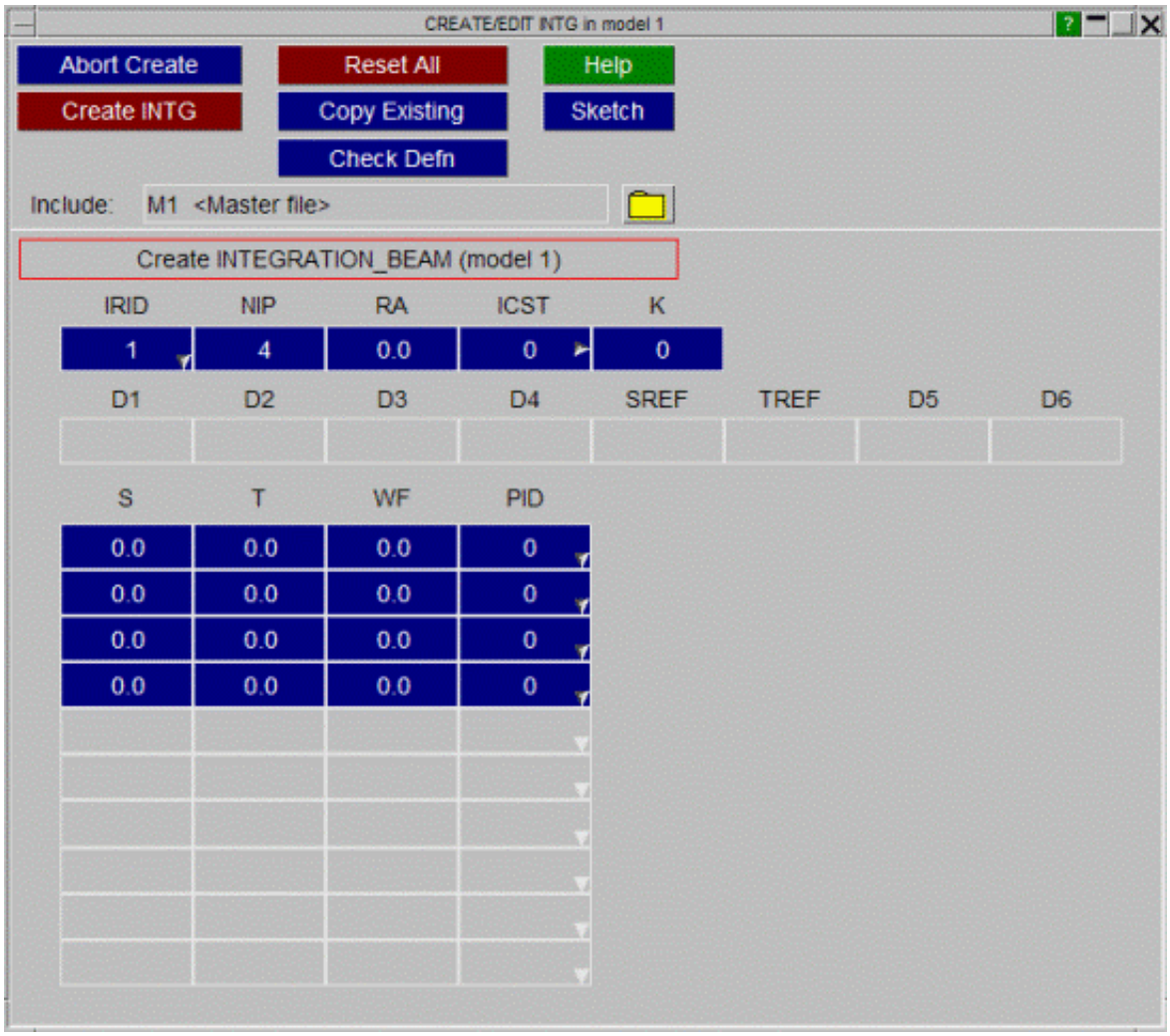
*INTEGRATION_BEAM
*INTEGRATION_SHELL

Select the integration rule to create from the keyword main panel, and the standard sub-menu will appear below.

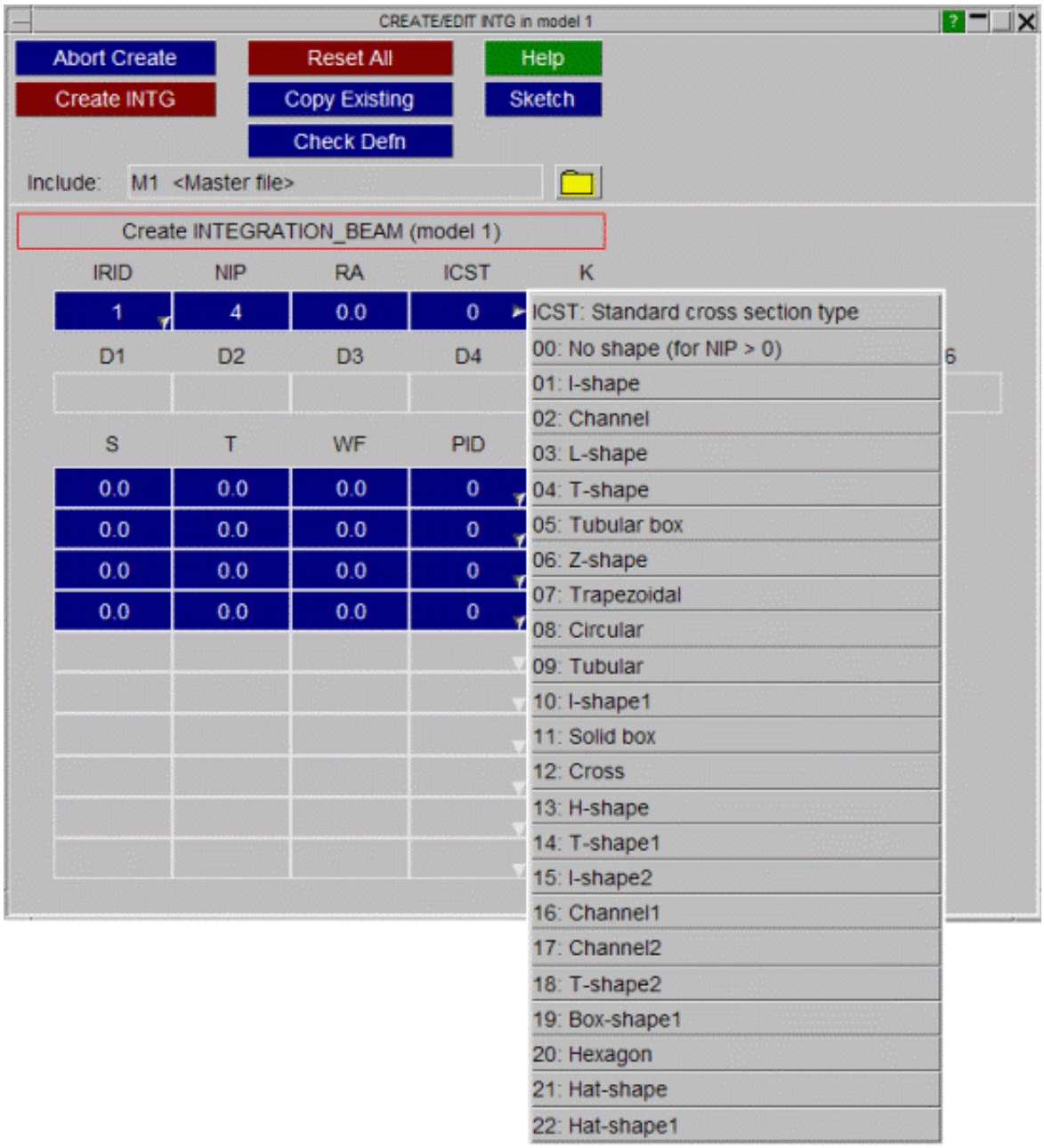
*INTEGRATION_BEAM



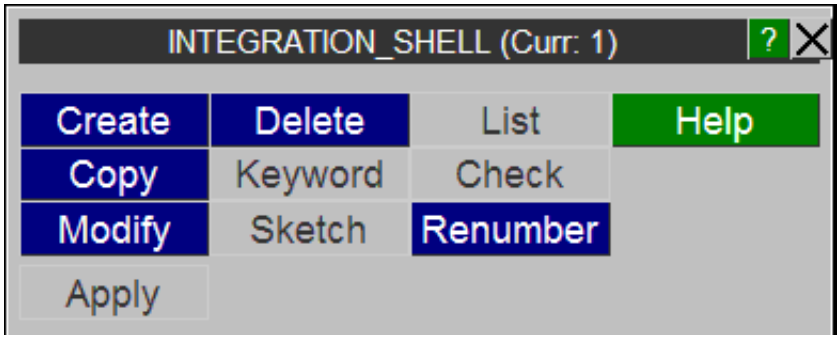
Enter the field to define the NIP for the INTEGRATION_BEAM, and ICST will be set to zero and the rows for each layer will appear below.



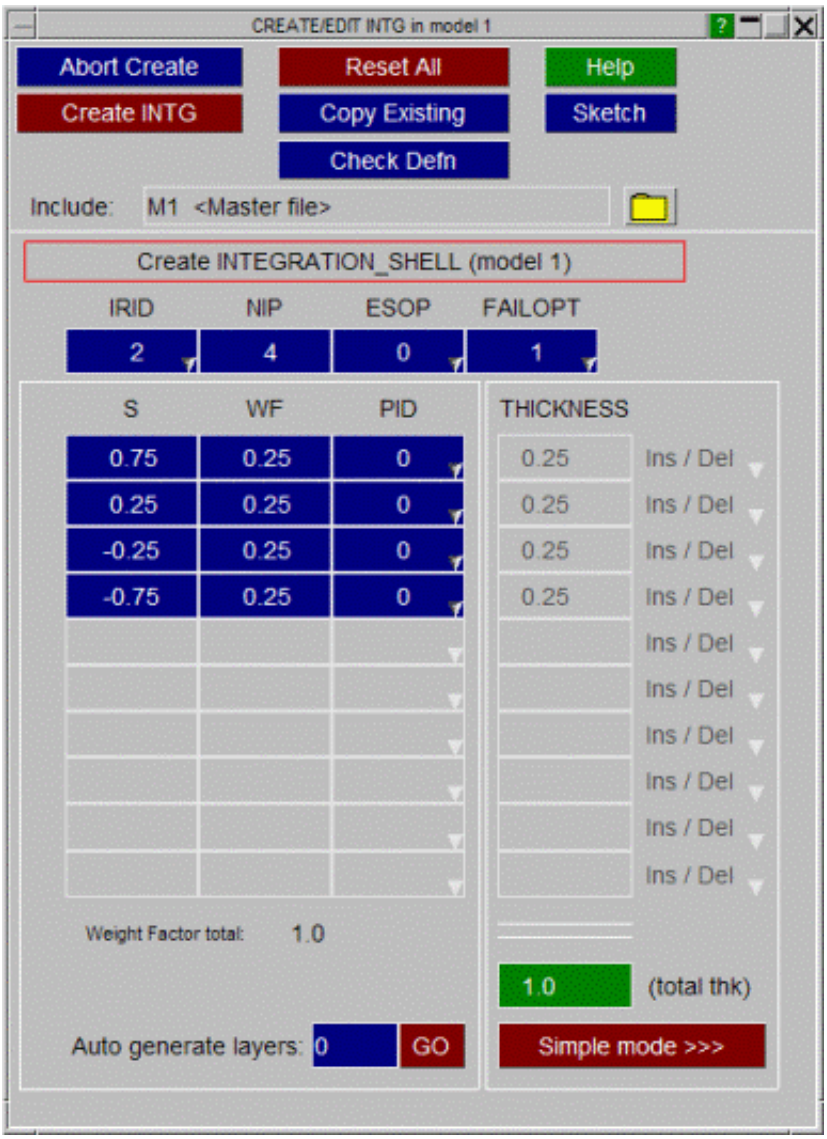
If the user wants to have a predefined shape, select the ICST type from the popuip and the NIP will be set to zero automatically.



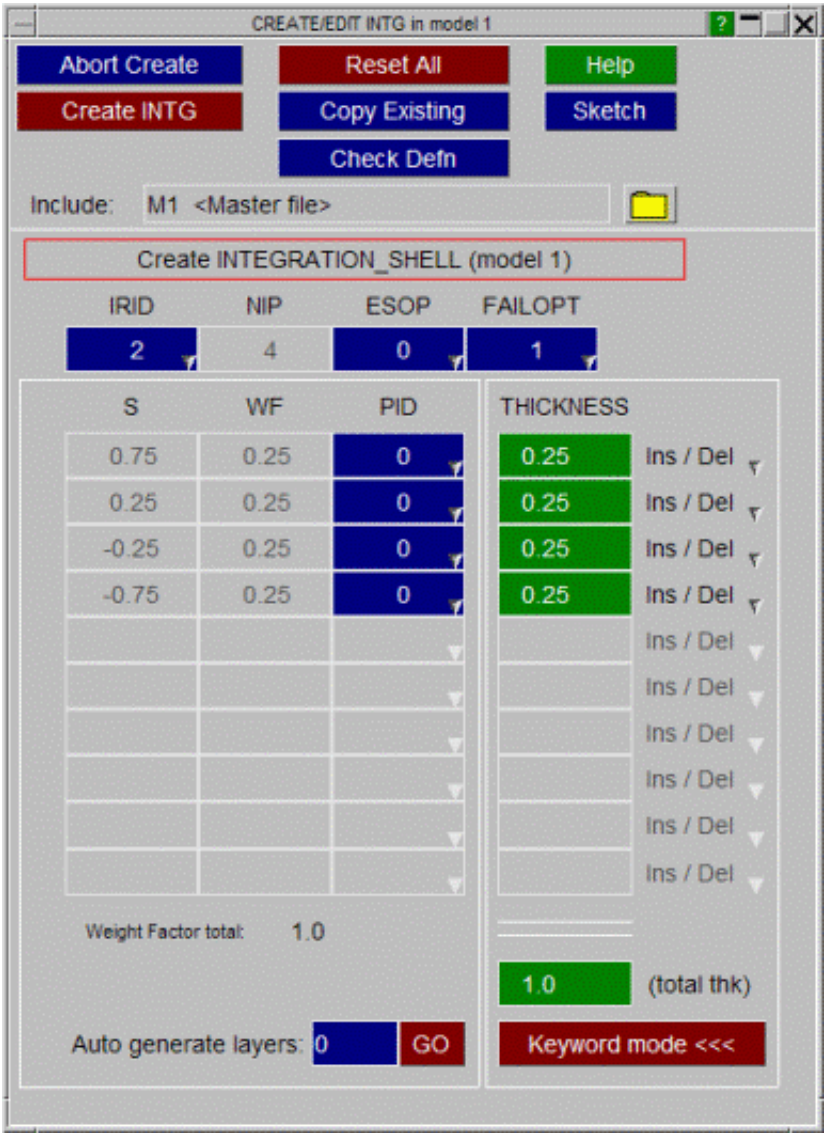
*INTEGRATION_SHELL



The standard panel is below, the user has to define the NIP and in each row below the weighting factor and coordinate have to be input manually. There is an **auto-generate layers** option that generates equally spaced layers for the user.



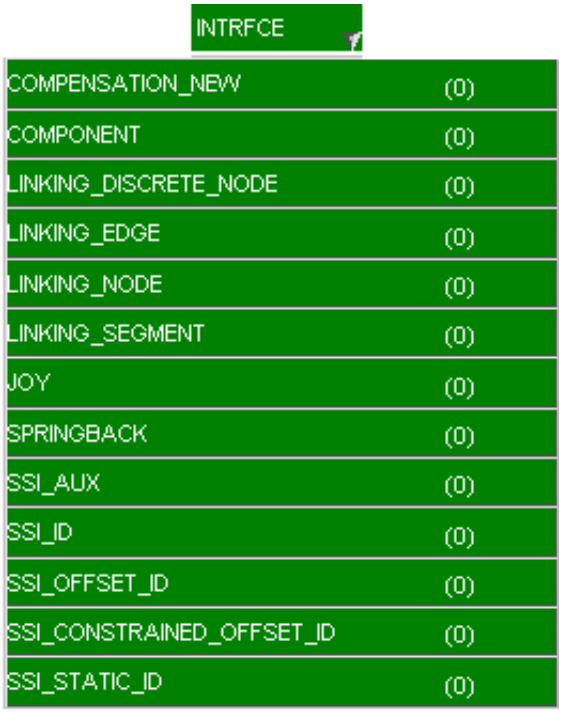
If the user wishes to have a more intuitive input panel, then select the **simple mode >>>** button. The following panel is displayed. It allows the user to define an overall thickness and insert or remove layers through the popups shown. The layers can be defined with thicknesses rather than a coordinate, and the panel automatically calculates the coordinate and weighting factor. Select **keyword mode <<<** to return to the basic panel.


CREATE/EDIT INTG in model 1
Abort Create
Reset All
Help
Create INTG
Copy Existing
Sketch
Check Defn
Include:
M1 <Master file>
Create INTEGRATION_SHELL (model 1)
IRID
NIP
ESOP
FAILOPT
2
4
0
1
S
WF
PID
THICKNESS
0.75
0.25
0
0.25
Ins / Del
0.25
0.25
0
0.25
Ins / Del
-0.25
0.25
0
0.25
Ins / Del
-0.75
0.25
0
0.25
Ins / Del
Ins / Del
Ins / Del
Ins / Del
Ins / Del
Ins / Del
Ins / Del
Weight Factor total:
1.0
1.0
(total thk)
Auto generate layers:
0
GO
Keyword mode <<<

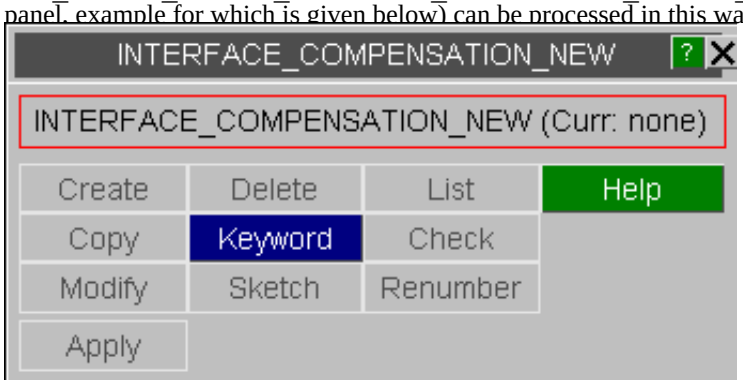
INTERFACE: Defining Interface Cards.

- Selecting the *INTERFACE sub-keyword
- Editing panels
- Special create/edit panel for **SSI_ID**

The ***INTERFACE** keyword has 13 sub-categories.
These can be selected from the drop down list available under **INTRFCE** keyword listing.

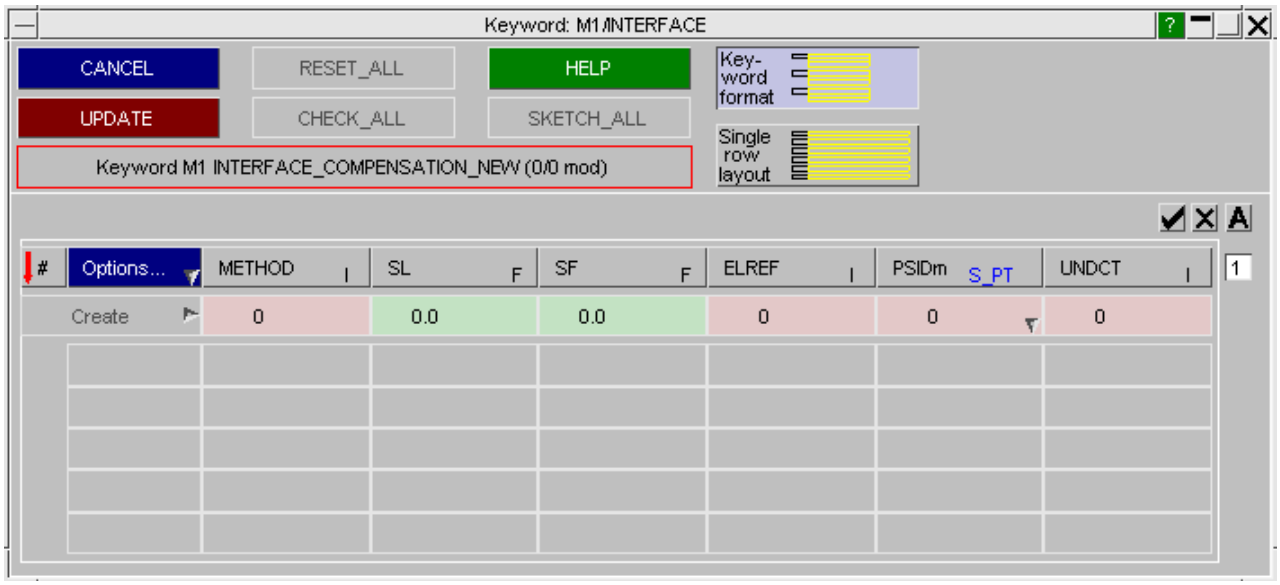


***INTERFACE** cards can be edited with the generic "Keyword" editor. All ***INTERFACE** sub-keywords except **SSI_ID**, **SSI_OFFSET_ID** and **SSI_CONSTRAINED_OFFSET_ID** (which have their own specific Create/Edit panel, example for which is given below) can be processed in this way.



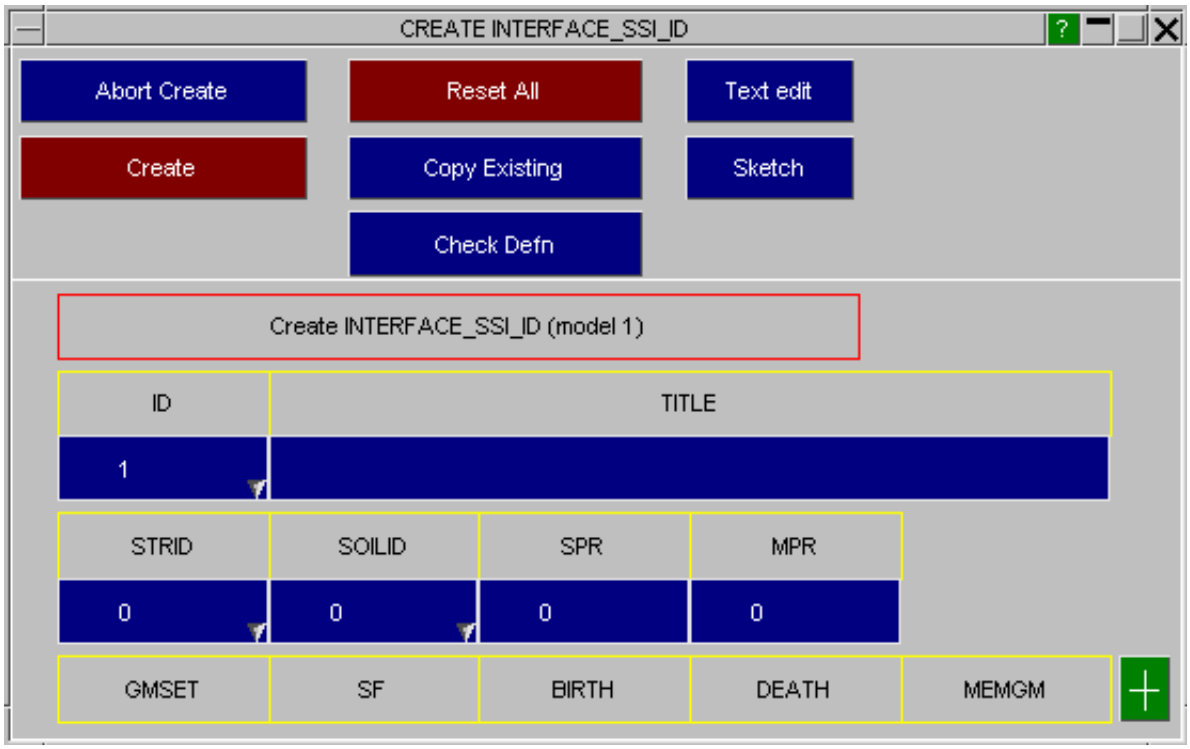
The other commands (**COPY, DELETE,** ...) function in the standard fashion as defined in section 5.1.1..

This shows an example of the Keyword editor for ***INTERFACE_COMPENSATION_NEW**.



INTERFACE_SSI_ID

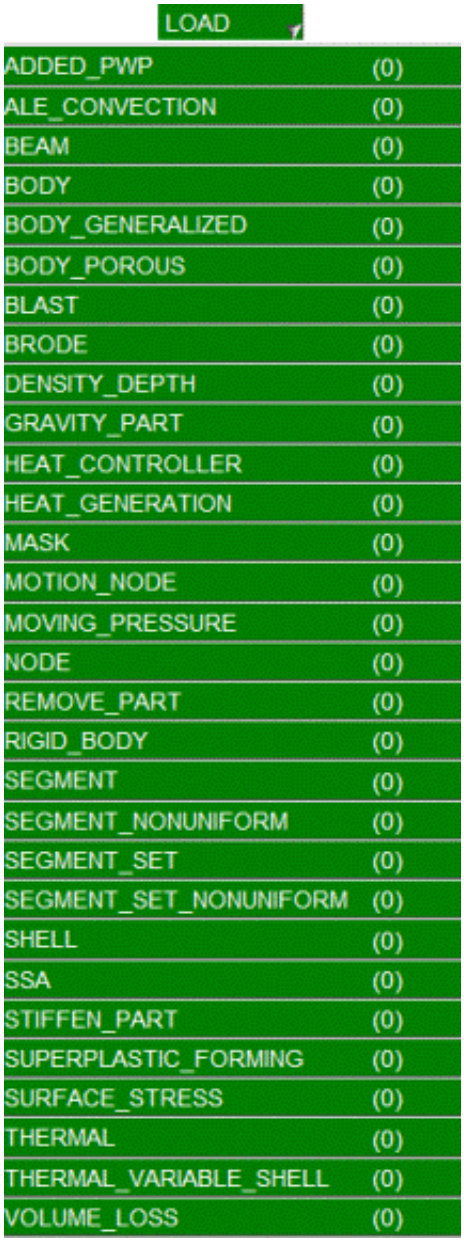
***INTERFACE_SSI_ID** items are created using a specific editing panel. New data rows can be added by clicking on the button.



LOAD: Defining Loading Conditions.

- Selecting the *LOAD sub-keyword
- Visualisation

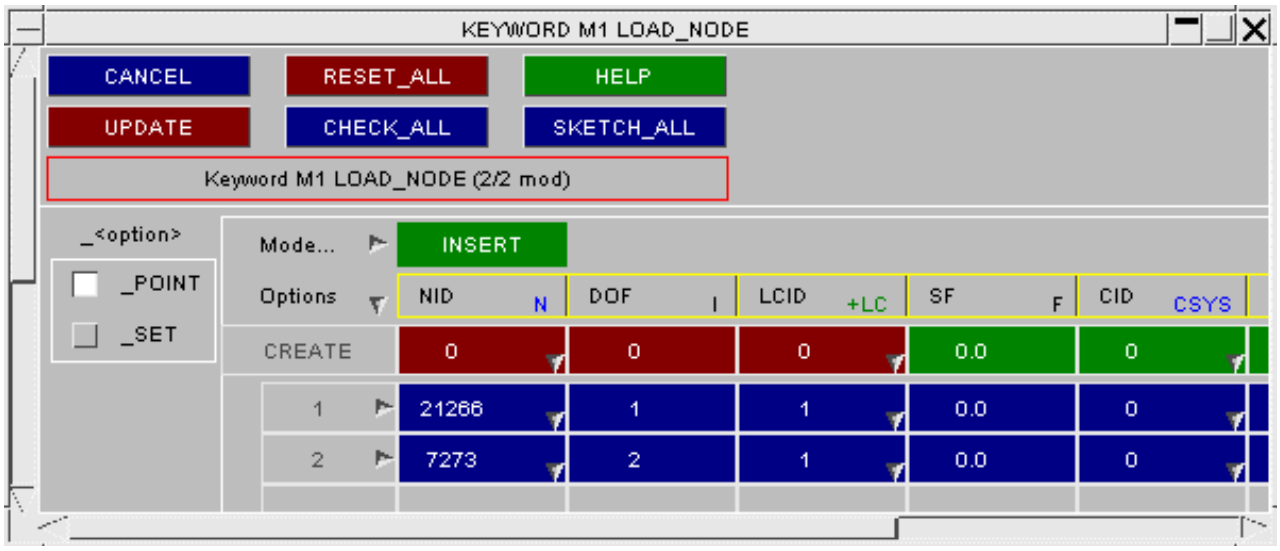
A range of different loading types can be defined in LS-Dyna
All *LOAD sub-keywords are editable within PRIMER.



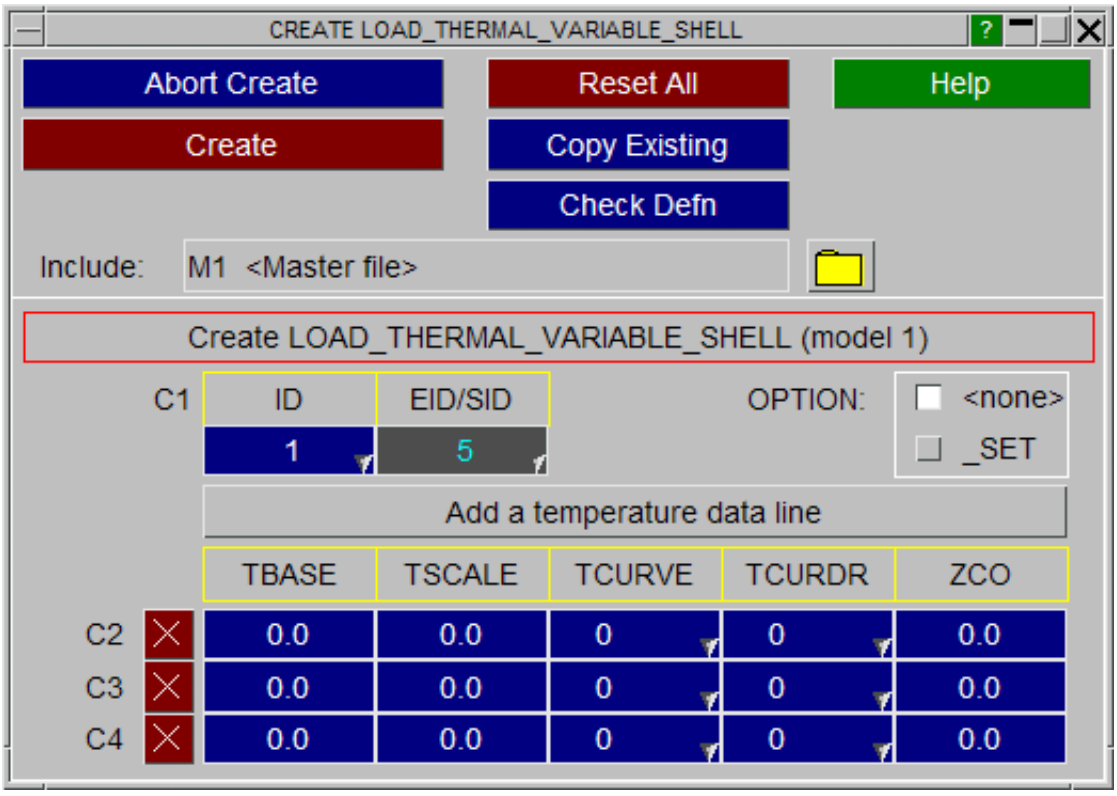
Most ***LOAD** cards can be edited only with the generic "Keyword" editor.

The other commands (**COPY, DELETE,** ...) function in the standard fashion described in section 5.1.1.

This shows an example of the Keyword editor for ***LOAD_NODE**. The **_POINT** variant has been chosen here.



Specific editing panels exist for ***LOAD_MOVING_PRESSURE** and ***LOAD_THERMAL_VARIABLE_SHELL** due to the nature of those keywords. These cards are similar. They both allow the user to add any number of additional lines of data per card. The ***LOAD_MOVING_PRESSURE** card is shown here.



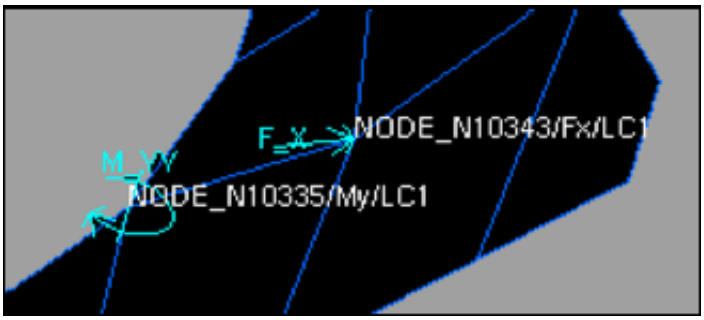
Additional rows of data can be added by clicking on the **Add a data line** button.

Visualising *LOAD items

***LOAD** items are not displayed by default, but can be selected for display and labelling in the **ENT**ity Viewing panel.

Only *LOAD_NODE symbols are drawn explicitly as arrows acting in the direction of the load/moment. Note that load symbols can be distinguished from *INITIAL_VELOCITY ones both by colour and because loads point to a node whereas velocities point away from it.

This example shows a force in X acting on Node 10343, and a moment about the YY axis acting on node 10335.



Other *LOAD sub-keywords are visualised only in terms of the sets, segments, elements or nodes upon which they act by turning on these additional items in ENTity Viewing. Labels are generated correctly for all sub-types.

MATERIAL: Defining Structural and Thermal Materials.

Structural Materials

- Top level menu
- Creating
- Copying
- Editing
- Deleting
- Importing
- Visualisation
- MAT_USER rules
- Encrypted materials

Thermal Materials
- Top level menu
- Editing

The **`*MATERIAL`** keyword in LS-DYNA includes both structural and thermal materials, however they are treated separately within PRIMER since their roles are different:

- Structural materials are required for all structural analyses. (In keyword format all types of structural material, including those for discrete and seatbelt elements, form part of the same numbering sequence. The distinction between "structural", "discrete" and "seatbelt" material numbering found in formatted input decks does not exist.

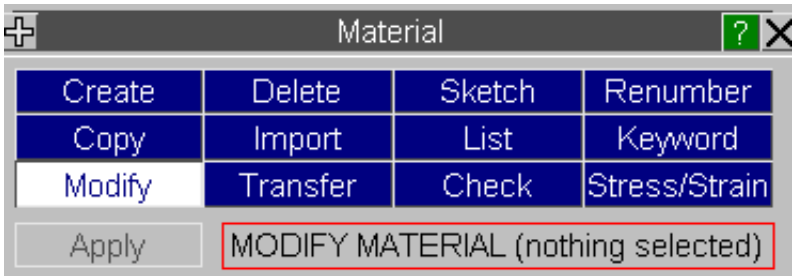
- Thermal materials are used for "thermal only" analyses, and may also be defined for combined structural and thermal analyses.

  Structural and thermal materials use separate labelling sequences, so it is possible to have both structural material #1 and thermal material #1.

STRUCTURAL MATERIALS

The structural material editing functions allow all structural material types to be processed.

PRIMER does not draw materials directly, but material labels may be appended to the graphics of structural items, see **VIS_1**; and Parts, Elements, etc may be selected by material for subsequent graphics operations such as **SKETCH** and **BLANK**



This figure shows the main structural material editing panel.
IMPORT permits material definitions to be "imported" from databases of material definitions to populate undefined materials in a model.

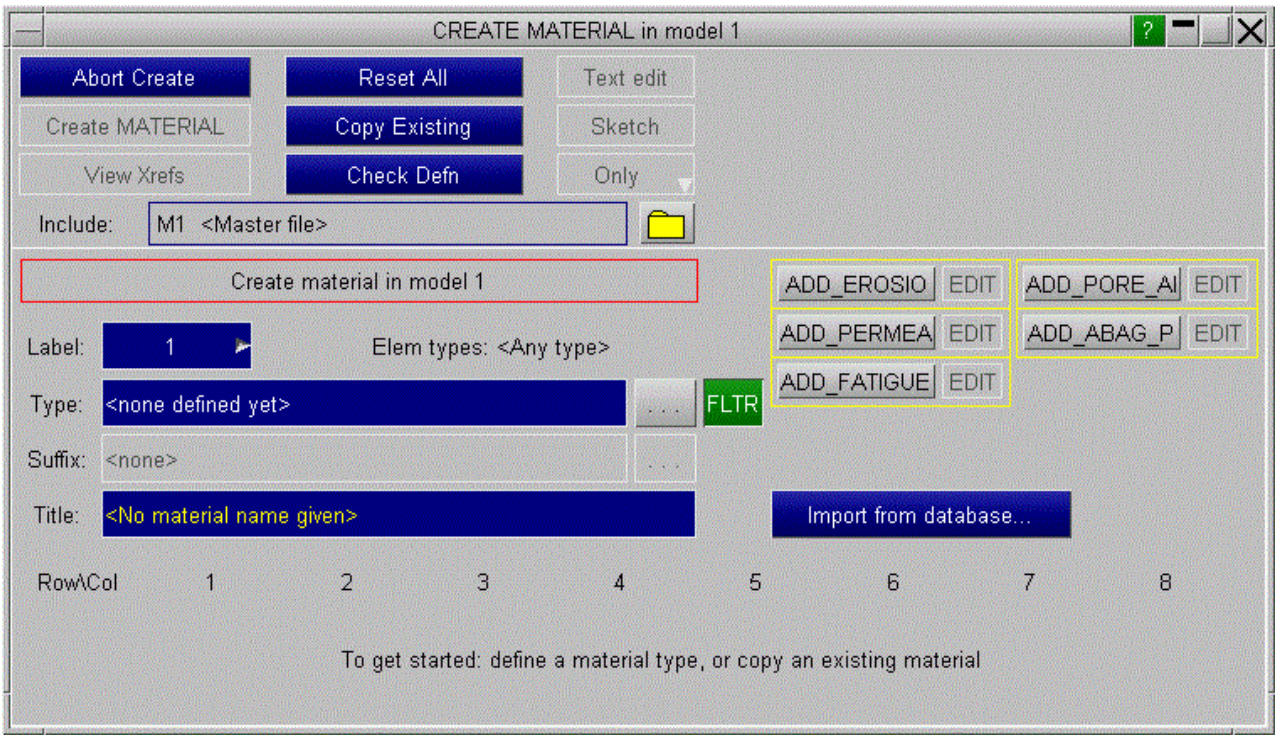
TRANSFER opens the main window for the transfer data function (see section 6.7 for more detail)

STRESS/STRAIN is used to visualise stress strain curves for a material.

The other functions currently available have their standard meanings. (See 5.1.1).

Switch to thermal materials toggles the editing panel (for more detail on thermal materials see below).

CREATE Making a new material definition.



CREATE produces this blank material creation panel, since no material type has been defined yet.

Elem types:

Indicates the types of elements which are applicable to the currently defined material type.

This is a brand new material definition with no previous context, therefore **<Any type>** is shown. Had this material been created from a **PART** of known element type the relevant type would be shown here, and only materials valid for this element type would be selectable.

Type:

The material type can be defined from this button.

The [...] browse button can be used to browse through a list of material types as shown here.

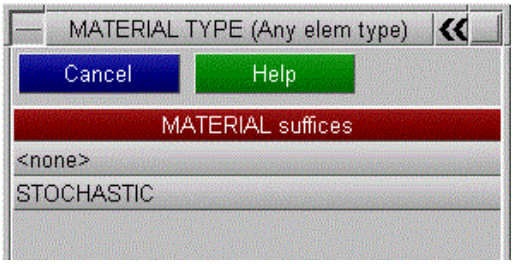
Each material is listed with its LS-dyna material number.

Note on selecting a Material:

A Material may be selected by one of three ways:

- by invoking the shortcut button and selecting the material with the mouse from the list
- by typing in the material number to the "Type" box, e.g. "1" for *`MAT_ELASTIC`
- by typing in the material name to the "Type" box, e.g. "rigid" for "*`MAT_RIGID`"

Suffix:



Underneath the material type you can define the suffix.

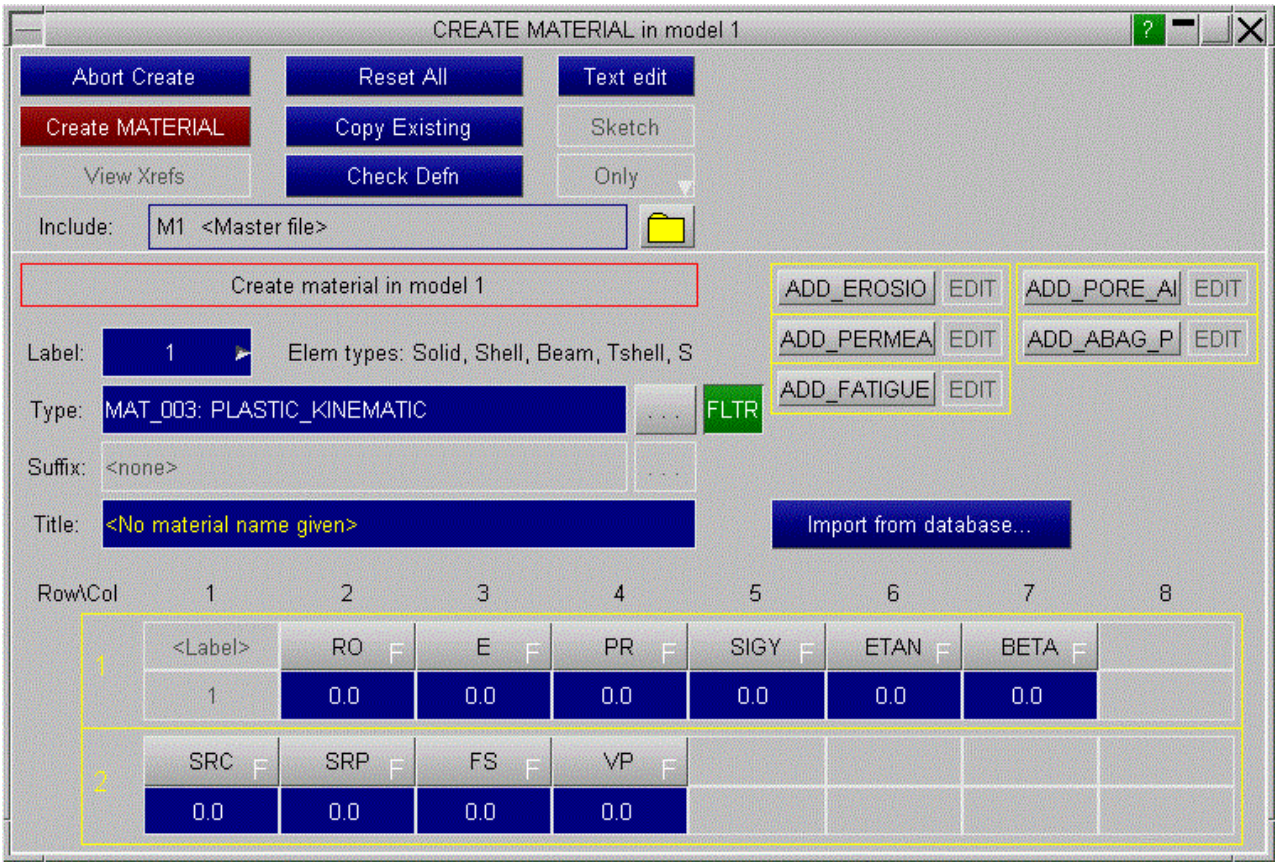
There are two ways of selecting suffices:

- by browsing with the [...] button
- by typing in the suffix name to the "Suffix" box, e.g. "stochastic" for the "*_`STOCHASTIC`" suffix

ROW/COL

The data relevant to each material type is displayed in row and column format identical to that of DYNA keyword.

Once a material type has been defined the panel will become populated with that material's format. For example the type ***MAT_PLASTIC_KINEMATIC** has been chosen here:



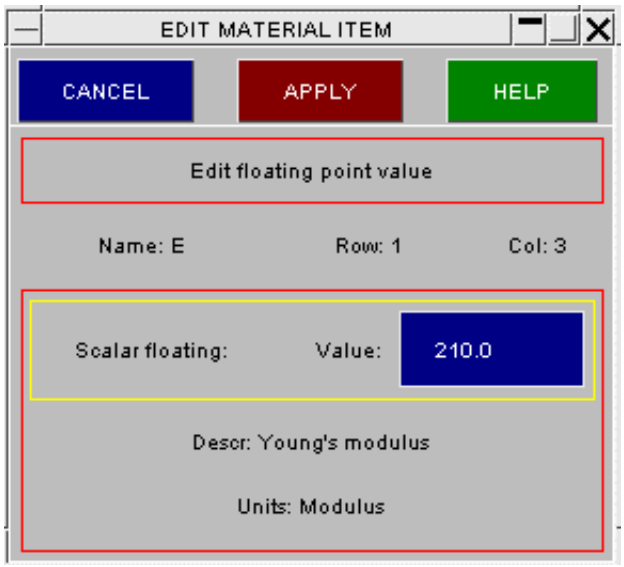
The material data can then be typed into the relevant boxes. The expected data type is indicated on the grey button, which also shows the acronym for that data value:

F	White **F** floating value
I	White **I** integer value
+LC	Green **LC** +ve loadcurve
-LC	Red **LC** -ve loadcurve

Information about each individual data component can be requested by pressing the grey data component button. For example; to request information about data component '**E**' (1st row, 3rd column) press the grey button with the E.

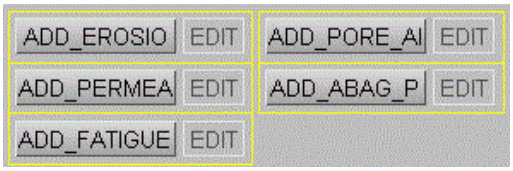
This will create a new window with detailed information about that data component showing:

- A one-line description of it;
- Its current units type
- Its current value.



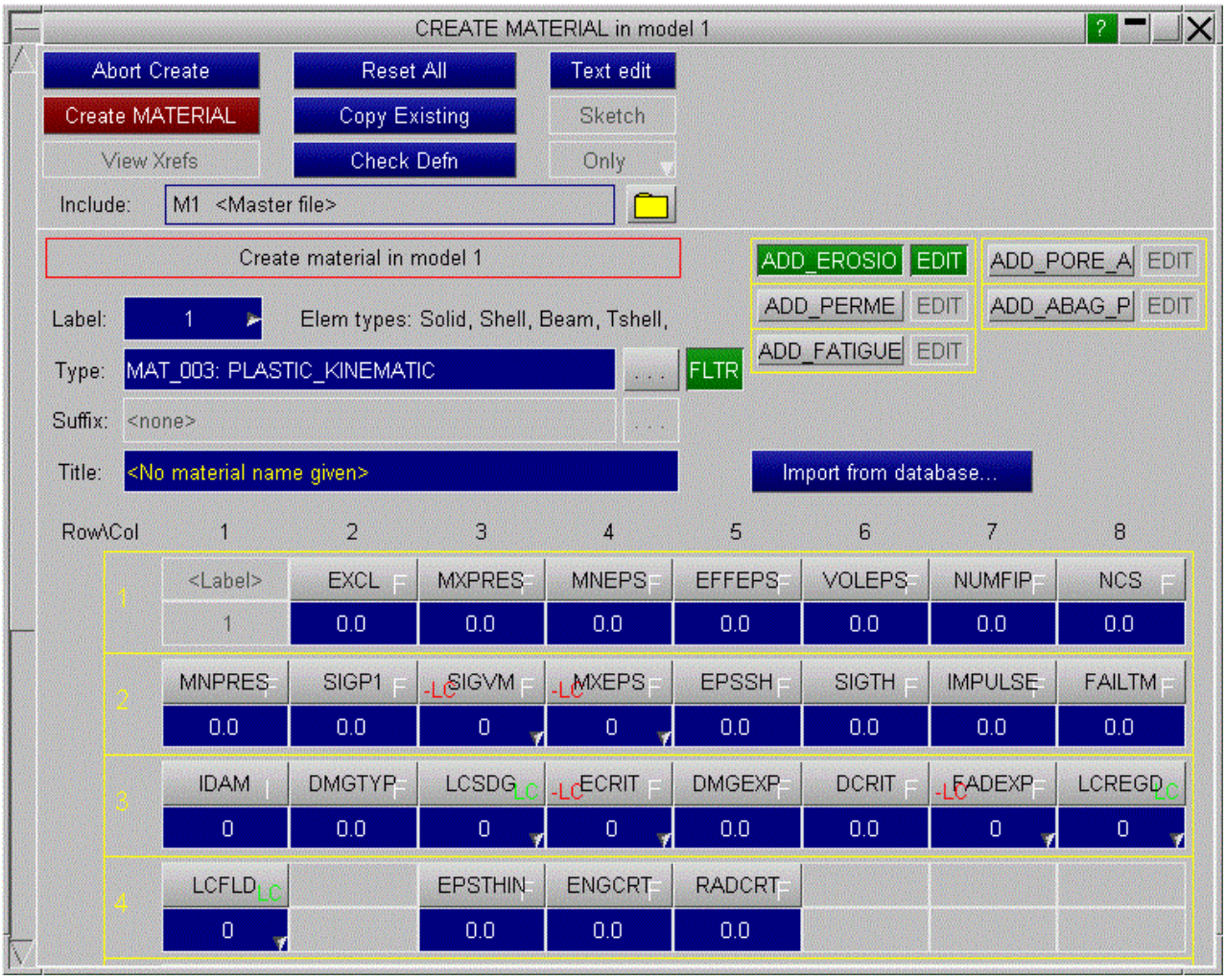
MAT_ADD_EROSION

From LS-Dyna version 950 onwards any structural material type can have "erosion" properties defined for it. This provides a range of failure parameters that can be used to delete ("erode") the elements using the material.



By default erosion properties are not defined for a material, and this option defaults to **Inactive**.

If it is made **Active** then you can **EDIT** it:



***MAT_ADD_PORE_AIR**, ***MAT_ADD_PERMEABILITY**, ***MAT_ADD_AIRBAG_POROSITY** and ***MAT_ADD_FATIGUE** are also available to edit in the same way as ***MAT_ADD_EROSION.**

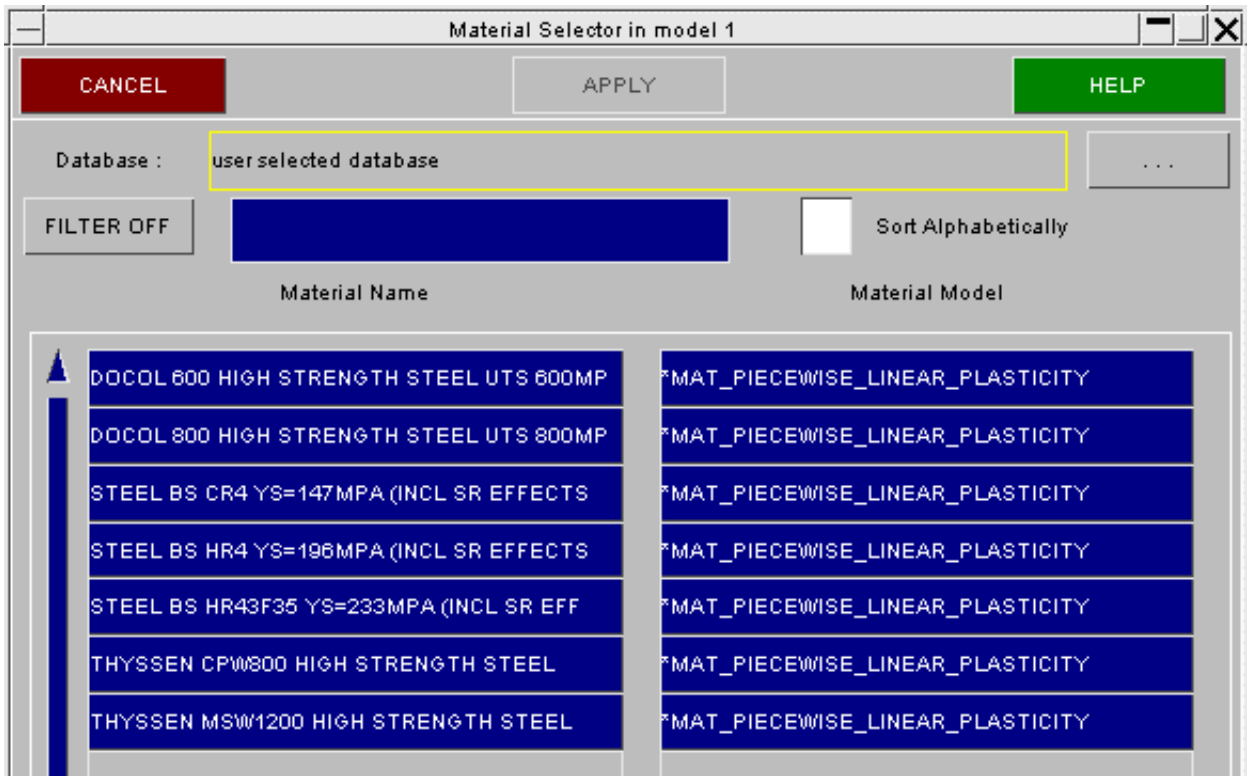
Import from database... Importing pre-defined material data from a database

Material data can be stored in a database, and "imported" to populate a definition. The information imported is:

- All the data fields in the material definition, except for its label, are overwritten.
- If the material in the database is of a different type to the current material, the type is also changed.
- If the database definition refers to loadcurves (***DEFINE_CURVE**) or tables (***DEFINE_TABLE**) then these will be imported too. They will be given labels <highest current curve/table number + 1> onwards.

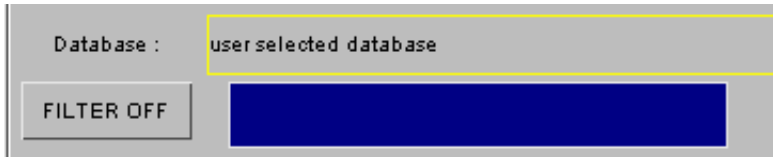
Note: **Import from database...** in this context differs slightly from the same command in the top material panel:

- Here: it imports data unconditionally for a single material definition.
- In the top panel it imports definitions for a range materials by matching up the names of the database definitions against those in the model.



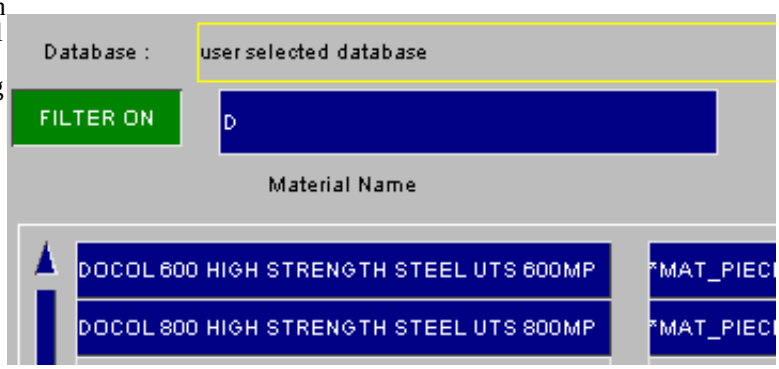
Here is an example of a database containing seven materials. The stored Material Name and the LS-Dyna **Material Model** type for each are listed.
To import a material click on its **Name** or **Model** definition, either will do, and press **APPLY**. This will overwrite the definition in the current create/edit panel with the imported data (only the label is left unchanged).

FILTER OFF/ON Filtering the material list
By default the filter is **OFF**, and all materials in the database are shown.



If the filter is **ON** then only those materials with names containing the character string given will be shown.
In this example "D" has been chosen, restricting the list to just two materials.

The filter is case-sensitive: "D" and "d" are treated as distinct.



Sort Alphabetically
By default materials appear in the order in which they are defined in the database.

Turning **Sort Alphabetically** on sorts them alphabetically by name. The sort is not case-sensitive.

Once all of the data has been input on the material card, press **CREATE_MATERIAL** to install the material permanently in the model.

As with other creation/editing functions a standard check is made of the new definition prior to saving it, and you are warned about errors found. However note that comprehensive checking of materials is nearly impossible, and for most

types PRIMER simply ensures that a positive density has been used.

MAterial database format and instructions for setting up a database are given in appendix 9.

COPY Copy existing material(s) to make a new material(s).

COPY makes new materials, in the same model(s), that are identical to their originals apart from their labels. By default only the material definitions themselves are duplicated.

Where **RECURSIVE COPY** is requested, all items associated with that material (i.e. elements, parts, etc.) are also copied.

MODIFY Modifying the attributes of an existing material.

MODIFY functions in the same way as **CREATE**. Obviously, the material type will already have been selected so the panel will resemble that shown in "populated" figure above.

If the material is in use by a `PART` which has elements then the "element type" of the material is locked to those elements, which will restrict the range of material types you can change it to. For example a material used by springs cannot be changed to a shell type.

Any modifications made to the material definition will not be made permanent until the **APPLY_MODIFY** button is pressed. At this point a the local copy which has been updated is used to overwrite the version in the model.

DELETE Delete existing material definitions.

The **DELETE** function deletes the selected materials. However you cannot delete a material that is in use by a `PART` unless you remove it from the part definition, or delete that too. To help with this the following two switches may be used:

DELETE_RECURSIVE Will select for deletion the parts, associated elements, and so on that reference this material.

REMOVE_FROM_SETS Is often also needed if parts are to be deleted, as their elements, the connected nodes, and often the parts themselves may be included in sets.

A good way of getting rid of surplus (unused) materials is to turn these two switches off, then select all materials for deletion. Only those which are not used by anything will actually get deleted.

IMPORT Matching up and restoring material definitions from a database.

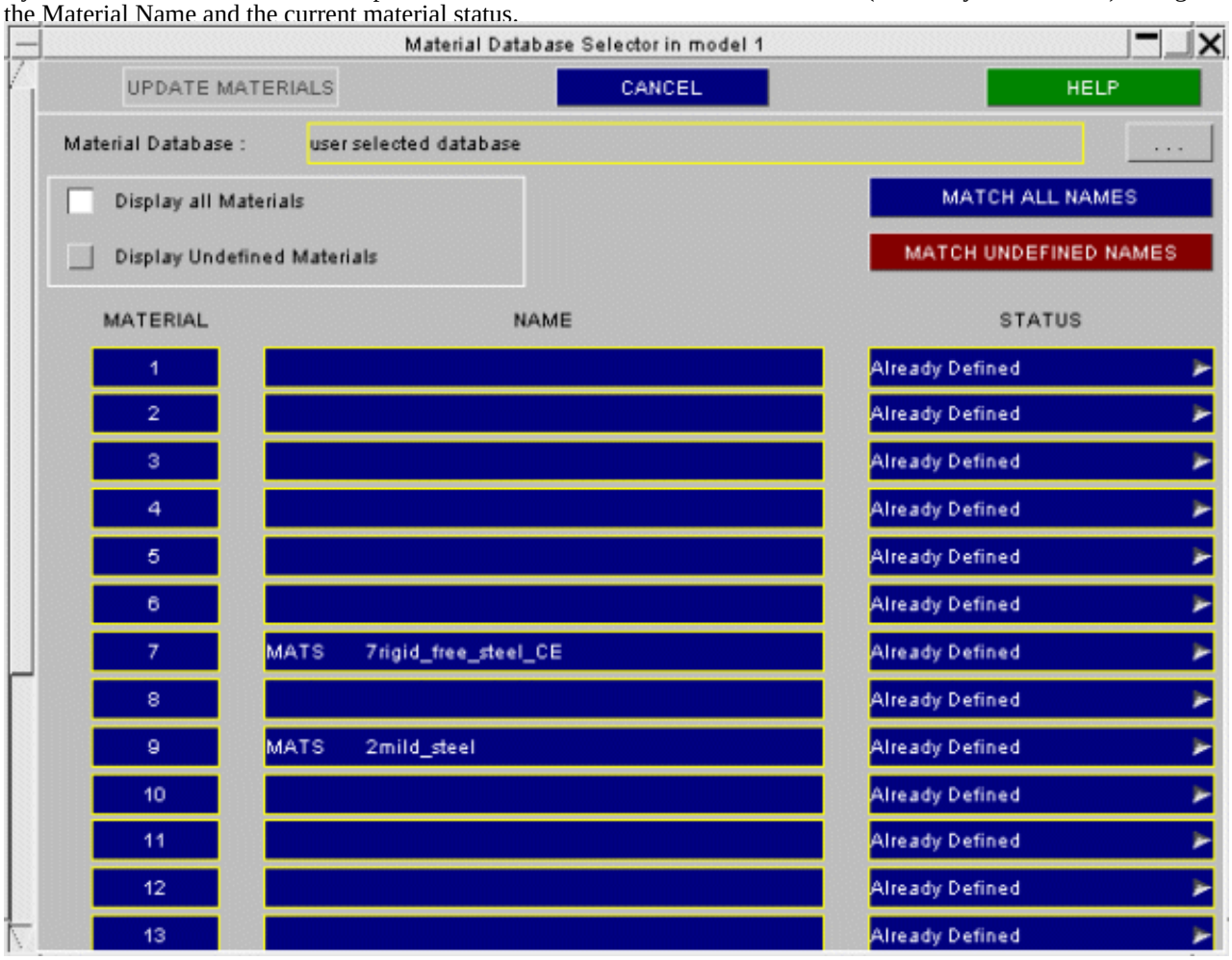
When a LS-Dyna model is taken into a 3rd-party pre-processor for modification it often loses most of its material definitions, as they are too complex for "general" software. This frequently happens when models are partially remeshed.

A model that comes back into PRIMER from such a source is referred to as being "stripped" of its materials.

This option allows you to maintain a separate database of material definitions, and to "marry" them up with the "stripped" model by matching their names.

This figure shows the material **IMPORT** menu.
At the top of the menu the current material database is displayed along with an option to select an alternative material database. If this option is greyed out then PRIMER has been unable to locate any material databases.

By default the menu contains a complete list of all the materials the model contains (sorted by Material ID), along with the Material Name and the current material status.



MATCH ALL NAMES

This option will search through **ALL** of the materials in the current model (ignoring the current material status) and will attempt to match them with the material definitions in the current material database.

MATCH UNDEFINED NAMES

This option will search through only the <<Undefined>> materials in the current model and will attempt to match them with the material definitions in the current material database.

If the current material database is changed after selecting either of the 2 options above and then the MATCH ALL NAMES option is selected then any material that was matched in the first database and also matches an entry in the second database will be replaced with the entry from the second database.

Material Names

The material name displayed for each material can come from a number of sources.

If the model being edited has been read in from a KEYWORD input deck then the material name is taken from one of the following sources (listed in order of preference)

- ***MAT_..._TITLE** card (LS-960 input files and above)
- **$HMNAME** - Hypermesh material name comment card.
- **$PRTITLE** - PRIMER material name comment card.

If the model has been read in from an IDEAS Master Series Universal file then the material names are taken from the material database cards in the universal file.

Material Status

Each material is displayed using one of the following statuses.

<< Undefined >>	The material is a Latent material without any material properties defined for it yet. (RED)
Automatically Found in Database	The material has been automatically matched with a material in the current material database. (GREEN)
User Selected From Database	A material definition in the current material database has been selected by the user for this material. (BLUE)
Defined by User	The material definition has been edited by the user. (BLUE)
Already Defined	The material is already defined in the model and has not been modified by the user. (BLUE)

In addition to the material status the entries in the list are colour coded using the colours above.

Automatic Material Import

The Material name can be used to automatically locate a material definition in the current material database. To locate a material in the database the following rules are applied.

- The material name in the current model is converted to upper case.
- The name is compared to each name in the material database (also converted to upper case) to see if there is an exact match. If the name matches more than one entry in the material database then the first entry in used.
- If none of the database names match exactly then a second search through the database is carried out to see if the material name is a subset of one of the names in the database of if the database name is a subset of material name.

If for example the current material database contained the following names

Database Names	: STEEL H350 - 1
	: STEEL H350
	: STEEL H420

Then the following materials would be matched as follows

Material Name	Database Material	
STEEL H350	STEEL H350	*(Exact match)*
NEW STEEL H350	STEEL H350	(Database subset of material name)
STEEL H	STEEL H350 - 1	(Material name subset of database 1st match)

Manual Material Import

In addition to automatically selecting a material from the database the user can manually select a material from the database by using the POPUP menu attached to the status button of each material.

SELECT FROM DATABASE...	This option will display a list of all the materials in the current material database.
EDIT...	This option will bring up the standard create (undefined material) or edit material panel.

SKETCH Sketch elements using a material on the current image.

SKETCH sketches on top of the current image the parts and elements that reference the selected materials.

LIST Summarise the attributes of selected materials.

LIST allows the user to individually select materials and display a summary listing of their attributes.

CHECK Check selected materials for correctness

CHECK runs the standard checking function on the selected materials, summarising any errors.

WARNING: Checking materials thoroughly is a mammoth task, which PRIMER does not at present attempt. Most structural materials are currently only checked for a positive density. Therefore that a material "Checks OK" does not mean that it contains no errors!

RENUMBER Renumbering material labels.

RENUMBER lets you change any or all material labels within a given model using the standard renumbering panel.

To change the label of an individual material it may be simpler just to **MODIFY** it.

STRESS / STRAIN Visualise stress strain information

STRESS / STRAIN lets you visualise stress / strain curves for various material types. Select the materials you wish to visualise and click **Apply**. A panel will open displaying the stress/strain curve(s).

If displaying information for more than one material, you can click through the materials displayed in a list on the right hand side of the panel.

The curves displayed are constructed from the appropriate material infomation. The following rules are used:

- If the material uses a *DEFINE_CURVE that is what is displayed. The curve data can be modified on the stress/strain panel.
- If the material used a *DEFINE_TABLE the first curve in the table is displayed. The curve data can be modified on the stress/strain panel.
- If the material has stress/strain data defined by a series of points on the material card (for example, with MAT24 you can specify stress/strain data using the EPS/ES fields), these points are displayed as a curve. The curve points can be modified, and the material data will be updated accordingly, however you will not be able to add more than the maximum number of points that can be defined on the card (in the MAT24 example there is a maximum of 8 points).
- If the material stress/strain data is defined by a yield stress and tangent modulus (for example SIGY and ETAN on MAT24) than a curve is constructed from these two values. SIGY is the first point, and the second point is derived from ETAN and a max strain value. This max strain value is 1.0 by default and can be changed on the options panel which can be opened from the stress/strain main panel. The curve points can be modified, and the material data will be updated accordingly, but for this method you cannot have more/less than 2 points.
- If the material stress/strain information is defined by a power law (for example MAT_JOHSON_COOK...), a curve is created from the card information. You can specify the number of points on the created curve (default 100) and also a strain rate (default 1.0) on the options panel which can be opened from the main stress/strain panel. The curve points cannot be modified using this method.

THERMAL MATERIALS

The thermal material editing functions allow all thermal material types to be processed.

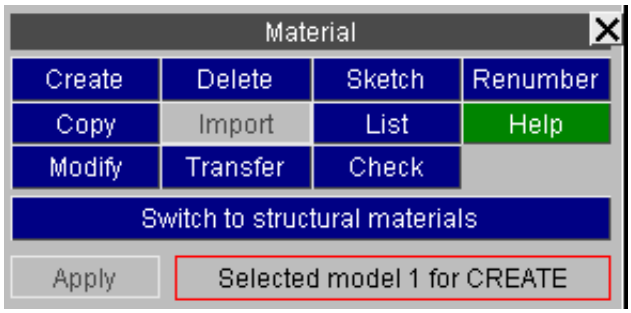
This is done in exactly the same way as structural materials, save that:

- The panels are somewhat simpler, reflecting the less complex nature of thermal materials.
- The concept of "Erosion" does not exist for thermal materials.
- There is no database "Import" capability for thermal materials.

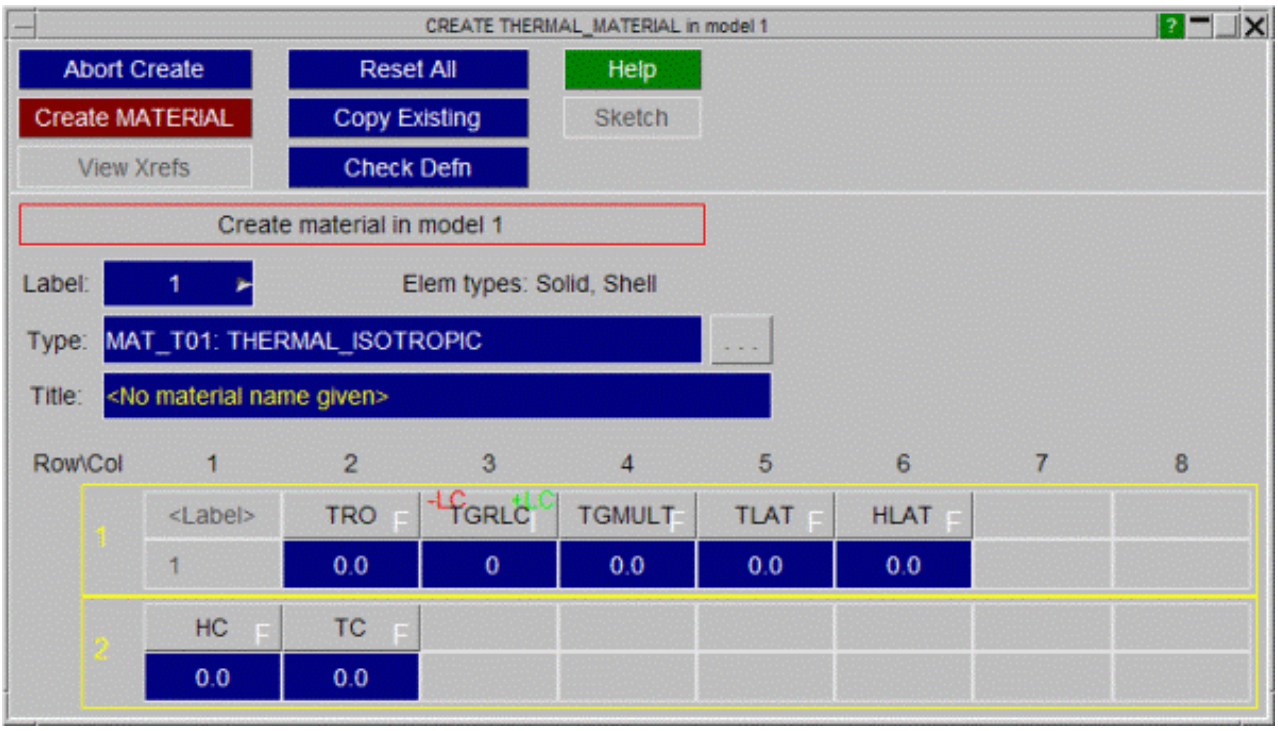
PRIMER does not draw thermal materials explicitly. Like structural materials they can be **SKETCH**ed by drawing the parts and elements that reference them.

This is the main panel for thermal material editing. The functions currently available have their standard meanings. (See 5.1.1).

(There is no **IMPORT** facility for thermal materials.)



This is a typical thermal material editing panel. It functions in exactly the same way as structural materials.



Visualisation of materials

Materials cannot be drawn explicitly (there is no **ENT**ity Viewing setting for them), however they can be visualised in the following ways:

- By a **SKETCH** command in the top menu or editing panels above;
- By selecting **MATERIAL** as the native colour for part-based elements: see "controlling the colour of elements".

Selected material properties can also be contoured: see section 4.2: "Data plotting commands"

Special rules for *MAT_USER and loadcurves

Material type **`MAT_USER`** (types 41 - 50) is special in that the input is a series of arguments for a user-supplied subroutine, but the definition of that subroutine is supplied separately outside the keyword deck.

In many cases such materials reference load-curves or tables, and this can cause problems during **Cleanup Unused** if these curves are not referenced by anything else because it makes them look "unused" and hence eligible for deletion during the Cleanup operation. Therefore from V11 onwards the following special rules apply:

- By default all **`*MAT_USER`** definitions in a given include file (or the master file) generate internal references to all loadcurves, tables or functions in that include file. This locks them against deletion, solving the Cleanup Unused problem described above.

- This behaviour can be modified using the preference **`primer*mat_user_ref_curves`** as described below.

Arguments for the **`primer*mat_user_ref_curves`** preference

IFILE_ONLY (default)	All ***MAT_USER** definitions in an include file (or the master file) make internal references to all loadcurves, tables and functions in that file **only**.
ALL	All ***MAT_USER** definitions in the model reference **all** loadcurves, tables and functions in the model - regardless of include file membership
NONE	No such references are made.

Special rules for encrypted materials

Input decks of proprietary models such as crash dummies often contain wholly or partially encrypted material data using PGP ("Pretty Good Privacy") encryption.

Since PRIMER does not have the private key required to decrypt these definitions it cannot "see inside" them, meaning that both material label and any references to loadcurves from within the material are unknown. Therefore PRIMER provides special support for such material definitions, and this is described in section 5.1.8 PGP Encrypted keyword data

NODE: Defining Nodes.

Top level menu
- Creating
- Copying
- Editing
- Deleting

Visualisation
Labelling

Nodes ("grid points") form the vertices of elements and wide range of other structural purposes: they are the "glue" that holds a finite element model together. Within an LS-Dyna analysis virtually all mass is lumped at nodes, providing a simple mass vector that doesn't require matrix inversion: the basis of the "explicit" solution method.

Special rules apply to the eligibility of nodes for screen-picking: they do not have to be drawn explicitly to be pickable: see the "Rules for screen-picking nodes" below.

Screen-picking The following special operations may also be performed on nodes.

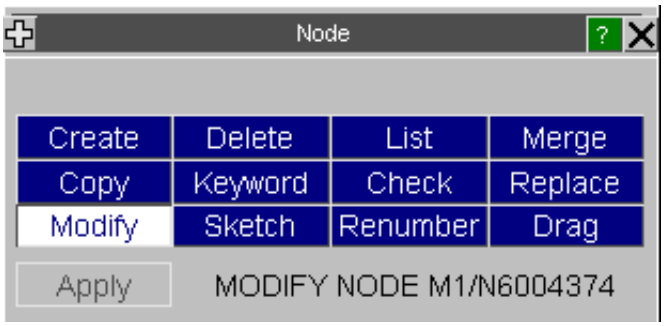
Replace Allows you to replace node A with node B

Drag Allows nodes to be dragged, morphing the model, using a range of geometrical and topological rules.

Duplicates When nodes have been multiply defined in different include files PRIMER merges them together using the same coincidence rules as LS-DYNA, creating "clone" definitions so that they are remembered. This option allows you to sketch and label these duplicated nodes.

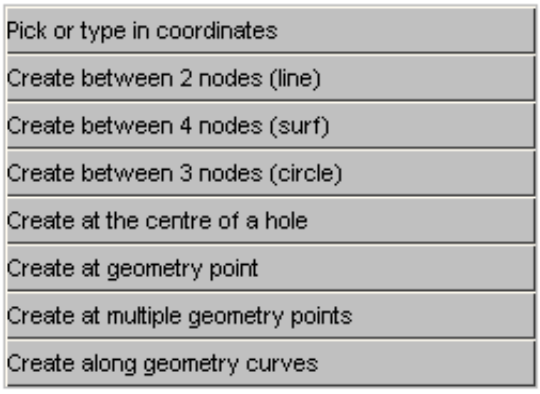
The nodes menu allows the creating, modification etc. of nodes in a keyword deck.
The functions currently available have their standard meanings. (See section 5.1.1)

Generic keyword editing is also available.



CREATE Making new node(s).

There are eight possible ways of creating nodes. These are selected by right clciking below **Method**. The following options are available



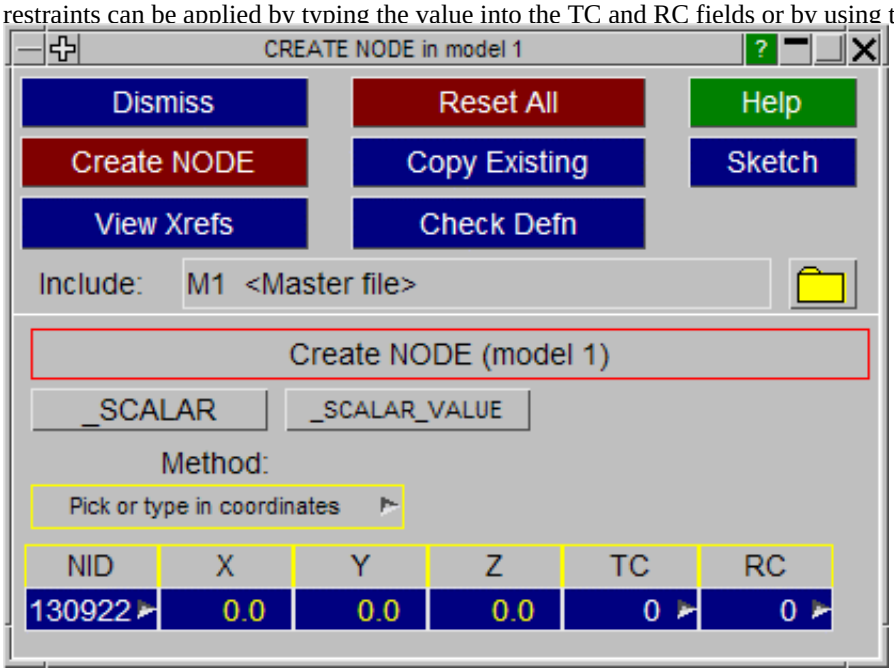
(1) Pick or type in coordinates

This figure shows the initial state of the nodes creation panel.
The default node label used is the highest node label in the model + 1.
The default coordinates used are (0.0, 0.0, 0.0).

Picking a node from the screen will set the X, Y and Z fields for the node you are creating to be the coordinates of the picked node. Alternatively you can just type in a new value for the X, Y or Z coordinate in the boxes.

The node label can be changed by typing in a new value or using the popup. If needed translational and rotational

restraints can be applied by typing the value into the TC and RC fields or by using the popups.



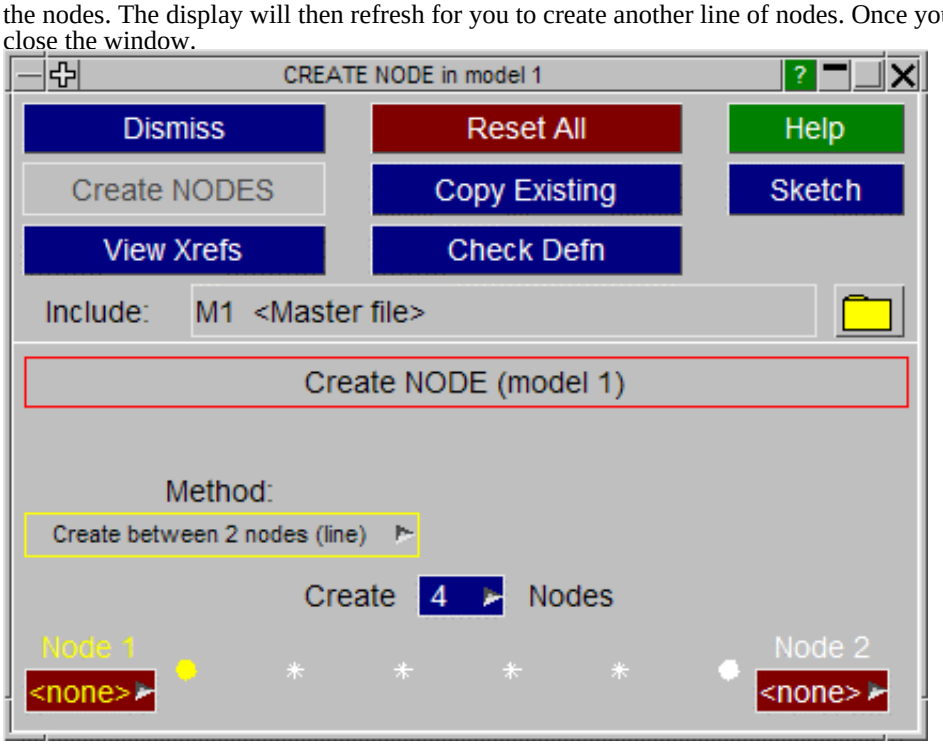
(2) Create between 2 nodes (line)

Instead of creating a single node, you can create any number of nodes in a line between 2 existing nodes. Initially the two end nodes are undefined so will be shown as <none> on a red background. Picking a node from the screen will update one of the two end nodes. The one which will be updated is shown in yellow instead of white, so in the figure to the right, Node 1 will be updated if a node is picked. Once this is picked then node 2 will be highlighted and can be picked from the screen. The two nodes can alternately be picked from the screen in this way.

Alternatively you can type in a node number or use the popups to create, select, sketch etc a node.

Any number of nodes can be created between the 2 end nodes. Either use the popup or type in a number to select how many to create. In this figure 4 nodes will be created.

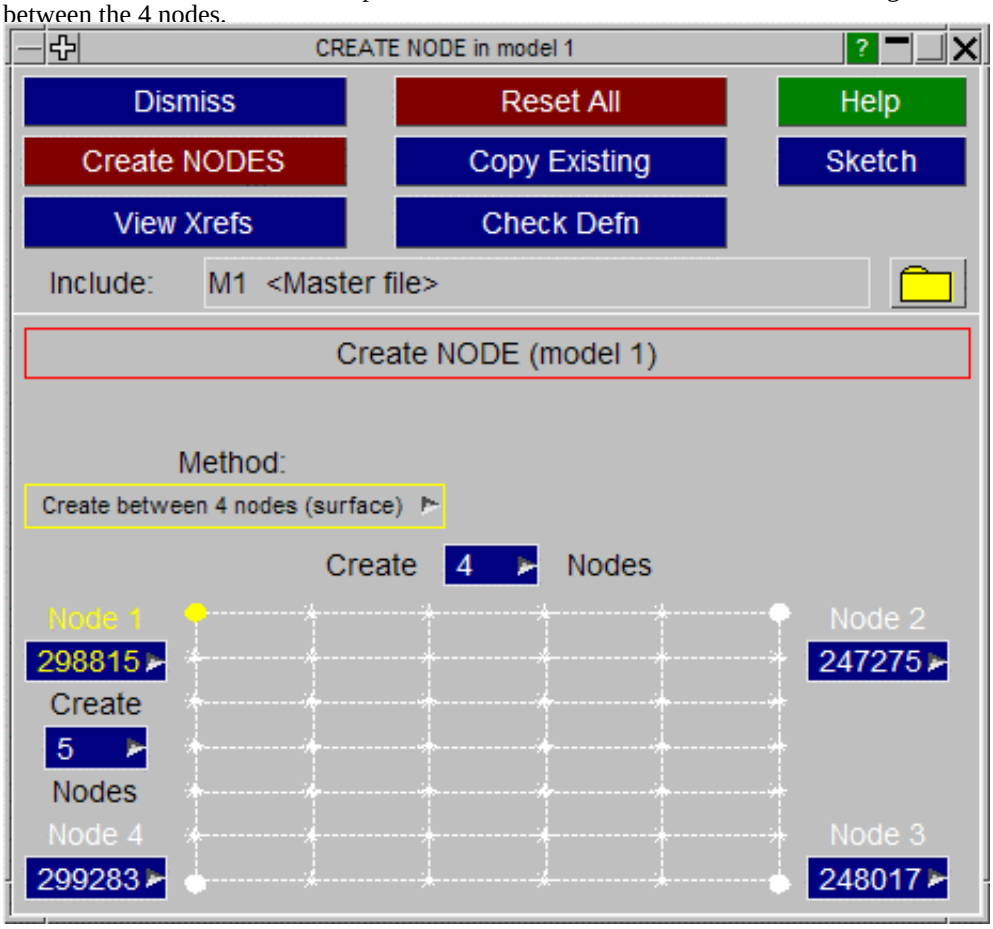
Once both end nodes have been defined the **CREATE_NODES** button will become active and can be used to create the nodes. The display will then refresh for you to create another line of nodes. Once you have finished **DISMISS** will close the window.



(3) Create between 4 nodes (surf)

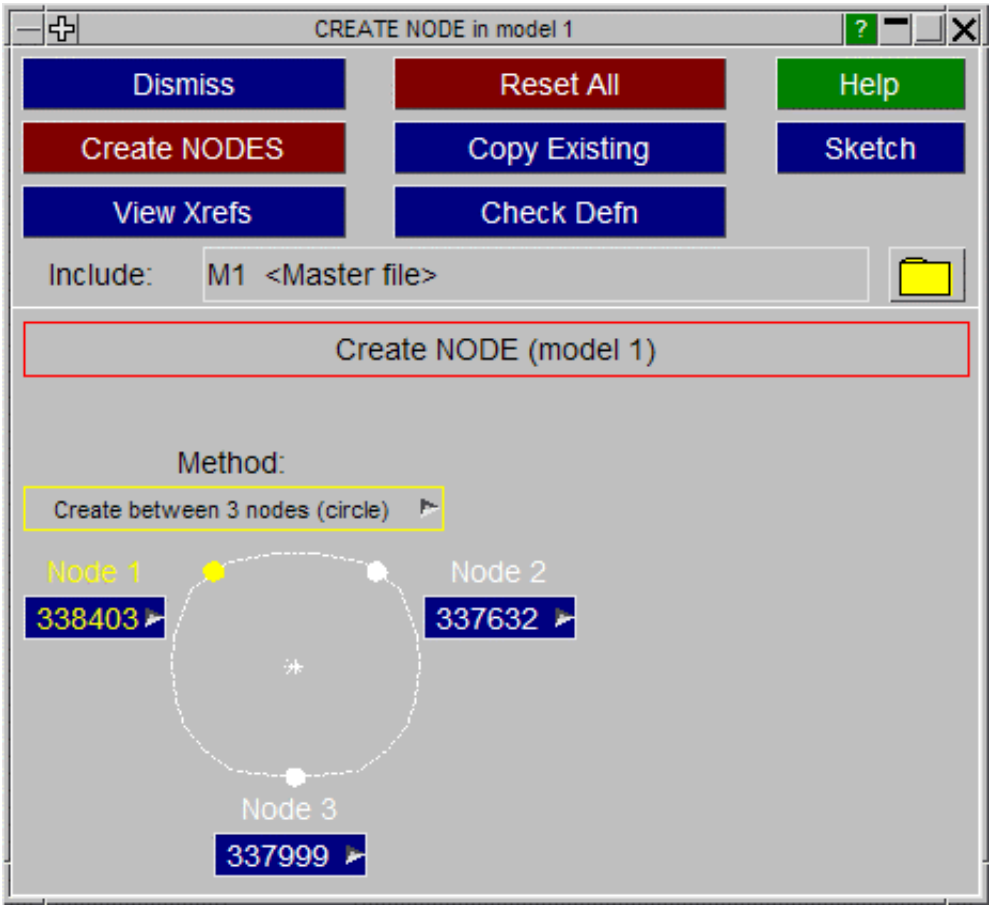
Creating a surface of nodes works in an identical way to creating a line of nodes except that 4 nodes need to be defined (one at each corner) and the number of nodes to create can be varied in both directions.
In this figure node 1 is the node currently highlighted for picking, and 4 nodes will be created in one direction, 5 in the other.

The nodes do not need to be on a plane. If the nodes are not then the nodes will be generated on a curved surface between the 4 nodes.



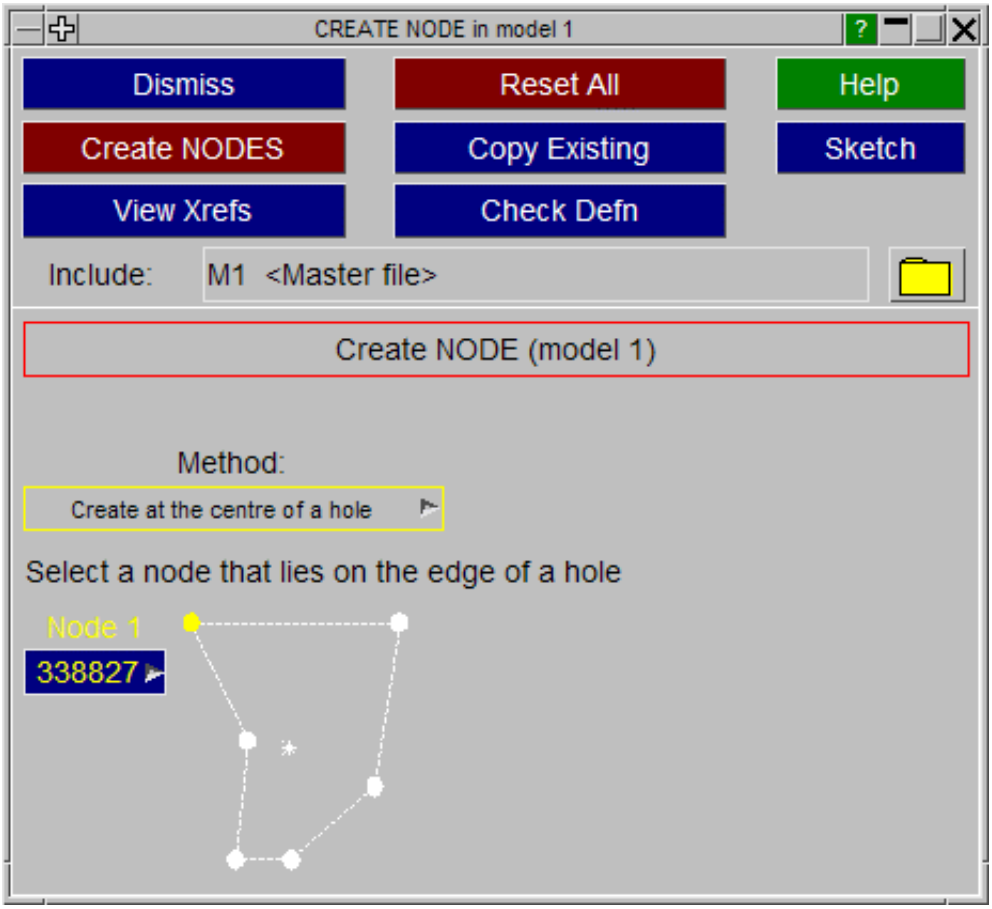
(4) Create between 3 nodes (circle)

Creating a node at the centre of a circle works in an identical way to creating a line of nodes except that 3 nodes need to be defined.
In this figure node 1 is the node currently highlighted for picking. The 3 nodes define a circle, and the node created will be at the centre of that circle.



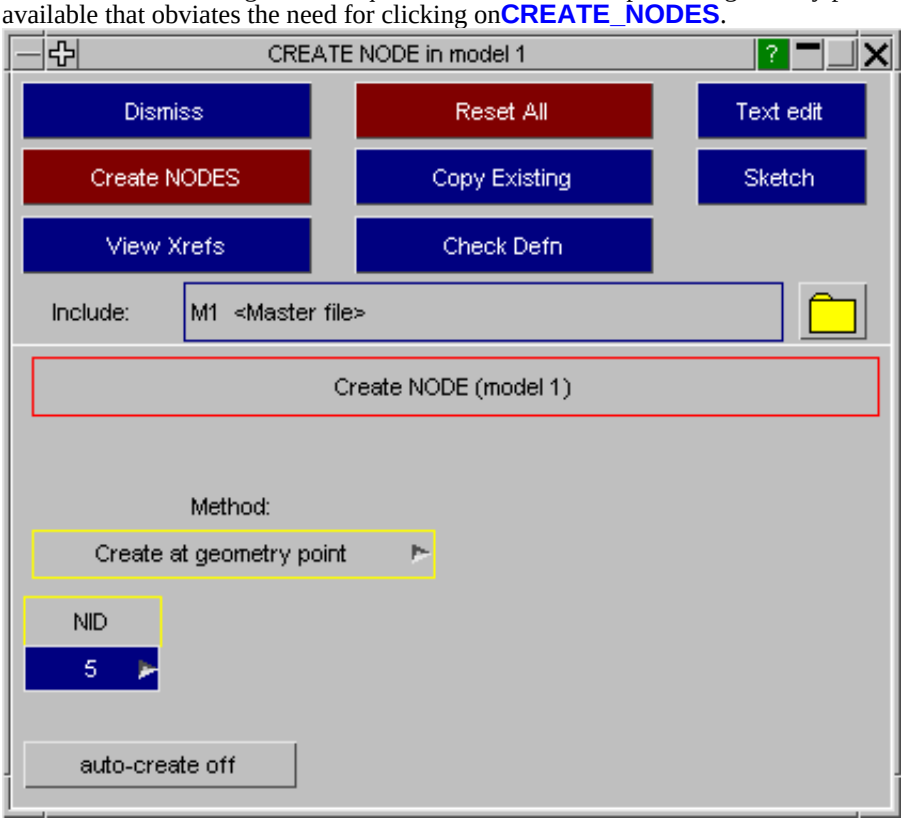
(5) Create at the centre of a hole

This method of creating a node only requires the user to select one node. This node must be on the free edge of a hole. Primer will determine the centre of the hole and create the node there when clicking on **CREATE_NODES**.



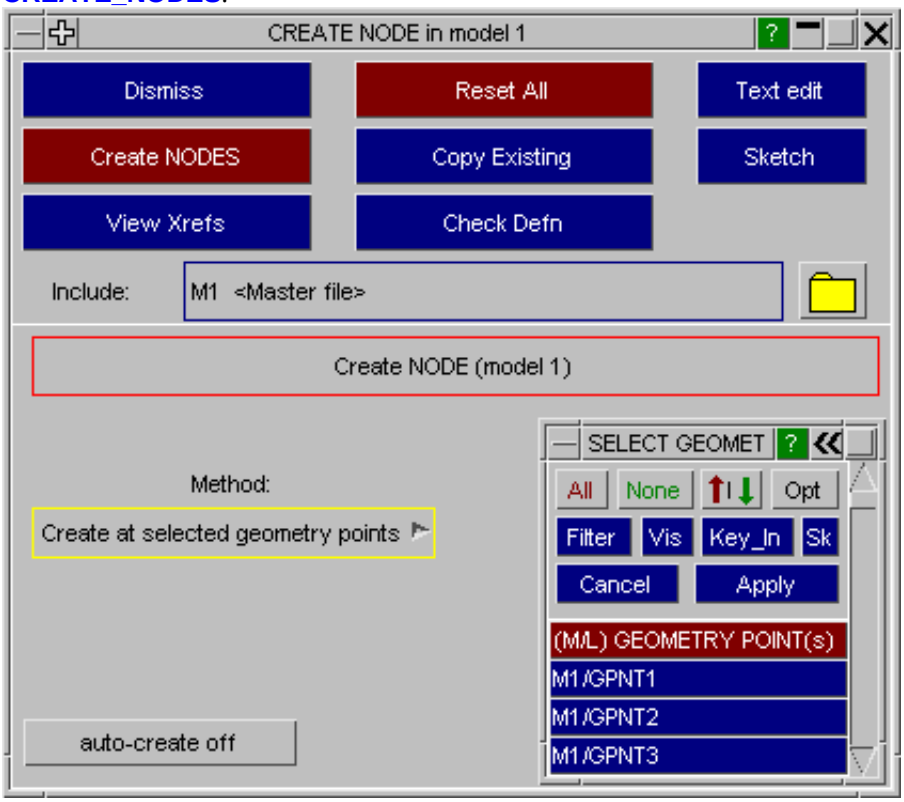
(6) Create at geometry point

This method of creating a node requires the user to screen pick one geometry point. An 'auto-create' option is also available that obviates the need for clicking on**CREATE_NODES**.



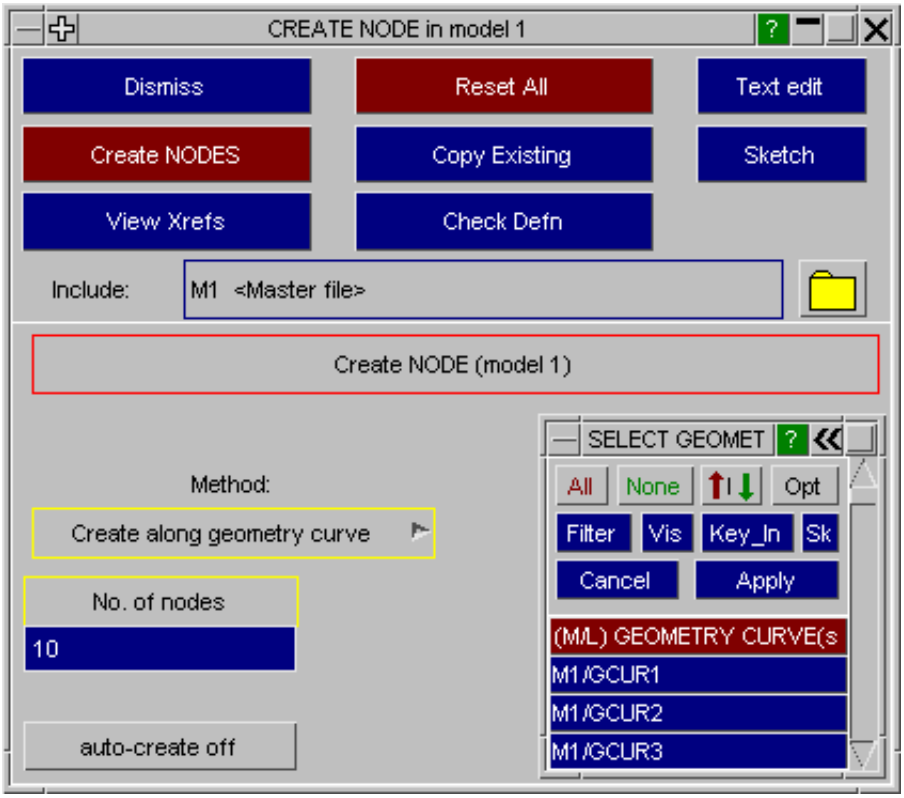
(7) Create at selected geometry points

Instead of creating nodes by picking one geometry point at a time, users may select multiple geometry points from a menu or off the screen. New nodes are then created at the location defined by all these geometry points. As in the case of the single geometry point method, an 'auto-create' option is available that obviates the need for clicking on **CREATE_NODES**.



(8) Create along geometry curves

Multiple geometry curves may be selected. Primer will then create the desired number of nodes spaced equidistantly along each of these geometry curves .



Other node creation commands:

DISMISS Aborts the current definition and returns to the main nodes menu.

RESET_ALL Resets all attributes to <null> for this definition: all data entered will be lost, and the panel will return to its initial default state.

COPY_EXISTING Copies the attributes of an existing node definition (in the current model). This may then be modified as required.

SKETCH Sketches the current definition on top of the current image.

LIST_XREFS Lists everything that references the current node definition.

CHECK_DEFN Performs a check of the current definition, listing any errors.

CREATE_NODE Saving the node definition.
Once you have entered the node information the **CREATE_NODE** button will save this definition. The definition will be checked and any errors listed, an then it will saved permanently in this model.

Until you press this the definition remains volatile, and will be lost if you exit this panel in any other way.

Once the node(s) has been defined the panel will refresh with the default values to speed up node creation.

COPY Copying existing nodes(s) to make a new one(s).

You can **COPY** any number of nodes, in multiple models.

When **APPLY** is pressed you are asked to confirm what is to be copied, and then the operation is carried out.

For each model the <n> extra nodes chosen in that model are copied using labels <previous highest + 1> to <previous highest +n>, there is currently no control available over the new labels assigned.

MODIFY Modifying the attributes of an existing node.

This functions in exactly the same way as **CREATE**, using the same panels as in the figures above. The only difference is that the initial state of the panels is already set with the attributes of the node to be modified. In the modify mode the options to create nodes along a line or surface are not available. Only a single node can be modified.

If you want to change values on multiple nodes then you should use the **KEYWORD** option instead

DELETE Deleting existing nodes

The **DELETE** operation works exactly the same way as **COPY** above, except that the chosen nodes are deleted.

- If **DELETE_RECURSIVE** is switched on any loads, constraints etc. referenced by the nodes to be deleted are marked for deletion.
- If recursive deletion is not used only the node definitions themselves are removed.

Note also that the standard deletion rules described in Section 6.4.1 still apply: nodes will only be deleted if nothing else (which is to remain) depends on them.

KEYWORD Generic keyword editor

KEYWORD starts the generic keyword editor which allows creation, deleting and modification of multiple nodes. This is useful for modifying multiple nodes in a single operation.

SKETCH Sketch the chosen nodes on the current image

SKETCH allows the user to select and sketch individual nodes on the current graphics image. Nodes are drawn with a star symbol.

CHECK

Runs the standard checking function on the selected nodes. Each node will be listed either as "OK", or a summary of the errors encountered will be printed. (This is the same as the **CHECK_DEFN** command during node editing.)

LIST

Writes a summary list of the selected nodes to the screen. This is just the total number of selected nodes in each model.

RENUMBER

Raises the standard renumbering panel for nodes in the chosen model, allowing you to renumber some or all of them.

MERGE

MERGE allows you to merge coincident nodes (or nodes within a specific tolerance) together. For more information see the **MERGE NODES** option in the **REMOVE** window.

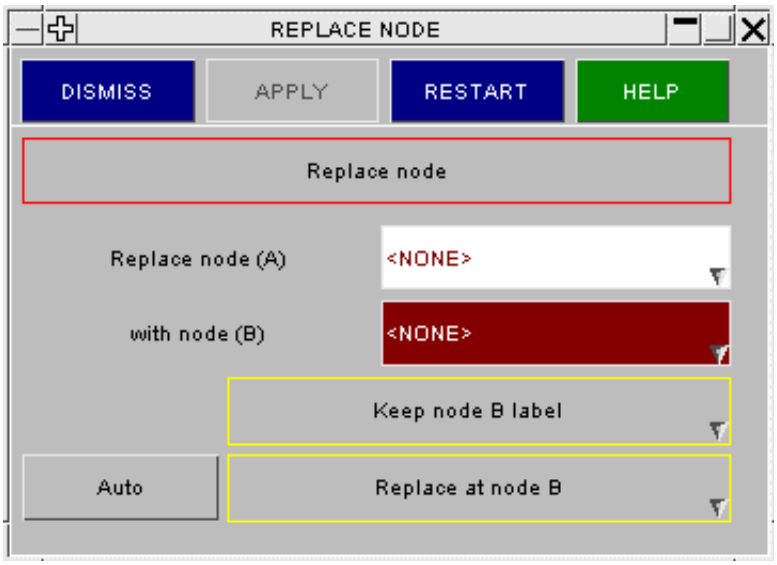
REPLACE

REPLACE works in an identical way to **MERGE** but only allows you to replace a single node with another node.

'quick picking' is enabled in the replace node panel so you can just click on the screen to select the 2 nodes. The node which you are currently picking is shown by the colours being inverted (i.e. in the figure on the right, node A is currently being picked).

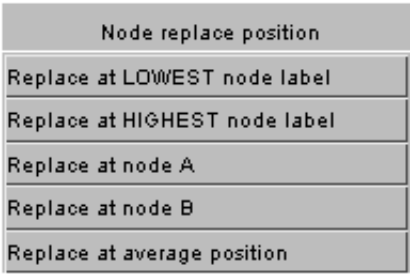
Alternatively you can type in the node numbers or use the popup to select the nodes

If **Auto** is selected then the node will be replaced as soon as both nodes are given. Alternatively press **APPLY** to replace the node.



By default the label of the second node you pick (B) will be kept. You can change this by using the popup. You can choose to keep either label or the highest or lowest label.

By default the node will be replaced at the location of the second node you pick (B). This can be changed by using the popup. You can force the node location to be at the position of either node A or B, the average position, or the position of the node with the lowest or highest label.



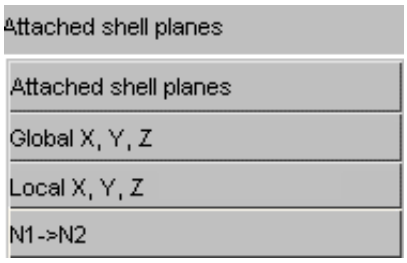
DISMISS returns the user to the main PRIMER window

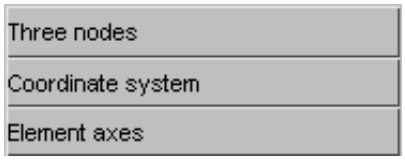
DRAG

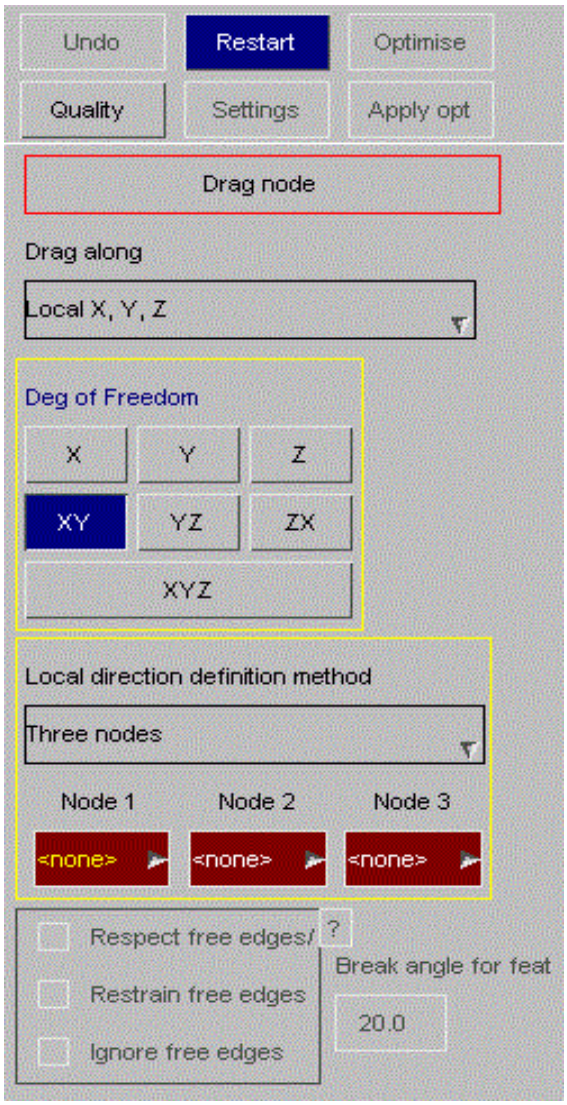
DRAG permits users to drag nodes based on certain constraints. The impact of such an operation on element quality can be viewed using the **Quality** button. Various individual quality metrics, as well as overall quality imperfection can be viewed using the **Settings** button.

Four methods are currently available for node dragging:

- The **Attached shell planes** popup option facilitates dragging along the planes of attached shells.
- The **Local X, Y, Z** option permits dragging along local X, Y, Z axes or along local XY, YZ, ZX planes. Appropriate degrees of freedom can be defined using the attached **X**, **Y**, **Z**, **XY**, **YZ**, **ZX** option buttons. A local coordinate system can be defined using an element, a coordinate system or a set of three nodes.
- The **Global X, Y, Z** option, likewise, permits dragging along global X, Y, Z axes or along global XY, YZ, ZX planes.
- Selecting the **XYZ** degree of freedom will translate mouse motion into movement on the current screen coordinate plane.
- The **N1->N2** option permits users to select a source node 'N1' and destination node 'N2'. After selection, N1 will shift to position of N2.





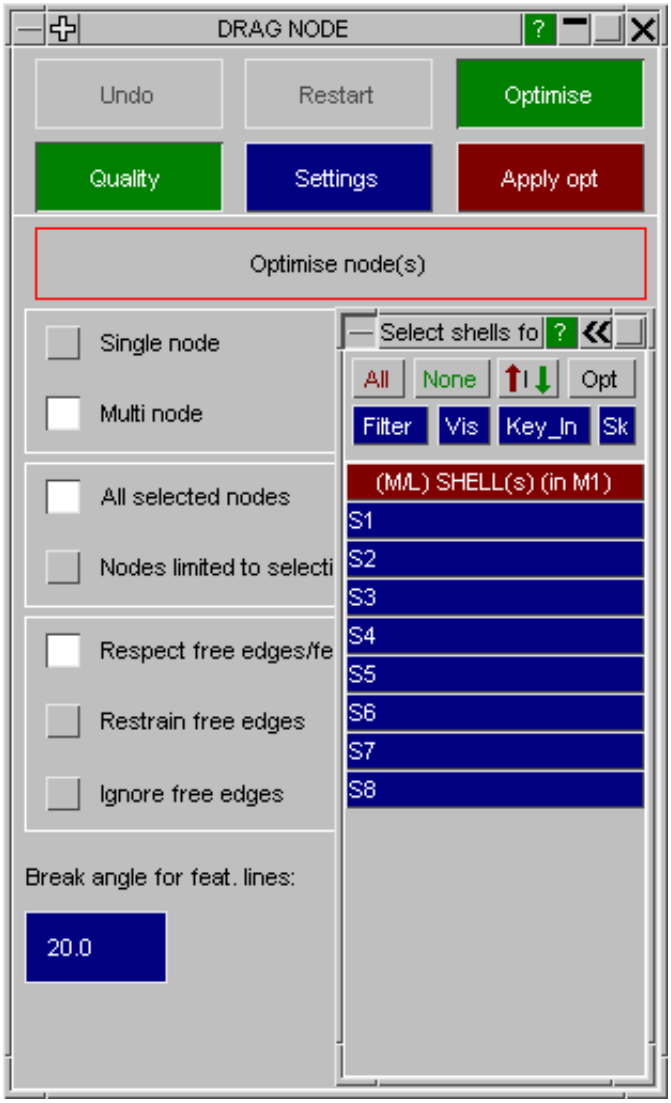


The **Optimise** button will instruct Primer to automatically reposition nodes for improved quality.

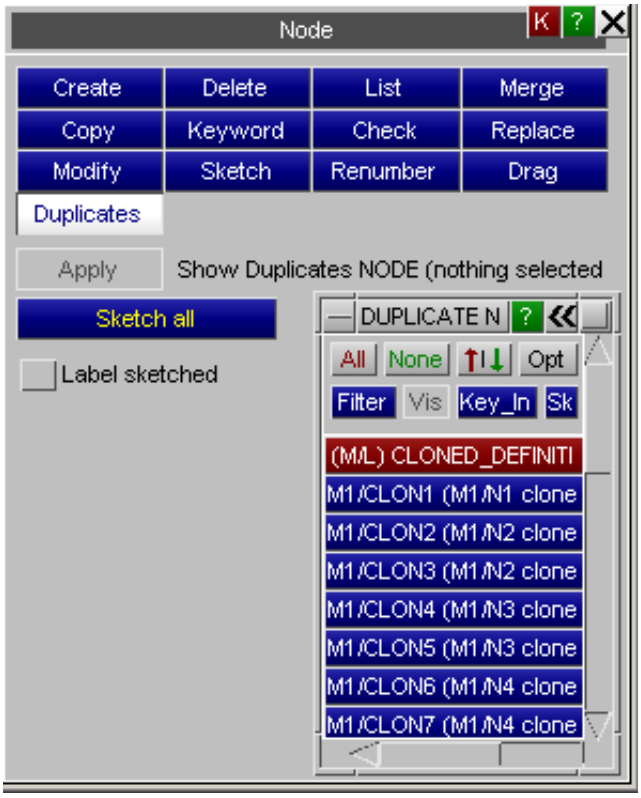
Two optimisation modes are available - single and multi node. The latter permits selection of one or more elements. Primer will then reposition attached nodes so that overall quality of the selected elements is improved.

Nodes that lie on a free edge or feature line can be restrained using an appropriate option.

Movement of nodes that also lie on unselected elements can also be restricted using an option.



DUPLICATES



Sketching and labelling duplicate coincident nodes, sometimes used to "stitch" models together.

Coincidence rules.

LS-DYNA has special rules to handle the case that node label N is defined more than once in different include files. It merges multiple definitions of node N into a single definition so long as:

- The restraint codes TC and RC are identical
- The nodal coordinates are coincident.

The test for "coincidence" of two definitions NA and NB with the same label is as follows:

```
xdist1 = max(1.0e-16, vector distance
of node from origin)
xdist2 = vector distance between
coords of NA and NB

xdist2 / xdist1 < 1.0e-8
```

In addition if a ***NODE_MERGE_TOLERANCE** card has been defined then the distance xdist2 must be greater than this tolerance value for the nodes to be considered "not coincident".

PRIMER uses the same rules as LS-DYNA.

How coincident nodes are handled inside PRIMER

PRIMER has the problem that nodes must be merged if coincident, but also that the duplicate definitions must be "remembered" so that they are written out again in the correct include files. It handles this by creating "clone" definitions of each node such that:

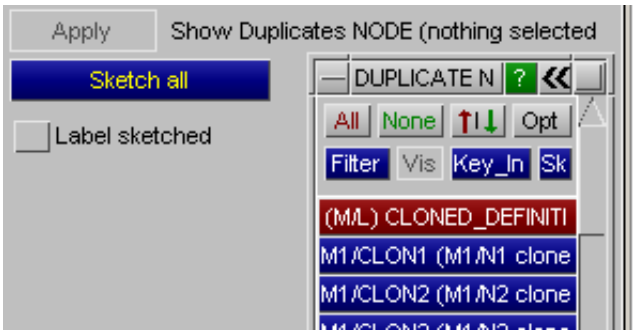
- The first definition of node A that is found is the "true" definition, which is the normal definition of the node.
- Any subsequent duplicate definitions make "clones" of this node where:

  - A clone is simply a reference to the "true" definition
  - It remembers the include file in which it exists.

- On keyword output the "true" definition of the node is repeated in every include file where there is a clone

Coincident nodes are normally merged silently during keyword input, but it is possible to list nodes merged during this process by using the "Save keyin log to file" option.

Visualising duplicate nodes.

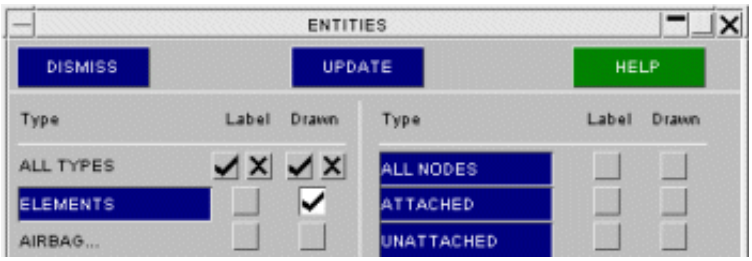


Since duplicate nodes are actually just references to the true node definition they do not appear as separate entities on plots, nor will you see them in menus that list nodes. However they can be sketched and labelled using this **Duplicates** option.

Either **Sketch all** to show all of them, or select the subset of nodes to be seen from the menu of cloned nodes and **Apply** to draw them.

By default only node symbols are shown, but **Label Sketched** will also turn on labels, which shows both node label and also the include file in which it resides.

Controlling the visibility and labelling of nodes.



Node visibility and labelling is controlled from the **ENT**ity Viewing menu.

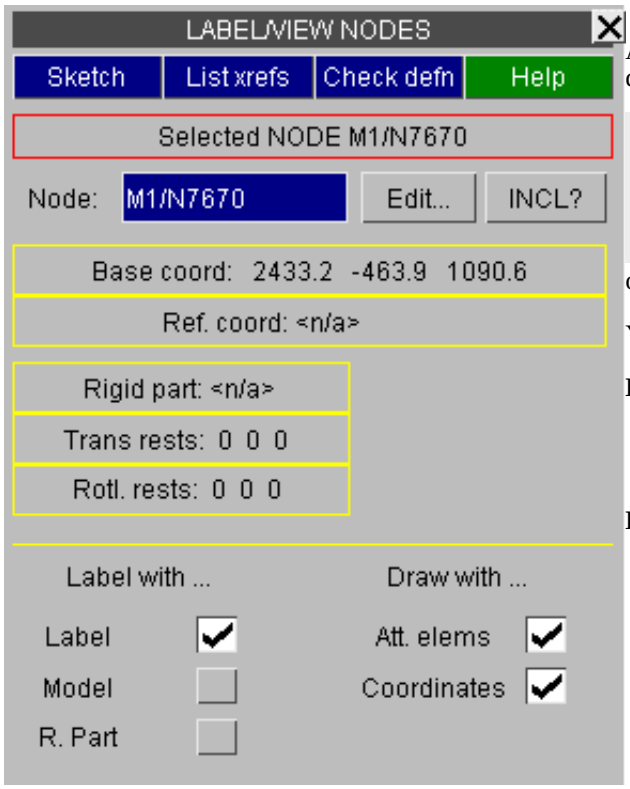
However its treatment is different to that applied to all other item types.

ALL_NODES	Draws all nodes, regardless of attachment
ATTACHED	Draws only nodes attached to items currently visible on the screen
UNATTACHED	Draws nodes that are ***not*** attached to any item.

Labelling of nodes is handled in much the same as their drawing.

Nodes do not have to be drawn explicitly in order to be labelled: for example selecting **ATTACHED** labels will display the node labels on visible elements etc.

Dynamic labelling and details of nodes.



As with elements, nodes have a special "pick to label and display details box" that is invoked by clicking on one of the

ALL_NODES	Buttons in **ENT**ity Viewing (doesn't matter which)
ATTACHED	
UNATTACHED	

or from Quick Pick.

You can control how nodes are labelled and drawn using:

Label with...

Label : The node's label
Model : Prefixes the Mnnn model id
R. Part : The part id of any "parent" rigid part.

Draw with...

Att. elems : The elements attached to the node
Coordinates : The node's global coordinates.

Rules for screen-picking nodes.

Nodes are treated in a non-standard way for screen-picking purposes. A node is pickable if:

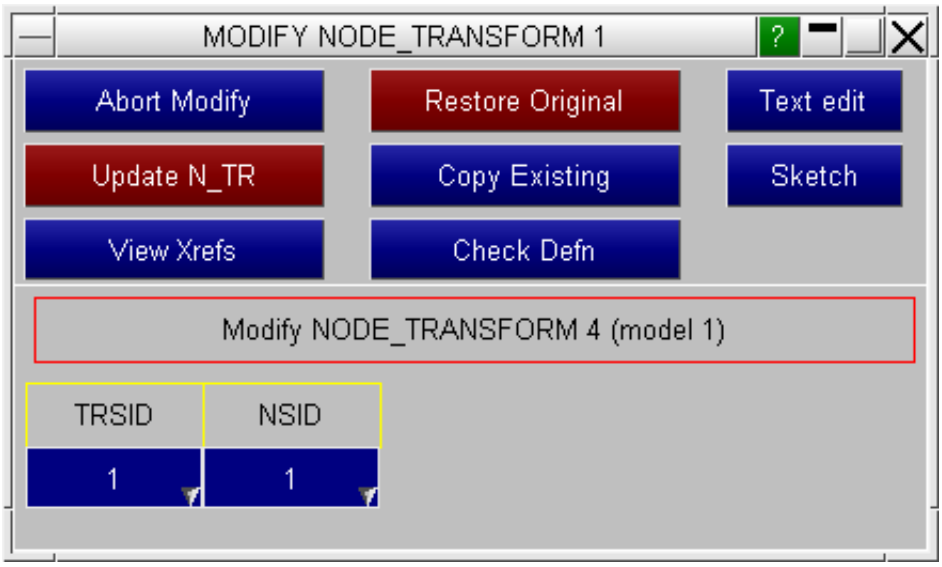
- It is drawn explicitly (using the * symbol)
- Or if an item which uses it (element, joint, restraint, ...) is visible.

The second condition may be thought of as "**ATTACHED**", without actually having to draw the nodes.

The reason for this anomaly is that it is extremely useful to be able to screen-pick nodes, but very annoying if they have to be drawn to make this possible: node symbols tend to obscure other useful information, and also slow down graphics.

NODE_TRANSFORMATION

Node transforms can be created and/or modified.

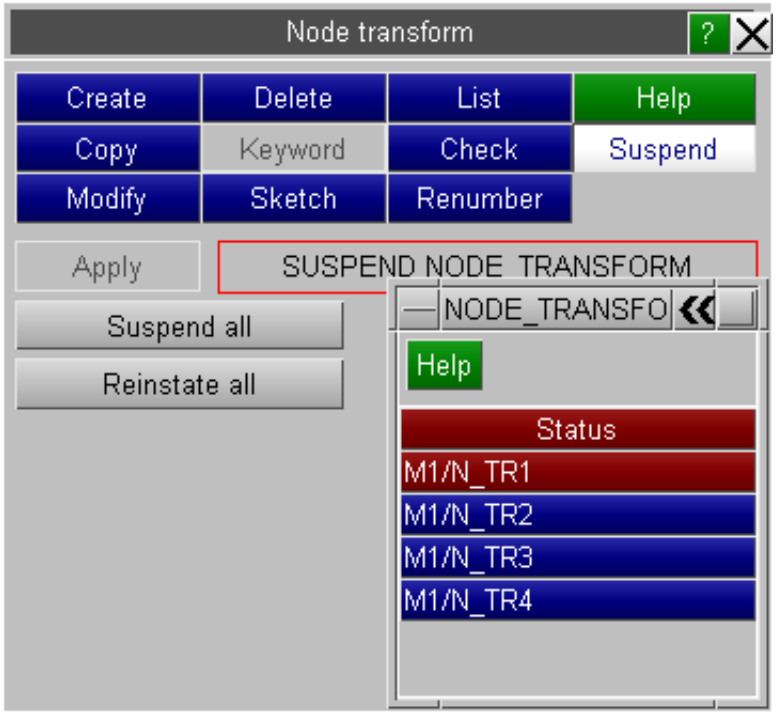


Select or create the TRSID, refer to section 3.14 for more information. The node set ID can also be selected or created using the usual methods.

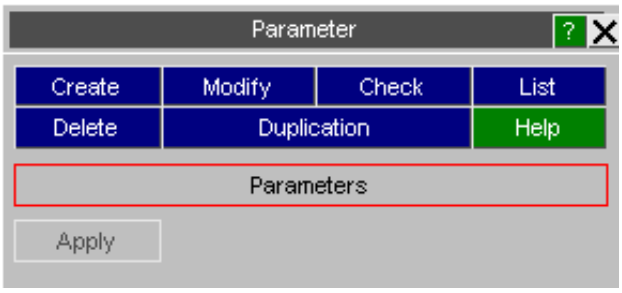
Suspending node transforms

Sometimes it may be useful to suspend node transforms. The **Suspend** option on the **Node transform** menu allows you to do this.

Transforms can be suspended/reinstated by clicking on them in the menu. If a transform is currently suspended it is shown in red (e.g. **M1/N_TR1** is currently suspended in the right hand figure).



PARAMETERS



The figure on the left is the ***PARAMETER** main menu in PRIMER.

From release 970 onwards LS-DYNA supports parameters. These are Integer or Real (floating point) values defined by name at the top of the input deck which can be used in any data field of the relevant type. From LS971 R6 LS-DYNA also offers character parameters.

In addition the ***PARAMETER_EXPRESSION** card allows a parameter to be defined by an arbitrary mathematical expression which may contain references to other previously defined parameters.

The ***PARAMETER** keyword in LS-DYNA has grown in capability over time, and the following is a summary of these changes and how they are supported in successive releases of PRIMER.

From release 9.3RC2 onwards PRIMER provides full support for parameters:
- Both ***PARAMETER** and ***PARAMETER_EXPRESSION** are evaluated on input.
- The association between a parameter and the data field in which it is used is "remembered" across all operations, and rewritten on keyword output.
- Parameters, including the _EXPRESSION variant, may be created, edited and deleted interactively.
- When a parameter's value is changed all affected parts of the model will be rebuilt to reflect the change.
- Either parameters or their underlying values can be displayed in all editing panels, and parameters may be typed into any valid field on these.

From release 10.0 PRIMER adds support for the LS971R5 keywords:
- ***PARAMETER_LOCAL**, which permits multiple parameters with the same name but different values in different include files
- ***PARAMETER_DUPLICATION**, which tells LS-DYNA how to process **_LOCAL** name conflicts is supported in a limited way.

From release 11.0 PRIMER adds support for LS971R6 options
- Parameter type character ("C")
- Parameter names may now be up to 9 characters long (formally the case in 971R6, informally in R5 and later releases of R4)
- The ***PARAMETER_DUPLICATION** card is now fully supported, and its effect on parameter usage mimics that of LS-DYNA

From release 12.0 PRIMER adds support for LS971 R7.1 options.
- The **_MUTABLE** suffix is now supported
- The **_TYPE** suffix is now supported
- Handling of multiple ***PARAMETER_DUPLICATION** keywords has been clarified in the LS-DYNA user manual, and PRIMER supports this.
- You can now Pre-read Parameters when reading keyword files, to avoid the "used before value is known" problem in complex cases.
- The potential to handle parameter names up to 19 characters long has been added, see Permitted length of parameter names in "large" format.

From release 13.0 PRIMER adds support for "implicit" parameters
- Implicit parameters defined within <...> in a keyword data field are supported.

From release 14.0 PRIMER adds support for LS_OPT expressions in data fields.
- LS_OPT expressions defined within <<..>.> in a keyword data field are supported to a limited extent. See Support for LS_OPT expressions below.

From release 16.0 PRIMER has improved the handling of accepting duplicate parameters.
- When a ***PARAMETER_DUPLICATION** card accepting duplicate parameters (option 2 or 4) is present, until version 15 PRIMER only used the youngest definition of parameters of the same name and marked all others as ignored. From version 16.0 onwards duplicate parameters are treated internally as **_MUTABLE** with these duplication settings, which seems to match better how LS-DYNA reads them.

Create Create a new parameter

Modify	Alter Parameter cardsl
Check	Check the parameter cards for errors and completeness.
List	Summarise parameter cards across all models including usage cross-references.
Delete	Delete parameters
Usage in panels	How to use parameters in editing panels and the generic keyword editor.
Implicit parameters	How PRIMER processes "implicit" parameters in <...> in data fields
LS_OPT expressions	How PRIMER processes LS_OPT expressions in <<...>> in data fields
Mis-matches	What happens when the true data value is changed so that it no longer matches the parameter's value.
***Include_Transform**	Special rules for dealing with parameters used inside *INCLUDE_TRANSFORM definitions
Use for labels	Why you should be very careful about using parameters for item labels
Definition order	How PRIMER copes with the order in which parameters are defined.
Units change	What happens to parameter data when a units change is applied to fields in which they are used
Text strings	How "&name" is processed when encountered in a title or other text string
8 or 9 character names	Using the undocumented ability of LS-DYNA pre 971R6 to handle 8 or 9 character parameter names
Duplication	How PRIMER handles the *PARAMETER_DUPLICATION card.
Character parameters	How PRIMER handles character parameter types.

Creating a Parameter

Parameters may be created in the following ways:

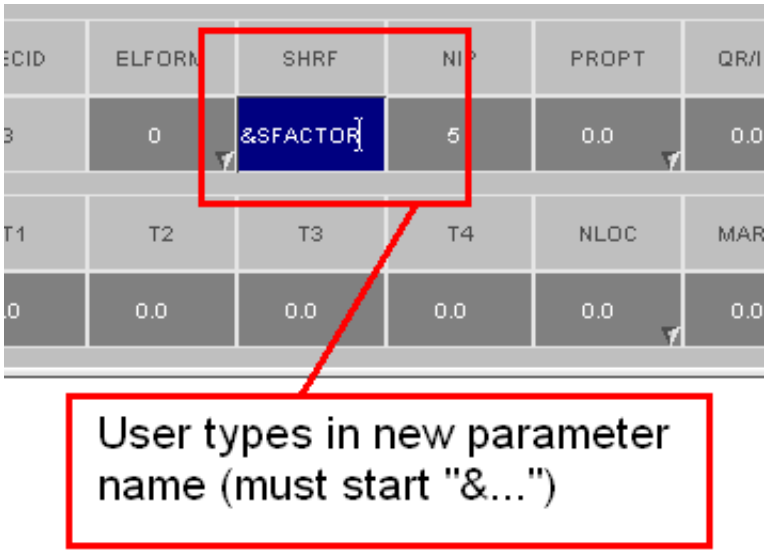
(1) By typing a new parameter name into any text entry box in an editing panel

This will map a creation panel populated with:

- the parameter name
- the type (Integer or Real) as deduced from the context
- a sensible default value for this context.

To create a character parameter type first create it in this way, and then change the given type (I or R) to C, and supply its name.

When you start typing "&..." a popup box showing all existing parameters in the model matching the characters typed so far will be mapped, allowing you to select one directly. For example if you type "**&B**" all parameters starting with "**B**" will be shown, and as you type more characters the list will be refined.



Using wildcard syntax to limit the list of parameter names displayed.

In models with many parameters this popup of matching parameter names can be very large so you can also use wildcard syntax to limit what is shown: '?' matches a single character and '*' any number of characters or none. This is best illustrated by example:

Consider a model that contains parameters **ALPHA, BAY, BAG, BACK, BODY ,B1, B10** and **CHASSIS**.

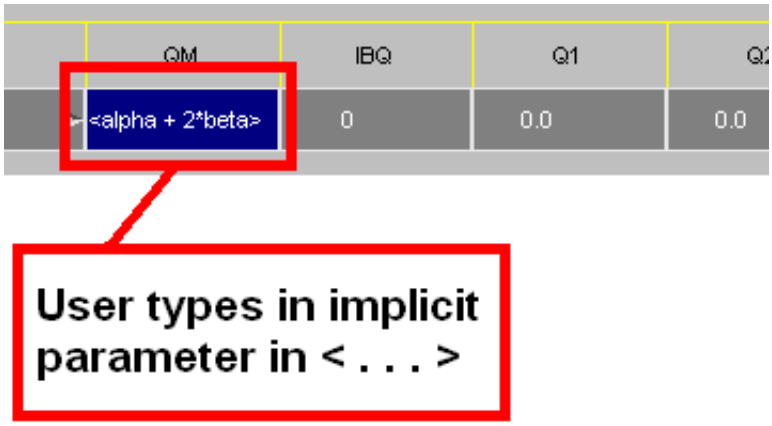
Characters typed	What will appear in the popup menu	The reason for this
& or **&***	All the parameters above	**&** on its own matches any parameter. ***** matches any number of characters.
&B	**BAY, BAG, BACK, BODY, B1, B10**	All parameters starting with **B**, equivalent to **B***.
&B?	**B1**	This is the only parameter of **B** following by one character
&B??	**BAY, BAG, B10**	These are the only parameters of **B** followed by two characters
&*A*	**ALPHA, BAY, BAG, BACK, CHASSIS**	These match [any characters or none] **A** [any characters or none]

If you use wildcard syntax remember that parameters must start with a letter A-Z, and may only contain the characters A-Z, 0-9 and _ (underscore). So if you have used wildcard syntax but have not found anything that matches in the popup list then the name you have, containing * and/or ?, is not a valid parameter name and will be rejected unless you correct it.

Creating an "Implicit" parameter in **<...>**

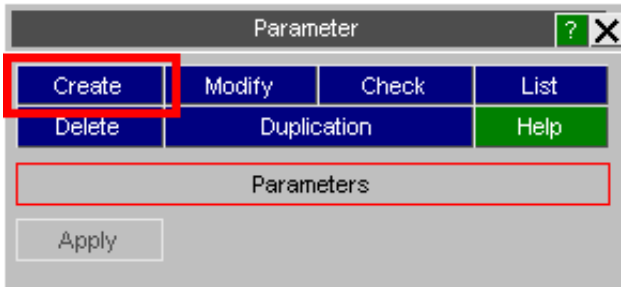
As an alternative to the &name syntax above it is also possible to create an implicit parameter expression by writing the expression directly in **<...>**

Implicit parameters are described in more detail below, but briefly they behave like ordinary parameter expressions but don't have an explicit name.



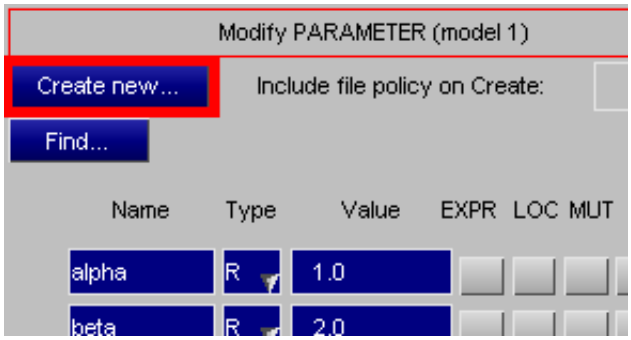
(2) From the Create button in the main Parameter panel

This will map an empty creation panel (see below).



(3) From the "**Create new...**" button on the Modify Parameter panel

As for (2) above this will map an empty creation panel.



The Create/Edit parameter panel

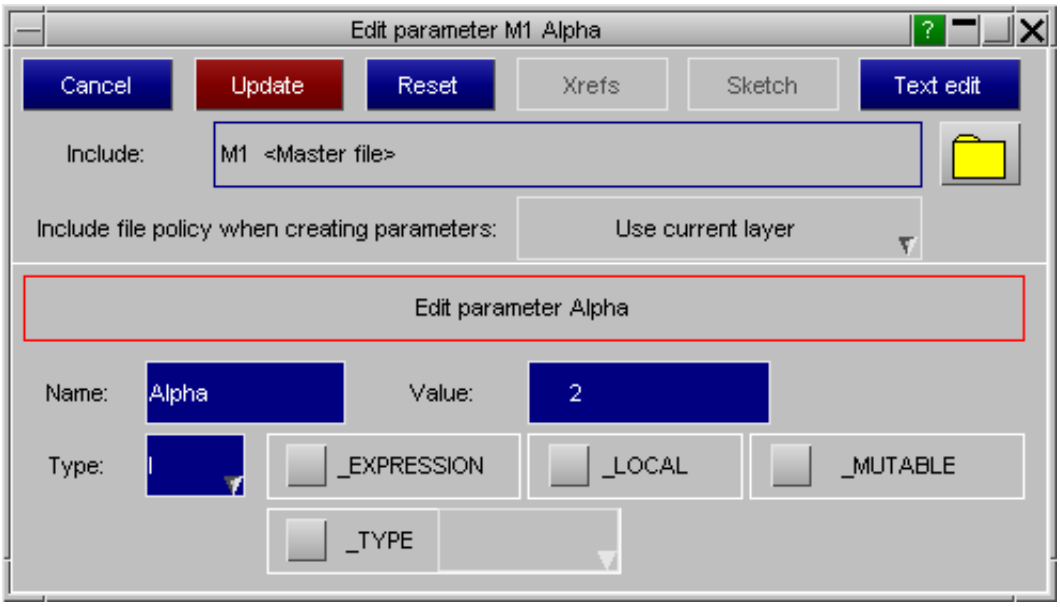
The panel allows you to edit all attributes of the parameter:

- Its name: up to 9 characters without the initial "&".
- Its type: **`I`**(nteger), **`R`**(eal) or **`C`**(haracter)
- Its value

- Whether or not it is an **_EXPRESSION** parameter
- Whether or not it is a **_LOCAL** parameter.
- Whether or not it is a **_MUTABLE** parameter

  alternatively

- Whether or not it is a _TYPE parameter, and the **`PRTYP`** argument if it is.

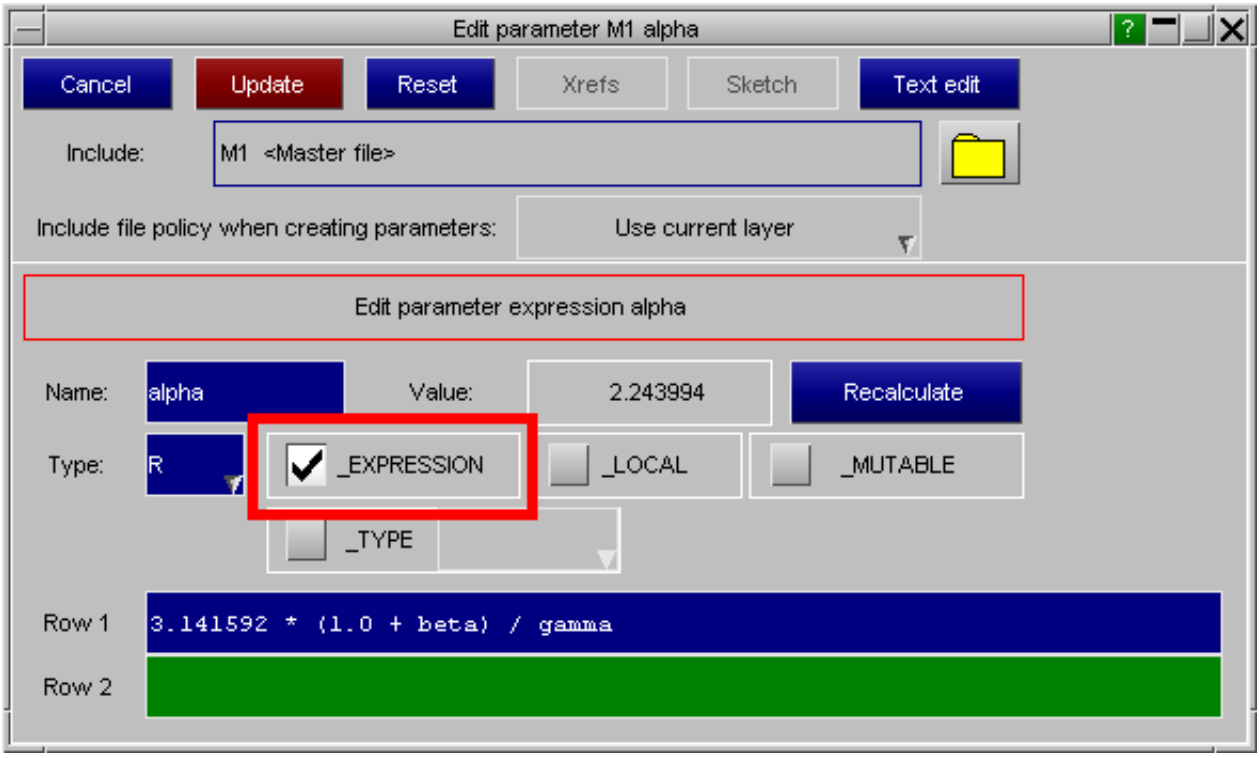


The value is simply typed into the "Value" box.

As with other editing panels this is a scratch definition, and the saved attributes of the parameter will only be updated when you use **Update**.

Using the **_EXPRESSION** option.

In the **`_EXPRESSION`** case the panel extends to provide editing rows to contain the expression.



As many rows as necessary to define the expression may be used. PRIMER will reformat the expression over as many lines as necessary to make it fit the 80 column width of the LS-DYNA input deck.

Its value is calculated when you hit <return>, but if a parameter referred to in the expression (for example Beta or Gamma here) is changed externally you can use **RECALCULATE** at any time to force a recalculation.

Arithmetic rules used to evaluate PARAMETER_EXPRESSION

PARAMETER_EXPRESSION uses syntax and standard operators common to most programming languages. The standard rules of arithmetic evaluation and operator precedence are obeyed, and built-in functions common to most programming languages are supported.

In particular note that integer expressions evaluate as integers, see Integer vs Floating evaluation of expressions below

Precedence of operators

Brackets `(..)`	are senior to
Unary `+/-` (eg -10, +3.141592)	are senior to
Raising to the power of (`**`)	is senior to
Multiply, divide, modulo (`*, /, %`)	are senior to
Add and subtract (`+, -`)	

Built-in functions

Trigonometric functions (all arguments in radians, not degrees)		**Hyperbolic functions**	
`sin(x)`	`asin(x)`	`sinh(x)`	`asinh(x)`
`cos(x)`	`acos(x)`	`cosh(x)`	`acosh(x)`

`tan(x)`	`atan(x)`	`tanh(x)`	`atanh(x)`
`sec(x)`	`atan2(x, y)`		
`csc(x)`			
`cot(x)`			
General maths functions			
`exp(x)` (e to the power of x)	`abs(x)` (absolute value of x)	`int(x)` (Integer part of x, truncated)	`float(x)` (x as a floating point value)
`log(x)` (natural log of x)	`mod(x,y)` (remainder when x divided by y)	`aint(x)` (Absolute value of integer part of x, truncated)	`ceil(x)` (nearest integer (as a float) larger than x)
`log10(x)` (log base 10 of x)	`max(x,y)` (maximum of x and y)	`nint(x)` (Nearest integer to x, rounded)	`floor(x)` (nearest integer (as a float) less than x)
`sqrt(x)`	`min(x,y)` (minimum of x and y)	`anint(x)` (Absolute value of nearest integer to x, rounded)	`sign(x,y)` (sign of y applied to |x|)

All functions return a floating point result except for:

`int(x), nint(x), aint(x), anint(x)`	All return integers
`max(x,y), min(x,y), mod(x,y), sign(x,y)`	Return integer if both X and Y are integers, otherwise floating point
`abs(x)`	Returns an integer if X is integer, otherwise floating point

Integer vs Floating evaluation of expressions

Programming languages make a distinction between an integer (eg 2) and a floating point value (eg 2.0). The same is true of expressions which give a result of the same type of their component parts, however when an expression is mixed the result is "promoted" to floating point. Generally this has little influence on the outcome, but there is one significant exception:

WARNING: integer division truncates!!

Consider the following examples

Expression	Gives result	Because
`5.0 / 2.0`	`2.5`	Both parts are floating point, giving a floating result
`5 / 2`	`2`	Both parts are integer, so result is truncated to an integer
`5 / 2.0`	`2.5`	Expression is promoted to float because ot has mixed types
`5.0 / 2`	`2.5`	Expression is promoted to float because ot has mixed types

These rules apply to subsets of expressions as well as the expression as a whole, for example:

Expression	Gives result	Because
`1.0 + 5 / 2`	`3.0`	The part (5 / 2) evaluates to 2, so result is 1.0 + 2 = 3.0 (result promoted to highest type)
`1 + 5 / 2`	`3`	All integer, so 1 + 2 = 3
`1 + 5.0 / 2.0`	`3.5`	1 (integer) + 2.5 (float) = 3.5 (float result)

Because it is easy to write an "all integer" expression by mistake when a floating point result is desired PRIMER performs a check for this on all **PARAMETER_EXPRESSION** definitions. It evaluates the parameter twice: once using the rules described above and a second time using "all floating point" arithmetic. If the results are different by more than 1 part in 1e6 of the magnitude of the result then a warning is issued stating that the result depends upon integer truncation. Almost always this is due to integer division as given in the examples above, but it is theoretically also possible as a result of the **mod(x,y)** function (or the expression **x % y**, which means the same thing) returning an integer result.

It is recommended that "real" expressions are always written using floating point numbers, ie write 2.0 rather than 2, as this reduces the chances of accidental integer truncation and the meaning is clear to anyone reading the expression.

Integer expressions may deliberately utilise integer truncation, but it is recommended that this is made explicit by using the **int(x)** function. It will make no difference to the outcome, but it will make it clear to the reader that truncation is intended.

When expressions refer to other parameters the type of each parameter's value is retained: integer parameters evaluate as integers, and real parameters as floats.

Conversion of results by Parameter type

You will see from the above that the results of parameter expressions may be floating point or integer, however parameters themselves may also be declared as being "real" (floating point) or integer.

Primer converts results as follows:

Result of evaluating expression	**Destination parameter type**	**How data is converted**
Floating point	Real	Takes result directly
	Integer	Truncates result to integer value (ie 2.5 becomes 2)
Integer	Real	Converts result to float (ie result 2 becomes 2.0)
	Integer	Takes result directly

Conversion between **R**eal or **I**nteger parameters and **C**haracter parameters is not permitted.

Precision of Parameters

Integer parameters use single precision signed integers, giving results in the range +/- 2**31 - 1; in decimal this is +/-2,147,483,647 or just over 9 significant figures.

Real parameters use single precision floating point value, giving a decimal mantissa of about 7.5 significant figures and a decimal exponent range of about 1.0e+/-38.

PARAMETER_EXPRESSIONs are evaluated as follows:

- All calculations are performed using double precision floating point variables, so all arithmetic is performed with a mantissa precision of about 16 significant figures and an exponent range of 1.0e+/-308.

- Where integer truncation is applied as described above ***within*** an expression the result is a "whole number" stored as a double precision float, retaining this precision.

- The result of a **PARAMETER_EXPRESSION** evaluation is converted to the intrinsic type of the parameter, thus:

  Real parameters convert the double precision floating result to a single precision floating value, in other words a mantissa of 16 sig figs is truncated to 7.5 sig figs, and the exponent is truncated from 1.0e+/-308 to 1.0e+/-38.

  Integer parameters truncate the double precision floating result to a single precision integer, ie a value in the range +/-2,147,483,647. Truncation to integer includes a small tolerance, the actual expression being:

  <Integer result> = "Integer part of" (<double precision floating result> + <tolerance>)

  where <tolerance> is +1.0e-6 for a +ve result, and -1.0e-6 for a -ve result.

  When a **PARAMETER_EXPRESSION** is used to calculate an integer result, typically for a label, it is advisable to use "all integer syntax" in the expression to avoid any possibility of ambiguity or rounding errors.

Parameters and Include files.

Early versions of LS-DYNA required all **`*PARAMETER`** statements to occur after **`*KEYWORD`** and before any other input card. This was quite restrictive, and the limitation was soon removed - in fact LS-DYNA 971R3 doesn't care what order parameters arrive in, of where they are written - see "the order in which parameters are defined" below.

PRIMER adopts the following strategy:

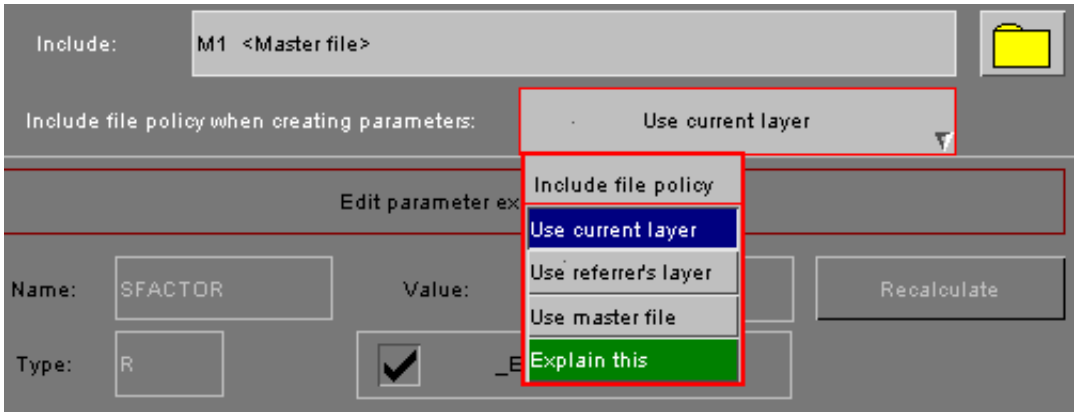
- It doesn't care where in the deck a **`*PARAMETER`** statement occurs.
- From release 9.3 onwards it no longer needs to be defined before it is used, although not doing so is deprecated.
- It doesn't care if a **`*PARAMETER`** statement occurs in an include file, and it will keep it in that include file.
- It permits parameters to be moved between include files and master file, just like any other keyword.

The use of parameters inside include files manipulated by `*INCLUDE_TRANSFORM` is a special case, please see below for more details.

The use of the **`_LOCAL`** suffix to the **`*PARAMETER`** card, introduced in LS971R5, permits parameters to be multiply defined with different values in different include files, and this affects the scope of their applicability. This is explained under "Using the _LOCAL option" below.

From LS971 R7.1 parameters may also have a **`_MUTABLE`** suffix, permitting their value to change as each duplicate definition in the input deck is read. If present, this supersedes any **`_DUPLICATION`** settings.

The policy used for the destination Include file of new parameters.



There are three options:

1. **Create the parameter in the current layer (include file).**

   This is the default, and is consistent with the creation policy for other items in PRIMER.

2. **Use referrer's layer.**

   This is only relevant in the case of a parameter being created via a new name (&xxx) being typed into a text entry box (option 1 above).

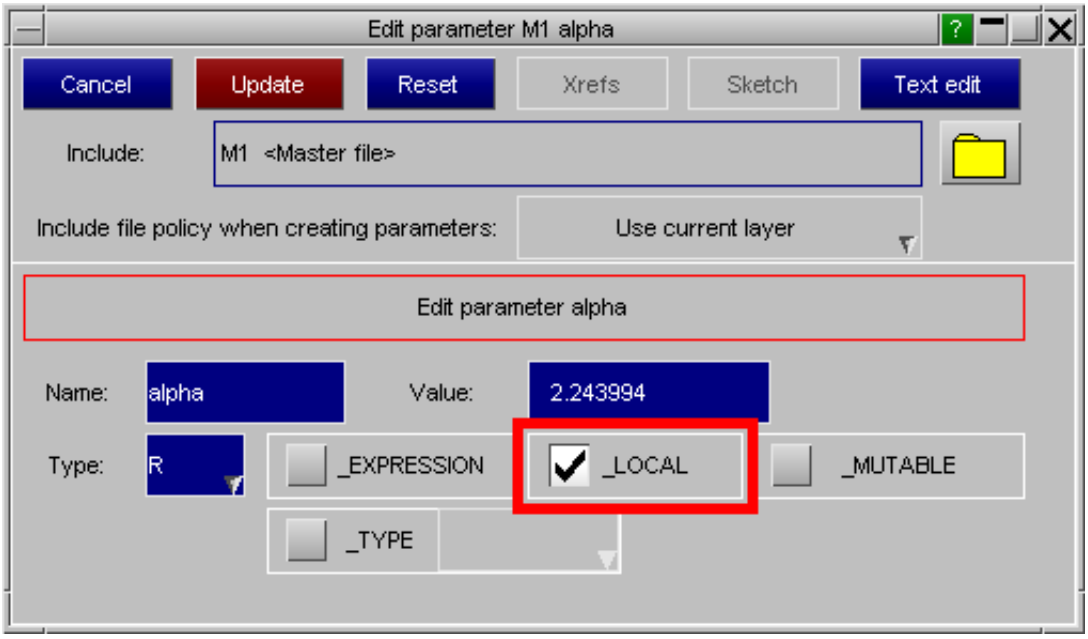
   In this mode the new parameter will "inherit" the include file of the item being edited. Parameters created by other means will use the default "current layer" to determine their include file.

3. **Use master file**.

   All new parameters are forced to reside in the master file. If the strict ls-dyna rules are applied this is the safest option, however where parameters are used in an include file it may result in "decoupling" of the parameter definition and its subsequent usage in a different context.

The include file of any parameter may be changed at will using the standard "include" editing buttons just as for any other in PRIMER.

Using the **_LOCAL** option



The suffix **_LOCAL** was introduced in LS-DYNA 971R5 (early 2011) and it is defined in the LS-DYNA user manual as working as follows:

- A conventional "global" parameter, plain or **_EXPRESSION**, is normally applicable throughout the whole model, including all include files, regardless of where or in which include file it is defined.
- A parameter with the **_LOCAL** suffix is only applicable in the include file in which it is defined.
- Multiple parameters of the same name, using different values, may exist in different include files in a deck so long as all definitions, or all but a global one, use the **_LOCAL** suffix.
- In a given include file file a parameter with the **_LOCAL** suffix defined in that file will "mask" all other parameters of the same name defined elsewhere in the deck, so that its local value is used.
- The ***PARAMETER_DUPLICATION** keyword defines how LS-DYNA will handle parameter **_LOCAL** and global name conflicts.
- The **_MUTABLE** suffix may or may not be applied at the "local" scope of a parameter if combined with **_LOCAL**. The LS-DYNA manual is silent on this point.

*PARAMETER_LOCAL rules used in PRIMER

The LS971R5 manual leaves certain details of this usage ambiguous, however reverse-engineering the behaviour of LS-DYNA suggests that the following rules are correct:

- A non_**LOCAL** (ie global) parameter can be defined anywhere in the input deck, in any include file, and it will be used to satisfy a parameter reference anywhere in the input deck except where superseded by a **_LOCAL** definition of the same name as described below.
- A **_LOCAL** parameter is valid only in the include file in which it is defined, and in any "children" of that include file, to any number of generations.
- A **_LOCAL** parameter "masks" (supersedes) a global parameter definition in the include file in which it is defined, and any children of that file to any number of generations. This is true even if a global parameter of the same name is defined in this file or its children.
- A **_LOCAL** parameter in an include file also masks a **_LOCAL** parameter of the same name in the parent (or grand-parent, to any number of generations) include file that "owns" this one. It also masks it in any children of this file.

These rules could be summarised as

- A global parameter is valid everywhere unless it is masked by a **_LOCAL** parameter in that context.
- **_LOCAL** parameters apply in the include file in which they are defined, and any child include files referenced in that file, to any include depth.
- **_LOCAL** parameters in an include file also mask other **_LOCAL** parameters defined in a "parent" include file.

How duplicate parameter names are handled

LS-DYNA is configured to permit duplicate parameter names to exist using a combination of two strategies:

1. **_LOCAL** parameters can legally mask global parameters of the same name, and they can also legally mask other **_LOCAL** parameters of the same name in different include files.

2. Duplicate global parameters may exist in a deck if a ***PARAMETER_DUPLICATION** card (see below) is defined and configured to permit this. The same card can also permit duplicate **_LOCAL** parameters in the same file.

   If duplicate global (anywhere) or **_LOCAL** (in the same file) parameters are found and no ***PARAMETER_DUPLICATION** card is found then LS-DYNA will terminate input with an error.

PRIMER V11 onwards mimics this behaviour as follows:

Once keyword input is complete all parameters are examined to see whether any name clashes exist. If there are some then an attempt is made to resolve these using global vs **_LOCAL** rules. If name clashes remain in a given scope then behaviour depends on whether or not a ***PARAMETER_DUPLICATION** card is found, and any settings on this card. The results will be as follows:

If a *PARAMETER_DUPLICATION card is present:

- If it is set to "accept" (options 2 or 4) then the ***youngest***, of any clashing definitions in a given scope will be used to resolve references to parameters of that name. For example, if a keyword referencing that parameter is defined between the first and the second definition of the parameter, it will use the first definition, whereas all keyword cards referencing that parameter name after the second definition will ignore the first definition if that parameter name. This is the same behaviour as for **_MUTABLE** parameters described below, although the **_MUTABLE** suffix will not need to be specified with this duplication setting.

- If it is set to "ignore but continue" (options 0, 1 or 5) then the ***oldest***, as in first read, of any clashing definitions in a given scope will be used to resolve references to parameters of that name. Any younger definitions in that scope will be stored and written out, but marked as ignored.

- If it is set to "ignore and terminate with error" (option 3) the oldest definition is used as above. Younger clashing definitions are stored and will be written out, but marked as "duplication error" and will also not be used.

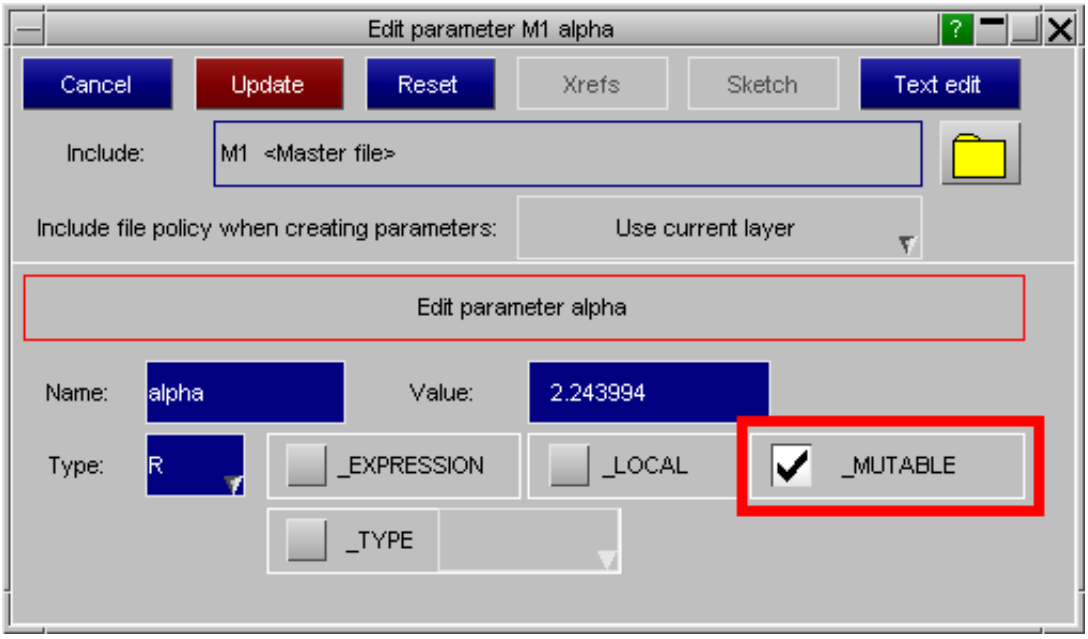
If a *PARAMETER_DUPLICATION card is *not* present:

- The effect is the same as option 3 above: the oldest definition is used, and younger clashing definitions are remembered and written out, but marked as being in error and to be ignored.

Note that there is an implicit hostage to fortune in this logic: the parameters that are actually used will depend not only on the settings of the ***PARAMETER_DUPLICATION** card, but ***also on the order in which include files are read***.

This won't matter if multiple include files contain the same (global) parameter names with the ***same*** values, since then it doesn't matter which definition is used, but if include files contain clashing global parameter names with ***different*** values then the potential for mix-ups is all too clear. In this latter situation it is strongly recommended that the **_LOCAL** suffix is used to make explicit which parameters are to be used in which files.

Using the _MUTABLE option



The ***PARAMETER_MUTABLE** suffix was added in LS971 R7.1 to provide an alternative way of handling multiple definitions of the same parameter.

The parameter can be defined multiple times in the input deck, with a different value each time.

References to it use the most recently defined value.

The rules for its use in LS-DYNA are:

- It only applies to **I**nteger and **R**eal parameter types, it is ignored for **C**haracter ones.
- The **_MUTABLE** suffix must be added to the first definition of a mutable parameter, it is optional on subsequent ones.
- The effect is that each new definition of the parameter supersedes the previous value, becoming the "current" value for any references.
- This behaviour is independent of the ***PARAMETER_DUPLICATION** setting.

To understand the implications of this more clearly consider the following simple input deck:

```
*PARAMETER_MUTABLE
I ID1                1
```
The current value of integer parameter ID1 is 1.
```
*MAT_ELASTIC
$      mid        ro         e        pr        da        db
&ID1              1.0       1.0       0.3       0.0       0.0       0.0
```
So the material above has label 1.
```
*PARAMETER_MUTABLE
I ID1               10
```
The current value of ID1 is now 10.
```
*SECTION_SOLID
$    secid    elform       aet
&ID1                1         0
```
So the section above has label 10
```
*PARAMETER
I ID1           100
```
The current value of ID1 is now 100
```
*PART
$ title
Example part
$      pid      secid       mid     eosid      hgid      grav    adpopt      tmid
&ID1                10         1         0         0         0         0         0
```
So this is part 100, referencing section 10 and material 1.

From this example several points will become clear:

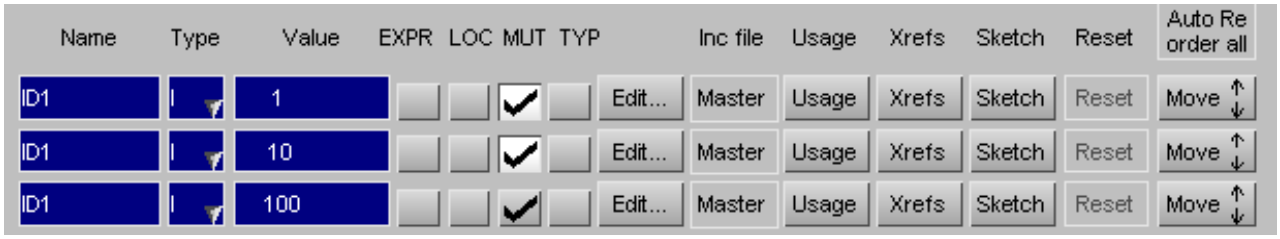
- The mutable parameter **ID1** has three distinct values: **1**, **10** and **100**.
- The order in which the parameter (re-)definitions appear in the input deck is significant.
- The order in which other cards referring to this parameter appear in the deck is also significant.

PRIMER has to handle this problem without losing definitions or muddling them up, and it does so as follows:

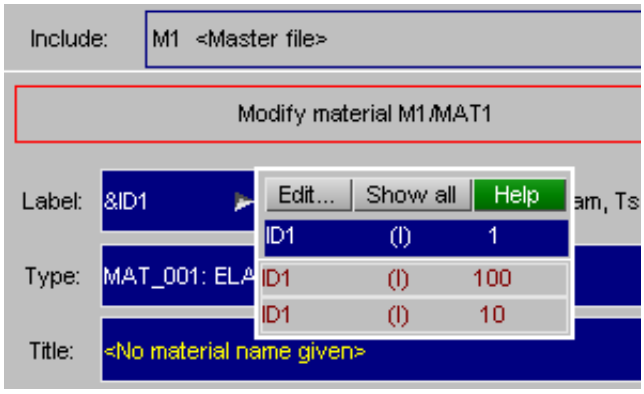
How PRIMER stores and manages mutable PARAMETER definitions.

- Each definition of a mutable parameter is stored separately as a distinct internal entity.
- Each reference to a mutable parameter "knows" which of these entities it is using.

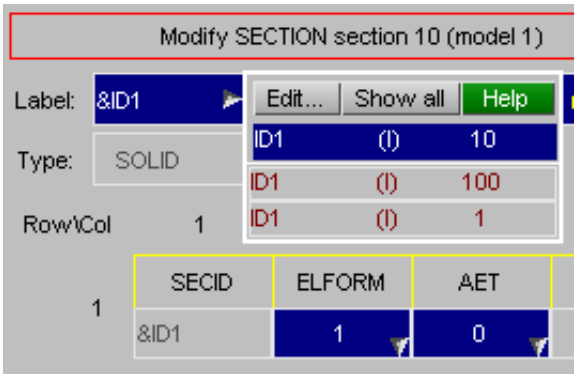
If we use the simple input deck above as an example the PARAMETER editing panel will show the following:



Each definition of ID1 is stored separately, in the order in which they appeared in the input deck.

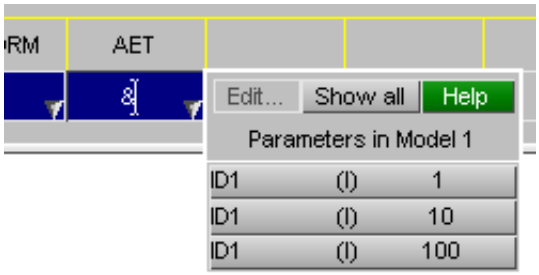


Hovering over the parameter used to define the material label shows that it uses the version of parameter ID1 with the value of 1



Hovering over the parameter used to define the section label shows that it uses the version of parameter ID1 with the value of 10

When you start typing "&" into a data field to define a parameter you will be given a list of all possible candidates to choose from.



How PRIMER deals with Mutable parameters during keyword output.

It will be clear from the discussion above that the order in which mutable parameters appear in the output deck, with respect to where they are referred to, will affect the value assigned to each reference. PRIMER does not always preserve the order of cards in an input deck since it sorts labelled items into ascending order, and moreover edits made during a PRIMER session may affect the order in which keywords are written.

In order to solve this problem PRIMER writes out mutable parameters in the following way:

(1) All mutable parameters in a given master or include file are written out at the top of the file along with other other parameter definitions.

So on keyout the simple deck used in the example above will write the following:

```
$
$ ===============
$ PARAMETER cards
$ ===============
$
*PARAMETER_MUTABLE
$PR_MPTAG 1
I ID1               1
$PR_MPTAG 2
I ID1              10
*PARAMETER
$PR_MPTAG 3
I ID1             100
```

This is done to establish the "native" location for mutable parameters, which may be necessary if definitions are repeated in include files as described in (4) below. It also defines the "tags" assigned to each parameter, as described in (3) below.

(2) If a reference to a mutable parameter does not use the "current" value a new *PARAMETER card is written.

Each time a mutable parameter card is written that becomes the current value of that parameter. So in the deck in (1) above the current value of ID1 is 100 at the end of the *PARAMETER cards, since that is the most recently written value.

However the *SECTION definition in that deck uses the version of ID1 that has the value 10, so immediately before the section card is written out a new *PARAMETER card for ID1 with the value 10 is output:

```
*PARAMETER_MUTABLE
$PR_MPTAG 2
I ID1              10
$
*SECTION_SOLID
$    secid    elform       aet
&ID1                1         0
$
```

Thus the current value of ID1 is now 10. Likewise the *PART definition in that file uses the version of ID1 with the value 100, so that parameter value is written first:

```
*PARAMETER_MUTABLE
$PR_MPTAG 1
I ID1             100
$
*PART
Example part
$      pid     secid       mid     eosid      hgid      grav    adpopt
tmid
&ID1               10         1         0         0         0         0
0
```

(3) Each parameter is "tagged" with its own unique $PR_MPTAG value to avoid fragmentation on reread.

It will be clear that in a reordered deck the same version of a parameter might be written out several times, and while this would be correct in the sense that the right values would be used it could lead to "fragmentation" of the parameter into several redundant instances of the same value if a deck is written out of PRIMER and then read back in again.

To avoid this, and also to prevent internal confusion, PRIMER assigns a unique internal "tag" to each mutable parameter. This is a small integer which is unique to that version of the parameter in this PRIMER session. These tags may not be sequential, and their values may change in successive PRIMER sessions, so no meaning should be inferred from them - their sole purpose is to distinguish between versions of the same parameter.

On keyword output a comment line starting $PR_MPTAG is written between the *PARAMETER keyword and the definition of the parameter definition itself, and this can be seen in the snippets of output above, for example:

```
*PARAMETER_MUTABLE
$PR_MPTAG 1
I ID1             100
```

When PRIMER rereads the input deck it looks at these "tag" values and amalgamates multiple definitions which have the same tag into a single definition, thus avoiding multiple rendundant instances of the same parameter.

(4) The "current" definitions for keyout of mutable parameters are reset in and after include files.

A further problem is the sequence

```
*PARAMETER_MUTABLE
$PR_MPTAG 1
I ID1             100
$
$
*INCLUDE
filename
```

This presents two difficulties:

- If parameter ID1 is used inside include *filename* then it will assume the most recent definition in the parent file. But if the definitions in the parent file change, or the order of writing include files changes, this may cease to be valid meaning that without any changes in the include file itself its contents could change on reread.

- Include *filename* may itself contain a *PARAMETER card that changes the current value of parameter ID1, meaning that the parent deck can not safely assume that the current value before the *INCLUDE card is still the same after it.

To avoid this problem PRIMER enforces the following rules:

- When writing out an include file the "current" value of all mutable parameters in the model is set to "undefined" at the start of that include file.

  This means that any reference to a mutable parameter within that file will trigger at least one *PARAMETER keyword for that parameter in order to make its current value explicit. This makes the include file free-standing, and also immune to external changes or the order in which include files are output.

- In the parent file (which contains the *INCLUDE keyword) the current value of all mutable parameters is set to "undefined" after any *INCLUDE keywords which define include files.

  This means that reference to a mutable parameter in the parent file will also trigger a fresh *PARAMETER card.

To make this clear the following flow diagram explains the implicit logic used during keyword output of a deck containing mutable parameters. Parent file in left column, include file in right column.

In Parent file	In Include *filename*

`*PARAMETER_MUTABLE`
`$PR_MPTAG 1`
`I EXAMPLE 3`
`$`
`$`
Current value of **EXAMPLE** for keyout is now **3**
`*INCLUDE`
`filename`	

Writing can be thought of as switching from parent file to include file filename ===>
 	
*(For clarity it should be stated that PRIMER does not actually write out include files in the sequence here, interrupting the output of "parent" in order to write "child", then resuming the output of parent. Rather each file is written out separately in its entirety. This flow diagram represents the **implicit** logic used and, more to the point, enforces this logic when the deck is reread.)*
	At the top of this include file PRIMER unsets the current value of all mutable parameters for keyout. Therefore when **EXAMPLE** is referred to, here on a *SECTION card with the value of 3, a new *PARAMETER card is written, *even though the current value in the parent file was also 3*.
	`*PARAMETER_MUTABLE`
	`$PR_MPTAG 2`
	`I EXAMPLE 3`
	`$`
	`*SECTION_SOLID`
	`$ secid elform aet`
	`&EXAMPLE 1 0`
	`$`
	
	<=== At the end of the include file writing can be thought of as switching back to the master file.

On return from the include file PRIMER resets the current value of all mutable parameters for keyout to undefined. Therefore when parameter **EXAMPLE** with the value 3 is referred to a new *PARAMETER card is written out, *even though the current value before the previous *INCLUDE card was 3*.
`*PARAMETER_MUTABLE`
`$PR_MPTAG 2`
`I EXAMPLE 3`
`$`
`*MAT_ELASTIC`
`$ mid ro e pr`
`&EXAMPLE 1.0 1.0 0.3`	

This behaviour means that PRIMER may write out more definitions of a mutable parameter than are strictly necessary, but this behaviour is conservative and should minimise the chances of the wrong version of a mutable parameter being used.

Using **`*PARAMETER_MUTABLE`** with **`_LOCAL`**

The LS-DYNA manual is silent about the scope of these parameter names if used in conjunction with the **`_LOCAL`** suffix and the actual behaviour in LS-DYNA is not known at the time of writing (Oct 2013).

PRIMER assumes that the **`_LOCAL`** suffix is honoured, so that a "local" parameter is independent of a global one or a different local one of the same name, meaning that the **`_MUTABLE`** suffix is only applied to other parameters in that local scope. However this is only a conjecture and may not replicate LS-DYNA behaviour correctly.

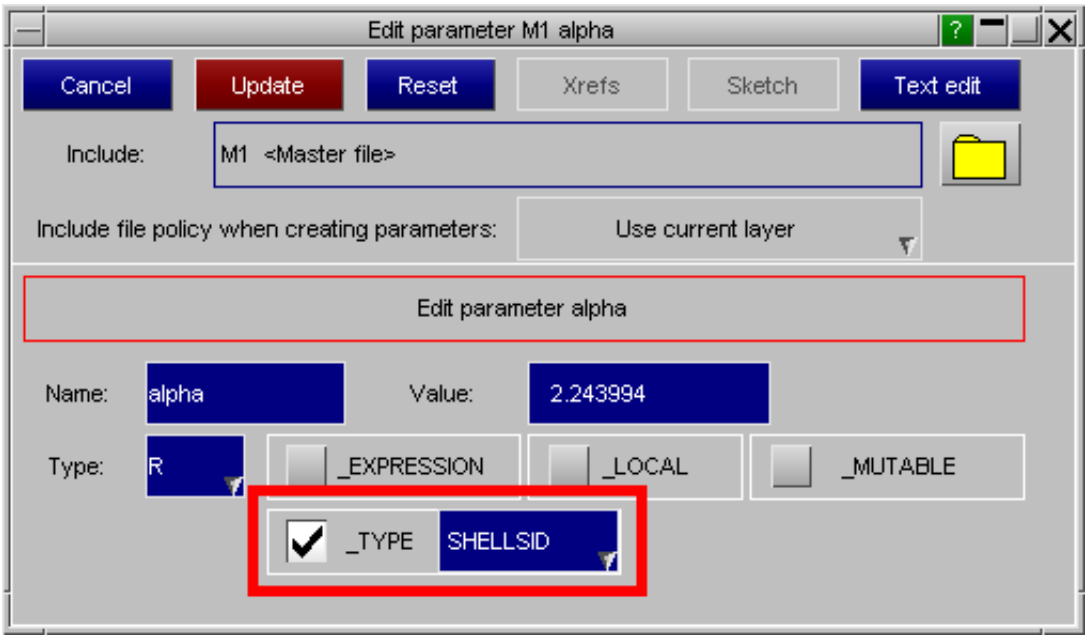
Pending resolution of this point ***it is recommended that the `_MUTABLE` suffix is not used in conjunction with `_LOCAL`.***

`*PARAMETER_MUTABLE` is not recommended!

From the discussion above it will be clear that the use of mutable parameters leaves considerable scope for confusion, unnecessary duplication (fragmentation) and possibly even errors if a deck containing them is re-ordered without sufficient care.

For these reasons their use is not recommended.

Using the **_TYPE** option

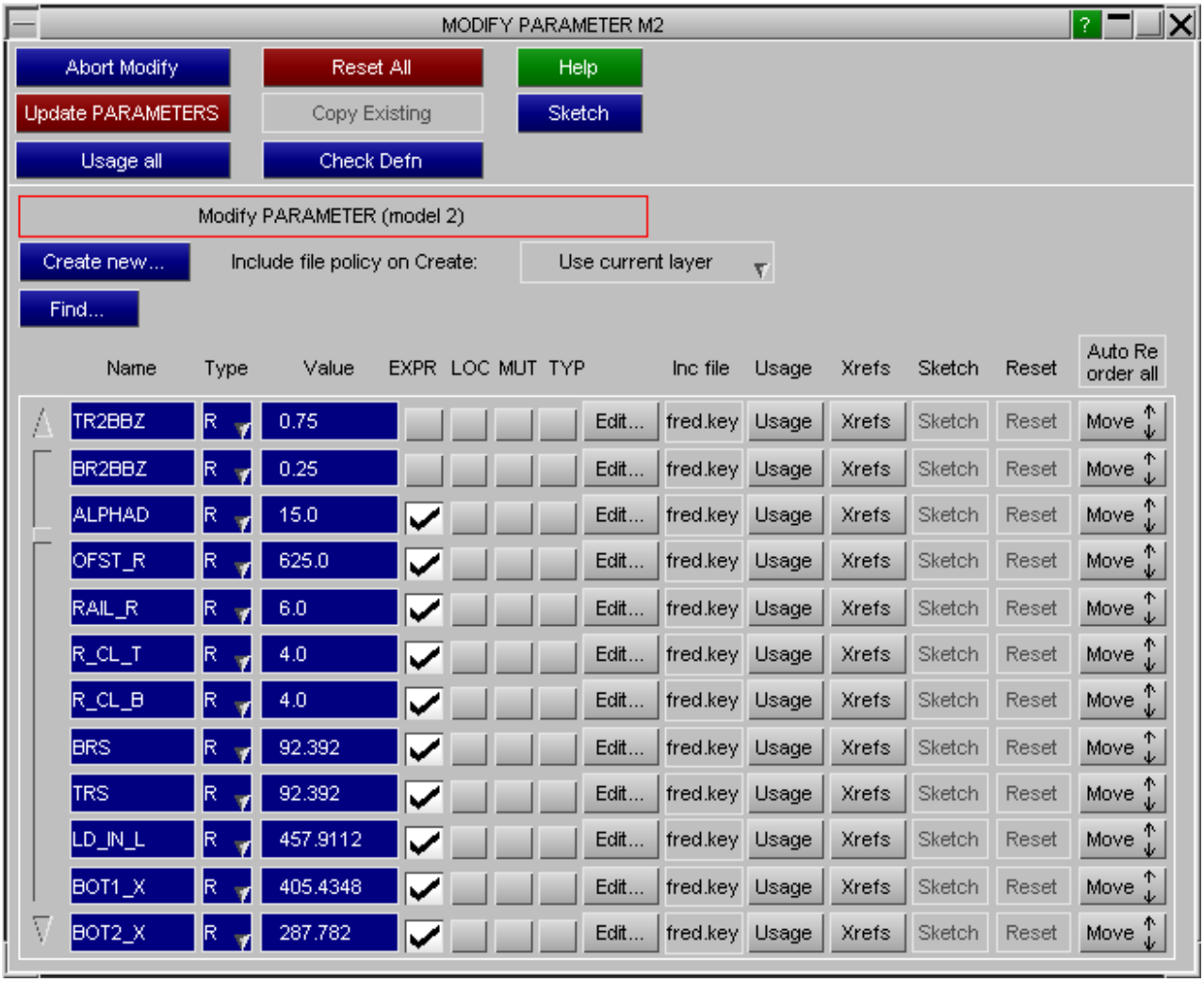


***PARAMETER_TYPE** was introduced in LS971 R7.1 and is intended to provide LS-PREPOST with information about how a parameter will be used in the analysis.

The **_TYPE** suffix is incompatible with **_EXPRESSION**, **_LOCAL** or **_MUTABLE** suffices, and PRIMER will treat an attempt to combine these suffices as an error.

Modifying Parameters

Parameter modification can be carried out at two levels:
The top level, shown here, lists all parameters in the model showing a summary of their attributes.



Changes made in this panel are applied to a scratch definition, and the permanent parameter definitions will not be updated until **Update PARAMETERS** is used. For more detailed editing, necessary when changing an EXPRESSION, use the **Edit...** button to map the detailed editing panel above.

The **Find...** button is an alternative way of selecting parameters to edit which may be useful in a model with many parameters. It works as follows:

- It maps a scrollable standard object menu of all parameter names.
- The standard filtering options, available from the **Filter** button in that menu, can be used to limit the list
- You select the wanted parameter from the list.

You may edit the following at this level.

Row header	Function
Name	Change the name of the parameter
Type	Swap between **R**(eal), **I**(integer) and **C**(haracter)
Value	The parameter value. For `_EXPRESSION` parameters changing this value will replace the expression with a constant value.
EXPR(ession)	Whether or not the parameter is an `_EXPRESSION` type
LOC(al)	Whether or not the parameter is `_LOCAL`

MUT(able)

Whether or not the parameter is **_MUTABLE**.

LS-DYNA requires the first of a series of **_MUTABLE** parameters to use this suffix, but it is optional for the 2nd and subsequent ones. Therefore the 2nd and subsequent **MUT** boxes will always be "ticked" but may not be "selected" unless the original input deck used the **_MUTABLE** suffix or the user has selected it explicitly. Consider the following example:

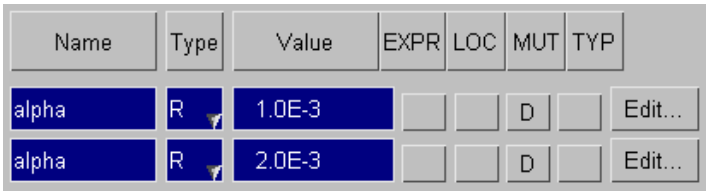
Here parameter alpha has been defined twice:

1st occurrence in the input deck:	***PARAMETER_MUTABLE**	Explicitly _MUTABLE, given the value 2.0
2nd occurrence in the input deck:	***PARAMETER**	Implicitly _MUTABLE, given the value 1.5

In this example the 2nd definition "wins", and the value used for this parameter in the model will be 1.5, which is why the first definition is dark grey since it is remembered but not used.

Also the **MUT** tick-box for the 2nd definition is "ticked but not selected", meaning that the **_MUTABLE** suffix will not be used in the keyword output file, relying on the fact that LS-DYNA does not require this.

When a **PARAMETER_DUPLICATION** card accepting multiple definitions (setting 2 or 4) is present, PRIMER internally treats all parameters with multiple instances of the same name as mutable. If they are not explicitly defined as **_MUTABLE**, they appear with a D symbol in the **MUT** column:

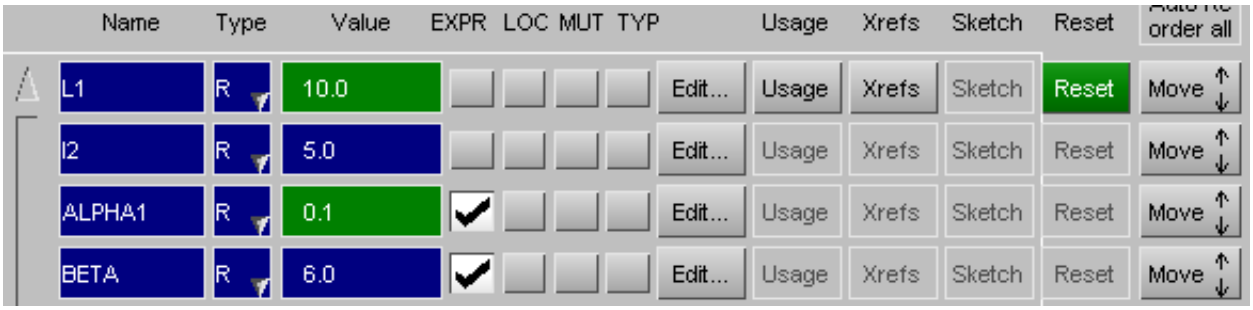


TYP(e)

Whether or not the parameter is **_TYPE**

The **_TYPE** suffix cannot be used in conjunction with the other suffices above.

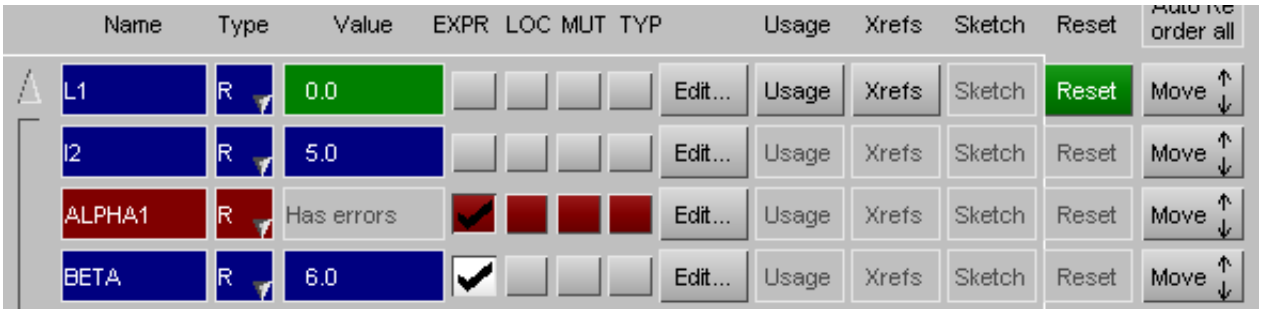
Changing a Parameter's value



When a parameter's attributes are changed the revised values are shown on a green background. In this example **L1** has been changed from 4.0 above to 10.5. In addition because the **EXPRESSION** in parameter **ALPHA** also refers to **L1** it too has changed.

Changes are "scratch", as explained above, and can be undone using the **Reset** button.

Dealing with errors caused by changes.



Sometimes changes to one parameter will create errors in an **`EXPRESSION`** that uses them.

In this example the parameter **`ALPHA`** has been changed to include "**`1.0 / L1`**" and the value of **`L1`** has been changed to zero. This gives a divide by zero error when evaluating **`ALPHA`**, and it is shown on a red background to emphasise this.

PRIMER can tolerate arithmetic errors in expressions, and substitutes a value of zero. However there is no guarantee that the analysis code will handle this and you will be warned if you attempt to update and save any errors.

Parameters that have clashing (duplicate) names

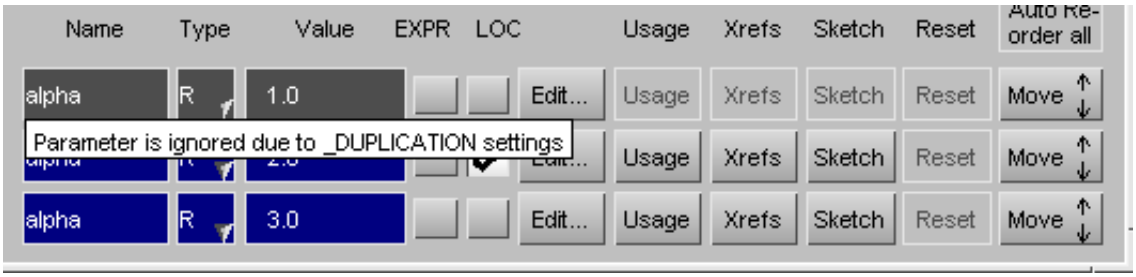
The example below uses three nested files, all containing a **`*PARAMETER`** definition called "alpha":

1. Master file, containing a global parameter called alpha, value 1.0, which includes ...
2. Child include file, containing **`_LOCAL`** parameter called alpha, value 2.0, which includes ...
3. Grandchild include file, containing another global parameter called alpha, value 3.0.

So the definition in file 2, being **`_LOCAL`**, does not conflict with the other definitions, but the two global definitions in files #1 and #3 are in conflict.

Where name clashes are resolved by a **`*PARAMETER_DUPLICATION`** card then rows which will be ignored are shown on a dark grey background, and hovering over the name will explain why the parameter is being ignored. You will also see that it has no "Usage" or "Xrefs".

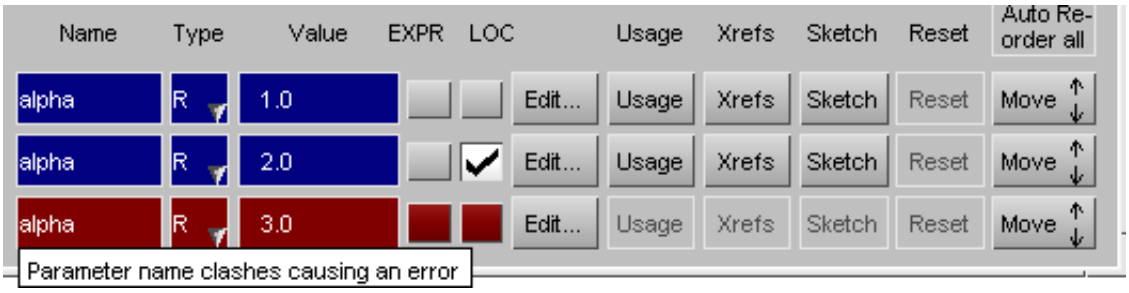
In this example the **`*PARAMETER_DUPLICATION`** card has been set to "accept", so the younger of the two clashing global definitions called "alpha" is used, and the older is ignored.



If there is no ***PARAMETER_DUPLICATION** card present, or its value of **DFLAG** is set to 3 (ignore and terminate with error) PRIMER will mark clashing parameter names on a red background, and hovering over the name will explain the cause.

Here is the example above with the ***PARAMETER_DUPLICATION** card removed, which has two effects:

- The effect is the same as "ignore and error", so the first encountered global definition of "alpha" is used.
- The 2nd definition is both ignored, and also treated as an error.



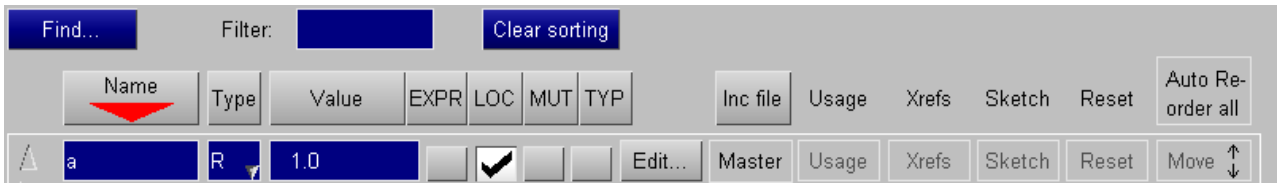
Note that the above logic can be superseded by the use of the **_MUTABLE** suffix on the ***PARAMETER** keyword. This permits multiple parameters in the same scope to have the same name so long as the first definition has the **_MUTABLE** suffix. The parameter can have multiple definitions, each with a different value, and usage on cards that reference the parameter depends on which value was "current" at the time of reading. See the section on Using the _MUTABLE option for more information.

Filtering and sorting parameters

By typing a wildcard pattern in the **Filter** box the panel only shows parameters whose name can be obtained from the pattern by replacing ? with any one character and * with any (possibly empty) sequence of characters. For example, when typing **a***, all parameters with name starting with **a** are shown. When typing a string without ? or *, only parameters matching the exact name appear on the panel.

The parameters on the panel can be sorted by name, type (character, integer or real), value, **_EXPRESSION** suffix, **_LOCAL** suffix, **_MUTABLE** suffix, **_TYPE** suffix or include file by clicking the column header. When clicking the column header again, the list will be sorted in reversed order.

The **Clear sorting** button resets the order of the parameters on the panel to their original order, which is the order in which the definitions have been read or they have been created.

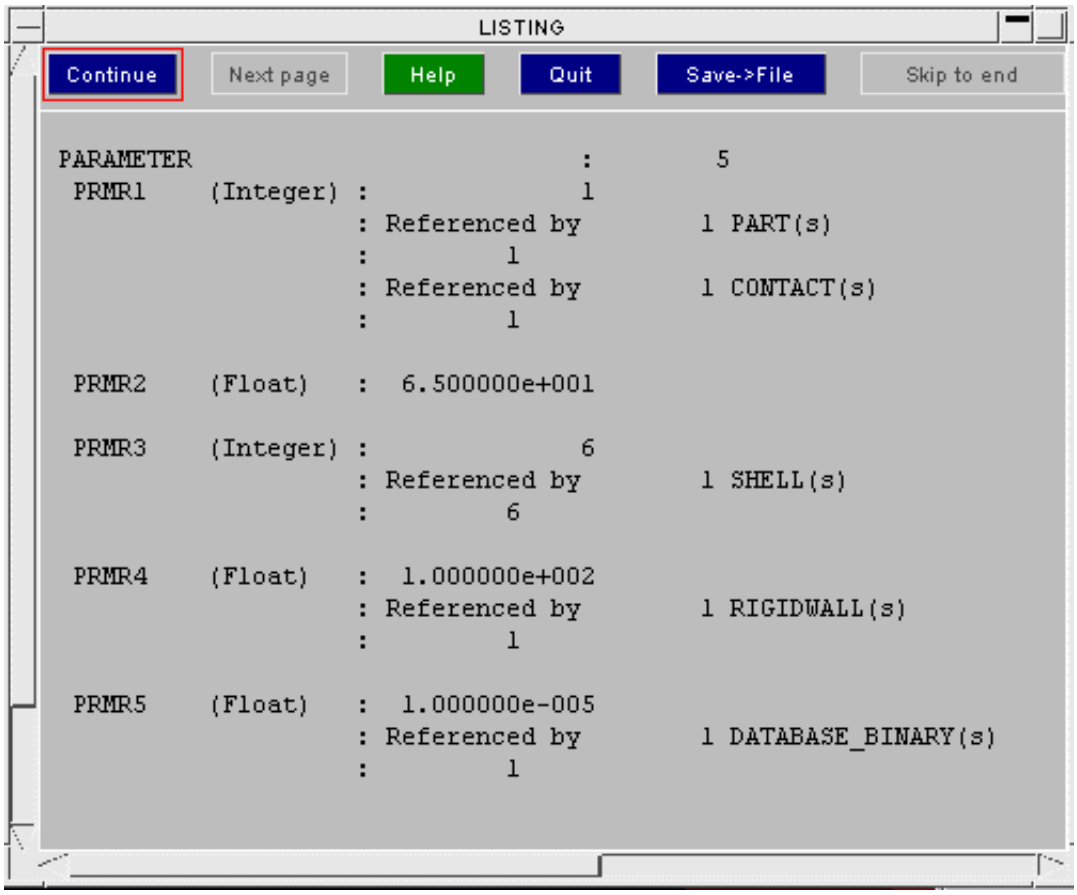


Check

This checks all parameter cards for errors. It is can also be accessed from the **CHECK_DEFN** button in the **Modify** window.

List

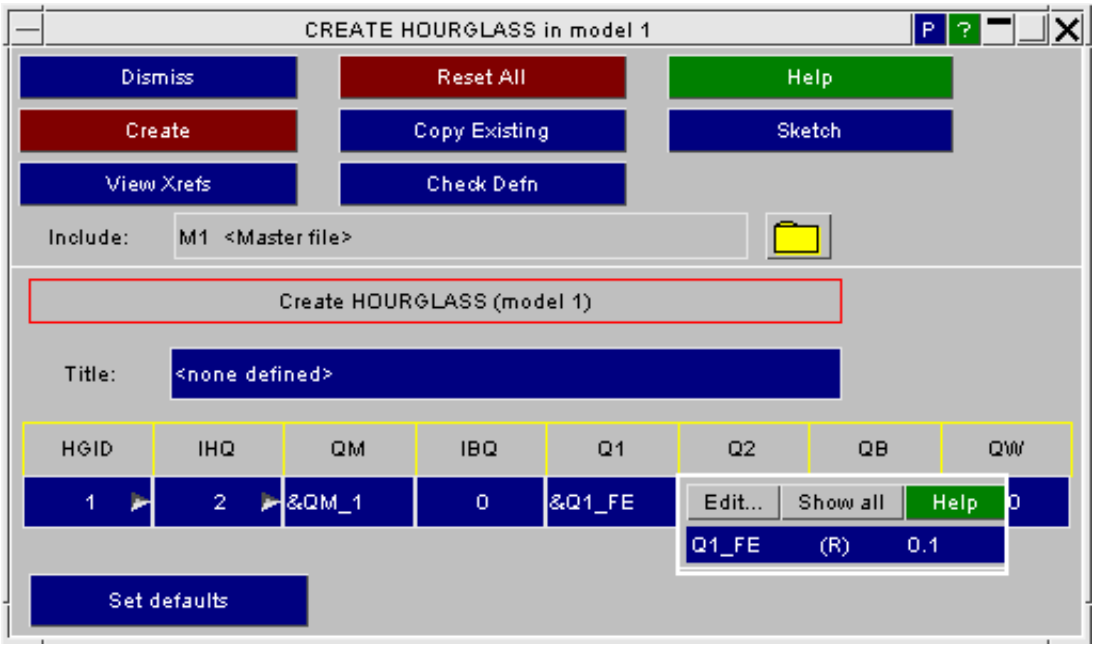
This provides a listing of all the parameters in the model and displays their numeric type, Value and what they are referenced by. The same listing can be accessed from **USAGE_ALL** in the **Modify** window.



DELETE deleting parameters.

Parameters may be deleted using the normal deletion logic in PRIMER. If they are referred to anywhere in the model, or within another **`PARAMETER_EXPRESSION`** statement, they will be locked against deletion.

Parameters in Editing Panels



From PRIMER 9.3RC2 onwards editing panels, including generic keyword editor ones, display parameters either by

name or by value wherever they are used.

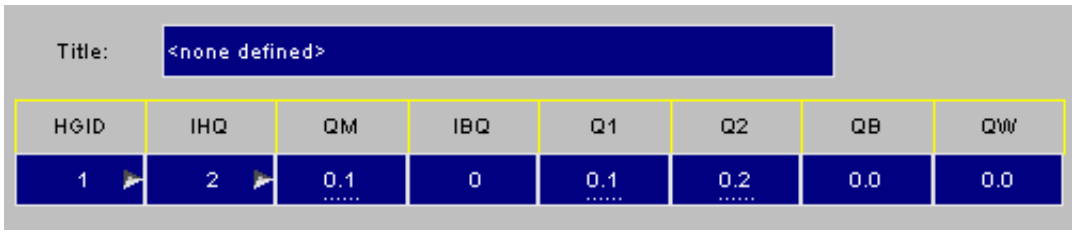
By default they are displayed in exactly the same format as that used in keyword output files: **&NAME**, but this can be toggled to show values instead - see below.

Hovering the mouse over such a parameterised field will display a popup box giving the parameter name, type and value. In this example the user has hovered over data field **Q1** which uses the parameter **&Q1_FE**. The popup box reveals that its value is **0.1**.

To edit this parameter use **Edit...**

Using

to toggle Parameter display mode.



An alternative to **&NAME** symtax is to show the value of the parameter, but underlined to emphasise that the value shown is parameterised. You can toggle between the two modes using the "**P**" button at the top right hand corner of any panel showing parameters.

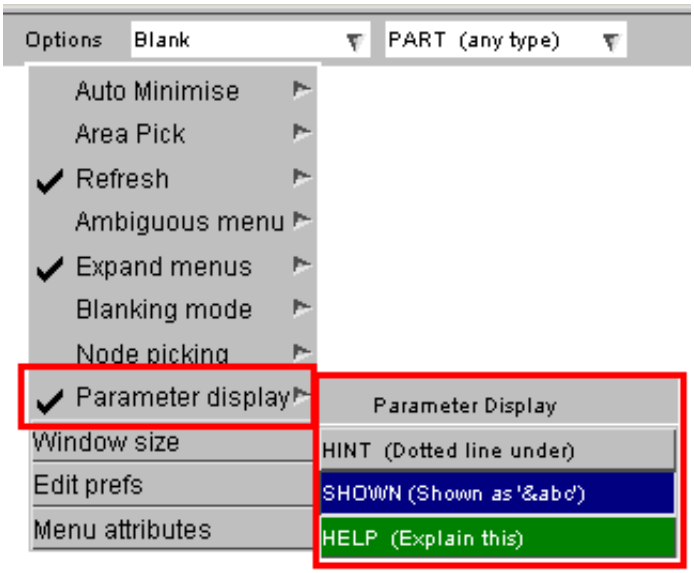
This image shows the panel above displayed in this alternative format. Hovering the mouse above an underlined field will map the parameter popup box as shown above, giving information about the parameter name.

Setting programme-wide parameter display mode.

The "**P**" button on an editing panel only controls display for that panel.

To set the default display mode for the whole programme use **Options, Parameter Display** and select:

HINT	Numeric values underlined with dots
SHOWN	Shown in **&NAME** format



Entering parameters in edit panel data fields.

In any data field where a parameter could be used in an LS-DYNA input deck the equivalent edit panel field may also use one. To enter a parameter do the following:

- Type an ampersand "**&**" into the text box field
- This will immediately map a popup box showing all currently defined parameters (if any exist)
- To choose an existing parameter simply select it from the popup.
- To narrow down the choice from many parameters type its initial letter(s), the list will contract to show only those that match.
- To enter a new, previously undefined, parameter type in its full name.

In the last case, entering a new parameter, the following happens:

- A new parameter definition using that name is created. It will be a plain parameter, ie not _EXPRESSION.

- It is given a type (Integer or Real) appropriate to the context
- It is given an initial value of zero, unless that would be illegal in that context in which case the smallest legal value is used.
- This value is entered into the data field, and the edit panel is updated.

The editing panel for the new parameter is then mapped, and you are invited to update its attributes. When you save these then its value on the "parent" editing panel will also be updated.

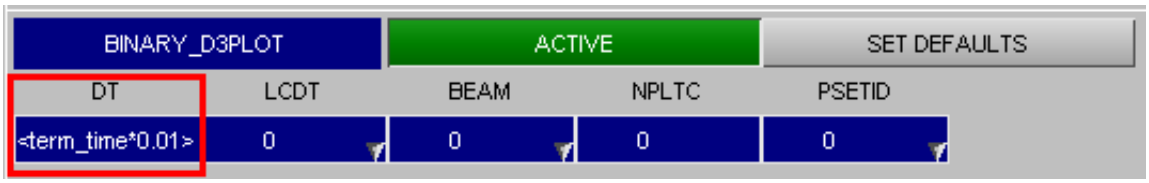
How to remove a parameter association.

To change a parameterised value to a plain number simply rub out the **&NAME** (or underlined) field, and type in a simple number. This will break the association between parameter and data field, even if the number you type in is the same as the parameter's value.

"Implicit" parameters

LS-DYNA permits the following syntax in any data field: **<expression>**

Where ***expression*** can be any parameter definition that would constitute a valid ***PARAMETER_EXPRESSION.** Here is an example on the ***DATABASE_BINARY_D3PLOT** card where the timestep dt has been set to the termination time * 0.01.



The rules for this in LS-DYNA are:

- The format of the keyword line must be comma-separated.
- The expression must be valid and enclosed in "<" ... ">".
- It must only refer to other parameters if they have previously been defined.
- The type (floating or integer) of the expression is not explicit.

The effect of using this syntax is identical to defining a ***PARAMETER_EXPRESSION *name*** using this expression, and referring to it by ***&name***.

How implicit parameters are handled inside PRIMER

In the following discussion the shorthand **`<...>`** refers to a valid expression inside "<" and ">" characters, for example **`<time + dt / 2.0>`**.

During keyword input:

- When the **`<...>`** syntax is encountered PRIMER creates an internal **`*PARAMETER_EXPRESSION`** definition that uses this string, of type **real**.
- This internal definition is given a name **`#P`*`nnn`*** where *nnn* is a unique integer starting from 1, and it is marked as being "implicit". This nomenclature is deliberately chosen to be an illegal parameter name in LS-DYNA in order to distinguish it from normal parameters inside PRIMER, and the differences between it and a normal parameter are described below.
- The data field is given the value of this expression, and is associated with the internal parameter. If the data field is in fact integer the nearest integer to the floating point (real) value will be used.
- If two of more <...> data fields use the same expression PRIMER will amalgamate them into a single parameter definition. However if the contents of a data field are subsequently edited the revised parameter in that field will become unique and other fields which originally shared the common definition will be left unchanged.

When editing values interactively in editing panels.

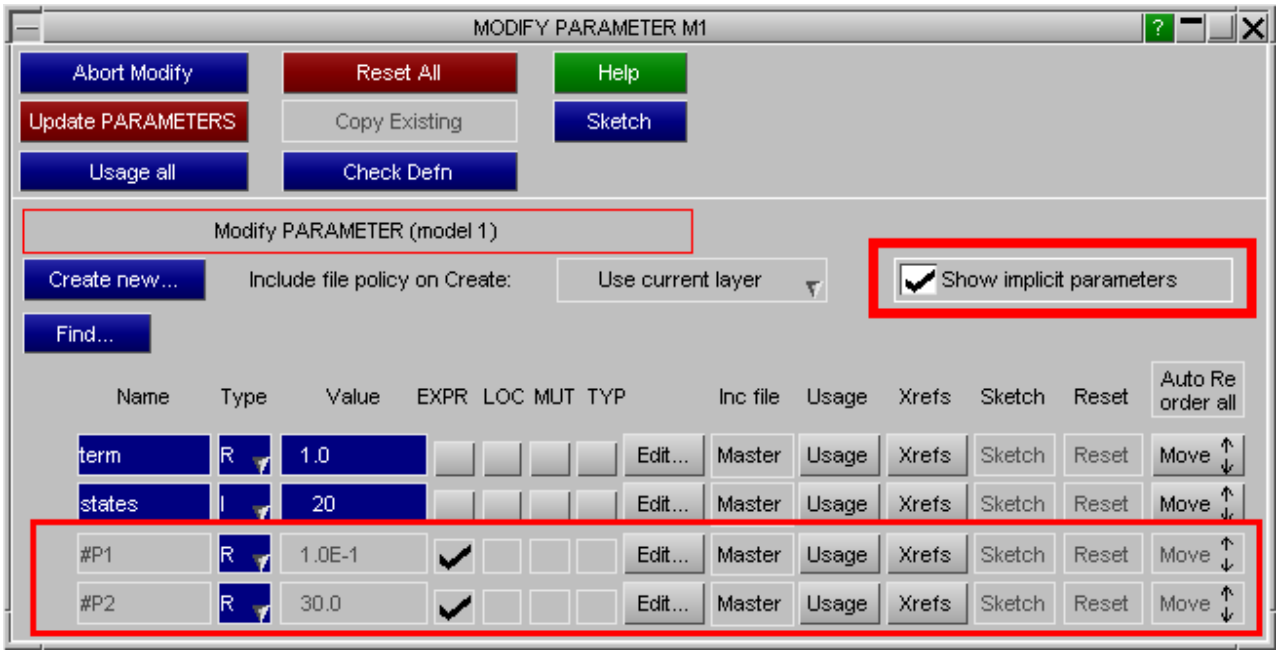
- The expression **`<...>`** rather than a reference to the parameter (**`&`*`name`***) is shown.
- The expression can be changed at will by simply typing in changes, alternatively by hovering over the expression the underlying parameter can be selected for editing.
- A new implicit expression can be created by replacing a number with an expression in **`<...>`**
- An implicit expression can be deleted by replacing it with an explicit number, or by a conventional **`&`*`name`*** parameter reference.

During keyword output

- The expression **`<...>`** is output in the relevant data field, and the line is converted to comma-separated format.
- The internal implicit parameter definition itself (**`#P`*`nnn`***) is ***not*** written out to the keyword output file.

Differences between implicit and normal parameters in PRIMER

These internal parameters are not normally shown in the top level **PARAMETER** editing panel, and it is necessary to turn on **Show implicit parameters** for them to be visible, as shown below.



Certain attributes of these parameters cannot be changed: they are forced to be of type **_EXPRESSION**, and cannot use **_LOCAL, _MUTABLE** or **_TYPE** suffices. You will observe that these buttons are greyed out for the implicit parameters in the image above.

Their name cannot be changed, and the syntax **#P*nnn*** would be illegal in LS-DYNA anyway. It is used to reinforce the point that these are not normal parameters.

It is normally the case that each implicit parameter is used only in a single context, which is where it has been written as **<...>** in a data field. PRIMER will permit you to enter **&#P*nnn*** into a data field to refer to an internal parameter, or to select it from a menu of parameter names, however the data field will always show the **<...>** expression rather than the **&*name*** reference.

The effect of editing a **<...>** field depends upon how you go about it:

- If you edit the **<...>** expression in the data field itself this will create a new **#P*nnn*** parameter exclusively for this data field, and any association with the "old" parameter definition will be lost. The old parameter remains in the model, and if it is used for any other data fields these will not be affected, otherwise it becomes an unused orphan. These orphans do no harm, but they can be deleted explicitly if desired using **Remove**, or automatically via **Cleanup Unused**.

- If instead you use the parameter editing panel to edit the **#P*nnn*** parameter definition itself this will behave like editing a normal parameter: the data field will continue to refer to the parameter and it will be updated as necessary to show the revised expression. Any other data fields referring to this parameter will also be updated.&

To avoid possible confusion it is recommended that you do not make explicit references to implicit **#P*nnn*** parameters by name, preserving their one-to-one relationship with **<...>** data fields. In particular do ***not*** attempt to refer to **#P*nnn*** parameters by name in other parameter expression definitions: this will not work because **#P*nnn*** parameters are not written out to the keyout file.

Support for LS_OPT expressions

LS-OPT will replace constants that are the subject of optimisation with the following syntax:

```
<<expression(:(i)nnn>>
```

Where:

- ***expression*** is an externally defined variable name
- **:(i)nnn** is an optional format statement modelled on Fortran syntax

For more information about this syntax see section 5 of the LS_OPT user manual.

This presents a problem for PRIMER since the external variable names and values are unknown, making it impossible to evaluate the expression. Therefore it adopts the following strategy:

- It creates an "implicit" PARAMETER_EXPRESSION with a synthetic name starting **#L...** which contains the string inside <<...>>
- This expression is stored but not evaluated, rather the parameter is given a constant value of 0
- Thereafter it behaves like a normal implicit parameter except that you can update the constant value assigned to it at will.
- Updating the numerical value does not affect <<expression>>, the two are decoupled.
- You can also change the contents of <<expression>> if you wish, which will not affect the numerical value used inside PRIMER

During keyword output any value assigned inside PRIMER is ignored and the current <<expression>> is written out verbatim in any data fields where it is used.

A special case is where a parameter itself is defined by an <<expression>>, for example:

```
*PARAMETER
R PNAME_123 <<variable_name:i10>>
```

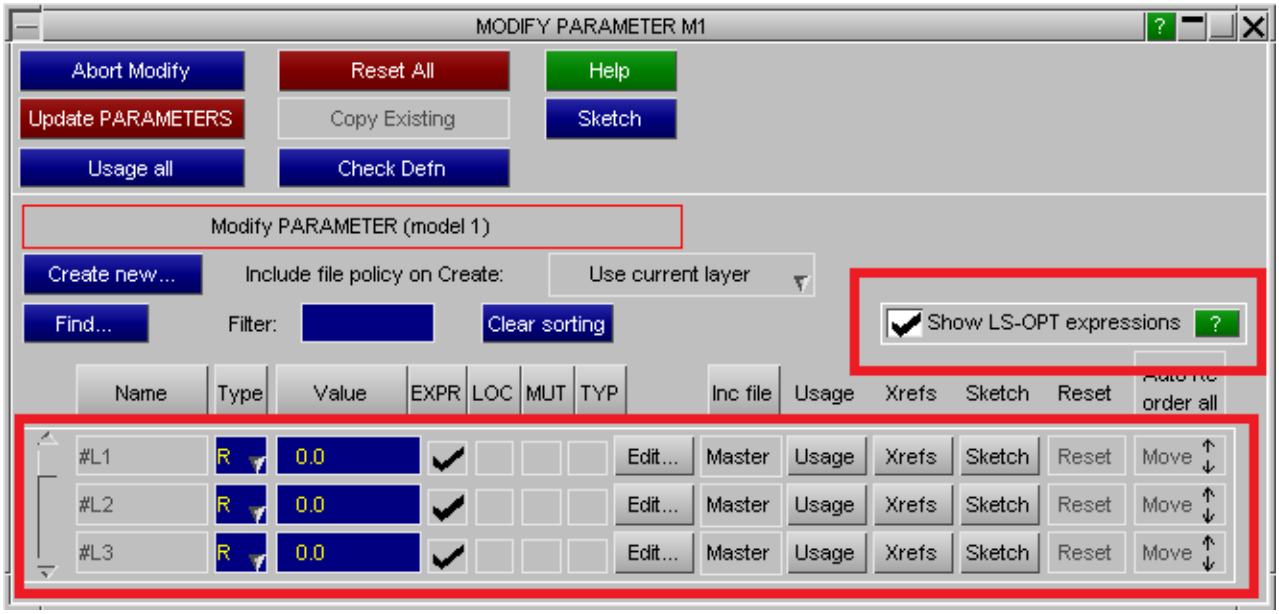
PRIMER does not permit ordinary parameters to be defined by PARAMETER_EXPRESSION, so instead it does the following:

- The <<expression>> is stored verbatim on the parameter structure.
- The parameter is given a constant value of 0.
- Thereafter it behaves like an ordinary parameter.
- You can change the numerical value at will.
- You cannot change the contents of <<expression>>.

During keyword output any value assigned inside PRIMER is ignored and the <<expression>> is written out verbatim in any data fields where it is used.

Visualising LS_OPT expressions

These internal parameters are not normally shown in the top level **PARAMETER** editing panel, and it is necessary to turn on **Show LS_OPT expressions** for them to be visible, as shown below.



Certain attributes of these parameters cannot be changed: they cannot use **`_LOCAL`**, **`_MUTABLE`** or **`_TYPE`** suffices.

Their name cannot be changed, and the syntax **`#Lnnn`** would be illegal in LS-DYNA anyway. It is used to reinforce the point that these are not normal parameters.

You can change the numerical value of these parameters, and such changes will be propagated through the model in the normal way.

> **Support for LS_OPT expressions is new in PRIMER V14 and is relatively undeveloped.**
>
> At present it is really only useful for visualising models that contain these expressions since PRIMER does not "understand" them in any meaningful way.
>
> So it is recommended that they are treated as read-only.

What happens when the value in a parameterised field is changed by some other means.

It is possible for the value in a parameterised field to be changed indirectly by some other operation in PRIMER, breaking the association between the field's true value and the associated parameter value. Consider the following sequence:

1. Parameter **`&X1`** is used as the X coordinate of a node.
2. The user chooses to **ORIENT** the model, translating it in X

After step (2) the the node's X coordinate will no longer match the value of parameter &X1. Other examples of such changes are "auto-fixing" in Check, contact de-penetration, mechanism/dummy positioning, item re-labelling, and there are many more such cases.

In these situations PRIMER will (silently) remove the association between the parameter and its usage in this context, so that if the deck is written out or the node displayed in an editing panel only the plain numeric value will be shown.

The test for whether or not there is a mismatch between data field and parameter value is as follows:

Parameter type	Description of test	Test as an algorithm
`Integer`	The test is absolute: any mis-match breaks the association	if(value != parameter)
`Real`	If the values differ by more than the absolute value of the parameter x 1.0e-6	if (abs(value - parameter) > abs(parameter) * 1.0e-6)
`Character`	The two character strings must match exactly. The test is case sensitive, so "A" will not match "a"; however leading or trailing white space is ignored.	if(strcmp(value, parameter) != 0)

The special case of using Parameters in *INCLUDE_TRANSFORM files

A special situation arises when parameters are used in **`*INCLUDE`** files that are subject to transformation by the **`*INCLUDE_TRANSFORM`** keyword. When dealing with such files there are two requirements:

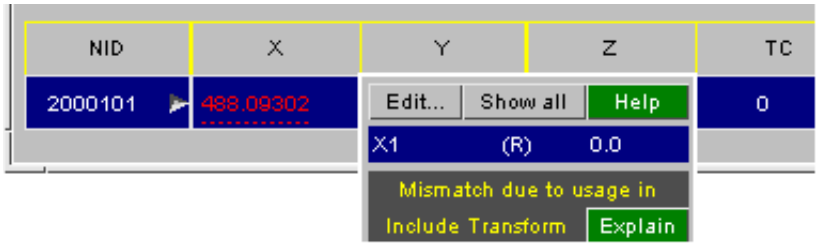
1. PRIMER must apply any transformations when it reads the input deck, so that it can display the model in its "as transformed" state.
2. PRIMER must "undo" the transforms on keyword output so that the original files are written out unchanged.

If parameters are used inside the **`*INCLUDE`** files this can lead to conflicts, since (1) may require the underlying values to change, invalidating the association between parameter and true values; yet (2) requires that the association be "remembered" in order that it can be restored on keyword output.

PRIMER deals with these conflicting requirements by setting a special internal flag against associations occurring inside **`*INCLUDE_TRANSFORM`** files. If the true value and the parameter value do not match in these cases then the following is done:

- The true value is used (as normal) for all calculations and display.
- In editing panels the true value is always displayed, but in red and underlined to designate it as a special parameter.
- On keyword output the inverse transformation is applied to the true value, and the parameter **`&NAME`** written if they still match (to within a small tolerance).

A typical editing panel field looks like this: here parameter **`&X1`** is the node's X coordinate. The parameter popup mapped in response to hovering over the panel shows the parameter's actual value and gives a warning message about the mis-match.



The effect of editing parameterised items used inside ***INCLUDE_TRANSFORM** files is as follows:

- Entering an explicit number in a parameterised field will destroy the association between data field and parameter - as in the normal case above.
- Entering a parameter into a field will result in the as-transformed value of that field changing to the parameter's current value. This may or may not result in the parameter definition appearing in the keyword output deck, as this will depend upon whether or not the pre-output inverse transformation changes its value so that it no longer matches the underlying value.

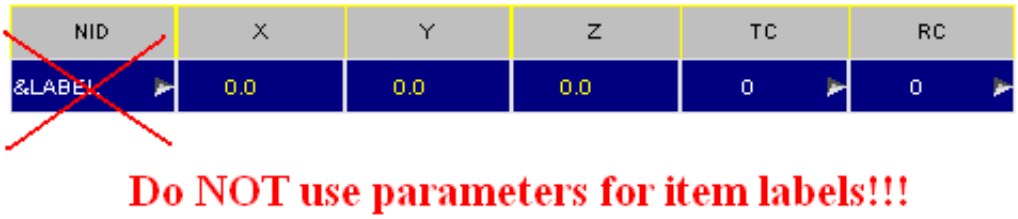
If the value of a parameter definition used inside a **`*INCLUDE_TRANSFORM`**, or a **`*DEFINE_TRANSFORMATION`** to which is refers, is changed then:

- The contents of the **`*INCLUDE_TRANSFORM`** are "untransformed".
- Any data fields using that parameter are changed to the new parameter value.
- The include file contents are "retransformed".

For large include files this may take a significant amount of time, and the "untransform, modify, retransform" cycle may result in small numerical errors in the result. The process works, but it is better if it can be avoided!

This is a complex problem, which only gets worse if the same include file is used multiple times in different **`*INCLUDE_TRANSFORM`** statements to create multiple instances of parameterised data fields! Please contact Oasys Ltd for help and advice if you are having problems with this.

Using parameters for the label fields of items!!!



It is best not to do this at all, or use them only if you won't change their values interactively during a PRIMER session.
It is ******STRONGLY***** recommended that you do ***not*** use parameters to define the labels of items. (ie the labels of nodes, elements, parts, etc).

PRIMER will try to cope if you do, but changing the value of such a parameter could lead to all sorts of conflicts and problems. For example if you accidentally change the parameter value so that you generate duplicate labels the internal "rebuild" of the affected items may fail because of errors, and this may lead to items disappearing - or indeed the whole model being deleted to avoid internal inconsistencies.

PRIMER will read in and process decks that use parameters in this way, and all will be well so long as their values are not changed, but you should regard this as extremely dangerous modelling practice which will, if treated as anything other than "read only", cause severe problems.

If you choose to use PARAMETER_EXPRESSION to build up and define labels remember that the arithmetic rules used to calculate expression values will truncate integer expressions to an integer result. This may have unexpected consequences if your expression includes integer division, and you should read the section "Integer vs Floating evaluation of expressions" above very carefully.

Using parameters in label field before the parameter values are known.

PRIMER's normal behaviour when it encounters a parameter in a keyword deck that has yet to be defined is:
- Substitute a value of 0 (integer) or 0.0 (floating point)
- Add the parameter name to the internal list of latent items, with its type being set to "unknown"
- Continue the input operation using this value of zero.

Once input is complete, and the parameter definition has now been read, PRIMER revisits all locations where items used the parameter before its value was known and recreates them with the correct value.

Generally this works well, but if a parameter is used to define the label of an item then substituting zero does not work because this would be illegal, and PRIMER will reject the definition with a BAD LABEL error.

Prior to release 11.0 this would result in some or all of the input deck being rejected, and possibly the model deleted from memory. This behaviour was criticised by users who wished to view include files or other subsets of models which might use parameters defined elsewhere.

From release 11.0 onwards this behaviour has been modified as follows.
- When an unknown parameter is encountered in a label field PRIMER now substitutes an internal, unique "unknown label" value.
- This value is -ve and is accepted by PRIMER as a "place-holder" value during keyword input.
- The parameter is marked as latent and unknown as before, but it is now given this palce-holder value.

Once input is complete:
- If the parameter has since been read the items that reference it are recreated using the correct labels instead of the place-holder value.
- If the parameter is still undefined the place-holder values are left unchanged.

In the latter case this means that PRIMER must manage the problem some item labels are unknown, and it does this as follows:
- In contexts where the label would normally be shown, for example in object menus or when labelling items on the screen, the parameter name (with &) is used instead.

  Therefore a part label might appear in an object menu as **`M1/P&abc`**

- Inspection of the parameter (using **PARAMETER, MODIFY**) will reveal that it is latent with a -ve value. This is its place-holder value, in a designated internal -ve label range set aside for this purpose, and you should only

change this if it is to become a "correct" +ve value.

Working with unknown parameterised labels imposes some limitations:

- Keywords such as *INCLUDE_TRANSFORM where label offsets are used will almost certainly not work. Adding an offset to an internal -ve number will probably create a value that is illegal, or which clashes with some other labels.

- Renumbering labels may not work, for much the same reason. If -ve place-holder labels are changed to lie outside the permitted range set aside within PRIMER the consequences are undefined.

  ... and it is likely that many other operations that imply changing labels or relying (directly or implicitly) on label ranges also will not work.

Therefore the use of undefined parameters in label fields should be regarded as a "read only" facility for inspecting input decks, and you should not expect everything to work correctly.

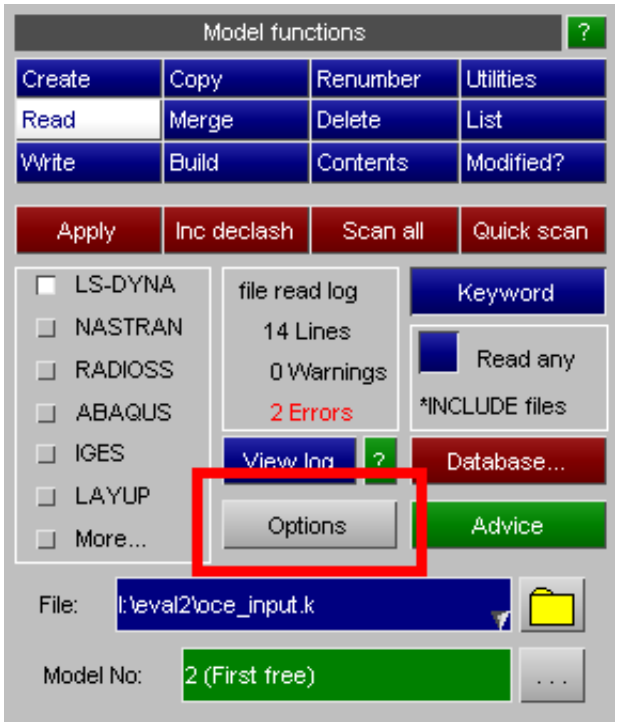
Pre-reading parameters.

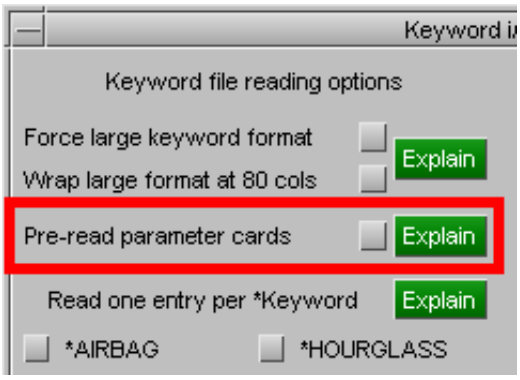
It is sometimes the case that PRIMER's "one pass" approach to reading decks with parameters may fail if the parameters are defined after they are used. This can happen if a parameterised field on a card determines the format of the rest of the card. Consider the case of *MAT_FABRIC, material 34, whose card format is

Card 1	**MID**	**RO**	**EA**	**EB**	**EC**	**PRBA**	**PRCA**	**PRCB**
Card 2	**GAB**	**GBC**	**GCA**	**CSE**	**EL**	**PRL**	**LRATIO**	**DAMP**
Card 3	**AOPT**	**FLC/X2**	**FAC/X3**	**ELA**	**LNRC**	**FORM**	**FVOPT**	**TSRFAC**
Card 4				**A1**	**A2**	**A3**	**X0**	**X1**
Card 5	**V1**	**V2**	**V3**	**D1**	**D2**	**D3**	**BETA**	**ISRFEG**
Card 6	**LCA**	**LCB**	**LCAB**	**LCUA**	**LCUB**	**LCUAB**		

Card 6 is only written if **FORM** on card 3 = 4.

Therefore if **FORM** is defined by a parameter, and the material card is encountered before the parameter value is known, then PRIMER's standard approach of inserting zero and then rebuilding the card later once the parameter's true value is known will not work, because card 6 will not have beeen read. The only solution to this problem is to scan the input deck for parameter values before reading it "properly", and this can be done via **Options** on the **Model Read** panel.





Pre-reading parameters in this way will take a bit longer, but this pre-read is much faster than the "proper" read which follows - and it is infinitely preferable to having the deck read wrongly or failing to read altogether. For this reason it is recommended that you locate parameters at the top of the master file, or in an include file that is read before other include files.

The order in which parameters are defined.

PRIMER 9.3RC2 required that parameters should be defined before they were used, since this was a requirement of contemporary LS-DYNA versions.

PRIMER 9.3 onwards permits parameters to be defined anywhere, including after where they are used, since this is now supported in LS-DYNA 971R3. However ***this practice is deprecated*** because:

- On keyword input PRIMER has to assign a default value of zero to undefined parameters, and then have a final pass to repopulate the internal definitions once the parameters' true values are known. This can slow down input, and if parameters are used inside **`*DEFINE_TRANSFORMATION`** definitions an "untransform, modify, retransform" cycle may be required which can lead to a build-up of small numerical errors.

- It is possible, albeit unlikely, that an input deck may fail to read into PRIMER if the temporary substitution of zero in a parameterised data field results in an invalid card definition. This is most likely to be the case with integer values, but there are a few cards, especially among the materials, where the detailed format depends upon floating point data field values.

- Versions of LS-DYNA prior to 971R3 may not read input decks in which parameters are used before they are defined. Documentation of earlier releases suggested this would be the case, and it is not clear exactly when this limitation was removed.

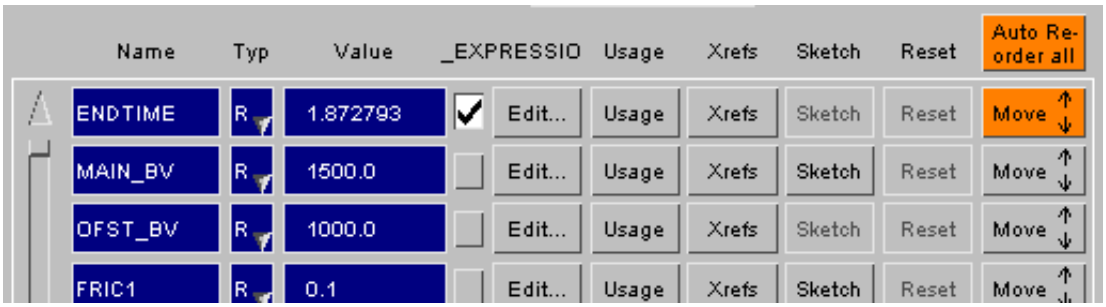
- All known versions of LS-DYNA require that parameters referred to in other **`*PARAMETER_EXPRESSION`** cards are defined before they are used.

- Other 3rd party software may also have difficulties reading keyword decks in which parameters are used before they are defined.

PRIMER 12 onwards adds a Pre-read Parameters option in the Model, Read, Options panel in section 3.2.1 to pre-read parameters in decks where the order in which they are defined causes problems.

Re-ordering parameter definitions inside PRIMER

Once inside PRIMER the order in which parameters are stored is irrelevant, since the code can access internal data in any order. However it contains logic to re-order parameters to deal with the problem of a **`*PARAMETER_EXPRESSION`** referring to another parameter that is defined after rather than before it.



In this example **`ENDTIME`** is a parameter expression which refers to other parameters, and it has been moved to become the first entry in the list so that it is "out of order", resulting in its "**Move**" button being coloured orange as shown here. This can be resolved as follows:

Auto Re-order all will attempt to resort the parameter list automatically to resolve any ordering conflicts. It moves all plain (non **`_EXPRESSION`**) parameters to the top of the list, and then tries to re-order any remaining **`_EXPRESSION`** definitions so that they only refer to their predecessors.

Move provides a manual alternative to the above. Click on this button, and then use the keyboard up or down arrow keys to move this row up or down. Once the ordering conflict has been resolved the button will go green.

As explained above it is not strictly necessary to re-order parameters since both PRIMER 9.3 onwards and LS-DYNA 971R3 onwards can cope with them in any order. The re-ordering logic described here was written to deal with the earlier situation when this was not the case, but its use is still recommended for the reasons given above.

What happens to Parameters during a Units change operation

If a units change is applied to scale parameterised data fields, but the parameters themselves do not also have their values changed, then the association between the parameters and the data fields will be lost. PRIMER tries to handle this problem by inferring scale factors for parameters and hence scaling their values; this process is described in section 6.6.2 How Units change affects Parameters.

Parameters used in titles and other text strings.

PRIMER does not attempt to evaluate parameters used in titles or other text strings, simply leaving them as "&*name*". It also does not create a cross-reference between such usage and the parameter, meaning that they do not "know about one another" and a parameter used only in text strings will not be locked against deletion.

LS-DYNA handles "&*name*" in titles and text strings as follows:

- Release 971R4 (mid 2009) onwards will replace valid definitions with their numeric values. If "&*name*" is not a valid parameter then it will issue an error message and exit.

- Releases prior to this had inconsistent behaviour: some simply ignored "&*name*", some attempted to substitute numeric values with varying degrees of success, some generated errors if "&*name*" was not a valid parameter definition and some did not seem to care.

Therefore from release 9.4 onwards PRIMER deals with "&*name*" in titles and text strings as follows:

- During normal interactive usage (editing, copying, etc) these definitions are simply reproduced verbatim and are treated as ordinary text strings with no special significance.

- During keyword output ***only*** for "&*name*" in a title or text string behaviour depends upon the

`primer*parameter_in_string` preference:

This preference can have one of four values:

<table>
<tr><th>Preference value</th><th>Effect during output</th></tr>
<tr><td><code>AUTOMATIC</code>
(default)</td><td>Processing on keyout depends on the LS-DYNA output version chosen:
<table>
<tr><th>LS-DYNA version selected for output</th><th>"&name" is a valid parameter name</th><th>"&name" does not match any parameter</th></tr>
<tr><td>971R4 and later</td><td>The string is output unchanged, ie as "&name"</td><td rowspan="2">A "@" character replaces "&" in all cases.</td></tr>
<tr><td>Pre 971R4</td><td>The numeric value of the parameter is substituted into the text string, replacing "&name" with the appropriate floating point or integer number.</td></tr>
</table></td></tr>
<tr><td><code>REPLACE_AMPERSAND</code></td><td>All "&" symbols in text and title strings are replaced unconditionally with "@", regardless of the LS-DYNA version chosen for output.</td></tr>
<tr><td><code>INSERT_VALUE</code></td><td>Where "&name" is a valid parameter its numeric value is inserted into the text string, regardless of the LS-DYNA version chosen for output. If it is invalid "@" is substituted for "&".</td></tr>
<tr><td><code>VERBATIM</code></td><td>The text string is unchanged, with no substitutions of any sort being made.</td></tr>
</table>

The default **`AUTOMATIC`** output policy may not suit all possible circumstances, but it has the merit that it should prevent initialisation in LS-DYNA failing due to malformed text strings in titles.

It is recommended that you do not use the "&" character in titles and text strings unless you are certain that it prefixes a valid parameter name, and even then it may cause problems in all but the most recent versions of LS-DYNA.

Allowing Parameters to use 8 or 9 character names

The LS-DYNA 971R4 user manual states explicitly that parameter names are limited to 7 characters; however experimentation by users has revealed that LS-DYNA will in fact accept and process 8 character parameter names, and there is some evidence to suggest that the 7 character limit stipulated in the user manual is a misprint.

The LS971R6 manual explicitly states that parameter names can be up to 9 characters long.

Since PRIMER needs to be able to read and process input decks that run in LS_DYNA it has been modified (in version 10.0) to increase the maximum length of parameter names from 7 to 8 characters., and release 11.0 of PRIMER will accept parameters 9 characters long.

A model "check" will issue a warning that this undocumented capability is being used if any 8 or 9 character parameter names are found, but this warning can be suppressed by setting the preference

`primer*warn_param_8_chars: off`

This check is not performed if the output file format is 971R6 or later, in which case parameters up to 9 characters long will be accepted silently.

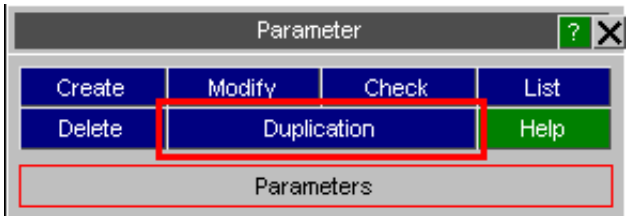
Permitted length of parameter names in "large" format.

In "large" keyword format all data fields are 20 columns wide meaning that there is sufficient space for **`&parameter`** syntax to use 19 characters. The LS-DYNA manual currently states that parameters are limited to 9 characters, making no mention of any extension in large format, so at the time of writing (Oct 2013) it is not known whether or not this is legal in LS-DYNA.

From V12 PRIMER is set up to use 19 character parameter names internally and will permit names this long to be defined, but the parameter checking functions will issue a warning if names >9 characters long are used.

This subject is under review and when the behaviour of LS-DYNA has been clarified the logic and checking functions in PRIMER will be amended to suit.

The *PARAMETER_DUPLICATION keyword



LS-DYNA 971R5 introduced the concept of "local" parameters, and these are described above. This creates the possibility of errors when both global and **_LOCAL** parameters are accidentally defined in the same include file, and the behaviour of LS-DYNA on input is determined by the settings on this card.

The LS971 R7.1 manual (October 2013) states explicitly that this is a "once only" keyword, and that if multiple definitions are found the first will be used and the remainder ignored, with a warning being issued. Earlier LS-DYNA manuals did not make this clear, so PRIMER was programmed to assume that this card could appear multiple times in both master file and any include file.

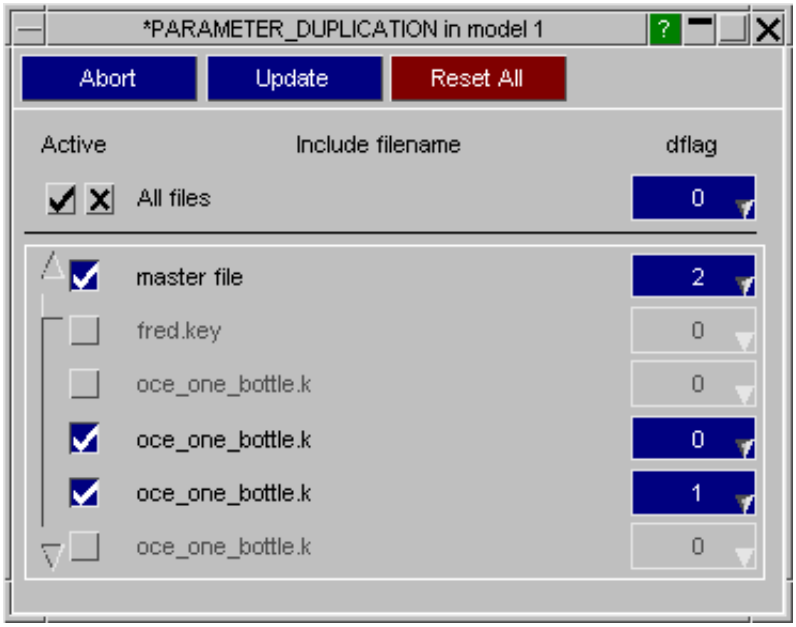
Since PRIMER must preserve input decks the ability to define and modify a ***PARAMETER_DUPLICATION** card in every include file has been retained, but from V12.0 onwards the following checks have been added:

- Multiple definitions in which all **DFLAG** values are the same just generate a warning.
- Multiple definitions in which one or more **DFLAG** values are different is treated as an error.

PRIMER permits each include file (including the master file) to have a separate, possibly different, definition of this keyword. It also permits these definitions to be created, edited and deleted using the **Duplication** option.

As this example shows you choose whether to have a ***PARAMETER_DUPLICATION** card in each include file by using the Active tick box.

In files where it is present you can set the **DFLAG** value accordingly.



*PARAMETER_DUPLICATION and error checking in PRIMER

From V11 onwards PRIMER considers the settings on any ***PARAMETER_DUPLICATION** card when it encounters clashing parameter names, taking the following actions:

If a *PARAMETER_DUPLICATION card is present:

- If it is set to "accept" (options 2 or 4) then the ***youngest***, of any clashing definitions in a given scope will be used to resolve references to parameters of that name. For example, if a keyword referencing that parameter is defined between the first and the second definition of the parameter, it will use the first definition, whereas all keyword cards referencing that parameter name after the second definition will ignore the first definition if that parameter name. This is the same behaviour as for **_MUTABLE** parameters described above, although the **_MUTABLE** suffix will not need to be specified with this duplication setting.

- If it is set to "ignore but continue" (options 0, 1 or 5) then the ***oldest***, as in first read, of any clashing definitions in a given scope will be used to resolve references to parameters of that name. Any younger definitions in that scope will be stored and written out, but marked as ignored.

- If it is set to "ignore and terminate with error" (option 3) the oldest definition is used as above. Younger clashing definitions are stored and will be written out, but marked as "duplication error" and will also not be used.

If a `*PARAMETER_DUPLICATION` card is *not* present:

- The effect is the same as option 3 above: the oldest definition is used, and younger clashing definitions are remembered and written out, but marked as being in error and to be ignored.

Error checking will warn about any clashing parameters, treating clashes that would be resolved successfully by a **`*PARAMETER_DUPLICATION`** during LS-DYNA keyword input as warnings, and those which would result in error terminations as errors.

(Prior to V11 PRIMER did not consider the settings on this card when checking for errors in parameters. It treated duplication of parameter names within a file as an unconditional error regardless of any settings here, or indeed whether this card is present or not.)

*PARAMETER_DUPLICATION and *PARAMETER_MUTABLE

LS971 R7.1 introduces the **`_MUTABLE`** suffix for the **`*PARAMETER`** keyword, providing an alternative way of handling multiple parameter definitions for the same name. If the first definition of parameter ***name*** is **`_MUTABLE`** then any number of further definitions of the same name can be read, with each one superseding the value of ***name*** so that its current value is that of the most recent definition found.

This behaviour is silent, and happens for **`_MUTABLE`** parameters regardless of the current **`*PARAMETER_DUPLICATION`** setting. The same behaviour also happens for the duplication settings 2 and 4 without explicit **`_MUTABLE`** suffix.

Character parameter types

From LS-DYNA 971 R6 onwards the character ("C") parameter type has been introduced, and this is supported from PRIMER release 11 onwards. The LS-DYNA user manual does not state how these parameters are expected to be used, so PRIMER makes the following assumptions:

- Character parameters can be used in item label and data fields where character strings would be acceptable. For example the labels of Material, Section, Equation of state, Hourglass and Thermal material which can all (optionally) use character labels.

- Character parameters will ***not*** be used "long" text fields such as title and header strings.

- Character and numeric (I or R) parameters are not interchangeable. PRIMER will only permit a parameter to be changed between these types if it is not referenced anywhere in the model.

These assumptions may or may not turn out to be right, if not the code will be modified in future releases.

PART: Defining Parts.

Top level menu
- Creating
- Copying
- Editing
- Deleting
- **Rigid Parts**
- **Visualisation**

Options:
- _INERTIA
- _CONTACT
- _REPOSITION
- _PRINT
- _ATT.._NODES

Sub-keywords:
- _ADA.._FAIL..
- _COMPOSITE
- _MODES
- _SENSOR
- _MOVE

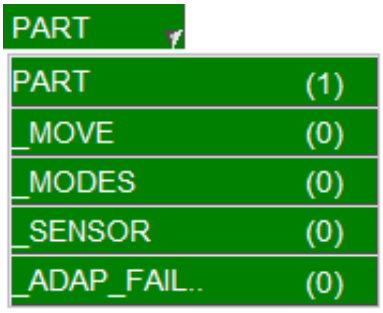
Moving elems in/out

Calculating properties

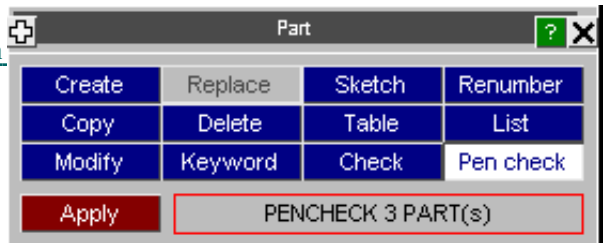
The `*PART` keyword is central to LS-DYNA usage: it acts as a "collector" of material, section property and other data for groups of elements. It is also commonly used as a means of splitting up large models into manageable components which can be given part names.

Parts in LS-DYNA are only used by **`BEAM, DISCRETE, SEATBELT, SHELL, SOLID`** and **`TSHELL`** element types, and elements of these types must reference exactly one part. Parts do not have an element type defined explicitly but, since LS-DYNA only permits a part to be referenced by a single element type, there is an implicit type associated with them. PRIMER honours this by associating an internal "type" with each part, which is defined by the first encountered element that references the part, and only material, section and other definitions valid for that element type are permitted.

`*PART_(OPTION1)_(OPTION2)_(OPTION3)_(OPTION4)` cards and **`*PART_COMPOSITE`** cards are defined by choosing **PART** from the drop-down list in the Keywords panel in Primer. The other **`*PART`** types (_MOVE, _MODES, _SENSOR and _ADAPTIVE_FAILURE) are defined from their own individual selection button from the part dropdown list.



This figure shows the main **PART** control panel.
All options have their standard meanings as described in section 5.1.1.



CREATE Creating a new part

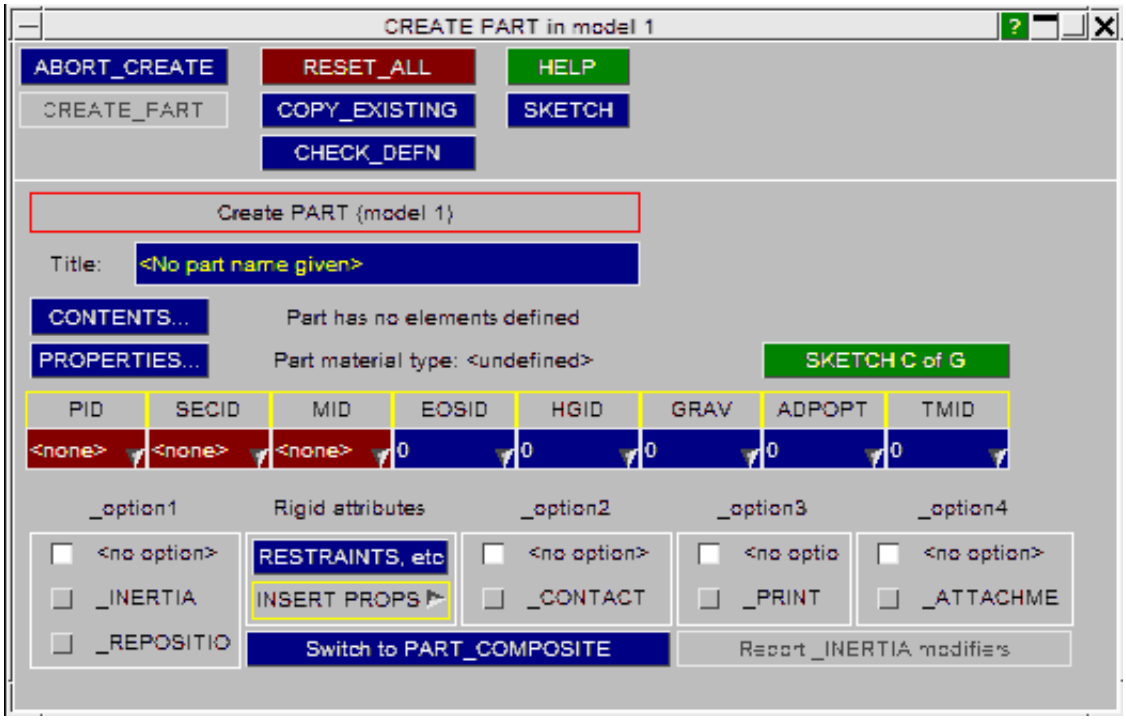
This figure shows the initial creation panel, in which nothing is defined.

In order to create the part you must define the following:

PID	The part id (label), which must be unique.
SECID	The section id. The section type must match the type of the elements in this part.
MID	Material id. Again, this must be a formulation valid for the elements in this part

This represents the bare minimum required to create a part, and you can now use **CREATE_PART** to make the definition permanent.

The remaining data fields (hourglass id, etc) are optional: limited on-line help is provided in their popup menus, but you should refer to an LS-DYNA manual for details of their meaning.



CONTENTS... The elements which make up the part. This is described below under section PART_CONTENTS.

PROPERTIES... Calculates and lists the structural properties (mass, C of G, inertia tensor) of this part - see PART_PROPERTIES

The optional sub-keywords (**`_INERTIA, _REPOSITION, _CONTACT, _COMPOSITE, _PRINT`**) are described in sections **PART_INERTIA** to **PART_PRINT** below.

COPY Copying selected parts.

The selected parts will be copied to the next free labels in the relevant models.

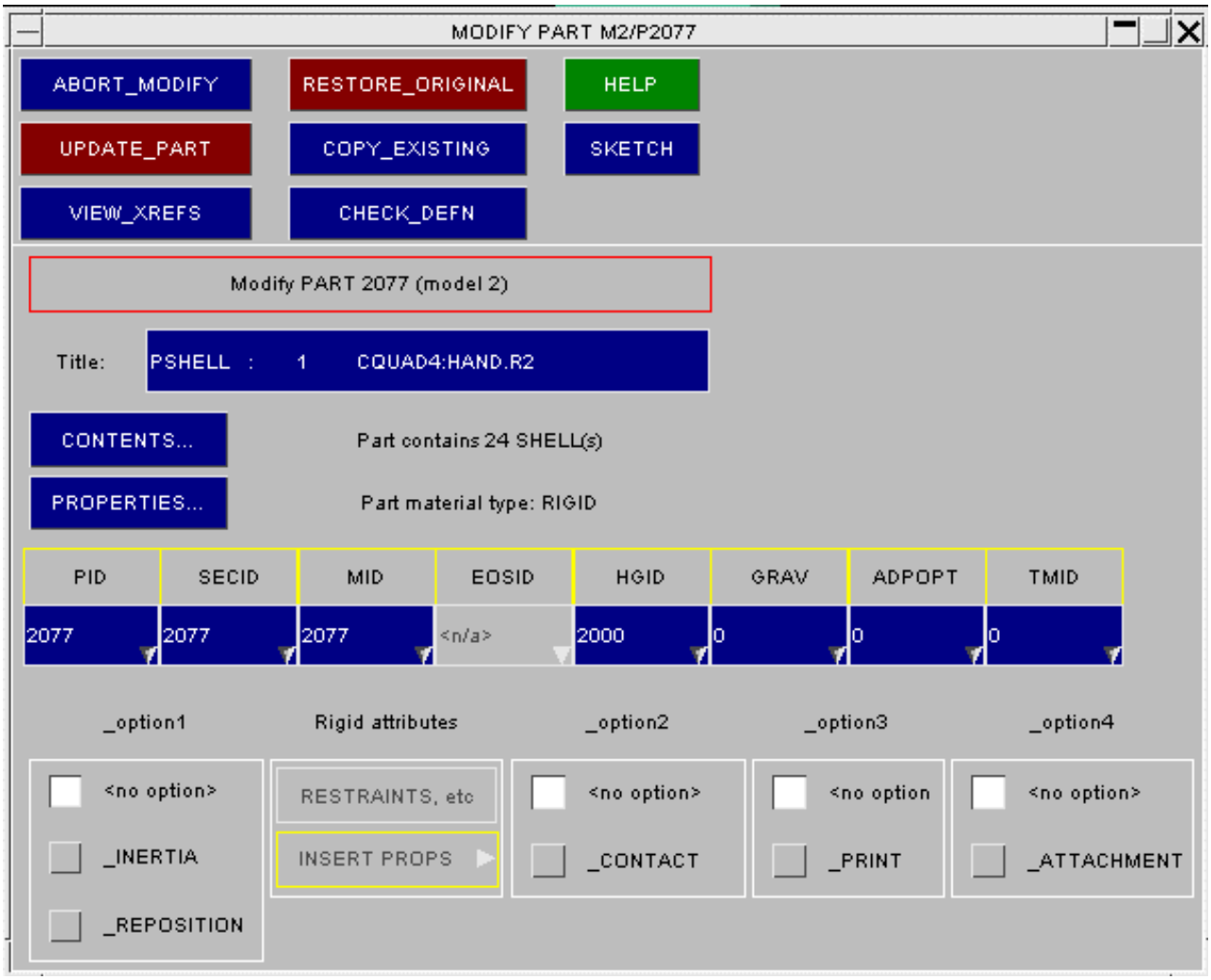
RECURSIVE_COPY is an extremely important flag in this context:

- When **OFF** Only the actual **`*PART`** card and the data on it are copied. The new definition contains no elements, but has the same properties as the original part.
- When **ON** Both the **`*PART`** card, and everything "owned by" (ie lower in the hierarchy than) the part are copied. Thus the elements and nodes will be duplicated, and the new part will refer to a set of new elements which are identical to but separate from the originals.

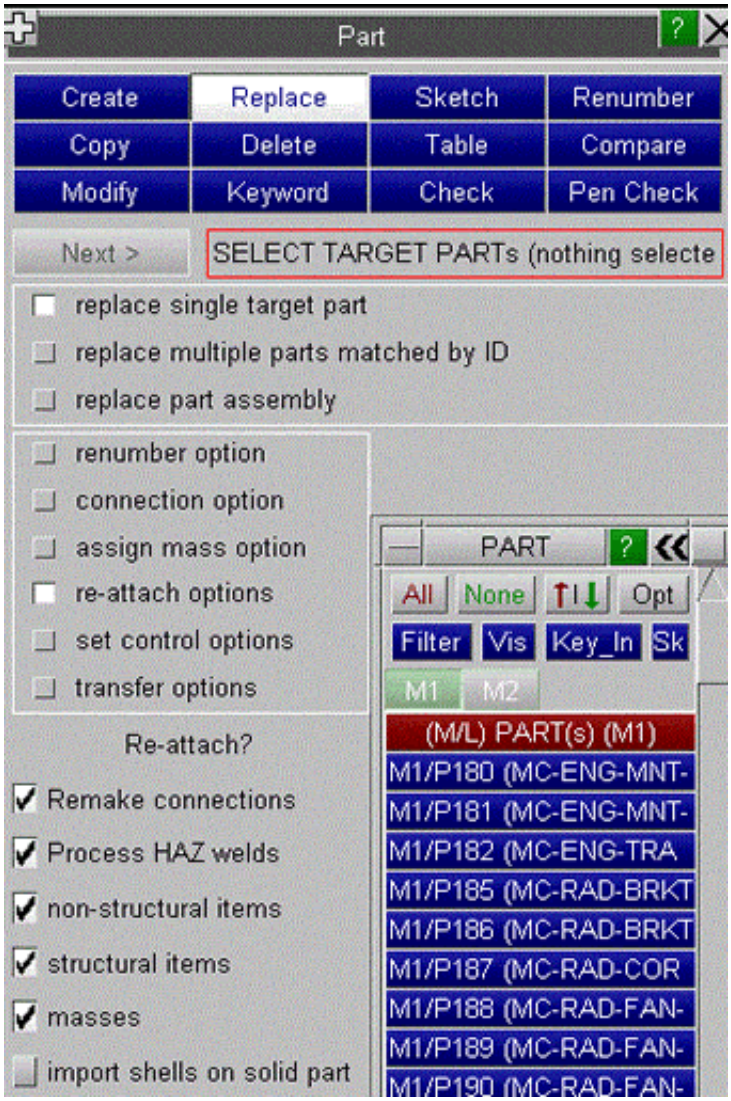
Note that the section and material definitions are ***not*** copied, since they are above parts in the programme hierarchy.

MODIFY Editing existing parts

The part editing panel is identical to the creation one, except that it will be populated with data when mapped. This figure shows a typical example, here for a part containing 24 **`SHELL`** elements. Changes are made as for **CREATE** above.


MODIFY PART M2/P2077
ABORT_MODIFY
RESTORE_ORIGINAL
HELP
UPDATE_PART
COPY_EXISTING
SKETCH
VIEW_XREFS
CHECK_DEFN
Modify PART 2077 (model 2)
Title:
PSHELL : 1 CQUAD4:HAND.R2
CONTENTS...
Part contains 24 SHELL(s)
PROPERTIES...
Part material type: RIGID
PID
SECID
MID
EOSID
HGID
GRAV
ADPOPT
TMID
2077
2077
2077
<n/a>
2000
0
0
0
_option1
Rigid attributes
_option2
_option3
_option4
<no option>
_INERTIA
_REPOSITION
RESTRAINTS, etc
INSERT PROPS
<no option>
_CONTACT
<no option
_PRINT
<no option>
_ATTACHMENT

REPLACE Replacing part(s) with those from another model



The process of replacing part(s)

The **REPLACE** function offers a powerful method of replacing a part or multiple parts in the target model with part(s) taken from **another** model (the source), typically a similar part remeshed or reshaped.

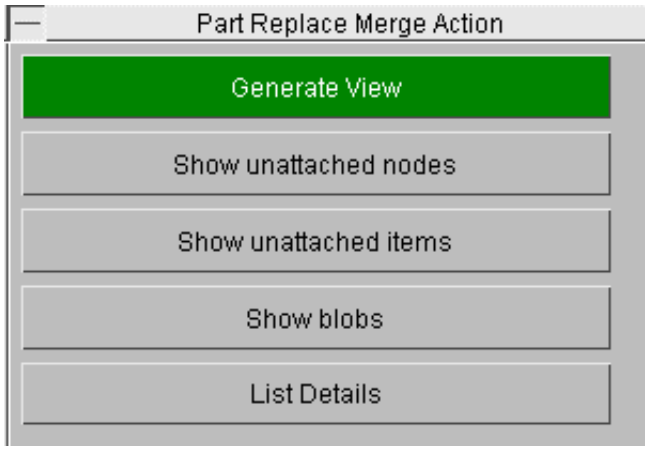
If replacing a single part, there is no need for the Part id to match.

If replacing multiple parts the target and source part ids must be the same. The element type (if specified) of the target and source parts must be the same. For multiple parts, all parts selected must have the same element type.

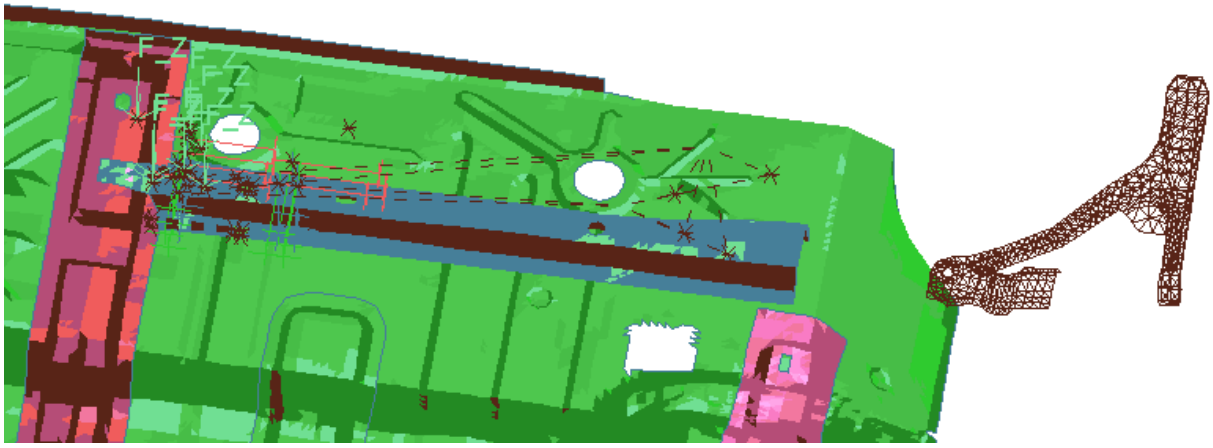
For single part, the replacement is achieved by simply selecting the target part and then the source part from the object menu.

For multiple parts, the target model is selected and then the parts from the source model which are to be imported. If the id of one or more selected source parts does not match any part is in the target model, the user is given the option of ignoring such parts or creating new ones in the target model. There is no longer a restriction on mixing parts of different element types, so an assembly of shells and solids can be replaced in one operation.

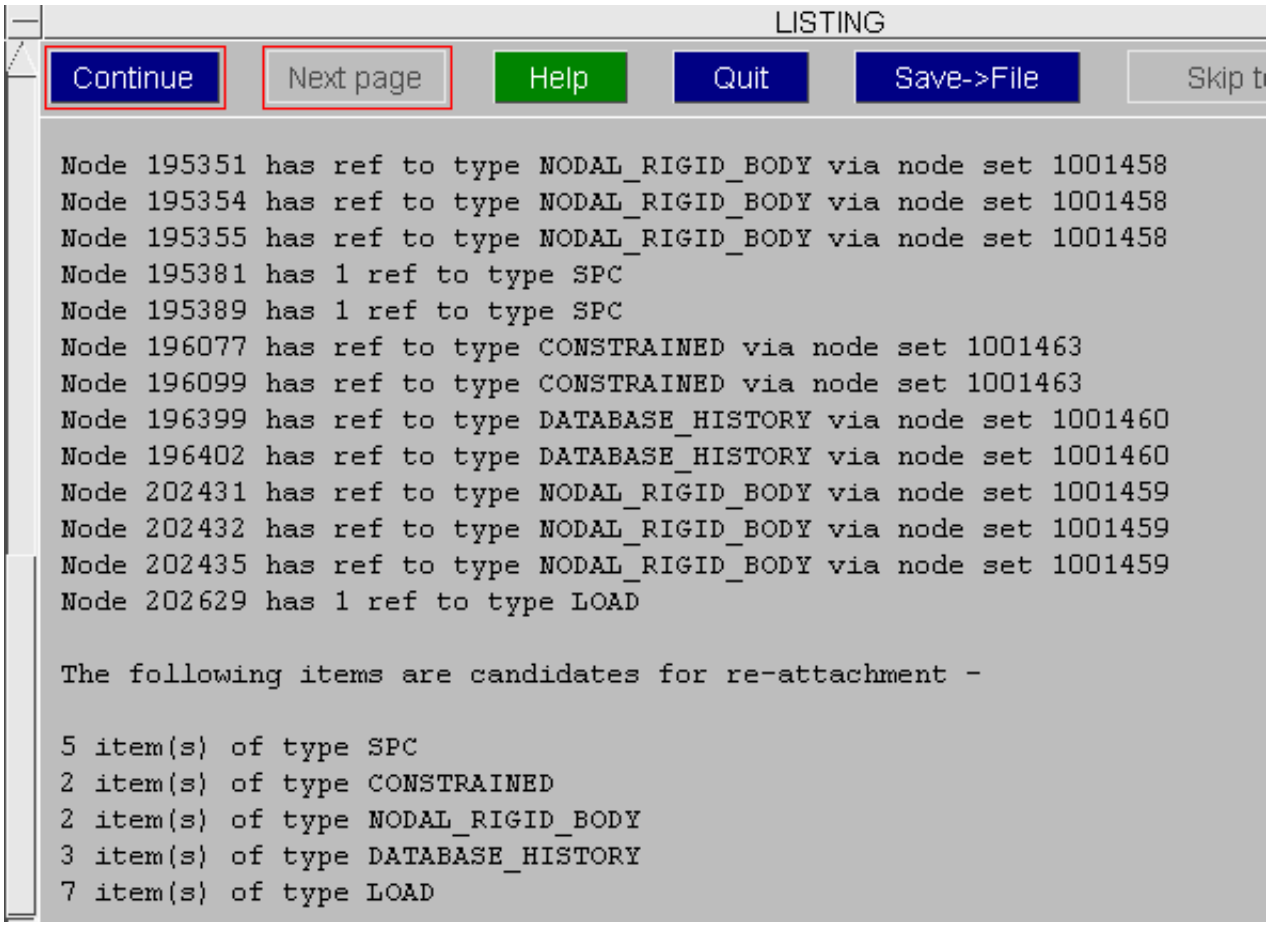
If any of the **re-attach** option are active on completion of transfer of elements & nodes into the target model, Primer will offer you a panel which allows you to view details of the possible merge options available and options on how to proceed.



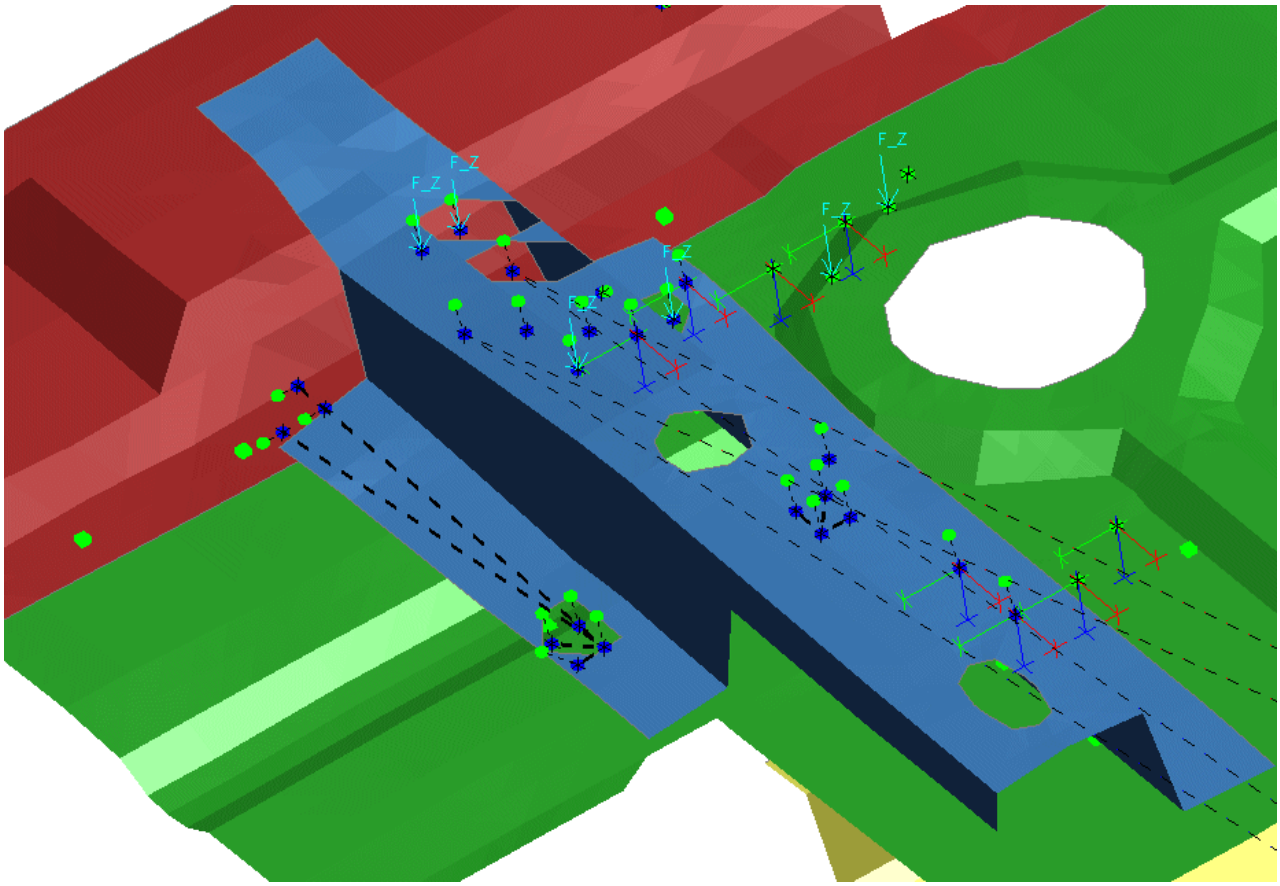
Show unattached nodes and **Show unattached items** will sketch the items concerned.



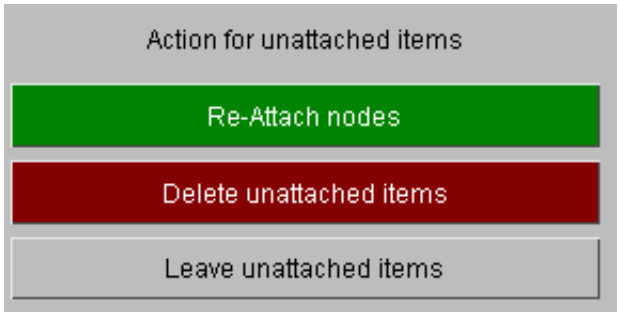
List Details will provide a detailed listing of what may require re-attachment.



Generate View will apply blanking automatically and give a blob plot to show how the nodes will be re-attached. The old nodes are plotted in blue and the corresponding new node in green. Any nodes which cannot be re-attached will be plotted in red.

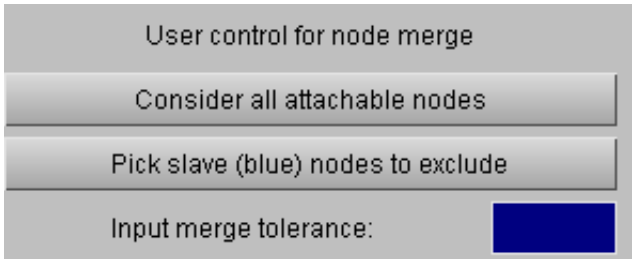


The user now has the option to **Re-Attach** the sketch items by merging the blue node to the green node, **Leave unattached** (for attention later) or **Delete unattached** items.

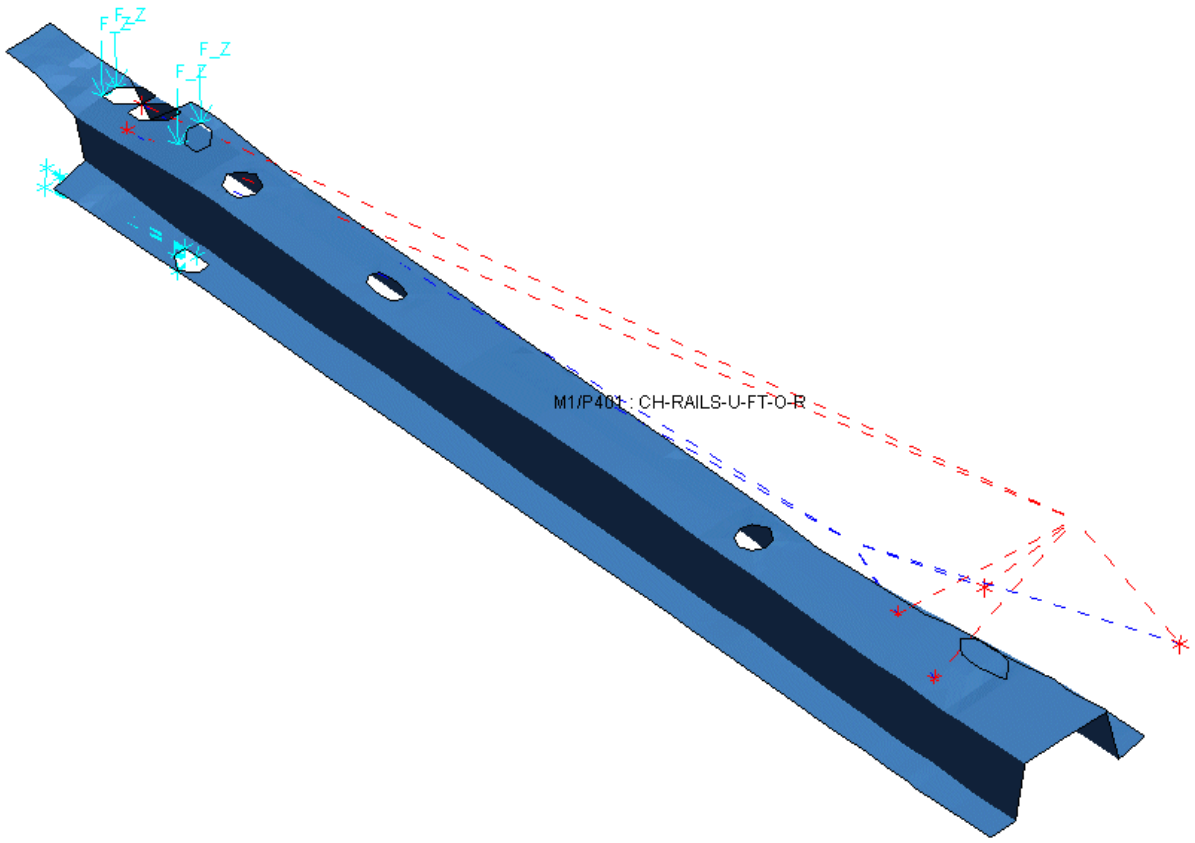


Delete unattached items will bring up the interactive deletion panel, giving the user control over what is deleted.

The user may control exactly which nodes are to be merged by applying a merge tolerance or picking slave nodes (blue blobs) to be excluded from merge.



If **Re-Attach nodes** is applied, the user may wish to Only the parts concerned and run a Find Attached operation to check that the process has been done correctly.



Re-attach options for part replace

Remake connections Primer will remake any connections attached to the target parts, once the part replace operation is completed. Any that fail to remake will be displayed on the connection table.

Process HAZ welds The default treatment of HAZ (meshed in) welds is

- undo the weld at the beginning of part replace (reverting any HAZ rings to the native part before deletion of elements)

- remake the weld on the new mesh if **Remake connections** is ON.

This has the disadvantage that shell mesh may be changed on layers of the weld which are NOT the part being replaced. If **Process HAZ welds** is switched OFF, the weld and the HAZ rings (they need to be different part to the native part) will not be deleted. If the imported source mesh contains holes which are configured to mate to the outer edge of the HAZ these can be re-attached using the merge function. There will be no remake of the HAZ welds, so there is no disturbance to mesh on other layers.

Non-structural items The nodal merge action will be offered for items such as *Initial_velocity, *Database_history_node or *Load_node (referenced directly or by conventional node set see below)

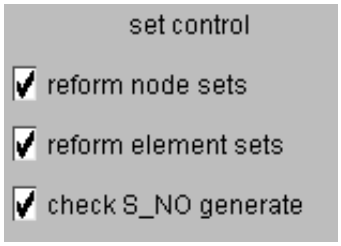
Structural items The nodal merge action will be offered for items such as node attached to structral element, node on

a *Nodal_Rigid_Body or a *Boundary_spc (referenced directly or by conventional node set see below)

Masses Preserve lumped masses (excluding Assign mass) on the nodes of the target part.

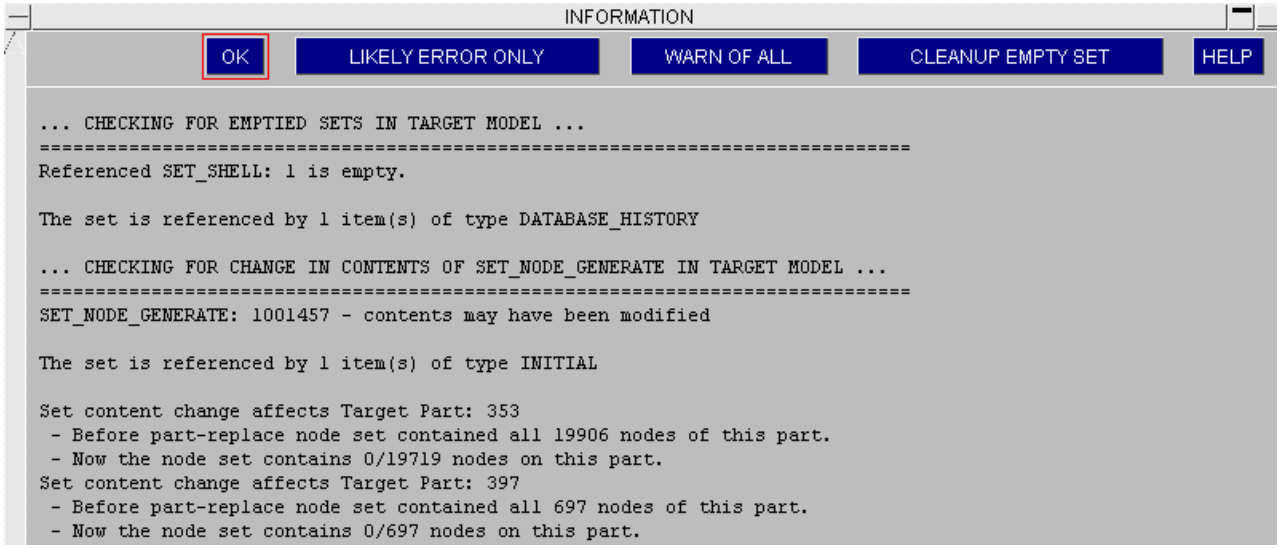
Null shells on solid part This is available for single part replace only. If active, when replacing a solid part coated with null shells, Primer will delete the old null shells and import those on the source part (if present) into the original null shell part in the target model.

Set control options for part replace

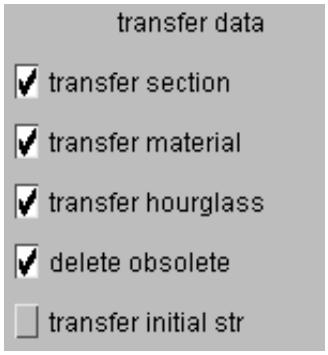


Reform node sets and **Reform element sets** These options apply only to conventional sets. This option ensures that if all nodes of a target part are in a node/element set, all nodes of the new part replacing it will also be in the set. Thus node sets for initial velocity or load can be maintained when the source part is more densely meshed than the target part.

Check S_NO generate As users may be using this type of set to deliberately control the contents Primer will make no attempt to modify the definition. It will however check the contents both before and after the part replace operation and report if there appears to be a discrepancy. Likely modelling errors are if all the nodes of a part were in the set and now some are missing, no nodes of a part were in the set and now some are, or some nodes of a part were in the set and now none are. Primer also warns of empty sets and offers to remove them.



Transfer options for part replace

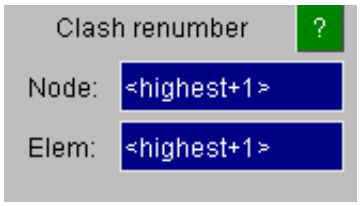


Transfer section / Transfer Material / Transfer hourglass Primer will import the section card, etc from the source model and change the target part to reference it. In the case of the Material card the load curves will be imported also.

Delete obsolete On completion of transfer, Primer will remove the original section, material, etc from the target model if it is unused by anything else.

Transfer Initial Primer will import any initial stress (or strain) cards which refer directly to elements of the source parts. Note - this will not apply if the _SET definition has been used in the source model.

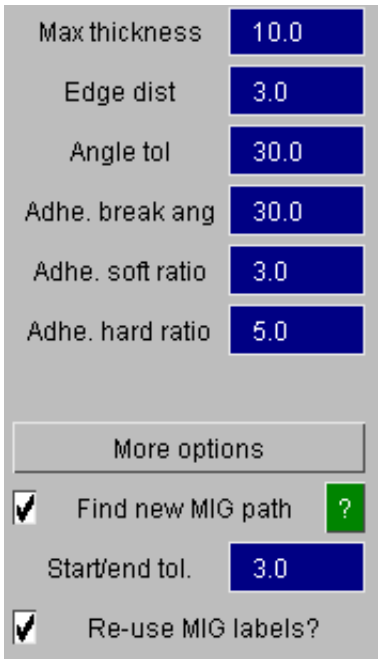
Renumber clash options for part replace



By default if any of the nodes/elements of the source part(s) clash with labels that already exist in the target model Primer will put these to highest label + 1. In models with set_generate, where an upper bound is above the highest label of any item, then the upper bound value will count as the highest label.

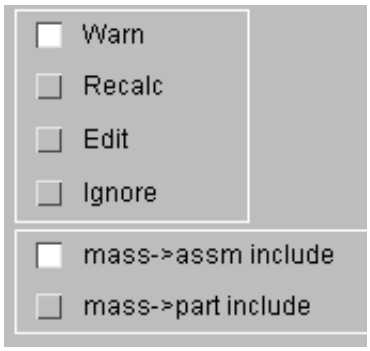
You may instead set a seed label. Note that if include label ranges are set on completion of part replace the generic relabelling routine will be applied to ant items that are out of their designated include range.

Connection options for part replace



You can set options for remaking welds and adhesive which will apply if the remake connections option is active. Additionally, you can activate/de-activate the option to reform the free edge geometry for MIG type spotwelds.

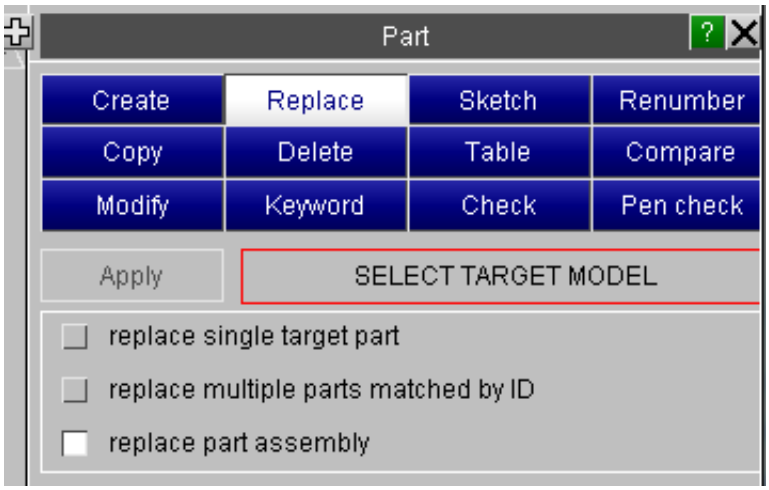
Assign mass options for part replace



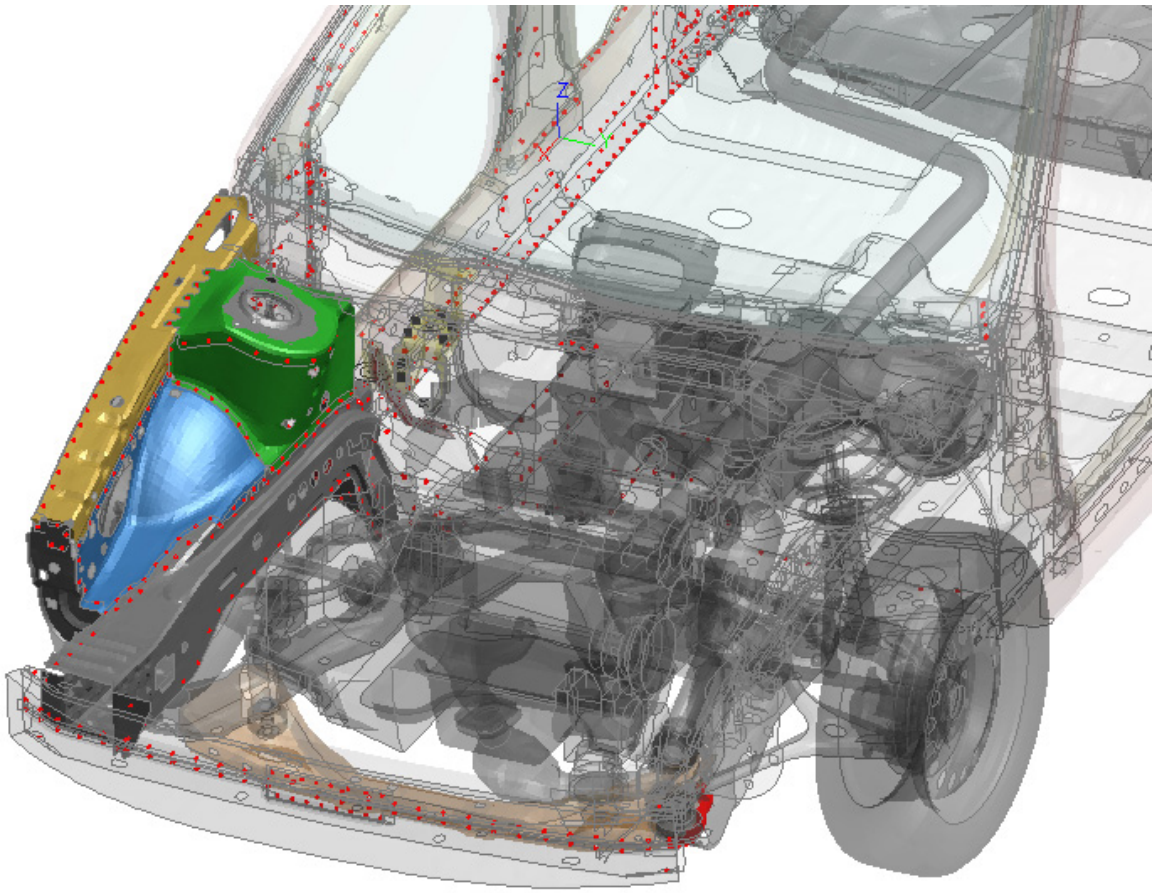
If the target parts are subject to Assign mass statement , Primer will (by default) warn of this and offer the option to recalculate/modify/ignore the assign mass. This option can be pre-configured.

The include for the mases created on the imported part will by default be that of the assign mass definition (as it is if we create an Assign mass), you can, however, switch this to be the same include as the nodes of the part.

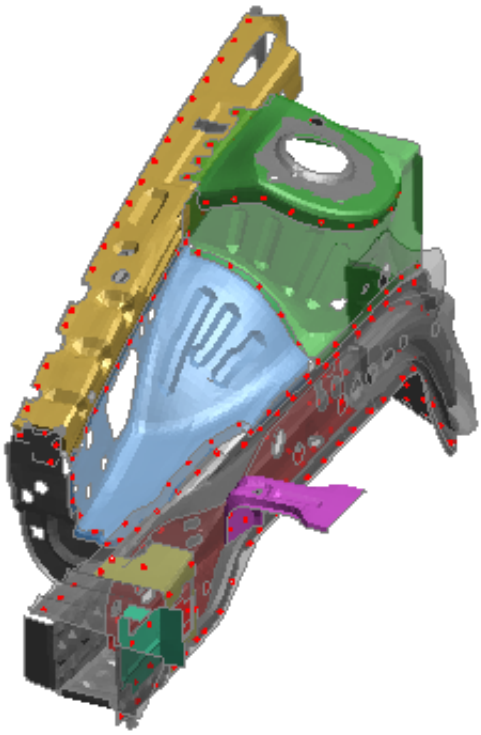
Replacing an assembly of parts using a model as source



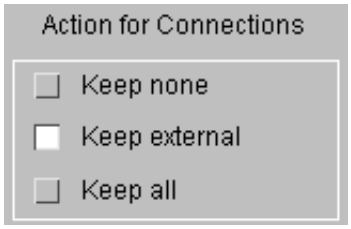
Multiple part replace as described in the previous secttion, serves well when there is a reasonable mapping between the target parts and the source parts which replace them. In such a case, the remake of old connections in the target model will hopefully give the correct connectivity for the new parts. However, consider the case where the new parts have completely different shape to the old, or where two parts replace one and the welds that join them do not exist in the target model. For such cases, we need a more generic tool which removes a set of parts (the target assembly) from the target model and all their internal connections and references, imports a complete source model which contains not only parts but their internal connections and then weaves the imported items into the target model making connections at the external boundaries of the assembly and reforms the relevant external references. It is not easy to achieve this.



The assembly to replace is best described using a Part Tree Assembly (in this case the parts shown with zero transparency). However, the parts may be described by any collector, e.g. part-set, group, include file or may simply be selected off the object menu.

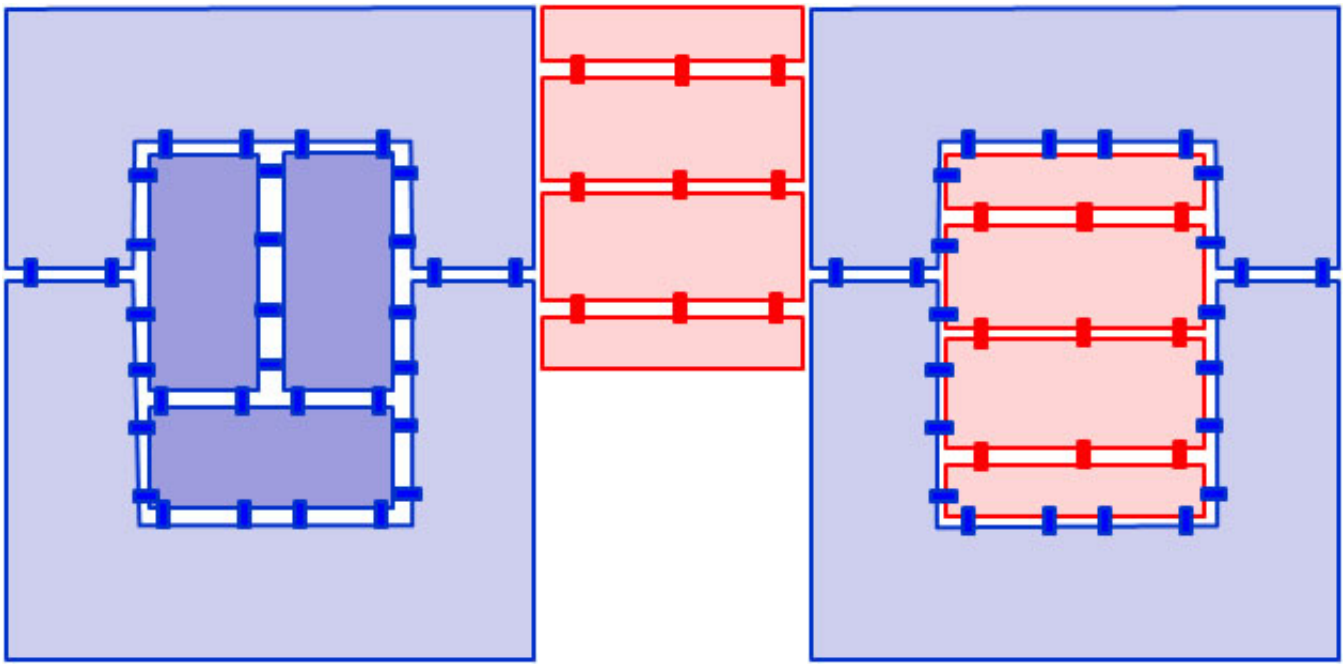


The source (or replacing) assembly is the complete contents of the designated source model. As this contains internal connections, the part geometry may be extensively different from the target model and there is no requirement that the parts map reasonably onto one another. Source parts that match target parts by label will implicitly be put into their correct set. If the source parts are at different labels, Primer may have difficulties putting the un-matched parts into the correct part-sets. This will also apply to any new parts. This problem may be obviated by the use of the preferred PART_SET_GENERATE instead of conventional sets. For a conventional set, the best automatic maintenance we can do is if all parts of the target assembly are in such a set, then all parts imported are put into the set.

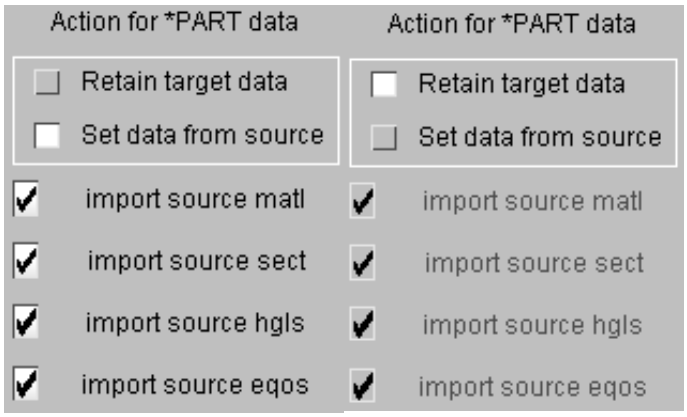


Action for (Primer) Connections - the default is to keep (and remake) only those that connect between the target assembly and external parts. This may be changed to keep none or keep all. The latter would only be appropriate if the source model contained no connections and the part mapping was very close and part labels correlated, i.e. a conventional part-replace operation.

The handling of connectivity is represented schematically in this diagram, where the dark blue target assembly is replaced by the modified pink source assembly.



Action for *PART data - If part labels match across source/target model, the default is **set data from source**, i.e. set the labels for matl, sect, etc on the target part to match that of the source part. If the option **import source matl, sect**, etc is active the matl, sect, etc card from the source model will be imported. Alternately, user may choose to **retain target data** which means reference on target part to matl, sect, etc will be unchanged.



DELETE Deleting parts

The selected parts are deleted. As with **COPY** the choice of flags is very important:

The **DELETE_RECURSIVE** flag determines whether or not the items "owned" by (lower in the programme hierarchy than) the part are handled:

- When **OFF** Only the part definition itself will be deleted, and this will only happen if it is not referenced by anything else, eg elements, that depend on it. If it has any dependants then it will not be deleted.
- When **ON** Its dependants (elements, nodes, ...) will also be marked for deletion. The part itself will still only be deleted if *all* its dependent items are themselves deleted.

The **Remove from sets** flag determines whether or not a part is barred from deletion by virtue of being part of ("owned by") a **`SET_PART`**.

- When **OFF** If the part is in a set it will not be deleted, even if it has no dependants, although its dependants may be deleted.
- When **ON** If the only thing stopping the part being deleted is its membership of a set then it will be removed from the set and deleted.

SKETCH Sketching parts

The selected parts are sketched in white on top of the current graphics image. Parts are drawn by drawing their constituent elements

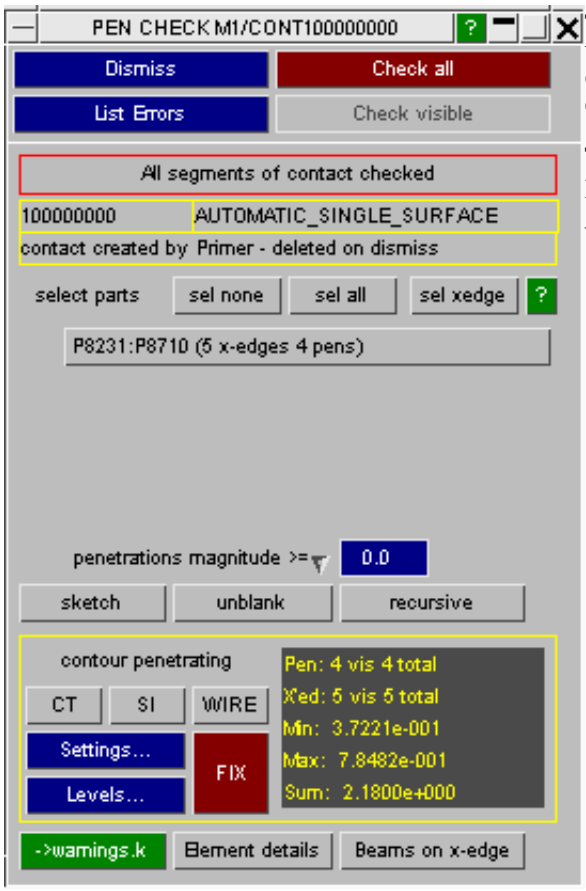
LIST Listing a summary of parts

A summary of the selected parts (name, material type, element type and #elements) is listed to the screen.

CHECK Checking parts

The selected parts are run through the standard checking routines, and any errors found are summarised on the screen.

PENCHECK Checking parts for penetrations



The selected parts will be put into a "private" automatic surface to surface contact (or single surface if more than 2 parts) and checked for penetrations. The standard penetration check panel is displayed.

This will allow you to contour any penetrations between the panels or if you switch mode, to contour gaps between panels.

When you dismiss the panel the vapid contact will be deleted.

RENUMBER Renumbering part labels

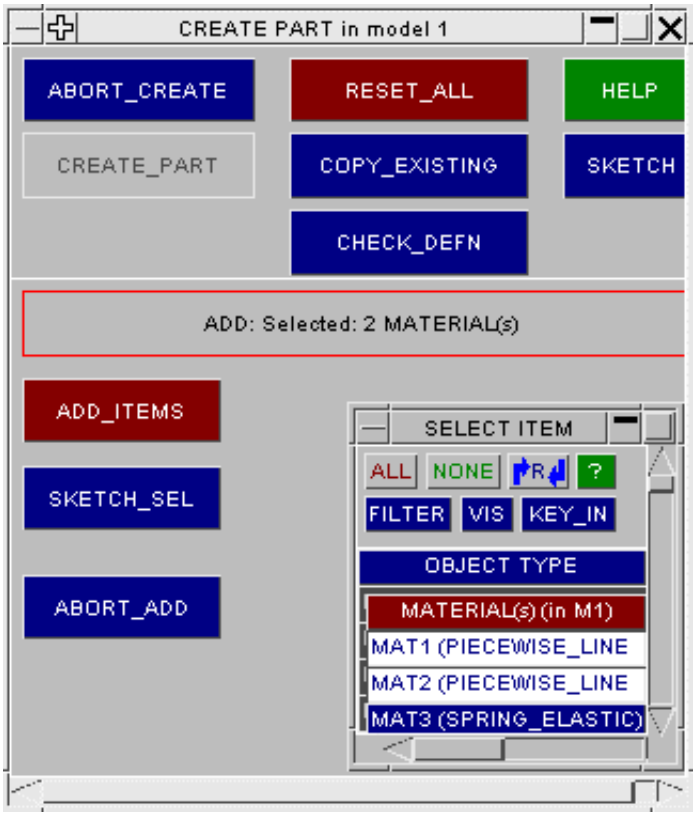
The standard renumbering panel is mapped for the relevant model, allowing part labels to be updated. To update a single part label it may be easier just to **MODIFY** it.

PART_CONTENTS... Adding and removing elements to and from parts

Both the **CREATE** and **MODIFY** functions use the **CONTENTS...** option to list part contents, and also to add and remove elements.
This maps the content editing sub-window, in which you can

ADD_ITEMS	To specify elements to be added to the part;
REMOVE_ITEMS	Specify elements to be removed from the part.
EMPTY_PART	Empty the part of all elements.

In both the **ADD** and **REMOVE** cases the standard object selection menu is shown, and you select (by element, part. model, or anything else) what is to be added or removed.



For example in this **ADD** case the user has selected two materials.

ADD_ITEMS	Adds the elements in these materials to the list of elements for this part.
SKETCH_SEL	Sketches the items that have been selected, to confirm they are correct.
ABORT_ADD	Aborts the ADD operation.

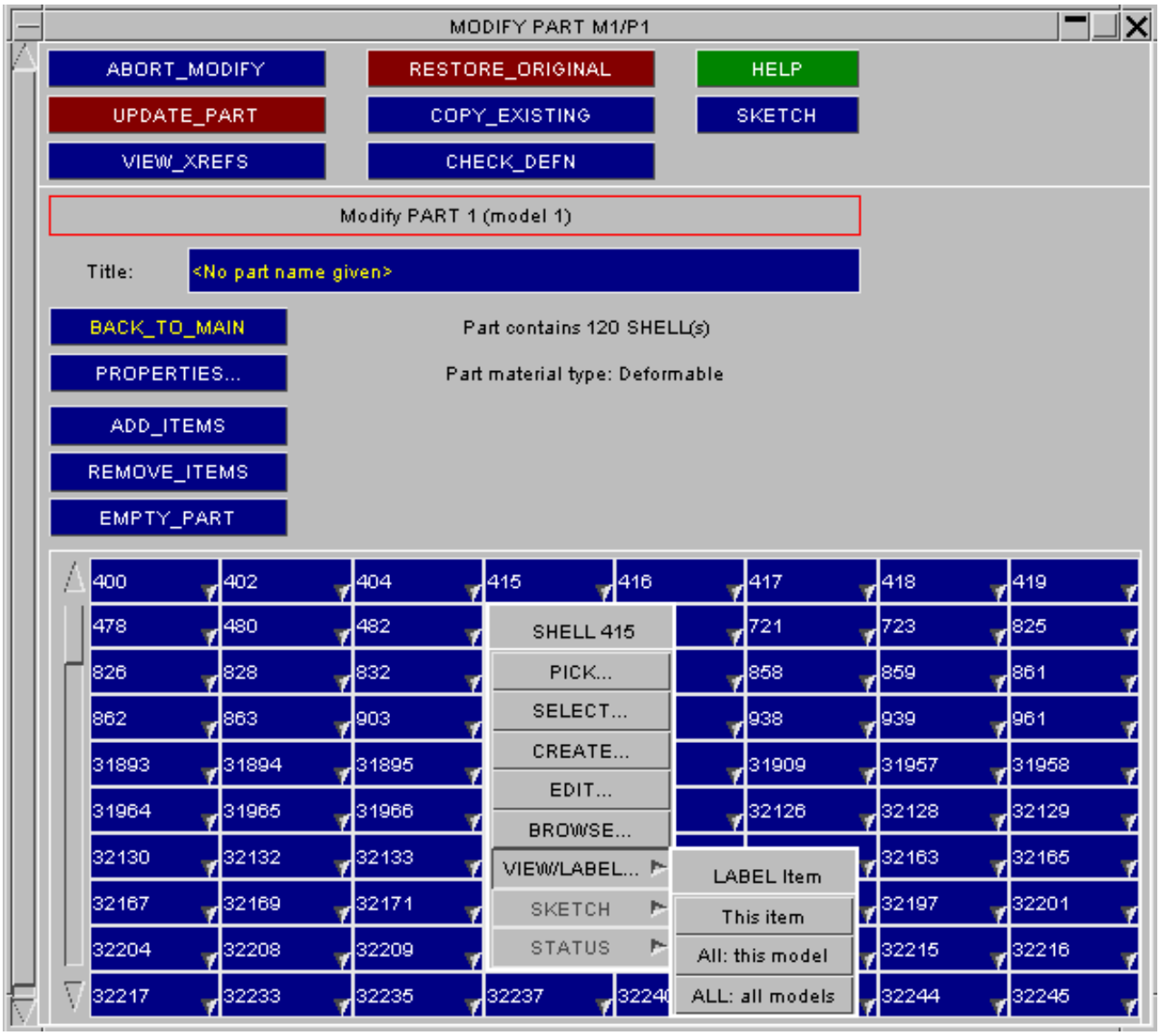
If the part already contains elements then its implicit type is already defined, and only elements of the relevant types are selected for processing.

If the part is empty then the first element type encountered in the addition list is used to determine its type.

Therefore when creating a new part take care to choose elements of the correct type. However when adding to or subtracting from an existing part you can choose broader categories, eg materials as here, knowing that only the relevant items will be selected for addition or removal.

It is legal to select for addition elements that are already in the part: they will not be duplicated since the outcome of an **ADD** operation is a logical (inclusive) OR of the existing and new elements.

Once the part contains some elements these become visible in the **CONTENTS...** sub-panel as shown below, and can be scrolled through individually.



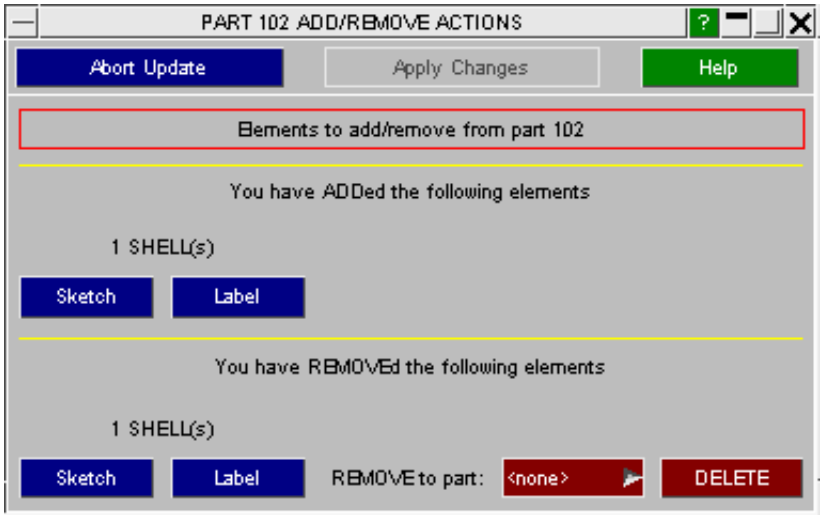
Popup menus are available for every element, giving a list of options that allow you to examine the elements in more detail. Here the user is about to look in detail at shell 415 using the **VIEW/LABEL** option.

> Note that while creating or editing a part, as elsewhere in PRIMER, you are always working with a "scratch" definition of the part.
> Elements are not actually transferred to or from this part at this stage, as its permanent definition is not updated until you **CREATE** or **UPDATE** it explicitly. At that time you will be required to confirm the transfer of elements between parts, and to determine how elements deleted from this part are to be disposed of. This is described in section PART_UPDATE below.

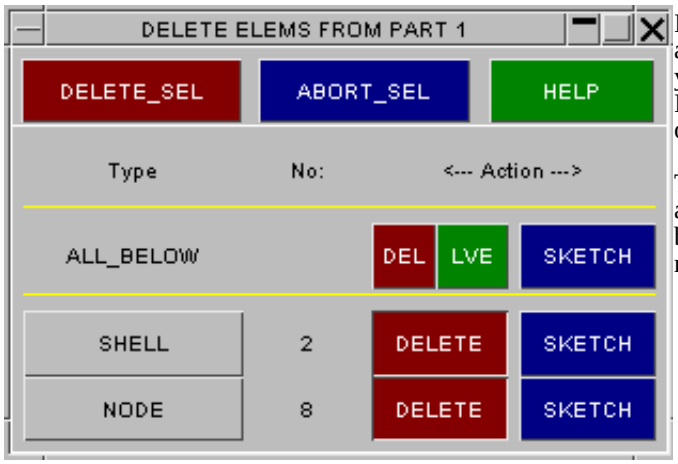
PART_UPDATE How element addition and deletion is processed on update/create

When part operations have added elements to or removed them from a part this only takes place in the scratch definition, so some extra processing is required when the permanent part definition is updated on **CREATE** or **UPDATE**.

This is done via the **ADD/REMOVE ACTIONS** panel, as shown here, which is automatically displayed.



In this example one shell has been added to this part and one has been removed from it.
Before user can apply the change he must determine what to do with the elements flagged for removal. These must be deleted or moved into another part.

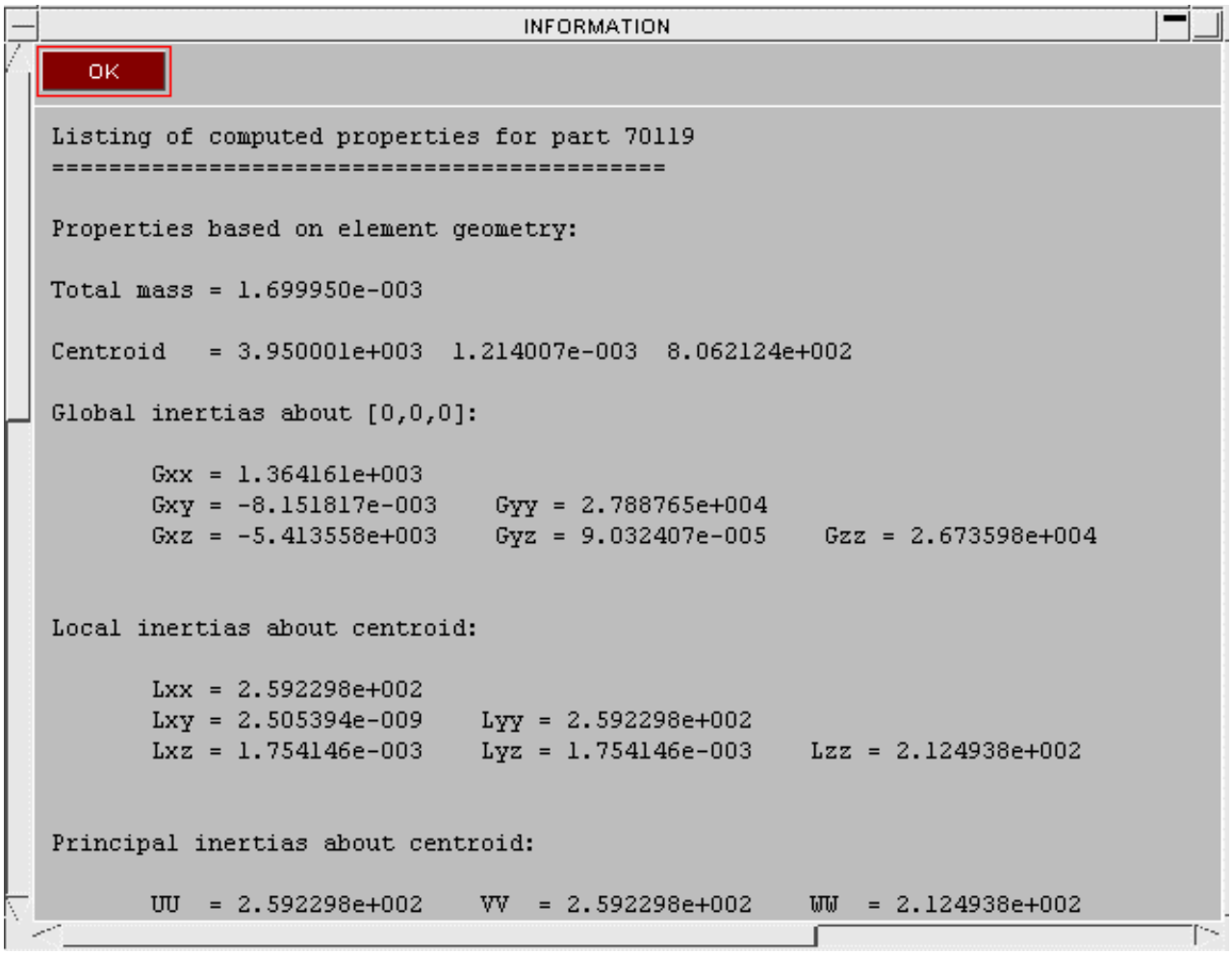


If the deletion option **DELETE** is chosen the elements and any associated items will be marked for deletion, and you will get the standard deletion confirmation panel.
In this example 2 shells are flagged for removal, and they own 8 nodes.

These will be deleted from the part once you confirm the action. If you abort the deletion the part update will be blocked as it cannot proceed without corrupting the model.

PART_PROPERTIES Calculating and displaying structural properties.

The **PROPERTIES...** command calculates the mass, C of G and Inertias of the part based on its known element properties. A typical output is shown:



"Known element properties" means:

- Element topologies must be defined, and node coordinates specified.
- Section definitions for shells and beams must be defined.
- Material definitions must be defined for all types to give density values.

Note the following exclusions from property calculations:

Any contributions due to items not in this part, but which are connected to it, are ***ignored.*** An example would be ***ELEMENT_MASS** and ***ELEMENT_INERTIA** elements which are connected to nodes on elements in this part, but not formally of it.

"Rigid" parts (see **PART_RIGID** below) for which mass and/or centroid and/or inertia have been externally specified still have their properties calculated on the basis of element properties, ***not*** on the stipulated values, and also do ***not*** include the effects of any rigid body merges.

PART_RIGID Parts using rigid materials (*MAT_RIGID)

"Rigid" parts in LS-DYNA are those which use the special rigid material ***MAT_RIGID,** and they have a whole range of special attributes which make them computationally efficient, but which can complicate their use. First a bit of theory - skip this if you already know about rigid bodies:

Rigid vs Deformable parts - some theory.

"***Deformable***" elements in LS-DYNA have their mass lumped at their nodes, and at each timestep the forces acting at each node are determined and Newton's 2nd law:

F = M.a (Force = Mass x Acceleration)

is used to determine the acceleration vector of each node, then by integration the velocity and displacement vectors are also found. Strains in elements arise due to differential displacements at nodes, and from these stresses are calculated leading to further forces for the next time-step. Each element can deform independently.

"***Rigid***" elements are treated quite differently: the total mass, C of G and inertia for each rigid part is calculated. Then at each timestep the forces acting on the centroid of this part are summed from all sources, and the resultant displacement vector calculated from its total mass and inertia terms. The resulting motion is then extrapolated to each node such that the whole part moves as a "rigid body", with no relative displacement between nodes. All element strains and stresses are zero.

This leads to the following special properties of rigid bodies:

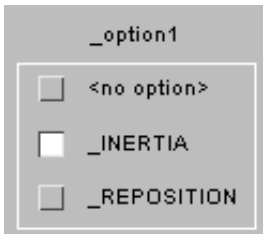
- Because in the "rigid" case motions are imposed at each node (the "a" part of Newton's equation above) by the rigid body formulation, all six degrees of freedom at each node on a rigid body are effectively constrained. Therefore no other conflicting constraints may act upon them.
- Although the default is for the properties (mass, etc) of a rigid part to be computed from its constituent elements this need not be the case, and these properties can be overwritten. The **_INERTIA** keyword appended to a part allows some or all of these properties to be defined.
- Rigid parts can be merged together such that one or more "slave" parts become subsumed into a "master", and all have their motion updated together during an analysis. The ***CONSTRAINED_RIGID_BODIES** card defines these merges.

This is not a complete treatment of rigid bodies in LS-DYNA, and you are referred to the theory, example and user manuals for more information, but it suffices for the purposes of explaining the options below.

PRIMER provides some special facilities for processing and checking rigid parts.

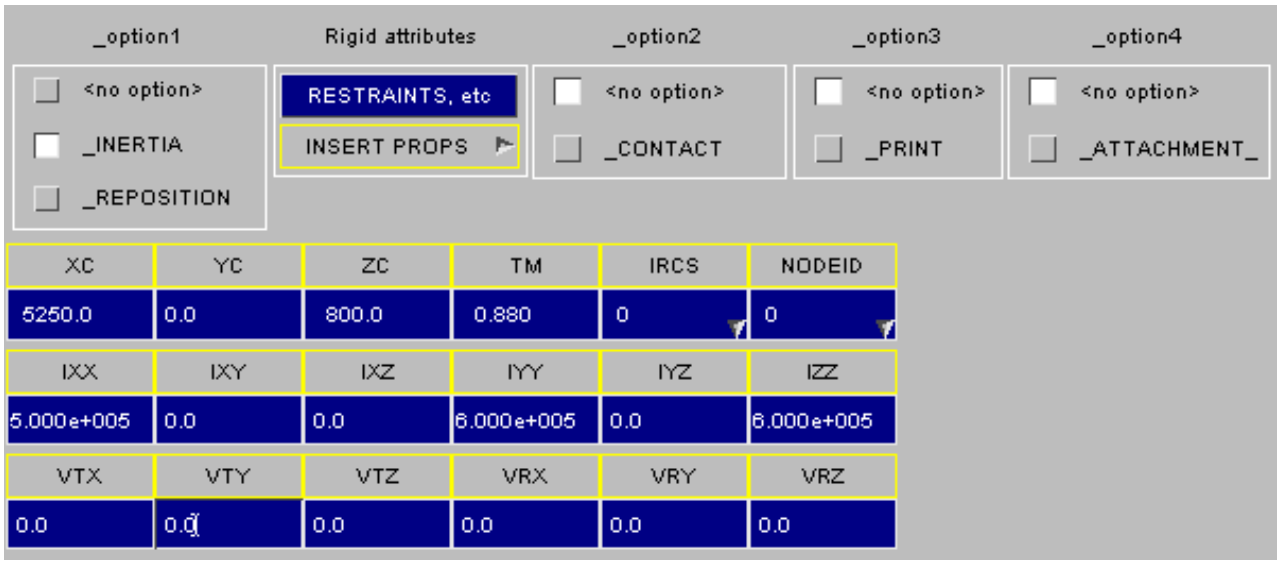
PART_INERTIA Overriding calculated properties for rigid parts.

The sub-keyword **_INERTIA** appended to a rigid part definition means that some or all of its mass, centre of gravity and inertia tensor terms are externally defined. For such a part no mass, C of G or inertia properties are calculated and the externally supplied values are used.



Initial velocities may also be defined: if they are not then the part is assumed to have no initial velocity.

PRIMER reports the rigid property fields in the create/edit panel, as shown below, and you are free to update any of these.



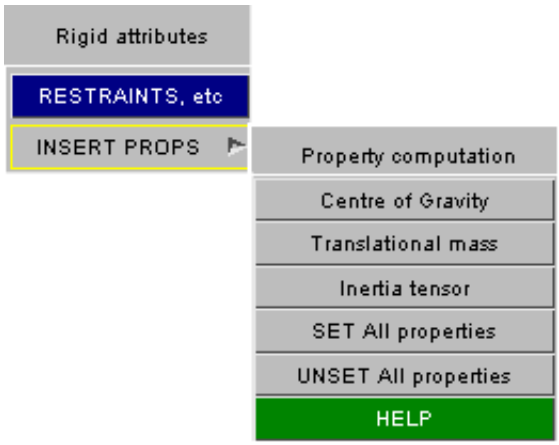
In this example the C. of G. (**XC, YC, ZC**), mass (**TM**) and inertia tensor (**Ixx .. Izz**) have all been defined. The tensor here is in the global system: a local system can be defined by setting **IRCS** to 1, then defining local axes.

Rigid attributes Calculating and (re-)setting rigid properties and restraints.

(At present **RESTRAINTS etc**, which will help with the definition of restraints and constraints for this rigid part, is not available. This is required since restraints and constraints applied directly to the nodes of rigid parts do not work, and these must instead be applied to the part itself. In addition rigid body merges, prescribed motion, "stoppers", and so on all merit special attention.)

INSERT PROPS calculates properties from the elements of this part, (exactly as described in PART_PROPERTIES above), and allows you to insert some or all of these as the "externally" defined terms above.

Use the popup menu, as shown here, to select which properties to overwrite.



Notes on using "rigid" parts

The definition of a "rigid" part is one that uses material type ***MAT_RIGID**.

- You should still give sensible density, Young's modulus and section properties for rigid parts. This is because these values are used when computing stiffness and geometry (for shells) for contact, and also mass, C of G and inertia properties if these are not externally defined.

- **_INERTIA** definitions are optional for rigid parts, and if omitted the properties will be calculated from the part's elements as described above.
- **_INERTIA** definitions *cannot* be defined for non-rigid parts. PRIMER will allow you to specify them in the editing panel on the premise that you will subsequently change the material type. However checking on **UPDATE**, or when using **CHECK_DEFN**, will flag this as an error.
- The rule that parts can only contain one type of element still holds true for rigid parts. To assemble a rigid body containing different element types create one (or more) parts for each element type, then merge them together using ***CONSTRAINED_RIGID_BODIES**. The parts need not be physically connected in any way.
- Rigid parts may have common nodes with other non-rigid parts, but they may not share common nodes with other rigid parts *unless* the parts have been merged as above. (Otherwise the common nodes would be subject to multiple constraints.)
- "Extra" nodes, not necessarily attached to any elements, may be added to rigid parts using ***CONSTRAINED_EXTRA_NODES**. These have their positions updated by the rigid body equations and can be extremely useful for connecting to rigid bodies.
- The LS-DYNA manual claims that rigid parts for which mass, C of G and inertia properties are defined need not contain any elements. In practice this seems not to work, and it is recommended that you have at least one element in a rigid part, even if it is a dummy that serves no purpose.

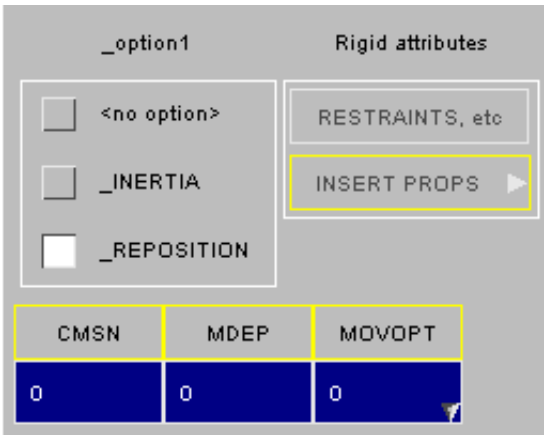
The part checking functions detect many errors associated with the use of rigid bodies, but for a comprehensive check it is necessary also to run the ***NODE** and ***CONSTRAINED** checking functions. These will, for example, detect multiple constraints on nodal degrees of freedom, and attempts to apply "rigid" constraints to non-rigid parts.

The **MODEL** > **CHECK** command, which runs all checking routines, will perform these checks for you.

PART_REPOSITION Special options for coupled analyses

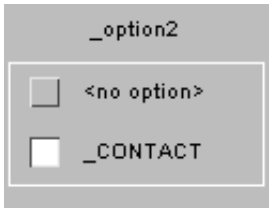
This is a specialist option which applies only to deformable parts that are to control the motion of rigid components in a coupled (CAL3D, MADYMO) analysis.

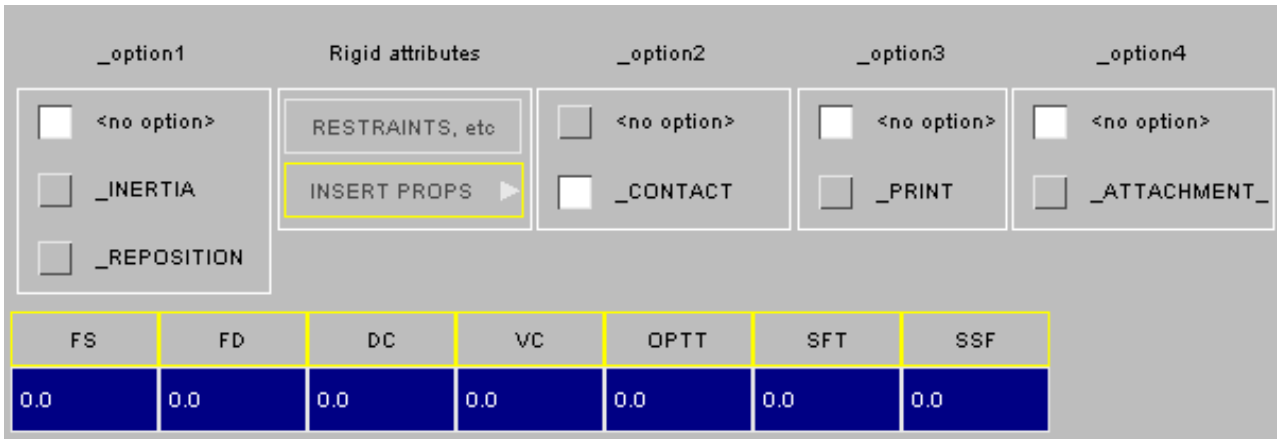
Refer to the LS-DYNA user manual for more information.



PART_CONTACT Specifying part-specific contact parameters for part-based contact.

When part-based contact (using parts or part sets) is used for the "automatic" contact types it can be convenient to have specific contact parameters for each part which supersede the default ones on the ***CONTACT** card.





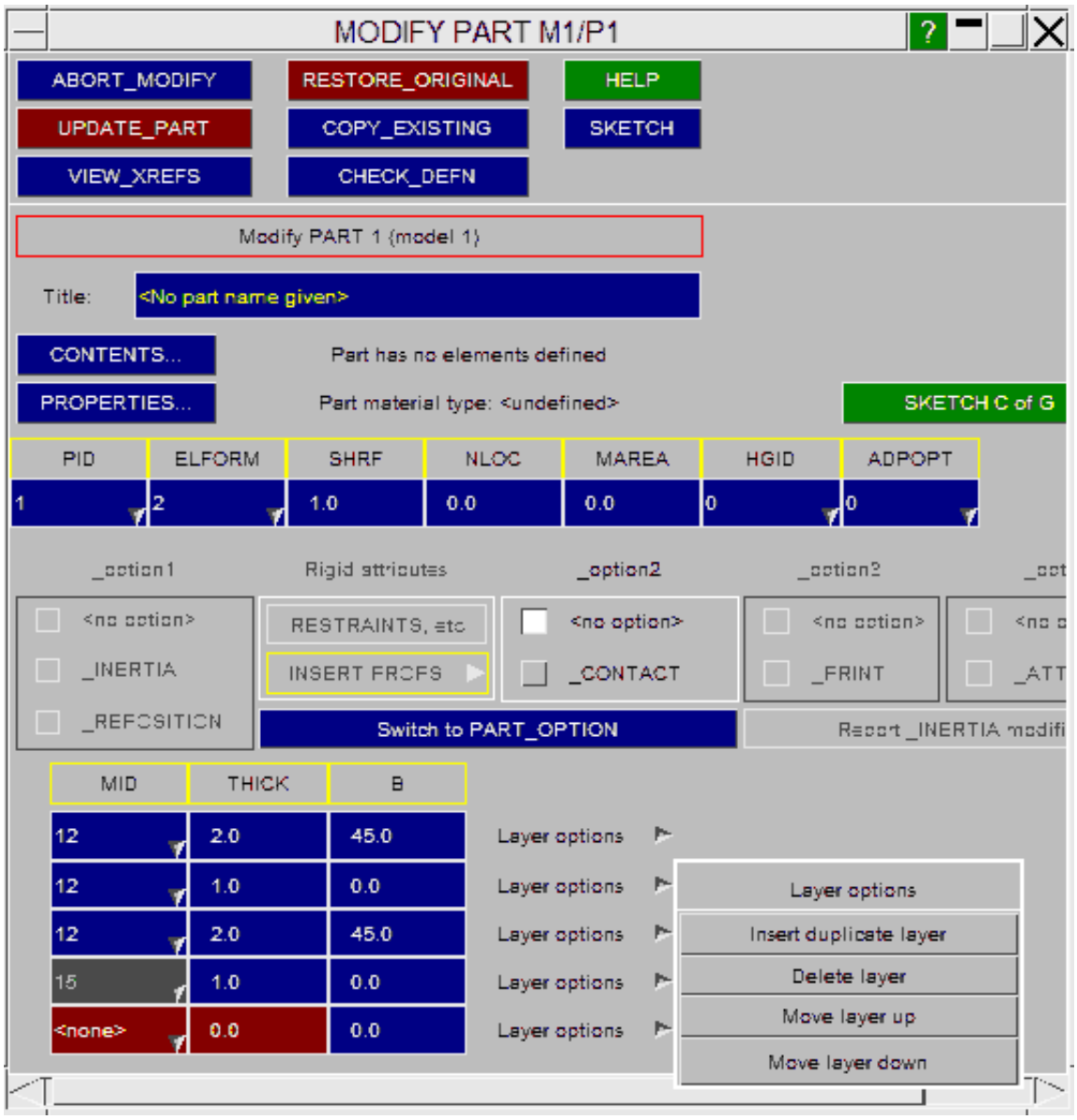
PRIMER allows you to set the **_CONTACT** parameter and define its values:

Refer to the LS-DYNA manual for the exact meaning of these parameters, and also the contact types to which they apply.

PART_COMPOSITE Specifying composite layers within a part.

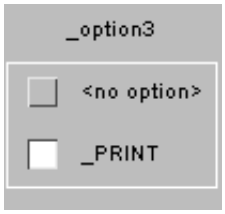
The part composite option changes the part panel to allow the user to create layers within the part. The MID and SECID are removed because it is all contained within the part cards

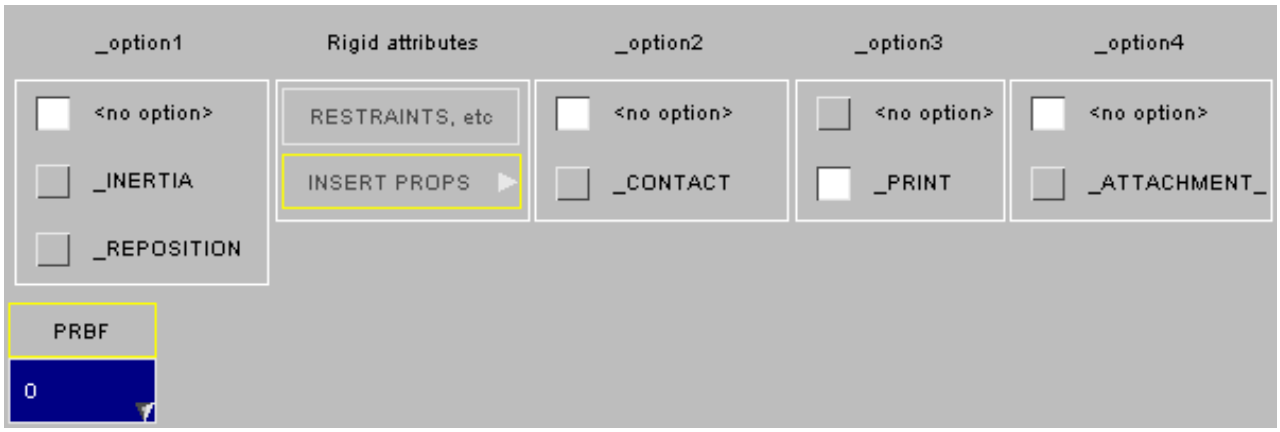
The user can enter layer information at the bottom of the panel, and it is possible to move layers up / down and create duplicate layers using the right-click popup to the right of each layer.



PART_PRINT Allows user control over whether output is written into the ASCII files MATSUM and RBDOUT.

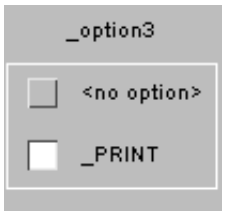
PRIMER allows you to set the **_PRINT** parameter and define its options. Refer to the LS-DYNA manual for the exact meaning of these.

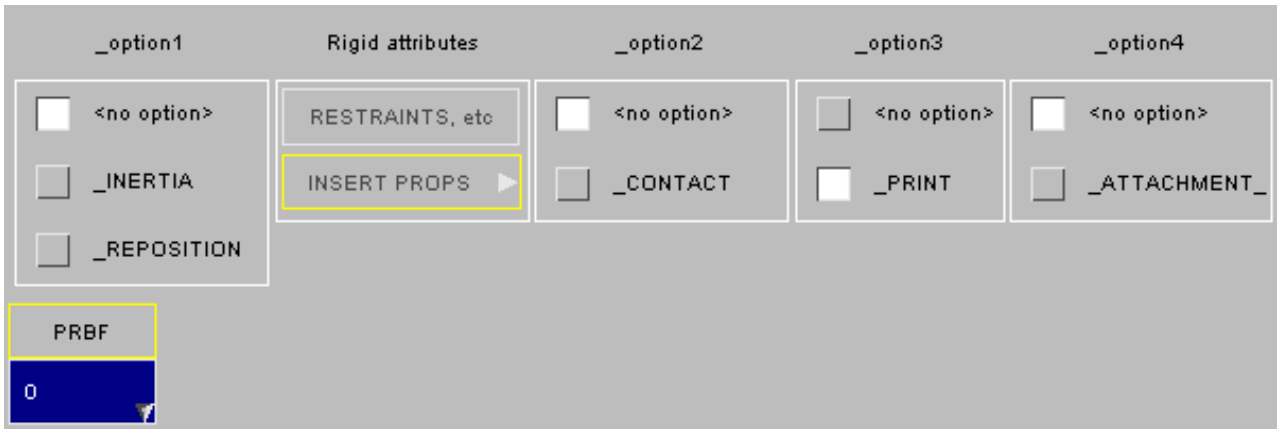




PART_ATTACHMENT_NODES Allows user control over which nodes are treated as attachment nodes.

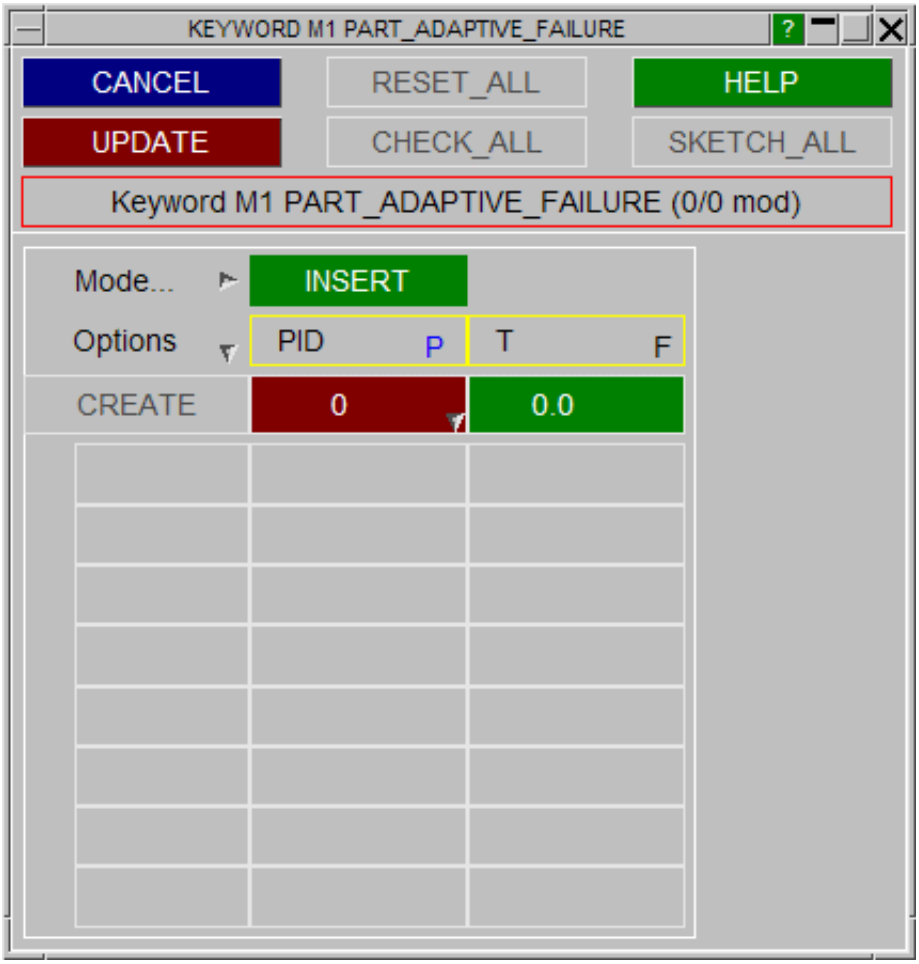
PRIMER allows you to set the **`_ATTACHMENT_NODES`** parameter and define its options. Refer to the LS-DYNA manual for the exact meaning of these.





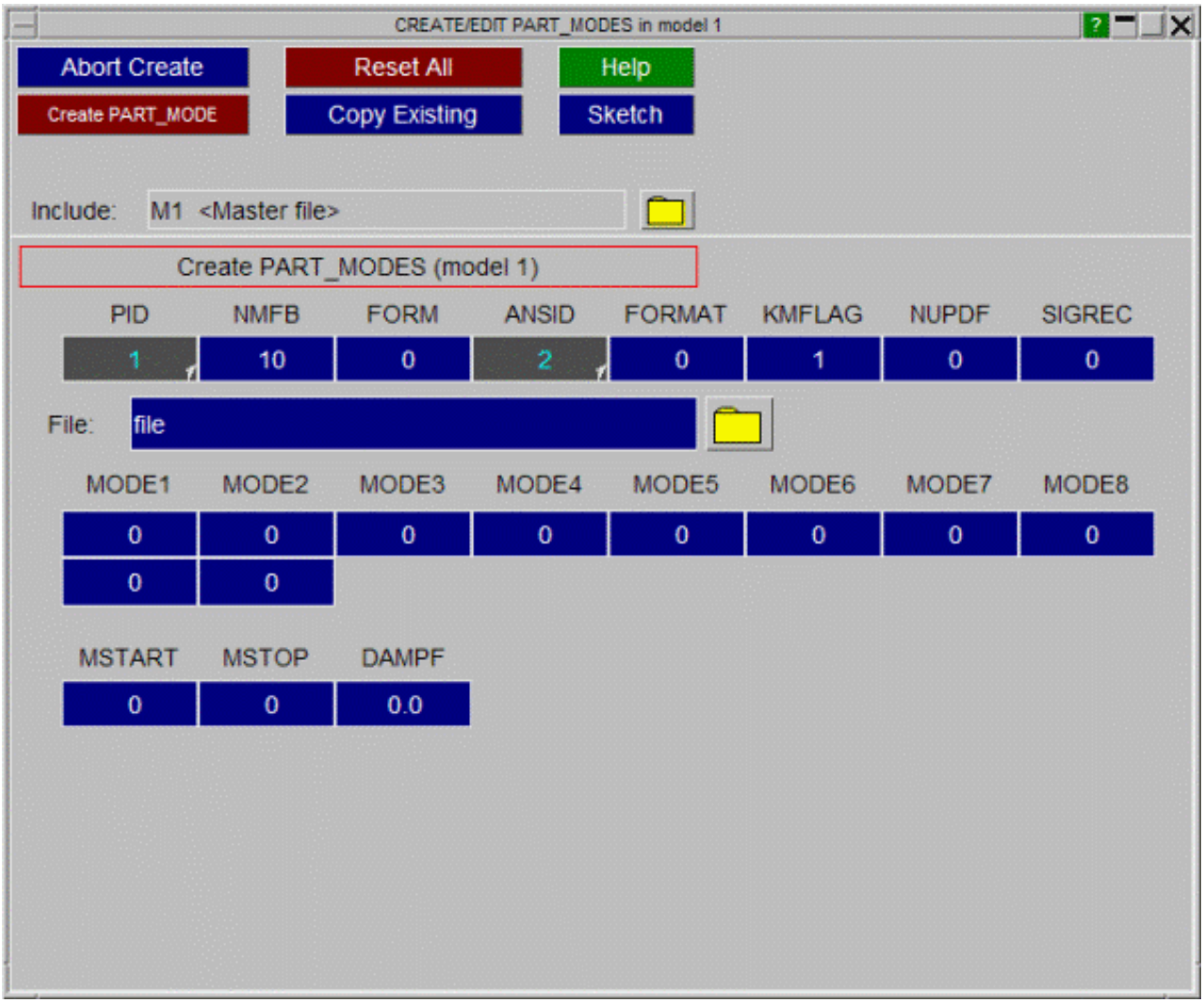
PART_ADAPTIVE_FAILURE Allows user control over which nodes are treated as attachment nodes.

PRIMER allows you to set up a **_ADAPTIVE_FAILURE** card. Refer to the LS-DYNA manual for the exact meaning of this card.



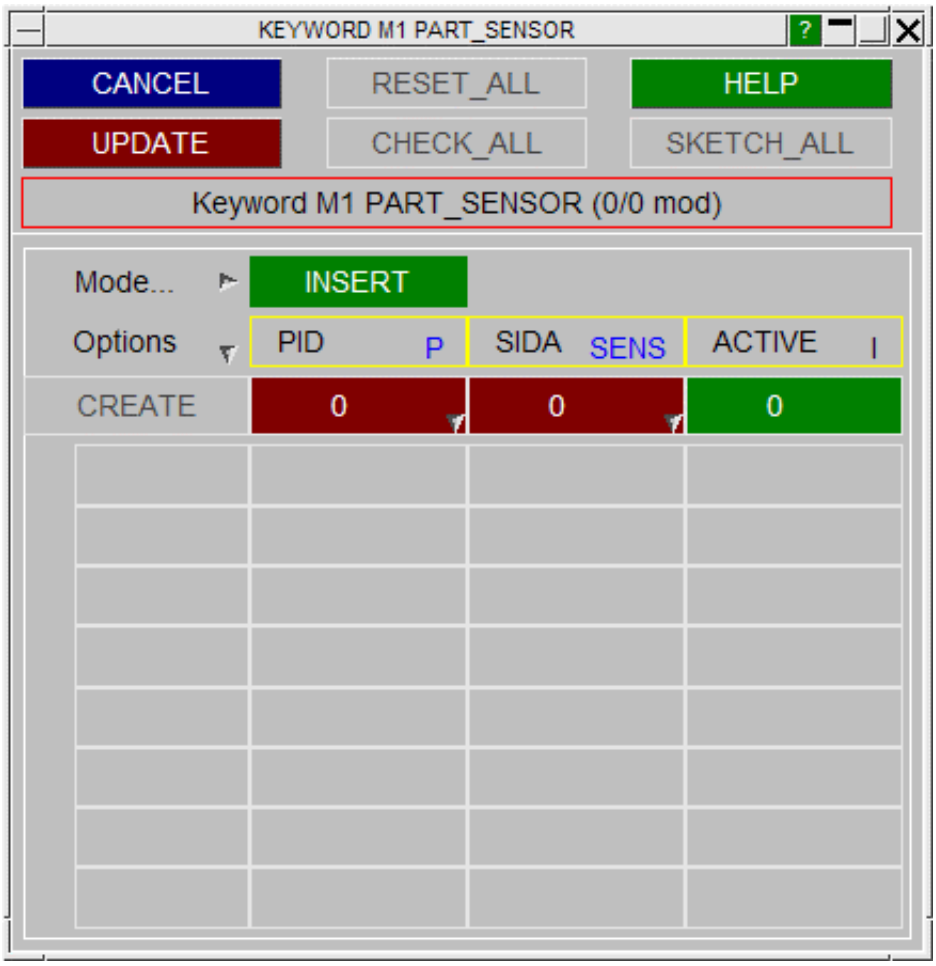
PART_MODES Allows user control over which nodes are treated as attachment nodes.

PRIMER allows you to set up a **_MODES** card. Refer to the LS-DYNA manual for the exact meaning of this card.



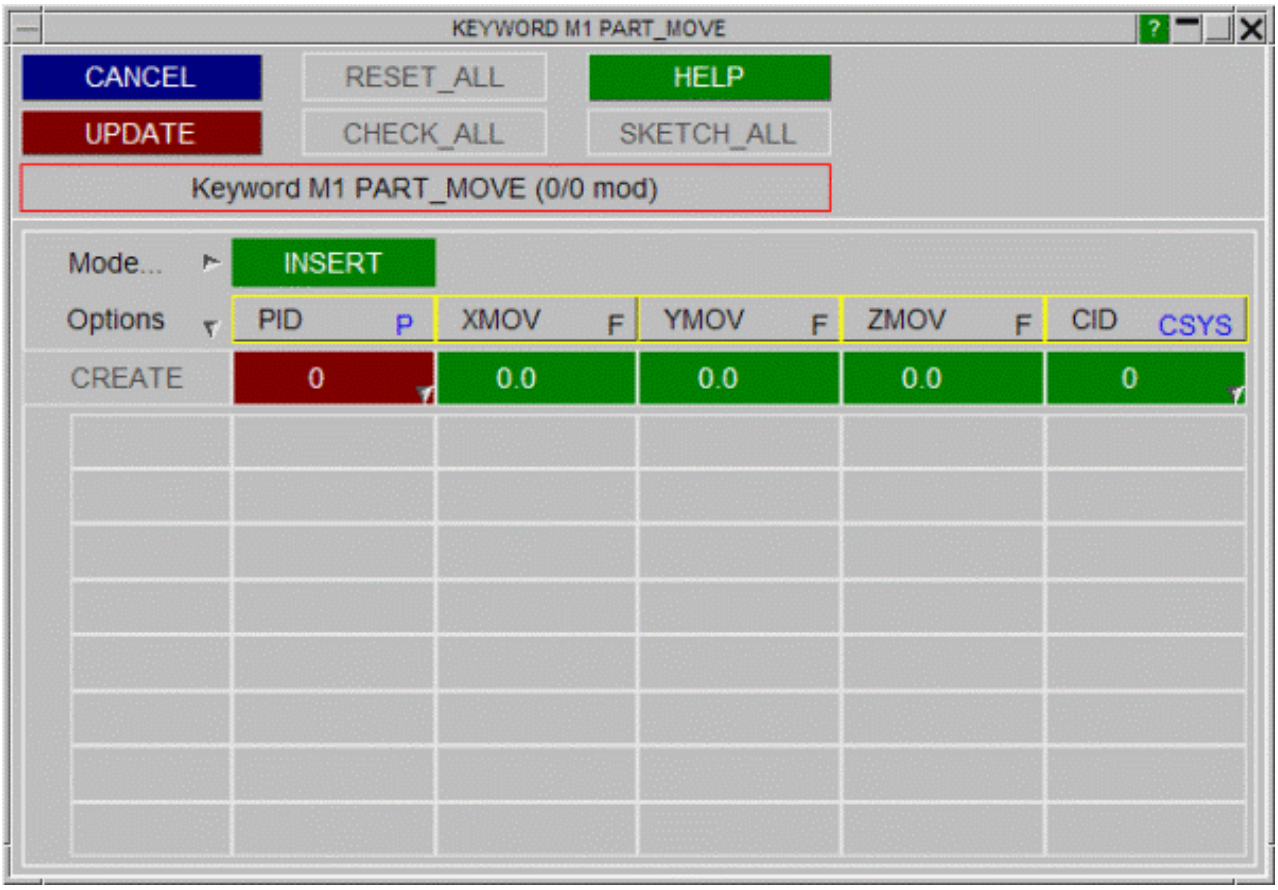
PART_SENSOR Allows user control over which nodes are treated as attachment nodes.

PRIMER allows you to set up a **_SENSOR** card. Refer to the LS-DYNA manual for the exact meaning of this card.



PART_MOVE Allows user control over which nodes are treated as attachment nodes.

PRIMER allows you to set up a **_MOVE** card. Refer to the LS-DYNA manual for the exact meaning of this card.

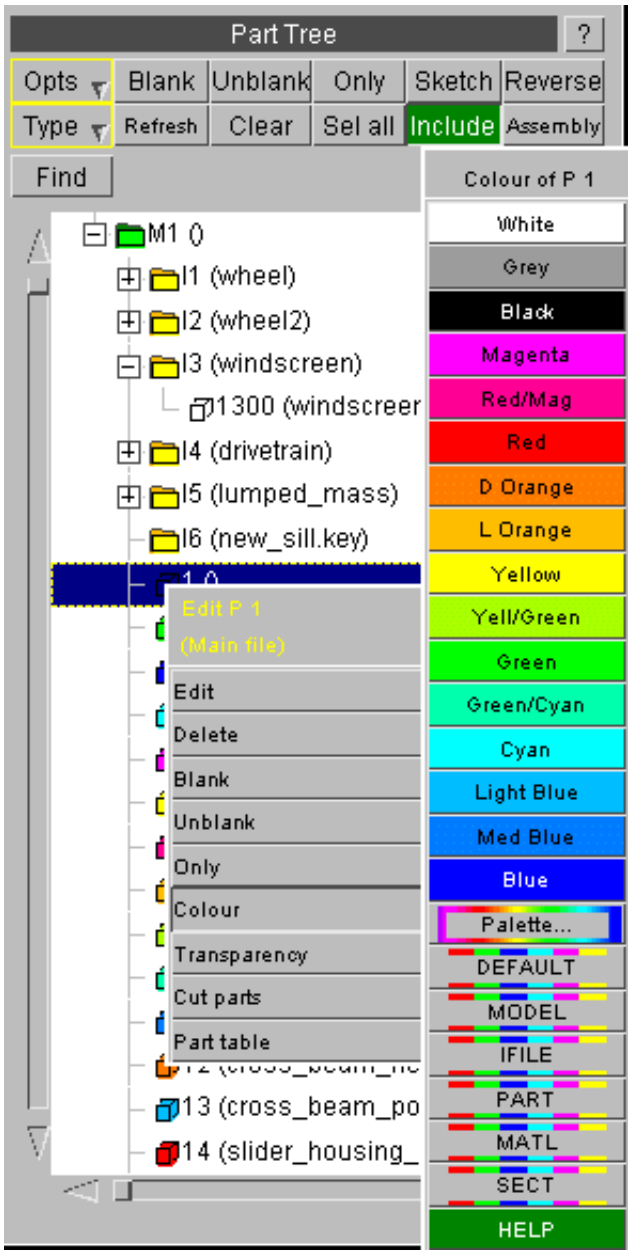


Visualising and Labelling parts

Parts are not drawn explicitly, but PRIMER can draw parts, by drawing their constituent elements, and can also label them by adding part ids to element label strings. Part visibility and labelling is therefore inherent in element visibility, controlled in **ENT**ity Viewing.

Parts may also be sketched in wireframe mode on the current image using the main and create/edit **SKETCH** options above.

COLOUR Colouring parts



Colours of parts can be set in two ways. The first method (as shown to the right) is to locate the part in the part tree, right click to bring up the edit pop-up menu and then select the desired colour option. You can select:

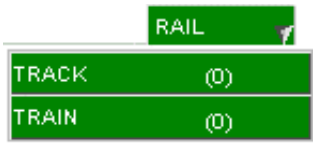
- A constant colour from the predefined range
- An arbitrary constant colour using **Palette**. This allows mixing of a colour from the full range of shades supported by the hardware.
- The **DEFAULT** colour for this item. The actual colour(s) are a function of item type, i.e. an element with a part id will inherit the colour of its part. Other items are based on their constituent sets or perhaps their labels.
- **MODEL** will apply the colour based on **Model** number modulo 13 (standard primer sequence red, green, blue etc).
- **IFILE** will apply the colour based on **Include file** number modulo 13 (standard primer sequence red, green, blue etc).
- **PART**, **MATL** (material) and **SECT** (section) colours only apply to element types with a

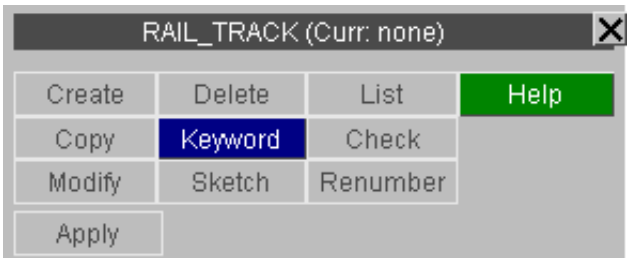
Alternatively, it is possible to use the Quick Pick tool. The same options as above are available which will then be applied to selected parts:

PERTURBATION

There are currently three sub-keywords available; **MATL**, **NODE** and **SHELL_THICKNESS**, as shown on the left. These can be created/edited with the standard keyword options in section 5.1.1 and create/edit options in section 5.1.2

RAIL





There are currently two sub-keywords available, TRACK and TRAIN, as shown on the left. These can be edited through the generic Keyword Editor.

RIGIDWALL: Defining Rigid ("stone") Walls.

Top level menu
Create/Edit
Keyword edit
Dragging walls
Editing planes
Visualisation

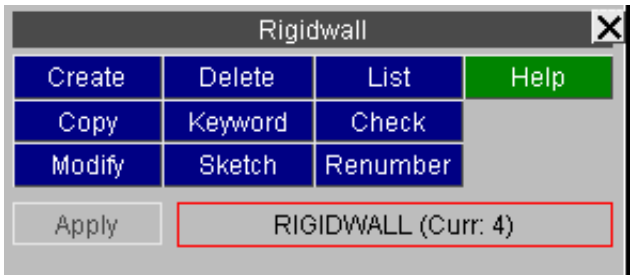
Rigidwalls in LS-DYNA are convenient and computationally cheap rigid geometrical shapes against which nodes on a deformable body can impact.

The ***RIGIDWALL** keyword in LS-DYNA supports the following sub-types:

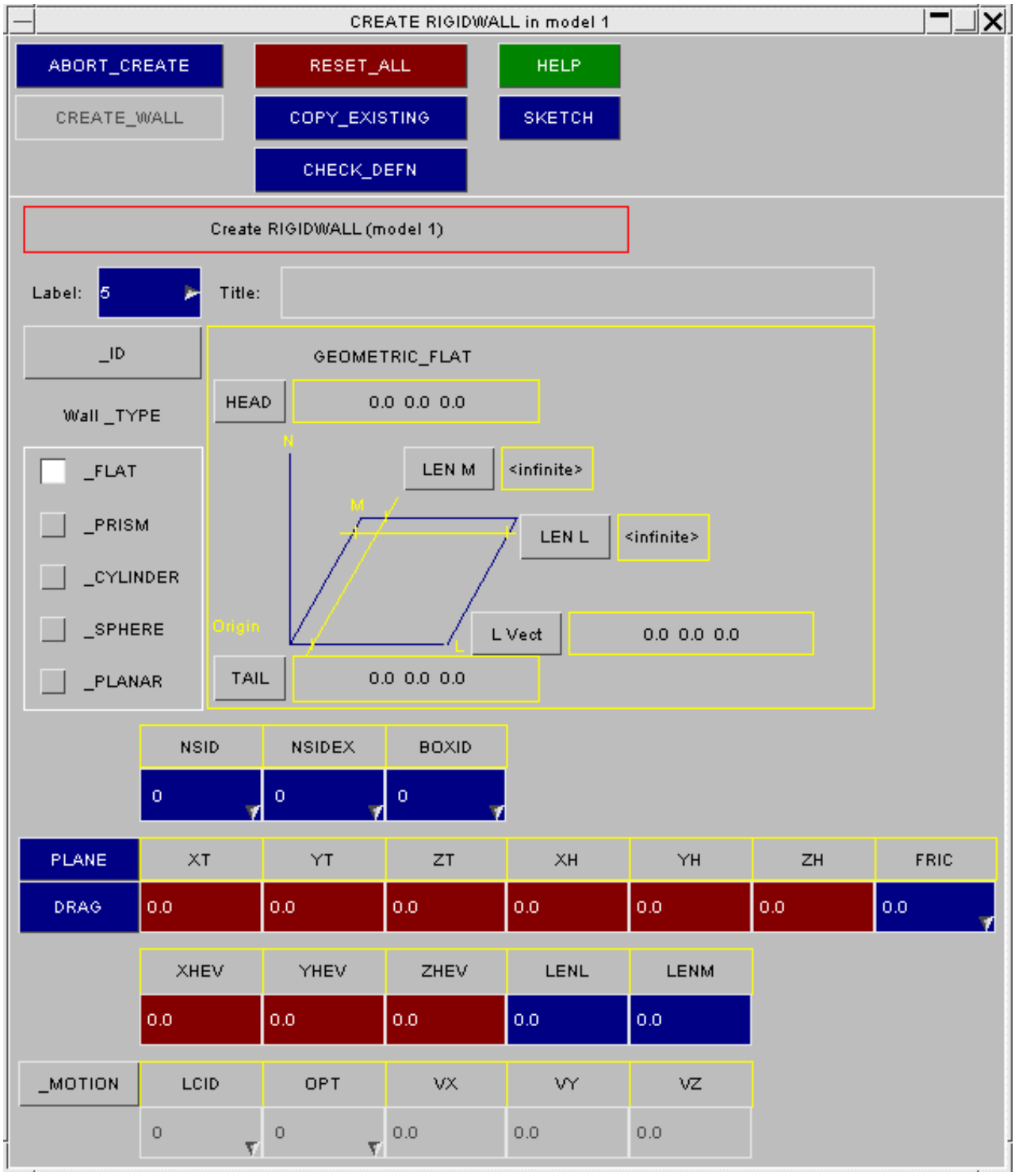
```
RIGIDWALL_GEOMETRIC_FLAT      RIGIDWALL_PLANAR
                   _PRISM
                   _CYLINDER
                   _SPHERE
```

This figure shows the top-level RIGIDWALL menu.
All rigidwall sub-types may be edited both explicitly (via Create/Edit panels) and via the generic Keyword editor.

Options have their standard meanings as defined in section 5.1.1.



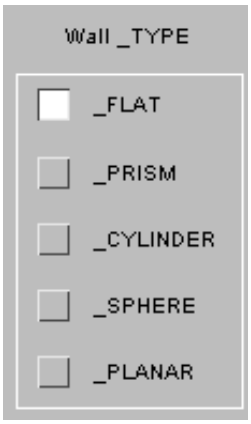
Create and Edit functionality



This figure shows the standard wall create/edit panel.

Its detailed layout changes with wall type: this example shows **`_GEOMETRIC_FLAT,`** although _PLANAR is the most commonly used option.

Selecting a different wall subtype



The detailed layout of the panel above changes as the different wall sub-types are selected.

In particular note that the ***RIGIDWALL_GEOMETRIC** types may only have the optional suffix **_MOTION**; whereas ***RIGIDWALL_PLANAR** may have a wider range of suffices. The LS-DYNA manual pages on the subject describe the various combinations of type and suffices available.

DRAG: Using the mouse to drag a wall into position.

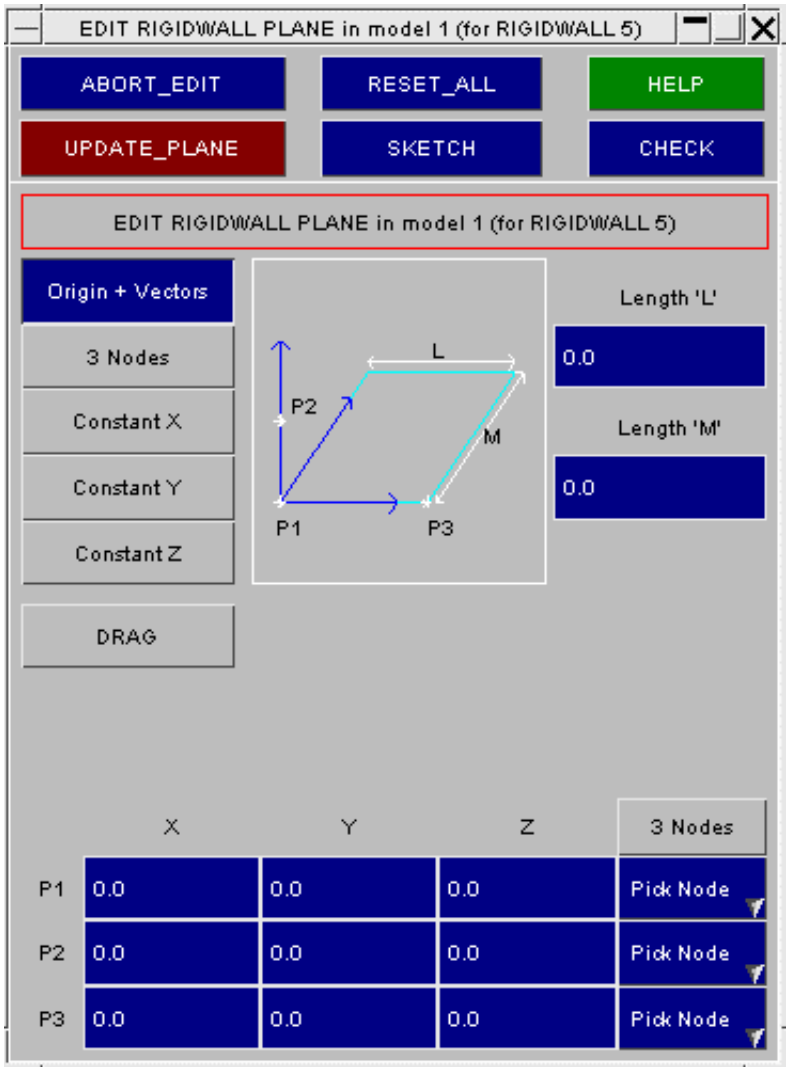
All rigidwall types can be dragged into position on the screen using the mouse. The mouse button determines the global axis along which it moves:

- X : Left mouse button
- Y : Middle
- Z : Right

END_DRAG terminates the dragging operation.

PLANE: For _GEOMETRIC_FLAT and _PLANAR wall types only

For walls defined by a flat plane the standard "plane" editor may be used.
This allows graphical definition of the plane geometry via a range of methods.



Rigidwall Keyword editing panel

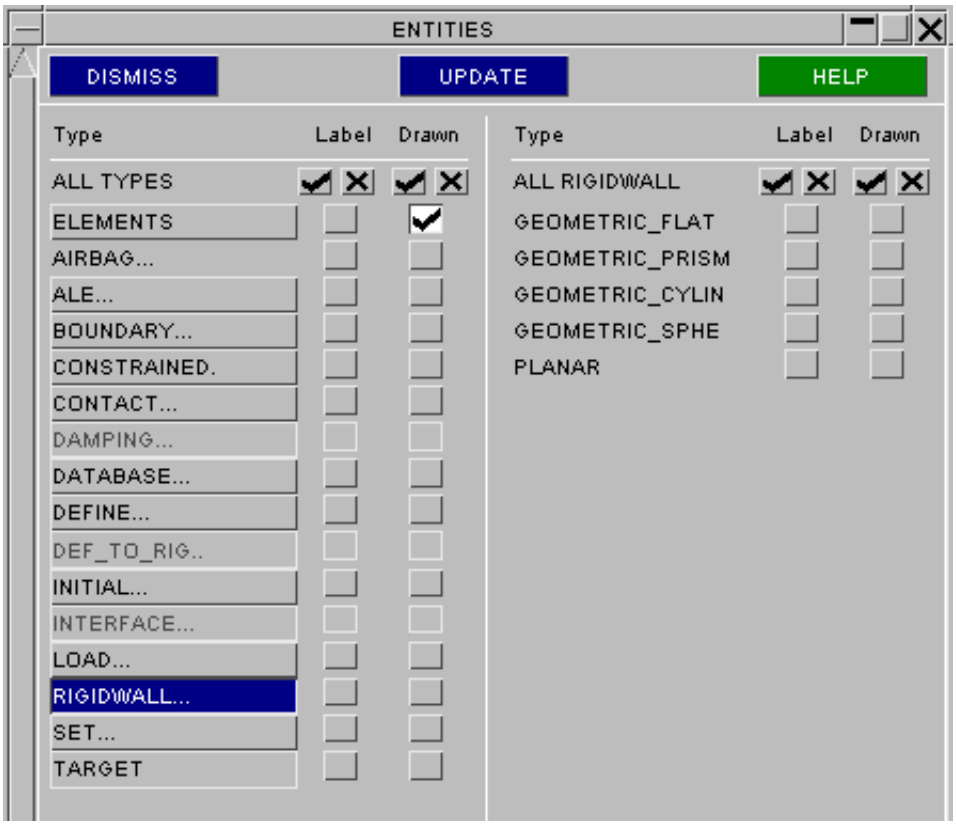
All rigidwall sub-types can also be processed using the generic Keyword editor panel an example of which is shown below.

The Keyword editor "AUTO _suffix" Displaying all _PLANAR suffices simultaneously
Because there are so many suffices to the `_PLANAR` rigidwall type, which may be used in many permutations, the **AUTO** suffix allows all such types to be displayed at the same time.

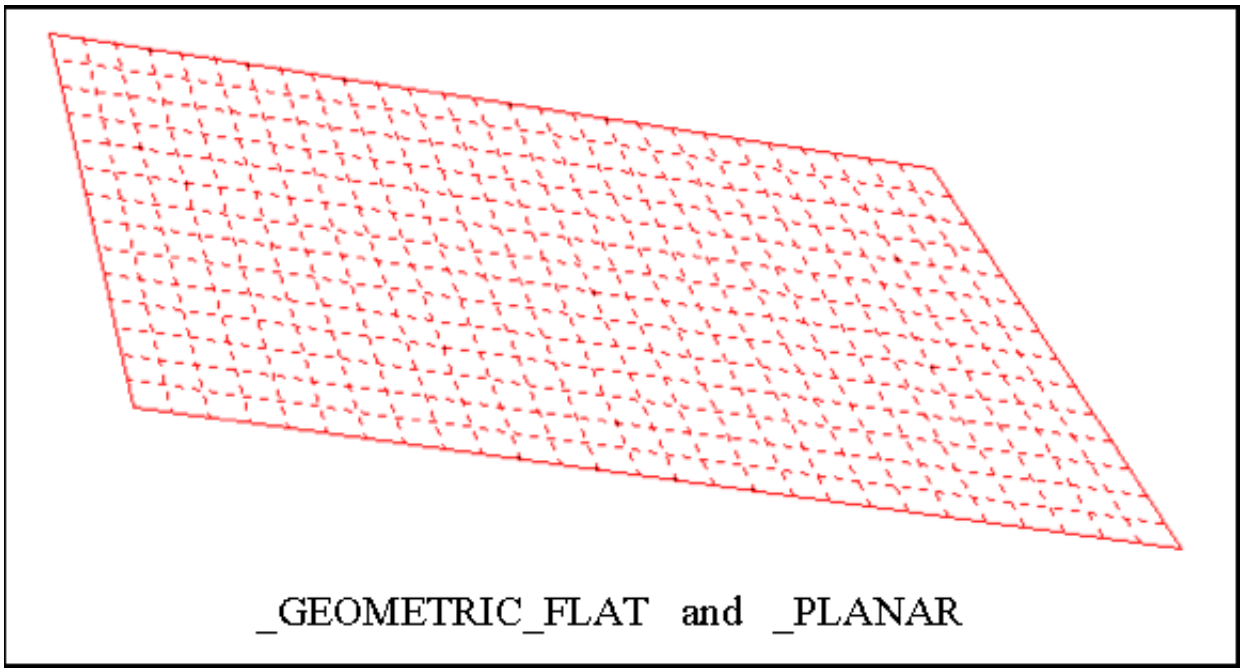
When **UPDATE** saves the editor status walls will only have a given suffix appended if the data fields for it are non-zero.

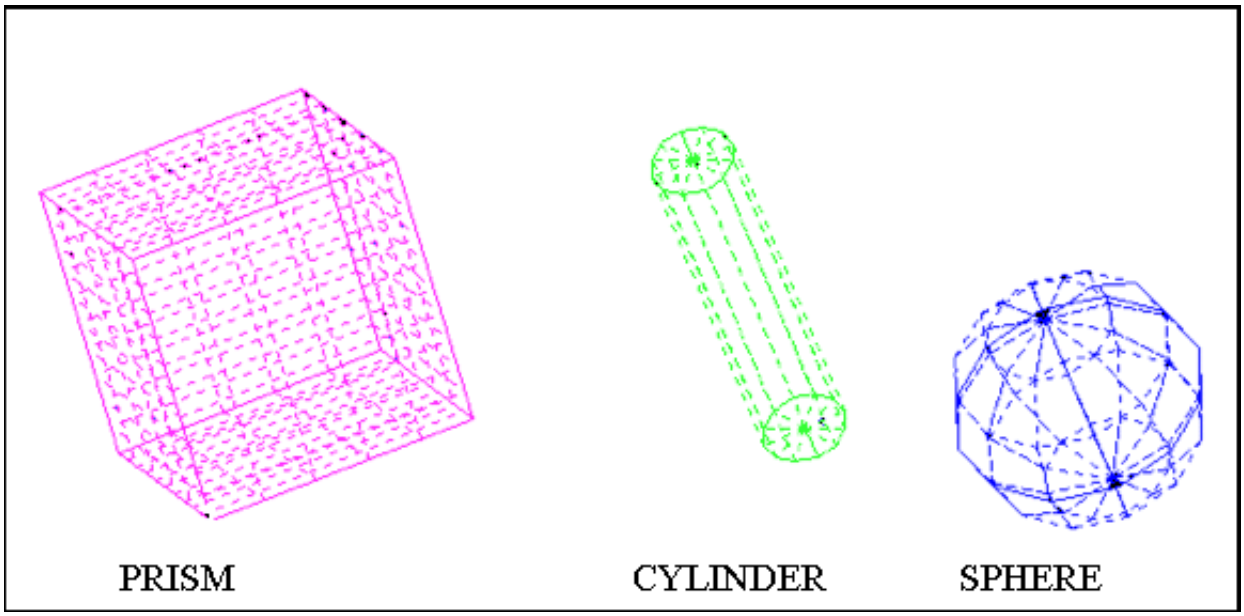
Visualising RIGIDWALLS

All rigidwall types may be visualised in **ENT**ity Viewing, also by the **SKETCH** functions above.



When **_FLAT** and **_PLANAR** walls have infinite side lengths then a dimension of approximately three times the diagonal of a box enclosing the model is used for graphical purposes. (Drawing an infinite object on a finite computer screen requires some compromise!). If sets and nodes are turned on as "extra" objects in **VIS_2** then the slave nodes for the walls will be drawn as well.





Note on scaling of finite RIGIDWALLS during **ORIENT** operations.

Prior to release 9.3RC2 PRIMER did not apply any **ORIENT** scale factors to the "finite" dimensions of rigidwalls. This was in keeping with the general policy of not applying Orient scale factors to "scalar" length dimensions since, for the most part, this is inappropriate.

However the finite dimensions of Rigidwalls are a special case, and the following scaling logic is now applied:

The local axis system of the rigidwall is calculated, that is:

- **N** is the normal axis (from tail to head)
- **L** is the first in-plane axis (defined by vector <hev>)
- **M** is the second in-plane axis, determined from the cross-product **N x L**

Any non-zero (ie non-infinite) "length" dimension is projected along the appropriate axis from the wall origin (tail coordinate) and the resulting vector is scaled by the [**Sx,Sy,Sz**] scale factors specified in the **ORIENT** operation. The length of the resulting vector is calculated and, corrected for sign if necessary, this becomes the new finite length.

The details of which dimension is projected where are as follows:

Wall type	Dimension	Projected onto	Notes
GEOMETRIC_FLAT PLANAR	LENL LENM	L axis M axis	
GEOMETRIC_PRISM	LENL LENM LENP	L axis M axis N axis	
GEOMETIC_CYLINDER	RADCYL LENCYL	L axis N axis	Asymmetric scale factors (Sx != Sy != Sz) will influence wall radius according to the wall's orientation, since radius operates in both L and M axes, but only L is used here.
GEOMETRIC_SPHERE	RADSPH	N axis	As for _CYLINDER case above

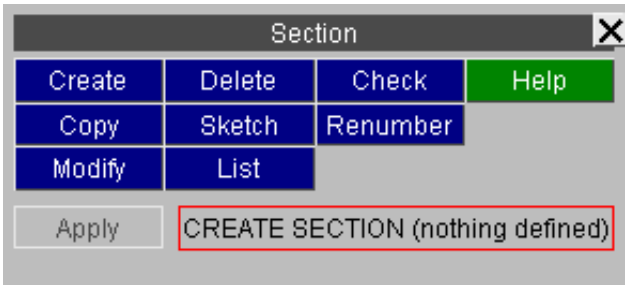
SECTION: Defining Element Sections.

Top level menu
Create
Copy
Edit
Delete
Visualisation

The ***SECTION** keyword in LS-DYNA are used to define the section properties of elements. Sections are referred to from ***PART** cards. For the possible options see the type pop-up menu in Making a new section definition below.

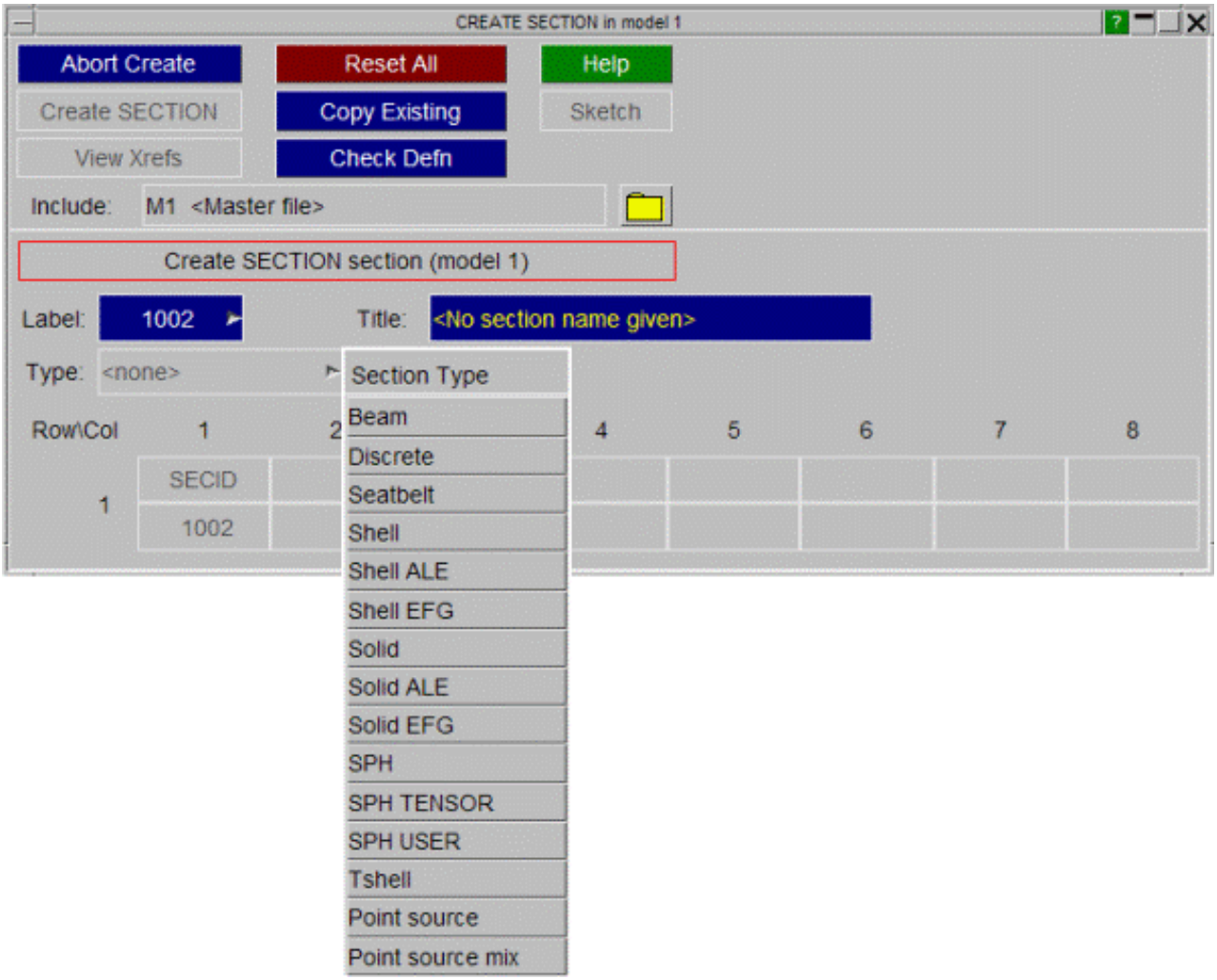
Sections of all types share a common numbering sequence (thus you cannot have ***SECTION_SHELL #1** *and* ***SECTION_SOLID #1**).

This figure shows the main section create/edit panel.
All functions have their standard meanings as described in section 5.1.1.

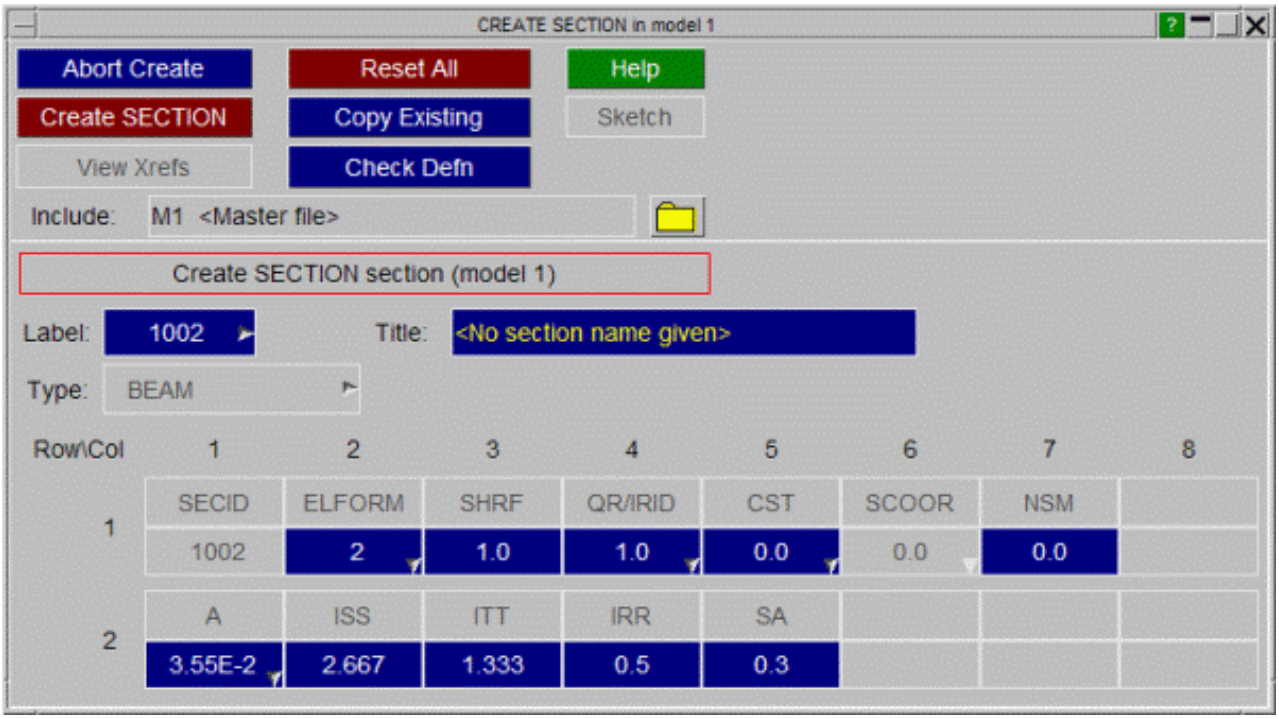


CREATE: Making a new section definition
Initially a new section has no **_type** defined, and it is necessary to define one.

Use the Type: popup menu, as shown in this figure to define an element type.



Once the section type has been defined, the relevant keyword cards appear on the editing panel, organised as shown in the LS-DYNA manual.
In this example the user has selected type **_BEAM**, and filled in the basic data for a section.



COPY Copy existing section(s) to make a new section(s).

When sections are copied the default is only to copy the section definitions themselves.

When **RECURSIVE COPY** is used all parts, elements, etc using the sections are also copied.

MODIFY Modify the attributes of an existing section.

Existing sections may be edited using **MODIFY**, which maps the same panel as above, except that data fields are already populated.

Where the section is referenced by a Part which contains elements, the topological type of the section cannot be altered. For example if a section is associated with a part containing shell elements, PRIMER will not allow the section type to be changed to **_SOLID**.

Any modifications made to the section definition will not be made permanent until the **UPDATE_SECTION** button is pressed. At this point a the local copy which has been updated is used to overwrite the version in the model.

DELETE Delete existing section definitions.

The **DELETE** function deletes the selected sections. However you cannot delete a section that is in use by a **PART** unless you remove it from the part definition, or delete that too. To help with this the following two switches may be used:

- **DELETE_RECURSIVE** Will select for deletion the parts, associated elements, and so on that reference this section.
- **REMOVE_FROM_SETS** Is often also needed if parts are to be deleted, as their elements, the connected nodes, and often the parts themselves may be included in sets.

A good way of getting rid of surplus (unused) sections is to turn these two switches off, then select all sections for deletion. Only those which are not used by anything will actually get deleted.

KEYWORD Generic keyword editor

KEYWORD starts the generic keyword editor which allows creation, deleting and modification of multiple section cards. This is useful for modifying multiple section cards in a single operation.

SKETCH Sketch elements belonging to sections on the current image.

SKETCH sketches on top of the current image the elements of the parts which reference the selected sections.

LIST List a summary of selected sections

The main attributes of the selected sections are listed to the screen.

CHECK Check selected sections for correctness

The selected sections are processed through the section checking functions, and any errors found are summarised to the screen.

Section checking is comprehensive: it detects both numerical errors (eg -ve area) and parameter errors (eg <type> indices out of range).

Individual sections may also be checked during creation and modification by the **CHECK_DEFN** command on those panels.

RENUMBER Change the labels of sections in a model

RENUMBER lets you change any or all section labels within a given model using the standard renumbering panel.

To change the label of an individual section it may be simpler just to **MODIFY** it.

END_SECTION terminates the section editing process.

Visualising *SECTION definitions

Sections are not drawn explicitly, but may be displayed by **SKETCH**ing the parts and elements that reference them.

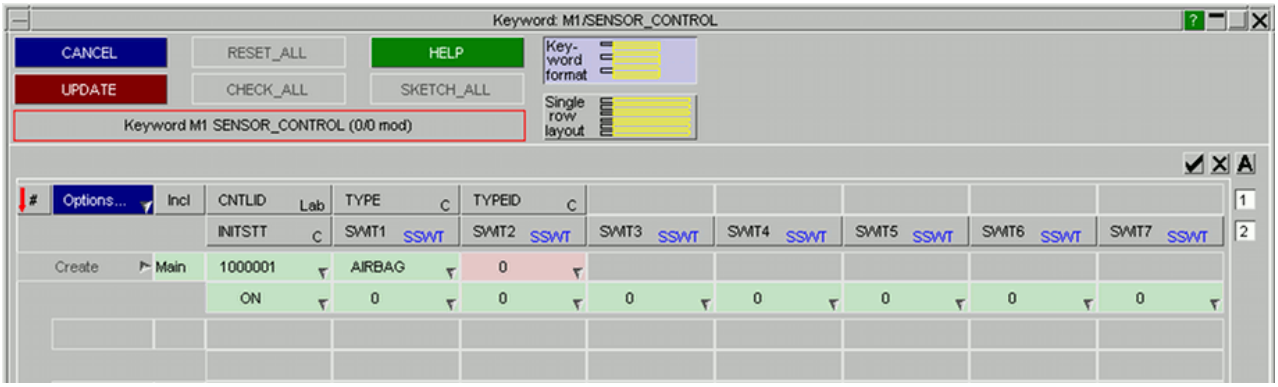
In addition the colour of part-based elements may be related to section id using the **Display > Colour > Colour all by...**. method as described in section 4.1.2

SENSOR

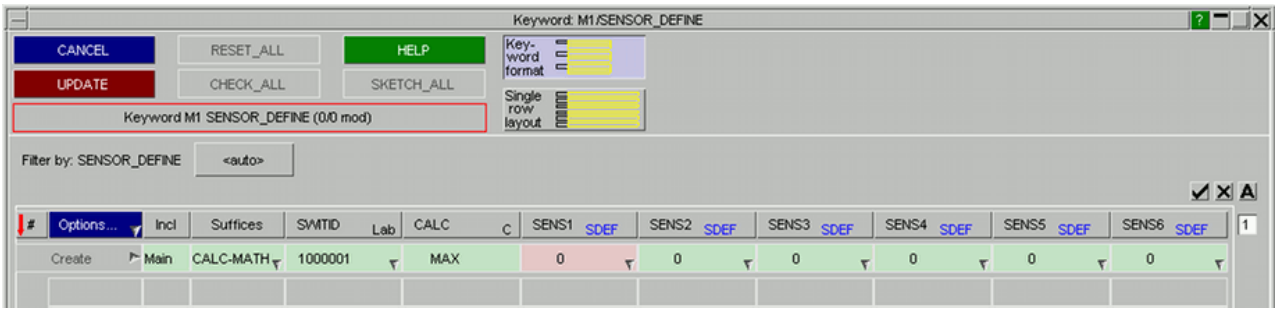
There are currently seven sub-keywords available, _CONTROL, _DEFINE_CALC-MATH, _DEFINE_ELEMENT, _DEFINE_FORCE, _DEFINE_NODE, _SWITCH and _SWITCH_CALC-LOGIC.

These can be edited through the generic Keyword Editor. There are 3 keyword editors used to create and edit these cards. One for the _CONTROL option, on for the _DEFINE options, and a third for the _SWITCH options.

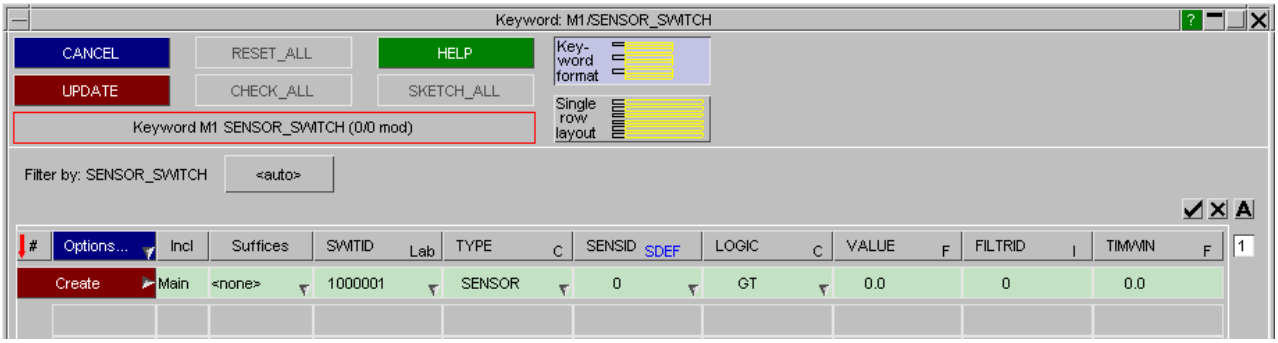
*SENSOR_CONTROL



*SENSOR_DEFINE_...



*SENSOR_SWITCH_...



SET: Defining Sets.

Top level menu
- **Create**
- **Copy**
- **Edit**
- **Delete**
- **Only**
- **Visualisation**

Set type:
- **_LIST**
- **_COLUMN**
- **_GENERATE**
- **_GENERAL**
- **_ADD**
- **_COLLECT**
- **_INTERSECT**

Set defaults

Segment sets

Locking set contents

The ***SET** keyword in LS-DYNA is used to define groups of items that can be used in many different contexts. Valid set types are:

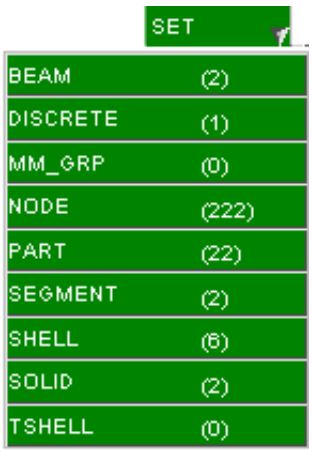
```
SET_BEAM
    _DISCRETE
    _MODE
    _MULTI_MATERIAL_GROUP
    _NODE
    _PART
    _SEGMENT
    _SHELL
    _SOLID
    _TSHELL
```

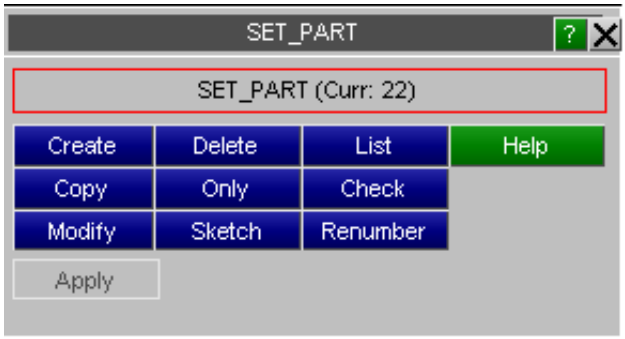
Each set types has its own numbering sequence, thus you can safely have ***SET_SHELL #1**, ***SET_SOLID #1,** etc.

With the exception of **SEGMENTS** sets simply reference other structural items, and do not themselves constitute "structure". **SEGMENTS** are a special case which are dealt with separately below.

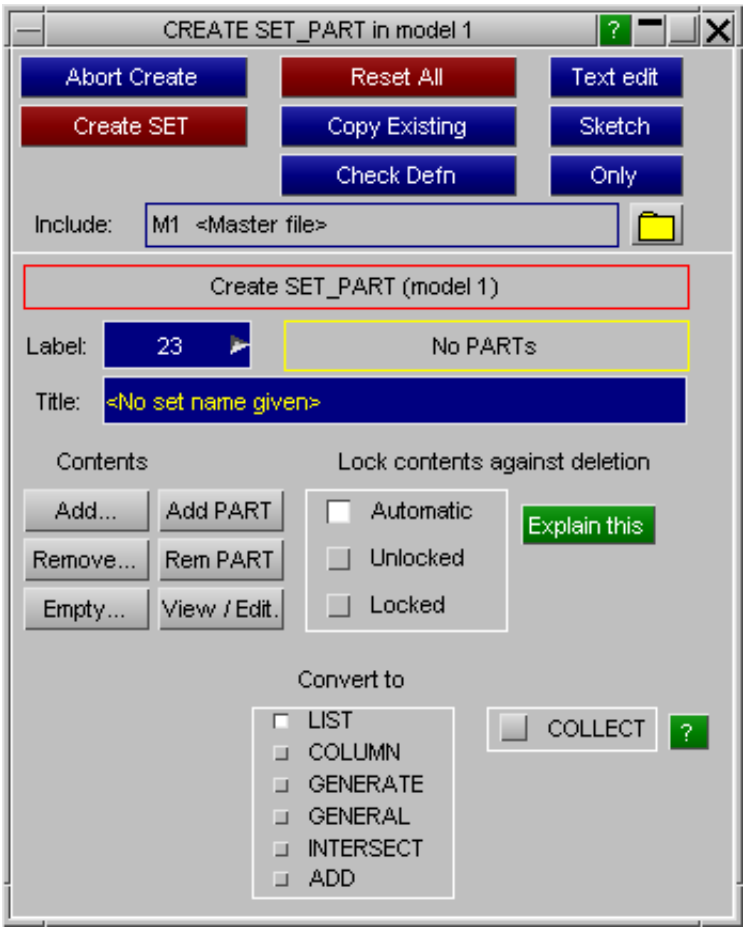
This figure shows the common top-level menu for all set types
On the right is shown the pop-up menu when **SET** is selected in the Keywords panel. The figure below shows a typical set main control panel, in this case for **SET_PART** definitions, but all are the same.

Commands have their standard meanings as described in section 5.1.1.





CREATE Creating a new set

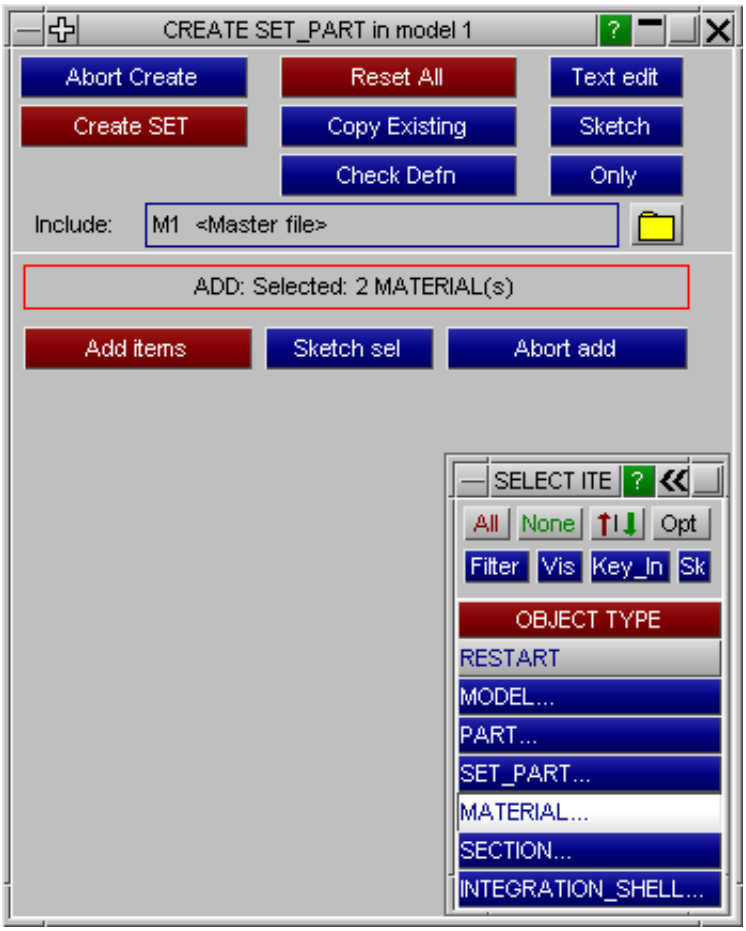


This figure shows the initial empty set creation panel.
Items, here **PARTs**, but the same applies to all valid set types, can be added to or removed from the set using:

ADD...	Inserts items into the set.
REMOVE...	Removes them from the set.
EMPTY...	Completely empty the set of all its contents.

Items are added or removed using the standard selection menu, as shown in figure below.

The set can also be converted to different formats by using the **Convert to** radio buttons. GENERAL sets have a different editing panel.



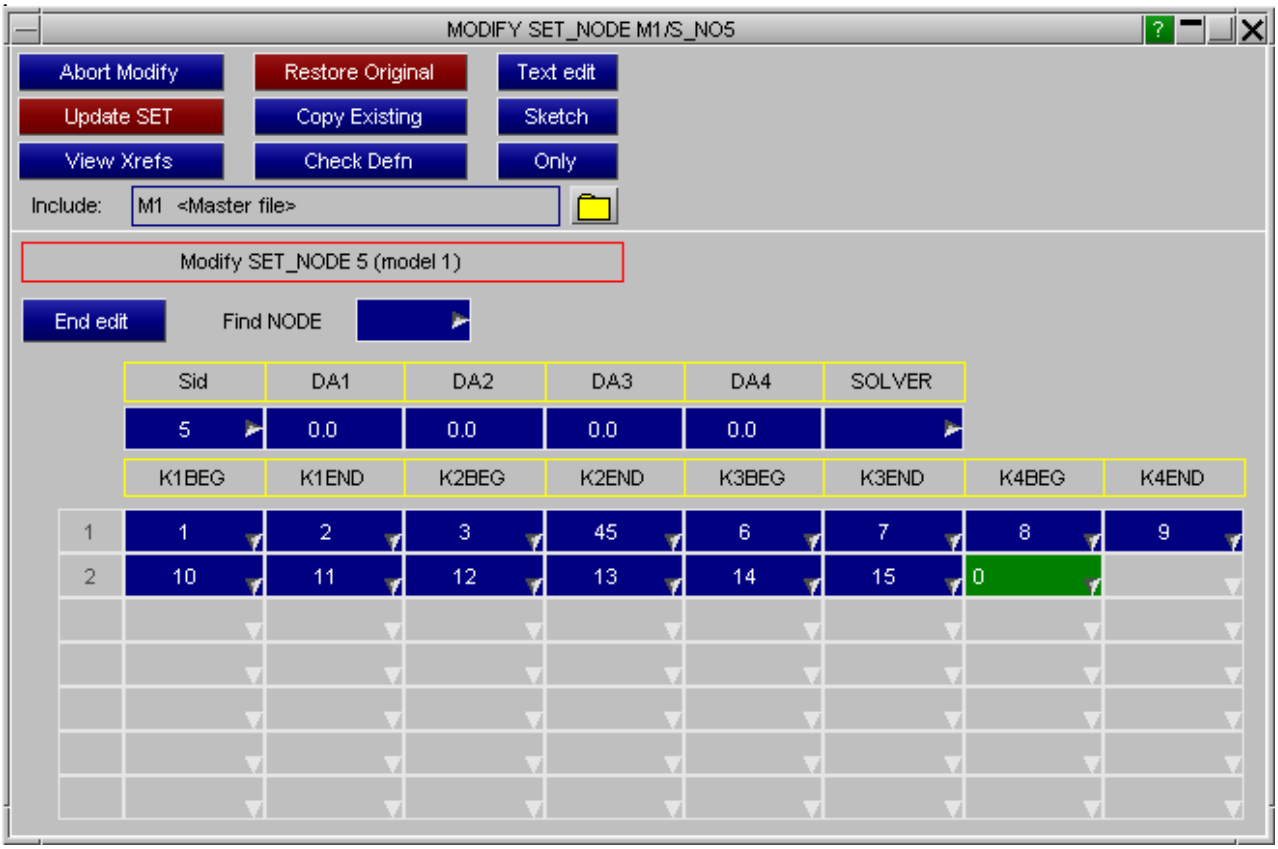
In this example the user is selecting parts by material: he has selected two materials and **ADD_ITEMS** will load the parts which reference these into the set.
Only items of the correct type, here parts, will be selected, therefore it is safe to select a super set of the objects required.

For example to load all parts in a model into a set you could just select the whole model.

It is legal to select items that are already in the set: they will not be duplicated as the addition operation performs a logical (inclusive) OR between the incoming items and the existing set contents.

Item removal operates in exactly the same way, except in reverse.

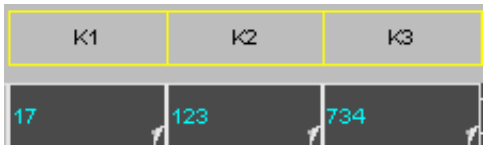
Once the set contains something you can use **VIEW/EDIT** to view the detailed contents of the set



This example shows the **VIEW/EDIT** panel for a set of nodes.

The popup options against each entry may be used to view details of that item, and different labels can be typed in to change the set's contents.

Defining "Latent" items in a set.



If a "latent" (referenced, but not yet defined) item is included in a set, its colour in the editing table changes to blue text on a dark background to warn you, as shown here.

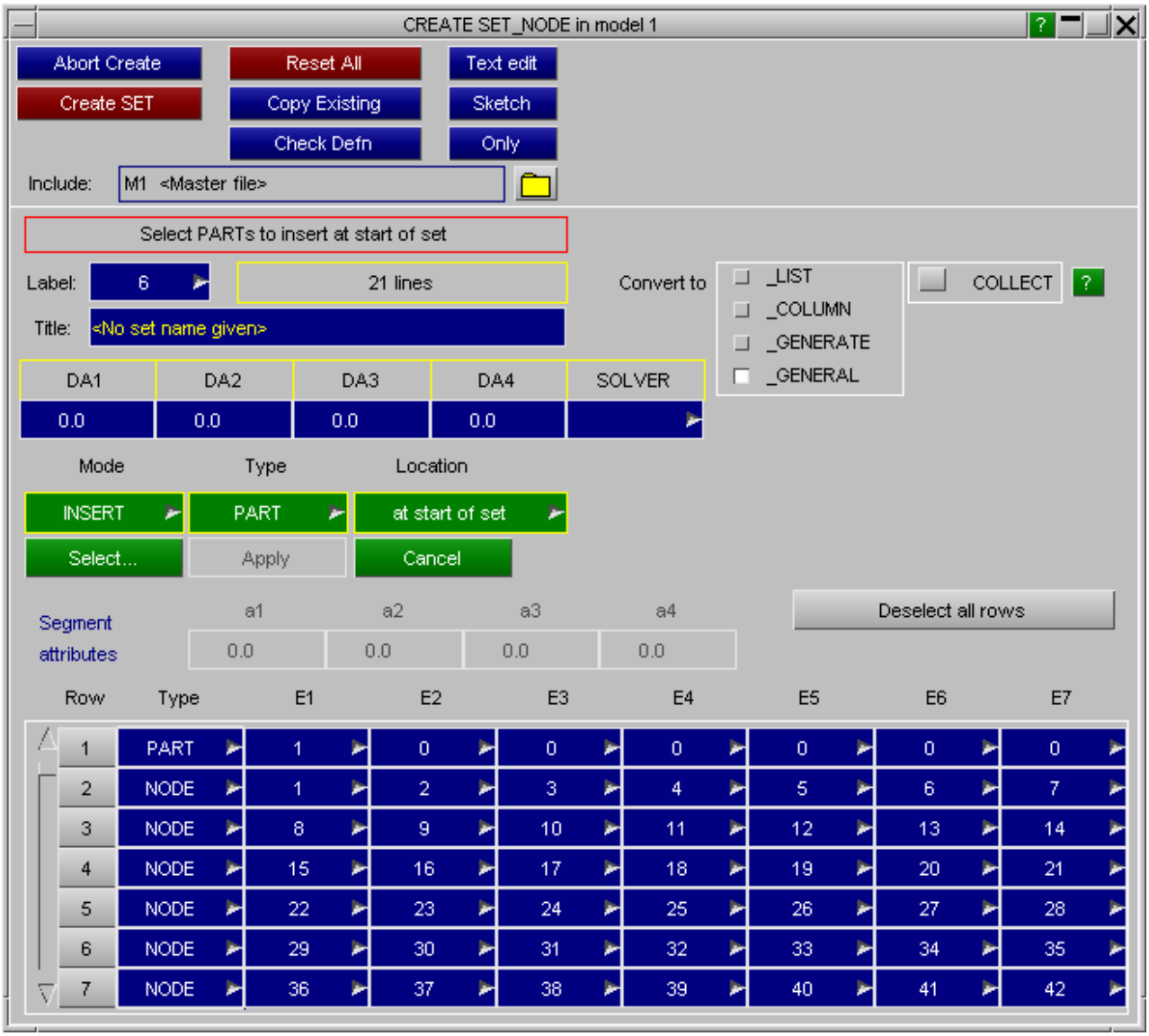
Including latent items in sets is legal, although they will show as an error when the set is checked. You must deal with this before running the analysis: by deleting them explicitly from the set, or by performing a **CLEANUP_UNUSED** operation on the model.

Defining set type and default parameters

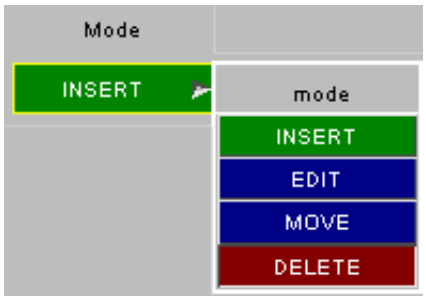
Sets may be of **_LIST** (default), **_COLUMN** ,**_GENERATE,** **_GENERAL** or **_ADD** types, default parameters may be defined, and individual parameters given in **_COLUMN** cases. These options are described in section SET_OPTIONS below.

Creating/editing _GENERAL sets

In LS-DYNA 960 _GENERAL sets have been introduced. These allow flexibility in how the set is defined. For example in a node set they can be used to add all nodes from part 10 and then remove all nodes inside box 1. As the way in which _GENERAL sets are defined is very different a separate panel is used for creating/editing them. The figure below shows a set with several rows already defined in it. Any of the existing rows in the set can be edited by using the usual popups and object menus in PRIMER or by typing in new labels for the items.



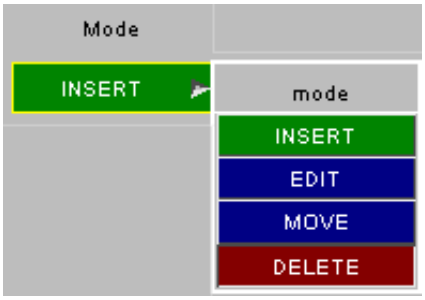
There are also 4 main functions that allow the user to insert, edit, move or delete rows in the set. The modes are selected by using the Mode popup.



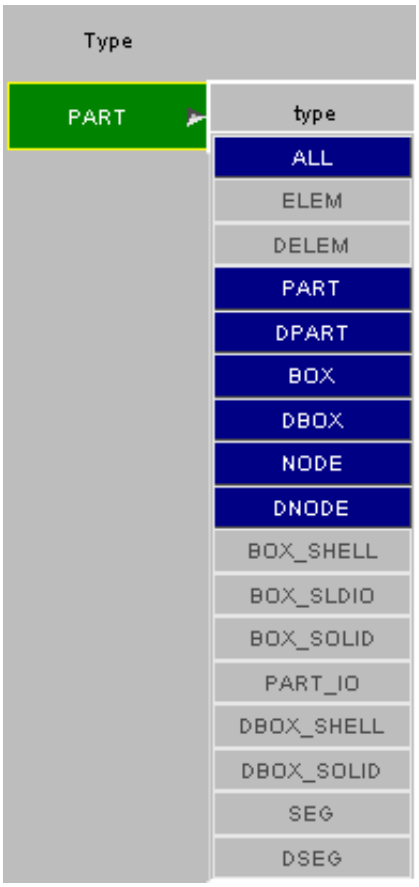
Inserting rows into a _GENERAL set

Example: inserting PARTS 4,5 and 6 to the set after row 4

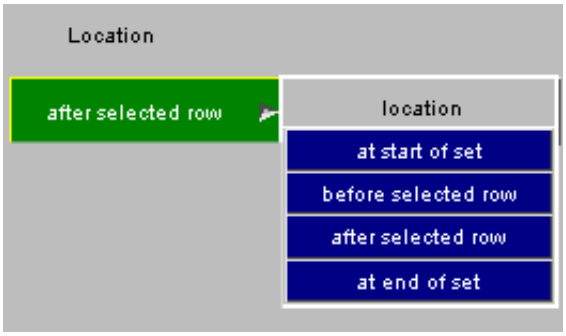
Make sure **INSERT** mode is selected by using the **Mode** popup.



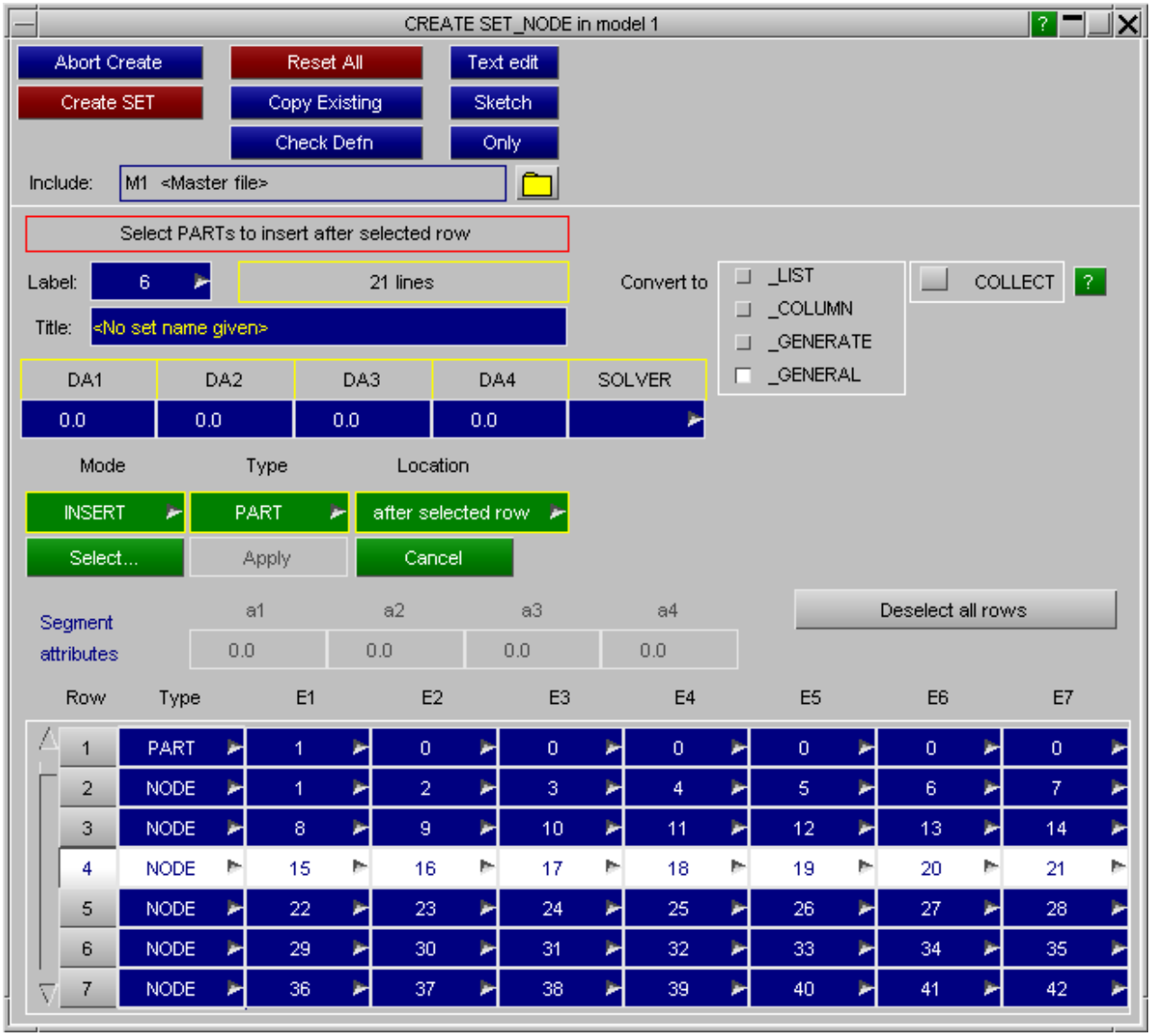
Select **PART** from the **Type** popup.



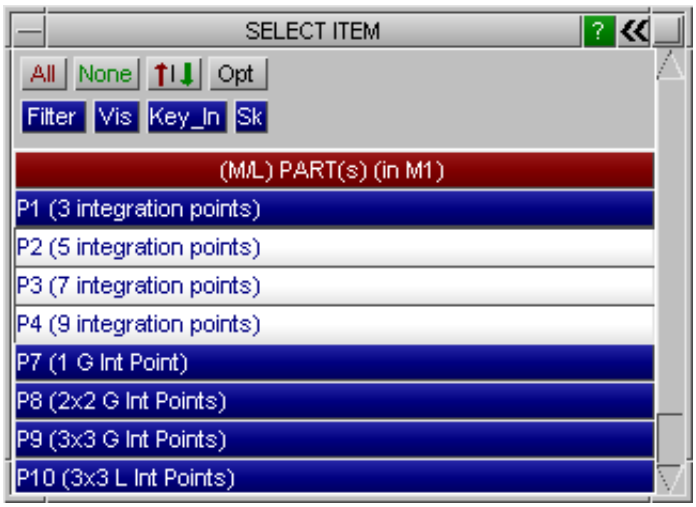
Choose **after selected row** from the **Location** popup



We want to insert the parts after row 4 so select row 4.



Press the **Select...** button and select the parts to add by either picking them or using the object menu.



Press **Apply** to add the parts to the set.

1	PART	1	0	0	0	0	0	0
2	NODE	1	2	3	4	5	6	7
3	NODE	8	9	10	11	12	13	14
4	NODE	15	16	17	18	19	20	21
5	PART	2	3	4	0	0	0	0
6	NODE	22	23	24	25	26	27	28
7	NODE	29	30	31	32	33	34	35

The parts have been added on row 5. All of the higher rows have been shifted up by one.

This method can be used to add a line (or more than one line if needed) to the set at:

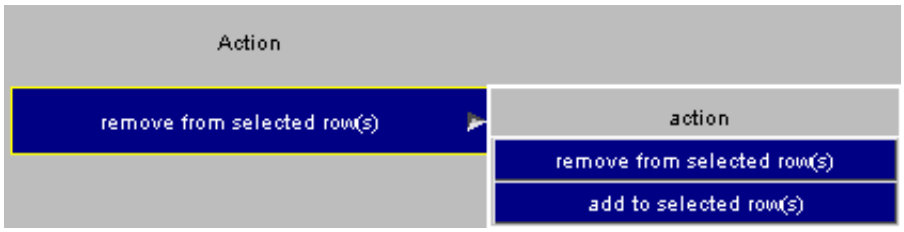
- the top (first row) of the set
- the bottom (last row) of the set
- before a selected row
- after a selected row

Editing rows in a _GENERAL set

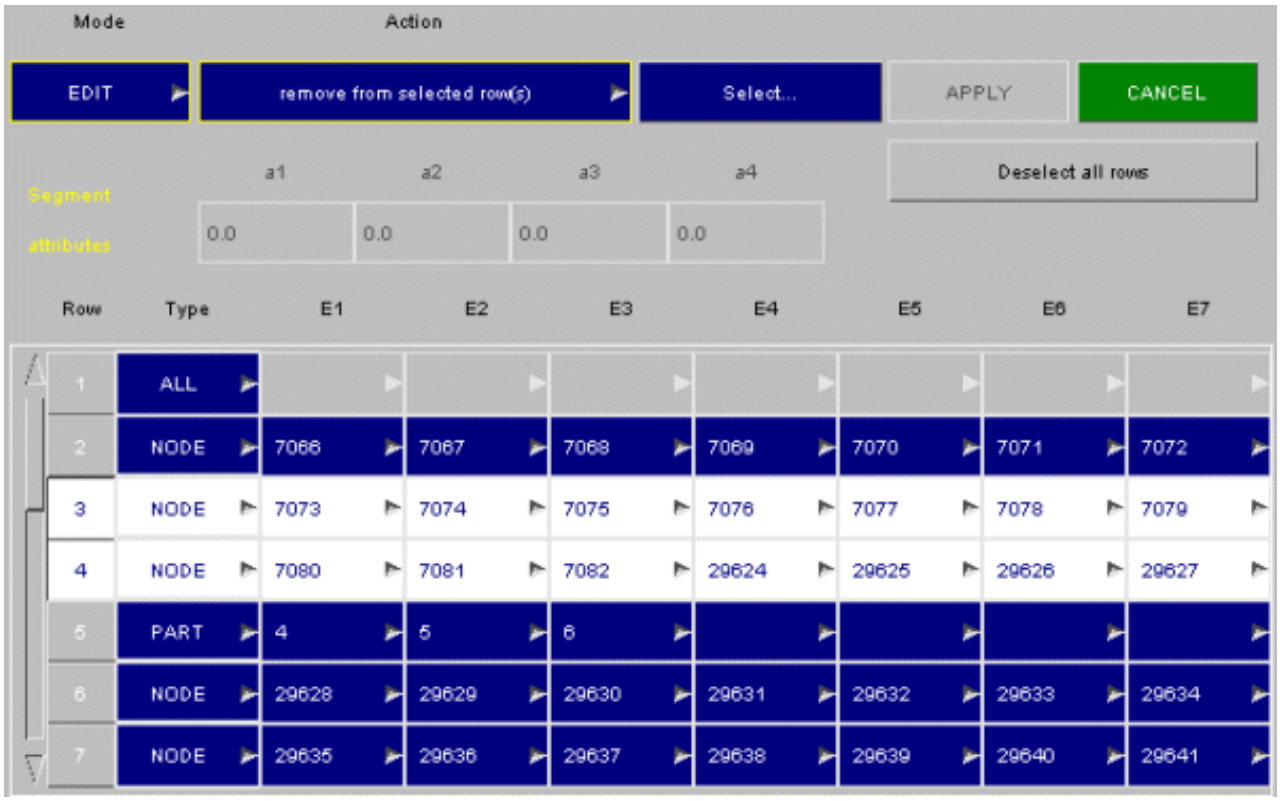
Example: removing BOXES from an existing row

Make sure **EDIT** mode is selected by using the **Mode** popup.

Select **remove from selected row(s)** from the **Action** popup.



Select the rows that we want to edit (in this example; rows 3 and 4).



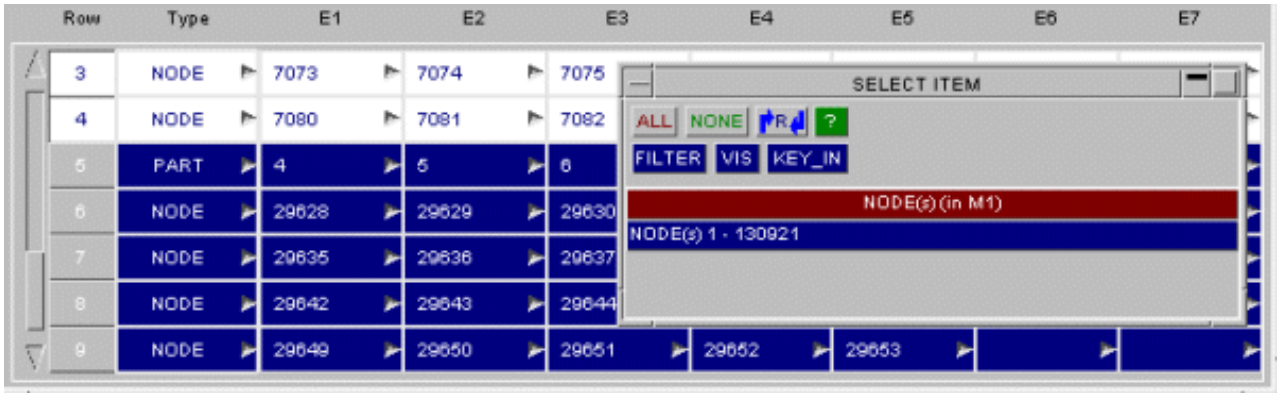
When selecting more than one row they must be:

- Of the same type. E.g. BOX, PART...
- Adjacent rows. E.g. selecting rows 3 and 4 is OK, selecting rows 3 and 7 is not.

To select a range of rows:

- Select the first row by left clicking
- Move to the last row and shift, left click to select the range

Press the **Select...** button and select the boxes to remove from the rows by either picking them or using the object menu.



Press **Apply** to remove the boxes from the selected rows in the set.

Row	Type	E1	E2	E3	E4	E5	E6	E7
2	NODE	7066	7067	7068	7069	7070	7071	7072
3	NODE	7075	7076	7082	29624			
4	PART	4	5	6				
5	NODE	29628	29629	29630	29631	29632	29633	29634
6	NODE	29635	29636	29637	29638	29639	29640	29641
7	NODE	29642	29643	29644	29645	29646	29647	29648
8	NODE	29649	29650	29651	29652	29653		

The boxes have been removed from the rows. As the entry now only requires one row (there are now only 4 boxes) the empty row is deleted and the higher rows have been shifted up by one.

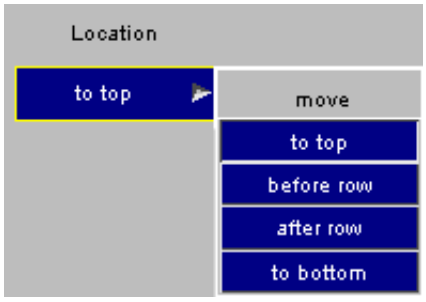
This method can also be used to add extra items to an existing line instead of removing items by choosing **add to selected row(s)** from the action popup. Extra rows will automatically be added if they are needed.

Moving rows in a _GENERAL set

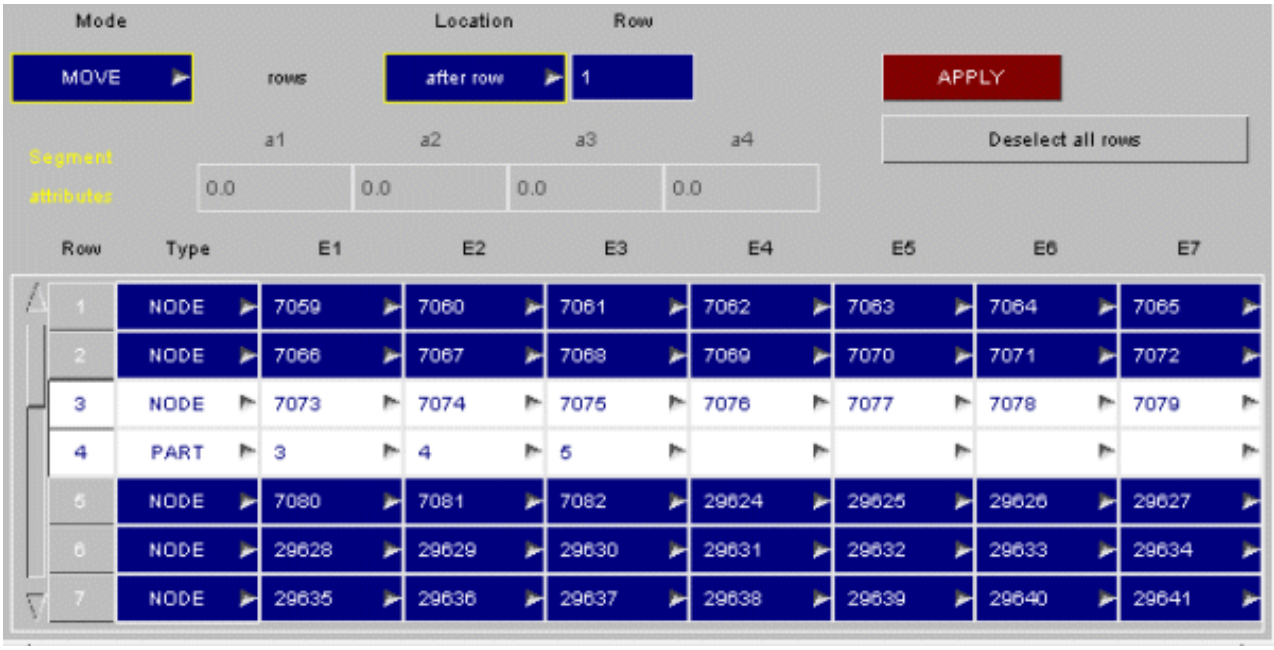
Example: Moving selected rows to be after row 1

Make sure **MOVE** mode is selected by using the **Mode** popup.

Select **after row** from the **Location** popup.



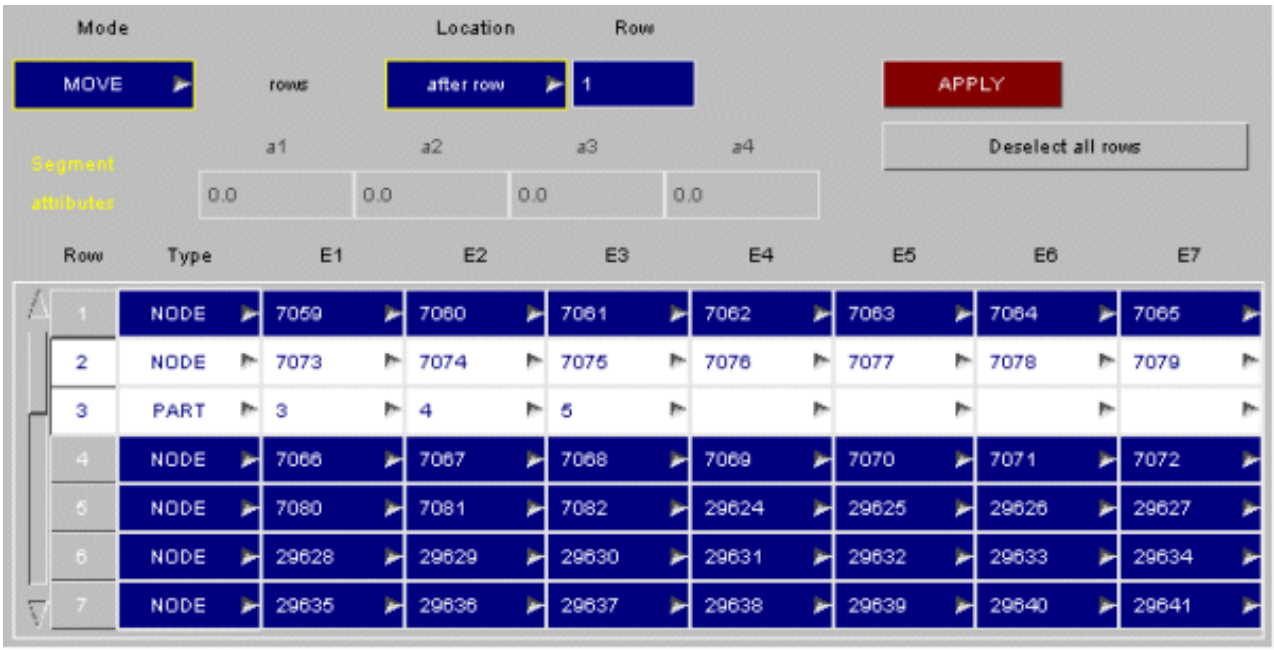
Select the rows that we want to move (in this example; rows 4 and 5) and type in row 1 for the **Location Row**.



When selecting more than one row they must be adjacent rows. E.g. selecting rows 3 and 4 is OK, selecting rows 3 and 7 is not.

To select a range of rows:
- Select the first row by left clicking
- Move to the last row and shift, left click to select the range

Press **Apply** to move the rows in the set to be after row 1.



The rows have been moved.

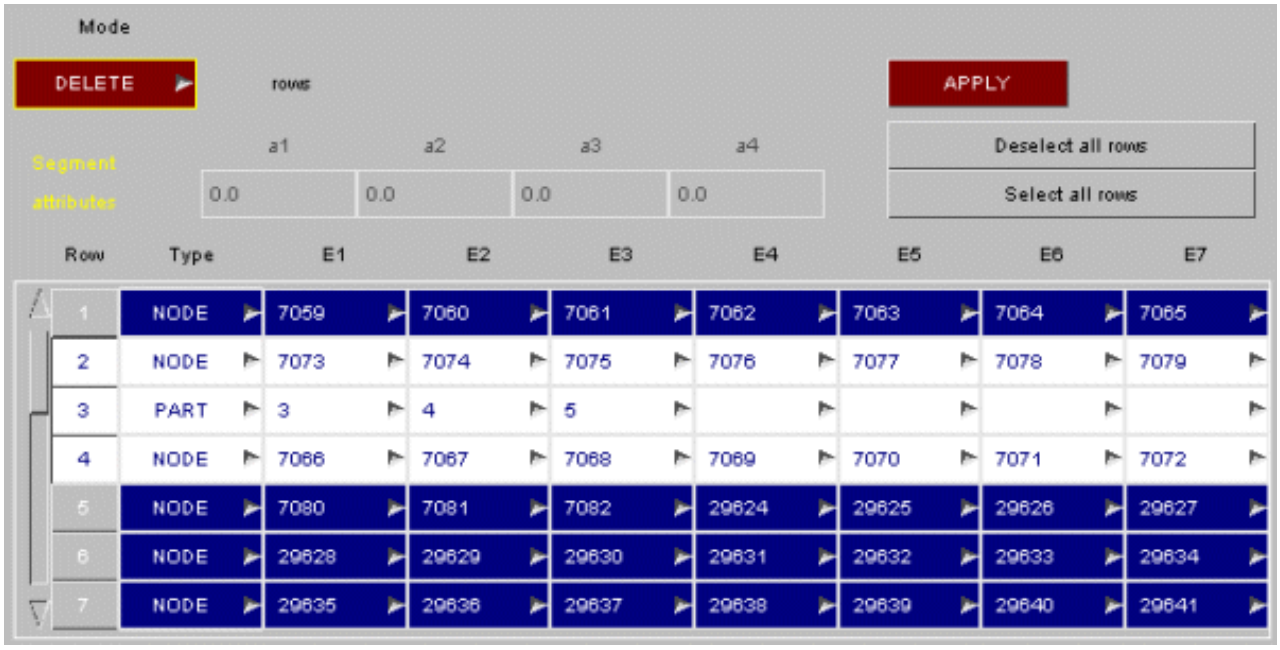
This method can be used to move a line (or more than one line if needed) to:
- the top (first row) of the set
- the bottom (last row) of the set
- before a selected row
- after a selected row

Deleting rows from a _GENERAL set

Example: Deleting selected rows

Make sure **DELETE** mode is selected by using the **Mode** popup.

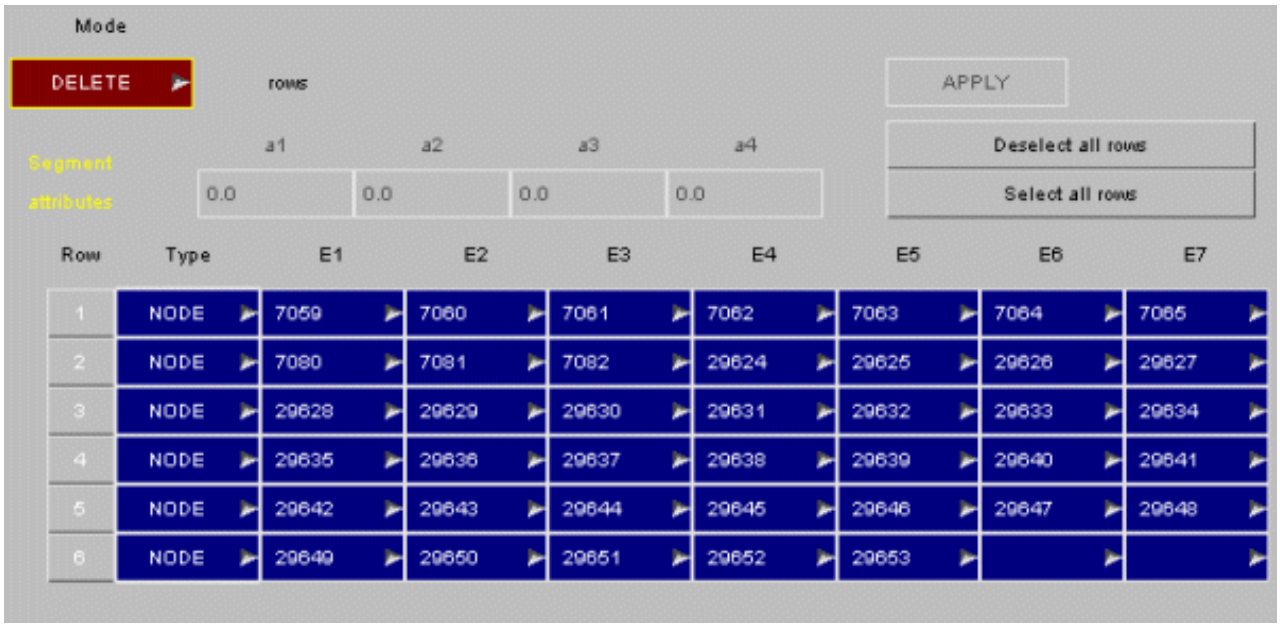
Select the rows that we want to delete (in this example; rows 2, 3and 4)..



Any rows can be selected. They do not need to be adjacent.

To select a range of rows:
- Select the first row by left clicking
- Move to the last row and shift, left click to select the range

Press **Apply** to delete the rows.



The rows have been deleted.

_ADD option

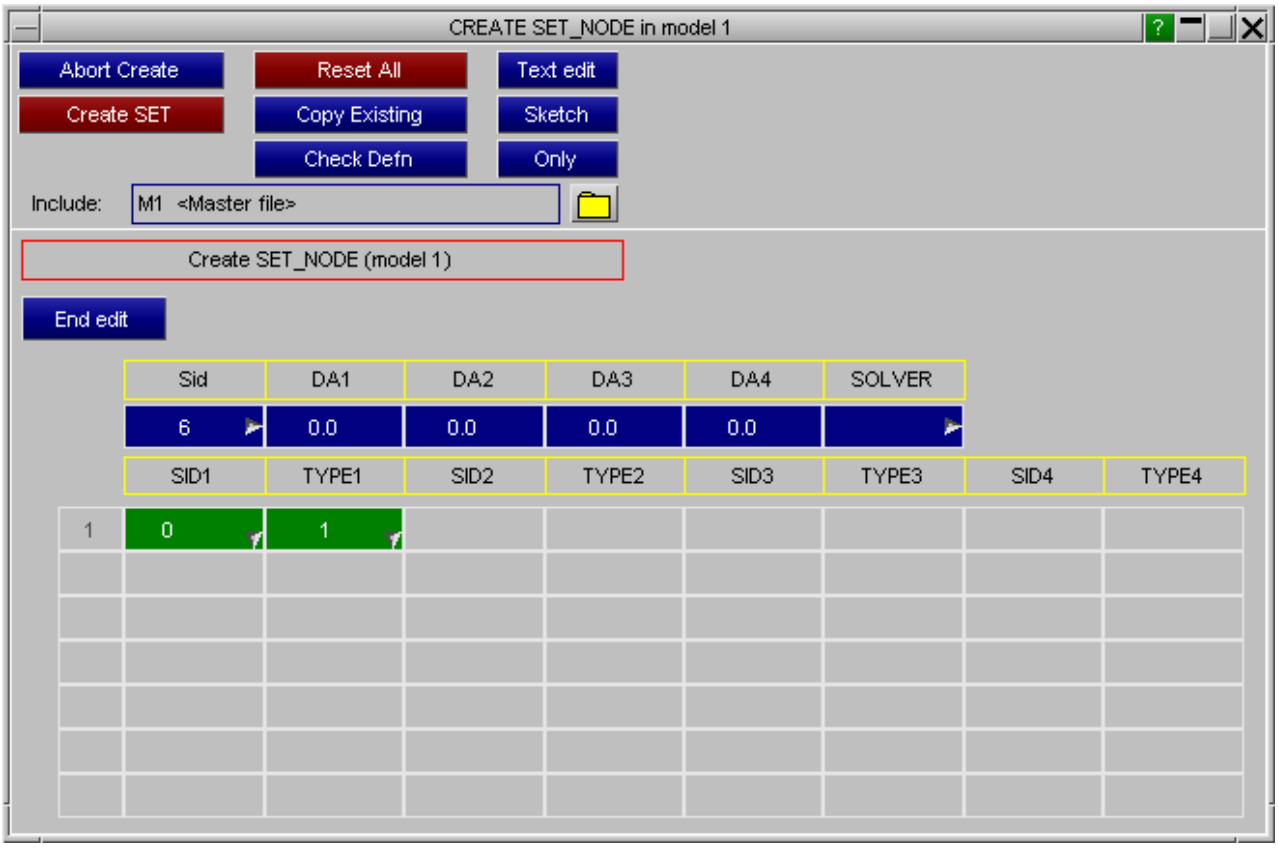
LS971 introduces the _ADD option for sets of type BEAM, DISCRETE, NODE, PART, SHELL and SOLID. Keyword *SET_PART_ADD is viable in LS970v6763 onwards. The _ADD option allows creation of a *SET that is itself composed of *SET definitions. For example a *SET_PART_ADD would list any number of other *SET_PART definitions. These sets may be created, edited, drawn, etc in exactly the same way as any other set types, with the limitation that LS-DYNA limits them implicitly to "list" syntax.

It is not clear from the LS-DYNA documentation whether or not these definitions can be nested to many levels (ie a *SET_PART_ADD contains sets that are themselves of type *SET_PART_ADD). So on the assumption that this is possible PRIMER permits them to be nested to any level, and does not treat this as an error.

PRIMER checks for recursive definitions (ie a *SET_PART_ADD contains a reference to itself, possibly several levels lower down) and flags these as errors. PRIMER handles such definitions by ignoring the 2nd definition, and anything below it.

A *SET_PART_ADD definition may easily contain multiple references to an underlying *PART. Again LS-DYNA does not state whether or not this is legal, although from past experience it probably is, so PRIMER also assumes that this is acceptable practice and does not mark this as an error. When using the set for internal purposes, for example graphics, PRIMER detects duplicate PART definitions and simply ignores the 2nd and subsequent ones.

For *SET_NODE_ADD the _ADVANCED option is available. This allows addition of other *SET types. NODE, SHELL, BEAM, SOLID, SEGMENT, DISCRETE and THICK SHELL sets can all be added. The keyword format for this option is a set ID followed by a type. The types available are NODE (type 1), SHELL (type 2), BEAM (type 3), SOLID (type 4), SEGMENT (type 5), DISCRETE (type 6)and THICK SHELL (type 7). Pressing **VIEW/EDIT** allows creation and modification of these keywords.



If a deck containing *SET_..._ADD is written out in a format pre-dating LS971R2 (LS970v6763 for *SET_PART_ADD), PRIMER will decompose the definition to a conventional *SET_... containing all underlying entities.

_COLLECT option

LS971 release 5 introduces the **_COLLECT** suffix for sets. This allows multiple definitions of a set to exist, all using the same label. These may be in the same file or (more probably) in different include files. During the LS-DYNA analysis

the contents of all the set definitions are "collected" together to form a single set.

This presents problems for PRIMER because it must handle the conflicting requirements of:
- Treating the set as a single entity for graphics, checking and when referenced from other cards.
- Maintaining the individual set definitions as separate for the purposes of editing, and for ultimate output to the keyword file.

PRIMER handles the problem as follows:

1. When a **`*SET_xxx_COLLECT`** definition of label N is first encountered the following internal definitions are created

   - A "parent" set definition which has the label N, but has no contents.
   - A "child" definition that holds the contents of this portion of set N, and this is associated with the "parent".

2. For each subsequent **`*SET_xxx_COLLECT`** definition of label N that is found a new "child" set definition is created, and associated with "parent" set N

For example if a model contains three definitions of *SET_SHELL_COLLECT all using label 10 then PRIMER will store:

Parent *SET_SHELL 10		This is the parent set, and does not contain any elements
	Child definition #1	This contains the elements of the 1st set definition
	Child definition #2	The contains the elements of the 2nd set definition
	Child definition #3	The contains the elements of the 3nd set definition

Creating a *SET_xxx_COLLECT definition

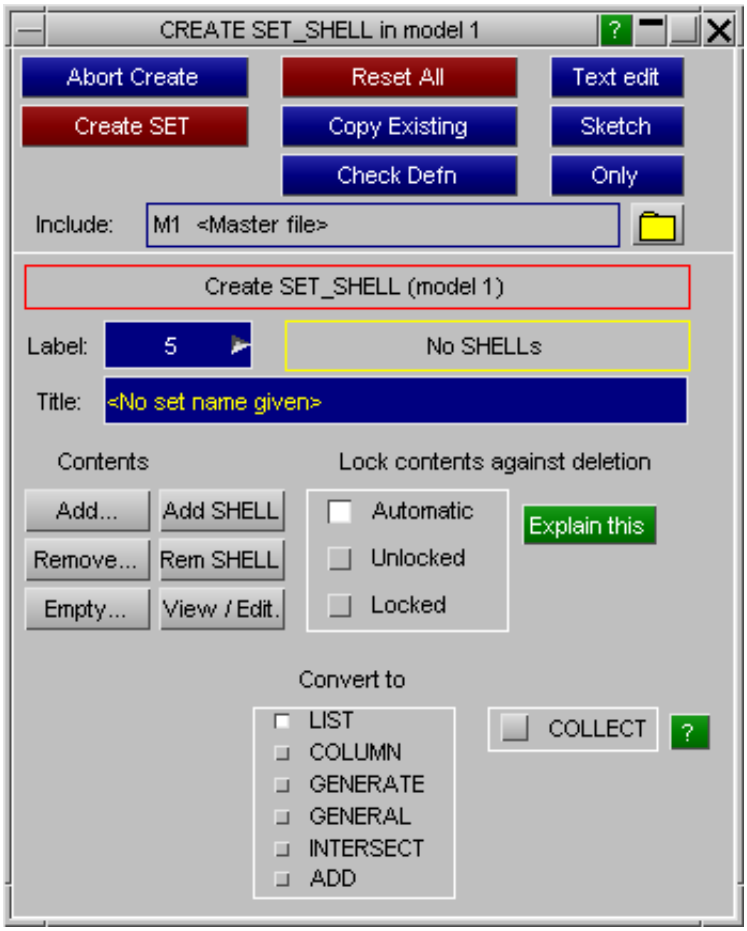
For the first set in the collection:
- Create the set in the normal way, giving it a normal label
- Tick the **COLLECT** box to designate it as the start of a collection
- **CREATE** it normally.

PRIMER will warn you about what it is doing, and will create both the parent set and its first child. The parent set will automatically be given the title "All collected sets", but you can change this is you wish.

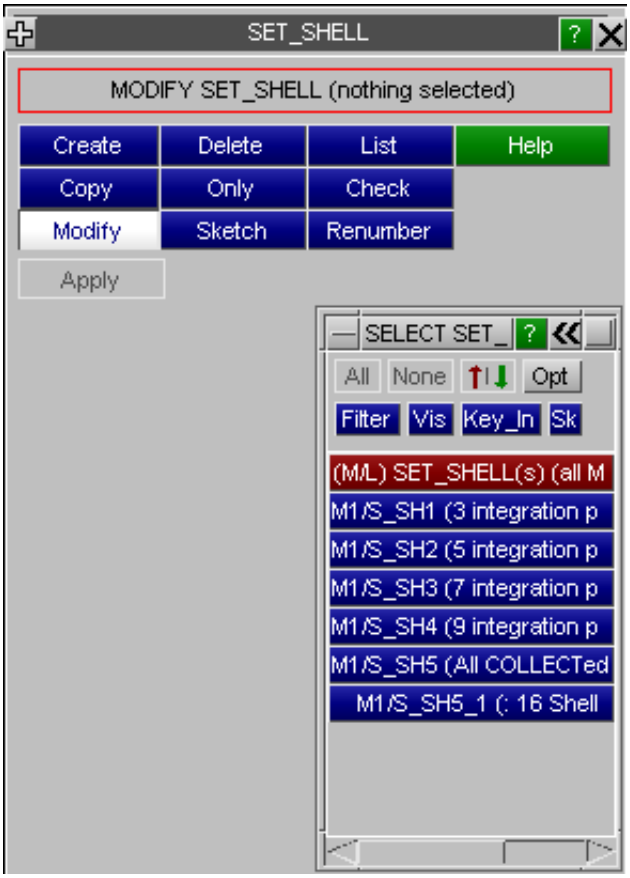
For the 2nd and subsequent sets in the collection:
- Create the set in the normal way
- Tick the **COLLECT** box as above
- Type in, or select, the label of the set used above.

PRIMER will automatically associate this set with the parent above, and this set definition will become its next child. Each child set may be given a separate title, and this is recommended as it will help to identify which is which.

There are no rules inside PRIMER about which include files the various sets "live" in: any set in the collection, including the parent, may be in any include file.



Editing a *SET_xxx_COLLECT definition



Firstly you must decide whether you want to edit the "parent" set, or one of its "children".

The menu from which you make this selection makes this parent/child relationship clear by indenting the children, and giving them label suffices _1, _2, etc as shown in this example.

Editing a ***child*** set is performed in exactly the same way as editing a normal set: you can add or remove contents, change its type, and so on, with two exceptions:

1. You can only change its label to that of a different *SET_xxx_COLLECT definition.

   PRIMER will not allow you to relabel a child set unless the new label is that of an existing (different) *SET_xxx_COLLECT definition.

2. You can remove it from the collection by unticking its **COLLECT** box.

   This will give it a new label, remove it from its parent, and turn it into a normal set. Note that is will not be referenced by anything else in the model as all existing references will be to its ex-parent.

Editing a ***parent*** *SET_xxx_COLLECT definition

The only operations you can perform on the parent set are:

- Change its label. This relabels the whole collection meaning that when the keyword deck is written out all child definitions will use this new label.

- Change its title. This title is only used within PRIMER, as the parent definition is not written out to the keyword file, so changing it will be of limited value.

- **Edit child** opens an object menu to select a child set for editing.

- **Merge** the collection into a single, ordinary (non _COLLECT) set definition.

  This operation modifies the contents of this parent set definition to be an ordinary set, using _LIST format, that combines the entire contents of all its child sets. It then deletes all the child sets and converts this parent set into an ordinary (non _COLLECT) set.

  Once updated this merge operation is not reversible.

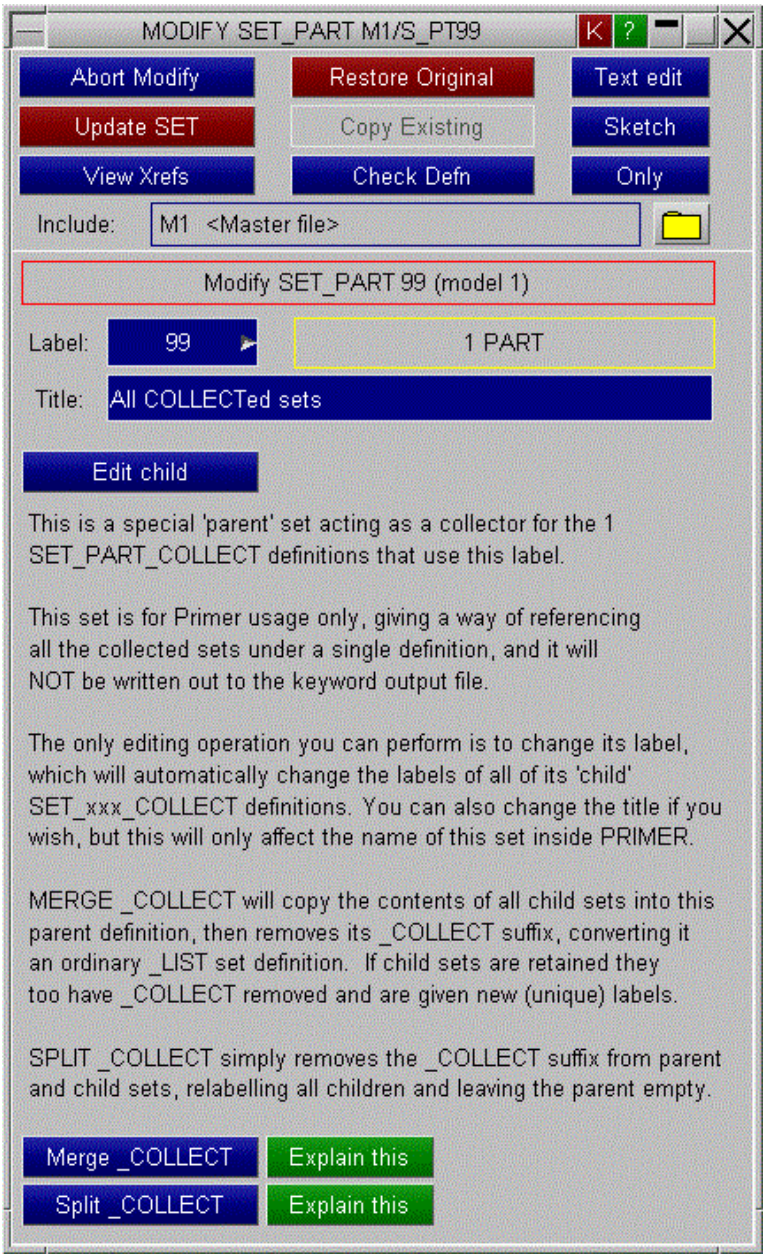
- **Split** the collection in N separate, ordinary (non _COLLECT) sets.

  Each child set is given a new (individual) label, and its _COLLECT status is turned off. This leaves it as an ordinary set, but note that it will not be referenced by anything else.

  The parent set remains, using its existing label and with its _COLLECT status also turned off. However it will have no contents. All references to the original collection will still refer to this set.

  Following a Split it will be necessary for you to sort out manually the consequences of having an empty set that is (possibly) referred to elsewhere in the model, and a series of child sets that are populated but not referenced.

  Once updated this split operation is not reversible.



Referencing and using a *SET_xxx_COLLECT definition elsewhere in PRIMER

From the point of view of the rest of PRIMER a _COLLECT set is seen as a single item, just like a normal set, and it will be referred to by the label of the parent definition. In other words no special actions need to be taken when dealing with a *SET_xxx_COLLECT definition, and it can be dealt with just like any other set. For example sketching the parent will automatically sketch all its children.

Object menus will only show the parent set, unless they are in a context in which it would make sense to offer selection of a child.

Changing an existing set to *SET_xxx_COLLECT

As well as creating a *SET_xxx_COLLECT definition from scratch you can edit an existing set to make it a _COLLECT one. PRIMER handles this as follows:

- By default the set retains its existing label, and it becomes the first set in a new collection using that label.
- You can also change its label to that of an existing collection, whereupon it will become a child of that collection.

This raises the question of "what happens to references to this set on other keywords?" The answer is:

- If this is the start of a new collection, using the same (or a fresh) label, then all references will be to that label.
- If the set is made part of an existing collection then all references to the set are ***changed to references to that collection***.

Really these two mean that same thing: references to a set that becomes part of a collection will become references to that collection.

Keyword output of *SET_xxx_COLLECT definitions

During keyword output PRIMER will write out each child set definition in its appropriate include file, using the _COLLECT suffix and the label of its parent.

The parent definition used inside PRIMER is not written out and will be lost when the model is deleted or PRIMER exits. It will be recreated automatically when the keyword deck is reread, or when _COLLECT sets are created interactively.

_INTERSECT option

LS971R6 introduces the _INTERSECT option for sets. The _INTERSECT option allows creation of a *SET that is itself composed of the intersection of other *SET definitions. These sets may be created, edited, drawn, etc in exactly the same way as any other set types, with the limitation that LS-DYNA limits them implicitly to "list" syntax.

It is not clear from the LS-DYNA documentation whether or not these definitions can be nested to many levels (ie a *SET_SHELL_INTERSECT contains sets that are themselves of type *SET_SHELL_INTERSECT) and whether they can be used in conjunction with other set types (e.g. *SET_SHELL_INTERSECT containing *SET_SEHLL_ADD sets etc). So on the assumption that this is possible PRIMER permits them to be nested to any level, allows somplex set definitions, and does not treat this as an error.

If a deck containing *SET_..._INTERSECT is written out in a format pre-dating LS971R6 the *SET_..._INTERSECT cards will be omitted.

COPY Copying sets.

The selected sets are copied. The **RECURSIVE_COPY** flag has an important influence on this:

- When **OFF** Only the set itself is copied. A new set referencing all the items in the original set is created.
- When **ON** All the items "owned" by the set are recursively marked for copying. This can select a considerable number of items: use with care.

Generally recursive copying will only be sensible for **`SET_SEGMENT`** definitions, refer to the special notes on segment sets below.

MODIFY Modifying sets

The set modification panel is identical to the **CREATE** one, except that it will already be populated when entered, and usage is exactly the same.

DELETE Deleting sets

The selected sets, and possibly their contents, are marked for deletion. What is actually deleted, and whether deletion of the set actually takes place, depends on the following switches:

DELETE_RECURSIVE Whether or not items "owned" by sets are marked for deletion.
- When **OFF** Only the set itself is so marked, its contents are not affected.
- When **ON** The contents, and anything they "own" are also marked for deletion.

REMOVE_FROM_SETS Whether flagged items can be removed from other sets.
- When **OFF** Items (marked recursively) will not be deleted if they are referred to by other sets. (But that won't stop this set being deleted.)
- When **ON** Items will be removed from any other sets in which they are referred to.

Deletion can only take place if the items referred to are not referenced elsewhere in the model. So deleting a set may fail if it is still in use somewhere, even though its contents may have been deleted leaving it empty! This will be picked up by global checking, and can be corrected with **CLEANUP_UNUSED** which gives the option of eliminating empty sets.

SKETCH Sketching sets

The selected sets will be sketched on the current graphics image. Sketching is performed by drawing the constituents of a set.

ONLY sets

The selected sets are displayed and everything else is blanked. An option is provided to display either the selected sets or the contents of those sets in the **ONLY** mode.

LIST Listing set summaries

A summary of the selected sets is listed to the screen.

CHECK Check sets for errors

The selected sets are processed through the standard checking routines, and the results summarised to the screen.

RENUMBER Renumber set labels

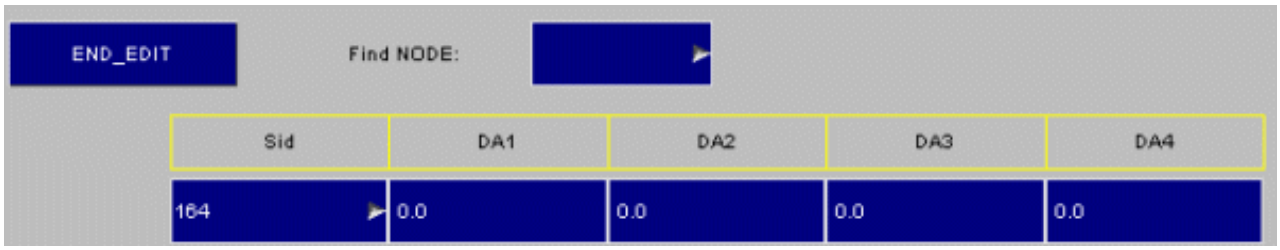
The standard item renumbering panel is mapped for the chosen model, and any or all labels can be changed. To change a single set label it may be easier just to **MODIFY** it.

SET_DEFAULTS Defining the default parameters for sets.

Set types **_NODE, _PART, _SEGMENT** and **_SHELL** can all be used in contexts where additional information may be required to complete input. Examples are:

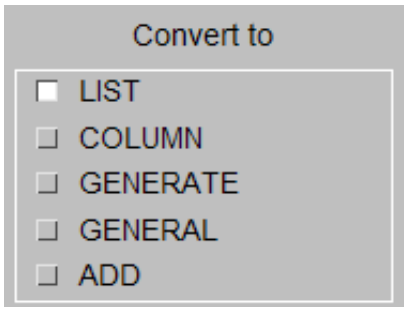
- In ***CONTACT_TIEBREAK** definitions you can define tiebreak failure parameters for a whole set, or on a per node/per element basis.
- In many other (see the LS-DYNA user manual) ***CONTACT_**... definitions it is possible to vary friction across a surface by defining individual friction parameters for segments or shells.
- In ***CONSTRAINED_TIE-BREAK** definitions individual failure parameters can be defined for each node.

In all cases default parameters may be defined for the whole set by filling in the **DA1 ... DA4** fields in the **VIEW/EDIT** panel as shown below.



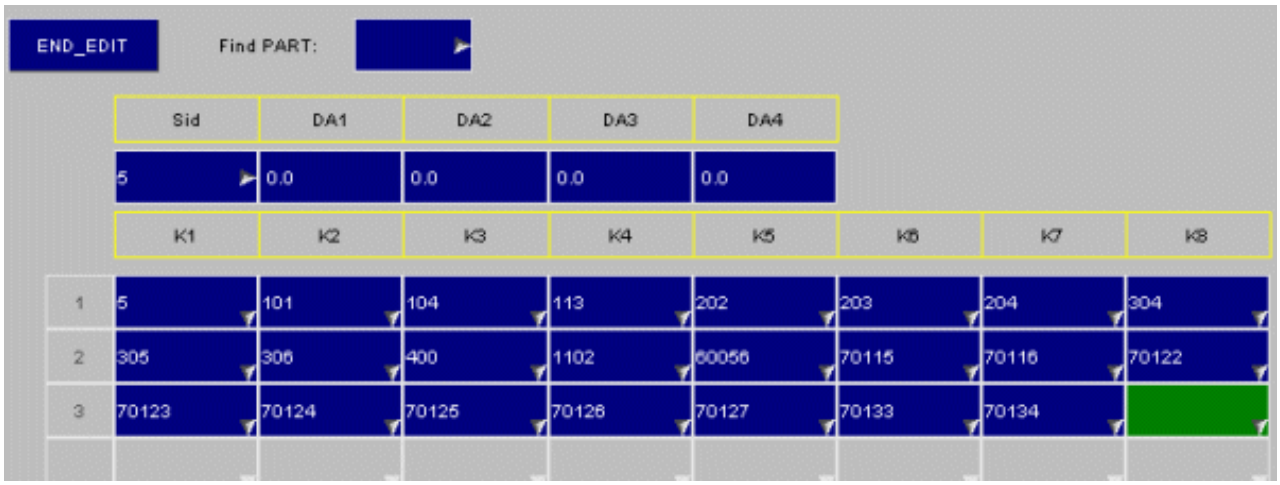
These will apply to every set entry except where individual entries have been made in **_COLUMN** mode.

SET_OPTIONS Using the _LIST, _COLUMN _GENERATE, _GENERAL and _ADD sub-keywords to change set layout.



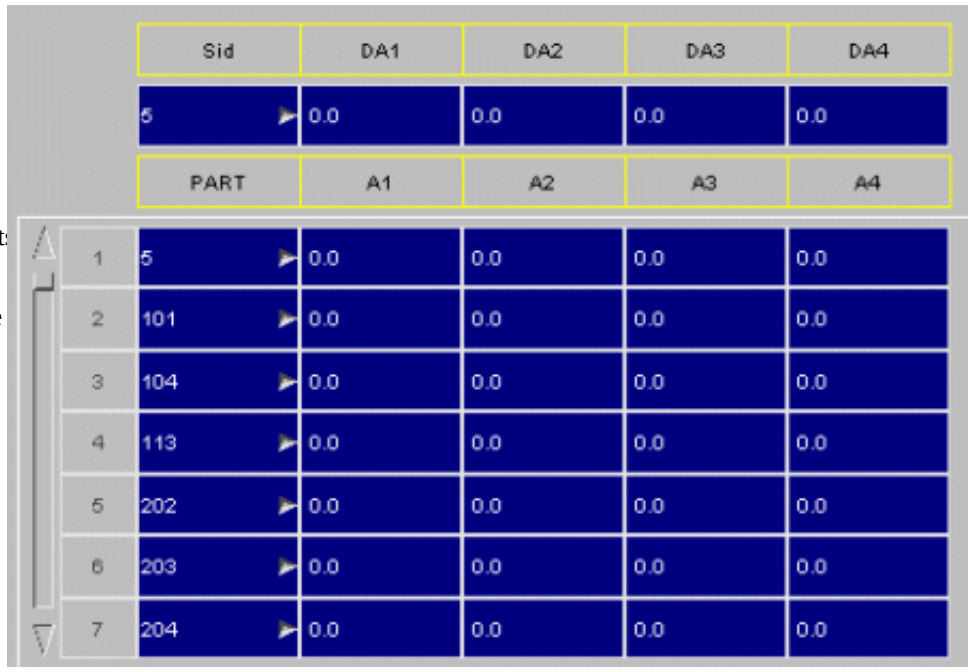
The simplest, default set layout is **SET_LIST** in which a simple list of constituent items is given 8 to a line, and in which no optional data is given for any item.
This example shows a typical **_LIST** layout for 23 parts.

Default attributes (**DA1 .. 4**) can be given, but no individual values can be defined.



Changing this definition to **_COLUMN** results in the revised layout shown here. It is now possible to give individual values for each item in the set.

When you change from **_LIST** to **_COLUMN** format in PRIMER the code automatically fills in all the new row entries with zeros, ie the default. You can then overwrite any specific values as required.



	Sid	DA1	DA2	DA3	DA4
	5	0.0	0.0	0.0	0.0
	PART	A1	A2	A3	A4
1	5	0.0	0.0	0.0	0.0
2	101	0.0	0.0	0.0	0.0
3	104	0.0	0.0	0.0	0.0
4	113	0.0	0.0	0.0	0.0
5	202	0.0	0.0	0.0	0.0
6	203	0.0	0.0	0.0	0.0
7	204	0.0	0.0	0.0	0.0

The third method of defining sets is to use **_GENERATE**, in which item labels are given in [<start><end>] pairs.

Converting between _LIST, _COLUMN, _GENERATE, _GENERAL and _ADD set formats

PRIMER will convert between any of these ***except to*** **_GENERATE** and _ADD format.

- When a set is initially converted *to* **_COLUMN** format zeros are inserted for all optional data.
- When it is converted *from* **_COLUMN** format the optional data is still stored, but not displayed or output, and is still available if you revert back to that format. So you can convert to and from **_COLUMN** format without losing information.

Data type and units of "default" and item-specific data

Because a set can be used in so many different contexts PRIMER does not attempt to determine the meaning of default or item-specific data.

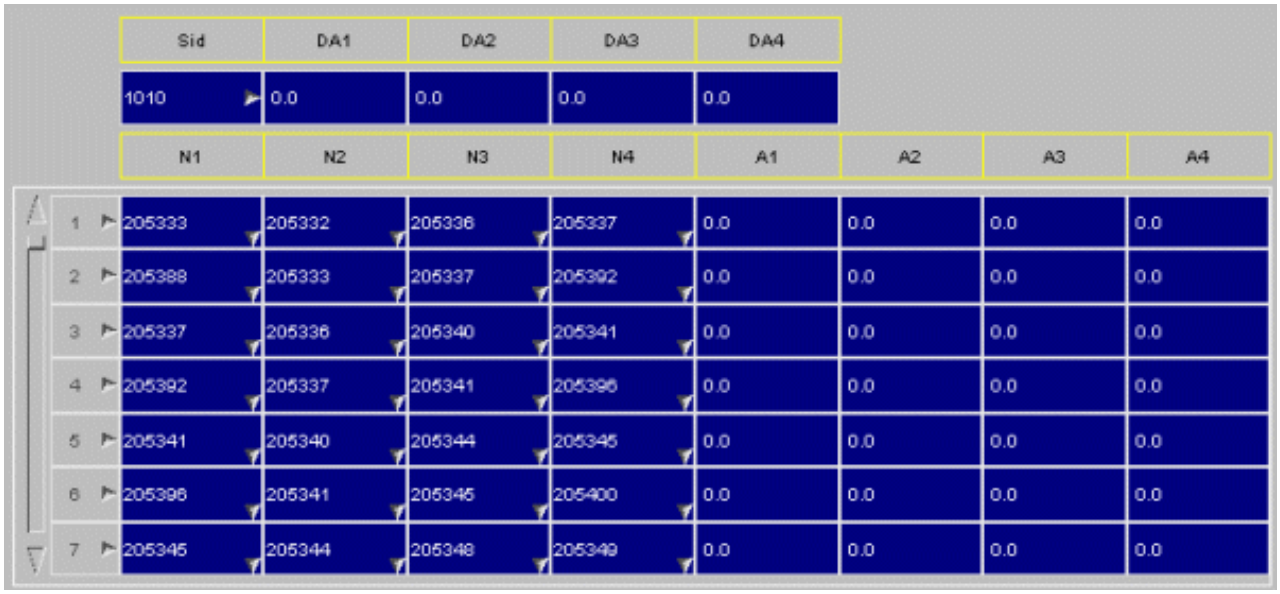
Therefore the data are treated as simple numbers, and no units conversion or checking is applied to them.

SET_SEGMENT The special case of segment sets

Segment sets are a special case for two reasons:

- A "segment" is not a structural item, although it may be drawn to look like one, and has no existence outside the context of its parent set definition (or load etc). It has to lie on top of a shell element, or the face of a solid or thick shell.
- Therefore segment sets actually create their constituent segments, rather than just referring to them, which makes the rules for their definition and deletion different to those of other set types.

As a consequence the layout and operation of the **`SET_SEGMENT`** editing panel is slightly different to that of the others.



Sid	DA1	DA2	DA3	DA4
1010	0.0	0.0	0.0	0.0

	N1	N2	N3	N4	A1	A2	A3	A4
1	205333	205332	205336	205337	0.0	0.0	0.0	0.0
2	205388	205333	205337	205392	0.0	0.0	0.0	0.0
3	205337	205336	205340	205341	0.0	0.0	0.0	0.0
4	205392	205337	205341	205396	0.0	0.0	0.0	0.0
5	205341	205340	205344	205345	0.0	0.0	0.0	0.0
6	205396	205341	205345	205400	0.0	0.0	0.0	0.0
7	205345	205344	205348	205349	0.0	0.0	0.0	0.0

Each segment is defined in terms of 4 nodes (n3 = n4 for a triangle), and the extra item-specific data occupies columns 5 to 8.

There is no choice of set format type.
Segments may be added to a set by duplicating those elsewhere in the model, or created using the COAT function.

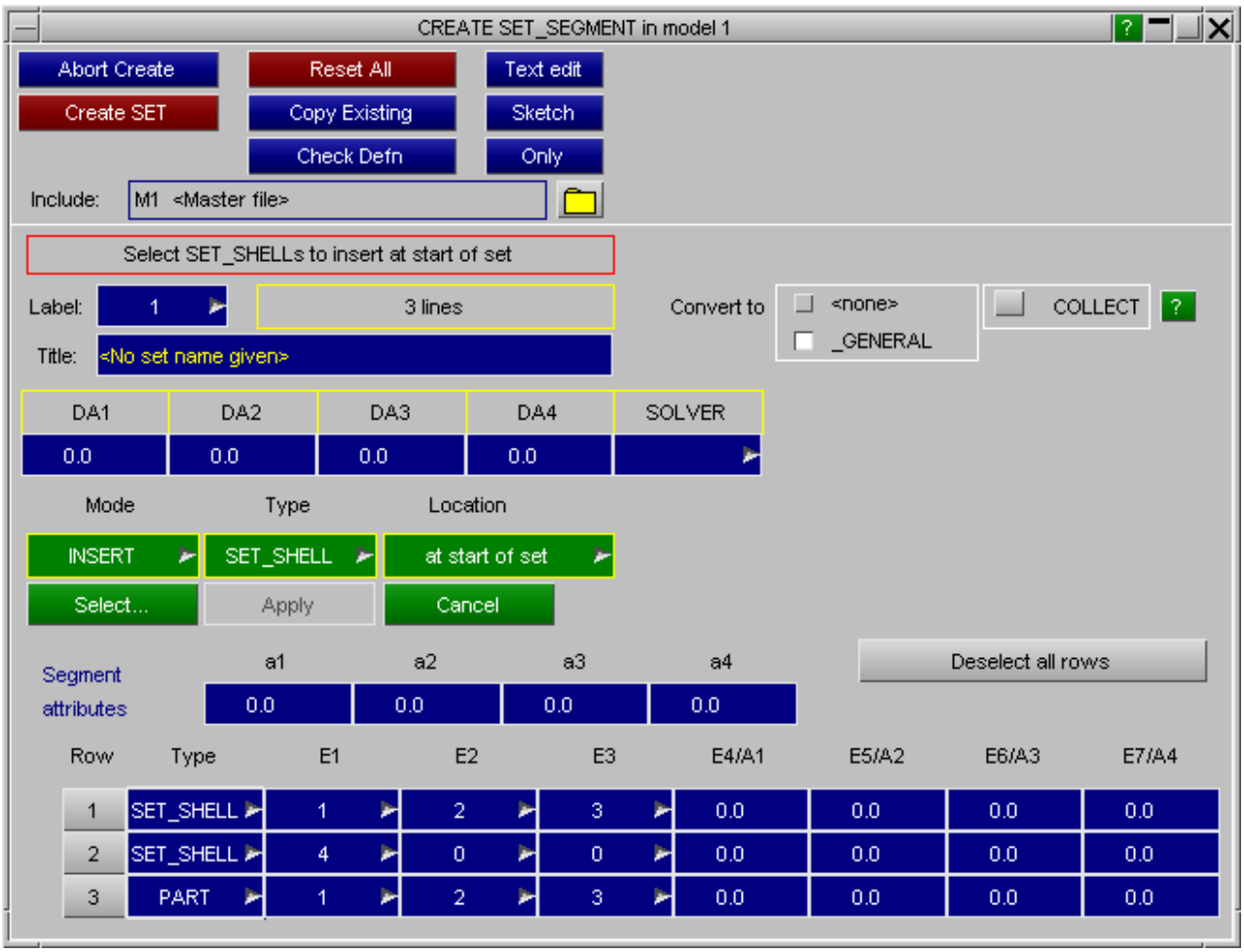
Creating/editing **`SET_SEGMENT_GENERAL`** sets

General segment sets are edited in the same way as other general sets. If needed segment attributes can be defined when inserting rows by using the **a1** to **a4 Segment attributes** boxes.

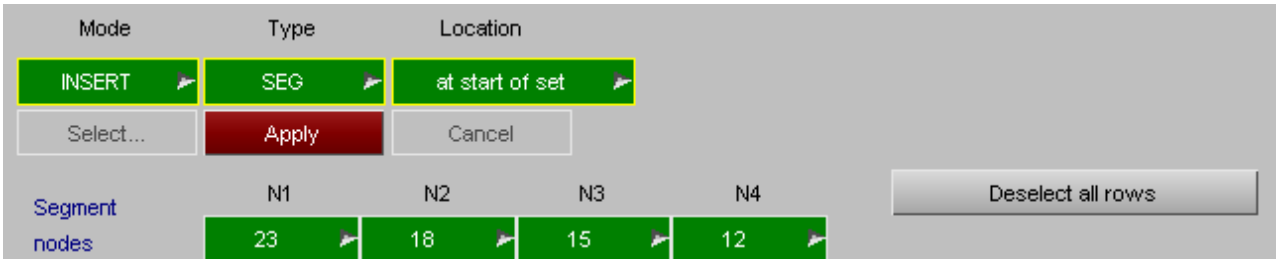
SEG and **DSEG** rows are added in a different way. These are inserted by selecting the four nodes.

Example: adding a **SEG** row to a **SET_SEGMENT_GENERAL**

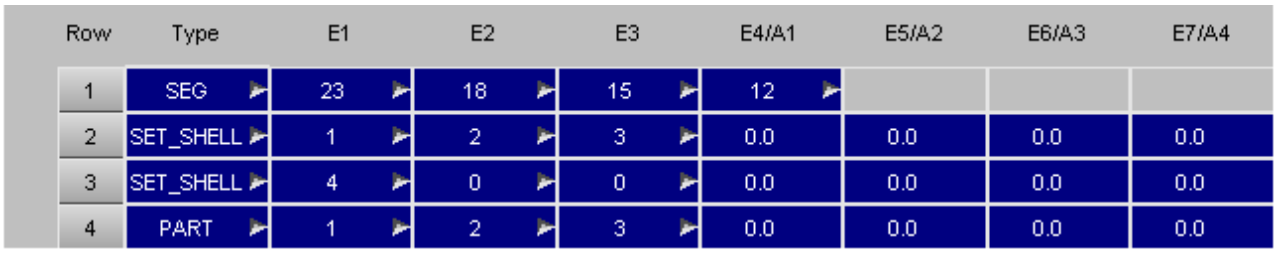
We want to add a SEG row to the top of the **SET_SEGMENT_GENERAL** set shown below:



Select mode **INSERT** , type **SEG** and location **at start of set** from the popups.



As we have selected **SEG** the Segment nodes popups are made live to be able to select the 4 nodes on the segment (for a triangular segment N3 = N4). Once the nodes are selected press **APPLY** to insert the row.



The row has been inserted at the top of the set.

Locking set contents against deletion

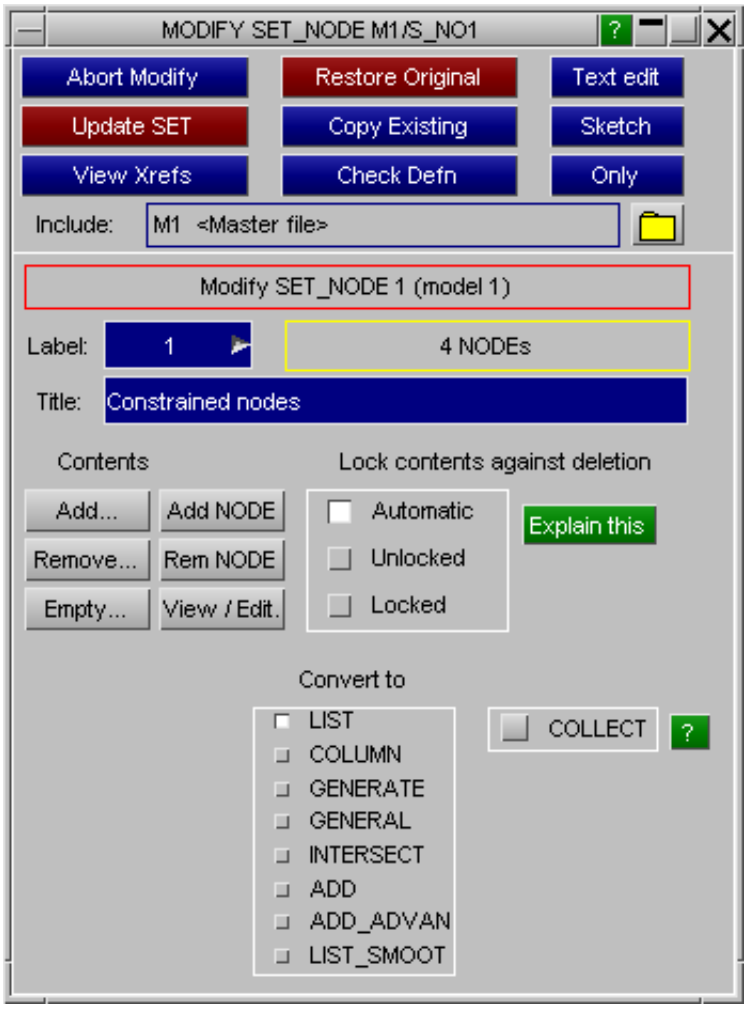
When items are deleted in PRIMER it has historically been the case that membership of a set does not - by default - "lock" an item against being deleted. This can be controlled globally on the deletion panel via the **Remove from sets** switch .

However there are a few cases where this logic can cause problems:

- Sliprings using 2D seatbelt elements rely on the contents of node and shell sets to define their geometry and connectivity
- Retractors using 2D seatbelt elements have a similar reliance
- Nodes and parts used in Dummy assemblies, where deletion of content is likely to be unintended.

  ... and other cases may arise in the future.

Therefore the concept of "locking" set contents against deletion on a "per set" basis has been added in PRIMER release 10.1.



Contents locking has three possible settings:

Automatic (default)	This is the default PRIMER behaviour that membership of a set does not lock contents against deletion ***unless*** the set is referenced by something known to be sensitive to this problem. At present such items are: • *ELEMENT_SEATBELT_SLIPRING when the slipring is for 2D seatbelt elements • *ELEMENT_SEATBELT_RETRACTOR when the retractor is for 2D seatbelt elements If either if thse types reference the set then its contents will be locked against deletion. This list of items may be added to in the future as the keyword format evolves.
Unlocked	This is the original PRIMER behaviour that membership of a set never locks contents against deletion, regardless of what references the set.
Locked	This is a new option, and if selected membership of a set always locks its contents against deletion

As stated above removal from sets during deletion also depends upon the **Remove from sets** switch being turned on.

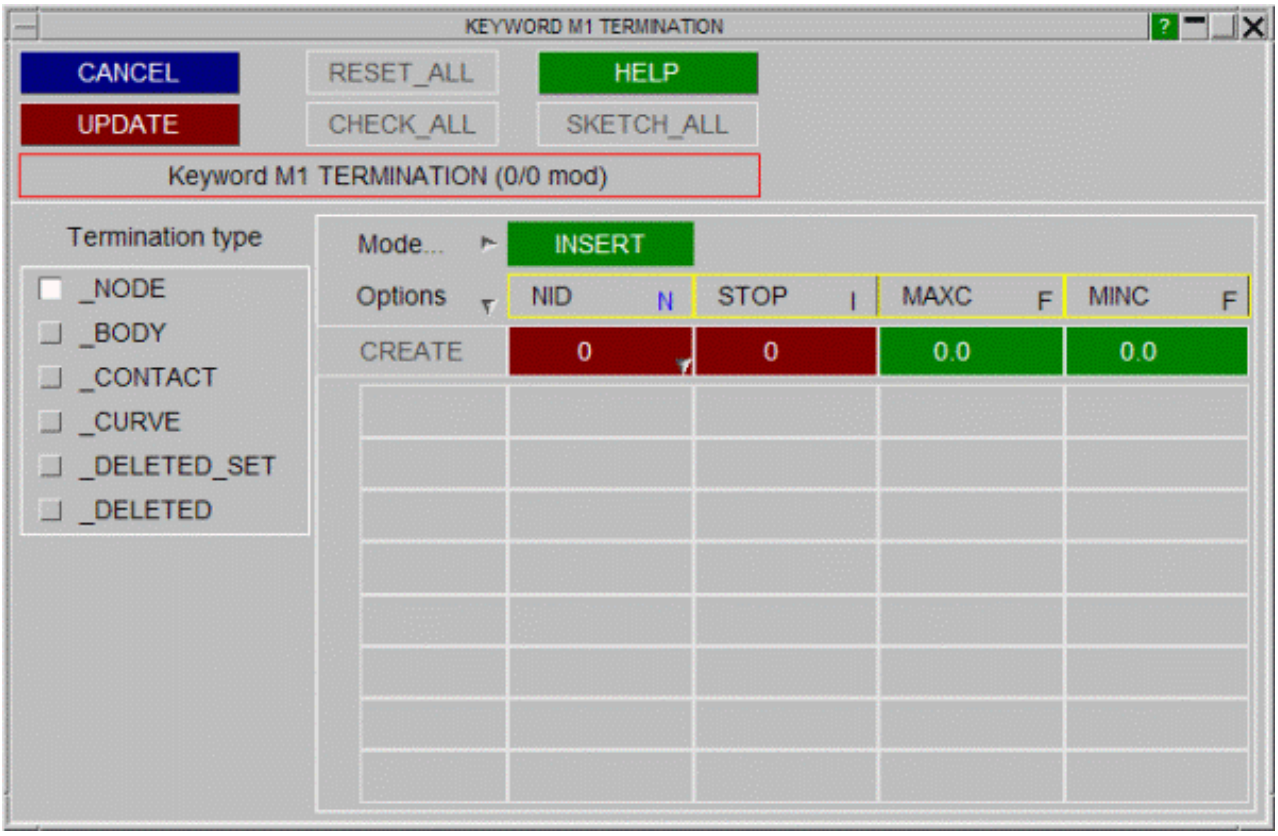
This setting is not "remembered" in the keyword output deck, so it will be lost when a model is deleted or a PRIMER session is terminated. It is hard to think of a situation in which the default **Automatic** setting will not be appropriate, and the use of this default is recommended, however the other two options are provided for completeness.

Visualising Sets.

Sets are not drawn explicitly, rather they are displayed by drawing their constituent parts, elements, nodes and segments. They may be **SKETCH**ed via the commands above, and in most contexts within PRIMER where sets are used it is possible to sketch them via their "daisy chain" popup menus.

TERMINATION

There are currently five sub-keywords available, _BODY, _CONTACT, _CURVE, _DELETED_SHELLS, and _NODE. These can be edited through the generic Keyword Editor.



5.2 Databases: Importing data from Pre-defined database files

Database files allow you to access predefined data of any type. Typical uses are material, section and loadcurve data; but anything which can be stored in tabular form may be accessed in this way.

The format of databases in PRIMER is described in appendix IX.

This section shows an example database and how it can be used in PRIMER.

An example of how to set up a database and how to use it.

A user 'guest' has set up:

- An **oa_database** file in his home directory (**$HOME**) which is **/guest**.
- There is also a global **oa_database** file in directory **$OASYS**.

These files are:-

in $OASYS

```
$ oa_database file for PRIMER in $OASYS
$
$ Any databases which are defined in here will be available to all users
$
$=========================================================================
$
$ type directory to find oa_index file in  database name
$ ==== ==================================  =============
$
LCUR*  /disk/database/loadcurve/seismic    seismic loadcurve database
LCUR*  /disk/database/loadcurve/material   material loadcurve database
MATL*  /disk/database/material             material database
MODEL* /disk/database/project1.dba         model build database proj 1
MODEL* /disk/database/project2.dba         model build database proj 2
```

in $HOME

```
$ oa_database file for PRIMER, user 'guest' in $HOME (/guest)
$
$ Any databases defined here will only be available to the user 'guest'
$
$=====================================================================
$
$ type directory to find oa_index file in database name
$ ==== ================================== =============
$
LCUR*  /guest/my_database/loadcurve       my loadcurve database
MATL*  /guest/my_database/material        my material database
MODEL* /guest/my_database/build.dba       my build database
```

Oa_index files

In this example when creating a material in PRIMER, we will import a material stress strain curve from a database. Note that this example describes a loadcurve database, while the more common database type is the material database. The material definitions then include the curve data (*DEFINE_CURVE as well as *MAT).The database we will import the curve from is 'example material loadcurve database'. This database is in the directory '**/disk/database/loadcurve/material**'. In this directory there MUST be an **oa_index** file which contains the database information.

The **oa_index** file located in this directory is shown on the next page.

The database contains 8 fields per entry and there are 10 entries (4 cold reduced steels and 6 hot rolled steels). Each

entry refers to a T/HIS curve file (as this is a **LCUR** database) in the directory/**disk/database/loadcurve/material**

```
$ Material stress-strain
curves
$

$==============================
$ NUMBER OF COLUMNS IN
DATABASE
8

$==============================
$ THE COLUMN NAMES (FILENAME
FIRST)
Filename
Material
Grade (GB)
Grade (Germany)
Grade (Japan)
Grade (USA ASTM)
Units
Description

$==============================
$ THE DATABASE ENTRIES
$
$ COLD REDUCED STEELS
$ -------------------
cr4_steel.cur
Steel
CR4
St12
SPCC
A366
MPa
Cold Reduced - Forming and
Drawing
$
cr3_steel.cur
Steel
CR3
St13
SPCD
-
MPa
Cold Reduced - Forming and
Drawing
$
cr2_steel.cur
Steel
CR2
-
SPCE
A619
MPa
Cold Reduced - Forming and
Drawing
$
cr1_steel.cur
Steel
CR1
RRSt14
SPCEN
A620
MPa
Cold Reduced - Forming and
Drawing
$
```

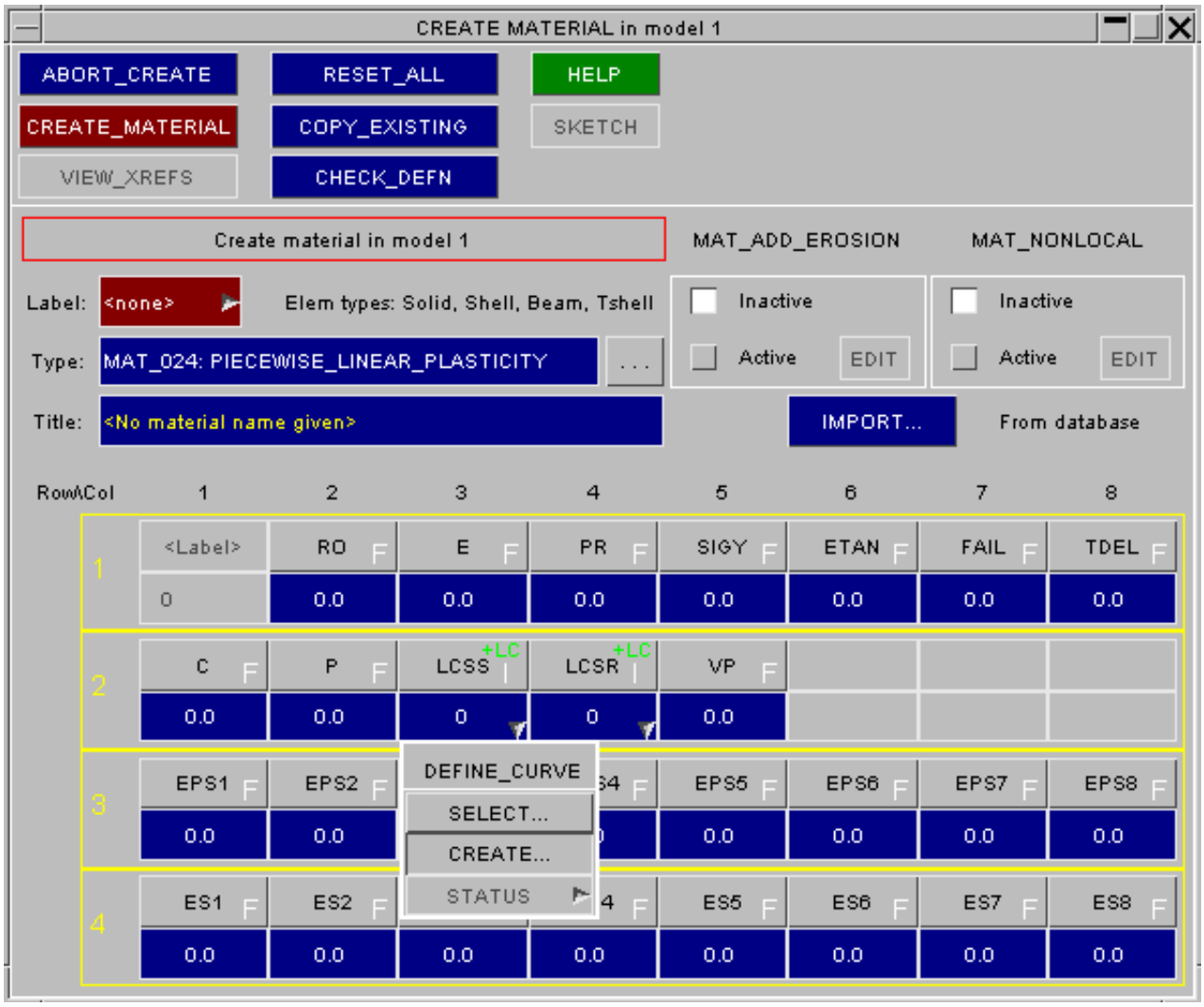
[continued on next column]

[from previous column]

```
$--------------------------------
$ HOT ROLLED STEELS
$ -----------------
hr15_steel.cur
Steel
HR15
-
-
-
Mpa
Hot Rolled - Forming and
Drawing
$
hr14_steel.cur
Steel
HR14
-
-
A569
Mpa
Hot Rolled - Forming and
Drawing
$
hr4_steel.cur
Steel
HR4
-
SPHC
-
Mpa
Hot Rolled - Forming and
Drawing
$
hr3_steel.cur
Steel
HR3
StW22
SPHD
-
Mpa
Hot Rolled - Forming and
Drawing
$
hr2_steel.cur
Steel
HR2
StW23
-
A621
Mpa
Hot Rolled - Forming and
Drawing
$
hr1_steel.cur
Steel
HR1
StW24
SPHE
A622
Mpa
Hot Rolled - Forming and
Drawing
```

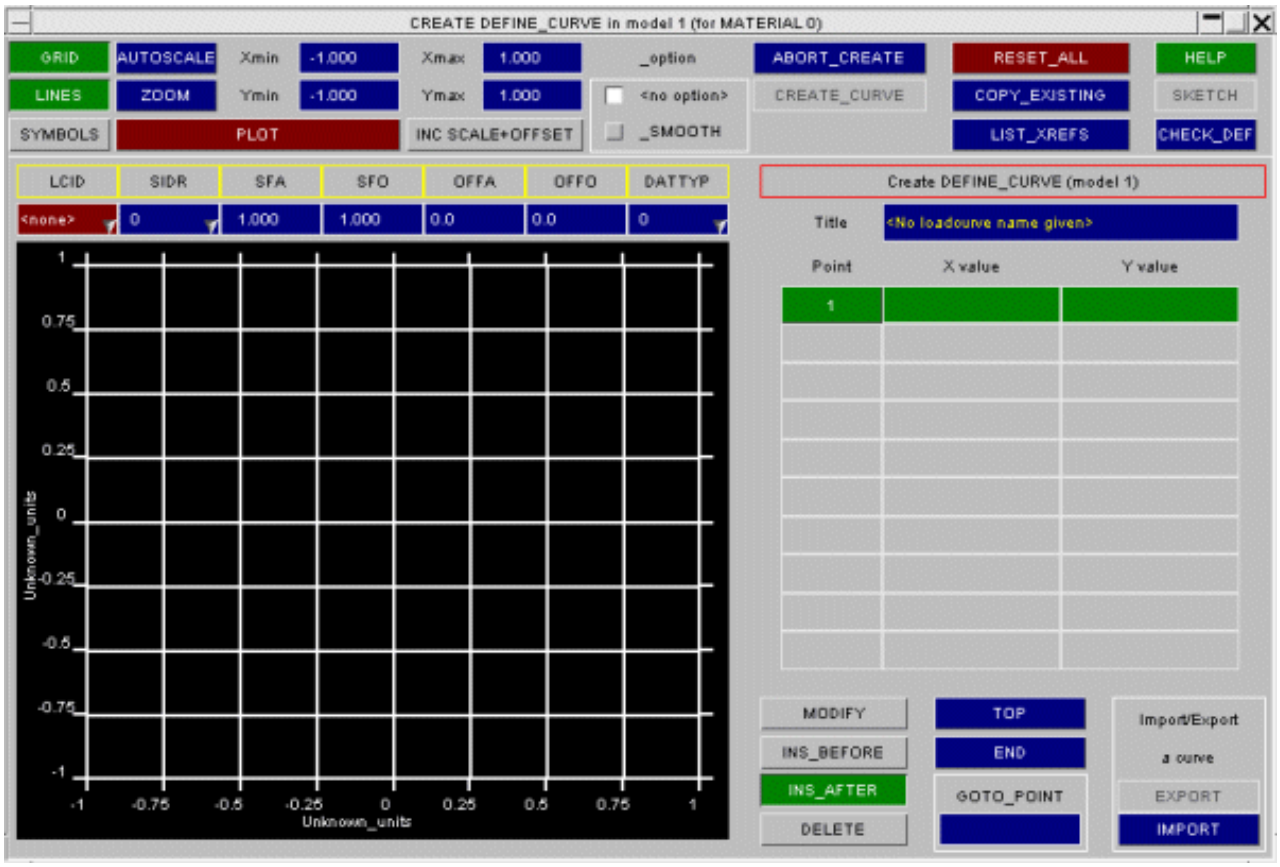
In the material creation window for PRIMER a material which refers to a loadcurve (**`PIECEWISE_LINEAR_PLASTICITY`**) is selected.

To import a loadcurve for the stress strain curve use the right mouse button to bring up the **LCSS** popup box. Select **CREATE** from the menu and this will start the Loadcurve creation box.

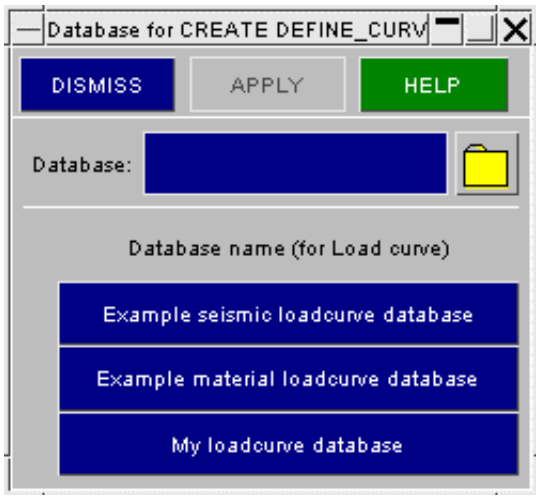


The loadcurve creation box allows a new curve to be created.

The **IMPORT** button on the bottom right of the window allows a curve to be read in from a file or database. Press this button.



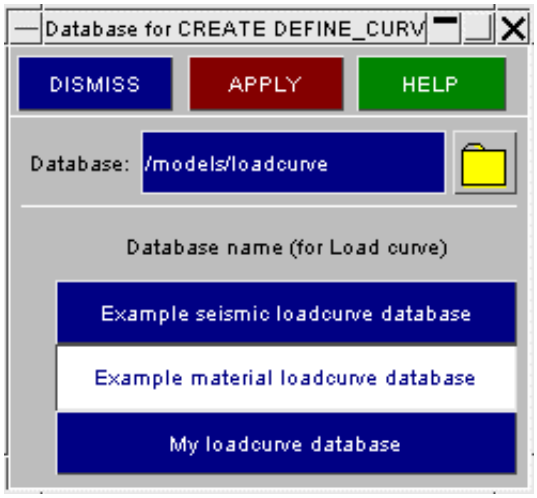
The Import part of the loadcurve creation box allows a curve to be read from a database by pressing the **DATABASE** button.



Pressing the DATABASE button in the loadcurve creation window starts the database selection window. Each button in the window corresponds to an entry in an oa_database file. The first two buttons are from the **oa_database** file in $OASYS. The last button is from the oa_database file in $HOME.

Only 3 databases are shown as these are the only ones which refer to **LCUR** databases.

Until a database is selected the **APPLY** button is inactive (greyed out).

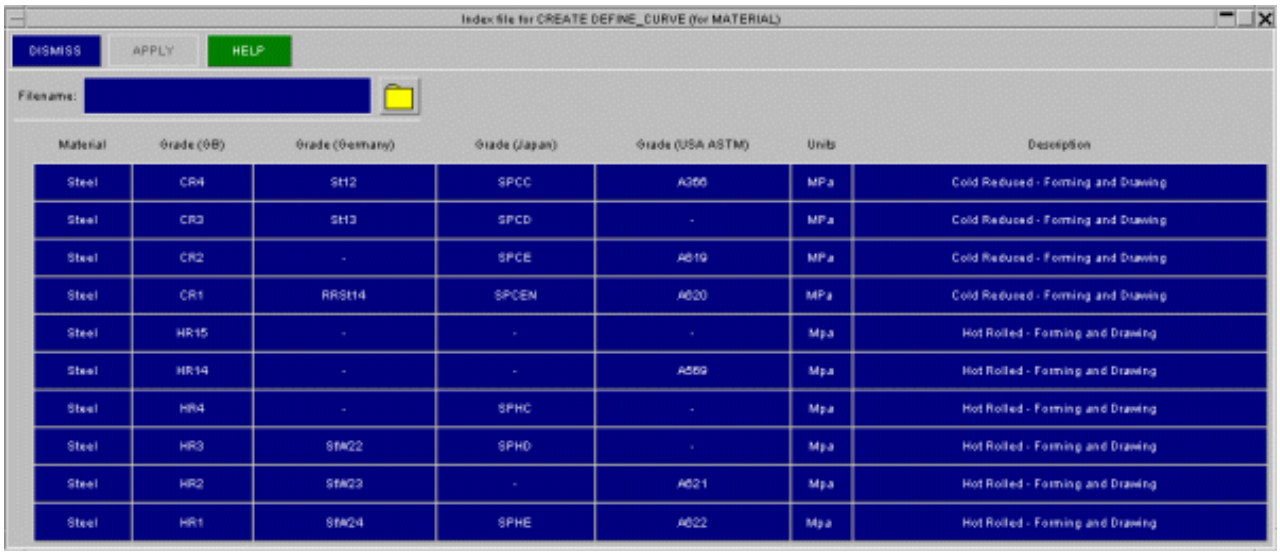


When a database file is selected it is highlighted and the **APPLY** button becomes RED allowing the user to select that database.

A different database can be selected as required. The one which is highlighted when **APPLY** is pressed is the one which will be read.

In our example we select a file from ’example material loadcurve database’.

When **APPLY** in the database selection window is pressed, PRIMER reads the oa_index file which is in that directory and creates a window with the entries from this file.

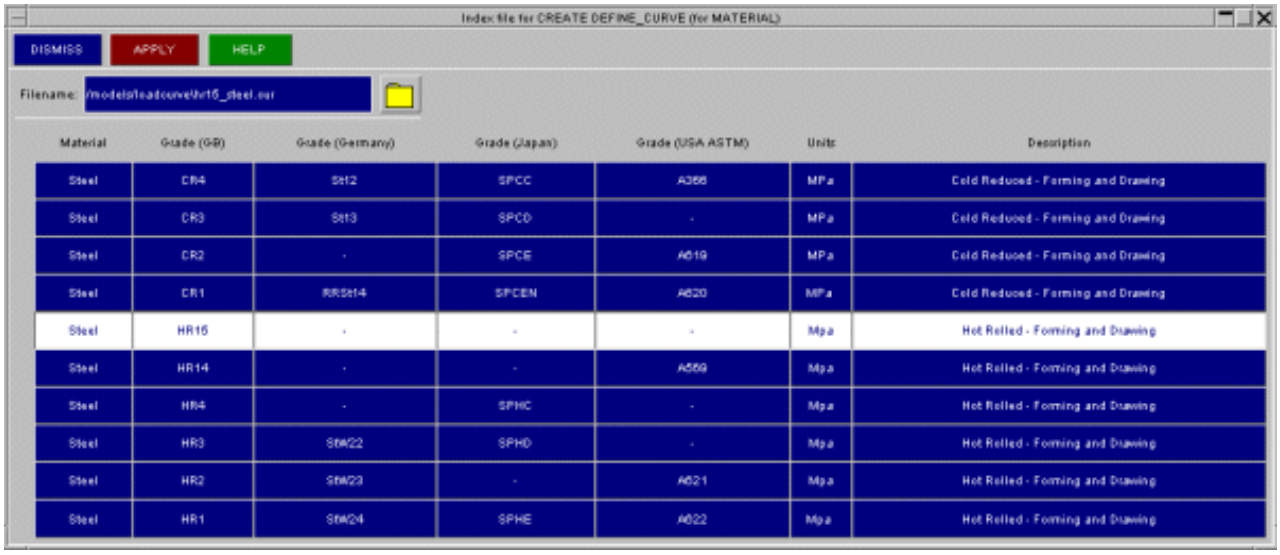


In this example you can see that the 10 entries which were in the oa_index file are all present in the window, each appearing on a row. If there were more then 10 entries in the oa_index file a scroll bar would allow you to scroll through the entries. Eight fields were defined for each entry. The first (the filename) has not appeared in the window but is stored internally. The remaining seven field headers appear above each column in yellow. If the number of fields does not fit on the window a scroll bar will be displayed.

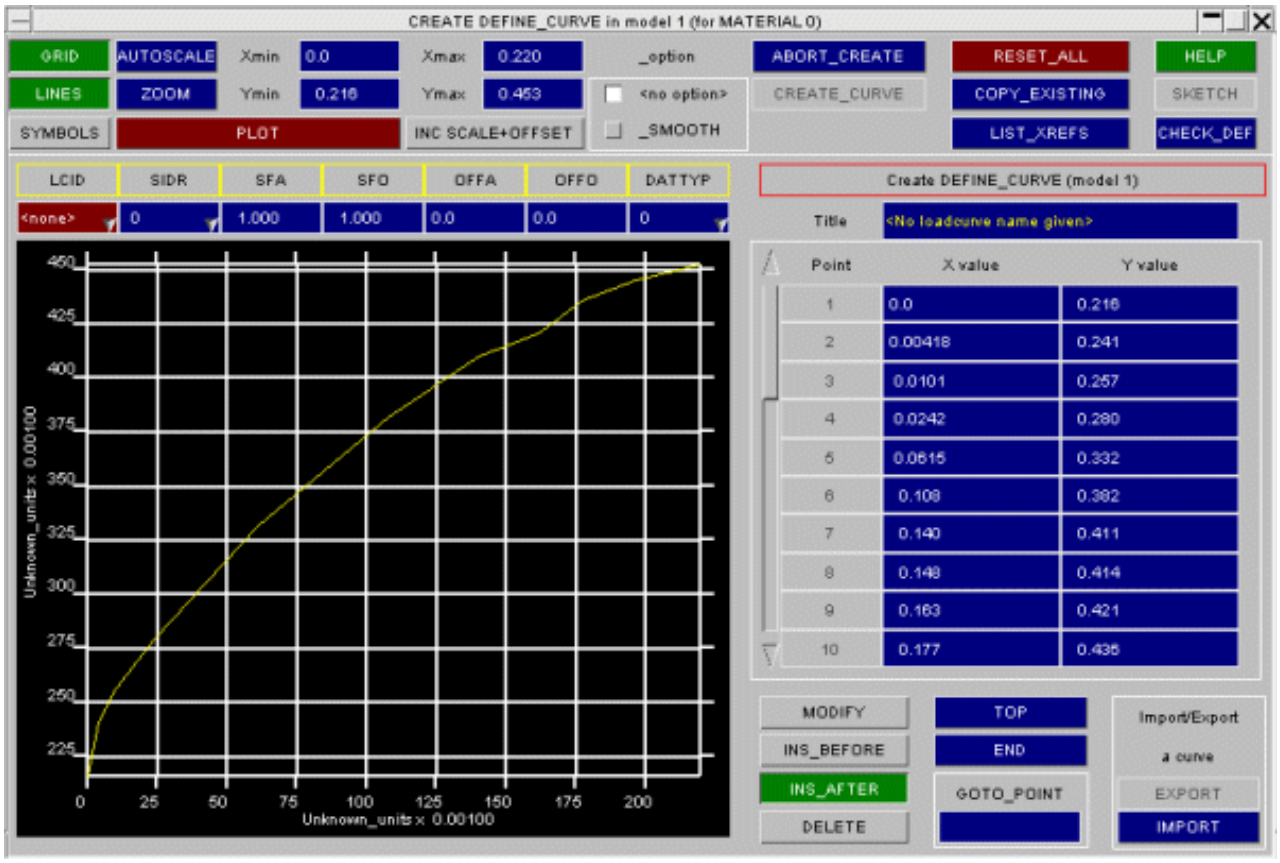
Until a file is selected the **APPLY** button is greyed out.

When a database entry is selected it is highlighted and the **APPLY** button becomes RED allowing the user to select that entry. A different entry can be selected as required. The one which is highlighted when **APPLY** is pressed is the one which will be read.

Here the HR15 steel is selected.



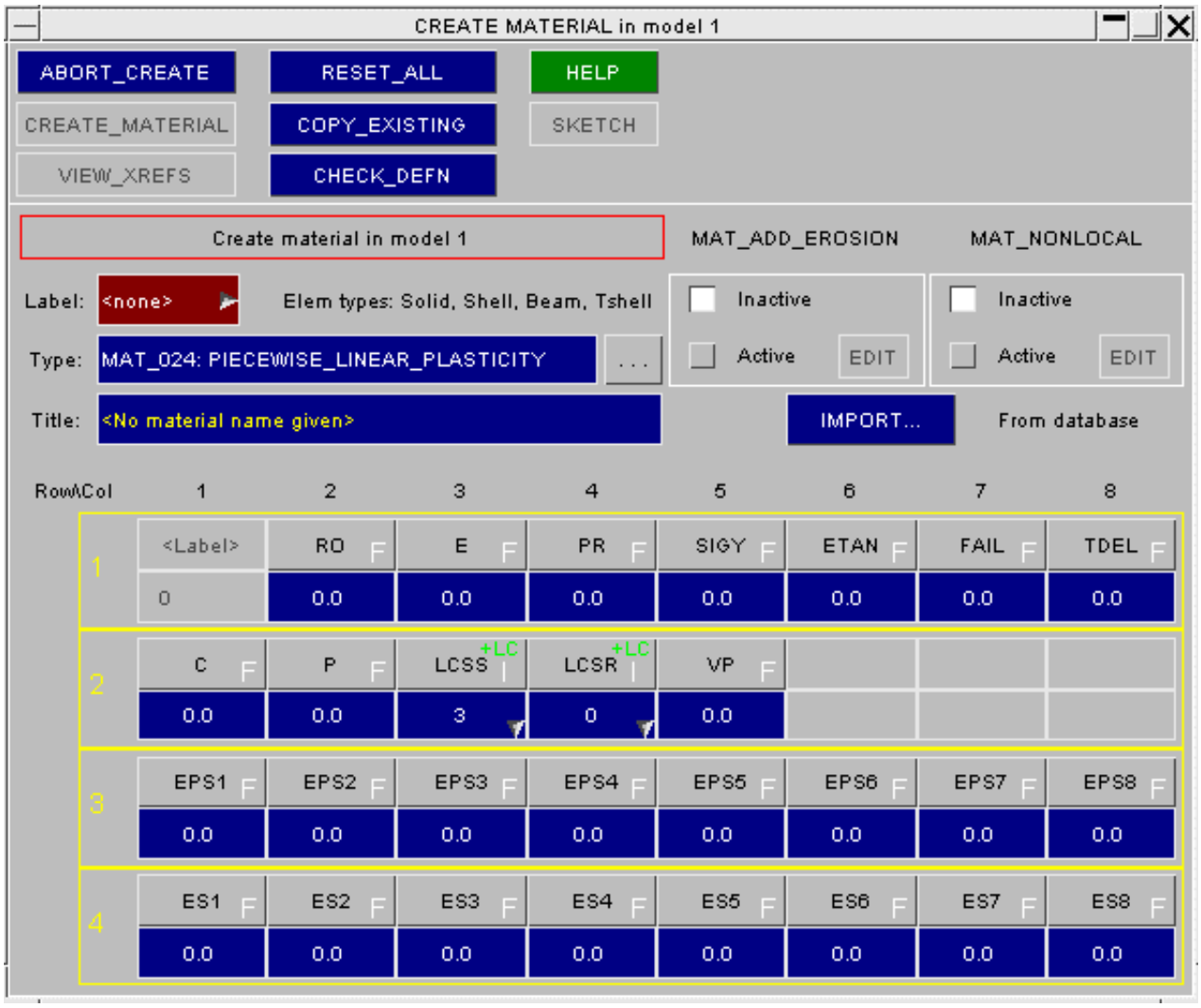
If **APPLY** is pressed this will be imported into the loadcurve editor and plotted.



The curve can be modified if required in the editor.

When a label has been given to the curve the **CREATE_CURVE** button will be ungreyed (made active) allowing the curve to be created.

When this is done the **LCSS** field in the material editor is updated with a new loadcurve ID referencing the imported data.


CREATE MATERIAL in model 1
ABORT_CREATE
RESET_ALL
HELP
CREATE_MATERIAL
COPY_EXISTING
SKETCH
VIEW_XREFS
CHECK_DEFN
Create material in model 1
MAT_ADD_EROSION
MAT_NONLOCAL
Label: <none>
Elem types: Solid, Shell, Beam, Tshell
Inactive
Active
EDIT
Inactive
Active
EDIT
Type: MAT_024: PIECEWISE_LINEAR_PLASTICITY
Title: <No material name given>
IMPORT...
From database
Row\Col 1 2 3 4 5 6 7 8
1 <Label> RO E PR SIGY ETAN FAIL TDEL
0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
2 C P LCSS LCSR VP
0.0 0.0 3 0 0.0
3 EPS1 EPS2 EPS3 EPS4 EPS5 EPS6 EPS7 EPS8
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
4 ES1 ES2 ES3 ES4 ES5 ES6 ES7 ES8
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0

5.3 Contact Penetration Checking

- What Checking Does
- Plotting errors
- **Settings...** Controlling plots
- **Levels...** Setting #contour bands
- Displaying local errors
- Creating null beams on crossed edges
- **LIST_ERRORS**: Error output listings
- Options...
- Notes on Contact Penetration Checking

Primer can check for and display initial penetrations and crossed edges in contact surfaces. It can also be used to find which nodes will be tied (or not tied) by *CONTACT_TIED.
This capability can be invoked from three separate locations:

CHECK (from Tools) > RULES
CONTACT > PEN_CHECK
CONTACT > CREATE/EDIT > PEN_CHECK

However the routines called are common to all cases, and their detailed use is described here.

Fixing contact penetrations can be sound in section 5.4

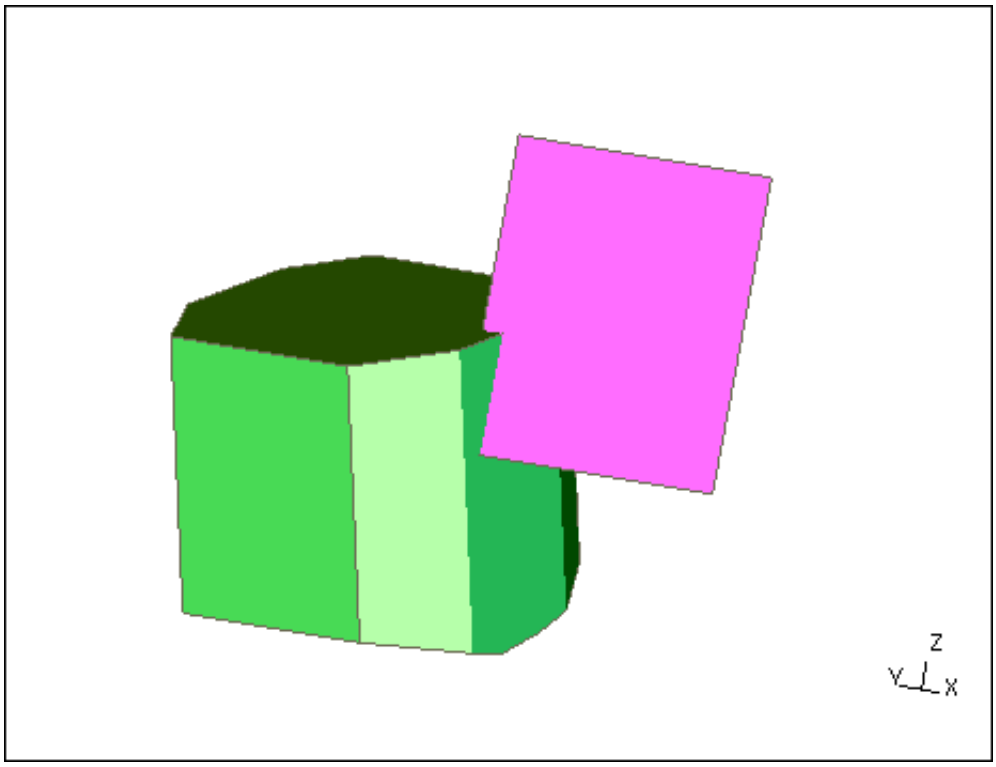
In this example a model contains two parts:

- Part 1 (pink) is a single shell.
- Part 2 (green) is a block of solids

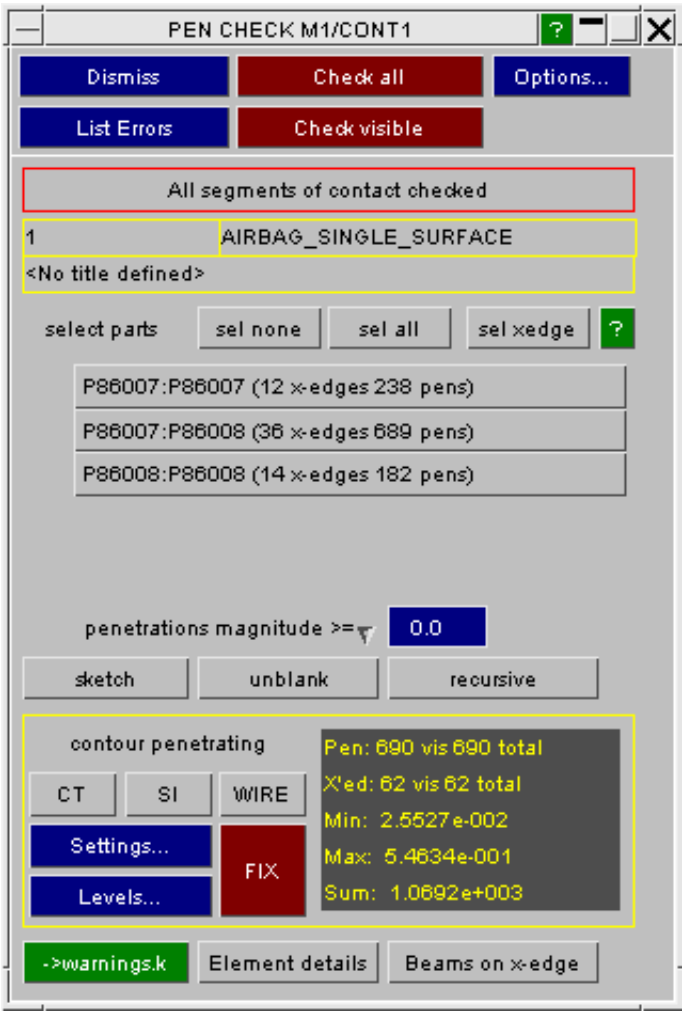
A contact surface (**`AUTOMATIC SURFACE TO SURFACE`**) between the two parts is defined:

- Part 2 is the slave side of a contact
- Part 1 is the master side

The two parts intersect by a small amount, as is clearly visible here.



5.3.1 Checking a Sliding Contact



Check all Checks the whole contact (recommended)

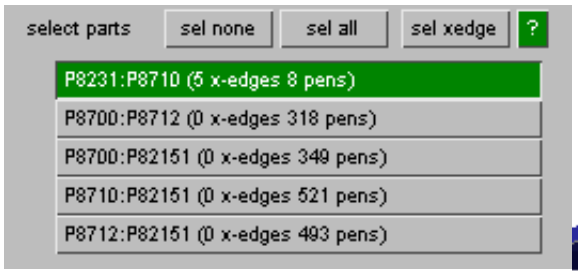
Check visible Only checks visible volume of the contact. If contact is large and substantial part of it is blanked, this option may save time particularly during iterative fixing procedures.

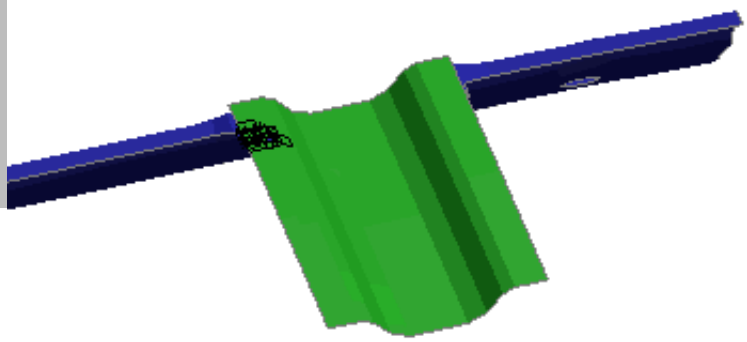
List Errors Lists all penetrating nodes, the element(s) they penetrate and the penetration distance. Also lists all crossed edges found.

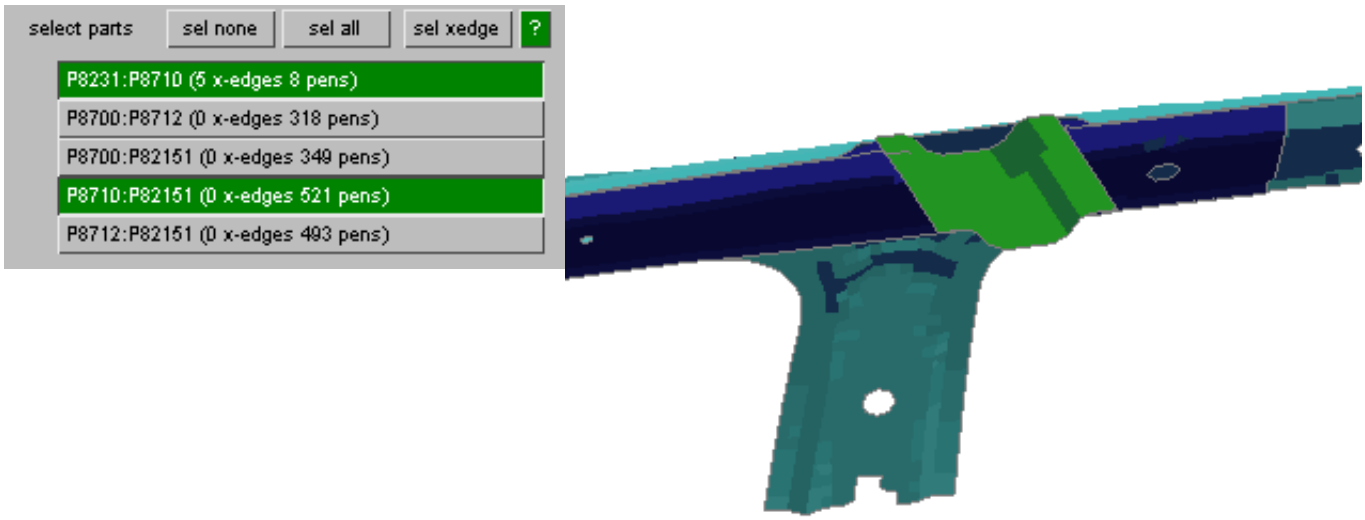
Options... Maps the contact penetration check options panel in which checker settings may be changed.

The front panel enables you to control visibility by unblanking interacting part pairs. It also pre-selects part pairs for fixing of crossed edges or penetrations in the default mode which is to observe blanking.

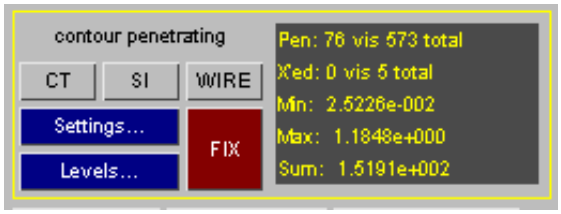
- Selection by a single click will exclusively select and "only" the display for a part pair.
- Ctrl-select will add another part pair to the seletion
- Shift-select in this context will select all part pairs which use P1. e.g. Shift-sel on P8710:P82151 also unblanks P8231as in interacts with P8710.
- Ctrl-shift-select does the same for all part pairs which use P2.





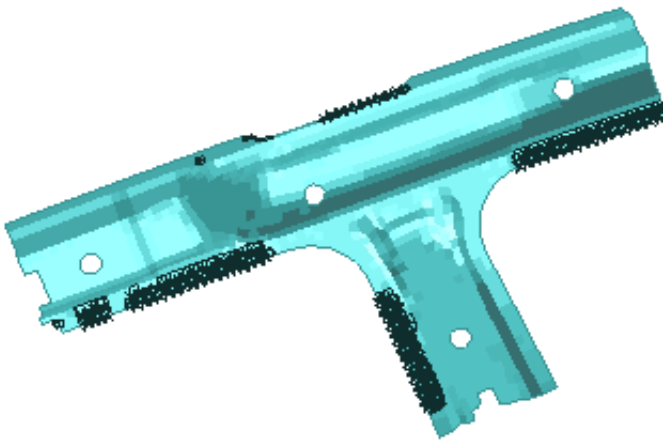


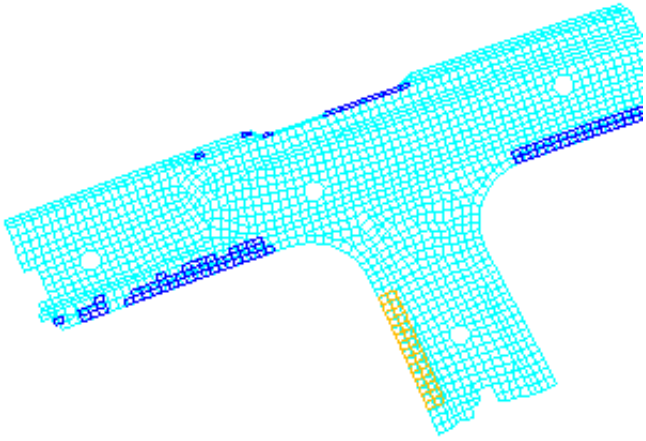
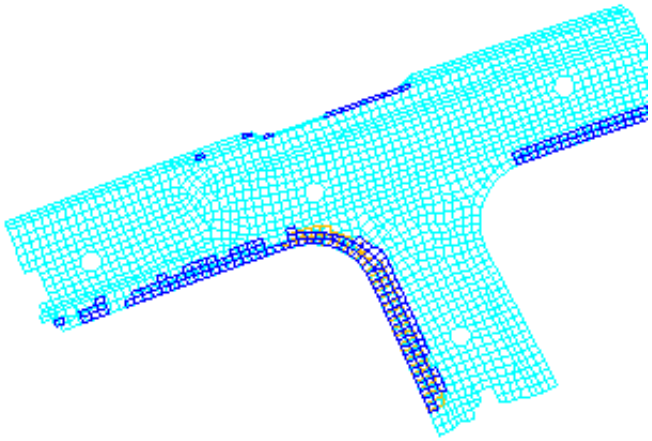
- **sel none** clears the selection and unblanks all parts in the contact
- **sel all** will "only" all the parts of the contact
- **sel xedge** will select all the interactions where crossed edges are found



The information panel gives the count of visible/total crossed edges and penetrations. Note - this is the (more useful) count of penetrating nodes not penetration events.

The following functions are especially useful when one part only is initially displayed.

	**sketch** will sketch all the visible segments involved in a penetration and all the segments that penetrate/cross them

	**unblank** will unblank underlying elements of all segments which penetrate/cross visible segments (i.e. blue and yellow shells)
	**recursive** will perform unblank until no more segments are found (note the extra blue shells which penetrate the yellow)

->warnings.k - this functions writes error sets to include file *warnings.k* which will be created if it does not exist. The same function is available for contacts from the model check error tree.

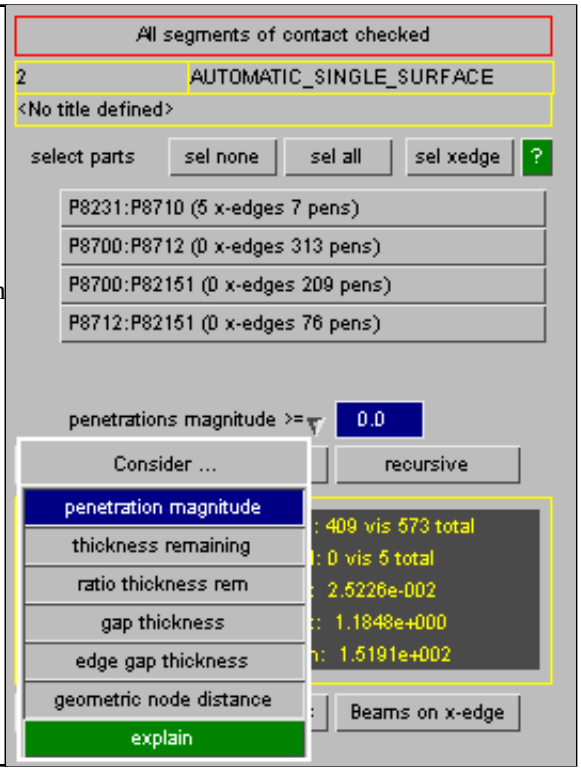
- If there are crossed edges in the contact, these will be written to a segment set named "*Contact <id>: Contact has crossed edges*".
- If there are penetrations, both a node set and a segment set will be created with matching name which also denotes the error, e.g. "*Contact <id>: penetration exceeds max allowable value @0.5*"

Filtering penetrations by magnitude

- Penetration magnitude - only consider penetrations greater than the given value
- thickness remaining - only consider penetrations where remaining unpenetrated thickness between segments is less than the given value. This is defined as 0.5*(t1+t2) - P, where t1,t2 are segment thicknesses and P is penetration magnitude
- thickness remaining ratio - only consider penetrations where remaining unpenetrated thickness expressed as ration of overall segment thickness is less than the given value

Changing this value will change the penetration count, the contour plot and the fixing procedure. Fixing will de-penetrate (not fully) but up to the specified limit.

This does not affect the handling of crossed edges.



CT, SI and **WIRE** provide Continuous Tone, Shaded Image and Wireframe plots respectively of the contact errors.

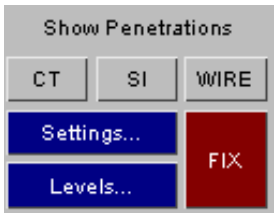
Settings... controls the parameters of these plots.
Levels... controls the contour bands used.

FIX accesses the de-penetration fixing function.

Element details permits more detailed examination of the errors in elements adjacent to a node, or in a particular part. The **CT, SI** and **WIRE** plotting modes are the same as above.

Beams on x-edge generates "null beams" on crossed edges. These can be used in external meshing programmes to identify where the problems occur, making remeshing easier.

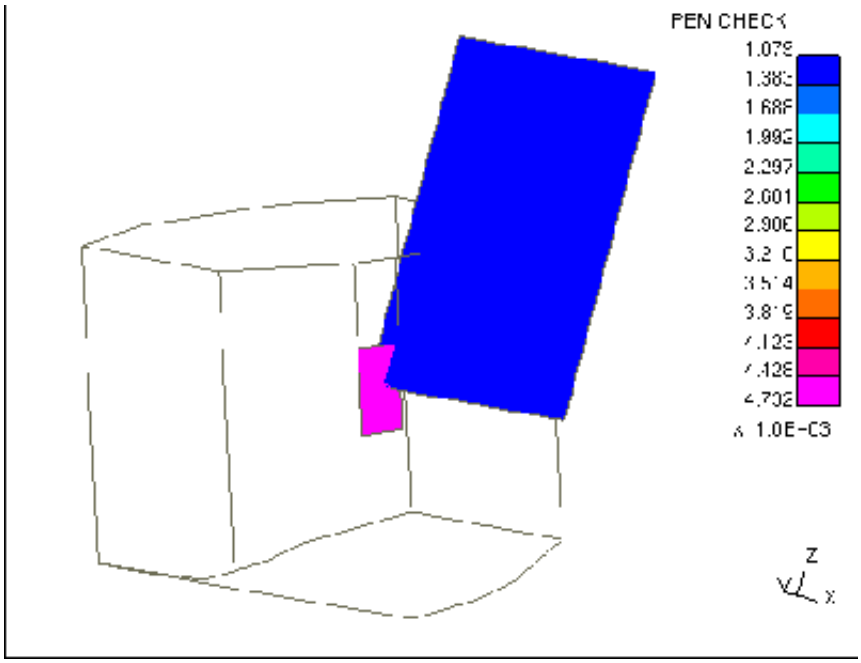
5.3.2 Plotting contact penetrations



This is a **CT** (Continuous Tone) plot of the contact penetrations.

Penetrated segments are drawn in a colour determined by the depth to which nodes penetrate them. The penetrating nodes and their "escape" vectors are drawn too.

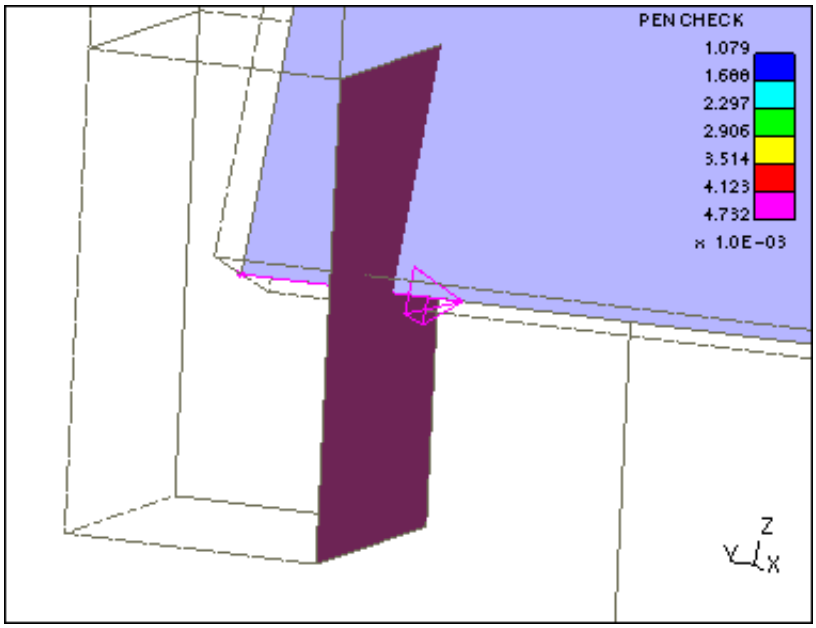
In this image the rest of the contact is drawn in "wireframe" mode: this, and other plotting parameters, are controlled in the **Settings...** panel.



This is a **SI** (Shaded Image) plot of the penetration region.

The escape vector of the shell node (to "escape" from the solid) is clearly visible.

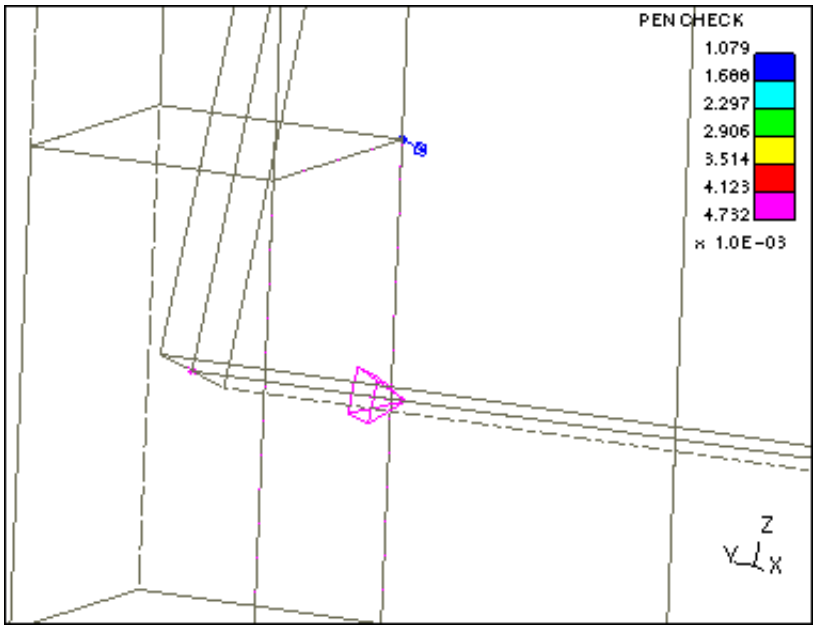
In addition the elements have been drawn "as thick" (controllable from the **Settings...** panel). This draws their thickness for contact purposes in grey lines.



This is a **WIRE** plot of the same region.

No shading or hidden surface removal takes place, and this makes it possible to see the other nodal penetration (of the solid node into the shell element). This was obscured in the previous plots.

The elements have still been drawn "as thick" here.

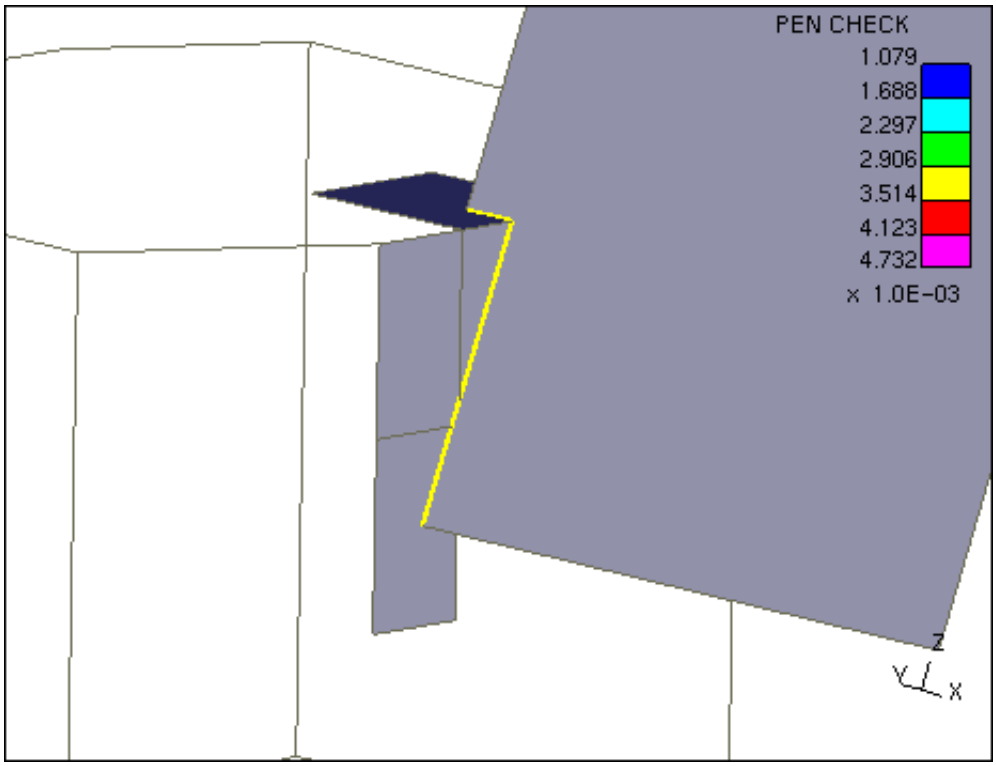


The plots above showed penetrations.

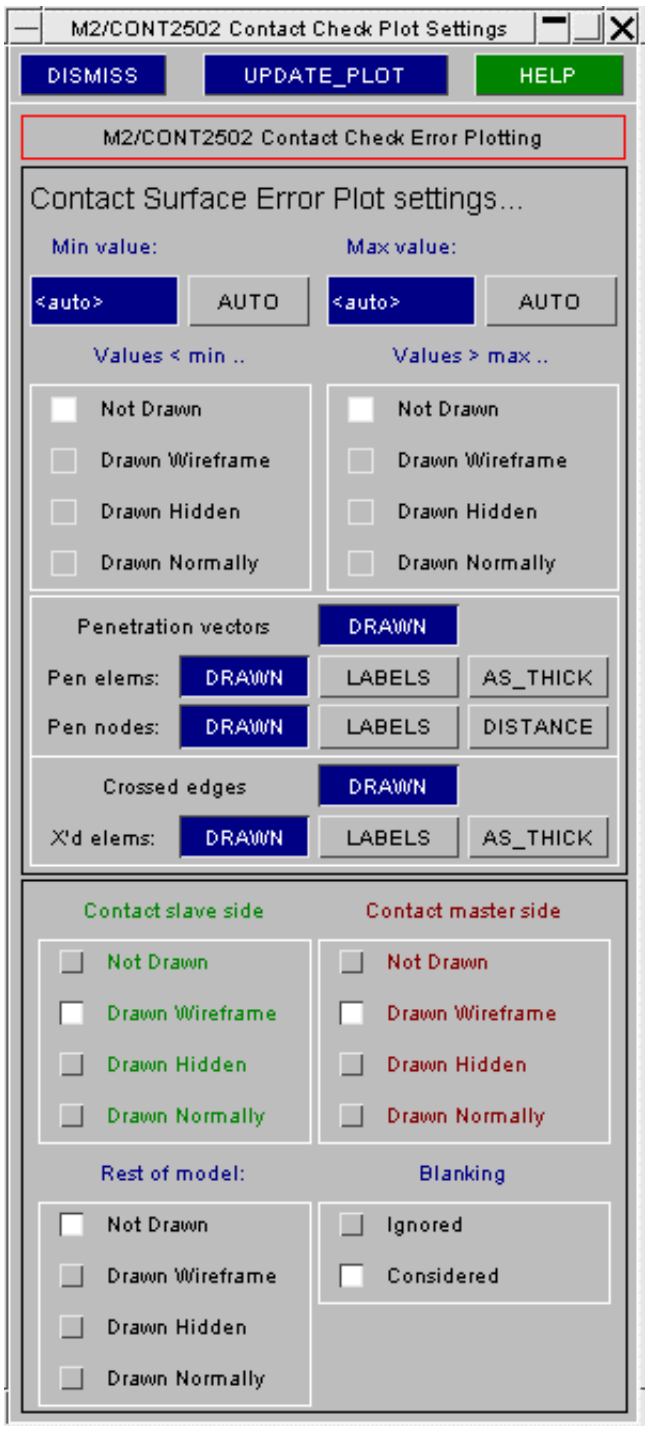
This is an `SI` plot of the crossed edges:
- Edges are drawn as thick yellow lines
- Penetrated elements are drawn in grey

It is possible to display penetrations and crossed edges on the same plot (the default) but this can lead to confusing images.

The display of each category of error is controllable separately in the **Settings...** panel.



5.3.3 **Settings...** Controlling plots



This section controls the min and max penetration distances that will be displayed (the contour band bounds).
By default the **Min value** and **Max value** are each automatic, displaying all penetrations, but each may be defined to a +ve value.

If either value is defined you can select how the segments which fall below/above this value are (or are not) displayed: [Not Drawn, Wireframe, ...].

This section controls what is actually drawn.

Penetration vectors: (which here are not drawn) are split into:

Penetrated elements	**DRAWN**	Whether or not they are drawn
	LABELS	Whether they are labelled (elem id)
	AS_THICK	Whether contact thickness is displayed
Penetrating nodes	**DRAWN**	Whether the nodes are drawn
	LABELS	Whether they are labelled (node id)
	DISTANCE	Whether penetration distance is added

Crossed edges:

Crossed elements	**DRAWN**	Whether or not they are drawn
	LABELS	Whether they are labelled (elem id)
	AS_THICK	Whether contact thickness is displayed

This section controls the display of each side of the contact [**Not Drawn**, **Wireframe**...] and also the rest of the model (same options).

Plotting contact offsets from the nodal plane in shells

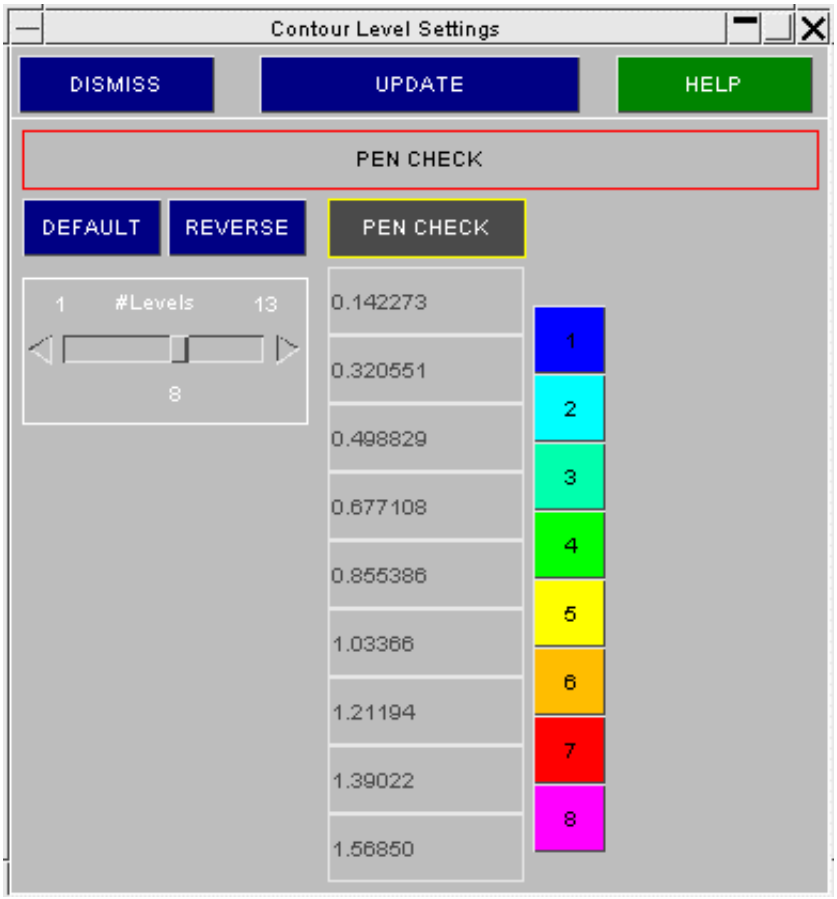
By default neither LS-DYNA nor PRIMER consider any offsets from a shell's nodal plane (as defined by field **NLOC** on ***SECTION_SHELL**, or explicit locations on ***INTEGRATION_SHELL**, or the **_OFFSET** suffix on ***ELEMENT_SHELL**) when calculating contact penetrations. However shell offsets ***are*** considered if:

- Field **CNTCO** on ***CONTROL_SHELL** is non-zero
  and
- The contact surface type is not **AUTOMATIC**.

In this situation PRIMER applies the offset during penetration checking in the same way as LS-DYNA.

Contact penetration plots will continue to draw segments and nodes in the "nodal plane" (ie not offset) by default, however if you turn on the **AS_THICK** option then the "true thickness" outline of contact segments on shells will be drawn in the offset position.

5.3.4 **Levels...** Setting the contour bands

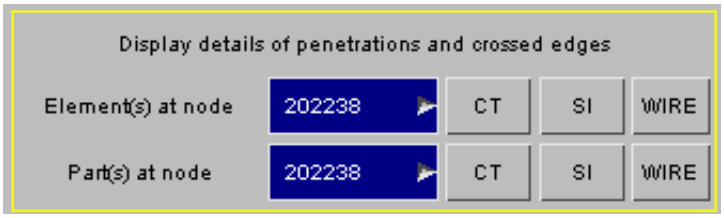


This panel (the standard panel in all contouring contexts) controls how many contour levels are displayed.

Note that it does not define the contour band values themselves:

> The upper and lower bounds are controlled in the **Settings...** panel, together with the display mode for items outside these bounds.

5.3.5 Details of errors local to elements and parts.



The main **CT**, **SI** and **WIRE** commands display errors for the whole model, as shown above.

The particular example illustrated is very simple, so display clutter is not a problem, but in a more complex model it is easy to imagine how confusing a plot of contact penetrations can get.

To make it easier to see what is going wrong you can select, by any means, a node (separate nodes may be selected for each case.)

Element(s) at the node

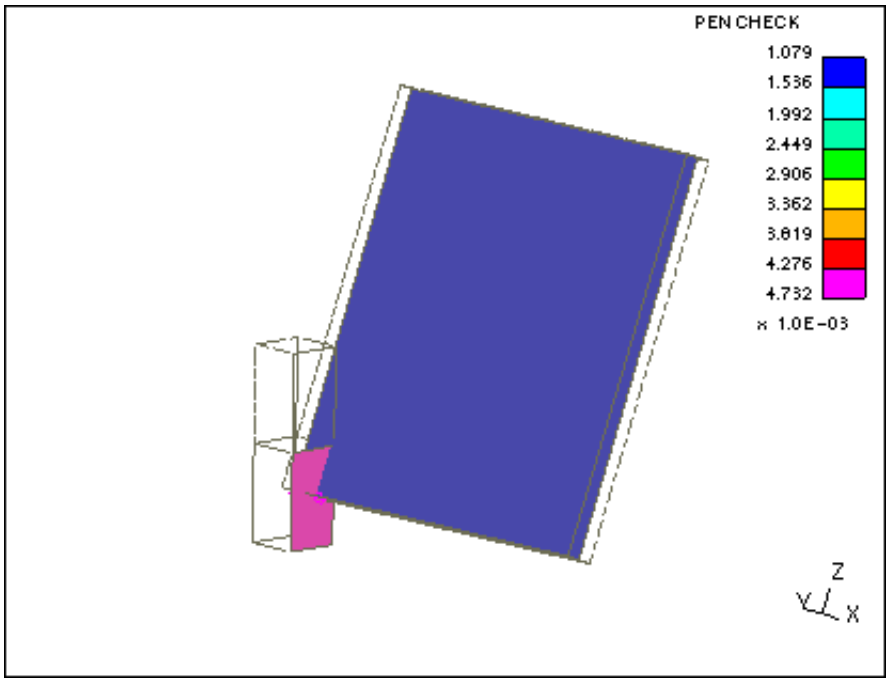
Only the elements related to this node are displayed, and the plot is autoscaled to these.

This means:

- All elements to which the node is attached
- All elements into which this node penetrates

This is an **SI** plot of the example.

All other plotting parameters (#levels, Settings... parameters, etc) are kept as before.

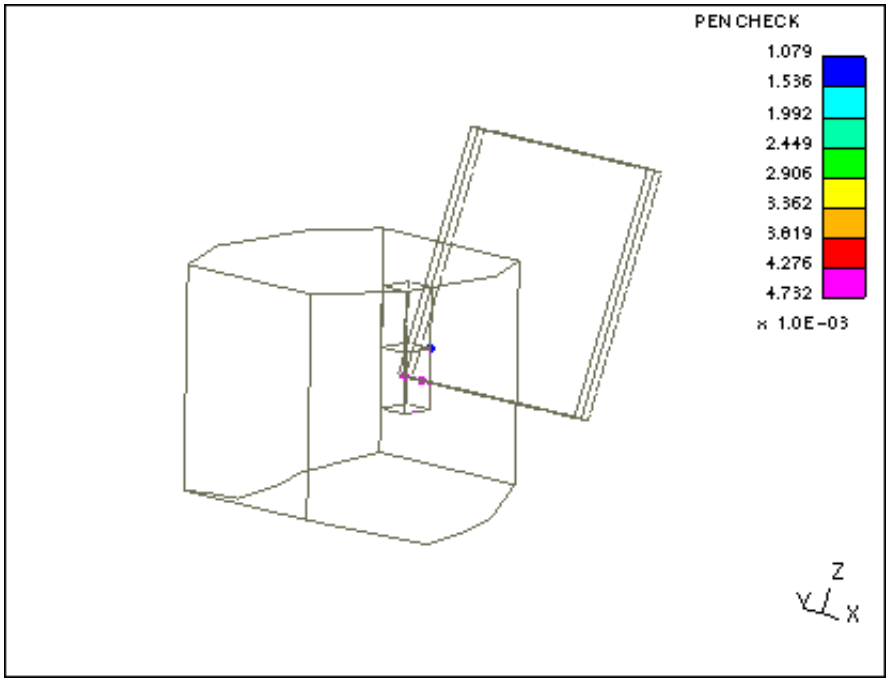


Parts(s) at the node

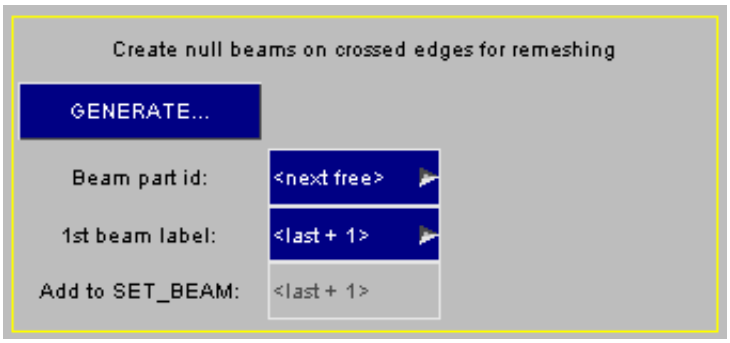
The elements related to this node are detected, as above; and all the elements of their respective parts are displayed in the mode selected in the **Settings...** panel.

This is a **WIRE** plot of the example.

Because this model is so simple the whole model, which only contains two parts, is drawn. However in a complex model this display mode allows you to determine how two parts interfere.



5.3.6 Generating "null beams" on crossed edges.



Generally the presence of crossed edges will require some remeshing, and this task will be performed outside Primer.

To make it easier to identify the edges externally you can generate "null beams" on these edges. A "null beam" normally (although you can change this) references **`*MAT_NULL`** and serves no structural purpose.

This procedure is entirely optional.
Primer allows you to control the following settings:

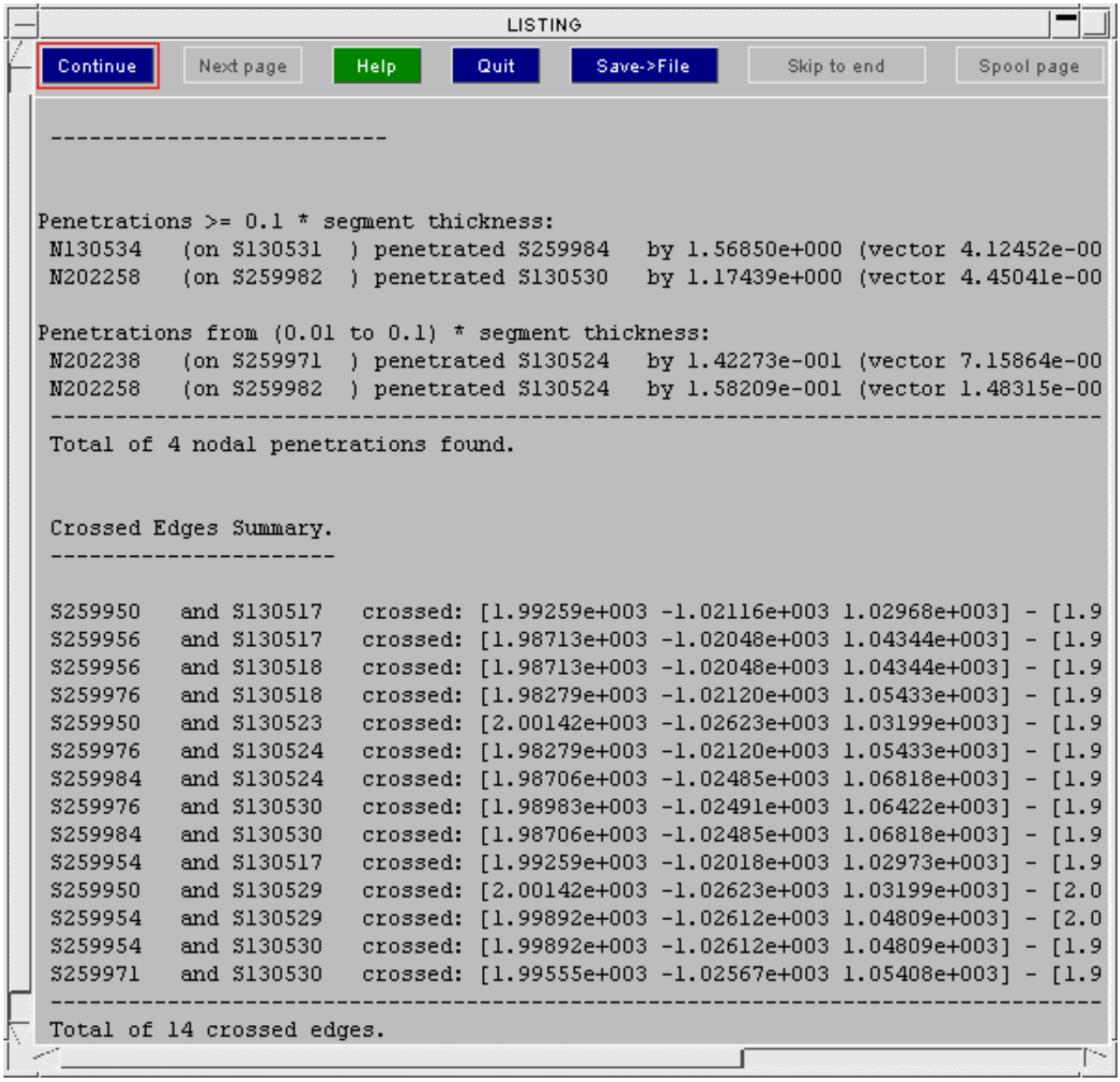
Beam Part id By default each set of null beams will be given new **`*PART, *SECTION_BEAM`** and **`*MAT_NULL`** definitions. Each will be given the "next free" label in its labelling sequence. If you want your beams to be generated in a specific part instead then define it here.

1st beam label By default beams will be generated using the "next free" beam label onwards. To generate them starting at a specific value (eg 100001) simply define the value: this may make them easier to identify.

Add to SET_BEAM By default the beams will not be placed in any sets. They may be easier to identify, and delete later when no longer required, if in a **`*SET_BEAM`** definition. If you want to add them to one simply give its label: it will be created if it doesn't already exist.

Once any settings have been made **GENERATE** will create the beams. All nodes used for beams will be new nodes, starting at the "next free" node label.

5.3.7 LIST_ERRORS: Listing penetrations and edges to screen and file.




```
-------------------------


Penetrations >= 0.1 * segment thickness:
 N130534   (on S130531  ) penetrated S259984   by 1.56850e+000 (vector 4.12452e-00
 N202258   (on S259982  ) penetrated S130530   by 1.17439e+000 (vector 4.45041e-00

Penetrations from (0.01 to 0.1) * segment thickness:
 N202238   (on S259971  ) penetrated S130524   by 1.42273e-001 (vector 7.15864e-00
 N202258   (on S259982  ) penetrated S130524   by 1.58209e-001 (vector 1.48315e-00
 ---------------------------------------------------------------------------------
 Total of 4 nodal penetrations found.


 Crossed Edges Summary.
 ----------------------

 S259950   and S130517   crossed: [1.99259e+003 -1.02116e+003 1.02968e+003] - [1.9
 S259956   and S130517   crossed: [1.98713e+003 -1.02048e+003 1.04344e+003] - [1.9
 S259956   and S130518   crossed: [1.98713e+003 -1.02048e+003 1.04344e+003] - [1.9
 S259976   and S130518   crossed: [1.98279e+003 -1.02120e+003 1.05433e+003] - [1.9
 S259950   and S130523   crossed: [2.00142e+003 -1.02623e+003 1.03199e+003] - [1.9
 S259976   and S130524   crossed: [1.98279e+003 -1.02120e+003 1.05433e+003] - [1.9
 S259984   and S130524   crossed: [1.98706e+003 -1.02485e+003 1.06818e+003] - [1.9
 S259976   and S130530   crossed: [1.98983e+003 -1.02491e+003 1.06422e+003] - [1.9
 S259984   and S130530   crossed: [1.98706e+003 -1.02485e+003 1.06818e+003] - [1.9
 S259954   and S130517   crossed: [1.99259e+003 -1.02018e+003 1.02973e+003] - [1.9
 S259950   and S130529   crossed: [2.00142e+003 -1.02623e+003 1.03199e+003] - [2.0
 S259954   and S130529   crossed: [1.99892e+003 -1.02612e+003 1.04809e+003] - [2.0
 S259954   and S130530   crossed: [1.99892e+003 -1.02612e+003 1.04809e+003] - [1.9
 S259971   and S130530   crossed: [1.99555e+003 -1.02567e+003 1.05408e+003] - [1.9
 ---------------------------------------------------------------------------------
 Total of 14 crossed edges.
```

This example shows the listing of penetrations and crossed edges generated for the model above.

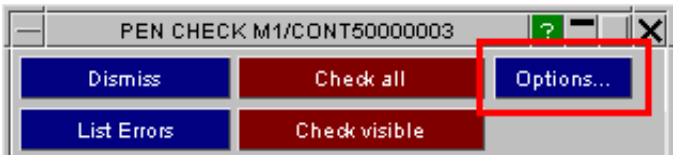
Nodal penetrations are grouped into the categories

- > 0.1 * Segment thickness
- From 0.01 to 0.1 * Segment thickness
- < 0.01 * Segment thickness

Crossed edges are simply listed in the order detected.

This listing is sent to the standard screen listing panel, and it may also be saved to file using the **SAVE->FILE** option.

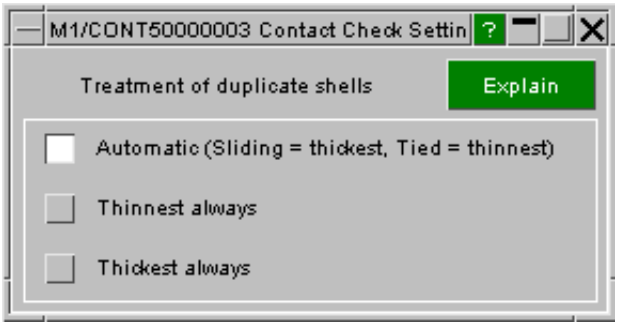
5.3.8 Options... Controlling penetration checking



Treatment of duplicate shells.

This option controls which shell is used to determine contact properties when a contact segment lies on two or more coincident shells.

This requires a little explanation:



When LS-DYNA receives a list of elements, segments, parts or sets to define the geometry of a contact surface it uses them as follows:

- It builds a list of 3 or 4 noded segments from the shells or faces of 3D elements. (Incoming segments are used verbatim)
- It culls any duplicate segments (identical topology) from this list
- Then it searches through the whole model (not just the elements defined for contact) to find shells or 3D element faces under these segments.
- The chosen element under each segment is used to determine the its thickness and stiffness properties.

Clearly two ambiguous cases can arise when finding "the element under the segment":

1. A segment lies on a 3D element face that is also overlaid by a shell.

   In this case LS-DYNA prefers the shell element unless the I2D3D flag on *CONTACT optional control card B is set to 1, in which case the solid element is preferred. The contact checker in PRIMER examines the I2D3D flag and applies the same logic.

2. A segment lies on a shell that is one of two or more duplicate shells sharing the same topology.

   In this case the shell element used by LS-DYNA is not determinate and it is not currently possible to state which shell element will be used for contact properties. Tests also suggest that the choice in the SMP and MPP versions may be different, so that if any initial penetrations exist they may be computed differently.

In order to address the second of these two cases PRIMER uses this option to define behaviour explicitly when a segment lies on duplicate shells, using one of three settings:

Automatic (default)	In this mode the thinnest shell is preferred for "Tied" contacts, and the thickest for all other types. This approach is chosen to give conservative behaviour since "Sliding" types may over-report penetrations when compared with LS-DYNA, and "Tied" types may under-detect ties. Therefore the results of a contact check in PRIMER may be pessimistic when compared to initialisation in LS-DYNA, but it will err on the side of safety. This is the default setting.
Thinnest always	The thinnest shell is always used.
Thickest always	The thickest shell is always used.

The default value of this option may be varied by the preference:

`primer*contact_penchk_dup_shells:` with the possible values **`automatic`**, **`thinnest`** or **`thickest`**

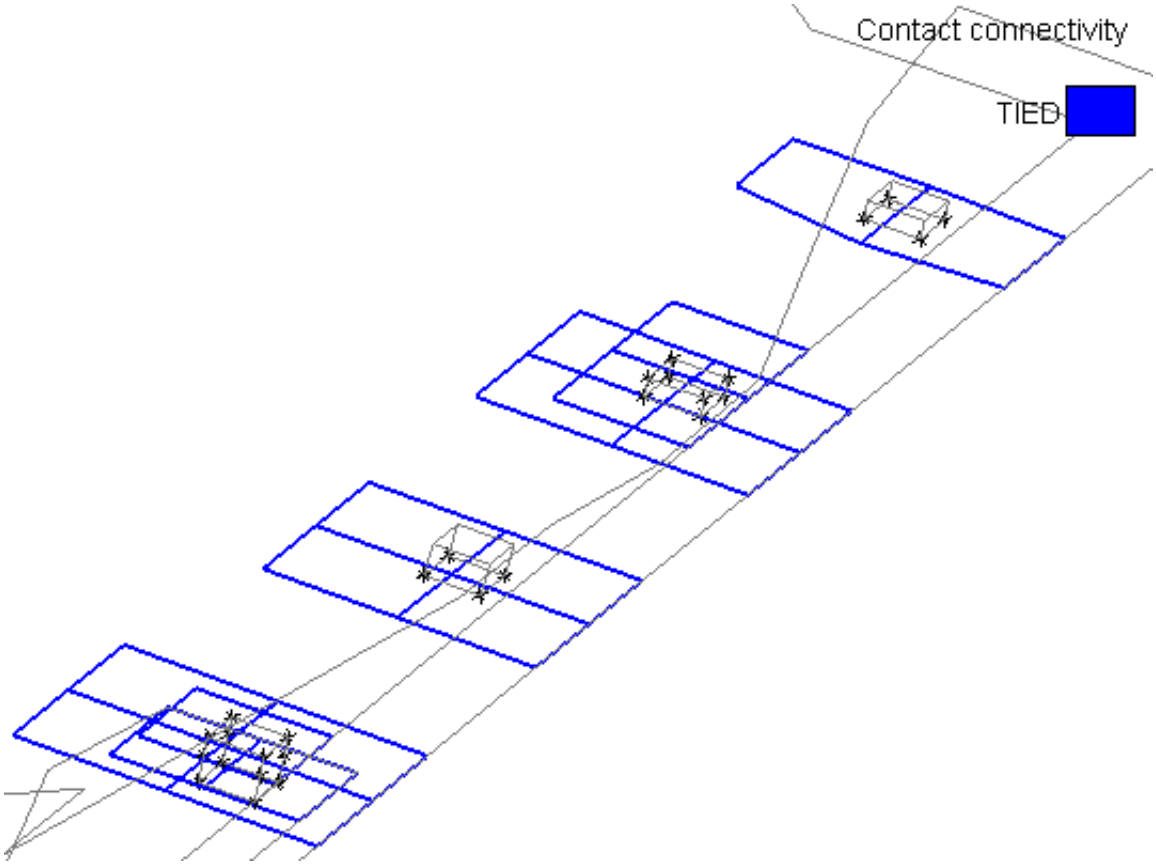
This was the status quo for Primer release 9.4, however it is likely that this setting - or at least the logic behind its default "automatic" option - will evolve as LS-DYNA develops to remove the ambiguity in the choice between duplicate shells.

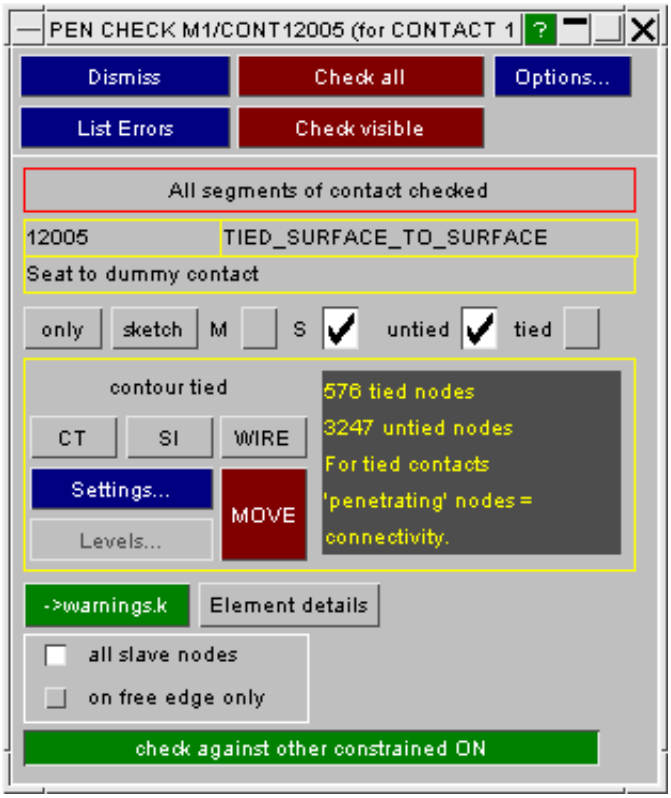
5.3.9 Checking a Tied Contact

The following contact types are treated as tied contacts in Primer:

```
*CONTACT_TIED_SURFACE_TO_SURFACE
*CONTACT_TIED_SURFACE_TO_SURFACE_FAILURE
*CONTACT_TIED_SHELL_EDGE_TO_SURFACE
*CONTACT_TIED_NODES_TO_SURFACE
*CONTACT_TIEBREAK_ SURFACE_TO_SURFACE
*CONTACT_TIEBREAK_ NODES_ONLY
*CONTACT_TIEBREAK_ NODES_TO_SURFACE
*CONTACT_SPOTWELD
*CONTACT_SPOTWELD_WITH_TORSION
```

For tied contacts, penetration of the slave node into the master segment means the node is tied.





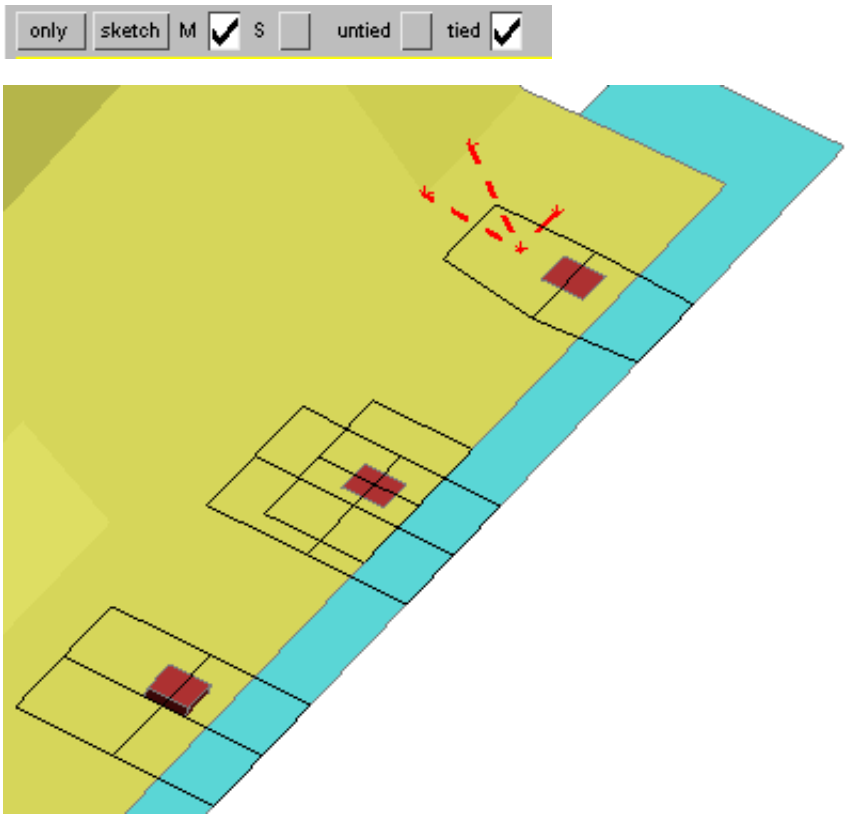
When contouring tied contacts, the tied node is sketched and blue is used to denote the segment to which it ties.

Primer will also the report the count of tied amd untied nodes. Some tied contacts may be expected to have zero untied nodes (e.g. those used for spotwelds) others may be using LS-Dyna's geometric tolerancing and intentionally contain untied nodes on slave side.

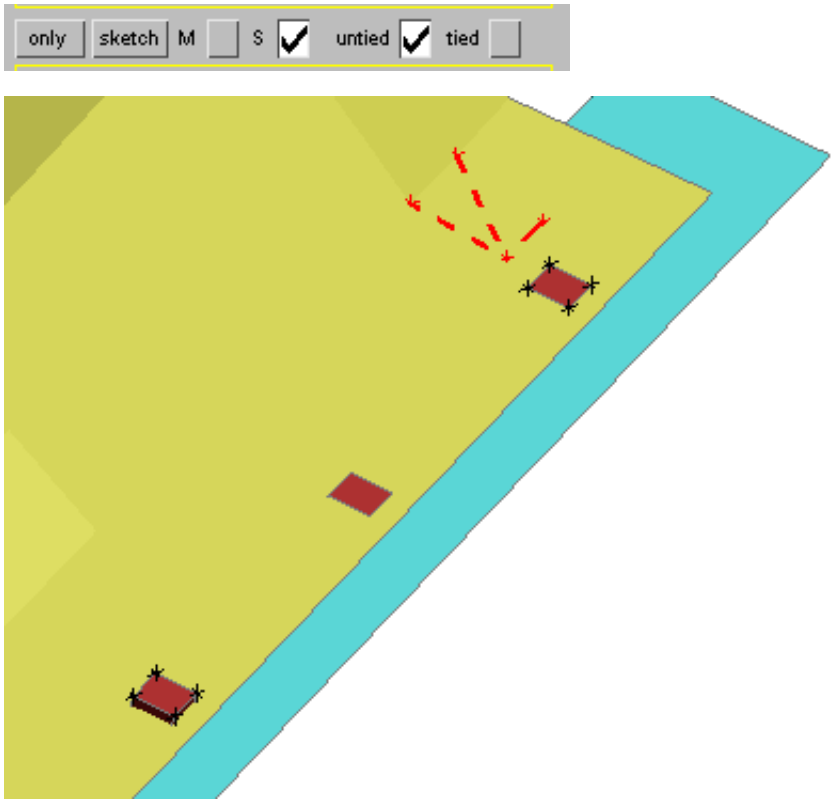
all slave node/free edge only option has no relevance for tied contact checking, it applies for MOVE option and only when slave nodes on shells

sketch & **only** - allow user to visualize what is **tied** and/or **untied** on master (**M**) and/or slave(**S**) side of the contact, according to which option is ticked. These settings do not affect the contour plot or node count.

all slave nodes/on free edge only - this option only applies if the slave side consists of nodes on shells. If set to **on free edge only** the reported count of untied nodes and **sketch** & **only** functions will only consider slave nodes on shells on free edges.



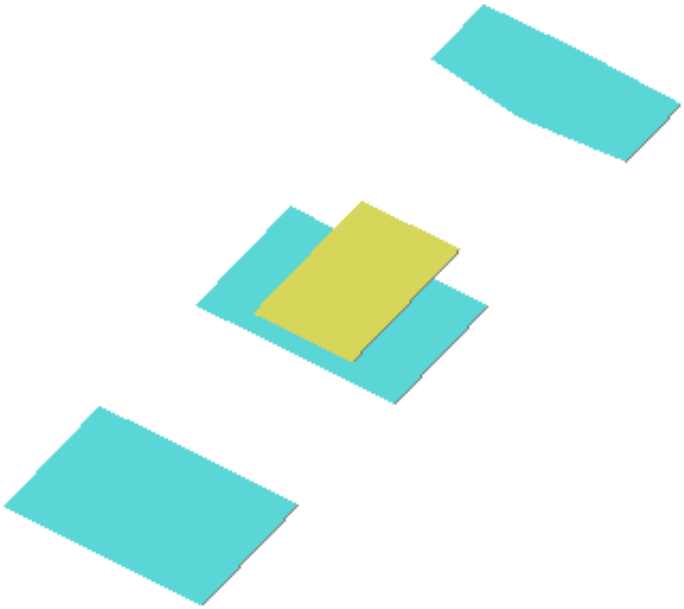
sketch applied to what is tied on the master side



sketch applied to what is untied on the slave side

some nodes do not tie because this is a constrained contact and the nodal rigid body shown interfers with it, others because they are too far away from their segment

only applied to what is tied on the master side



->warnings.k - for tied contacts this function will write untied nodes to a node set, appropriately named, in include file *warnings.k*. Untied elements will also be written to a set.

Check against other constrained - by default if this is a constrained (not penalty) contact and if other constrained contacts exist in the model, Primer will check for clashes with other constrained contacts. A slave node cannot be tied successfully to a segment, if another contact ties to this segment or a segment which shares a node with this one. You can turn this option off in CHECK > OPTIONS > CONTACT.

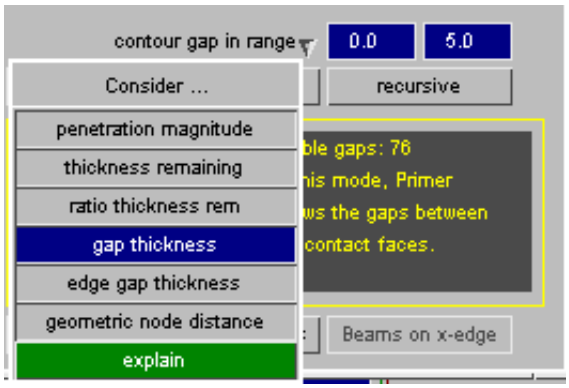
5.3.10 Notes on Contact Penetration Checking.

- The checking algorithms used in PRIMER aim to mimic those in LS-DYNA, but they are not identical. You must expect that the penetrations detected will differ slightly, although they should agree well in most cases. An exception is self-contact where LS-DYNA's initialization process of depenetrating nodes as it finds them will always yield appearance of less penetrating nodes (in otf file) than Primer's static geometric approach.

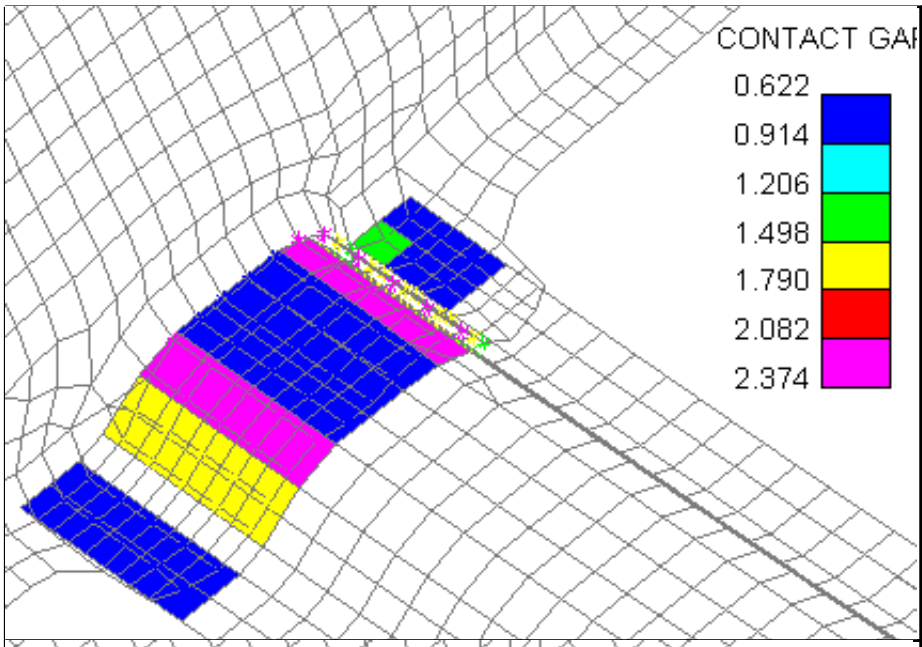
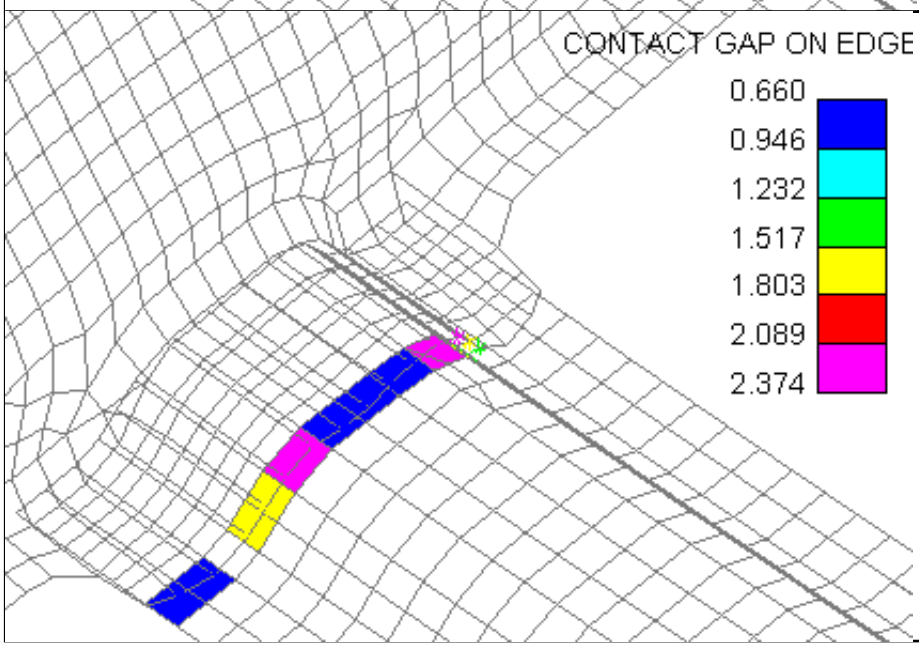
- It is a known "feature" of LS-DYNA that the contact penetrations reported by the SMP and MPP versions are different in some cases. Where a contact has the MPP flag set PRIMER knows it is destined for the MPP version and will use the relevant logic, otherwise where differences between the two versions of LS-DYNA are known to exist it will use the current penetration checking mode as set in the Check, Options panel, which defaults to MPP mode.
- When "old type" segment-based contacts (non-automatic surface to surface, nodes to surface and singe surface) are used, the default penetration depth "behind" a segment in LS-Dyna is "infinite" (= 1e20 or thereabouts). For efficiency in its bucket sort Primer limits the depth behind such a segment to be the longest distance from a node on it to an imaginary box containing the contact. Therefore "as_thick" plots, and reported penetration distances, will be limited to this value.
- Certain 2D geometries are not yet checked, these are:
  - "Edge to ..." contacts: Shell or segment edge contact is not considered
  - "Beam to ..." contacts: Contact along the length of a beam is not considered.

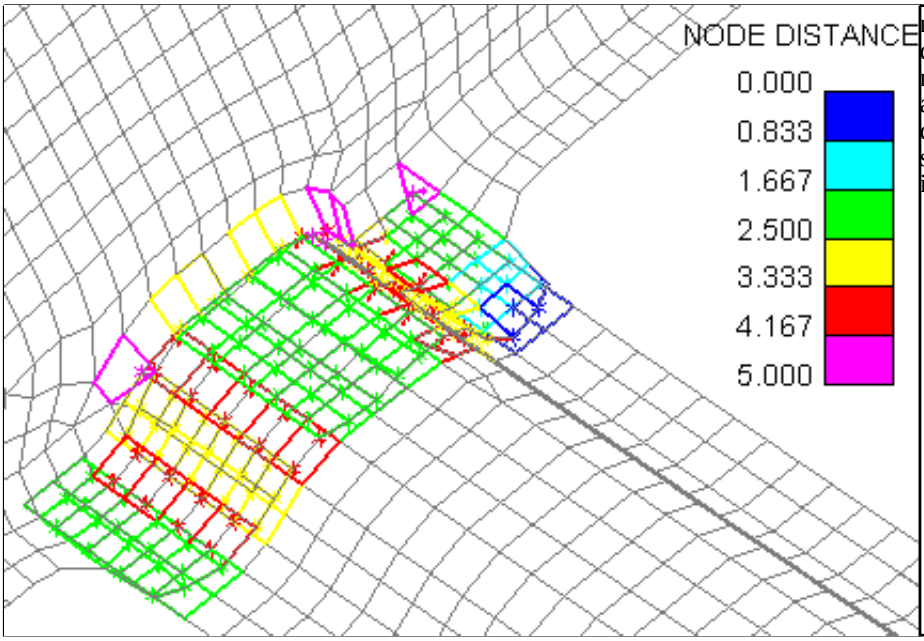
  However the (one way) penetration of nodes on edges and beams into segments is considered.

5.4 Contouring panel gaps for sliding contact



Penetration check module is now capable of contouring gaps in a user defined range between panels. Three modes are available. In gap thickness mode the **FIX** function becomes a **MOVE** function which removes gaps.

	gap thickness - will contour the gap between slave side nodes and master side shells for surface-surface contact. In single surface contact every node is treated as a slave node.
	edge gap thickness - as above but will only consider slave nodes on shells on a free edge



node distance - measures the distance of each slave node to mid-plane of its master segment and reports cases where the distance is less than the user specified value. This test is independent of contact thickness

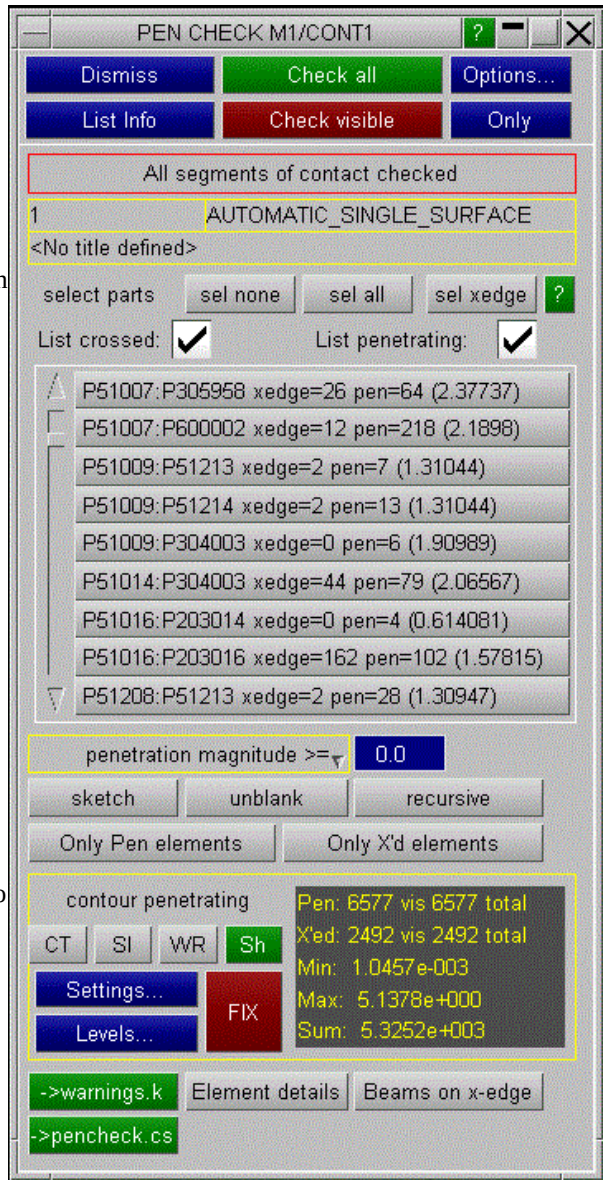
5.5 Contact Penetration Fixing

- Accessing the penetration fixing panel
- Correcting crossed edges
- Correcting penetrations

Primer can help fix initial penetrations and crossed edges in contact surfaces. This capability can be invoked via the penetration checker window accessed via:

CHECK (from Tools) > RULES
CONTACT > PEN_CHECK
CONTACT > CREATE/EDIT, PEN_CHECK

5.5.1 Penetration Fixing panel



The pen check panel shown will provide data on the number of crossed edges and penetrations in the model. For more information on penetration checking, see section 6.10 Contact Penetration Checking.

The penetration checker panel allows you to easily correct any crossed edges and initial penetrations in your model. This option can be accessed by pressing the fix button.

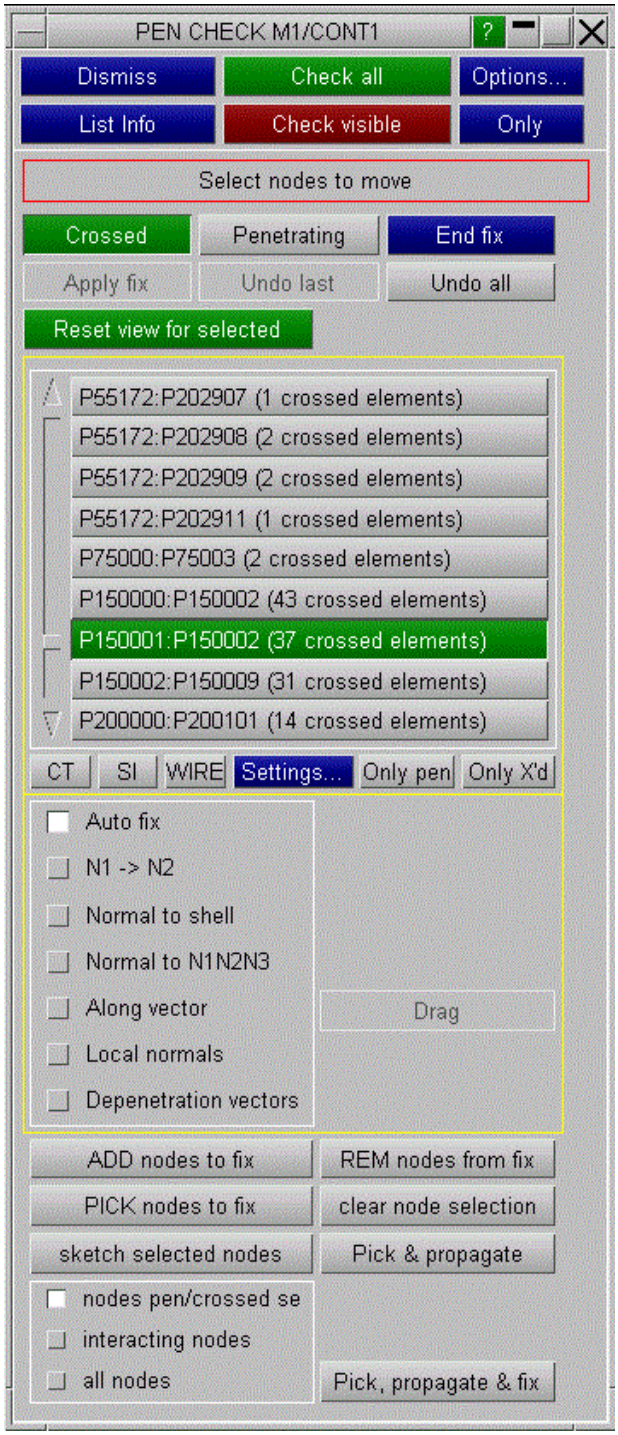
By default fixing is only applied to *visible entities* so you can use visibility control of the check panel to determine what is to be fixed. The crossed edge fixing panel then offers further visibility control As these need to be fixed one-on-one). If you pre-select *all the interacting parts or none*, all crossed edges will be up for consideration. Normally, it is recommended that you **remove all crossed edges** before attempting to remove initial penetrations.

The list of pairs of interacting parts can be filtered by unticking **List crossed** or **List penetrating**.

Penetrations and Crossed edges can be removed using both automatic and manual methods. The Penetration checker can be used to check that all initial penetrations have been removed.

Penetration mode. Fixing will be applied using the *active check penetration mode*.The setting both limits to which nodes the fix is applied (e.g. ignore penetrations below a certain value) and controls the magnitude of the fix (e.g. setting **ratio thickness rem** < 0.7 will mean that *minimal depenetration* is performed to meet this target)

5.5.2 Correcting crossed edges

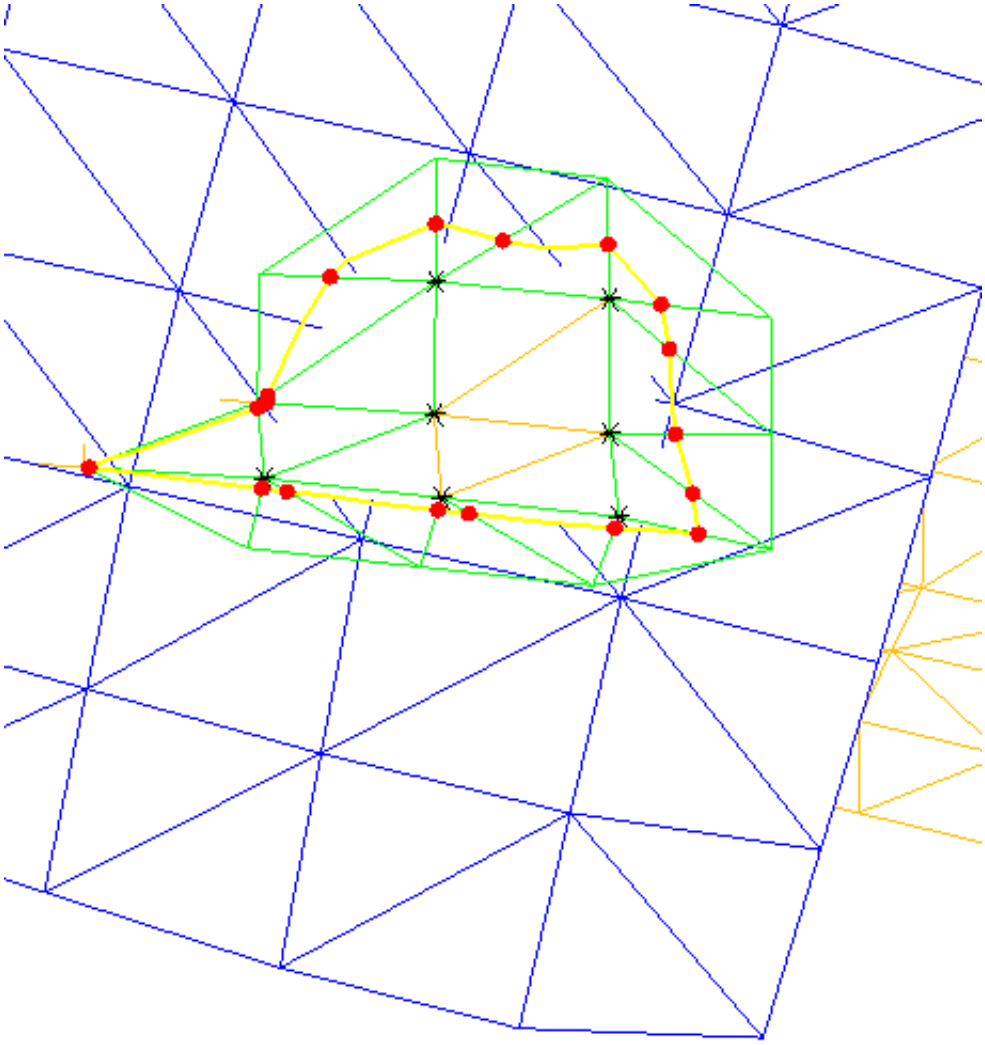


In the penetration fixing panel, select the CROSSED option. The panel will display all pre-selected crossed panels present in pairs as shown. Select the pair of crossed panels you wish to correct. Primer will highlight the 2 panels and show where the crossed edges exist and the fixing options will ungrey.

Primer allows you to correct crossed edges by moving selected nodes. **ADD nodes to fix** will allow you to add nodes to the selection, either by elements or directly by nodes. By default the object menu will limit the selection to nodes of crossed segments, but you may switch this to consider **all visible nodes** which will enable you in manual modes to smooth the mesh. **PICK nodes to fix** allows you to pick nodes directly from the screen without opening the object menu.

For some cases the **Pick & propagate** method may be suitable. This will apply if the crossed edge is a closed loop or runs up to a free edge and so a single pick of a node on the side of the edge to be moved can be propagated to determine all the nodes which should be moved.

Pick & propagate will detect the closed path of crossed edges and make the nodes selection.



The direction in which the selected nodes are moved can be specified by using one of the options

AUTO FIX

N1 -> N2

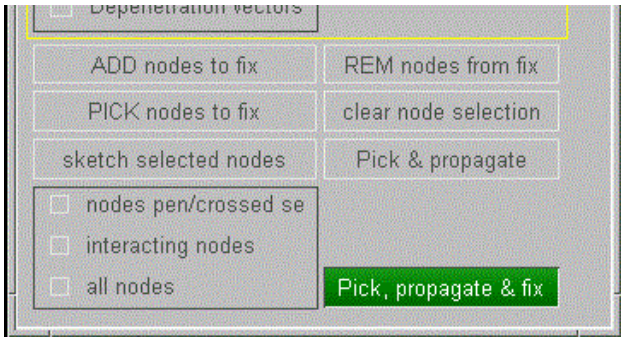
NORMAL TO SHELL

NORMAL TO N1N2N3

ALONG VECTOR

LOCAL NORMALS

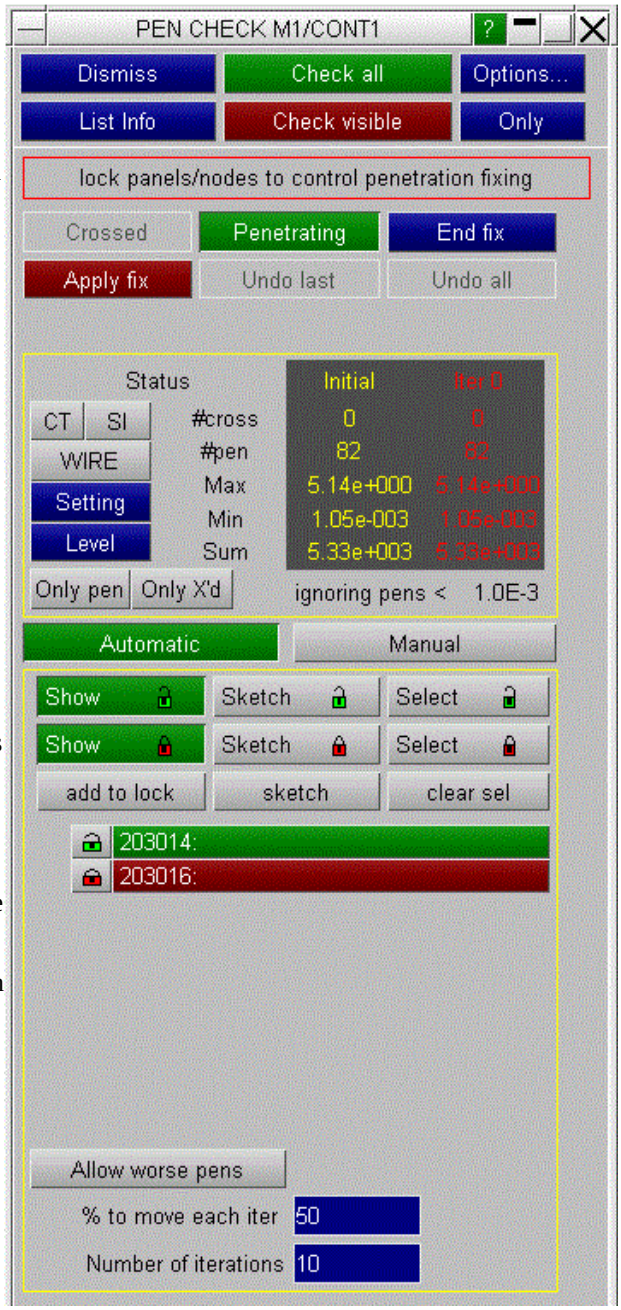
DEPENETRATION VECTORS



For closed loop cases a fully automatic fixing mode is available. **Pick, propagate & fix** will make the selection (as **Pick & propagate** above) and apply an iterative fixing function which attacks the wave front of the crossed edge until no nodes remain to move. Pressing this button disables all other fixing functionality and those buttons will remain greyed out until this one is unpressed.

When the desired fixing method has been set, press the **Apply fix** button in the Autofix case. In the other modes type in or drag a distance. Press **End fix** (as appropriate) to lock the change in.

5.5.3 Correcting initial penetrations



In the penetration fixing panel, select the **PENETRATING** option. The panel will display information on the number of crossed edges, the number of initial penetrations which exceed the define penetration parameter, the maximum and minimum penetration and the sum of all penetrations. Primer offers both automatic methods and manual methods for fixing penetrations.

When the desired fixing method has been defined, press the **APPLY FIX** button.

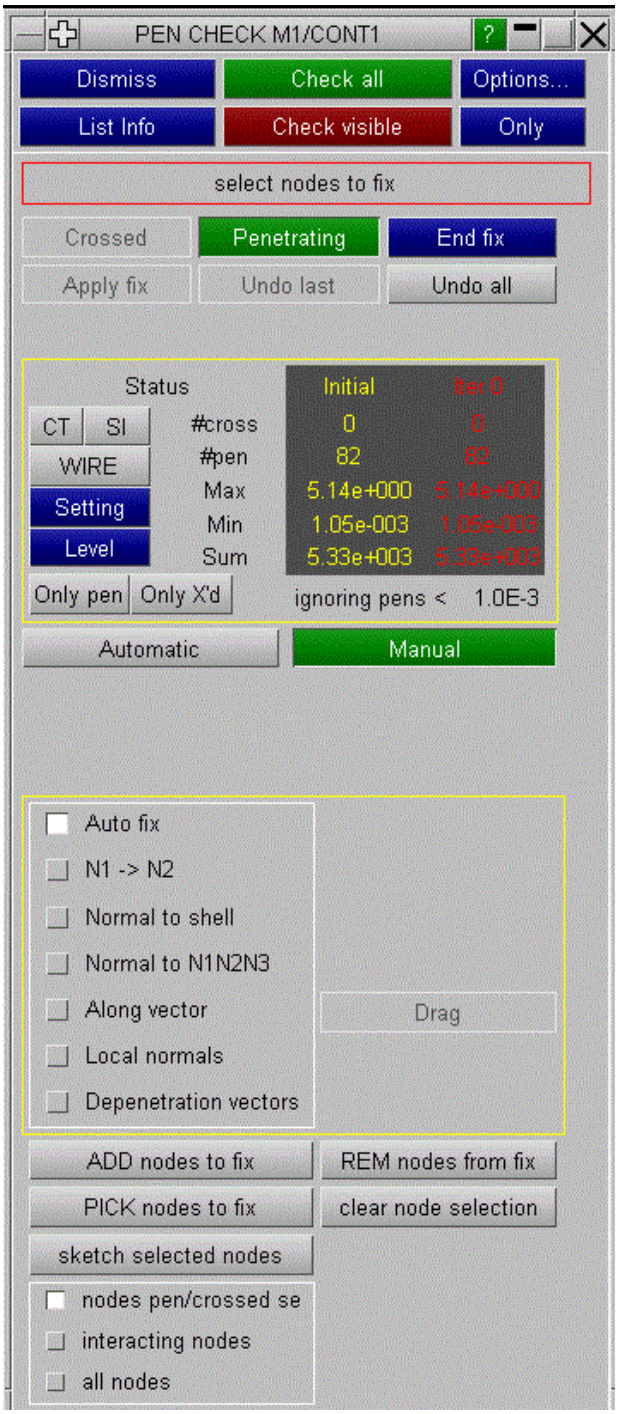
5.5.3.1 Automatic Fixing

This can be accessed by selecting the **AUTOMATIC FIXING** tab. Parts can be locked (indicated by a red, closed padlock) and unlocked (indicated by a green, open padlock) by clicking on the padlock tabs. Those parts with crossed edges will be automatically locked. Only unlocked parts will be moved in the fixing process. Only unblanked entities will be moved in the automatic fixing process. **add to lock** can be used to lock nodes additionally the the panel lock.

The Automatic Fixing method uses an iterative method. Nodes are moved only a percentage of the penetrating distance. To specify this percentage, enter the desired percentage in the % to move each iter box. The number of iterations to be carried out can be specified in the Number of iterations box. In some cases it may be necessary to allow worse penetrations initially in order to fix all penetrations in the model. If this could be the case, select the **Allow worse pens** tab.

Nodes will be fixed sufficiently to meet the current penetration criterion, this may be less than full depentration.

5.5.3.2 Manual Fixing



This option can be accessed by selecting the **MANUAL FIXING** tab.

Fixing is applied to selected nodes.

The direction in which the selected nodes are moved can be specified by using one of the options;

AUTO FIX

N1 -> N2

NORMAL TO SHELL

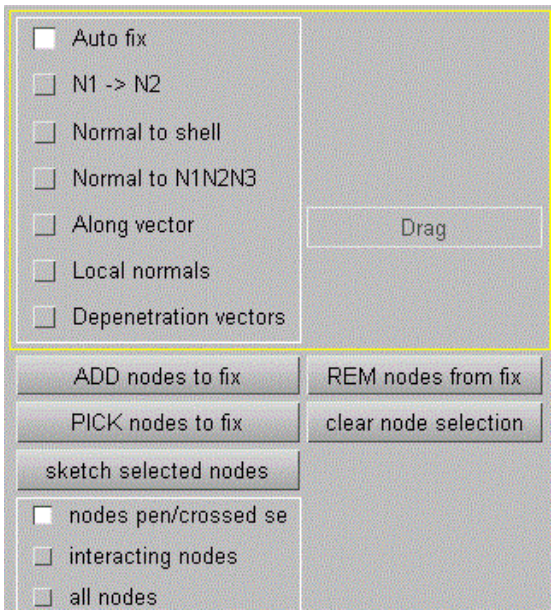
NORMAL TO N1N2N3

ALONG VECTOR

LOCAL NORMALS

DEPENETRATION VECTORS

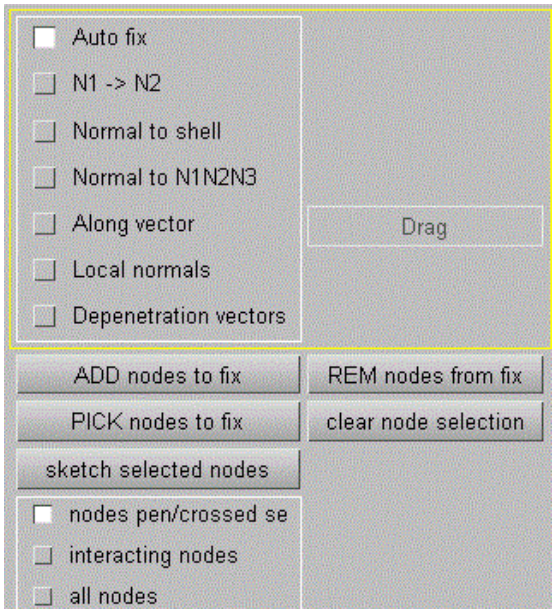
Selecting nodes to fix



Primer allows you to correct initial penetrations by moving nodes selected by the **nodes to fix** function. By default the selection is limited to penetrating nodes, but you may select other nodes for manipulation to smooth the mesh changes by using **consider all nodes**.

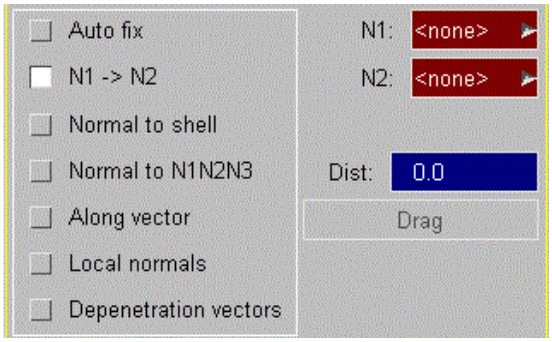
The direction in which the nodes are to be moved can be specified by using the options; N1 -> N2, NORMAL TO SHELL, NORMAL TO N1N2N3, ALONG VECTOR, or alternatively the AUTO FIX option can be selected.

Auto fix



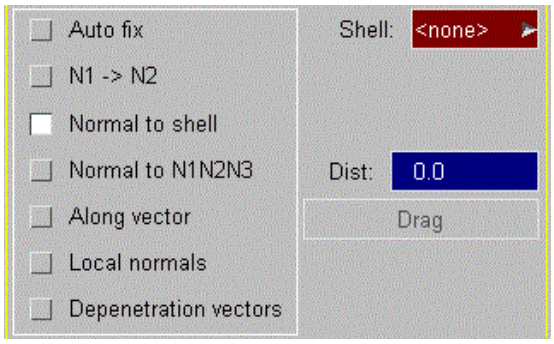
This option will move the selected node or nodes normal to the penetrating shell in order to remove the penetration.

N1 -> N2



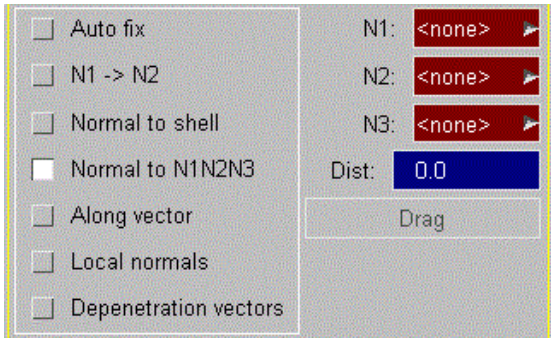
The selected node or nodes will be moved along a vector defined by 2 nodes. The nodes can be selected using any of the usual procedures. The extent to which the penetrating nodes are moved can be defined by distance, specified in the distance box, or by dragging the nodes, specified by selecting the **DRAG** button and dragging the nodes across the screen using the cursor.

Normal to shell



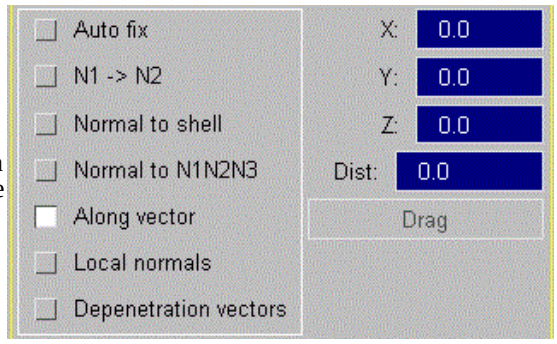
Normal to shell allows you to move the selected node or nodes normal to a shell. The desired shell can be specified using any of the usual procedures. The extent to which the selected nodes are moved can be defined by distance, specified in the distance box, or by dragging the nodes, specified by selecting the **DRAG** button and dragging the nodes across the screen using the cursor.

Normal to N1N2N3



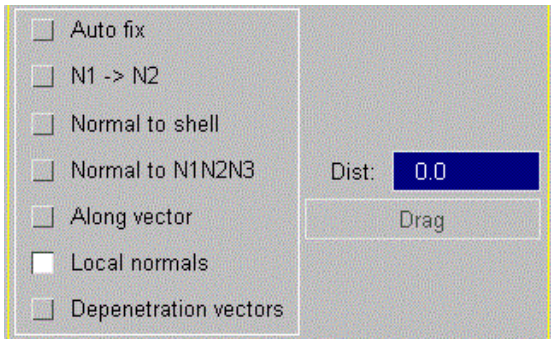
Selected nodes will be moved normal to a plane defined by 3 nodes. These 3 nodes can be selected using any of the usual procedures. The extent to which the penetrating nodes are moved can be defined by distance, specified in the distance box, or by dragging the nodes, specified by selecting the **DRAG** button and dragging the nodes across the screen using the cursor.

Along Vector



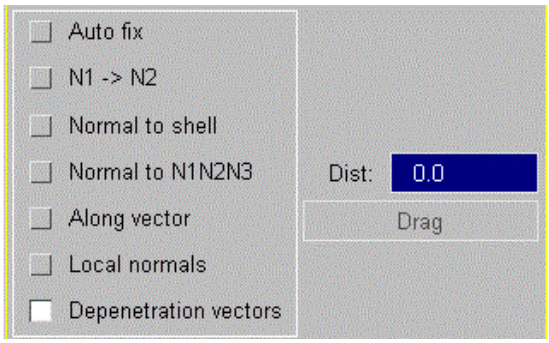
This option allows you to define a vector using co-ordinates along which the selected nodes are to be used. The extent to which the penetrating nodes are moved can be defined by distance, specified in the distance box, or by dragging the nodes, specified by selecting the **DRAG** button and dragging the nodes across the screen using the cursor.

Local normals



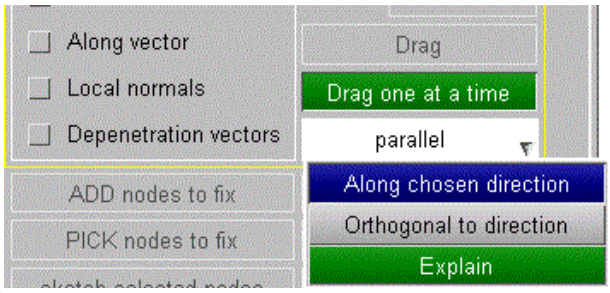
For every node a local normal will be calculated as a weighted average of the normals of adjacent shells. All selected nodes are moved by the same distance specified by the text box or by dragging, but the direction will usually be different for every node. This is useful for fixing nodes on curved parts.

Depenetration vectors



This option uses the internally calculated depenetration vectors like Autofix, but allows you to specify the distance manually in the text box or by dragging. The node with the longest depenetration vector will be moved by the indicated distance, and all other selected nodes are moved in a proportion corresponding to their depenetration vector magnitude.

Dragging modes

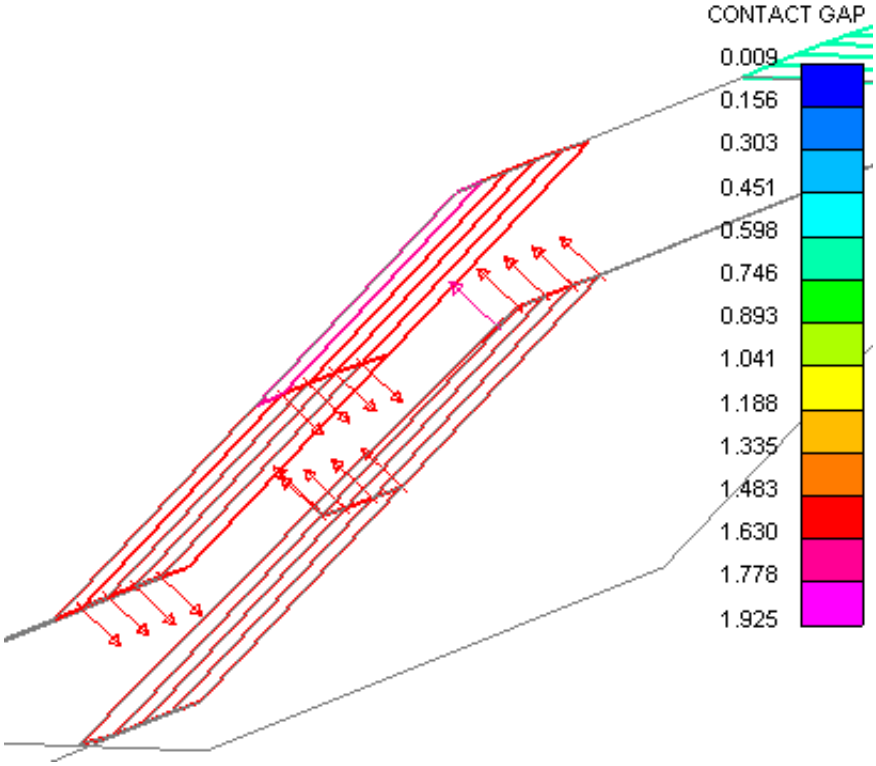


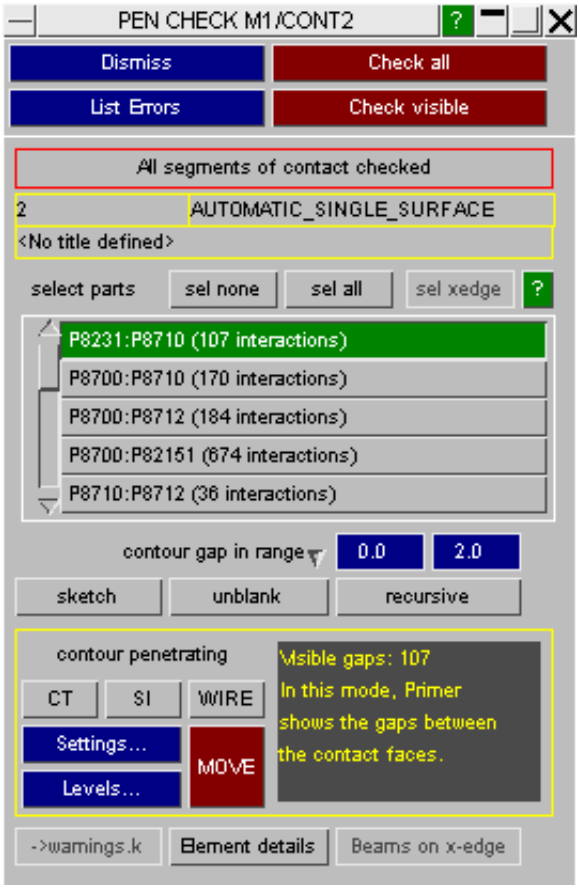
The distance which the nodes should be moved by can be specified by typing in a number into the text box. Otherwise it is possible to drag the nodes in several ways: The **DRAG** button allows you to visually determine the distance you want the selected nodes to move by continuously dragging them. **Drag one at a time** allows you to drag the nodes separately without selecting them, where the option in the radio button still determines the direction as a degree of freedom. Nodes can be moved either parallel to that direction or in a plane perpendicular to the direction.

5.6 Contact gap fixing

Primer contours gaps in the user defined range.

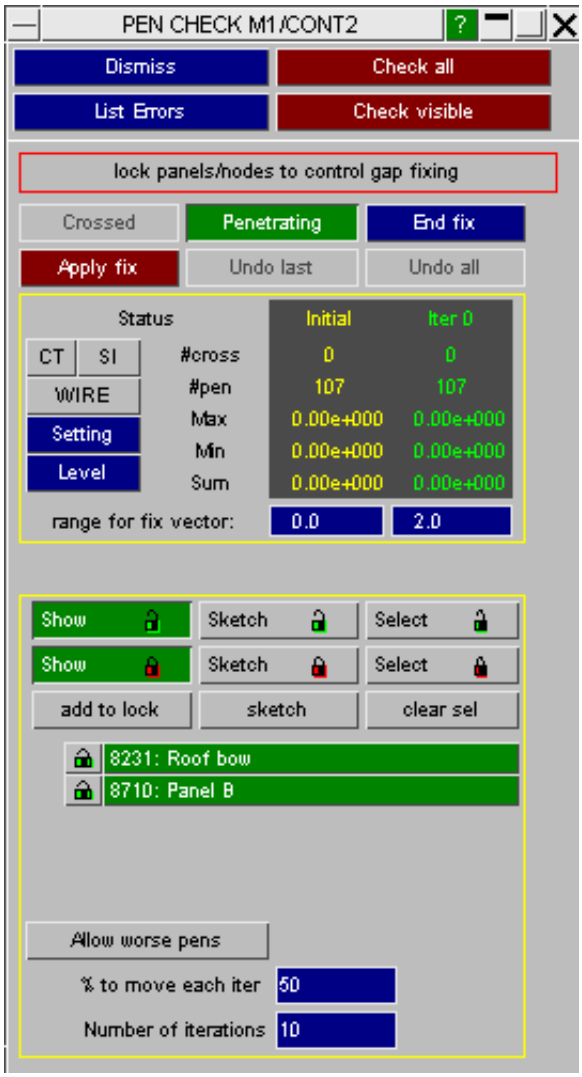
In gap mode the fixing function becomes **MOVE** - a function to remove gaps between visible segments.





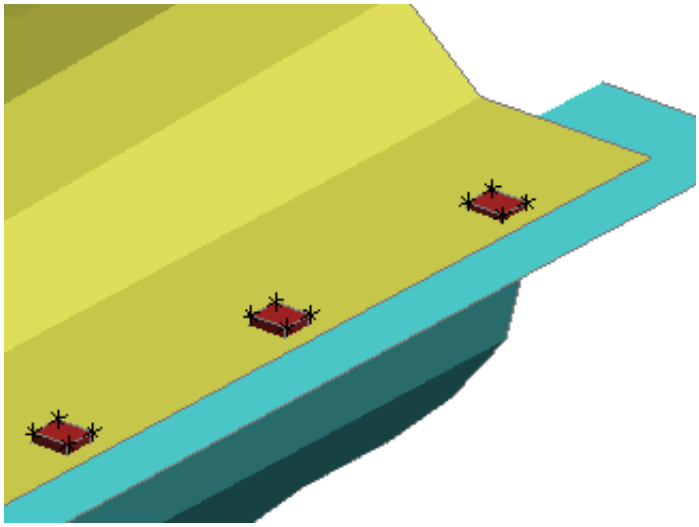
Gap fixing will observe panel and node locking in the same way as penetration fixing.

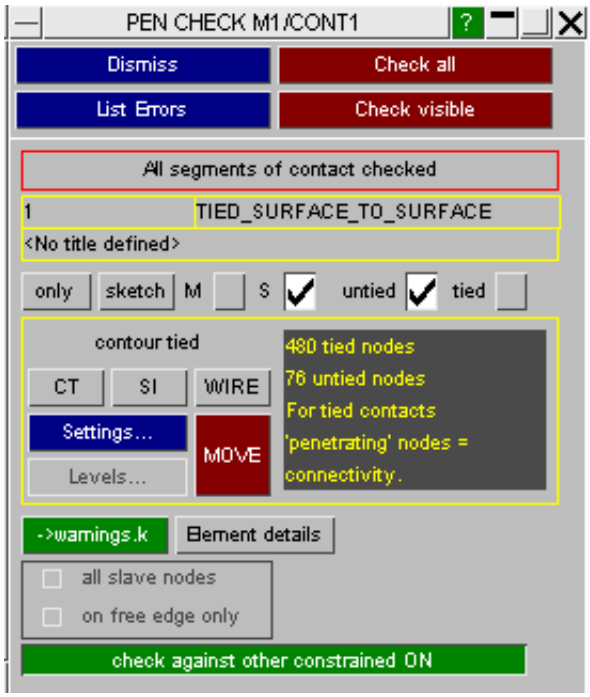
Any unlocked nodes which bear an approach vector in the CT plot will be moved along the vector iteratively, until the gaps are removed.



5.7 Tied contact fixing

When checking a tied contact the fix function becomes MOVE. This will move slave nodes to "best position".





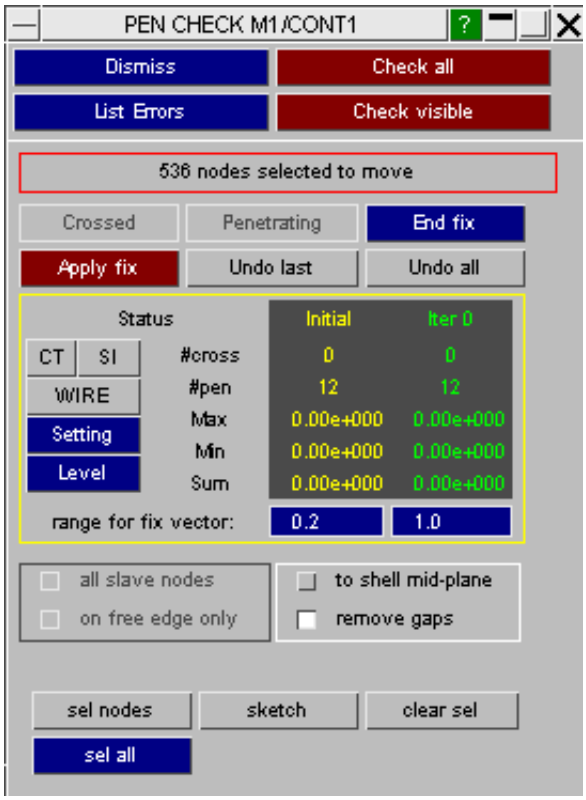
The range for fix vector needs to be set. CT plot will now show all the moveable nodes with a vector. Nodes to move must then be selected using **sel nodes** or **sel all** and **Apply fix** will become active.

to shell mid-plane - will configure the vector to move the node to the mid-plane of the shell to which it ties. This is recommended.

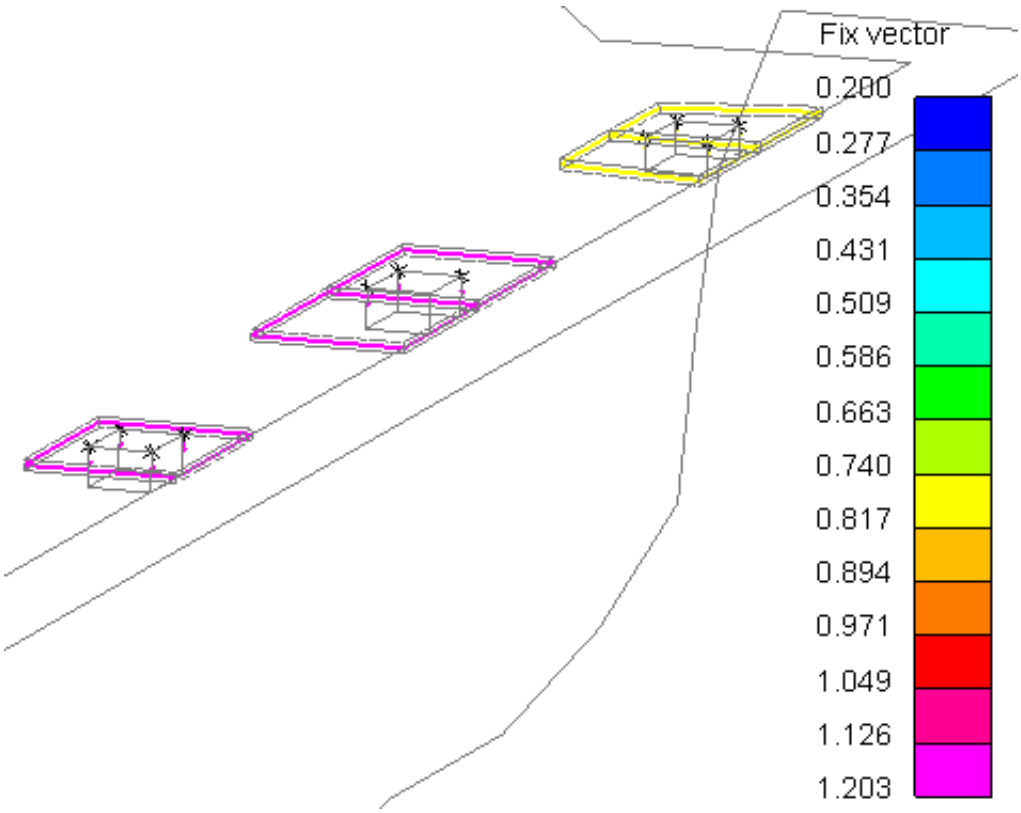
remove gaps - will configure the vector to move the node so there is no remaining gap. For offset contacts this fix may be more appropriate.

Either of the above methods will ensure that the node is in a position to tie geometrically (a node may not tie due to other reasons of course)

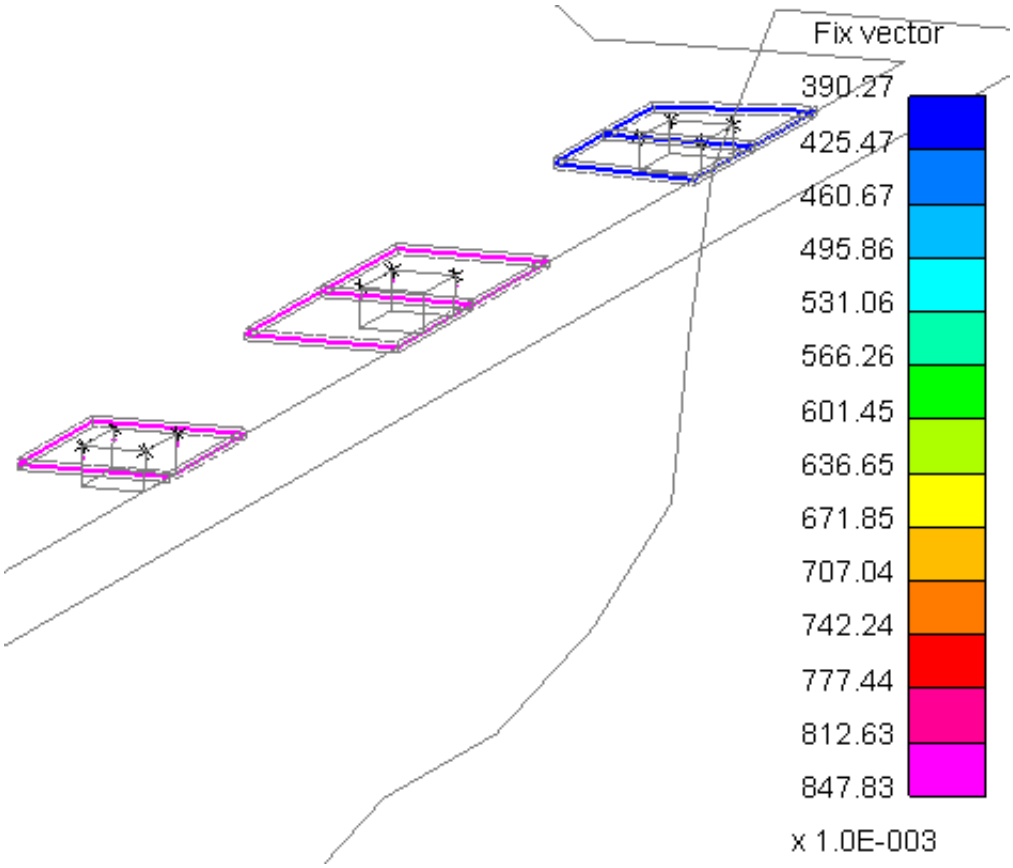
LS-Dyna has rules for determining whether or not a node will tie which depend on mesh size as well as distance off segment (see the manual entry under *CONTACT). It may be that some nodes with vectors drawn on them *are actually tied* and, therefore do not necessarily need to be moved.



Fixing to the shell mid-plane (shell thickness is shown)

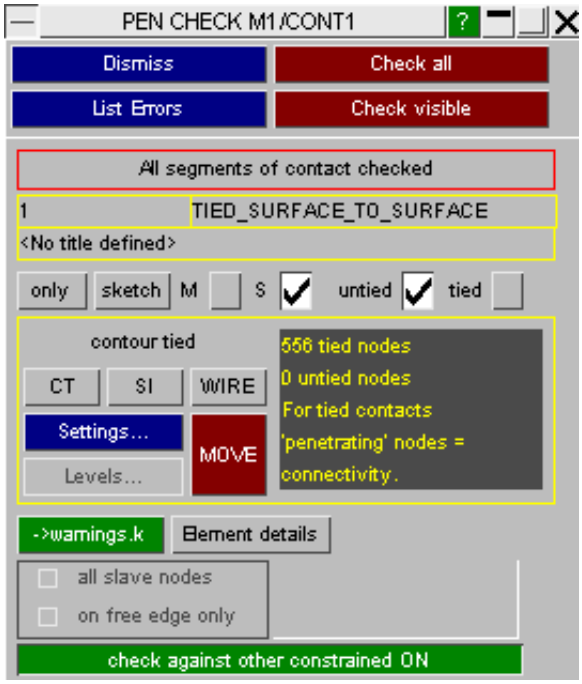


Fixing by removing the gap (shell thickness is shown)



End fix will return you to the tied contact check panel, where you can re-check the count of tied nodes

all slave nodes/**on free edge only** - the option only applies if slave nodes are on shell elements. If set to **on free edge only** it will limit the MOVE option to apply only to nodes on free edges of shells



6 Tools

6.0 TOOLS panels

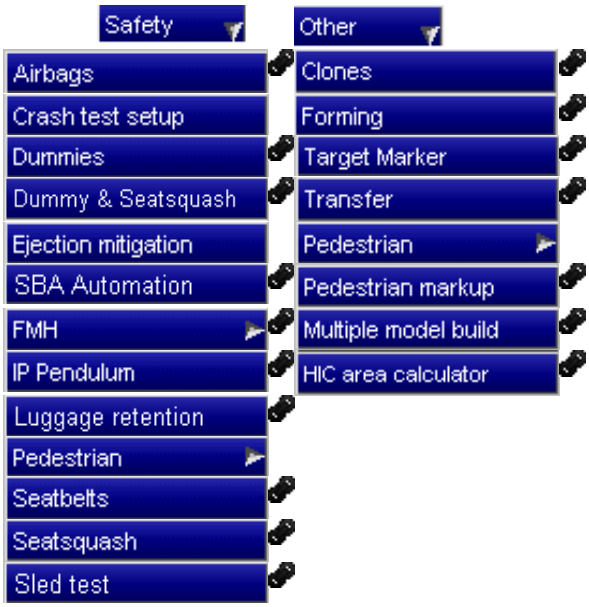
6.0.1 Tools panel

The main Tools panel provides top-level access to the following functions.

For more details click on the links below

Assign mass	Composite	Macro	***Safety***
Attached	Connection	Mass Prop	Script
Blanking	Cut Sect	Measure	Text Edit
Bill of materials	Explode	Mechanism	Units
Check	Find	Orient	X-references
Clipboard	Groups	***Other***	
Coat part	Include	Remove	
Compare	Load Path	Rigidify	

The following operations have been grouped under pull-down menus by category. For details click on the buttons or the symbols:



6.0.2 Mesh tools panel

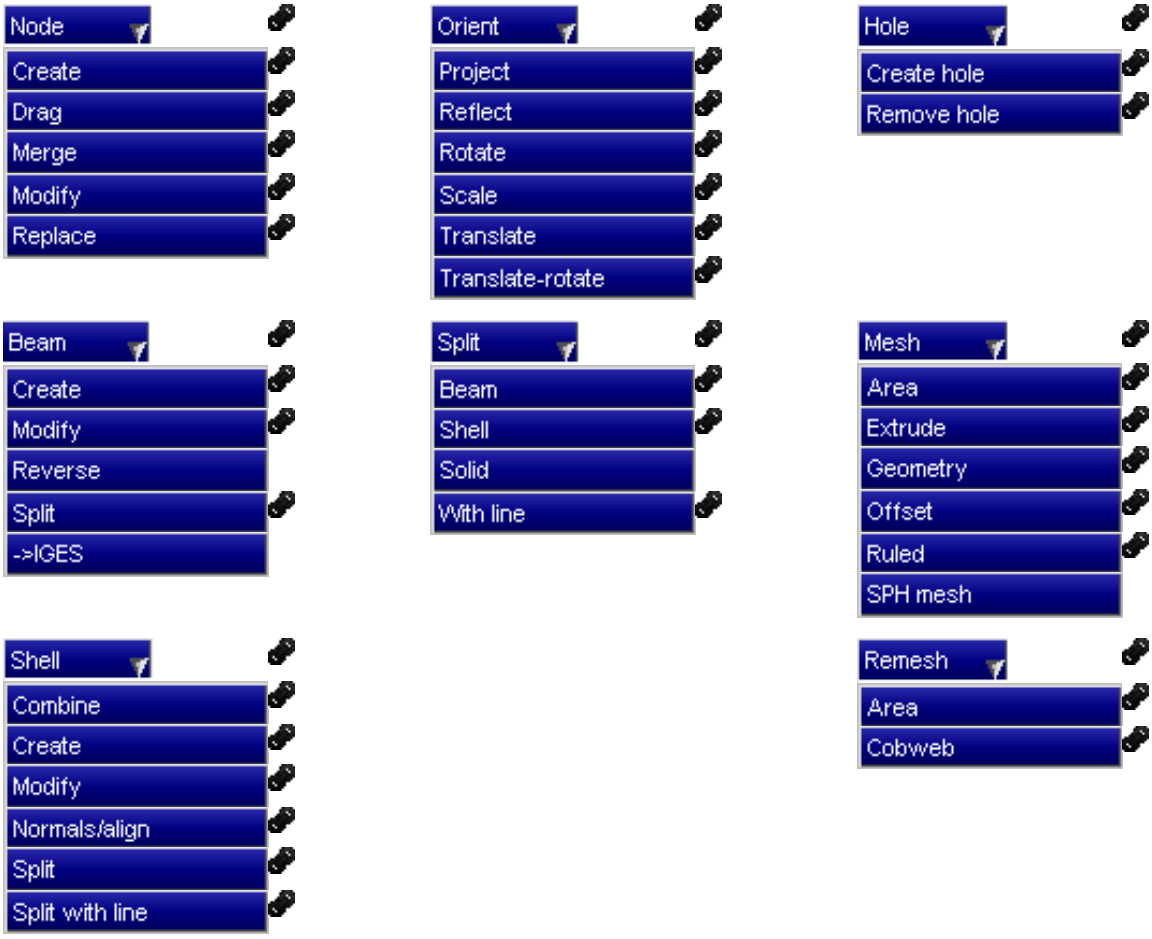
In version 12 of PRIMER the various mesh modification tools have been moved into a new **Mesh tools** panel.

For more details click on the links below

Node	Detach	Coat	Quality
Beam	Fix transition	Hole	Mesh->Surf
Shell	Orient	Mesh	Morph
Solid	Split	Remesh	Tet Mesh
TShell	Warped->tria	Shapes	

The **Tools** and **Mesh tools** buttons can be used to toggle between the two views. The other tools are available at any time by using the popups. For example the 'normal' tools are still available in the image above by right clicking on **Tools**.

The following operations have been grouped under pull-down menus by category. For details click on the buttons or the symbols:



6.0.3 POST: Linking PRIMER with Post-Processors

In Version 15 it is possible to link PRIMER with the D3PLOT and T/HIS post-processors via shared memory.

This makes it possible to open both "pre" keyword file in PRIMER and "post" results in D3PLOT and T/HIS, and to exchange data and information between the codes. In the case of D3PLOT it is also possible to synchronise graphics so that the view and otehr visual attributes in both codes match.

This is covered in more detail under section 3.17 Model POST

6.1 AIRBAGS

6.1.1 Folding Airbags

The airbag folder is designed to produce folded meshes from ones that are initially flat. Additionally, some facilities are provided to deal with 3-D initial configurations

The airbag folder can also generate an airbag mesh from scratch for some pre-defined geometries. At present this is limited to a star folded or a circular folded airbag. In future releases this functionality will be further enhanced.

During folding the airbag can be checked for distorted elements and initial penetrations. Once folded the airbag it can be positioned.

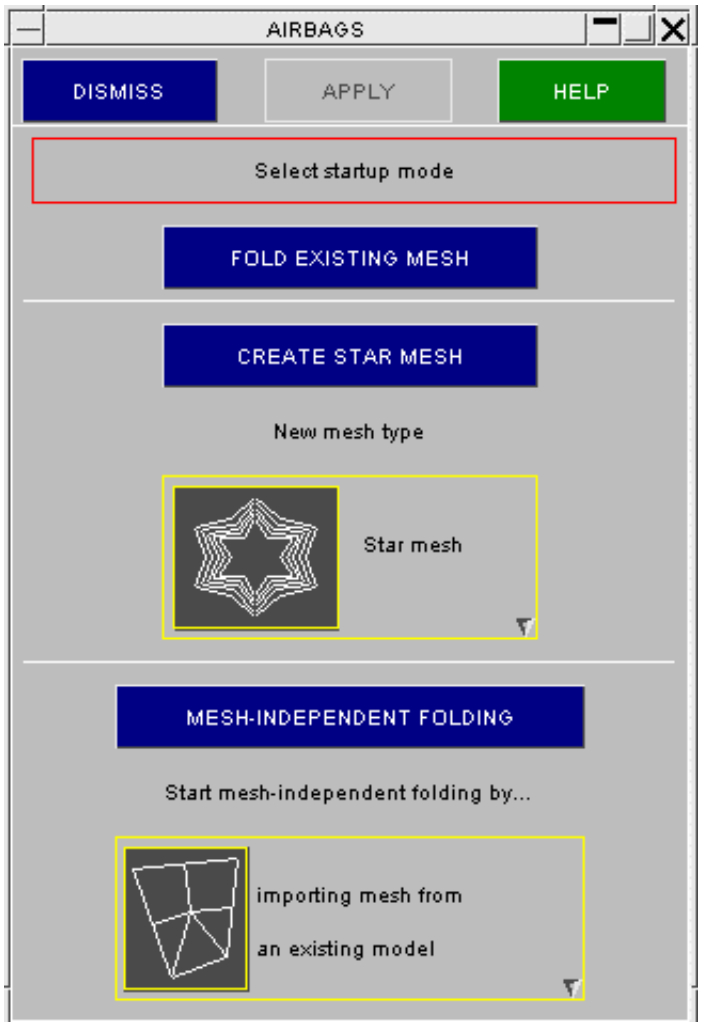
If you are starting from an existing mesh then each airbag must be a subset of only one model. Airbags usually consist of shell elements only and other element types should be avoided.

The airbag start up screen allows you to choose one of 3 possible modes.

1) Selecting **FOLD EXISTING MESH** will start the airbag folding process with an existing mesh.

2) **CREATE STAR MESH** can be used to create a new star or circular fold. To select whether to create a star fold or circular fold use the new mesh type popup box.

3) **MESH-INDEPENDENT FOLDING** allows you to create a circular airbag or import a mesh that is suitable for mesh independent folding.



Definitions: ORIGAMIs, FOLDs and ORIENTs

This section defines some of the terms which are used in the airbag folder.

The use of the term "Airbag", has the potential to cause some confusion. The actual **`*AIRBAG`** card in LS-DYNA consists of the surface of an airbag and the physical properties (gas thermodynamics) of the inflator. The tethers which might be included in a bag, for example, do not form part of the free surface of the airbag control volume, and thus they are not included in an LS-DYNA "airbag". The tethers, however, must be folded along with the remainder of the airbag, so they may need to be included in a geometrical definition.

Thus, to distinguish between the LS-DYNA **`*AIRBAG`** and the airbag as described here, the term ORIGAMI is used in

PRIMER. In fact, an ORIGAMI could potentially involve things which are completely unrelated to airbags: it is the umbrella definition containing everything required to define the geometrical extent of an airbag, and its associated folding operations:

- The origami label (which must be unique within a model) and title;
- A list of elements and nodes (as sets) which comprise the bag;
- A list of folds;
- A local coordinate system.
- A list of orientations

A **FOLD** is a generalised term for the many fold types available; eg Rolling a bag up is described as a "FOLD", as is a "Tuck" or a "Scrunch". Each FOLD definition contains:

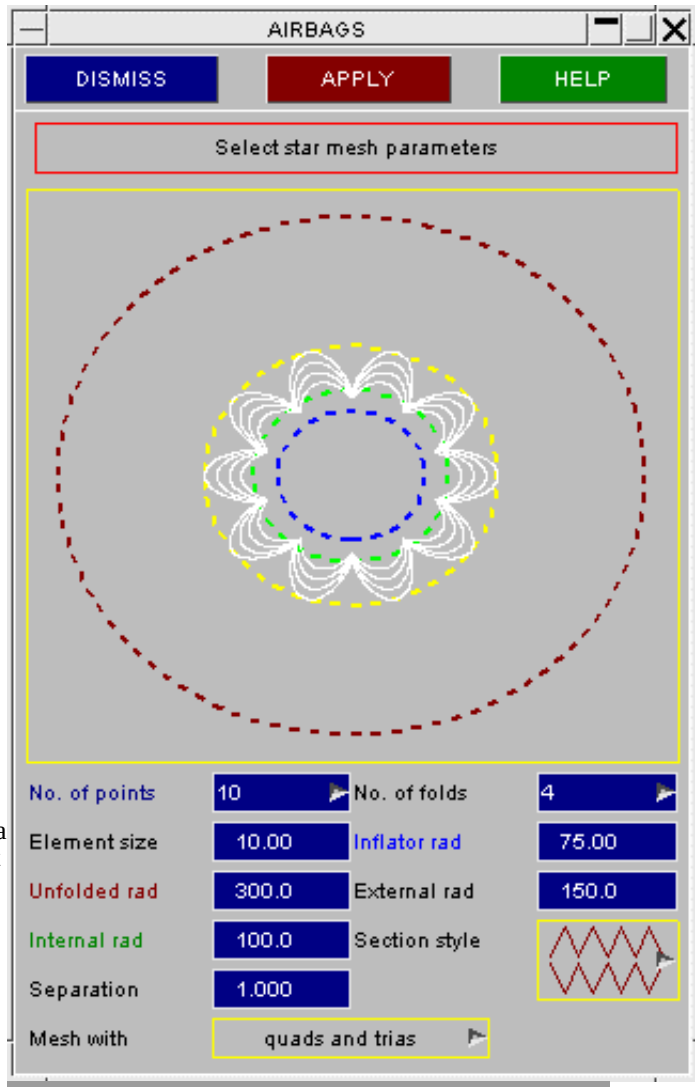
- The fold number and type (thin, thick, roll, tuck, ...)
- An optional coordinate system;
- Geometrical data (location, direction, angle, thickness, ...)
- Optional subsets of nodes and elements for special cases.

An **ORIENT** is a transformation which is applied to the folded airbag to position it in the model. The different types available are translation, rotation and scaling. Each ORIENT contains:

- The orient number and type (translate, rotate or scale)
- Geometrical data (location, distance, angle ...)
- Optional nodes for special cases.

An ORIGAMI definition may contain any number of folds and orients, and a model may contain any number of ORIGAMIS. Elements and nodes may be referenced in more than one ORIGAMI, but this would not normally be sensible as the different fold operations might conflict: remember that a node can only have one current coordinate!

Creating and folding a new airbag from scratch



At present the following types of airbag mesh can be created:

A circular folded airbag. The airbag is compressed radially. To enable this the excess fabric forms folds.

A ‘star’ folded airbag. This is formed identically to the circular bag but an extra operation is performed. The airbag is pushed inwards at various points forming a star shape.

The star fold panel is shown in the figure to the right. This panel allows you to define the parameters which will be used to generate the mesh for the airbag. The parameters that can be changed are:

No. of points The number of points on the star fold.

No. of folds The number of out of plane folds which are done.

Element size The optimum size of elements which will be used when generating the airbag mesh.

Inflator rad The radius of the inflator at the centre of the airbag. This area will not be folded in any way. The inflator area on the bottom surface of the airbag will also be put into a different part. This allows the inflator part to be made rigid if necessary.

Unfolded rad The external radius of the airbag when unfolded flat.

External rad The maximum radius of each point on the star. i.e. the radius at the point tip.

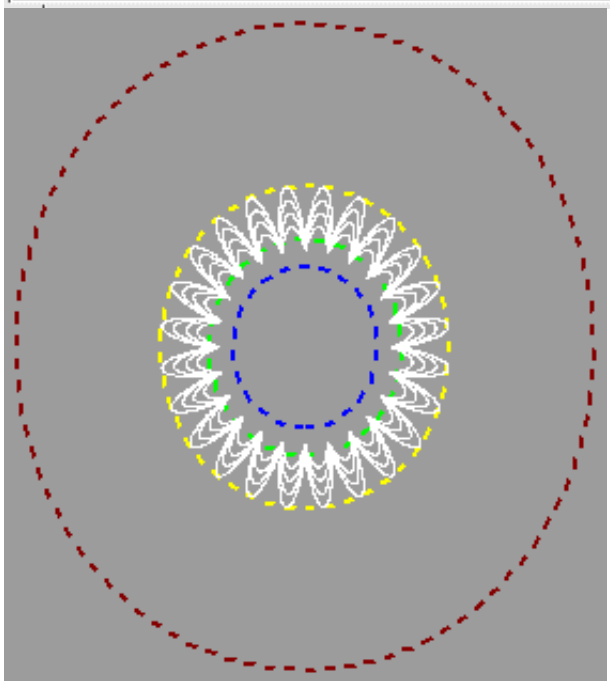
Internal rad The minimum radius of each point on the star.

Section style The cross section style of the folds used. The popup can be used to select two different styles.

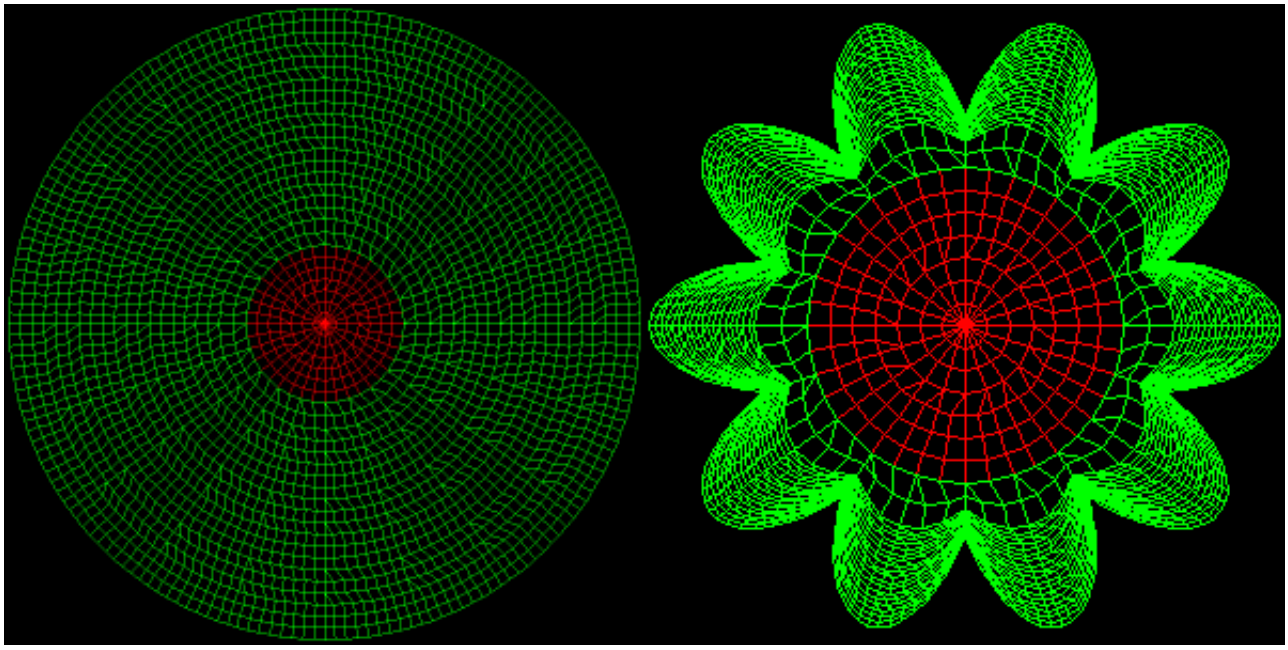
Separation The separation between the upper and lower surfaces of the airbag when unfolded flat.

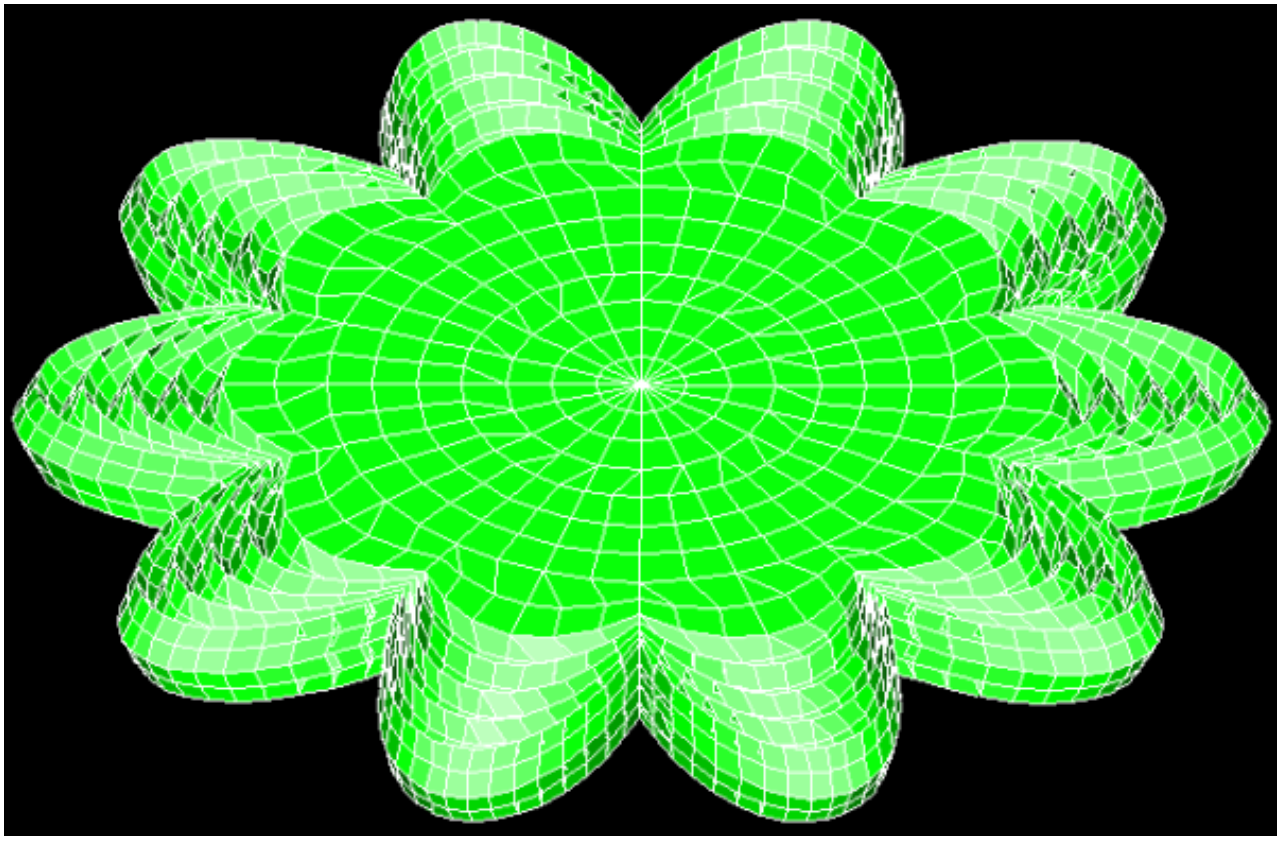
Mesh with The airbag can be meshed either with trias or with a mixture of (mainly) quads and trias. This popup can be used to select which you want

If you change any of the options the graphic also changes showing you the effect of the change. For example, the right-hand figure shows the effect of changing the number of points in the star fold to 24.



The following figures show a starfold with 10 points before and after folding.

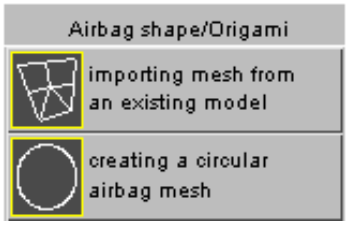




Once the star fold has been created you will be placed in the normal airbag folder. You can then position the bag, check for distorted elements etc.

Once the star fold has been meshed (created) you cannot change the number of points or other parameters. This is because the mesh is dependant on these parameters. If the airbag is not what you expected or required then the process needs to be repeated.

6.1.2 Mesh independent folding



Mesh independent folding allows you to create folds on the airbag at any position, regardless of the mesh on the airbag. It does this by remeshing the airbag after each fold to create a suitable mesh.
For this to work the airbag (or origami) has to be set up in a specific way.

Currently there are 2 methods for doing this.
1) Create a flat circular bag from scratch.
2) Import a mesh

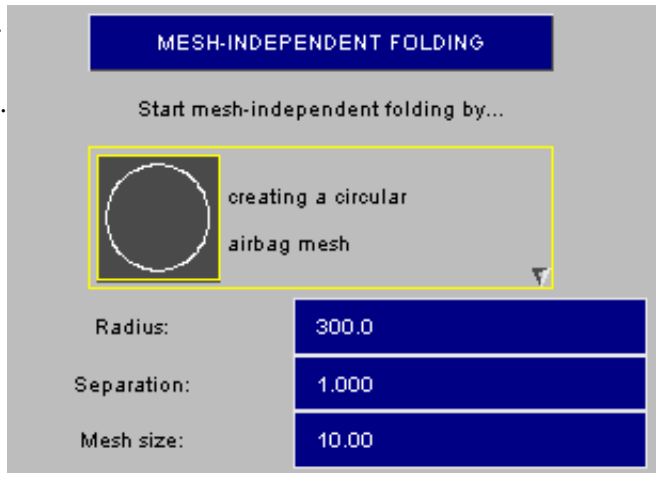
Alternatively, if an airbag has already been defined by one of these methods you can select the existing airbag.

Creating a circular airbag

This option is only available if you have not read an airbag into Primer.

If you have read a model then the option will be greyed out. To enable it **DELETE** the models or restart Primer.

1) From the airbag shape popup select the Circular option.
2) Give suitable values for:
a) the airbag radius.
b) the separation between the top and bottom fabric layers.
c) the element size for the initial mesh.
3) Press the **MESH-INDEPENDENT FOLDING** button.



The airbag will be created and you will be placed in the main folding screen. See section below for details of how to perform mesh-independent folding.

Importing mesh for mesh independent folding

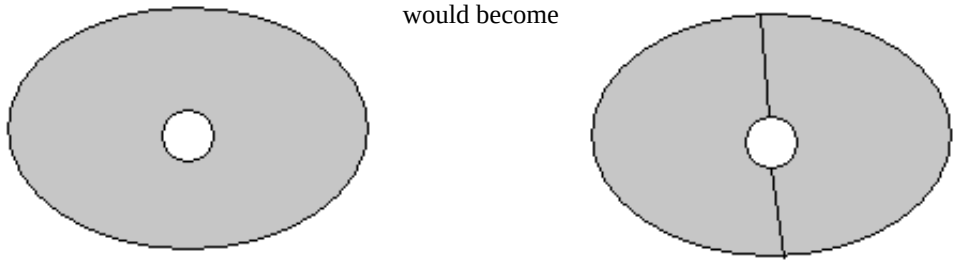
This option allows you to use a mesh that has previously been read into folder for mesh-independent folding.

1) Read an airbag file into Primer.
2) From the airbag shape popup select the Mesh option.
3) Press the MESH-INDEPENDENT FOLDING button.
4) Select the parts that make up the airbag.
5) Select fixed points on the airbag for remeshing.

Selecting parts for mesh-independent folding

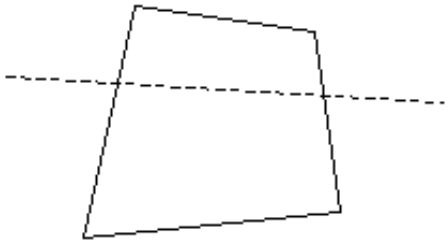
Folder needs to know which parts make up the airbag before mesh-independent folding can be done. There are limitations on parts that can be used:

1) Parts must be flat
2) Parts must not contain holes.
The algorithms that Folder uses when folding mesh-independent bags do not allow parts to have holes. If your airbag contains holes the parts need to be split.
For example

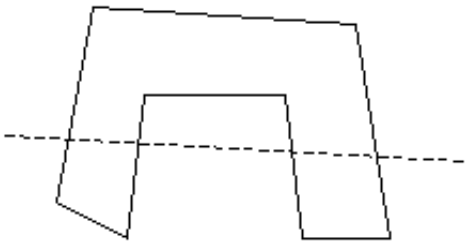


3) The top and bottom surfaces of the airbag must be different parts
4) If a fold will occur on a part, there must only a single fold line.

This part can be folded without problem.



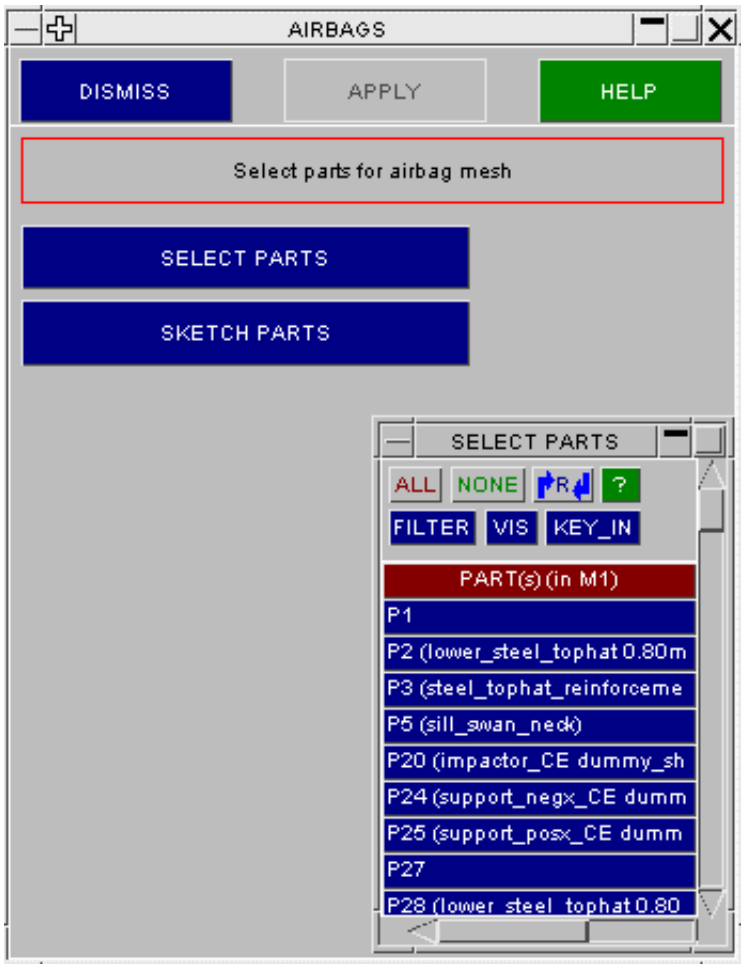
This part cannot be folded as the fold line cuts the polygon in two places.



Select the parts that make up the airbag and press **SELECT PARTS**.

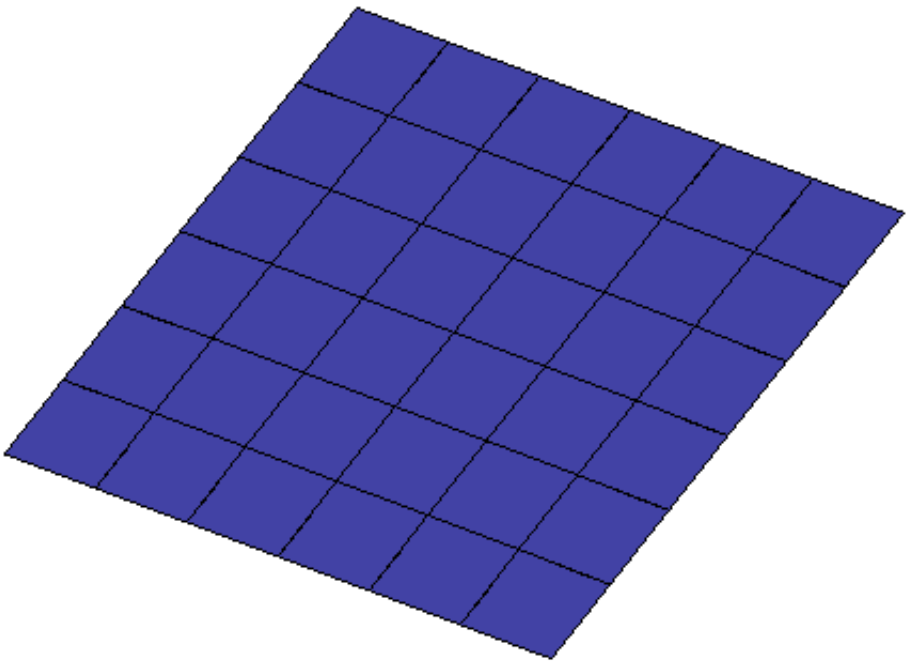
You can sketch the parts you have selected by pressing **SKETCH PARTS**.
In this figure 4 parts are selected.

The following section shows some examples of meshes suitable for mesh-independent folding

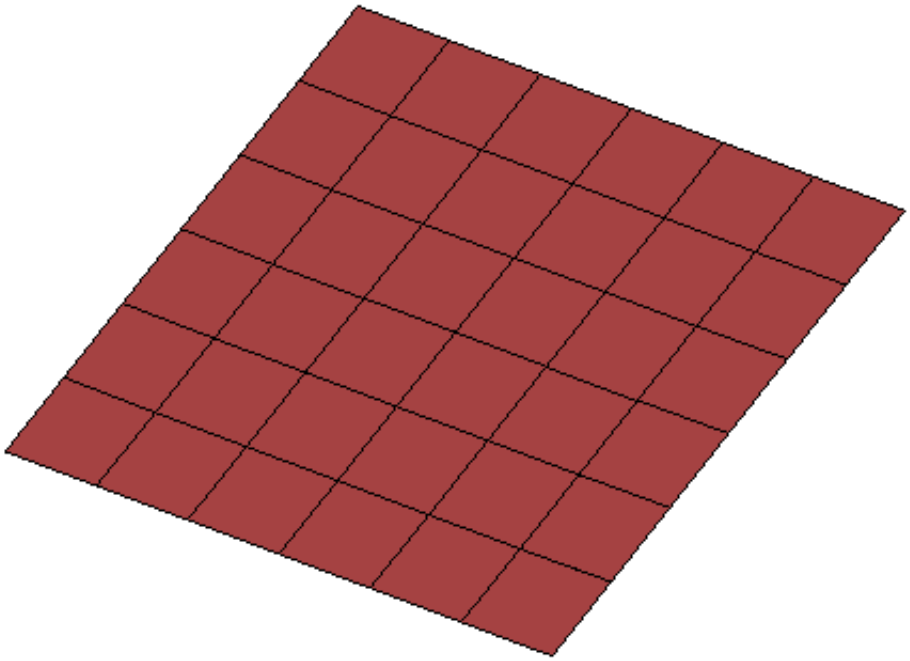


EXAMPLE 1

A square airbag is needed for mesh-independent folding.

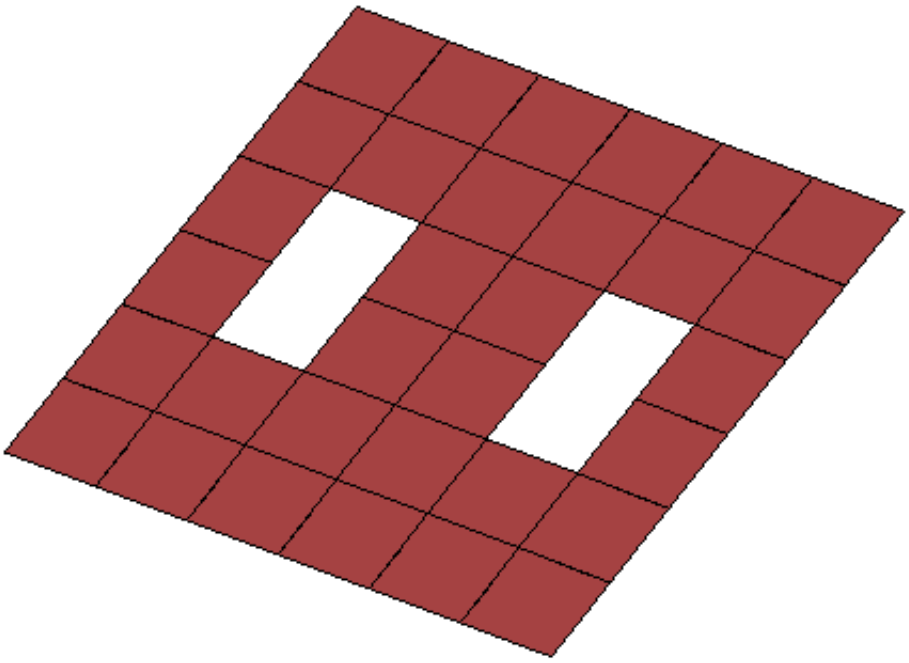


Top surface of airbag

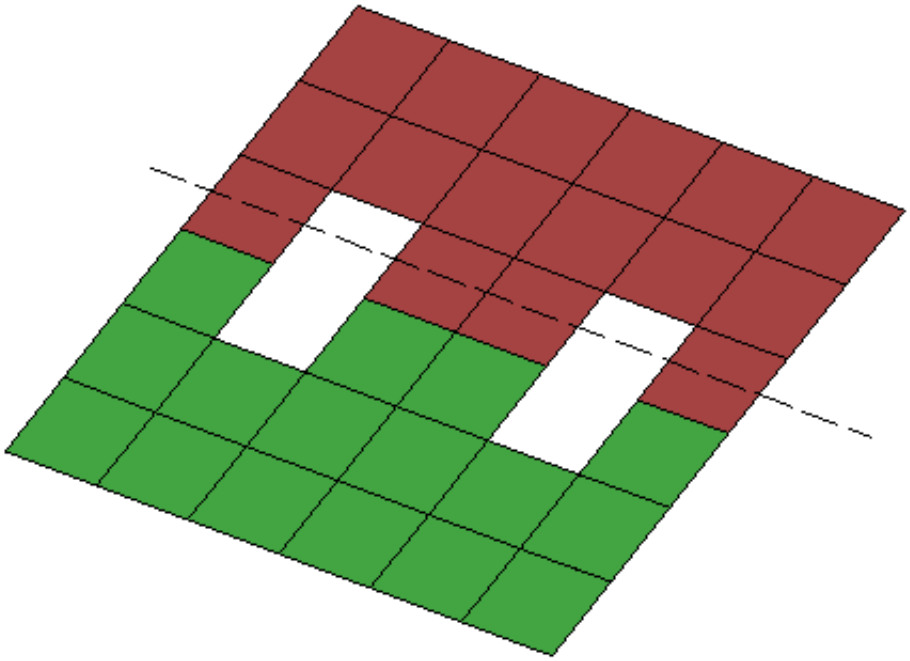


Bottom surface of airbag
This airbag can be used for mesh-independent folding. The top and bottom surfaces of the airbag are different parts, both are flat, and neither have holes.

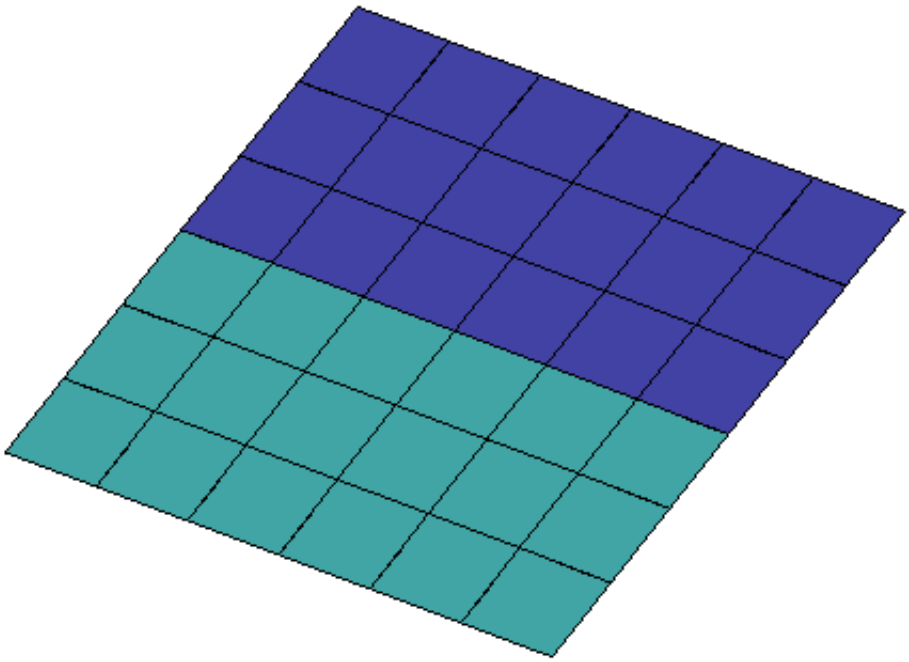
EXAMPLE 2



Bottom surface of airbag
This mesh cannot be used . The bottom surface of the airbag has holes in it.
To overcome this we can split the bottom surface into 2 parts. We also split the top surface to match



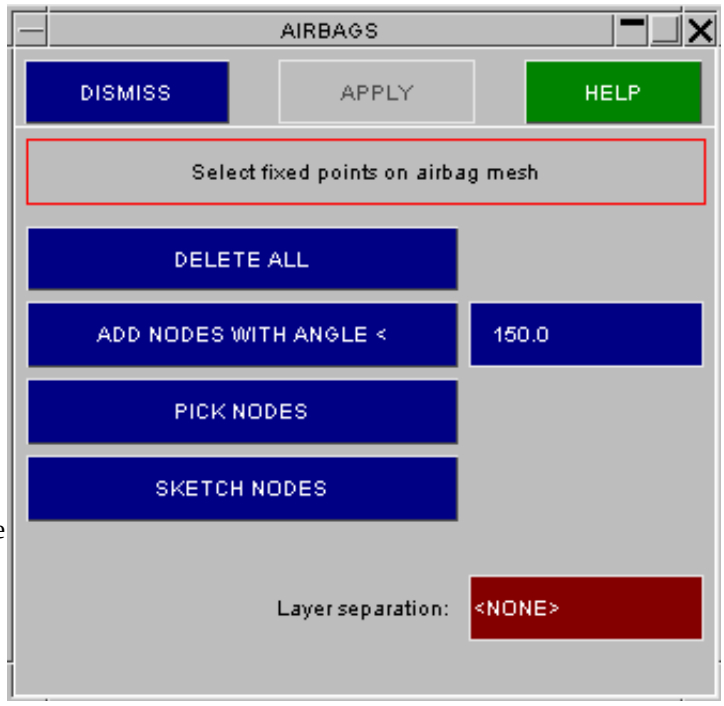
Bottom surface of airbag



Top surface of airbag

We have now eliminated the holes on the bottom surface so this airbag mesh can be used. Note that there would still be a problem if we were going to fold the airbag along the dashed line. In this case the fold line cuts the part in more than one place so will cause problems. If folds such as this are not going to be done the mesh is suitable.

Selecting fixed points for mesh-independent folding



Once the parts have been selected, folder needs to know about the fixed points on the mesh.

The fixed points are required so that folder can remesh the parts in the airbag as a folding is done.

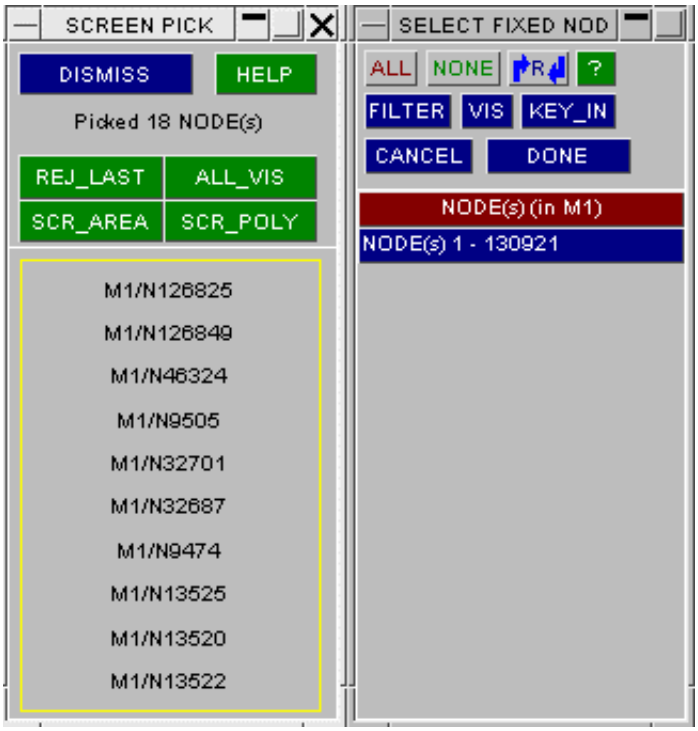
For example, to remesh the simple airbag shown in example 1 above folder would need to know that the corners of the square are 'fixed' in space and so cannot be moved. All the other nodes on the boundary of the parts can be moved without changing the airbag shape.

The 4 corner nodes must be selected as fixed points. If no fixed points are selected folder does not know which nodes are essential and so cannot remesh the airbag as folding is done.

ADD NODES WITH ANGLE can be used to add nodes on the boundaries of parts which have an angle less than the specified value. For example the corner nodes on the square mesh in example 1 have edge angles of 90º so will be selected (< 135º). All the other boundary nodes have angles of 180º so will not be selected.

To enable folder to determine the different surfaces of the airbag you must enter the layer separation (the distance between the top and bottom surfaces of the fabric). In the screenshot above the value has not been entered yet so the **APPLY** button is not active.

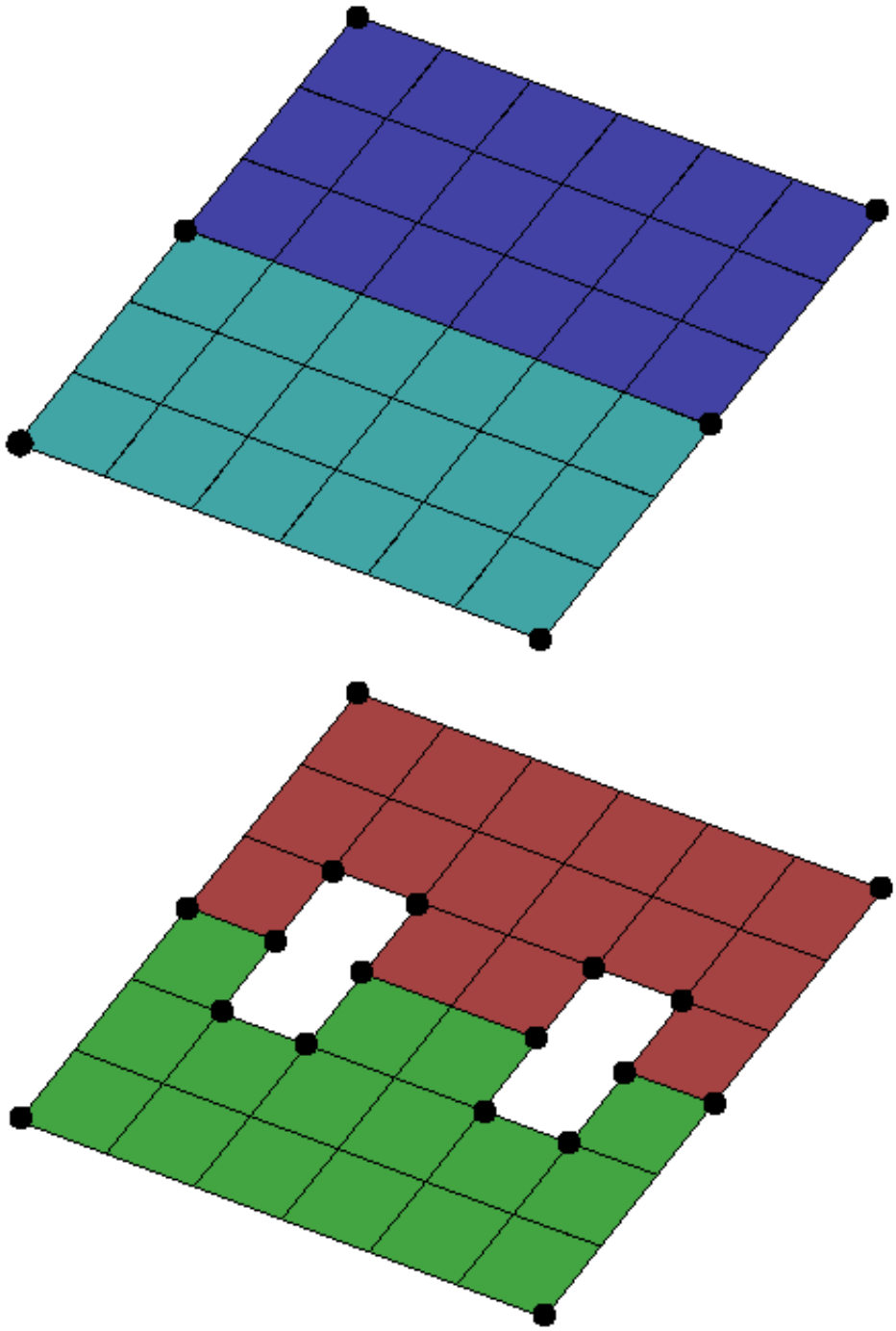
Once all the fixed nodes that you require are selected, **APPLY** will create the origami and take you to the main folding screen



PICK NODES can be used to manually pick nodes to become fixed nodes from the airbag.

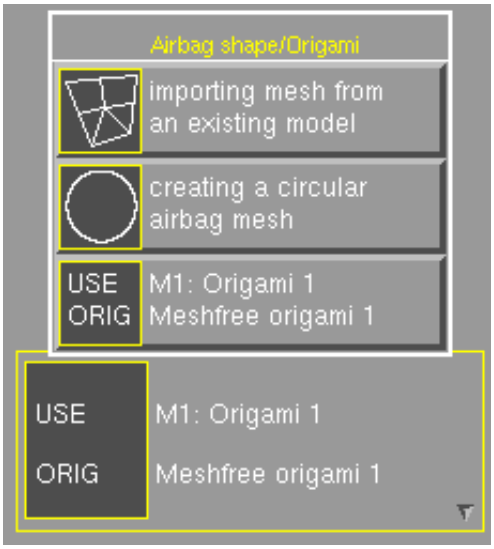
When you have selected the nodes on the screen press **DONE** to add them to the fixed nodes

For another example, the mesh with holes in the bottom surface in example 2 earlier would need the following fixed nodes selecting (shown by the dots).



Selecting an existing mesh-independent origami

If you have previously created a mesh-independent origami you can return to the mesh-independent folder by selecting the origami. In the screenshot on the right another option **USE ORIG** is available. For each mesh-independent origami that exists an option will be shown.



An origami will only be shown if **ALL** the folds in it are mesh-independent folds. If it is not shown it is because one or more folds have been done in the normal airbag folder. In this case to be able to return to the mesh-independent folder these folds would need to be deleted.

Performing mesh independent folding

If the airbag has been created for mesh-independent folding or imported the **SPLIT MESH** and **REMESH** buttons will be available in the main folding window.
Mesh-independent folding is available for the following fold types:

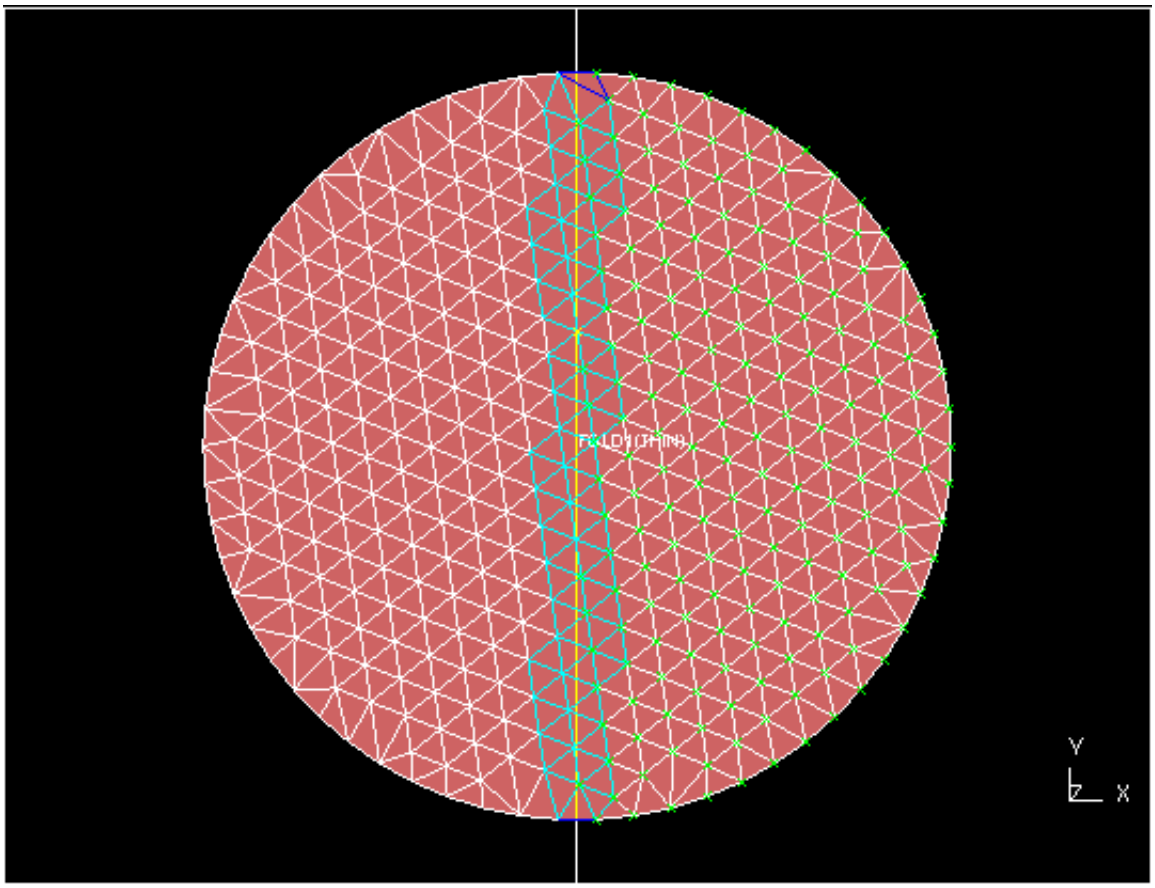
- Thin folds
- Tuck folds
- Thick folds
- Spiral folds

Other fold types are incompatible with mesh free folding and are not allowed (as the airbag is remeshed after each fold the fold definitions would change).

To perform a mesh-independent fold select the fold type, position angle etc as normal (see section 5.5.9). In the example a thin fold will be performed.

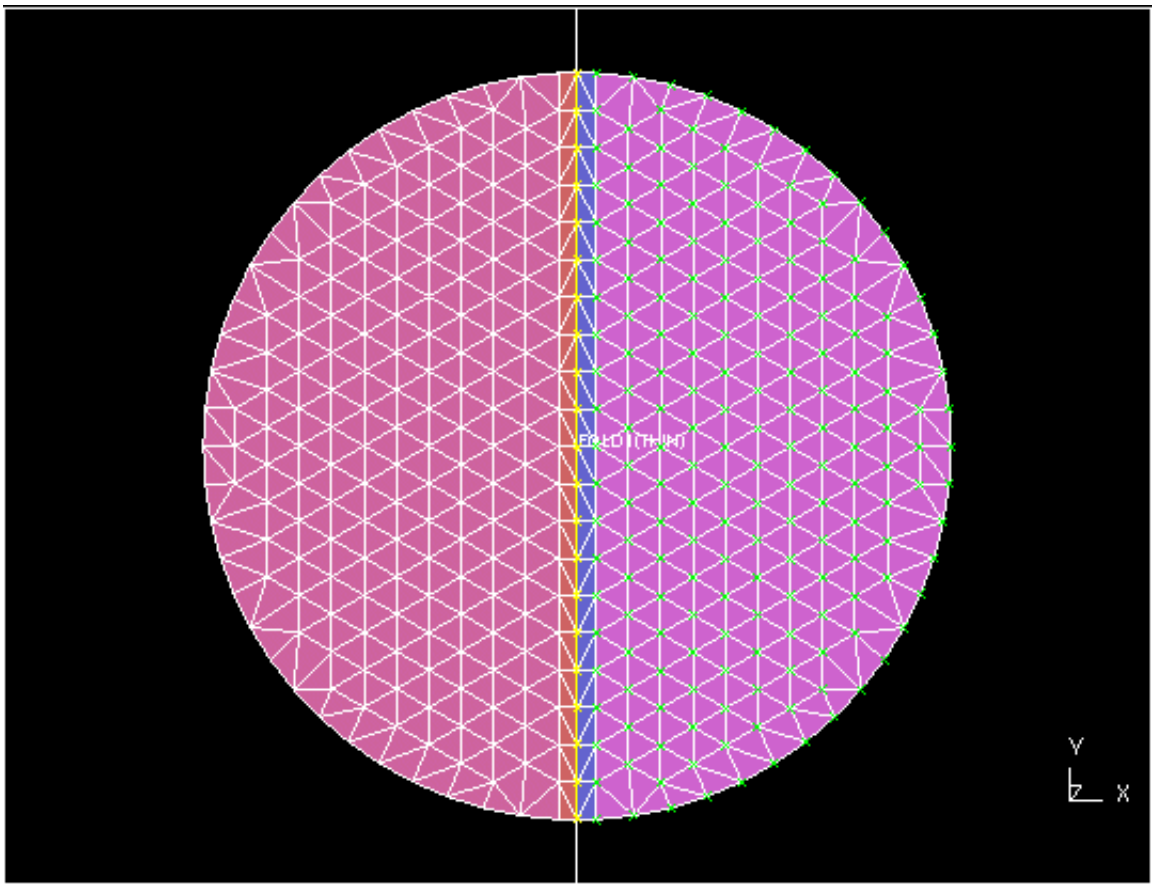
Set the tramline size and element size to the required values in the options panel (see section 5.5.14)

Press **SPLIT MESH**.



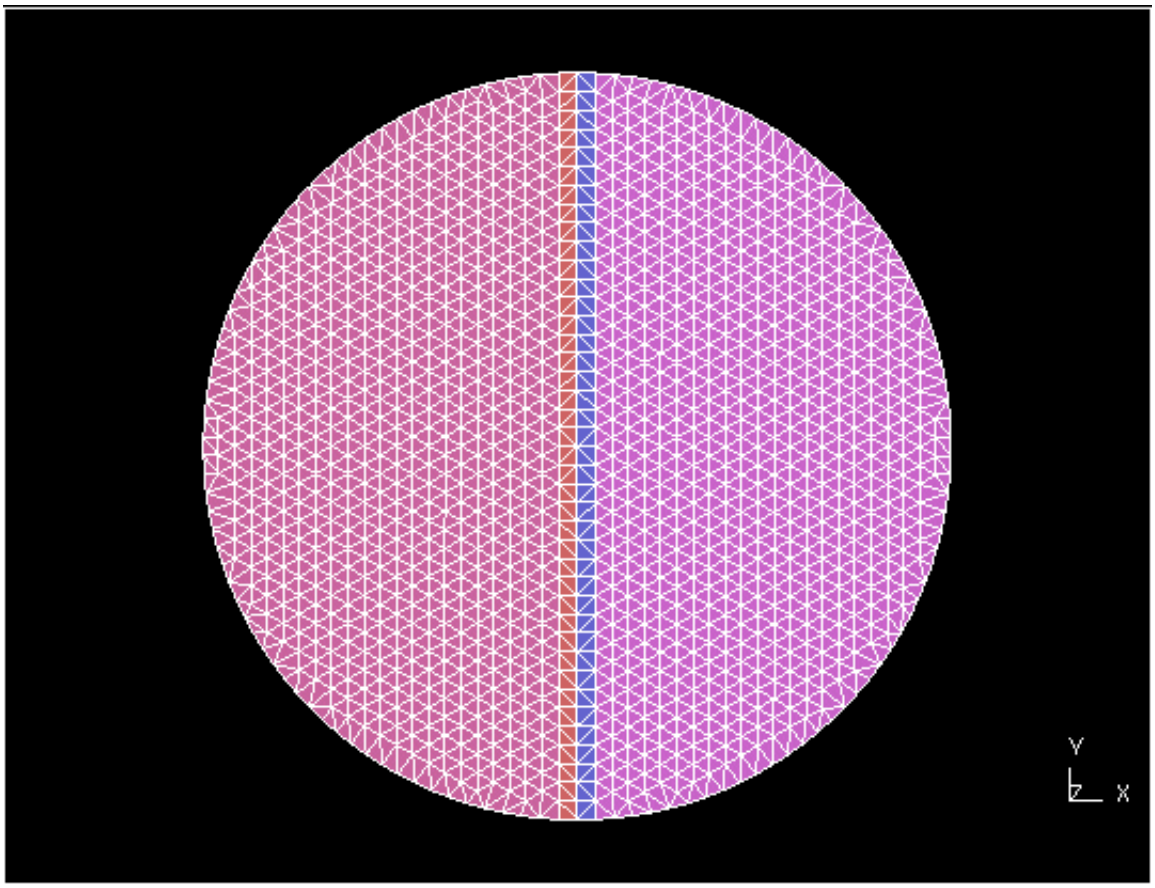
The mesh is split and remeshed. The following example shows the result of a thin fold.
The top and bottom surfaces of the airbag are split into new parts as required (as shown by the different colours) and then remeshed. Fixed points are automatically added to the airbag as required.

To undo a split **DELETE** the fold. The parts will be recombined and remeshed (as long as there are fixed points to enable folder to remesh)



At any time the element size in the options size can be changed and the airbag remeshed using the **REMESH** button. For example the airbag above has a tramline size of 5mm and an element size of 10mm. Changing the element size to 5mm and remeshing gives the mesh on the right.

Note that as the thin fold was made with a tramline size of 5mm, the airbag should not be meshed with elements smaller than 5mm as this could lead to more than one element across the tramline parts. This will cause problems with the thin fold.



6.1.3 Folding an existing mesh

The following sections deal with folding an existing mesh. There are several parts to this so it has been split up into several sections.

- Mesh orientation, airbag reference geometry and main orient command (this section).
- Airbag folding summary.
- Creating an origami definition.
- Creating a local coordinate system.
- Plotting modes.
- Using the main folding panel.
- Creating a new fold.
- Fold types.
- Subset folding.
- Options available in the airbag folder.
- Positioning the origami.
- Saving/reading origami and fold definitions.
- Tips and notes for successful folding.
- Folding example.

Suitable initial orientation of the mesh

The ideal way to fold an airbag is to define the unfolded airbag in the X-Y plane, fold it, and then orient it in space with respect to the vehicle space. If it is not defined topologically to be in the X-Y plane it would be sensible for the user to define a local co-ordinate system such that the local system is planar with the airbag (see Section 6.1.6). When folding then takes place it will visually appear to be in the global X-Y plane. On exiting the airbag folder the bag will return to the global space.

However, it is strongly recommended that folding should start with the initial geometry in the global X-Y plane, as this will reduce the chances of confusion about where the geometry actually is, and make life simpler!

How the ***AIRBAG_REFERENCE_GEOMETRY** interacts with folding

The ***AIRBAG_REFERENCE_GEOMETRY** definition is also crucial to folding, as it is used as the basis for all folding operations. If no such definition exists when folding starts, then PRIMER will automatically copy the nodal coordinates *in their current configuration* into the reference geometry. (Any existing reference geometry for a node will not be over-written.)

This is not a wholly satisfactory solution, as an initial reference geometry which gives a satisfactory element shape for stress calculation during analysis may not prove an ideal starting point for folding operations: typically where a 3D bag needs to be split into several 2D panels for folding, and then reassembled. In this situation it may be necessary to save the "true" reference geometry (for analysis) in a separate file, delete it from the input deck used for folding in PRIMER so that a more satisfactory geometry can be used for folding each panel, then re-introduce it prior to analysis.

This is not a wholly satisfactory state of affairs, and the geometrical basis for folding may in future be changed to be independent of the reference geometry.

Interaction between the **ORIENT** command and folding

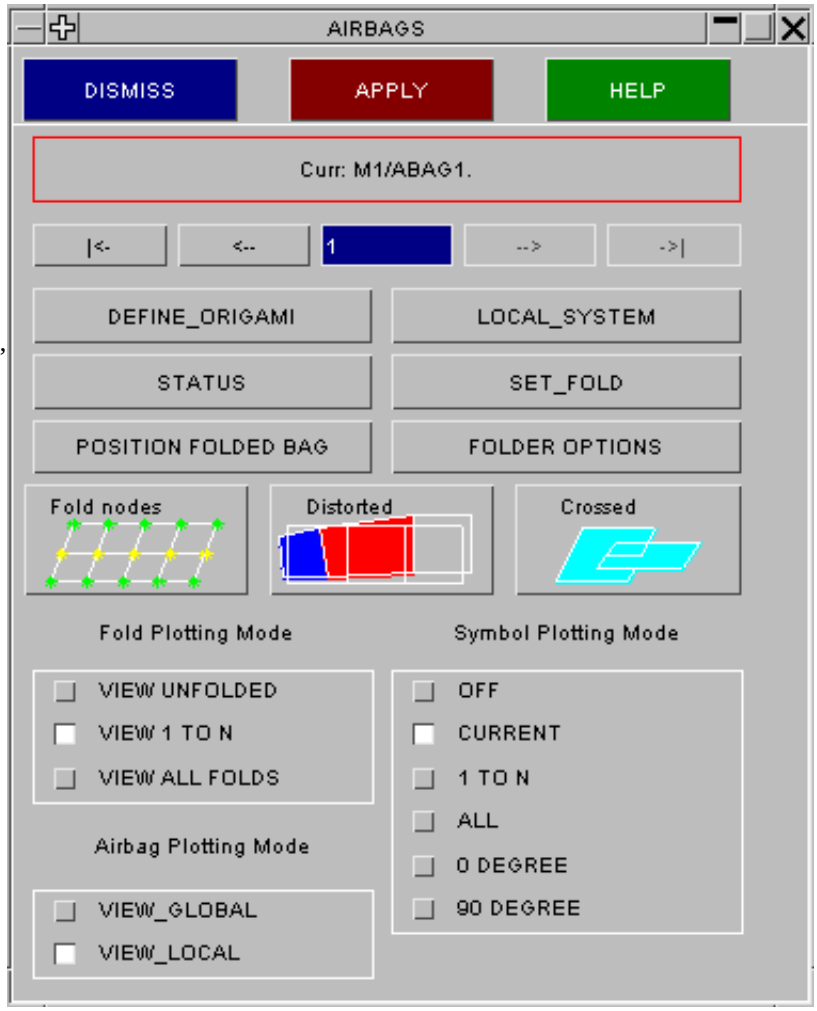
Both **ORIENT** and folding act to change the current coordinates of nodes, and the order in which they are applied is important since the most recently used will "win" in determining the final position of nodes. This is important since folding is always based on the *reference* geometry (which is unaffected by orientation commands), whereas **ORIENT** always operates on the *current* (ie as-folded) geometry. If you use **ORIENT** to position an airbag after folding and then return to the airbag folder at a later date to refold the airbag any orientations will be lost. To prevent this problem the airbag folder has built in orientation functions. These orientations are saved just like folds and so if a bag is refolded it can be repositioned using these stored orientations.

It is strongly recommended that you use the orientation functions build into the folder rather than the general orient functions of Primer so that any orientations you create are saved and can be modified or replayed in the future.

6.1.4 Airbag Folding Summary

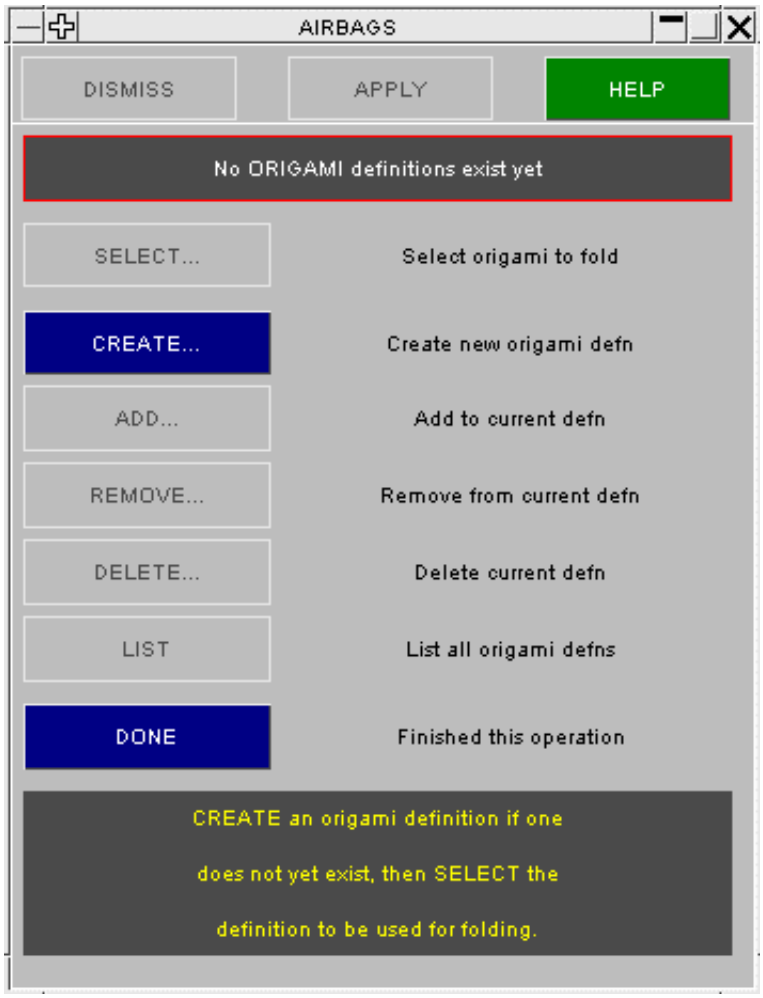
The following process should be followed when folding an existing airbag mesh:

- Read in a model containing airbag geometry (and possibly Origami definitions), and enter the PRIMER airbag module by using the **AIRBAGS** buttons.
- Define an ORIGAMI or select an existing one using **DEFINE_ORIGAMI**.
- Set the local coordinate system if your airbag is not in the global xy plane using **LOCAL_SYSTEM**
- Set the plotting modes that you require using the Fold Plotting Mode, Airbag Plotting Mode and Symbol Plotting Mode radio buttons.
- Set any options for the airbag folder using **FOLDER OPTIONS**
- Define or modify folds using **SET_FOLD** using subset folding, sets and layers as appropriate.
- If required position the folded bag using **POSITION FOLDED BAG**
- Exit airbag folder by pressing **APPLY**. Note that other methods of exiting from this model will lose current fold definitions.
- Save the model to disk.

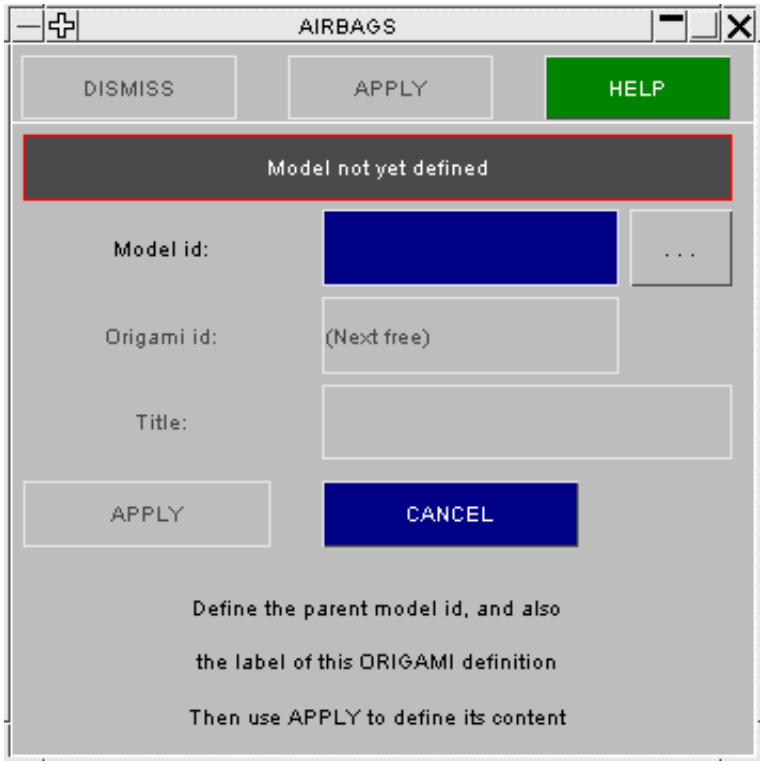


6.1.5 Define Origami: Selecting or creating an ORIGAMI Definition

Before you can fold anything you must have a current ORIGAMI definition. **DEFINE_ORIGAMI** in the main folding screen above gives the Origami definition menu (above left). In this example there are no existing definitions, so it is necessary to **CREATE...** one:.



An **ORIGAMI** definition can only exist in a single model, so the first phase of creation is to select a model, and then to define the Origami label and title. Any number of Origamis may exist within a model, but they must have unique labels.



Once the basic data has been defined you are presented with the standard selection menu which will allow you to define the SETs and/or PARTs and/or ELEMENTs which constitute this Origami definition. These define the nodes and elements which are to be folded.

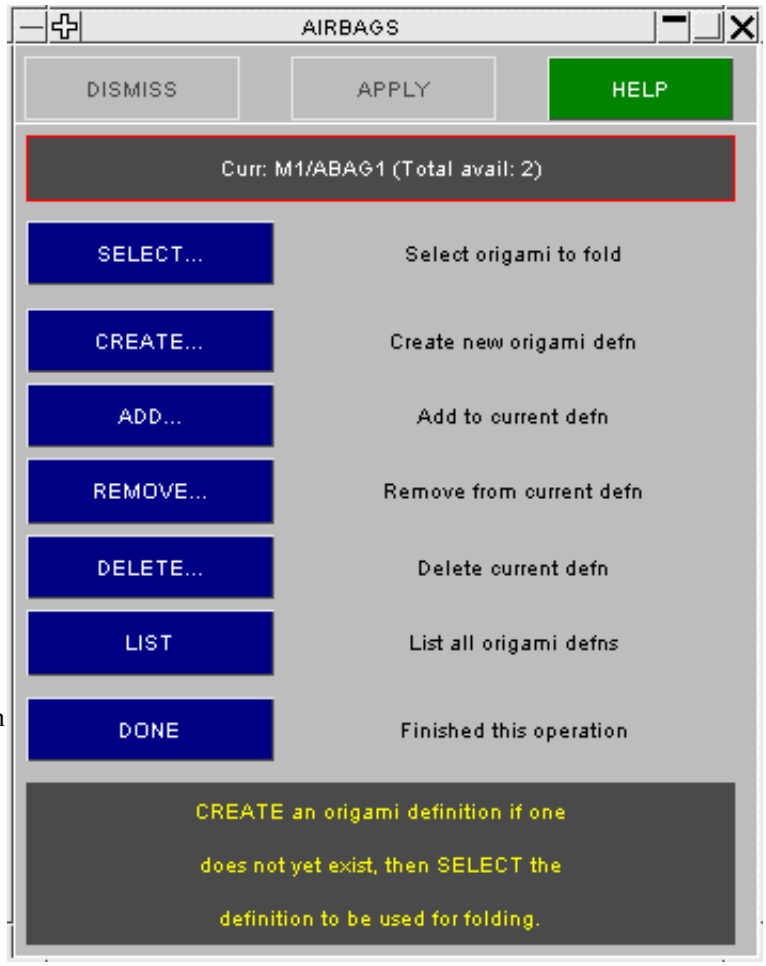
Finally the "AIRBAG REFERENCE GEOMETRY", which is used as the starting point for folding, is set up automatically by PRIMER. If this does not exist for any nodes to be folded, then it is created at this stage by copying the *current* geometry of those nodes.

The definition is now complete, and the Origami definition panel will now be fully populated as shown in the adjacent figure.

You can now **SELECT...** which definition you want to fold.

The other operations here, **ADD..**, **REMOVE..**, and so on are self-explanatory. Origami definitions can be edited at will by re-visiting this panel and manipulating them as required.

Use **DONE** to return to the main folding menu in order to proceed with folding.



The Origami definition is now a permanent part of your model, and will be written out after the **`*END`** card in a LS-DYNA deck so that it can be re-read in future PRIMER runs. (Note that **`*ORIGAMI`** is not a standard LS-DYNA keyword, and it is placed *after* the **`*END`** card so that it will be ignored by the LS-DYNA analysis code.)

It is possible to edit the **`*ORIGAMI`** data in a file by hand - Appendix III describes the format of this data - but it is ***strongly*** recommended that you do not attempt this as the data stored is quite complex: if you want to edit Origamis read them back into PRIMER and do the work there.

Also you should be very careful not separate **`*ORIGAMI`** definitions from their "parent" input files, because they contain references to coordinate systems, sets, elements and nodes that exist uniquely within those files. If you want to keep standard airbag files "on the shelf" make sure that they are complete with geometry and folding data kept together: PRIMER will merge airbags and structural models for you.

6.1.6 LOCAL_SYSTEM: Defining Airbag Local Axes

In general it is easiest to fold a bag if it is oriented initially in the global X-Y plane. However, if it is already in a vehicle this is unlikely to be the case. The user can therefore move the bag to a more convenient folding position.

This is done by pressing **LOCAL_SYSTEM** which will invoke the menu shown on the right. The user should then define the local X-Y plane (referred to as the origami local system) by selecting three nodes:

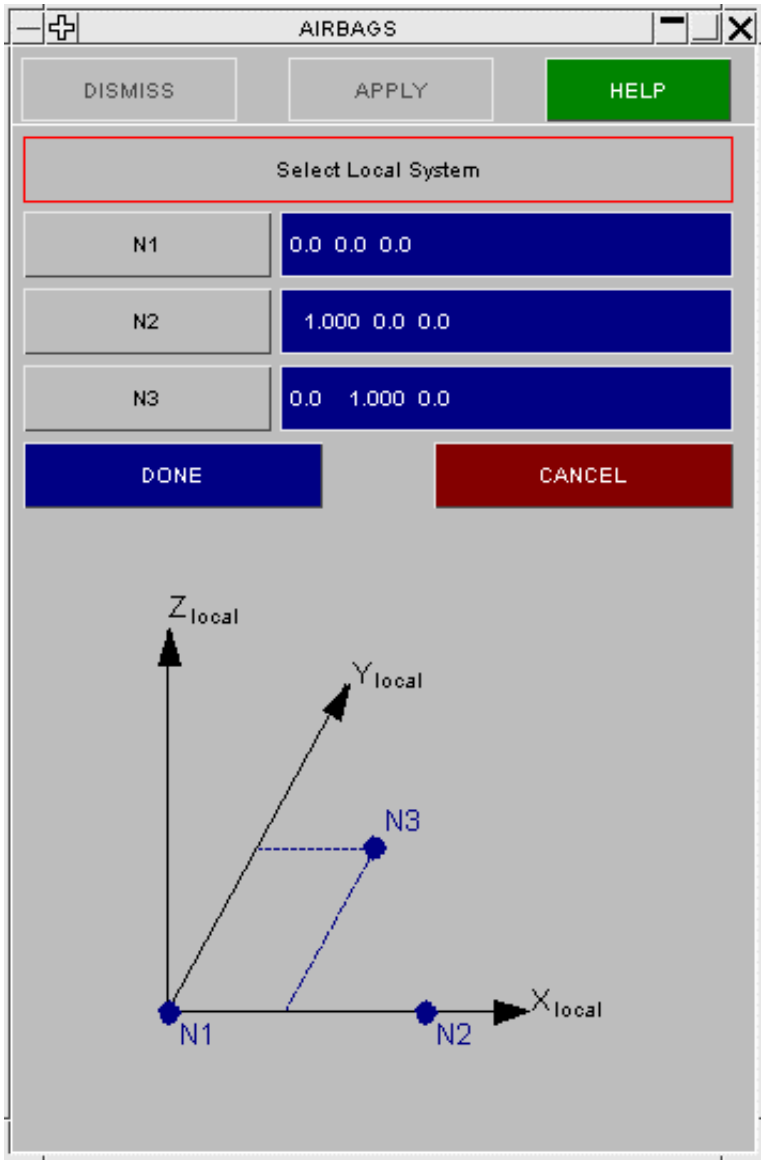
N1 defines the local axis origin;

N2 defines the local x-axis;

N3 defines another point in the X-Y plane.

Alternatively you can type in the local axis vectors, which PRIMER will normalise for you.

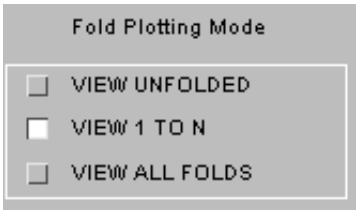
If this option is invoked when the current fold number is non-zero (see Section 6.1.8) the user can define a local axis system that affects the current fold only (referred to as the fold local system). Otherwise it applies to the Origami definition as a whole.



6.1.7 Plotting Modes

You can exercise control over how the Origami definition is drawn and annotated in a variety of ways. The following buttons are on the main folding window.

Fold Plotting Mode



VIEW UNFOLDED displays ORIGAMI without applying the folds to the displayed geometry;

VIEW 1 TO N displays the ORIGAMI folded up to the current fold (see Section 6.1.9) excluding any other folds;

VIEW ALL FOLDS displays all folds defined on the current ORIGAMI. This is not affected by the current fold number.

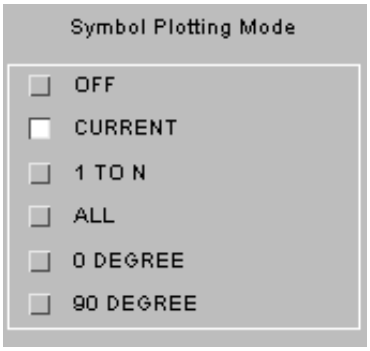
Airbag Plotting Mode (The coordinate system used for airbag display)

VIEW_GLOBAL Displays the airbag in the global cartesian system;

VIEW_LOCAL Displays it in its local system (if defined).

Symbol Plotting Mode (Plotting of fold line symbols)

Note that where folds slice through elements the actual fold line (ie along element edges) is shown as well as the defined fold line (ie across elements). The fold line symbol can be:



OFF No fold symbols are displayed;

CURRENT Only the current fold is displayed;

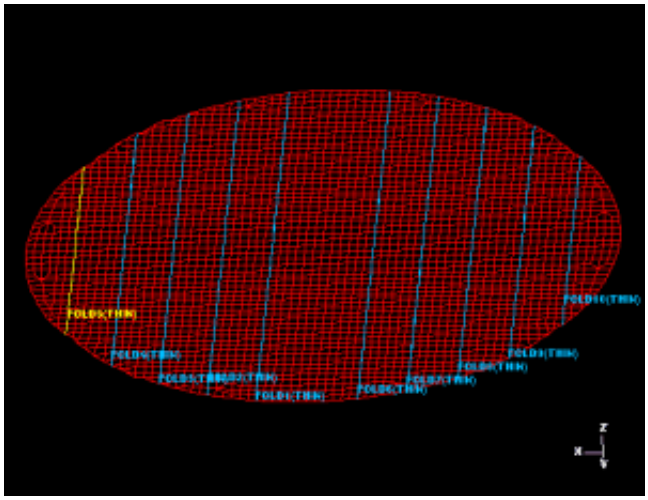
1 TO N Displays all folds up to the current fold;

ALL Displays all folds;

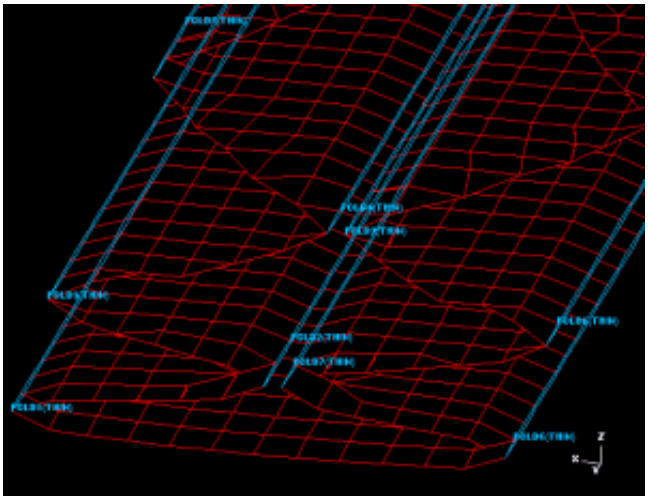
0 DEGREE Displays all folds perpendicular to the (local) X-axis;

90 DEGREE Displays all folds parallel to the local X-axis.

The following two examples show typical "unfolded" and "folded" displays of the same airbag, with fold line symbols superimposed in both cases.



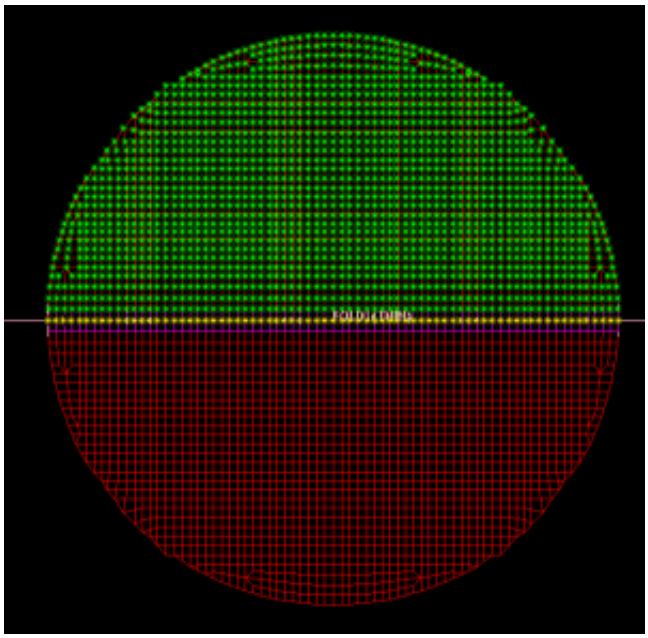
Unfolded mesh drawn in the global system showing all fold lines superimposed



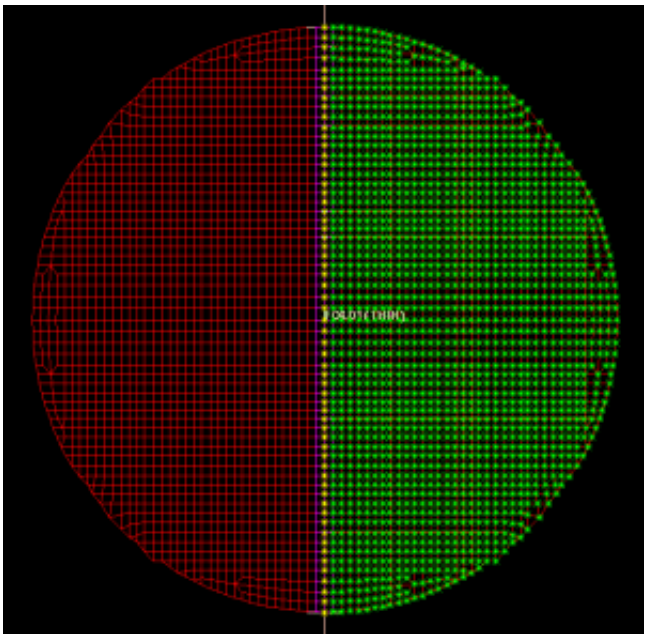
Folded mesh (all folds) showing fold lines superimposed.

Fold Node plotting (Visualisation of fold nodes)

By default when you make a new fold the nodes which will be moved in the fold are highlighted. This helps to show if you have the correct nodes selected for the fold. For example the next 2 figures show which nodes will be folded in a 0º or 90º fold.

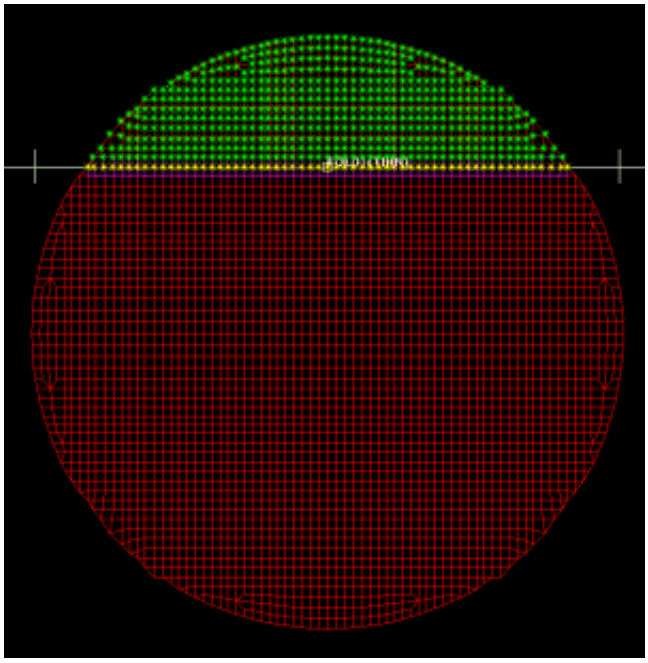


Fold nodes shown for a 0º fold from right to left.

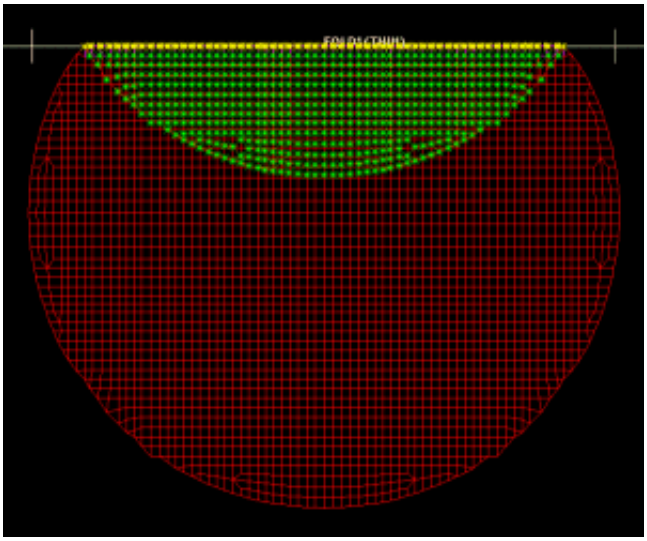


Fold nodes shown for a 90º fold from top to bottom.

Changing any fold parameter such as the fold point, fold direction, fold angle automatically updates the display.



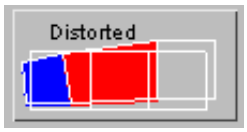
90º fold after changing fold point



90º fold after applying thin fold

Automatically drawing the fold nodes can be turned off by using the folder options (see section 6.1.11). At any time the fold nodes can be redrawn by using the **FOLD NODES** button.

Crossed and distorted element plotting



By default when doing a fold, elements which are distorted in the folding process and elements which have penetrations or are crossed are highlighted on the origami. Just as with the fold nodes in the previous section, when any fold parameters are changed the display is automatically updated.

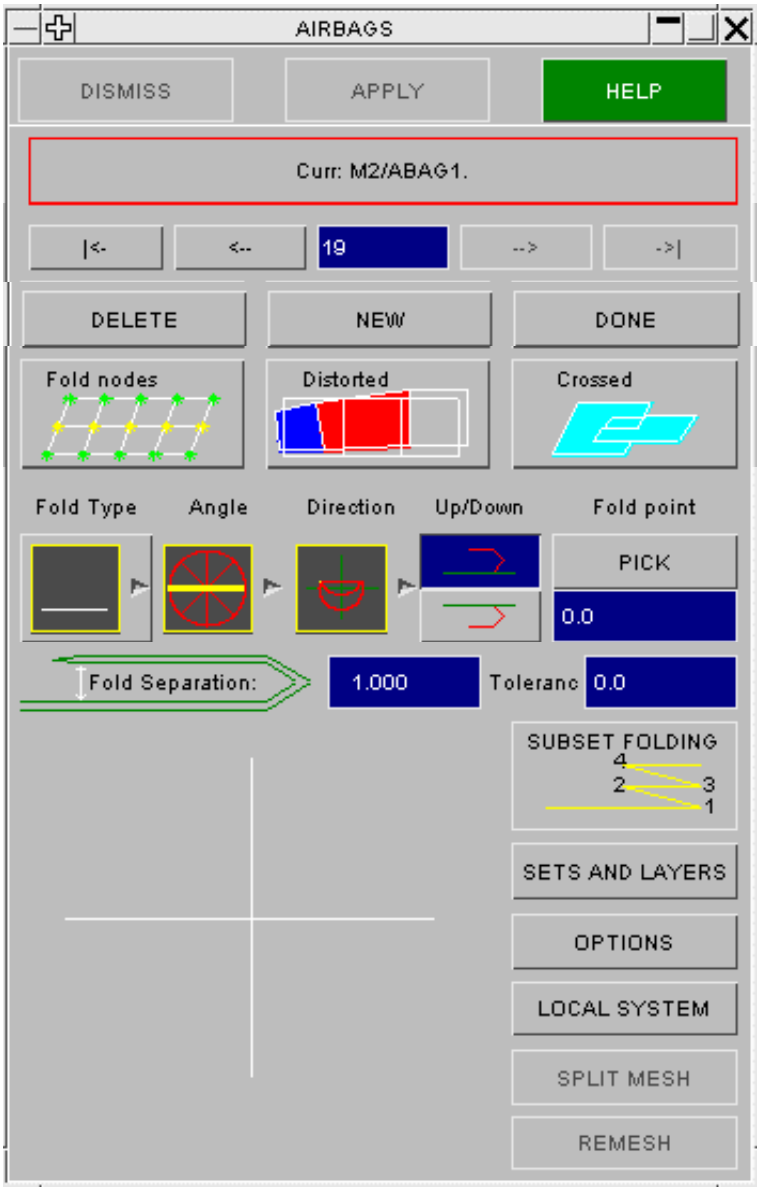
Automatically drawing of crossed, penetrating and distorted elements can be turned off by using the folder options (see section 6.1.11). At any time the elements can be redrawn by using pressing the **DISTORTED** and **CROSSED** buttons.

Elements are crossed when a node from one element has passed through the mid plane of another element. Elements are defined as penetrating if a node from one element is within the thickness of another element but has not passed through the mid plane. These features are very useful when adjusting the tip scale factors for thin folds. As the fold tip is adjusted the display will show if there are any penetration problems. In this way any potential penetration problems can be visualised and fixed. The thickness which is used for the penetration check can be altered in the folder options (see section 6.1.11).

Element distortion is defined as the ratio of the current element side or diagonal length divided by the reference element side or diagonal length. Therefore if an element is stretched the number will be greater than 1. If the element is shrunk the number will be less than 1. Three different contour bands are available for plotting distorted elements. The ranges and the colours for each contour can be altered in the folder options (see section 6.1.11).

6.1.8 SET_FOLD Creating Fold Definitions

Pressing **SET_FOLD** in the main folding window invokes the folding menu in the adjacent figure.



The fold number is adjusted by the **|< ... >|** buttons.

Fold control and saving are controlled here.

Fold node, distorted and crossed element plotting buttons.

The current fold type (null, thin, etc.), fold angle, fold direction, fold orientation and fold location.

The default/fold thickness and tolerance are set here.

Subset folding button

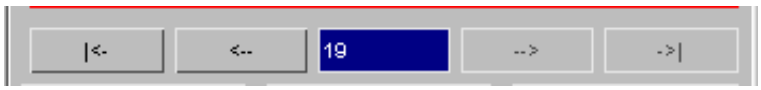
Sets and layers (section 6.1.10)

Airbag folder options

Local coordinate systems

Mesh independent folding

Selecting the current fold number



The currently active fold is set via the buttons or text box in this region. In the figure above fold zero is shown, which implies "the whole origami", and which allows you to set values for the whole airbag. Once finite fold ids (in this example #3) are shown, then operations apply to that fold only.

Controlling fold progress

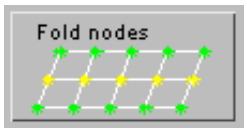
DELETE Deletes the current fold definition. All folds above this one have their number decremented by one.

DONE Finishes the folding process and returns the user to the main FOLDING menu.

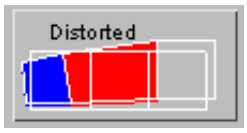
NEW Creates a new fold. If used in the middle of a sequence a new fold is inserted between the previous and next folds, and all folds above this point have their number incremented by one.

Viewing fold node and element information

FOLD_NODES Redraws the fold nodes at any time. By default this is done automatically but if this is turned off with the folder options this button can be used



DISTORTED Redraws the distorted elements at any time. By default this is done automatically but if this is turned off with the folder options this button can be used to plot the distorted elements.



CROSSED Redraws any crossed elements. By default this is done automatically but if this is turned off with the folder options this button can be used.

Selecting the current fold type

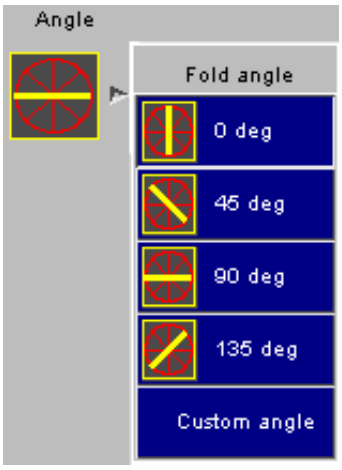
The **Fold Type** popup button controls the current fold type, which will be one of

- **Null fold**
- **Thin fold**
- **Thick fold**
- **Tuck fold**
- **Spiral fold**
- **Scrunch fold**
- **Align fold**
- **Scale fold**
- **Translate fold**
- Star fold
- Circular fold.

In this example a "thin" fold is current, but the type can be changed dynamically by simply selecting a new type with the popup menu. More information about each fold type is given below in section 6.1.9. Star and circular folds are unavailable from this menu. These are selected to be created from the initial airbag start up screen. For more information on these types see Creating and Folding a new airbag.

The fold type button will toggle between the currently selected fold type and a null fold.

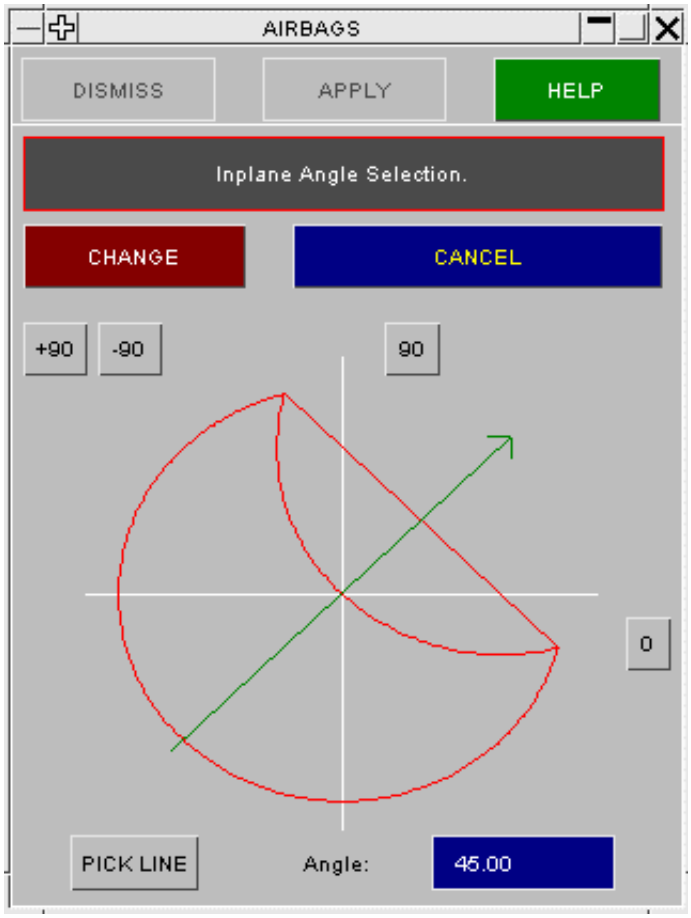
Setting the Fold Angle



The Angle popup button can be used to select the angle of the fold. In this diagram the current fold angle is 0º . The popup contains default fold angles of 0, 45, 90 and 135º . Most folds will take place in either the X (0º) or Y (90º) directions, but if necessary, you can use Custom angle to set the fold to an arbitrary angle.

If a new fold angle is chosen the graphic on the Angle button will be updated so you can easily see the angle of the current fold.

Defining an arbitrary XY fold angle using Custom angle



The adjacent figure shows the box after pressing Custom angle. This allows the user to define an arbitrary fold line.

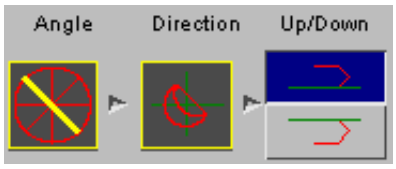
The angle may be selected in either of the following ways:

PICK LINE Calculates a line, and hence an angle from two screen-picked nodes.

Angle: Accepts a typed in angle, which may also be modified by **[0]**, **[90]**, **[+90]**, **[-90]**.

To accept the current definition press **CHANGE**, which will return to the fold definition menu.

Setting the Fold Direction



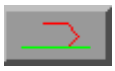
As well as setting the fold angle, the direction in which the fold is done needs to be set. In the adjacent figure the fold angle is set to 45º (you can tell this because of the angle of the yellow line). The graphic shown for the direction of the fold is also drawn at the 45º angle but there are two possible ways to do this fold

The upper right part of the airbag can be folded onto the lower left part of the airbag (**Forward**).
The lower left part of the airbag can be folded onto the upper right part of the airbag (**Reverse**).

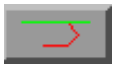
You can select the fold direction by choosing either Forward or Reverse. The graphic on the **Direction** button will change to indicate your selection.

Setting the up/down direction

When a fold is done the folded material can either be folded on top of or underneath the unfolded material. The Up/Down buttons are used to set this.



Folded material is folded on top of unfolded material.



Folded material is folded underneath unfolded material.

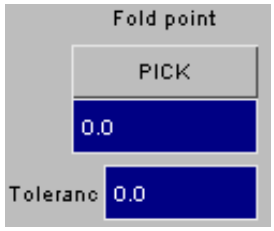
Setting fold separation distance

When a fold is current this sets the distance between layers, the thickness.

If fold #0 is current, this sets instead the Default Thickness to be used for all folds in the airbag, which may be overridden for individual folds.

When folding takes place the larger of the default and individual fold thicknesses is used, so a useful way to visualise an airbag is to set the individual fold thicknesses to their correct, relatively small, values; but to set the default thickness to a larger value, so making it easier to view folds. When folding is complete the default value can be reset to a more realistic value.

Setting the FOLD_POINT and Tolerance



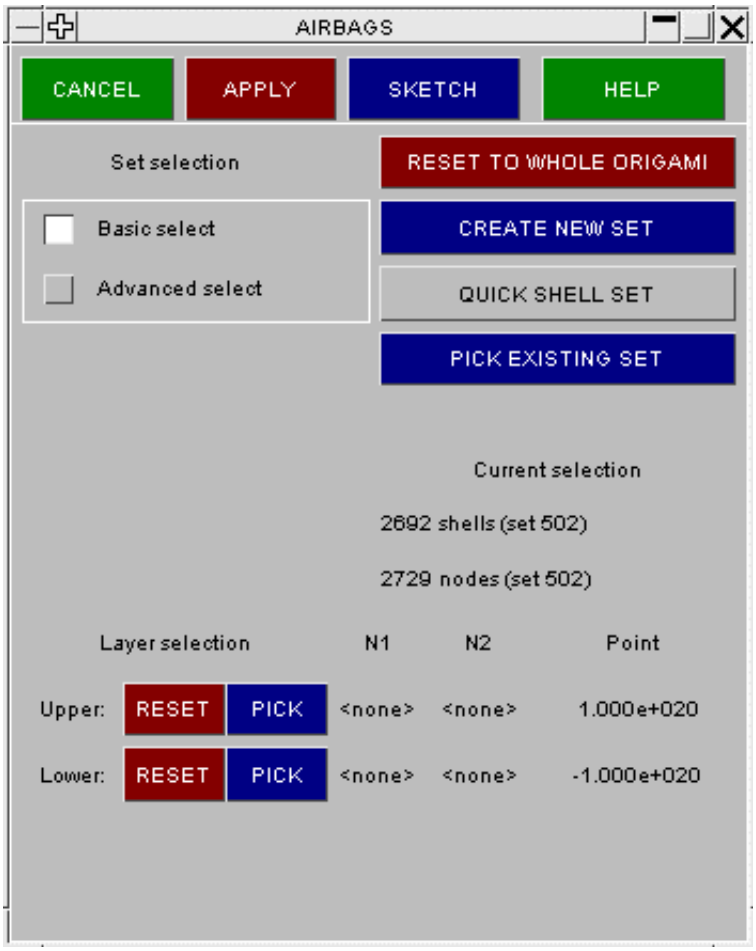
The fold point identifies an XY coordinate through which the fold line passes. This can be defined by using **PICK** to screen-pick a node, or by typing in an explicit value (which is a distance down the relevant axis). For thin and tuck folds this point should lie on, or be very close to, a node; for other fold types any reasonable location may be chosen.

Tolerance defines the tolerance margin either side of the fold line used when searching for nodes that lie on the line. You should aim to choose the smallest value that will include all nodes on the fold line.

6.1.9 SETS AND LAYERS Selecting a subset of the airbag for folding

By default the whole origami is considered for folding when a fold is defined. Sometimes you do not want to fold the entire airbag. For example, if you have done several folds already, on your next fold you may only want to fold the top layer of the airbag rather than the whole airbag. Sets and layers are used to select which bit of the airbag you want to fold. They work in different ways. Normally you will only need to use one or the other but both can be used together if needed.

Using Sets



A set is a collection of shells from the origami. If a set is defined then rather than folding the entire origami, just the shells in the set will be considered for folding. The set could contain a single shell from the origami or it could contain the whole origami.

There are two methods for selecting sets: **Basic select** and **Advanced select**. It is strongly recommended that you use **Basic select** as this is much simpler. If this cannot do what you need (which is unlikely) then use the **Advanced select** option.

The sets works by using sets of shells. These can be created inside the airbag folder, or if your model contains sets already they are available for use. To pick an existing set use the **PICK EXISTING SET** button. You can then choose from the list of available sets.

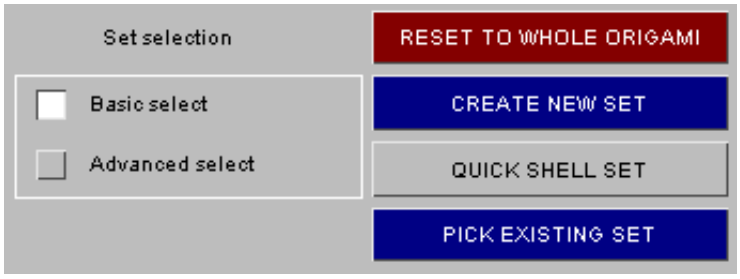
To see the current set selection you can use the **SKETCH** button. This will sketch which shells are in the set on the graphics window. You can reset the selection to the whole origami by pressing **RESET TO WHOLE ORIGAMI**.

To create a new set you can use **CREATE NEW SET**. This will start the standard set creation panel. Once the set has been created it will automatically be chosen for folding.

Alternately you may use **QUICK SHELL SET** to create the set at highest label + 1.

It is important to realise that just because a shell is in a set does not mean that it will definitely be folded. The shells that are in the set are the ones that will be considered for folding. As an example consider the first fold of an airbag where we have defined the set to be the entire origami. When the fold is actually performed the folder checks all the shells in the set. Some of these shells will be on the wrong side of the fold line and will be left in their original positions. Only the ones which are on the correct side of the foldline are actually folded. So even though all the shells were considered for folding, the folder actually only folded the shells on the correct side.

Quick set creation



The **QUICK SHELL SET** button allows you to create a shell set in a much quicker way than the normal creation method.

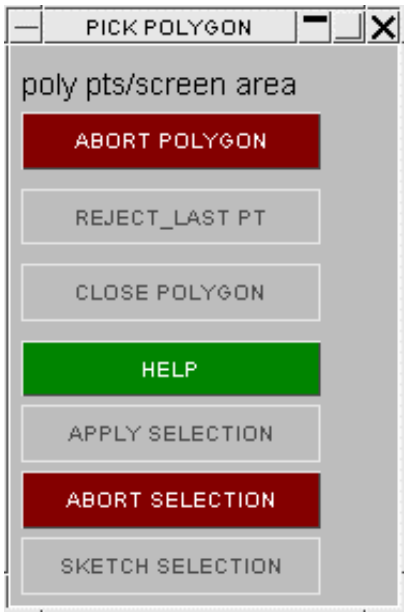
A shell set is automatically created for you with the next free label.

Shells can be added or removed from the set by the following methods

1. Clicking and dragging with the left mouse button will add shells to the set by box.
2. Multiple left mouse button clicks will add shells to the set by polygon.
3. Clicking and dragging with the right mouse button will remove shells from the set by box.
4. Multiple right mouse button clicks will remove shells from the set by polygon

To finish selecting the shells and create the set, press the **APPLY SELECTION** button.
When selecting by polygon, if you close the polygon by clicking back onto the first point, the shells will be added.

Alternatively you can press **CLOSE POLYGON** to select the shells. **REJECT LAST PT** will delete the last point in the polygon. **ABORT POLYGON** will restart the polygon.



Advanced set selection

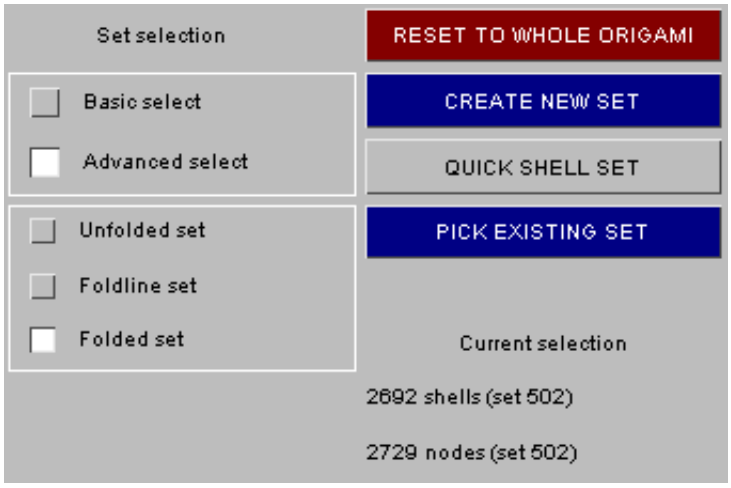
The advanced set selection works with sets just like the basic set selection. The difference is that instead of using a single set (basic select), 3 sets are used.

When a fold is performed there are three distinct regions of the fold.

The **Unfolded set**. The shells that will not move. i.e. they will be unaffected by the fold.

The **Foldline set**. The nodes which are actually on the fold line.

The **Folded set**. The shells that will be folded. i.e. the shells that will move during the fold.



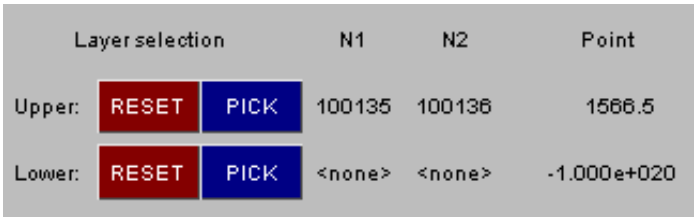
Each of these 3 sets can be selected individually. They can be completely different sets.

It is important to realise that the logic from the basic selection method still applies here. The folded set contains the shells that will be considered for moving during the fold. The foldline set contains the shells (actually the nodes from these shells are considered) that will be considered for the fold line etc.

In actual fact the basic selection method works by setting all 3 sets to be exactly the same. It is then the folder which works out which shells should be folded, which should be left in place and which nodes are on the fold line.

Layers

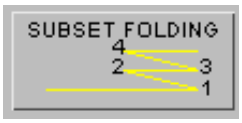
By default all layers (through the thickness of the airbag) will be folded, but sometimes it is convenient to restrict this to only layers within a given +/- Z coordinate.



Upper and Lower layer selection define respectively the upper (positive) and lower (negative) Z limits within which material will be folded. Selecting **PICK** from either prompts you for two nodes, and the Z limit used is set to the average of these. (The reason for having two nodes is that you need to define a point between the outermost layer to be folded, and the layer beyond that, and usually there will only be empty space there!). The default values that are used for the upper and lower limits are 1.0e+20 and 1.0e-20 respectively so by default the whole origami (or set) will be folded. **RESET** can be used to set the lower/upper layer back to this default. In this example the upper layer is 1.0e+20 and the lower layer is -0.3 so anything which is less than Z=-0.3 will not be considered for folding.

Airbag Folder options

Various options can be set for the folder. See section 6.1.11

Subset Folding 

Subset folding can be used to quickly create folds. See section 6.1.10

Fold Creation

The following process should be followed to create folds:

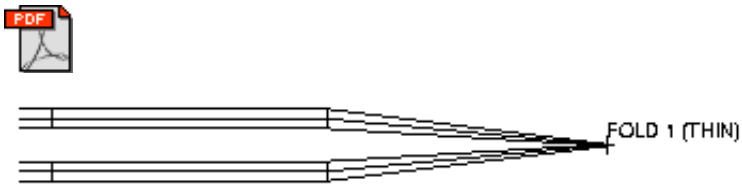
1. Create a **NEW** fold. By default a point point at 0 and left-right folding is done.
2. Fold angle: Set the fold angle by using the popup button to choose one of the preset angles or a custom angle. As you change the angle, the highlighted nodes on the screen change.
3. Fold direction: Decide on whether material is being folded in the forward or reverse direction. As you change the direction the highlighted nodes on the screen will change accordingly.
   Decide on whether folded material should finish above or below unfolded material. The top of the bag is defined as being in the direction of increasing Z.
4. Select material to be folded if necessary: ie define **Upper** or **Lower** layers and/or a set in the **SETS AND LAYERS** menu. As you change the layer or set selection the highlighted nodes on the screen will change accordingly.
5. Decide on the fold type (**Thin**, **Thick**, etc) - see Section 6.1.9.
6. On pressing the fold type required PRIMER folds the ORIGAMI. This may require adjustment of parameters (Separation, Tolerance) to achieve a good result.
7. Continue creating new folds until the complete ORIGAMI is folded.

6.1.10 Fold types

Null Fold (Set attributes, but don't fold mesh)

The **NULL** fold allows the user to define all the fold data without actually folding the ORIGAMI. This is useful when the current fold appears to be incorrect (eg too many layers have been folded). Pressing the NULL fold option effectively unfolds the current fold and allows the user to change the fold parameters (eg entity selection). The nodes which will be folded are still highlighted

Thin Fold (Perform a sharp "crease" fold)

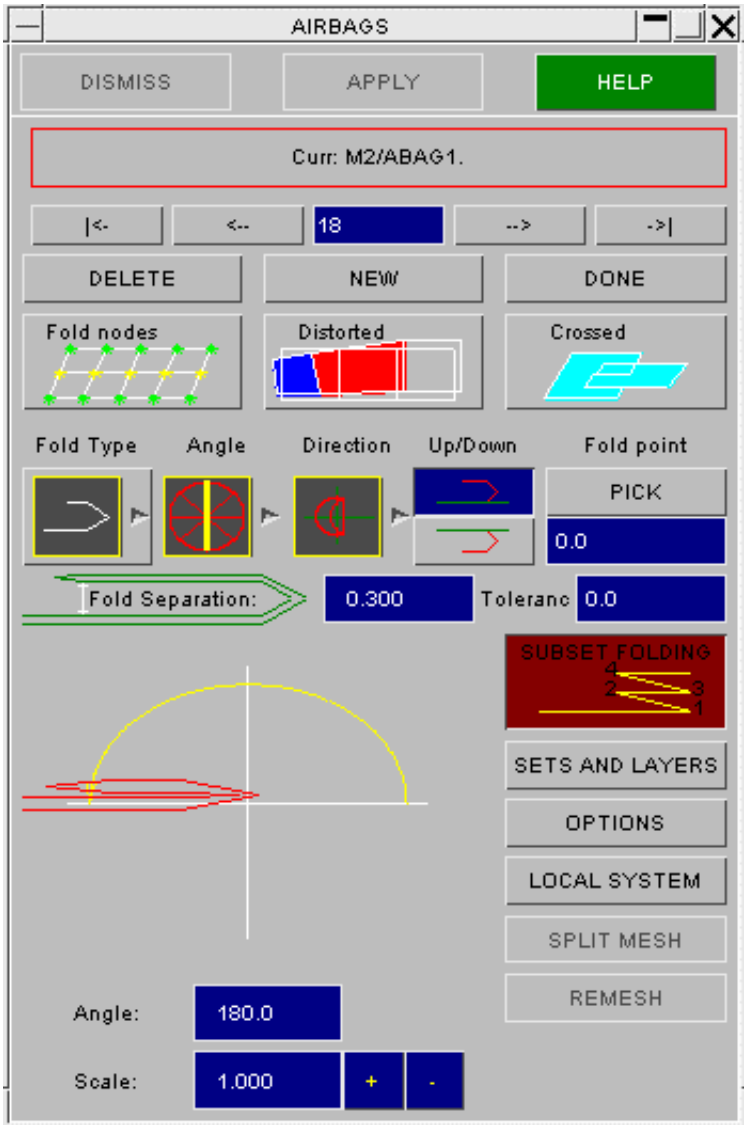


For airbags, the most common fold type is the thin fold. Material is creased sharply along a line of nodes to give a precisely defined shape.

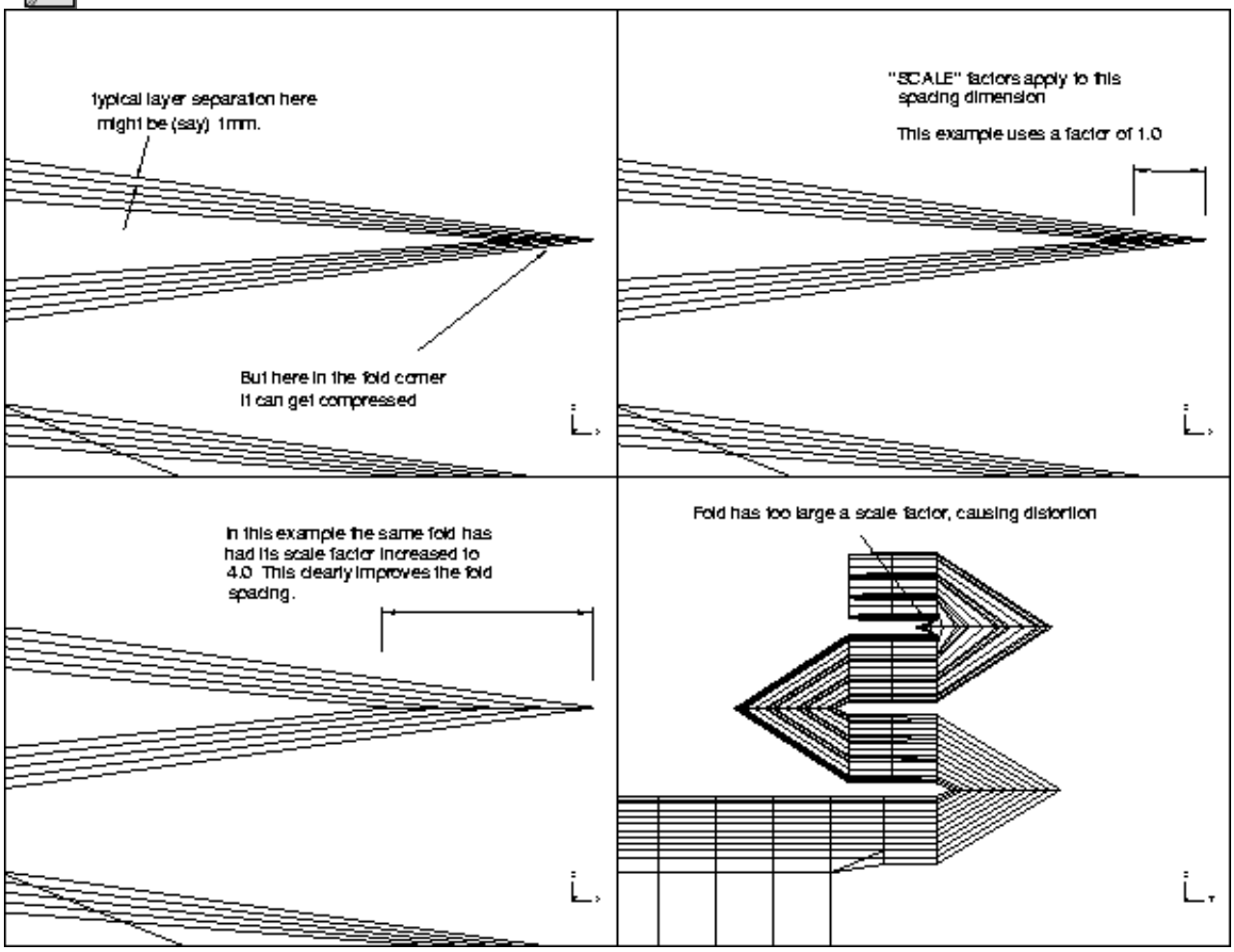
The definition and control panel for **thin** folds are shown in the figure below:

The panel shows the creation of a thin fold. the airbag has been folded from right to left - the right hand side ending up on top of the left hand side. The fold thickness is 0.3.

Typical thin folds, have a total fold angle of 180º with the centre portion rotated 90º though this is not a requirement. Angles larger than 180º are not permissible. These defaults are being used here. The thin fold graphic above shows that there is some pinching at the fold tip. The pinching can be reduced by increasing the **Scale** option to spread the layers at the fold tip.

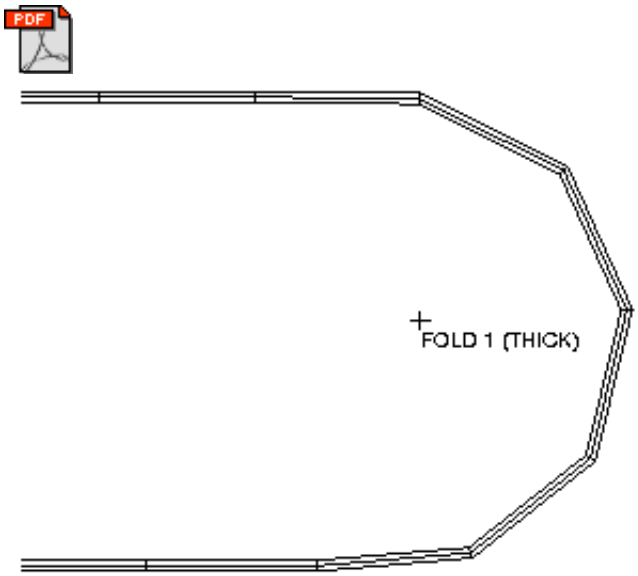


The figure below shows the effect of **Scale** on typical thin folds. It is generally desirable to try to select a value which causes the inner and outer material surfaces to be parallel, as this will lead to the contact algorithms being more reliable. However, the user should take care that this doesn't lead to gross local element deformations: as in the bottom right quadrant of the figure. This can also be visualised by using the **CROSSED** element button. If there are problems with element penetrations this will easily show it. Changing the scale factor will automatically update this plot so you can tell when any penetrations are eliminated.



By default all the elements and nodes selected for the Origami definition are included in the fold, but it is possible only to operate on a sub-set of these. The **SETS AND LAYERS** button can be used to select a subset to fold

Thick Fold (A radiused fold spanning > 1 element)

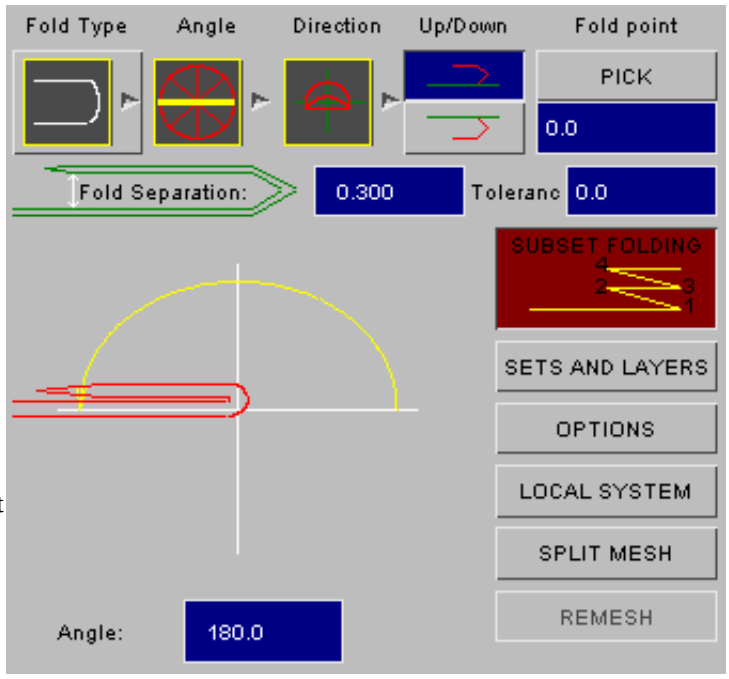


Thick folds are used when the mesh is extremely fine or when there are a large number of layers to be folded. In general the element size should be smaller than the fold radius or it is unlikely that a satisfactory fold will be created.

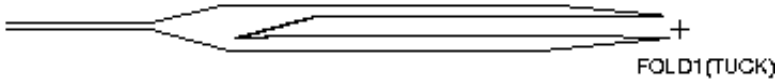
As this figure shows the effect is to create a radiused fold, using several elements, giving an effect similar to rolling the mesh round a circular former of the given radius.

The adjacent figure shows the fold creation menu - the material has been folded from right to left and the folded material is folded on top of the unfolded material. The only option which is available for a thick fold is the angle. In this figure the angle is set to 180º . The thick fold does not have a precise centre like the thin fold, but it does allow for arbitrary angles of orientation.

Usually, a fold point is located towards the packaging limits of a airbag. In this fold, the fold point is offset ahead of the centre point for the fold by the radius. The radius is equal to half the separation distance between layers. If more precise control over the selection of elements for this fold is needed, then the **advanced set selection** option in **SETS AND LAYERS** can be used to select the portion of the airbag to be folded. The **unfolded** set is used only to determine clearances in this fold type.



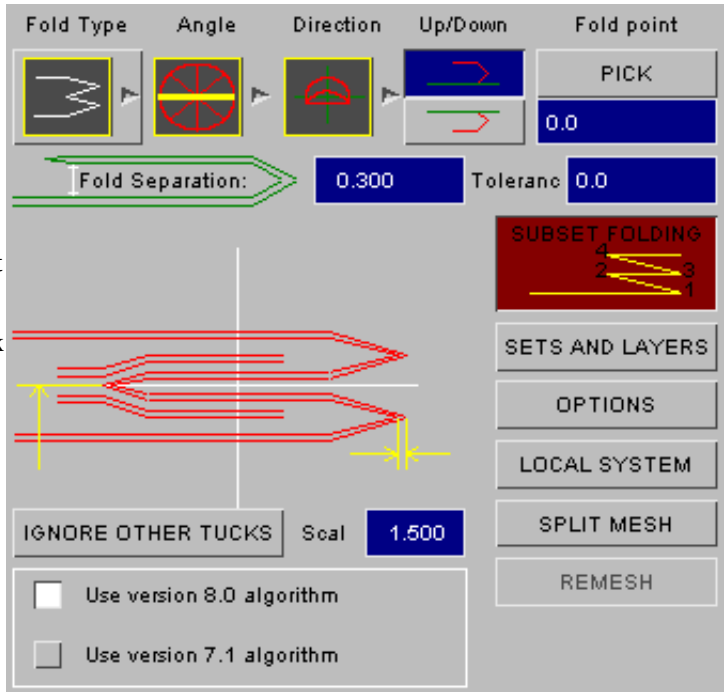
Tuck Fold (A thin fold tucked into the mesh centre)



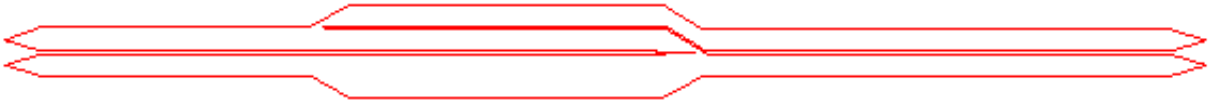
The tuck fold is also common in airbags. The material is folded inside the outer layers to form a "tuck". ("Up" and "Down" have no meaning here.)

The adjacent figure shows the tuck fold creation panel.

In version 8.0 a second tuck fold algorithm has been added. This is not meant to replace the version 7.1 tuck fold as there will be situations when the version 7.1 fold will perform better than the version 8.0 tuck fold. However the new version 8.0 tuck fold will perform much better in situations where two tuck folds interfere with each other. To illustrate the point the next two figures show an cross section through an airbag with 2 interfering tuck folds (one from each side of the bag) folded with the version 7.1 tuck fold and the version 8.0 tuck folds.



Two interfering tuck folds using the version 7.1 tuck fold algorithm

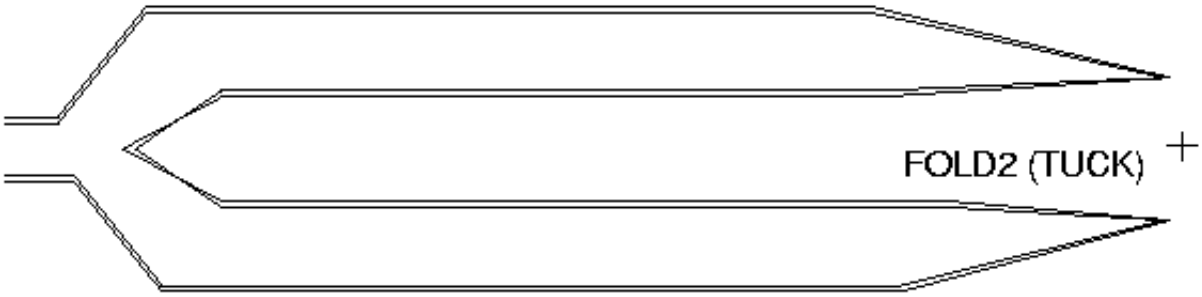


Two interfering tuck folds using the version 8.0 tuck fold algorithm

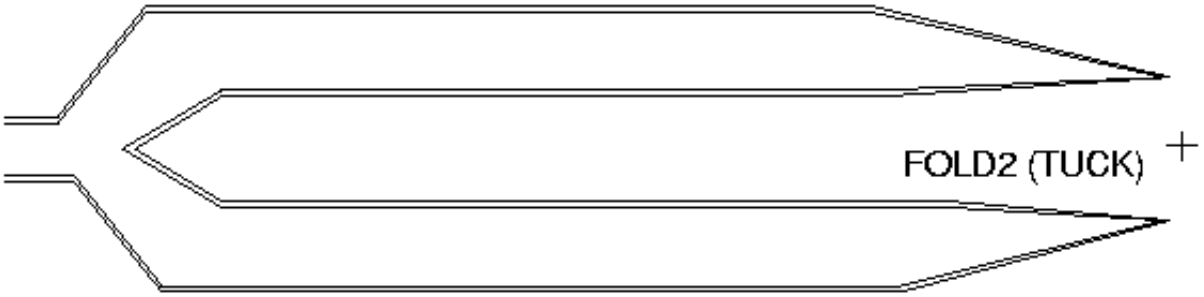
The tuck folds using the version 7.1 tuck folds penetrate through each other. If this airbag was deployed there would be problems with contacts and the airbag forming knots. Additionally the nodes on the fold tip are not in the correct place. The version 8.0 folds do not penetrate through each other and so this airbag will deploy correctly and the nodes at the tip of the fold are still in the correct position.

The default for tuck folds is to use the version 8.0 algorithm. If the fold cannot be performed with this algorithm you can still use the older 7.1 algorithm. This may be better for tuck folds which use multiple material thickness as there are some additional options which may help.

The following two figures illustrate the use of these options for the version 7.1 algorithm, the left hand figure shows that problems can occur with penetrations when using tuck folds for multiple layers. If problems occur then selecting **[>>]** (double layer mode) may help resolve the problem (right hand figure). But, the double layered mode is only valid if the fold tip lies along a line of nodes. If it does not then the single layered mode should be used.



Penetrations at tip

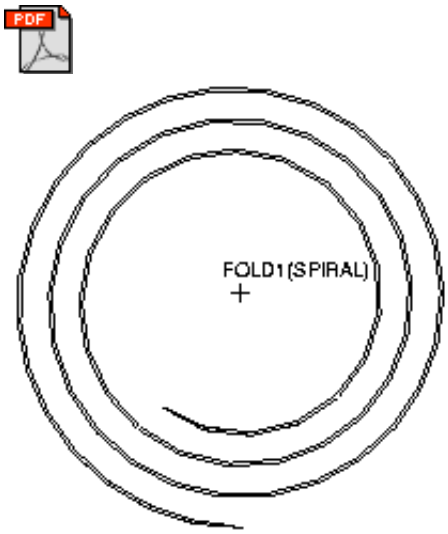


No Penetrations

By default, the folder attempts to locate the middle fibre of the unfolded material. Everything above the middle fibre is pushed up and everything below the middle fibre is pushed down so that the tip can be inserted and clearance maintained. This can be overridden if PRIMER selects the wrong location using **ZSPLIT** which prompts the user to pick two nodes. These define a plane whose normal vector starts mid-way between these nodes. The layers are then separated above and below this plane.

SCALE allows the user to reducing the pinching that occurs at the fold tip by increasing the node separation.

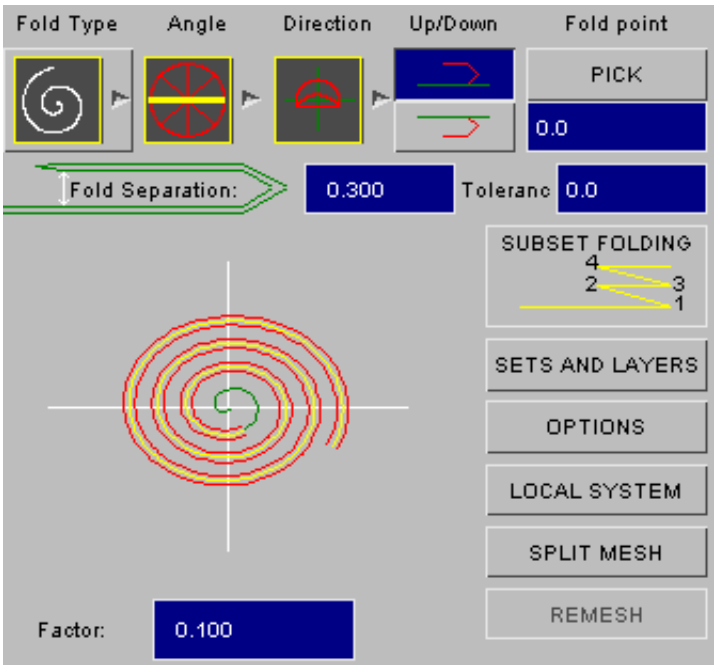
Spiral Fold (Rolling layers into a spiral)



The Spiral fold is used to roll up a flat bag.

An Archimedean spiral (radius is proportional to angle) is used, and PRIMER attempts to keep the characteristic element length constant at the middle fibre of the bag.

The adjacent figure shows the fold definition and options menu for the spiral fold.



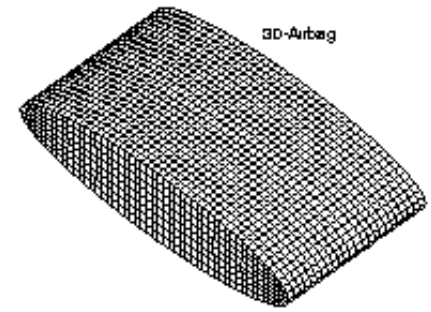
Controlling the spiral internal radius

PRIMER tries to preserve a constant arc length for the middle fibre of the bag as it is rolled, which presents problems at the spiral centre where the radius tends to zero and will always be smaller than the elements.

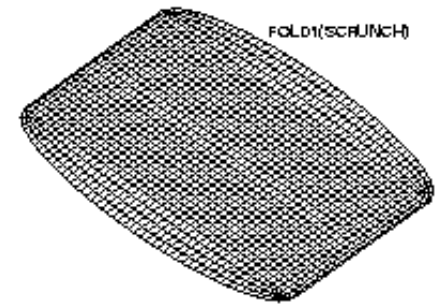
In practice a rolled airbag has a finite thickness and therefore will not use the early portion of this curve. PRIMER uses a **Factor**, of the original arc length of the airbag to specify that portion of the curve that is not to be used. For example, a Factor of 0.5 increases the total arc length to 1.5 times the airbags arc length and leaves the first 0.5 times this arc length unused. A factor of 0 would have no unused portion and the airbag would be rolled from the spiral centre. The options menu shows how much of the spiral is mapped and how much is unmapped. The default factor is 0.1.

The user can also specify a subset of the airbag to be folded using the **SETS AND LAYERS** options as for other types.

Scrunch Fold (Compressing a 3D bag to a flat shape)



Before scrunch



After scrunch

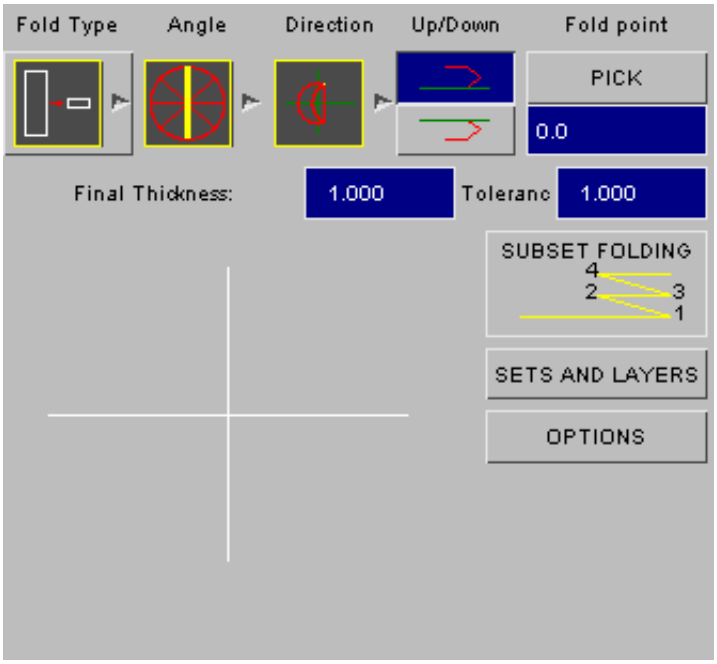
The adjacent figures show a 3D airbag being **scrunch**ed to a flat (2D) shape.

The option of splaying the sides out has been used.

This fold type can accomplish two separate functions:

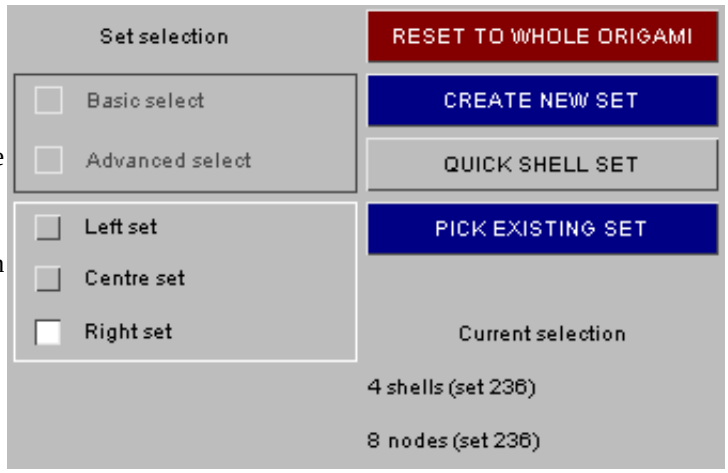
1. It can simply scale an existing bag in the local Z-direction so that it has a smaller final thickness.
2. It can flatten a 3D airbag so that the 2D folder (thin, thick etc) can be used. The bag is reduced in the Z-direction and the sides are pushed out.

The **scrunch** fold definition and options are shown in the adjacent figure.



The user must tell PRIMER which elements form the side to be pulled out. This is done using the **Left** and **Right** sets from the **SETS AND LAYERS** menu. If neither of these sets is chosen, then a simple scaling is used. In the case shown above this could lead to the vertical elements having a zero side length (which may not be illegal if airbag reference geometry is used during the analysis).

When forcing the sides outwards, the top and bottom of the bag are located above and below a side node. The node is then pushed outwards based on the nearest distance to the top or bottom. When using this capability for pushing out side walls, it is important how the bag is oriented. The axis of the cylinder must be parallel with the local X-axis. The sides must be in the YZ plane.



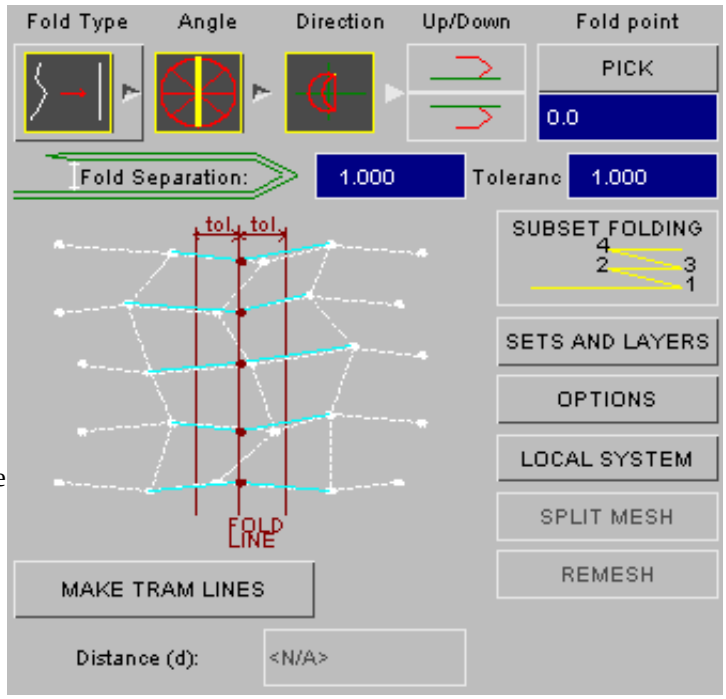
FOLD_POINT has no effect here the ORIGAMI is scrunched about the local z=0 plane.

Align Fold (Aligning nodes on a fold line)

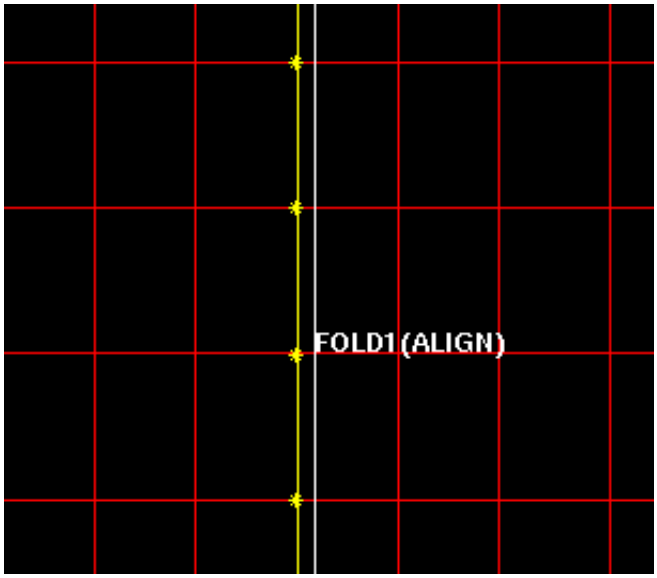
The align fold is used for projecting nodes onto a fold line. This is very useful if the nodes do not line up exactly on a fold line or if the fold line is in slightly the wrong position. The adjacent figure shows the align fold creation menu. In this example the **TRAM LINES** option has been set. This can be turned on or off by using the **MAKE TRAM LINES** button.

The normal options for selecting the nodes for the align fold can be used. You can select the nodes you want to fold using the **SETS AND LAYERS** options. Alternatively, you can use the nodes which were folded in the previous fold by using the **SUBSET FOLDING** option.

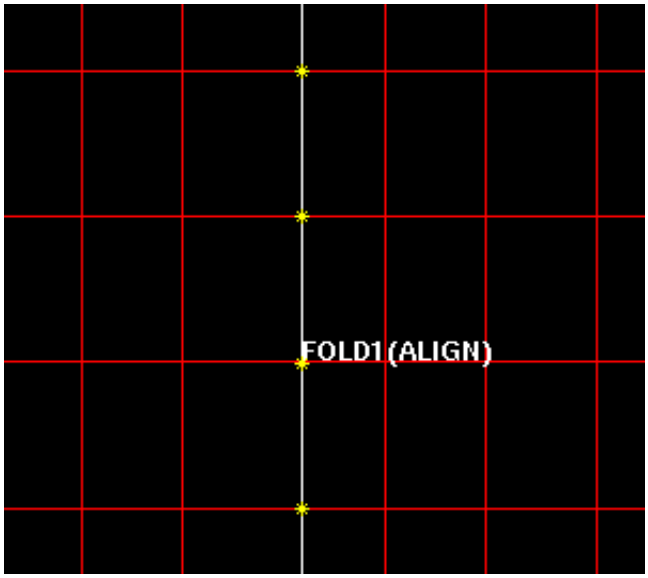
For both options you give a fold point and a fold angle to define the fold line. The nodes which will be aligned are highlighted on the screen. The **Tolerance** option can be used to increase the tolerance for selecting the nodes which will be moved onto the fold line.



The default options for the align fold just move the selected nodes onto the fold line. The following figures show the effect of this option with pictures before and after the align fold.



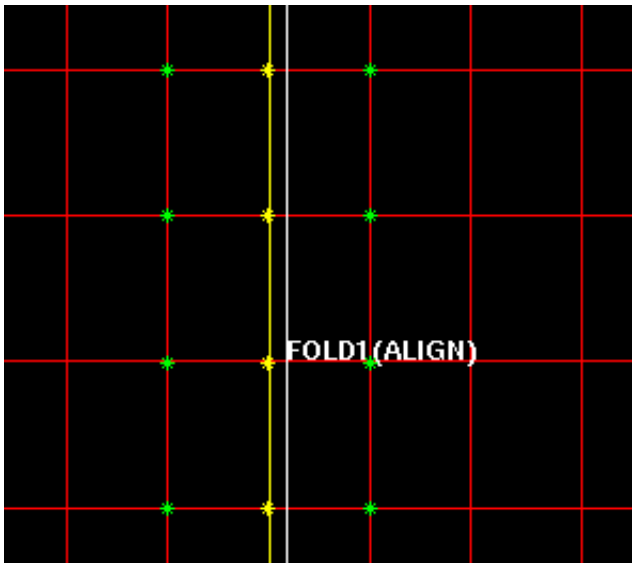
Before align fold



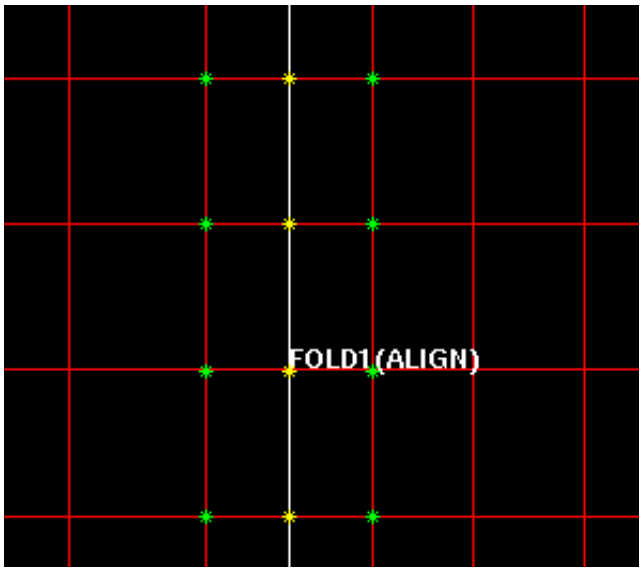
After align fold

Folds work best if the nodes adjacent to the fold line are a constant distance from the fold line. The **MAKE TRAM LINES** option does this. If the option is set then if a node from an element is moved onto a fold line the other nodes on the element which are not on the fold line will be moved to be a constant distance away. You can enter the distance to position the nodes with the **Distance** box. The diagram in the panel updates to show you that you are using the tramlines option.

The following figures show the effect of the tramlines option with pictures before and after the align fold.



Before align fold



After align fold

With the tramlines option set the adjacent nodes are a fixed distance from the fold line. Compare the above pictures to the ones with the tramlines option not set.

Scale Fold

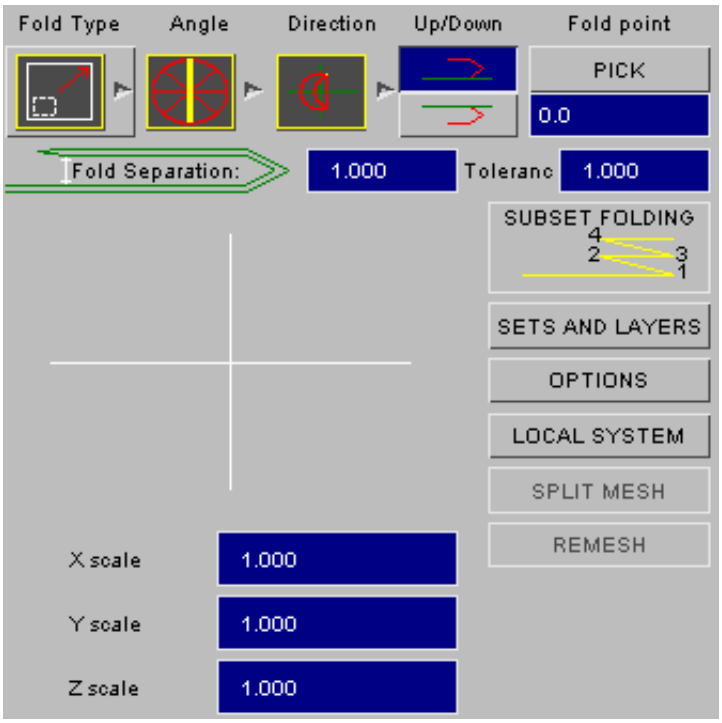
The scale fold is used for shrinking or enlarging parts of the origami.

The scaling is done in the local co-ordinate system of the fold, NOT the global co-ordinate system.

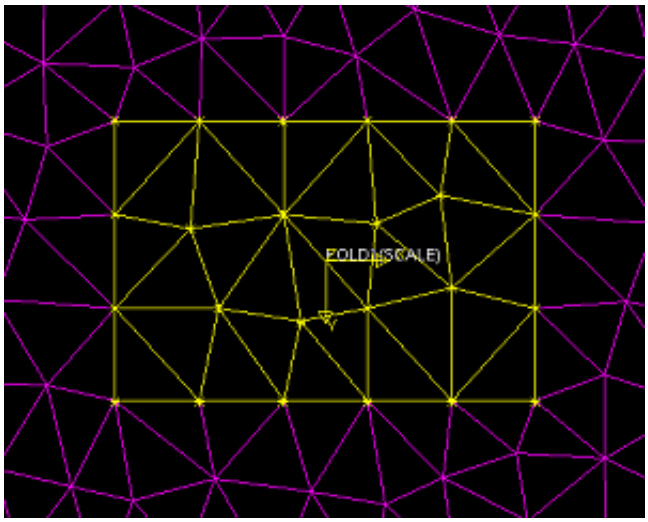
To help you visualise the local co-ordinate system the axis triad is drawn on the airbag.

The nodes/elements that you want to scale are selected in the normal way using sets and layers as required.

You can give separate scale factors for the X, Y and Z axes of the fold.



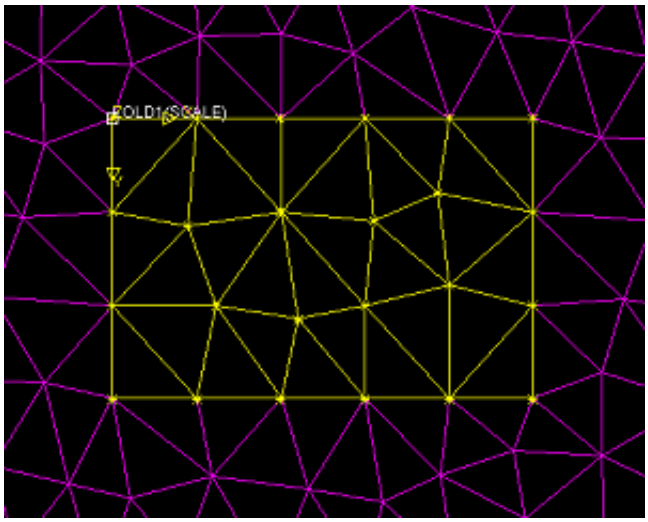
Examples of scale fold



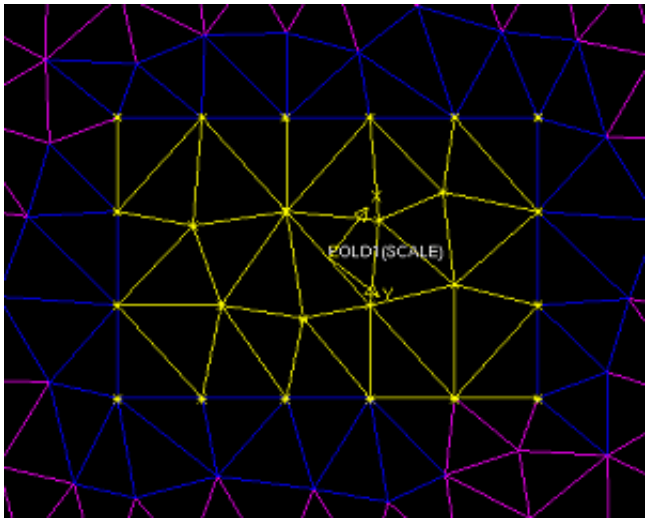
The yellow portion of the airbag has been selected using the sets and layers option (the fold nodes are shown highlighted by yellow stars).

The local co-ordinate system is shown on the diagram.

In this example the fold point is at the default (0) position.



In this example the fold point has been moved to the top left of the yellow portion by **PICK**ing a new fold point.



In this example the fold coordinate system has been changed by using the **LOCAL SYSTEM** function. Note that the directions of the local axes have changed but the origin is still at 0. To change the origin a new fold position would need to be picked.

Translate Fold

The translate fold is used for moving parts of the origami.

The translation is done in the local co-ordinate system of the fold, NOT the global co-ordinate system.

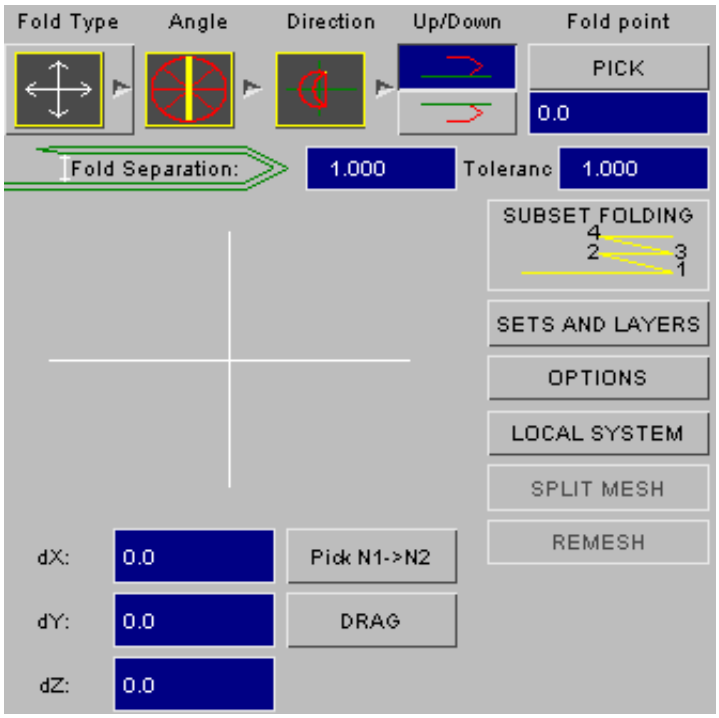
To help you visualise the local co-ordinate system the axis triad is drawn on the airbag.

The local axes can be changed using the **LOCAL SYSTEM**. The fold point (local axes origin) can be changed by selecting a new Fold point. See the examples for the scale fold as the method is identical.

The nodes/elements that you want to translate are selected in the normal way using sets and layers as required.

You can translate the airbag by one of 3 methods:

1. You can type in separate X, Y and Z distances in the text boxes.
2. You can pick 2 nodes using **Pick N1 -> N2** to define the distance.
3. You can drag the selection interactively using **DRAG**.



Press **DRAG**. The window will change as shown on the right. All other functions are unavailable while dragging.

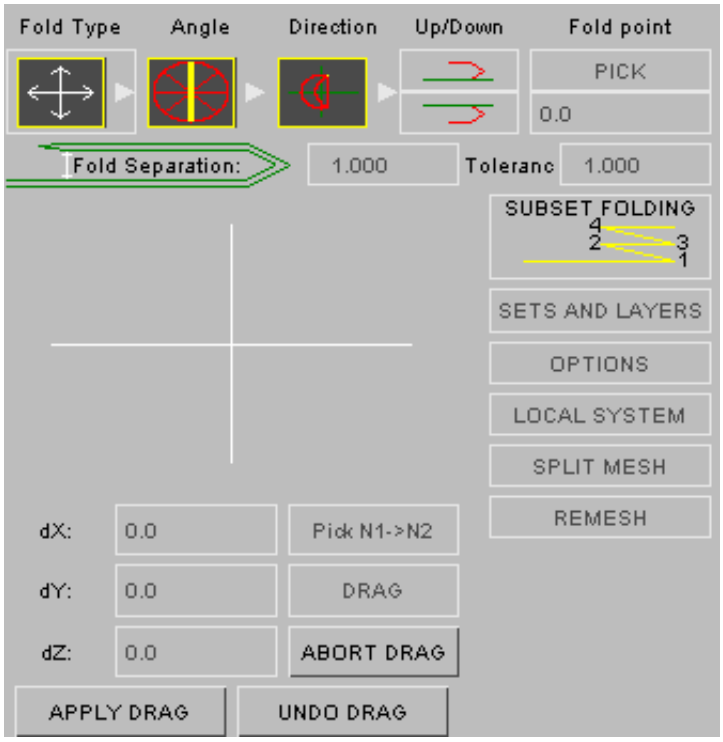
Click and drag with the mouse and the selection will be moved on the screen and the distance boxes updated. When the mouse button is released the screen will be redrawn. Make sure the Fold type button is toggled to show the translate fold type (not the null fold type) or you will not see the translation!

You can only drag in the XY plane of the fold. You can continue dragging the selection until you are satisfied.

You can cancel dragging at any time by pressing **ABORT DRAG**.

You can start dragging again (undoing the last drag you made) by pressing **UNDO DRAG**.

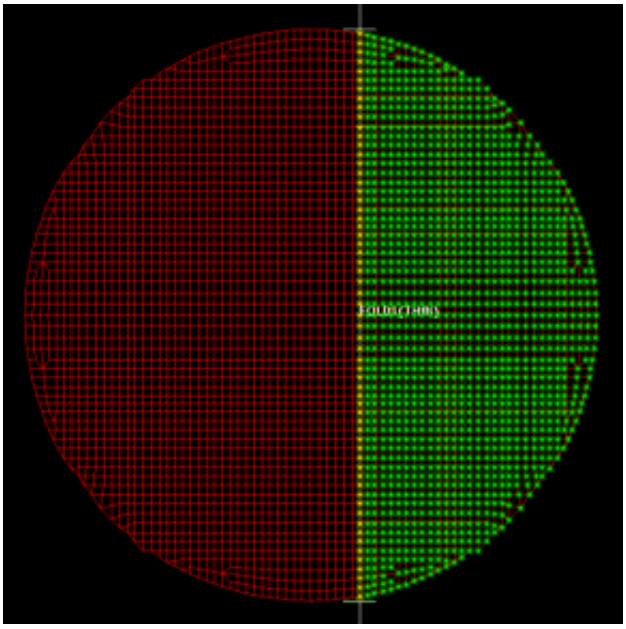
To finish the drag operation press **APPLY DRAG**. The distances will be saved.



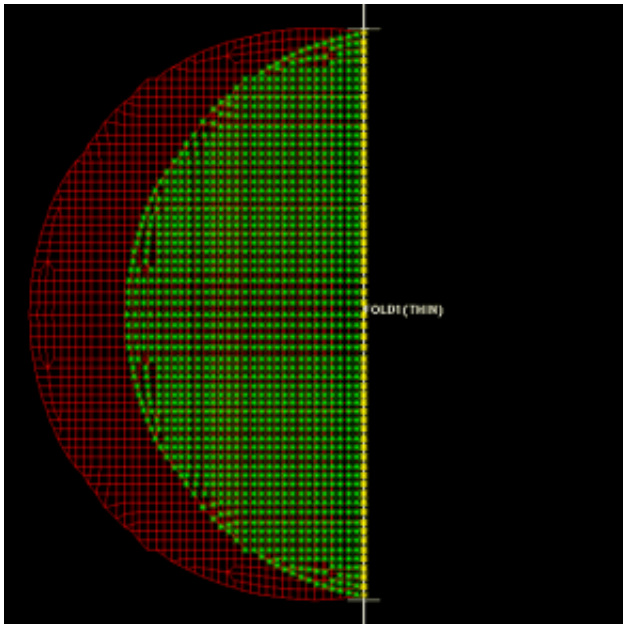
6.1.11 Subset Folding

Subset folding can be used with thin, thick, version 8.0 tuck folds and align folds. It can make the process of making multiple folds considerably easier. Every time you do a fold a list of the nodes which are folded is saved. If you now want to do a new fold in which the nodes you want to fold are a subset of the previous fold (i.e. all the nodes were folded in the previous fold) instead of defining a set or layers you can just press the **SUBSET FOLDING** button and the nodes from the previous fold will be used as the input to this fold. The following figures illustrate the use of subset folding when creating thin folds.

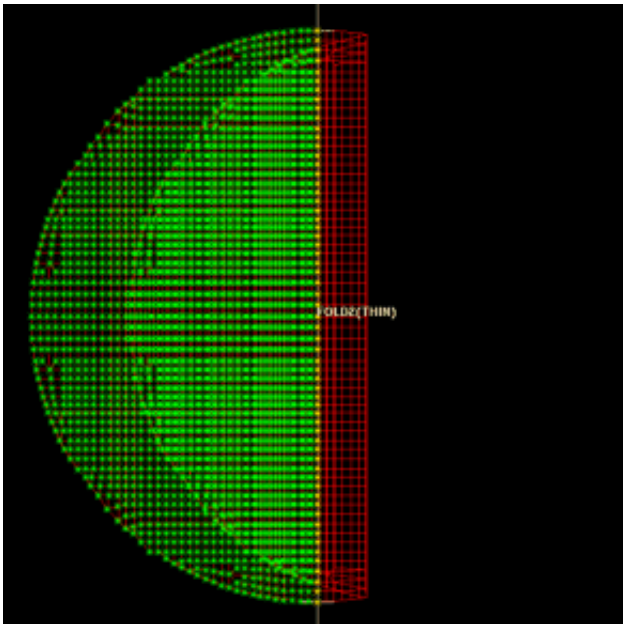
a)



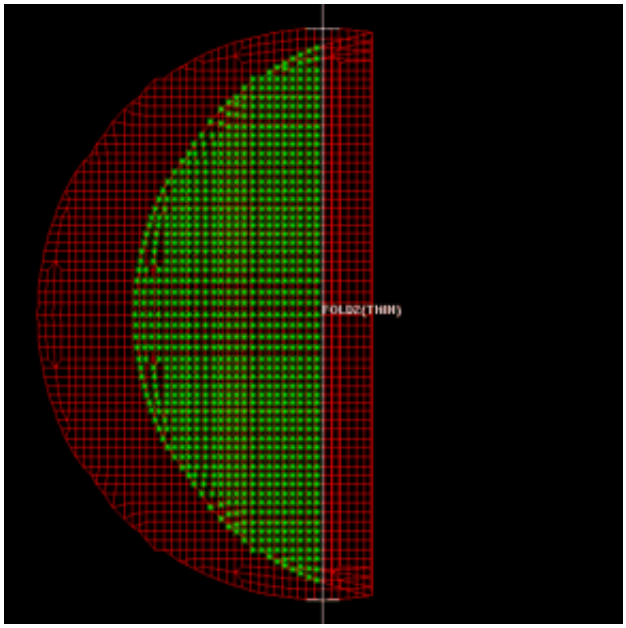
b)



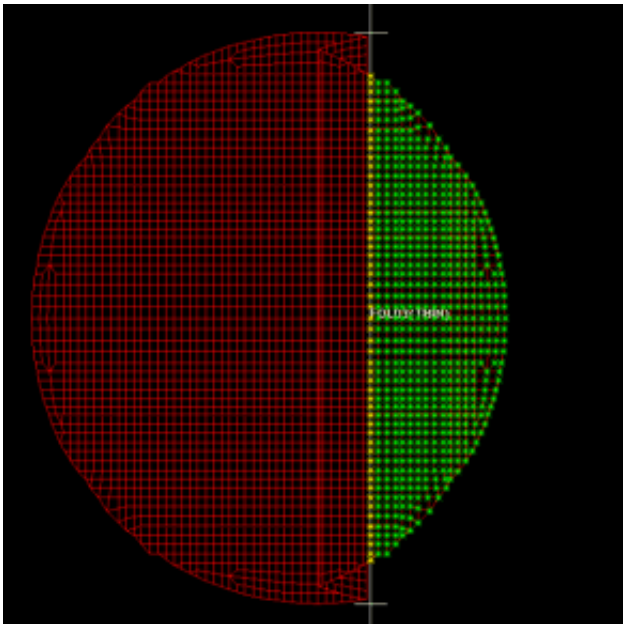
c)



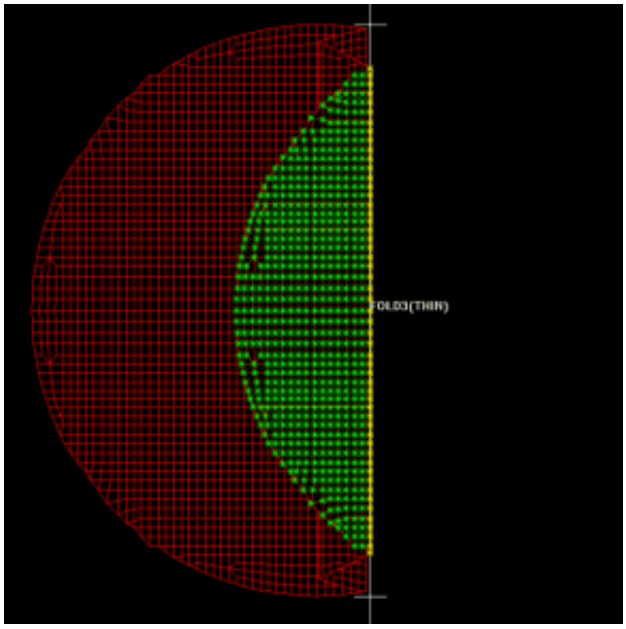
d)



e)



f)



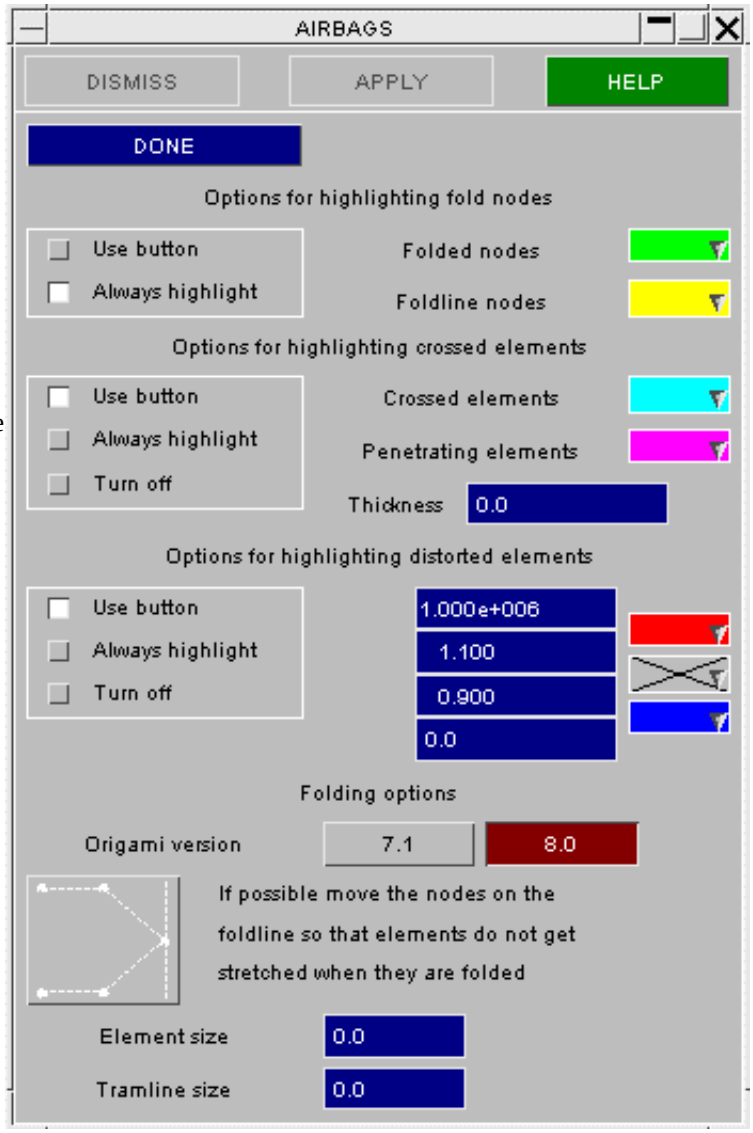
A new fold (fold 1) is created on the airbag folding from right to left. a) shows the airbag before the fold, b) shows the airbag after the fold. If a new fold (fold 2) is now created folding from left to right by default everything to the left of the fold point will be folded (c). At this point you could define a node set or use layers to select the upper layer of the airbag to fold. Instead you can press the **SUBSET FOLDING** button. Now the nodes which will be used for fold 2 are the nodes which were folded in the first fold (d). If this fold is completed and the folding process continued with subset folding then when the third fold is defined (e) the nodes are automatically selected and the fold direction swapped from forward to reverse. (f) shows the airbag after the third fold. If the folding process was continued and a fourth fold created using subset folding the fold direction would automatically swap over to be forward. This process can be used to very quickly create >zig-zag= folds on an airbag. It is much quicker as no sets or layers need to be created when doing the folds. You can turn subset folding off at any time and continue folding by the normal methods.

6.1.12 Folder Options

The **FOLDER OPTIONS** panel enables you to set various options which alter the way the folder works and what is drawn.

For fold nodes, crossed and penetrating elements and distorted elements you can individually choose to always plot them or to only plot them if the appropriate button is pressed. This option is set by using the radio buttons and setting the option **Use button** or **Always highlight**.

As the distorted and crossed element checking is very complicated it can take some time. If the delay when folding is unacceptably slow then these features can be turned off by using the **Turn off** option. If this is done then you will not be able to plot the distorted or crossed elements until one of the other options is chosen again.

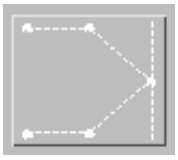


For each type of entity (e.g. Foldline nodes) you can also choose the colour that the entity is drawn in (or if it drawn at all) by using the popup menus. Each popup menu brings up a selection of colours to choose.

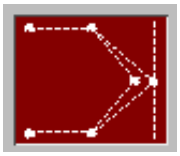
The colour popup allows you to choose from 15 basic colours. The cross at the bottom right of the panel (the X button) stops the entity being drawn completely. E.g. in the above figure, element distortion between 0.9 and 1.1 will not be highlighted.

The element thickness which is used for checking penetration can be altered by typing in a new thickness in the text box. The ranges for distorted elements can similarly be changed by typing new values in the text boxes.

Version 8.0 has some new fold options such as a new tuck fold algorithm and an element stretching option. As these options could change the way an existing origami is refolded there is an option to say if your origami will be folded using version 7.1 techniques or version 8.0 techniques. By default any new origami you define will be a version 8.0 origami and you will have access to these new functions. If you read in an old origami (version 7.1 and before) the origami version is set to be 7.1. A warning will be output when you first select the origami to say that you are using an old origami. The reason a warning is given is to bring to your attention that if you change the origami to be version 8.0 the folds may change. If you have a previously correlated airbag then leave the version at 7.1 and the folds will not be changed. You will not be able to use the new functions until the origami is changed to be version 8.0.



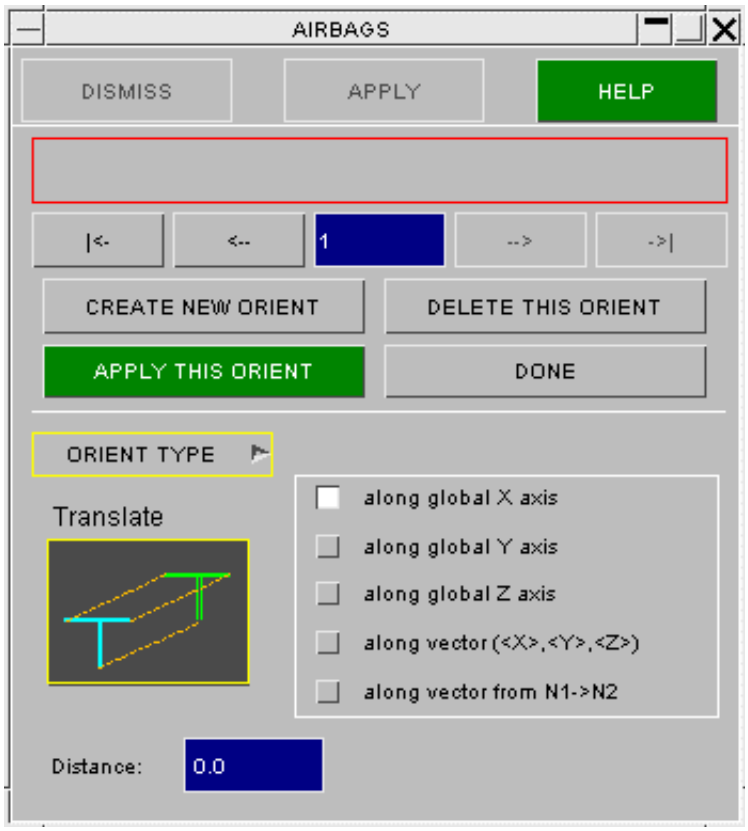
The final button on the panel is used to try to stop stretching of elements during folding. This only applies to thin folds. By default elements will be stretched as the centre plane of the elements which are being folded is placed at the fold point. Elements on one side of the plane will get smaller, elements on the other side of the plane will get larger.



If the button is pressed then the folder will attempt to stop any elements from stretching. To do this the fold point has to be moved slightly. For folds which are less than 180º this can generally be done. If the fold is a 180º fold then if you try to fold multiple layers which are thick a point is reached when the fold cannot be done without stretching any elements. If the thickness is less than this then the stretching will be eliminated. If the thickness is more than this then the stretching will be reduced as much as possible.

6.1.13 Positioning Folded Airbags

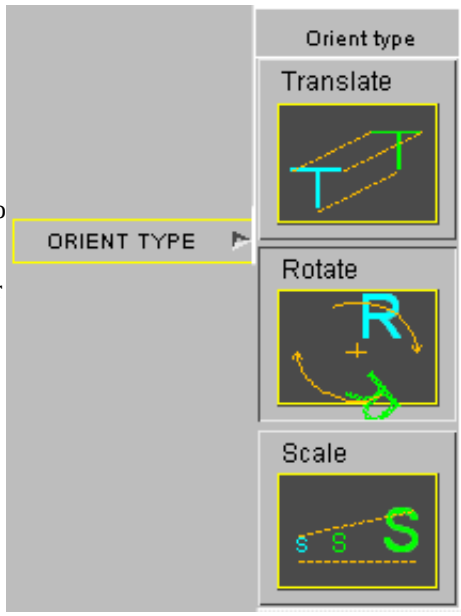
Once the airbag has been folded it can be positioned where you want it by using the **POSITION FOLDED BAG** option. The reason for using this rather than the normal **ORIENT** option in PRIMER is so that you do not lose the orientations if you refold the airbag. The orientations are created and viewed just like folds so you can create, delete or edit any orientation. They are cumulative transformations on the folded airbag. For example if you define a translation and then a rotation first the translation will be applied and then the rotation so the order of the orientations is important.



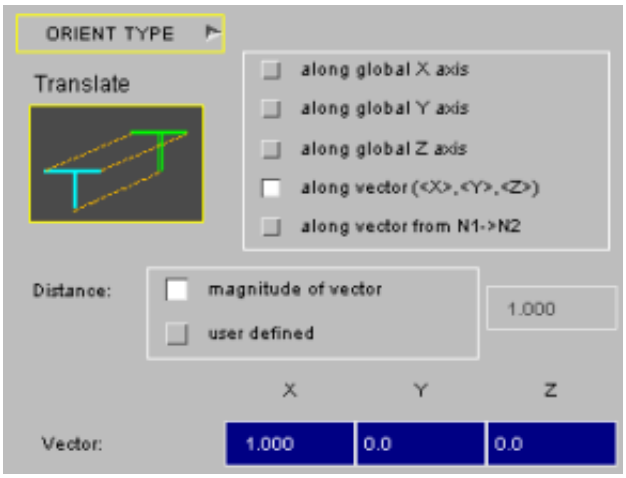
The left hand figure shows the initial window when you enter the positioning section. As there are no orientations the top row of buttons are greyed out. As orientations are defined these buttons can be used to move backwards and forwards through the orientations in exactly the same way as folds in the **SET FOLD** menu. The **DONE** button will return to the main folding window. To create a new orientation the **CREATE NEW ORIENT** button can be used. The default new orientation type is a translation (right hand figure).

At any time an orientation can be deleted by using the **DELETE THIS ORIENT** button. Once the necessary values have been given for the orientation (for example the translation distance) the orientation can be applied by using **APPLY THIS ORIENT**. Once applied the button will change to **UNDO THIS ORIENT**. This button can be used to continually toggle between the unapplied and applied view of the orientation.

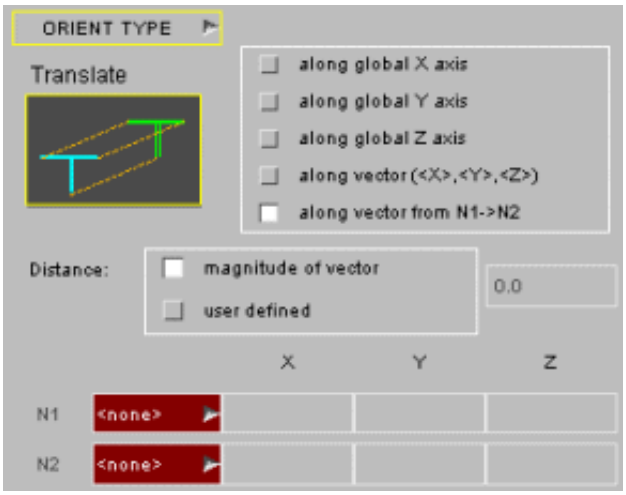
Three different orientations are available; translation, rotation and scaling. To change an orientation type use the top row of buttons (**|<** , **<** , **>** , **>|**) to select the orientation number to change and then use the **ORIENT TYPE** popup menu (figure on right) and choose either **TRANSLATE**, **ROTATE** or **SCALE**. The different options for translation, rotation and scaling are described in the following sections.



Translation



Translation along a vector <X>,<Y>,<Z>



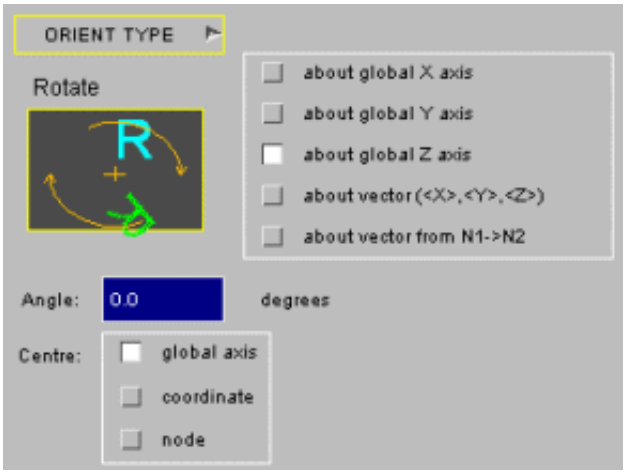
Translation along a vector from N1->N2

There are 5 different translation options. The first 3 are translating along the global X, Y or Z axis. For these options the translation distance must be given by typing the value in the text box.

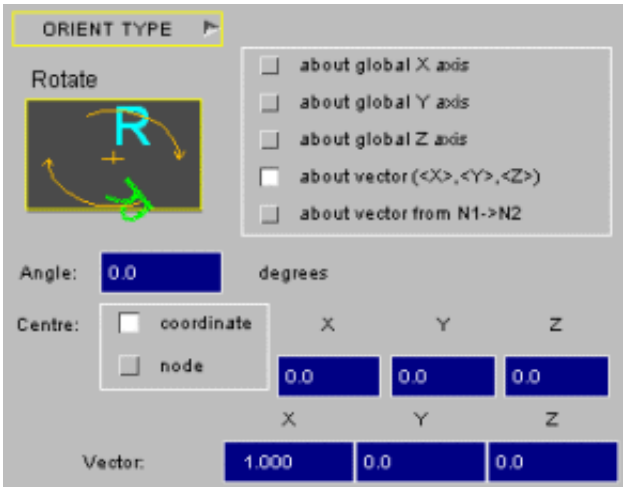
The fourth option allows you to give a vector to translate along by typing in the X, Y and Z components of the vector (left hand figure). The distance that the airbag is translated along this vector can either be a user defined distance or the magnitude of the vector.

The fifth option translates the airbag along a vector defined from N1 to N2 (right hand figure). The 2 nodes can be typed in or picked using the popup menus. The translation distance can either be a user defined distance or the magnitude of the vector.

Rotation



Rotation about a global axis



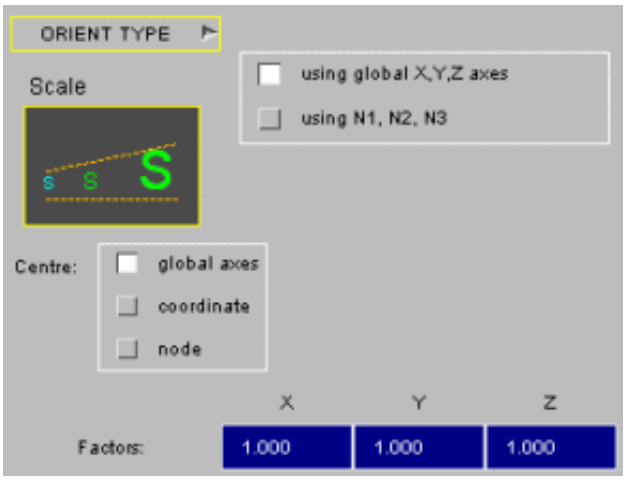
Rotation about a vector

There are 5 different rotation options. The first 3 are rotating about the global X, Y or Z axis (left hand figure). For these options the rotation angle must be given by typing the value in the text box. There are 3 possible methods for specifying the centre of rotation. The centre can be the global axis, about a coordinate which you can type in, or about a node number which you can type in or select using the popup menu.

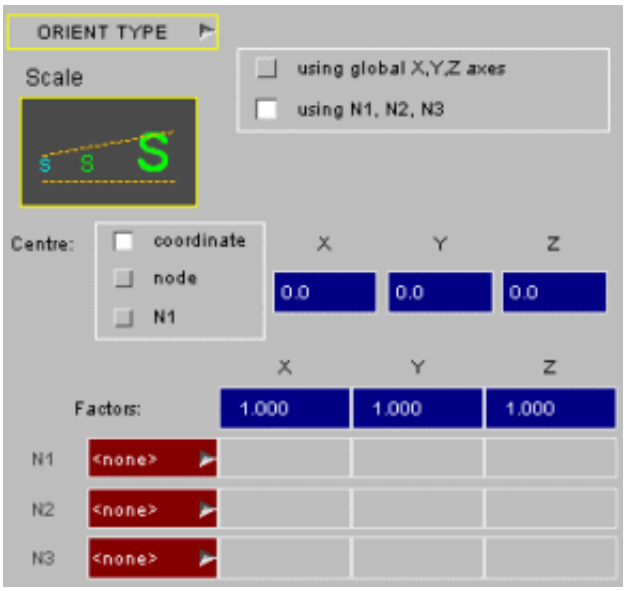
The fourth option allows you to give a vector to rotate about by typing in the X, Y and Z components of the vector (right hand figure). The centre of the rotation can either be a coordinate or a node.

The fifth option rotates the airbag about a vector defined from N1 to N2. The 2 nodes can be typed in or picked using the popup menus. The centre of rotation can be a coordinate, N1 on the vector or another node.

Scaling



Scaling using global axes



Scaling using local axes

There are 2 methods available for scaling the airbag. The first method allows you to scale the airbag in the global axes (left hand figure). Different scale factors can be used for the X, Y and Z directions if necessary. The centre for the scaling operation can be defined as either the global origin (0, 0, 0), a coordinate which you can specify by typing in the X, Y and Z values or a node number which you can pick or select by typing the number.

The second scaling method allows you to scale an airbag in directions other than the global axes by using three nodes. The three nodes are used to define a local coordinate system for the scaling. N1 is the origin for the local coordinate system. The vector from N1 to N2 is the local x axis. N3 defines another point which lies in the xy plane. This method is the same as **`*DEFINE_COORDINATE_NODES`** and is used a lot in LS-Dyna. For further information look at the user guide. The 3 nodes can be typed in or picked using the popup menus. As for the global scaling option the centre can be the origin, a coordinate or a node.

6.1.14 Saving and Reading ORGIAMI/Fold Definitions

There is no capability to directly read and write the ORIGAMI and FOLD definitions to a file. However, the information is stored in a section labelled ***ORIGAMI** at the end of the LS-DYNA (after ***END**) keyword file which includes all the fold information. This is automatically added when a LS-DYNA keyword file is written. The format of these is included in the comments (see also Appendix III). When read back into PRIMER, these definitions are available to the airbag folder.

To stop any ***ORIGAMI**, ***FOLD** and ***ORIENT** definitions from being output, the ORIGAMI definitions must be deleted.

Although the ***ORIGAMI**, ***FOLD** and ***ORIENT** definitions are available in ASCII form at the end of the LS-DYNA input, it is recommended that hand editing be avoided as it is error prone: to modify fold and orient definitions read them back into PRIMER.

Note also that ***ORIGAMI** definitions should not be separated from their "parent" input decks, since they make reference to nodes, sets and coordinate systems within those decks.

6.1.15 Additional Airbag Folding Notes

The following will help users to fold airbags successfully

- For **thin** and **tuck** folds make sure that mesh lines follow fold lines exactly, (or at least within **Tolerance**). To improve accuracy of these folds it is usually important that the mesh lines adjacent to a fold line are also straight and have a constant spacing (perpendicular to the fold) from the fold mesh line. This is not so critical for other fold types. If your mesh lines do not follow the fold lines exactly this can easily be fixed by using an **ALIGN** fold.
- Be sure that the airbag does not have any cuts: it should be a closed surface. Circular holes are not a problem, but there may be computational problems if there any internal free edges.
- Thick folds and spiral folds can result in penetrations between shells on different layers. These need to be done selectively and the radius may need to be increased to avoid penetrations.
- Plan ahead if possible. Frequently, there are many different orders in which folds can be done which will result in the same final folded configuration. One order is usually much easier to accomplish than the opposite order for complex bags. If difficulty results from trying to fold a bag in one order, then perhaps try the opposite order. Subset folding can make the folding process much easier but this can only be used when the nodes for a fold are a subset of the nodes from the last fold (i.e. if the fold order in the bag is from the centre towards the edge, not from the edge towards the centre).
- Be sure always to select **NEW** before selecting options for a new fold.
- If necessary a fold can be **DELETE**d and the user can start it again if something goes wrong.
- If nothing appears to happen when creating a fold ensure that sets, carried over from the previous fold, are not defined in this definition. This can happen as much of the previous fold’s data is carried across when creating a new fold. Simply go to the **SETS AND LAYERS** feature and change the set or layer definition.

6.1.16 Folding Example

The above may sound somewhat complicated. In fact the easiest way to learn is to try the process and see what the various features do. To help this learning process the user should follow the example shown in Appendix IV.

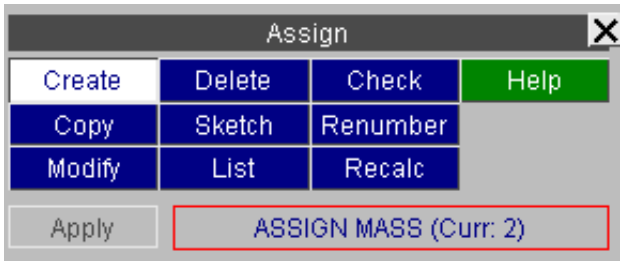
6.2 ASSIGN MASS

The assign mass panel allows you to add mass and change the centre of gravity of a group, part-set or assembly. This is done by adding lumped masses on the nodes in the group (or a subset of the group if you use the subgroup option) or by creating/modifiying *ELEMENT_MASS_PART(ADD) definitions. Movement of CofG is more restricted if using the EMP method but the principles of mass distribution are the same.

A new assign mass may be created using either method, and an existing one may be swapped from one to the other.

It is impossible to assign mass to nodes of a part which is defined with a **`*PART_INERTIA`** card as the lumped masses will be ignored by LS-DYNA. However, this does not mean that part inertias cannot be present in a massing up operation. They can be, but must be wholly contained in the group to be valid and included in the mass calculation. See the part inertias section in the problems below for more details.

Similarly in part mode, Primer will expect any *ELEMENT_MASS_PART_SET(ADD) definitions to be fully contained within the group,etc. *ELEMENT_MASS_PART_FINMASS definitions are analogous to *ELEMENT_MASS_NODE_SET in the node method and will be locked against change of mass.



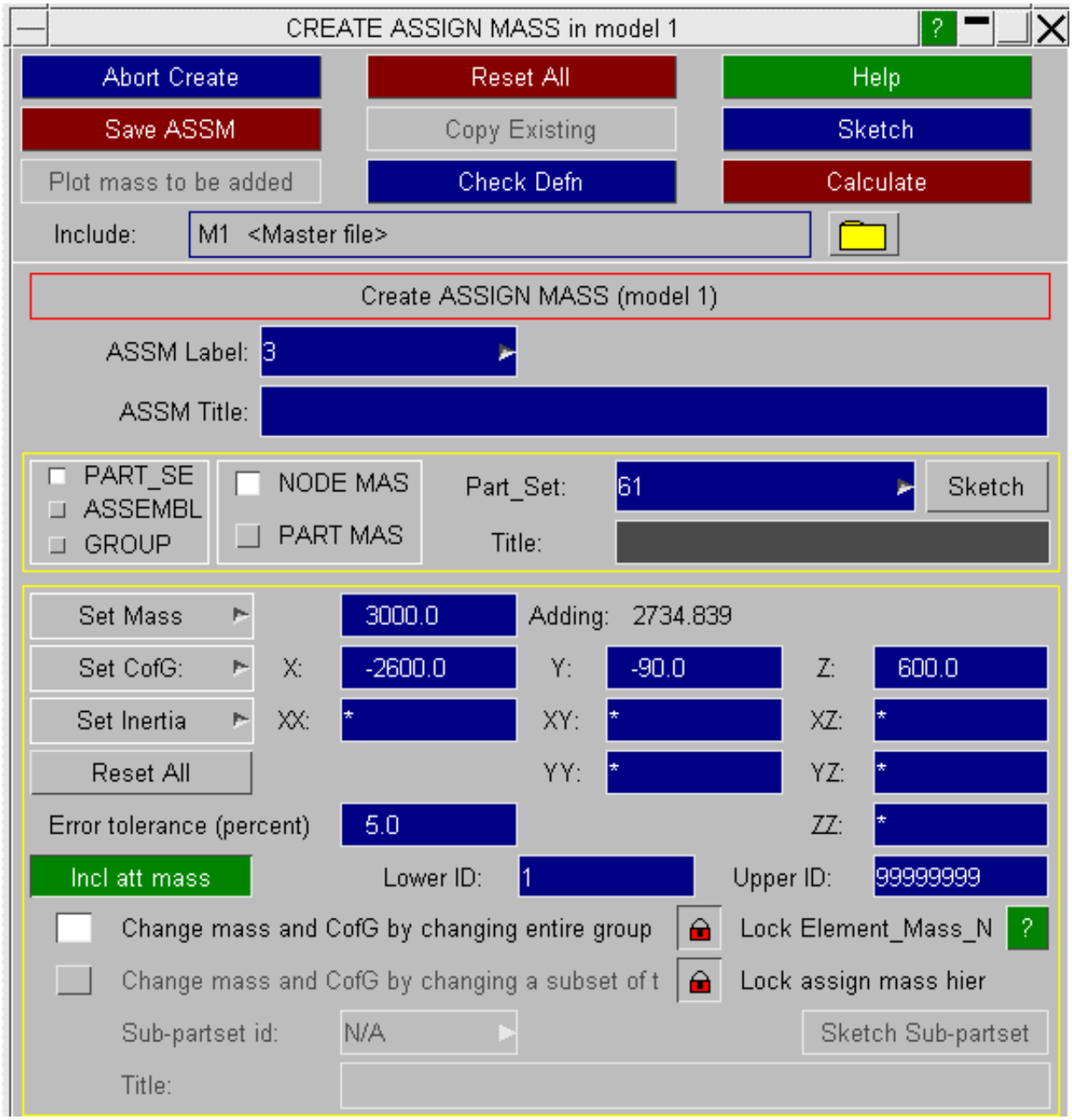
Basic assign mass operation

- Creating
- modifying
- recalculating

Advanced assign mass operation

- Including part inertias in the assign mass operation
- Hierarchy of assign mass
- Problems with assign mass
- Controlling suppression of text box warnings

Creating an Assign mass



Assign mass may be used to add mass to a group of entities (typically a set of parts or part assembly) or to achieve a target mass (which must exceed their native mass). The drop-down under **Set Mass** will set this mode.

The user needs to select (a) the group which constitutes that mass of interest and (b) the group to which mass is to be added. (b) may be a sub-group of (a) or both groups may be the same (as in the above example). By default, Assign mass is hierarchical, meaning that ASSM at label n+1 will not apply mass to elements massed by ASSM at label n. Thus the definitions should be ordered component (e.g.wheel), sub-assenbly (e.g. fron suspension), whole vehicle. This feature may be unlocked to allow ASSM to add mass to definitions at lower labels.

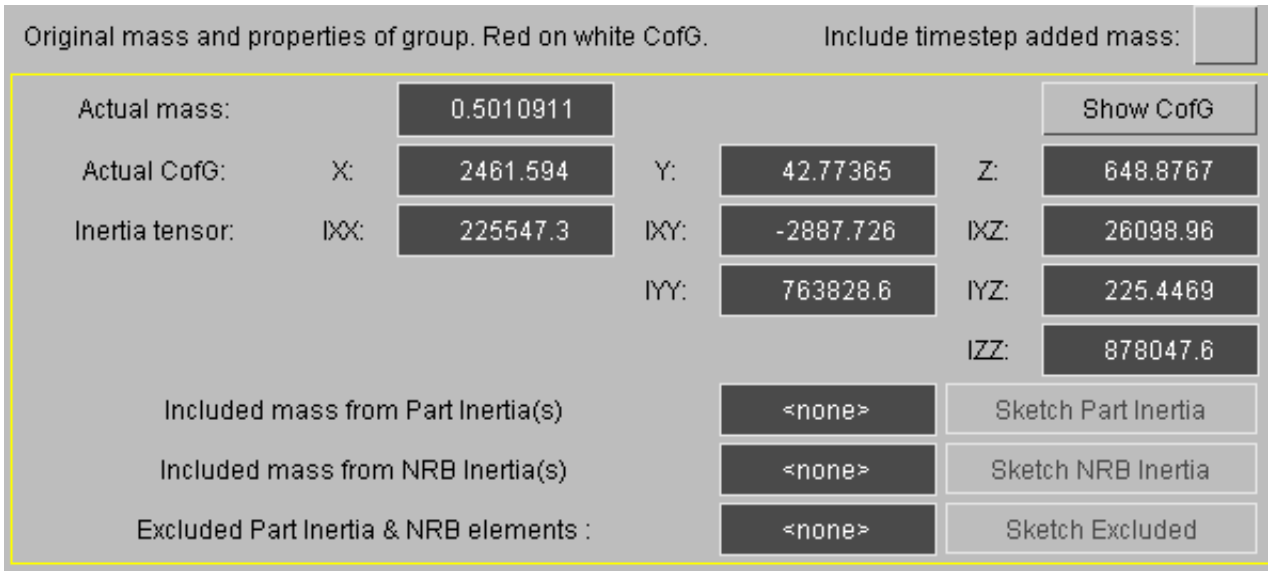
Group (a) may consist of a Primer group, a part-set or an assembly. Group (b) may consist of a Primer group or a

part-set.

The Primer group definition is versatile, but will require maintenance by the user, should the contents of the model change, therefore use of *ASSIGN_MASS_PART_SET (or *ASSIGN_MASS_ASSEMBLY) is often preferred as the contents are easier to maintain as the model updates (e.g. by using PART_SET_GENERATE).

*ASSIGN_MASS(_GROUP) is still available for backward compatibility.

Once the selection is made mass properties are displayed at the bottom of the panel the panel.



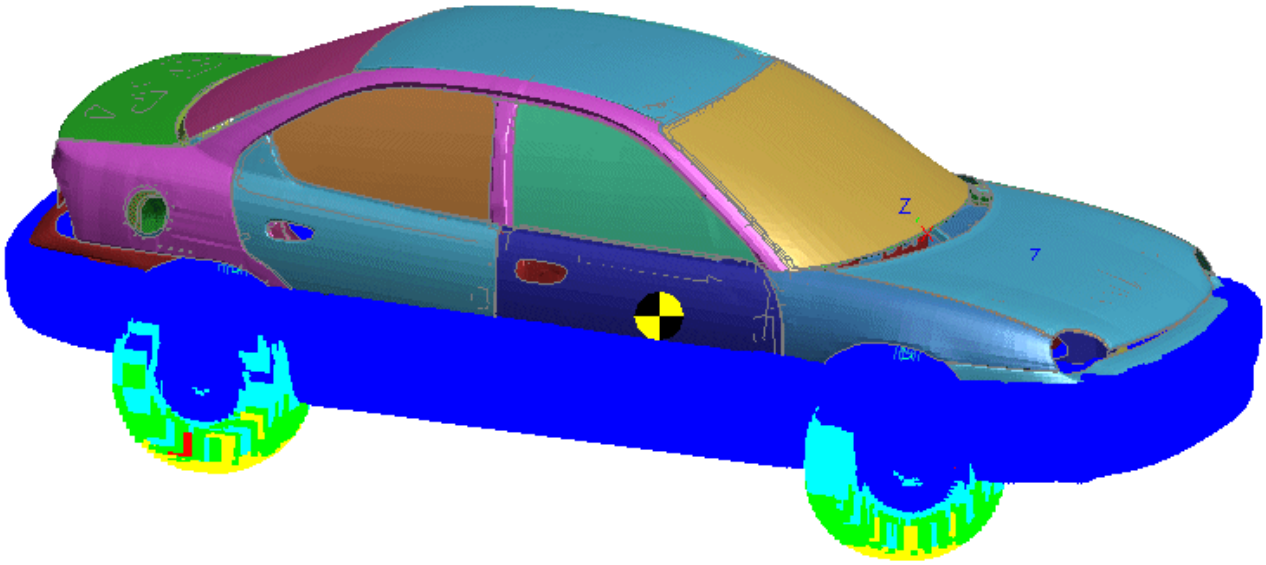
By default, **Incl attached mass** ia active, which means the mass of any attached lumped masses is implicitly included.

Timestep added mass is not included in the calculation by default, but may be by activation of the setting **Include timestep added mass**.

Reset All will set the target mass and CofG to the original properties of the selection. The drop-downs off **Set CofG** and **Set Inertia** may also be used to set the original values

By typing in to the text boxes, target values may be set for individual terms of centre of gravity and inertia or these may be left free (indicated by the wildcard symbol) to assume their own value.

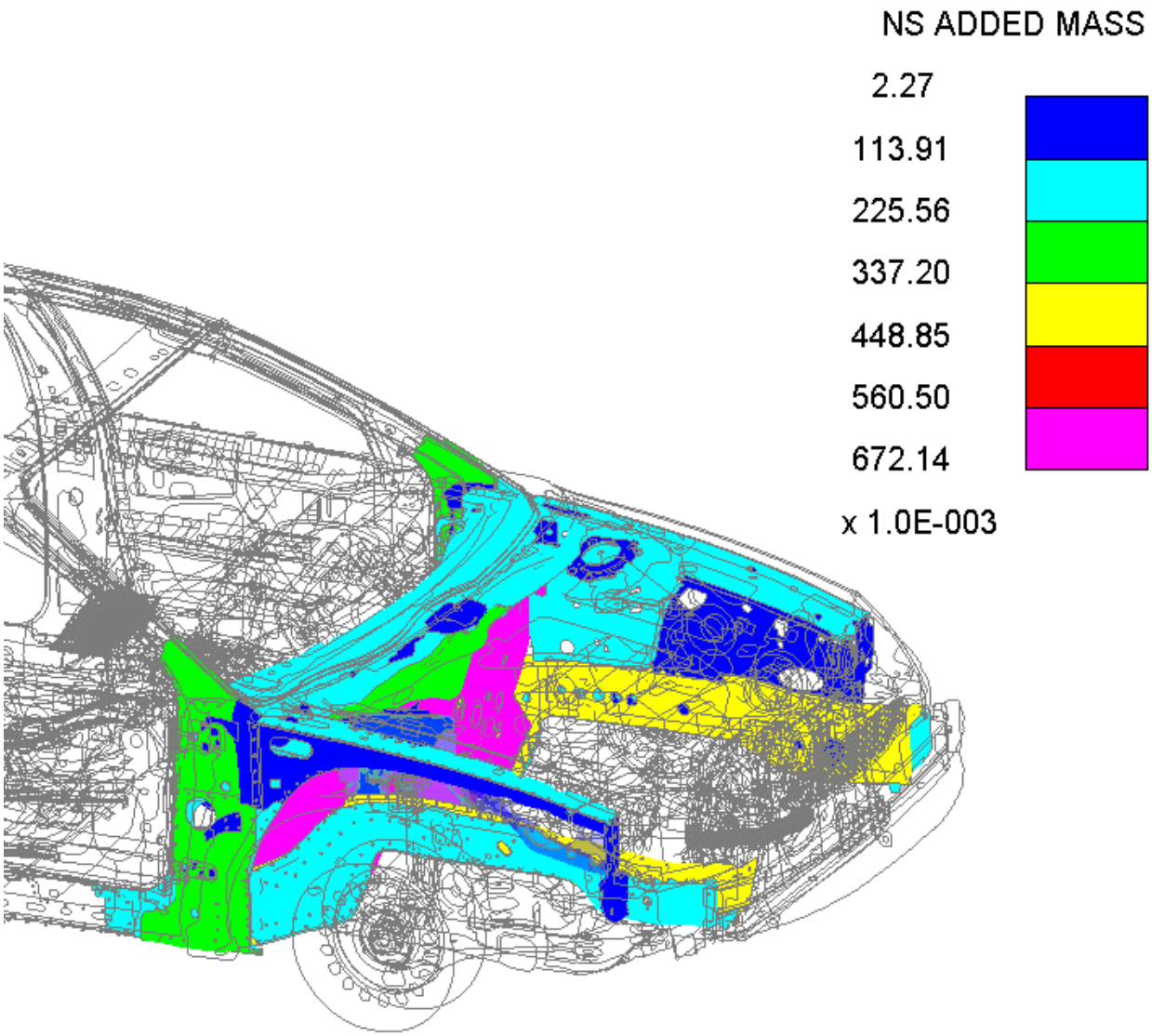
Target mass must be increased above the original mass. Press **CALCULATE** to determine the mass distribution which will best meet the assigned properties.



Plot mass to be added will show the proposed mass distribution. In this case, lowering the CofG has biased the mass toward the bottom

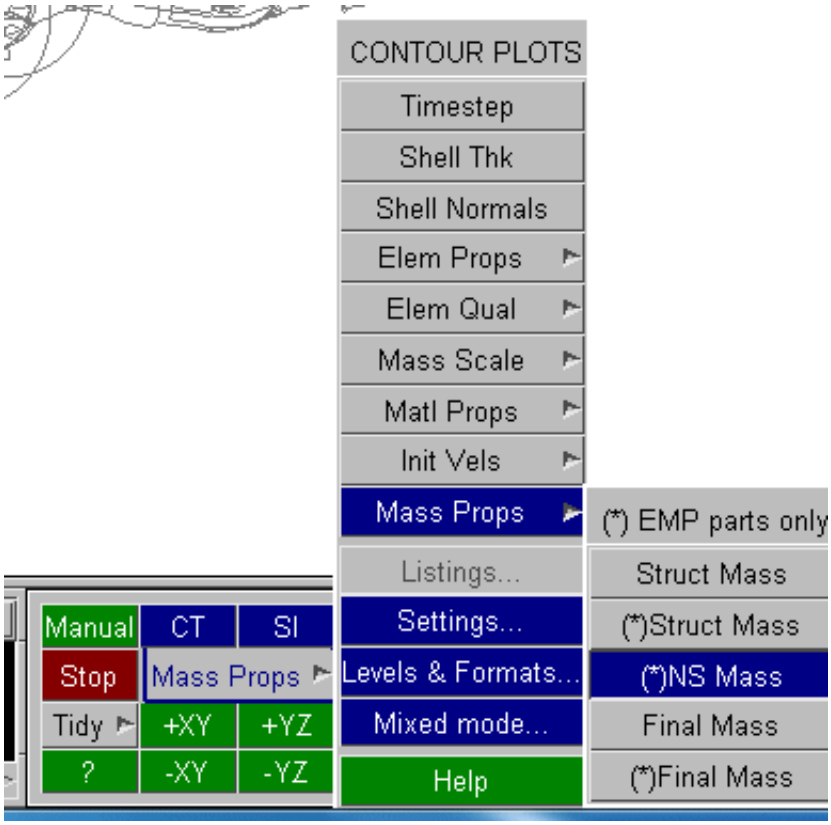
Create ASSM will then implement and save the solution.

Assign mass by Element Mass Part



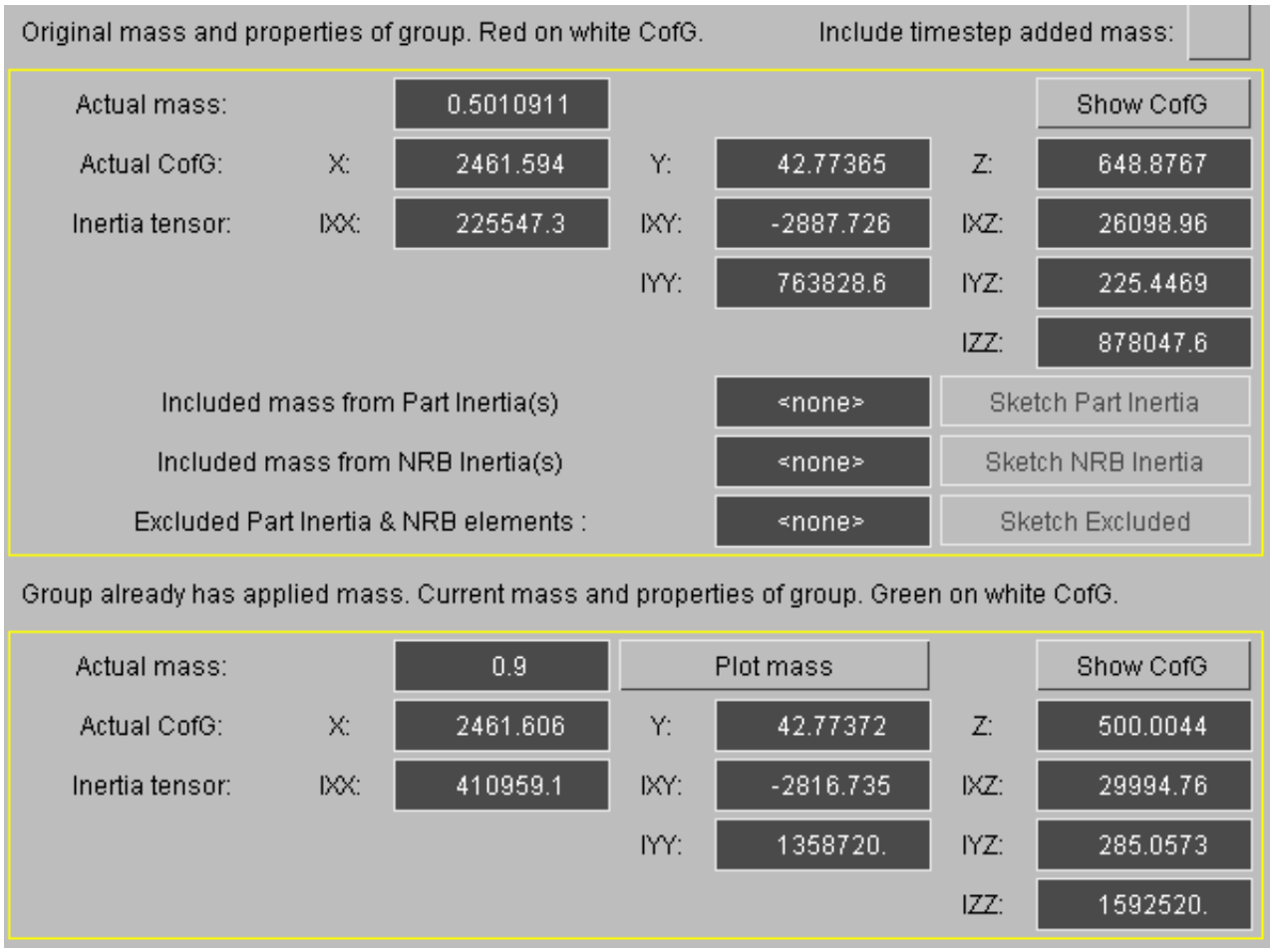
In part mode, **Plot mass to be added** will show the mass added per part by *ELEMENT_MASS_PART(ADD) definitions.

You may also use the mass properties contour tool to show (*)Struct Mass, (*)NS Mass or (*)Final Mass - that is the sum of structural and non-structural mass for all parts with *ELEMENT_MASS_PART.

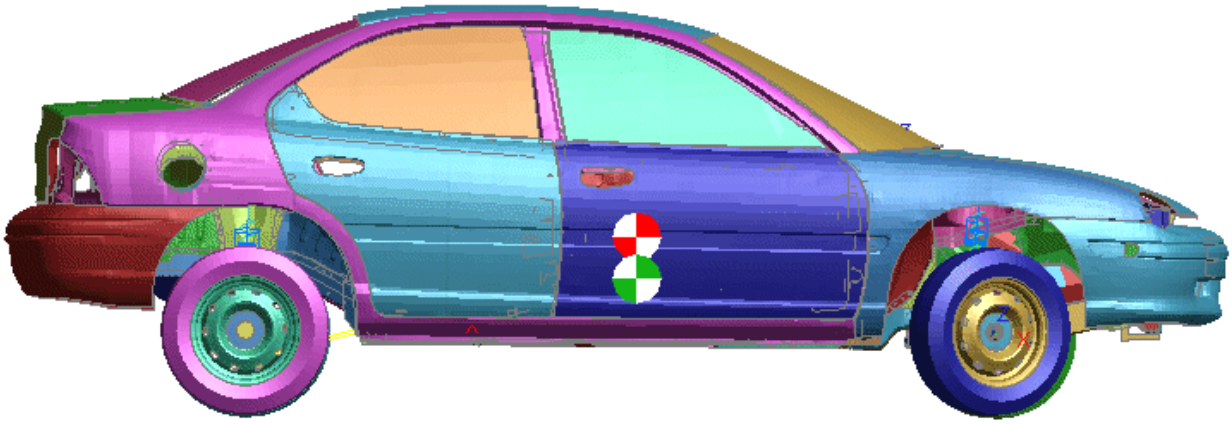


Modifying an assign mass

If we modify the previously created definition the panel will display both the original and the current mass properties.



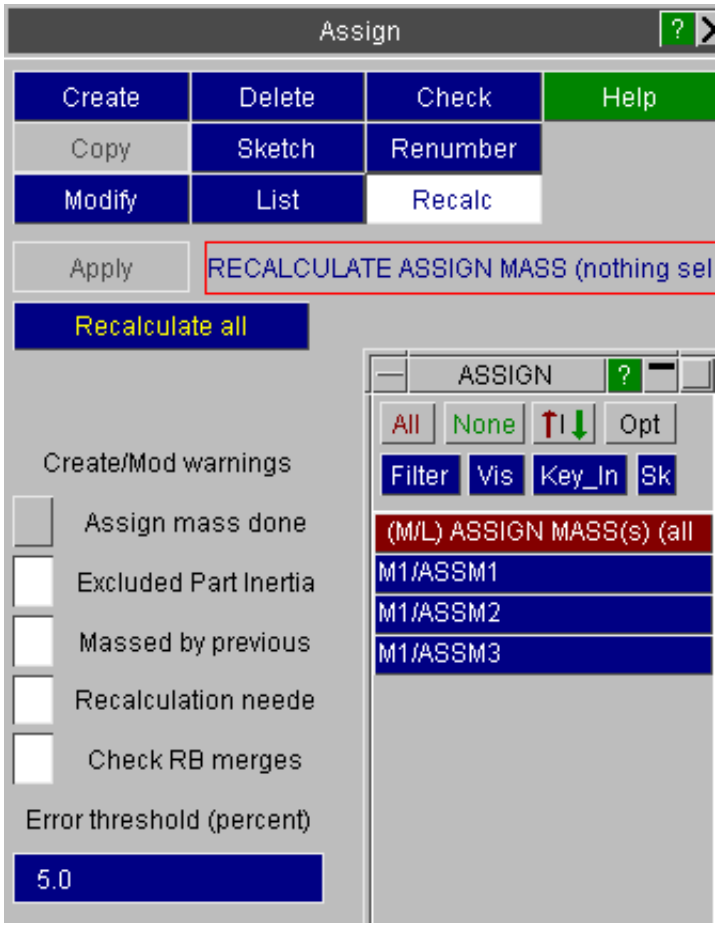
The modified CofG may be observed by using the **Show CofG** buttons.



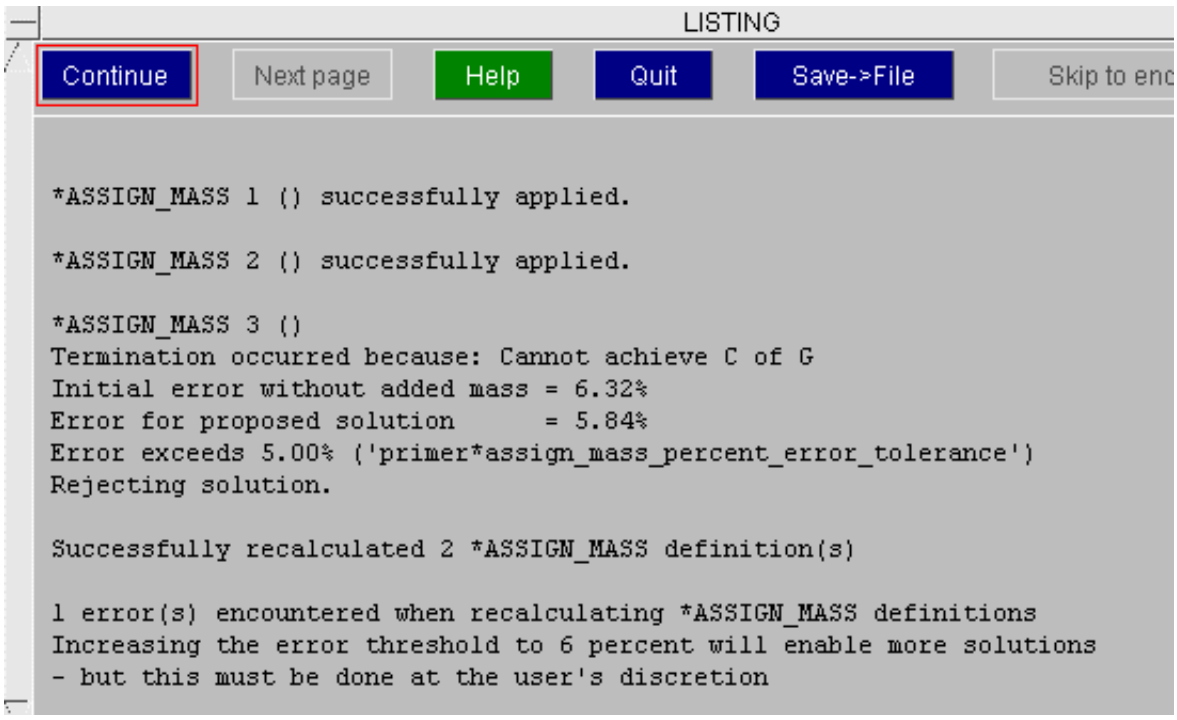
The definition may be modified and re-made with different target properties by using **Calculate** and **Update ASSM.**

Recalculating an assign mass

If you have a model which has been modified (remeshed, material properties changed, etc) with multiple assign mass, these can easily be remade by the **Recalculate All** function.



On completion, a listing panel will report any problem definitions.



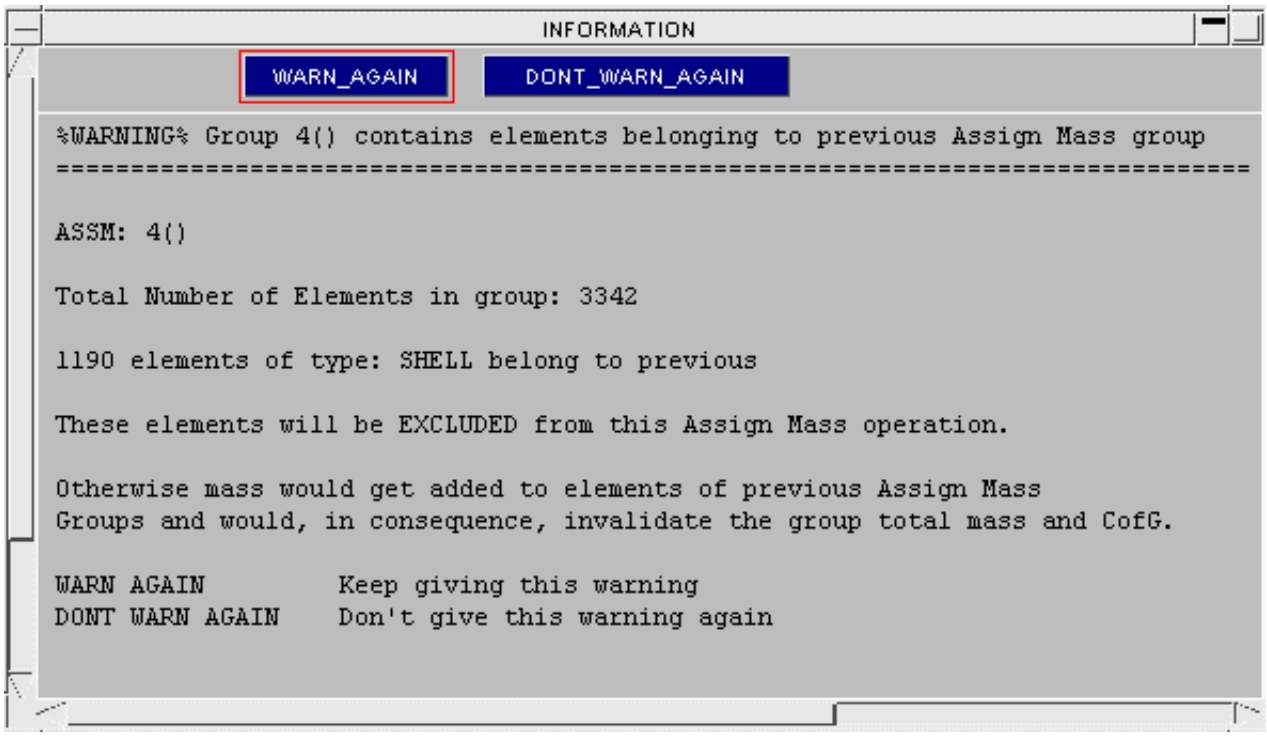
The same function can be accessed from the command line by the syntax ASSIGN APPLY_ALL.

Hierarchy of assign mass

If you are massing assemblies of components which have themselves been massed up, you must observe the hierarchy that the assign mass statements of the components precede (i.e. are at a lower label than) the assign mass statement of their corresponding assembly.

When you come to mass the assembly, you may either define a subgroup which contains all parts of the group except those which have already been massed, or, as this may be rather inconvenient, you can allow primer to **automatically exclude** those elements which have been massed previously.

When you **CALCULATE** the assign mass, you will get the following warning:



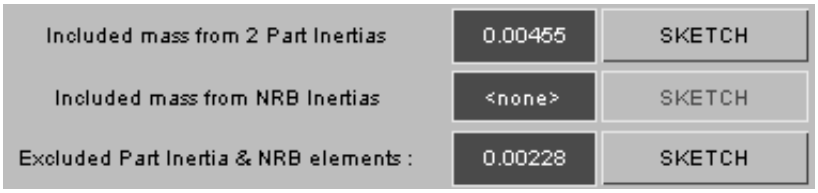
If we modify an Assign Mass group which contains elements which are used by a later (hierarchically higher) statement, a warning will be given and the user urged to apply the **RECALC** function. This will remake all the assign mass statement which have labels above the current one, thus accomodating the affect of modifying the mass of the lower group. In default mode, the elements will not be remassed. To maintain the integrity of the assign mass statements, it is recommended that the function be used in this way. However, some users have requested the ability to add mass to items already massed up. This may be done by setting the **OVERMASS** flag on both the overmassed and the overmassing assign mass statements.

Including part inertias in the assign mass operation

If a group contains *PART_INERTIA or *CONSTRAINED_NODAL_RIGID_BODY_INERTIA cards they will be included in the assign mass calculation if they are completely contained in the group. For example if you mass up an entire car that contains an engine which is a part inertia that will be fine. If you try to mass up the rear 2/3 of the car so only half of the engine is in the group, the engine part inertia will not be included.

If your group does not contain any inertia definitions then the panel will be displayed as shown on the right.

If your group does contain some inertia definitions then the panel will be displayed as shown on the right. The mass will be shown for each type and the elements can be sketched.



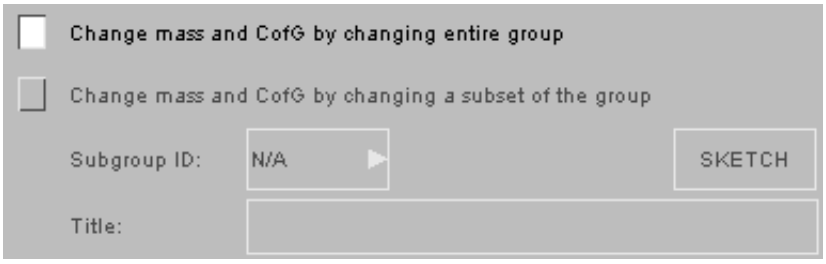
The included mass from parts and NRB's is shown. These are inertias that are completely contained in the group and so are included in the calculation.

The Excluded part inertia and NRB elements are from inertias that are not completely contained. If this occurs Primer will give a warning and they will not be included in the mass calculation (see the problems section)

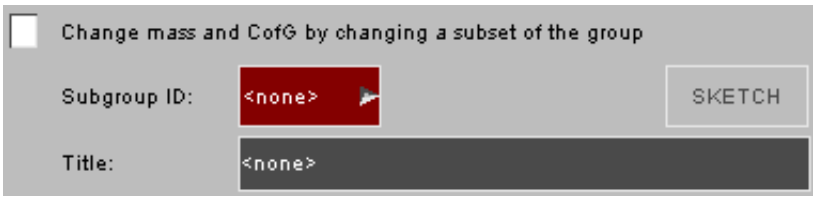
LS-DYNA will ignore any lumped masses that are on inertia definitions. They will be overwritten by the part inertia when DYNA initialises. Primer will not create any lumped masses on inertia definitions.

Changing the mass of a group by only adding mass to a subgroup

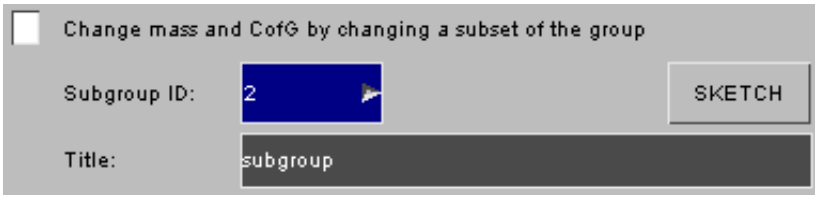
By default the assign mass function will try to change the mass and centre of gravity of the group by adding mass to all the nodes in the group (except the nodes on *PART_INERTIA and *CONSTRAINED_ NODAL_RIGID_BODY_INERTIA cards.



If you only want to change the mass on a certain part of the group instead then select **Change mass and CofG by changing a subset of the group**.



You can then select a subgroup which will be used by Primer for adding lumped masses to instead of the main group. This group **MUST** be a subgroup of the main group for this to work. If you try to use a group that is not a subgroup of the main group Primer will warn you.

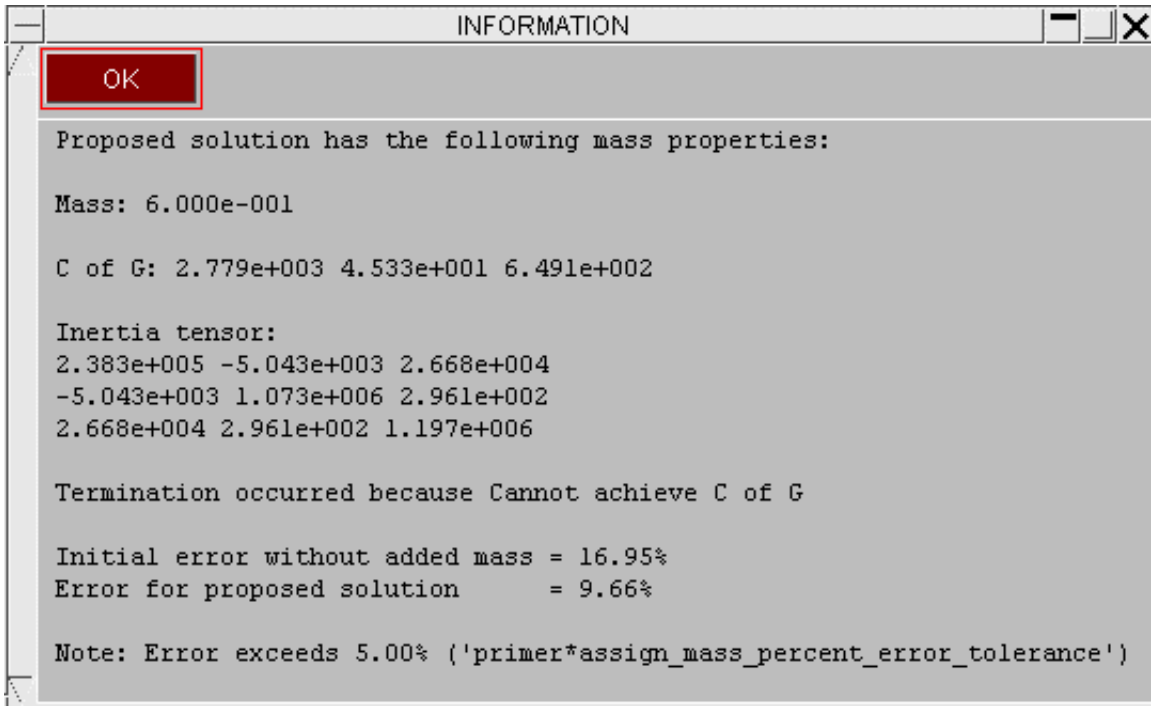


Problems with assign mass

Unable to achieve centre of gravity or inertia

The assign mass function in Primer works by adding lumped masses to (some but not necessary all of) the nodes of the group in an iterative process which attempts to meet the requirements of total mass, CofG and Inertia.

If the error of solution falls below the specified error tolerance (5% by default) the the mass distribution is simply applied as above. If a satisfactory solution cannot be reached an information panel will be offered giving the magnitude of the error.

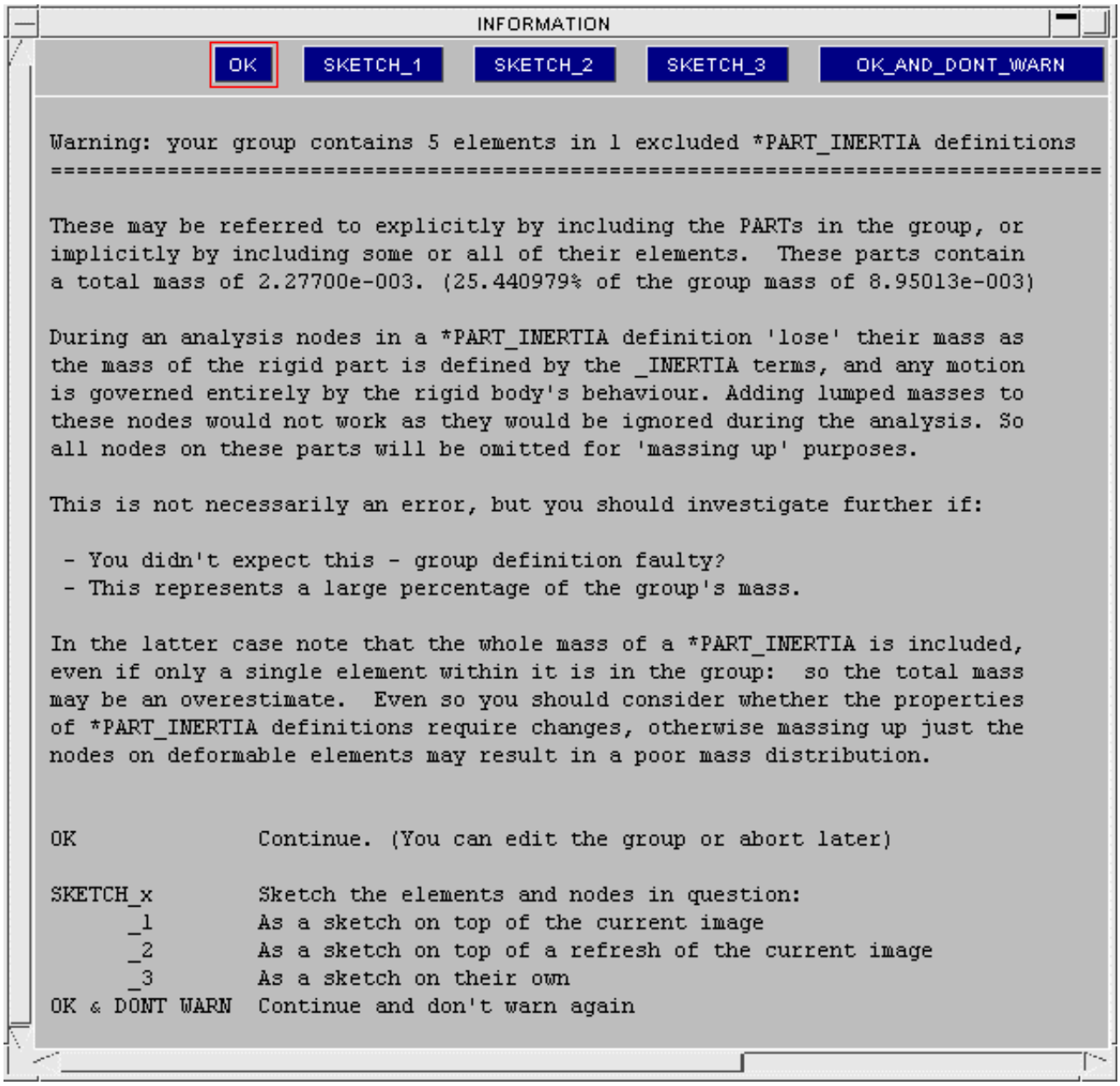


This will usually occur because for the given amount of mass being added the CofG cannot be shifted by the required amount. Primer can add mass to nodes but it cannot remove mass.

Part inertias

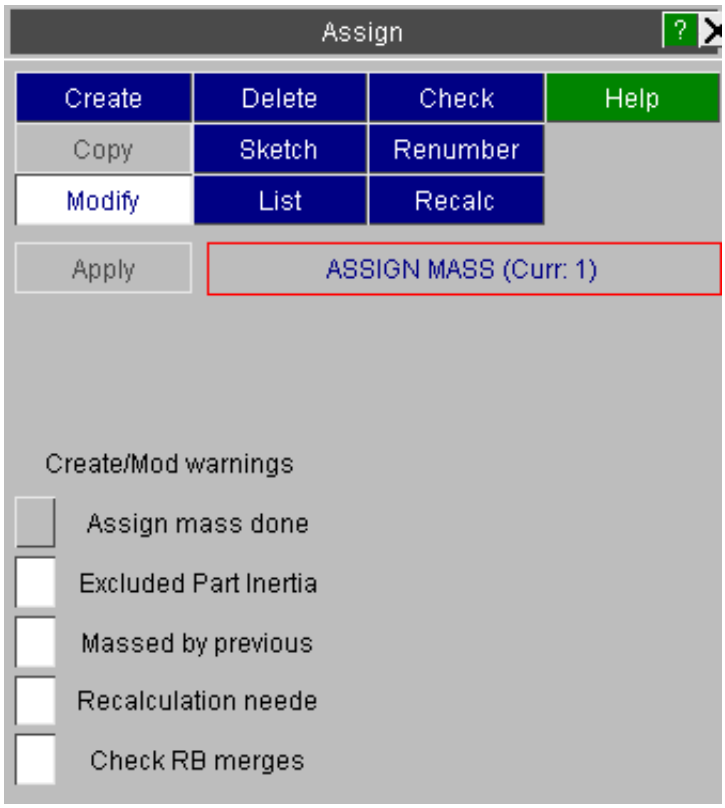
The assign mass function in PRIMER works by adding lumped masses to the group you have specified. This works perfectly OK (even on rigid materials). However, if you have a rigid material that has a **`*PART_INERTIA`** card, LS-DYNA ignores the mass of the elements and lumped masses and imposes the mass and inertia properties from the **`*PART_INERTIA`** card. This means that any lumped masses that are added in PRIMER during the assign mass function to nodes that are in a **`*PART_INERTIA`** will be ignored. An identical problem occurs if a node is part of a **`*NODAL_RIGID_BODY_INERTIA`**. Primer will include any inertia definitions that are completely contained in the group in the mass calculation but will not produce any lumped masses on the nodes in the inertia definition (see Including part inertias in the assign mass operation)

If an inertia definition is not completely contained it will not be included. To warn you of this, when you select the group in the assign mass panel, PRIMER checks to see if any of the nodes are on a **`*PART_INERTIA`** card or a **`*NODAL_RIGID_BODY_INERTIA`** card. If any of the nodes in the group are part of an **`_INERTIA`**, then Primer checks to see if the inertia is completely contained. If it is not, a warning screen is printed and the nodes will be excluded from the assign mass calculation.



Controlling suppression of text box warnings

The parent panel of assign mass has several warning options which the user can select or deselect as they wish:



The selection boxes allow the user to suppress warnings and errors that they feel are unnecessary.

assign mass done - reports the completion of the assign mass and the number of lumped masses added

excluded part inertia - gives the warning described above that part/nrbc _inertia cannot be included

massed by previous/recalculation needed - warn that there are hierarchy conflicts in the definitions which require resolving

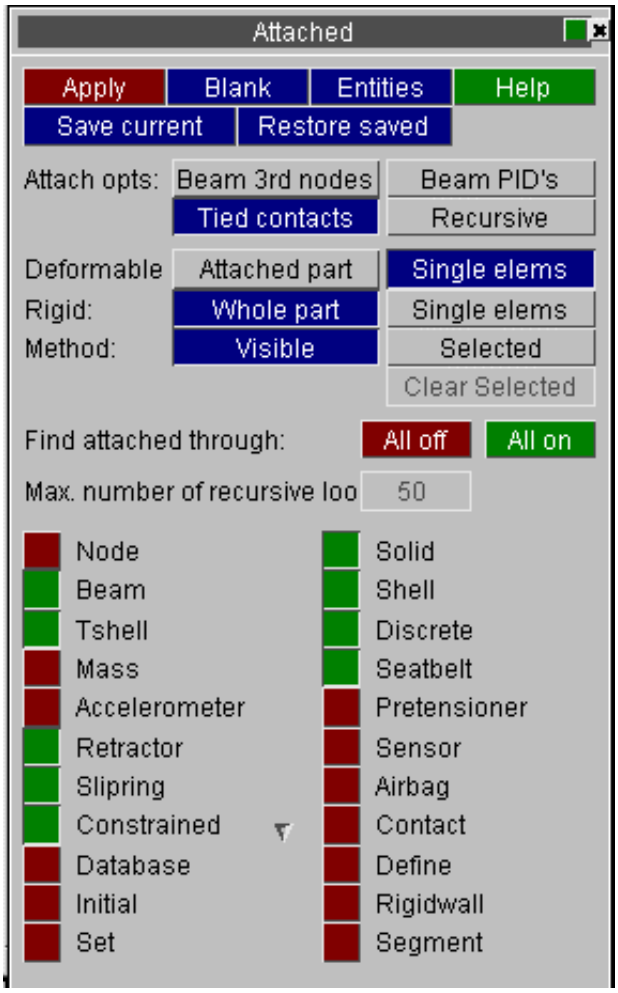
check RB merges - warns that a subset of slave/master rigid parts are included in the assign mass group. Whilst not an error this can cause confusion about the total mass (as Dyna assigns mass of slave parts to the master).

6.3 ATTACHED Displaying what is "attached to" things

- The ATTACHED menu
- What does "attached to" mean?
- Restricting its extent
- Attached options
- Controlling the "saved" status
- Interaction with Entity Viewing and BLANK
- Some Limitations

The **ATTACHED** menu is invoked from the **Tools** panel.

6.3.1 Top level menu



This figure shows the top level "attached" menu.

When you enter the **ATTACHED** menu the following happens:

- Everything that is currently drawable (ie unblanked with its entity switch turned on) is unblanked.
- Everything else is blanked.
- This blanking status is "remembered".
- Sets the attached switches to find anything physically attached.

At this stage performing a drawing operation (**LI**, etc) will not result in any change to what is currently visible.

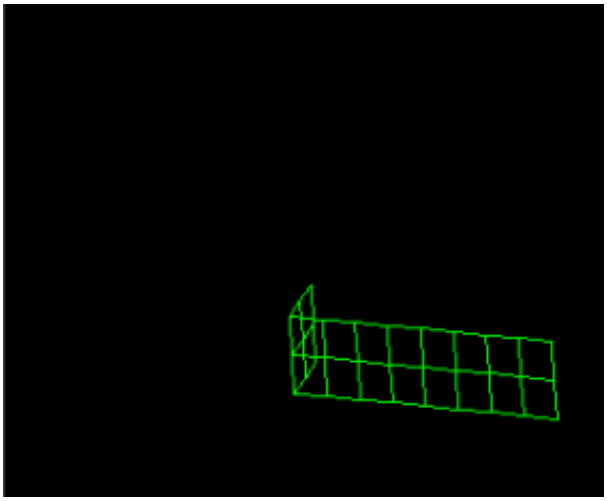
However each time you press **APPLY** Primer does the following:

- Looks at what you want to find attached (shells, constraints etc)
- Finds what is immediately "attached to" what is currently visible.
- Unblanks these newly found items.
- Redraws the image.

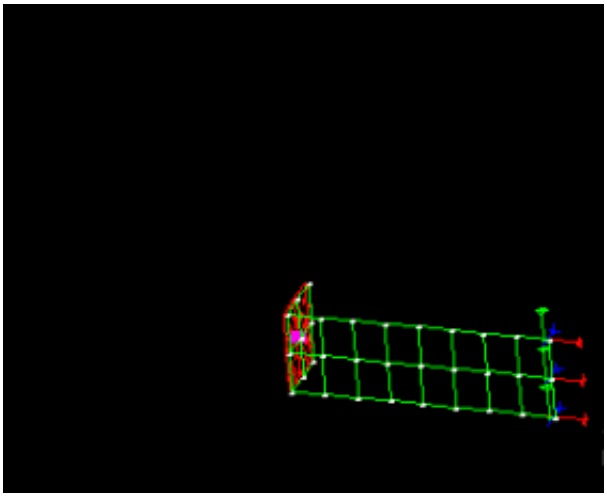
This results in progressively more and more of the model being drawn until nothing attached to what is currently visible (which is not necessarily the whole model) remains to be unblanked and drawn.

The following six images demonstrate how **ATTACHED** makes progressively more and more of a model visible:

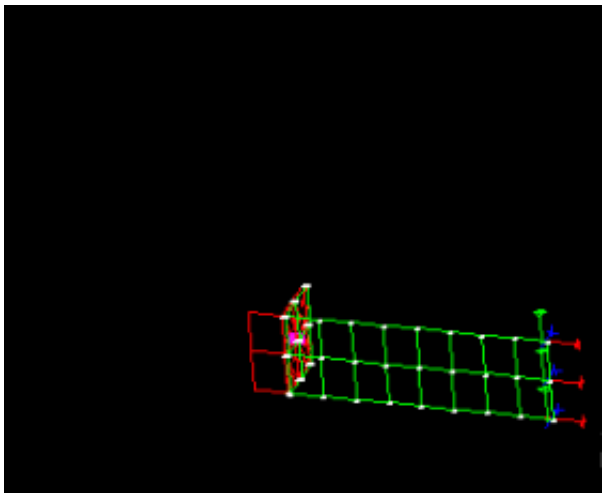
(1) Just one part visible



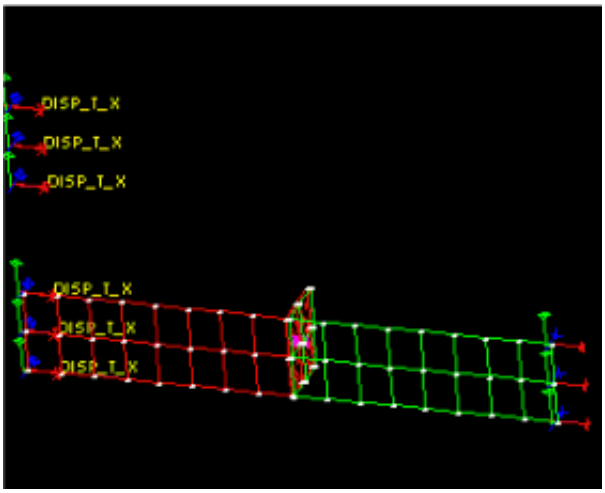
(2) Restraints, contact and spotweld to next part



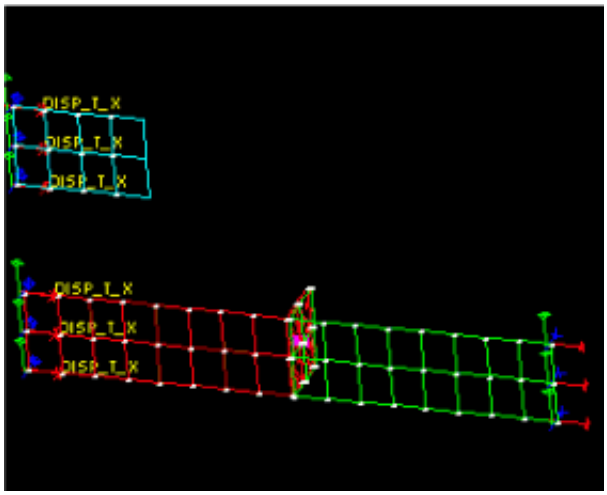
(3) Nearest elements on next (red) part attached to spotweld.



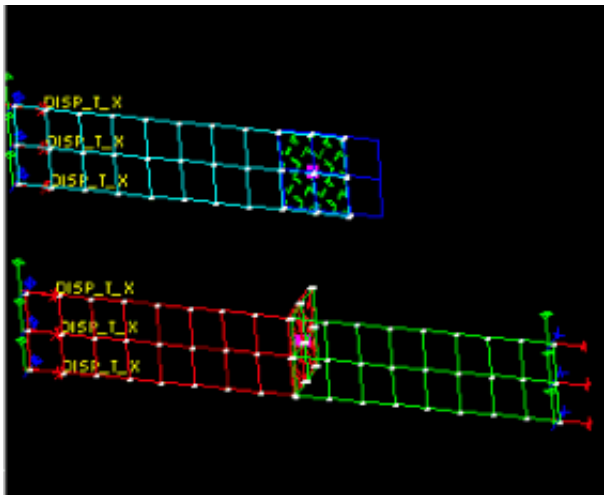
(4) (After a few **APPLY** operations) restraint set on red part



(5) (After more **APPLY**s) elements on the light blue part



(6) More **APPLY**s: spotweld to & elements of dark blue part.



6.3.2 What does "attached to" actually mean?

In this situation attached generally means "***anything connected to, or referenced by, what is currently visible***".

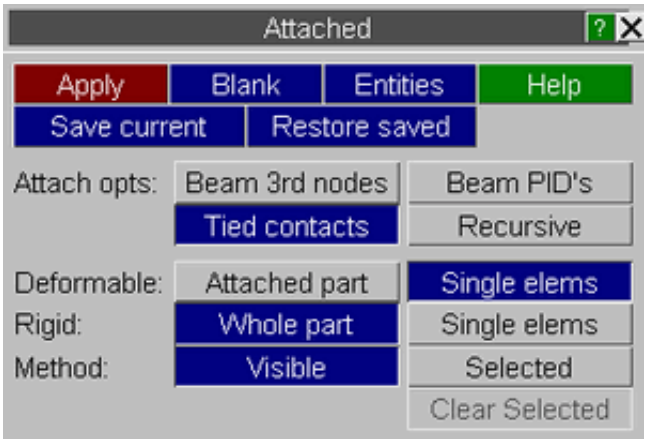
This is a wider definition than the strictly structural "*any elements attached to visible nodes*" which, initially, might seem to be the more obvious method. The figures above show why this is so:

- Figure (4) shows the **`*BOUNDARY_PRESCRIBED_MOTION_SET`** (Labelled as `DISP_T_X`) applied to all the edge nodes on the left of the picture.
- Because the set used by this includes nodes on the end of the light blue part (visible in figure (5)) then elements attached to this part have become visible.
- Consequently the light and dark blue parts eventually become visible, even though they have no physical connection to the red and green ones.

6.3.3 Attached options

There are several options available to the user to increase the flexibility of the attached panel.

- Beam 3rd nodes
- Beam PID's
- Tied Contacts
- Recursive



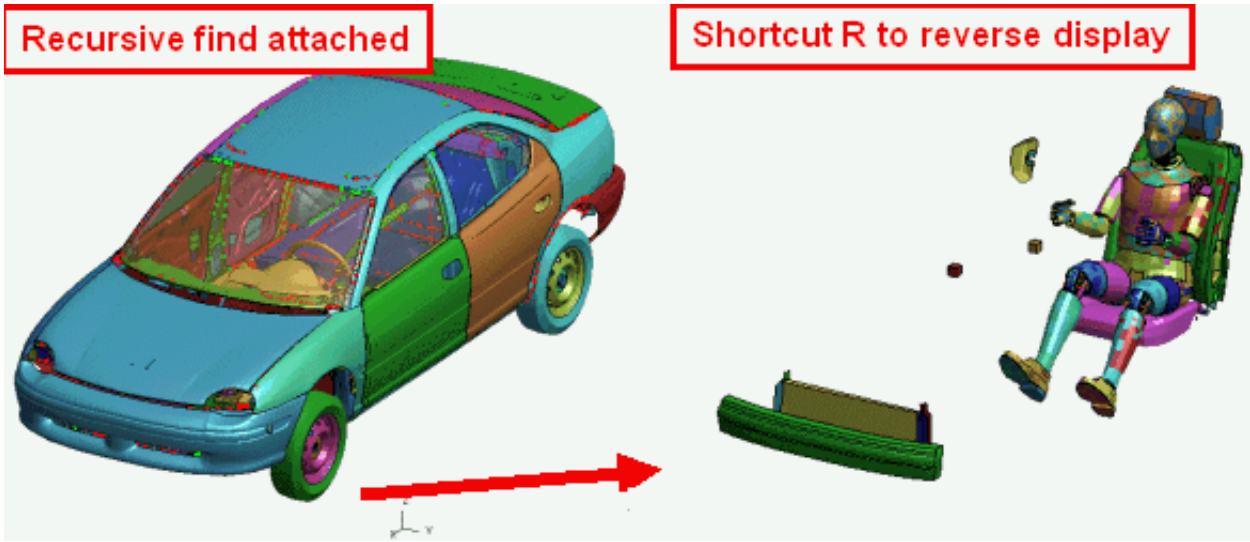
Beam 3rd nodes will find attached entities through a beam's 3rd node (and vice versa).

Beam PID's will find attached beams that refer to a part displayed through their PID1 and PID2 fields.

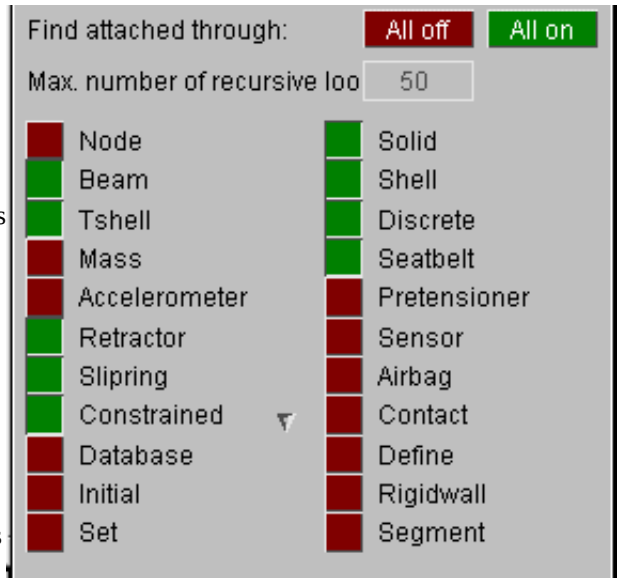
Tied contacts will find attached elements through DYNA tied contacts using PRIMER's contact penetration checker.

Recursive will iteratively keep finding attached until no more can be found - note there is a failsafe value in PRIMER to allow for any anomalies that might cause this routine to go on forever (the STOP button also works here). Pressing reverse all (shortcut R) will then show any unattached parts.

Instead of finding attached to all the visible entities, the user can select the entity/entities they wish to find entities attached to. This is done by selecting **Selected** for the method instead of **Visible**. In selected mode, an object menu is used to select the "seed" items. Use **Clear Selected** to reset your selection.



6.3.4 Restricting the extent of "attached to" propagation

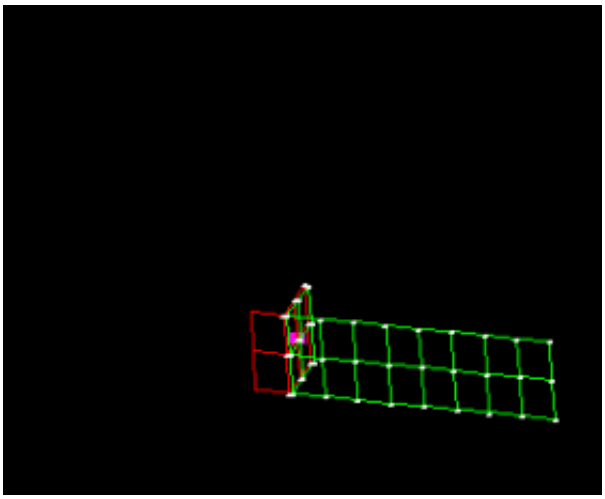


Because the definition in section 6.3.2 is all-embracing it may lead to too many things being made visible. Therefore it is possible to limit what is found attached through entity switches .

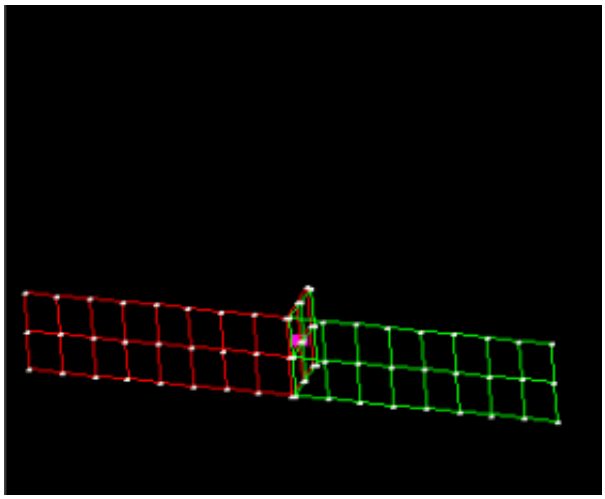
For example, you can still display shells, solids, beams ect but just find attached beams. Note that there is a triangle to the right of the **Constrained** entity switch. Right clicking here allows you to select to **Filter** the constrained entity types. A new panel will open up allowing you to turn on or off entity switches for the different constrained types. This allows you, for example, to find attached through nodal rigid bodies, but ignore constrained spotwelds.

In the example below the user has selected nodes and elements only, which results in the narrower "structural" definition of attachment referred to above. Starting from the same point as figure (1) above a series of **APPLY** operations gives rise to figures (7) and (8) below:

(7) The spotweld connects to the 2nd red part as before



(8) The final result: only nodes and elements are drawn



Now the blue parts are not diagnosed as being attached to the red and green ones, since the connection between them (the node set used by an initial velocity definition) has not been drawn.

6.3.5 Using and updating the "SAVED" status

When you enter **ATTACHED** the current visibility status is saved in a backup blanking table. All **APPLY** operations operate only on the current blanking table, leaving this backup unchanged.

The reason for this is simple: most usage of **ATTACHED** reveals too much information in the first pass, and it is necessary to go back and repeat the process with some **attached** categories switched off.

RESTORE SAVED	Copies the backup blanking tables to the "current" ones, effectively restoring the initial state. You can **RESTORE_SAVED** as many times as you like within a given usage of **ATTACHED**. Because the backup tables are always rebuilt from what is currently visible whenever you (re)enter **ATTACHED** they are effectively "lost" whenever you close this panel. To maintain a backup while performing other operations you can iconise this panel rather than closing it.
SAVE CURRENT	Copies the current blanking tables, ie what is currently visible, into the backup ones (overwriting them). This then becomes the new "saved" state. You can **SAVE CURRENT** at any time, but doing so loses your original saved state irretrievably.

6.3.6 How ATTACHED inter-reacts with Entity Viewing and Blanking

ATTACHED has to modify entity drawing and labelling settings (the province of **ENT**ity Viewing) and also blanking tables (**Blank**).

The entity switches you find attached through will inevitable effect the entity panel. If shells are searched for, and the entity panel doesn't have shells displayed, then the attached panel will turn the entitiy switch on too.

For convenience you can access the relevant panels directly from the **ATTACHED** panel, as shown here (they are no different when accessed here as opposed to from the main panel).

6.3.7 Some limitations of ATTACHED - mainly due to using `*SET_xxx`

Because the definition of "attached" is wider than the purely structural one of connected nodes and elements (see section 6.3.2 above), some problems can arise when definitions which use sets are diagnosed as being "attached".

Consider the following case:

- Initial velocities for the whole model are defined by `*INITIAL_VELOCITY` using a `*SET_NODE` that contains all nodes in the model.
- Once a single node in that set is detected as being "visible" then the set itself is also made visible. This has the consequence of drawing all nodes (with their initial velocities) in the model ....
- .... and the next thing you see, after a single **APPLY** operation, is the whole model being drawn.

This presents a dilemma: should **ATTACHED** track through `*SET` definitions or not?

If it does then the problem defined above occurs.

If it does not then, for example, extra nodes on a rigid part (defined by `*CONSTRAINED_EXTRA_NODES_SET`) will not be drawn properly when the part is visible.

At present **ATTACHED** ***does*** track through `*SET` definitions, leading to the problem outlined above. (Although the case of set zero, where it means "all nodes in the model", is detected and trapped.)

You can stop specific cases happening by turning off their switch (for example to stop the initial velocity display turn off `INITIAL` switch). You can also **BLANK** specific items (although they will probably unblank themselves again after the next **APPLY** operation).

6.4 BILL OF MATERIALS

The Bill of Materials section in PRIMER enables you to check and if necessary update part names, the material title used for a part and the gauge of a part. This is done by reading a file containing the necessary information to check for each part.

6.4.1 File format

The bill of materials reader is designed to be able to read files produced by hand, spreadsheet programs and other programs. The files produced from spreadsheets are commonly known as CSV or comma separated files.

The format of the file has to follow the following rules:

- Blank lines in the file are skipped.
- A comment line can be included anywhere in the file by starting the line with a specific character (which you can define when reading the file into PRIMER).
- Lines that contain specific strings or characters can be skipped
- The Bill of materials file should contain one line for each part entry you want to check/modify.
- Each line should contain 'fields' that are separated with a specific character (which you can define when reading the file into PRIMER).
- The fields must be in the same order for every line.
- One of the fields MUST be the part ID.
- For auto-recognition of columns to work, columns must be given specific names.

Example

An example Bill of materials file is shown below.

```
$ Bill of Materials file:Example Bill of Materials
$ Date produced: April 2001
$ Produced by: Miles Thornton
Vehicle X,Bill of Materials version,8.6,Date,20/02/01

Part No,Title,Part ID,Material,Supplier,Gauge,Part mass
AA51201,sill_swan_neck,5,HP37  ,Company X,2.2,9.64E-03
AA51202,front_support_mem diagonal,101,HP37 ,Company X,2.2,4.74E-03
AA51203,Bumper_ft,104,HP37,Company X,1,3.71E-03
AA51204,A_pillar_lower_support_a,113,CR4  treatment C,Company Y,1.2,2.28E-03
AA51205,cowl.1,200,CR4 treatment C,Company  Y,1.2,6.40E-03
AA51206,A_pillar_lower_support_b,202,CR4 treatment C,Company Y,2,6.60E-03
AA51207,dash_x_member,203,CR4 plt3 grade,Company Y,1.2,2.25E-03
AA51208,dash_panel,204,CR4  plt3 grade,Company Y,1.2,4.48E-03
AA51209,front_floor_panel,304,CR4 plt3 grade,Company  Y,1.2,7.17E-03
AA51210,Tunnel_wall_FR,305,CR4 plt3 grade,Company Y,1.2,5.76E-03
```

We want to skip the first 3 lines and so have started them with a $.
The fourth line does not begin with a '$' but we can force the reader to skip the line if needed.
The fifth line is blank and so will skipped.
The next line describes the fields. This is not necessary but helps to see which fields are which. If you want to do automatic recognition of columns then a line that contains the column names is essential. This can also be skipped.
The following lines contain the information. Each line contains the information for one part. The fields are separated by a ',' (comma). The following sections show this file being read and used by PRIMER.

Automatic recognition of columns in Bill of materials file

If **Auto-detect columns** is switched on then when PRIMER reads a bill of materials file it looks for specific column headers on any of the lines which are not blank or comments. The first 50 lines in a bill of materials file are shown as a preview when reading the file. Primer will look at each of these lines in turn and look to see how many of the column names it can match to the standard column names. The line that matches the most headers is the one that will be used for the column titles. If you do not want PRIMER to recognise the columns this can be turned off

Rules for matching column titles

1. Each 'field' is read from the line in turn.
2. Any spaces are removed from the field.
3. The 'field' text is converted to lower case.
4. The text is compared to the headers below. If an exact match is found then that 'field' type is set to the type that matched

Standard Headers

Field type	Allowed headers	Meaning	Field contains
Part ID	pid, part, number, partid	Model Part ID (compulsory)	integer
Part title	title, parttitle, description	Part title	characters
Material title	material, mat, matname, materialname	Material name	characters
Material ID	materialid, matid, matnumber, materialnumber, mid	Material ID (for referencing standard database of materials)	integer
Cad part number/description	cad, cadpart, cadpartno, partno	Cad part number (not currently used by PRIMER)	characters
Gauge	gauge, thickness, thk	Thickness of part (on *SECTION_SHELL card)	real
Hourglass type	hourglasstype, hgtype	Hourglass type (on *HOURGLASS card)	integer
Hourglass coefficient	hourglasscoeff, hgcoeff	Hourglass coefficient (on *HOURGLASS card)	real
Lower ID	lowerlabel, lowerid,lower,low	start id for renum of nodes/elements/masses on part	integer
Upper ID	higherlabel,higherid,higher,high	end id for renum of node/elements/masses on part	integer
Element formulation	elform, formulation, element	ELFORM on *SECTION card	integer
Section ID	sectionid,secid,secnumber,sectionnumber,sid	Section ID	integer
Hourglass ID	hourglassid,hgid,hgnumber,hourglassnumber	Hourglass ID	integer

Examples

As spaces are removed and the fields are case insensitive all the following could be used for the Part ID field:

- Part ID (would match 'partid')
- P I D (would match 'pid')
- Part (would match 'part')
- PARTID (would match 'partid')
- Number (would match 'number')

The following could not be used:

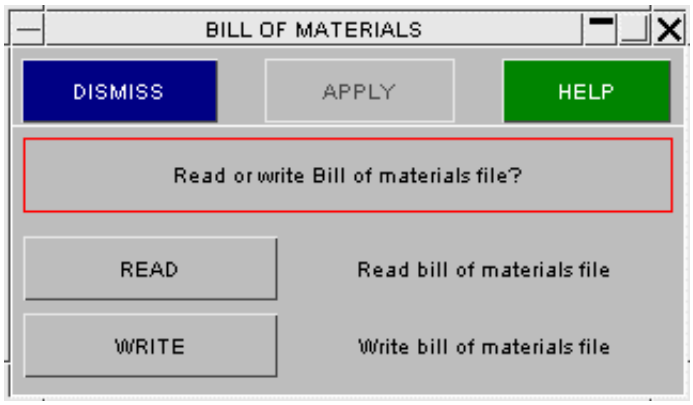
- Part number (As the string would not exactly match 'number')
- ID (as ID is not a valid title)

In the example file shown above all of the columns except for `Supplier` and `Part mass` would successfully matched.

6.4.2 Initial Screen

The initial screen allows you to choose whether to read or write a Bill of materials file.

The two options available are **READ** and **WRITE**.



6.4.3 Reading a Bill of Materials file

Selecting the file

The initial Bill of Materials screen is shown in the figure below.To select the Bill of materials file either type the name into the blue **File:** box or press the **?** button to bring up the file selection panel.

Until a file name is given the **APPLY** button will not be active. When the button is active, pressing it will scan the file and any comment lines can be selected.

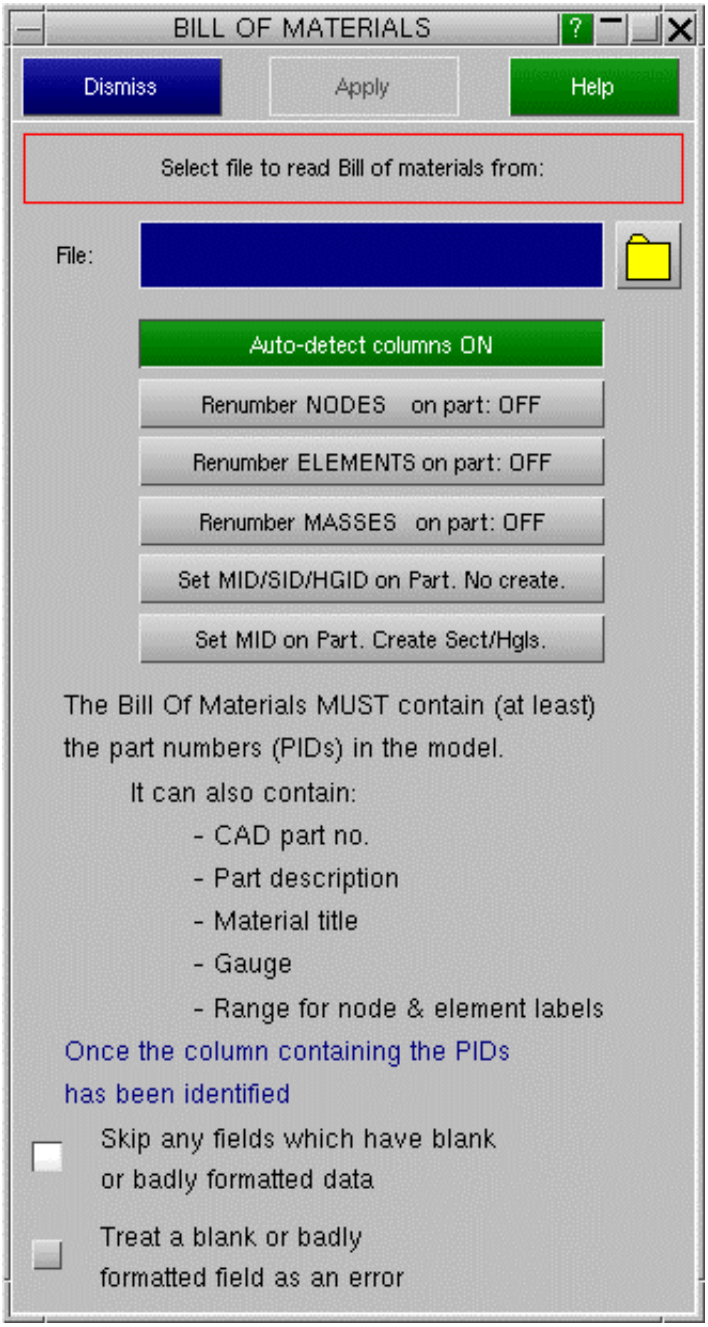
In the image on the right **Auto-detect columns** is turned **ON**. PRIMER will try to recognise each field by using the column titles.

If **Renumber NODES/ELEMENTS/MASSES on Part** is active and label ranges are defined for this part, the renumbering function will be activated. On large models this may slow the read of the BOM considerably. Label ranges may overlap one another.

Two methods are available for handling materials, section or hourglass cards when these are shared by more than one part. The default option **set MID/SID/HGID** will set the part to reference the specified material, etc. It will the only adjust the material title, section properties, etc. if no other part references it. The option **Modify/Create Matl/Sect/Hgls** will always make a material/section/hgls card with the specified data, creating a new one and ignoring the specified ID, if this proves necessary, because another part refers to that card.

If the material, section and hourglass card for each part are kept unique , both methods will give the same result.

The option **Set MID on Part. Create Sect/Hgls** if set will over-ride the other. The material on the part card will be set to match the (first found) material of the given name or the given material id if no name is specified. A section card will be created at the same id as the part and the gauge,etc will be updated with the given data. Similarly hourgalss cards will be created if the data is non-zero or one previously existed.



The radio buttons enable/disable error trapping when reading the part number field. In the example bill of materials file in the previous section the line

```
Part No,Title,Part ID,Material,Supplier,Gauge,Part mass
```

is not a comment line. If this line is read as an actual line of data an error could occur as instead of reading a number for the part ID, the string `'Part ID'` would be read instead.
If `'Skip any lines which have a blank or badly formatted PID'` is selected, the line would be skipped, a warning printed and the read will continue.
If 'Treat a blank or badly formatted PID as an error' is selected, this would be treated as an error and the read will stop.

Comment lines

Once the bill of materials file is selected it is scanned and a preview of the file is shown (the first 50 lines of the file are shown).

This preview can be used to help answer the questions which PRIMER asks. The scrollbars can be used to scroll the preview up and down and from left to right.

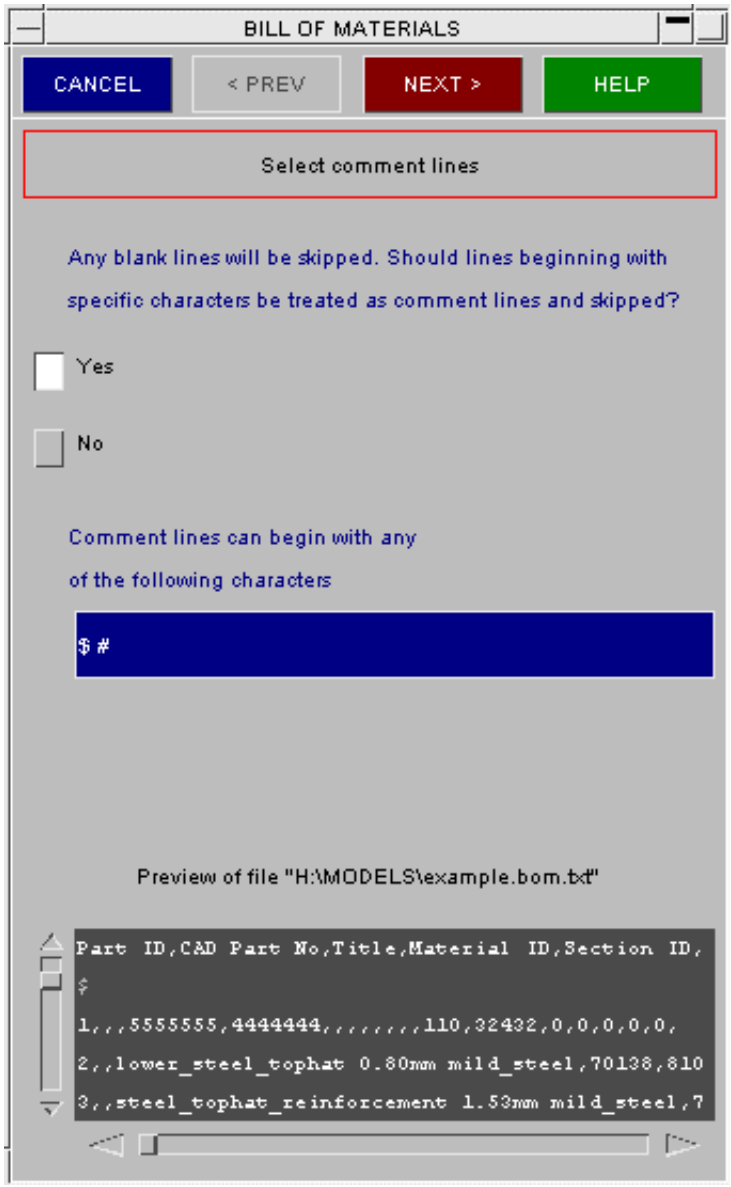
The default is not to skip any lines. To skip the comments the switch must be set to **Yes.** If the file does not contain any comments this step can be skipped.

To cancel reading and return to file selection press **CANCEL**.

To go on to the next step (skipping specific lines) press **NEXT >**

By default comment lines can begin with a $ or a #. Type the characters that you want comments to begin with into the blue box.

In the file preview any lines that will be treated as comments are shown in grey text instead of white text.



Skipping specific lines

In this example we want to skip the line that begins

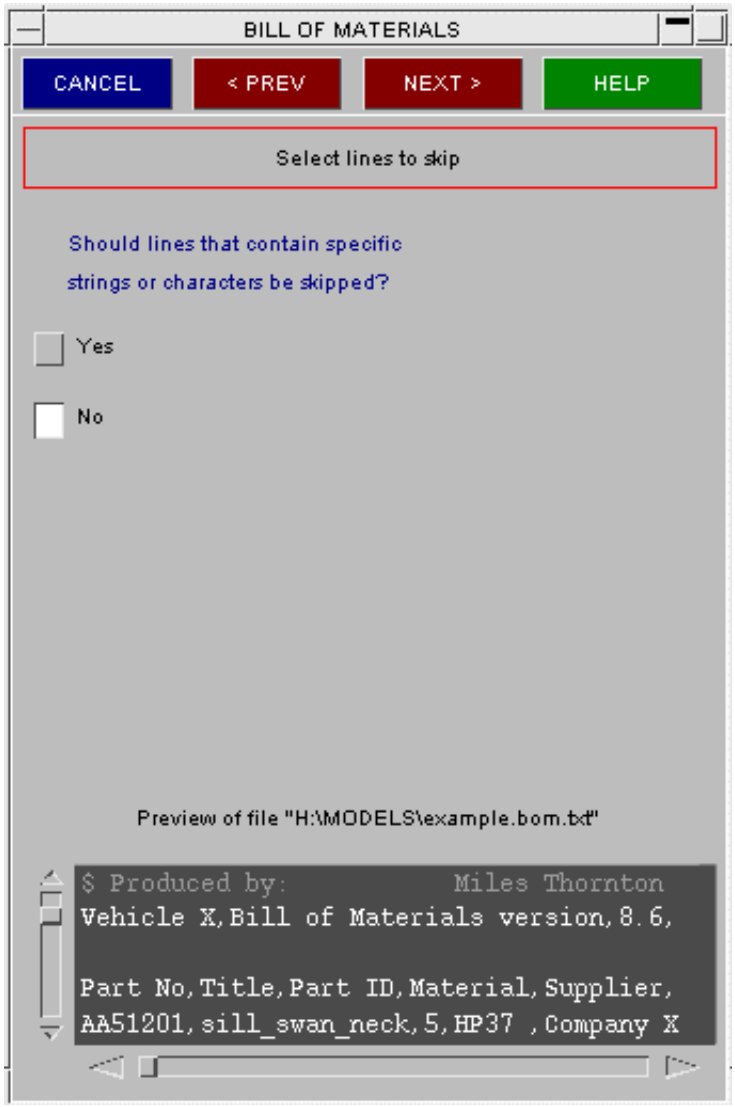
```
Vehicle X, Bill of Materials
```

The default is not to skip any lines containing specific strings. To skip the line the switch must be set to **Yes.** If no lines need to be skipped this is not needed and can just be left at the default value.

To cancel reading and return to file selection press **CANCEL**.

To go back to the previous step (comment lines) press **PREV>**

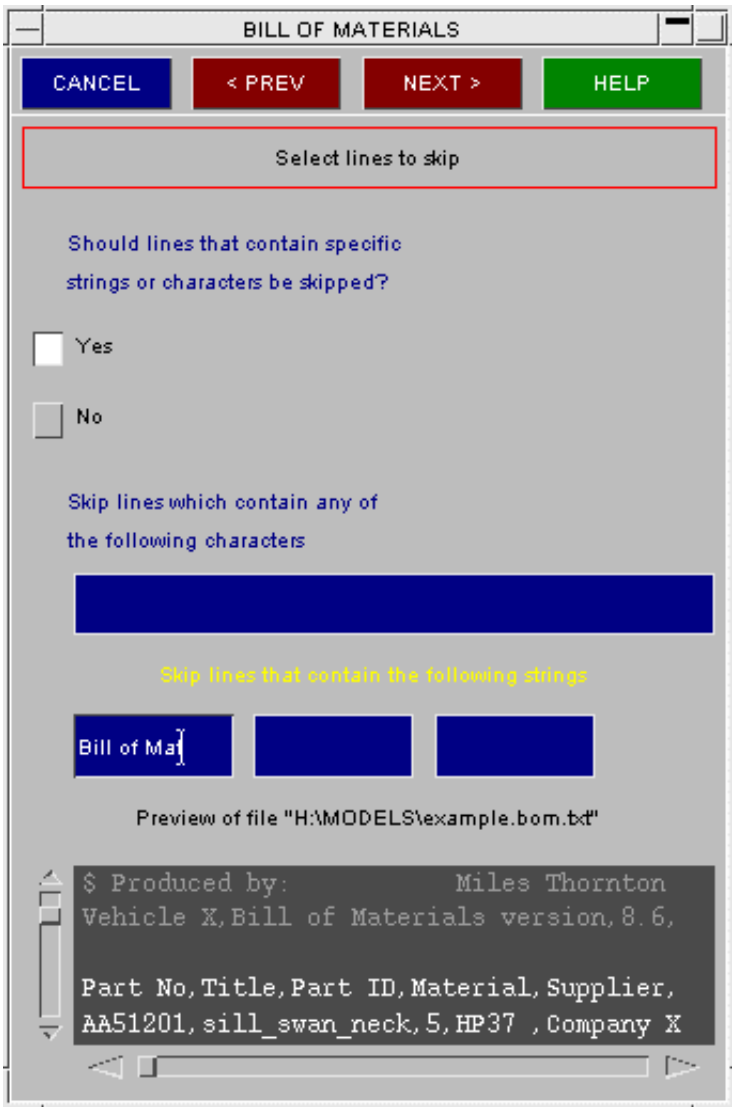
To go on to the next step (selecting delimiters) press **NEXT >**



A line can be skipped that either contains a specific character or a specific string. Type the characters or strings into the blue boxes. Text is case sensitive.

In this example we have chosen to skip any lines that contain the string '`Bill of Mat`'.

In the file preview any lines that will be skipped because they contain specific strings or characters are shown in grey text instead of white text.



Selecting delimiters

In this example the fields are separated by commas. e.g.

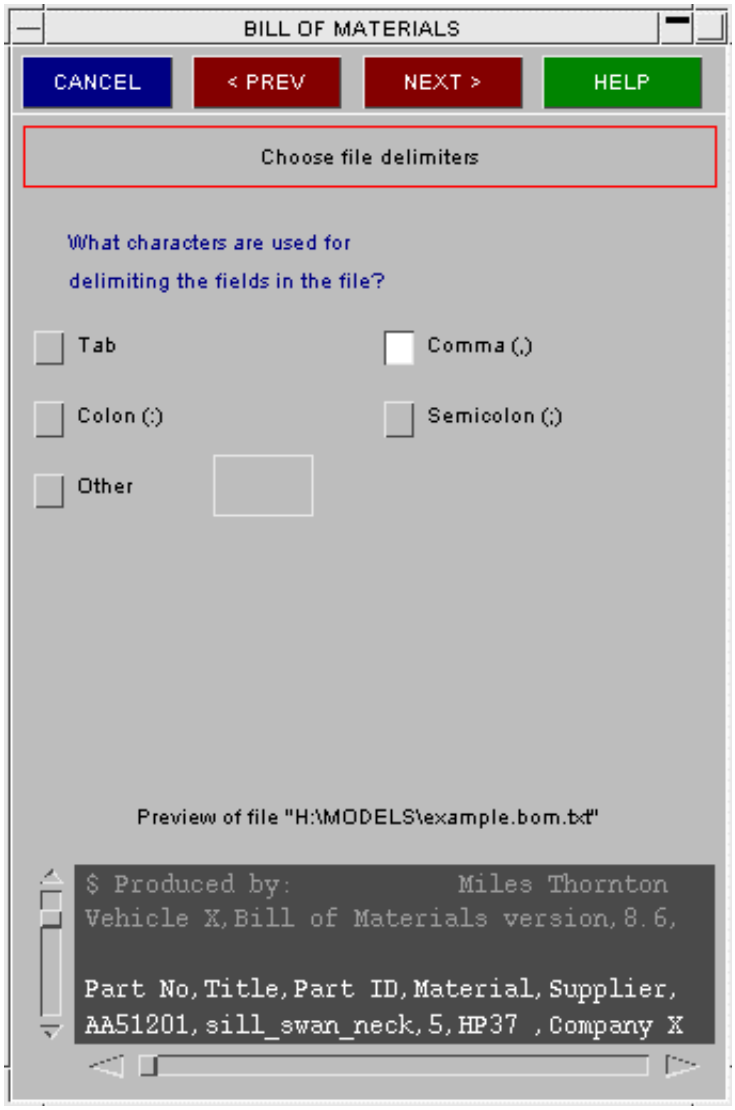
```
Part No,Title,Part ID,...
```

The default delimiter is a comma so this is OK for this example. If the data is separated by another character it can be chosen here. Other buttons are available for common delimiting characters. If your data is separated by a character that is not in the list press the **Other** button and type the character in the box. A space cannot be used to separate fields.

To cancel reading and return to file selection press **CANCEL**.

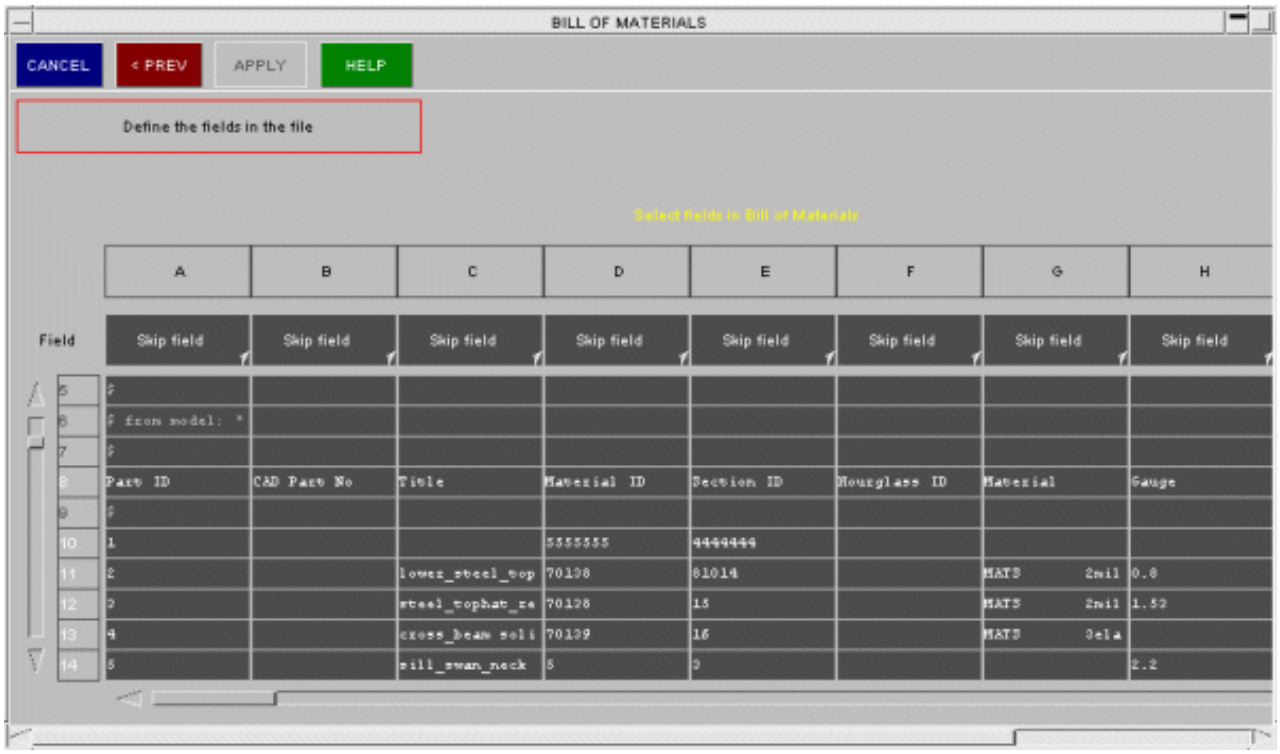
To go back to the previous step (skipping specific lines) press **PREV>**

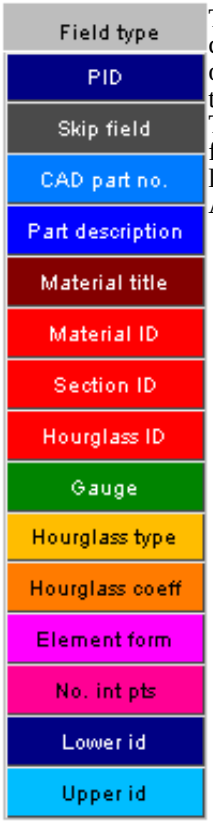
To go on to the next step (defining fields) press **NEXT >**



Defining fields

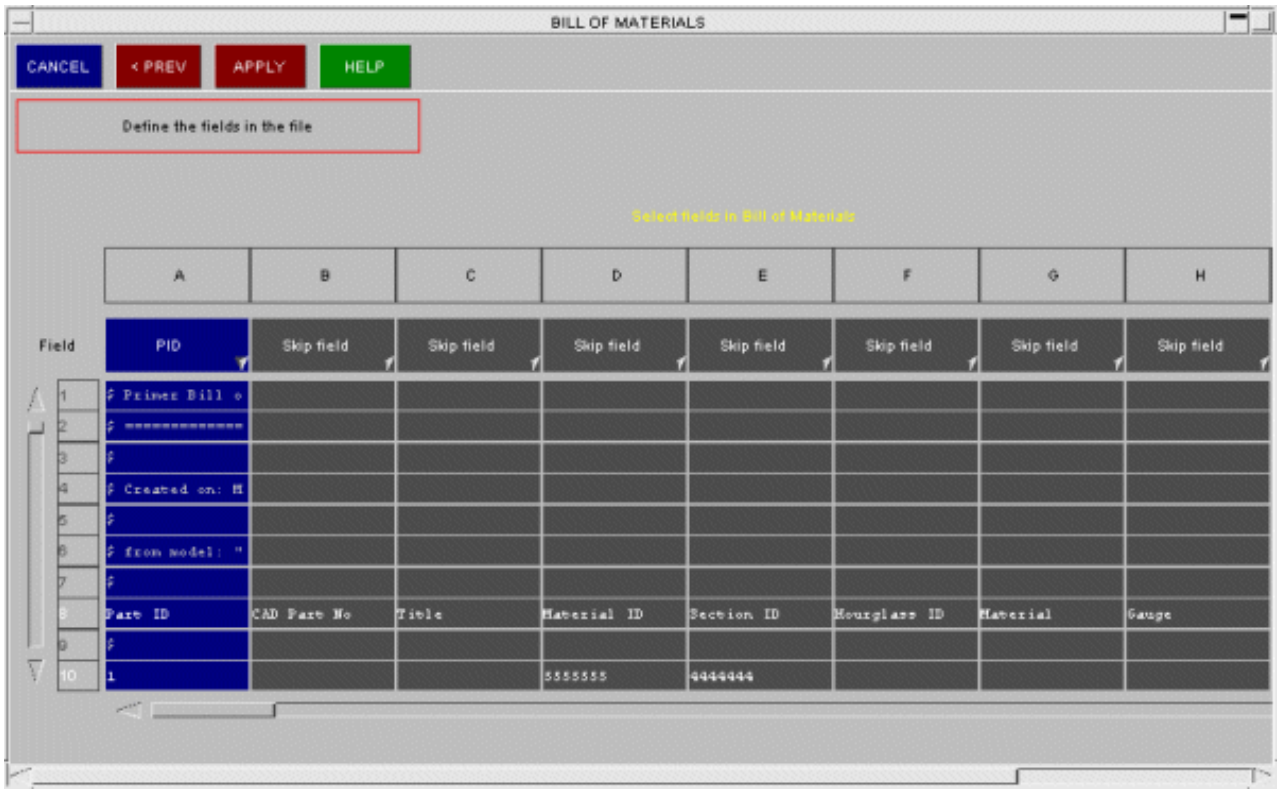
This panel enables you to choose which columns of the bill of materials to use and what the columns mean. A preview of the bill of materials is shown below.





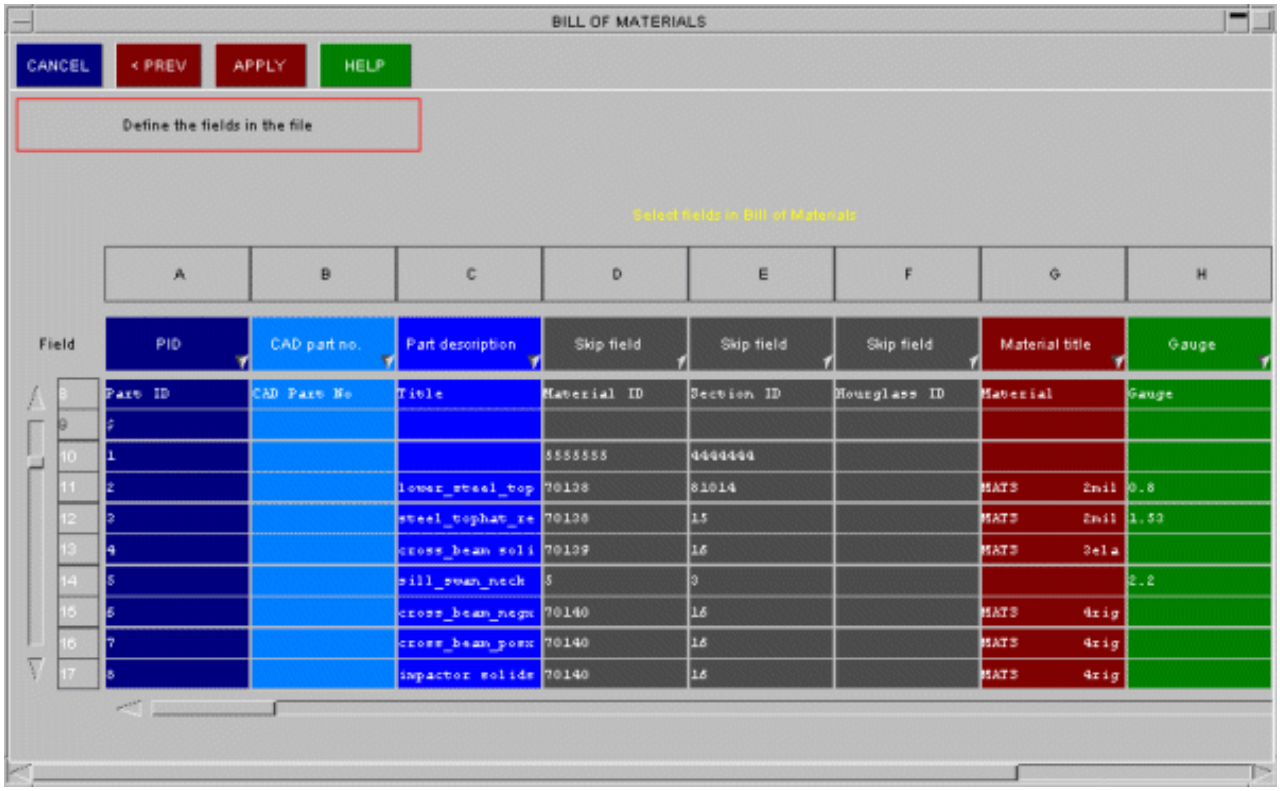
The lines that are going to be skipped are shown in grey rather than white. The data is shown in columns to make it easier to read. If there are more than 10 columns a scrollbar is used to view the other columns. To be able to do anything useful PRIMER needs to know which columns you want to use and what those columns mean. This is done by using the **Field** popup buttons in each column. The default action for each field is 'Skip field'. This can be changed by selecting any of the options from the popup. Once an action is selected the column will change colour and 'Skip field' will no longer be shown.
A field can be unset at any time.

For example, if the field for column A is set to be 'PID' it will be coloured dark blue as shown below.



At least the PID and one other field must be selected. The **APPLY** button will not be active until this is done. Once the button is active, **APPLY** will start reading the file and altering the selected fields.

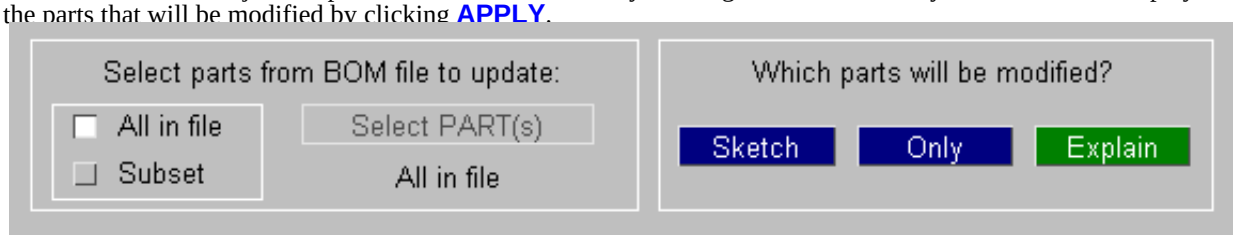
In the following example the CAD part no, Part description, PID, Material title and gauge have been selected.



Only the selected fields will be altered using the bill of materials. In the above example the CAD part no, Part description, Material title and gauge will all be altered as they have been selected. If only the PID and gauge were selected then only the gauge would change.

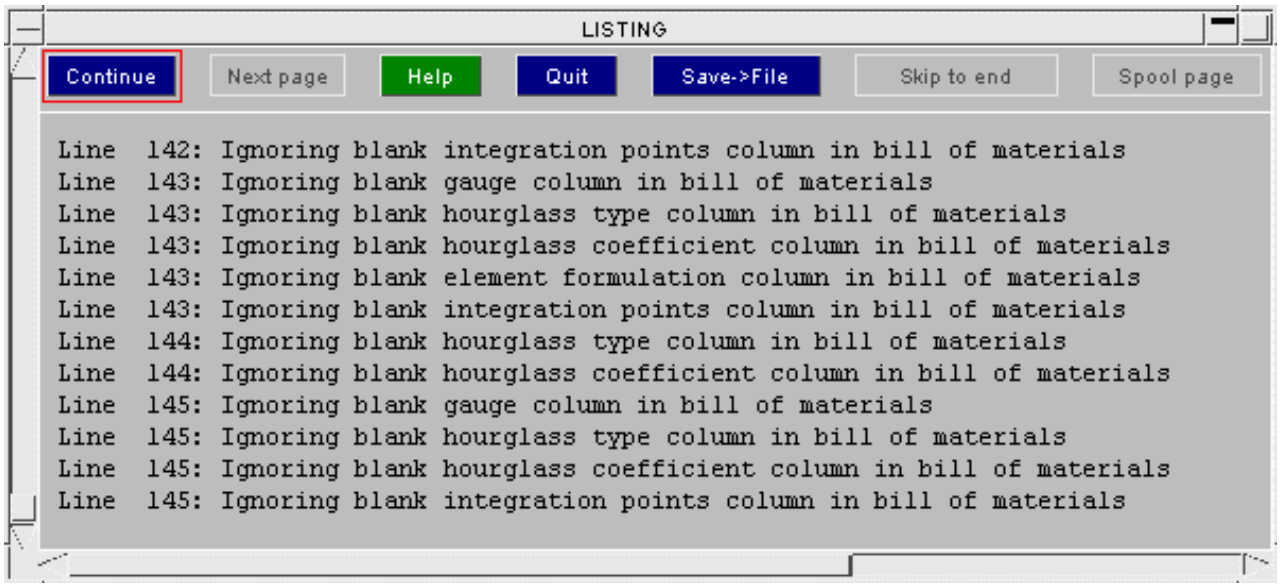
There are two options to select parts from BOM file to update (see below). 'All in file' option will allow you to update all the parts which have differing information to that in BOM file and 'Subset' will enable **Select PART(s).** The

SKETCH will sketch just the parts that will be modified by clicking **APPLY**. Similarly the **ONLY** will display only the parts that will be modified by clicking **APPLY**.



Listing output

As the Bill of materials file is read messages are copied to a listing window.



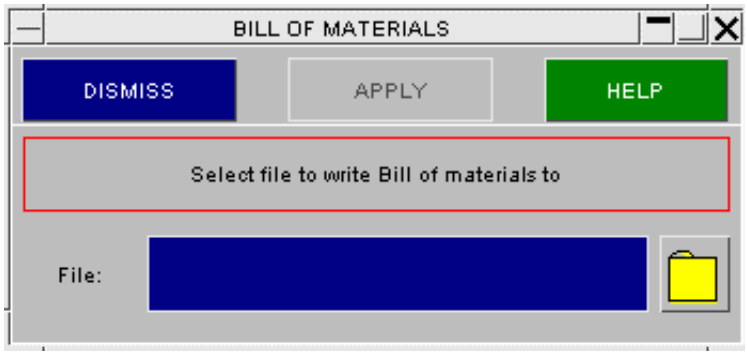
This gives information about what the bill of materials is changing. If needed it can be saved to file by pressing the **SAVE -> FILE** button.

6.4.4 Writing a Bill of Materials file

To write a Bill of materials file type a filename into the text box or use the button to choose a file to overwrite. Once you have given a filename the **APPLY** button will become active and you can write the Bill of Materials.

For each part in the model all the fields given in the standard headers above will be written. In addition 4 extra fields will be written:

- The part mass
- Mass added to the part from assign_mass structures
- Non structural mass added to shell parts (LS 960 and greater only)
- The total mass

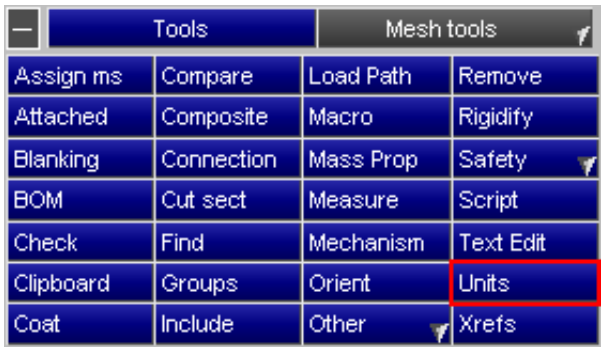


6.5 BLANKING Setting entity visibility.

The blanking menu in the **Tools** panel is covered in a separate section of the manual - see section 4.5 for details.

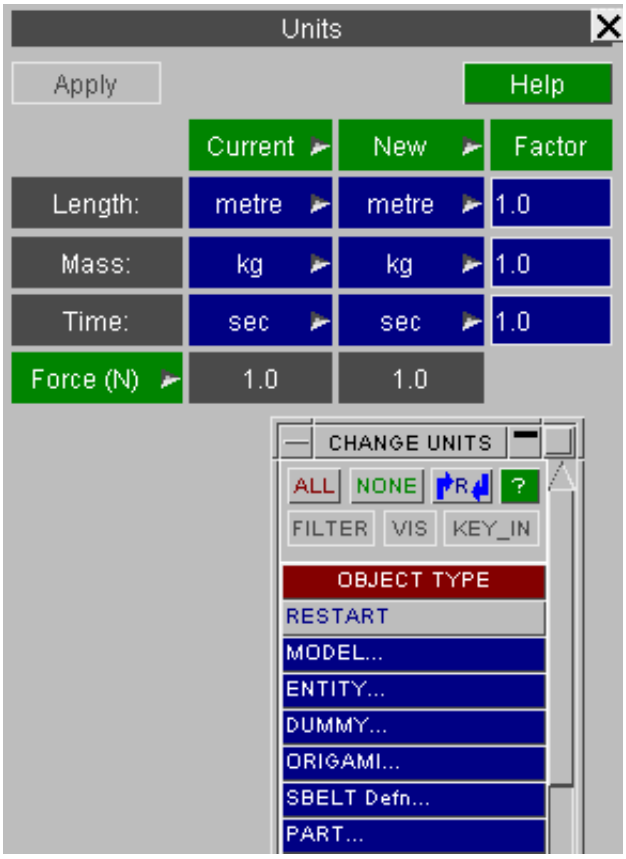
6.6 CHANGING UNITS

Length, Mass & Time



- Units of Length, Mass & Time may be scaled
- A whole model or a selection of items may be converted
- Curves belonging to multiple keywords will be copied automatically, if necessary

6.6.1 Selecting Units



To perform a unit conversion select the current units and the required new units. These are likely to be available under the Current unit and new unit popups, but if not may be selected from the individual unit popups.

The conversion factor will appear in the right hand column. If your units are not available from a popup, you may set the factor directly.

The force unit arising from the choice of units is displayed for your information.

Finally, select the items to which you want to apply the unit change. Primer will automatically propagate the change onto the appropriate items, e.g. selecting a part will get the section and material also.

Special treatment of loadcurves: if the same loadcurve is used by multiple keywords, primer will check the unit type of each and, if different unit types are found, offer the option of copying curves.

6.6.2 How Units change affects Parameters.

When a model uses parameters, and data fields defined by parameters are affected by a units change, then if no action were taken the value of the data fields would no longer be the same as the value of their parameters thus breaking the association between them. This would be unsatisfactory and parameters need to be updated during a units change, but this is not straightforward because parameters do not have intrinsic units.

PRIMER handles this problem as follows:

- During a units change operation any data fields which use parameters "tell" their parameter definition what factor has been applied to them.

- Each affected parameter "remembers" this factor temporarily.
- When the units change is complete all parameters are scanned to see whether factors should be applied, and for those affected:

  Scalar parameters are simply multiplied by this factor to give a new value.

  Expression parameters are more difficult since they may be affected both by modifed scalar ones and by changes to each other. Therefore:
  - Each expression parameter is re-evaluated, and the outcome compared with the <original value> * <any factor from units change>.
  - If there is a mismatch then the outcome is factored to give the required value.
  - This process is repeated iteratively, since expression parameters may reference one another in an arbitrary order, until no further changes are required.

    Once the correct factor has been determined for each expression then it is applied to the text string of that expression as a multiplier

    factor * (*original expression***)**

This process effectively assigns an implicit unit type to parameters and scales them appropriately, so that their new values still match the scaled data fields to which they apply, which means that the association between parameters and data fields remains unbroken.

The particular case of expression parameters referring to other parameters in a conflicting way.

Consider the following two parameters:

*PARAMETER	**TIME_1**	Value = **1**
*PARAMETER_EXPRESSION	**TIME_2**	Value = "**TIME_1 + 1**" = 2

If the model is now scaled from seconds units to milliseconds each of these two parameters must be multiplied by 1000. This gives the result:

TIME_1 becomes	1 * 1000	Correct. Should be 1000
TIME_2 becomes	either (TIME_1 + 1) = 1001 or (TIME_1 + 1) * 1000 = 1001000	**Wrong**!! Should be 2000

Prior to PRIMER V13 this resulted in the connection between the data field (which would be scaled to the correct value) and the parameter (which would have the wrong value) getting broken. So the model would be numerically correct, but there would be a loss of parameter usage.

From PRIMER V13 onwards the problem is solved by converting expressions such as the above so that parameters inside an expression are scaled by the reciprocal of their "own" factors, effectively restoring them to their pre-scaled state, then the end result is scaled by the correct factor for the expression as a whole. So in the case above the result would be written:

TIME_2 becomes	((TIME_1 / 1000) + 1) * 1000 = 2000	**Correct**!! Should be 2000

If for some reason the above logic fails to give the expected result then the behaviour in V13 onwards is:
- The new definition, here TIME_2, is converted to an expression that is a simple number equal to the wanted result, here 2000.
- The original parameter definition is copied verbatim to TIME_2_OLD, providing a record of its contents.

This means that the connection between the parameter and its usage in the model is not broken, and that there is also a record of what the parameter used to be before its expression was changed to a plain number. The user is warned about this, and needs to resolve the situation manually if required.

The "_OLD" definition is not referred to in the model, and can be deleted if not required.

Situations in which units change of parameters may fail.

This process normally works well, but can fail for either or both of the following reasons:

If a parameter is used inconsistently:

For example if a parameter is used in a data field of length units, and also one of time units, then two different factors may be notified to the parameter definition. In this situation the most recently notified definition "wins" and will dictate the factor applied to the parameter, meaning that its association with some of the other data fields will be broken.

The solution is obvious: don’t use a parameter in incompatible contexts.

If only part of a model is subjected to units change, but parameters are used in all of it:

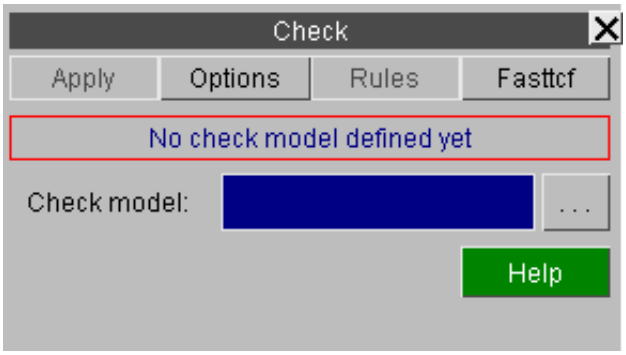
In this situation factors from changed fields will cause the parameters to be updated, which may break their association with unchanged data fields.

The solution is not to change units of a subset of a parameterised model. It is normally the case that a units change is required when models from different sources are merged together, and the units change should be applied to the model as a whole ***before*** it is merged.

6.7 CHECK Running the model checker, and setting its options.

The model checker is run from the **CHECK** option in the **Tools** panel - see section 3.9 for details.

6.7.1 The CHECK popup menu



APPLY Runs the model checker as described in section 3.9.

OPTIONS Maps the **CHECK** options panel as described in section 3.9.1

6.8 CLIPBOARD

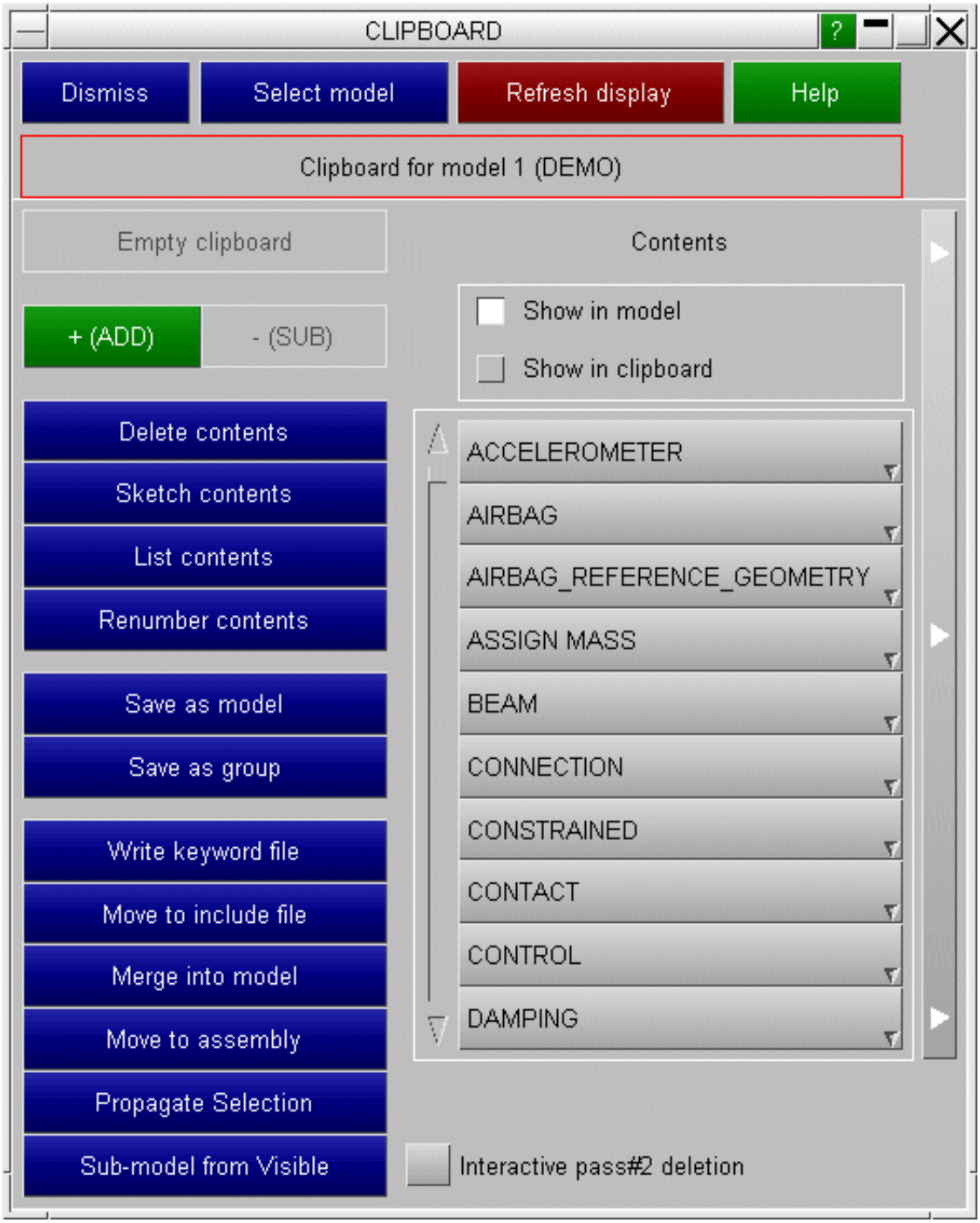
The clipboard function allows the user to work on a subsection of a model. Each model has its own clipboard and the clipboard contents remains part of the model. The clipboard contents can be;

- renumbered - this is useful for renumbering nodes and elements by part
- interrogated for their locations - useful when a model contains many include files
- saved as a group or new model - a quick way of obtaining a complete sub model
- written as a separate keyword file - useful when only specific parts need to be modified in a separate application
- moved to a new or existing include file - useful for model organisation
- merged into another model - a quick way of building new models
- reuse of selections in general object menus

The clipboard panel is displayed below

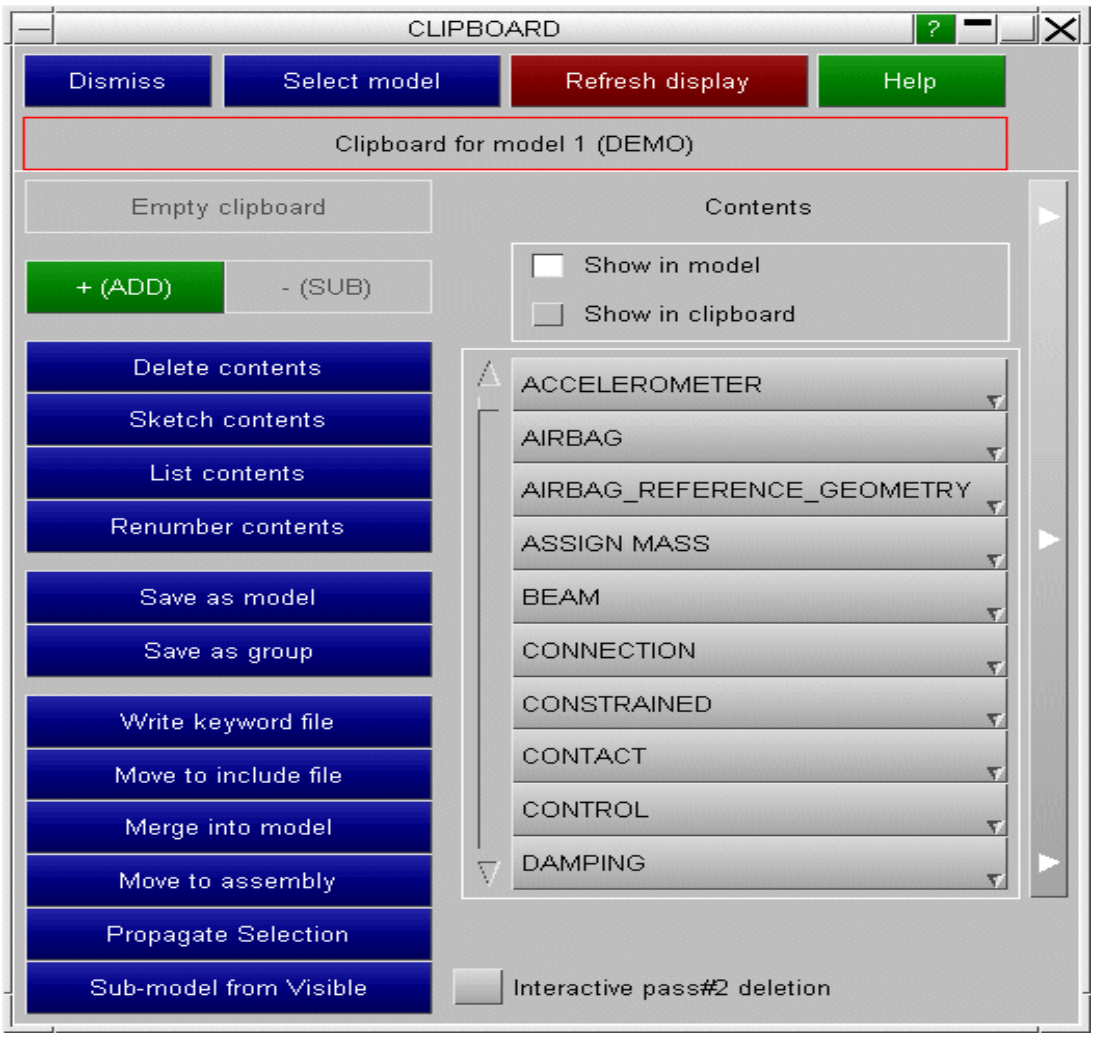
6.8.1 Adding and Removing items from the clipboard

Items can be added and removed from the clipboard by using **+(ADD)** and **-(SUB)**

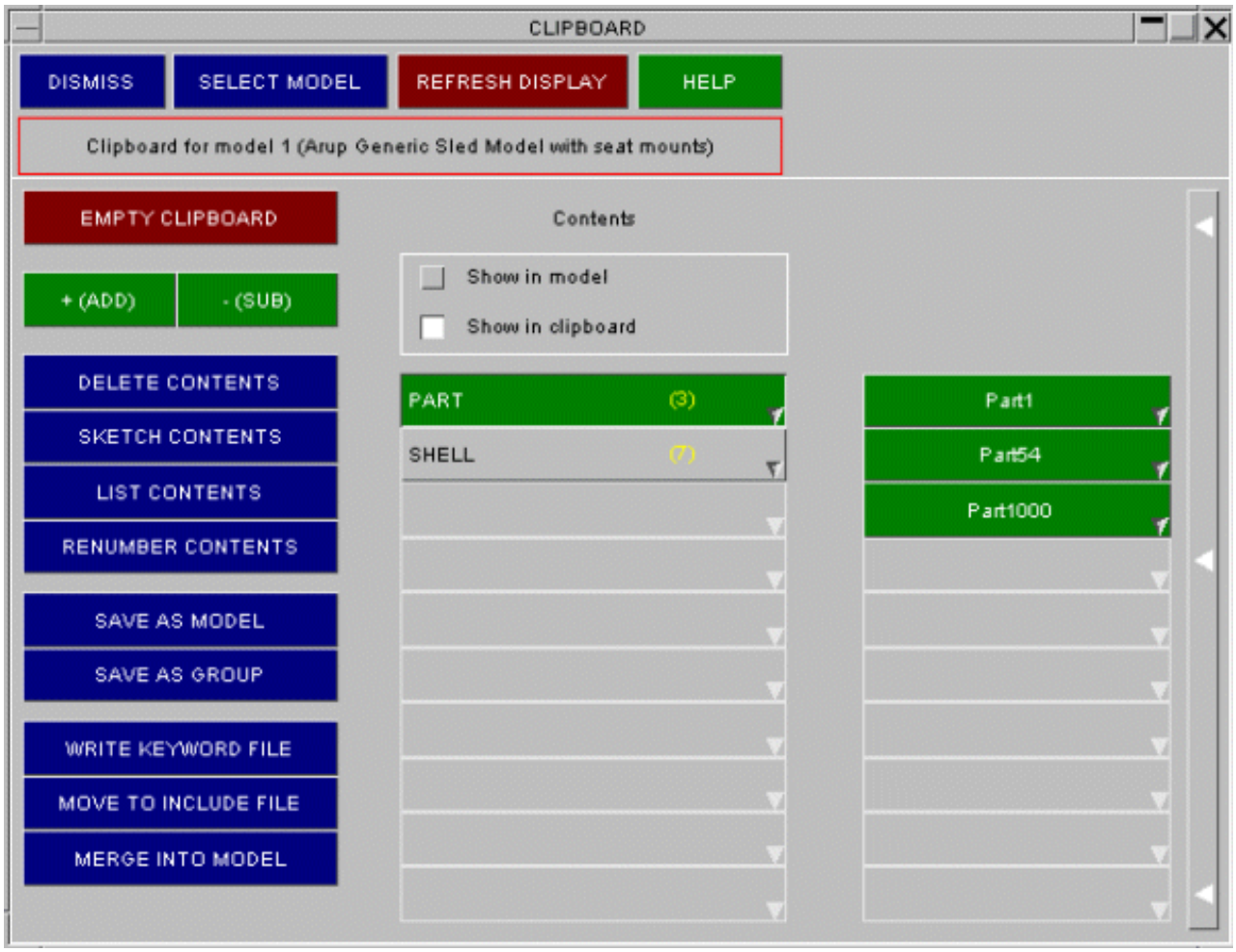


6.8.2 Display of Items on the Clipboard Panel

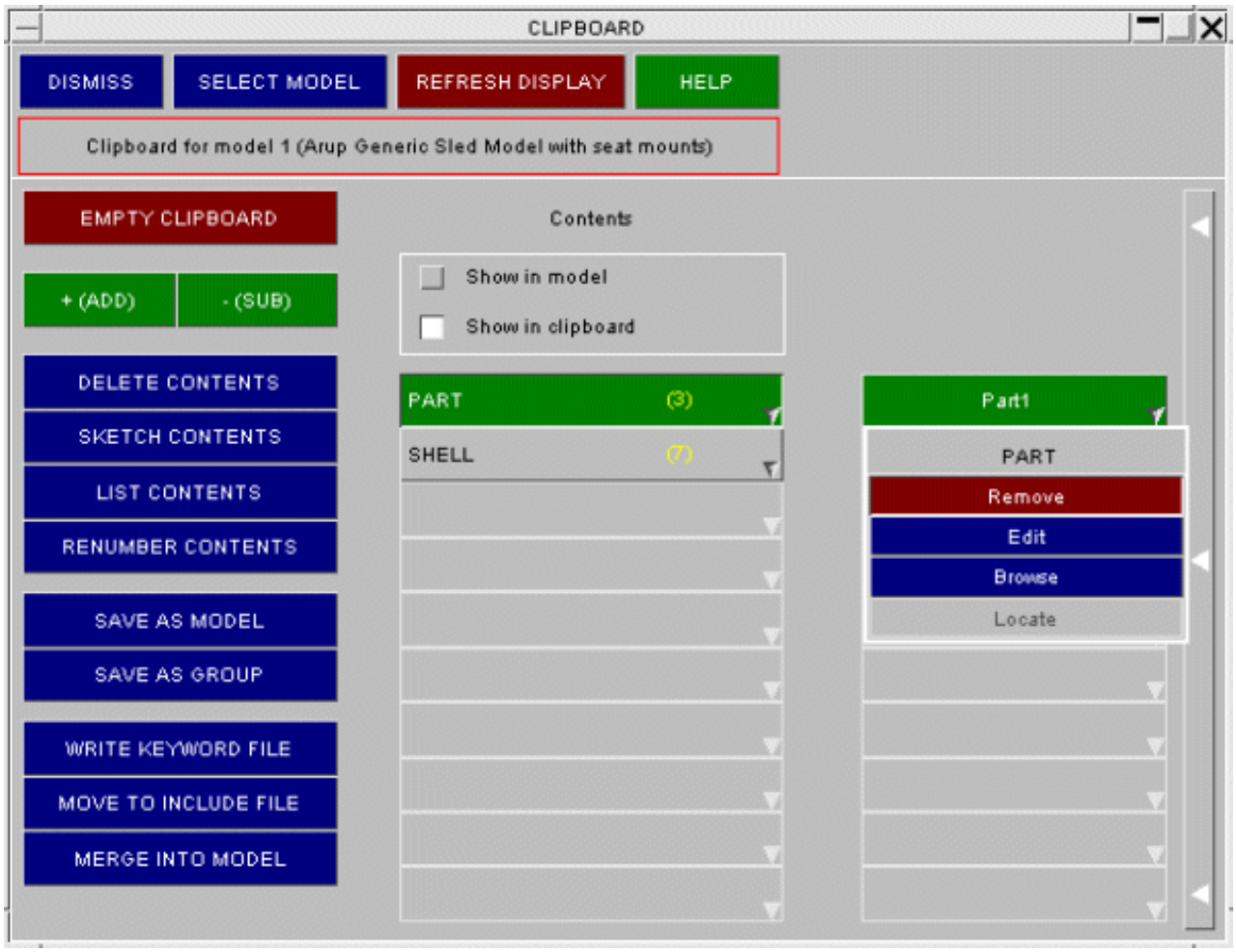
There are three modes of displaying items in the clipboard panel. The contents display can be set to show all types of entities in the model by activating **show in model**. If entities are in the clipboard the number present will be displayed in brackets. **Show in clipboard** will show only entities present in the clipboard with the number present again displayed in brackets.



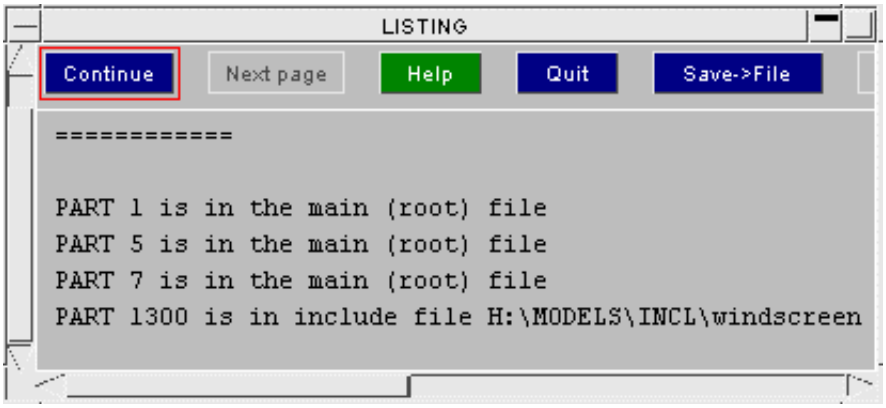
The bar at the right hand side expands the clipboard panel to show all members of an entity group present in the clipboard once that entity is selected by left-clicking the mouse over its button.



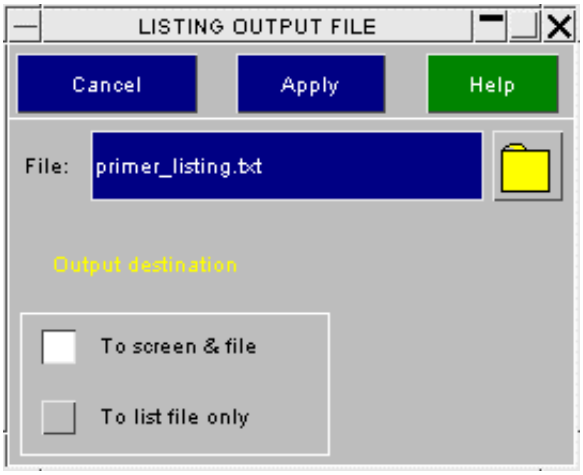
Right-clicking on the items in the right hand column will allow removal, editing or location of that item.



The location of all of a particular group of entities can similarly be found by right-clicking on the relevant entity button. Selection of **locate** will bring up a panel that will scroll through all entities present in the clipboard with their locations.



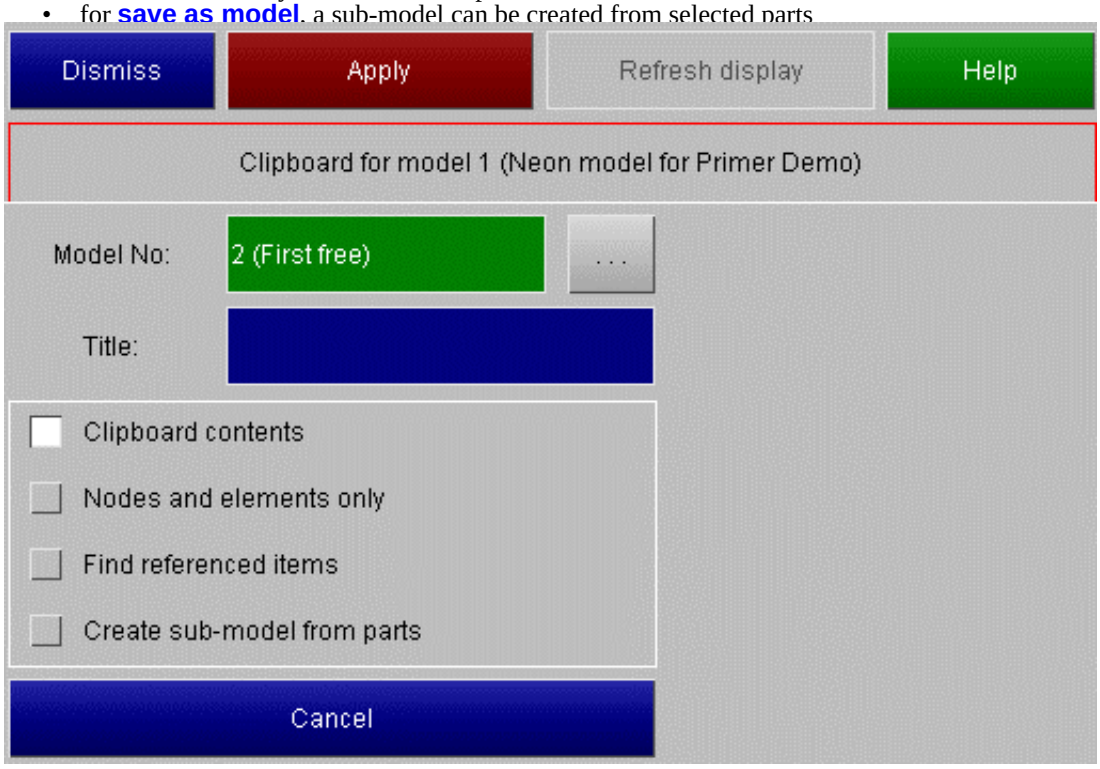
This information can also be output to a text file.



6.8.3 Referencing of Clipboard Items

When saving the clipboard contents to a new model or writing a keyword file there are options to select the entities that are written.

- The clipboard contents can be written as they are.
- Nodes and elements in the clipboard and those belonging to parts in the clipboard can be written.
- Items referenced by entities in the clipboard can be included
- for **save as model**, a sub-model can be created from selected parts

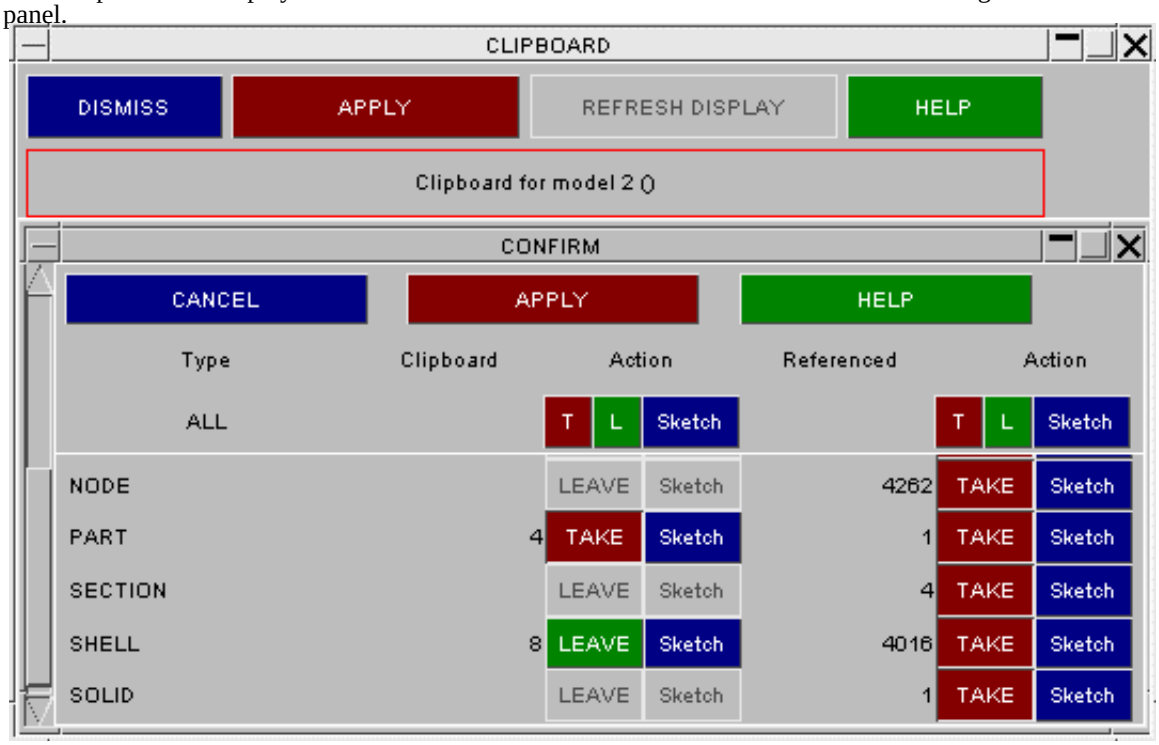


Find items referenced is useful as PRIMER will find cross-referenced entities necessary to produce a more complete model. It will not, however, find items such as *CONTROL or *DATABASE which have no direct reference to the selected items - **sub-model from parts** and **sub-model from visible** are also worth considering

PRIMER finds entities according to a hierarchy. Some examples of this follow;

- When finding referenced items, a clipboard containing an element alone would first find the nodes of that element and the part referenced by the element. However this is not a complete model as the material and section of the part also need to be included. PRIMER therefore carries on checking until all necessary entities are found. A clipboard containing only nodes would not find any elements or parts as these are not referenced and are 'downstream' of nodes in the PRIMER hierarchy.
- Clipboard contains a *`CONTACT`. If the contact referenced a *`SET_PART` then this would be found together with the parts on that *`SET_PART`. Elements and nodes of the parts would also be found.

When PRIMER has finished finding all entities the following panel is displayed and at this stage the user has control over selection of items by clicking on the relevant button to take or leave each group of entities. Items originally present in the clipboard are displayed in the centre column and referenced items are shown in the right hand column of the panel.

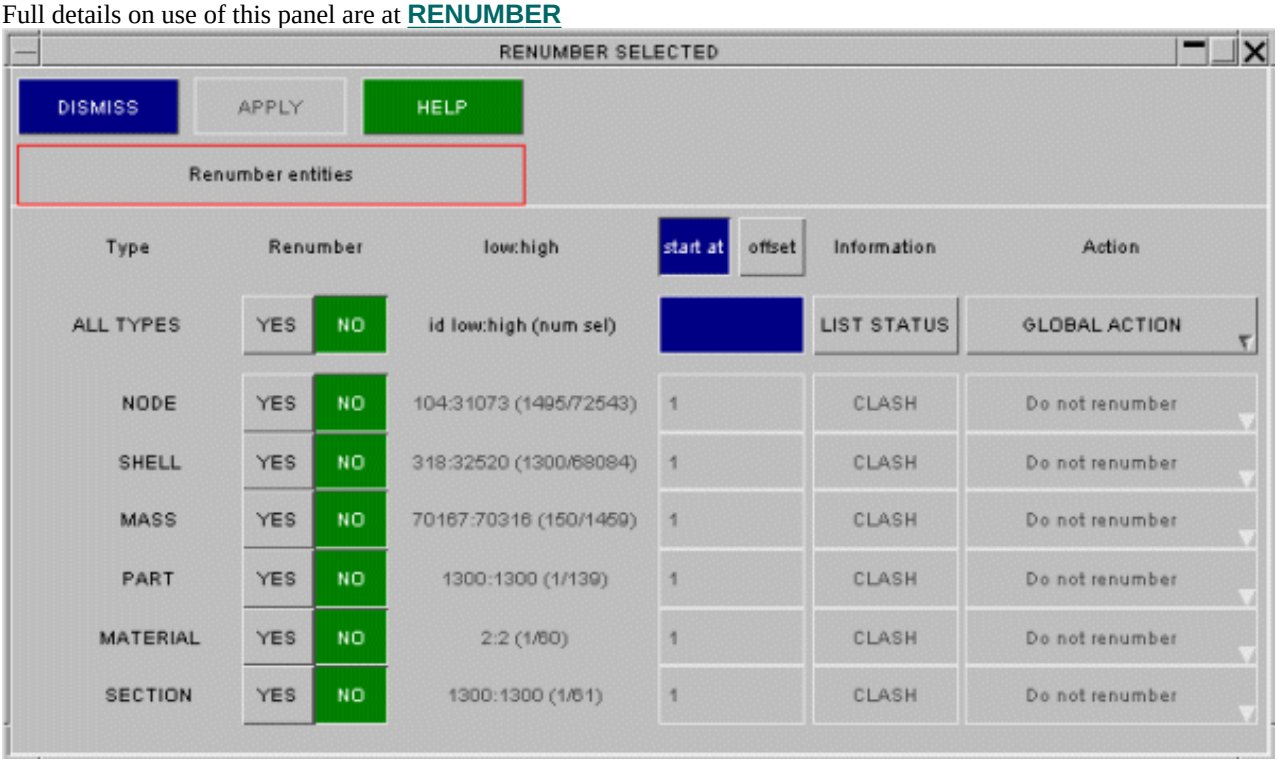


6.8.4 Renumbering of Clipboard entities

This function makes node and element renumbering by part association a simple task.

The panel shows what appears when the clipboard renumber button is operated. The clipboard contains one part and referenced nodes and elements are found and automatically selected for renumbering.

Full details on use of this panel are at **RENUMBER**



6.8.5 Saving Clipboard entities as a new model/keyword file

Sub sections of a model can be added onto the clipboard and then saved as a new model in PRIMER or written out as a new keyword file. The clipboard allows you three options:

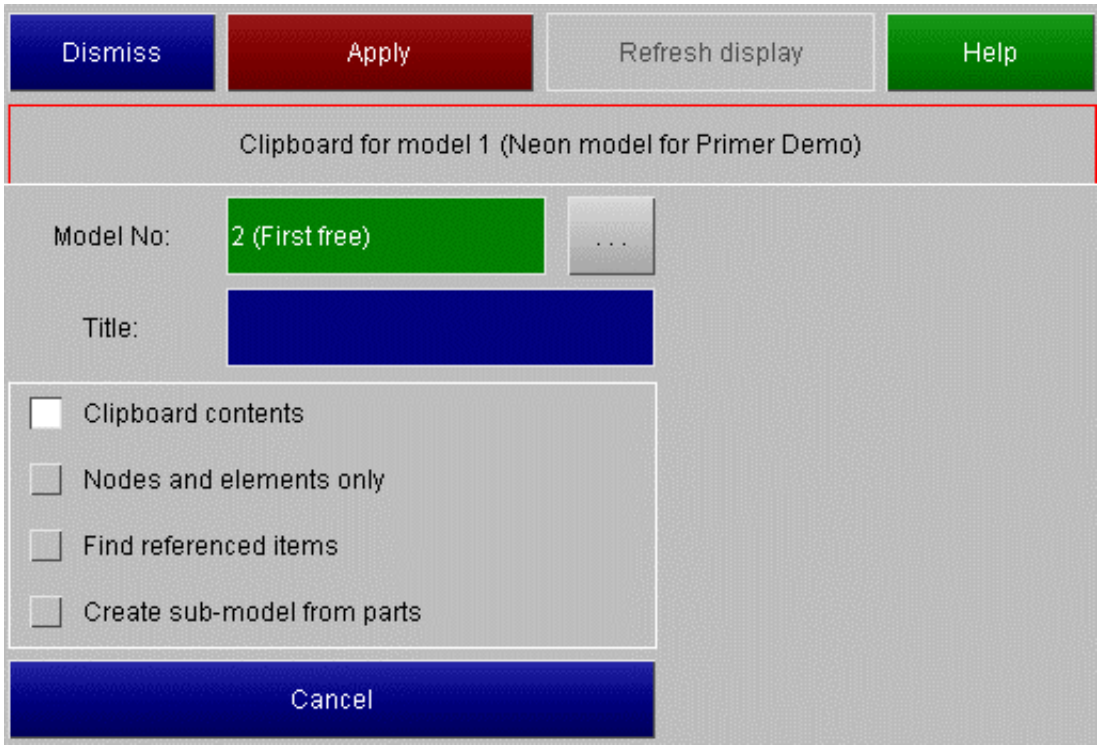
1/ The clipboard contents can be written as they are.

2/ Nodes and elements in the clipboard and those belonging to parts in the clipboard can be written.

3/ Items referenced by entities in the clipboard can be included if desired. In this case a complete stand alone model will be produced but it won't necessarily contain everything wanted.

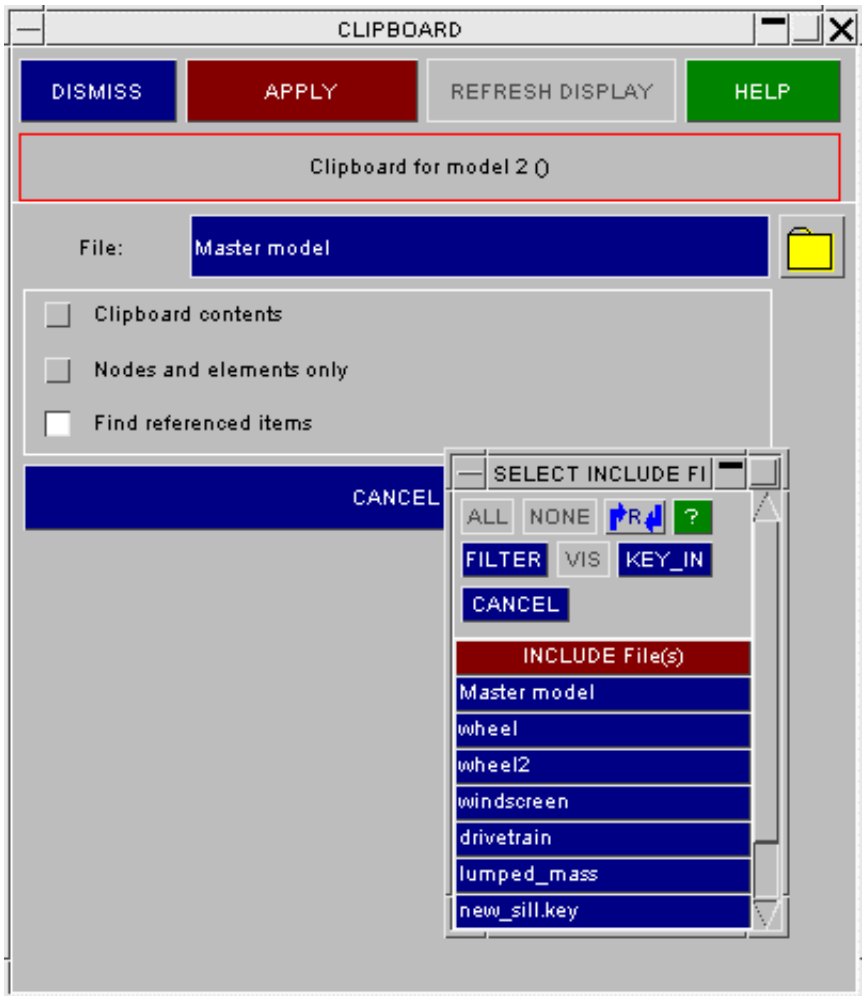
4/Create sub-model from parts is designed to generate a model from selected PARTs (items of other type are ignored). It will propagate the part selection, through sets as necessary and give a model which contains items such as sets, contacts, database cross-section, etc which are deemed to belong with the parts. Additionally, the function will find Primer connections which attach to the parts and (optionally) export them to the sub-model.

When writing the clipboard contents to a new keyword file, you can access the writing options by clicking on the **>>> LS-Dyna output options** button.



6.8.6 Moving Clipboard entities into include files

The clipboard provides a powerful and controllable way of moving items into include files or from one include to another.



The panel above appears when the **move to include file** button on the main clipboard panel is operated. There are three possible options that allow the user control;

1/ The clipboard contents can be moved

2/ Nodes and elements in the clipboard and those belonging to parts in the clipboard can be moved

3/ Items referenced by entities in the clipboard can be moved

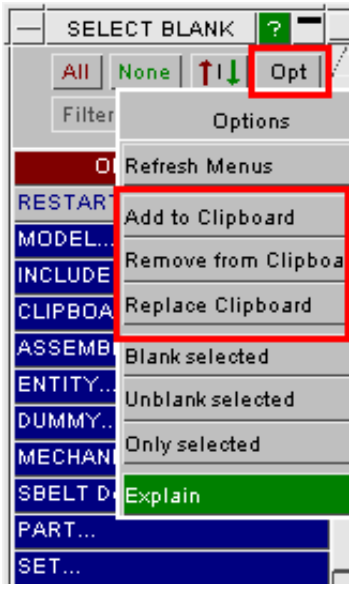
Depressing the [?] button will bring up current include files. These can be selected directly from the subpanel or a new filename typed in the box after **File:**

Selecting **Find referenced items** will bring up a panel allowing the user control of referenced items

6.8.7 Clipboard merge into model

This allows items in the clipboard to be merged into another existing model in the PRIMER session. This function works in exactly the same way as **MODEL > MERGE**

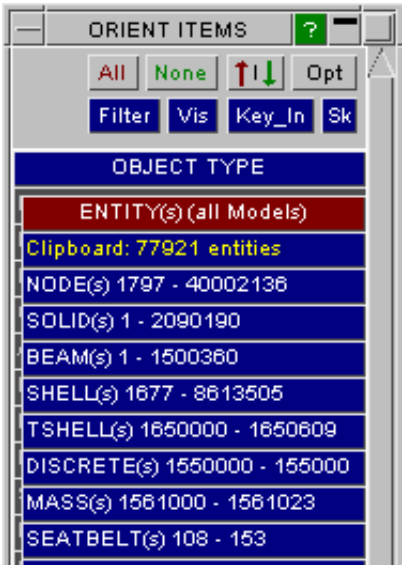
6.8.8 Clipboard usage in Object Menus



The Options button in all standard Object Menus includes options to:

- Add the current selection to the clipboard
- Remove it from the clipboard
- Replace the clipboard with the currently selected objects

This acts as an alternative way of manipulating the clipboard contents.



The clipboard contents may also be used for selection in object menus, as this example illustrates.

In all object menu contexts where:

- Multiple selection is legal

  and

- The clipboard contains 1 or more items of the specified type

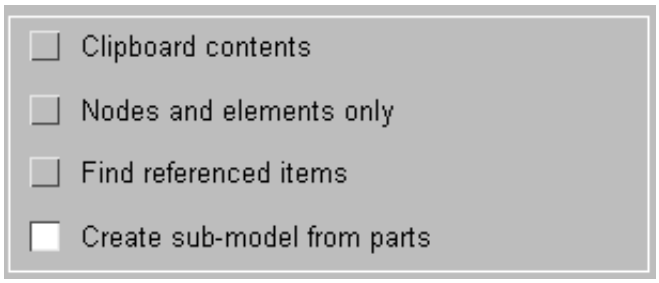
Then a "**`Clipboard: nnnn <item type>`**" row will appear at the top of the menu selection.

If this is selected then all items in the clipboard which are also legal in this context may be selected in a single click by choosing this "Clipboard" row. Items in the clipboard which do not match the currently specified type are not included.

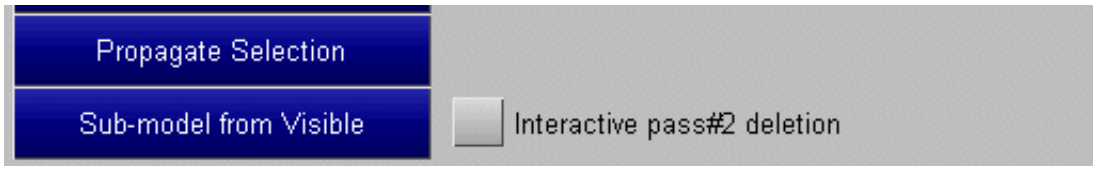
This provides a means of reusing selection in a range of different contexts.

6.8.9 Sub-model creation

The find referenced item option (under save as model) described above can be used to create a sub-model but will require the user to make an detailed selection of items and propagation may not give the expected results, e.g. if a contact using part-set is selected, then all parts of the set will be found, not just the ones on the clipboard. Try putting contact on clipboard and pressing **propagate selection**.



If parts only have been selected **create sub-model from parts** option may work better. This will propagate the selected parts to find sets, contacts, database cross-sections,etc that use those parts but will NOT bring in the unselected parts of those sets (or their elements). Additionally, it will find Primer connections on the selected parts and offer to export them to the sub-model. As with find referenced, it will not capture items (such as *CONTROL) which have no direct reference to the selection.



Sub-model from visible is a tool which applies deletion and cleanup processes to make a sub-model.

- The entire model is copied to the next free slot
- Pass#1 deletion is applied to all items which have been blanked (and elements for which the entity switch has been turned off)
- Iterative cleanup-unused is then applied.
- Pass#2 deletion is then applied to remove items which have no visibility in themselves (e.g. seatbelt defs) but may be blocking deletion of items which should go (e.g.seatbelt elements)

The result should be a sub-model consisting of the visible items of the original model but retaining the all important *CONTROL, *DATABASE,etc cards internal connections (such as *CONSTRAINED_NODAL_RIGID_BODY) even if they were not visible.

Pass#1 deletion is the removal of

6.34.1 CLONES

How PRIMER handles duplicate definitions using Clones.

Theoretically an LS-DYNA input deck should only have a single instance of any labelled item A, for example ***NODE** label 1 should only ever appear once. "Unique" keywords such as ***CONTROL** should also only occur once. However LS-DYNA tolerates duplicate definitions in a range of ways:

- ***NODE** definitions may be repeated so long as they obey certain rules about coincidence. See **[NODE] Duplicates** for more information on how these are processed in PRIMER.
- ***PARAMETER** definitions can be repeated by using suffices such as **_LOCAL** and **_MUTABLE**, and handling of repeated definitions is further controlled via the ***PARAMETER_DUPLICATION** card. See **PARAMETER**s for more information on how these are processed in PRIMER.
- LS-DYNA also tolerates repetition of some other keywords, usually by ignoring all but the last instance found in the input deck. PRIMER tolerates this subject to the [Options] Permit duplicates keyword input setting.

Unlike LS-DYNA, which can ignore all but the solitary definition it chooses to use, PRIMER must "remember" these duplicate definitions so that they can be written out again correctly, otherwise the sequence "read and write model" will end up removing entries from keyword files, and it does this by creating "clones".

What is a clone?

A cloned definition is not a copy of a keyword, rather it is a ***reference*** to the original "true" definition, and is subject to the following rules. In the definitions below "include file" can also mean the master file.

- There is only ever a single true definition of an entity inside PRIMER. This will normally be the first such definition read from the input deck, and this will dictate which include file it exists in during the PRIMER session. However if differences are detected in duplicate definitions you can choose which to use, see below.
- Any number of "clones" may refer to the true definition A, subject to the limitation that an include file may only contain a single definition of item A. This means that:

  - A clone of A may not exist in the same include file as the true definition of A.
  - An include file may only contain a single clone of item A.
- During keyword output the external keyword format of the true definition is written out in its "true" include file and also in every include file in which a clone of it exists.

Inside PRIMER each model has a list of clones, which can be thought of as pseudo-keywords. They show up in cross-reference lists of true items, but otherwise they are mostly invisible and can be ignored.

What happens if duplicate definitions contain differences?

Special rules apply to the processing of ***NODE** and ***PARAMETER** cards, but for all other duplicates PRIMER applies the following rules during keyword input.

- Each duplicate definition is stored temporarily until keyword input is complete.
- Then each duplicate definition is compared with the original "true" definition, searching for any differences.
- If no differences are found then the first definition is used as the "true" one, and subsequent definitions become clones. This process is silent.
- If differences ***are*** found then these are listed and the user is asked how to handle them. The options available are:

USE_LAST	Typical LS_DYNA behaviour. The **last** read definition becomes the true one, and all others become clones.
USE_FIRST	The **first** read definition is the true one, the remainder become clones.
MASTER_OR_LAST	If a definition exists in the master file this becomes the true one, otherwise the last read is used. The rest become clones.

MASTER_OR_FIRST	If a definition exists in the master file this becomes the true one, otherwise the first read is used. The rest become clones.
ABORT	Keyword input is aborted and the model is deleted from PRIMER's memory.

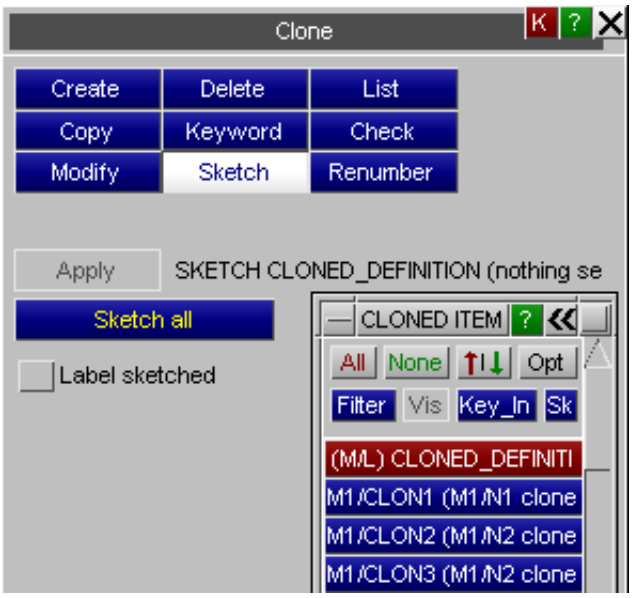
It must be emphasised that inside PRIMER only a single true definition exists, and on keyword output this true definition is simply repeated in each clone location.

Therefore any differences in the content of duplicate definitions will be lost when the deck is read into PRIMER.

This may seem like a limitation, but in fact it is a safety measure. If your deck contains duplicate definitions that have different contents this may well be an error, and even if it is not an error then having differences in these definitions may lead to subtle and hard to find errors during the analysis since the values used will depend on the order in which include files are read. In addition LS-DYNA does not state explicitly how it processes duplicate definitions, so any change in its behaviour in future versions may also affect your results.

It is ***strongly*** recommended that you treat any differences that PRIMER detects as an error, and sort them out manually.

The top level **CLONE** panel



Although the Clone panel is accessed from **[Tools] Other**, it is really a pseudo-keyword and as such its functionality mimics that of conventional keywords.

Create, **Modify** and **Keyword** functions are all performed using the keyword editor, see Keyword editing below. **Delete**, **List**, **Check** and **Renumber** all work as for other keywords.

Drawing Clones

Clones are not drawn separately on plots because they have no separate existence, and there is no separate `CLONE` entry in the **ENTITY** panel for the same reason, so the only way of visualising them explicitly is to the **Sketch** options in this panel, as shown here. You can select a subset of definitions from the menu for sketching, perhaps using filtering options to narrow down what is shown and then using **Apply** to draw them. Or **Sketch all** will draw everything.

By default only the items themselves are sketched, however if you turn on **Label Sketched** then the original item labels, the clone label and also the incoude file in which the clone lives will all be added to the plot.

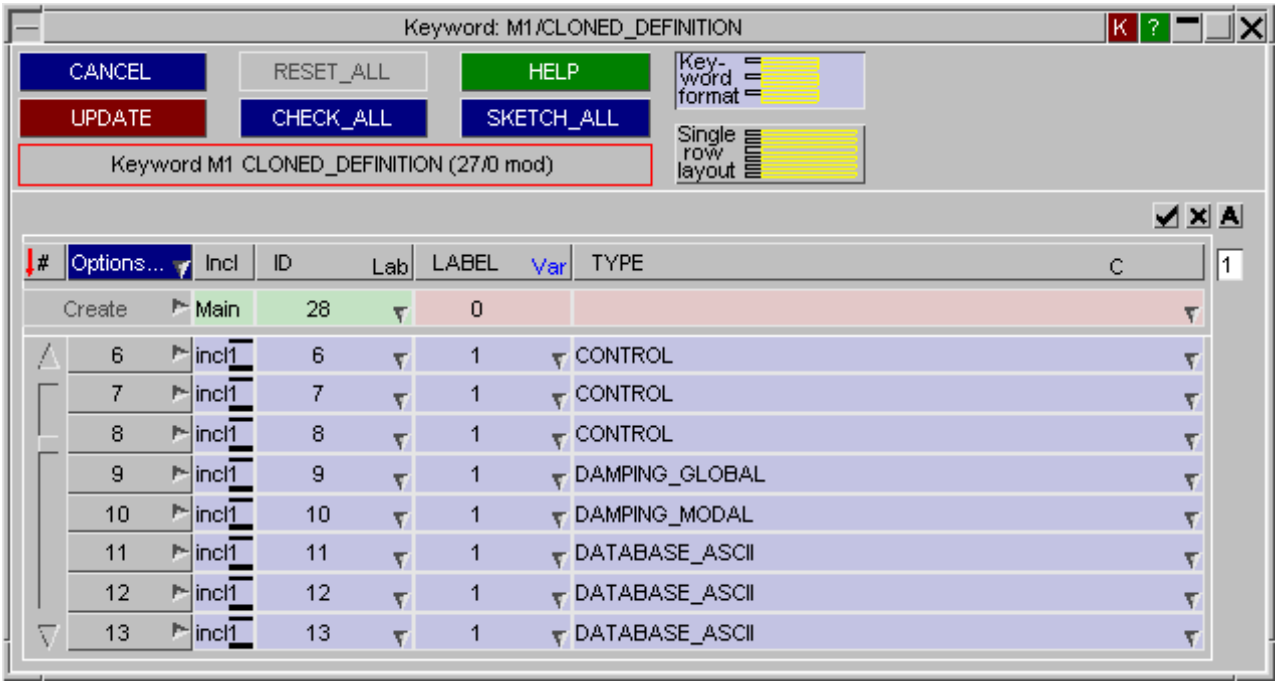
Keyword editing clones

Since clones are pseudo keywords you can think of them as having the keyword format:

ID (clone label)	**TYPE** (keyword type)	**ITEM** (label of duplicate item)
label 1 .. n	eg NODE, SOLID, MAT	label 1 .. n

The clone ID is purely for reference inside PRIMER, and clones will be created sequentially from 1. It has no relationship with the actual duplicate **ITEM**.

Here is an example of the clone keyword editor in a model containing duplicate items of various types.



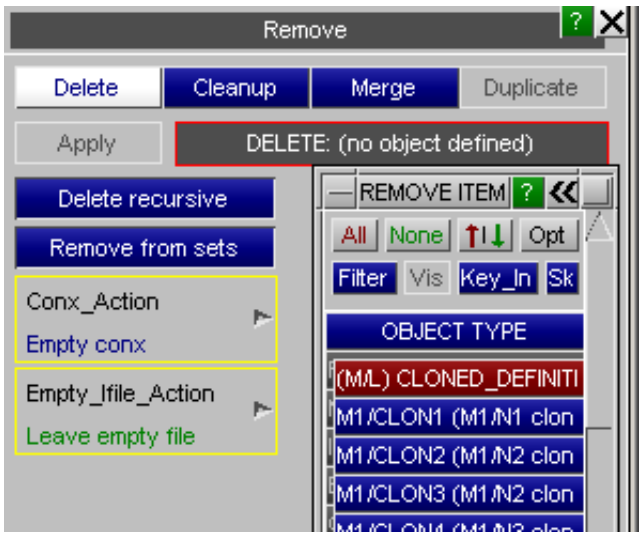
Definitions can be created, edited and deleted here. The **`Incl`** column shows the include file in which each clone exists, ***not*** the include file of the true definition.

Deleting clones

Although clones can be deleted in the keyword editor above this is a cumbersome method, and they can more easily be deleted using the normal **Remove** logic.

Simply select the category **Cloned Items** and then select clones for removal in the normal way.

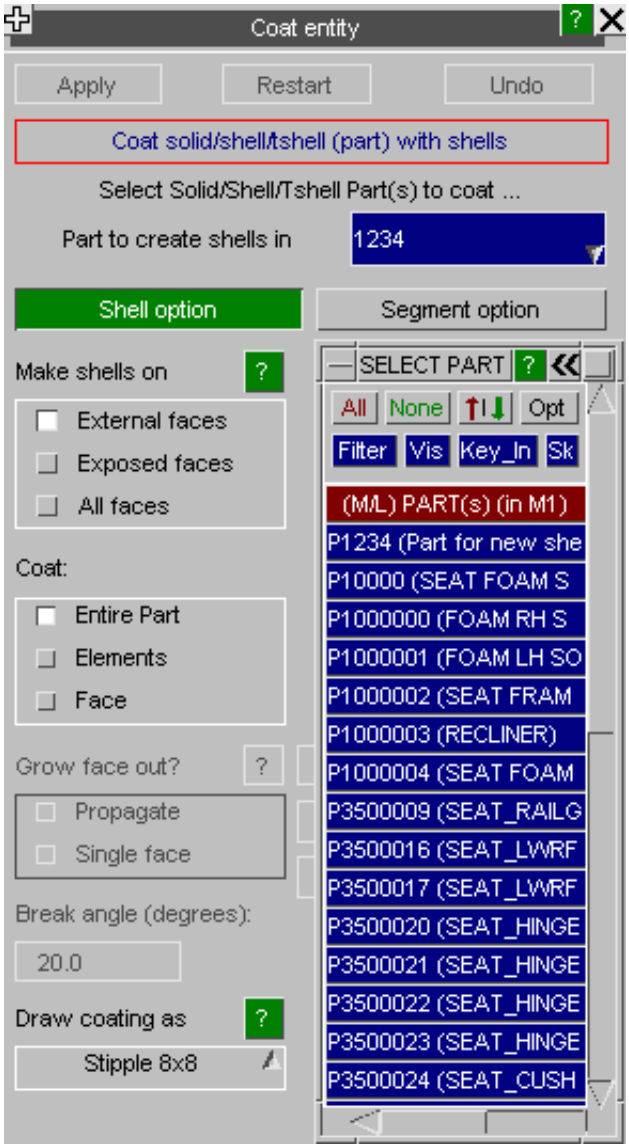
Deleting a clone has no effect on the "true" definition of the item, it only removes the clone definition itself.



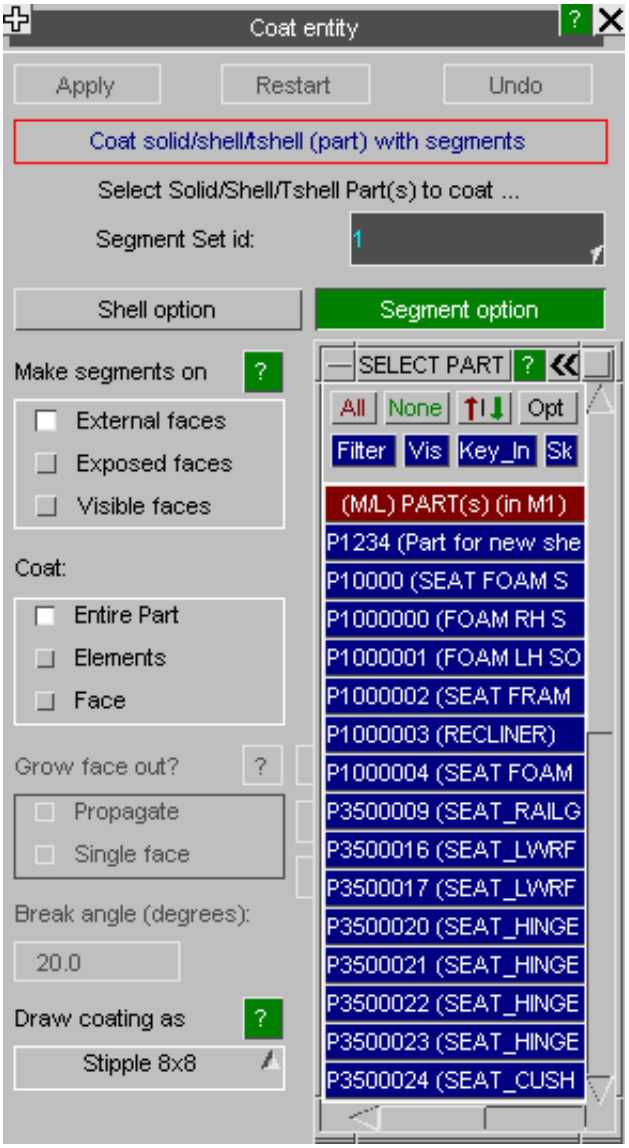
.

6.9 COAT ENTITY: Coating entities with shells or segments

It is often the case that a solid part needs its external surface coating with shell elements (usually null shells) for purposes such as defining contacts. This process is laborious when carried out manually, and the **COAT ENTITY** function is designed to do it automatically.



Coating with shells



Coating with segments.

Coat Part

The function is generalised so that it will coat any solid or shell part(s) with either shells or segments. Furthermore, when coating solid parts with shells the option to coat internal faces is offered.

Any number of solid/shell parts may be selected through the respective object menus. For each coating operation, the new shells/segments created will always reside in a single part/set. Thus if two (or more) adjoining solid parts are *selected and coated together* (with the internal face option off for shell case) only the external faces will be coated.

Coat Elements

Selecting this option will permit users to coat specific solid or shell elements instead of a part.

Coat Face

This option allows users to coat specific solid faces with shells/segments. Two additional options are available for coating solid faces:

- **Propagate** - Users are directed to pick a face on any solid or shell element, and the associated break angle. All faces on that element and on adjacent elements that define an angle with the selected face that is less than the break angle are coated. This is the default mode for the **Coat Face** option.
- **Single Face** - The selected face on a solid element is coated. All other faces are ignored.

Coating with shells:

Part to create shells in may be an existing or new part in which the "coating" shells will be created. This can be any valid part, with "null" or ordinary structural materials.

Three options are available for coating element faces with shells:

- **External faces** - For 3D elements, shells are created on topologically external faces of the selected elements. These would be faces that would be visible if no elements of this type were blanked.
- **Exposed faces** - For 3D elements, shells would be created on the exposed faces of the selected elements. These would be the elements that would be visible if all but the selected elements were blanked.
- **All faces** - All selected faces including internal faces will be coated.

Coating with segments:

Set for segments. By default the highest+1 set id will be displayed. The user may select any existing set or type in the id of a new one. In the former case the new segments will be added (without duplication) to the set.

Three options are available for coating element faces with segments:

- **External faces** - For 3D elements, segments are created on topologically external faces of the selected elements. These would be faces that would be visible if no elements of this type were blanked.
- **Exposed faces** - For 3D elements, segments would be created on the exposed faces of the selected elements. These would be the elements that would be visible if all but the selected elements were blanked.
- **Visible faces** - For both 2D and 3D elements, segments are created on only those faces that are visible in the current view.

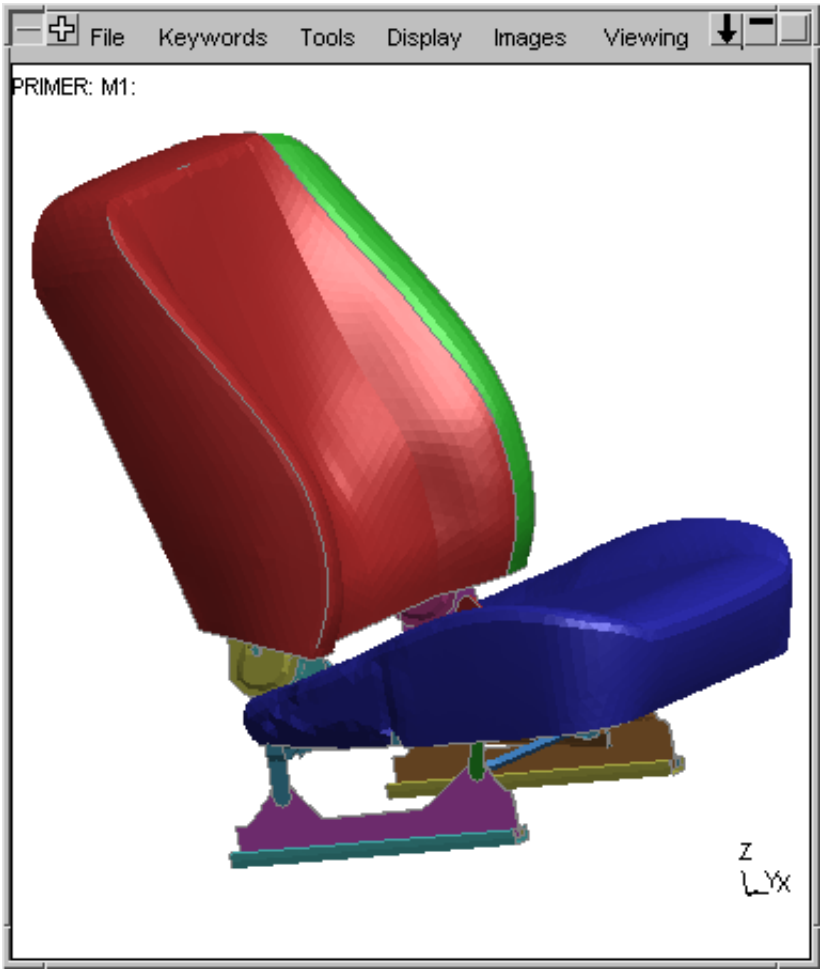
Once the parts have been defined **APPLY** creates shells/segments.

NOTE: Segment sets may also be created through **SET->SEGMENT->CREATE->COAT ELEMENTS** option.

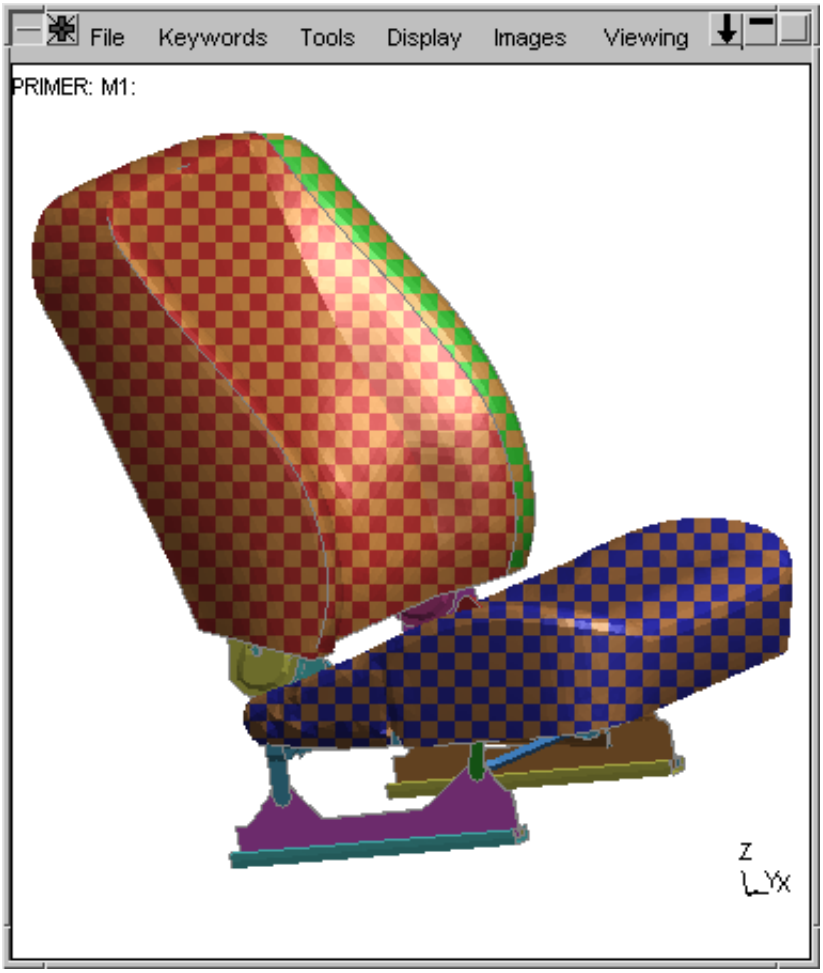
Multiple (coincident) coatings may be applied to a part by calling this function repeatedly - any existing shells attached to solid faces are ignored.

Coat Example:

The following example shows how Coat Part can be used to cover solid parts with shells



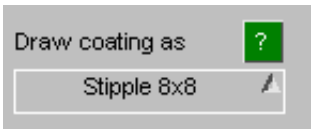
Initial model
In this first image the seat back has been modelled in solids, using two different solid parts (red and green), and the seat base is also solids using a third part (blue).



After coating with shells
Here all three parts have been coated with shells, the part id (1234 has been chosen at random) happens to have a default colour of orange, and this shows how the default "stippled" (checquer-board) rendering of the new part shows where it is applied.

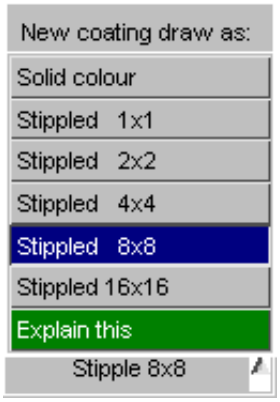
When you exit the coat part panel the rendering of the newly created parts will revert to conventional solid colouring.

Stippled (chequer board) drawing: **Draw coating as...**



Controlling the stippling (checquer-board) rendering of new parts / segments.

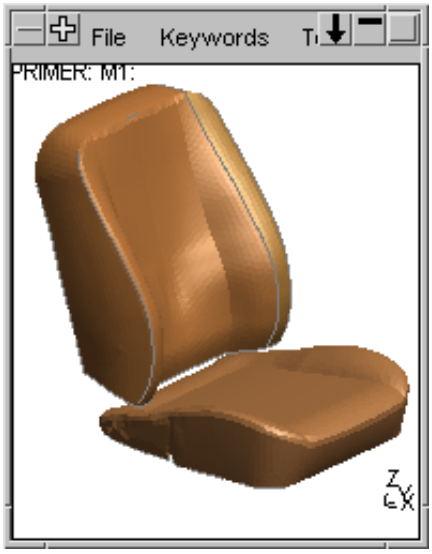
As the example on the right above shows newly created parts are drawn using stippling, which is equivalent to drawing the new part "through" a grid of squares in which alternate squares are opaque, so the underlying element can be "seen through" the new part. This is intended both to make new parts visually obvious, and to provide confirmation that you have coated the correct part.



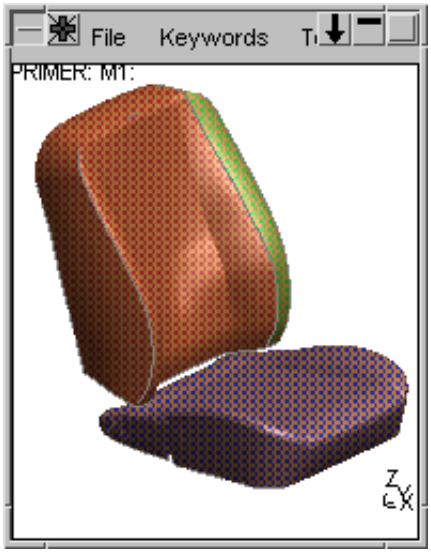
The size of the "chequer-board" pattern can be controlled using the popup on the "Draw coating as" button to be one of the options shown on the right. Examples of various settings are shown below.

The default setting is 8x8, giving the pattern shown on the right above. This default can be controlled by setting the preference

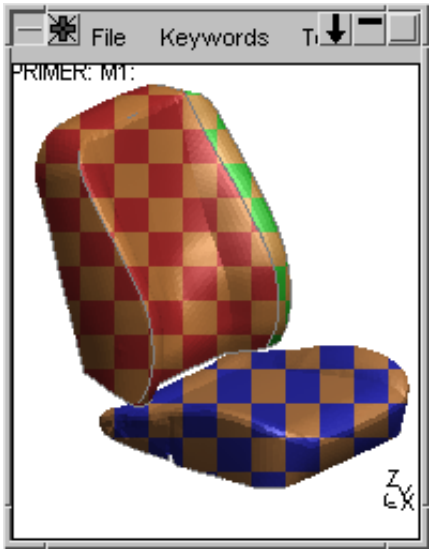
```
primer*coatpart_display:   solid  |  stipple_1  |
stipple_2  |  stipple_4  |  stipple_8  |  stipple_16
```



This image uses "Solid colour" and you can see that it covers the underlying elements completely



This image uses 2x2 stippling. The effect is one of "coarse transparency" in which the underlying colour shows through and mixes with the new colour.



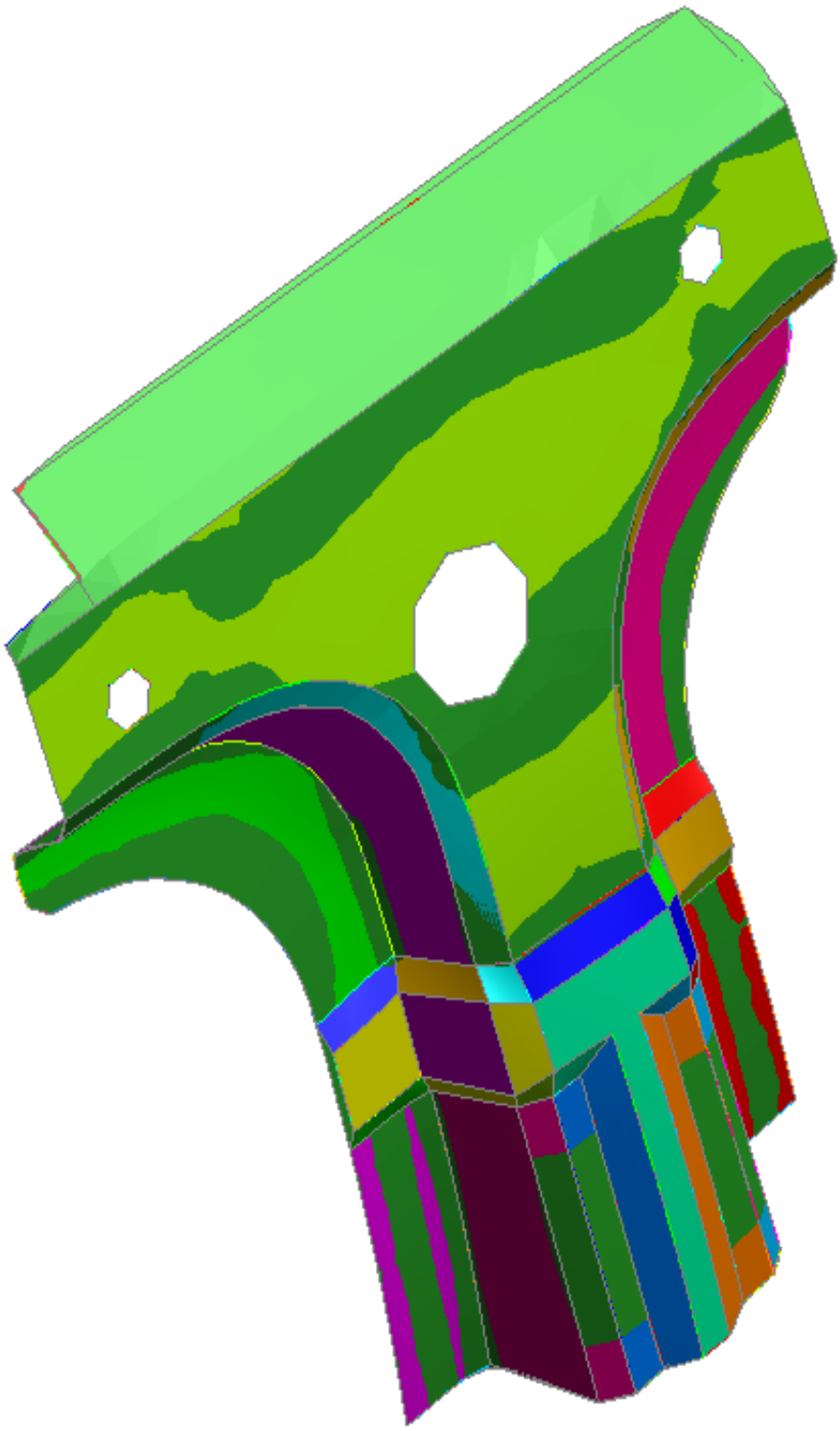
This image uses 16x16 stippling. To the author this is reminscient of "Coco the Clown": it certainly has the merit of being striking!

6.10 COMPARE: Comparing FE and CAD data

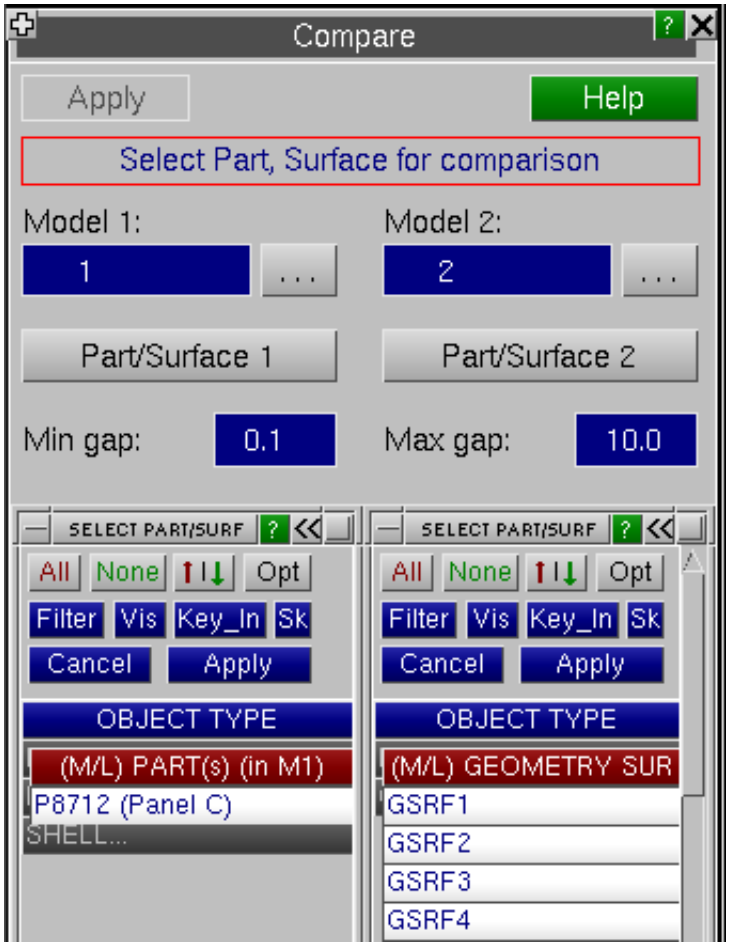
The COMPARE tool allows comparison of two sets of FE data or CAD geometry as defined by one or more shells, shell sets, shell parts or geometry surfaces.

Potential applications include comparison of an FE mesh against the corresponding CAD geometry. This function may also be used to compare two sets of parts.

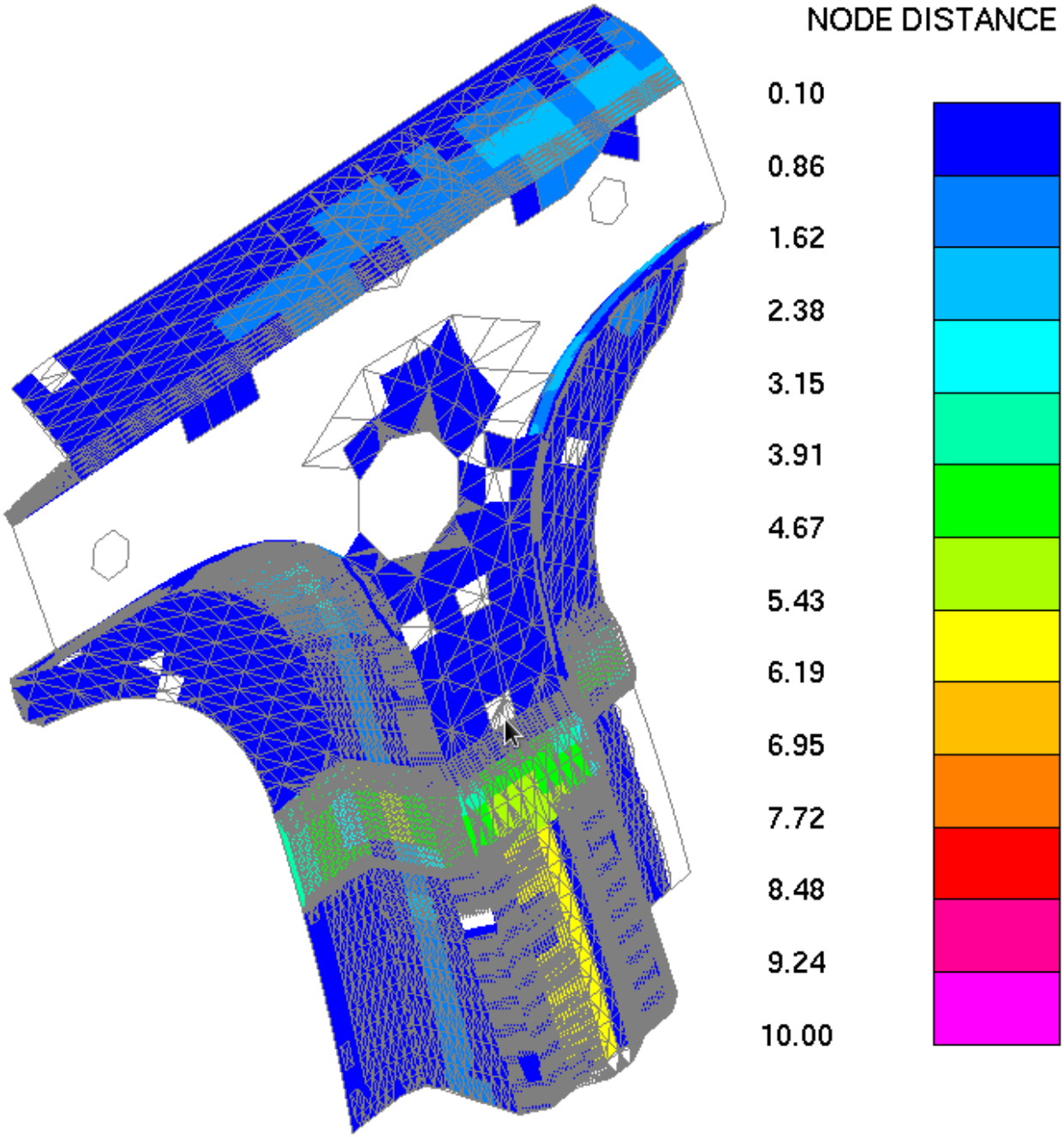
The selected entities may occupy different points in space and have different orientations, but they must have roughly the same shape for the comparison to work.



To start the comparison, two sets of data must be specified. Minimum and maximum search distances may also be controlled.



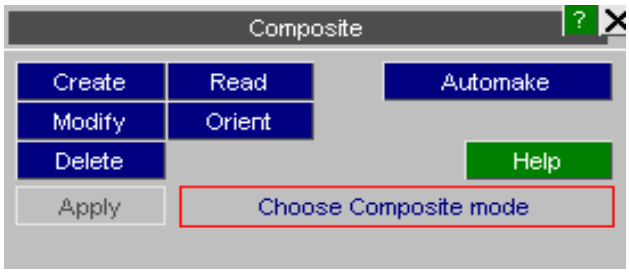
Results are displayed as contours. Various drawing modes are available as in the case of the contact penetration checking function



6.11 COMPOSITE

The **COMPOSITE** command is invoked from the **Tools** panel at the top of the screen.

The composite tool is a tool which able to manage (read, create, modify, delete) composite layup. When creating a composite layup from the composite tool, it will automatically transform the elements inserted into the layup in *ELEMENT_SHELL_COMPOSITE.

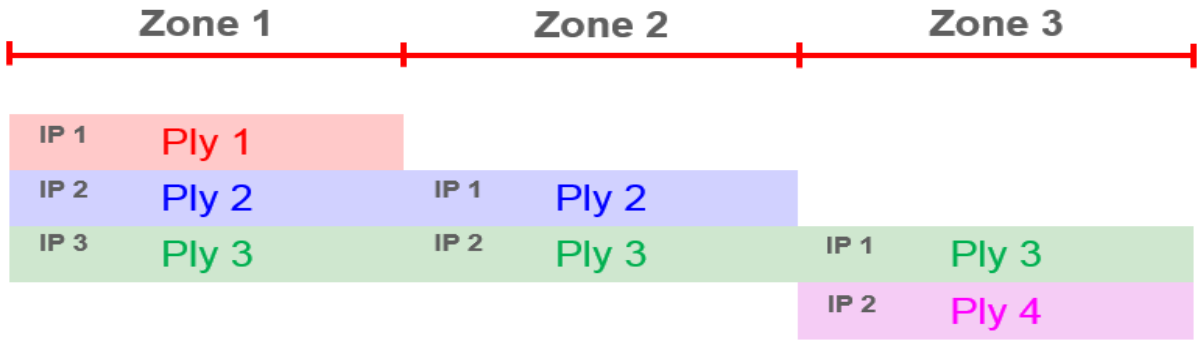


6.11.1 Purpose of the Composite Layup

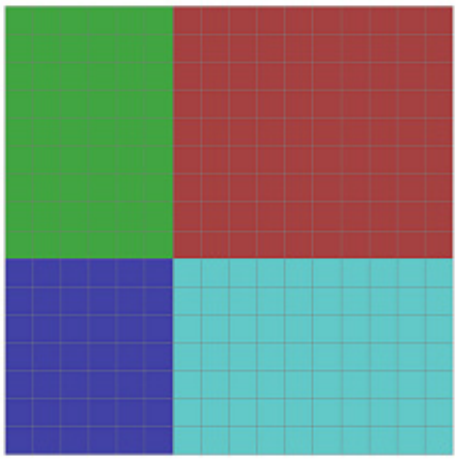
In a model of a composite component, different combinations of layers or plies may be present in different zones.

Each different combination of layers requires a different *PART_COMPOSITE or *ELEMENT_SHELL_COMPOSITE definition.

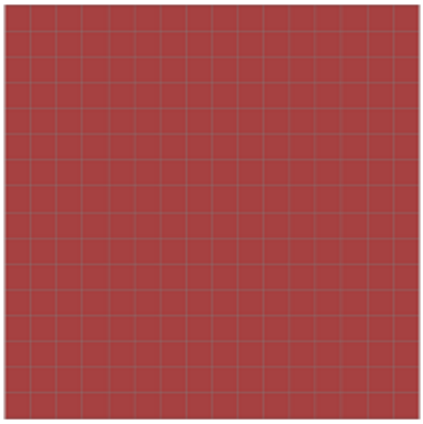
Information about the continuity of plies is not preserved.



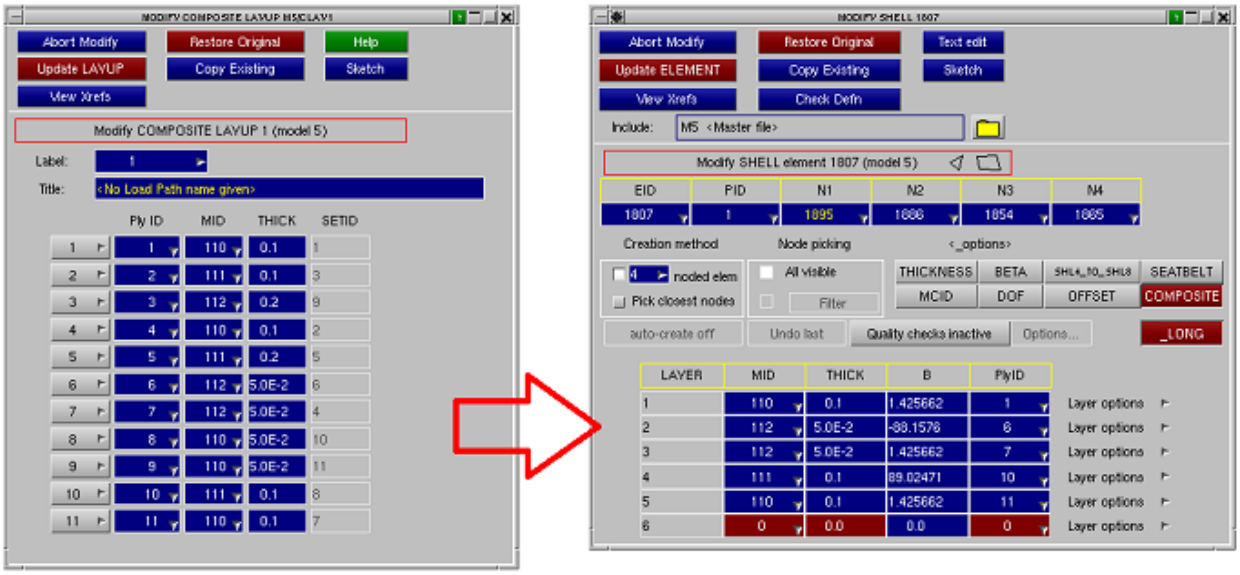
With *PART_COMPOSITE one Part has to be defined for each combination of layer as below.



Using *ELEMENT_SHELL_COMPOSITE, the combination of layers is directly contain into the element definition. Each physical componant only need one Part to be defined.



When Creating/Updating a Composite Layup, Primer will automatically update all the Elements which are part of the Layup. As in the example below, Element with EID 1807 is included in Ply 1, 6, 7, 10 and 11 of the Layup. After updating the Layup, elements are updated and the layers are added to there definition (see definition of *ELEMENT_SHELL_COMPOSITE).

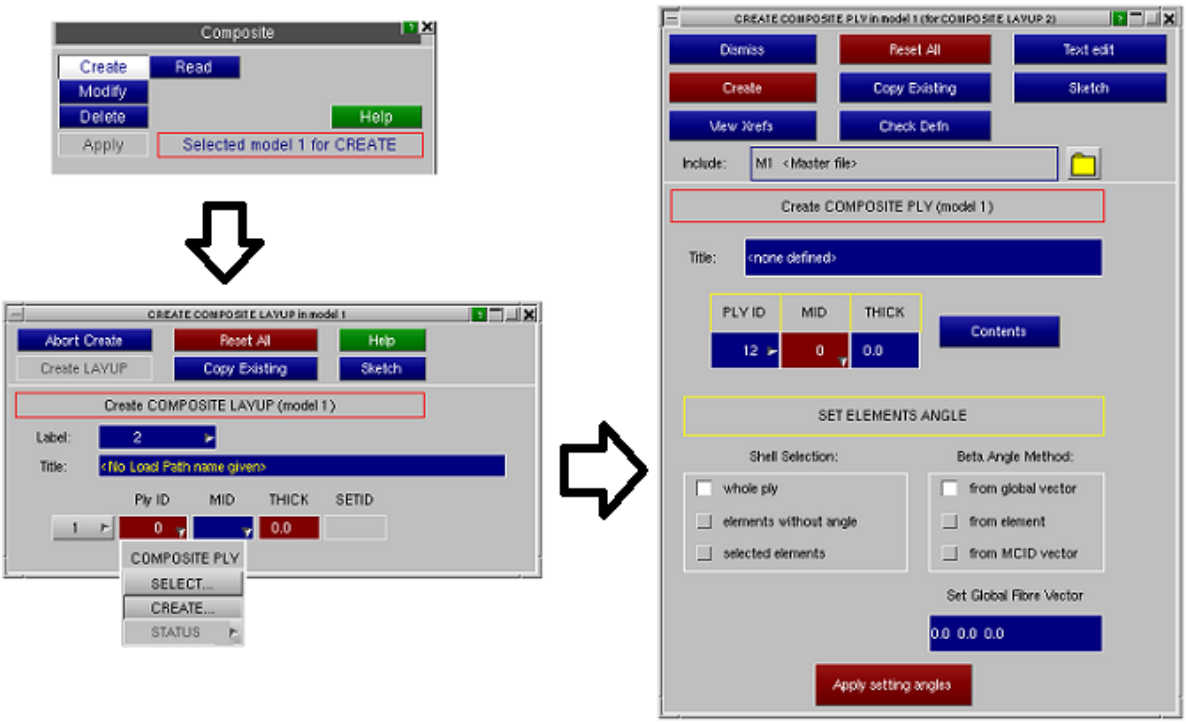


6.11.2 Create a Composite Layup

1. From the Composite tool, when pressing Create, a CREATE COMPOSITE LAYUP panel will appear.

2. On this panel it is possible to create from an existing mesh a composite Layup. Adding new plies to the Layup by pressing CREATE from the drop down panel under Ply ID.

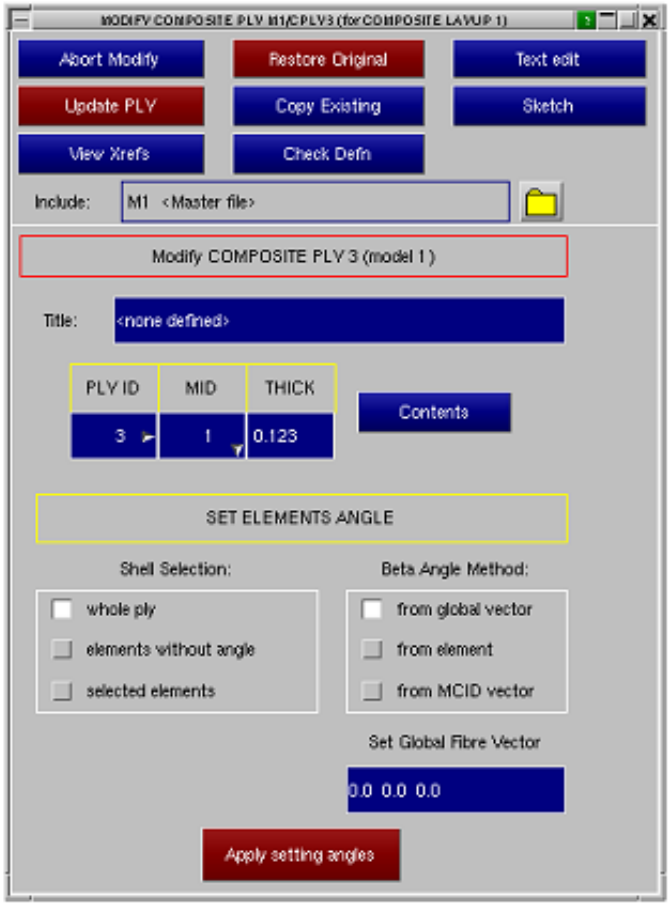
3. From this CREATE COMPOSITE PLY panel it is possible to assign a material (MID) and a thickness (THICK) to the ply which will be created. The Contents button allows management of the Set of Shells which are part of the Ply. Below is a tool to assign angles to the elements of the Ply.



6.11.3 Ply options

On the Ply panel it is possible to modify the material assigned to the ply (MID), the thickness (THICK) or the contents of the ply. The contents of the ply is defined by a set of shells, by pressing Contents on the COMPOSITE PLY edit panel, a classic set edit panel appears, user can from there add/remove elements from the definition of the ply.

Another option available from this edit panel is to set the Beta angle to the elements of the Ply, this option is at the bottom of the panel, the different methods are explained below.



To set an angle to the elements of a Ply there is different methods, three differents shell selection and three differents way to assign the angle.

Shell Selection:

- whole ply: the angle will be set to all elements of the ply

- elements without angle: the angle will be set to all elements of the ply which hadn't already got an angle assigned (example, elements newly added in a ply)

- selected elements: the angle will be set to every elements the user has selected

Beta Angle Method:

- from global vector: the fiber direction will be the projection, on the element, of the global vector typed in the textbox at the bottom of the panel.

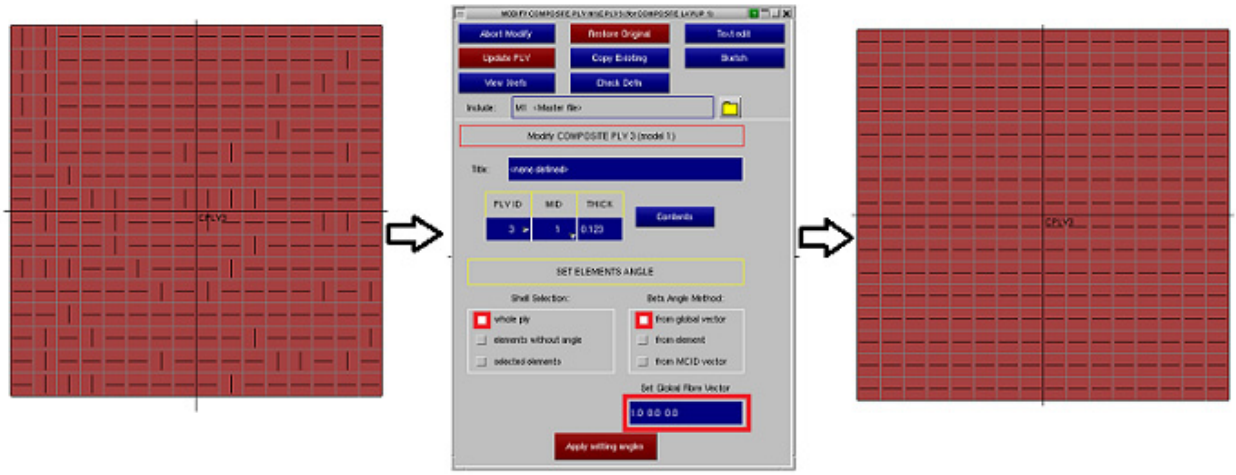
- from element: allows the user to pick an element and duplicate the fiber direction of this element to others.

- from MCID vector: the fiber direction will be the projection, on the element, of the vector, expressed in material local coordinates (available only from certain types of materials), typed in the textbox at the bottom of the panel.

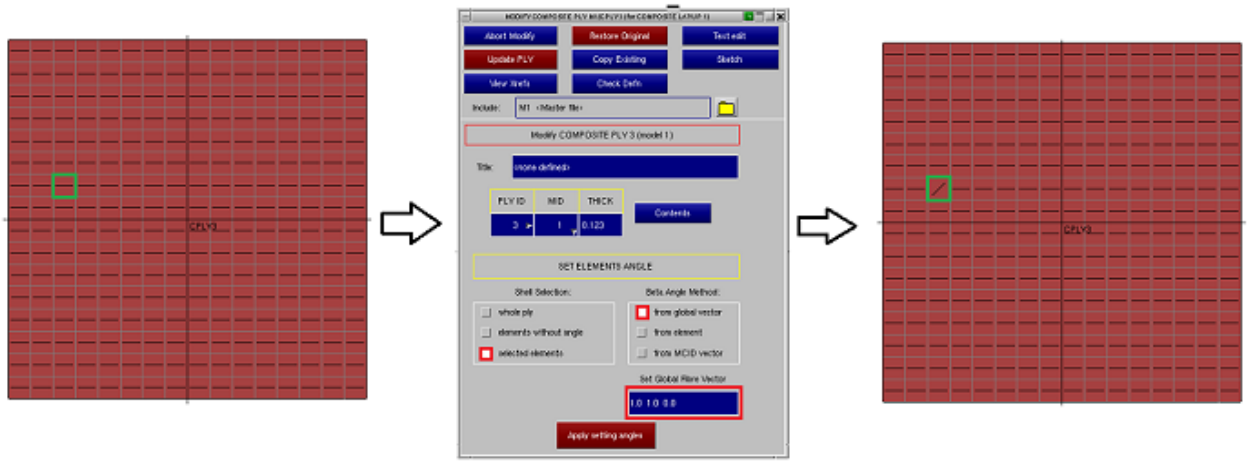
After selecting the method and the angle press the red button Apply setting angles so it will assign the new angle to the elements selected. To check if the result is as expected Sketch the ply.

Following are some examples:

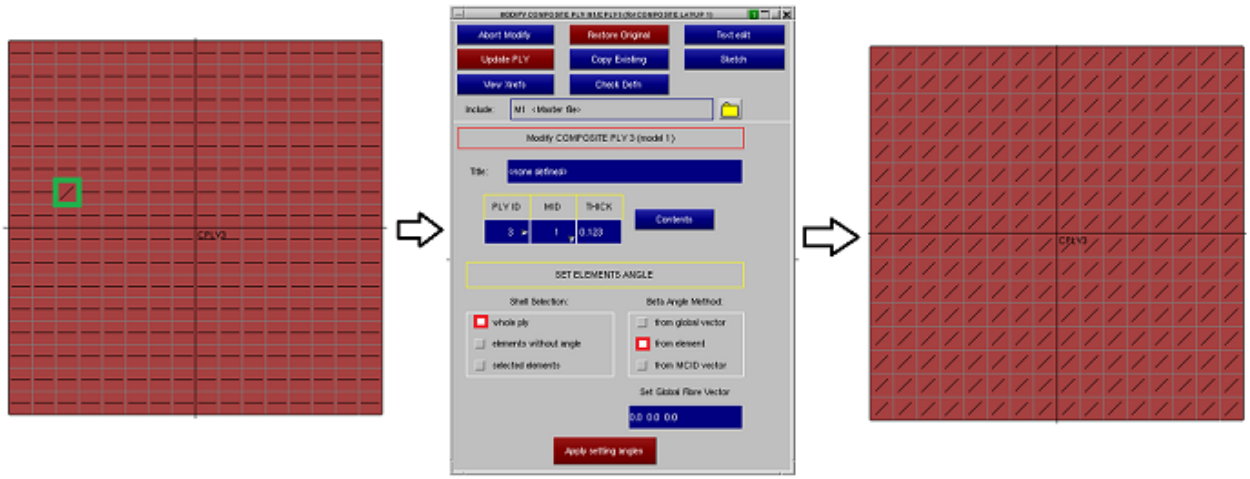
Below, first image is a sketching of the ply and fiber direction before setting the angle. In the second user is setting angle to the whole ply from global X vector. The last image is after setting the angle.



Below, first image is a sketching of the ply and fiber direction before setting the angle. In the second user is setting angle to the element in green from global XY vector. The last image is after setting the angle.



Below, first image is a sketching of the ply and fiber direction before setting the angle. In the second user is setting angle to the whole ply from the elements in green angle. The last image is after setting the angle.



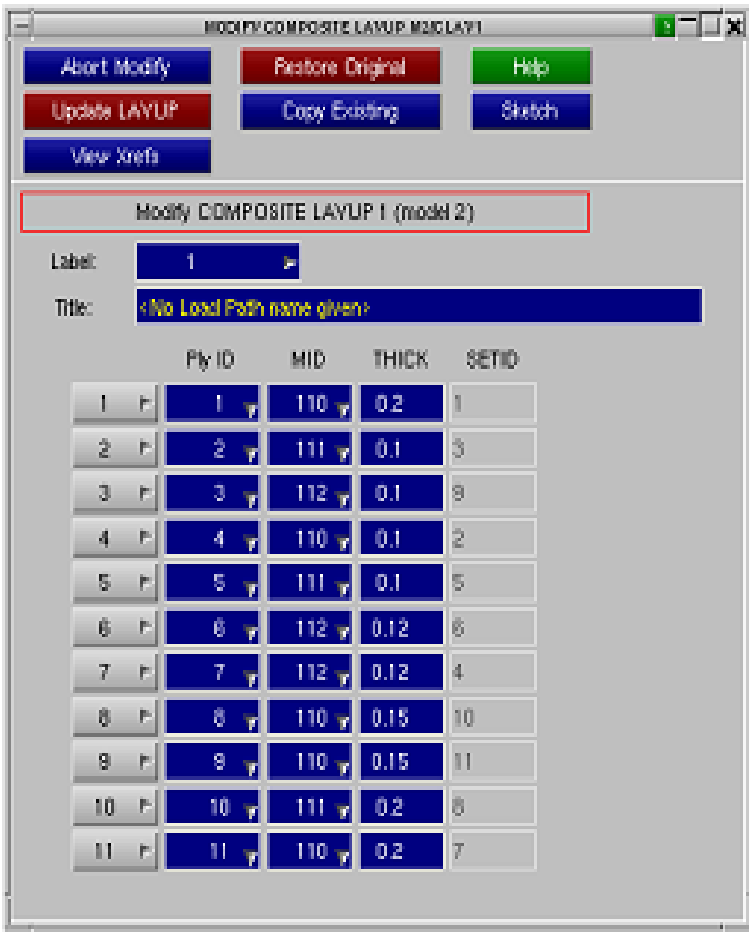
6.11.4 Layup options

The edit panel for COMPOSITE LAYUP display the plies of the layup, the material, thickness and setid assigned respictivelly for each ply of the layup.

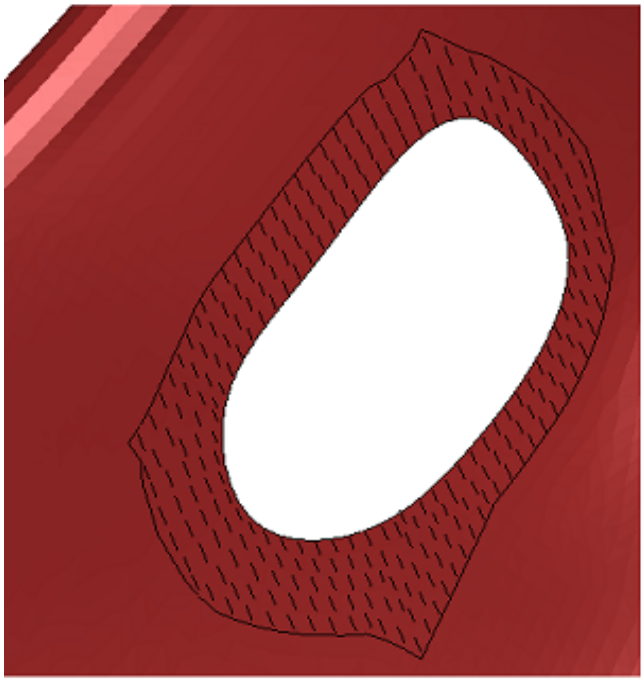
It is possible to modify the material or thickness of a ply directly from the layup panel.

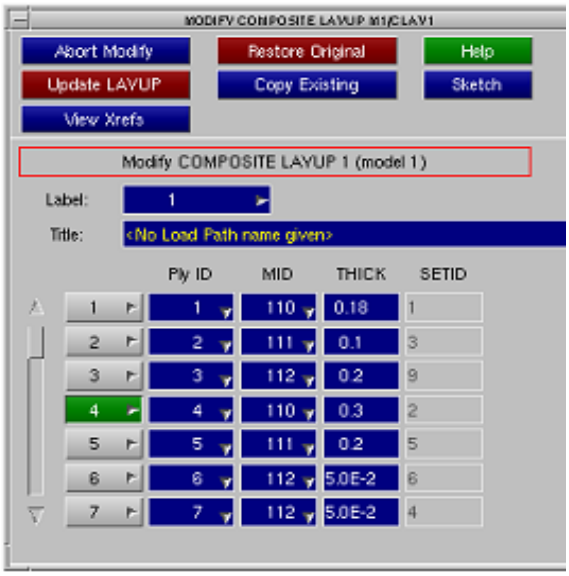
It is also possible to change the thickness and materials of multiple plys at the same time by selecting mupltiple rows of the layup panel by using the ctrl and shift buttons. After the selection is made, any material or thickness which is chosen for any one of the plys in the selection, is applied to all plys in the selection.

Similar selections can also be done from the PART_COMPOSITE and ELEMENT_SHELL_COMPOSITE edit panels as well.



To sketch a ply easily from the layup panel, press the grey button on the left hand side of the panel. The button will turn green (as in the right image), the ply with the fiber orientation of each elements composing this ply will be sketch (as in the left image).





If right clicking on the button on the left hand side, few options for the ply will appear.

- ins before/after: insert a new line on the layup before/after the line selected

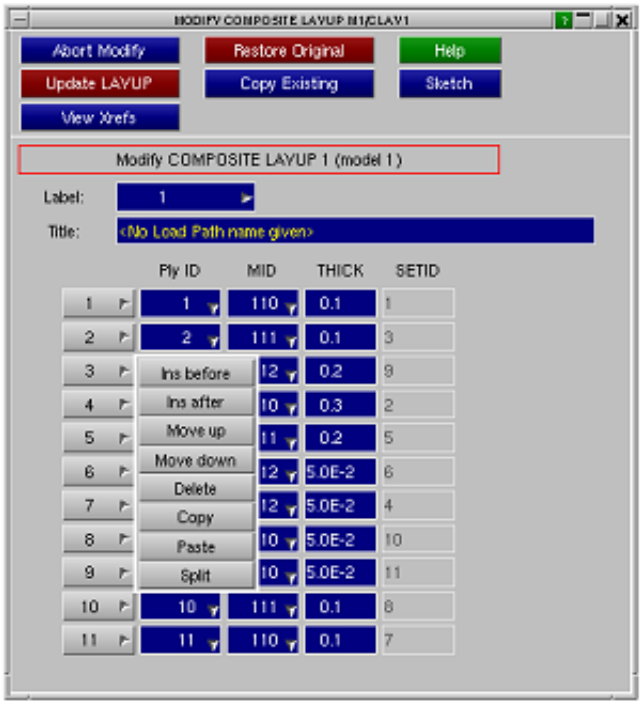
- move up/down: will swap position between the ply selected and the one before/after

- delete: delete ply from the layup definition

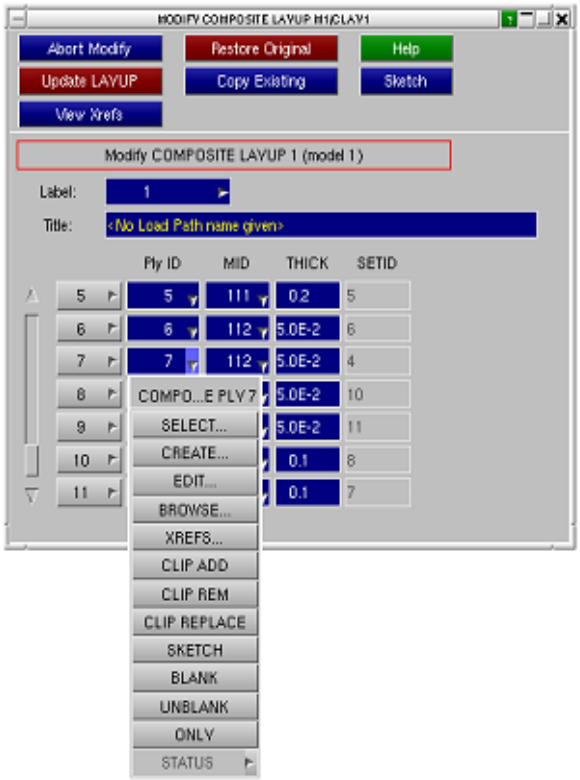
- copy: copy informations of this ply

- paste: create a new ply with the informations copied previously

- split: split the ply in two plies with half of the thickness

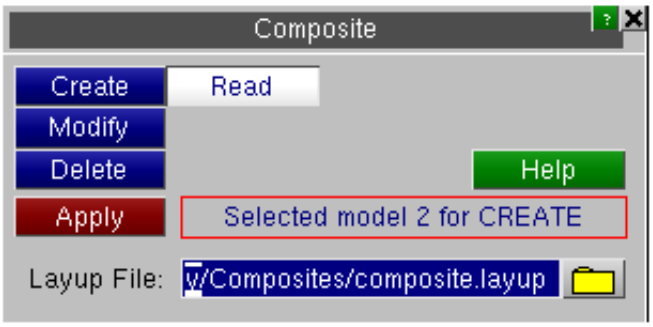


A drop down menu is available for every ply of the layup, with the usual options available as for any component within Primer.

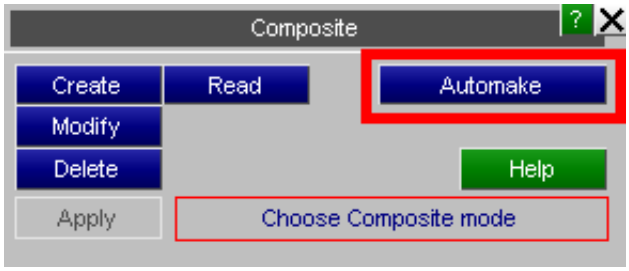


6.11.5 Read a .layup file

The .layup file format is a format of data storage for composite layup. This format can contain a mesh and all the informations needed to describe a composite layup (number of plies, materials, thickness,...). This format is readable from the Composite tool.



6.11.6 **Automake** : creating plies and layups from _COMPOSITE data in the model.



Sometimes a model wll contain one or more of the following keywords:

***PART_COMPOSITE**	Part defines multiple layers, with thickness, material and beta angle at each layer
***PART_COMPOSITE_LONG**	As ***PART_COMPOSITE**, but also including a PLY number at each layer
***ELEMENT_SHELL_COMPOSITE**	Shell element defines multiple layers, with thickness, material and beta angle at each layer
***ELEMENT_SHELL_COMPOSITE_LONG**	As ***ELEMENT_SHELL_COMPOSITE**, but also including a PLY number at each layer
***ELEMENT_TSHELL_COMPOSITE**	Thick shell element defines multiple layers, with thickness, material and beta angle at each layer

However the model may not contain ***COMPOSITE** keywords after ***END** meaning that plies and layups will not be generated.

Automake attempts to solve this problem by generating plies and layups automatically from the cards above. This can be particularly useful if plies and layups are written to the ZTF file since they become available in Oasys D3PLOT. This means that the beta angle of each layer of each shell is known during post-processing making it possible to view results in local material axes.

Automake works as follows:

1. Any ***ELEMENT_xxx_COMPOSITE_LONG** keywords which define explicit ply numbers will have resulted in latent ply definitions being created inside PRIMER.

   These plies are populated with the material and thickness of the first element that references, and converted from latent to installed so that they become "visible" in PRIMER.
   A layup is created for each part (pid field on the element card) that these elements reference, and the plies are added to these layups.

2. Any ***PART_COMPOSITE_LONG** definitions which define explicit ply numbers will have resulted in latent ply definitions being created.

   These plies are populated with the material and thickness of the relevant layer on the part, and any elements referring to the part which are not themselves _COMPOSITE are added to the plies.
   A layup is created for each such part, and the plies referred to in the part are added to this layup.

3. ***PART_COMPOSITE** definitions are processed next.

   This (non _LONG) card format does not permit ply ids to be specified, so a new ply will be created for each layer of the part using that layer's material id and thickness.
   All elements that reference the part will be added unless the element is itself _COMPOSITE
   A layup is created for each part, and all plies in the part are added to that layup.

4. ***ELEMENT_xxx_COMPOSITE** definitions are processed last.

   These present a problem since it is difficult to identify logical plies from disparate element definitions. Therefore the following logic is adopted:

   - A ply is created for each [Part | Integration point | Material id | thickness] combination encountered on an element.
   - A layup is created for each referenced material if it doesn't already exist.
   - A ply is added to the layup of any material referred to by an element in the ply.

   So for example two elements which are in the same part, and which have the same material and thickness at the same integration point will be placed in the same ply.

This process tends to result in more plies being generated than there are"true" continuous layers of material in the model, and there can be some illogical overlap between plies in layups. It is no substitute for having proper ply definitions supplied via a layup file or manual editing, but it is better than nothing.

Automake and ZTF file creation.

The main purpose of Automake is to make it possible to post-process composite shell data in material local axes. Users

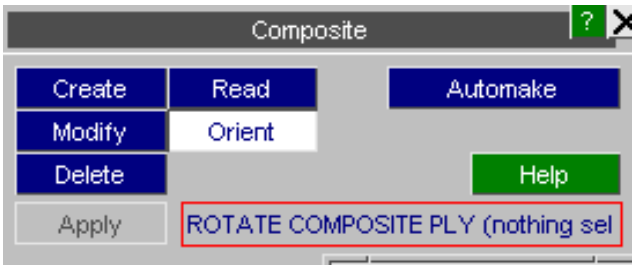
of Oasys D3PLOT who read a ZTF file containing ply information will be able to do this, so prior to writing a ZTF file a check is made for composite parts and elements that have not been put into plies. If any are found you are given the option of running Automake prior to generating the ZTF file so that the data gets embedded (this option can be pre-selected using the preference 'ztf_output_composite_action'). See section 3.8.2 for more information about writing ZTF files.

Automake and saving Layup and Ply data in the keyword output file.

Using **Automake** will result in Plies and Layups being created and installed in your model. If you write the model out to a keyword file this information will be remembered via special PRIMER-specific *COMPOSITE cards after *END, and recreated when the model is reread.

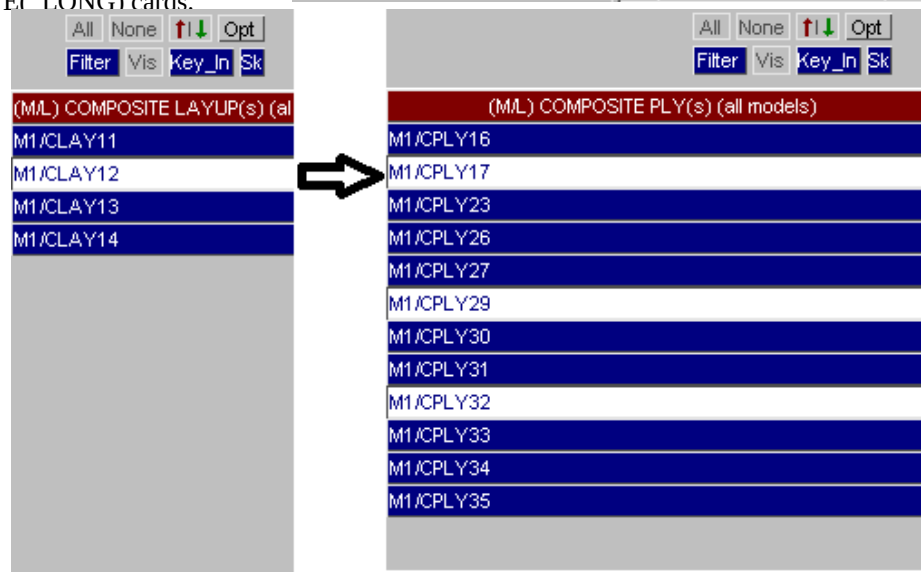
Since the plies and layups have been synthesised and will almost certainly not be a true representation of the physical construction of your model it is recommended that you don't save them. Either don't write out a keyword file, or delete the plies and layups prior to doing so.

6.11.7 Orient: An easy way to set composite beta angles across plys in a layup.



The "Orient" panel offers several techniques with which users can set Beta angles on composites. Please note that all the options on this panel at the moment only apply to ELEMENT_SHELL_COMPOSITE(_LONG) cards and if a ply contains other dyna cards, these plys are automatically filtered out and are not shown in the object menu.

Please note that all operations done on this panel are binding, i.e they modify the actual values on the ELEMENT_SHELL_COMPOSITE(_LONG) cards.



All Operations in this tab apply to a selection of plys in a layup. The user is first presented with an object menu from which to select any layup in the model. Next This opens up an object menu will all the plys in the selected layup. The user can then select the required plys and the apply any of the orient operations to these selected plys.

At the moment there are four options which the user can use to set composite angles, these are:

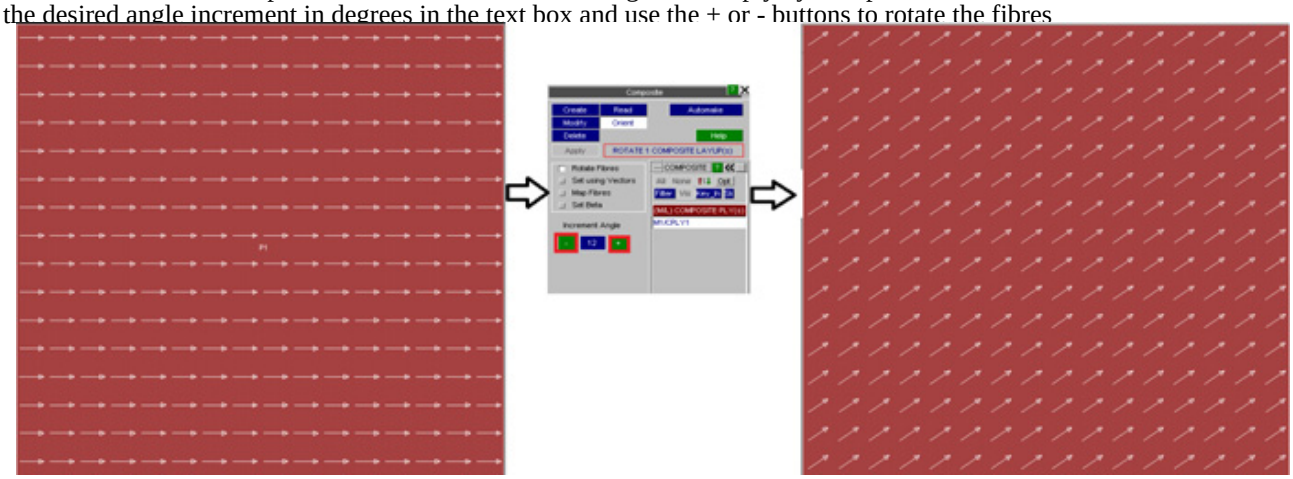
1. Rotate Fibres

2. Set using Vectors

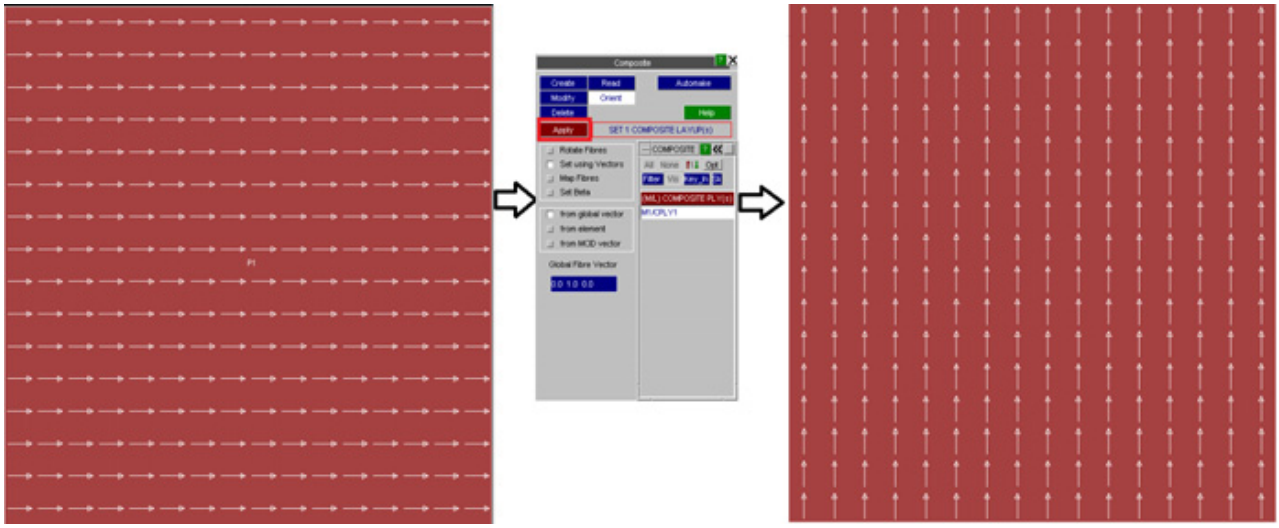
3. Map Fibres

4. Set Beta

1. Rotate Fibres: This option allows users to rotate the angles in the ply by the specified amount. The user has to enter the desired angle increment in degrees in the text box and use the + or - buttons to rotate the fibres

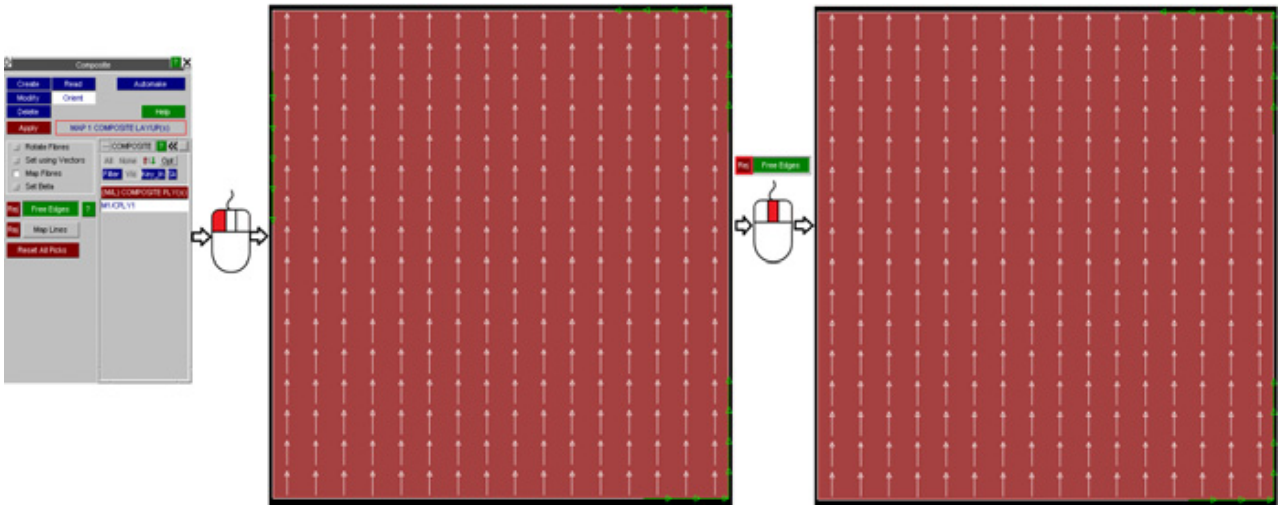


2. Set Using Vectors: This option basically consists of options which are present in the ply panel, but form here, they can be applied to a selection of plys rather than just one. For more details on these options, please refer to the documentation on the ply panel.



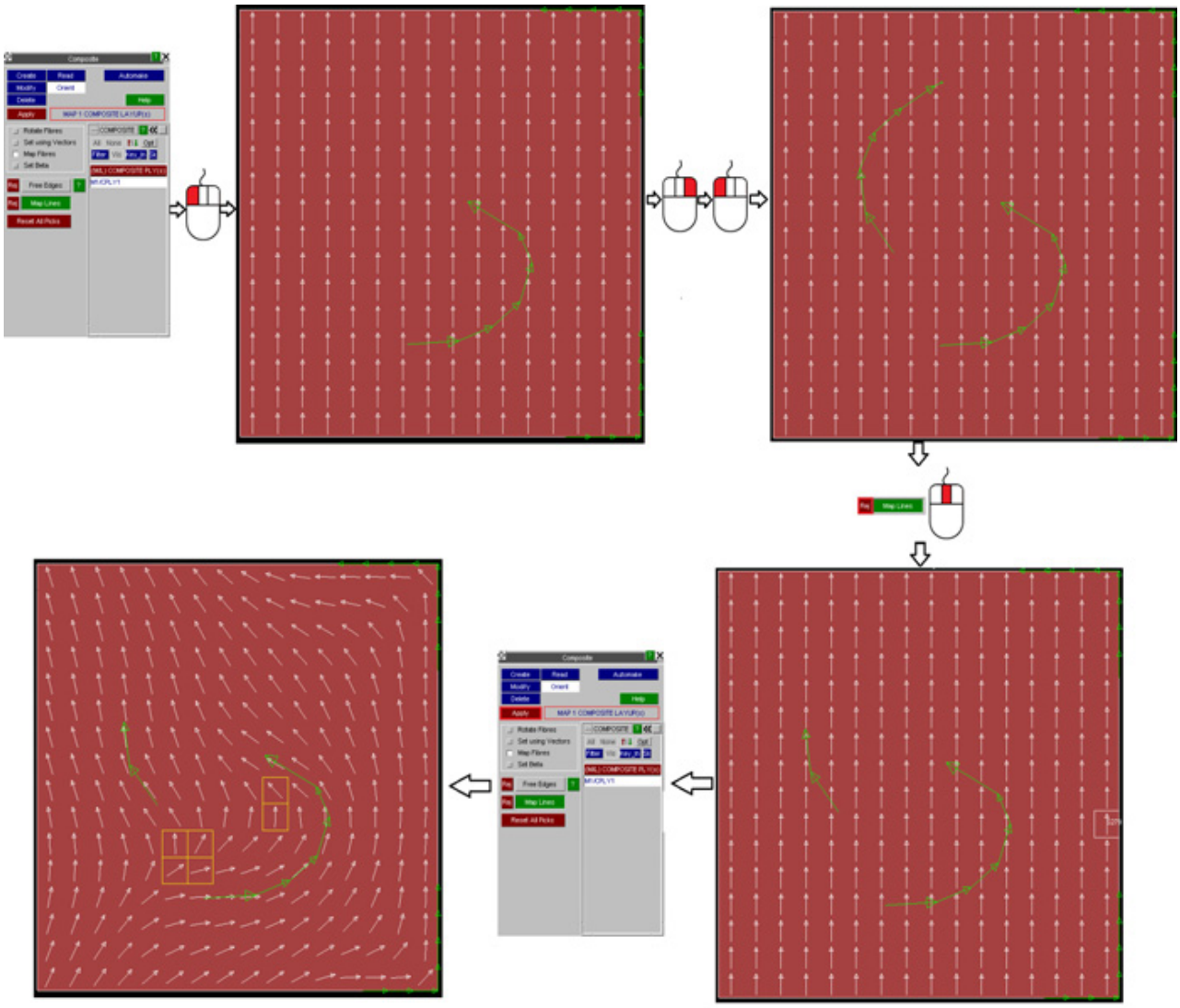
3. Map Fibres: This function can be used to interpolate fibres along map lines which can be specified by the user. The user has a choice of selecting free edges or map lines. The interpolation takes place based on the inverse (distace)^(2 x mapping parameter). The value of the mapping parameter is set to 1.5 by default but this can be changed by the user in Options -> Program Options -> Composites -> Mapping Parameter.

a) Free Edges: Using the free edges option, the user can select free edges along the model by using the left mouse button. The user can select discrete sections of the model and any selection can be undone by using either the middle mouse button or the "Rej" button on the panel.

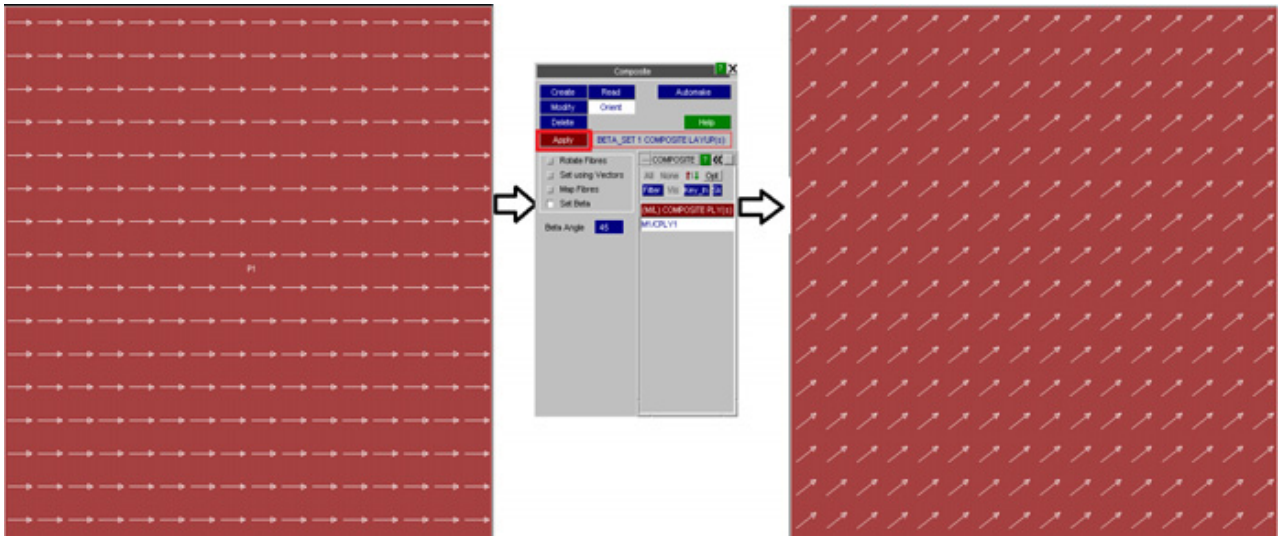


b) Map Lines: The user can also select lines along the surface of the model using the Map lines option in conjuction with any free edges.

When you are satisfied with the selection of free edges and map lines. Click on 'Apply' which will interpolate the composite beta angles based on your selection of lines. The "Reset All Picks" button will be used to reject all free egdes and map lines and should be used only when the user wants to start the mapping process again from the beginning.



4. Set Beta: This option can be used to directly set the value of beta angles on the entire ply.



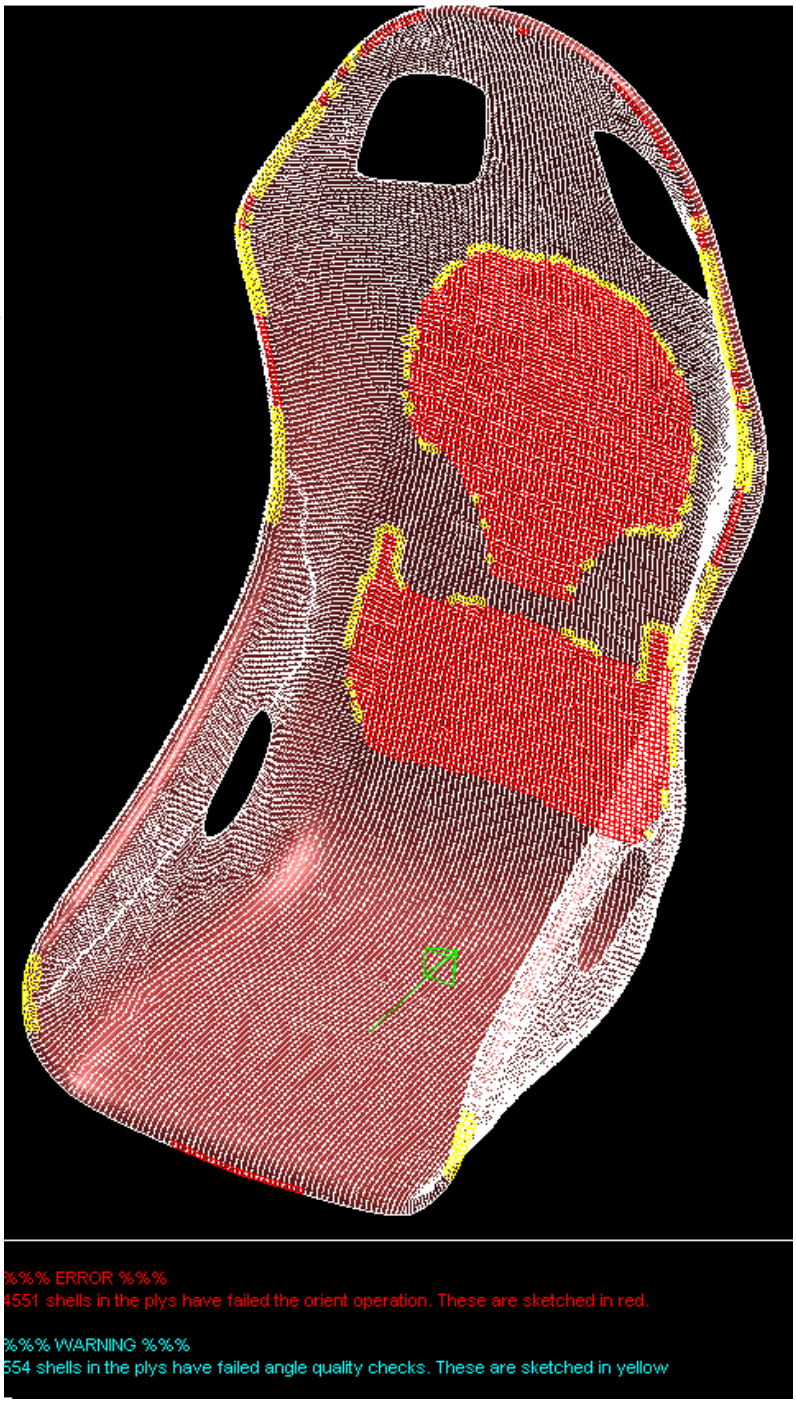
Composites Angle Quality Feedback:

While setting composite ply angles through the orient panel, PRIMER warns the users if some angles have not been set or if some beta angles change direction too drastically compared to their neighbours.

In some cases it is possible that the input map line or vector is too close to the shell normal and it is not possible for PRIMER to correctly compute the beta angles, in all these cases, the shell's beta angles remain unchanged and the offending shells are sketched in red.

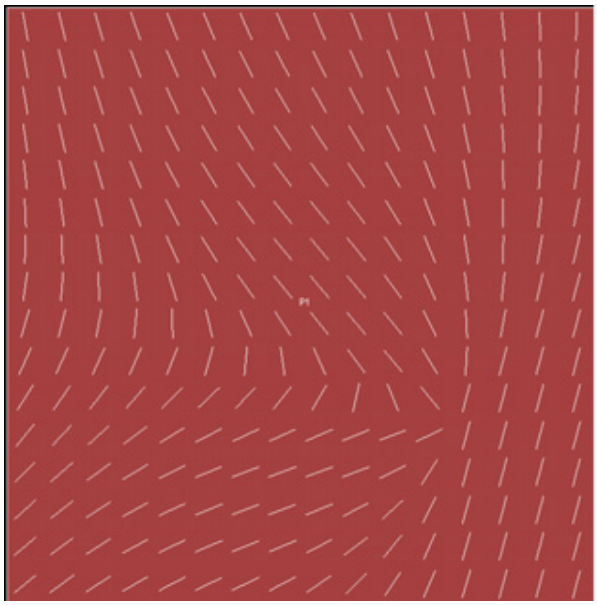
PRIMER also checks for shells which change angles too drastically compared to their neighbours and sketches these shells in yellow. The default angle for this check is set to 45 degrees, but this value can be changed by the user in Options -> Program Options -> Composites -> Shell quality angle.

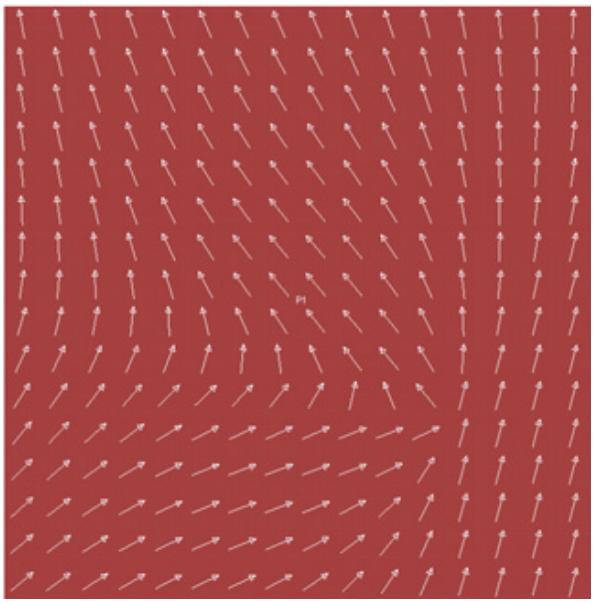
The number of offending shells in either case is also listed in the message box in the bottom left corner of PRIMER.



Composites Sketching options:

Users can now sketch composite ply angles using the previously available lines or alternatively they can now sketch them using arrows. This option can be set by the user in Options -> Program Options -> Composites -> Sketch Method.

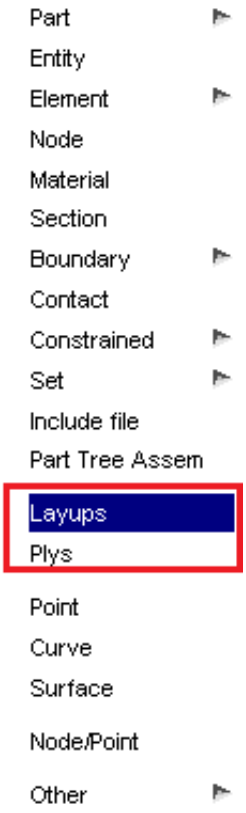




6.11.8 Composites Graphics Options

It is now possible to select and edit composite plys and layups using the quick pick menu . The current options available for both plys and layups are: Blank, Unblank, Only, Information, Delete, Edit, Sketch, Find, Xrefs and Set current layer.

Please note that deleting plys and layups only deletes PRIMER's internal *COMPOSITE_LAYUP and *COMPOSITE_PLY cards does not remove the corresponding data on the part/shell. This has to be done from their corresponding edit/keyword panels.



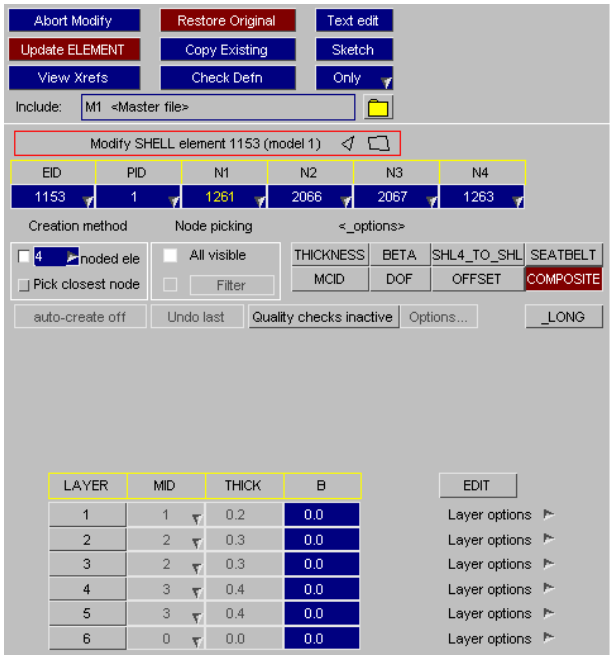
6.11.9 Composites Manual Editing

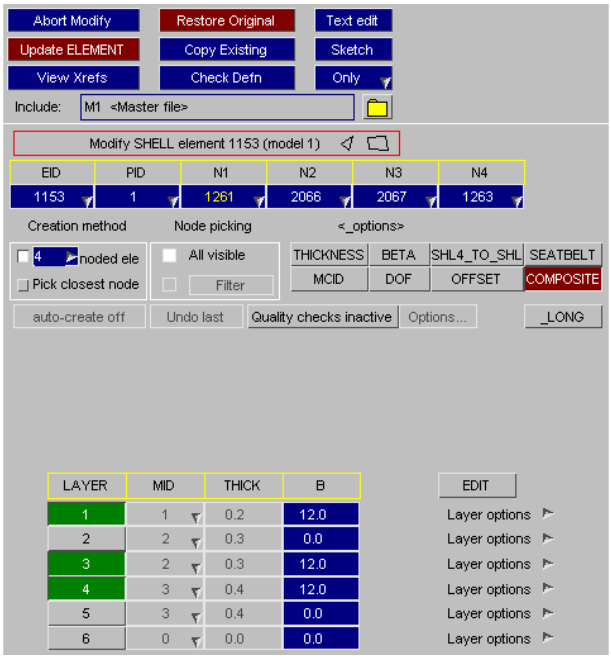
It is possible to edit composite values directly from the PART/ELEMENT edit panels. Users are discouraged from editing compopsite materials and thicknesses from these panels for ELEMENTs because these values are common across a ply and editing these values on the ELEMENT may cause PRIMERs internal ply data to do out of sync. It is however safe to modify the beta angles as these are element specific and are not specified on the ply itself.

Keeping this in mind, PRIMER now automatically greys out these fields if it determines that the edit panel in question belongs to PRIMER's internal ply/layup cards. The user can choose to override this functionality by clicking on the "EDIT" tab.

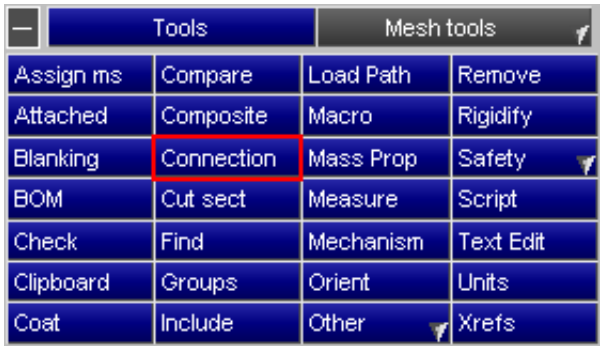
Users can now edit multiple ply values at the same time by using the "ctrl" and "shift" buttons and clicking on the layer buttons. Now any value set on any of the selected plys is automatically copied over to all selected layers.

This functionality is available on the PART, (T)SHELL and COMPOSITE_LAYUP panels.





6.12 CONNECTIONS



A connection is a new PRIMER entity introduced in version 9.3. It allows PRIMER to create/modify/delete **mesh independent** spotwelds, bolt connections and adhesive runs. Spotwelds consist of beams or hexahedral elements tied to the panels using a tied contact. Bolts are rigid connections between panels. Adhesives consist of runs of hexahedral elements tied to the panels using a tied contact.

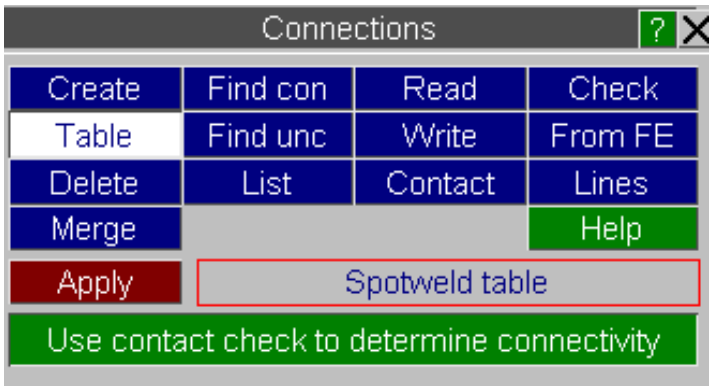
The **Connection** panel is used for all aspects of managing connection data.

The connection entity allows PRIMER to store all of the information that makes up the appropriate connection entity. That means that for example, it is possible at any time to change a beam spotweld into a solid spotweld or a bolt. As PRIMER knows what entities make up the connection it can delete the old entities and make new ones as required. The connection can be drawn (a 'blob' is drawn at the connection point, or a line indicating the path of an adhesive run) or labelled using the entities panel. The colour of the connection is drawn in depends on the state of the connection. The following colours and their meanings are used.

Colour	Meaning
Green	Realized. The connection is made and it does not have any errors
Blue	Provisionally realized, no contact check has been done
Red	Bad. The connection cannot be made because there is a problem
Orange	Invalid. The connection has been made but there is something wrong with it (e.g. the node is not tied correctly)
Yellow	Not checked. The connection has been made but PRIMER has not yet checked it to see if it is OK or not
Cyan	Latent. The connection point exists but it has not been made yet

The panel allows you to create, review, modify and delete connections. A 'connection file' can also be read by PRIMER to connect an entire structure very easily. Additionally, tools are available for checking and correcting bad connections as well as finding connected or unconnected panels. The initial spotweld panel is shown below. If your model does not contain any mesh independent spotwelds only the **Create**, **Read** and **From FE** options will be available.

The following options are available from the **Connection** panel.

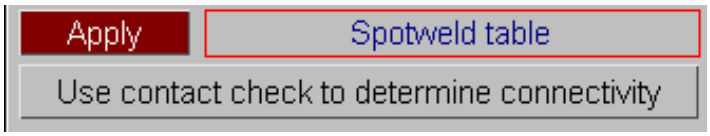


Creating spotweld connections or bolt connections
Modifying/Reviewing an existing connection using the TABLE
Deleting spotwelds
Merging connections

Using find connected to find connected panels
Using Find unconnected to find unconnected panels
Listing connection data
Reading a connection file (PRIMER spotweld file format)
Writing a connection file (PRIMER spotweld file format)
Checking the spotweld contact
Checking spotwelds
Creating connection from existing FE entities
Modifying connections by creating lines

There are several options that control how connections in PRIMER work.

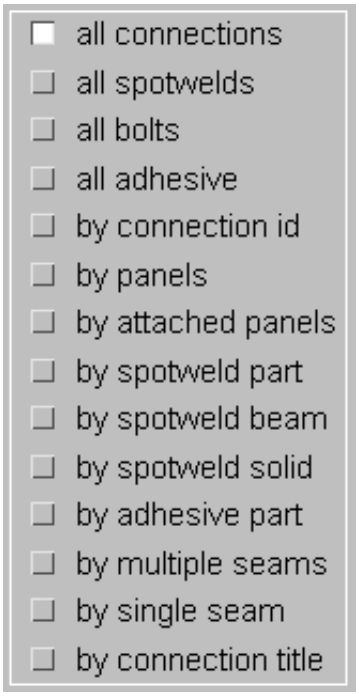
To achieve realized status a connection must be checked using the contact checker. Normally this is done automatically before the table is displayed. For very large models with multiple contact definitions this may take while, so the user may elect to postpone the connectivity check. In this case a simple geometric check is made, there is no guarantee that the weld will tie or even be present in a tied contact! Hence the connections will be displayed as blue - provisionally realized.



6.12.0 Methods of selecting connections

Several of the connections functions (e.g. **Table**, **Delete**, **List** etc.) allow you to select which connections you want to work on by several different methods. At the top of the panel you can select which connection types you wish the different methods to apply to.

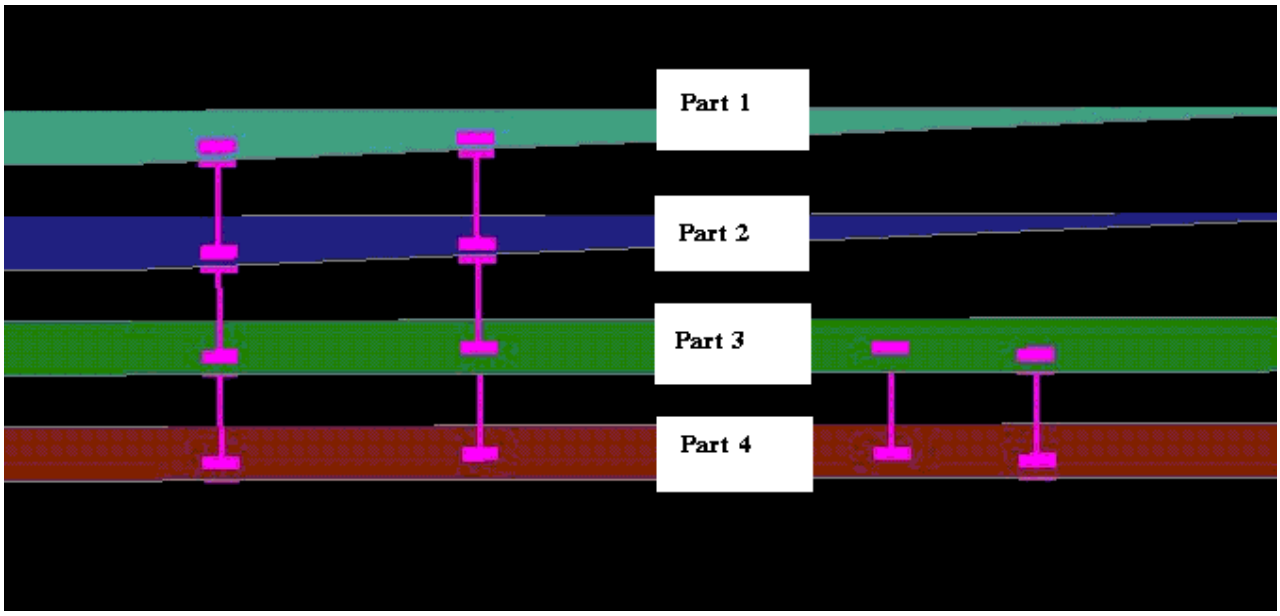
- **all connections**. All the connections in the model are selected.
- **by connection id**. You can select which connections to modify by picking or using the object menus.
- **by panels**. Connections that use any of the selected parts in their layer definitions. Note that the connection does not have to be made for this.
- **by attached panels.** Any connections that are attached to any of the panels you select. Note this implies that the connection is 'realized'
- **by spotweld part**. Any spotweld beams or solids using the specified part(s).
- **by spotweld beam**. Choose connections by spotweld beam.
- **by spotweld solid**. Choose connections by spotweld solid.
- **by adhesive part**. Any adhesive runs using the specified part(s).
- **by multiple seams**. Any connections that only use some (or all) of the selected parts (see Multiple or single seam selection for a more detailed description).
- **by single seam**. Connections that use all of the selected parts (see Multiple or single seam selection for a more detailed description).
- **by connection title**. A box opens up to enter a title search string.



Multiple or single seam selection

When selecting by **multiple seam** the connections and/or their related entities that are attached to **ANY** of the selected parts **AND NOT** attached to **ANY** deselected parts will be selected. For example, in the figure below, if part 3 and part 4 are selected then the two beams on the right will be chosen. If parts 1, 2, 3 and 4 are selected then all 4 beams will be chosen.

When selecting by **single seam** the connections and/or their related entities that are attached to **ALL** of the selected parts **AND NOT** attached to **ANY** deselected parts will be selected. This will only ever be one seam. For example, in the figure below, if part 3 and part 4 are selected then the two beams on the right will be chosen. If parts 1, 2, 3 and 4 are selected then ONLY the two beams on the left will be chosen.


Part 1
Part 2
Part 3
Part 4

6.12.1 Creating connections

Automatic creation of connections from welds

Management of spotwelds by connection entities is fundamental to PRIMER - weld creation, deletion of welded shells, weld checking, find attached, etc. All welds created in PRIMER will have a corresponding connection, maintained as post-end keyword.

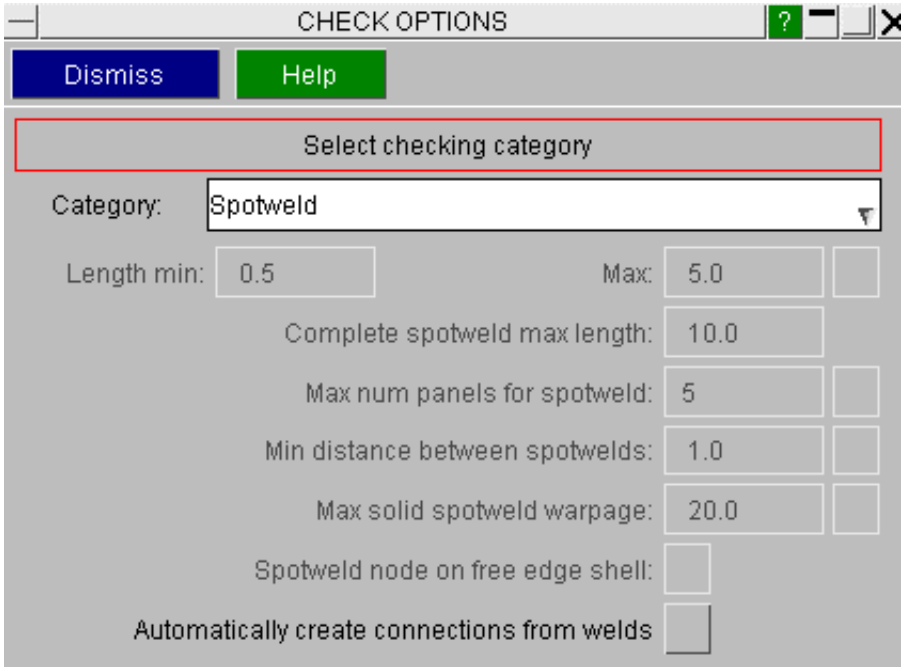
As read models may, however, contain welds which do not have connections. By default, PRIMER will attempt to create connections from any existing MAT100 welds (beams, single solid or solid 'nuggets' [with define hex spotweld assemblies]) which do not already have them

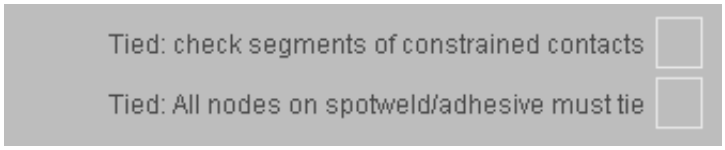
- when the connections tool is activated
- when a model check is done
- when shells or shell parts are being deleted

PRIMER will warn in the dialogue box when a model check or deletion operation has created connections. These connections are marked and will be ignored when the model modified function is applied.

```
%%% WARNING %%%
Primer *CONNECTION entities have been created.
These enable checking and management of spotwelds in the model.
Connection entities will be written as post *END data.
```

Whilst this methodology is recommended, it is possible for the user to inhibit the automatic creation by the setting under **CHECK > OPTIONS > SPOTWELD**. This will also inhibit the checks which rely on connection logic **for all spotwelds**.

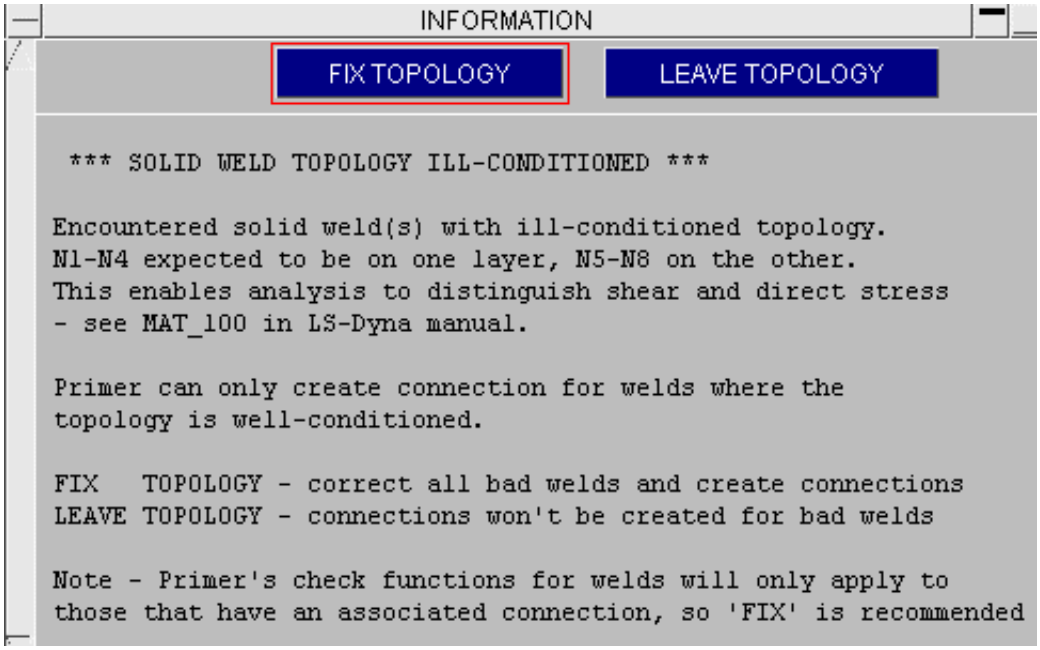




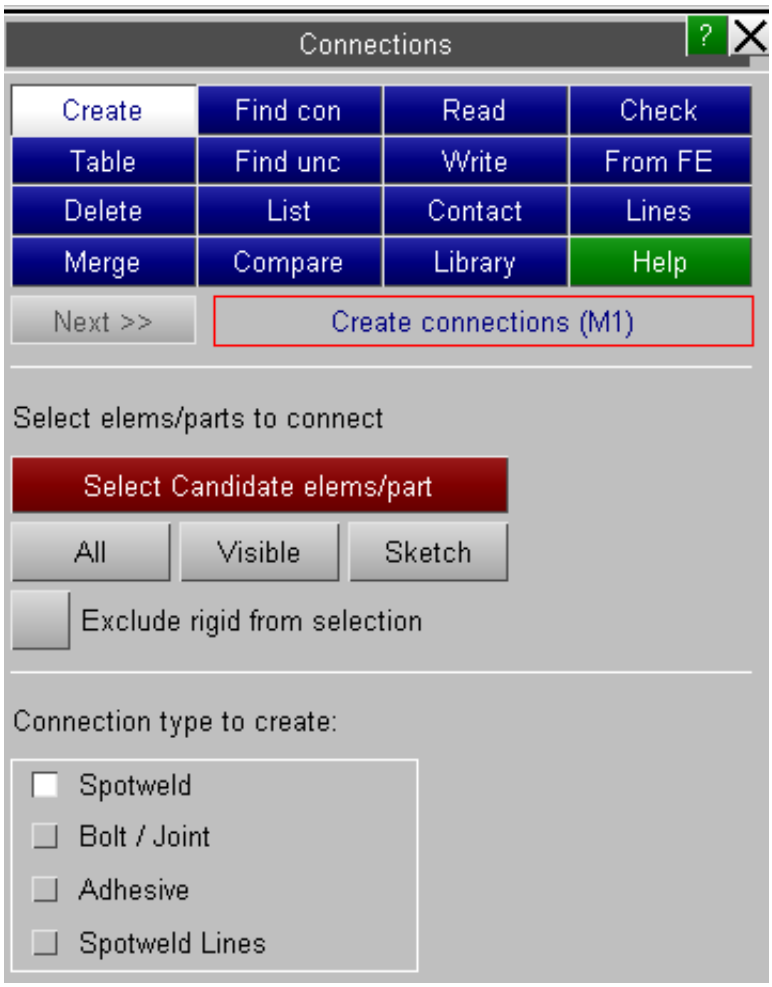
Fixing of solid topology

Creation of connections from solid welds (excluding nugget welds) requires that the solid topology is correctly configured. Nodes N1-N4 should be on one layer and nodes N5-N8 on the other. LS-Dyna actually requires this if direct and shear stresses are to be correctly calculated. An information panel will report the problem and give the user

the option of fixing the topology.



Creating Connections: spotwelds, bolts, adhesive



The connection creating panel allows you to select creation option - spotweld, bolt/joint, adhesive or spotweld lines and to select the shells to be considered.

Selecting which elements to connect

This may be as simple as just selecting all of the panels in the vehicle or you may just want to select 2 or 3 panels to connect

PRIMER v16 considers SHELLS, SOLIDS, THICK SHELLS and SHELL_NURBS_PATCH as candidates for spotweld/adhesive connection

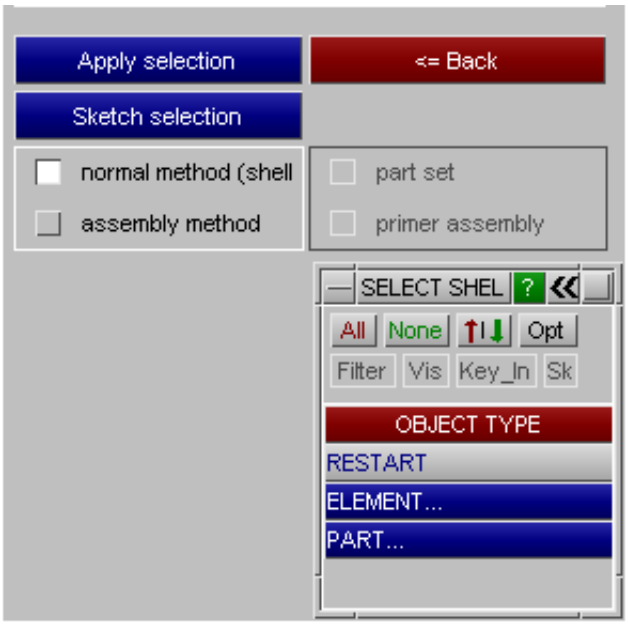
Previous versions treat only shells and even in v16 bolts will only connect to shells.

All will select all elements in the model

Visible will select all visible elements (i.e. entity switch ON and unblanked)

Exclude rigid from selection is active the selection will ONLY be applied to deformable elements

Select candidate elems/part allows you to select on the object menus



When PRIMER creates a spotweld from a point you give it, it considers which elements near the point are selected for welding and tries to create a spotweld between these elements. If you do not want a certain panel to be welded, do not include it in the selection.

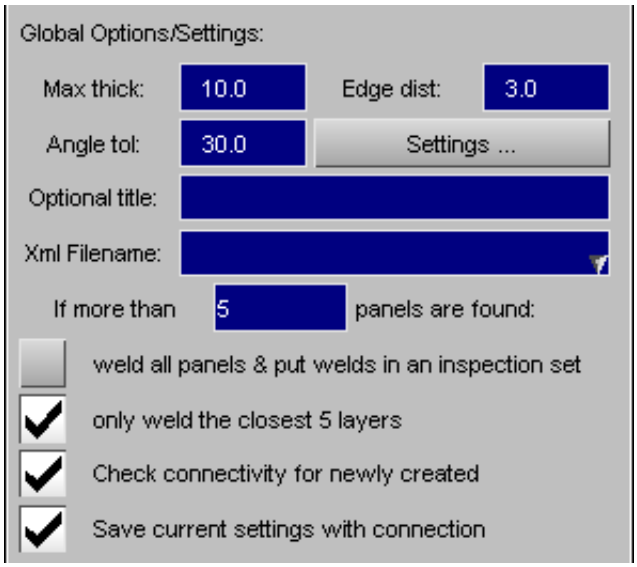
An alternative method for selecting the elements to connect is by selecting the **assembly method** rather than the normal method. Using the **assembly method** you can select a part set or PRIMER assembly as your selection of source elements. When using this method PRIMER will store the part set/assembly with each connection created. This means that if the part set/assembly is modified (parts added/removed) and the connections remade, PRIMER will update the connection so that the layers take into account the changes made to the part set/assembly. This method is most suitable if your connection information is based on assemblies of parts, rather than refering directly to parts they are connecting.

To finish selecting the elements to weld press the **APPLY SELECTION** button. You can sketch the elements/parts that you are selecting at any time by pressing **SKETCH SELECTION**.

Once done **Next >>** reconfigures this panel to display the global options

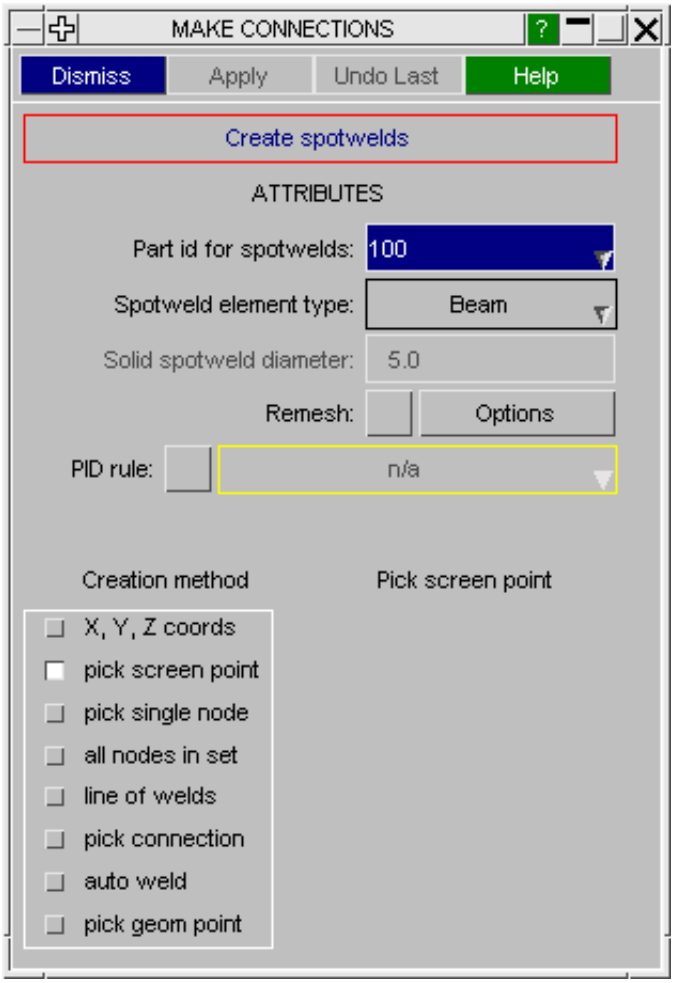
When creating any connection, a title can be added to the connection by typing the title in the optional title box and one can set an xml filename to be associated with the connection (this assumes they will be exported to a file of that name when connections are written.

Note when connections are created all the various settings used during creation are stored with the connection entity. This means that when remaking the connection the saved settings are reused. This is new functionality added in v14 onwards. This can be turned off in the settings panel by unticking **Save current settings with connection**. When turned off, PRIMER will use the current program settings when remaking connections,



The creation panel will now open in the appropriate mode

Creating Spotwelds



Before any connections can be created the user should specify

1. The spotweld element type
2. A part to put the spotweld elements in.
3. spotweld diameter for solid welds

When these steps are done you can start creating spotwelds. You can make a spotweld by either:

- Typing in the X, Y, Z coordinates. The spotweld will be created at this point.

- Picking an arbitrary screen point. This does not have to be a location of a node. The spotweld will be created at this point using the element found under the pick.
- Picking a connection from the model. The spotweld will be created at the connection location.
- Picking a single node from the model. The spotweld will be made at the node location.
- Picking a minimum of 2 screen points to create a line. Spotwelds will be created on this line. The number of spotwelds can be defined by number or pitch
- Using a node set. The default behavior is now to replace each *Constrained weld which uses any nodes of the set. Previous method of making a spotweld at each node is still available.
- automatically detecting flanges and creating welds.
- Selecting geometry points that exist in any model read into PRIMER.

There are various options that can be set to control how spotwelds are made.

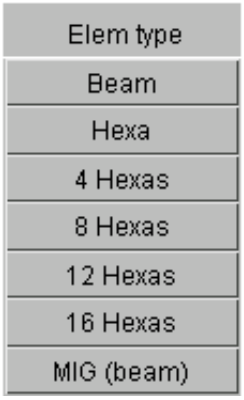
PRIMER v14 added the ability to remesh panels around spotwelds. This can be used to represent heat affected zones around welds more accurately. The **Remesh** options control this. If you want the spotweld connection to remesh panels then select the checkbox. Pressing **Options** sets the options for how the remshing is done. See the Spotweld remeshing section for more details. In this case only shells should be selected.

PRIMER v15 added the ability to give different PIDs for beams/solids between each layer of the spotweld instead of using the same PID using 'rules'. The **PID rule** option controls this. See the PID rule section for more details.

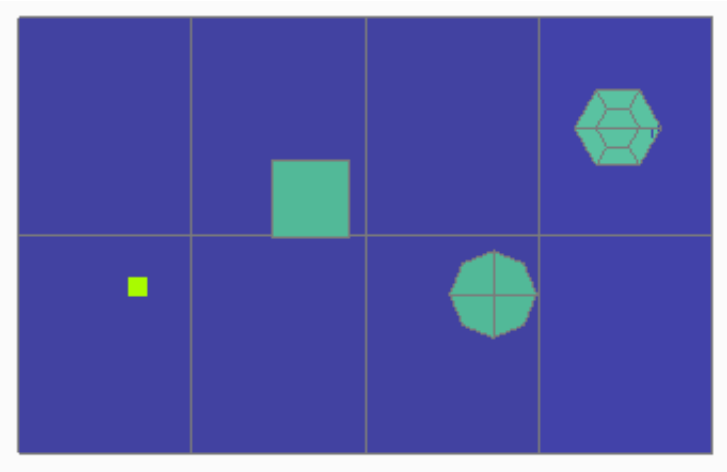
Choosing the spotweld element type

First, the type of spotweld connection must be defined. Spotwelds can either be beams or solid elements.

To create spotweld connections, PRIMER offers the following options:



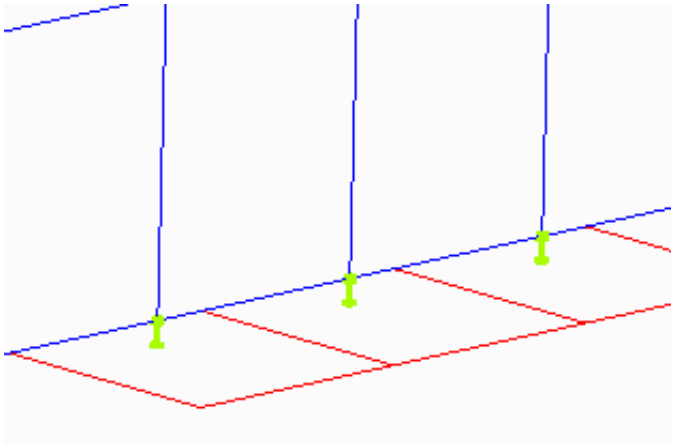
The Beam and Hexa options allow you to create mesh independent spotwelds using a single beam, a single solid or multiple solids between panels. The image on the right shows examples of Beam, Hexa, 4 Hexa and 8 Hexa welds.



If one of the solid Hexa element options is selected, the spotweld nugget diameter can be modified.

The MIG (beam) option allows you to create a beam to represent a portion of a MIG weld. Typically many of these connections would represent a MIG weld seam. The beam is meshed in (shares a node with the shell) at one end (the blue part in the figure on the right). The other end of the beam is projected onto the other panel and is mesh-independant (like the normal beam weld).

Also see converting MIG weld to beamless.



Choosing a part for the spotweld elements

PRIMER needs to know which part to put the spotweld elements into. If there is only one part in the model that is suitable (i.e. for beams if the part uses material `*MAT_SPOTWELD` and section type `*SECTION_BEAM`, or for solids if the part uses material `*MAT_SPOTWELD`) then PRIMER will automatically select it. Otherwise you will have to select it.

To select a part type in the part number, or you can use the standard popup functions (right click) to select or create the part. Generally, the part **must** use material type `*MAT_SPOTWELD` (material 100). It is possible to specify other valid material types by setting the following preference:

additional_valid_spotweld_material_types: 196, 240

In the above example, the preference is set to allow MAT_196 and MAT_240 as valid material types to used for spotwelds along with MAT_100.

When using MAT_100, if the spotwelds are defined as beam element, the part **must** use a section type `*SECTION_BEAM` type 9.

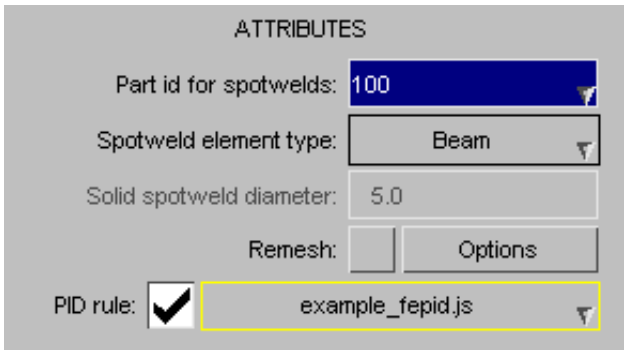
Once the part has been selected or created the part number will be displayed in the box:

Note that when creating spotwelds, one part is used for all beams/solids created. After the connection has been created, you can modify the part to be different between layers on the connections table. So, for example for a 3T weld, the beam/solids between layer 1 and layer 2 can reference a different part to the solids between layer 2 and layer 3.

PID rules

By default PRIMER uses a single part for all of the beams/solids created for a spotweld. However you may want to give different PIDs for some of the beams/solids when joining panels of different thicknesses and/or different material properties to give different spotweld properties. In versions prior to version 15, once the weld had been created, the PIDs could be changed for each layer by using the Parts columns in the connection table.

In version 15 PRIMER added the ability to use a PID rule when creating the spotweld to set the PID for each layer of the weld. This is similar to the connection rules used in spotweld remeshing. As well as specifying a default **Part ID** to use for the spotweld you can also specify a **PID rule** to use. In the image below the PID rule `example_fepid.js` has been used.



A PID rule is a special JavaScript which PRIMER runs for each pair of layers when creating the spotweld. For example if a spotweld connects two panels together the rule will be run once, if it connects three panels together the rule will be run twice.

Each time the rule is run it is passed information about the spotweld and it can return the PID to use for the beam/solids. This means that a different PID can be used for layer if required.

PRIMER will look for PID rules in the directories

$OA_ADMIN/primer_library/connection_rules
$OA_INSTALL/primer_library/connection_rules
$OA_HOME/primer_library/connection_rules

There is one simple example PID rules that has been give out with PRIMER to help understand how they work:

example_fepid.js	This example rule sets the PID to 1000 + the layer number. e.g. if the spotweld connects three panels together the rule will return PID 1000 for the beams/solids between the first pair of panels and 1001 for the beams/solids between the second pair of panels.

Each time the rule is run it can specify the PID to return using the special function `Conx.SetRuleFEPID()`. If the PID is not set then the default PID from the creation panel will be used.

The data for the connection is passed to the rule using the arguments array. The data is:

arguments[0]	The absolute name of the connection rule
arguments[1]	Model object for the model that the connection is being made in
arguments[2]	An object containing the remeshing data

The remeshing data object contains the following properties

Property	**Description**
label	Connection label
type	Connection type (Conx.SPOTWELD, Conx.ADHESIVE or Conx.SPOTWELD_LINE)
subtype	Connection subtype (Conx.SPOTWELD_BEAM, Conx.SPOTWELD_SOLID1, ADHESIVE_SOLID etc)
nlayers	How many layers the connection has
nrings	How many rings the connection has [if remeshing]
coords	The coordinates of the connection (array of length 3)
layer	The layer we want to specify the PID for
pid	Array of layer parts (length nlayers)
mid	Array of layer materials(length nlayers)
thickness	Array of layer thicknesses (length nlayers)
weldpid	Array of spotweld parts (usually only 1)
weldmid	Array of spotweld materials (usually only 1)
remesh	false as script not called as remesh rule
fepid	true as script called as a FE PID rule

so, for example, the script can get the data using:

```
var rule  = arguments[0];  // The name of the rule
var model = arguments[1];  // The model the connection is being made in
```

```
var data  = arguments[2];  // The remeshing data
```

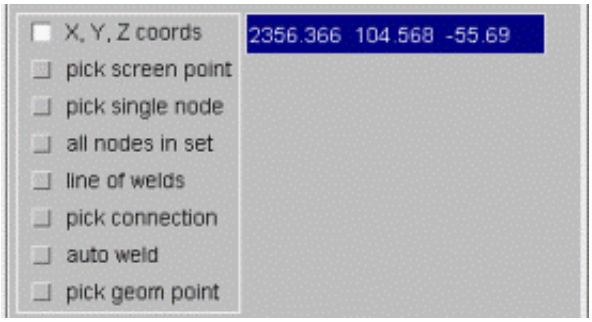
The layer would then be available as
`data.layer`

For a simple example of a rule, the example rule `example_fepid.js` sets the PID to 1000 + layer

```
if (data.fepid)
{
// Add 1000 on to layer ID
    Conx.SetRuleFEPID(1000 + data.layer);
}
```

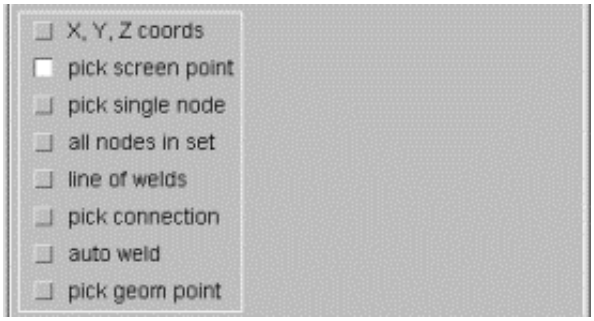
Once the PID rule is run and PIDs are assigned for the different layers they are automatically assigned to the PID (L2-L3), PID (L3-L4) fields for the connection. These can be viewed and changed by using the Parts columns in the connection table.

Using coordinates



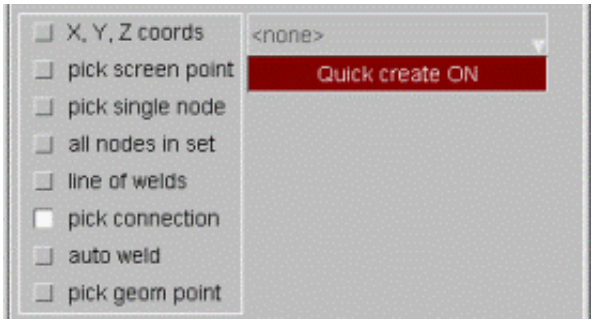
Type the X, Y, Z coordinates into the box and press the **APPLY** button. The spotweld will be created if it is possible. If the weld cannot be made an error message in the dialogue box will give the reason why. You can undo the spotweld if it is not what you want by pressing **UNDO CREATE**.

Using a screen point



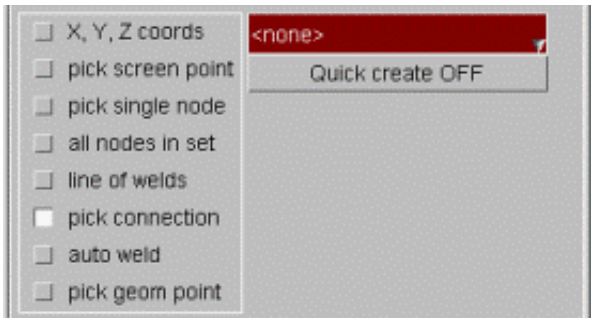
Using the cursor, select a point on the screen at which you wish the spotweld to be created. The spotweld will be automatically created at the point selected.

Using a connection

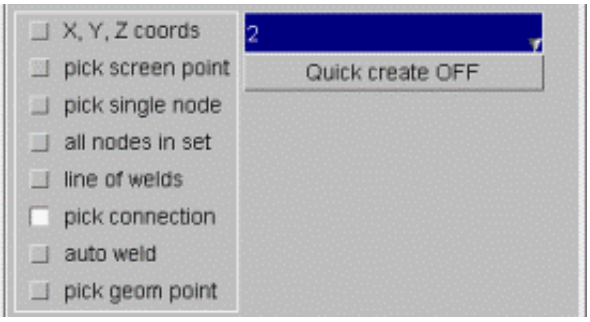


If quick create is turned on, you can just pick an existing connection from the screen. The spotweld will be created if it is possible. If the weld cannot be made an error message in the dialogue box will give the reason why. You can undo the spotweld if it is not what you want by pressing **UNDO CREATE**.

If a spotweld or bolt is already defined for the selected connection, you can substitute this existing element with the new spotweld by selecting the Delete old connection option

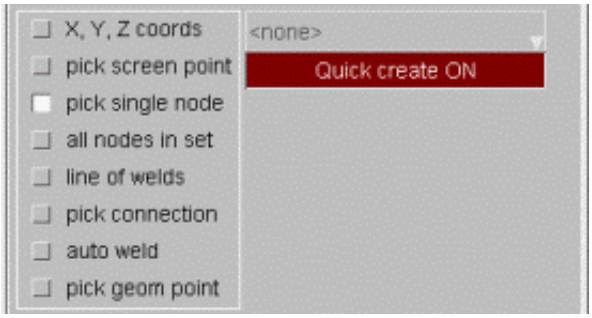


If quick create is turned off, you can type the connection number into the box or use the normal popup functions to select a connection.

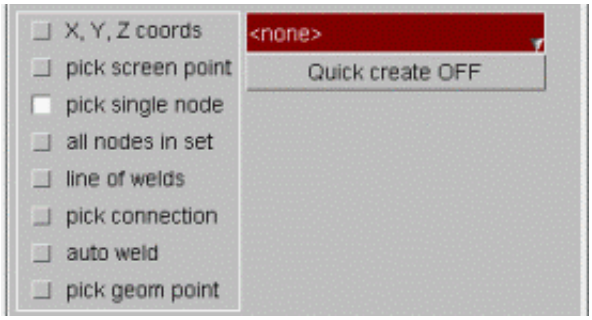


Once the connection has been selected the spotweld can be created by pressing the **APPLY** button. You can undo the spotweld if it is not what you want by pressing **UNDO CREATE**.

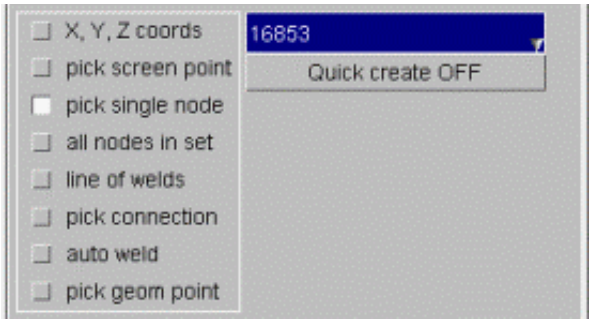
Using a node



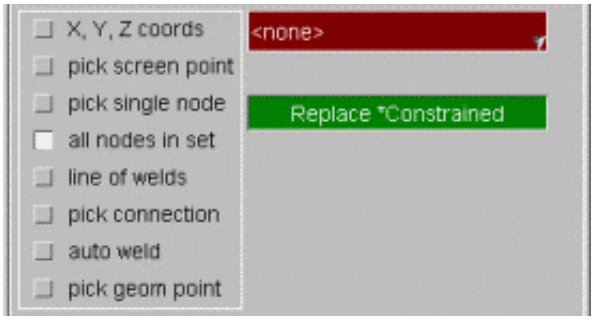
If quick create is turned on you can just pick a node from the screen. The spotweld will be created if it is possible. If the weld cannot be made an error message in the dialogue box will give the reason why. You can undo the spotweld if it is not what you want by pressing **UNDO CREATE**. If quick create is turned off you can type the node number into the box or use the normal popup functions to create or select a node.



Once the node has been selected or created the spotweld can be created by pressing the **APPLY** button. You can undo the spotweld if it is not what you want by pressing **UNDO CREATE**.



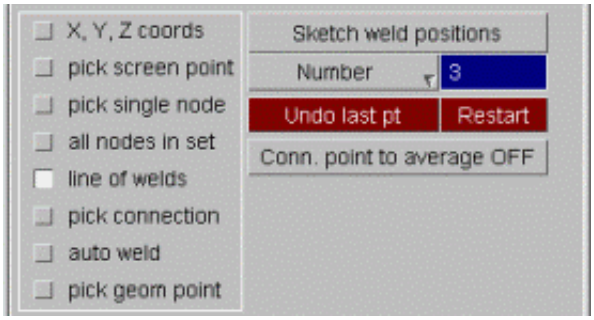
Using a node set



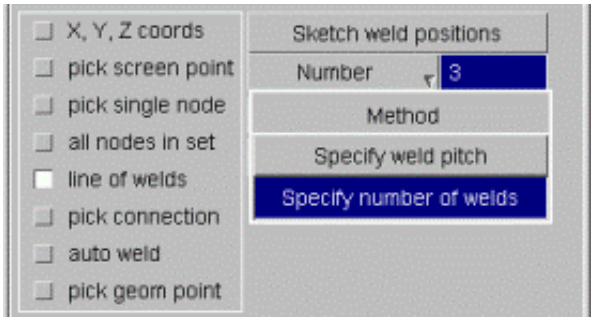
The "Replace *Constrained" option is designed to replace with a spotweld every *CONSTRAINED_SPOTWELD or *CONSTRAINED_GENERALIZED_WELD which has at least one node in the selected set. The spotweld will be created between panels selected for welding. NOTE - this may not be all the panels which the old weld joined if the original shell selection was incomplete. In this case the connection table will be invoked for these welds.

The old option "weld every node in set" is still available. In this mode PRIMER will attempt to create a spotweld at every node in the set and dump to a node set any nodes where the weld could not be made.

Using a line of welds

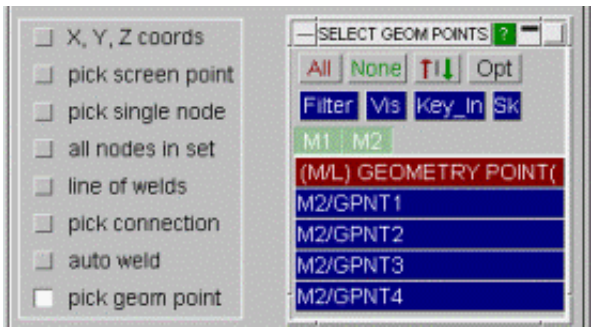


Using the cursor, select 2 or more points in order to create a line along which you wish to create spotwelds. For the MIG type spotwelds, this mode works in a different way. Here you select 2 nodes along a free edge/feature line. PRIMER will determine all the nodes along the free edge/feature line between the two nodes chosen. Clicking on **Apply** after this will create MIG spotwelds at all the nodes along the free edge/feature line between the 2 selected nodes. If you choose the same node twice for this operation, PRIMER will create MIG weld beams for every node around the free edge.



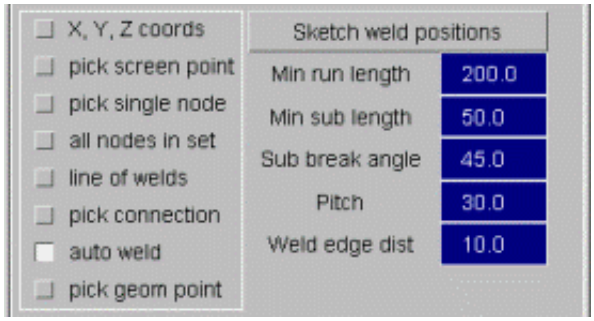
By either clicking on the tab or using the available popup function, specify whether the quantity of spotwelds you require is determined by **Number** or by **Pitch**. If using number, type the number of spotwelds required along the line in the box. If pitch is required, type the desired distance between spotwelds in the box. Once completed, press the **Apply** button to create the spotwelds.

Using geometry points

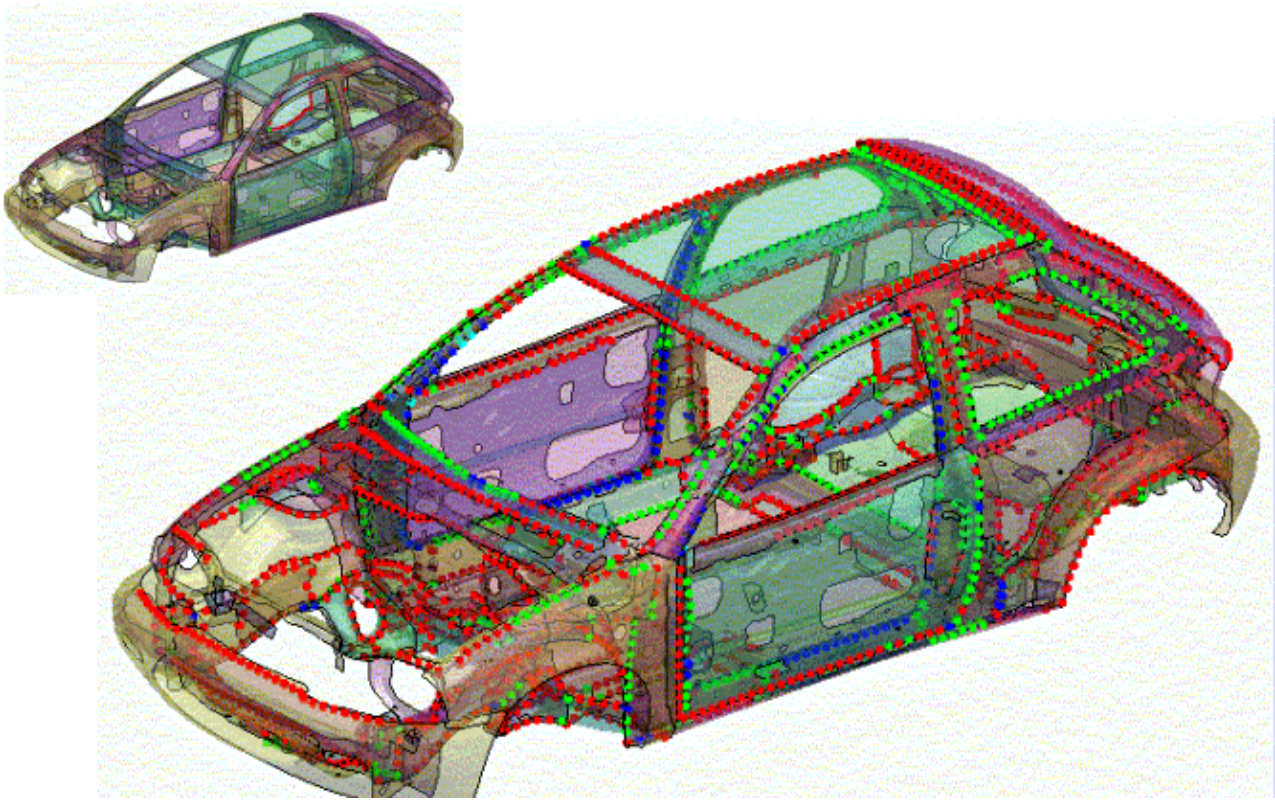


Geometry points can be selected from any model in PRIMER. The coordinates of these points are used as the coordinates of the connections to be created.

Auto Welding



PRIMER has the ability to automatically weld panels together with the only input being the shells to weld and a few user defined parameters, for example:



The selected shells are searched through, and any shells that are close together are flagged for the 2nd stage of the auto welding process. The second stage takes these shells and highlights any model free edges that belong to the shells - these are called **free edge runs**. Finally, each of these free edge runs are split into sub sections (at feature edges defined by a user defined angle **"sub break angle"**) and spotwelded at a user defined pitch and distance from the edge. The weld run is centred so there is an equal amount of space at the beginning and end of the weld.

Any welds that are too close together are discarded - eliminating the chance of multiply welded parts.

You can sketch the possible weld points using the **Sketch weld positions** button; note that this shows all attempted weld points - weld runs that lie on top of one another will no doubt have many weld points unmade because they are too close.

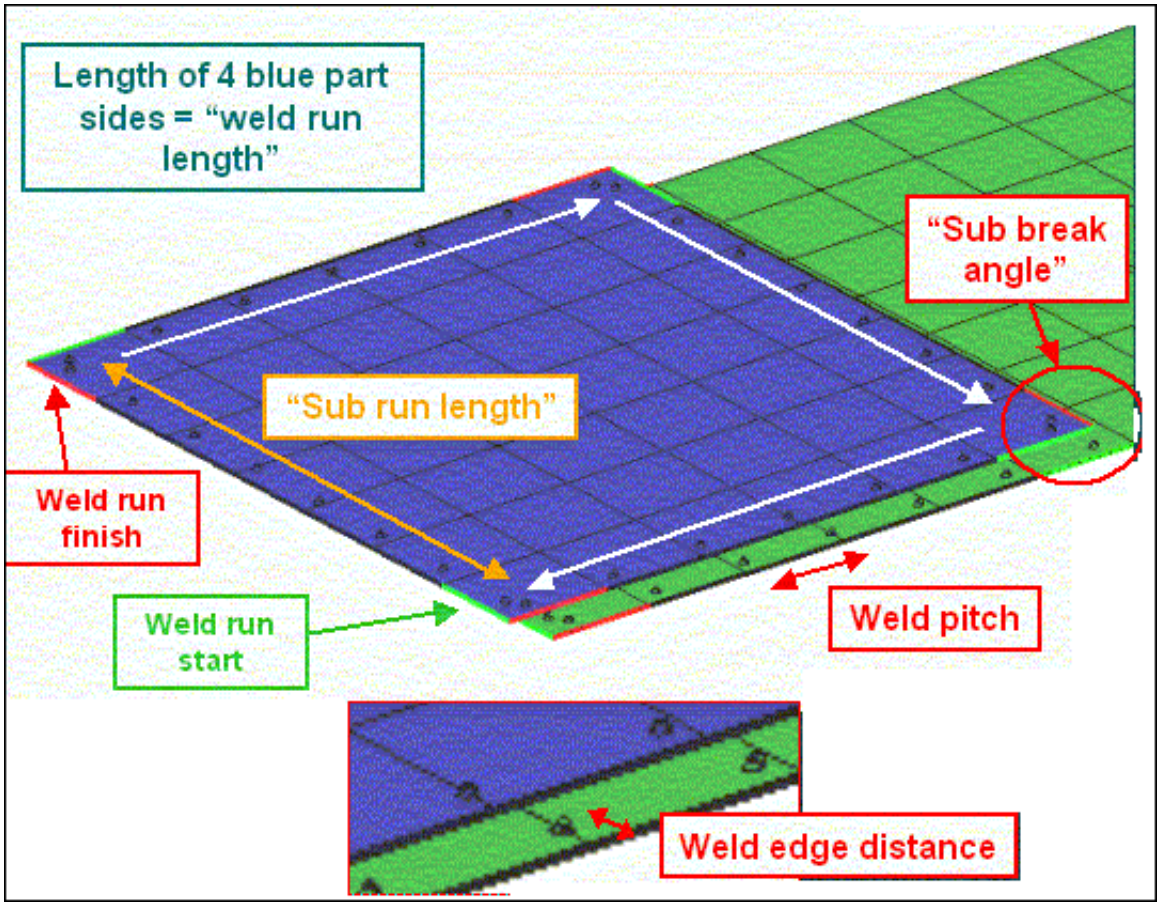
The user defined parameters are as follows:

- **Min run length**: any free edge runs that are less than this amount are discarded.
- **Min sub length**: any sub sections that are smaller than this amount are discarded.
- **Sub break angle**: the angle that determines how the free edge runs are split up.
- **Pitch**: pitch of the welds
- **Weld edge dist**: distance to weld in from the free edge run.

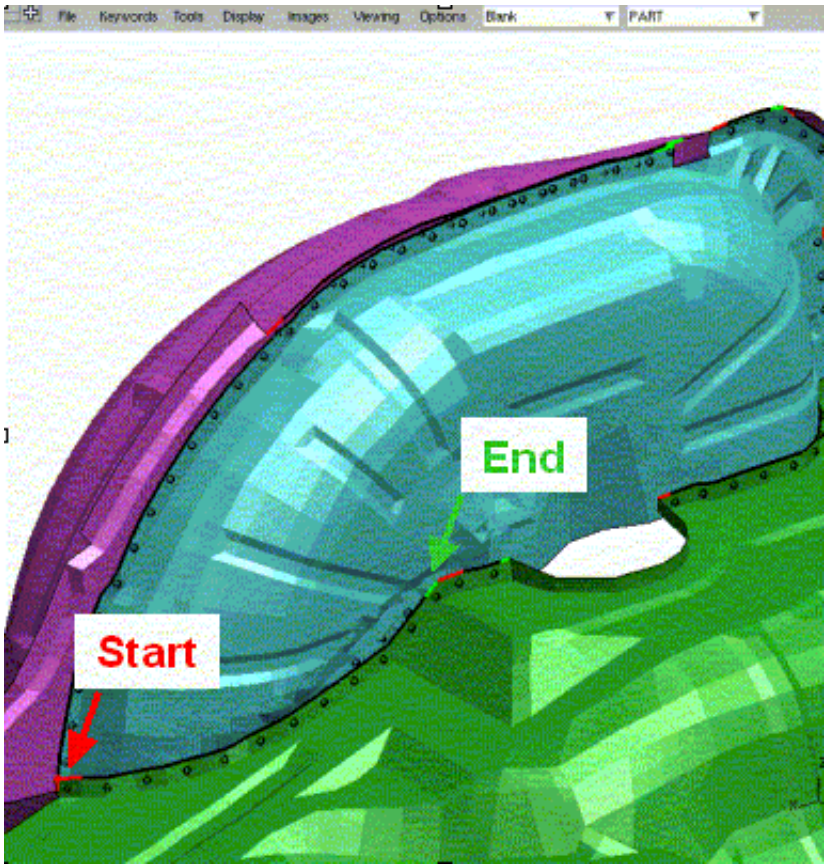
A master part or a master part set can be used to specify which panel(s) are used to determine the free edges. If

specified, only free edges on the master part(s) are used to construct spotwelds. Without a master part/part set selected, all shells selected for connection are considered when determining free edges.

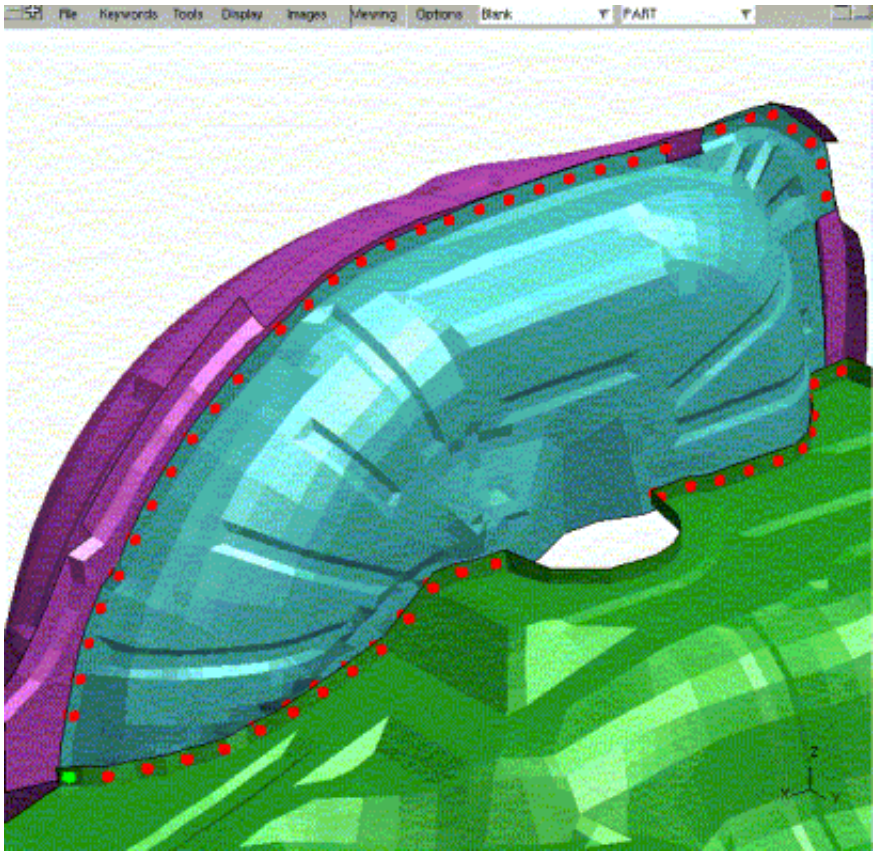
The following shows an example of sketching the positions, where the weld runs are shown as green lines (start position), red lines (end position) and black lines (min run lengths). Each possible position on each weld run is sketched as a square. Explanations of the parameters are also shown:



Sketch weld positions will sketch all the potential weld points PRIMER has calculated to attempt to weld. A green line marks the start of a run, and a red marks the end

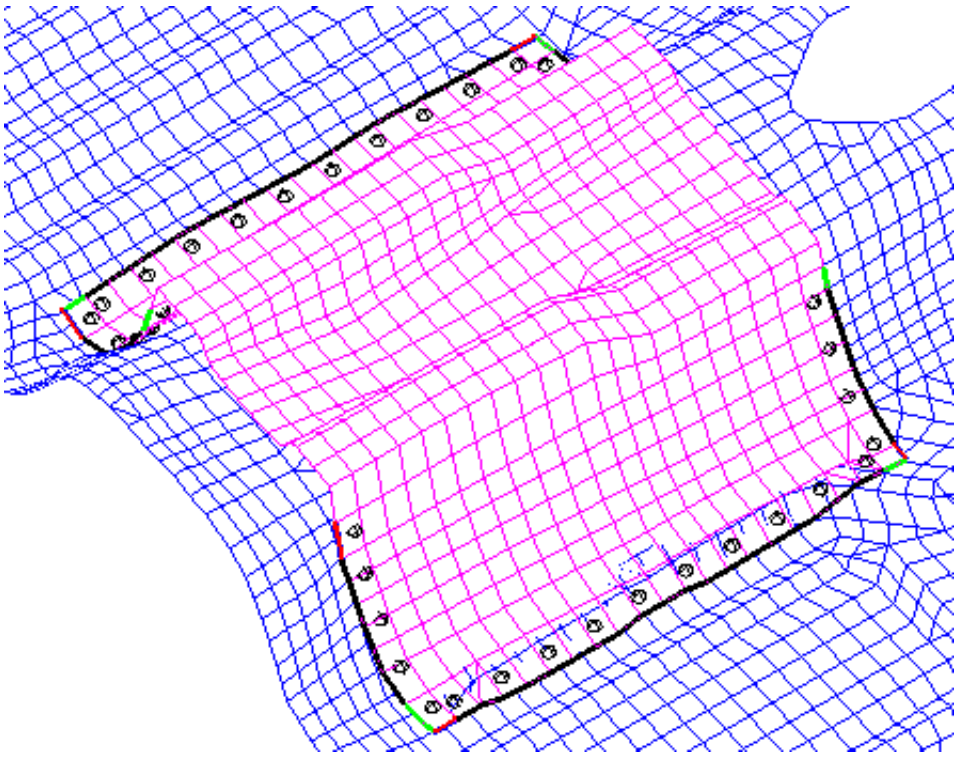


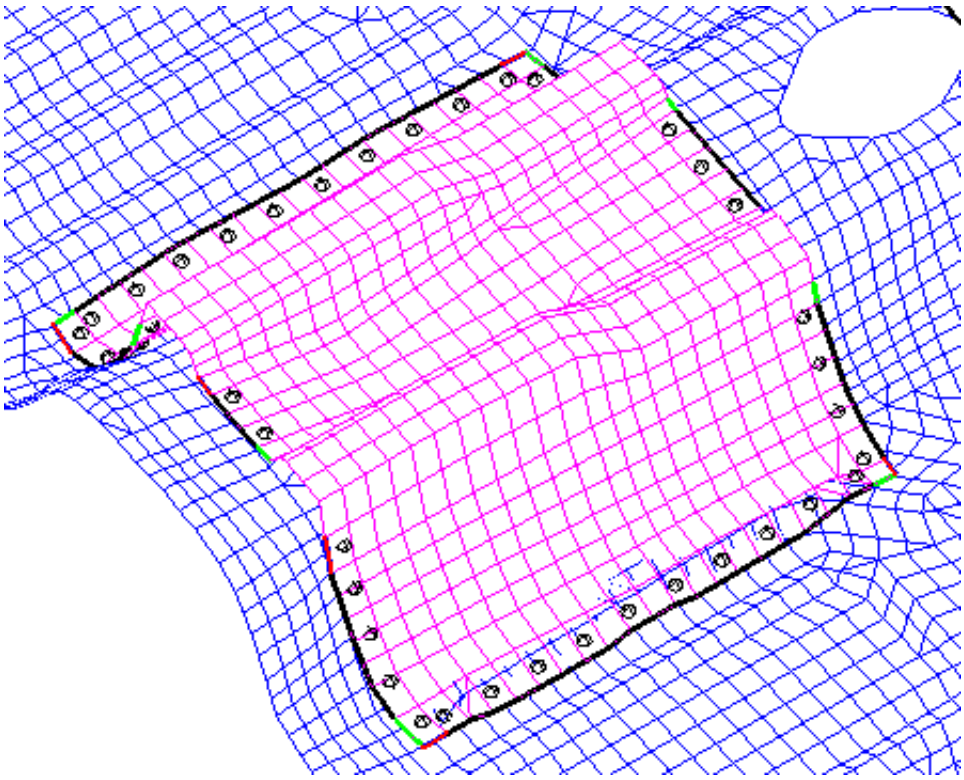
The following shows an example of the automatic welding, the red dots are the successful 2-panel welds, and the green dots are the 3-panel welds. Even though there were weld runs (and therefore weld points) that were next to each other, the auto weld routine checks for proximity and will not weld any points that are too close to each other.



The next image shows the result of changing the minimum sub length to create welds on the two short sections at either

edge of the magenta panel. On the first image, the sub length isn't enough to allow the sections to be welded, on the second image, the sub length has been reduced so the auto spotwelder allows these welds:





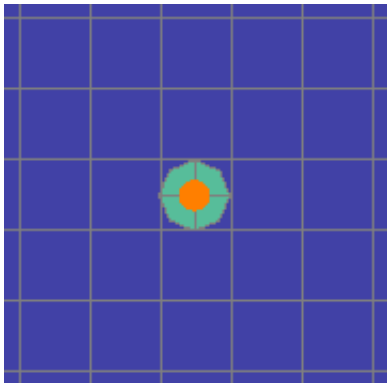
Note: There are 2 oa_pref options to control how tolerant the autoweld feature is when checking close welds. They are factors on the spotweld pitch and are used to calculate the distance to check for close welds on nearby seams.

The first is `autoweld_diff_seam_proximity` (0.5) which is a factor for checking welds on a different seam.
The second is `autoweld_same_seam_proximity` (0.8) which is a factor for checking welds on the same seam.

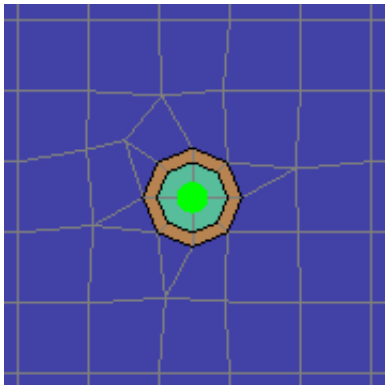
e.g. distance to check = Pitch * autoweld_same_seam_proximity

Spotweld remeshing

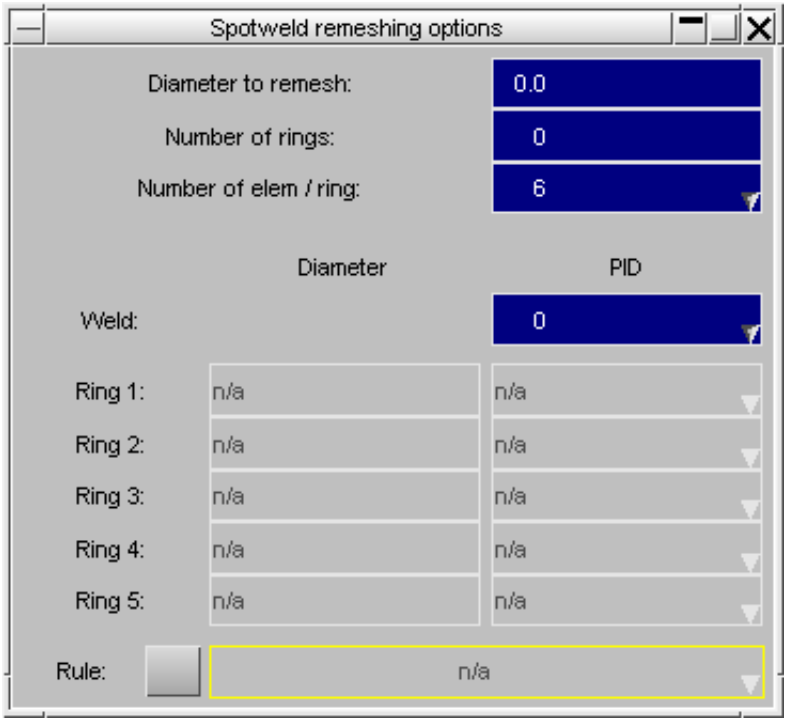
Spotweld remeshing is used to remesh the panels that are welded so the spotweld is directly meshed into them. For example the following image shows a normal 4 solid nugget spotweld which is attached using a tied contact.



The image on the right shows the same spotweld connection but using the spotweld remeshing, creating a ring of elements around the weld to represent the heat affected zone. These elements are moved to a different part so the material properties can be different.



When creating spotwelds the remeshing is controlled in the **Spotweld remeshing options** panel.



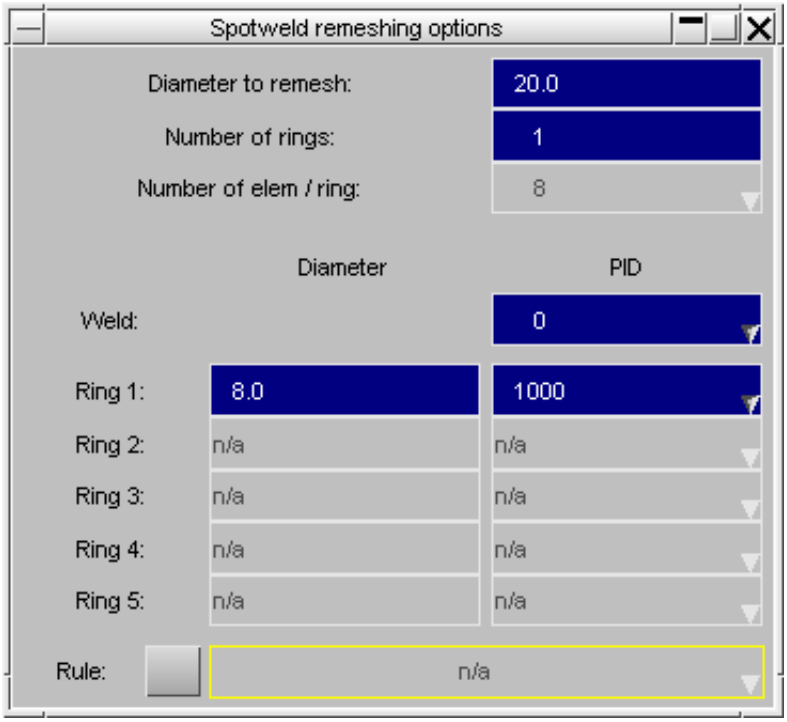
PRIMER nneds to know how much of the panel to remesh when creating the spotweld. The **Diameter to remesh** setting is used to control this. If set to zero then PRIMER will automatically choose which elements to remesh, otherwise give a diameter and PRIMER will select any elements that have a node inside that diameter to remesh. The default diameter (if setting is zero) is:

largest ring diameter + (2 x spotweld diameter)

If the number of rings is zero then this is then (3 x spotweld diameter)

The **Number of rings** setting controls how many rings of elements will be created around the spotweld to represent

the heat affected zone. If set to zero no rings will be created. A maximum of 5 rings is allowed. In the image above no rings have been selected so the data for the 5 rings is greyed out. In the image below one ring has been selected so data can be given for that ring.
The **Number of elem / ring** setting can only be changed for beam spotwelds. When remeshing around beam spotwelds PRIMER will either make a hexagonal or octagonal mesh representing the weld. In the image above it is set to 6 so a hexagonal region will be created. In the image below we are creating a solid spotweld so the option is not available to change but it will indicate how many elements will be used per ring. In this case we are making a 4 solid nugget so the value is 8.



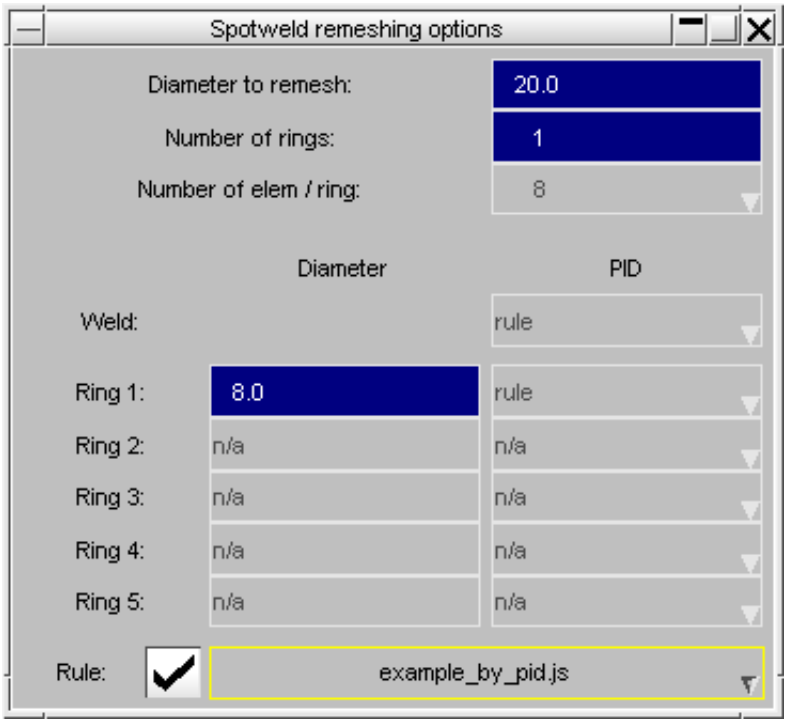
When remeshing the panel around the spotweld for solid spotwelds PRIMER creates shell elements 'on' the spotweld. For example for a single solid spotweld PRIMER will create a shell on the spotweld for each layer. For a 4 solid nugget PRIMER will create 4 shells for each layer. The part that these are created in is given by **Weld PID**. If it is zero then PRIMER will use the part ID from the panel being welded.

The diameter of each ring and the **PID** for the shells creating each ring can then be given. If the **PID** is zero then PRIMER will use the part ID from the panel being welded.

Connection rules

The method decribed above of specifying the number of rings to create and a PID for each ring around the spotweld may be sufficient for modelling basic spotwelds. However, it may be desirable to have much more control of the part IDs created for the spotweld rings. For example different PIDs may be required for joining panels of different thicknesses and/or different material properties.

This can be done in PRIMER by using a connection rule when remeshing the spotweld. Instead of specifying a **PID** to use for each ring you instead specify a **connection rule** to use. In the image below the connection rule `example_by_pid.js` has been used.



A connection rule is a special JavaScript which PRIMER runs for each ring and each layer when remeshing the spotweld. For example if a spotweld connects 2 panels together and three rings have been defined the rule will be run 8 times.
For each layer (2 layers) the connection is run for each ring (3 rings) and also the central weld portion. So it is run 2 x (3 + 1) times.

Each time the rule is run it is passed information about the spotweld and it can return the PID and/or the diameter. This means that a different PID can be used for each ring and/or layer if required and the diameter can be different for each layer if required.

PRIMER will look for connection rules in the directories

$OA_ADMIN/primer_library/connection_rules
$OA_INSTALL/primer_library/connection_rules
$OA_HOME/primer_library/connection_rules

There are 3 simple example connection rules that have been give out with PRIMER to help understand how they work:

example_by_pid.js	This example rule adds 1000 to the PID for rings around the spotweld e.g. if the spotweld connects parts 100 and 101 the PID for elements in the rings will be 1100 and 110
example_by_nlayer.js	This example rule sets the PID for rings around the spotweld depending on how many layers the connections has. If there are 2 layers the PID is set to 1000 If there are 3 layers the PID is set to 2000. Additionally for a 3 layer connection if there is one ring the diameter of the ring is changed to be 9mm
example_by_thickness.js	This example rule sets the PID for rings around the spotweld depending on the thickness of the panel being welded. If the panel is < 0.7mm the PID is increased to 1000 Additionally if there is one ring the diameter of the ring is changed to be 9mm. If the panel being welded is >= 0.7mm then the defaults are used

Each time the rule is run it can specify the PID to return using the special function `Conx.SetRulePID()` and the diameter can be changed (if required) using `Conx.SetRuleDiameter()`. If the PID is not set then the PID from the layer definition will be used. If the diameter is not set then the default diameter given for the ring will be used.

The data for the connection is passed to the rule using the arguments array. The data is:

arguments[0]	The absolute name of the connection rule
arguments[1]	Model object for the model that the connection is being made in

arguments[2]	An object containing the remeshing data

The remeshing data object contains the following properties

Property	Description
label	Connection label
type	Connection type (Conx.SPOTWELD)
subtype	Connection subtype (Conx.SPOTWELD_BEAM, Conx.SPOTWELD_SOLID1 etc)
nlayers	How many layers the connection has
nrings	How many rings the connection has
coords	The coordinates of the connection (array of length 3)
layer	The layer we are currently running the rule for
ring	The ring we are currently running the rule for
pid	Array of layer parts (length nlayers)
mid	Array of layer materials(length nlayers)
thickness	Array of layer thicknesses (length nlayers)
ringdiameter	Array of ring diameters (length nrings)
weldpid	Array of spotweld parts (usually only 1)
weldmid	Array of spotweld materials (usually only 1)
remesh	true as script called as remesh rule
fepid	false as script not called as a FE PID rule

so, for example, the script can get the data using:

```
var rule  = arguments[0];  // The name of the rule
var model = arguments[1];  // The model the connection is being made in
var data  = arguments[2];  // The remeshing data
```

The layer and ring would then be available as

```
data.layer
data.ring
```

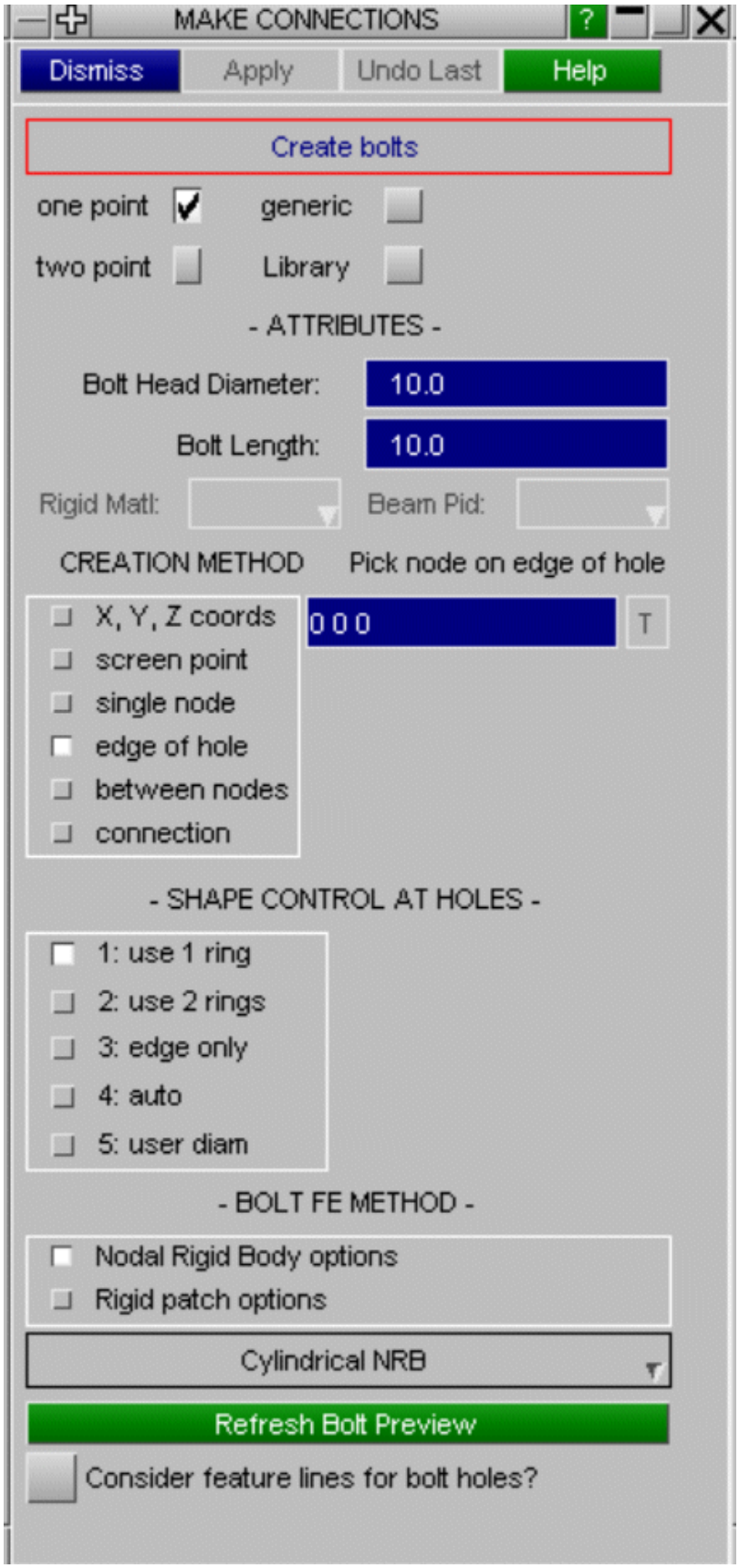
Taking the above example where a spotweld connects 2 panels together and three rings have been defined the rule will be run 8 times.
data.layer and data.ring will have the following values for each time the script is run.

data.layer	data.ring	Description
0	-1	Central ’spotweld’ region on first layer
0	0	First ring on first layer
0	1	Second ring on first layer
0	2	Third ring on first layer
1	-1	Central ’spotweld’ region on second layer
1	0	First ring on second layer
1	1	Second ring on second layer
1	2	Third ring on second layer

For a simple example of a rule, the example rule `example_by_pid.js` adds 1000 to the PID for the rings but not the central ’spotweld’ portion using

```
// Add 1000 on to PID
if (data.ring != -1)
{
    Conx.SetRulePID(1000 + data.pid[data.layer]);
}
```

Creating bolts



A bolt may be a single rigid entity (RB merge or NRB) joining multiple panels or when 2 panels are being joined, it may be 2 rigid entities (rigid patches or NRBs) connected by a beam, zero length discrete beam, revolute joint or ball joint.

The geometry of a **one point bolt** is described by a point, a diameter and a maximum length. The orientation of the bolt is calculated by Primer from the shells to be joined. This method is appropriate for short bolts.

The geometry of a **two point bolt** is described by two points and a diameter. In this case, "length" is max thickness of the connection at each end, the actual bolt length being unrestricted. The axis of the bolt is defined by the vector between the points.

In the case of merge or rigid patch bolts, extra master parts (with no elements of their own) are created which allow the unlabelled rigid body merge to be identified.

These also provides parts which can be converted to PART_INERTIA should the mass properties of the bolt require adjustment.

Optional material id. The user may enter a rigid material id to be used for the rigid patches, otherwise Primer creates one with properties derived from the panels being attached.

If creating a bolt at a hole, the **maximum washer diameter** (Settings ...) must be set to a sufficient value. The default of 20mm should be sufficient for most models.

Bolts are now **drawn in preview** and will only be created when **APPLY** is pressed.

Creating single point bolts

A single point set by a variety of methods (see below) determines the position of the bolt head.
The **Max length** parameter sets the maximum possible length of the connection. PRIMER will search for shells along the bolt axis in both directions to this length from the connection point. So the connection point should be defined at the

head/nut of the bolt not in the middle. The table function **update & remake (repos)** can be used to reposition the connection point.

For bolts which are created on panel mesh (i.e. excluding those at a hole) **Bolt head diameter** is specified. If a bolt is being created at a hole, the diameter is determined by the choice of shape control flag (see below). Bolts may be made at positions which combine meshes both with and without holes, in which case the shape control will be applied at the holes.

NRB methods use NODAL_RIGID_BODY to rigidify the panel shells. Spherical/cylindrical NRB form a single rigid body, the other methods will form 2 rigid bodies connected by the applicable FE.

Cylindrical methods determine a bolt axis and will only incorporate shells where the normal aligns (using Angle tol) with it.

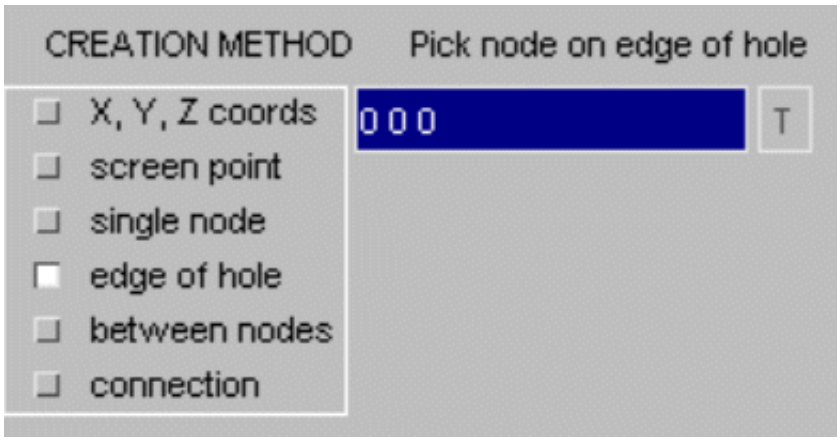
Spherical method requires no alignment of shells and will simply sweep out a radius from the connection point.

For bolt at hole, spherical methods are not applicable and therefore greyed.

Cylindrical merge forms rigid patches connecting layers which are slaved to single rigid master part. Other methods form 2 rigid patches connected by the applicable FE.

These methods will attach to rigid parts by making additional *Constrained_rigid_body definitions as necessary.

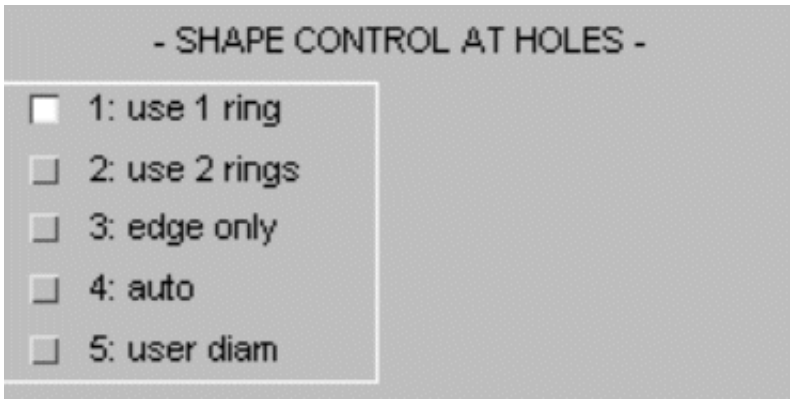
Single point bolt at hole



Picking a node on the edge of a hole in the mesh will set up the data for creating a bolt with its head at the centre of the hole.

The prospective bolt will be sketched in preview.

The diameter of the bolt is determined by the shape control setting.



For NRB type bolts the options are

1 ring - rigidify nodes of all shells around the hole

2 rings - rigidify node of all shells around hole and all that attach to them

edge only - rigidify only the nodes around the hole

auto - calculate diameter from mesh density

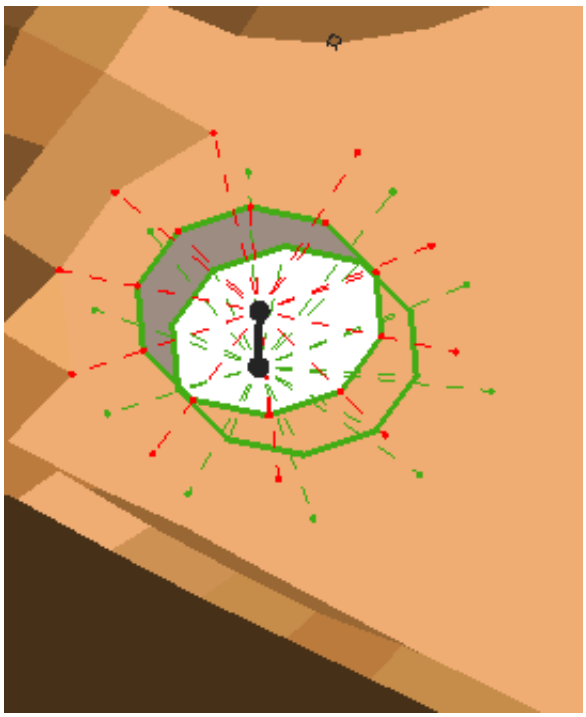
user diam - use the input diameter

For patch type bolts the options are

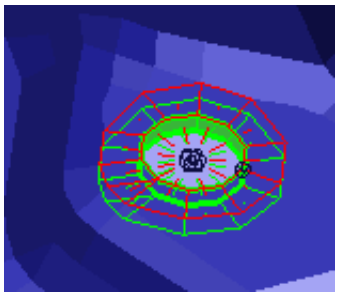
1 ring - rigidify all the shells around the hole

2 rings - rigidify those around the hole and those that attach to them

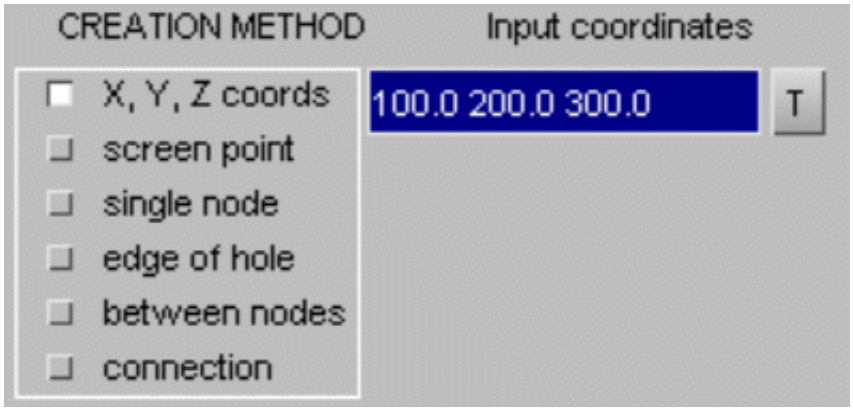
Preview of Nodal Rigid Body bolt with deformable beam formed by picking node on edge of hole. The free edge detected is drawn in green.



Preview of patch type "bolt" with ball joint connecting the rigid parts.



Single point bolt from coordinates



Type the **X, Y, Z coord**inates into the box and the bolt will be previewed if it can be made.

If the bolt cannot be made an error message in the dialogue box will give the reason why.

T option enables you to drag translate the point in global system

APPLY will create the bolt

UNDO CREATE will remove it

Single point bolt from screen point

screen point

Using the cursor, select a point on the screen at which you wish the bolt to be created.

The bolt will be previewed if it can be made.

APPLY will create the bolt **UNDO CREATE** will remove it

Single point bolt from existing (empty) connection

connection

If you pick an existing empty connection, PRIMER will preview the bolt.

APPLY will create the bolt **UNDO CREATE** will remove it

Single point bolt at a node

single node

If you pick a node, PRIMER will preview the bolt that can be created.

APPLY will create the bolt **UNDO CREATE** will remove it

Single point bolt between picked nodes

between nodes

The "between nodes" function provides a way to create a bolt between N nodal points. Once you have selected a minimum of 2 nodes the connection can be previewed by pressing **PREVIEW**.

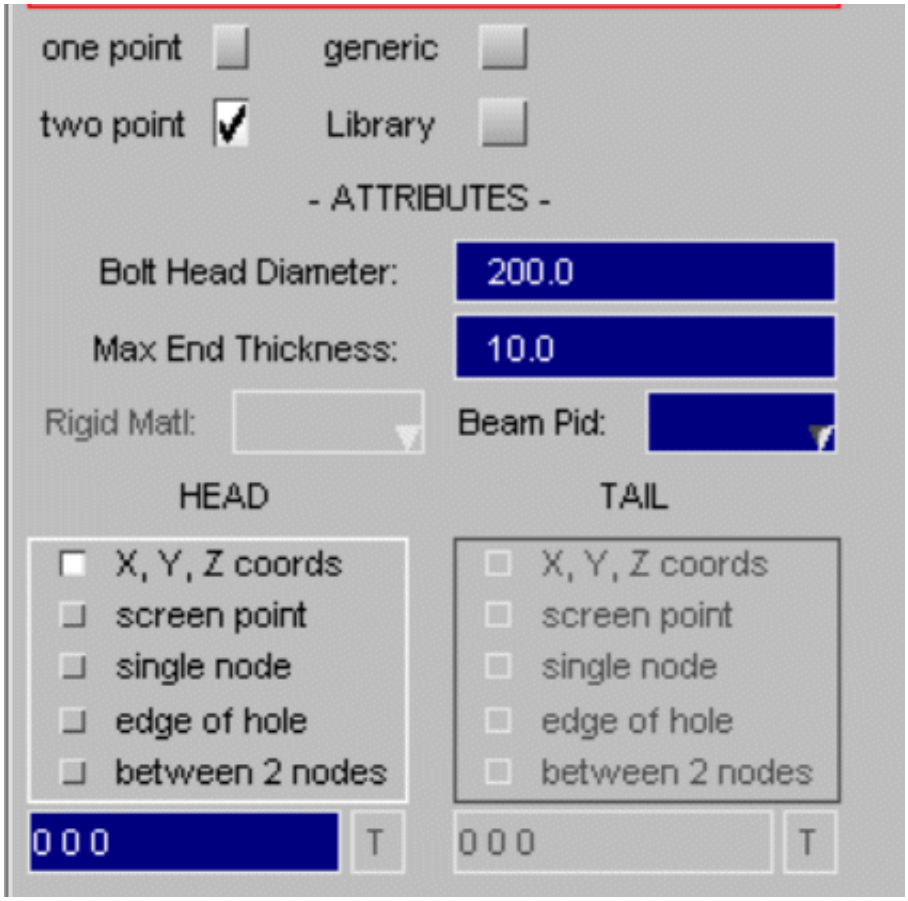
Nodes are picked by a left click and may be deselected by a middle click on the mouse.

Creating two point bolts

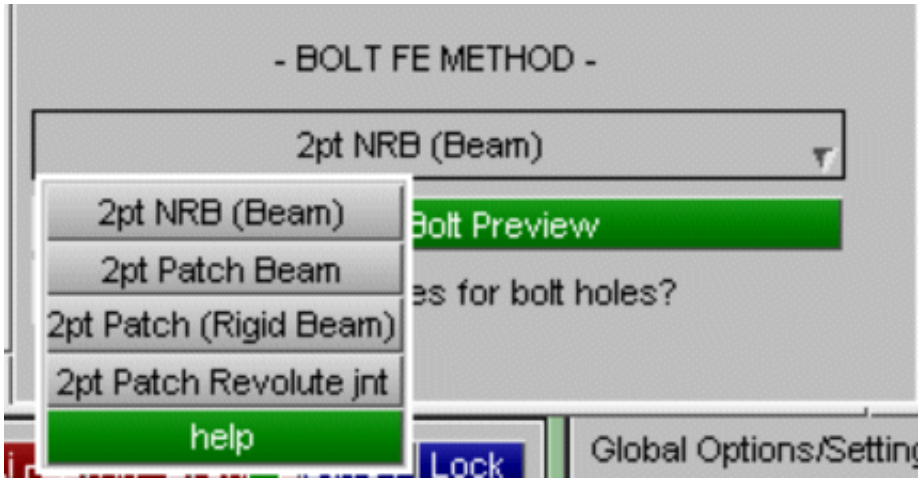
For longer bolts the 2 point method is recommended. Each point may be defined by coordinate, screen point pick or node pick. If a node on the edge of a hole is picked, the centre of the hole will be taken as the connection point.

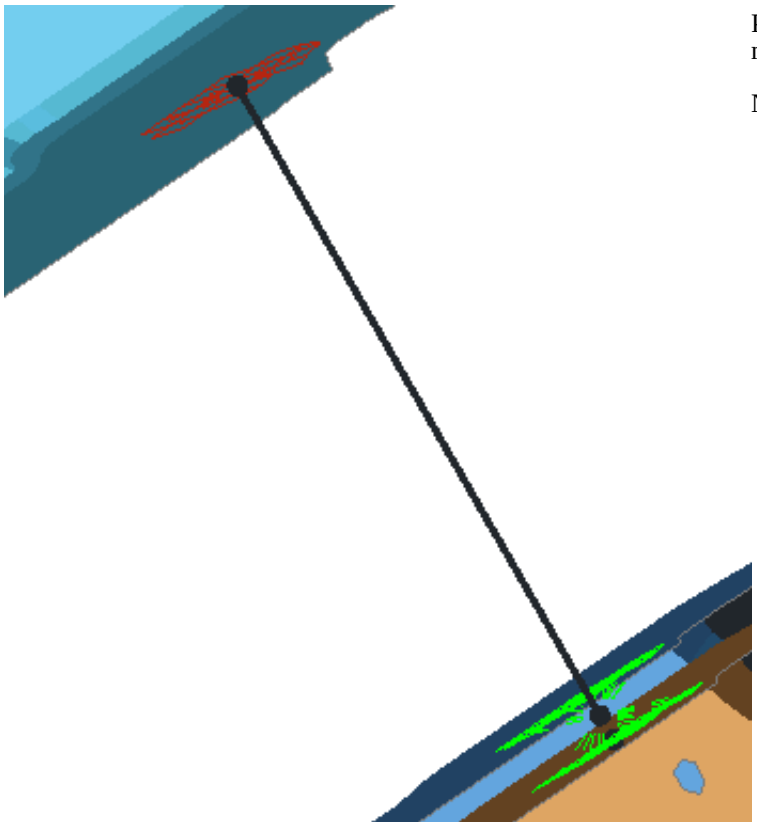
In this mode each of the two layers may contain multiple parts. Only one value of **Max end thickness** is specified. If less than half the distance between the two points the same value will be applied at each end. If greater this length will be applied at end 1 up to a maximum thickness of 90% and what remains (with a minimum of 10%) will be applied at end 2.

Once picked, the point may be dragged in global XYZ using **T** tool.



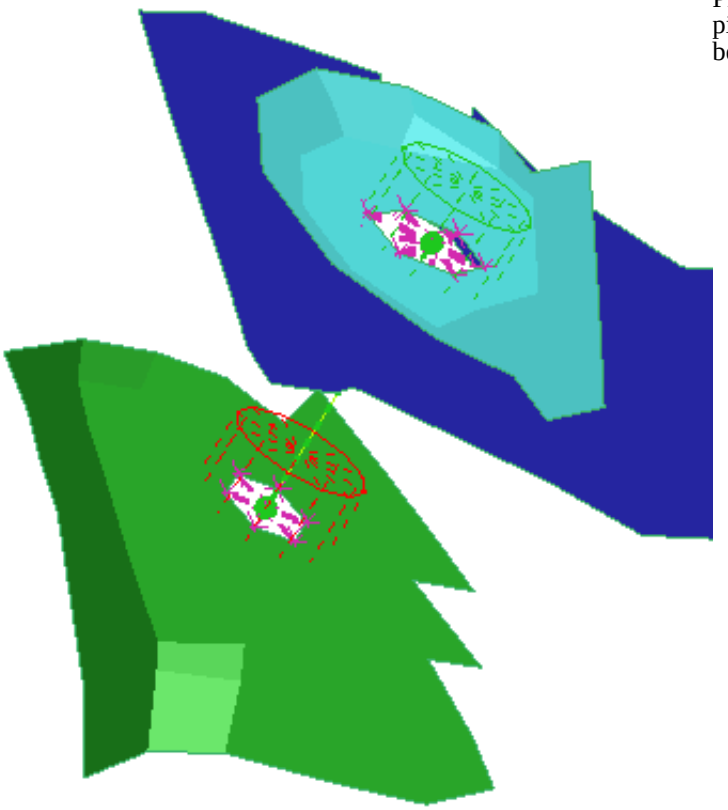
Two separate rigid bodies are made (NRB or rigid parts) and may be joined by deformable beam, optional rigid beam or revolute joint according to the method option.





Preview of two point patch type bolt with beam made by pick screen point.

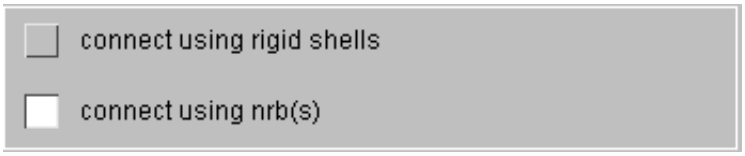
Note - multiple parts may included at each end



Preview of two point NRB type bolt made by picking a node on edge of hole with beam between rigid parts.

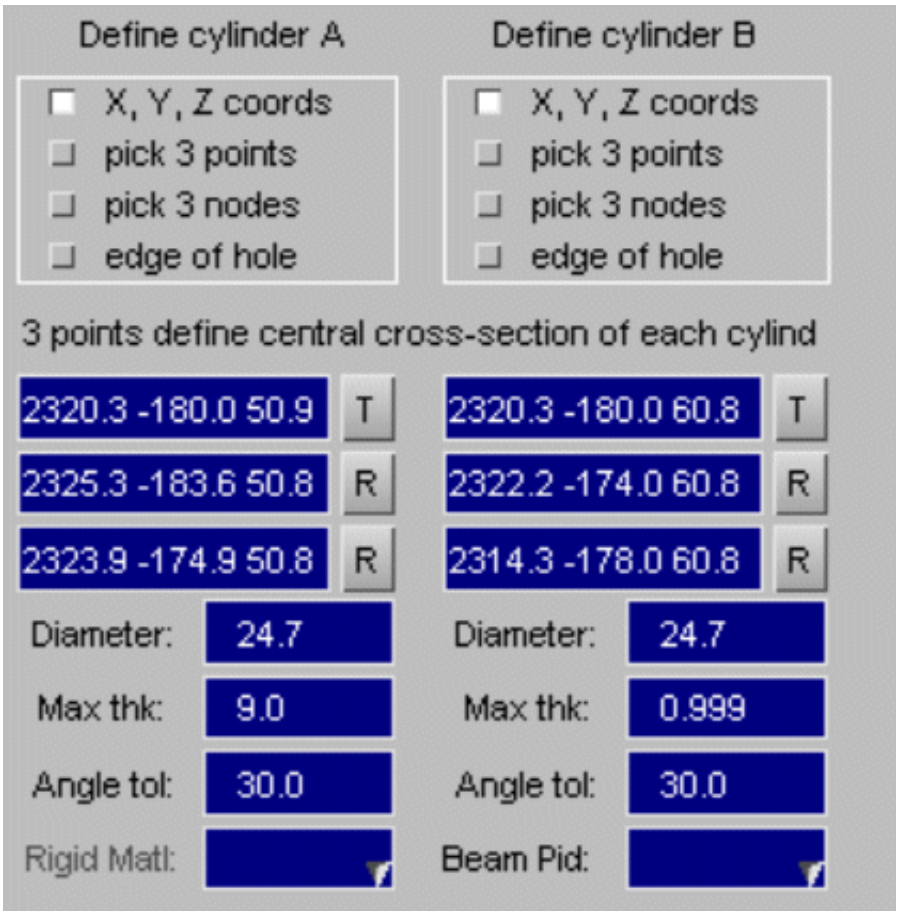
Creating generic two point bolts

These bolts are designed to give a generic method for two point connection. They may be formed using nodal rigid bodies or overlying rigid shells.

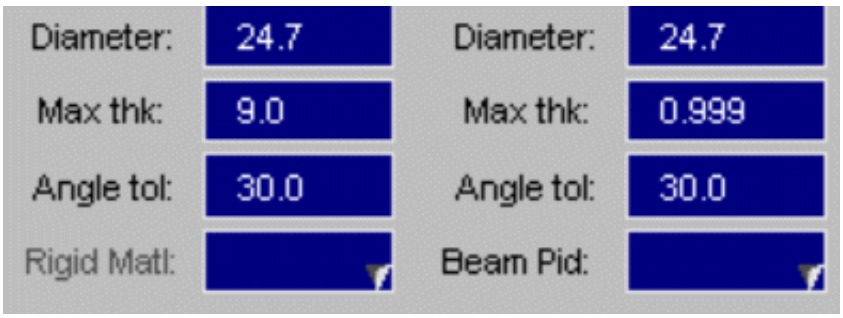


If no Beam Pid is specified a rigid connection will be made encompassing both ends. Otherwise a beam will be created between the centre points of each cylinder or a zero length beam if the section is of type discrete.

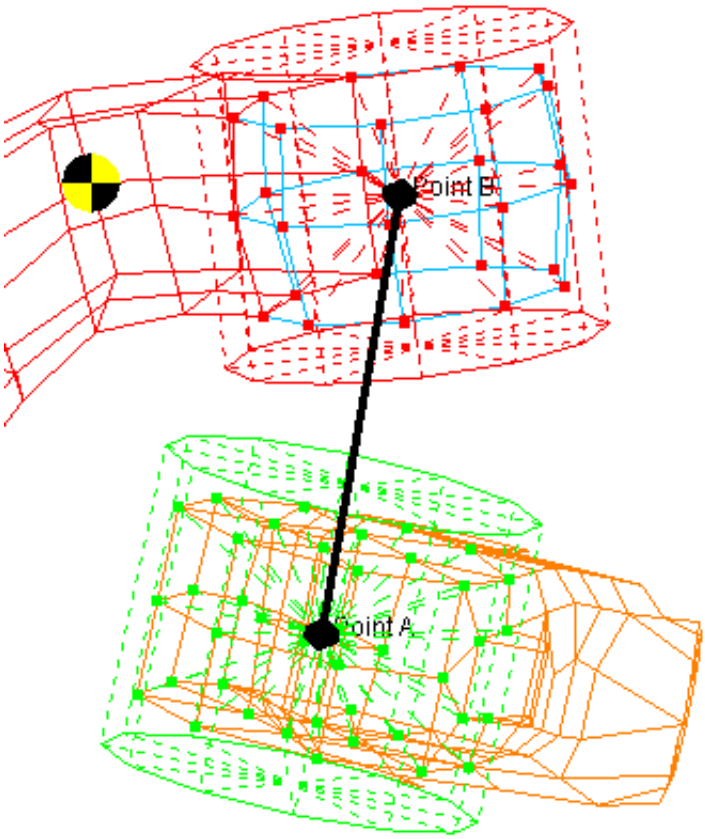
For generic bolts the layer definition is pre-determined by the create panel with the **Set layer A** and **Set layer B** functions. The method should then be chosen. With **edge of hole** a single pick on a hole edge will establish the centre of the hole P1 and the points P2, P3 such that P1P2 and P1P3 are non-parallel vectors which describe the central plane of the cylinder. Alternately, **pick 3 nodes** or **pick 3 points** may be used to establish the points or they may be typed in. Once established the coordinate may be adjusted by XYZ dragging **T** or rotation of the cylinders **R**.



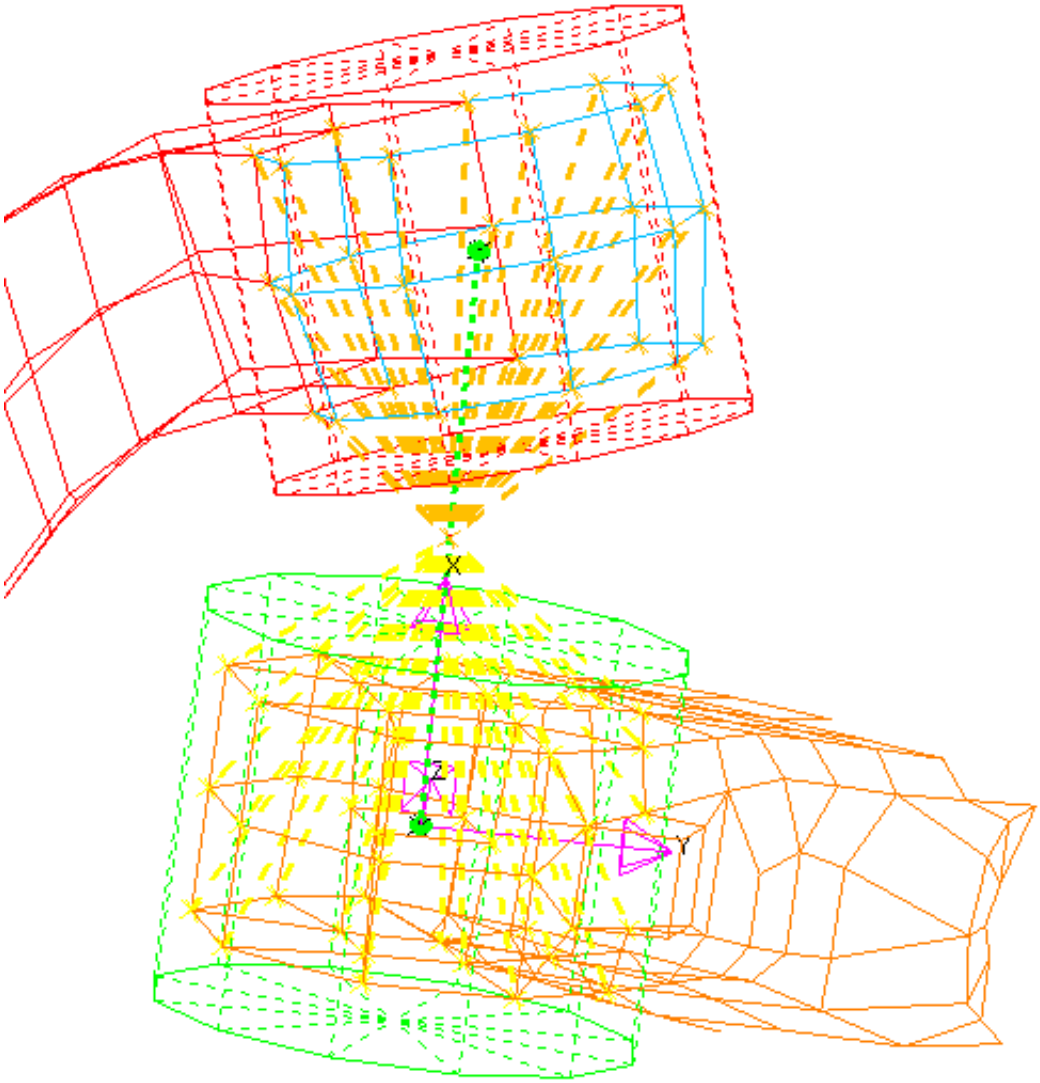
Once the orientation is correct, the diameters and lengths (thicknesses) of the cylinders can be increased. Also the angle tolerance which compares shell normal to cylindrical axis may be set.



Primer will then preview the prospective bolt. Adjusting the "recipe" (e.g. the diameters) should dynamically update the preview, if it does not **Refresh Preview Bolt** can be pressed. Once the bolt looks right it may be created by pressing **Apply**



In this example Part 612 is a discrete beam part. Consequently a zero length ELEMENT_BEAM_THICKNESS has been created wihich references *DEFINE_COORDINATE_SYSTEM to describe the base orientation of the bolt axis.



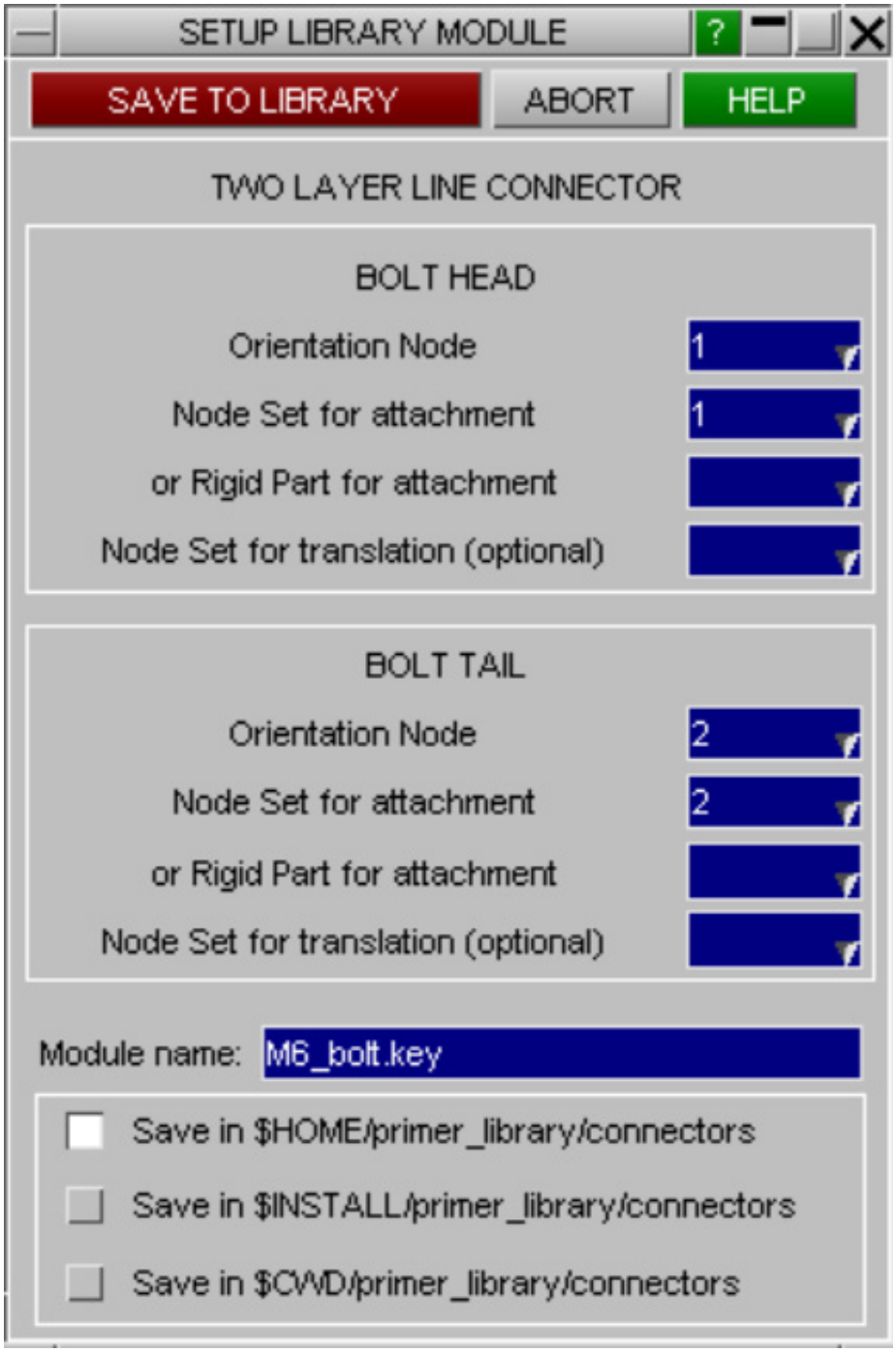
Creating library bolts

The library bolt is an FE model created by the user which represents a two layer line connection. This feature allows the user to create multiple complex bolts designed to his own specification.

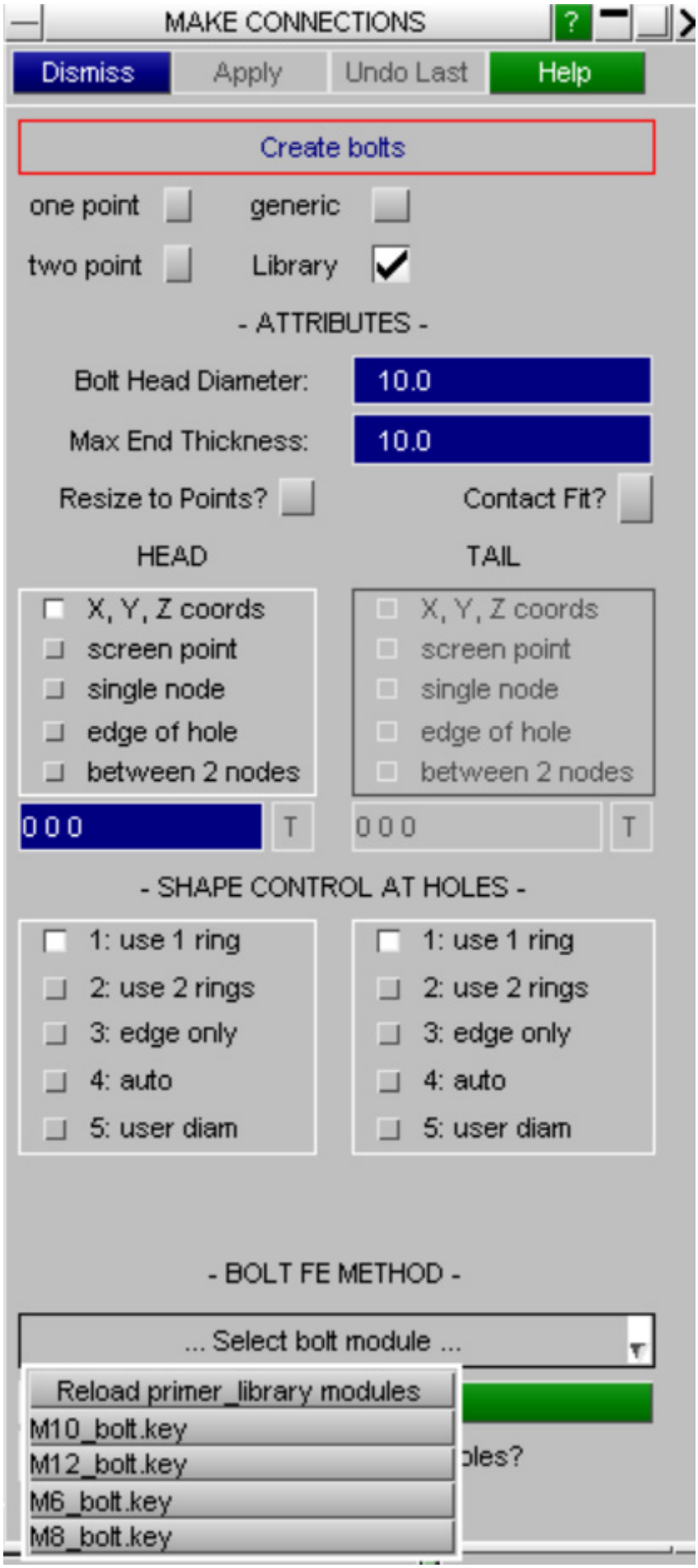
The model contains additional information needed to connect is to deformable shell panels in a bespoke *COMMENT. This is best set up by selecting a model and using the **library** function under connections

With this tool the user can select the head and tail nodes which will locate to the connection points, the nodes sets to attach by *CONTRAINED_NRB or the rigid parts to be connected by *CONSTRAINED_EXTRA_NODES. By default stretch will applied to the nodes which lie between the head and tail nodes and others will be translated. But the user may configure this by setting translation node sets. More detail is available under **HELP**.

Once set up the module can be saved with a unique name to CWD or HOME area (single user) or INSTALL area (all users). Whenever Primer is started all library modules will be loaded and available. Note - on windows machines if using CWD you will need to read a file to set cwd and then tick the **Library** option on the bolt creation panel.



Creation is very similar to two-point bolt with options for node/point picking and shape control, the first point will locate the head, the second the tail. A module must be selected from the library these are all listed in a dropdown..


MAKE CONNECTIONS
Dismiss
Apply
Undo Last
Help
Create bolts
one point
generic
two point
Library
- ATTRIBUTES -
Bolt Head Diameter:
10.0
Max End Thickness:
10.0
Resize to Points?
Contact Fit?
HEAD
TAIL
X, Y, Z coords
screen point
single node
edge of hole
between 2 nodes
X, Y, Z coords
screen point
single node
edge of hole
between 2 nodes
0 0 0
T
0 0 0
T
- SHAPE CONTROL AT HOLES -
1: use 1 ring
2: use 2 rings
3: edge only
4: auto
5: user diam
1: use 1 ring
2: use 2 rings
3: edge only
4: auto
5: user diam
- BOLT FE METHOD -
... Select bolt module ...
Reload primer_library modules
M10_bolt.key
M12_bolt.key
M6_bolt.key
M8_bolt.key

If **Resize to points** is set the bolt will be stretched so that the head/tail node coincide with the picked points.

Contact Fit will do the same and further adjust the bolt so there are no contact penetrations.

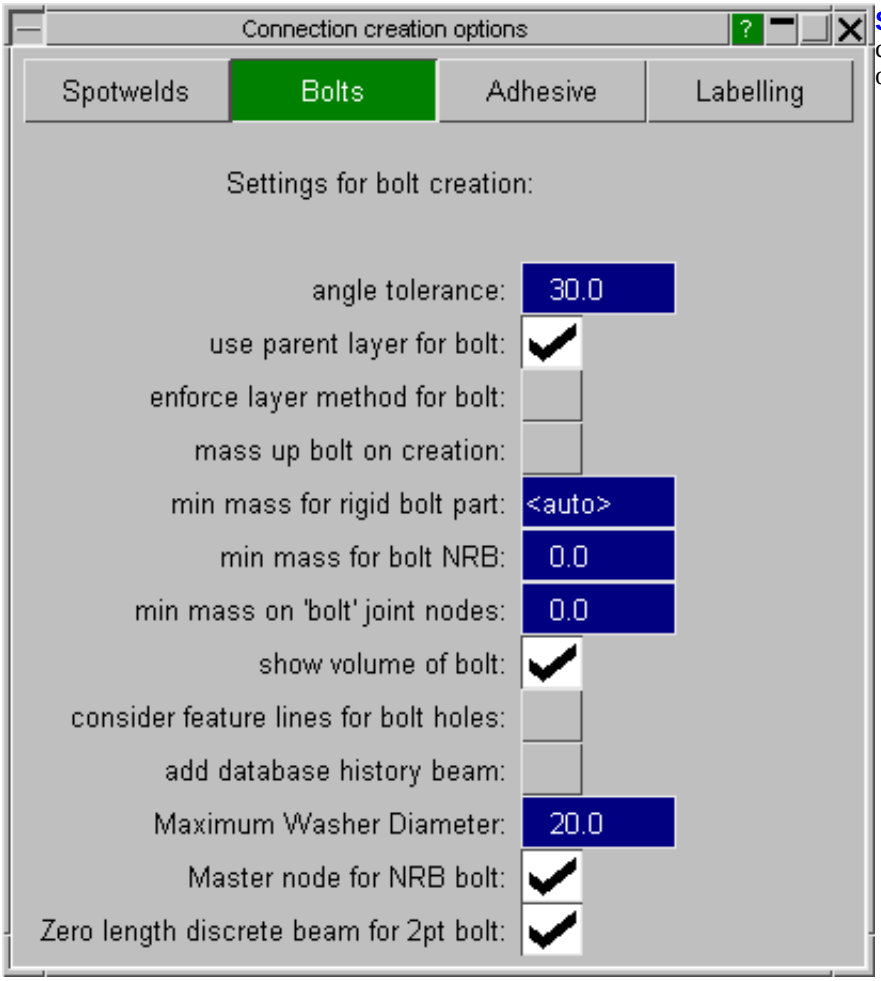
A *CONSTRAINED_NRB or EXTRA_NODE will be created to attach to deformable shell panels according to the information contained in the *COMMENT of the module.

As with standard bolts the library module is also supported by in the xml connection file so multiple bolts can easily be created in batch process. The unique module name should be given with path.

To avoid un-necessary creation of deformable parts, sections and materials, the titles of these are checked against those already in the model and if matched the extant card will be referenced rather than imported. The titles for these should therefore be carefully chosen in the module. When connections are deleted these parts will not be and may be left empty in the model.

```
<connection type="rigid">
  <method>bolt_module</method>
  <title>Bolt at X=2337.52 Y=44.5245 Z=68.85</title>
  <id>1</id>
  <coord x="2337.518311" y="44.524529" z="68.849998" />
  <coord2 x="2337.518311" y="44.524529" z="78.849991" />
  <diameter>12.500000</diameter>
  <length>8.999993</length>
  <shape>edge only</shape>
  <angle_tol>30.000000</angle_tol>
  <shape2>edge only</shape2>
  <angle_tol2>30.000000</angle_tol2>
  <resize>0</resize>
  <fit>1</fit>
  <module>M6_bolt.key</module>
  <layer type="PART_ID">
    <part id="102" />
  </layer>
  <layer type="PART_ID">
    <part id="103" />
  </layer>
</connection>
```


Bolt options



Settings ... on the main connection create panel gives access to global options for bolt creation

min mass on 'bolt' joint nodes - will add mass to the nodes of the joint to meet the target value (if necessary). Increasing mass of a joint increases its stiffness and stability.

minimum 'bolt' joint length - will increase the length of the revolute joint (if necessary) to meet the target value

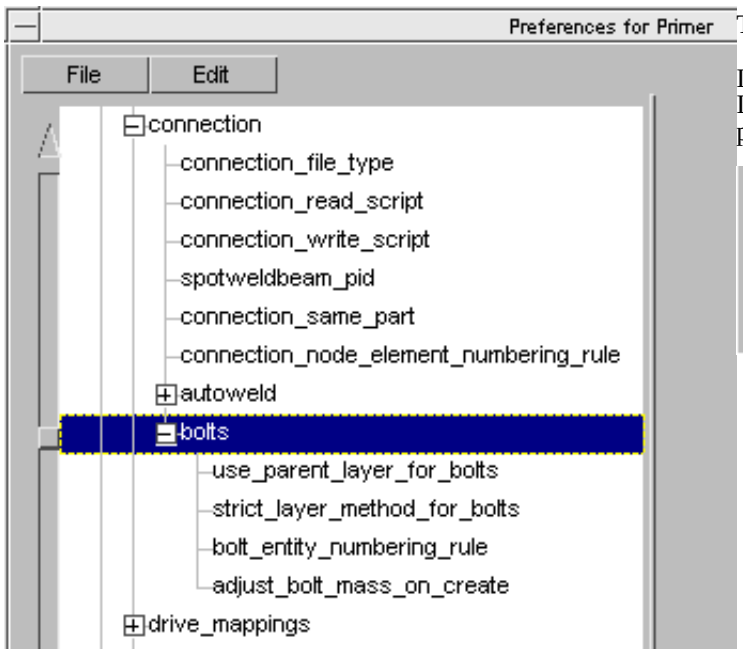
sketch volume of bolt - PRIMER will sketch the cylinder or cylinders (for 2 pt bolts) which show the nominal geometry of the bolt. This is useful for adjusting the diameter.

add database history beam - on creation of bolt, PRIMER will automatically make a Database History Beam id using the title of the bolt.

use parent layer for bolt FE - creation normally will put all FE into the current layer. However, for bolts if this option is set, bolts will try to use the parent layer. If parent panels are all in the same include, all FE will go into that include. If they are in different includes, shells/parts/ materials will be put with the overlaid parent part and any other items will go into the current include. On the connection table the layer of the connection itself and layer(s) of the FE data can be displayed using **view ...** conx include/FE include.

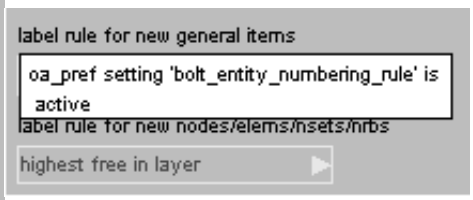
enforce layer method for bolt FE - if numbering rule is set to a layer type method and this option is set, it will block creation of the bolt if include file number ranges have not been set. Without this setting, PRIMER will create the bolt using highest+1 in model when number range is missing.

entity numbering rule - by default the same rule applies to welds/adhesive and bolts. The default is *model highest+1* but oa_pref **connection_node_element_numbering_rule** can modify this. For bolts only, oa_pref **bolt_entity_numbering_rule** will always over-rule any other setting.



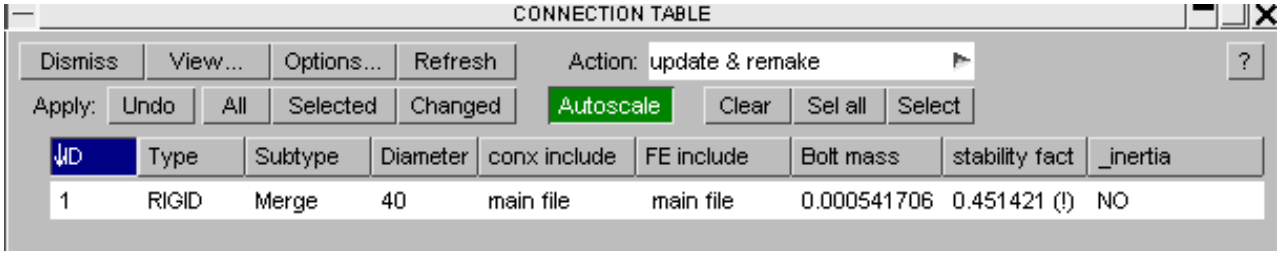
These may be configured using oa_pref settings

If **oa_pref bolt_entity_numbering_rule** is set. It will overrule the label rule set in the options panel which is therefore greyed.



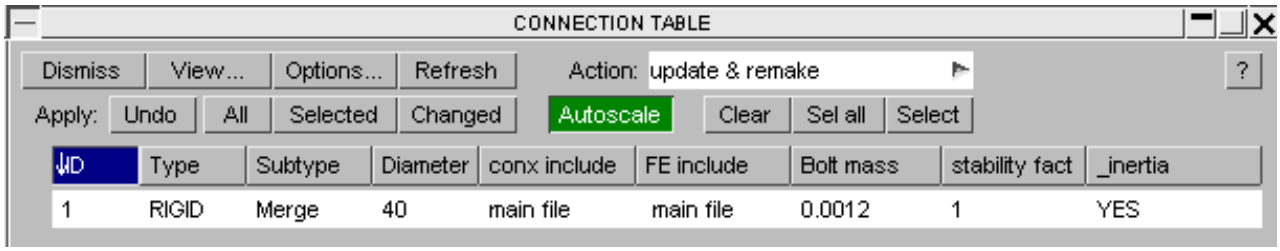
mass up bolt on creation - small merge type bolts can lead to stability problems in mass scaled models. The problem arises because added mass on the deformable elements which attach to the rigid bolt gets lost. The mass of the bolt itself needs to be sufficient to compensate for this loss. By default PRIMER will automatically calculate the mass required, however user may set a target mass which will be used in preference (check > options > rigid). Nodal rigid bodies are implicitly stable in LS-Dyna so there is no stability criterion for these type of bolts, however user may set a target mass for these too.

If this option is active and the requested mass exceeds the actual mass, PRIMER will convert the master part (merge type bolt) or nodal rigid body to an _INERTIA definition with mass properties appropriately scaled up. If it is not the mass of the bolt created may be too small for stability. The table will give the mass and the factor which should be >= 1.0 for stability.

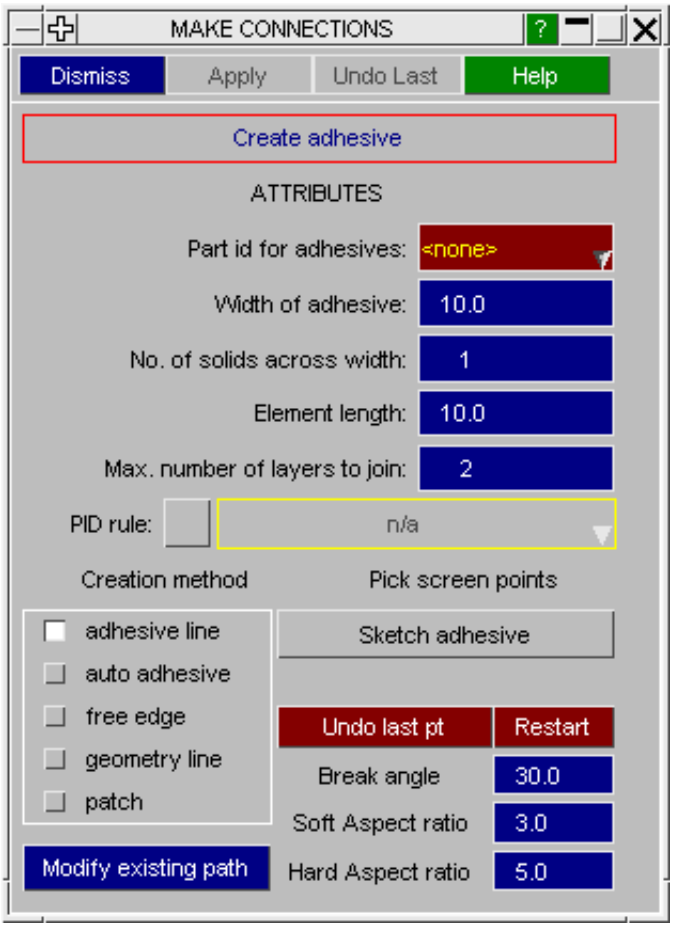


ID	Type	Subtype	Diameter	conx include	FE include	Bolt mass	stability fact	_inertia
1	RIGID	Merge	40	main file	main file	0.000541706	0.451421 (!)	NO

The mass can be corrected by re-making the bolt with **mass up bolt on creation** active or by performing a model check and auto-fixing the master part/nodal rigid body. The table will then show that _INERTIA is being used.



Creating adhesive



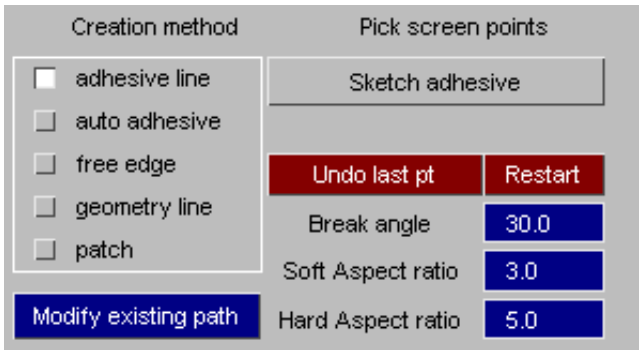
The adhesive creation panel is shown on the left. Adhesive is either defined as a constant width run of solid elements or a patch of element solids created between the panels you wish to connect.

Constant width adhesive lines are created by defining a line along the panels you wish to join. The line can either be defined manually by clicking on the screen or through automatic methods, similar to automatic spotwelding. Patches of adhesive are created by defining areas of shells to use as a source for creating solid elements.

Before any adhesive can be created, the part ID that new adhesive solid entities are put in needs to be specified.

Various inputs are defined to determine the final adhesive run. These include the width of adhesive, number of solid elements across the width, element length and various inputs to aid the adhesive definition when going round corners. These are explained in more detail below.

Generic adhesive buttons

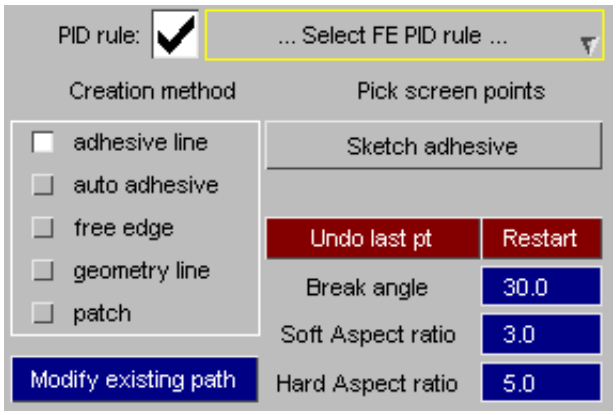


The available methods of creation are **adhesive line, auto adhesive, free edge, geometry line** and **patch**. These can be selected on the radio buttons on the left. **Sketch adhesive** is used for sketching the proposed adhesive before creating it. This is particularly useful when creating long runs of adhesive. **Modify existing path** is used to modify the path points on an existing adhesive run.

Break angle controls how the defined path is split into sections. PRIMER will also try to meet the **soft aspect ratio** for the solid elements created, however this will not prevent creation. If a solid is found to fail the **hard aspect ratio** for the solids, then that solid will not be created. Note the aspect ratio check does not take into account the through thickness of the solid element, i.e. it is just in-plane aspect ratio.

When creating adhesive PRIMER will create all the solid elements it can. Some solid elements may not be possible (for example due to holes in the mesh). PRIMER will skip over these sections and create what it can. Because of this it is useful to use the **sketch adhesive** button as you are creating the adhesive to see what will be made and what will not.

When **adhesive line** is selected, the **undo last point** and **restart** buttons are available. These can be used as you are defining the adhesive path to undo points you have created and to start again.

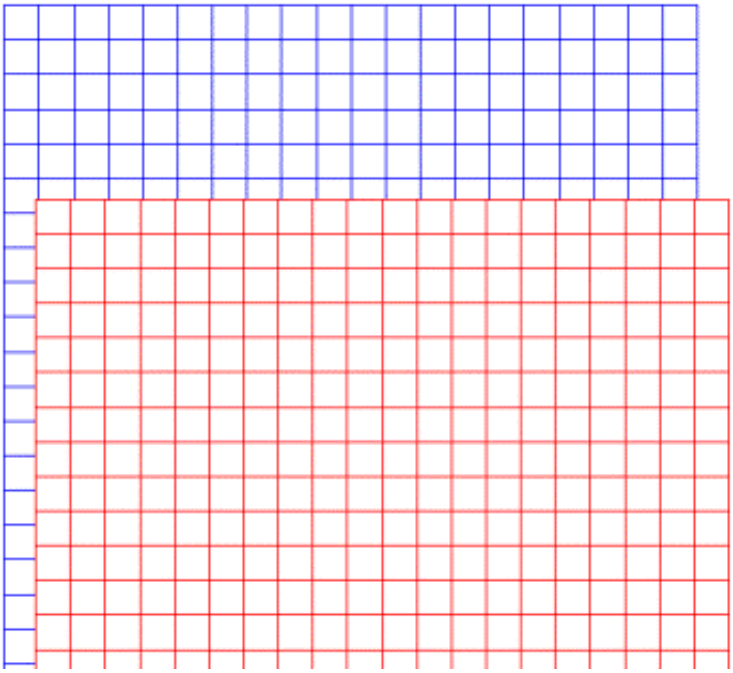


The **part id for adhesives** is the part that the created adhesive solid elements will end up in. This must be defined before adhesive can be created.
The **width of adhesive** is the width across the run of adhesive (not applicable for patch creation method).
The **number of solids across the width** of the adhesive can also be specified on this panel (not applicable for patch creation method).
The **element length** is the desired size of the solid elements along the length of the adhesive path (not applicable for patch creation method).
You can also increase the **maximum number of layers to join** from the default of 2.

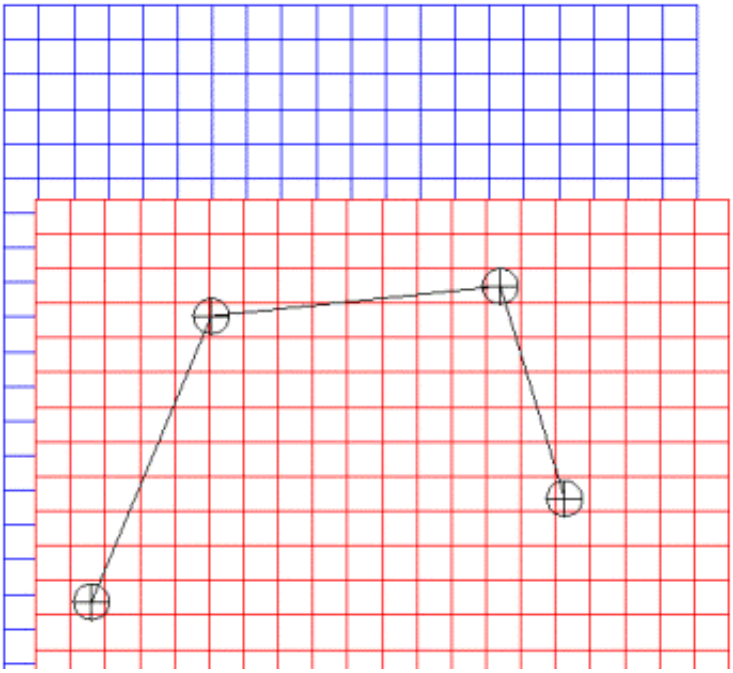
A **PID rule** can also be used to define the PID used for the adhesive layer(s). This is documented in the PID rule section for spotwelds.

Adhesive line creation method

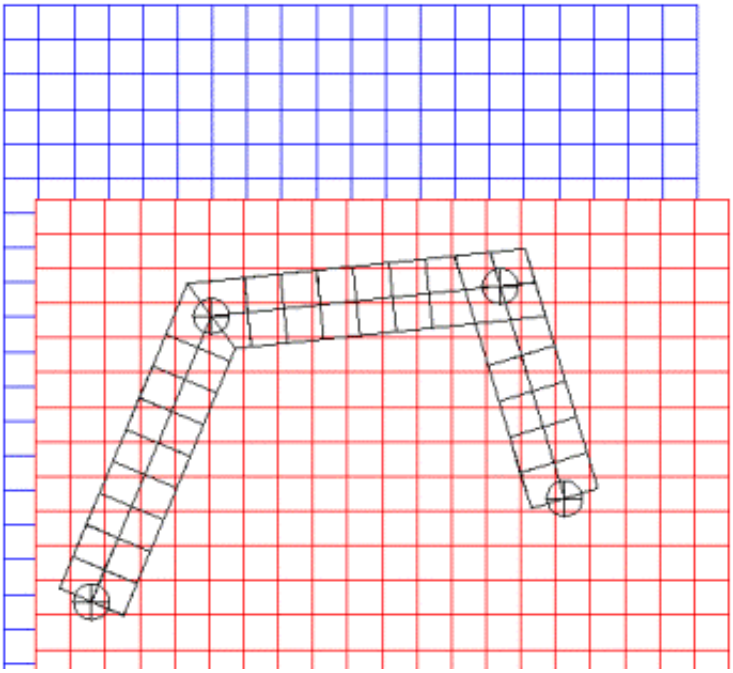
The following example shows how to create a simple run of adhesive using the adhesive line creation method.



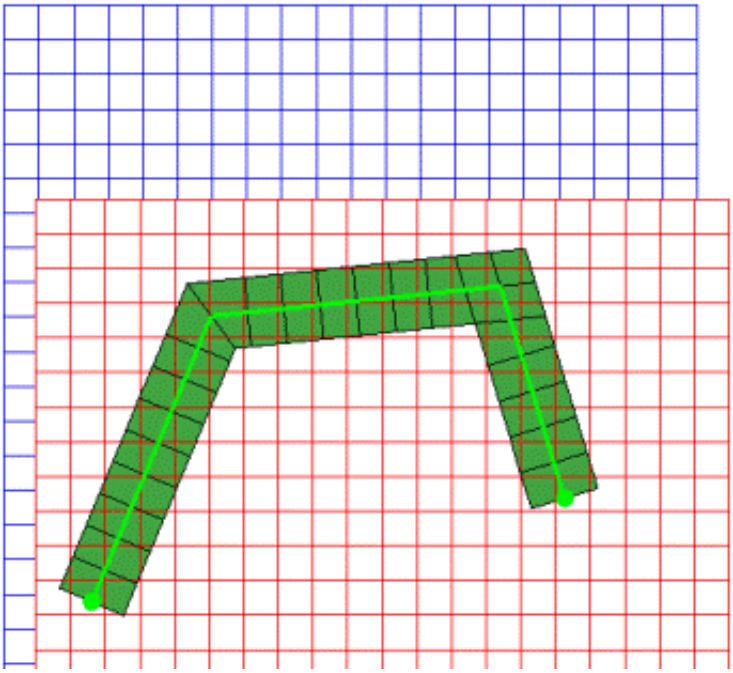
Say you want to join together two panels with an adhesive run using the **adhesive line** method. Remember, before you can create adhesive you must choose the part ID you wish the adhesive solids to end up in, and also the shells you wish to connect. You should also set your **adhesive width**, **number across width** and **element length** values.



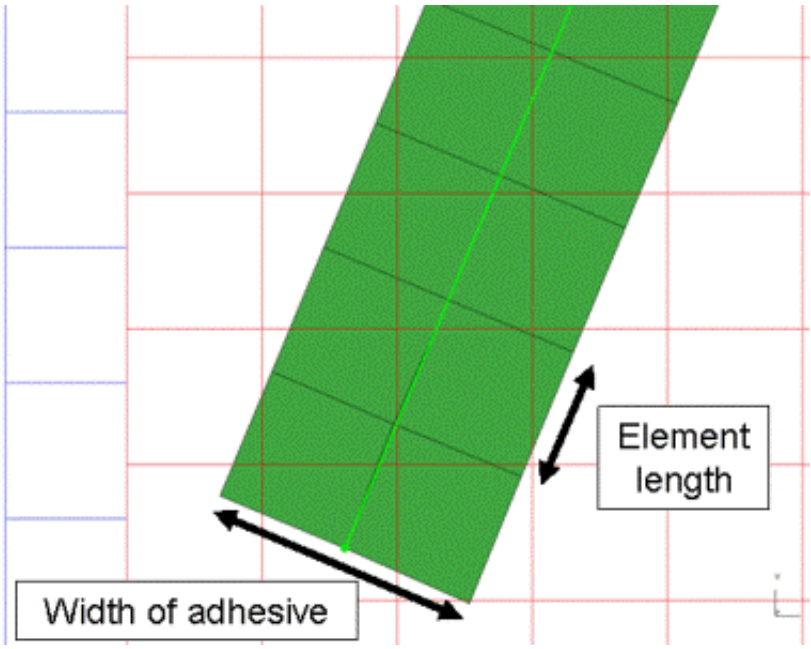
With **adhesive line** selected, click on the panels you have chosen to join to define points in the run. PRIMER will sketch the points and the line as you go along.



Clicking on the **sketch adhesive** button as you go along will allow you to preview the adhesive before actually creating it.



After you are happy with your defined path, clicking Apply will create the solid elements and create the connection entity. The connection entity is drawn as two blobs connected by a path line. The colouring of this connection entity is dependent on connection status and follows the same scheme as spotwelds.

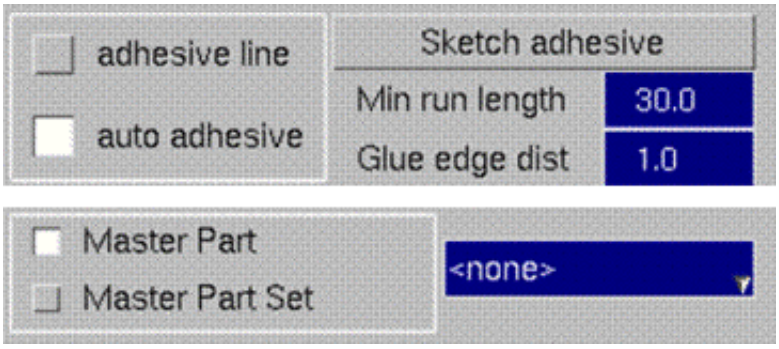


In the example shown, the **width of adhesive** has been defined as 20mm. The **number of solids across the width** has been defined as 2. The **element length** along the length of the adhesive has been defined as 10mm.

Auto adhesive creation method

Auto adhesive allows the user to automatically create adhesive runs between selected panels. The method works in a similar way to the automatic spotwelding feature described above.

The following additional buttons/inputs are available for auto adhesive creation:

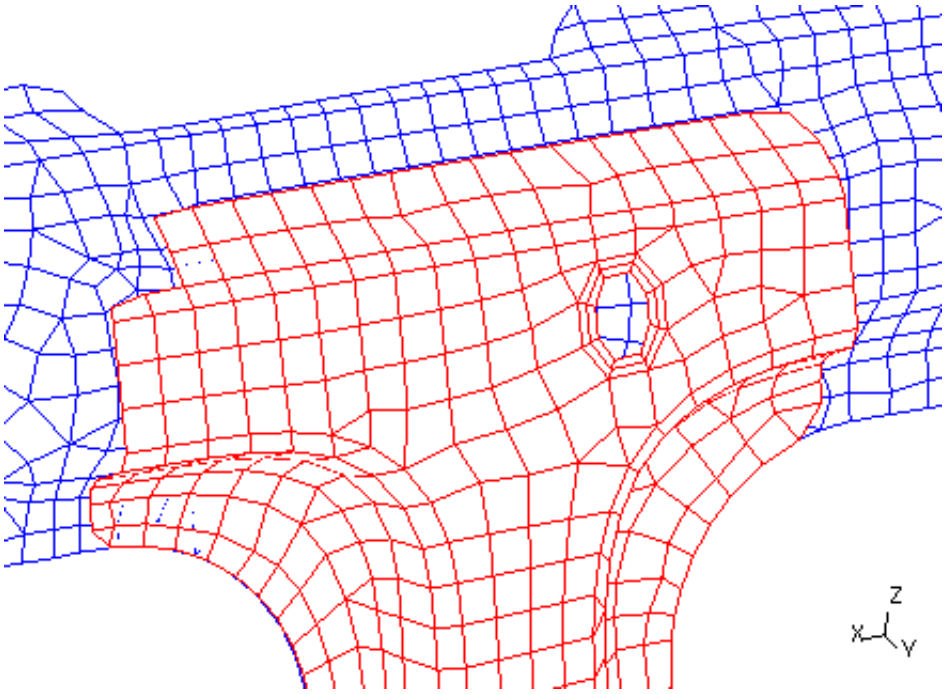


Min run length: any free edge runs that are less than this amount are discarded.

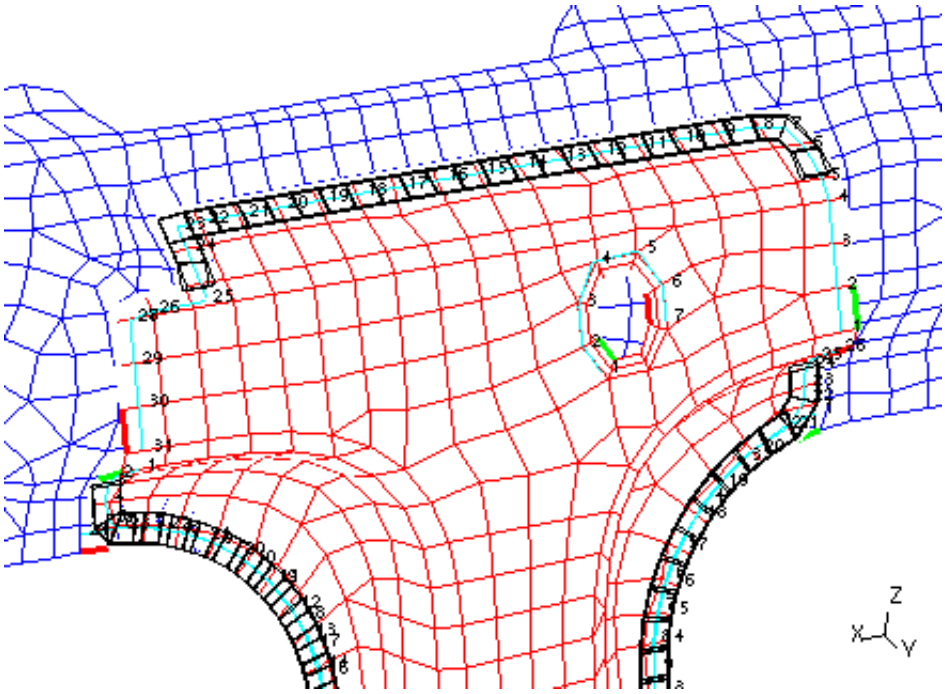
Glue edge dist: distance between the edge of the panel and the edge of the solid elements.

Master Part: a master part or a master part set can be used to specify which panel(s) are used to base the auto adhesive on. If specified, only free edges on the master part(s) are used to construct adhesive paths.

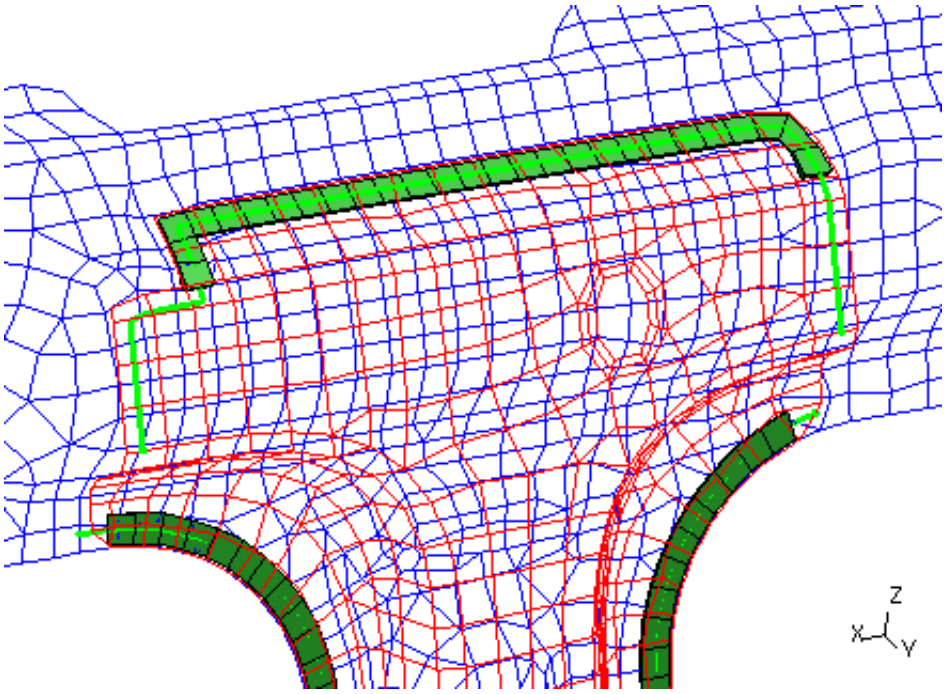
Without a master part selected, PRIMER will attempt to create adhesive from all free edges on the shells selected for connection.



To the left is an example of auto adhesive creation. In this case, the **adhesive width** is set to 10mm, the **number of solids across the width** is set to 1 and the **element length** is set to 10mm. The **glue edge distance** is set to 1mm, and the front part (red part) is set as the **master part**, meaning it is the part used to determine free edges and hence adhesive runs.



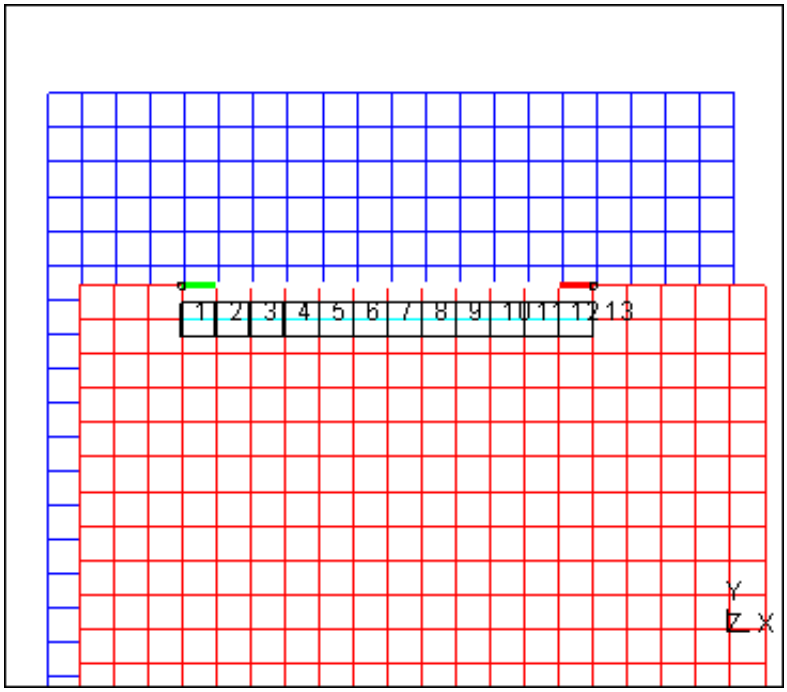
After setting the desired parameters, the **sketch adhesive** button can be used to preview the adhesive that PRIMER will create. The user can now change any of their inputs now and re-sketch the adhesive until they are happy with what will be created.



Clicking **Apply** will now create the adhesive. Note that **modify existing path** can be used to modify any runs created.

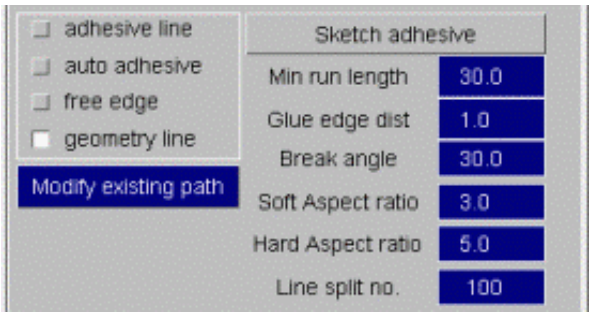
Free edge adhesive creation method

This is similar to the auto-create method in that it is based on free edges, but here the free edges are defined by the user rather than automatically determined by PRIMER.



The free edge length is defined by clicking on two nodes along the free edge. PRIMER will determine all the nodes along the free edge between the two selected nodes. Adhesive created using this method will follow the free edge. The distance between the edge of the adhesive solids and the free edge can be specified using **Glue edge dist**.

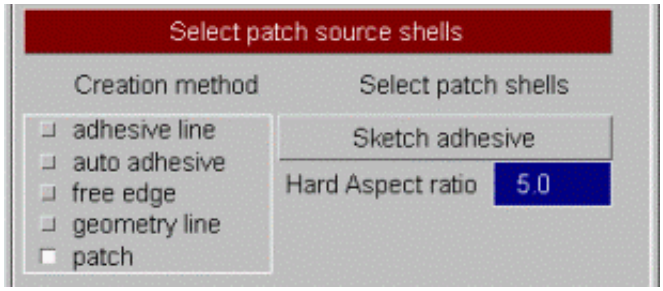
Geometry line creation method



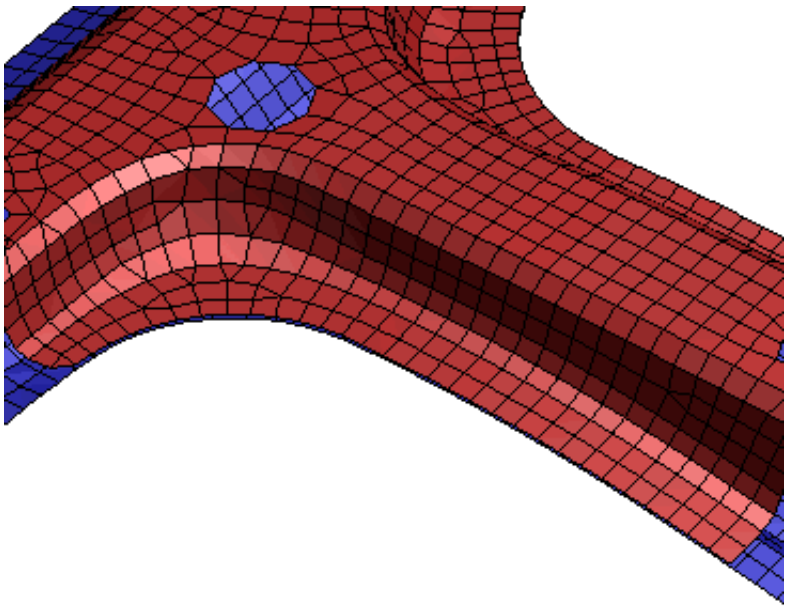
The **geometry line** creation method can be used to create adhesive runs from geometry lines that exist in any model in PRIMER. Use **Line split no.** to specify how many increments the line is split into when creating the information for the adhesive path.

Patch adhesive creation method

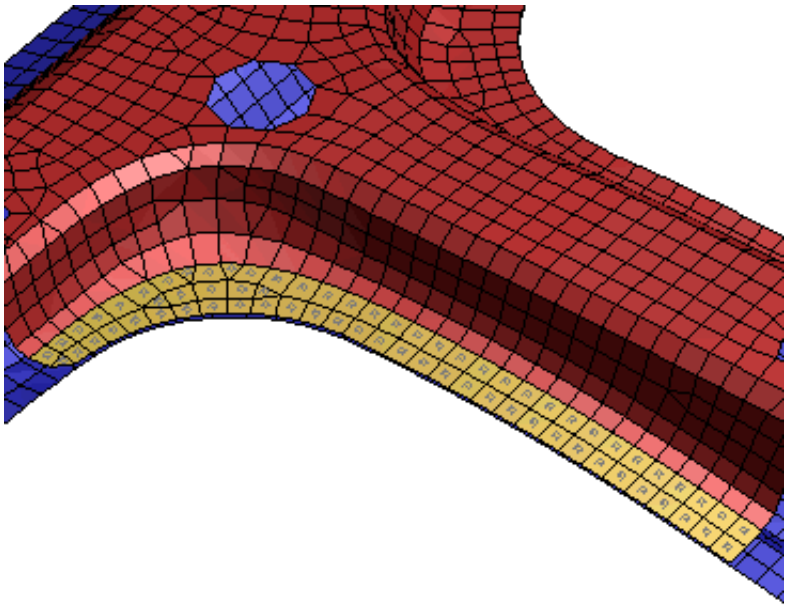
The patch creation method should be used if you want to create adhesive based on an area rather than a constant width line. Source shells are selected and used to project between the panels to create solid elements. The source shells do not need to be in the model you are creating adhesive in, i.e. the source shells can be meshed "ribbons" of adhesive that exist in a separate model.



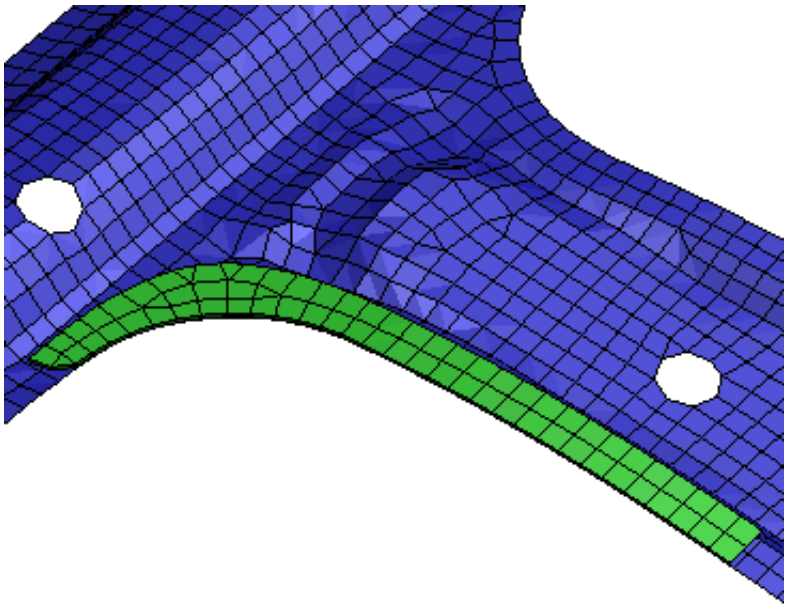
The **patch** creation method can be used to create adhesive from shells that exist in any model in PRIMER. Use **Select patch source shells** to select the source shells for creating solid adhesive elements.



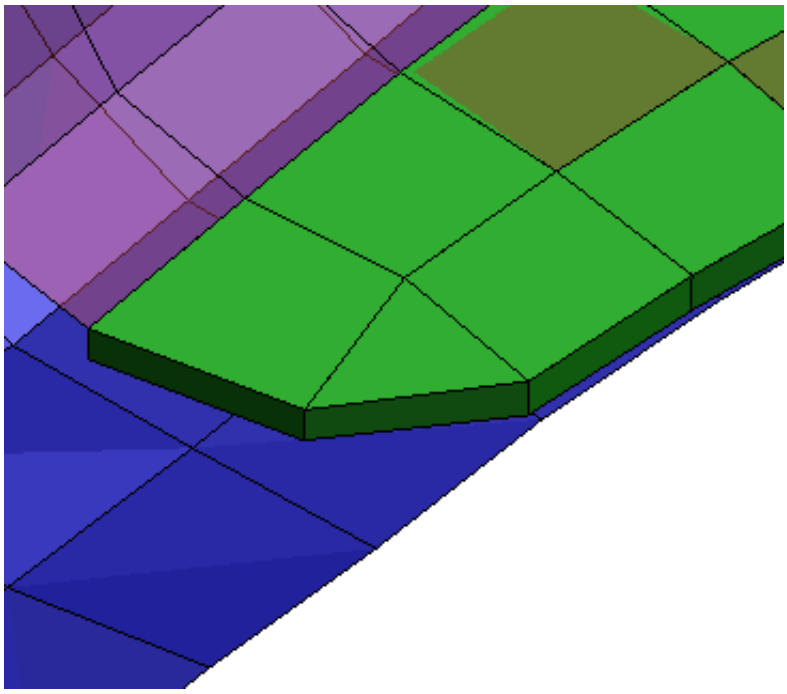
In this example we want to create a patch of adhesive covering the flange shown between the red and blue part. The flange is not constant width so the patch method is appropriate.



Click on **Select patch source shells** to select the shells along the flange (highlighted in this example). Note in this example the source shells are within the model we are creating adhesive in. The source shells could also be in a separate model, for example if your adhesive information has come direct from CAD and been meshed as shell ribbons.



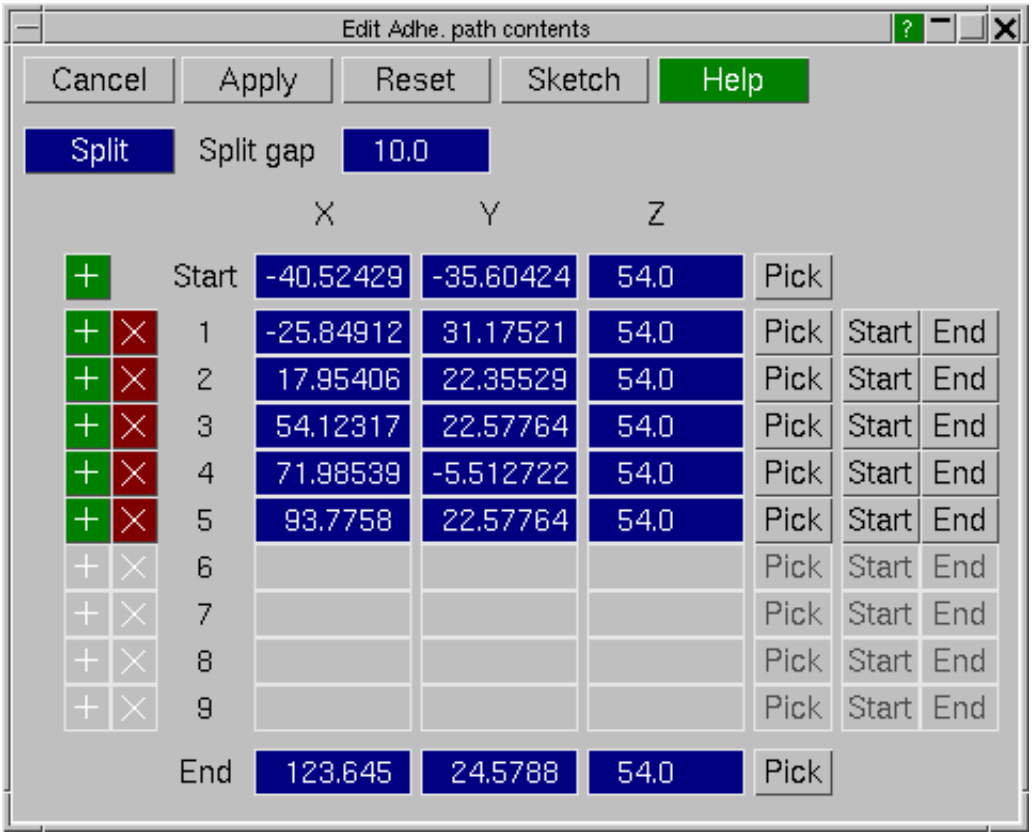
Click **Apply** to create the adhesive patch. The image to the left shows the created solids with the red part blanked. As with other connection types, a PRIMER connection entity has been created as well as the solid elements. The colour of this connection entity gives the status of the connection entity (for example an orange colour means the solids are not tied to the surface). The existence of the connection entity means the adhesive patch can be easily modified/reprojected in the connections table.



A close-up showing the solids created between the panels.

Modifying the adhesive path

The path of an adhesive line can be modified by clicking on the **Modify existing path** button in the create panel. The path can also be modified in the same way through the connections table (see section 6.10.2). After clicking the button, you select the adhesive you wish to modify, and the following panel will appear.

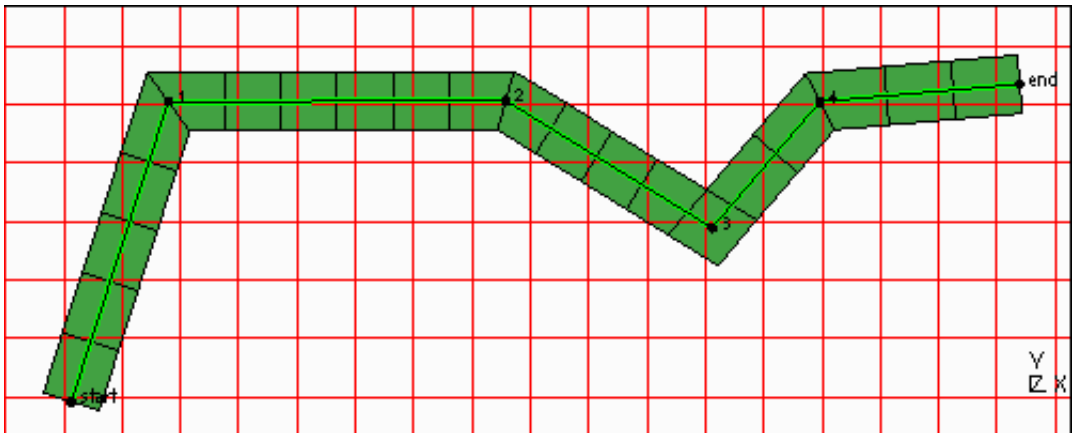


The path modification panel displays the adhesive path information. The coordinates of the start and end points are displayed, as well as the in-between path points. Through this panel it is possible to carry out the following functions:

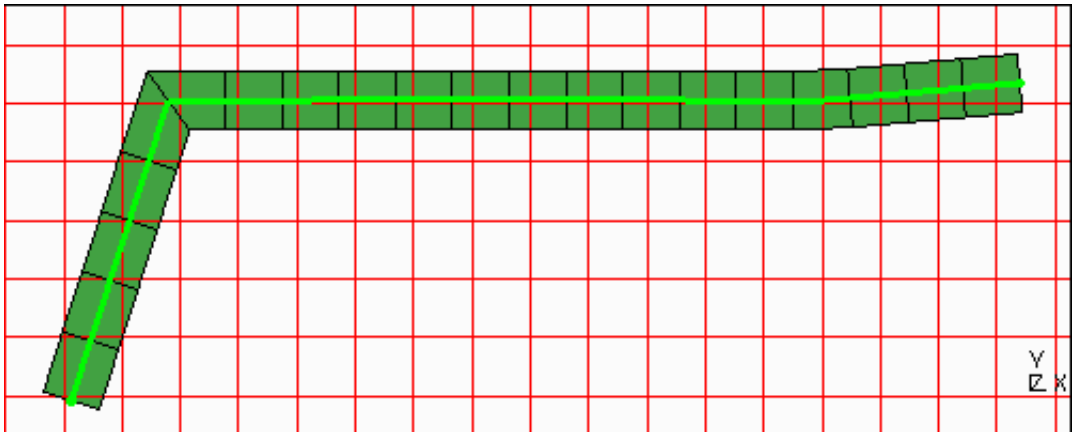
- Add or remove path points to the adhesive run.
- Modify the position of point on the adhesive path, either by typing in new coordinates or by clicking on the

screen.
- Setting any of the existing path points to a new start or end point.
- Splitting the path at any point along the adhesive run with a defined gap.

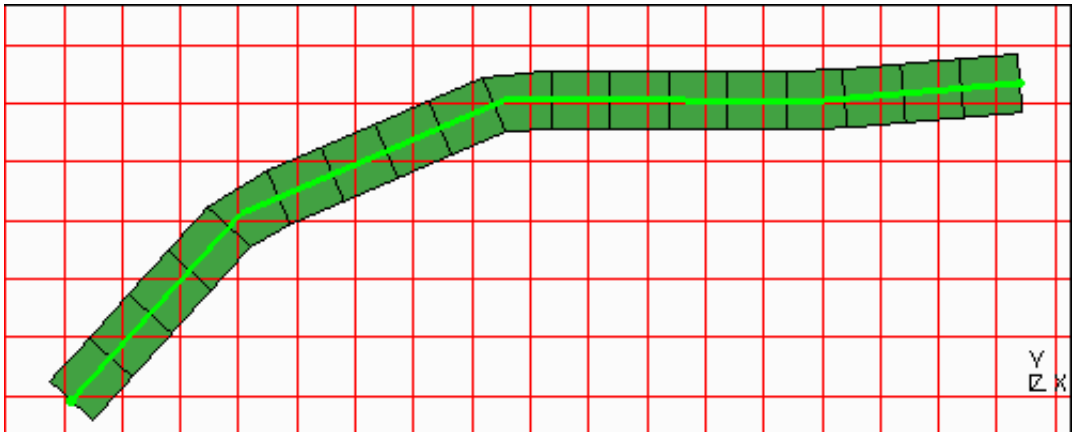
To demonstrate some of these features, take the following example:



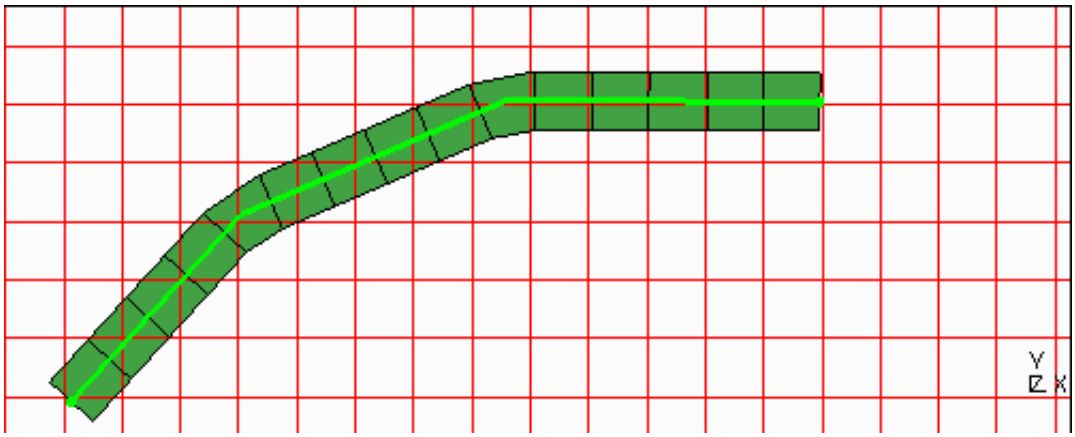
The above example shows a typical adhesive run. Clicking **Sketch** in the modify path panel will sketch the path points with numbers next to the points corresponding to the numbers shown on the panel. Note, you can return to the original path data by clicking on **Reset** (must be done before clicking on **Apply**).



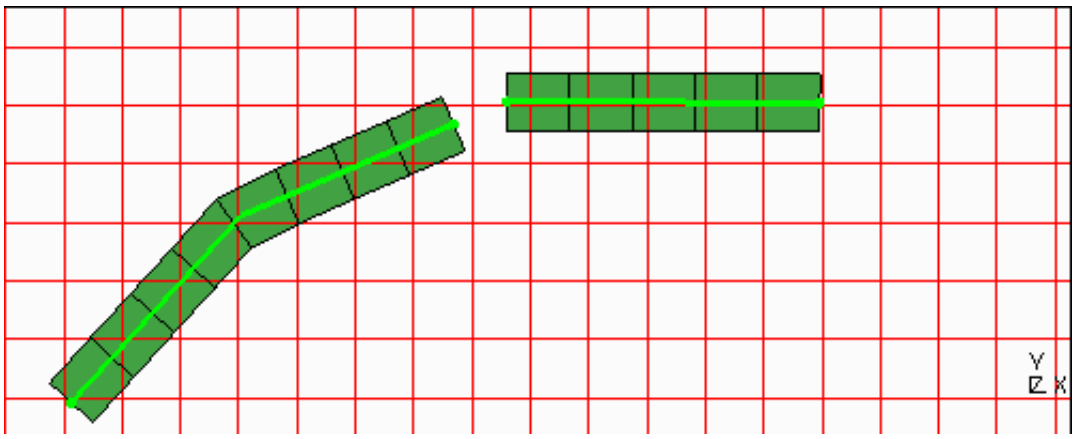
By clicking the red **X** button next to any of the points, that point will be removed. Clicking **Apply** will remake the adhesive run without that point. The image above shows the 3rd point removed. Similarly a point can be added by clicking on the green **+** button next to any of the points. This will create a new point after the point you clicked on. You can then select the position of the point by either typing in the coordinates or by clicking on **Pick** next to the new point, and then choosing a point on the mesh.



You can modify an existing point either typing in the coordinates or by clicking on the **Pick** button next to the point, and then choosing a point on the mesh.



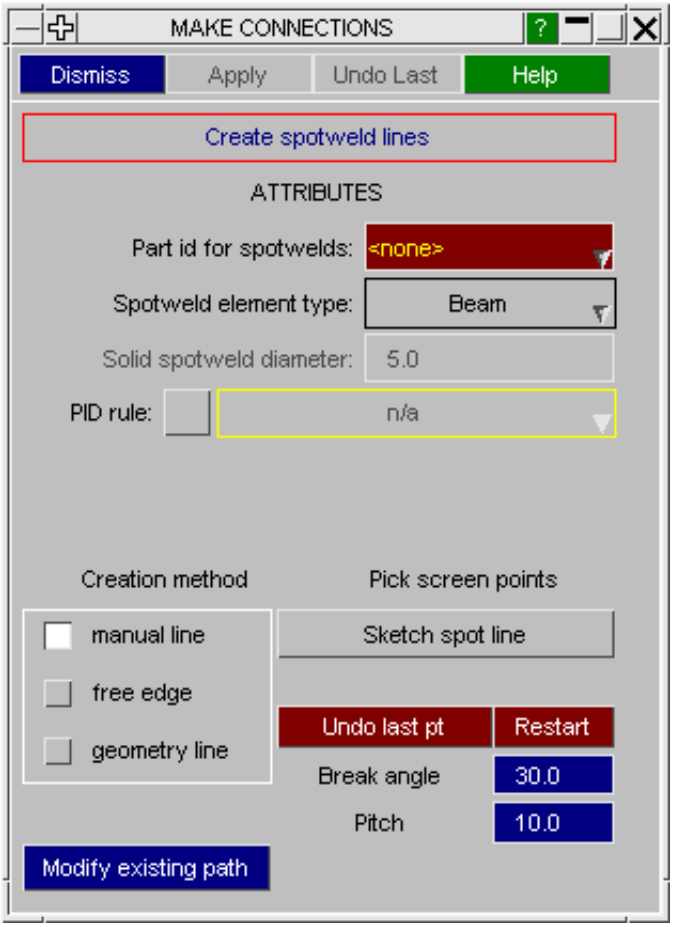
You can set any of the path points to the new start or end points by using the appropriate **Start** or **End** buttons next to the path point in the panel.



You can split the adhesive path by clicking on the **Split** button on the adhesive path panel, and then clicking on a node on the adhesive run. PRIMER will split the path at this point with the gap specified in the adhesive path panel. Note that this operation cannot be undone. After the split, the path information retained on the path panel is for the first of the two resulting adhesive runs.

Creating Spotweld Lines

Spotweld line entity types are an extension to individual spotwelds where you can create lines of spotwelds which are associated with one connection entity type. This is beneficial as the connection entity can be easily modified to have a different pitch, or to follow a new free edge should the part the connection attaches to be re-meshed.



The spotweld lines creation panel is shown on the left. Spotweld lines are defined as a run of beam/solid spotwelds elements created between the panels you wish to connect.

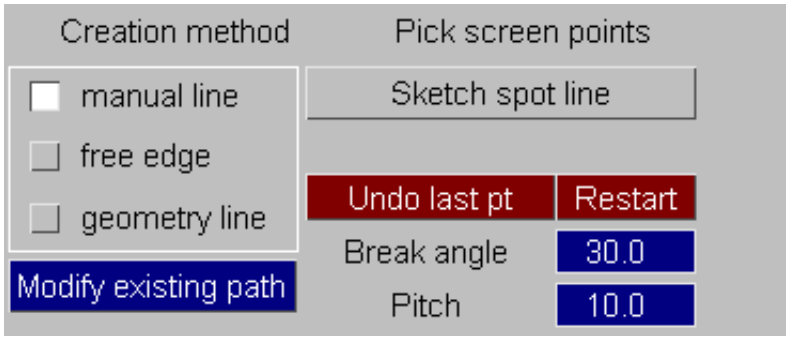
Spotweld lines are created by defining a line along the panels you wish to join. The line can either be defined manually by clicking on the screen or through automatic methods, similar to automatic spotwelding.

Before any spotweld line can be created, the part ID that new beam/solid entities are put in needs to be specified. If creating solid spotwelds you will also need to specify a diameter (default 5.0).

Various inputs can be defined to determine the final spotweld line run. These are explained in more detail below.

A **PID rule** can also be used to define the PID used for the beams/solids created for each layer. This is documented in the PID rule section for spotwelds.

Generic spotweld line buttons



The available methods of creation are **manual line, free edge** and **geometry line**. These can be selected on the radio buttons on the left. **Sketch spot line** is used for sketching the proposed spotweld line before creating it.

This is particularly useful when creating long runs of adhesive. **Modify existing path** is used to modify the path points on an existing adhesive run.

Break angle controls how the defined path is split into sections. The **Pitch** input is where you specify the pitch of the spotwelds along the path.

When creating spotweld lines PRIMER will create all the beam/solid spotwelds it can when progressing along the run path. It may not be possible to create some spotwelds along the path (for example due to holes in the mesh).

PRIMER will skip over these sections and create what it can. Because of this it is useful to use the **sketch spot line** button as you are creating the connection to see what will be made and what will not.

When **manual line** is selected, the **undo last point** and **restart** buttons are available. These can be used as you are defining the path to undo points you have created and to start again.

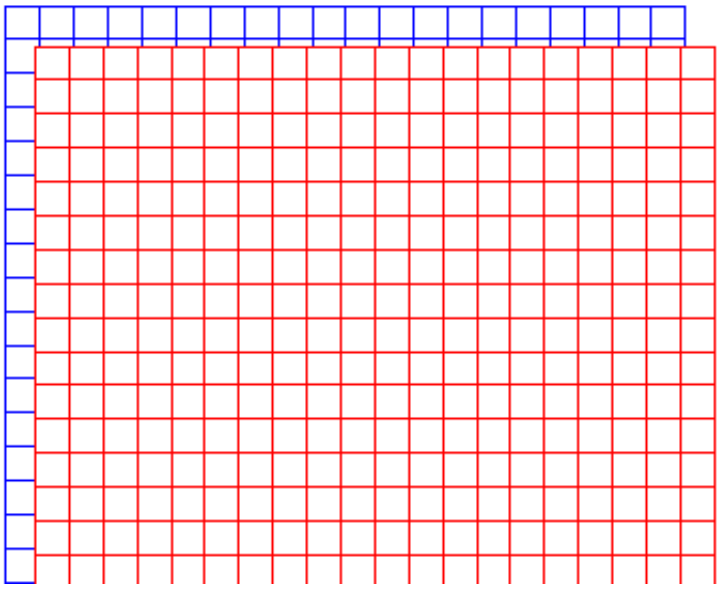
For information on specifying the part ID for spotwelds, the spotweld element type and diameter, please refer to previous **Choosing the spotweld element type** section.

Manual line creation method

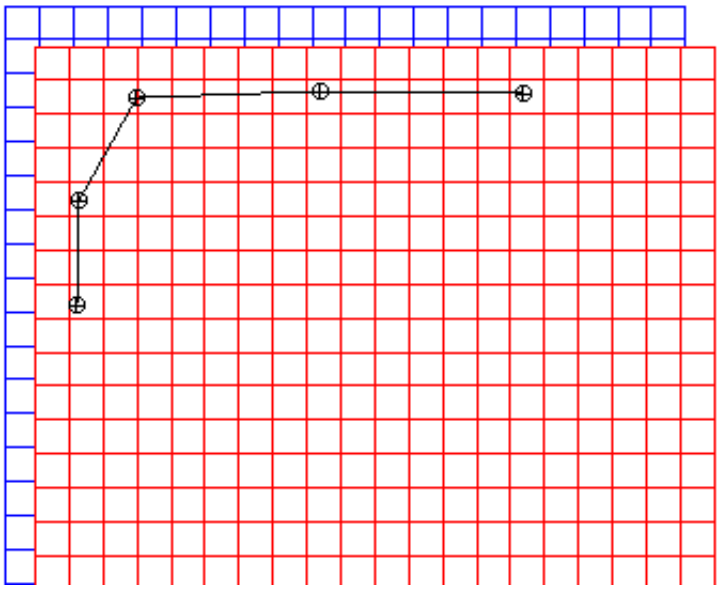
The following example shows how to create a simple run of spotweld using the spotweld manual line creation method.

Say you want to join together two panels with an spotweld line run using the **manual line** method.

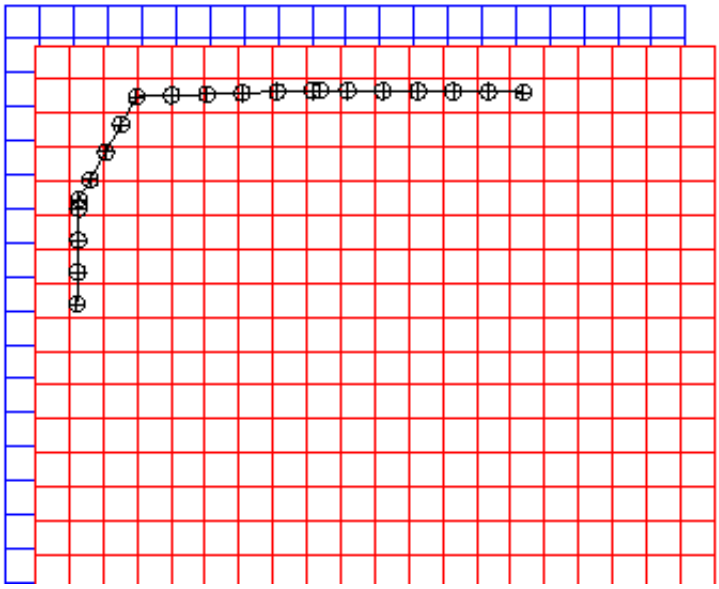
Remember, before you can create spotweld lines you must choose the part ID you wish the beams/solids to end up in, and also the shells you wish to connect.



With **manual line** selected, click on the panels you have chosen to join to define points in the run. PRIMER will sketch the points and the line as you go along.



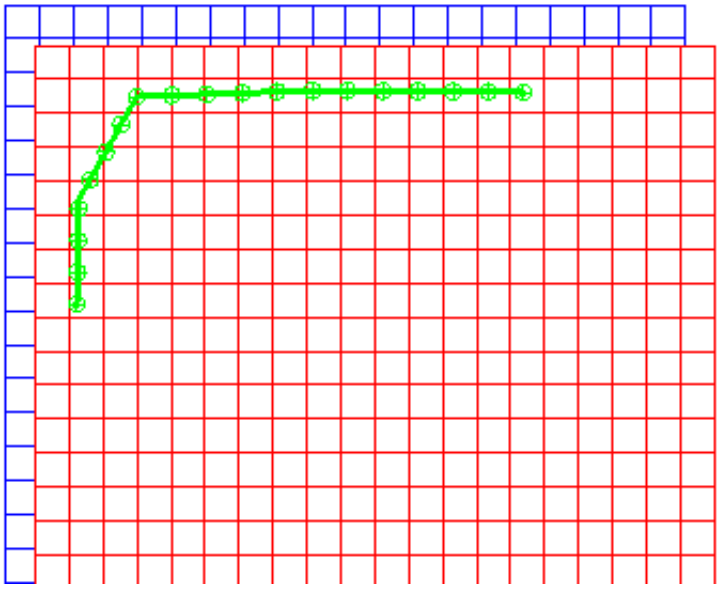
Clicking on the **sketch spot line** button as you go along will allow you to preview the spotwelds before actually creating it.



After you are happy with your defined path, clicking Apply will create the beam/solid elements and create the connection entity.

The connection entity is drawn as two blobs connected by a path line.

The colouring of this connection entity is dependent on connection status and follows the same scheme as spotwelds.

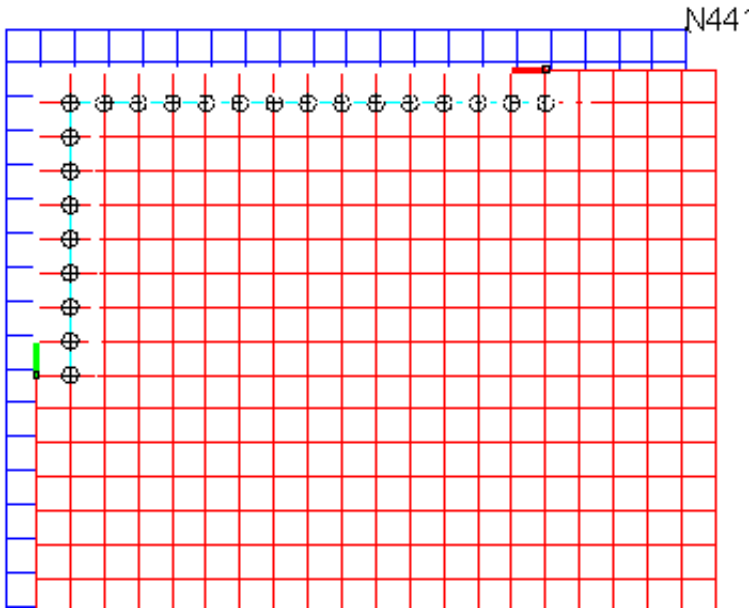


Free edge spotweld lines creation method

This method is used to create spotwelds along a free edge, using the free edge to define the path.

The free edge length is defined by clicking on two nodes along the free edge. PRIMER will determine all the nodes along the free edge between the two selected nodes.

Spotwelds created using this method will follow the free edge. The distance between the centre of the spotwelds and the free edge can be specified using **Edge dist**.



Geometry line creation method

The **geometry line** creation method can be used to create spotweld line runs from geometry lines that exist in any model in PRIMER.

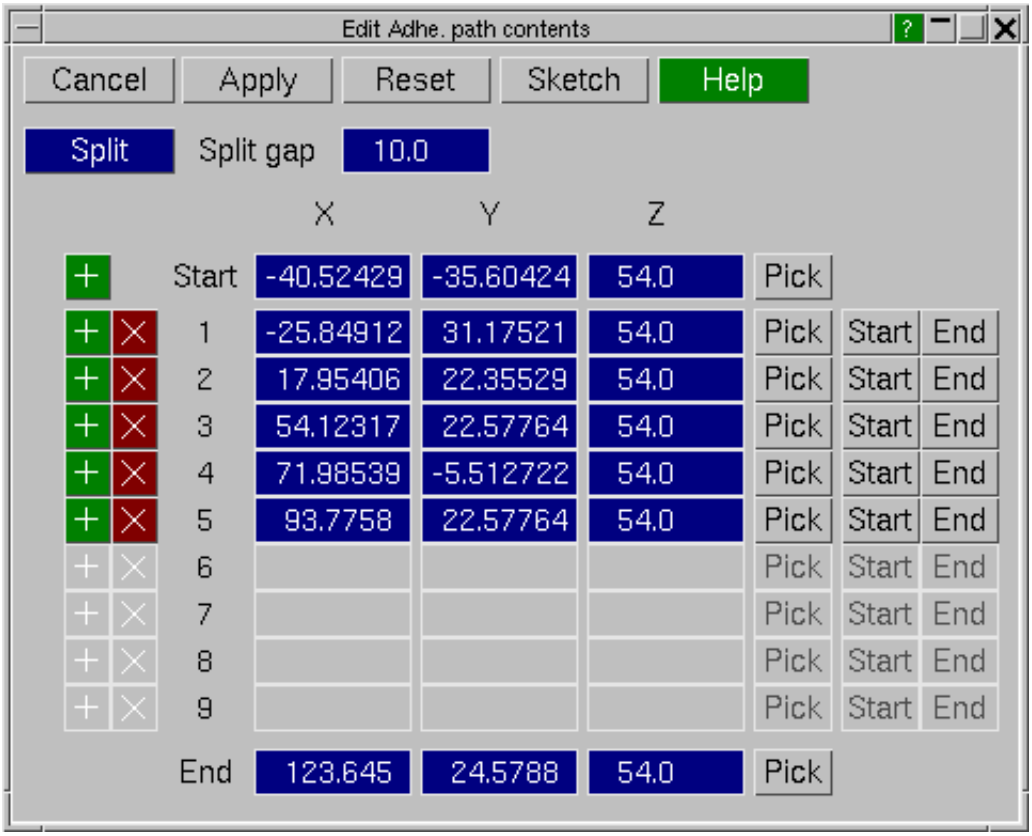
Use **Line split no.** to specify how many increments the line is split into when creating the information for the spotweld line path.

Modifying the spotweld line path

The path of a spotweld line run can be modified by clicking on the **Modify existing path** button in the create panel.

The path can also be modified in the same way through the connections table (see section 6.10.2).

After clicking the button, you select the spotweld line connection entity you wish to modify, and the following panel will appear.



The path modification panel displays the path information. The coordinates of the start and end points are displayed, as well as the in-between path points. Through this panel it is possible to carry out the following functions:

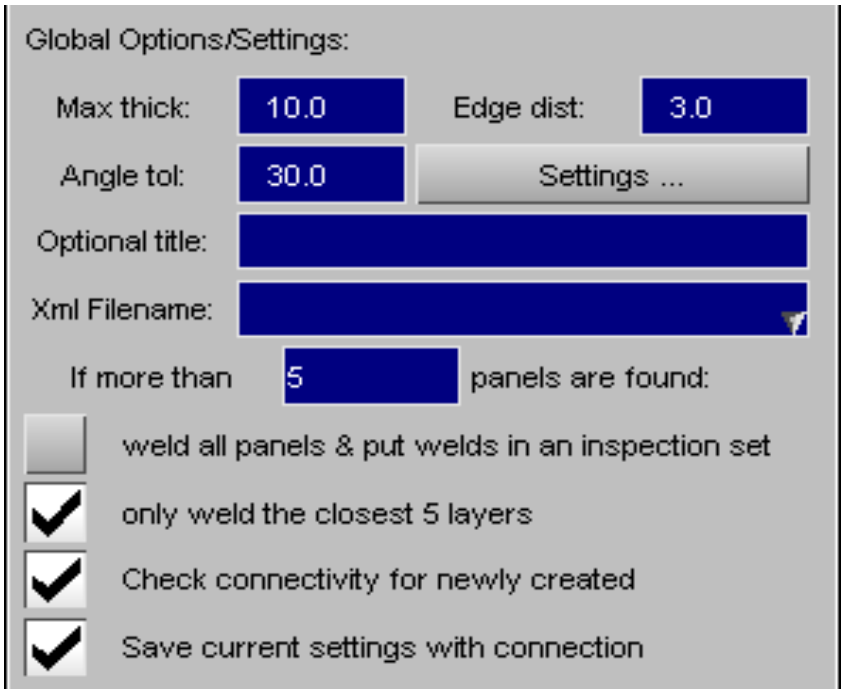
- Add or remove path points to the spotweld line run.
- Modify the position of point on the spotweld line path, either by typing in new coordinates or by clicking on the screen.
- Setting any of the existing path points to a new start or end point.
- Splitting the path at any point along the spotweld line run with a defined gap.

This panel works in the same way as modifying an adhesive path run. For more information, see Modifying the adhesive path above.

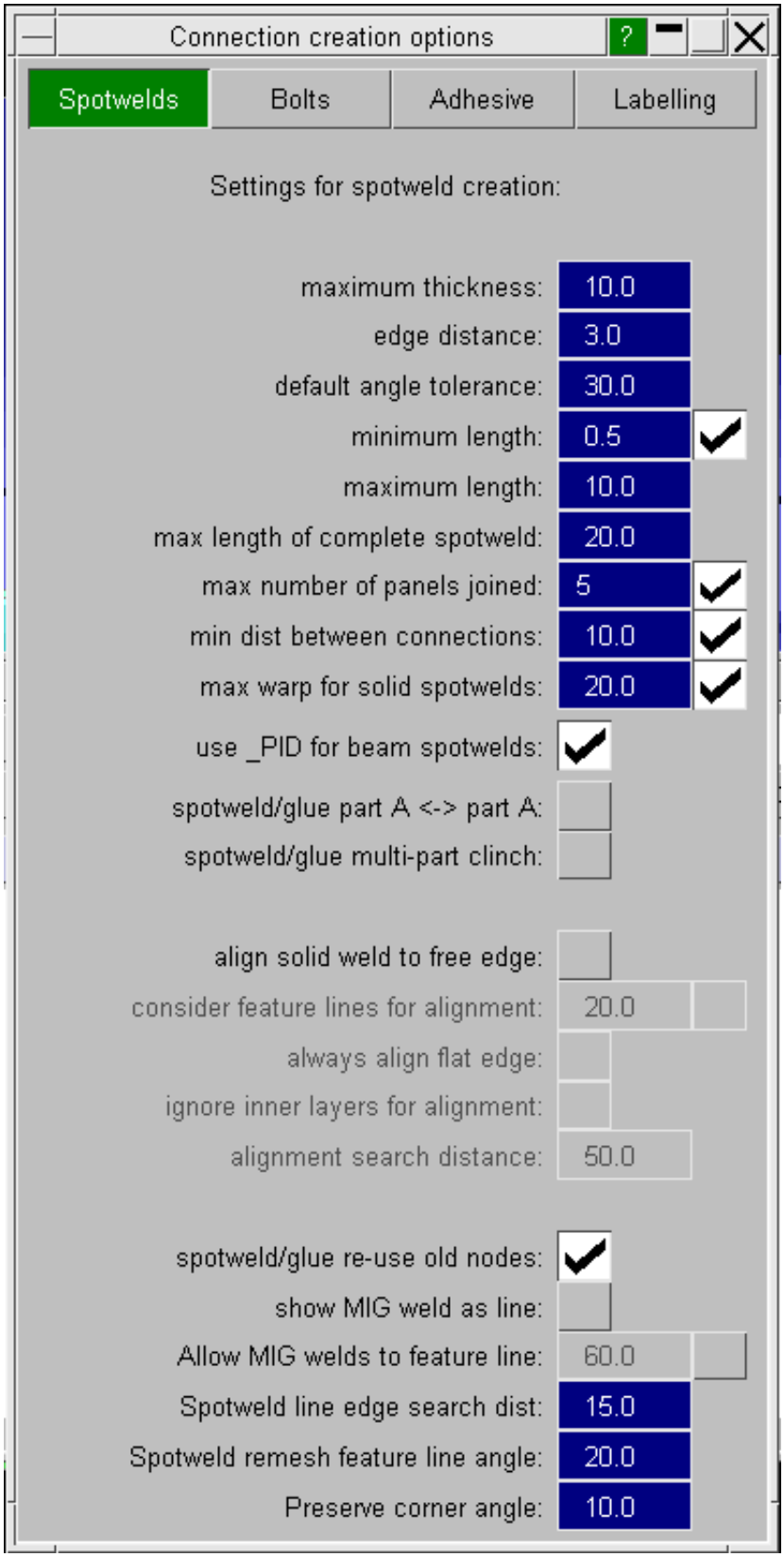
Connectivity check

Upon exiting the connection create panel, PRIMER will automatically check the connectivity status of all newly created spotwelds and adhesive connections.

This check (on by default) can be turned off on the global settings panel.


Global Options/Settings:
Max thick:
10.0
Edge dist:
3.0
Angle tol:
30.0
Settings ...
Optional title:
Xml Filename:
If more than
5
panels are found:
weld all panels & put welds in an inspection set
only weld the closest 5 layers
Check connectivity for newly created
Save current settings with connection

Settings

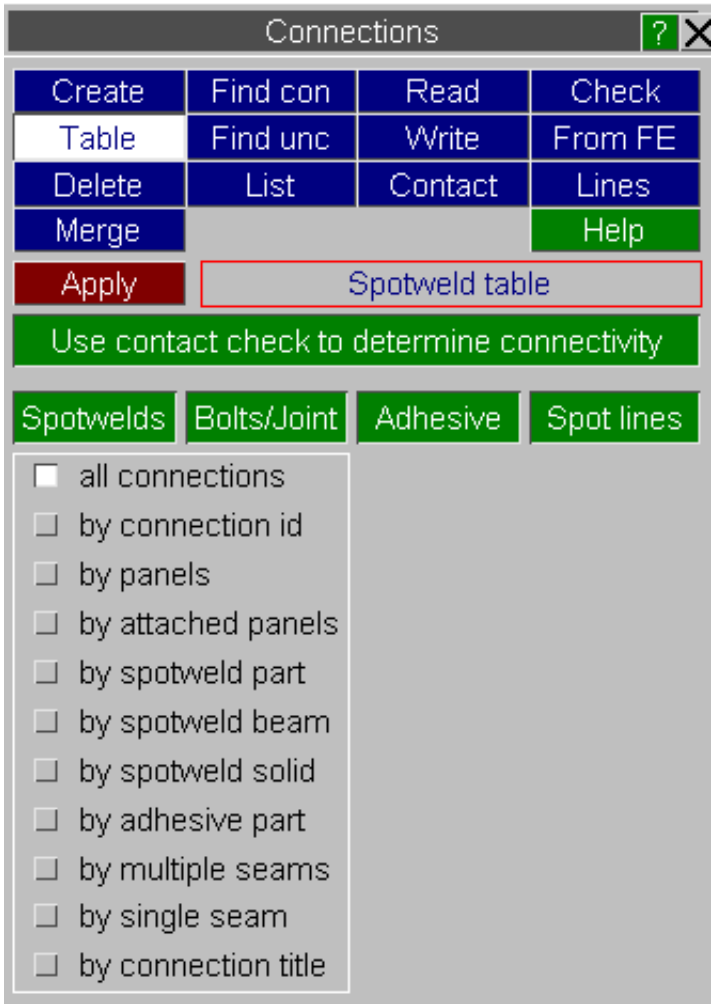


See the connection options section for more information.

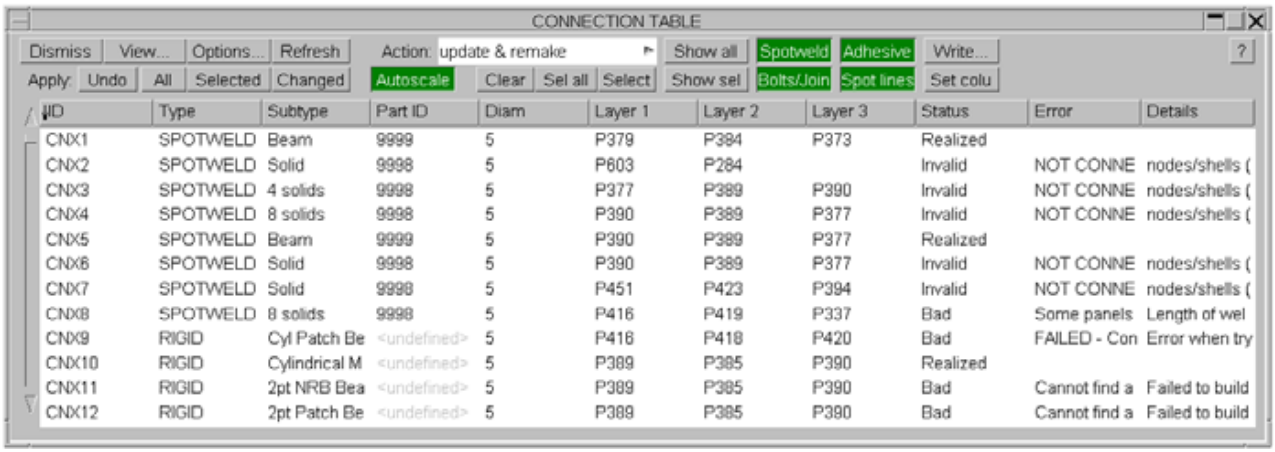
6.12.2 Connection table

The connection table allows you to review and modify connections data.

You can specify which connections to review a number of different methods. For more details of the different methods see section 6.12.0.



After specifying the connections to be reviewed, press the **Apply** button. The connection table window will appear.



ID	Type	Subtype	Part ID	Diam	Layer 1	Layer 2	Layer 3	Status	Error	Details
CNX1	SPOTWELD	Beam	9999	5	P379	P384	P373	Realized		
CNX2	SPOTWELD	Solid	9998	5	P603	P284		Invalid	NOT CONNE	nodes/shells (
CNX3	SPOTWELD	4 solids	9998	5	P377	P389	P390	Invalid	NOT CONNE	nodes/shells (
CNX4	SPOTWELD	8 solids	9998	5	P390	P389	P377	Invalid	NOT CONNE	nodes/shells (
CNX5	SPOTWELD	Beam	9999	5	P390	P389	P377	Realized		
CNX6	SPOTWELD	Solid	9998	5	P390	P389	P377	Invalid	NOT CONNE	nodes/shells (
CNX7	SPOTWELD	Solid	9998	5	P451	P423	P394	Invalid	NOT CONNE	nodes/shells (
CNX8	SPOTWELD	8 solids	9998	5	P416	P419	P337	Bad	Some panels	Length of wel
CNX9	RIGID	Cyl Patch Be	<undefined>	5	P416	P418	P420	Bad	FAILED - Con	Error when try
CNX10	RIGID	Cylindrical M	<undefined>	5	P389	P385	P390	Realized		
CNX11	RIGID	2pt NRB Bea	<undefined>	5	P389	P385	P390	Bad	Cannot find a	Failed to build
CNX12	RIGID	2pt Patch Be	<undefined>	5	P389	P385	P390	Bad	Cannot find a	Failed to build

Each row in the table represents a connection. At a glance, you can review connection Type (Spotweld, Rigid bolt, adhesive or spotweld line), element type (Beam or Solid), connection coordinates and the layers (Parts) connected together. The **Status** column tells you whether a connection has been defined and realized properly. If the connection has not been realized then the **Error** and **Details** columns give you information on the error. Often the error messages are too long to be shown in the column. In this case if you leave the mouse over the column the whole message will be shown in hover text.

Changing the table columns

The connection table can show various properties of connections.

Set columns will activate the most commonly applicable fields for the connections that are currently on the table. For example, if they are all welds you won't see any bolt parameters.

To add or remove columns press the **View...** button which will bring the window shown below. The fields that are currently shown are marked with a tick symbol.

Unset All can be used to de-activate all sttings but ID. **Reset** will return to the last selection. **Save Settings** will record your preferred defaults in the oa_pref file.

The **View...** panel has various sub-headings to make it easier to find and turn on/off table columns. These sub-headings are:

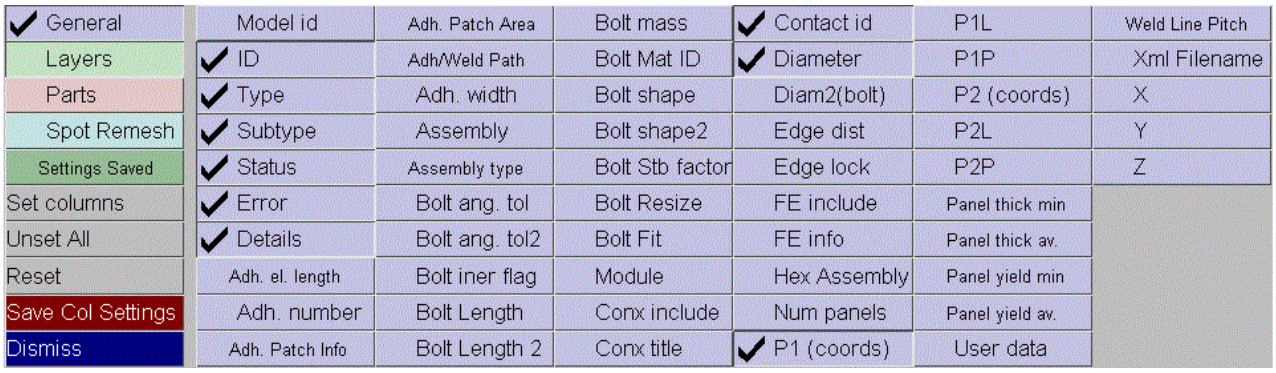
General - General connection properties not related to the other sub-headings.

Layers - Up to 10 layers can be specified per connection - this panel gives access to them on the table.

Parts - A different part ID can be specified for each layer of beam/solid elements in the connection - this panel gives access to them on the table.

Spot remesh - Gives access to all columns relating to remesh settings/properties for spotwelds.

Settings saved - By default, all settings used during creation of a connection are saved with the connection entity. This means that when the connection is remade, the saved settings are used rather than the defaults. This panel gives you access to them on the table.



The columns can be made wider or smaller by dragging the sides of them in the header. The column order can be changed by dragging a column to a new position.

By default the rows are sorted by connection ID. You can sort by a different column by pressing on the column header. Pressing once will sort in ascending order. Clicking the column again will sort by descending order. The column that is currently used for sorting has an arrow drawn on it. This also shows if the sort is ascending or descending.

The **Set columns** function will try to display those most relevant to the current selection of connections on the table and may be useful after on has applied the dynamic filters.

Available table columns

The following columns are available for display in the connections table under **View...** (seperated into the sub-headings described above).

General

Column	Explanation
Model	Model label
ID	Connection ID
Type	Type of connection (spotweld, rigid etc.)

Subtype	Subtype of the connection (beam, solid etc.)
Status	Status of connection (realized, invalid, bad etc.)
Error	Error code for that connection
Details	More details on error
Adh. width	Width of adhesive run
Adhe. number	Number of elements across adhesive width
Adhe. el. len.	Length of adhesive element along adhesive run
Adhe/Weld Path	Number of path points between start and end of adhesive/spotweld lines (can also modify path through this column)
Adhe. Patch Info	Provides number of source "shells" for an adhesive path, plus a method for modifying the source "shells"
Assembly type	Instead of defining individual layers, an assembly of parts can be specified for each connection. The type can be PART_SET or PRIMER assembly
Assembly	Instead of defining individual layers, an assembly of parts (part set or assembly depending on type above) can be specified for each connection.
Bolt length	Max length for 1 pt bolt, max thickness at end 1 for 2pt bolt
Bolt length 2	Max thickness at end 2 for 2 pt bolt
Bolt Ang tol	Angle tolerance for shell normals for 1pt bolt or at end 1 for 2pt bolt
Bolt Ang tol2	Angle tolerance for shell normals at end 2 for 2pt bolt
Bolt Shape	Shape control (edge of hole, 1 ring,etc) for 1 pt bolt or at end 1 for 2 pt bolt
Bolt Shape2	Shape control for end 2 for 2 pt bolt
Bolt mass	Mass of bolt connection type
Bolt stb factor	Ratio of the required mass of the rigid bolt for stability to the total mass of the rigid bolt
Bolt Iner flag	Part inertia flag for bolt
Bolt Resize	Flag to resize library bolt to points
Bolt Fit	Flag to resize and then apply contact fitting algorithm
Bolt Mat ID	Material ID for bolts (optional)
Module	Name of bolt library module
conx include	Include file location of connection entity
Conx Title	Title of the connection
contact id	Contact relating to connection entity
Diameter	Diameter of spotweld or bolt (end 1 for 2 pt bolt)
Diam2	Diameter of bolt at point 2
Edge dist	Distance of spotwelds from free edge (only applies to Edge lock = TRUE)
Edge lock	Set to TRUE to lock a spotweld line to a free edge.
FE include	Include file location of FE data within connection entity
FE info	Information on FE that makes up the connection entity
Hex Assembly	Labels of Hex assemblies (*DEFINE_HEX_SPOTWELD_ASSEMBLY) of solid elements related to a connection
Num panels	Number of panels in connection (2T, 3T etc.)
P1	Coordinates of connection (start point for adhesive/spotweld line/bolt)/centre of cylinder for end 1 of generic bolt
P1L	
P1P	vectors P1-P1L and P1-P1P describes normal plane of cylinder which describes end 1 of generic bolt
P2	Coordinates of end point of adhesive/spotweld line/2 point bolt - centre of cylinder for end 2 of generic bolt
P2L	
P2P	vectors P2-P2L and P2-P2P describes normal plane of cylinder which describes end 2 of generic bolt

min panel thick	minimum thickness of attached panels
av panel thick	average thickness of attached panels
min panel yield	minimum yield stress of attached panels
AV panel yield	average yield stress of attached panels
User data	Any typed in user data. This is not written out, so just remains for the current Primer session
Weld line Pitch	Pitch of spotwelds along path for spotweld lines connection type
Xml Filename	Filename of connection source XML file (if applicable)
X, Y, Z	Coordinates of connection (start point for adhesive/spotweld line/bolt)

Layers

Column	Explanation
Layer 1,2,3,...	Layer information

Parts

Column	Explanation
Part ID	Part ID of the connection entities (applies to all layers if PID (L2-L3) etc. not specified, else applied to entities between L1 & L2.
PID (L2-L3), ...	Part ID of the connection entities between specified layers.

Spot Remesh

Column	Explanation
Remesh	Flag to say whether we remesh around the sportweld or not.
Remesh diam	Diameter around the spotweld that will be remeshed.
Remesh Nrings	Number of rings around the spotweld to create.
Remesh R1 diam	Diameter of ring 1.
Remesh R2 diam	Diameter of ring 2.
Remesh R3 diam	Diameter of ring 3.
Remesh R4 diam	Diameter of ring 4.
Remesh R5 diam	Diameter of ring 5.
Remesh R0 PID	Part ID of layer mesh at centre of spotweld. If not specified, Part ID will match layer part.
Remesh R1 PID	Part ID of ring 1 of spotweld. If not specified, Part ID will match layer part.
Remesh R2 PID	Part ID of ring 2 of spotweld. If not specified, Part ID will match layer part.
Remesh R3 PID	Part ID of ring 3 of spotweld. If not specified, Part ID will match layer part.
Remesh R4 PID	Part ID of ring 4 of spotweld. If not specified, Part ID will match layer part.
Remesh R5 PID	Part ID of ring 5 of spotweld. If not specified, Part ID will match layer part.
Remesh Rule	JavaScript rule for specifying ring diamter and Part ID based on certain rules.

For more information on spotweld remeshing and what these mean please see the spotweld remeshing section in spotweld creation.

Settings Saved

Column	Explanation
Store/Default	Flag to say whether we store settings with the connection or not.

Length check	Length check of this connection is ON or OFF.
Maximum length	Maximum length allowed for this connection.
Minimum length	Minimum length allowed for this connection.
Total length	Total length allowed for this connection.
No. panel check	Check against max number of panels ON or OFF.
Max number of panels	Maximum number of panels joined for this connection.
Warpage check	Solid element warpage check ON or OFF.
Maximum warpage	Maximum warpage allowed for this connection.
Use _PID	Setting for if this beam spotweld sets _PID on beams created or not.
Allow same part	If set to ON, this connection is allowed to attach a part to itself.
Allow clinch	If set to ON, this connection is allowed to attach a shell to it's neighbour (clinch situation).
Align solid	If set to ON, this connection (if a solid spotweld) will align with nearby panel free edges.
Spot line tol	For spotweld lines, distance searched for free edges if LOCKED to a free edge.
Patch check angle	Adhesive patch angle tolerance.
Glue break angle	If adhesive, break angle used to determine where to position nodes along the length of an adhesive run.
Glue soft aspect ratio	If adhesive, aspect ratio check used to determine when solid elements should be modified.
Glue hard aspect ratio	If adhesive, aspect ratio check used to determine when solid elements should not be created.
Max thickness	Search distance to find shells to attach to.
Edge distance	Distance away from an edge that a connection can still project to the surface.
Angle tolerance	Check on angle between elements joined.
Bolt mass adjust	For bolts, flag for adjusting mass.
Bolt part min mass	Minimum mass for rigid bolt part.
Bolt nrb min mass	Minimum mass for nrb bolt.
Boilt feature line	Consider feature lines for bolt holes.
Bolt dth beam	Add database history beam to bolds containing beams.
Bolt zero len dscr	Create a zero length discrete beam for 2pt bolts.

Changing the default table columns

By default the connection table will show the Connection ID, type and subtype, Part, Diameter, 3 layers, status error and error details columns. If you want to change which columns are shown by default then change the columns shown to be the ones you want and press **Save Settings** in the **View...** popup. This will automatically add a preference `primer*conx_table_columns` to your home oa_pref file with the appropriate columns.

Selecting connections

Connection rows on the table can be selected in a number of ways. The easiest way is by left mouse clicking on the row. A selected row will be highlighted blue. Multiple lines can be selected using the shift or ctrl key combined with the mouse. There are also buttons at the top of the table to aid you in selecting and viewing connections in the table. The **Clear** button clears all current selections. The **Sel all** button selects all connections currently displayed on the table. **Select** will bring up an object menu and allow you to select connections using that method (i.e. being able to use various filters to select connections). **Show sel** will display in the table only those connections currently selected. **Show all** will bring back and display the original connections on the table.

Modifying connection data

The table window allows you to easily modify connection data. In the selected table row, right click on the field that you want to change. A popup menu allows you to change the option. Additionally, from all of the columns the following common options are available:

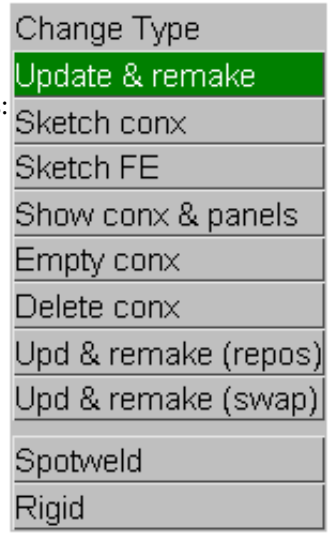
- **Update & remake**. This will remake any of the selected connection(s).

- **Sketch conx**. This will sketch the selected connection(s).
- **Sketch FE**. This will sketch the FE entities relating to the connection.
- **Show conx & panels**. This will blank everything apart from the selected connection(s) and associated panels.
- **Empty conx**. Empties the connection of it's FE entities, leaving the connection DORMANT.
- **Delete conx**. Delete the connection (and optionally the connection FE entities).
- **Upd & remake (repos)**. This will remake any of the selected connection(s), and create the connection entity at the average of the nodal coordinates related to that connection (spotwelds and rigid bolts only).
- **Upd & remake (swap)**. The same as "update and remake", but the layer order is reversed. This is useful for material types where the orientation of a solid element within a spotweld is important..

The popup changes depending on the column. Some examples of connection modifications are given below.

Modifying connection Type

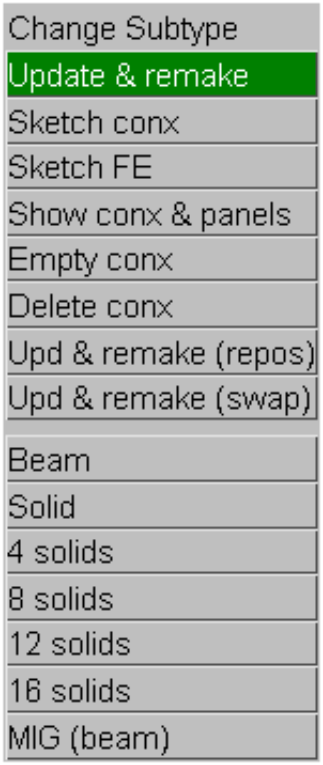
Right click a field of the **Type** column and the Change Type popup menu appears:



You can then choose between **Spotweld** or **Rigid** (Bolt) type connection.

Modifying connection Subtype

Right click a field of the **Subtype** column and the Change Subtype popup menu appears:



If the connection Type is **Spotweld**, the Change Subtype popup menu looks like this:

You can then choose the connection element type among the following options:

- Single beam element
- Single hexahedral solid element
- 4 hexahedral solid elements
- 8 hexahedral solid elements
- 12 hexahedral solid elements
- 16 hexahedral solid elements
- MIG beam elements

If the connection Type is **Rigid**, the Change Subtype popup menu looks like this:

You can then choose the connection entity type among the following options:

- Various RIGID_BODY_MERGE types of rigid connection
- Various NODAL_RIGID_BODY types of rigid connection

For more information on different bolt types, see section 6.10.1.

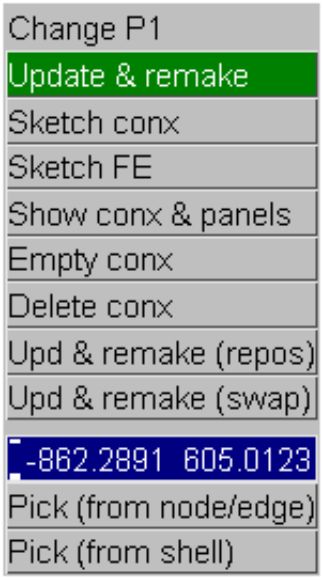
Modifying connection element Part

Right click a field of the **Part ID** column and the Change Part popup menu appears:

To modify the element Part data, you can type a new part ID in the text box. Alternatively, use **Select/Create/Edit** to choose a part from the part list, create a new part or edit a new part respectively.

If just the Part ID column is used, the part specified is applied to elements created between all layers in the connection. You can also specify different parts between layer pairs. To do this, specify parts for columns PID (L2-L3), PID (L3-L4), etc. in the same way as above. If these columns are used, then the part specified in the Part ID column only applies between layer 1 and layer 2.

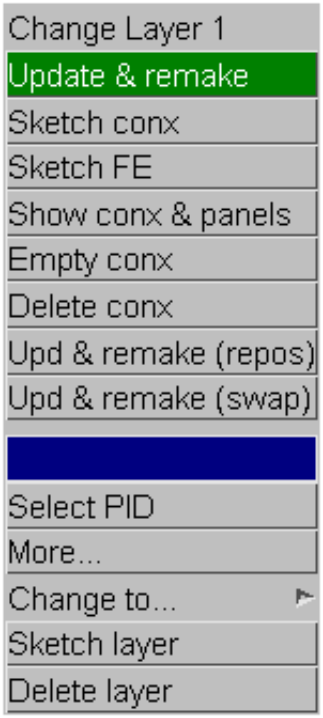
Modifying connection coordinates



Right click a field of the **P1** columns and the Change coordinate popup menu appears:

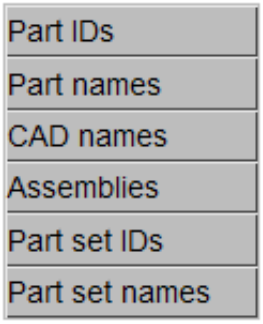
To modify the connection coordinates, you can type the new coordinates in the text box or pick a node. Alternatively, you can choose the Pick(from shell) option and using the cursor, select a point on a shell where the connection needs to be located.

Modifying connection layers



Right click a field of the **Layer** columns and the Change Layer popup menu appears:

The vast majority of layer definitions will be a single part ID. In this case to modify the layer, you can type a new part ID in the text box or use **Select PID** which allows you to pick a part or select a part from an object menu. Wildcards are allowed when defining connection layers (Part ID type only) on the connections table. A "?" represents one digit, and a "*" represents any number of digits. So, a layer part ID of 10?? will reference any shell part in the model with a label between 1000 and 1099. A layer part ID of 10* will reference any shell part in the model with a label that begins with "10".

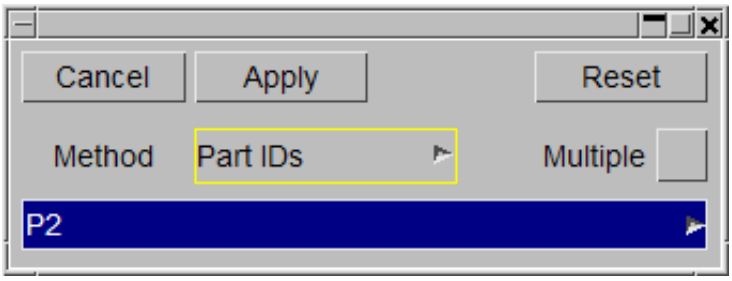


However, layers do not have to be defined by part IDs. You can also define layers by:

- **Part IDs**
- **Part names**
- **CAD names**
- **Assemblies**
- **Part set IDs**
- **Part set names**

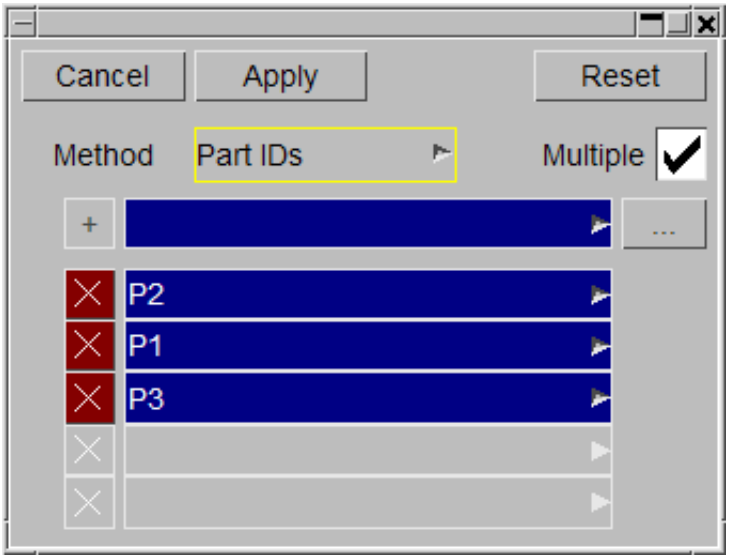
The **Change to...** option allows you to change the layer definition to be a different type. Click on the option of your choice and PRIMER will automatically update the connection layer type. For example you may want to use CAD names for the layer definitions instead of part IDs. CAD names means PRIMER will look for any matching CAD name set through the BOM feature. Failing that PRIMER will look at part titles and will look for the CAD name string within the titles. If possible PRIMER will try to change any existing definition to the new type (e.g. if you change the layer definition from Part ID to Part name PRIMER will change the definition if the existing part has a name)

Alternatively, if you press **More...** a more detailed panel allows you change the layer definition.



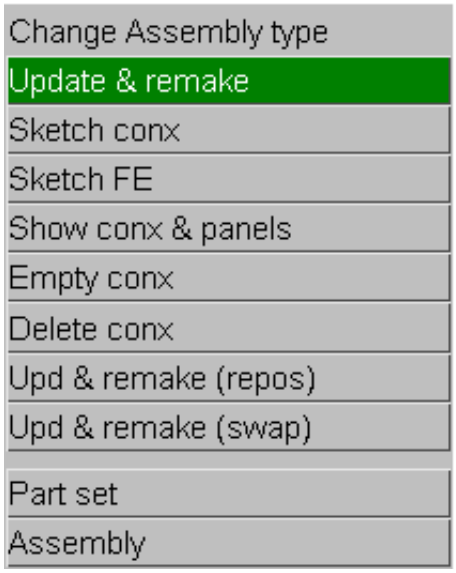
Layers can also be defined with multiple parts, part sets etc.. If you press **More...** to access the detailed layer panel then you can select multiple parts by selecting the **Multiple** checkbox. In this case you can add/remove multiple items from the layer definition.

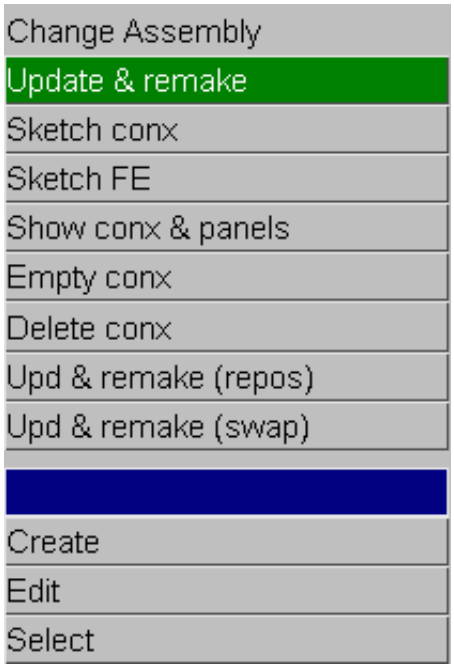
This is useful in some circumstances. e.g. if you are spotwelding a tailor welded blank then there could be multiple parts that represent the entire panel (as there are different thickness' for each part). If you wanted to make spotwelds involving this then it is much easier to include all of the parts for the layer definition for the blank. If you didn't then the part could vary depending on the position of the weld.



Modifying connection assembly

A connection can reference one assembly of parts, rather than referring to different parts in different layer. This means you specify one assembly of parts, and PRIMER will create the connection attached to any parts within that assembly in the vicinity of the connection point specified. This method is most suitable for situations where CAD part information is stored in assemblies, and there is one file per assembly containing connection information - i.e. the connection information refers to an assembly rather than specifying individual layers. Right click a field of the **assembly type** columns and select either **Part set** or **Assembly**. **Part set** means you select a *SET_PART definition to specify parts in the assembly. **Assembly** means you select a PRIMER part tree assembly to specify parts in the assembly. After choosing the type, right click a field of the assembly column to select/modify the part set/part tree assembly containing the parts.





Modifying connection include

Right click a field of the **conx inc** columns and the Change conx include popup menu appears:

To modify the connection coordinates, click on **Change include**. The standard include select panel will open. See section 5.1.6 for more information on the include selection panel.

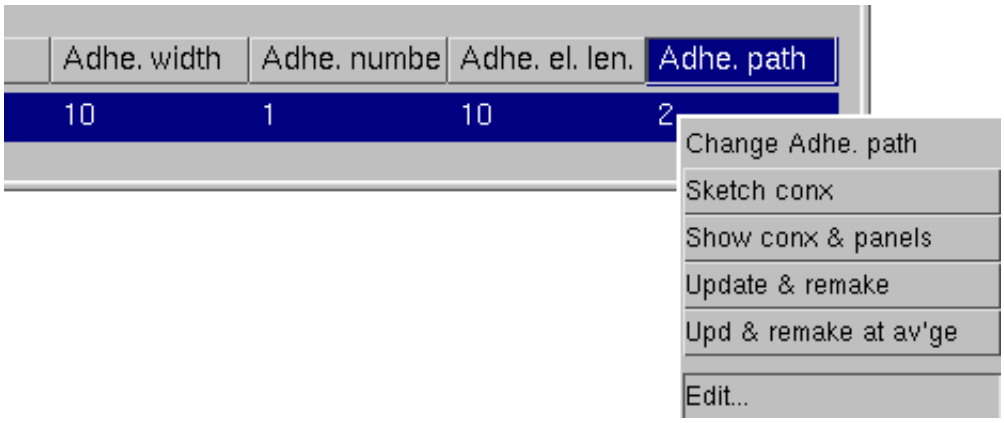
Listing the connection FE includes

Right click a field of the **FE inc** columns and the FE include popup menu appears:

This will print a listing to the screen with information regarding which include the various FE entities within the connection are in (nodes, beams etc.).

Modifying adhesive data

Adhesive data can be modified on the connection table. Adhesive width, number of elements across the width and adhesive element length can call be modified by right clicking on the field and typing in a new value. The path data can also be modified on the connection table. Under the **adhe. Path** column, the number shown is the number of path points between the start and end points of the adhesive run. Right clicking on the field and choosing **Edit** opens up the adhesive path modification panel allowing the user to modify the path data. For more information on the adhesive path modification panel see section 6.10.1.



Modifying spotweld line data

Spotweld line connection specific information can be modified on the connection table. The path can be modified in the same way as adhesive (see above), but only if it is not locked to a free edge. The columns that specifically relate to

spotweld lines are **Pitch** (the pitch of spotwelds along the line), **Edge Lock** (lock a spotweld line entity to a free edge so the path is determined by the nearest free edge of connecting layers) and **Edge dist** (distance from spotweld centres to free edge in edge lock mode).

Deleting a layer from a connection

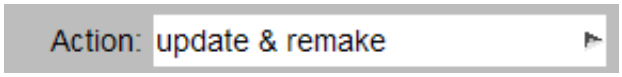
Sometimes you might want to delete a layer from a connection. For example in the connection window example at the top of this section, connection 1 has 3 layers. Layer 1 contains part ID 1, layer 2 contains part 3, and layer 3 contains part 3. If you wanted to delete layer 2 from this connection thus making a spotweld between parts 1 and 3 then right click on the **Layer 2** definition for the connection and select **Delete layer**. The entry will become blank. If you **Update & remake** the connection, the layer will be deleted.

Using the Assembly method to specify panels the connection joins together

As an alternative to the standard method (connections refer to parts or groups of parts for each layer) the assembly method can be used to specify one assembly of parts that the connection uses to determine which panels it connects. In this case the layer information in automatically modified/changes when the connections are remade, and PRIMER will just use shells within the specified assembly within the vicinity of the connection location to determine what it connects to. To switch connection to the assembly method, specify

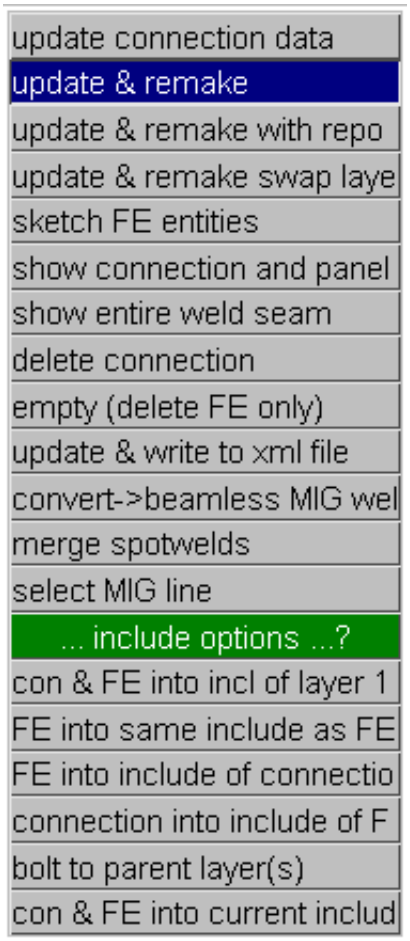
Changing the action for connections

The current action for the connections table is shown in the **Action** field.



Right clicking on the button will show the possible actions (shown on the right). The available options are:

- **update connection data**. The connection is updated with the current values in the table.
- **update & remake**. The connection is updated with the current values in the table and then remade.
- **update and remake with repos**. The same as above, but the connection entity is created at the average position of the nodes associated with the connection.
- **update and remake swap layers**. The same as "update and remake", but the layer order is reversed. This is useful for material types where the orientation of a solid element within a spotweld is important.
- **sketch (with FE entities)**. The connection is sketched.
- **show connection and panels**. Everything apart from the selected connection(s) and associated panels will be blanked.
- **Show entire weld seam**. All connections that use the same layers are shown.
- **delete connection**. The connections are deleted. This gives the option to delete both the connection itself and the FE entities, or just the connection itself, leaving the FE entities unchanged.
- **empty (delete FE entities only)**. The FE entities that make the connection are deleted but the connection definition is left 'latent'
- **update & write to file**. The connection is updated with the current values in the table and then written to file
- **convert->beamless MIG weld.** Converts the connection to a beamless MIG weld.
- **merge spotwelds**. Merges spotwelds that are close to each other. For example two 2T welds can be converted to a 3T
- **select MIG line**. Select all MIG welds in a line with currently selected MIG welds.



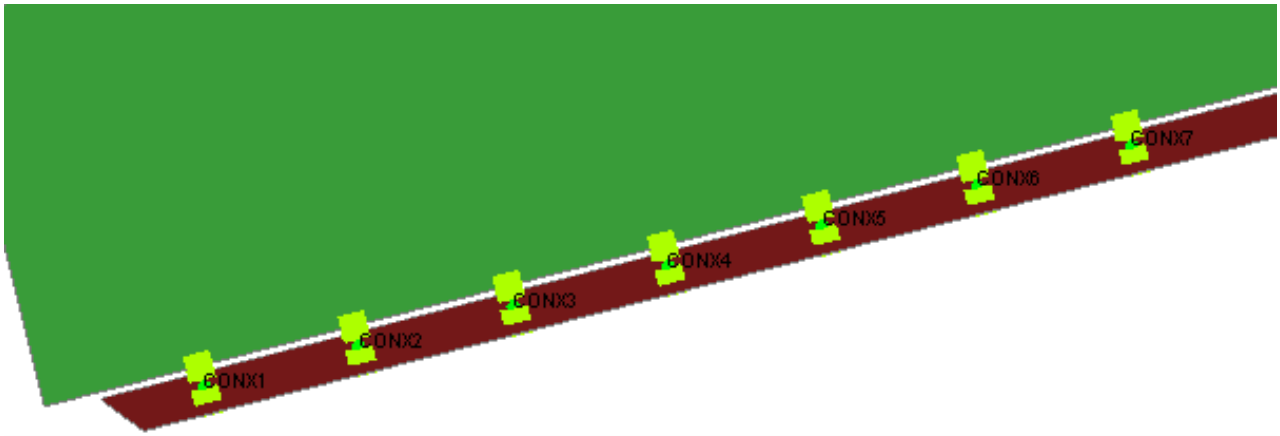
Additionally there are options for controlling which include file the connection entity and the FE entities are in.

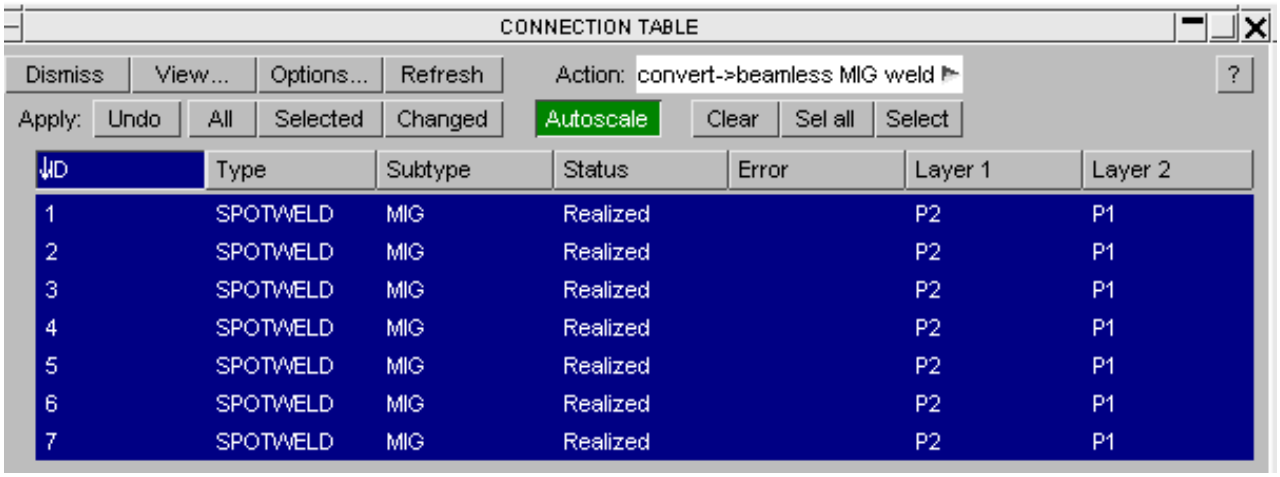
Converting MIG weld to beamless

Primer supports a MIG beam weld which is meshed to a shell on one side and tied using spotweld contact on the other. If weld failure is not an issue, users may prefer to model this weld simply using *CONTACT_TIED_SHELL_EDGE_TO_SURFACE_BEAM_OFFSET with a node set on the slave side.

The function **convert -> beamless MIG weld** can be applied to a selection of conventional beam MIG welds (their

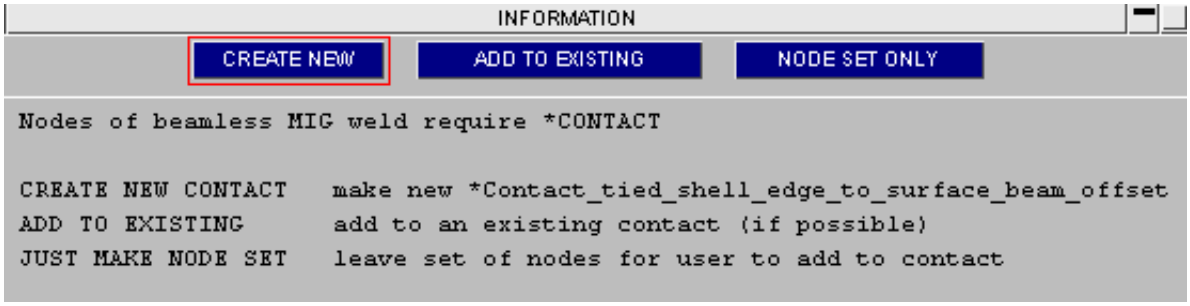
status may be REALIZED or INVALID).



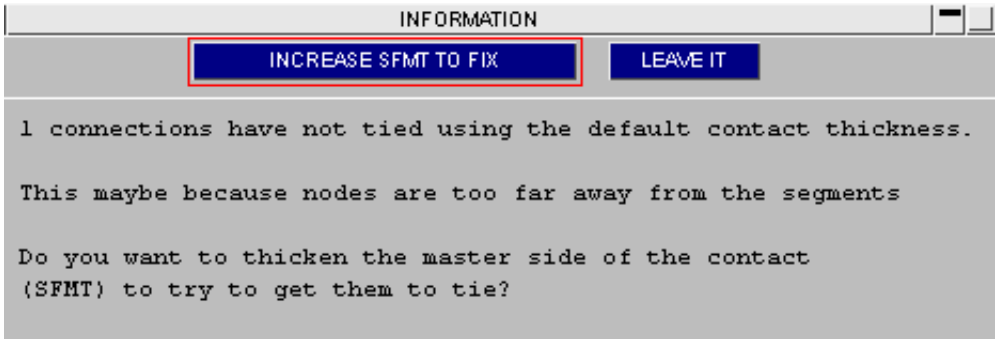


ID	Type	Subtype	Status	Error	Layer 1	Layer 2
1	SPOTWELD	MIG	Realized		P2	P1
2	SPOTWELD	MIG	Realized		P2	P1
3	SPOTWELD	MIG	Realized		P2	P1
4	SPOTWELD	MIG	Realized		P2	P1
5	SPOTWELD	MIG	Realized		P2	P1
6	SPOTWELD	MIG	Realized		P2	P1
7	SPOTWELD	MIG	Realized		P2	P1

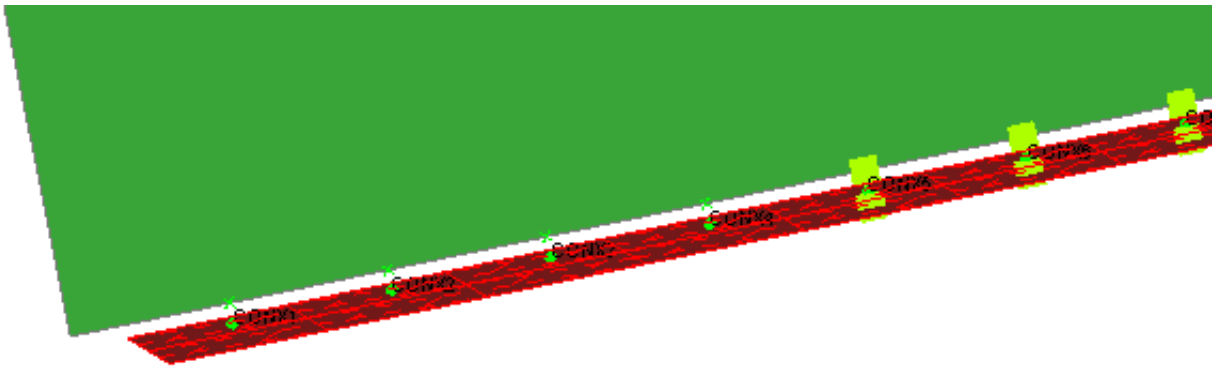
You may create a new _OFFSET contact, add nodes to node set of an existing one which is suitable (if any is found) or just dump the nodes to a set for sorting out later.



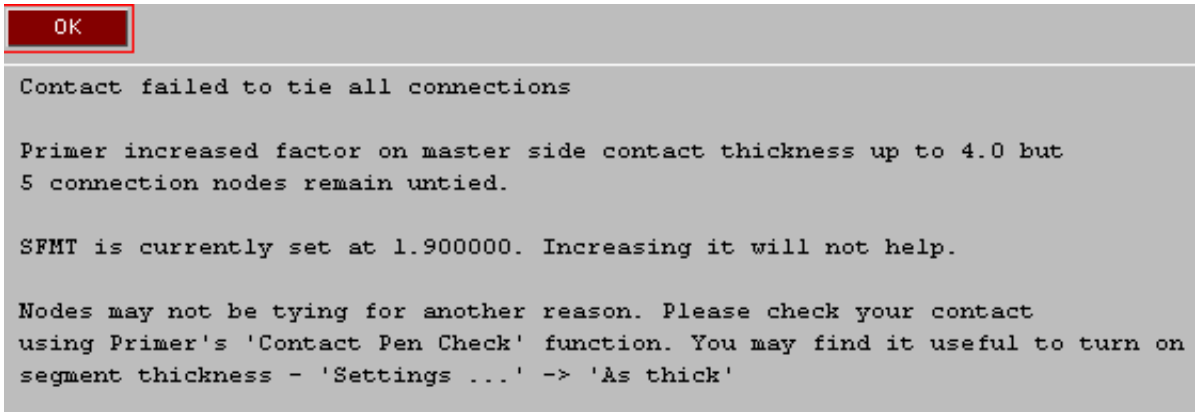
If the nodes are found not to tie because they are too far away you can run **INCREASE SFMT TO FIX** which will thicken the master side of the contact iteratively until all nodes are tied.



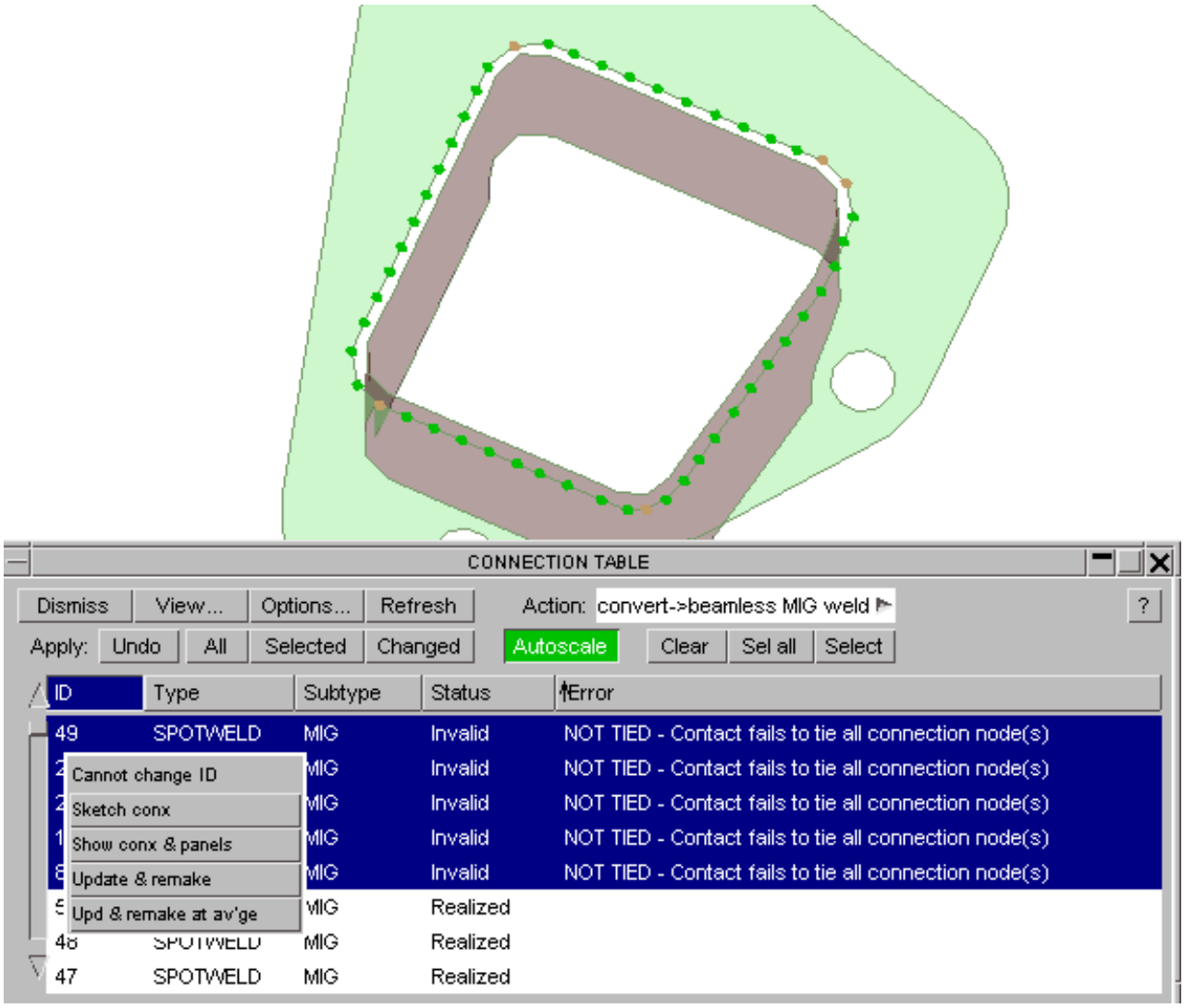
The contact alone then provides connectivity between the panels.



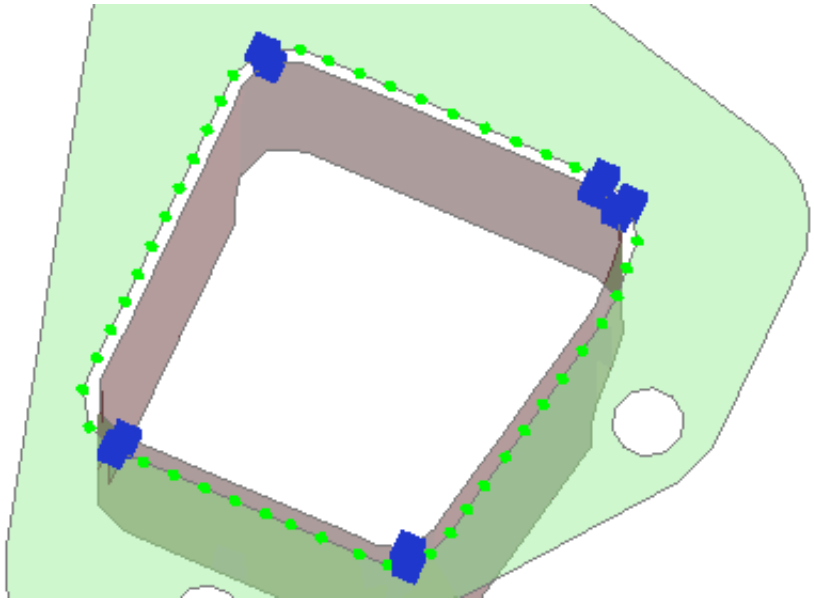
If Primer fails to tie all nodes, you will get the following error message.



The connections that failed to convert are left with **NOT TIED** error and **invalid** status (denoted by orange colour).



You can use **update & remake** to reform these as conventional beam MIG welds. Alternately, you may be able to get them tie by adjusting parameters on the tied contact which control the search depth, such as MAXPAR. Such tuning is beyond the scope of this function.

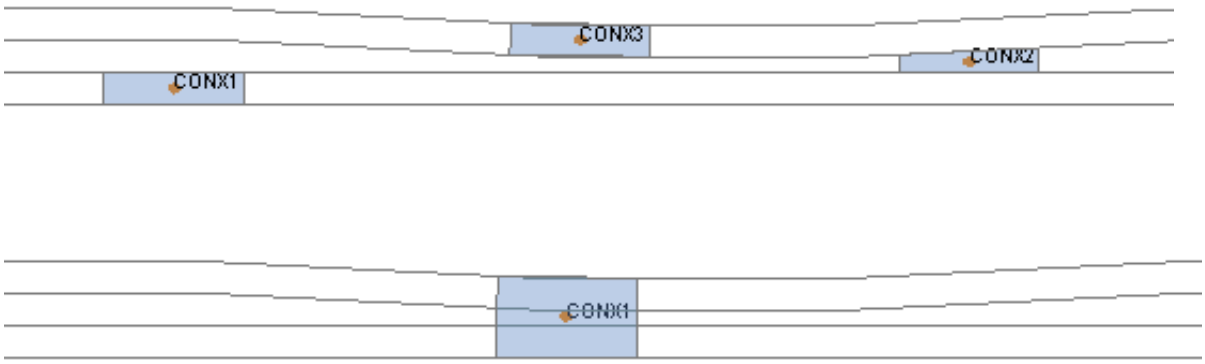


Merging spotwelds

The action **merge spotwelds** provides an alternate method to deletion for dealing with conflicting welds. The function uses parameter **min dist between connections** if it is non-zero (see settings). Two or more spotweld connections may be selected on the table and the action applied. Primer will then calculate the average position of the selected welds. To proceed the function requires that

- all welds are within **min dist between connections** of the average position (if set to zero this restriction is ignored)
- all welds must share at least one layer with another selected weld
- all welds must have the same sub-type, PID and diameter

Primer will then attempt to make a weld at the average position which connects all the layers involved. If this is successful the old welds will be deleted, if it fails they should be left unchanged.



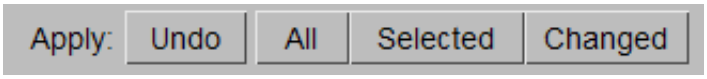
Modifying the include (layer) of connections and their FE

- **Con & FE to include of layer 1** - move connections and FE into layer of first found part in layer definition 1
- **FE into same include as FE** - move all FE of connection into layer of primer FE element
- **FE into include of connection** - move all FE into same layer as connection itself
- **Connection into include of FE** - move connection (and FE) into same layer as primary FE element
- **bolt to parent layer** - if all connected shells in same layer, move connection and all FE into that layer. If in different layers, move rigid shells/parts/materials to same layer as overlaid parent shells, nut connection/master part/C_RBOD/NRBC remain unmoved
- **Con & FE to current layer** - move connection and all FE into current layer

Primary element in this context means first found beam/solid for spotweld, NRBC or master part for bolt.

Applying/undoing connection action

Once you have selected the action you require pressing the **All**, **Selected** or **Changed** buttons will apply the action to all connections, the connections that you selected, or only the connections that are changed respectively.



To Undo any connection modification that you made (shown in red in the table), press **Undo**.

Selecting connections in the table

If you want to select connections in the table you can clear the entire selection or select all connections using **Clear** or **Sel all** respectively. Alternatively you can use **Select** to pick connections from the screen or select them using an object menu.

Display only selected on the connections table

The connections displayed on the table can be changed to show a subset of the original connections placed on the table. This is done by selecting on the table the connections you wish to show (either by clicking on them or using the Select button). Then click on the button **Show Sel**. This will update the table to show just the connections selected. Any **Apply** operation (**Undo, All, Selected, Changed**) will now just apply to those connections displayed on the table. To display on the table all the connections that were originally on the table click on **Show all**.

Filtering the types of connections shown on the connections table

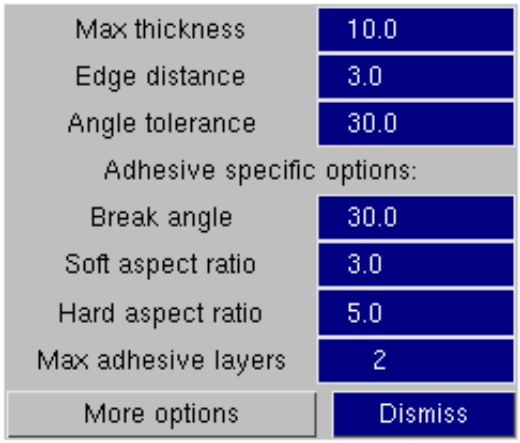
Types of connection shown on the table can easily be filtered by turning on/off the connection type filter buttons at the top of the connections table. By default, these are all on (green).

Write the connection table data as a CSV file

The **Write...** button on the connections table allows you to write the current table contents as a CSV file.

Options in the connections table

Pressing the **Options...** button shows a popup allowing you to change options for how connections are treated in PRIMER. For more details see the section on connection options.

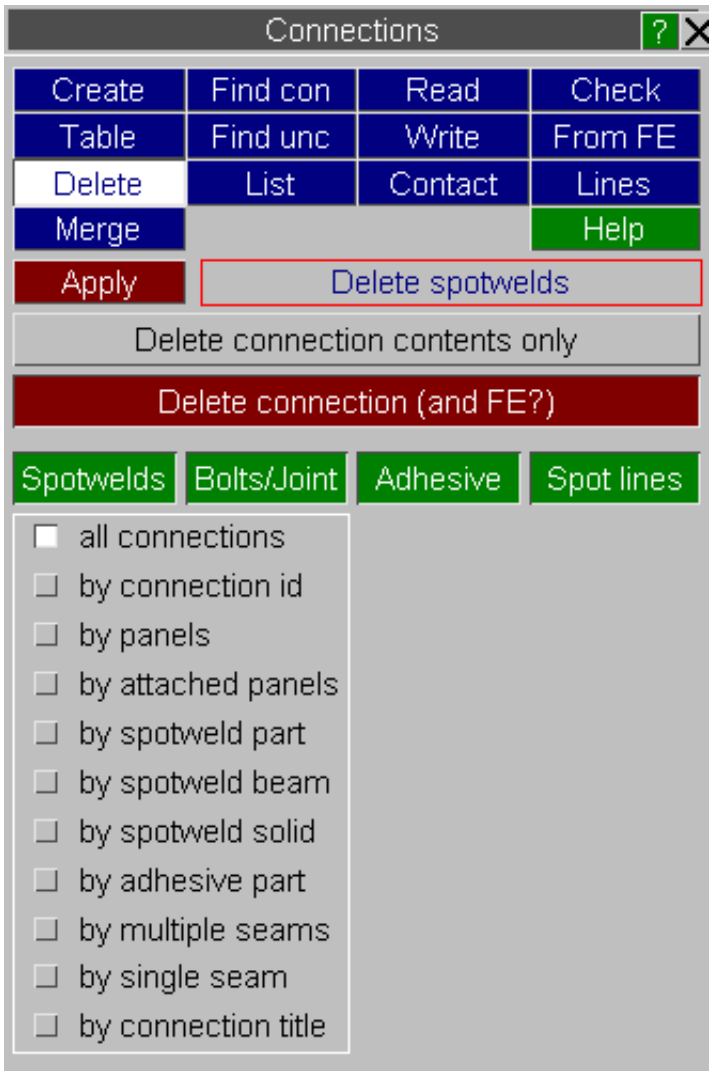


6.12.3 Deleting connections

This panel allows you to delete connections and their related entities.

You can specify which connections to delete by a number of different methods. For more details of the different methods see section 6.12.0.

Once you have selected which connections you want to delete and the method for deleting them pressing **Apply** will delete the selected connections.



Selecting the deletion method

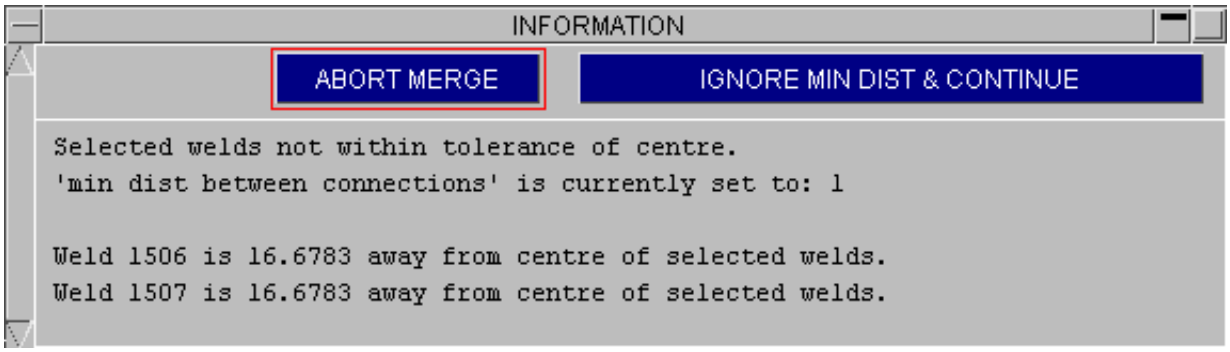
If **Delete connection (and FE?)** is set , connections and related FE entities (spotweld beams, NRBs, etc.) will be deleted all together. The connection point is lost.

If **Delete connection contents only** is set, the connection related FE entities will be deleted but the connection data (coordinates, layers, etc.) will not be deleted and therefore the connection can be remade later.

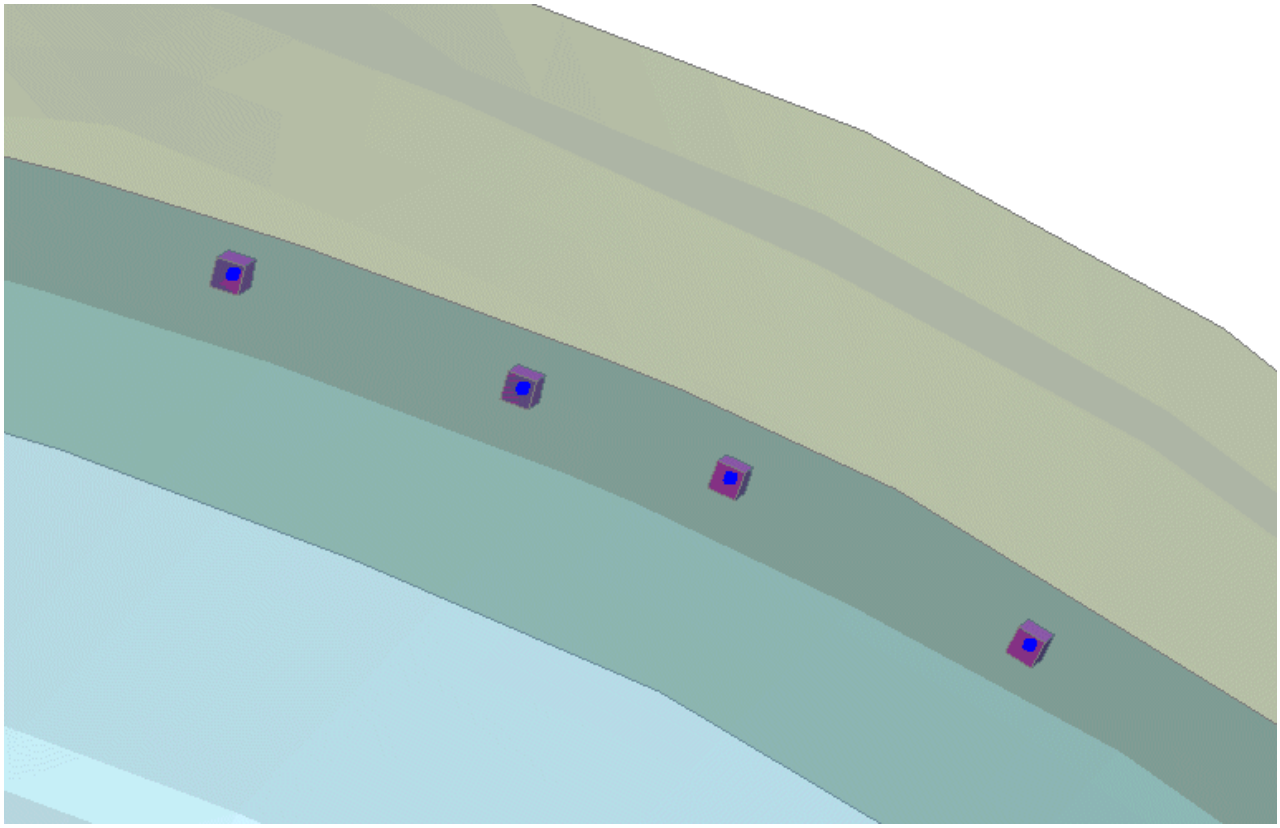
Merging spotweld connections

Two or more welds can be merged together by selecting them and applying the merge. A weld will be made at the average coordinate of the selected welds, re-making the layer definition as necessary.

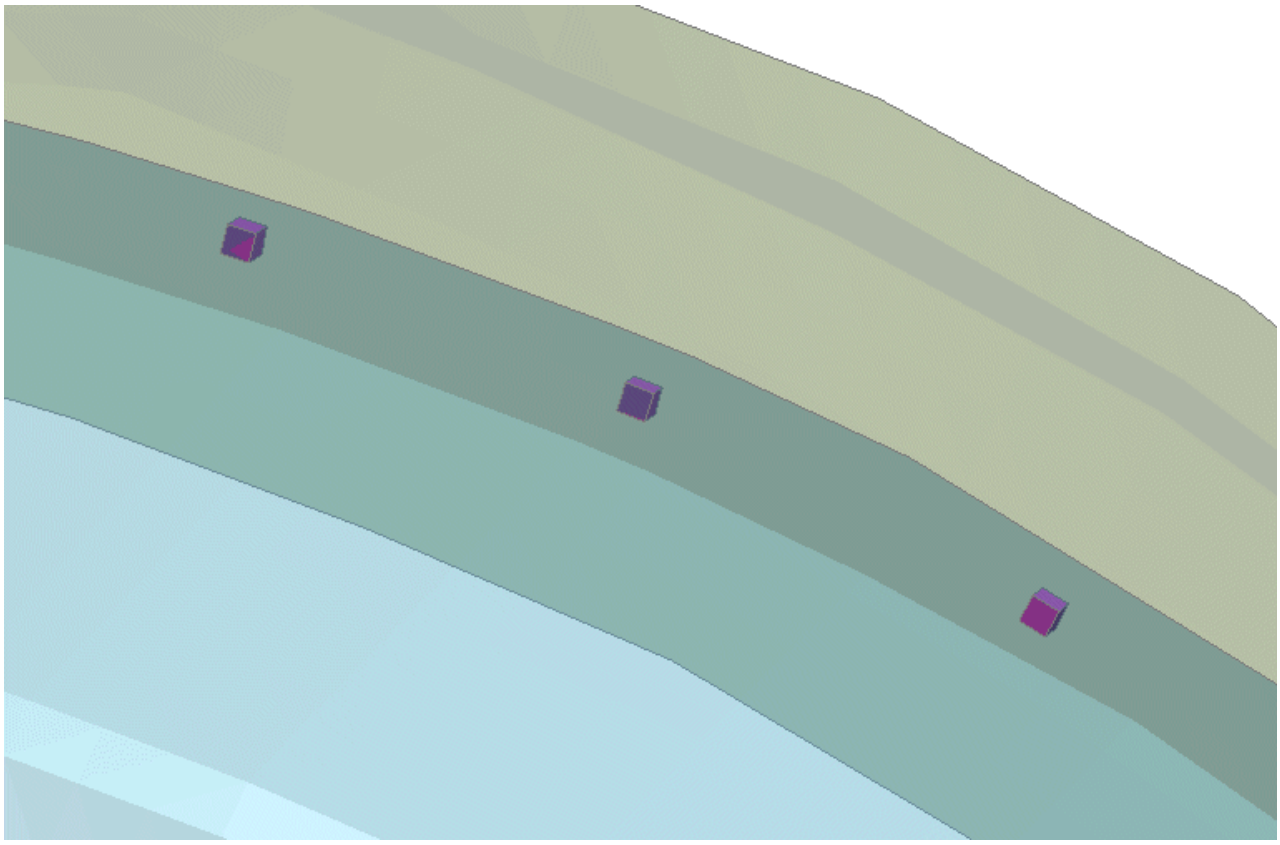
If the option **min dist between connections** is set, Primer will expect all selected welds to be within that range of the averaged position. If not a warning will be isssued.



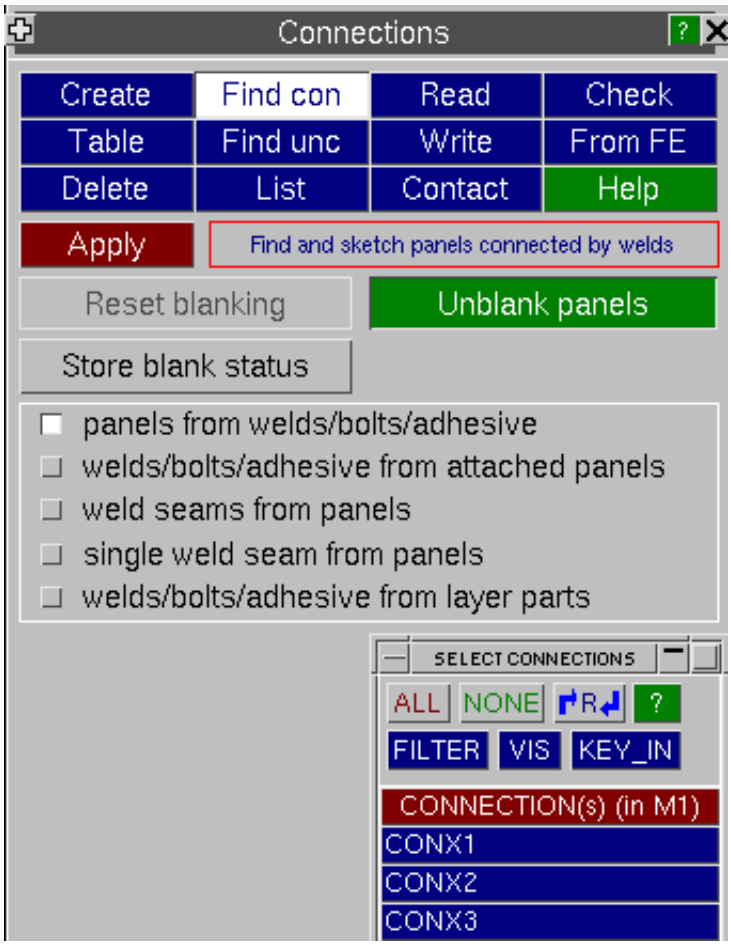
The selected welds must share at least one part, have the same pid, configuration and diameter. Then the merge will proceed.



The two cetral welds have been merged to form a new weld.



6.12.4 Finding connected panels



This panel allows you to find connections that are tied to panels. It also allows you to find panels tied to connections.

There is also a switch that alters which panels are displayed.

When finding panels tied to connections (**panels from welds/bolts/adhesive**), select the connection and press **Apply**. PRIMER will blank the whole model then unblank the connection and all panels attached to it. In order to undo the blanking that PRIMER has just done, press the **Reset blanking** button.

When finding connections tied to panels (**welds/bolts/adhesive from panels**), select the panel you wish to find the connections attached to by any of the usual methods and press **Apply**. This option will only find attached if the connection has been made, i.e. contains FE entities. To find connections associated to a panel by connection layer (i.e. the connection may not be realized and may not contain FE entities) use **welds/bolts/adhesive from layer parts**. If the **Unblank panels** button is set, PRIMER will blank the whole model then unblank the selected panel, the attached connections and any other panels tied to these connections. If the **Unblank panels** button is not set, PRIMER will blank the model then unblank the selected panel and attached connections. In order to undo the blanking that PRIMER has just done, press the **Reset blanking** button. You can store the blanking status by hitting **Store blank status**.

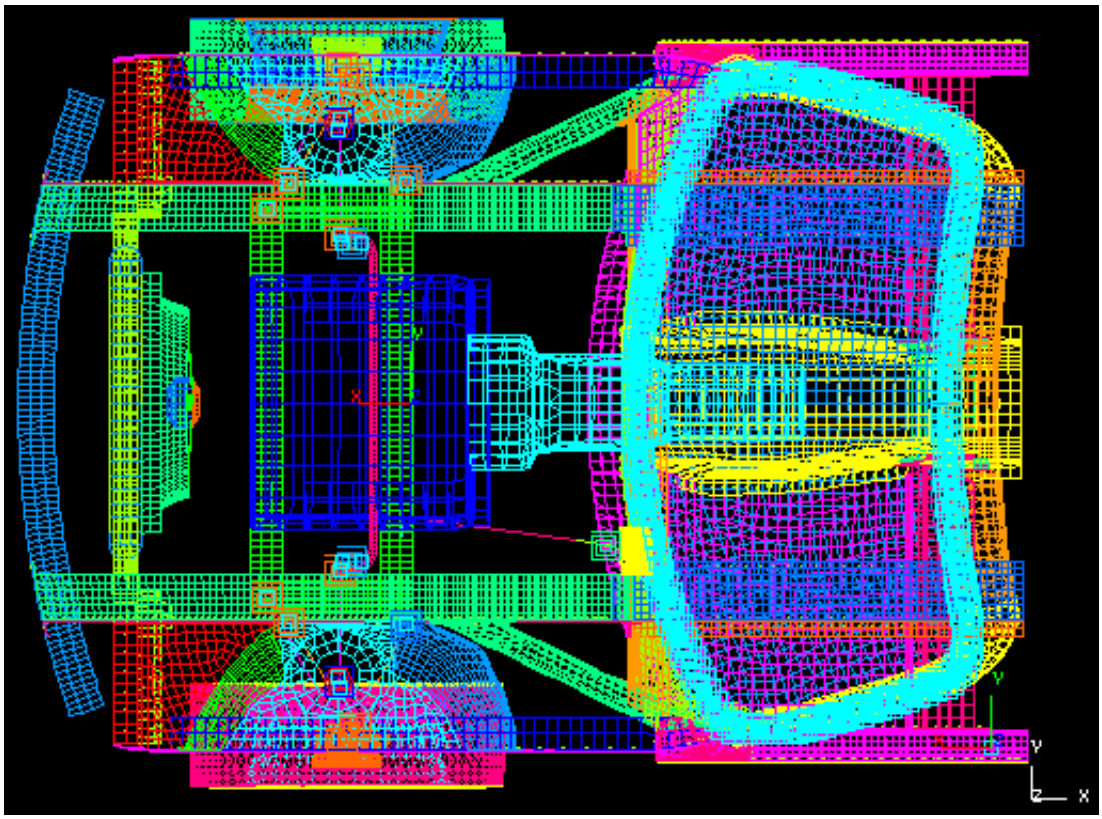
When finding seams connected to panels - select the panels you want the seams to join and press **Apply**. If you select parts 1, 2, 3 and 4 then **ANY** connection which ties **ANY** combination of these parts together will be shown.

When finding a single seam from a panel, select the panels you want the seam to join and press **Apply**. If you select parts 1, 2 and 3 then **ONLY** the connections joining **ALL** of these panels will be displayed.

For more on how the function works look at the following example.

Example

The following figure shows the front of a vehicle. It has been welded together using the PRIMER spotwelding ability. We want to find which panels are attached to the floorpan by spotwelds or bolt connections.

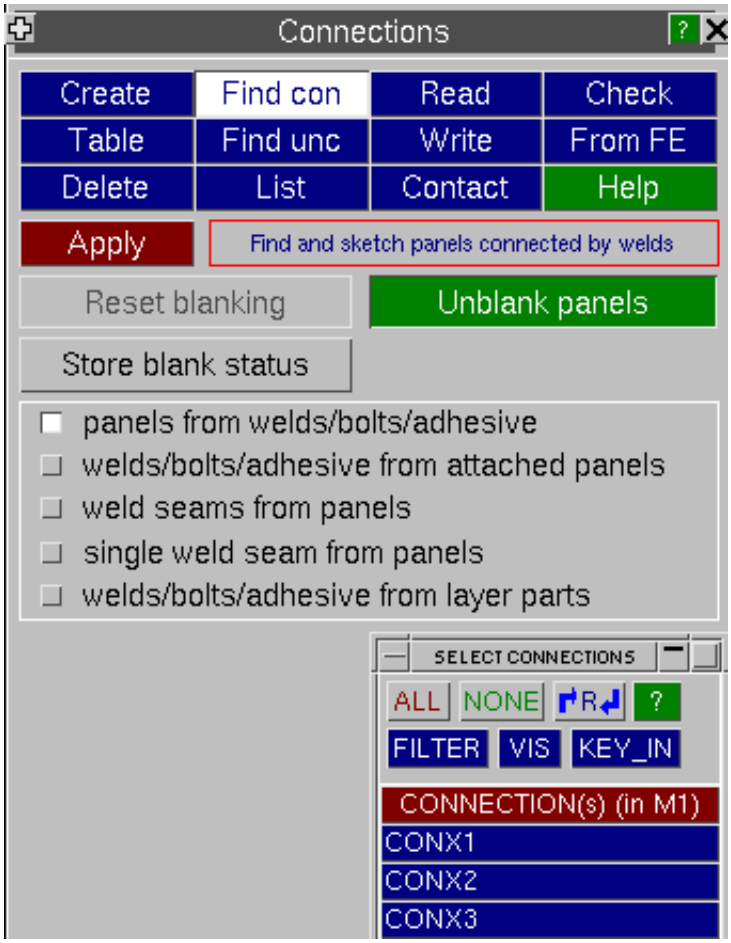


First, we select the panels we want to find connections attached to by either picking the panel from the screen or selecting the panel from the list.

Secondly, set/unset the **Unblank panels** switch.

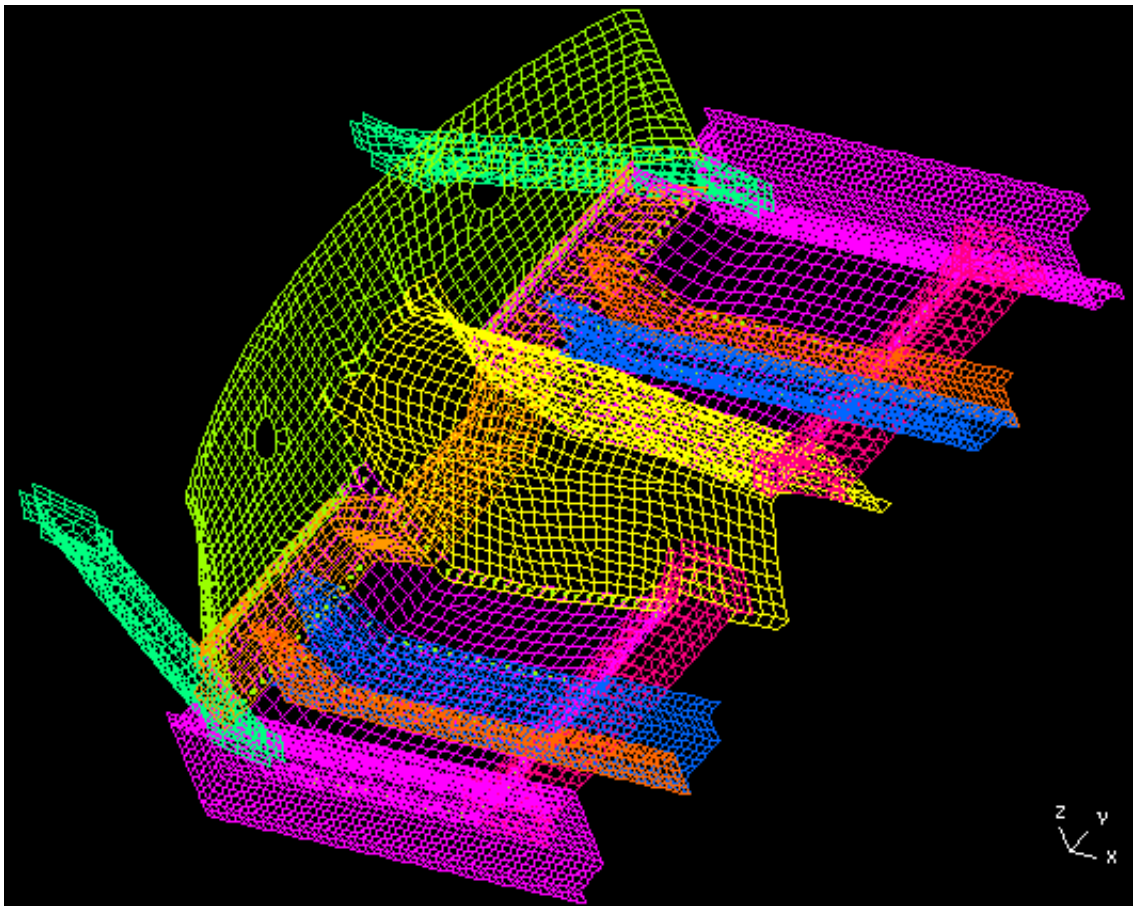
Press the **Apply** button.

Result if **Unblank panels** is set.
Result if **Unblank panels** is unset.



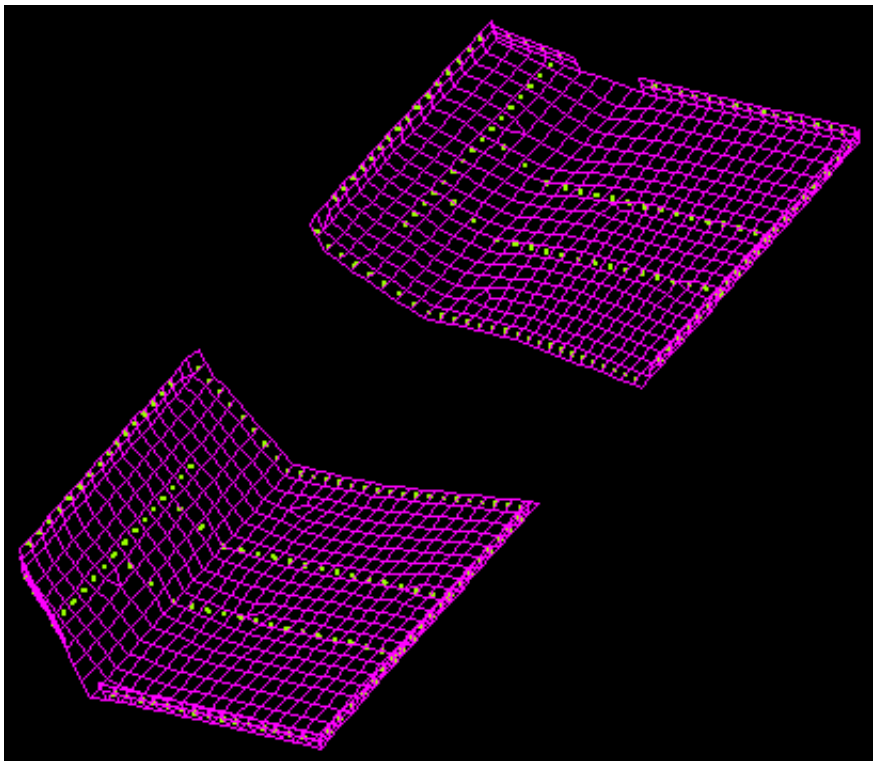
Result if **Unblank panels** set

If the switch is set PRIMER will blank the model, unblank the part you selected, find and unblank the connections attached to that part, and also find and unblank the panels that are attached by those connections.



Result if **Unblank panels** unset

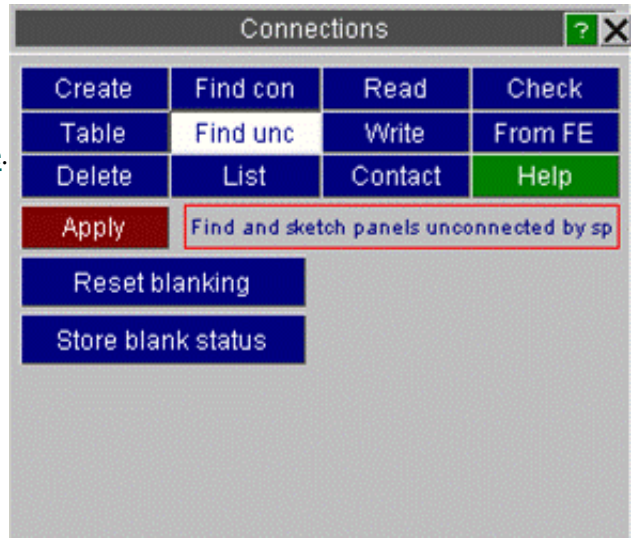
If the switch is unset PRIMER will blank the model, unblank the part you selected and find and unblank the connections attached to that part.



6.12.5 Finding unconnected panels

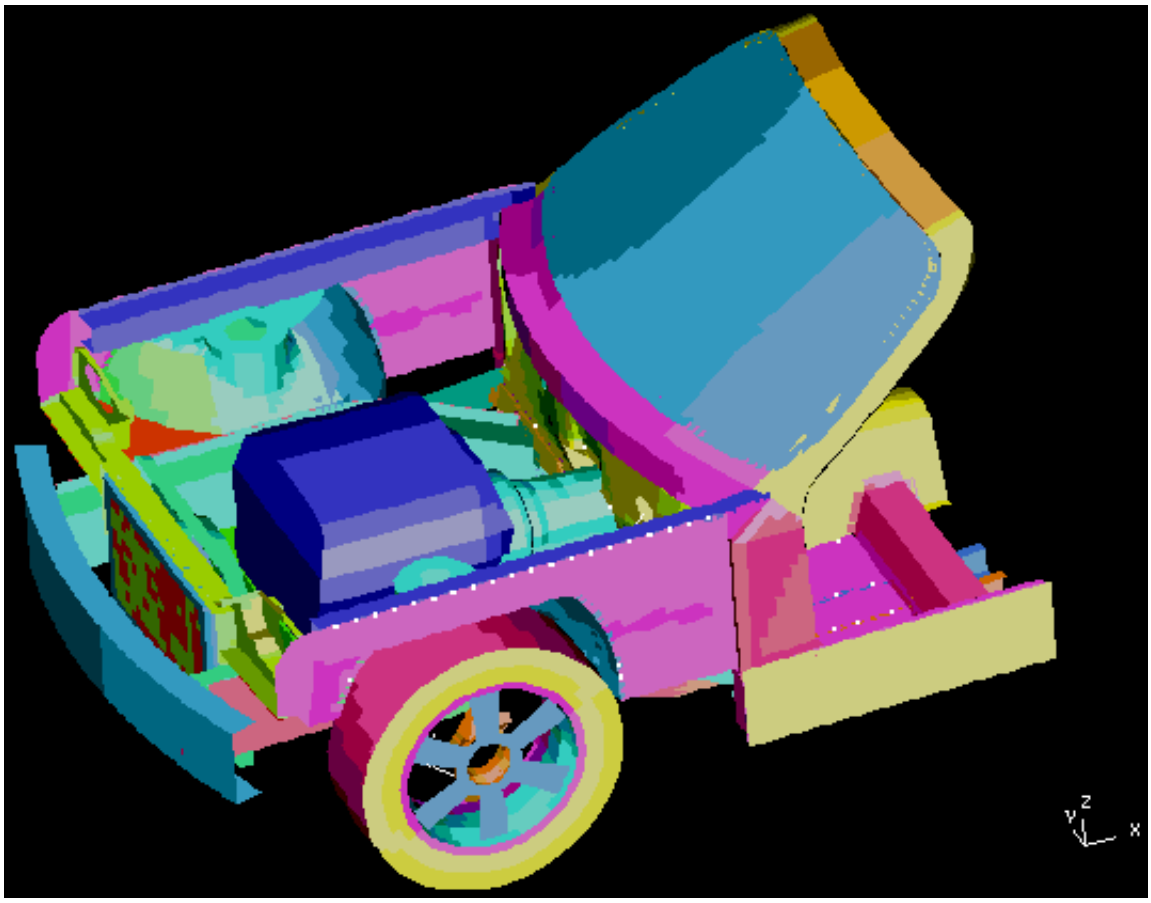
This tool allows you to find panels that are not attached to any connection. This facility enables you to quickly check that the panels you expect to be connected together,either by bolts or spotwelds, actually are welded together!

To see how the function works look at the following example.



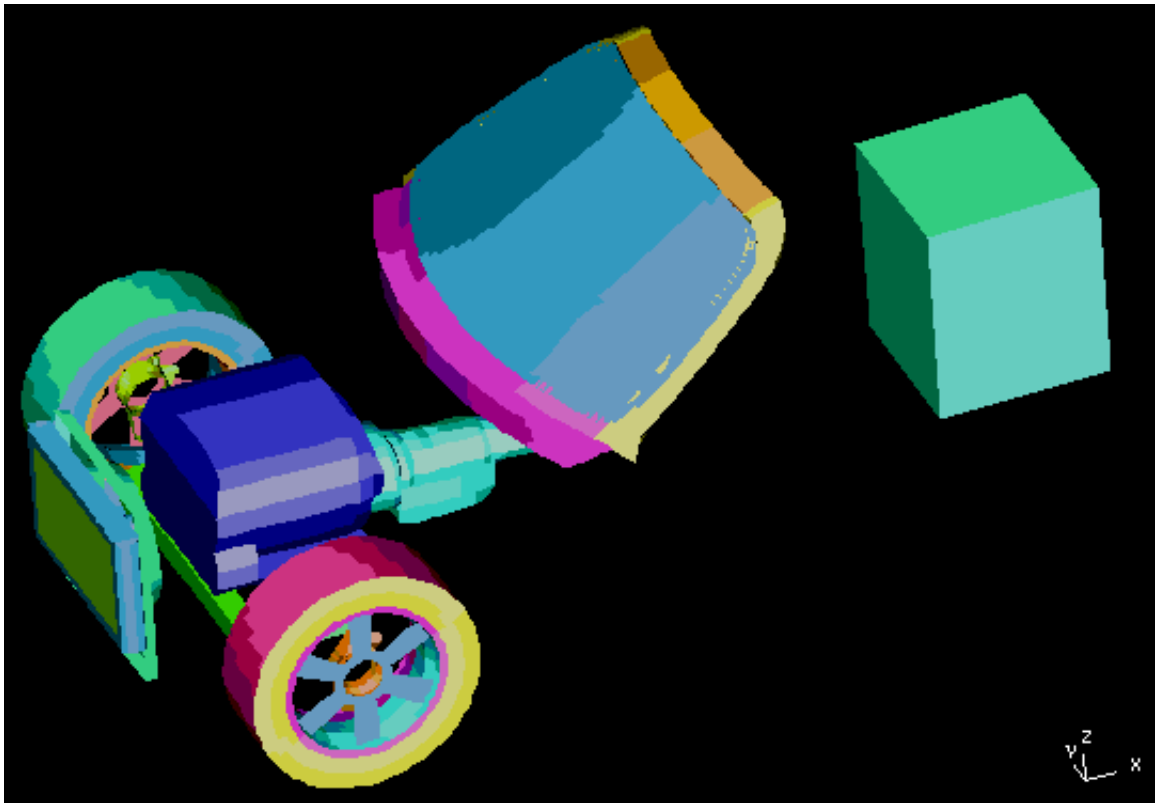
Example

The following figure shows the front of a vehicle. It has been welded/bolted together using the PRIMER connections ability. We want to find which panels are not attached by any spotweld nor beam connection.



Press the **Apply** button.

PRIMER blanks the model, and unblank any parts in the model that are not attached together by any connection.



This shows that the radiator, wheels, engine, gearbox, windscreen and screenrail are not connected to the rest of the vehicle. In this case the screenrail should be connected by spotwelds! We can now go back and fix this before submitting the job. Doing this quick check can help find problems which may be missed otherwise.

Reset blanking and Store blank status

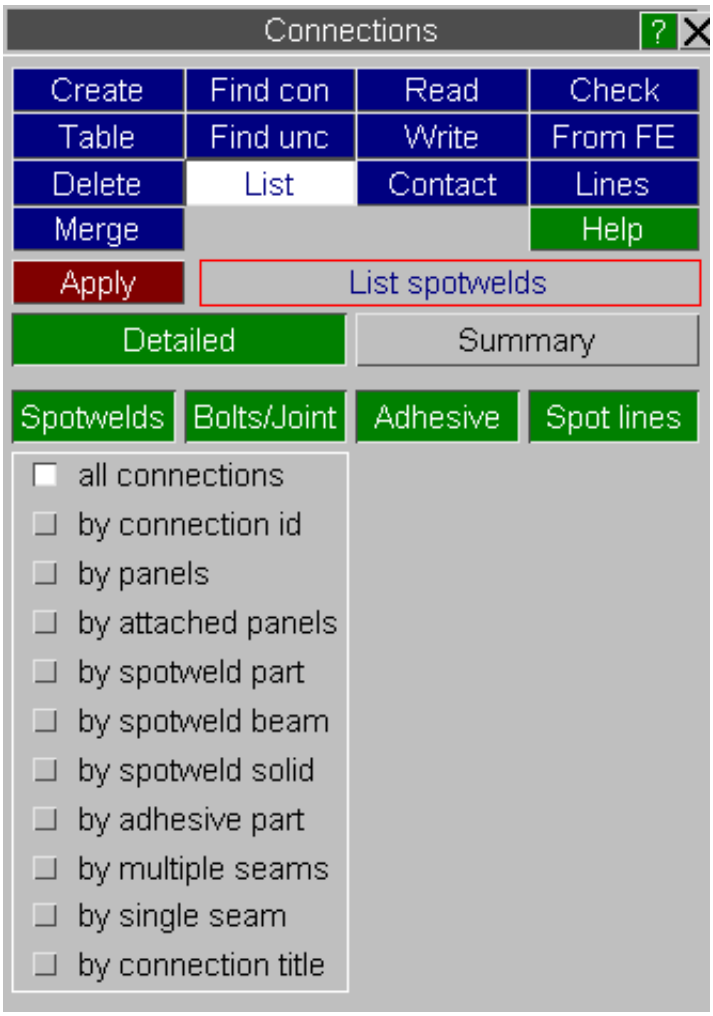
If you want to undo the blanking that PRIMER has just done you can press **Reset blanking**. PRIMER will reset the blanking to the previous state. Using **Store blank status** will store the current blanking status - this is useful because when you exit the panel it goes back to the stored status.

6.12.6 Listing connections

This panel allows you to list connections.

You can specify which connections to review by a number of different methods. For more details of the different methods see section 6.12.0.

There is also an option that effects how the spotwelds are listed :
Summary or **Detailed**.

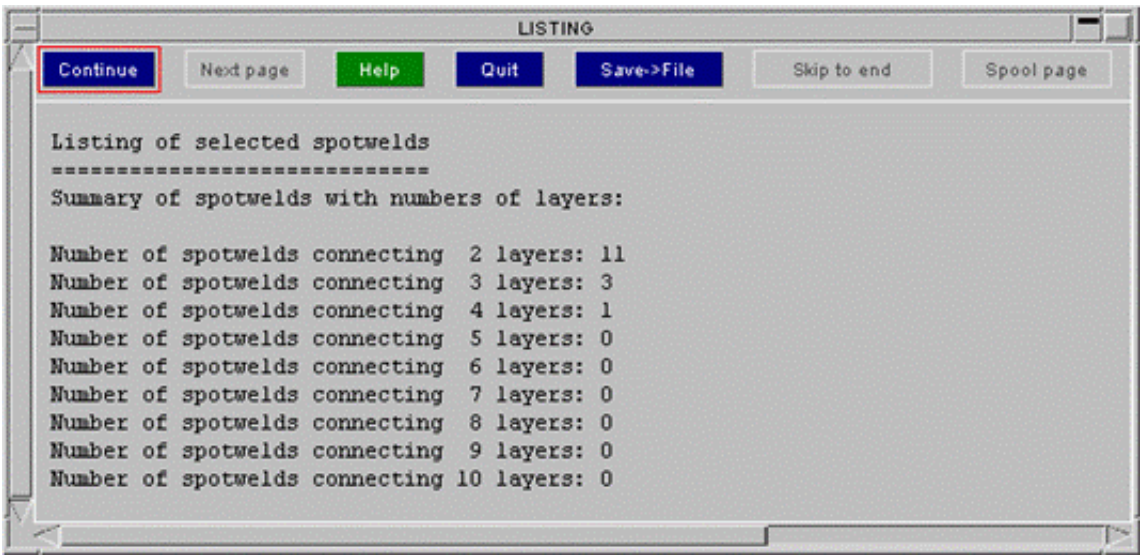


Summary or detailed listing

When you select connections to list you can choose how connections are listed. The options are either **Detailed** or **Summary**.

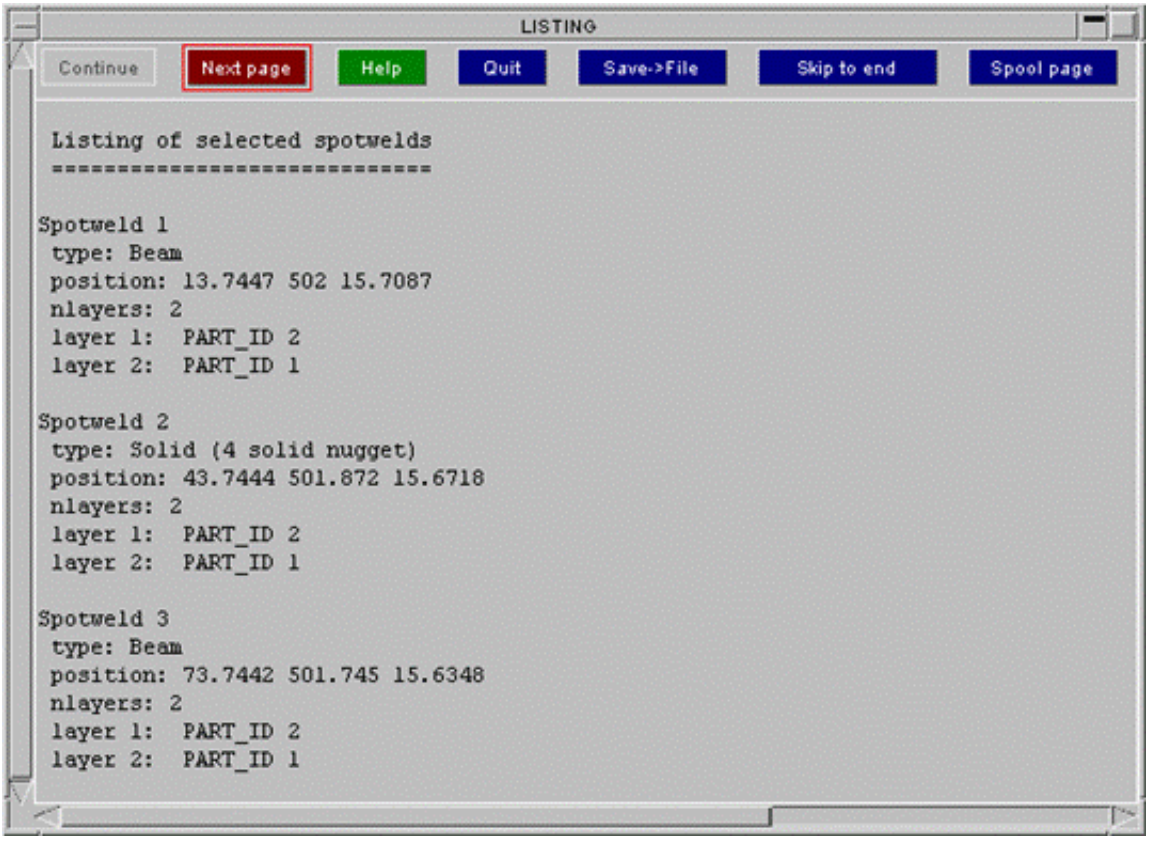
Summary listing

In summary listing PRIMER shows how many connections tie 2, 3, 4... panels together. This is useful as a quick check. If you know that the maximum number of panels that you weld for any spotweld is 3, and there is a 4 noded weld then there is an incorrect weld.



Detailed listing

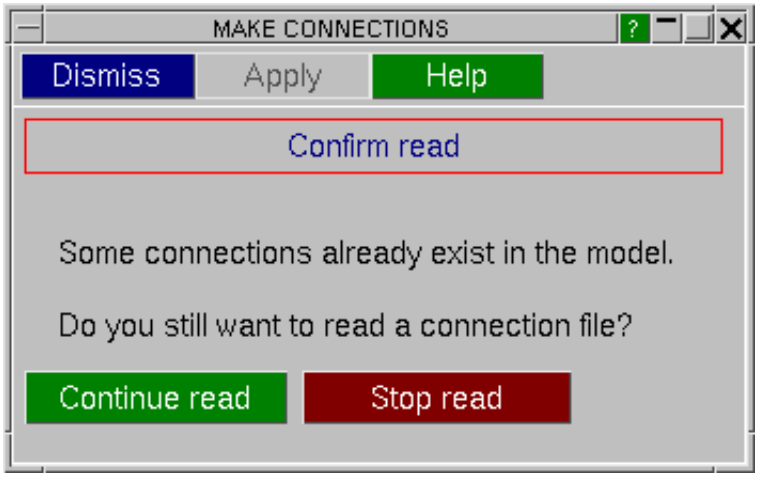
In detailed listing PRIMER shows the coordinates and panels of each connection. The format of the listing is similar to a PRIMER connection file.



6.12.7 Reading connections from a file

When you go to read a connections file, if your model already contains connectionsPRIMER will ask you if you want to continue with the read or not (see figure on right). Press either **Continue read** to continue or **Stop read** to finish.

If your model does not contain any connections the main read panel will be displayed.



The main read panel allows you to read mesh independent connections from a PRIMER spotweld file, an xml connection file, a Catia spotweld file, a UG file or other type of file. PRIMER can also store a user defined script using PRIMER's javascript functionality. The script can be written to read a particular file format and then is used through this panel to read connection data into PRIMER. See section 10 for information on PRIMER's scripting ability.

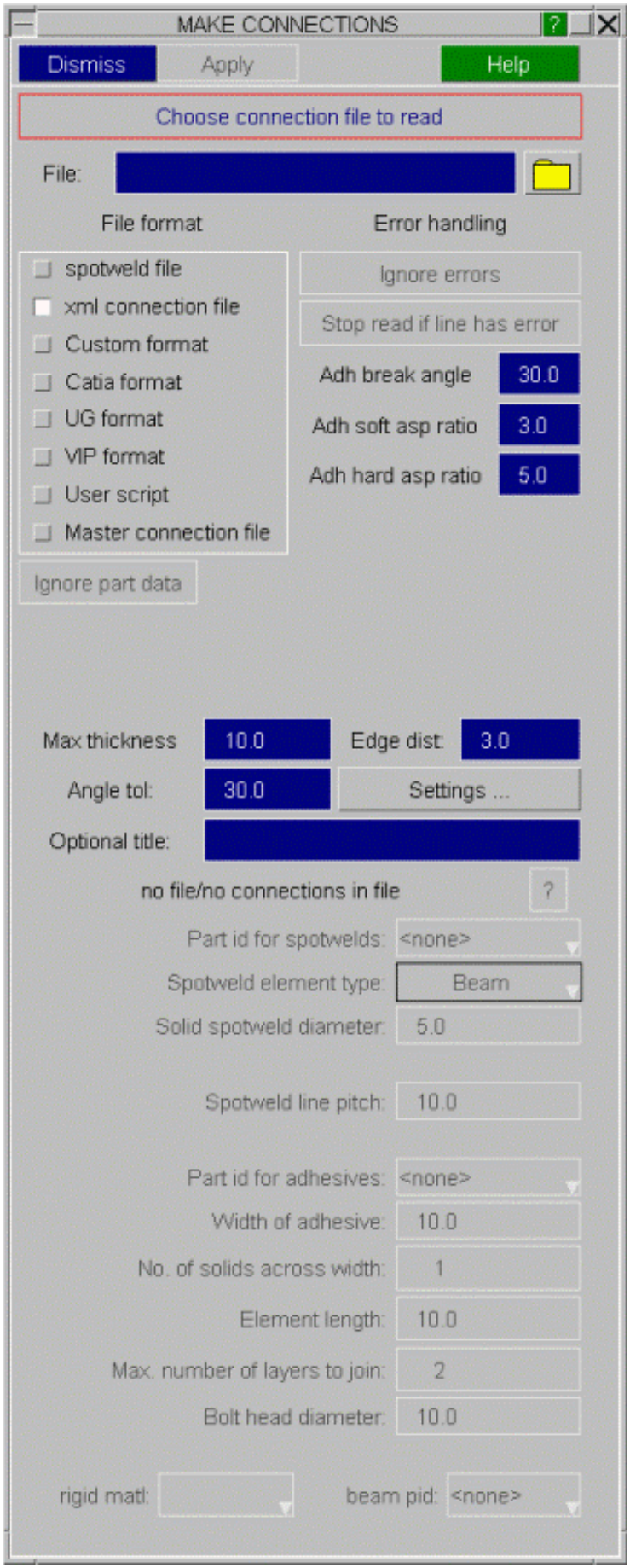
Choosing part for beams/solids and filename

Choosing file format

Error handling

Options

Actually reading the file



Specifying a title

An optional title can be specified when reading in connections. This title will be applied to every connection read in. This is useful for keeping track of where connections came from in your model. For example, you can set the title to be the same as the name of the file read in. Connection titles can be displayed in the connection table.

Choosing part for beams/solids and filename

PRIMER needs to know what type of spotwelds to create and which part to put the spotwelds into. These can be set with **Part id for spotwelds** and **spotweld element type**. Additionally if you are making spotweld solids PRIMER needs to know what size to make the solid spotwelds. This is set with **solid spotweld diameter**.

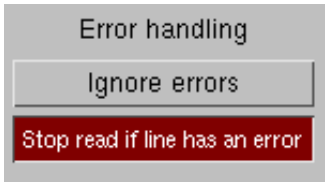
To select a part type in the part number, or you can use the standard popup functions (right click) to select or create the part. The part **must** use material type **`*MAT_SPOTWELD`** (and **`*SECTION_BEAM`** type 9 for beams). Once the part has been selected or created the part number will be displayed in the box:
If there is only one part in the model that is suitable (i.e. for beams if the part uses material `*MAT_SPOTWELD` and section type `*SECTION_BEAM`, or for solids if the part uses material `*MAT_SPOTWELD`) then PRIMER will automatically select it. Otherwise you will have to select it. Other inputs are available and will become ungreyed once the file to be read in has been selected.

Choosing file format

PRIMER is designed to be able to read a wide variety of files. PRIMER has it's own spotweld file format. If this format is used then PRIMER knows which fields are which in the file and so it can be read in one step. Select **Primer spotweld file**. PRIMER also supports Catia and UG spotweld files and the format is set automatically in PRIMER so it can also be read in one step. Select **Catia format** or **UG format**.

If the file you want to read is not a PRIMER or Catia spotweld file then you need to tell PRIMER how it should be read. Select **Custom format**.

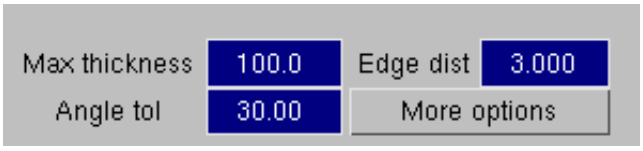
Error handling



When PRIMER is reading a **PRIMER spotweld file** there are 2 possible actions that can be taken if an error occurs when reading the file: Either stopping the read completely when an error is found, or ignoring an error.

To show the two options, consider the following example. PRIMER is reading a line from the spotweld file and has already read the X, Y and Z coordinates and panels 1 and 2. Now, when trying to read the third panel, an error occurs as PRIMER reads the string 'aaaa' from the file and it is expecting to read a panel number. If **Stop read if line has an error** is selected PRIMER will stop and give the line number of the error. If **ignore errors** is selected, PRIMER will just ignore the third panel and make the weld (if possible) between panels 1 and 2 at (X, Y, Z).

Options



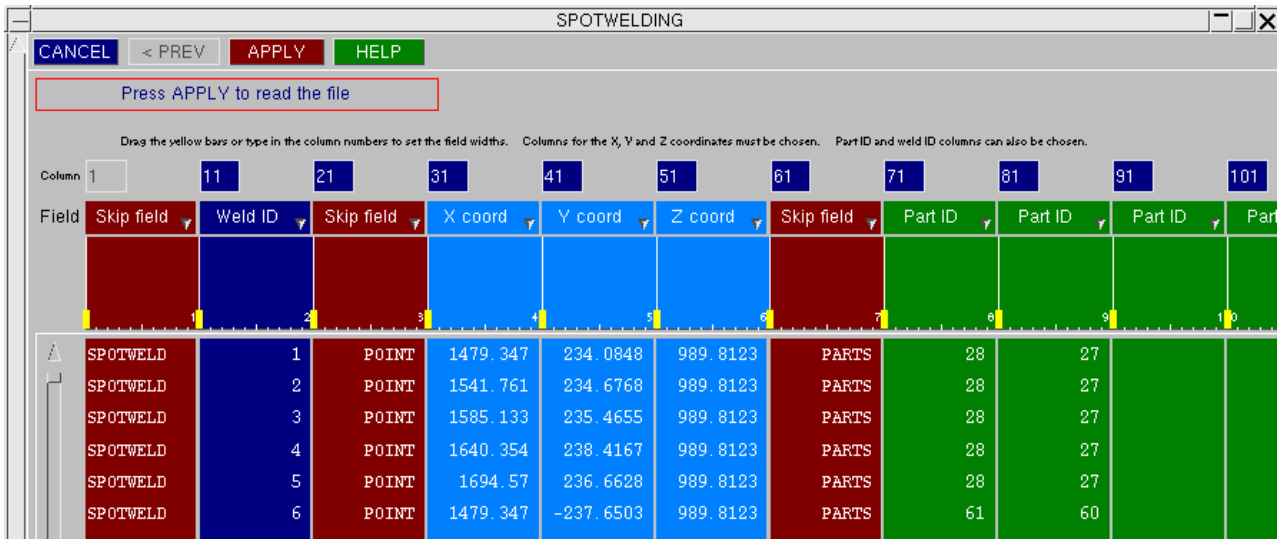
These options control how the spotwelder works. They are identical to the options that are used when creating spotwelds. For more information see the section on spotweld options

Reading the file

Pressing **APPLY** will start reading the file. How the file is read is dependant on the file format chosen: PRIMER spotweld format, Catia spotweld format or Custom format.

PRIMER spotweld file

PRIMER will read the first 50 lines of the file and put a preview on the screen.



The preview should be similar to the image above. PRIMER knows what each field is so everything is set for you automatically. Pressing **APPLY** will now actually read the file. If the file is not a PRIMER spotweld file or you want to stop the read press **CANCEL** to return to the main reading panel.

XML connection file

From version 9.3 PRIMER can use a new xml format for connections. This contains more information than the PRIMER spotweld file and can be used to describe bolts and adhesive lines as well as spotwelds. Each connection entry consists of connection type, point (or line info) and a list of the layers to be connected. A layer is typically a part, but it may consist of multiple parts, see modifying connection layers.

Layers provide a powerful way of ensuring that correct connectivity in a model is achieved, as a connection will only be passed as valid (or realized) if it successfully joins all the layers in its definition.

For spotwelds, the file includes the spotweld part ID, the FE type (beam, solid or multiple solids) and the diameter (for solid welds). For bolts, FE type is defined as NRB or merge (for these an optional rigid material id may be given) and the diameter. For adhesives, adhesive width, number of elements across the width and element length are stored. Also for adhesive, the part ID of the solids in the connection are stored, along with additional information for the path of the adhesive.

Prior to read, PRIMER will scan the file to check if any required information is missing (e.g. part ID or diameter), and you will be able to supply this to the edit panel. Such information will **only** be used for connections which have parameters missing in the xml data. See Choosing part for beams/solids and filename for more details.

On completion of read, if any connections have not been made they will be put on the connection table for you to investigate and fix. For more details see section 6.12.2.

For models with existing connections, the xml connection file may be written out from the connection table using the "Update & write to file" function.

For more information on the format see the Spotweld file formats section.

Catia spotweld file

PRIMER will read the Catia spotweld file automatically and go straight to step 7 at the end of the reading process. The format of the file is set within Primer and therefore it is important to check that the Catia weld file matches the Catia format set by PRIMER.

UG spotweld file

PRIMER will read the UG spotweld file automatically and go straight to step 7 at the end of the reading process. The format of the file is set within Primer and therefore it is important to check that the UG weld file matches the UG format

set by PRIMER.

VIP spotweld file

PRIMER will read the VIP spotweld file automatically and go straight to step 7 at the end of the reading process. The format of the file is set within Primer and therefore it is important to check that the VIP weld file matches the VIP format set by PRIMER.

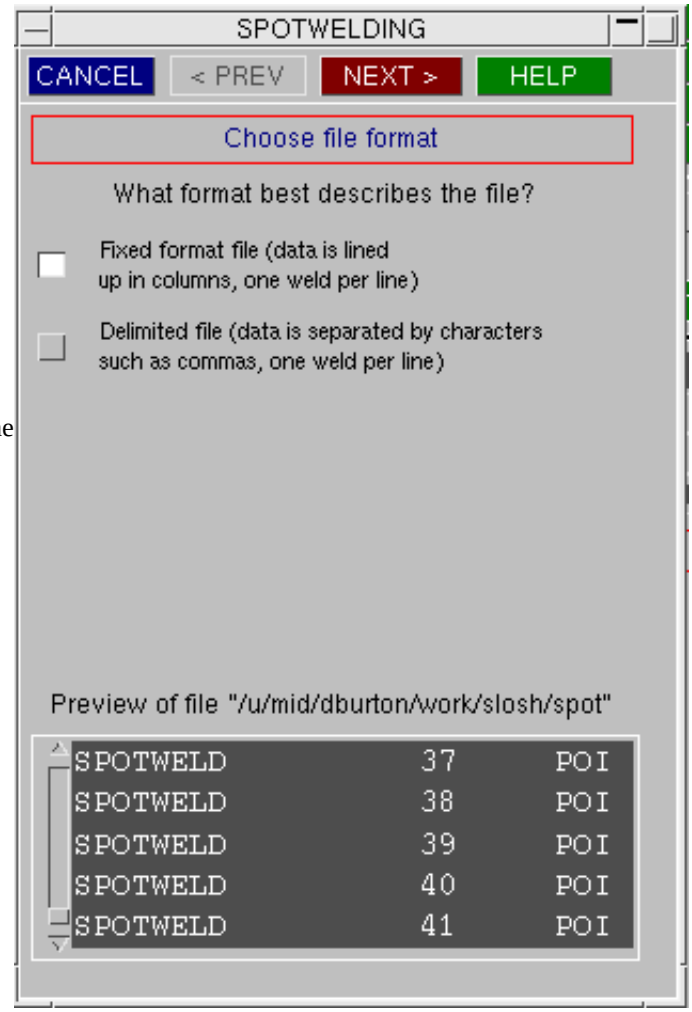
Master connection file

PRIMER will read the Master Connection File (MCF) automatically and go straight to step 7 at the end of the reading process. The format of the file is set within Primer and therefore it is important to check that the MCF matches the MCF format set by PRIMER.

Custom format

If the file is a custom format, PRIMER will ask you a series of questions to determine the format of the file. Once the format has been determined you will be able to read the file.

Step 1: Fixed/Delimited



The first step is to determine the format of the file. PRIMER will try to read 2 types of files:

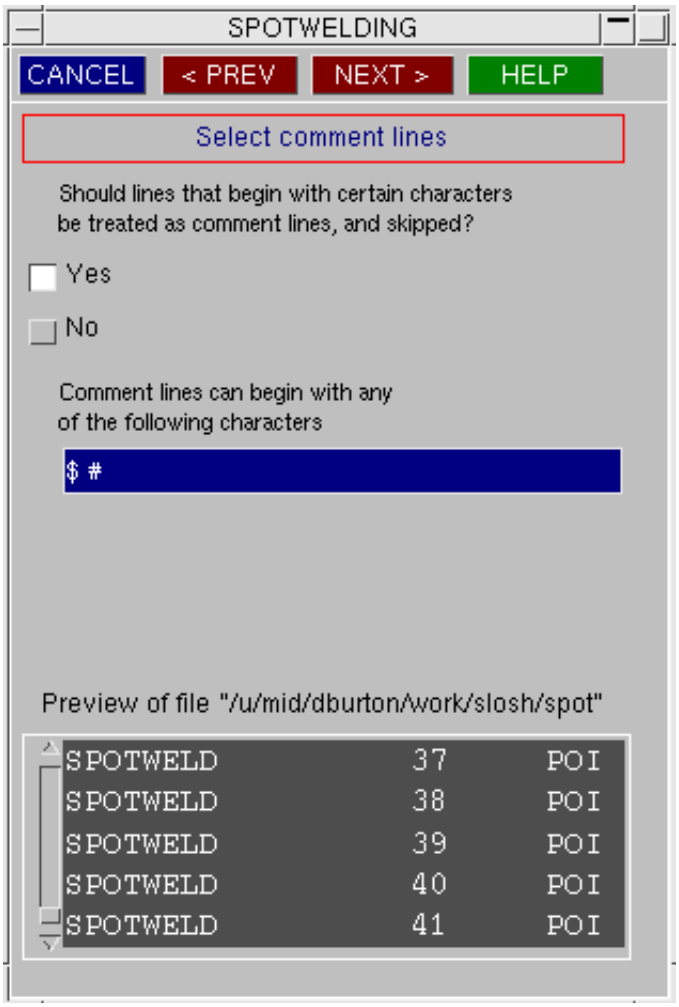
Files that have fields of fixed widths (these are like the fields in LS-DYNA keyword files that are generally 10 characters wide).

Files that have fields that are separated by a specific character such as a comma. An example of a file like this would be a CSV file produced by a spreadsheet program.

PRIMER shows a preview of the file at the bottom of the panel. You can use this to view the file and determine which of the 2 formats best describes the file.

Once you have chosen the format that best describes your file press **NEXT >** to go onto the next step. **CANCEL** will return you to the main screen.

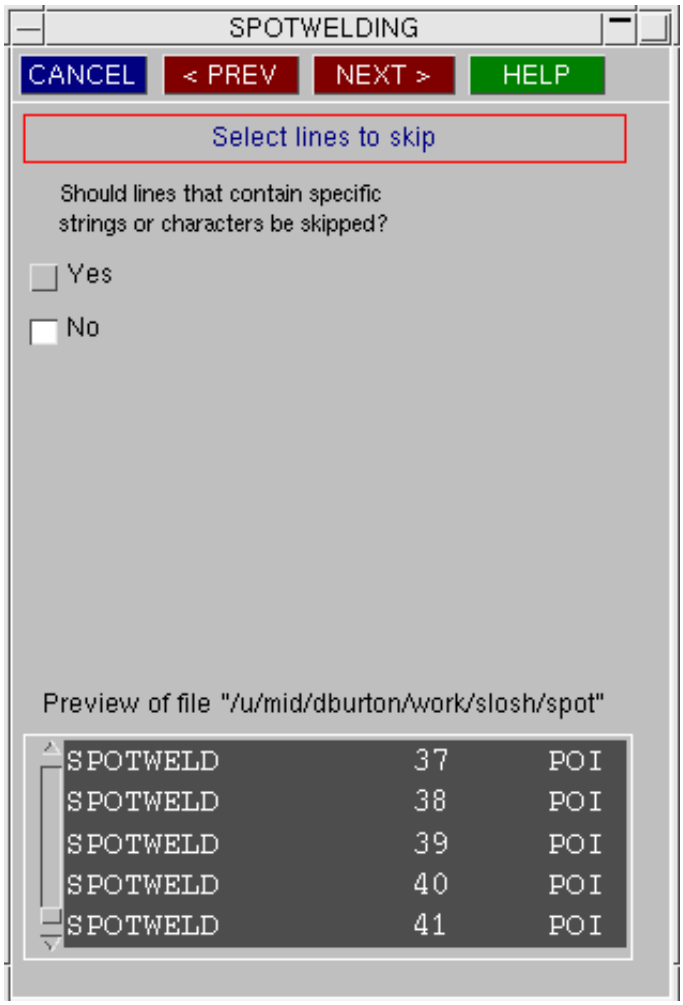
Step 2: Comment lines



The second step is to determine if any lines in the file should be treated as comment lines and skipped. This is like comment lines in a LS-DYNA keyword file that can begin with a '$' character.

Once you have chosen the comment setting press **NEXT >** to go onto the next step. To go back to the previous step press **< PREV**. **CANCEL** will return you to the main screen.

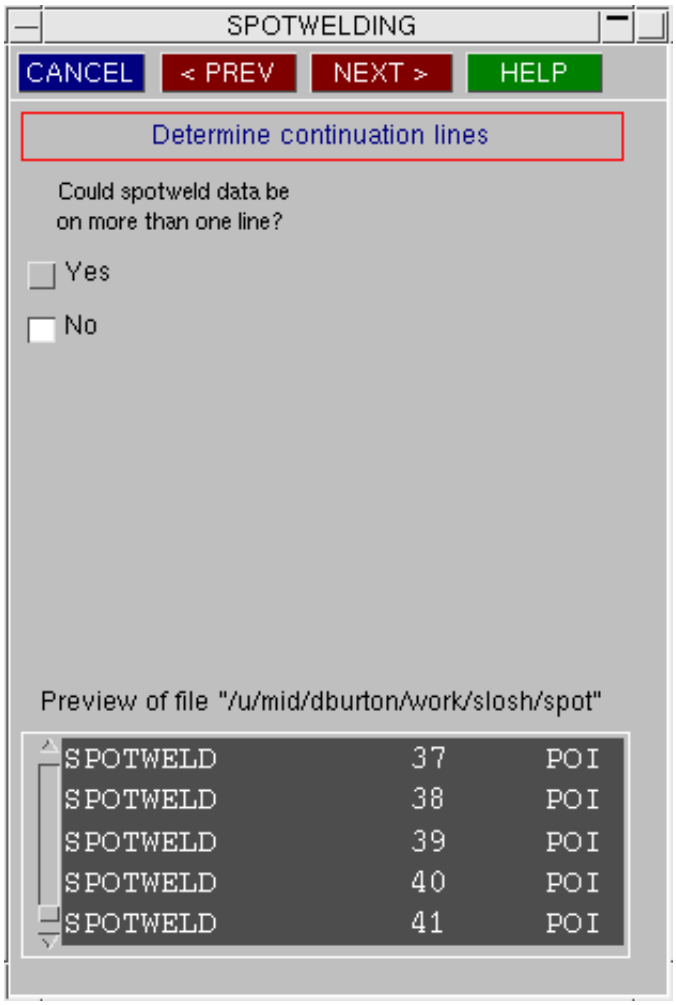
Step 3: Skip strings and characters



The third step is to determine if any lines in the file that contain specific strings or characters should be skipped.

Once you have chosen the string and character settings press **NEXT >** to go onto the next step. To go back to the previous step press **< PREV**. **CANCEL** will return you to the main screen.

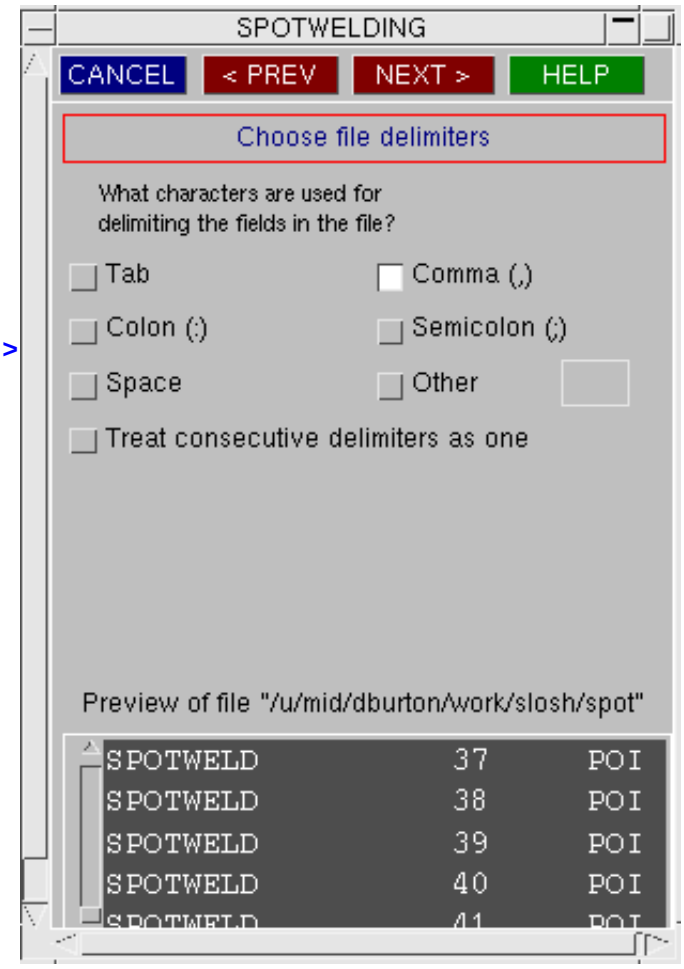
Step 4: Continuation lines



The fourth step is to determine if spotweld data can continue onto a second line. It is strongly recommended that you have one line per spotweld. However, if spotweld data can continue on to a second line PRIMER will try to read it with these settings.

Once you have chosen the continuation setting press **NEXT >** to go onto the next step. To go back to the previous step press **< PREV**. **CANCEL** will return you to the main screen.

Step 5: Choosing delimiters



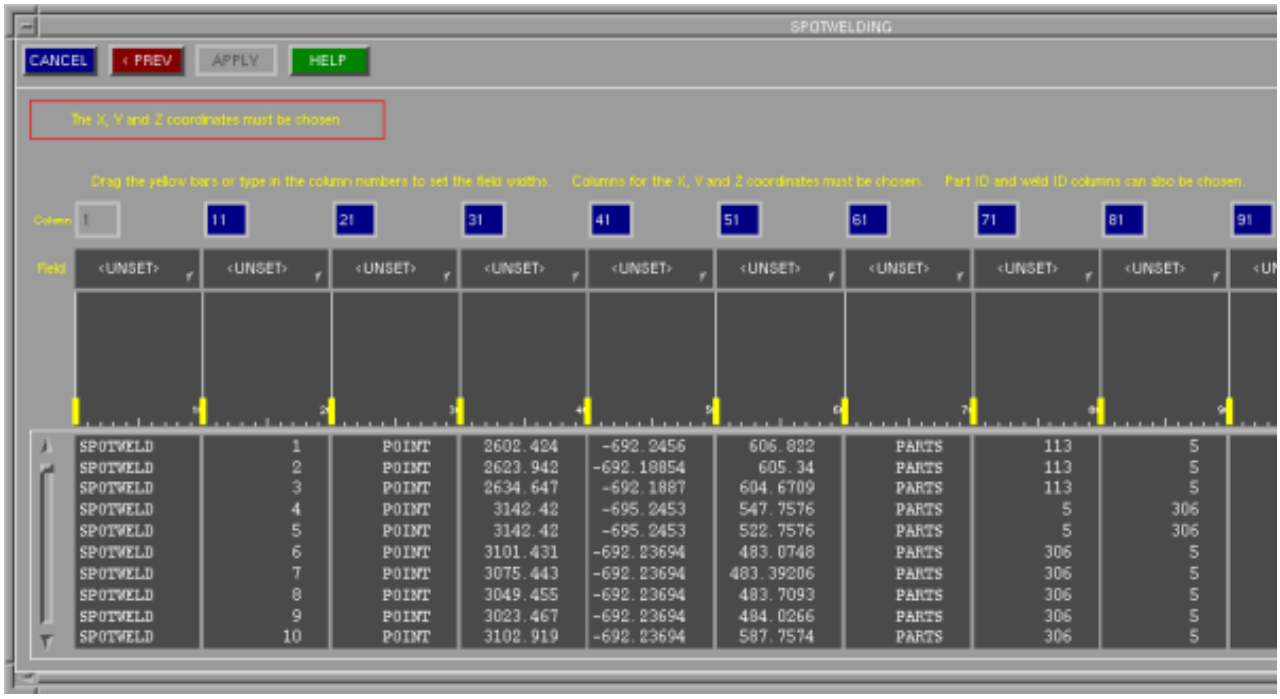
The fifth step is only done if you are reading a file in delimited format. You need to tell PRIMER what character(s) to use as field delimiters. Additionally there is a switch to **treat consecutive delimiters as one** delimiter. This is most commonly used when the 'space' character is used as the field delimiter. If some of the fields are separated by more than one 'space' then PRIMER will treat it as a single 'space'.

Once you have chosen the delimiter setting press **NEXT >** to go onto the next step. To go back to the previous step press **< PREV**. **CANCEL** will return you to the main screen.

Step 6: Choosing fields

The sixth step allows you to choose which fields are which. PRIMER shows a preview (the first 50 lines) of the file showing how it will decode the fields from the settings you have chosen in the previous steps.

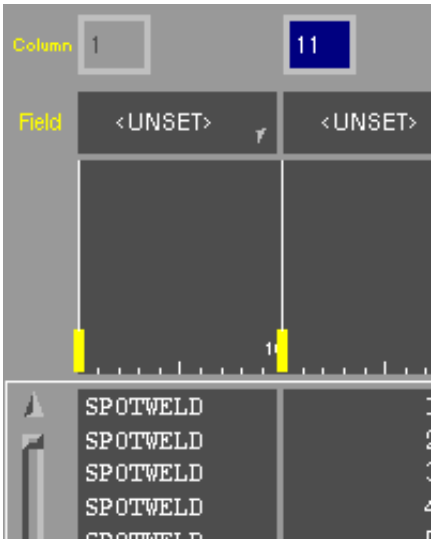
Choosing field types



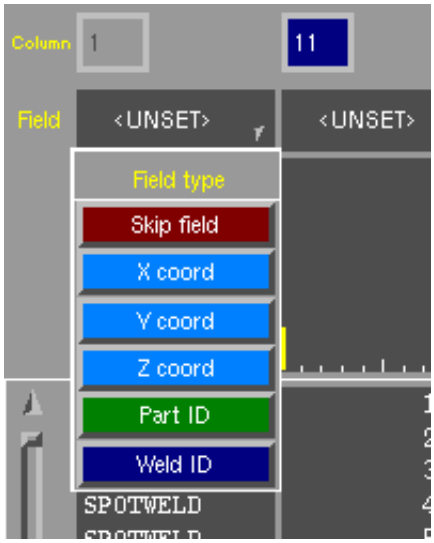
The image above shows the fields that PRIMER has read. If it is incorrect you can go back and change the settings as necessary. In this example the fields are:

Field	columns	description
1	1-10	Skip this text
2	11-20	Weld ID
3	21-30	Skip this text
4	31-40	X coordinate
5	41-50	Y coordinate
6	51-60	Z coordinate
7	61-70	Skip this text
8	71-80	Panel ID 1
9	81-90	Panel ID 2

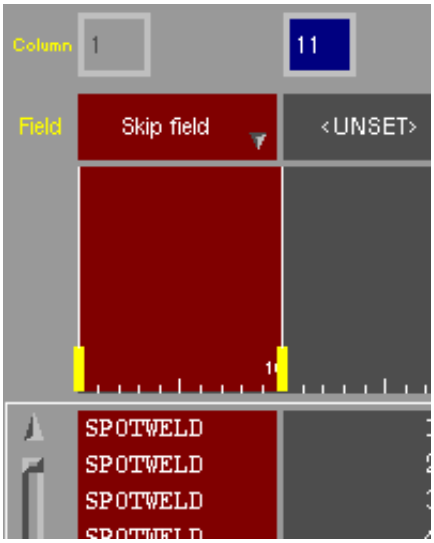
Initially all the fields are **<UNSET>**. Use the popup to change the field to the required type. For example to change field 1 to 'skip this field':



Field is initially unset

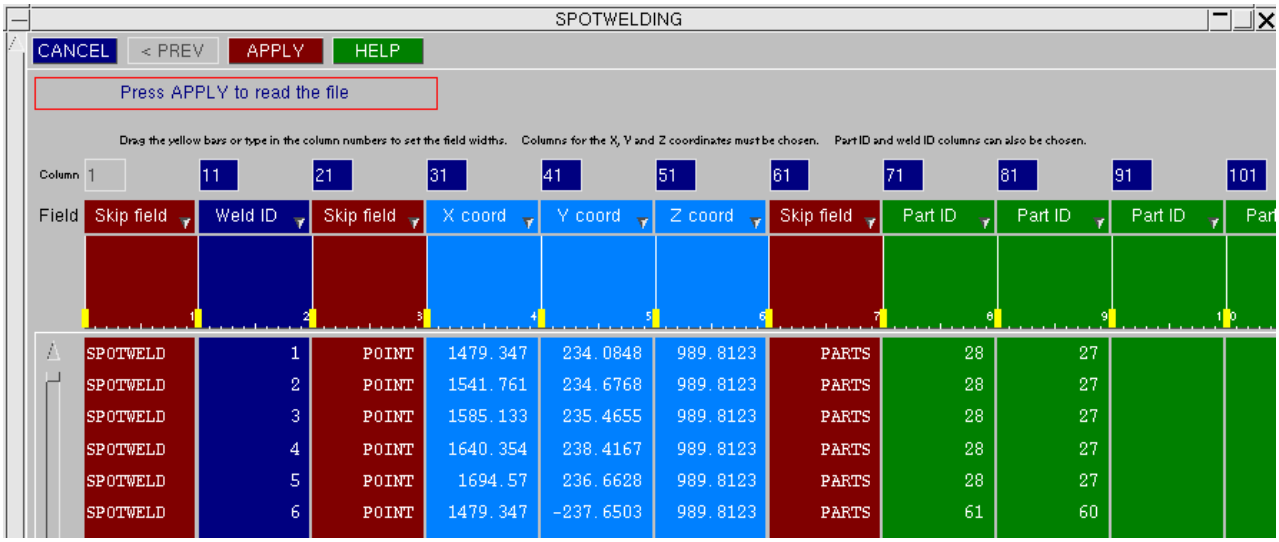


Use the popup and select **Skip field**



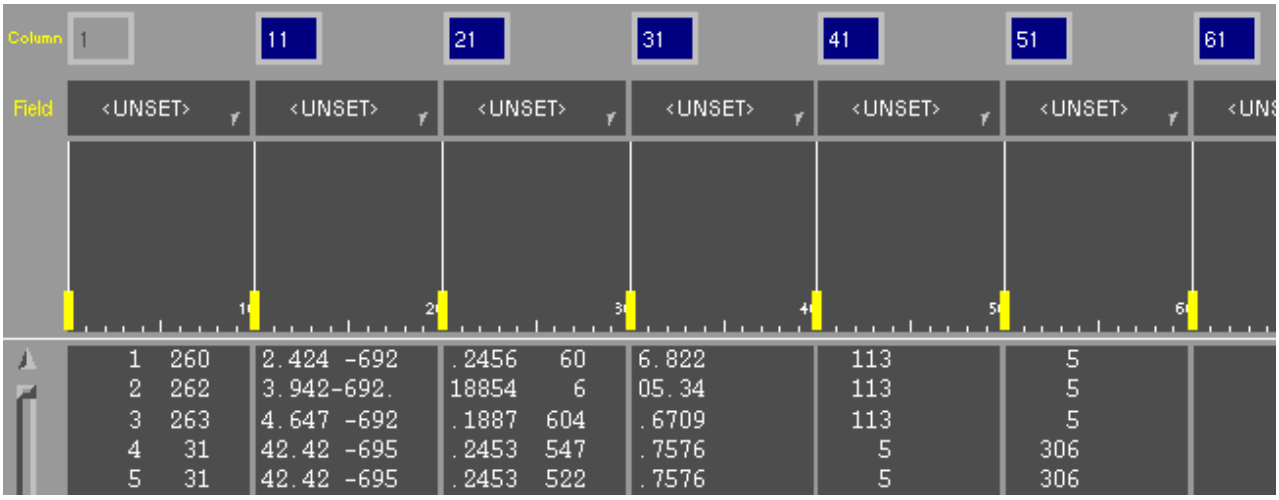
Field is now set to **Skip field** and coloured to show it is set

Repeat this until all the fields have been set to the required values. You **MUST** define the **X coord**, **Y coord**, **Z coord** and **Part IDs**.

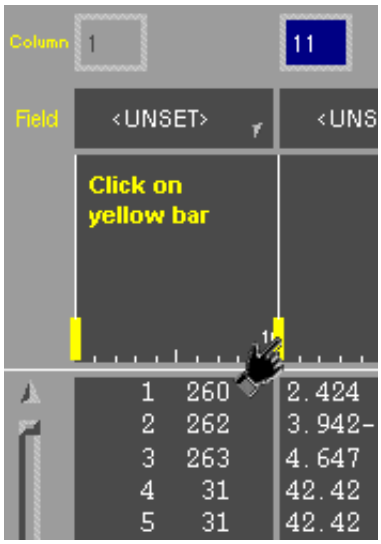


Choosing field widths

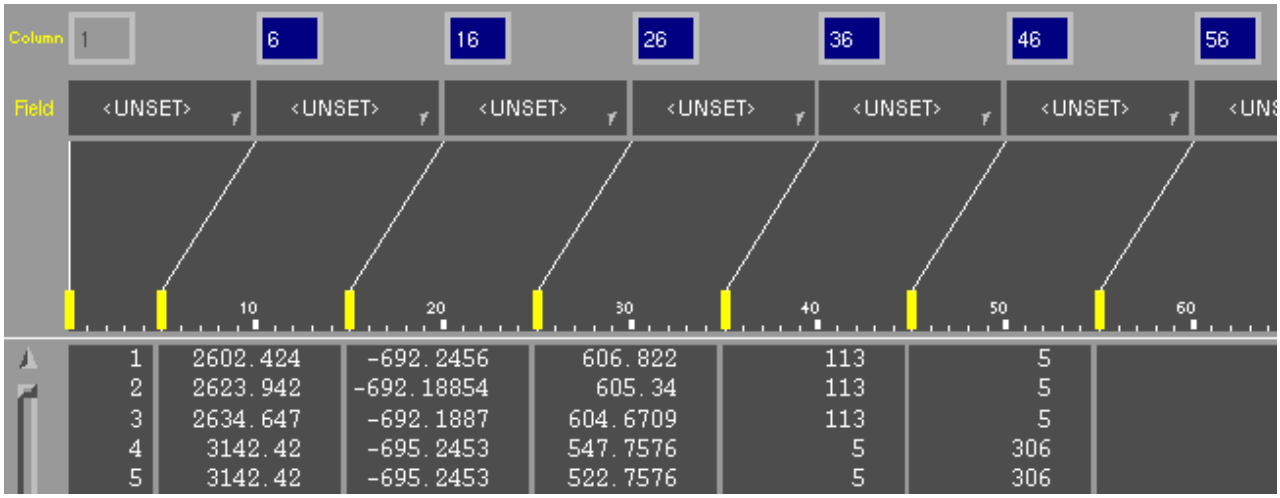
In the example image below the fields are not just 10 columns wide. We need to change the column widths.



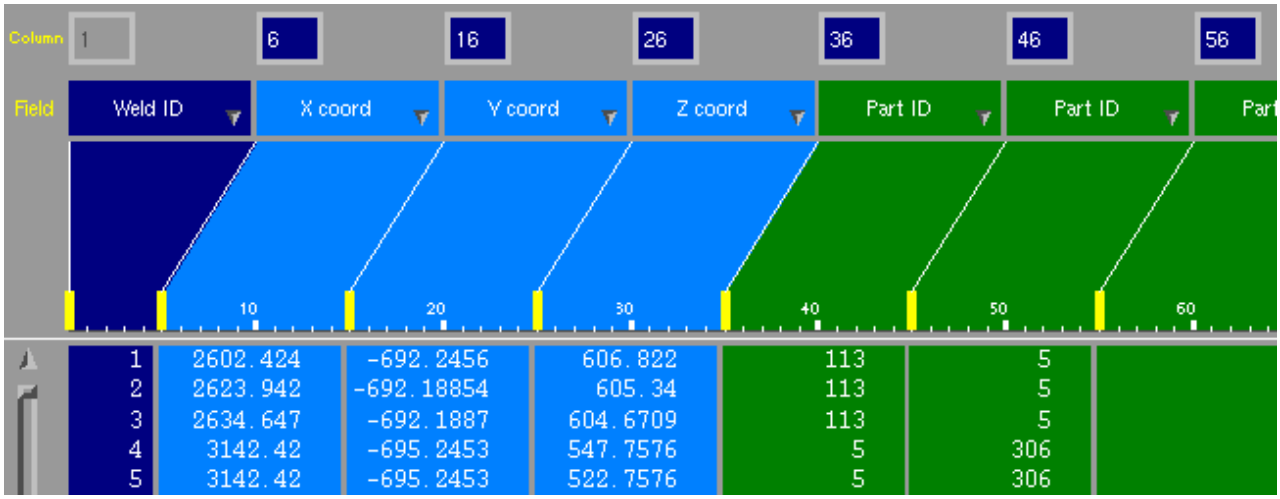
Field 1 should be columns 1-5, not 1-10. To change this you can either type in the new column numbers in the blue boxes or you can drag the columns to the correct sizes. The yellow bars enable you to drag the columns by clicking on one of them with the mouse and dragging it to the left or right until it is in the correct place.



Repeat this process until all the fields are the correct width



The field types can then be chosen as necessary.



Step 7: Warnings and errors after reading the file

After PRIMER has finished reading the spotweld file it will display a listing panel giving information on the welds it has not been able to create.

PRIMER will also do a check of all the welds that it has created to see if any are too close together (the pitch between the welds is too small).

Step 8: Fixing bad welds

If any of the welds in the file could not be created PRIMER will put them onto the connection table and you can use it to visualise and fix the welds. For more details see section 6.12.2.

6.12.8 Writing spotwelds to file

This panel allows you to write mesh independent spotwelds to a PRIMER spotweld file, an XML connection file, a UG weld file, an IGES file or a Master Connection File. Select which file type using the lower radio button selection.

The XML file also supports the export of bolts and adhesive connections.

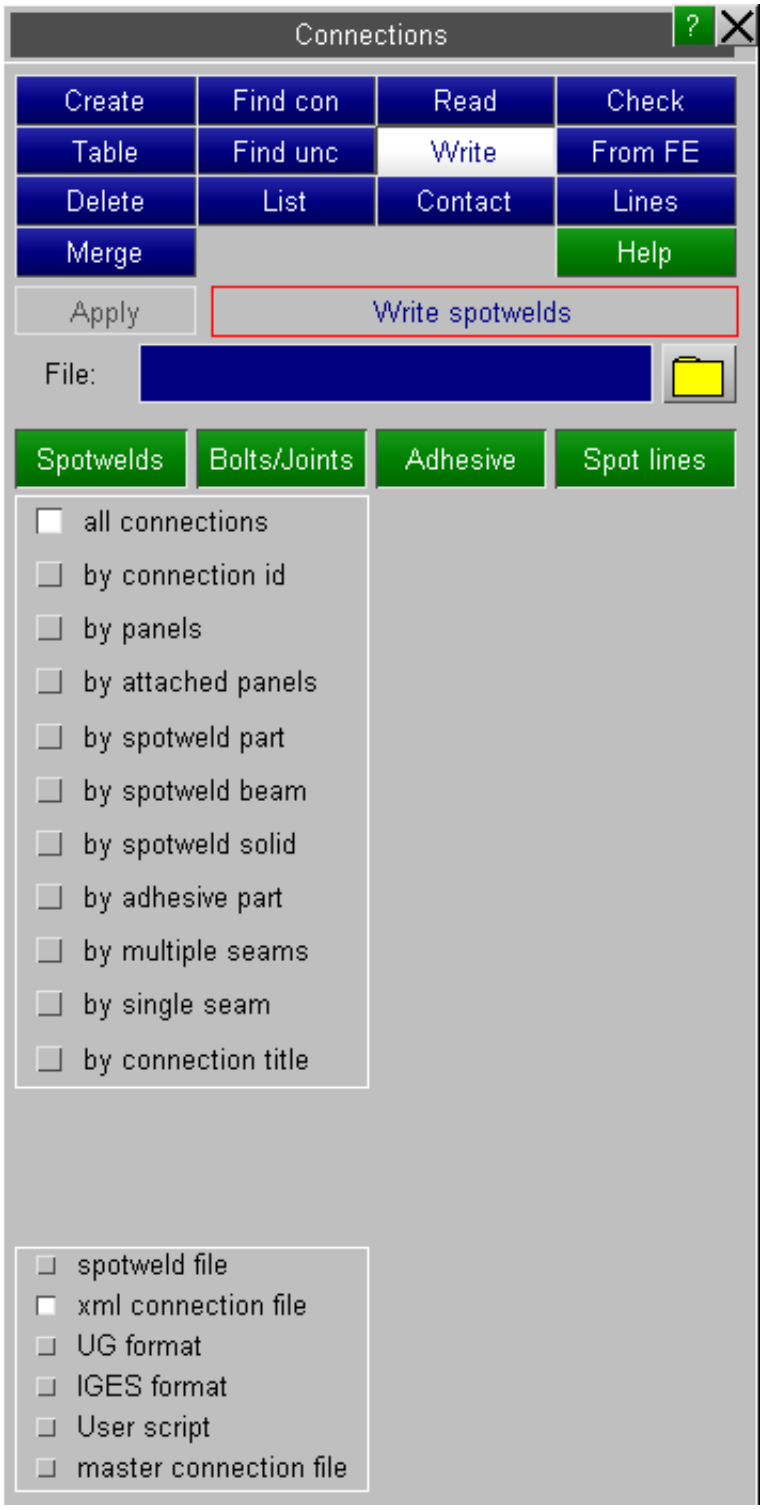
When choosing the IGES option, adhesive connections are written as IGES lines. All other connection types are written as IGES points.

You can specify which connections to write by a number of different methods. For more details of the different methods see section 6.12.0.

Type in the name of the file you want to write or use the file selector button.

Once you have specified the filename, which connections you want to write and the file format press the **Apply** button to write the file.

A script can also be used to write connection data out in a user defined format. The user would create a javascript that would write connection data in the format desired, and then this script can be used through the connection write panel to write the connections in that format. See section 10 for more on PRIMER's scripting functionality.



6.12.9 Connection contact

Mesh independent spotweld beams/solids and adhesive solids (connections in Primer) are tied to their respective panel shells using tied contacts in LS-DYNA. For solids the preferred contact is *CONTACT_TIED_NODES_TO_SURFACE, for beams it is *CONTACT_SPOTWELD. These constrained contacts give the correct shear stiffness for a weld connection. Penalty (_OFFSET) contacts will generally not give adequate stiffness - vehicle models can show 10% underestimate of torsional stiffness when penalty contact is used for all spotwelds.

Modelling constrained contacts: Constrained contacts require rigorously correct modelling as they are incompatible with other forms of constraint and may interfer with one another. For example, if a *CONSTRAINED_NODAL_RIGID_BODY attaches to a node of a shell to which a spotweld beam is attached by *CONTACT_SPOTWELD, the spotweld will be released. Similarly, if a piece of foam is tied to a panel by a constrained tied contact, spotwelds cannot be attached to the same shells by a *CONTACT_SPOTWELD. The best solution to this problem is to transfer the nodes which will not tie out of the constrained contact to an _OFFSET contact.

Primer's **CONNECTION > CONTACT** function has been designed to create spotweld/adhesive contact(s) with automated deployment of _OFFSET contact where necessary.

6.12.9.1 Checking the connection contact

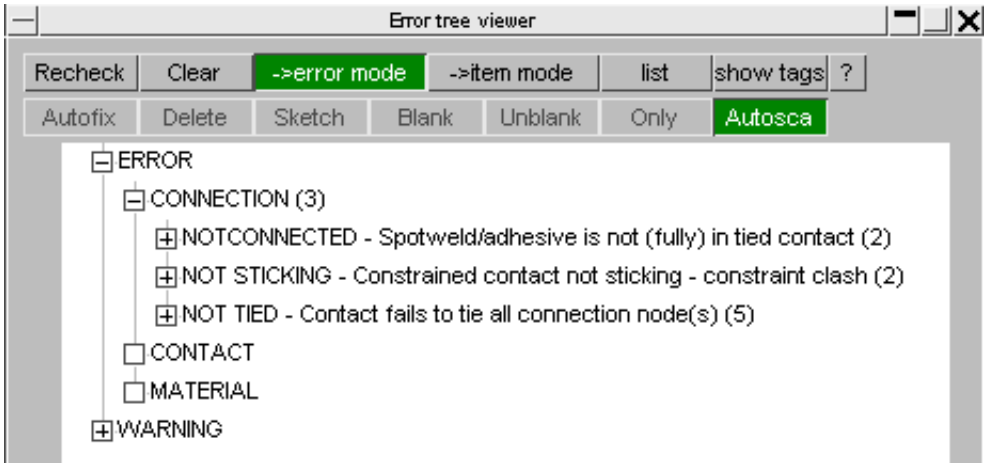
Connection contact is checked in a variety of contexts.

- when you run a model check - to report connections in error
- whenever connections are put onto the table - so their status is reported correctly. This is true so long as you have not de-activated the button **Use contact check to determine connectivity**. This option is only intended for quick edit of connections in very large models with complex contact definitions where user has no interest in their connectivity status.
- using **CONNECTION > CHECK > CONNECTIVITY**
- using **CONNECTION > CONTACT**

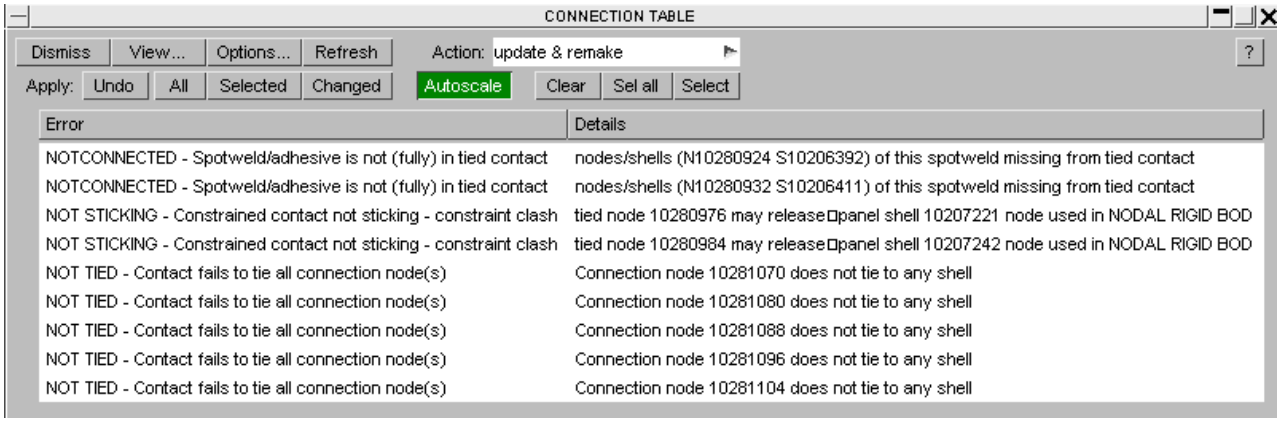
All use the same checking function and will report error code for each connection such as

- NOTCONNECTED - nodes or shells are missing from the contact definition
- NOT TIED - node and shell are in contact but fail to tie, usually because they are too far away
- NOT STICKING - constraint clash prevents contact from working, typically an NRB on the same panel shell as the weld
- BAD CONTACT - contact type is invalid

The results of model check are displayed in a tree view. The drop-down allows easy transfer of connections to the table, e.g. for re-making.



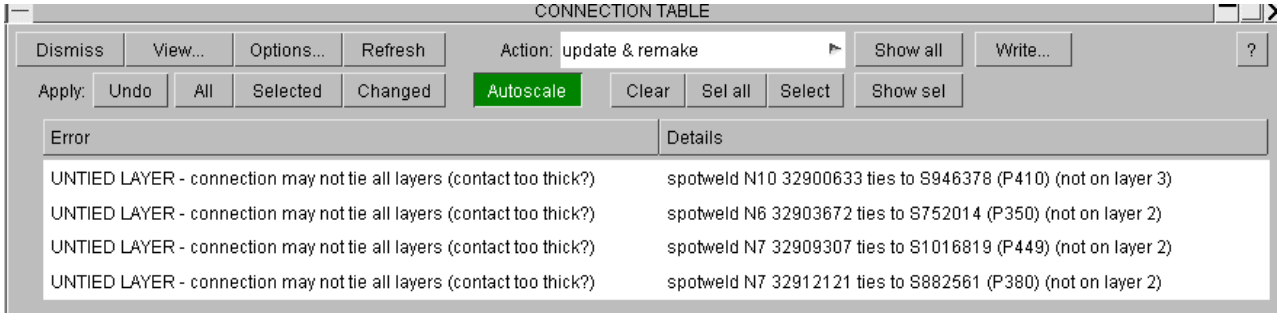
Details on the table provides useful information about the error.



6.12.9.2 UNTIED LAYER Connections tie but to wrong layer

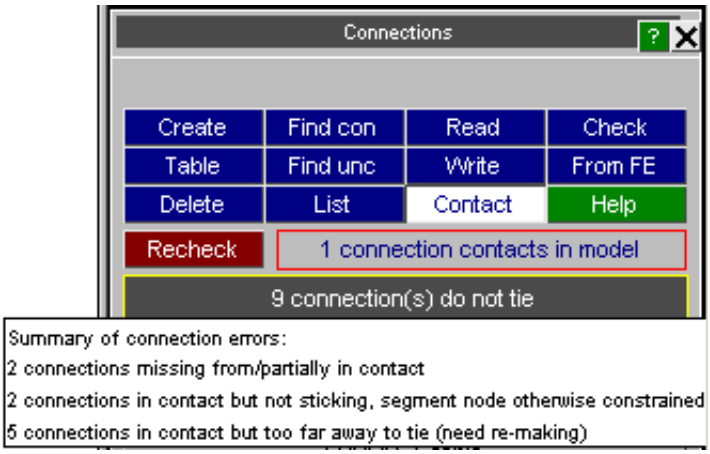
This error appears to be unfixable. Spotwelds are correctly projected, nothing is missing from the contact and all the nodes and their corresponding shells are tied.

It has actually resulted from the fact that the tied contact is defined with an excessive thickness wrt the shells being tied. This means that it is ambiguous to which shell a node is tied and Primer suspects that the actual shell tied will not be on the part in the layer definition. This error will not arise if the contact thickness is consistent with the shell thickness.

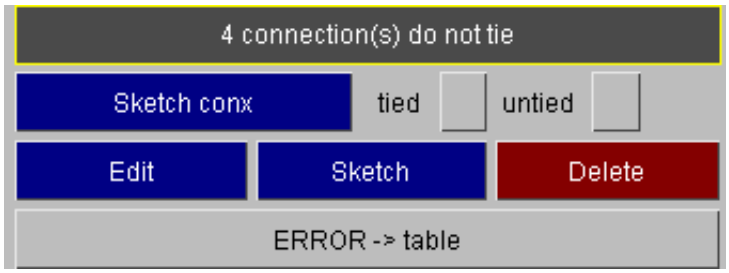


6.12.9.3 Using Connection > Contact

This function will report a summary of errors in hover text activated when the mouse is over the dark grey button.

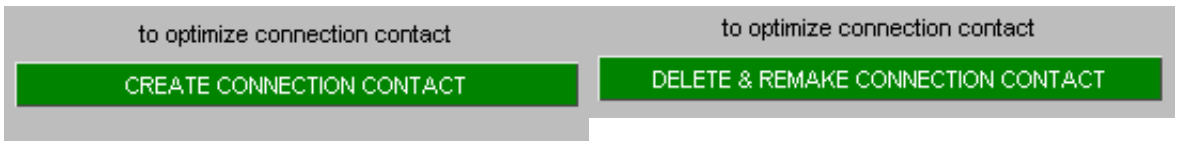


Tied and untied connections may be sketched with **Sketch Conx** and contacts used for connections may be accessed directly by **Edit Contact**. Problematic connections may easily be put on the table using **Error->table**, where the **show conx & panels** feature will enable you to display the area of interest. **Delete** may be used to remove contacts selected on the object menu.



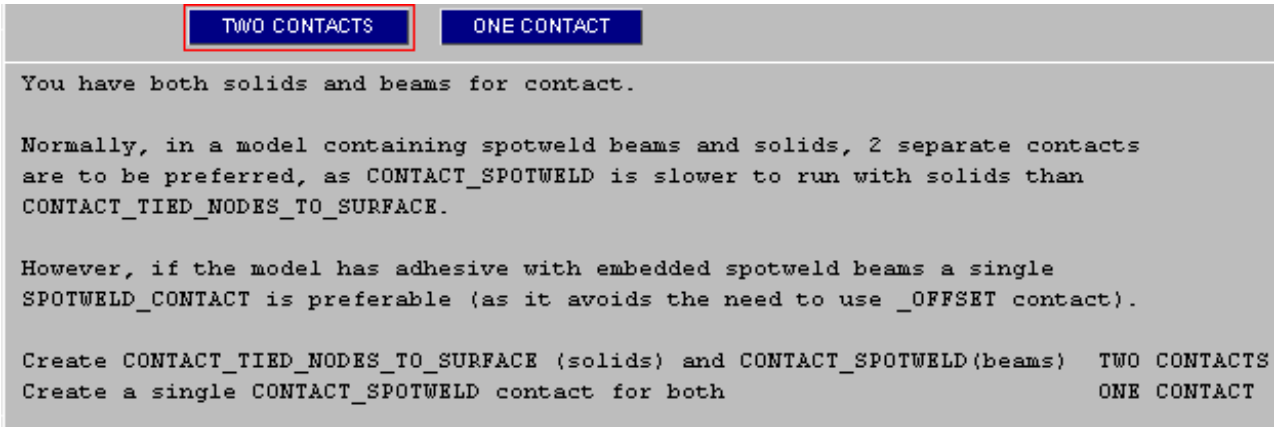
6.12.9.4 Connection > Contact: fully automatic fix

If there is no connection contact **CREATE CONNECTION CONTACT** will make one. If contacts already exists, the sister function **DELETE & REMAKE CONNECTION CONTACT** will delete the existing connection contacts and re-make them. These functions work on all connections and will yield an optimum solution with the minimum number of connection contacts. In a model with multiple contacts you may prefer to fix a sub-set of connections. To this end the functions **NOT-CONNECTED->contact** and **NOT-STICKING->offset** contact are available.



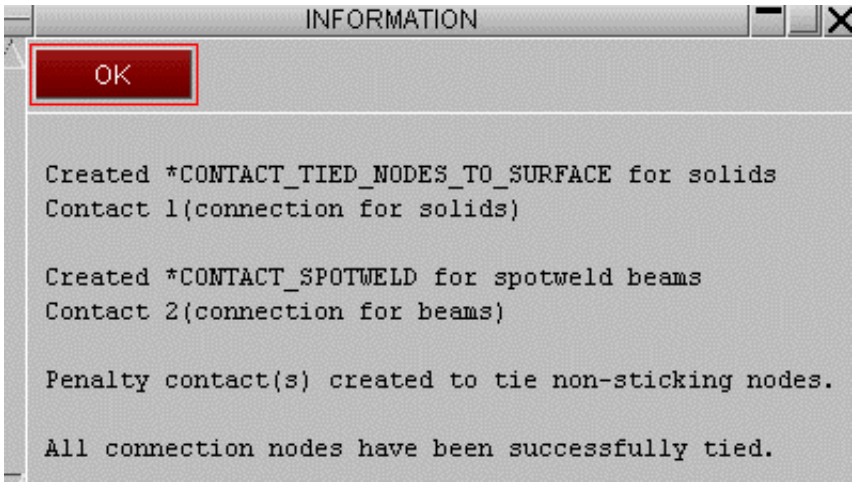
Models with beams and solids: A model with both spotweld beams and spotweld/adhesive solids will normally require separate contacts as CONTACT_TIED_XXX_TO_SURFACE cannot be used for beams (as it has no rotational constraint) and CONTACT_SPOTWELD is inefficient for tying solid elements.

However, if spotweld beams are embedded in a solid adhesive one may prefer to use a single CONTACT_SPOTWELD rather than have to create an _OFFSET contact (to avoid constraint clash). Therefore Primer will offer the choice.

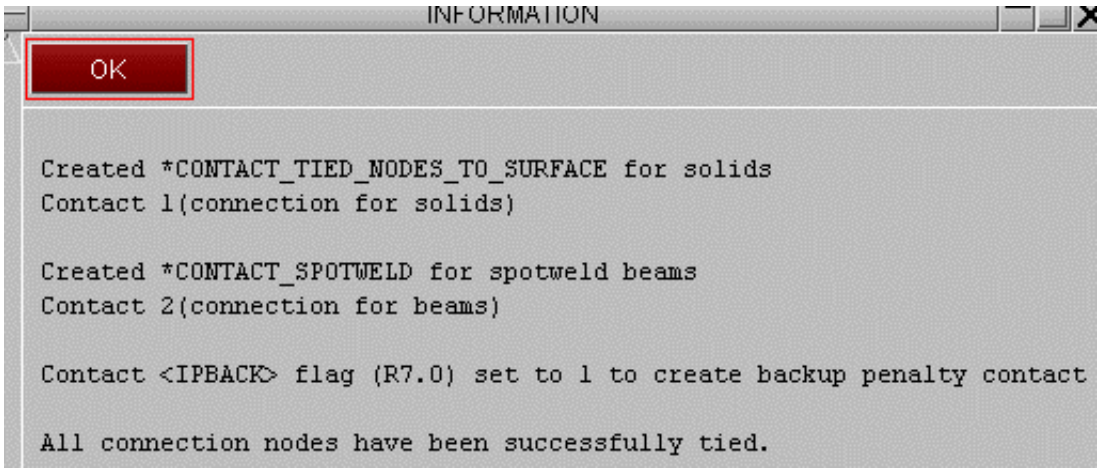


If we take the **TWO CONTACTS** option Primer will create both contacts with the slave side initially defined by PART or PART-SET. Primer will then automatically check for any constraint clashes.

Offset backup contact. If clashes are detected, Primer will create a copy of the original contact which is identical but for being of type _OFFSET (penalty). This preserves the preferred definition by PART on the contact. For all nodes which tie by the constrained contact contact, the penalty one is an irrelevance, but those which release due to constraint conflict will be captured by the offset contact.



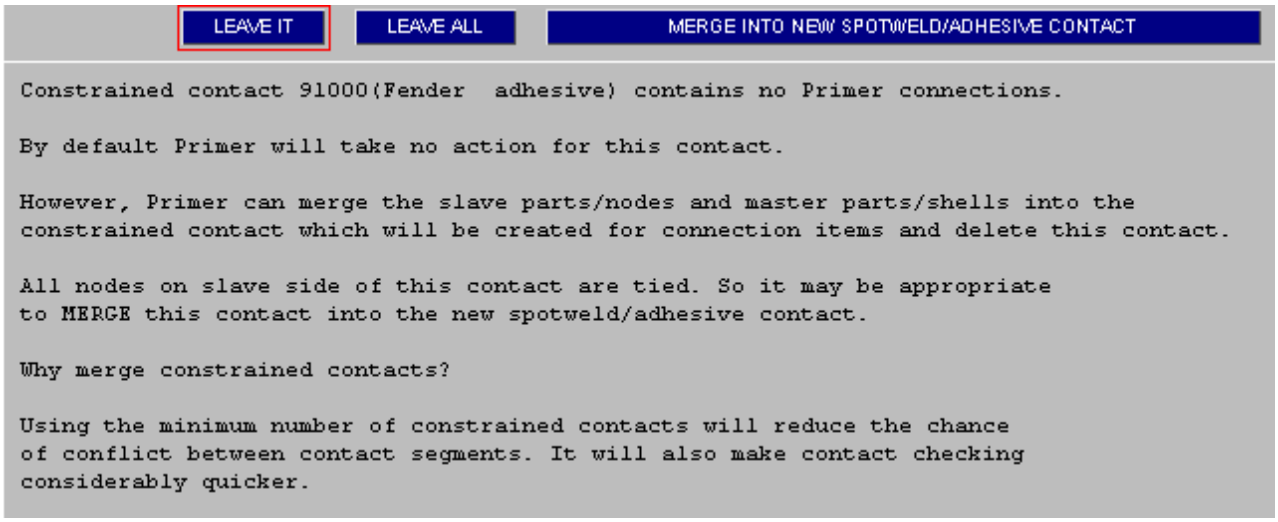
If the output version is R7.0 (or higher) there is no need to create an explicit second contact as setting the IPBACK flag on the constrained contact will achieve exactly this.



Keeping contacts defined by PART/PART-SET. This is the method much preferred by users and Primer will attempt to maintain definition of master and slave slides in this way. However, to tie internal nodes of solid elements (such as occur with 3T nugget welds or adhesive runs) LS-Dyna requires definition of the slave side by node set. The internal nodes do not tie if definition is by part set! Therefore Primer will use NODE_SET_GENERAL referencing parts in this case to overcome the problem.

Handling mixed contacts on re-make: Contacts may already exist which contain on the slave side both connection and non-connection items (e.g. solid spotwelds and solid foam). In this case, Primer will create connection contact as before, but in addition will add the non-connection nodes to the slave side, and their corresponding parts to the master side. Thus connectivity of the non-connection part should not be lost.

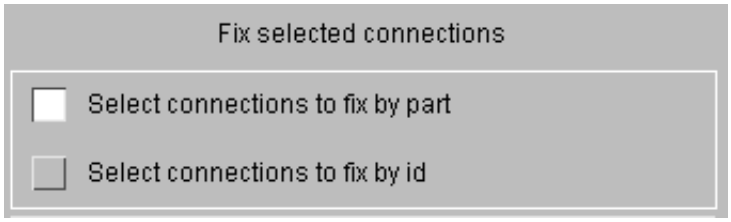
Handling constrained contacts with no connections: To avoid clashes it it advantageous to have the minimum number of constrained contacts in a model. If Primer finds, when creating or re-making connection contacts, that constrained contacts already exist in the model which have no connections on the slave side, you will be given the option to merge these into the connection contact to be created. This action is recommended if you intend all the slave nodes to tie. If you choose to **LEAVE** you may find that Primer has to convert more of the connection contact to _OFFSET.

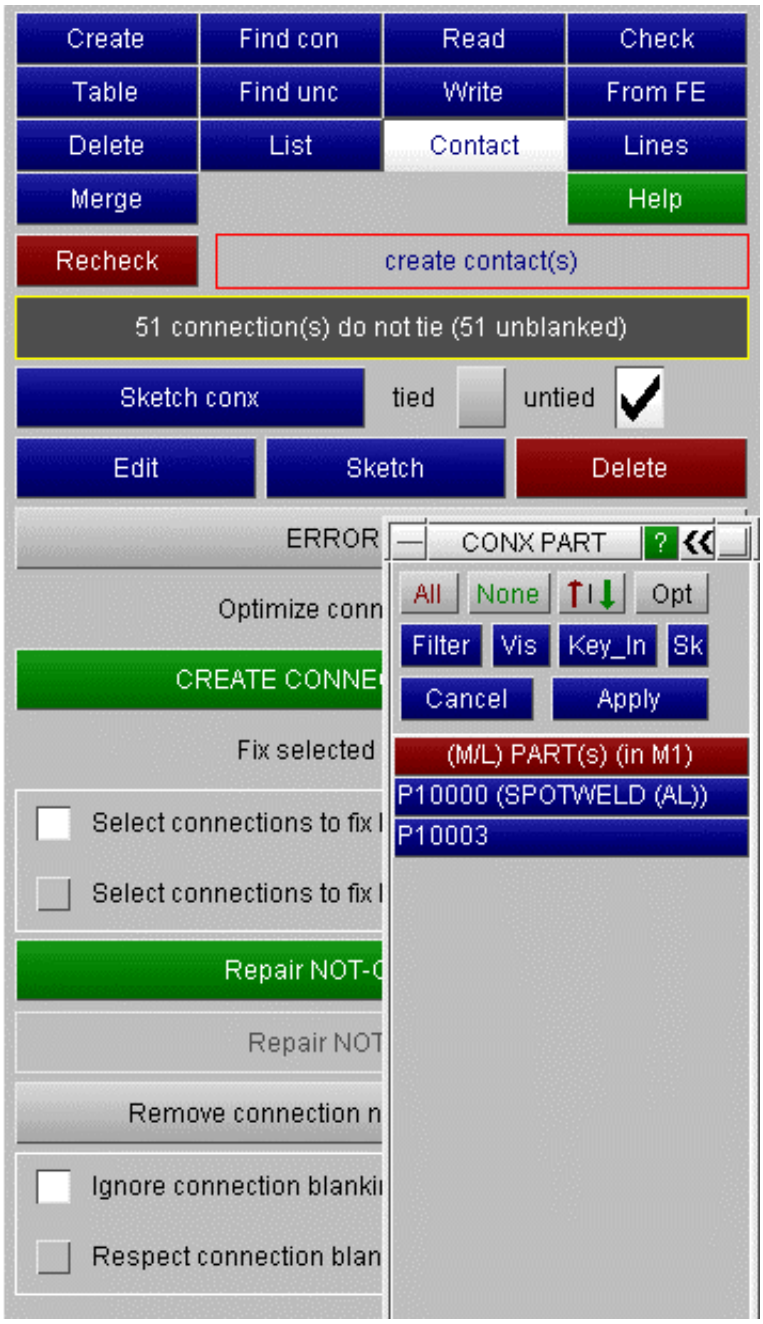


6.12.9.4 Connection > Contact: semi-automatic fix

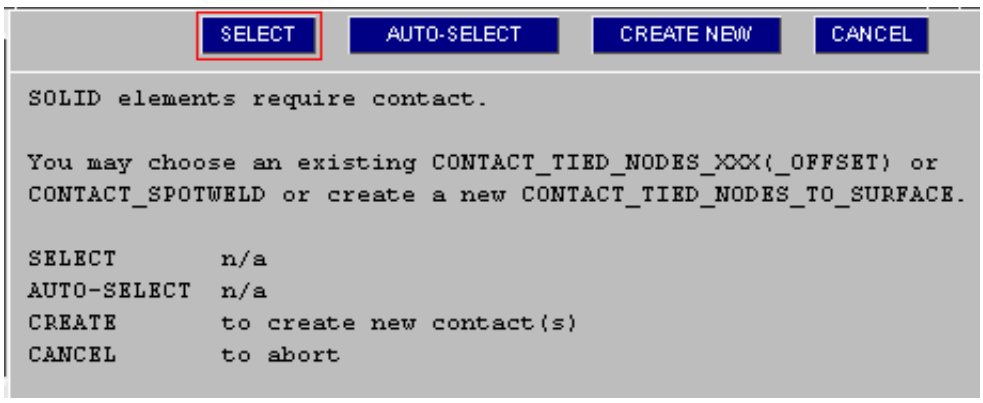
Repair NOT-CONNECTED. This function is designed to handle cases where the user does not want to re-make *all connection contacts* but prefers to fettle the individual contacts.

Connections to be fixed may be selected by part, which gives a better chance of preserving definition by part on the slave side of the contact or by connections which will incur conversion of slave side to SET_NODE_GENERAL and addition of nodes to the contact.



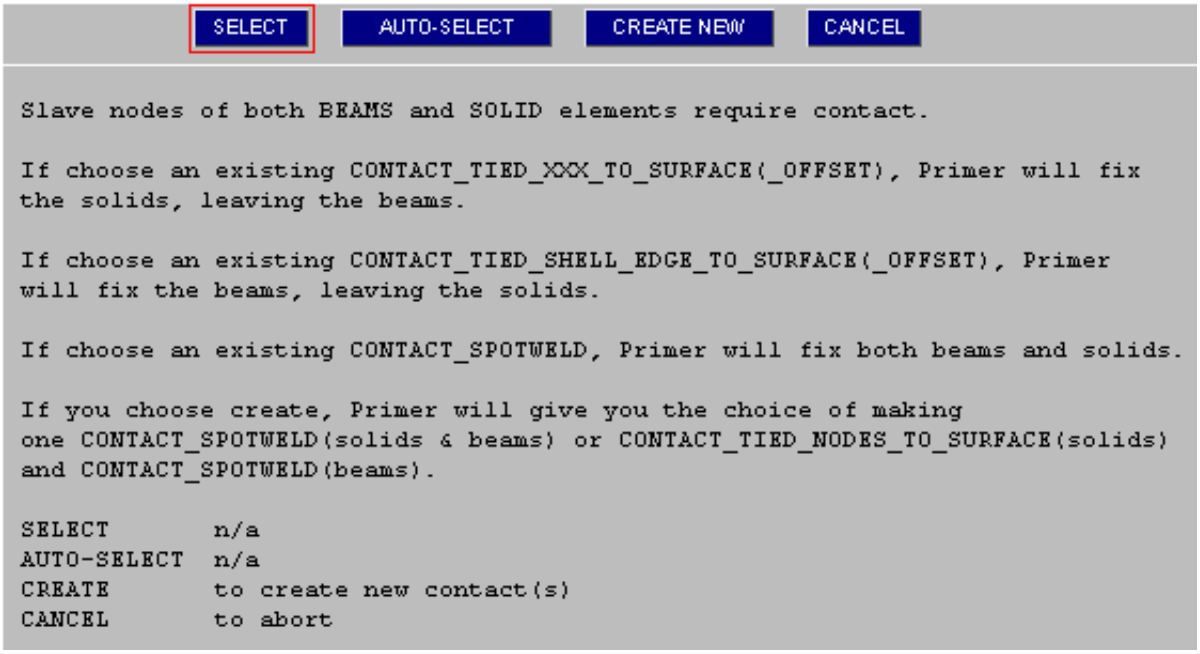


CREATE NEW will create a new constrained connection contact (by PART/PART-SET) of the appropriate type. You may also **SELECT** an existing contact and add the connections to it. Primer will refuse so to do, if the contact type is incompatible.

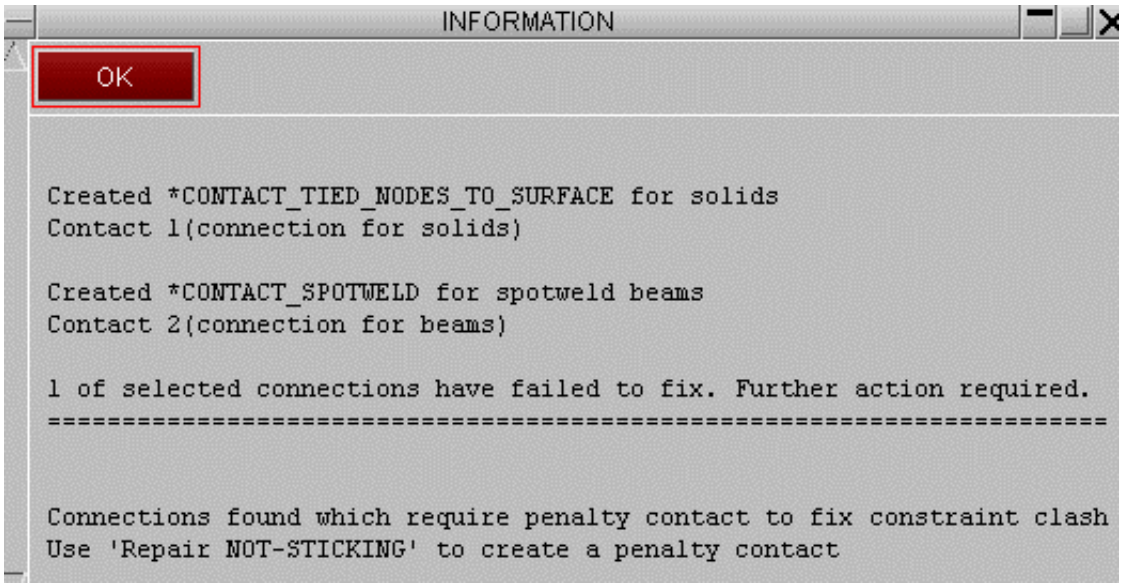


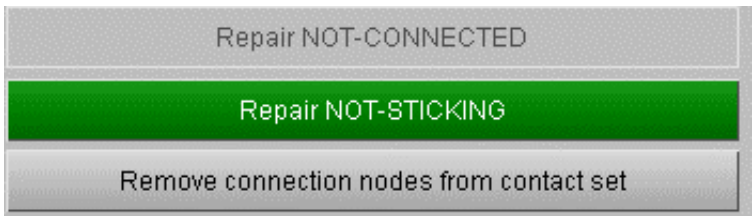
If both beam and solid parts have been selected for contact Primer's action will depend on the contact you select. The

information on the panel will guide you.



On creation of the contact, Primer will check for constraint clashes and invite you to fix them using **Repair NOT-STICKING** function which will have become active.





Connections may again be selected by part or by connection. Primer will then repair the non-sticking contact(s) by creating a duplicate backup penalty contact (or setting IPBACK on the constrained contact).

On completion of the process, on the main panel you should get the report that **all valid connections tie** and the semi-automatic connection fixing options will be greyed.

As a further check on the table all weld/adhesive connections should have REALIZED status and there should be no error messages.

Create
Find con
Read
Check
Table
Find unc
Write
From FE
Delete
List
Contact
Lines
Merge
Help
Recheck
2 connection contacts in model
all valid connections tie

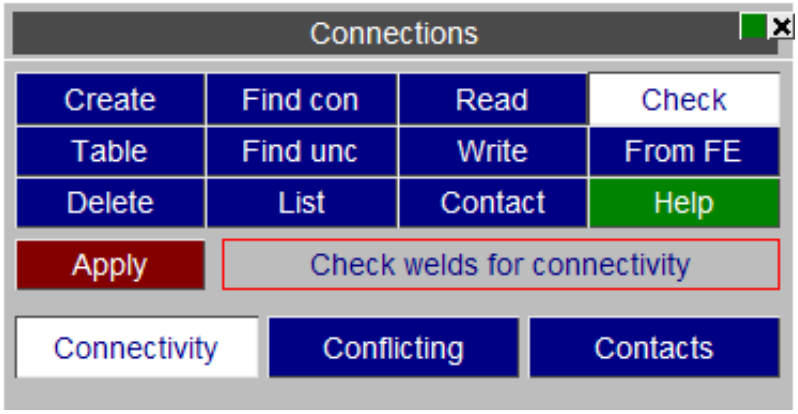
6.12.10 Checking connections

As well as checks under the normal model checking, there are three PRIMER functions that allow you to check properties of spotwelds.

Connectivity checks the quality of spotweld connectivity.
Conflicting, where PRIMER checks for the distances between welds and offers to delete welds that are too close together.
Contacts where PRIMER looks to see if the spotwelds are part of a tied contact in the model.

Checking connectivity

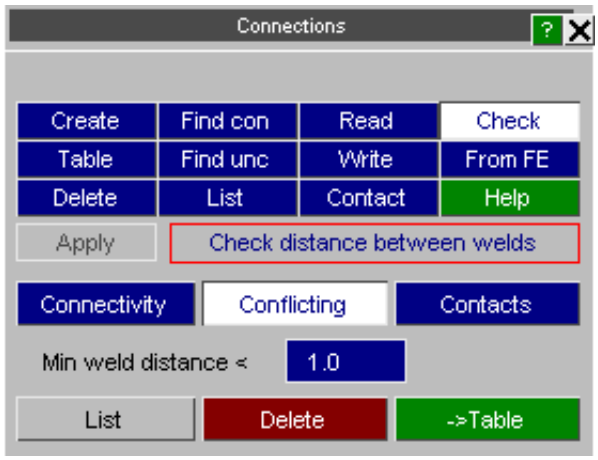
PRIMER will check all of the spotweld connections in the model to see if they are tied correctly. If any of the weldsare not PRIMER will put them onto the connection table and you can use it to visualise and fix the welds. For more details see section 6.12.2.



Checking conflicting welds

PRIMER will check to see if any spotweld connections in the model are too close to each other. Enter the **Min weld distance** and then you can either

- **List** - list all conflicting the welds
- **Delete** - delete a subset of welds to remove the conflict (same as model autofix)
- **->Table** - send all conflicting welds to table where you can delete them or merge them

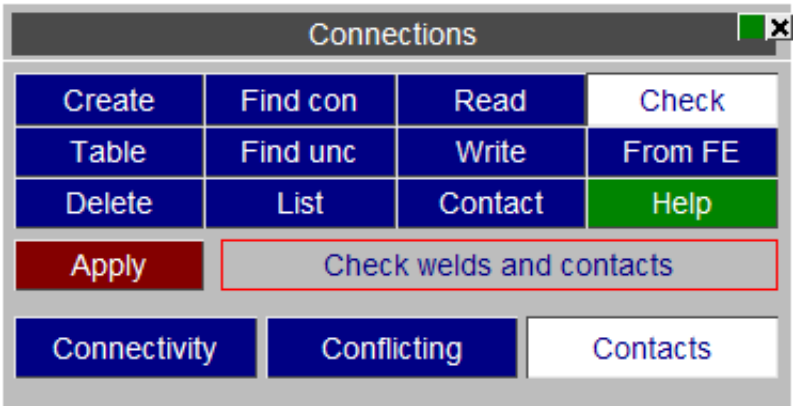


Checking weld contacts

PRIMER will check to see which spotweld connections are not in a tied contact.

Any beams, solids or parts that are not in a tied contact will be put into sets so that you can visualise them.

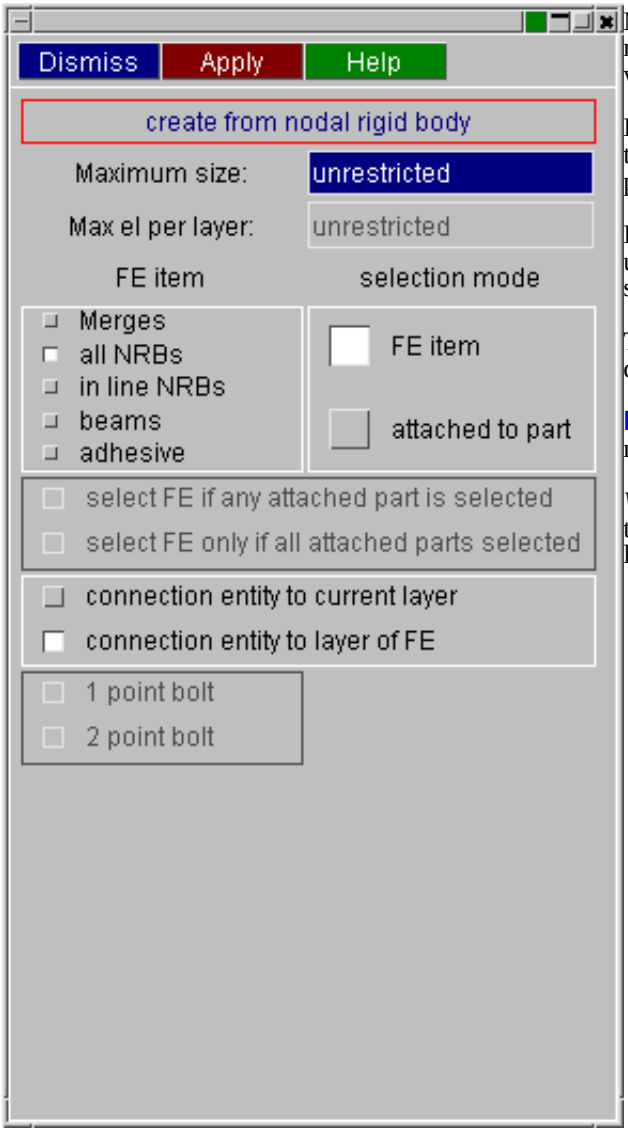
Note - this function will only work when you have a **single connection contact**. For a more generic treatment you should use the function under **CONNECTIONS > CONTACT**.



6.12.11 Creating bolt connections from FE data

PRIMER will automatically create connections for any existing beam spotwelds and solid spotwelds in your model. At present PRIMER will not make connections for ’nugget’ solid spotwelds (i.e. spotwelds that use multiple solids and a *DEFINE_HEX_SPOTWELD_ASSEMBLY card. Primer can also make connection entities for existing solid adhesive runs in the model which conform to "Primer like" adhesive connection entities.

For ’bolt’ type connections PRIMER cannot make the connections automatically. The **FROM FE** panel enables you to create them from selected Nodal Rigid Body or Constrained Rigid Body definitions.



Nodal rigid body or one (of a possible chain) of rigid body merges may be selected from the object menu in the usual way.

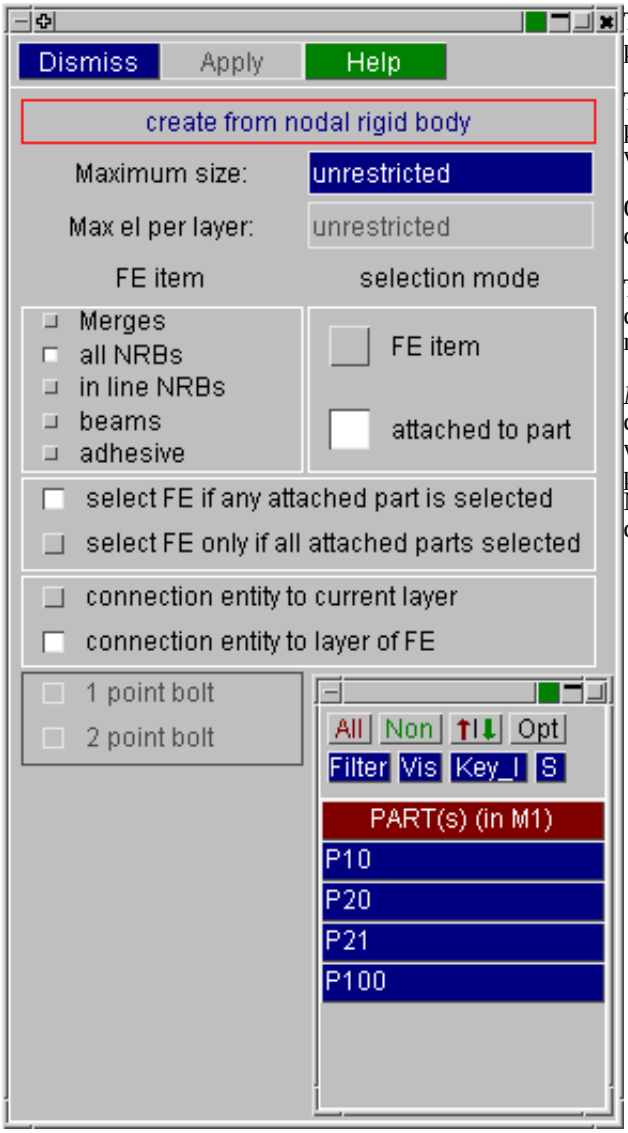
Beams may be selected to create connection beam bolts. In this mode, you may set Primer to create one point or two point connections.

For each item selected a calculation of dimension is made and user has the option to exclude those that exceed the maximum size (if specified).

The dimension is an approximate measure of the connection’s diameter.

line nrbs option will limit the object menu to offer only nodal rigid bodies which form a straight line

Where is the connection put? If the model contains include the default behaviour is to put the connection entity into the layer of the NRB or the C_RBOD from which it is formed.



The FE items may also be found by selecting by attached part.

The options are to select the item if it attaches to any or the parts selected or, more restrictively, to select only those items where all the attached parts are selected.

On **APPLY** connection with correct layer data will be created. The FE data itself *will not be changed.*

The user is recommended to invoke the newly made connections on the TABLE and check their diameter, as this may require adjustment.

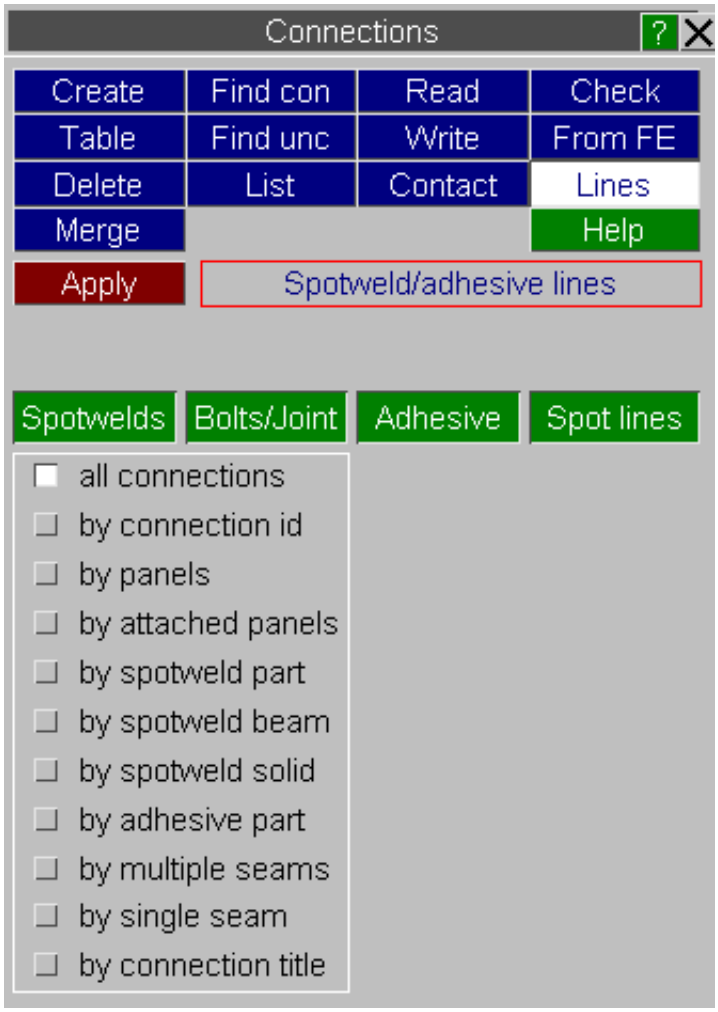
Note on remake of connections - the FE items may be changed slightly after remake. Merge type bolts will remake with a master part (itself containing no elements) and slave parts overlaying the panel, so an extra part has been created. NRB type bolts may remake at a slightly different size. In all cases, connectivity of the layer panels will be maintained.

For adhesive connections, you select the solid element runs you wish to create connection entities for. On **APPLY** Primer will create connection entities. The FE data itself *will not be changed.*

6.12.12 Modifying spotwelds and adhesives through lines

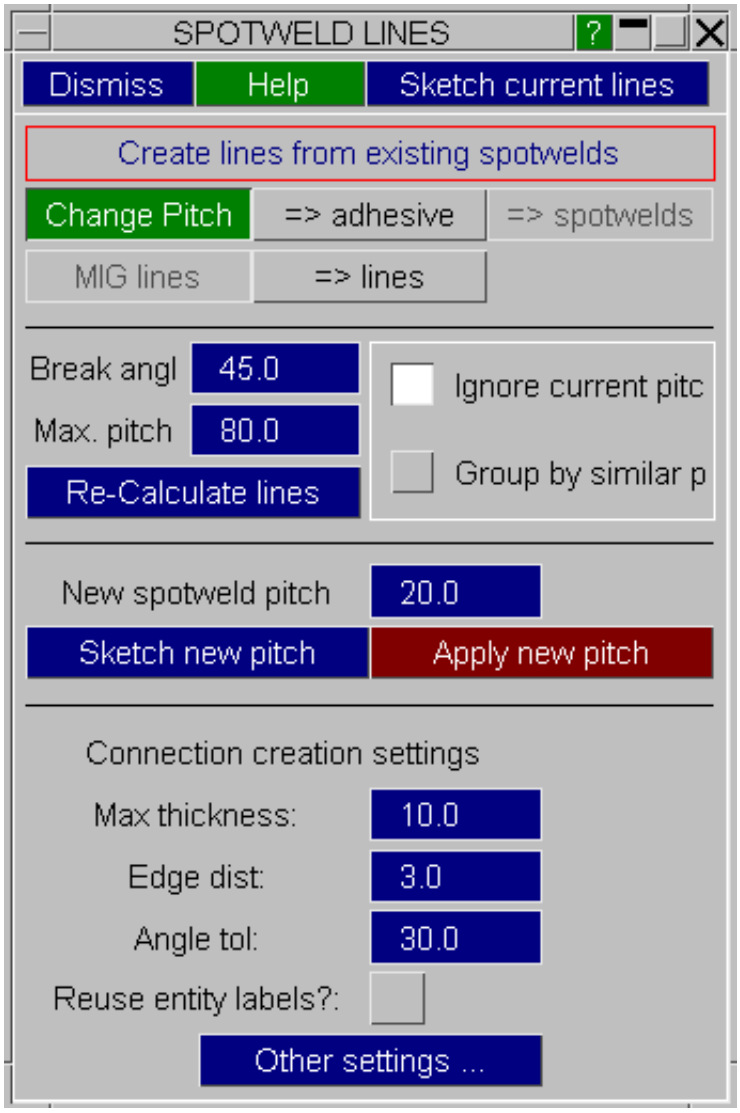
Spotweld connections exist as individual entities within PRIMER, but the **LINES** tool can be used to modify groups of individual spotwelds that form lines. A common use of this feature is to modify the pitch of a run of individual spotwelds. This tool can also be used to convert a line of spotwelds to a run of adhesive, or vice-versa

The connection lines feature can be accessed by selecting LINES in the connections panel. You can then use the standard methods for selection of connections. For more details of the different methods see section 6.12.0. Note that you can only modify spotwelds or adhesive, not spotwelds or adhesive together.

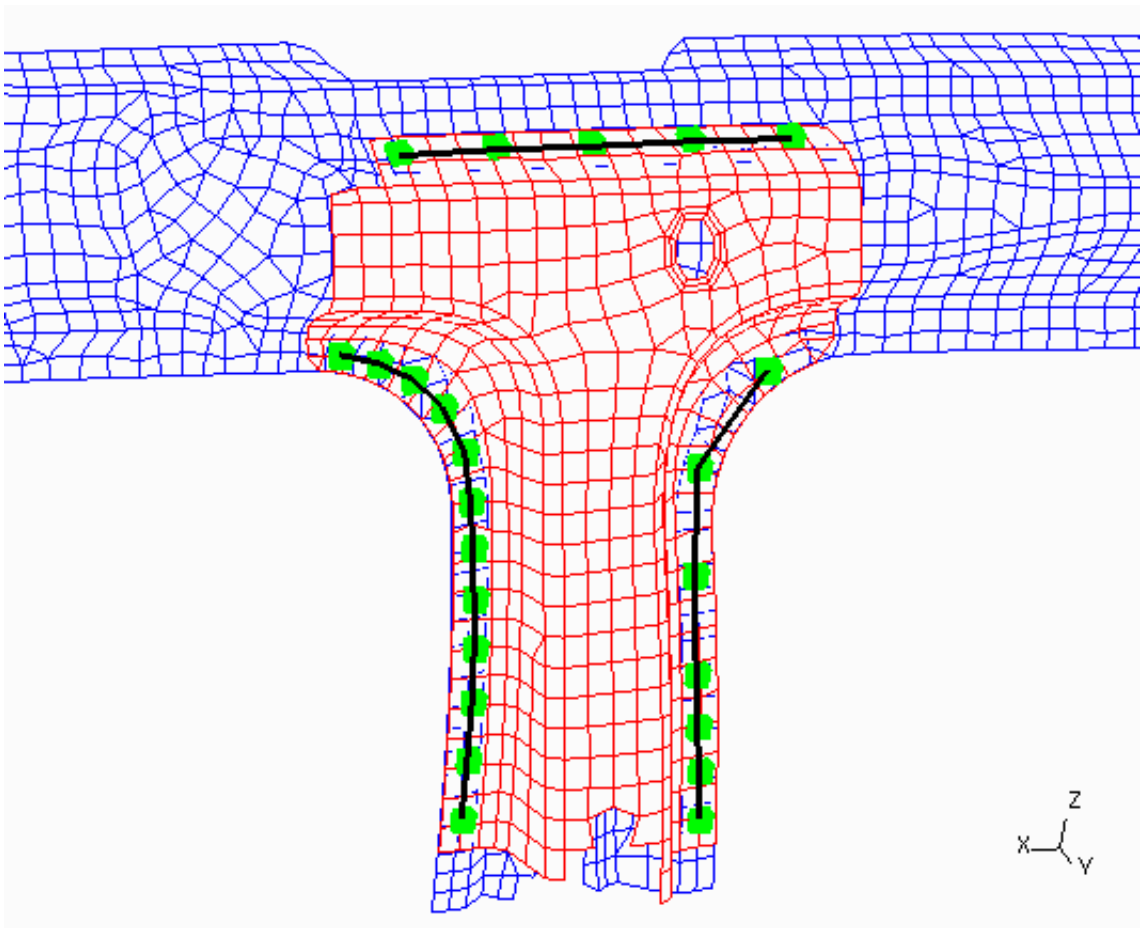


After selecting the connections you wish to modify, PRIMER will group the the spotweld lines panel will open up. The panel will look different depending whether you select spotwelds or adhesives.

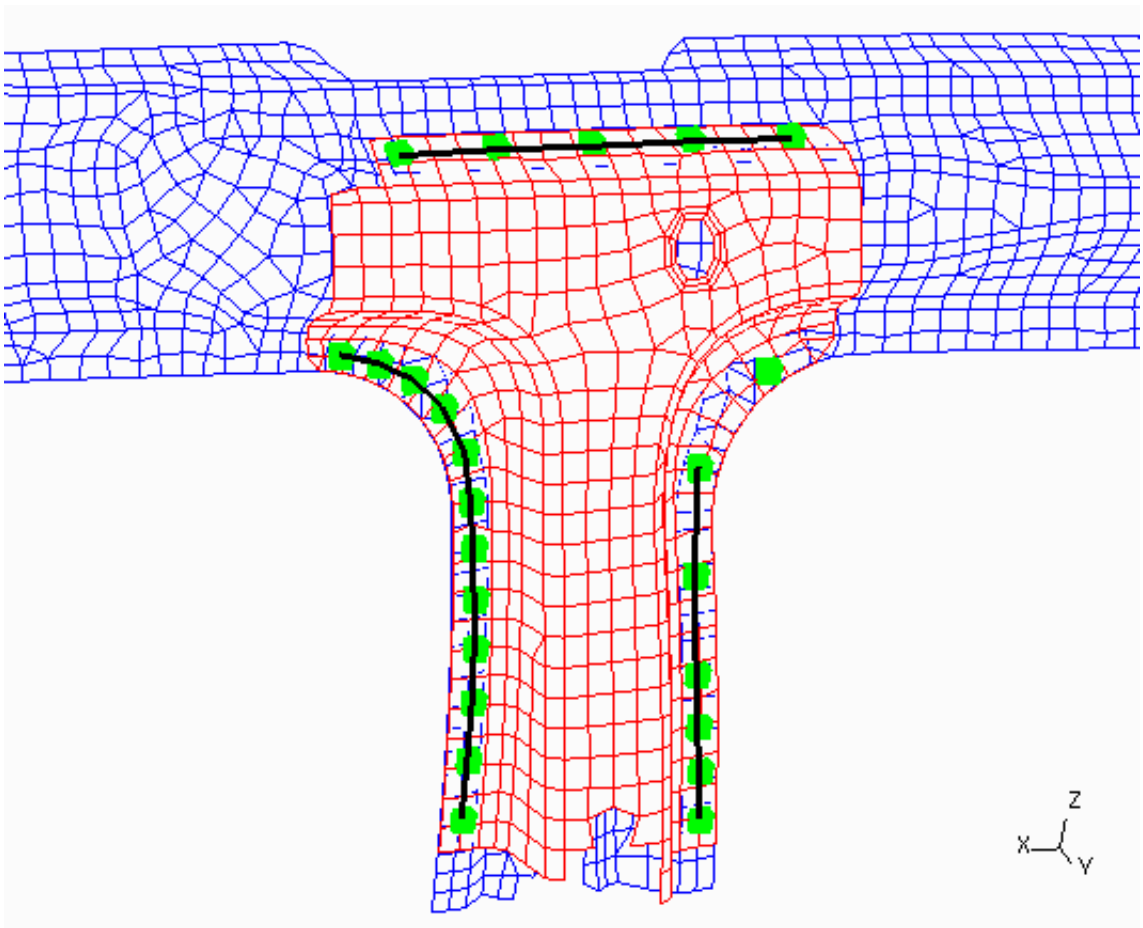
Modifying spotwelds



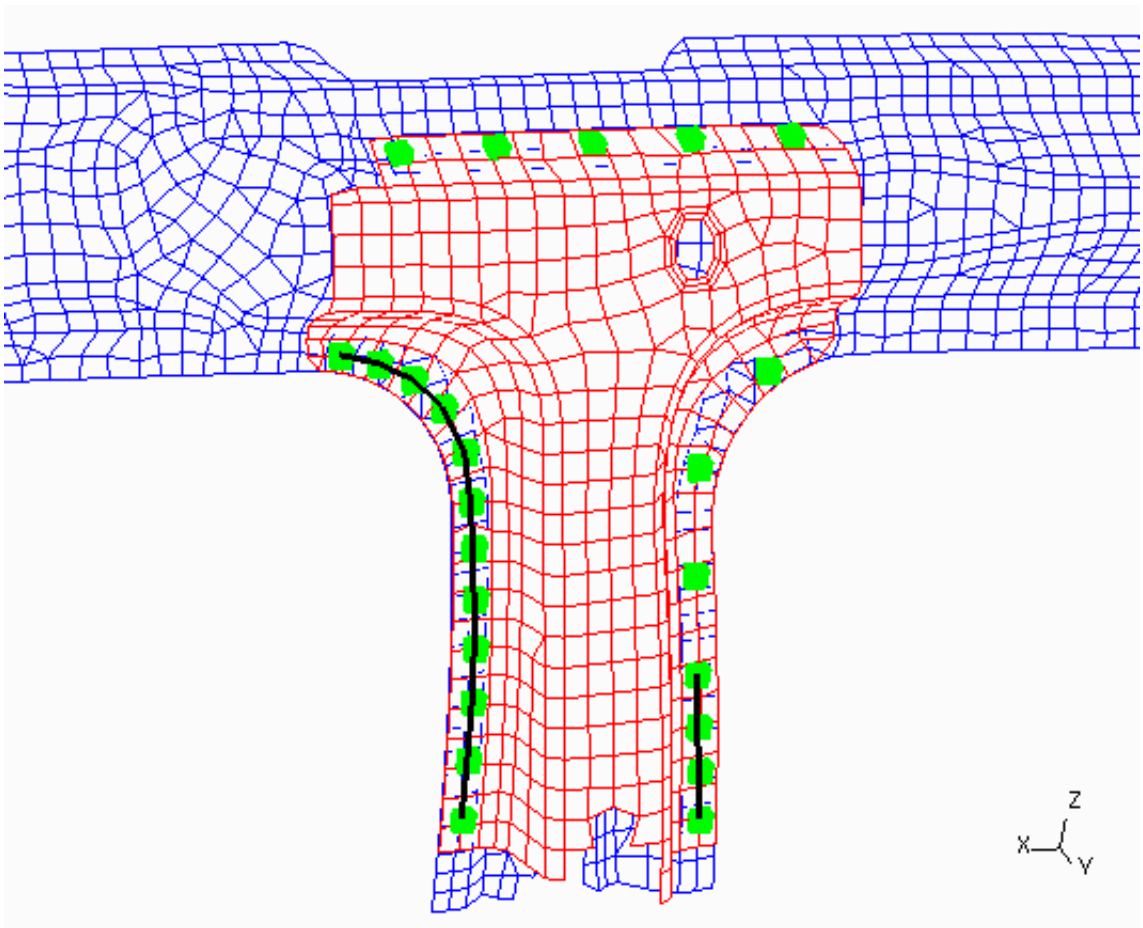
The above panel opens up when modifying spotwelds. When opening up the panel, PRIMER will group the spotweld connections in the model together into line groups, and sketch the lines on the screen.



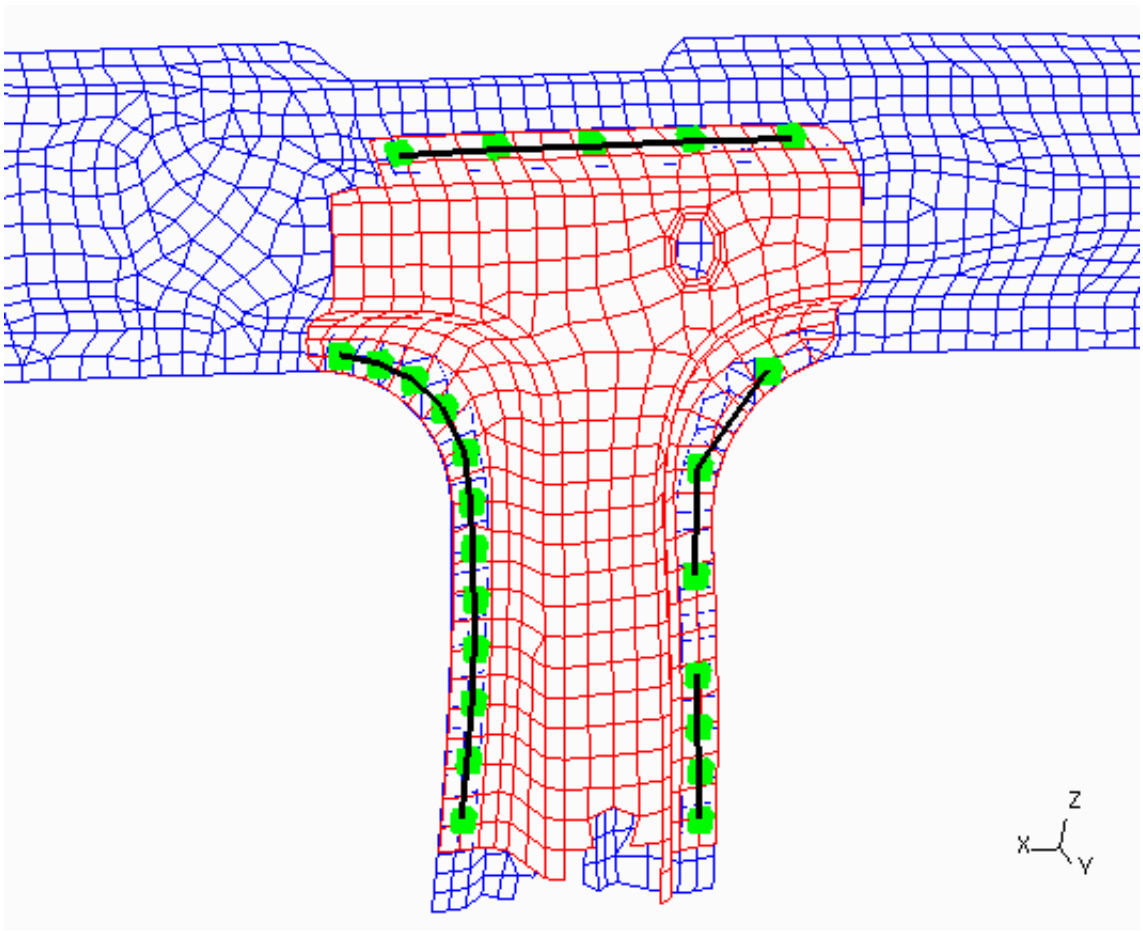
The default when opening the panel with spotwelds is **Change Pitch** mode, which can be used to modify the pitch of the selected spotwelds. At the top of the panel, there is a **Sketch current lines** button. This can be used to sketch the lines currently being used by the panel should you lose the original sketching (can occur if you redraw for example). The top part of the panel can be used to modify the inputs PRIMER uses to calculate the groupings of spotwelds to form lines. The **Break angle** value is used to separate the lines should the angle between two sections of the line by greater than this value. The following image shows the affect of changing the break angle from the default of 40.0 to 20.0.



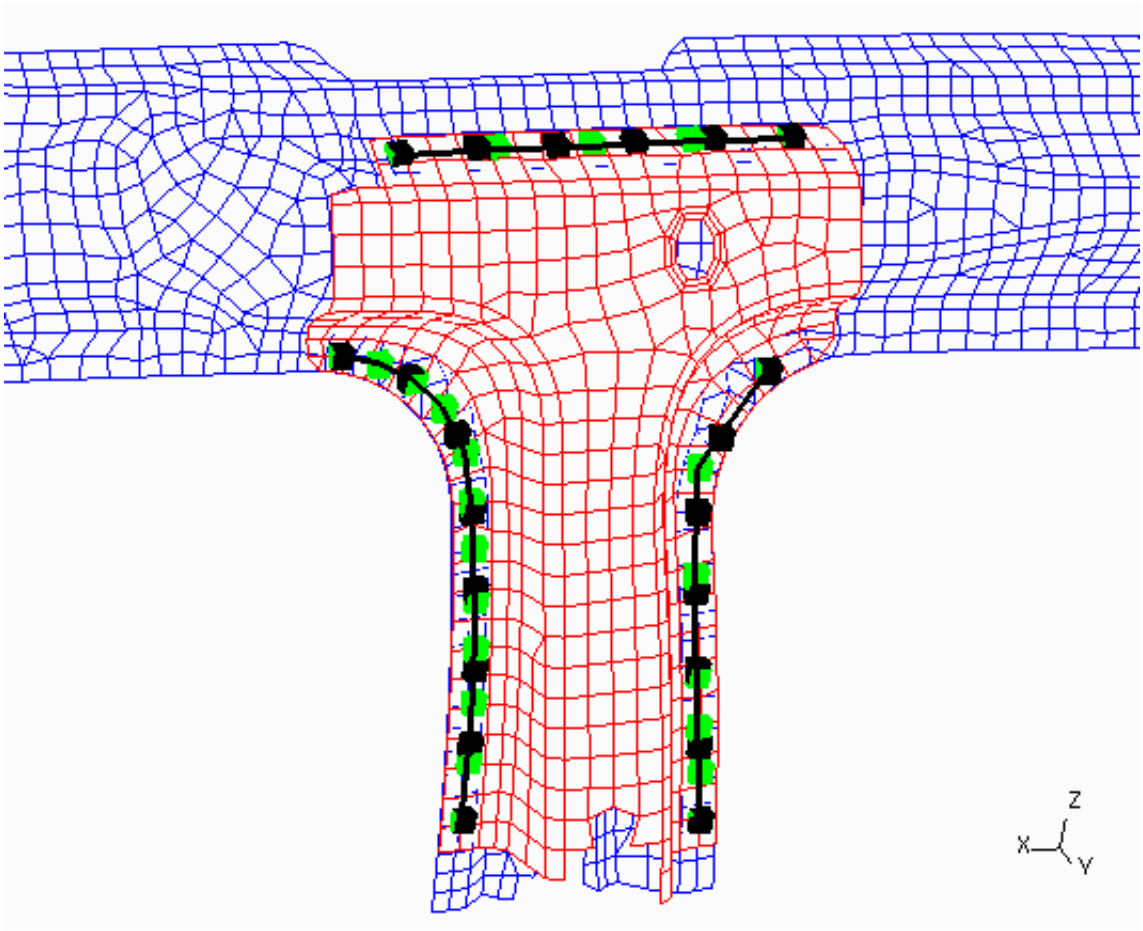
The **Max. pitch** value is the maximum allowed distance that two spotwelds can be apart and still considered to be part of the same line. The following image shows the affect of changing the maximum pitch from the default of 80.0 to 40.0.

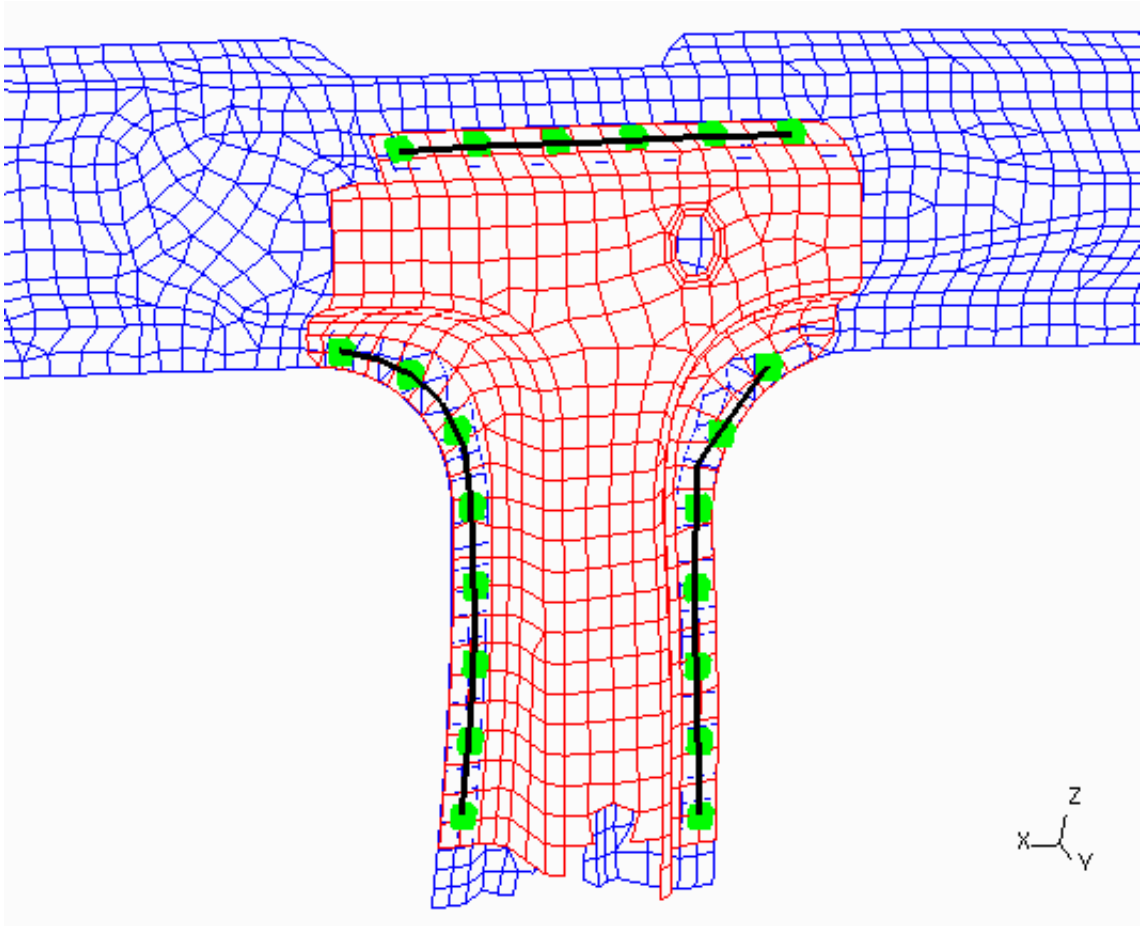


If **Group by similar pitch** is toggled, PRIMER will group together spotwelds that have a similar pitch (as, for example, spotwelds may be irregular along a flange and you may want these to be considered separate lines). The default of **Ignore current pitch** will not group spotwelds according to this rule. The following image shows the affect on the example of toggling **Group by similar pitch**. Note the line on the lower right flange is now split into two, and the spotweld pitch is inconsistent along the flange.



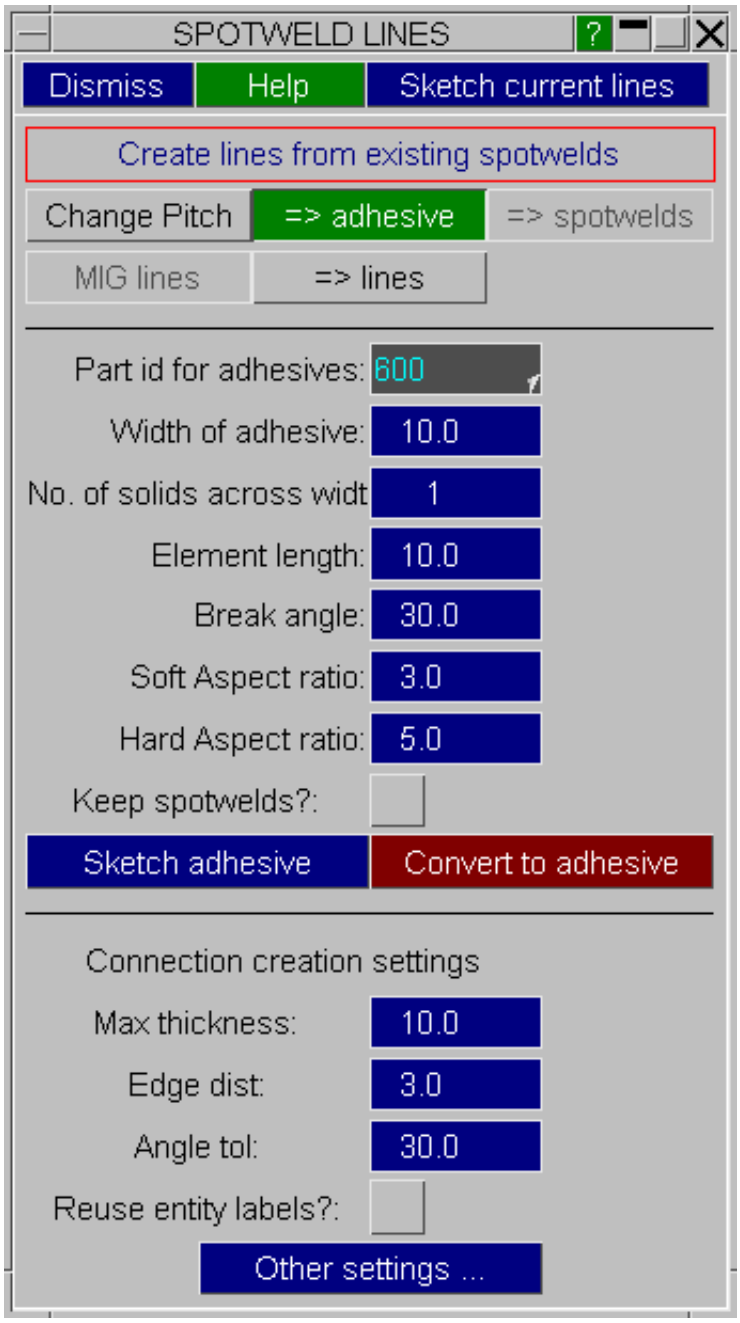
Clicking on **Re-calculate lines** will re-calculate the line groupings based on the current settings. The new pitch you want to apply to the spotweld line groupings in typed into **New spotweld pitch**. The proposed new spotweld positions can be sketched by clicking on **Sketch new pitch**, and the new spotweld pitch can be applied using **Apply new pitch**. The following images show the sketch case and the apply case for the above example when implementing a new pitch of 40.0mm.


Z
X
Y

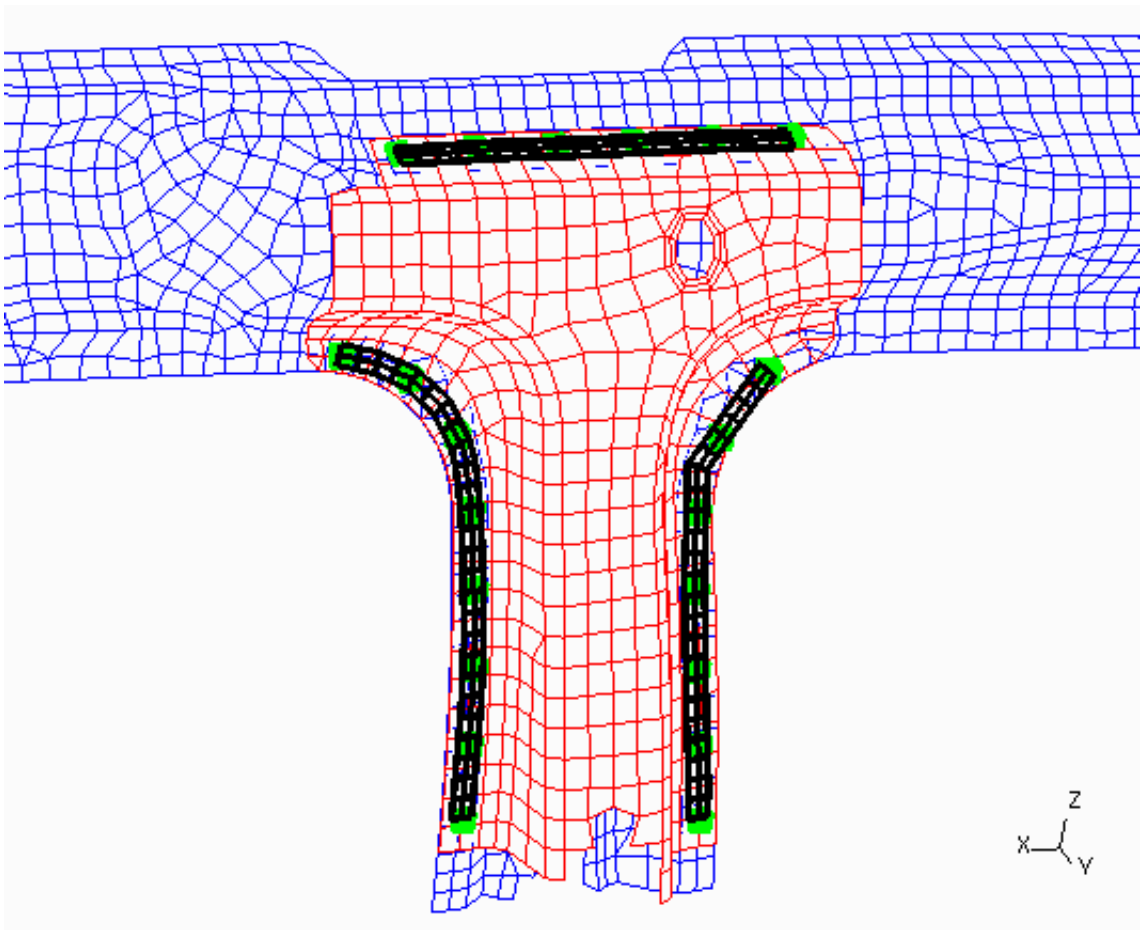

Z
X
Y

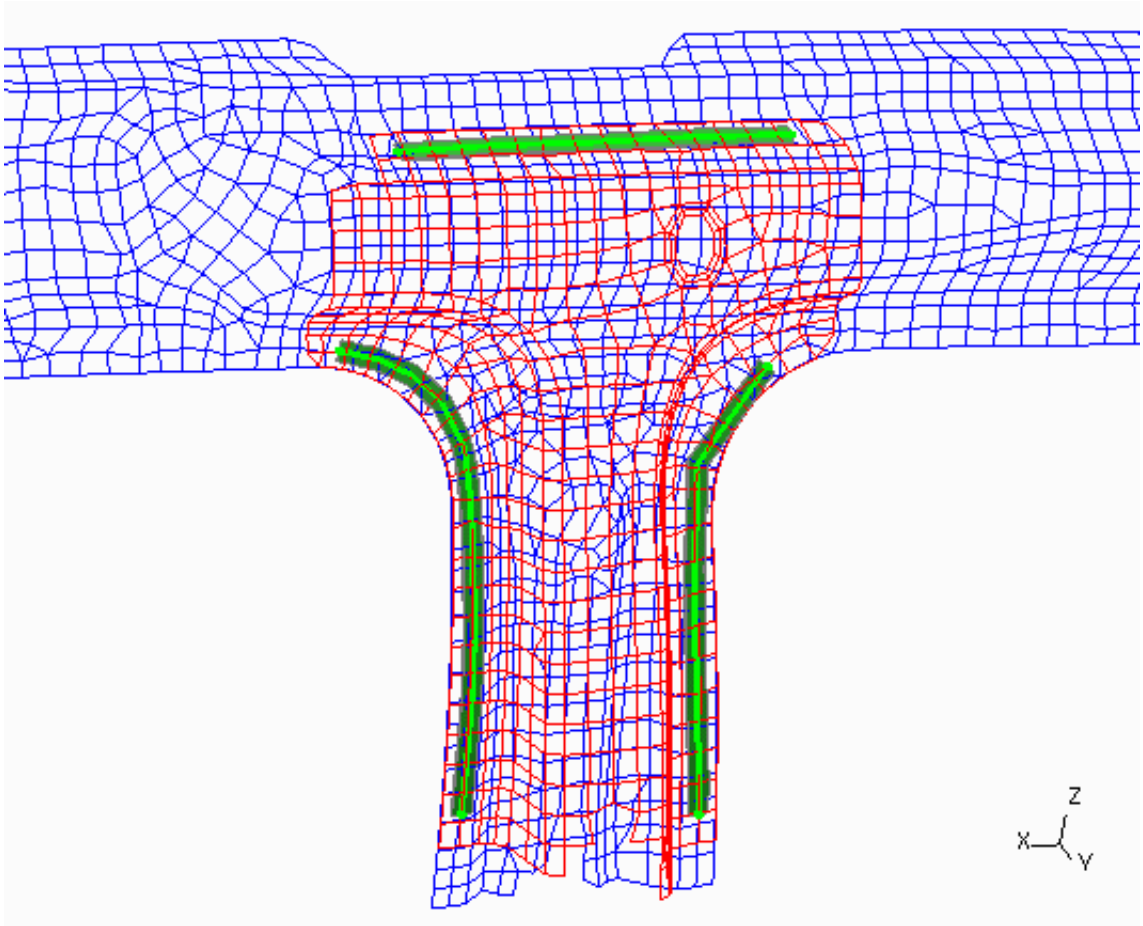
Note that before applying spotwelds, PRIMER will check that the information held for all the spotwelds in the line is consistent. For example, the diameter could vary between spotwelds, or the include file the spotwelds are in. If PRIMER finds inconsistent data, a message will appear asking the user if they wish to proceed. If the do proceed, PRIMER will use information from the first spotweld in the line to create all new spotwelds along the line length.

As mentioned previously, this panel can also be used to convert lines of spotwelds into adhesive runs. Click on **=> adhesive** to see the adhesive creation options.



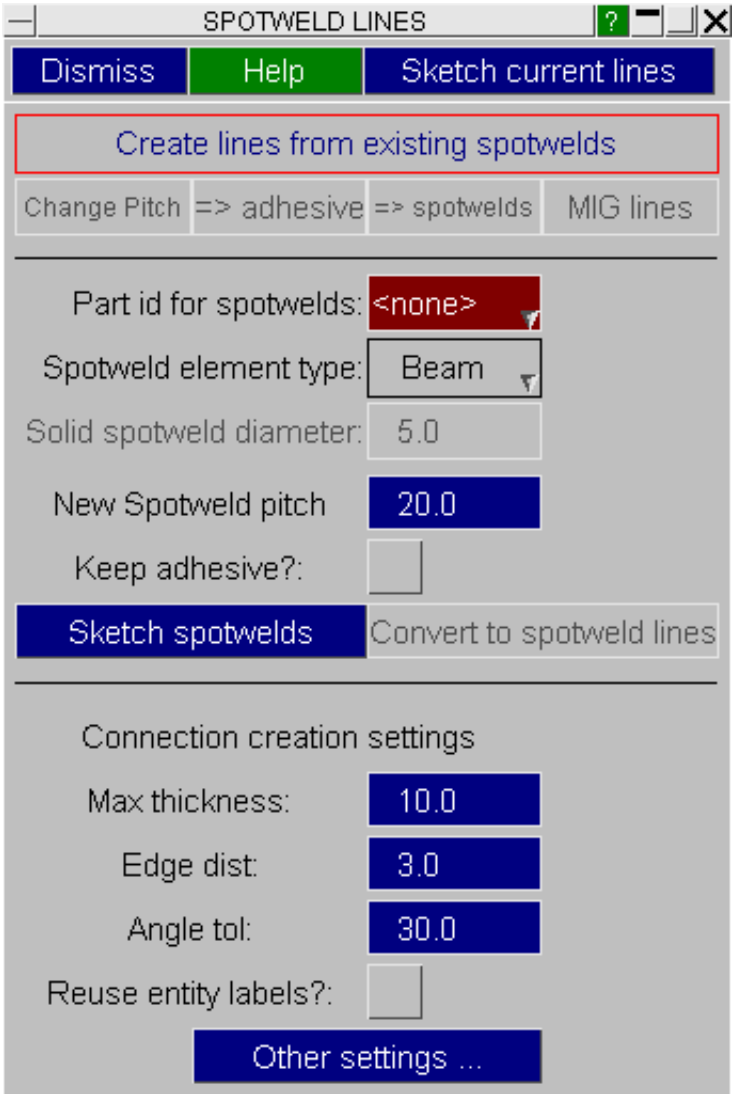
See the create adhesive section for details on the inputs for adhesive creation. The part ID of the solids that the adhesive solids will be created in is the only required parameter. Check the **Keep spotwelds?** tick box should you wish to create adhesive runs but retain the original spotwelds. Again, the user can sketch the proposed adhesive before creation (**Sketch adhesive**). Clicking on **Convert to adhesive** will apply the creation of the adhesive.


Z
X
Y


Z
X
Y

Modifying adhesive

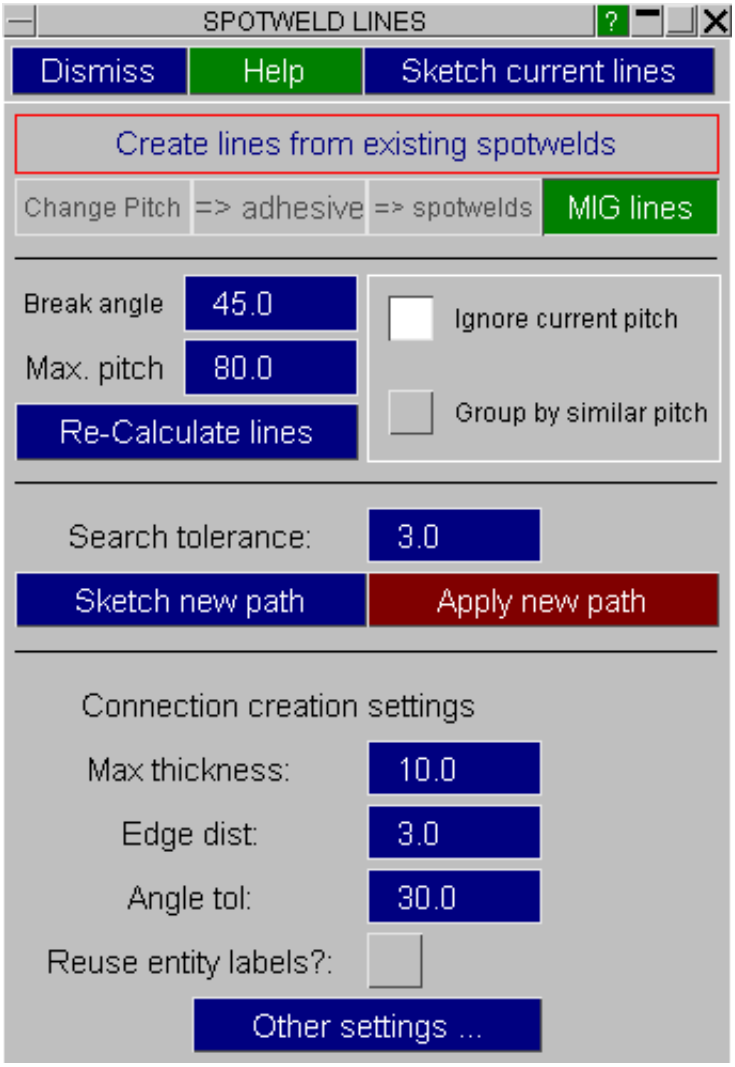
Converting adhesive runs to spotweld lines is also possible. This option is available to you when selecting adhesives when opening up this panel.



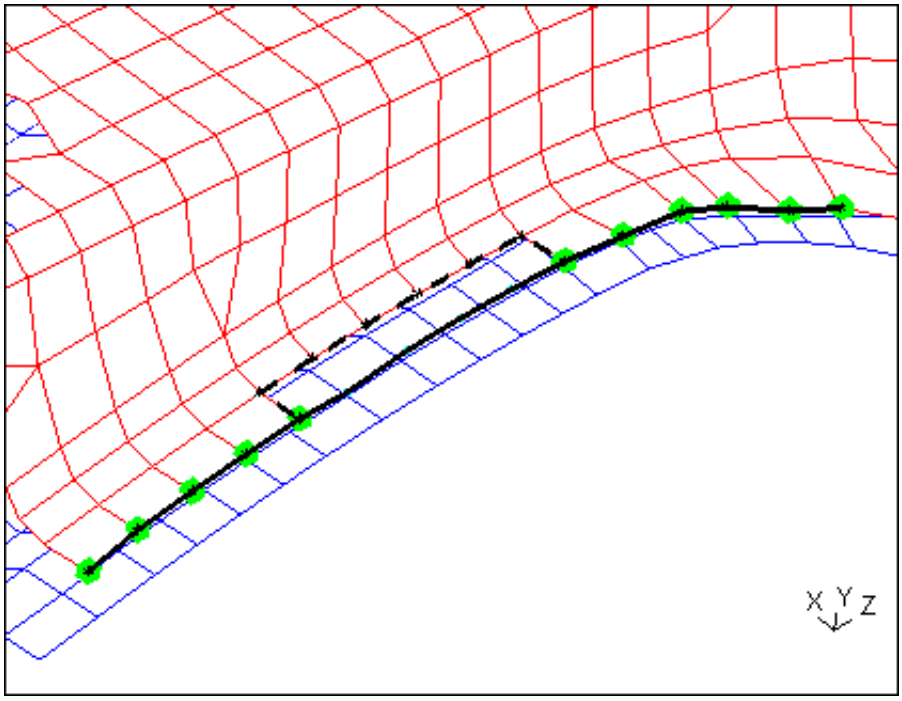
See the create spotweld section for information on the inputs. The part ID of the spotweld solids or beams will be created in is the only required parameter. Check the **Keep adhesive?** tick box should you wish to create lines of spotwelds but retain the original adhesive. Again, the user can sketch the proposed spotwelds before creation (**Sketch spotwelds**). Clicking on **Convert to spotweld lines** will apply the creation of the spotwelds.

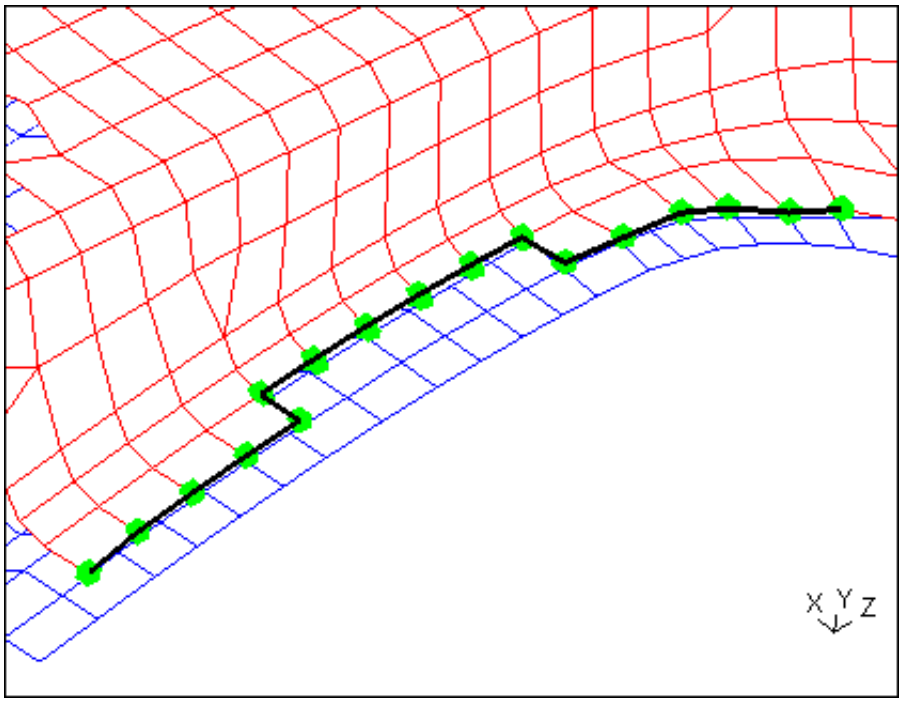
Modifying MIG spotweld types

When openning the spotweld lines panel with only MIG type spotwelds selected, the following panel is presented.



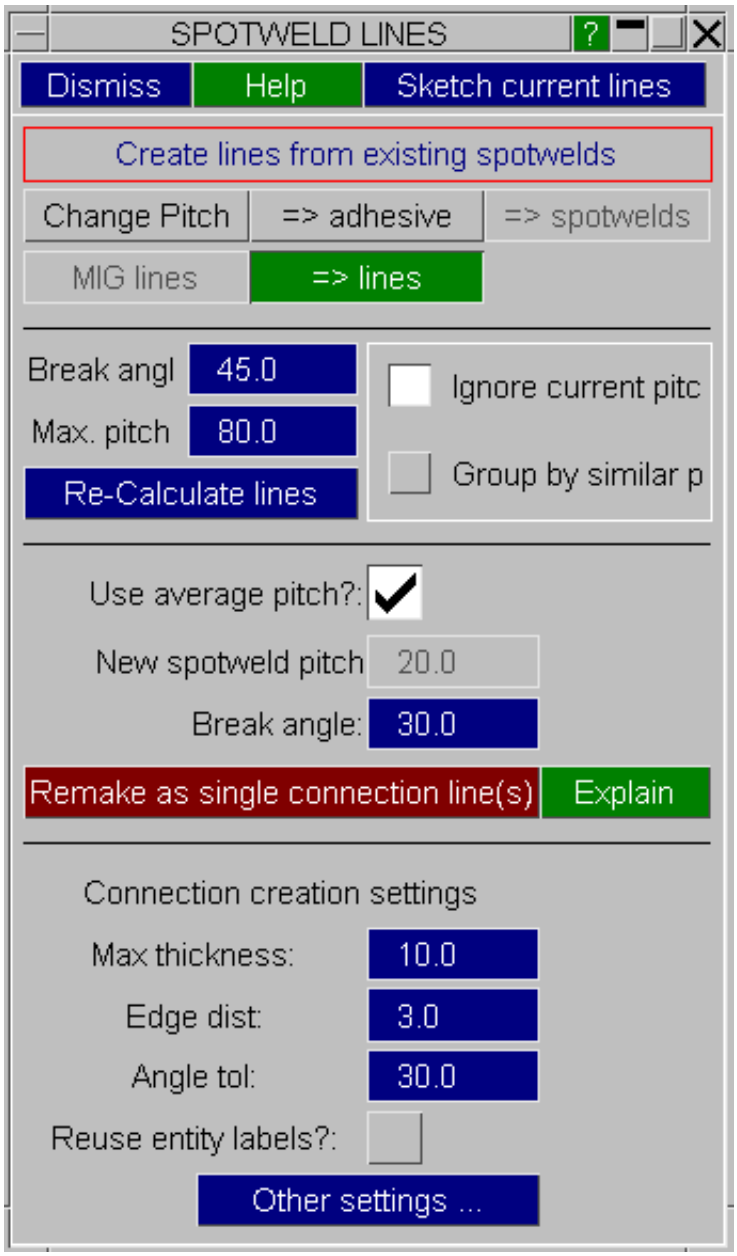
This panel is used to reapply MIG type spotwelds should the mesh of the free edge the MIG welds are attached to change. The panel will again automatically group the MIG welds into lines. The search tolerance is used by the line end point to look for a new start node on the updated free edge mesh. Clicking on **Sketch new path** will display the new path the MIG welds will be applied to. Clicking **Apply new path** will reapply the MIG spotwelds to the new mesh.





Converting individual spotwelds to spotweld line connection types

From v11 onwards, PRIMER has a connection type called "spotweld lines". These allow you to contain many individual spotwelds in one connection entity. The benifit of this is you can easily change the pitch of these spotwelds on the connections table, and reproject the line along a free edge when the panel mesh has changed. Groups of individual spotwelds can be converted to spotweld lines via the **=> lines** option.



Connection creation settings

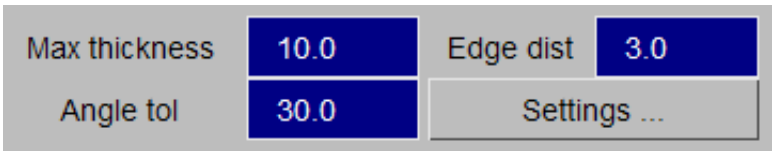
At the bottom of the spotweld lines panel are connection creation settings. These settings are used when creating new spotweld and adhesive entities. During creation, some spotwelds may not be made correctly due to the settings. A common example of this is if the user sets the new pitch to be smaller than the value set for minimum distance between welds (default 10.0mm). Should PRIMER note be able to make new connections for whatever reason, the user is given the opportunity to open these on the connections table for investigation.

For more information on the connection settings see connection settings.

Check the Reuse entity labels? if you wish PRIMER to reuse the entity labels of the previous spotwelds/adhesive when creating new connections.

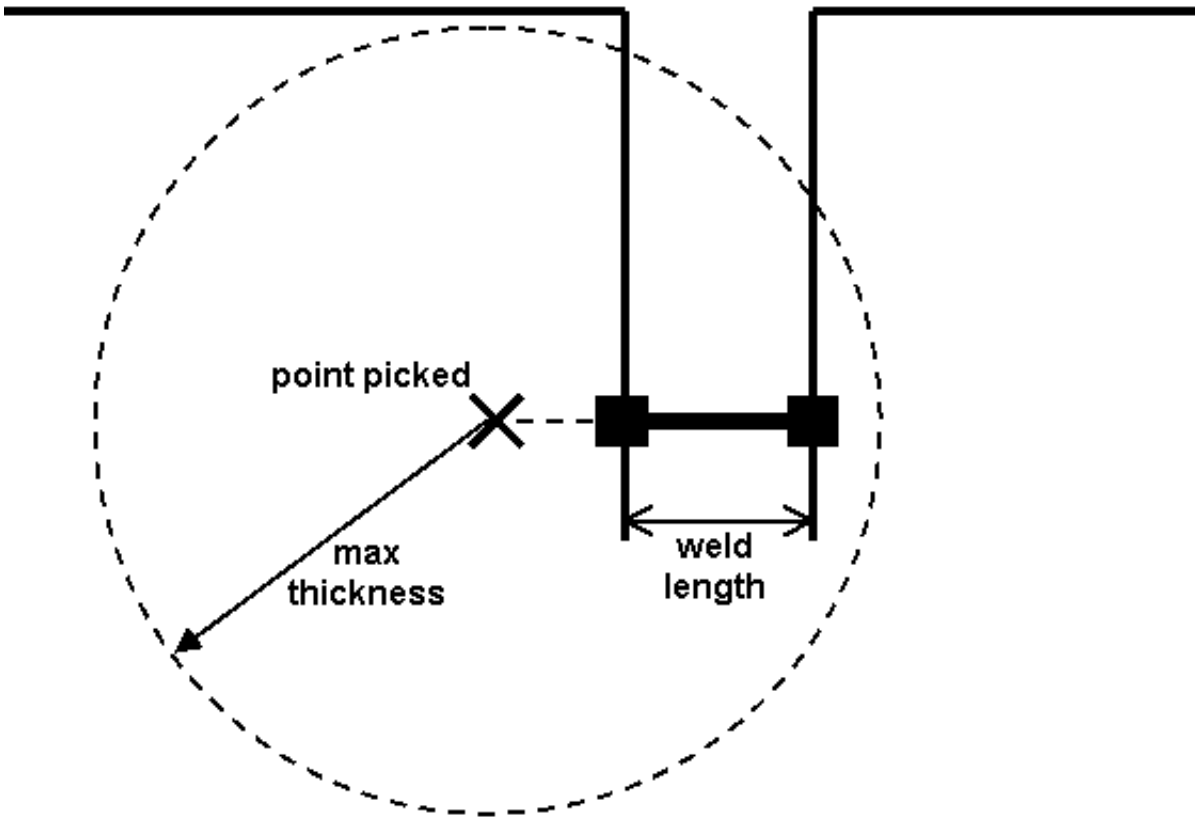
6.12.13 Connection options

There are several options that alter the way that connections are created. Some are found in the connection creating panel



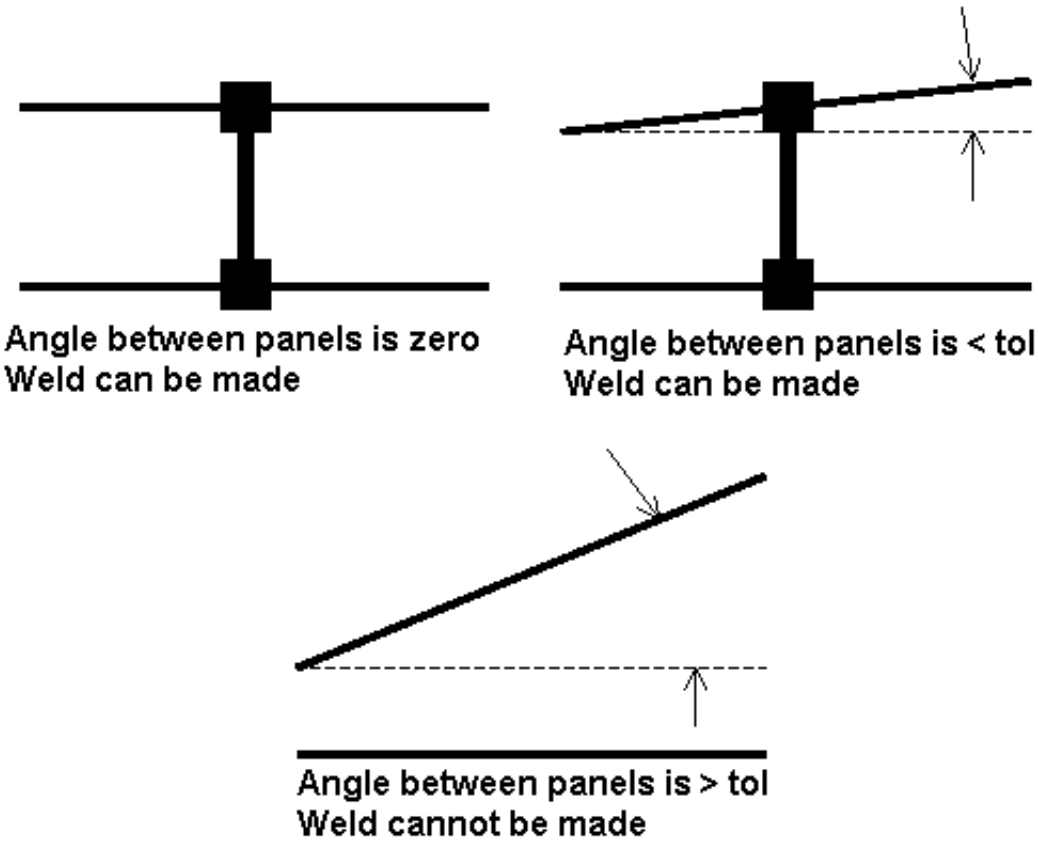
Max thickness

The **Max thickness** dimension is the radius from the point you pick that PRIMER will search for panels to try and weld. In the figure below both panels are inside the circle and so can be welded. If one of the panels was outside the circle the weld would not be able to be made.



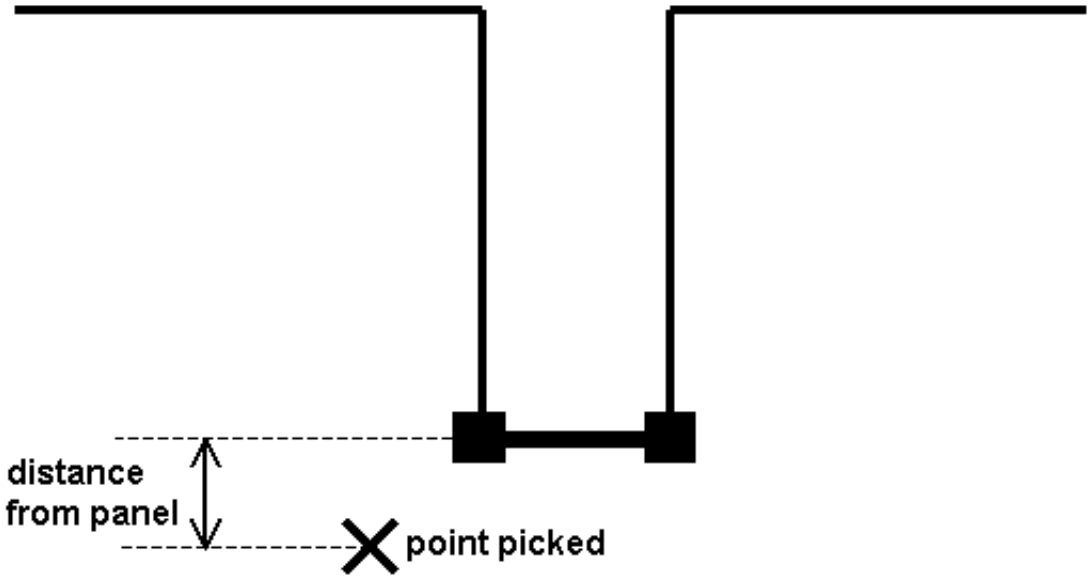
Angle tolerance

PRIMER uses the **angle tolerance** to check elements that it can weld. If the angle between the elements is less than the tolerance, the weld can be made. The figure below shows examples of welds that can and cannot be made.

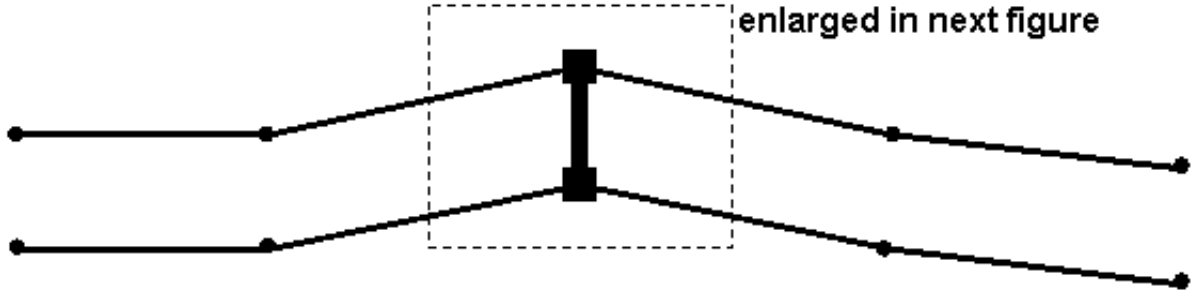


Edge tolerance

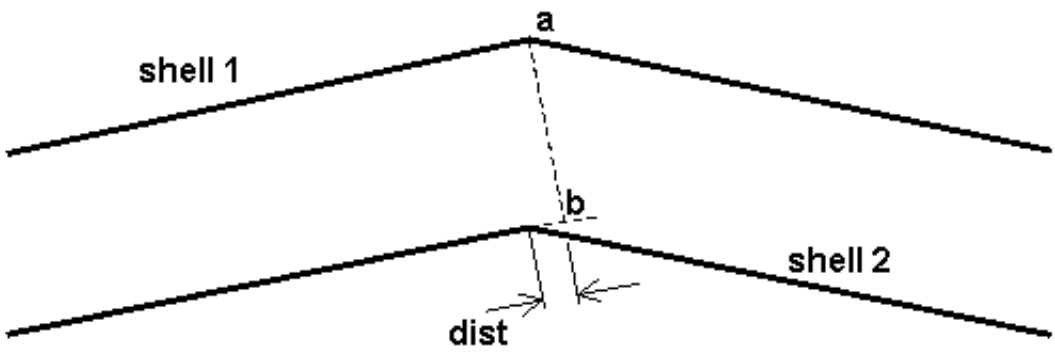
The **edge tolerance** is used to try to find elements if the point you pick is not on a flange. The figure below shows 2 panels with flanges. We want to weld the 2 panels together. The point that is picked is not actually on (or near) the flange. PRIMER checks to find the distance from the panel flanges. If this is less than the edge tolerance then the weld will be made at the end of the flanges. This may not be ideal and in reality you do not want welds on the edges of flanges. To avoid this try to ensure that the weld points are on flanges.



The edge tolerance is also important when welding curved panels. The spotwelder in PRIMER works by locating a point on a panel/element and then creating a beam that is perpendicular to that element. For a curved panel this sometimes does not work. The weld that we actually want to make is shown in the top figure.



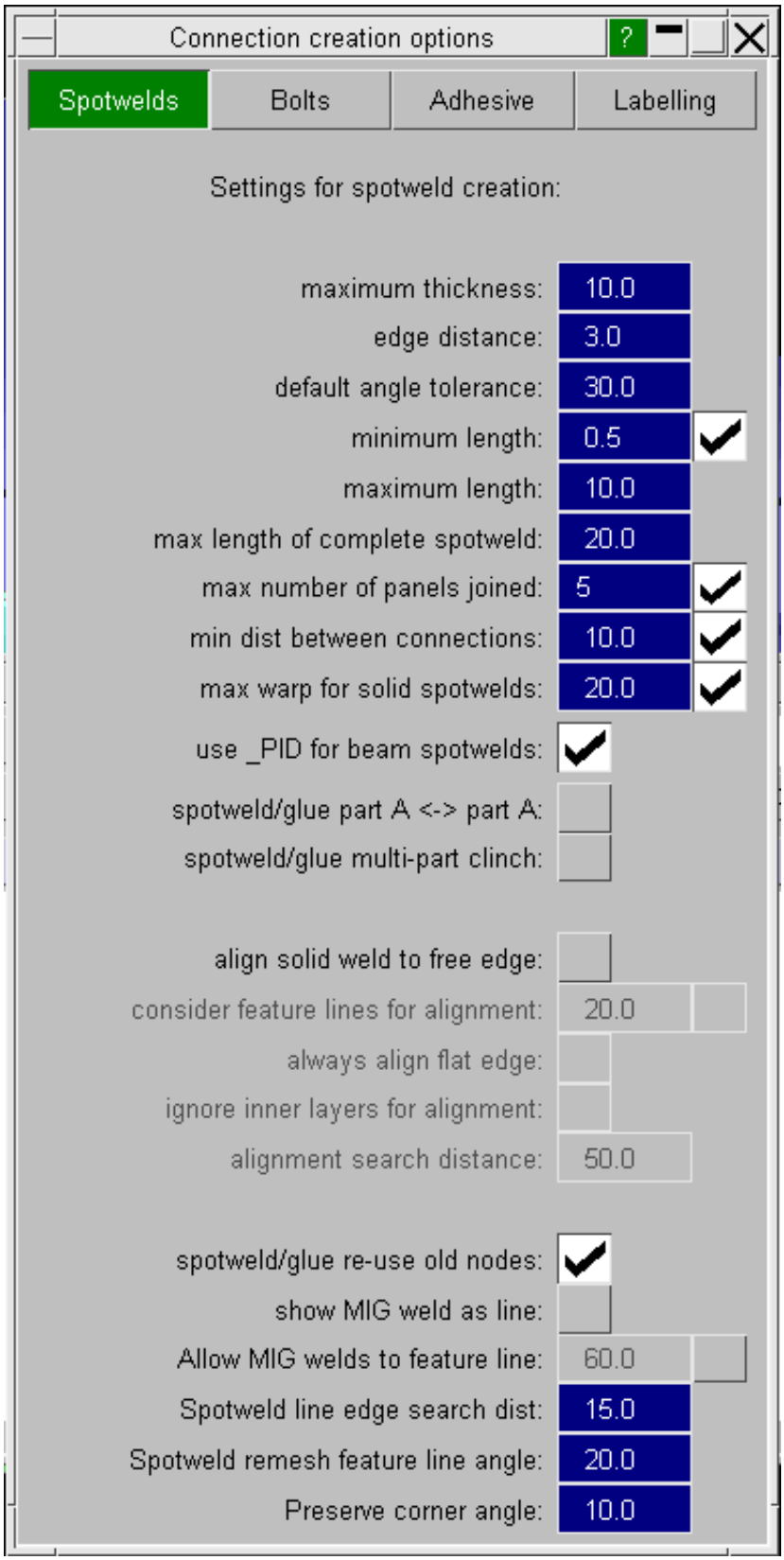
Point **a** is selected for the weld and PRIMER chooses **shell 1** for one end of the weld. PRIMER then projects perpendicular to **shell 1** and looks for elements to weld. PRIMER finds **shell 2**. If PRIMER used this point the weld may not be able to be made as the angle between the panels may be greater than the angle tolerance. Instead, PRIMER checks to see if the distance, **dist**, is less than the edge distance. If it is then the weld (as shown above) can be made.



Other options used when checking/creating connections

Some other options are used in the spotwelder. These are found in the **Settings...** panel. These are seperated into spotwelds, bolts, adheisve and labelling.

Spotweld options



The **minimum length** and **maximum length** set the minimum and maximum allowed length for a single spotweld beam/solid. This is different to the **max length of complete spotweld** which is the total length of all the spotweld beams/solids in the weld. The figure below shows the difference between these.

max number of panels joined sets the maximum number of panels that PRIMER will allow to be connected together.

The **min distance between spot** allows PRIMER to check the pitch between connections. If a panel has 2 connections that are closer than the minimum distance a warning will be printed. This is very useful to checking for bad

weld positions or possible manufacturing problems.

max warp for solid spotweld sets how distorted solid spotwelds can be before PRIMER refuses to create them.

The **use _PID for beam spotwelds** option sets the _PID option on a beam element when creating a spotweld, and supplies the appropriate part ID's.

When **spotweld/glue part A <-> part A** is active, PRIMER is able to connection the same part together (useful for parts folded on themselves or clinches).

When **spotweld/glue multi part clinch** is active, PRIMER is will allow multiple part clinches to be created.

Align solid weld to free edge is used to align the olsid by checking to se if any free edges are nearby.

Consider feature lines for alignment – Consider local feature lines as well as free edges when aligning spotweld – a break angle can also be modified to specify the feature line.

Always align flat edge – always align a flat edge of a solid spotweld with the closest free edge/feature line.

Ignore inner layers for alignment – only consider the outer layers of parts being joined together for alignment – i.e. ignore free edges/feature lines on the inner layers.

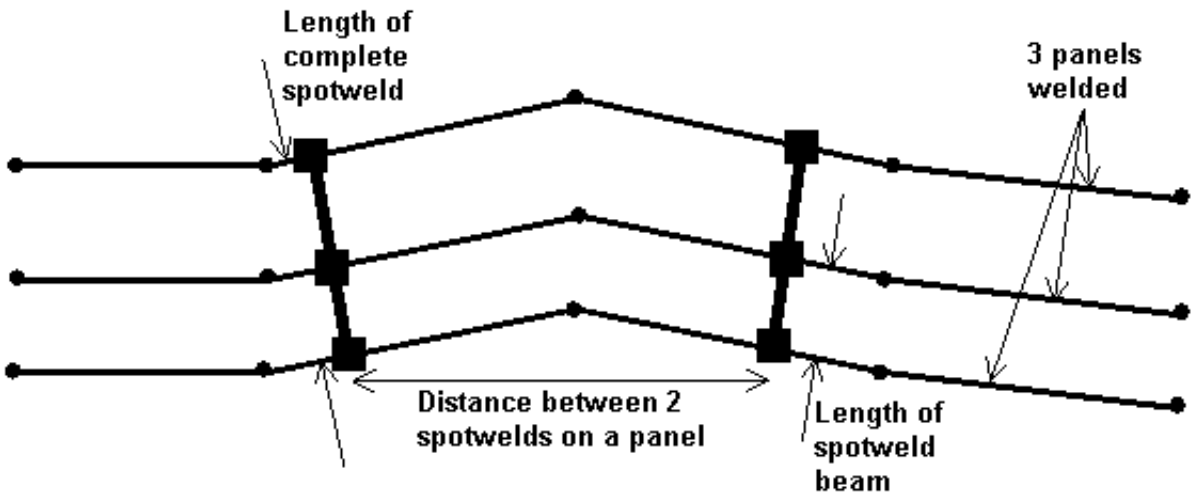
Alignment search distance – Search distance for finding free edges/feature lines near the spotweld.

Reuse old nodes is used when remaking spotwelds - node ID's in the original FE are used again in the new FE.

Show MIG weld as line is a graphics option that displays a line through groupings of MIG welds.

Allow MIG welds to feature line is a setting for allowing MIG welds to connect to a feature line as well as free edges. A break angle is given to specifiy what constitutes a feature line.

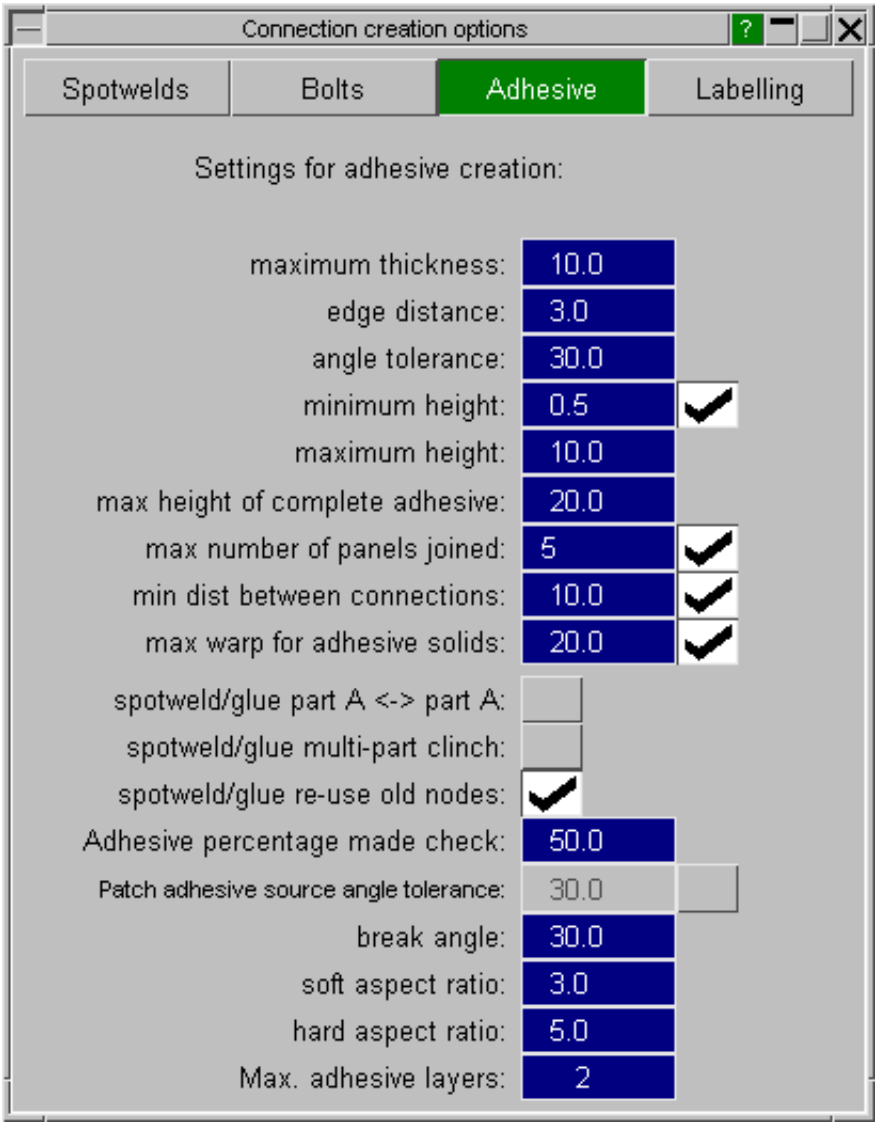
Spotweld line edge search distance is used when a spotweld line connection type is "edge locked" and you wish to re-create the path along a new free edge. This search distance is how far from the start and end points of the path PRIMER will search for a new start and end point on the new free edge.



Spotweld remesh feature line angle is the angle above which defines feature lines during spotweld remeshing.

Preserve corner angle is the minimum angle above which corners at nodes on free edges are preserved during spotweld remeshing. A value of zero would preserve all nodes on free edges; a value of 180 would not preserve any nodes, potentially leading to rounding of corners.

Adhesive options



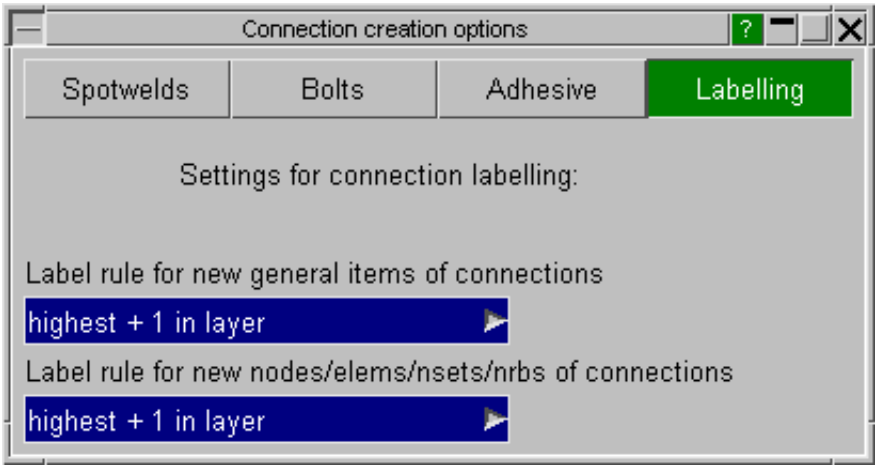
Some settings for adheisve are the same as for spotwelds as described above. Adheisve specific options are:

Adhesive percentrage check is used when checking adhesive connection types. If the percentage of solid elements created along the path compared to the maximum number possible along its path is less than the percentrage value specified, then an error will be given when checking the connection.

Patch adhesive source angle tolerance is used when projecting shells to create patch adhesive. Switch this on to make the resulting solids align more closely to the source shells.

More information on adheisve settings can be found in the adhesive creation section.

Labelling



The **label rule for new general items** and **label rule for new nodes/elems/nsets/nrbs** allow you to set what labels are chosen for new entities that are created.

Bolts

For belts, see the bolt creation section.

6.12.14 Spotweld file formats

PRIMER spotweld file format

The PRIMER spotweld file format is designed to be able to be easily read by PRIMER and people. The file should contain either comment lines or spotweld data lines

Comment lines

Any line that begins with a $ (dollar symbol) is treated as a comment line and skipped. This is the same as LS-DYNA.

Spotweld data lines

Any line that is not a comment line is treated as a line containing spotweld data.
Each line contains data for one spotweld.
Each line consists of up to 15 fields.
Each field is 10 characters wide (like most LS-DYNA keyword data) giving a maximum line length of 150 characters.

The fields are defined as follows:

Field number(s)	Column numbers	Description
1	1-10	Field skipped. Usually contains string 'SPOTWELD' for readability
2	11-20	Spotweld ID number
3	21-30	Field skipped. Usually contains string 'POINT' for readability
4	31-40	Spotweld point X coordinate
5	41-50	Spotweld point Y coordinate
6	51-60	Spotweld point Z coordinate
7	61-70	Field skipped. Usually contains string 'PART' for readability
8-15	71-150	Part ID number. Up to 8 panels to weld together.

Example file

```
$ Primer spotweld file
$ ====================
$
$ Created on: Thu Feb 15 15:04:34 2001
$
$ from model: "ARUP CRASH MODEL - Training (workshop6)"
$
$         < weld ID>          < X coord>< Y coord>< Z coord>          < Part 1 >< Part 2 >< Part 3 >
$
SPOTWELD           1     POINT  2602.424 -692.2456   606.822     PARTS       113         5
SPOTWELD           2     POINT  2623.942-692.18854    605.34     PARTS       113         5
SPOTWELD           3     POINT  2634.647 -692.1887  604.6709     PARTS       113         5
SPOTWELD           4     POINT   3142.42 -695.2453  547.7576     PARTS         5       306
SPOTWELD           5     POINT   3142.42 -695.2453  522.7576     PARTS         5       306
SPOTWELD           6     POINT  3101.431-692.23694  483.0748     PARTS       306         5
SPOTWELD           7     POINT  3075.443-692.23694 483.39206     PARTS       306         5
SPOTWELD           8     POINT  3049.455-692.23694  483.7093     PARTS       306         5
SPOTWELD           9     POINT  3023.467-692.23694  484.0266     PARTS       306         5
SPOTWELD          10     POINT  3102.919-692.23694  587.7574     PARTS       306         5
```

Catia spotweld file format

The Catia spotweld file has a set format and is read into PRIMER in the following way:

The lines are split into words and parsed until one matches the following format:
<string> <string> <string> <floating point number>,<floating point number>,<floating point number> <strings>

The first string is stripped of non-digits to obtain a weld ID. e.g. 4118-5o converts to 41185.
The second string identifies how many parts the weld should attach and the reader should look for.
The three floating point numbers are stored as X Y and Z coordinates for the weld.

The reader then takes the amount of parts to look for (from the second string) and looks in the following lines for a part ID. The part ID is taken from characters 28 to 34 inclusive and should result in a 7 digit number. If it's identified as a 2 thickness weld, then Primer will expect a part ID in each of the next 2 lines.

An example file is as follows:

```
=====================================================================
| BIW Assembly Features :                                           |
=====================================================================

SPOT WELDS
==========

ID      Feature Location (X,Y,Z)
--      ------- ----------------
4118-5o 2T Spot 4074.49,-295.82,1680.00 Ordinary class 3 Jul 29,2002 No
   - NEED TO BE FIXED - 1: 55354119AB A (MO-6000 44A, 0.79 mm)
   - NEED TO BE FIXED - 2: 55396175AA A (M3-67, 2.03 mm)
4118-7o 2T Spot 2041.68, -301.24, 1680.00 Ordinary class 3 Jul 29,2002 No
   - NEED TO BE FIXED - 1: 55354119AB A (MO-6000 44A, 0.79 mm)
   - NEED TO BE FIXED - 2: 55396175AA A (M4-67, 2.03 mm)
4118-9o 2T Spot 2038.04, -356.46, 1680.00 Ordinary class 3 Jul 29,2002 No
   - NEED TO BE FIXED - 1: 55354119AB A (MO-6000 44A, 0.79 mm)
   - NEED TO BE FIXED - 2: 55396175AA A (MY-67, 2.03 mm)
```

UG spotweld file format

The UG spotweld file has a set format and is read into PRIMER in the following way:

The first line is skipped, and after that the only lines PRIMER reads are lines containing "spot". PRIMER expects the following comma separated order:

<string>,<weld id>,<number of panels to weld>,<X coord>,<Y coord>,<Z coord>,<part strings>

The part strings should match exactly what PRIMER contains in the *PART title fields.

An example file is as follows:

```
WELD_TYPE,ID,NUMBER OF SHEETS WELDED,X_POS,Y_POS,Z_POS,CONNECTED PART
1,CONNECTED PART 2,CONNECTED PART 3,CONNECTED PART 4
SPOT_WELD_TYPE_UNKNOWN
resistance
spot,2,2,2623.941895,-693.717041,605.340027,A_pillar_lower_support_a,sill_swan_neck,
resistance
spot,3,2,2634.646973,-693.717102,604.670898,A_pillar_lower_support_a,sill_swan_neck,
resistance
spot,4,2,3142.419922,-693.741089,547.757629,sill_swan_neck,seat_xmember_outer,
resistance
spot,5,2,3142.419922,-693.741089,522.757629,sill_swan_neck,seat_xmember_outer,
resistance
spot,6,2,3101.430908,-693.741089,483.074829,seat_xmember_outer,sill_swan_neck,
resistance
spot,7,2,3075.443115,-693.741089,483.392059,seat_xmember_outer,sill_swan_neck,
resistance
spot,8,2,3049.455078,-693.741150,483.709290,seat_xmember_outer,sill_swan_neck,
resistance
spot,9,2,3023.467041,-693.741150,484.026611,seat_xmember_outer,sill_swan_neck
```

VIP spotweld file format

The VIP spotweld file has a set format and is read into PRIMER in the following way:

The only lines PRIMER reads are lines containing "WSPOT". PRIMER expects the following comma separated order:

<string>,<weld id>,<diameter>,<X coord>,<Y coord>,<Z coord>,<part strings>

The part strings can either be numbers to represent part ID's, or strings to represent CAD name. With the CAD name method, PRIMER will look for any parts which have a specific CAD name specified through the BOM feature. Failing that, PRIMER will look at the heading/title of the *PART cards to see if they contain the string specified. Once found, these parts are used to create the connection. Note that for the CAD name method, PRIMER will reference all parts that

contain the string in it's heading/title, not just the first one found. Note that id a diameter is not set in the VIP file, PRIMER will use the diameter set on the connection read panel (default 5.0).

An example file is as follows:

```
WSPOT,1,5.0,2634.646973,-693.717102,604.670898,A_pillar_lower_support_a,sill_swan_neck,

WSPOT,2,5.0,3142.419922,-693.741089,547.757629,sill_swan_neck,seat_xmember_outer,

WSPOT,3,6.0,3142.419922,-693.741089,522.757629,sill_swan_neck,seat_xmember_outer,

WSPOT,4,6.0,3101.430908,-693.741089,483.074829,seat_xmember_outer,sill_swan_neck,

WSPOT,5,5.0,3075.443115,-693.741089,483.392059,seat_xmember_outer,sill_swan_neck,

WSPOT,6,4.5,3049.455078,-693.741150,483.709290,seat_xmember_outer,sill_swan_neck,
WSPOT,7,4.0,3023.467041,-693.741150,484.026611,seat_xmember_outer,sill_swan_neck
```

Master Connection File (MCF) format

An MCF file is a connections file written in XML format. An example of an MCF file is given below:

```
<?xml version="1.0" encoding="UTF-8"?>
<xml_connection_file xmlns:xsi="http://www.w3.org/2000/10/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="mcf.xsd">
  <version> 1.0.0 </version>
  <connection_group>
    <id> 1 </id>
    <connected_to>
      <pid> 1 </pid>
      <pid> 2 </pid>
    </connected_to>
    <connection_list>
      <connection_0d>
        <id> 100001 </id>
        <type>spotweld</type>
        <loc> 42.39  83.91  1.94 </loc>
        <info>  dia  5. </info>
        <info>  comment This_Is_A_Comment </info>
      </connection_0d>
      <connection_0d>
        <id> 100002 </id>
        <type>spotweld</type>
        <loc> 74.29 49.12 1.94 </loc>
        <info>  dia  5. </info>
      </connection_0d>
      <connection_1d>
        <id> 100009 </id>
        <type>adhesive</type>
        <loc_list>
          <loc> 1.0  50.0  0.0 </loc>
          <loc> 2.0  50.0  0.0 </loc>
          <loc> 3.0  50.0  0.0 </loc>
          <loc> 4.0  50.0  0.0 </loc>
          <loc> 5.0  50.0  0.0 </loc>
          <loc> 6.0  50.0  0.0 </loc>
          <loc> 7.0  50.0  0.0 </loc>
        </loc_list>
        <info> width 7. </info>
      </connection_1d>
      <connection_2d>
        <id> 4429 </id>
        <type>adhesive</type>
        <loc_list>
          <loc>  2748.69 -349.67 1386.83 </loc>
          <loc>  2749.49 -359.62 1386.08 </loc>
          <loc>  2739.23 -350.69 1389.11 </loc>
          <loc>  2740.04 -360.63 1388.34 </loc>
          <loc>  2729.78 -351.71 1391.38 </loc>
          <loc>  2730.59 -361.65 1390.60 </loc>
          <loc>  2720.32 -352.73 1393.65 </loc>
```

```
          <loc>  2721.15 -362.67 1392.86 </loc>
          <loc>  2710.86 -353.75 1395.93 </loc>
          <loc>  2711.70 -363.69 1395.12 </loc>
        </loc_list>
        <face_list>
          <face> 1 2 4 3  </face>
          <face> 3 4 6 5  </face>
          <face> 5 6 8 7  </face>
          <face> 7 8 10 9 </face>
        </face_list>
        <info>  height  2.0 </info>
      </connection_2d>
    </connection_list>
  </connection_group>
</xml_connection_file>
```

The main tag of the file must be `xml_connection_file`. Each selection of parts that are to be connected then form a `connection_group` and the part ids of the connected parts are listed in the `connected_to` tag.
Each connection is then given inside the `connection_list` tag. There are currently three available types:

- `connection_0d`. These are connections that can be represented by a single location in space.
  The `spotweld` and `gumdrop types` can be read into PRIMER.
  The `loc` tag describes the x, y and z location of the connection in global coordinates.

- `connection_1d`. These are connections that can be represented by a curve in space.
  The `spotline` and `adhesive types` can be read into PRIMER.
  The `loc_list` tag encompasses the x, y and z coordinates that make up the connection curve.

- `connection_2d`. These are connections that can be represented by a surface in space.
  The `adhesive type` can be read into PRIMER.
  The `loc_list` tag encompasses the x, y and z coordinates that make up the mesh used to define the connection plane.
  The `face_list` tag then describes the tesselation that defines the mesh, where the numbers in the `face` tags are the indices of the nodal points in the `loc_list` section.

In each of the above cases, the optional `info` tag can be used to define extra information e.g. `dia` is the diameter of a 0d connection and `width` is the width of a 1d connection. If the size of the connection (diameter, width) is not specified the PRIMER defaults will be applied.

PRIMER XML connection file

A small example of a PRIMER XML connection file is given below.

```
<?xml version="1.0"?>
<!-- Primer connection file -->
<!-- ====================== -->
<primer_connections version="14.0">
  <connection type="spotweld">
    <title></title>
    <id>1</id>
    <coord x="11.292786" y="70.951309" z="21.500000" />
    <diameter>5.000000</diameter>
    <method>hexa</method>
    <spotweld_part_id>101</spotweld_part_id>
    <layer type="PART_ID">
      <part id="1" />
      <part id="3" />
    </layer>
    <layer type="PART_ID">
      <part id="2" />
    </layer>
  </connection>
</primer_connections>
```

The main tag of the file must be `primer_connections`. Currently the only version supported is `9.3`.
Each connection is then given inside a `connection` tag. The currently available types are `spotweld, rigid` and `adhesive`. Several tags are then used inside the `connection` tag to define the connection property. Most tags should be obvious.

For connection type `spotweld`, the available `methods` are `beam`, `hexa`, `4_hexa`, `8_hexa`, `12_hexa` and `16_hexa`.
For connection type `rigid`, the available `methods` are `rigid_body_merge` or `nodal_rigid_body`.

For connection type adhesive, the only available method is solid.
For connection type spotweld_line, the available methods are beam, hexa, 4_hexa, 8_hexa, 12_hexa and 16_hexa
The method tag is optional. If it is omitted then PRIMER will use the default option when reading the connection file.

Spotweld connections can contain the spotweld_part_id tag which tells PRIMER what part to create the spotweld in. It is optional. If it is omitted then PRIMER will use the default option when reading the connection file. For rigid connections there is an equivalent optional tag to specify the material called rigid_material_id, and for adhesive connections there is an equivalent optional tag to specify the material called adhesive_material_id.

Adhesives and spotweld lines contain more information than individual spotwelds and bolts (an example is shown below). The adhesive and spotweld line information contains the end point (coord2) and the path data (points between the start and end points). Adhesive also contains adhesive width, number of elements across the width and element size.

```
<?Xml version="1.0"?>
<!-- Primer connection file -->
<!-- ===================== -->
<primer_connections version="14.0">
  <connection type="adhesive">
    <title></title>
    <id>2</id>
    <coord x="-13.702406" y="-39.206017" z="54.000000"    />
    <coord2 x="100.094444" y="-33.088951" z="54.000000"    />
    <method>solid</method>
    <adhesive_part_id>600</adhesive_part_id>
    <adhesive width="10.000000" number="1" size="10.000000"    />
    <path x="18.048088" y="-27.360266" z="53.999996"    />
    <path x="42.419262" y="-18.718697" z="54.000000"    />
    <path x="82.520035" y="-17.456446" z="54.000000"    />
    <layer type="PART_ID">
      <part id="10" />
    </layer>
    <layer type="PART_ID">
      <part id="21" />
    </layer>
  </connection>
/primer_connections>
```

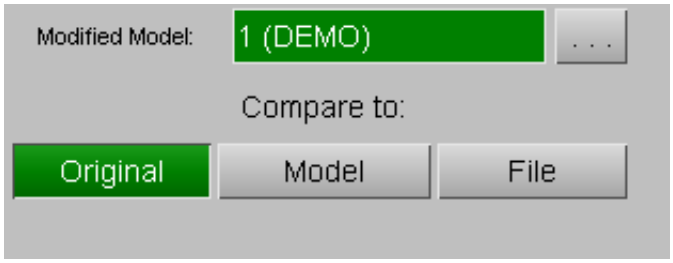
A layer tag is then given for each layer of the connection. The available layer types are PART_ID, PART_NAME, CAD_NAME, ASSEMBLY, SETPART_ID, and SETPART_NAME. The layer tag then contains the item(s) that are used for the layer. The tag and attribute change depending on the layer type. The following table shows the values.

Layer type	Tag	Attribute	Example
PART_ID	part	id	<part id="1" />
PART_NAME	part	name	<part name="panel12345" />
CAD_NAME	part	CADname	<part CADname="CATIA_12345_xyz" />
ASSEMBLY	assembly	name	<assembly name="biw" />
SETPART_ID	partset	id	<partset id="1" />
SETPART_NAME	partset	name	<partset name="body_side_parts" />

6.12.15 Connection compare

The connection compare feature allows users to compare connections across models or with connection files.

There are three ways in connections of a model can be compared to:



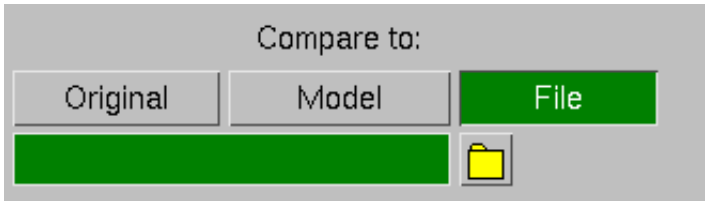
Original - Compare with the original model connections.

Model - Compare with the any other model in Primer.

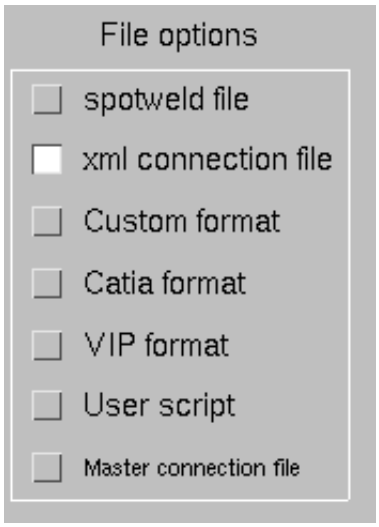
File - Compare with the connections present in connection files.

In case you select **Model** , following buttons would be enabled to select/input model to be compared with.

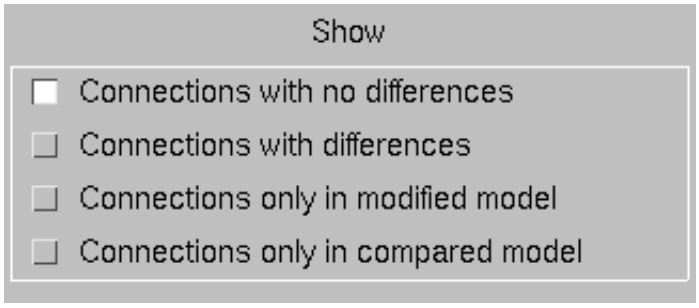
In case you select **File** , following buttons would be enabled to select/input connection file to be compared with.



Also, **File options** are enabled to select the type of connection file to be compared with.



One of the follwing modes of comparison should be selected while comparing:

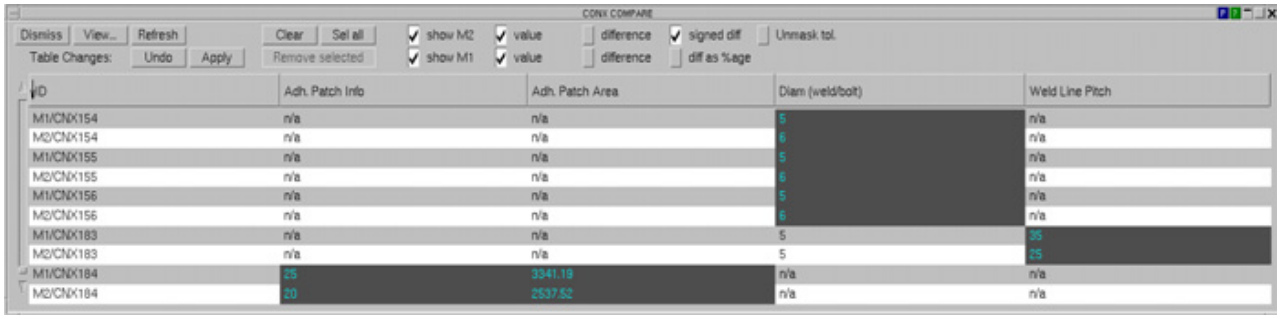


The **Tolerance** is the spatial tolerance to be used while comparing connections.

After selecting the required comparison options hit **Apply.**

This brings up a connection compare table or a normal connection table depending on which mode is selected.

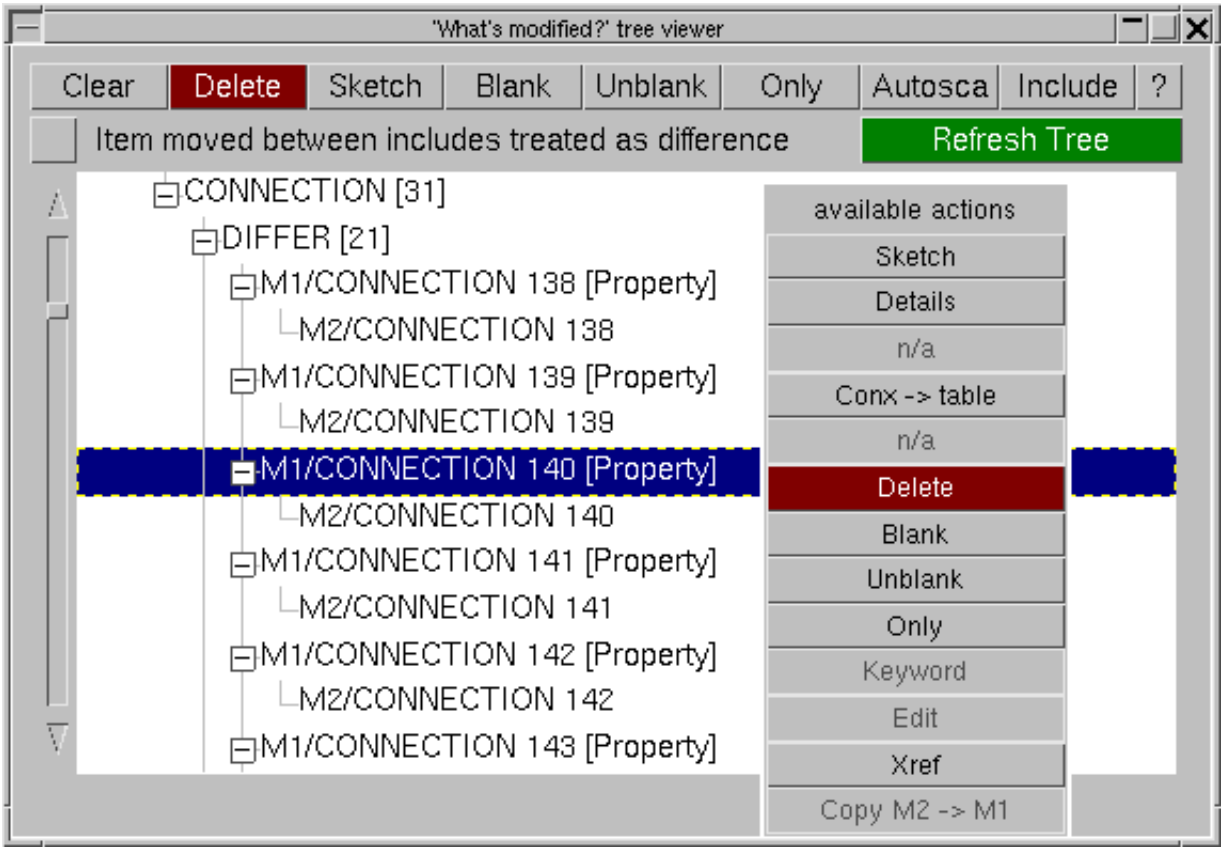
Connection compare table



The connection compare table shows the differences in the properties of spatially matched connections. This can be accessed from:

a) From Model modified - when **Properties** " button is turned on under futher options for comparing connections.

Then selecting **Conx-table** from the popup as shown below.

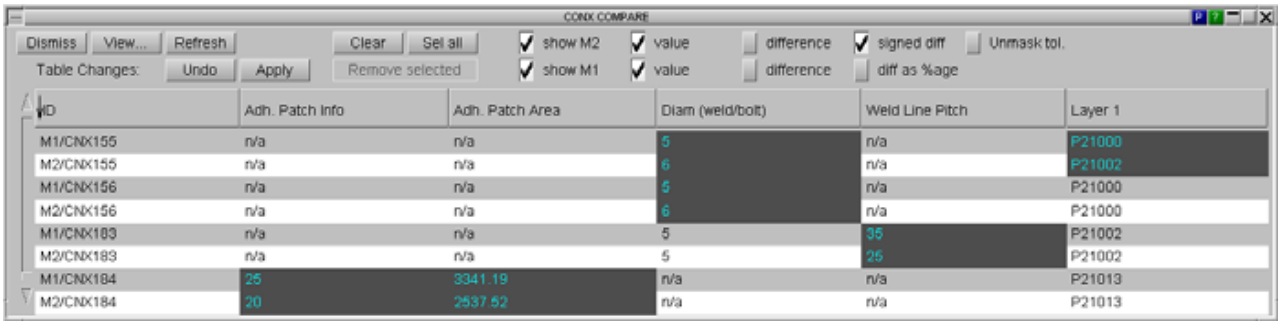


b) From Connection compare - when show **Connections with differences** is selected.

All the matched connections are checked against one another for all the values that the connection table treats (see section 6.12.2) and a connection compare table is constructed for the connection pairs which show differences.

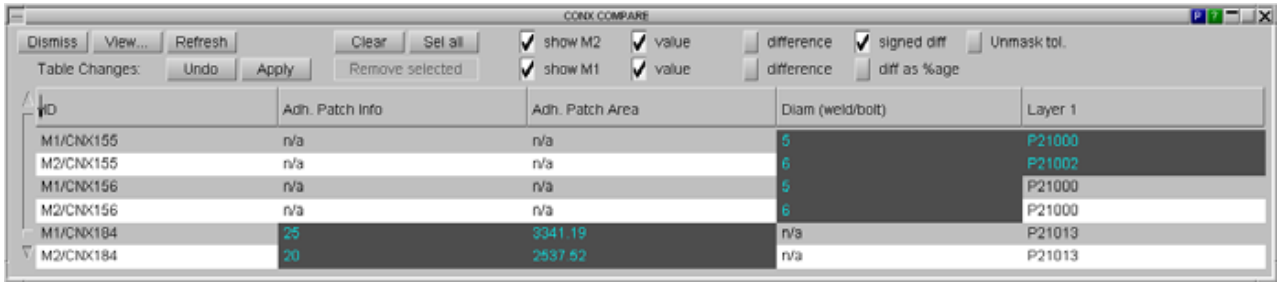
The connections are listed in the form M1/CNXy, M2/CNXy, M1/CNXz, M2/CNXz, etc and the sorting of the Connection ID column will always restore this order.

All the appropriate columns are displayed and the difference is highlighted. In example below, differences in bolt size/diameter, Layer, Patch area, Weld line pitch are shown.

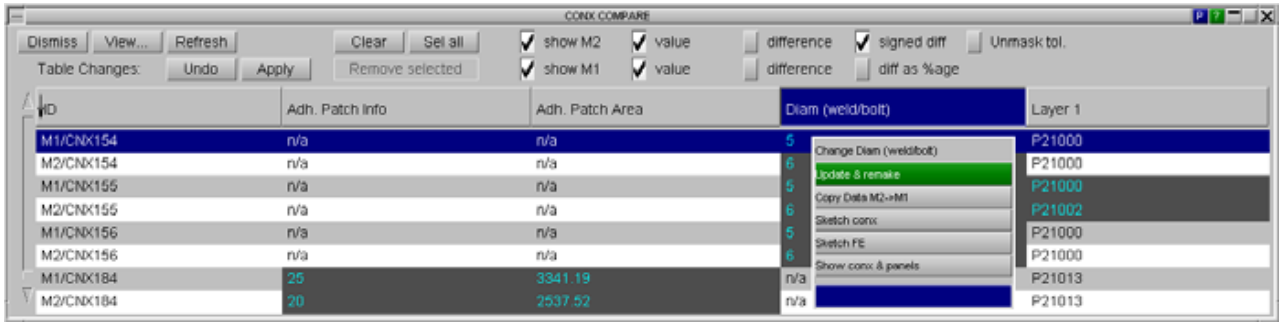


You may use shift-select to select unwanted connection pairs and then apply **Remove Selected**.

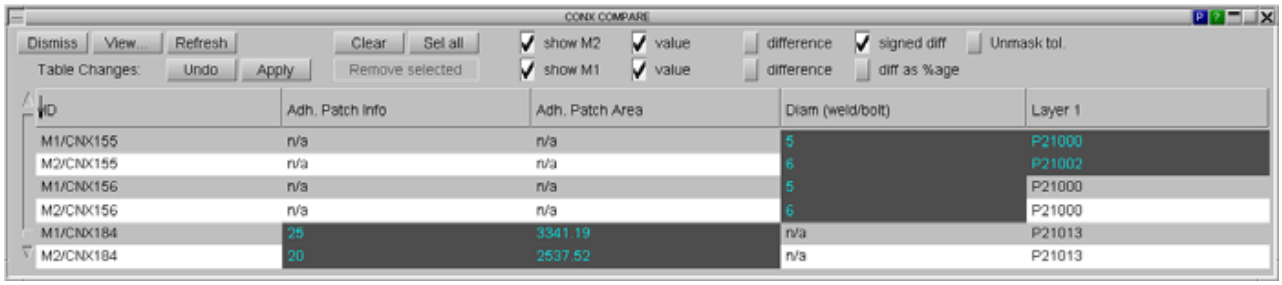
If we are not interested in the weld line pitch change, we can use **View..** to de-activate that column and **Refresh** to rebuild the table. Any connection pairs from which the **only** diference is their include are now removed from the data stacks.



We may then use the connection table functionality to investigate and edit data as appropriate. For example, click on diameter column of M1/CNX154, in the popup enter "6" and hit **Apply**.

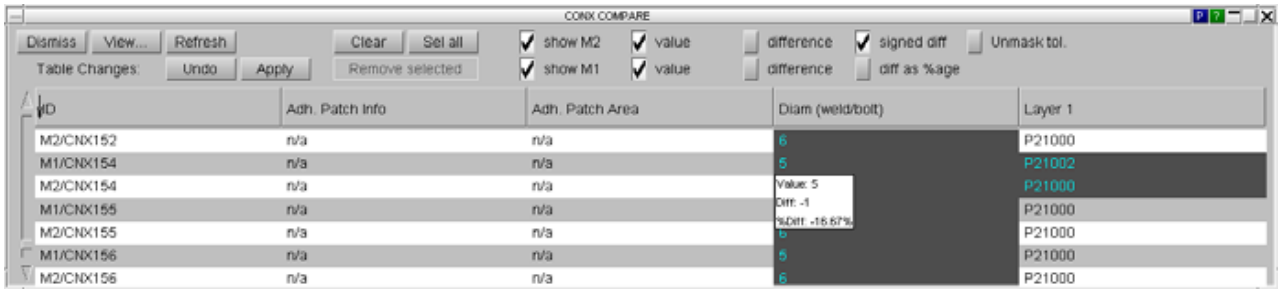


Refresh of the table will then remove connection 154 as the data is consistent across models

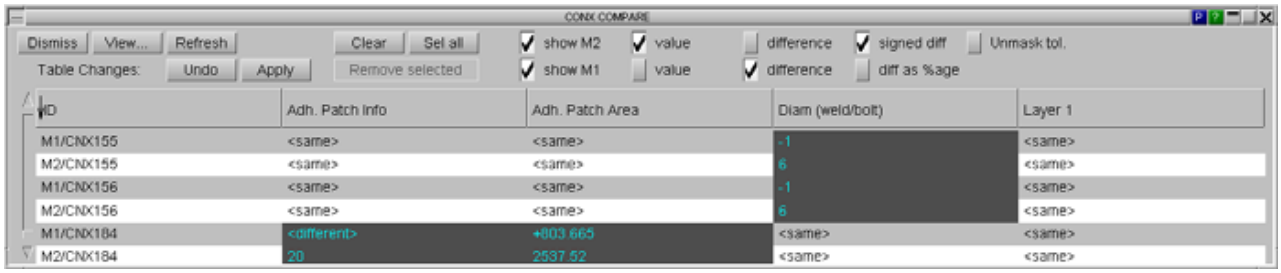


Showing the difference

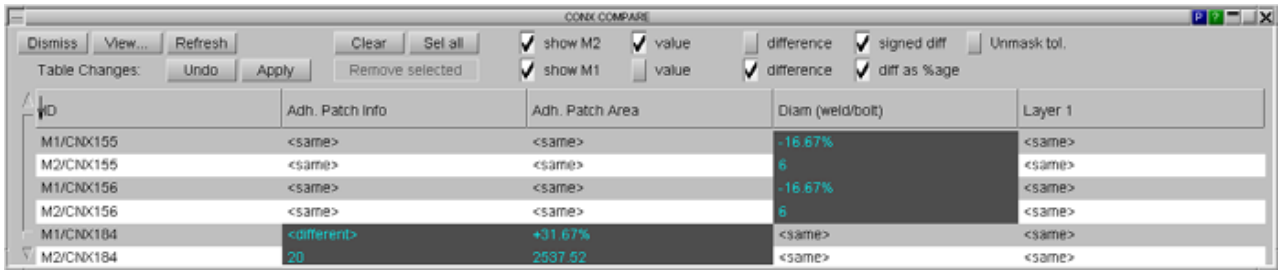
By default the table shows values with hover text to show the absolute and percentage differences wrt the other value.



By activating the difference switch you may show the absolute difference for floating point numbers. For other types the string <different> will be written.

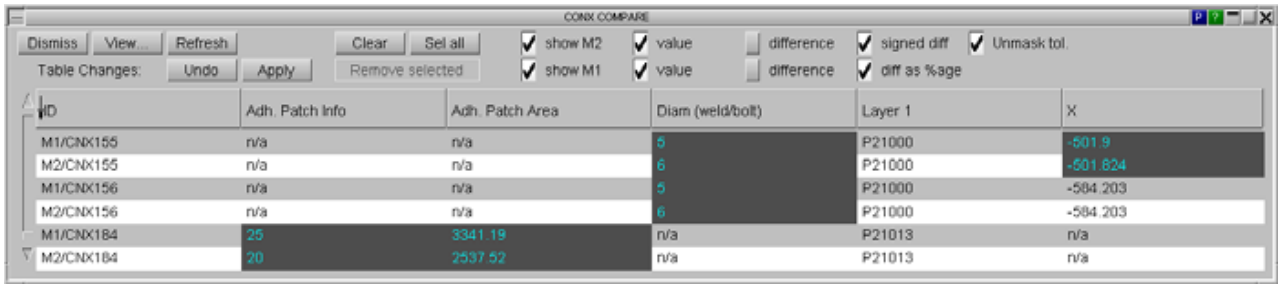


The difference for floating point numbers may also be usefully expressed as a percentage.



Unmask tol, when enabled, will unmask the tolerance used while spatial matching of connections and will show differences in location of the connection, if any.

In the below example, the difference in X coordinate for connection 155 is shown.



6.13 CUT SECTIONS

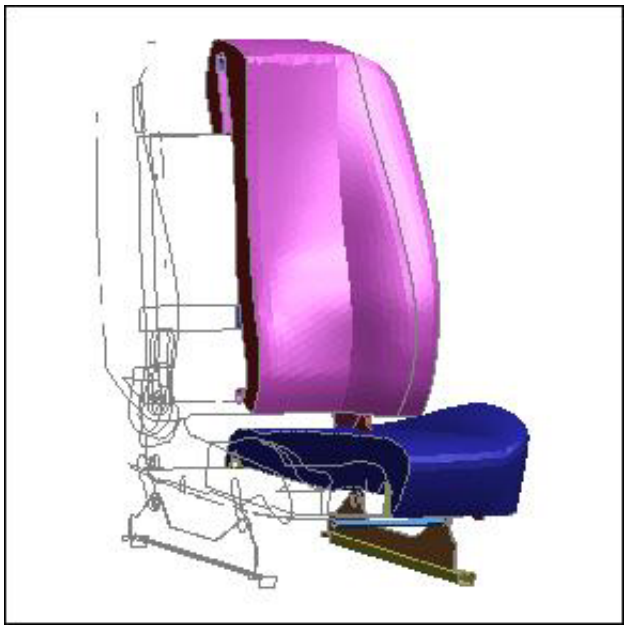
The **Cut Section** menu is invoked from the Tools menu or from keyboard shortcut X.

A cut-section, sometimes referred to as a "cutting plane", is a flat plane that cuts through the model. It may be located anywhere in space and oriented at any angle.

When the **Cutting switch** is turned on the intersection of the plane with the model is calculated and the interpolated cut plane is drawn.

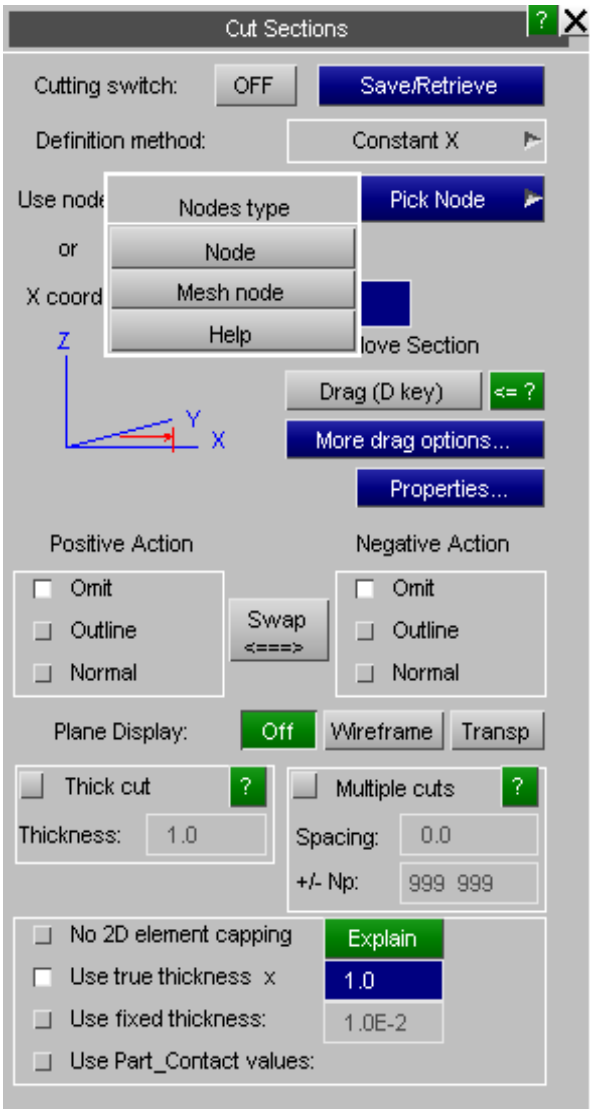
Various options, described below, define if and/or how the model either side of the plane is drawn.

This image shows a conventional plot of a seat model, with no cut sections active



In this image the a cut section has been turned on, located roughly half-way across the seat. The +ve side (on the right) is drawn normally, the -ve side (on the left) is drawn in outline.

6.13.1 The Cut Sections panel



The parts of this panel are summarised below. Click on one to jump to the more detailed description.

Cutting switch	Normally OFF, in which case cut-sections are inactive. May be toggled on/off at any time.
Save/Retrieve	Saves cut-sections, and restores previously saved ones
Definition method	Toggles between the six different ways of creating a cut section
Properties	Extracting engineering properties (Area, I, etc) of the cut elements
Move Section	Various options for dragging the section with the mouse
Positive & Negative action	How the +ve and -ve sides of the plane are displayed
Plane display	Whether, and how, the actual plane itself is displayed
Thick cut	Optional "thick" (finite thickness) plane display
Multiple cuts	Optional multiple parallel cutting planes
2D element capping	How the cuts through 2D elements are displayed

6.13.2 Cutting Switch

Controls whether or not the cutting plane is active

Initially cut sections are turned off and no plane is active. When a plane's properties have been specified turn this on to see the effects. It can be toggled on/off at any time.

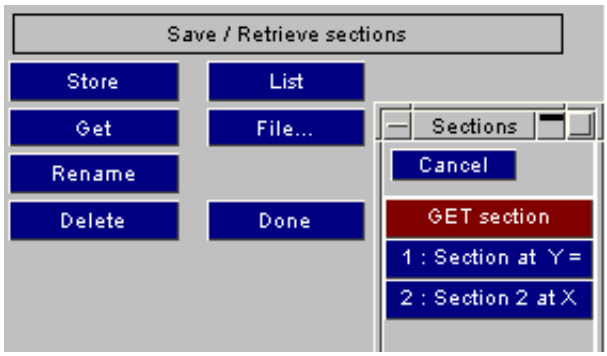
6.13.3 Save/Retrieve

Controls the saving and retrieving of cut-section definitions.

Only one cut section can be active at any time, but any number of cut section definitions can be saved to disk for subsequent retrieval.

To save a section:
- Open a file with **File...** (default "section.cut")
- **Store** the file, giving the section a name.

To retrieve a section:
- Open a file if not already open
- **Get** the section from the list. It will be applied immediately



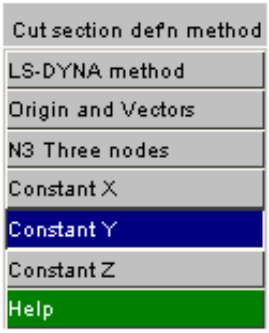
Sections in the file can also be **Delete**d and **Rename**d at will. To return to the main cut section panel use **Done**.

You can open a different cut section file at any time, and have any number of such files on disk. The default name is "section.cut", but any name may be used.

The file format is common with D3PLOT, so the same section definitions can be used in both programmes.

6.13.4 Definition method

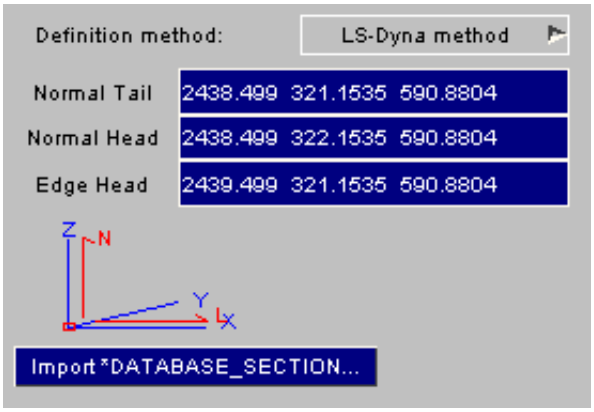
There are six different ways of defining a cut section, chosen from the pull-down menu. The data entry panel changes for each mode. Click on a method below for details.



LS-DYNA method	Tail, Head and Edge head coordinates are defined.
Origin and vectors	Origin coordinate is defined, then vectors for local X axis and XY plane
N3 Three nodes	Three nodes: N1 at origin, N2 giving X axis, N3 the XY plane
Constant X	Cut at constant X value
Constant Y	Cut at constant Y value
Constant Z	Cut at constant Z value

Regardless of how the plane is defined its actual characteristics and geometry will be the same.

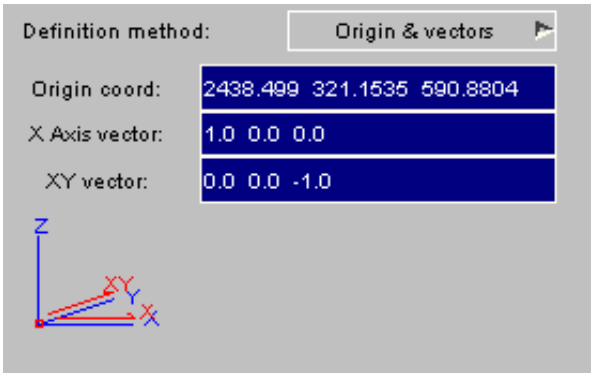
LS-DYNA method



This entry method mimics the data format of the *DATABASE_CROSS_SECTION card in the LS-DYNA input deck. You define:
- The Tail coordinate of the normal vector (origin)
- The Head coordinate of the normal vector (local Z axis)
- A Head coordinate of a vector on the XY plane

If there are any *DATABASE_CROSS_SECTION cards in any models the definition can be built from those using **Import...**

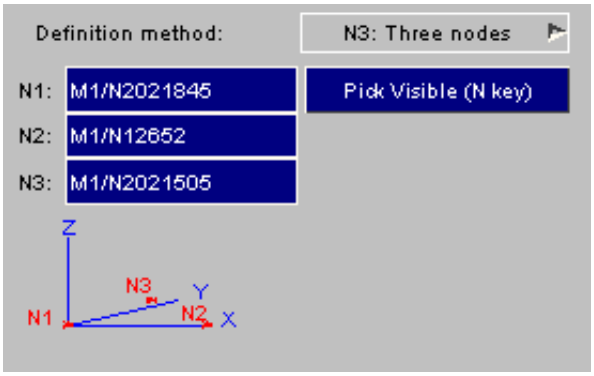
Origin and Vectors



Here you give:
- An origin coordinate
- A vector defining the local X axis
- A vector on the local XY plane

The normal (local Z) vector is obtained from the vector cross product of these.

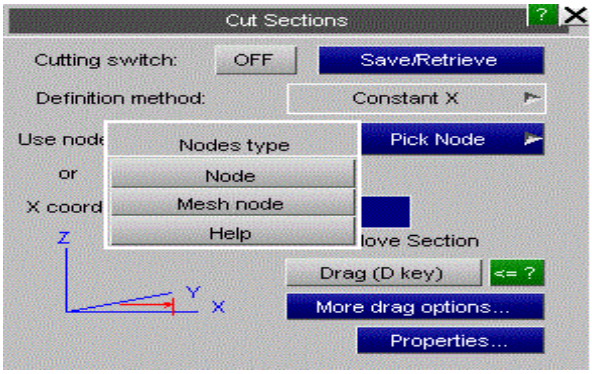
N3 Three nodes



Here three nodes are defined:
- N1 is the plane origin
- N2 is on the local X axis (vector N1N2)
- N3 is on the local XY plane

The normal (local Z) vector is obtained from the vector cross product of these.

Constant X
Constant Y
Constant Z



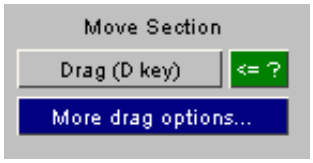
In these cases define either:
- A coordinate on the relevant axis

  or

- A node, or a mesh node, from which the relevant coordinate will be extracted. (The **N** keyboard shortcut will jump straight to this mode.)

A plane will be defined at a constant value of the relevant axis at that point.

6.13.5 Dragging the cut-section



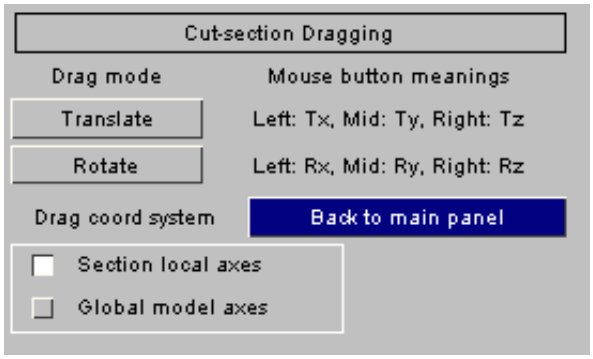
Once the cut-section has been defined it can be moved to a new position and orientation by dragging with the mouse.

Drag (D key) Either clicking on the button, or using the **D** keyboard short-cut invokes this mode.

The Cut-section panel acquires control of the mouse (the cursor symbol changes to "sect drag" to signify this) and the mouse buttons work as follows:

Mouse button	Cursor Symbol	Action
Left	Tz	Translates the plane in the normal (local Z) direction
Middle	Rx	Rotates the plane about its local XX axis
Right	Ry	Rotates the plane about its local YY axis

More drag options...



This gives access to a more complex set of options for dragging the section. You need to choose:

- Drag mode: either translate or rotate
- Drag coordinate system: section local or global

Mouse buttons then translate/rotate in/about axes:

Left button : Tx / Rx
Mid button: Ty / Ry
Right button : Tz / Rz

How mouse motion is interpreted when dragging

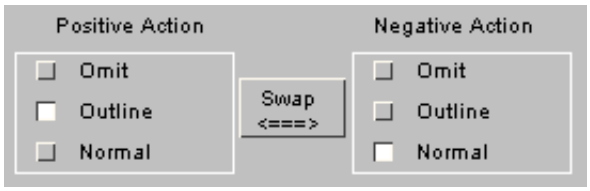
In most cases the mouse motion is projected onto the section axis to be dragged, as shown on the screen, giving an intuitive result as if you had grabbed the section with the mouse and dragged it.

However this method fails when the section axis to be dragged points directly in or out of the screen since the dot product of its vector (screen Z) with mouse motion (screen XY) is zero. Therefore when the axis to be dragged lies within approximately 1 degree of screen +/-Z then an alternative method is used:

- +ve mouse motion in screen X or Y equates to +ve motion down the section drag axis.
- -ve mouse motion gives the opposite effect.

Put more simply: in these cases mouse motion to the right (+X) or up (+Y) results in +ve motion down the section axis, and left (-X) or down (-Y) gives -ve motion.

6.13.6 Positive & Negative action



Controls how the image on either side of the plane is rendered.

The cutting plane itself is always rendered in the current display mode, but for each side of the cutting plane you must choose how the image is to be rendered:

- **Omit** means that it will not be drawn at all
- **Outline** means that it will be drawn in wireframe outline, in the edging mode of the current display mode.
- **Normal** means that it will be drawn in the current display mode.

Swap <===> simply swaps the +ve and -ve display modes around and redraws.

Any permutation of modes can be drawn on either side, here are some examples for the model above:

In this example both +ve and -ve sides have been set to **Outline**.

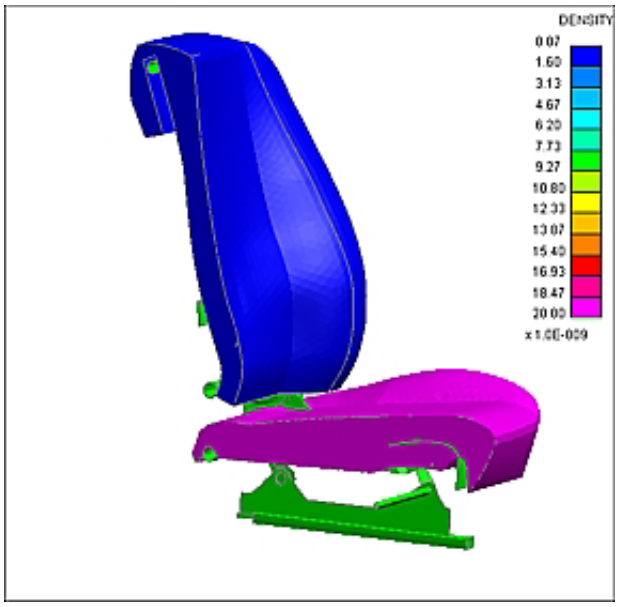
Because the current display mode is **Shaded**, with free edge outlines, this means that they are rendered in free edge wireframe mode.

The cutting plane itself is clearly visible in shaded mode, and because this model contains solid elements these are capped on the cut plane and therefore easy to see.

Here the +ve (far) side is displayed in **Normal** mode, and -ve side has been **Omit**ted.

The display mode is SI this time, showing element density.

This demonstrates the cut-sections can be used with any plotting mode, includingdata-bearing ones.

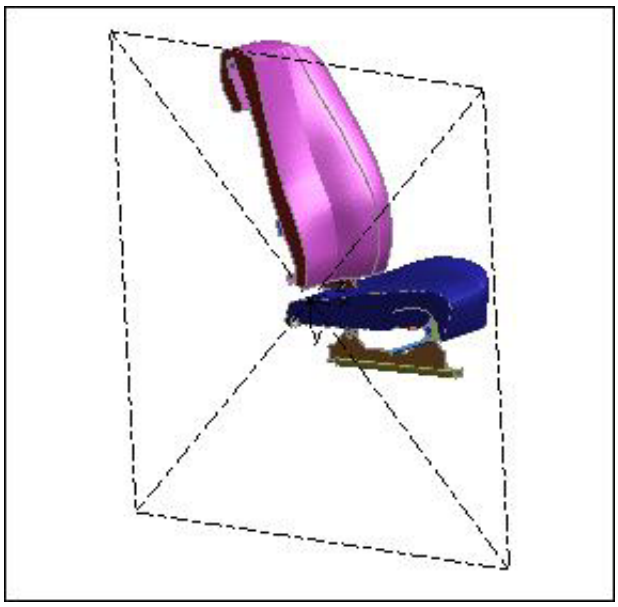


6.13.7 Plane Display

Whether, and how, the actual plane definition itself is displayed. By default plane display if Off and it is not shown. However it is possible to draw the plane in one of two modes:

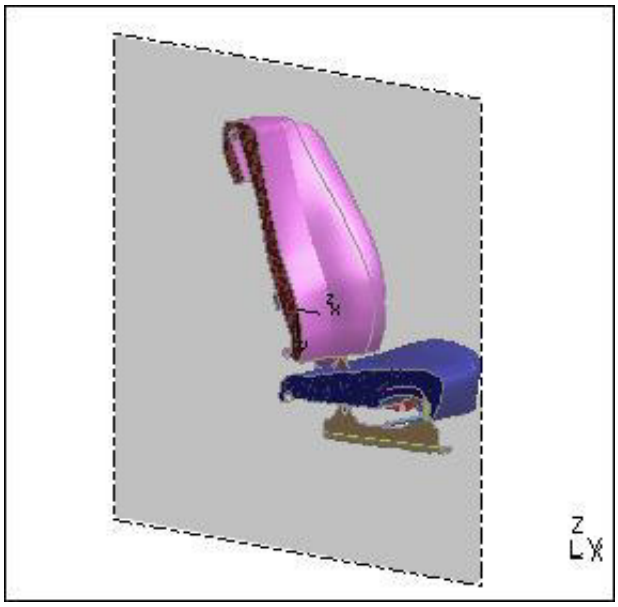
Wireframe

Draws the plane boundaries and a diagonal as a wireframe overlay on the plot



Transparent

Draws the plane as a partially transparent square, occupying model space and intersecting the structure.



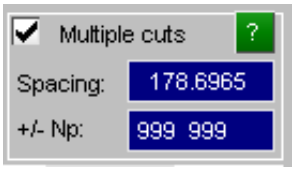
6.13.8 Thick cut

Alternative "thick" plane display mode. In this mode the plane is extruded in the local Z axis by +/- Thickness/2.0 either side of its "thin" position, effectively forming two planes with solid structure in between.

This example shows the model above rendered in this way.

Note that nothing is displayed outside the extruded +ve and -ve planes, (and the **Positive** and **Negative** action options are inoperative).

6.13.9 Multiple cuts



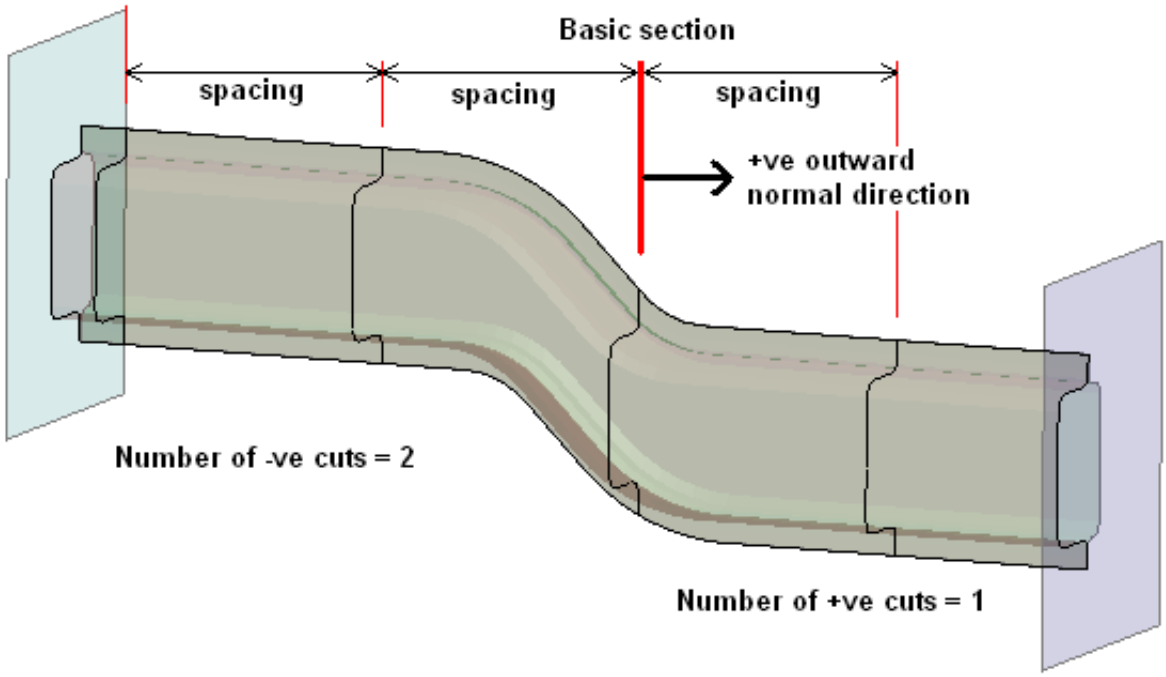
By default only a single cutting plane is used, however you can choose to display multiple parallel planes by using **Multiple Cuts**.

1. Turn this mode on by selecting the tick box.

2. Define the **spacing** between parallel cuts. By default approximately 10% of the diagonal across the bounding box that contains the model(s) will be used.

3. Define how many planes (**Np**) are to be drawn either side of the basic cut section position.

The **Np** values require some explanation

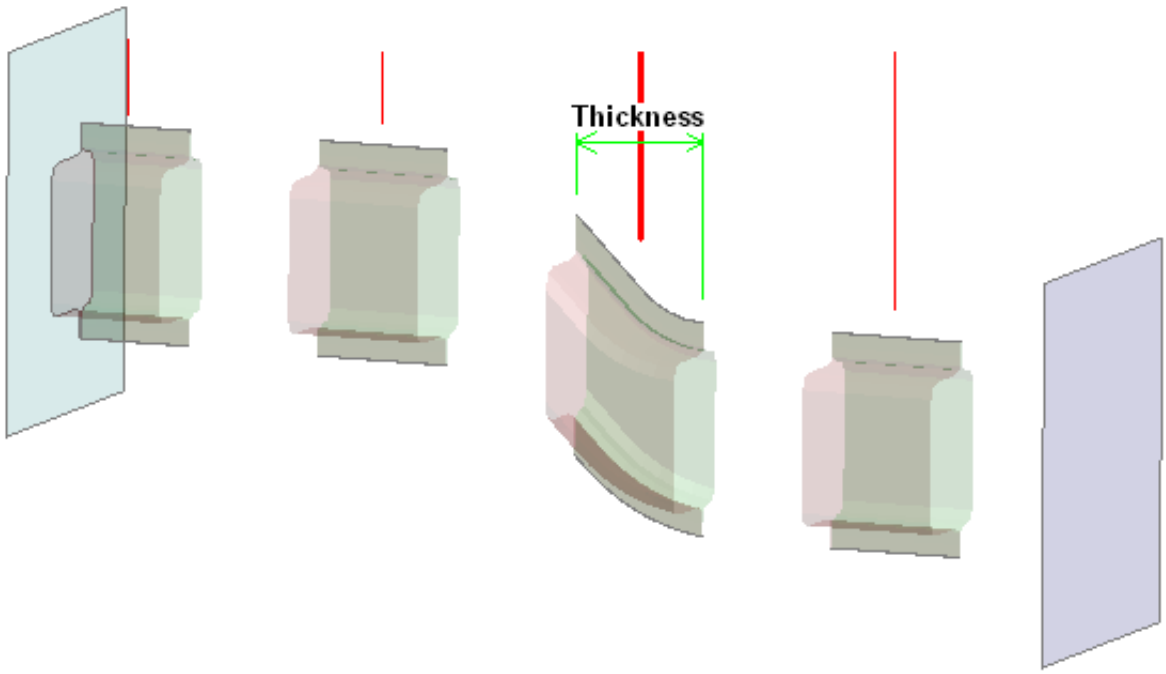


In this example the thick red line shows the basic plane definition, and the thin lines show the repeated multiple planes.

Np has been set to "`1 2`" meaning "**One** cut on the +ve side and **Two** cuts on the -ve side". The positive side of the plane is based on the outward normal of the cut section definition.

It does no harm to set **Np** values that are many times greater than the number of planes that could intersect the model, and in fact the default values are 999 meaning "effectively unlimited". PRIMER will only ever use the actual number of planes that can intersect the model.

Multiple cuts may be combined with "thick" sections.

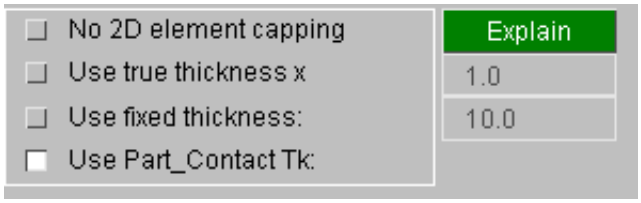


This image shows the same model as above with the "thick sections" switch turned on. It can be seen that multiple thick sections are now drawn.

Warning: Multiple thick sections can be slow.

Multiple thick sections require repeated passes through graphics rendering, so drawing and screen-picking can both become slow if many such cuts are used. The time taken to render a plot will be some function of model size times number of cuts, so it is recommended that you use this combination sparingly when processing large models.

6.13.10 2D element capping

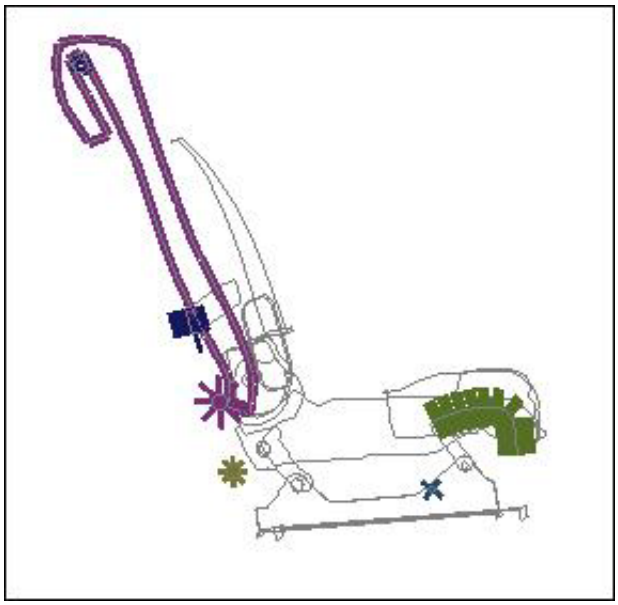


Controls how the cut edges of 2D elements (shells) are displayed.

When shell elements are cut it is possible to draw their cut edges in three modes:

- **No 2D capping**. The cut is simply a line with colour but no thickness.
- **True thickness x factor**. Extracts the true shell thickness, multiplies it by <factor> and uses that value. This is probably the most useful since it shows actual model dimensions, although a factor > 1.0 is often necessary to visualise thicknesses.
- **Fixed thickness**. Uses a constant value in model space units for all shells.
- **Use Part_Contact values**. (See also notes on plotting contact thickness below)

For shells on a **`*PART_CONTACT`** card: If explicit thickness **`OPTT`** is defined this is used. or If scale factor **`SFT`** is defined then the truethickness x **`SFT`** is used or The unscaled true thickness is used.
For other shells the unscaled true thickness is used.



In this example solids have been turned off leaving only shells, which have been rendered using True thickness x 15.0 to make them stand out at this scale.

Showing shell offsets in cut sections.

Normally the neutral axis of a shell lies in the plane of its nodes, but it is possible to offset it in various ways:

- Defining field **NLOC** on ***SECTION_SHELL**
- Using ***ELEMENT_SHELL_OFFSET**
- Using ***INTEGRATION_SHELL**

In addition composites may be modelled using ***ELEMENT_SHELL_COMPOSITE** or ***PART_COMPOSITE**.

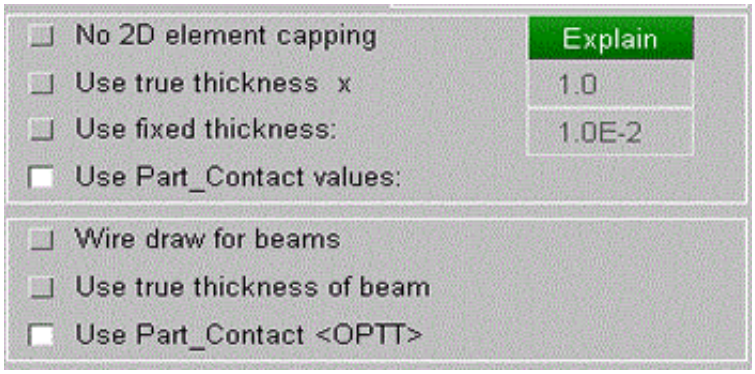
All these methods can result in a shell's neutral axis being offset from the nodal plane, and in the case of composites the shell may have many layers through its thickness at different offsets from the nodal plane.

By default using "true" shell thickness capping does ***not*** take into account these offsets, since that would conflict with the normal PRIMER graphics which render shells as infinitely thin plates on the nodal plane, ignoring both thickness and any offsets. This is because applying the offset to the cut section but not to the uncut element would give a visual offset that could be confusing.

However PRIMER can render shells in "true thickness" mode if the Display Options, Shell, True thickness options are used. This not only draws shells as "thick" using their actual thickness, but also takes into account any offsets to their neutral axis. If this option is in force then cut-section capping of shells will also show any offsets, retaining the logic that the capping cuts the element "as drawn". In addition if composites are being used the various layers will be shown at their correct locations through the thickness.

So to summarise: if you want to see shell offsets in cut sections it is necessary also to turn on "true thickness" display for all shells.

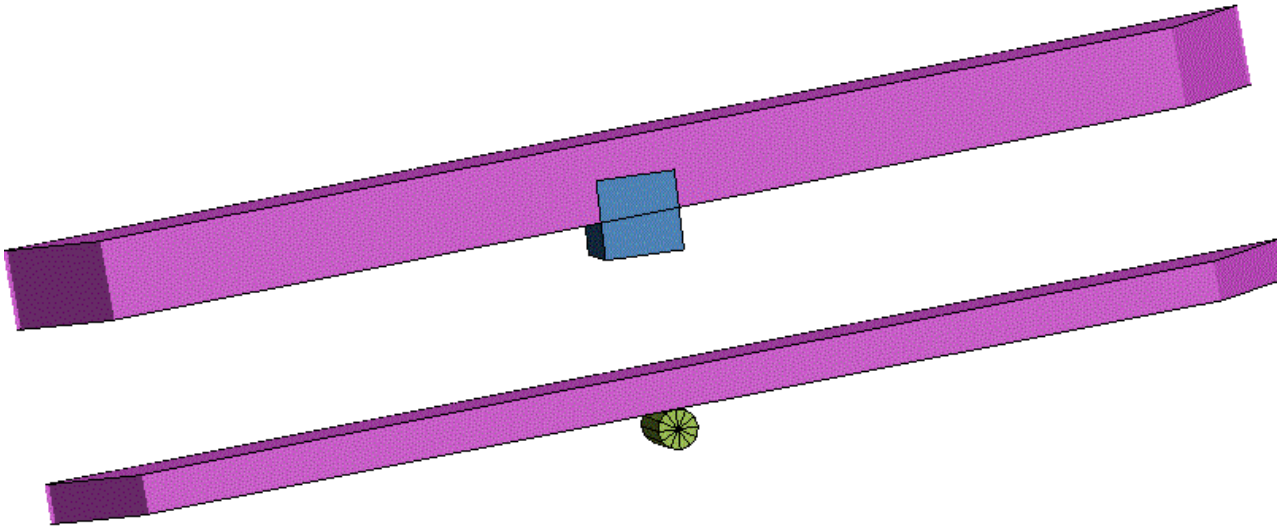
Showing beam thickness in cut sections.



By default beams in cut sections are drawn wireframe. New options allow the beam to be draw as true section or as circular if <OPTT> applies.

Both beams below are 2x2 rectangular section and both shells have physical thickness = 2.

The lower pair of parts have OPTT = 1 both on the shell section and on the beam section so the display options above change how they plot.



Notes on using cut sections to plot contact thickness

There have been requests for cut sections in PRIMER to show contact thickness generally, but this is not really practical for two reasons:

- A given element may be in more than one contact, and the thickness used can be influenced by the contact type and the settings on the contact card itself, so there may not be a unique value.
- Inside LS-DYNA the relationship between elements specified for contact and those actually used is rather weaker than it may at first appear. When a contact surface is created the following process is used to determine the geometry of the contact:
    - Segments are built from all shells or 3D element faces in the contact definition, or explicit segments are used directly.
    - Duplicate segments are eliminated.
    - The element under each segment is then used, whether or not it was specified in the original contact definition.
    - If shells overlay solids, or coincident shells are present, the choice of element is complicated further.

Therefore in a model with more than one contact surface it is nearly impossible to determine a general "thickness used for contact" for every shell, and the only real solution is to limit display to elements in a given contact.

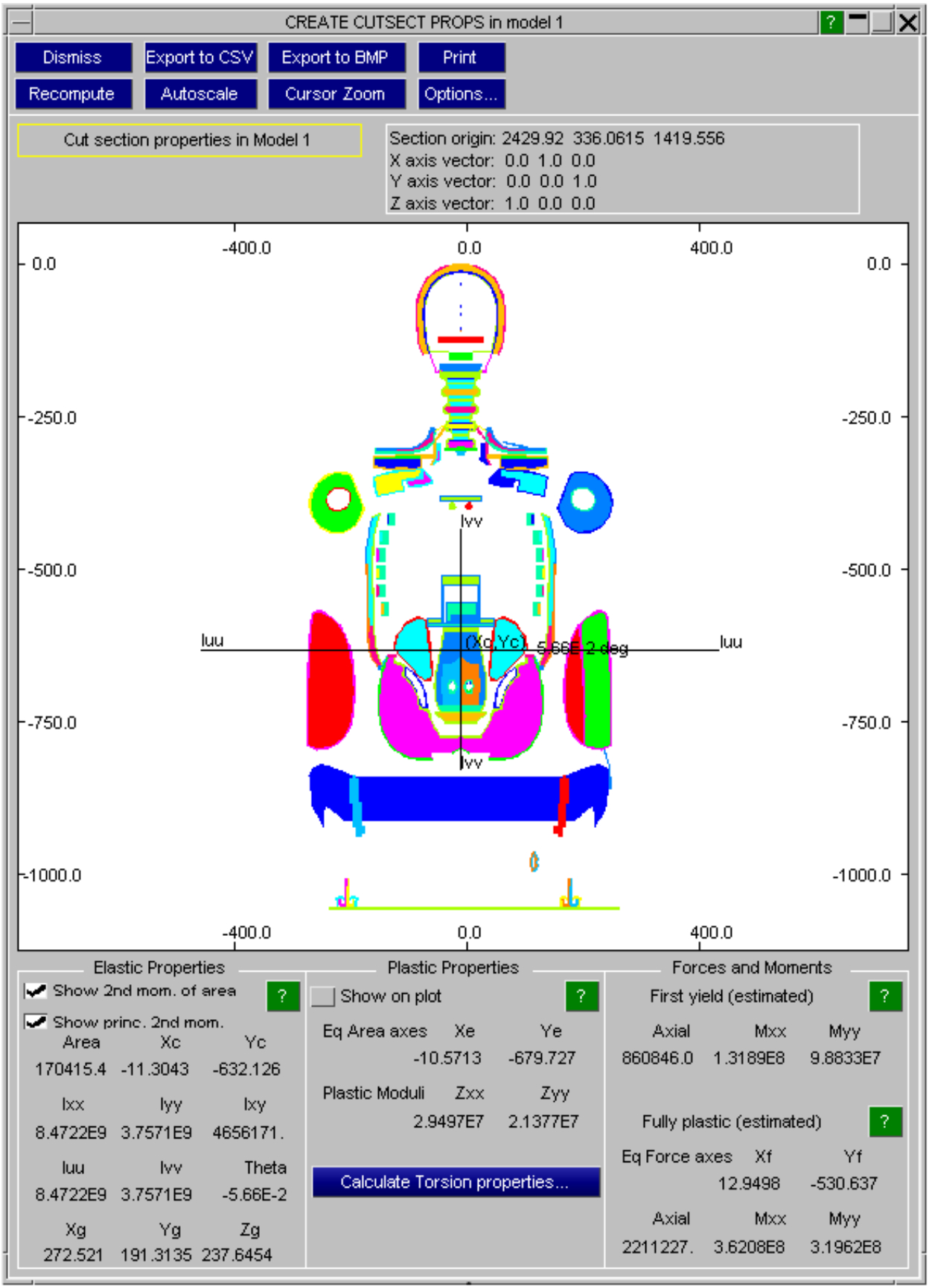
The contact penetration checker performs all these calculations for the specified contact surface, and if Settings, As Thick is chosen when displaying penetrations then the thickness of each segment will be shown. If cut sections are then turned on they will apply to these penetration plots, and in this way it will be possible to visualise penetration thicknesses.

6.13.11 **Properties**: Computing cut section properties

PRIMER is able to derive section properties (Area, 2nd moments of area, plastic moduli and section capacity) from the elements cut by the plane. Properties:

- Are derived from 3D (solid and thick shell), 2D (thin shell) and 1D (beam) elements only. All other types cut by the plane are ignored.
- Are calculated only from what is currently visible, so blanking and entity switches may be used to limit what is used in the calculation
- Use the local XY plane of the cut-section as their frame of reference.

An example of a cut through a dummy positionined on a seat is shown below



The method used to calculate properties.

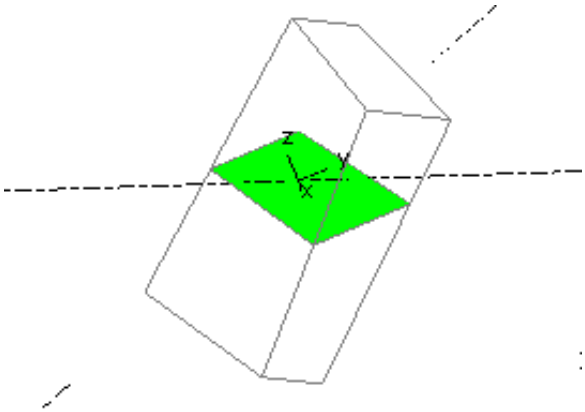
In general terms the cut area through each element is computed as a flat 2D polygon, or series of polygons in the case of complicated beam sections, and transformed into 2D cut section XY space. The origin of the cut section is implicitly the origin of this 2D space system, its X axis becomes left/right and its Y axis becomes up/down. Properties are then calculated in that XY space system using standard mathematical formulae.

The way that PRIMER calculates cut sections through the various element types is described below.

Cutting through 3D (solid and thick shell) elements

This is simple: the polygon generated where the plane intersects the element is used unconditionally. It will have between 3 and 6 sides depending on the location and orientation of the cut.

This example shows a plane cutting an 8 noded "brick" element at some oblique angle.

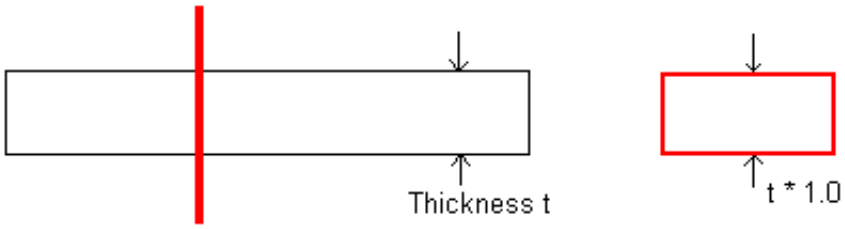


Cutting through 2D (thin shell) elements

Where the cutting plane intersects the shell cleanly at 90 degrees to its plane, the well-conditioned case, the cut plane through the element gives a good representation of the true element thickness.

However when the plane cuts the plane at an oblique angle as shown in the images below the calculation of the cut place depends on the "2D and 1D section cut"setting in the **Options** panel.

This has two possible settings



Plane cuts shell at 90 degree angle

This is the normal case of a well-conditioned cut in which the plane cuts at 90 degrees to the plane of the element. The cut face represents the section through the shell correctly.

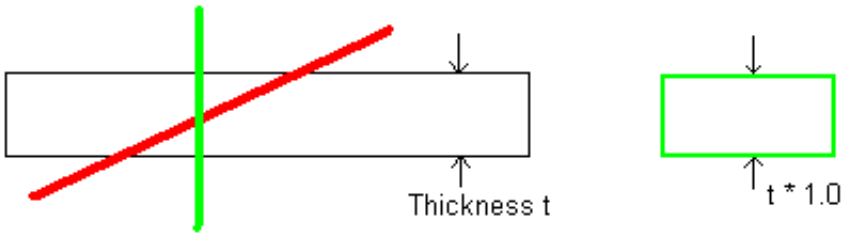
- **90 degree cut**, in which the cut surface calculated is always that of the well-conditioned orthogonal cut, which is the default because it is a "safe" option when computing section properties.
- **True cut**, in which the actual surface generated by the intersection with an oblique plane is computed. This is useful for visualisation of actual geometry, but will tend to over-estimate section properties.

These two options are illustrated below:

90 degree cut
(default)

In this case the effective cut plane through the element is always orthogonal, ie at 90 degrees to its plane, regardless of the actual angle at which the section cuts the element.

This is a "safe" method when calculating section properties, since otherwise a very oblique cut through a shell can result in a an area much deeper that its actual thickness which, in turn, could lead to the section properties being over-estimated.



Oblique case, 90 degree cut option

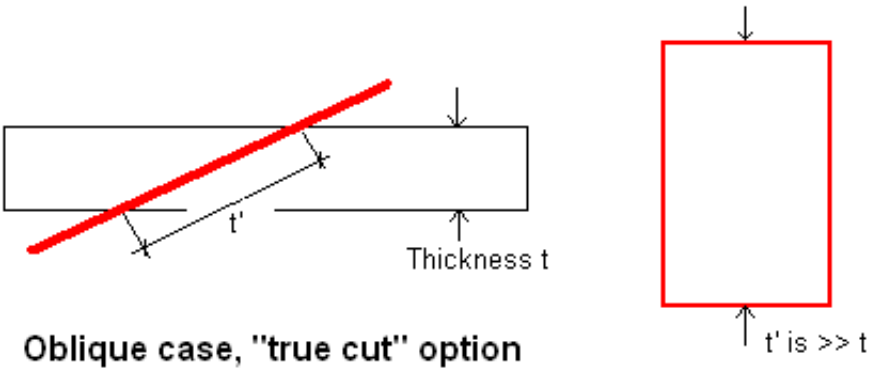
Even though the cut section (red line) cuts the element at a shallow angle the cut surface is computed at 90 degrees to the plane of the shell (green line).

This is a "safe" solution since it will not over-estimate section capacity.

"True" cut

In this case the true intersection between plane and element is computed, and for oblique cuts this can give apparent sections much deeper (t') than the element thickness (t) as shown here.

This can be useful when you wish to visualise the genuine cut geometry, but it is strongly deprecating if you are making use of the section properties calculated by PRIMER as it is likely to lead to their being an over-estimate of the true capacity.



Oblique case, "true cut" option

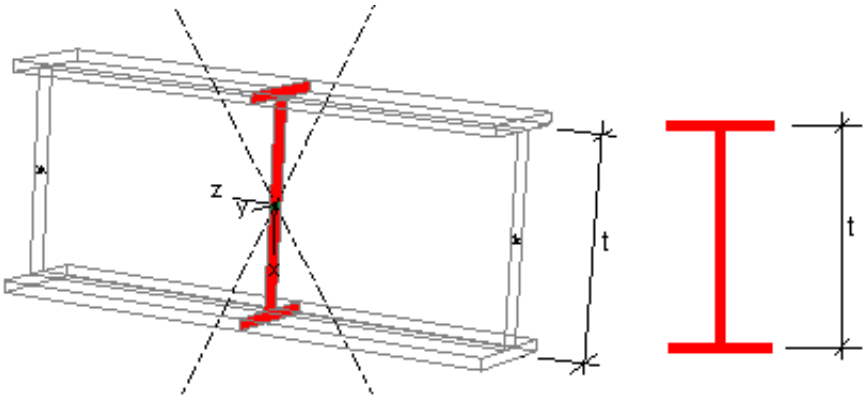
The actual cut surface is used, giving a "deeper" section as the cut becomes more and more shallow.

This can over-estimate section capacity, so it should not be used for calculating properties, but it is sometimes useful to be able to visualise the true intersection geometry.

Cutting through 1D (beam) elements

In order to calculate the intersection PRIMER expands beams to their "true" 3-dimensional section shape, and then cuts through that using the same "90 degree" or "true" cut rules that are used for shells.

(Note that by default PRIMER does not show true beam sections in the main graphics window, but this can be turned on using the **Display Options** panel. However the cut-section properties display window will always show the true beam shape, as this is required in order to compute properties correctly.)



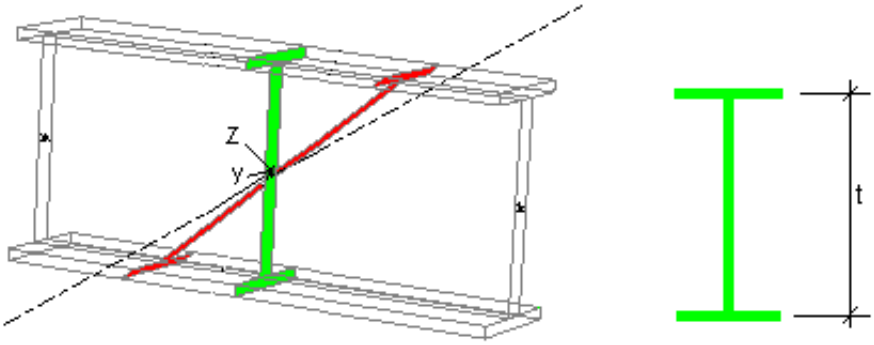
Plane cuts beam at 90 degrees

Intersection is the expected "I" shape of true section depth

90 degree cut option

Calculates the section at exactly 90 degrees through the element, regardless of the actual angle of intersection between beam and plane.

This is a "safe" option when calculating section properties, and hence the default.



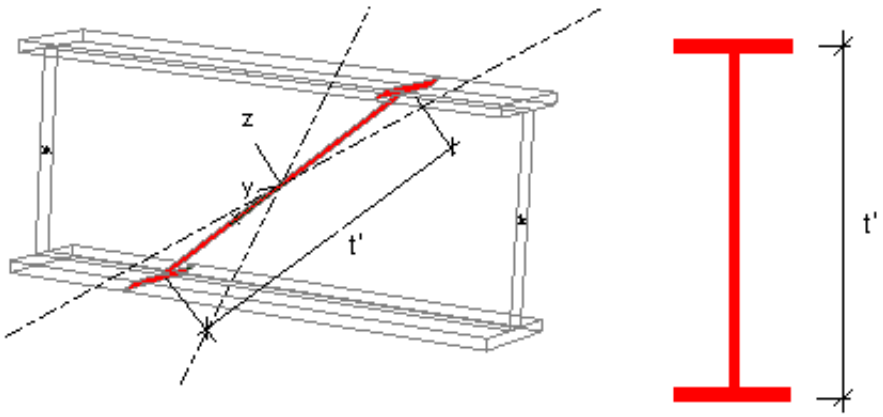
Plane cuts beam obliquely "90 degree cut" option used.

Again, the intersection is an "I" shape of the correct proportion and the actual section depth, regardless of the angle of the cut.

True cut option

Used the actual intersection between beam and plane, resulting in the section becoming taller as the cut angle gets shallower, as in this example.

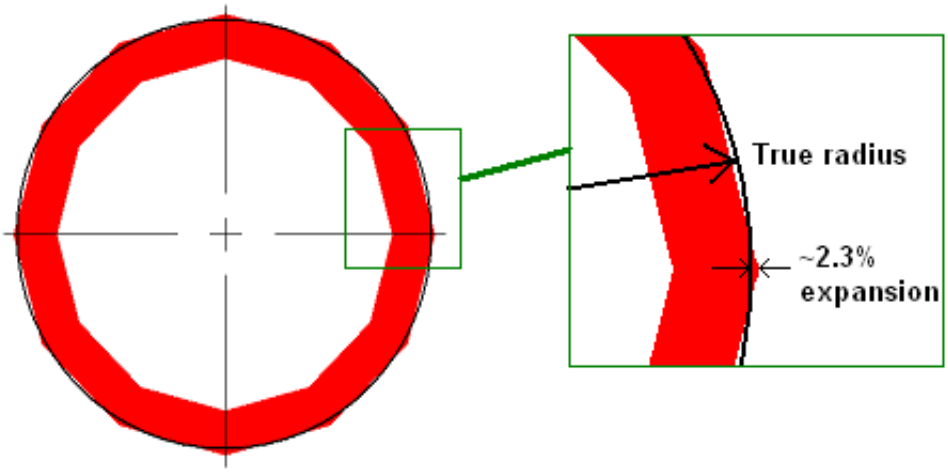
This is useful if you need to visualise the true geometry of the cut plane, but it should not be used when calculating section properties as it will over-estimate the section capacity.



Plane cuts beam obliquely "True cut" option used.

The section shape is now elongated and its depth (t') is greater than the actual section depth (t)

Special note on "true" sections used for circular beams



The "true" section shape used for solid circular and hollow tubular beams is an approximation that uses 12 facets, at 30 degree intervals around the circle, to represent the actual section. This is the same approximation as that used for "true" beam section graphics.

However to give approximately the correct Area and Inertia values in the cut-section property calculation the radii used to define the facets are approximately 2.3% greater than the actual radii in the true circular section.

This factor compensates for the fact that each facet is slightly smaller in area, and its centroid closer to the section centre, than the true section arc. It is based on the ratio of "true" area of a 30 degree arc vs that of the triangular slice subtended by 30 degrees. In the calculations to the right R = 1.0.

This solution is not precise, but it should be more than good enough for practical purposes.

$$Area\ of\ sector\ of\ arc = \frac{\pi . R^2}{12} = 0.2618$$

$$Area\ of\ 30^o\ triangle = R^2 . \cos(15) . \sin(15) = 0.250$$

$$Ratio\ of\ areas = \frac{0.2618}{0.250} = 1.0472$$

$$Ratio\ of\ radii = \sqrt{1.0472} = 1.0233 \sim 2.3\%$$

The torsion constant (J) of circular and rectangular beam sections is calculated analytically from the true section dimensions, and used directly in the torsion calculations below.

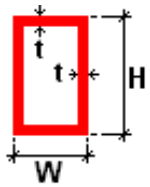
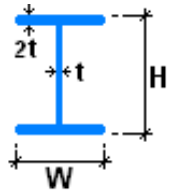
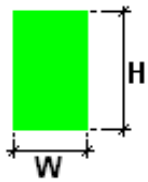
Special note on beams with explicit Area, Iss and Itt values.

Some beam element formulations, notably **`elform`** = 2 and 12, do not provide section dimensions but rather the engineering properties: **`Area, Iss, Itt, J, SA`** and **`Ist`**. These get special treatment in the cut section calculations.

Section shape for beams with explicit section properties

PRIMER attempts to synthesise a symmetrical section shape based on **A**, **Iss** and **Itt** (ignoring any torsion constant **J**, shear area **SA** or cross inertia **Ist**) and this shape is used both for "true" beam section graphics and for calculating derived cut-section properties.

The shape used will be one of:

A thin-walled rectangular hollow section (RHS)	Preferred shape, so long as the aspect ratio of width W to height H does not exceed 2 : 1 (or 1 : 2). All sides have a constant wall thickness t.	
An I beam	Alternative shape where the RHS shape would look stupid because of extreme W : H aspect ratios, generally because Irr and Itt are significantly different in magnitude. The web has a wall thickness t, and the flanges 2t.	
A solid rectangular section	Fall-back shape for cases where neither of the two sections above can be calculated from the properties supplied. (Properties supplied for a composite beam may not give a "physical" result for a homogeneous beam as assumed here.)	

There are no explicit solutions for calculating RHS or I beam shapes, so an iterative solution method is used. This usually gives a plausible shape, but it will not be an exact match for the original section shape from which the properties were derived. Also if a cross inertia term **Ist** is supplied this usually implies a cross section that is not symmetrical. This term is ignored in the section synthesis above since there is no way of knowing what shape to use, so if **Ist** is non-zero treat the derived shape with circumspection.

Elastic section calculations for beams with explicit section properties

Where these explicit properties are available and 90 degree cuts are in use then PRIMER uses the supplied **Area, Iss, Itt** and **Ist** for elastic properties directly in the Calculation of Elastic Section Properties, instead of calculating these values from the geometric shape.

If true cuts are in use then elastic properties are calculated from the geometric shape since it is assumed that the area-based properties of the skewed section are required.

Plastic section calculations for beams with explicit section properties

For the Calculation of Plastic Section Properties PRIMER always uses the geometric shape as synthesised above. This is necessary because these calculations require knowledge of actual shape and area, and you should be aware that such calculations - based on a synthesised shape - will only ever be approximate.

Torsion calculation for beams with explicit section properties.

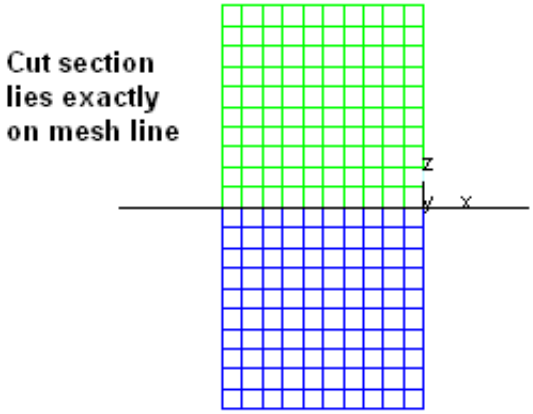
If Calculating Torsion Properties for a section using these beams then the **`J`** value (**`Irr`** in earlier LS-DYNA versions) supplied is used directly, and no attempt is made to derive a J value from the synthesised shape.

Not only is this much faster, but the J value is not considered when synthesising a shape; also there are no considerations of calculating plastic properties. This does mean that if J on the original **`*SECTION_BEAM`** card is zero then the beam will not contribute to the overall torsion constant - but then it would not contribute during an analysis either, so this is correct.

How ill-conditioned cuts are handled

It is normally the case that the cutting plane intersects elements cleanly, leaving no doubt about which elements are being cut. However where a mesh is rectilinear, and the cut plane is positioned exactly on a line of nodes, then a problem can arise as it is not clear whether the plane:

1. Lies in the gap between adjacent rows of elements, and doesn't cut anything
   or
2. Cuts elements both above and below the plane
   or
3. Only cuts elements on one side of the plane



Clearly (1) is not satisfactory, and (2) would be dangerous since it would tend to double the true properties, therefore (3) is adopted as follows:

- For each cut element a check for ill-conditioning, as in cut exactly at node (or nodes) location is made.
- Where this is found to be the case the cut plane is temporarily shifted by a very small amount in its +ve Z (outward normal) direction, and the cut calculation is repeated.

The effect of this is always to consider the element on the +ve Z side (in plane local axes) of such a cut, and to ignore the element on the -ve Z side. In the image here this means that the light green elements would be considered, and the dark blue ones ignored.

This solution is safe, but it may not always give the result that you want. The best solution is to avoid this problem by moving the plane so that it cuts near to the centre of elements, rather than at mesh lines, since in this way you control explicitly which elements are cut.

Calculation of Elastic Section Properties

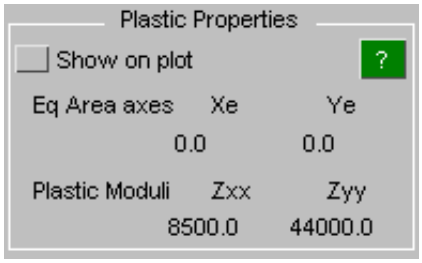
Note that X and Y axes here are the cut-section local (X,Y) plane, and the centroid position is given relative to the origin of the plane.

The following engineering properties are calculated:

Total **`Area`**	The sum of all cut-section polygon areas.
Geometric centroid (**`Xc,Yc`**)	The result of the 1st moment of area about X and Y axes, divided by the **`area`**
Geometric centroid in global coordinates (**Xg, Yg, Zg**)	Displays the geometric centroid in the global coordinate system.
2nd moments of area **`Ixx, Iyy, Ixy`**	In each case the sum of the local I value for each polygon, + its area * distance squared from the relevant axis.
Principal 2nd moments of area **`Iuu`** (max) and **`Ivv`** (min), and the angle **`theta`** between **`Iuu`** and **`Ixx`**	The result of transforming the tensor [Ixx, Iyy, Ixy] using Mohr's circle to give principal values and the angle between **`Iuu`** and **`Ixx`**.

Calculation of Plastic Section Properties

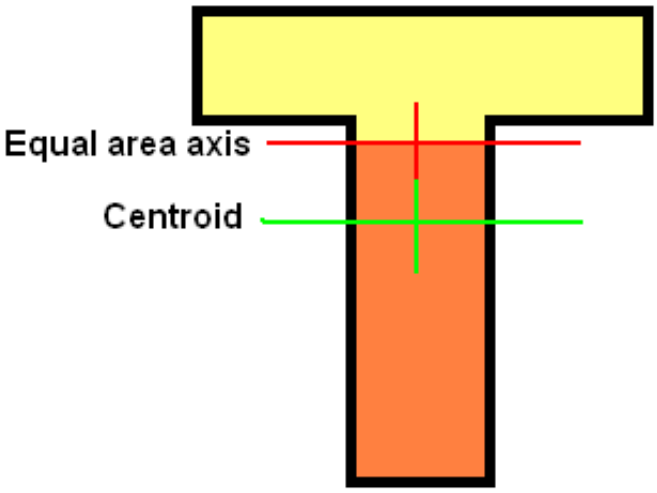
The following plastic properties, commonly used by structural engineers, are calculated:



Equal area axes (**`Xe,Ye`**).

These are the axes which give equal areas about X and Y respectively.

For an unsymmetrical section such as the Tee shape here the equal areas axis will not lie on the centroid in the depth axis. It is located where the areas above and below it, drawn here in yellow and orange, are equal.



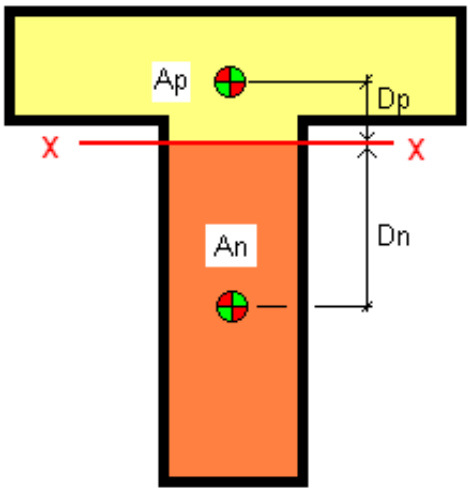
Plastic moduli (**`Zxx,Zyy`**) [sometimes **`Sxx,Syy`** in North America]

These are the sum of "area * distance from centroid of area to equal areas axis" about X and Y respectively.

This example shows the computation of the **`Zxx`** value (**`XX`** is the horizontal equal area axis here) for the Tee section above.

The plastic modulus **`Zxx`** about equal area axis **`XX`** is given by:

```
Ap * Dp + An * Dn
```

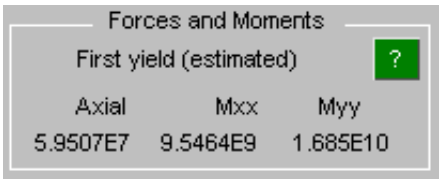


Where:

Ap = Area on +ve side of XX axis (yellow) } these areas are
An = Area on -ve side of XX axis (orange) } equal in value
Dp = Distance from centroid of area **Ap** to **XX**
Dn = Distance from centroid of area **An** to **XX**

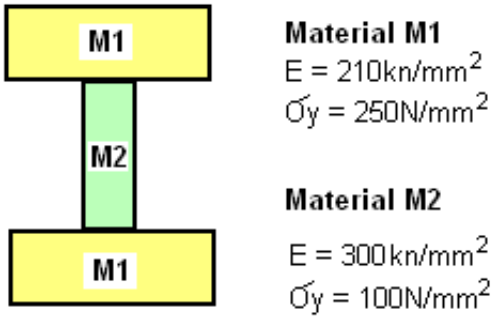
If all elements in the section have the same yield stress then multiplying the modulus by the yield stress gives a crude approximation of the fully plastic bending capacity of the section. See also the Force and Moment calculations below which may give more useful results for a typical section made up of multiple material types.

Calculation of "First Yield" capacity of the section



This gives the axial force and bending moments at which the first part of the section to reach yield stress is exactly at that stress, taking into account all the different material and element types that comprise the section.

To illustrate this consider the following imaginary section with two different material properties, M1(yellow) and M2 (green), that have different Young's Modulus (E) values and yield stresses (σy).



In all cases we use the elastic properties (area, Ixx, Iyy) and make the assumption that plane sections remain plane, ie that the distribution of strain through the section depth is linear. In the following images the distribution of strain through the section is shown, annotated with the corresponding stresses.

Please read the warnings below before using these values.

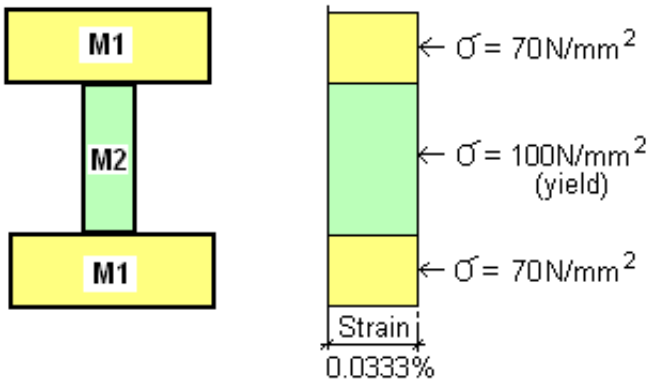
Axial case.

Here the strain in the section is constant through its depth, and first yield occurs when the the weaker material M2 reaches yield at 0.0333% strain.

M1 has an E value of 210kN/mm2, so at 0.0333% strain its stress is 70N/mm2

The axial force capacity of the section is then simply the sum of stress * area:

```
axial force = (area of M1 * 70)
            + (area of M2 * 100)
```



Axial Case

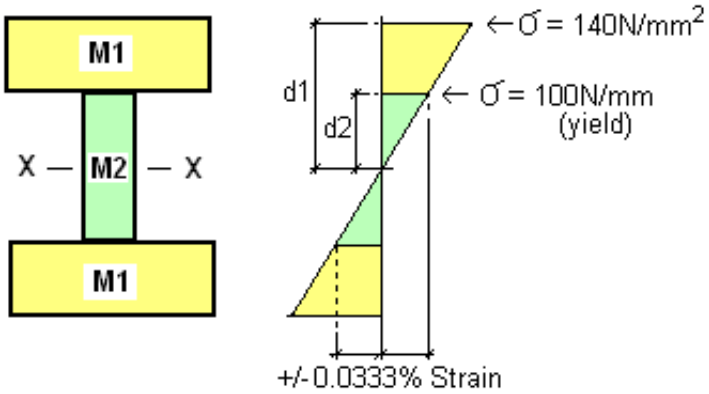
Mxx Bending case

Here the limiting strain of 0.0333% is still controlled by the outer fibre of material M2 at depth d2 from the XX axis, which reaches yield first.

However the linear strain distribution through the section means that the outer fibre of material M1 reaches about 0.0666% strain (assuming d1 = 2 x d2), so the peak stress in material M1 is about 140N/mm2 .

Therefore the bending moment capacity of the section about its XX axis, **`Mxx`**, is, obtained from the standard formula **`M/I = stress/y`** giving:

```
Mxx = (Ixx for M1 * 140 / d1)
    + (Ixx for M2 * 100 / d2)
```

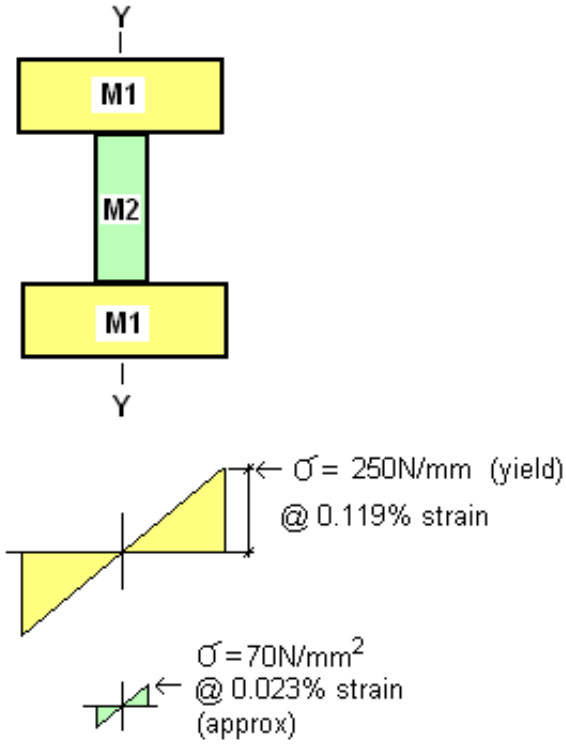


Mxx Bending Case

Myy Bending case

This is calculated using the same method as **`Mxx`**, but now the narrowness of the web means that first yield is reached at the outer fibres of material M1 at a strain of 0.119%.

The strain in the outer fibre of the web, material M2, will be about 20% of this, approximately 0.023%, giving an outer fibre stress in M2 of about 70N/mm2.



Myy Bending Case

Warnings about the "First Yield" calculation

1. It will be clear from the calculations above that both Young's Modulus (E) and yield stress (σy) must be available for every material in the section if this calculation is to be valid. These values are obtained from the material (*MAT) cards, but for some material types - especially brittle and crushable ones - either or both may not be well defined. Also it is possible for that the relevant material cards may not be present in the deck.

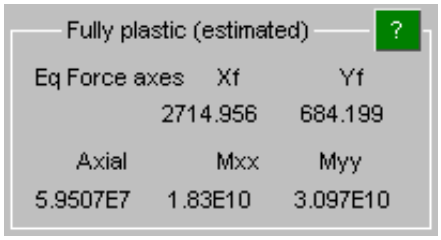
   Please see the **Options** panel below to see the alternatives that PRIMER uses when either of these properties are missing. Regardless of the choices you make there if any element in the section cannot extract either E or yield stress from the material card then the results will be marked as "Estimated".

2. This is a purely elastic calculation and, in the case of Mxx and Myy, assumes that all materials in the section behave symmetrically in tension and compression in the elastic regime, and have the same yield stress in both tension and compression. For ductile materials, eg steel, this is likely to be true; however for brittle materials (eg concrete) yield in tension may be very different to yield in compression.

3. It is assumed that each cut element is homogeneous with a single E value and yield stress σy. This will not be the case for composites, made up of layers of different materials; nor may it be a valid assumption for orthotropic materials.

Please consider the sections being cut through when you use this feature, and satisfy yourself that the calculation is valid for your model.

Calculation of Fully Plastic capacity of the section

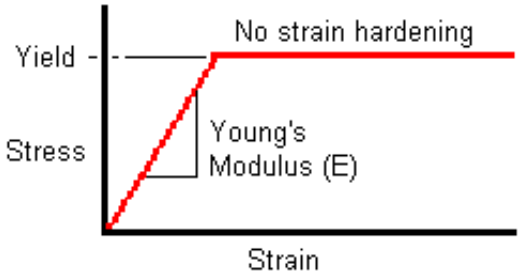


These values calculate the axial force and bending moment capacity of the section assuming that all materials are at their yield stress σy.

This is very similar to the calculation of plastic moduli, except that the actual yield stresses of all materials in the section are used, making it possible to estimate the plastic capacity of a section comprising multiple different material types.

Instead of calculating "equal area" axes PRIMER now calculates "equal force" axes in which the force (the sum of area * yield stress for all cut elements) is equal on both sides of the axis.

All materials are assumed to behave in an "elastic / perfectly plastic" stress strain curve, symmetrical in tension and compression. Since plane sections must remain plane it is necessary that each material be able to maintain a constant yield stress over a wide range of strain values, hence the requirement for no strain hardening.



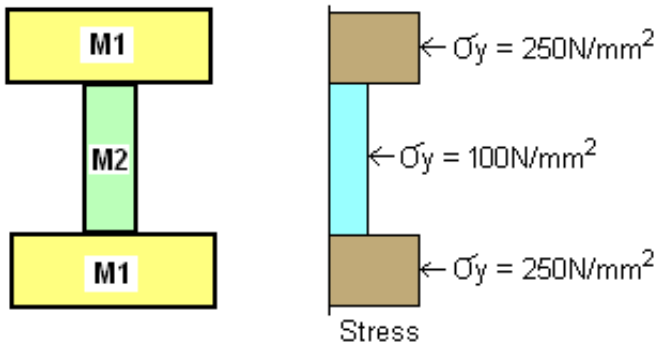
Elastic / perfectly plastic Stress / strain curve

The illustrations of the plastic capacity below use the same section as used in the "First yield" examples above, but note that the images now show the distribution of ***stress*** through the section depth (rather than strain above).

Fully Plastic axial capacity

Each material is at yield, so the axial force capacity of the section is then simply the sum of yield stress * area:

```
axial force = (area of M1 * 250)
            + (area of M2 * 100)
```

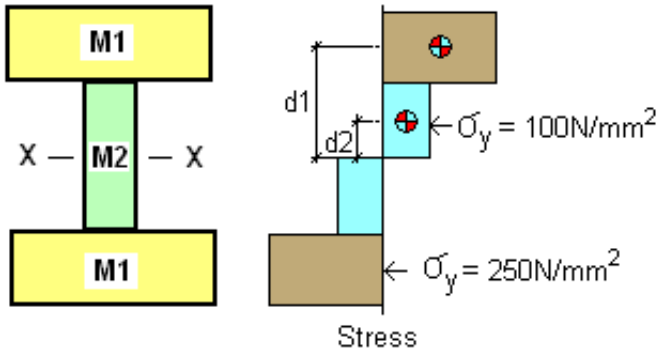


Axial case: Fully plastic

Fully plastic Mxx bending capacity

Again each material is at +/- yield stress, so the total bending capacity ois given by the sum of (area * yield stress * distance from centroid to XX axis) for all cut sections. So in this example

```
Mxx = (Area M1 * 250 * d1)
    + (Area M2 * 100 * d2)
```

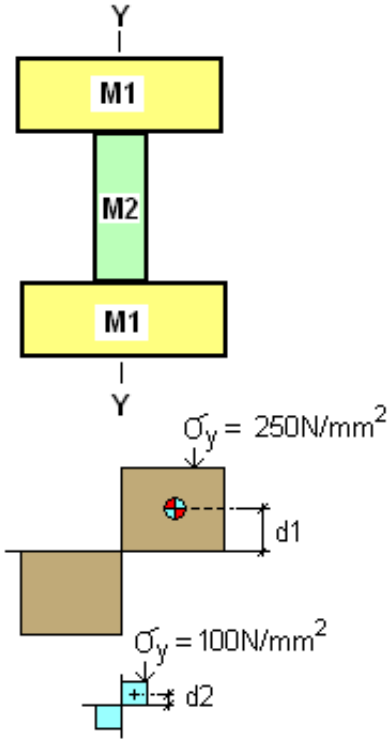


Mxx Bending Case: fully plastic

Fully plastic Myy bending capacity

Essentially the same calculation as Mxx. Each material is at +/- yield stress, so

```
Myy = (Area M1 * 250 * d1)
    + (Area M2 * 100 * d2)
```



Myy Bending Case: fully plastic

Warnings about the "Fully Plastic" capacity calculation

1. These calculations require the yield stress σy of every material in the cross section. These values are obtained from the material (*MAT) cards but for some materials, for example crushable or brittle ones, a yield stress may not be well defined. In this situation PRIMER has various options for determining a yield stress, see the **Options** panel below.

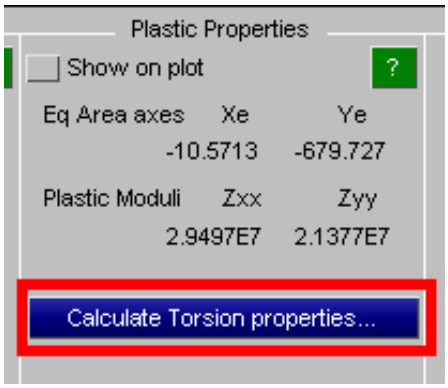
2. Fully plastic calculations like this are inherently unrealistic since the vast majority of real materials do not exhibit "elastic / perfectly plastic" behaviour. Moreover LS-DYNA material models tend to define quite complex post-yield stress/strain characteristics, all of which are ignored here, so these values should not be considered to be anything more than a crude estimate of plastic capacity.

3. In the case of bending (Mxx and Myy) these calculations assume symmetrical yield behaviour, with the same yield stress in tension and compression. This may be reasonable for ductile materials (eg steel) but can be hopelessly wrong for brittle ones (eg concrete).

4. It is assumed that each cut element is homogeneous with a single yield stress. This will not be the case for composites, made up of layers of different materials; nor may it be a valid assumption for orthotropic materials.

Please consider the sections being cut through when you use this feature, and satisfy yourself that the calculation is valid for your model.

Calculating Torsion Properties

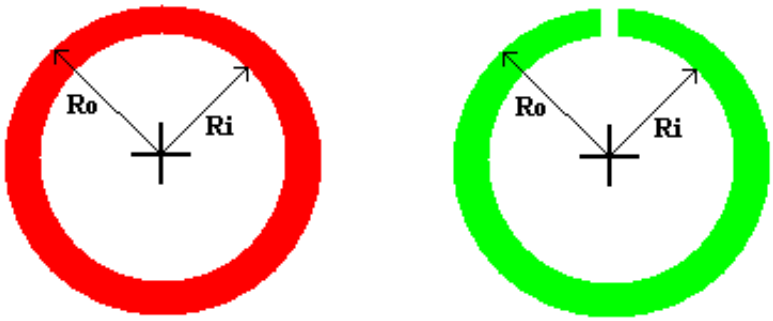


The torsion constant J of the section is not calculated by default since this a complicated and - sometimes - slow process, and for complex sections there can me more than one way of interpreting the answer.

If you want to compute J values use **Calculate Torsion properties...** to switch to the torsion calculation mode, which will show this alternative panel base:

Why a separate panel?

Generally an analyst wants a simple J value for his section, but unfortunately life is not that simple. Unlike the more common section properties such as Area and 2nd moment of inertia the torsion constant depends upon both section shape and - crucially - connectivity. Consider the following examples of open and closed tube sections:



Closed case **Open Case**

$$J = \frac{Pi \cdot (Ro^4 - Ri^4)}{2} \qquad J \approx \frac{(Ro - Ri)^3}{3} \times 2\,Pi \cdot Rav$$

Despite having very similar shapes and almost the same amount of material the J value of the closed (red) section is many times greater than that of the open (green) section, and the calculation process must reflect this. A calculation based on shape alone will not do, it must also consider connectivity.

Why split the calculation up by element types?

The calculation method that works well for a multi-cellular structure made with thin shells, which treats the shear stress distribution across cell walls as constant, is wholly inappropriate for sections through "thick" structure made of solids and thick shells, where there will be a variation of shear stress.

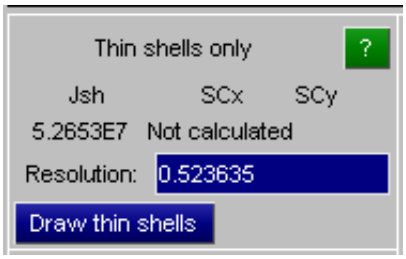
Calculating beam properties is yet another problem since each beam's shape, derived from its section properties, must be considered.

Finally in a composite section that contains solids and/or thin shells and/or beams is it true to say that each element type is working separately? Or are they all working together in a composite form?

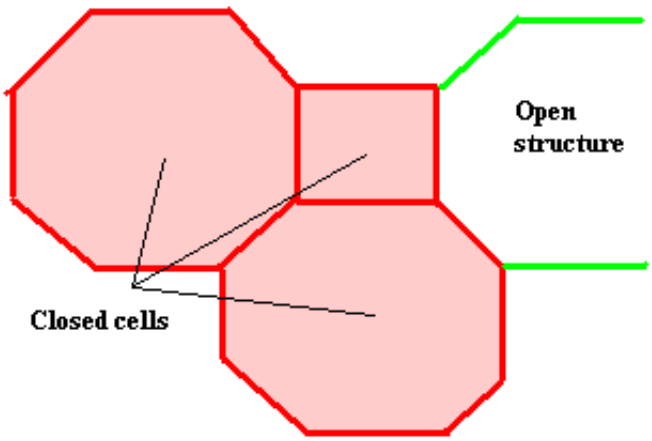
This is why each class of elements is treated separately, and two sorts of combined results are presented. Each class of elements will be described separately below, and then the different ways of combining results will be discussed.

Jsh: Thin shells only

Calculation of open and closed-cell structures modelled with thin shells, using a "thin" calculation method suited to these elements.



For thin shells the structure is broken down into "closed cells" and "open structure".

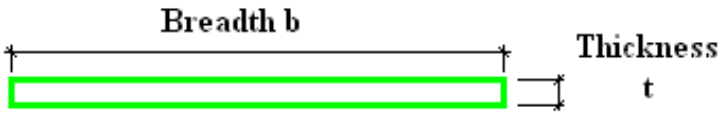


Thin shell torsion calculation method

Difference between closed cells and open structure

The torsion constant for the **closed** cells (red) is calculated by solving a series of linear equations based on shear flow in the walls and the constraint that the angular twist of the whole section must be constant. This approach assumes that the distribution of shear stress in the cell walls is constant, which is a reasonable assumption for "thin" elements - hence its use for thin shell elements.

The torsion constant for the **open** structure (green) is the sum of the individual torsion constants of each separate shell, treating each one as an independent thin flat plate of breadth b and thickness t as follows:



$$J = \frac{b \cdot t^3}{3}$$

Torsion constant of a thin flat plate

For an arbitrary structure of multiple thin flat plates

$$J = \sum \frac{b \cdot t^3}{3}$$

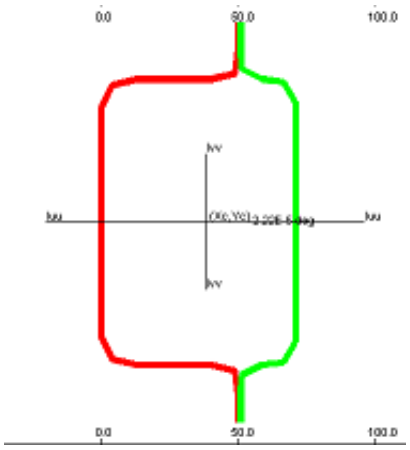
Thus the total **Jsh** value is `Sum (Jclosed) + Sum (Jopen)`

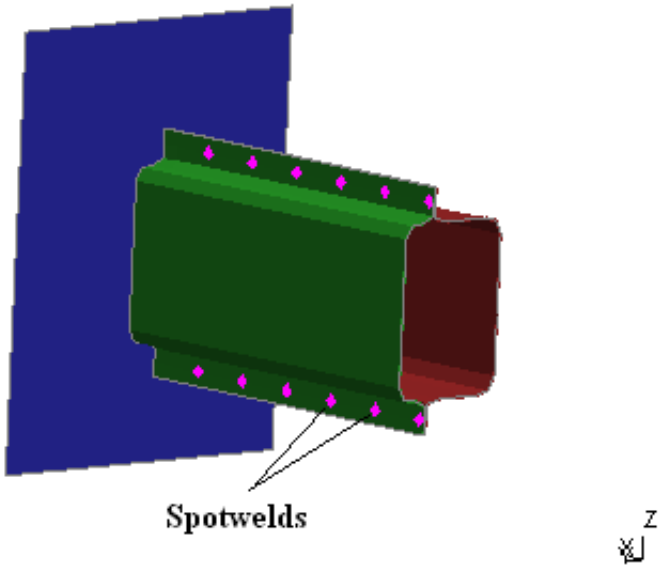
Draw thin shells - checking that the diagnosis of closed cells is correct.

The determination of what is in a closed cell is ***not*** based on the connectivity of the original mesh, rather it is performed by projecting the cut section onto a raster grid and determining connectivity from adjacent cells being occupied. It is done this way, rather than by mesh connectivity, since many "real world" structures connect materials by spotwelds, contacts or other non-topological means. Consider the following example of a crush tube:

In this example a platen (blue) will crush a tube made of two separate "U" sections (red and green) that are spotwelded together (purple). The "U" sections form separate meshes, the only connectivity between them being the welds.

A section has been cut through the tube giving the cut section shown below:

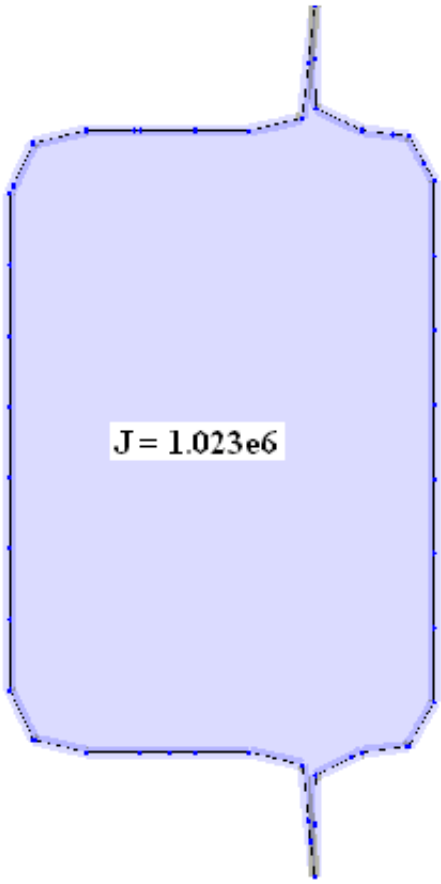


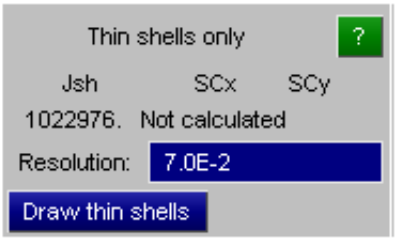


Calculating the torsion properties and then selecting **Draw Thin Shells** gives the following results. The fact that the difference in J values between "closed" and "open" solutions is a factor of 1000x illustrates the importance of making sure that the structure's connectivity has been interpreted correctly.

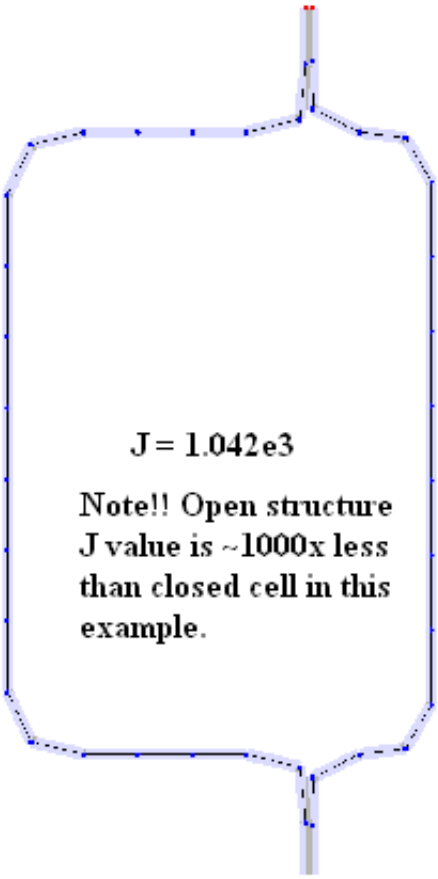
On the left hand side the two halves of the structure are close enough to be treated as "connected", and the structure as displayed by Draw Thin shells is filled in confirming that it is a closed cell. Its J values is ~ 1e6.

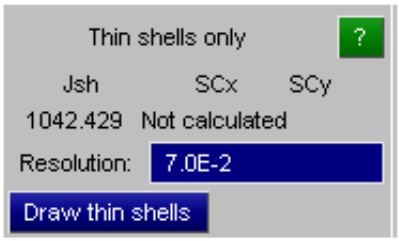
On the right hand side the original geometry has been moved apart so that the two halves are not longer close enough to be "connected", and the structure is no longer treated as a closed cell, but rather as "open structure". The J value is three orders of magnitude less at ~ 1e3





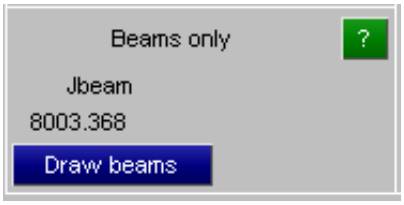
Structure halves close enough to be treated as a closed cell.





Structure halves separated, so treated as all "open structure"

JBeam Beam elements only



Beam elements are considered to act independently in torsion, so that the value **Jbeam** is the sum of the J values of each individual beam.

This section describes how the J values are calculated for beams.

LS-DYNA provides several different ways of defining beam section properties, and these affect how the J value is calculated.

Definition method	How J value is calculated here.
By supplying A, Ixx, Iyy and - for some section type - a J value	The explicit J value is used directly if it is non-zero, otherwise the method below is used.
By supplying dimensions for rectangular or circular shapes	The section is represented as a series of polygons, as used for graphics, and the "thick" calculation method is used to derive a J value.
By defining dimensions for one of a library of standard shapes.	

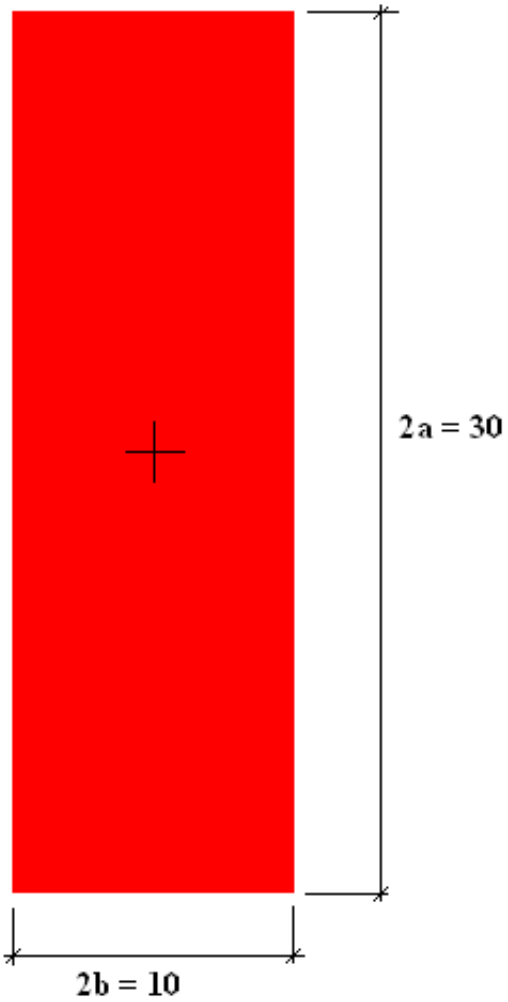
Some examples of using the "thick"calculation method to derive values are given below.

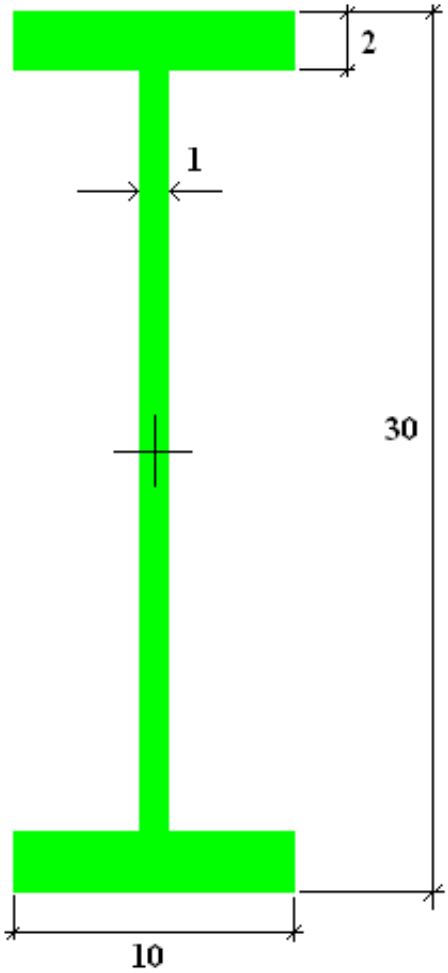
The left hand side shows a solid rectangular section for which a published solution for J is available.

Correlation between published formula and calculated solution is better than 3%

The right hand side shows an imaginary I section, and correlation between the value based on the sum of flanges and web treated as thin plates and the calculated value is reasonable.

As noted below some texts would apply a multiplier > 1.0 to the "sum of thin plates" solution for this section. It is difficult to say what the right answer is.



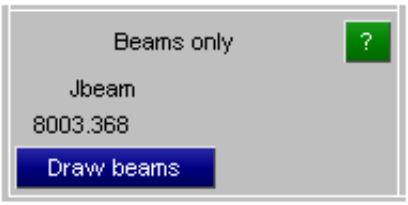


The analytical value of J for this section is

$$J = a.b^3\left[\frac{16}{3} - 3.36\,\frac{b}{a}\left(1 - \frac{b^4}{12a^4}\right)\right]$$

Which evaluates to 7.902e3

PRIMER obtains 8.003e3 using the "thick" analytical method

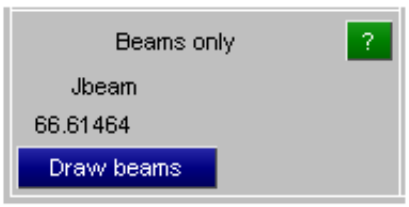


Using $J = \sum \frac{b\,.\,t^3}{3}$

for flanges and web gives J =62.0

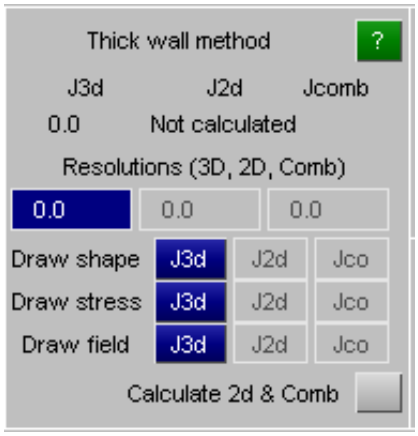
(Some texts would apply a factor n to this equation for an I beam, with n a bit greater than 1.0)

PRIMER obtains 66.61 using the "thick" analytical method



The J value for each beam is calculated separately, then **Jbeam** is the sum of J for all beams in the section.

J3d solid and thick shell elements



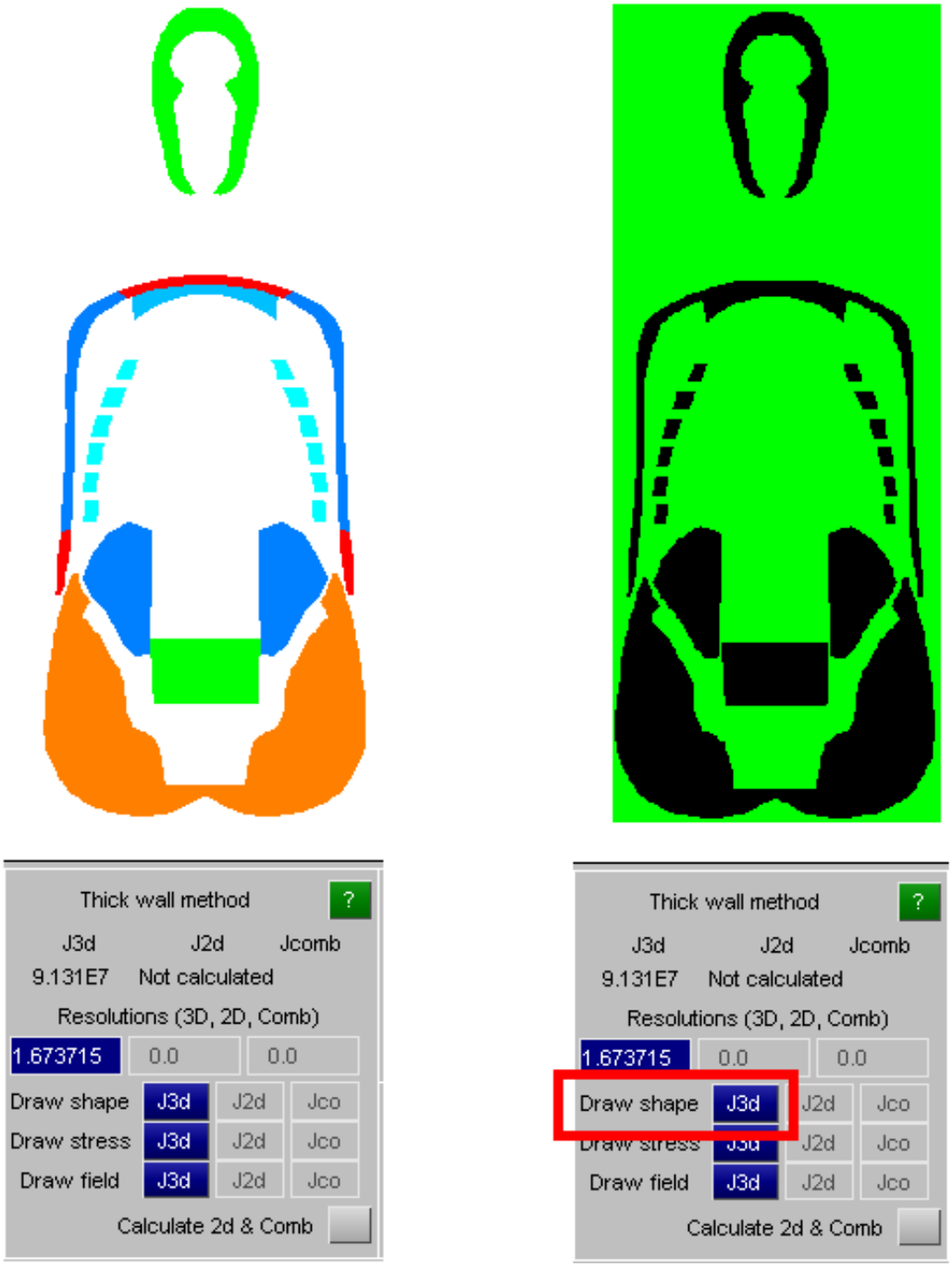
J values for 3d solids and thick shells are calculated via a "thick walled" method that evaluates the Prandtl torsion function.

Also **J2d** and **Jcomb** which, optionally, apply this "thick" method to both 2d thin shell structure and the combination of 2d and 3d elements if both are present in the section. These are greyed out by default and their use is discussed later.

Example of cut section through an arbitrary thick walled structure made from solids and thick shells. In this example the torsion constant has been evaluated as 9.1e7 using the default resolution

This first image shows just the shape of the section, which is a simplified cut through a dummy model showing head, torso, pelvis, ribs and some internal structure.

This plot is the "shape" which shows how the structure has been rasterised onto a grid for calculation. Check this to make sure that no structure has been omitted or mis-diagnosed.



Two further plotting modes are provided: **Draw stress** and **Draw field**. These are really just for debugging purposes and can be ignored.

Resolution: controlling the raster resolution

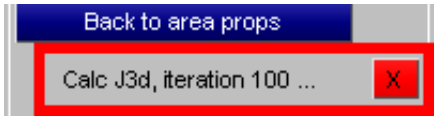
For "thick" torsion calculations the default resolution is based on the smaller dimension of the bounding box round the section (ie the green rectangle above right) divided by 500. In this example the default resolution is 1.6737.

If you feel, for example from the "shape" plot, that some structure is not being rasterised correctly then you can make the resolution finer (smaller). This might be necessary to resolve narrow gaps between elements, or perhaps if some narrow regions are present. As a rule of thumb a "narrow" section should have at least 5 cells across its width, and 10 would be better.

However for most genuinely "thick" examples the default resolution is satisfactory: changing the resolution in this example from 1.6737 to 1.0 changed the J value from 9.131e7 to 8.908e7.

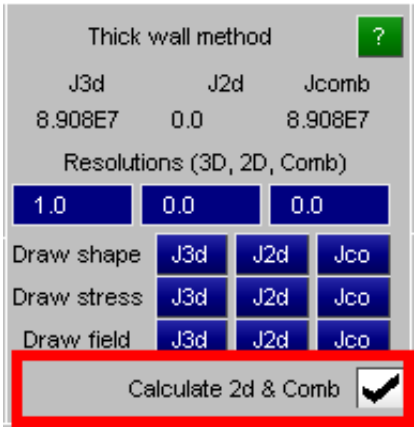
WARNING: The calculation of the Prandtl torsion field is an interative procedure using finite differences. Calculation time is a function of resolution ***cubed***, so making the resolution unnecessarily small will result in very long run times.

Halting the calculation if it is taking a long time



The progress of the torsion calculation is reported at the bottom right hand corner, and if it seems to be taking a long time you can halt it prematurely with the red [X] button. A result will be reported but you should regard it with some suspicion.

Calculate 2d & Comb



Combining "thin" and "thick" results using the "thick" calculation method.

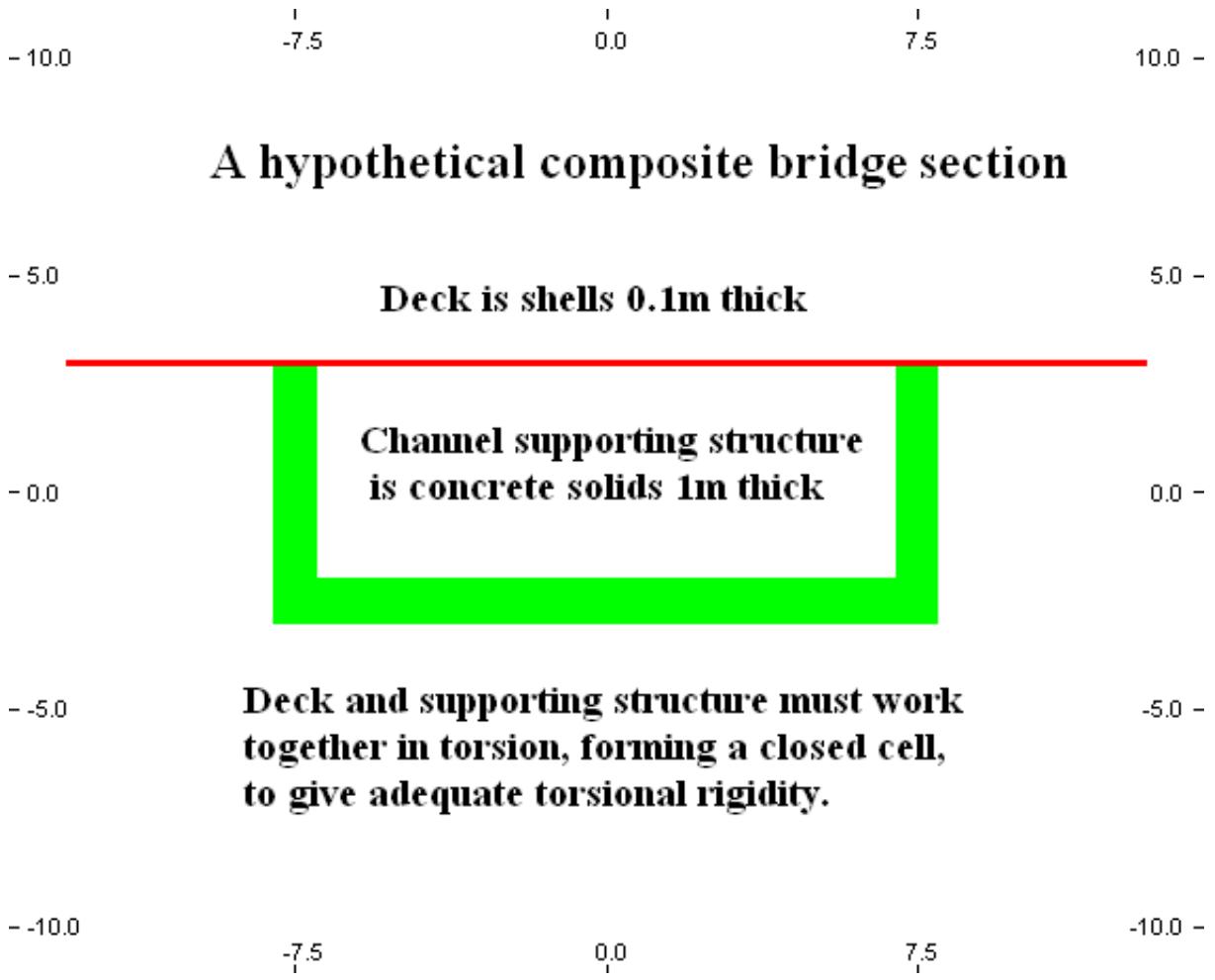
Useful when shell and solid structure is working together in torsion.

You may have the situation that a structure combines both 2d thin shells and 3d solids, and that for the purposes of torsion these have to be considered as acting together. In this case calculating **Jsh** and **J3d** in isolation and then adding the results may result in an understimate of the torsion capacity of the structure, and it becomes necessary to combine the two elements types in a single calculation.

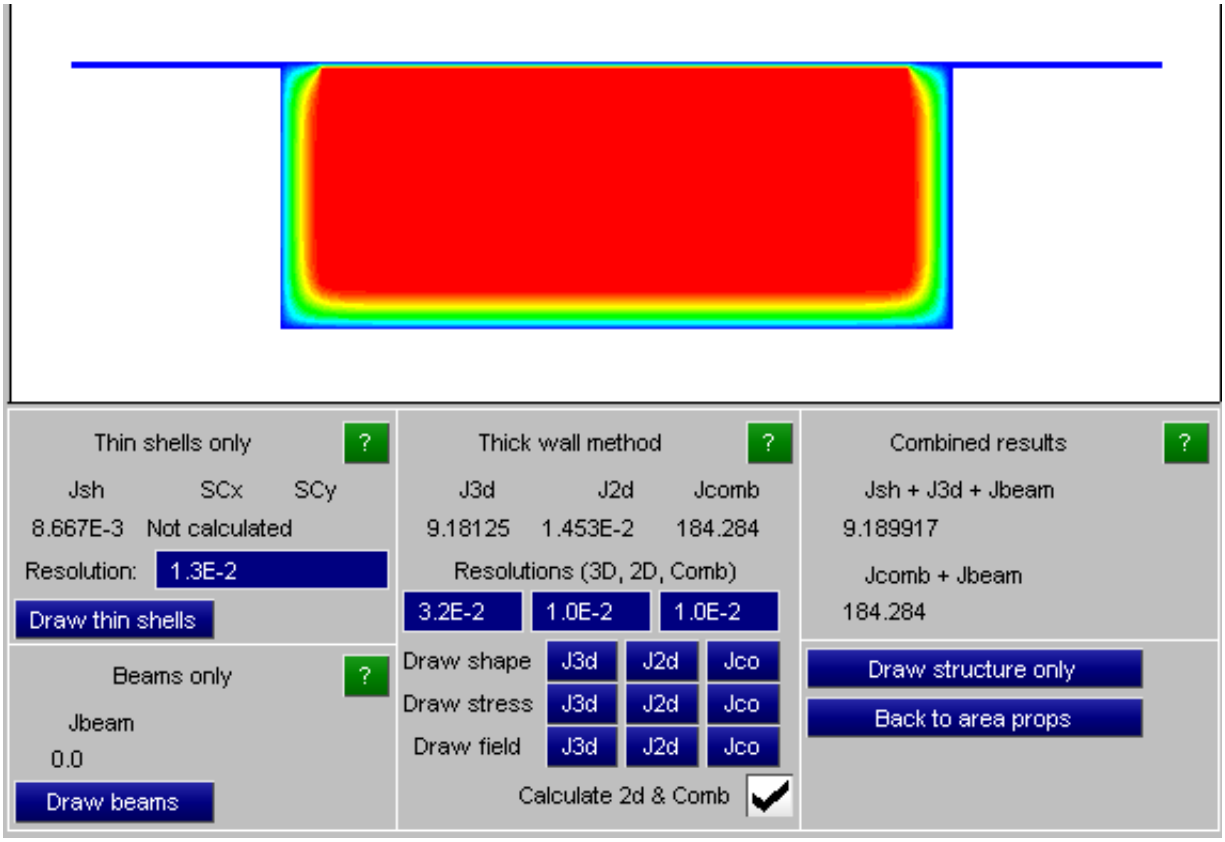
Consider the following (artificial) example of a bridge section in which

- the deck is modelled with shells 0.1m thick (red)
  and
- the channel structure below is modelled in solids 1m thick (green).

Clearly the torsional capacity of the combination of deck and channel working together to form a closed cell is many times greater than that of the individual parts, as the images below will show.



Below is a **Jco** plot of the Prandtl field for the combined case, demonstrating that the two structural components are working together. The various results and their meaning are explained in the table underneath.



Understanding the results from the combined image above.

Solid element results	J value	Explanation
J3d (under thick wall method)	9.18125	This is the "thick" calculation of the solid elements only, and represents the torsional capacity of the open "U" shaped channel that they form.
Thin shell results	J value	Explanation
Jsh (under thin shells only)	8.667e-3	This is the "thin" calculation of the top deck shells in isolation, based on 1/3.b.t**3
J2d (under thick wall method)	1.453e-2	This is the "thick" calculation of the top deck shells in isolation. It acts as a check on the accuracy of the "thick" calculation as applied to this "thin" structure. If the value of **J2d** is very different to **Jsh**, and this difference is deemed to be important, then it will be necessary to refine the resolution of the **J2d** calculation. **J2d** is likely to be less accurate than **Jsh**, and should not be used as an independent property. It is shown here to give a measure of the accuracy - or otherwise - of the calculation method.
Beam results	J value	Explanation
Jbeam (under beams only)	0.0	There are no beams in this model, so the result is zero. However had there been any beams the J value for each one would have been calculated separately, then **Jbeam** would be the simple some of these individual J values.
Combined results	J value	Explanation

Jcomb (under thick wall method)	184.284	This is the result of running the combination of thin shells and solid elements through the thick wall calculation, resulting in the two forming a closed cell. It is clear from inspection that the result of this calculation is over 20x greater than that of the individual components acting alone, and this demonstrates clearly the importance of getting the connectivity of the structure right for this calculation.
Jsh + J3d + Jbeam (Under Combined results)	9.1899	This is the linear sum of the individual thin shell, solid and beam torsional constants **calculated in isolation**. Therefore it does **not** consider the possibility of these element types acting together.
Jcomb + Jbeam (Under Combined results)	184.284	This is the linear sum of the **combined** solid and shell **Jcomb** value plus the value for beams, and is the result of the structure "working together". (Beams are always considered to act independently in torsion.)

Adjusting the resolutions of the combined calculation process

If the combined calculation method has been enabled all three resolutions will be enabled for editing. From left to right they are

- That for the solids only J3d value
- That for the shells only "thick method" J2d value
- That for the combined Jcomb value

The default values are chosen to give a reasonable compromise between accuracy and speed (recall that the time taken for a "thick" calculation rises as a function of resolution cubed!). You will normally only have to adjust these values if:

- The calculation method fails. This will generate a warning message explaining why, and the relevant result will be reported as "Failed". This is normally due to selecting too coarse a resolution resulting in a failure to project the section correctly onto the raster grid. Choosing a finer value will normally cure the problem.

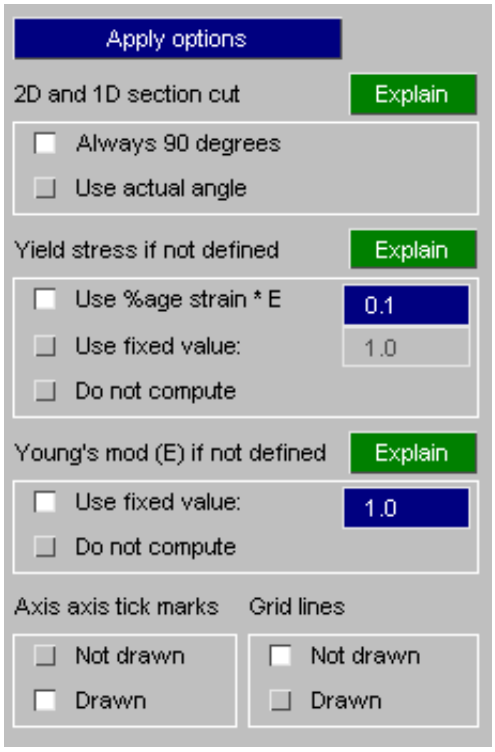
- Examination of the "shape" (or other) plots reveals that the section is not working as intended. Again, selecting a finer resolution will usually solve the problem.

- In the combined case the thin shell **Jsh** and **J2d** values are significantly different, and thin shells contribute a significant amount of the torsional stiffness. In this case it will be necessary to refine the **J2d** resolution until a reasonable result is obtained, and then to apply this result to the **Jcomb** case as well so that the combined value is calculated accurately.

Options

Controlling calculation and plotting.

2D and 1D section cut



Controls how 2D (shell) and 1D (beam) sections are generated when the angle between element and cutting plane is not orthogonal.

- **Always 90 degrees** gives a cross-section that cuts the element at right angles to its in-plane axis, giving a "safe" shape for calculating section properties.
- **Use actual angle** calculates the actual area cut through the element, which can be useful if you need to visualise this correctly. However this is not recommended for section property calculations as it can result in over-estimates of area and hence capacity.

The effects of these two options on shells and beams is illustrated above for shells and beams.

Yield stress (σy) if not defined

When calculating both First Yield and Fully Plastic section capacities PRIMER needs to know the yield stress of all cut element materials, and it will normally extract this from their material (*MAT) cards.

However that card might not be present, or for some more exotic material types it may not be possible to calculate a yield stress, in which case some other value must be used, and the following three options are provided:

Use %age strain * E	Calculates a yield stress by multiplying the Young's Modulus (E) by a %age strain value.
Use fixed value	You define a yield stress to be used if a value cannot be found.
Do not compute	Elements for which a yield stress cannot be found are omitted from the calculation.

Young's Modulus (E) if not defined

When calculating First Yield section capacity PRIMER needs to know the Young's Modulus (E) of all materials so that it can compute the stress at a given strain. As with yield stress this value is normally extracted from the material card of the cut element, but this may be missing or the material may not have a well-defined E value. When this value cannot be found you have the following options:

Use fixed value	You define a Young's Modulus value to be used
Do not compute	Elements for which an E value cannot be found are omitted from the calculation.

Axis tick marks and grid lines

This control whether or not tick marks showing the dimensions of the cut-plane coordinate system are shown on the image, and also whether or not grid lines are drawn between these tick marks.

Image Capture options

The information on the cut-section properties panel can be captured both numerically and graphically as follows:

Export to CSV

Sends the numerical information at the bottom of the panel to a Comma Separated Variable (.csv) file in a format suitable for import into a spreadsheet. This is an ASCII text file so it is also suitable for import into any external programme, and can also be read by humans.

Its format should be self-explanatory from this example:

```
Cut section properties for model 2
Section origin,          0.000000e+000, 0.000000e+000, 0.000000e+000
X axis vector,           0.000000e+000, 1.000000e+000, 0.000000e+000
Y axis vector,           0.000000e+000, 0.000000e+000, 1.000000e+000
Z axis vector,           1.000000e+000, 0.000000e+000, 0.000000e+000
Cut area,                2.968603e+005
Cut centroid Xc Yc,      2.498571e+003, 5.909979e+002
2nd moms Ixx Iyy Ixy,    4.275801e+010, 1.890375e+011, 2.316741e+010
2nd moms Iuu Ivv Angle,  1.926190e+011, 3.917650e+010, 8.121208e+001
Equal area axes Xe Ye,   2.714534e+003, 6.853174e+002
Plastic moduli Zxx Zyy,  9.119786e+007, 1.544448e+008
1st yield Axial Mxx Myy, 5.950707e+007, 9.546364e+009, 1.685284e+010
Equal force axes Xf Yf,  2.714956e+003, 6.841990e+002
Eq force Axial Mxx Myy,  5.950706e+007, 1.830389e+010, 3.096836e+010
```

Export to BMP

Create a bitmap (.bmp) file of the panel contents. This format has been chosen since it is readable by any 3rd party graphical software, and because it does not use data compression the quality of the lines and text on the image is not degraded.

Note that this panel, as with any sub-window in PRIMER may also be copied to the system clipboard by using the "Copy->Clipboard" option in the drop-down menu under the `[-]` button in the top left corner of the window.

Print

Where supported this will send a copy of the panel to the printer.

Image manipulation options

Recompute

Recalculates and redraws the current image, taking into account any changes tothe model that may have occurred since the results were last updated.

Moving the cut-section, blanking elements or changing entity visibility does not result in an automatic update of this panel, and Recompute should be used to see the results of any such changes.

Autoscale

Redraws the image autoscaled to the current menu panel size. Use this to return to the original image size following a zoom operation, or if you have resized the menu panel and the image no longer fits properly.

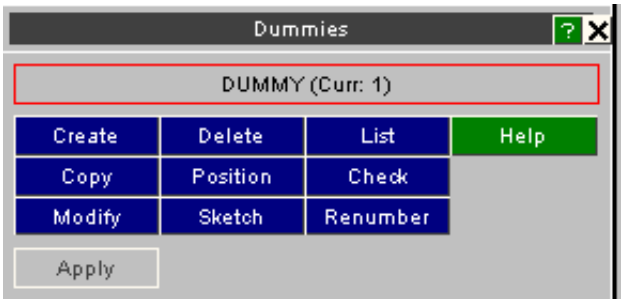
Cursor zoom

Click on this to enable cursor zooming in this panel. When selected you can drag out a rectangular area with the cursor, which will be enlarged to fill the window.

6.14 DUMMIES Positioning Occupants

Dummies positioning capabilities have been enhanced In PRIMER release 9.3. The main changes are:

- Dummy definitions may be created and edited interactively
- Positioning has been enhanced to include a "free dragging" mode.
- "Points" may be added to dummy assemblies to give further restraint and positioning options.



- Dummies may be "child" definitions of general mechanisms, and positioned in conjunction with those mechanisms.
- Dummy positions may be saved and restored, and also interchanged between models via an independent positions file.
- Dummies may be positioned from the command-line and thus also in batch mode.

What is a "dummy"?

A "dummy" is simply a collection of ordinary nodes, elements, materials, etc. but with the extra property that a *DUMMY definition has been created for it. This is a special set of additional data which defines how the parts of the dummy are connected together, and in what order. It is described fully in Appendix II, but the key points are:

- A dummy is made up of a series of "assemblies", each of which contains one or more parts and node sets. For the purposes of positioning each assembly is assumed to move as a rigid body, although its definition may include both rigid and deformable components for the purposes of analysis. There is a limit of 100 assemblies per dummy.
- Assemblies are connected together in a hierarchy, in which "parent" and "child" relationships are strictly defined. For example the torso is "parent" to the upper leg, which in turn is "parent" to the lower leg, which is "parent" to the foot; thus the foot is a "child" of the lower leg, and great-grandchild or the torso.
- Generalised Stiffness definitions between assemblies define the (local) axes a child may rotate about with respect to its parent, and a node must also be specified which gives the position on the parent about which the child rotates.
- The "root" part, generally the lower torso or pelvis, is assumed to rotate about the dummy "H-Point".
- Dummies may have a coordinate system defined, in which the angles of the "root" part are expressed, if none is defined then the dummy is assigned a system which is initially aligned with the global cartesian system. This system will be rotated as the dummy as a whole is rotated.

Dummies may constitute separate models, or may be part of a complete structural model, it makes no difference. **`*DUMMY`** definitions are appended to the file after the **`*END`** keyword, and so are ignored by the analysis code.

6.14.1 Creating and Editing dummies

Dummy definitions contain Assemblies and Points.

Assemblies are collections of one or more parts, which may be any permutation of rigid or deformable, that make up body components (torso, head, limbs, etc.).

Points are optional, and any number may be defined. They are coordinates in space, "tied to" and a property of their parent assembly, that may have restraints in any combination degrees of freedom. If a local coordinate system is defined for a point then any restraints act in that system.

Points may also be used for positioning: if a new coordinate if specified for a point then the "free drag" positioning algorithm will move the dummy accordingly.

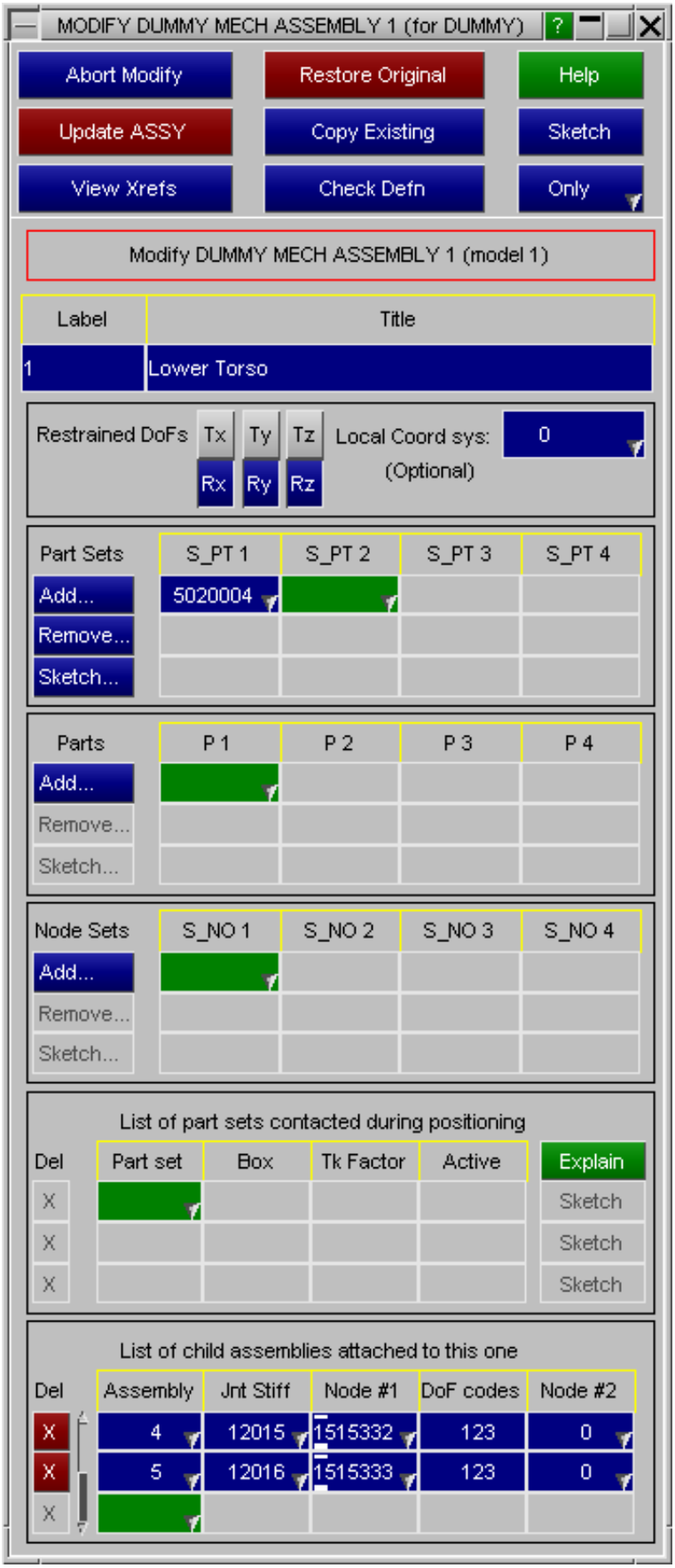
Stretch definitions allows parts of the structure are not part of the dummy to be reshaped ("stretched") by interpolating from the motion of the dummy's assemblies.

In addition from V12 onwards PRIMER will automatically determine **`*DATABASE_CROSS_SECTION`** definitions that "belong to" assemblies and update their motion with those assemblies, see Applying motion to Database Cross Sections below for more details.

Use **Create...**, **Copy..., Edit...**, etc. to select and operate on the relevant items.

When the definition is correct use **UPDATE_DUMM** to save it.

Assembly Creation and Editing



Label and title

An assembly must have a label. Labels are "private" to this dummy, thus in a model with 2 dummies each may have assembly label 1 and they will not clash.

A title is optional but is recommended.

Restraints and coord system

Assemblies may have optional restraints. These are used during "free drag" positioning to constrain movement.

If a coordinate system is defined restraints are in that system, otherwise they are in the global system.

Part sets and parts

An assembly must contain at least one part, and any number may be used in any permutation of explicit parts and part sets. It is legal to define a part more than once (e.g. in a set and explicitly): it will only be used once.

See "Good modelling Practice" below for some further rules and suggestions.

Node sets

Assemblies may also contain node sets. These can be a useful way of "associating" nodes with an assembly since all nodes in the set will be moved with it.

Contact

Contact between dummy assembly and structure.

Any number of part sets of "structure" may be specified, with an optional box to limit the volume of space in contact.

Tk factor is a factor on true thickness used for contact, and Active enables contact to be turned on/off at will. (It can also be switched on/off on the positioning panel.)

This is not a true *CONTACT definition, but rather a pseudo contact used during positioning only.

Child assemblies

Dummies assume an explicit parent -> child hierarchy, and an assembly may have any number of children (or none).

Child assemblies are connected at a point (node #1) on the parent. Rotation axes are defined by one of:

- A joint stiffness (*Constrained Joint Stiffness) giving local axes and stop angles.
- A second node #2, limiting rotation to about the axis N1N2.
- Neither stiffness nor N2, leaving rotation axes implicitly global.

The <DoF code> is also used at joints which only permit rotation about a single axis to limit rotation to that axis: 1 = about X, 2 = about Y, 3 = about Z

Visualising assemblies.

An assembly is made up from standard components, parts and sets. To visualise an assembly in isolation use the standard **Only** button at the top of the panmel, which will blank everything except for this assembly.

Assembly contents and special rules for deletion

It is normally the case that Dummy "tree files" are made by specialist suppliers and that users should not change the dummy contents. Therefore the following exceptions apply to PRIMER's normal deletion logic when using **REMOVE, DELETE_UNWANTED**.

- Parts, Part sets and Node sets used to define the structure of a Dummy assembly are "locked" against deletion.

  However part sets used to define contact between a Dummy assembly and structure are not locked, since removing them will not affect the validity of the assembly structure.

- This introduces an inconsistency in that the ***contents*** of Part and Node sets used to define a Dummy assembly, ie the individual parts and nodes, are not locked by standard PRIMER deletion rules which allow items to be removed from sets.

  From V11 onwards, where locking of set contents is available (see Locking set contents against deletion in section 5, SET) this inconsistency is removed because the contents of these Part and Node sets are also locked against deletion.

Note that there is a distinction between the treatment of Dummy and Mechanism assembly contents: Mechanism assemblies do not "lock" their contents in this way.

Good Assembly modelling practice

When defining assemblies the following rules should be borne in mind:

- **It is illegal for a part to be defined in more than one assembly.**

  The reason is obvious: since moving an assembly will move a part then having the part in two assemblies would cause conflicting movements at the assembly nodes. This will be detected and flagged as an error during the pre-positioning check.

- **It is illegal for a node in assembly A also to be defined in assembly B - with three exceptions.**

  The reason is the same as for parts above: a node may not have more than one possible source of motion, otherwise its position will be updated wrongly. This will be detected during pre-positioning checks and flagged as an error.

  However there are three important exceptions to this rule:

  1. **It is legal to have "extra nodes" on rigid parts in assembly A when their parent part is in assembly B.**

     This is a special case that is detected during the pre-positioning checks, and the normal PRIMER logic that moving a rigid part also moves any "extra" nodes is not applied in this case. This permits joints to articulate during positioning, but then to be locked during analysis in their "as positioned" configuration. (However where extra nodes on a part are in the same assembly as the part they will be moved in the normal way.)

2. **It is also legal to have a slaved rigid part (from a rigid body merge) in assembly A when its master part is in assembly B.**

   This is another special case, like "extra nodes" above, that is detected and handled differently during positioning. This is again to permit articulation during positioning but locking during analysis.

3. **Parts made of null shells (Material type 9) may legally cover multiple assemblies, but must not be defined in more than one.**

   Many dummies are covered by a "skin" of null shells that allow contacts to track across the gaps between the various assemblies. Inevitably these will include nodes common to assemblies A and B at such a gap, and may in fact extend further over the dummy model so that a single null shell part covers several assemblies.

   The pre-positioning check detects these parts, which must be shells referencing material type 9, and excludes them from the positioning algorithms - effectively making them non-structural in this context. But see (4) below for an exception.

   However if any nodes on these null shell parts are not also "structural" nodes on assemblies their positions will not be updated during positioning. This would be bad modelling practice anyway since all nodes on null shell parts need to be attached to real structure in order to give them stability during the analysis and stiffness for the contact.

4. **An exception to an exception (3) is made for "isolated" null shell parts, which *are* moved with their assembly**

   Experience has shown that some modellers include target markers on their dummies made up of pairs of null shell parts, and they attach these using tied contacts. Exception (3) above resulted in these target markers being left behind when an assembly was moved because they did not share any structural nodes with the assembly.

   Therefore the "null shell" test has been extended and these parts ***are*** moved with the assembly if they do not share any structural nodes with it.

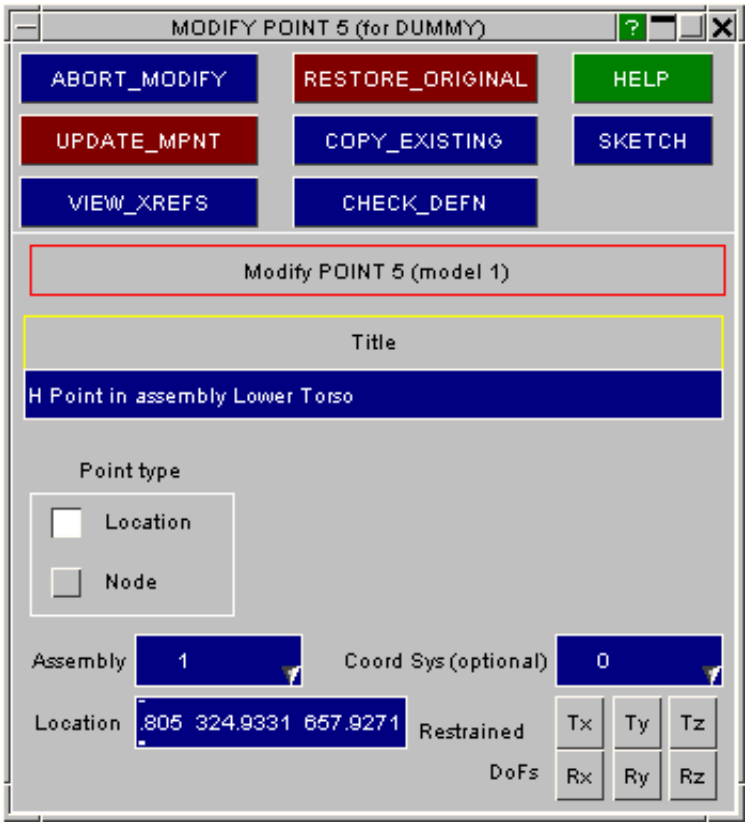
   The "is it isolated?" test needs to be quick as it is invoked during dragging operations, so it is not fool-proof and sufficiently devious modelling practice may defeat it. Users taking advantage of this should ensure that the null parts used for "isolated" parts are unique to their assembly.

- **Coordinate systems used for Joint Stiffness definitions must be defined by nodes in the relevant assembly.**

  When defining coordinate systems for Joint Stiffnesses it is important that these use nodes (*Define Coordinate_**NODES**) on the assembly to define their axes, otherwise they will not be updated as the assembly moves and the joint axes will not rotate correctly. PRIMER can visualise coordinate systems and joint stiffnesses, but some modellers add a triad of rigid beams at these locations to visualise coordinate systems during post-processing.

Further hints on good modelling practice may be found in the section on Rules for "tree" files in Appendix II.

Point creation and editing



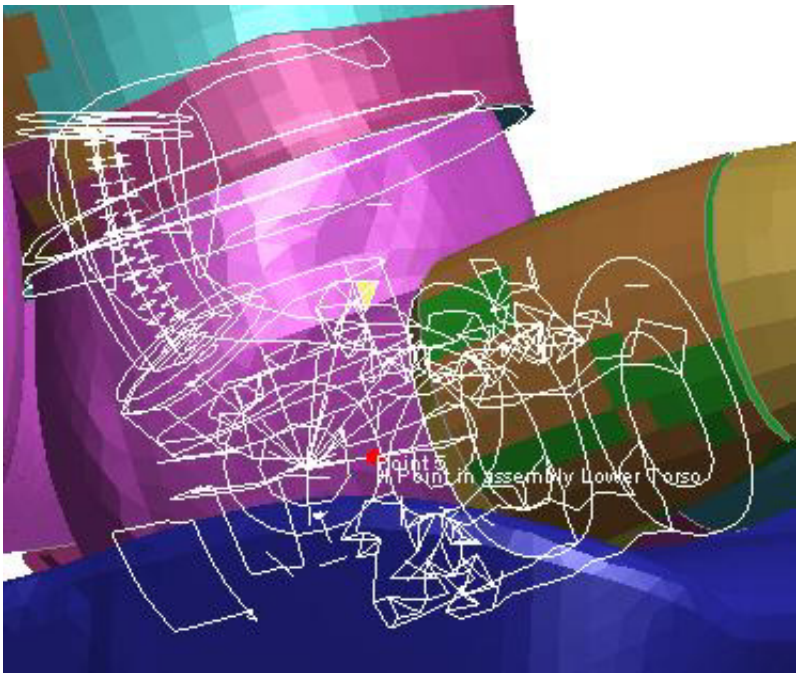
Title	A title will be generated automatically, but you can supersede this with your own. Points do not have labels
Point type	A point defined by Location is a coordinate in space that is attached to, and moves with, its parent assembly. A point defined by node is essentially the same: it obtains it current coordinate from the node. The node should normally be part of the parent assembly, but this is not mandatory.
Restraints and coordinate systems	A point's movement may be restrained in any combination of degrees of freedom (or none). If a local coordinate system is defined restraints act in that system, otherwise they are global.

Visualising Points

Points may be visualised by using the **Sketch** options both on the parent dummy panel and on their edit/create panels.

Here is a picture of the point in the example above: it is the dummy's H point, located in the centre of the pelvis. (This point is created automatically if the dummy definition includes an H_POINT card.)

It is shown as a circular red symbol, labelled with its title, with the free edges of the parent part also displayed.



Automatic creation of a point at the H-Point.

If your dummy definition includes an H-Point then a "point" as described above will automatically be created at that location, attached to the root assembly of the dummy.

STRETCH definitions

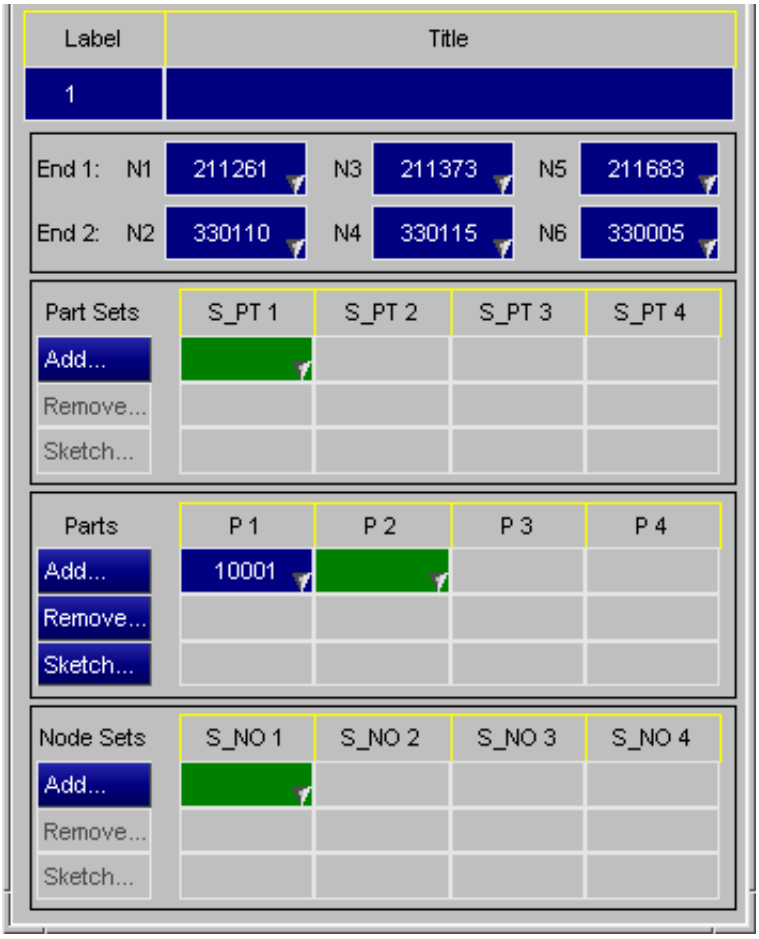
These allow you to define parts of the structure which are not part of the dummy, but which will be "stretched" by dummy assembly movement. Typical examples might be fabric spanning between two dummy assemblies, but not part of either, which needs to have its shape changed when the related assemblies move.

At least one node must be defined at each "end" of a stretch, and at least one end needs to be on a dummy assembly. You also define parts, part sets or node sets of structure that will be "stretched".

When the dummy is articulated and the assemblies move the relative motion between the two ends is interpolated onto the parts and nodes defined within the **Stretch** definition.

A single node at an end gives a "Pinned" definition, and three nodes forming a triad give an "Encastre" definition. Pinned ends only result in translation being interpolated, whereas encastre ends will result in both translation and rotation being interpolated.

This topic is covered fully under the Mechanism section, please refer to that for more information.



Dummy "Tree" files.

The information describing the dummy is saved in special keywords following the *END card.

Collectively these data cards are know as a "Dummy Tree file" and their format is described in Appendix II. This appendix also gives further rules that apply to dummy models, and advice about good and bad modelling practice. It is recommended reading if you are creating a dummy model as there are some tried and tested techniques that are known to work ... and equally some common pitfalls!

Applying motion to *DATABASE_CROSS_SECTION definitions.

From release 12 onwards PRIMER will automatically find **`*DATABASE_CROSS_SECTION`** definitions in a model that "belong to" a dummy assembly, and will update its motion as the assembly moves. In order to belong to and assembly a cross section must obey the following rules:

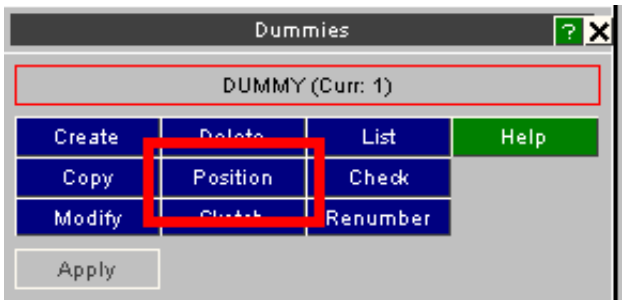
Rules that a *DATABASE_CROSS_SECTION must satisfy to "Belong to" a Dummy assembly	
It must be of type **`_PLANE`**	Sections of type **`_SET`** are ignored by the dummy positioner since they don't have an explicit location. You can still use them in a dummy, and they will be carried through to the analysis, it is simply that their location in space is determined by the nodes and sets defining the section so the positioner does not have to worry about them. In fact they may be a good solution if you want a cross-section to span elements in multiple assemblies.
It must have part set **`PSID`** defined.	Sections with **`PSID = 0`**, ie all parts in the model, are ignored.

At least one part in set **PSID** must be in this assembly, and other parts must either be in this assembly or not in any assembly in this dummy. Shell parts referencing ***MAT_NULL** are omitted from this check.	If no parts are in this assembly it is ignored. If parts are in both this assembly and also one or more other assemblies in this dummy then motion is ambiguous so the section will be ignored.

If a section has parts in more than one assembly in this dummy then a warning is issued prior to positioning, and you are given the option of "cloning" the section into as many definitions as necessary to create sections that are unique to each assembly. Each "clone" is a new section definition that is geometrically identical to the original, but in which **PSID** only contains the subset of parts present in a given assembly. These clones can be positioned with their respective assemblies since their motion is no longer ambiguous, but the original section definition (which is left unchanged) will not be moved.

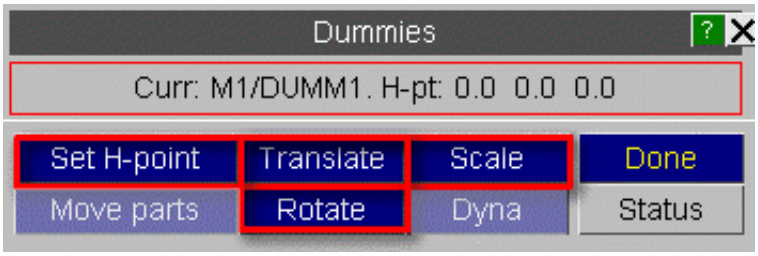
By default "move cross-section with assembly" is on, but you can turn it off in the Options panel of the positioner. You also have the option of turning it off if the pre-positioning warning detects sections spanning multiple assemblies. This on/off status is recorded in the dummy section of the keyout file - see Appendix IIa for details.

6.14.2 **Position**: Positioning dummies



Once a dummy has been defined, or read in from file, it can be positioned in a variety of ways.

Operations that apply to the whole dummy as a rigid unit:



Translate, Rotate, Scale act in exactly the same way as under the (main programme) **Orient** command. The whole dummy is moved to its new position either by explicit commands or by dragging with the mouse.

Set H-Point is simply **Translate** by another method: the dummy is translated by the difference between the current and the required H-point position. However this is often a convenient method when a specific H-Point is required.

It is important to appreciate that these commands move the dummy as a rigid whole, with no articulation of its limbs.

6.14.3 **Move Parts**: Positioning Dummy Assemblies

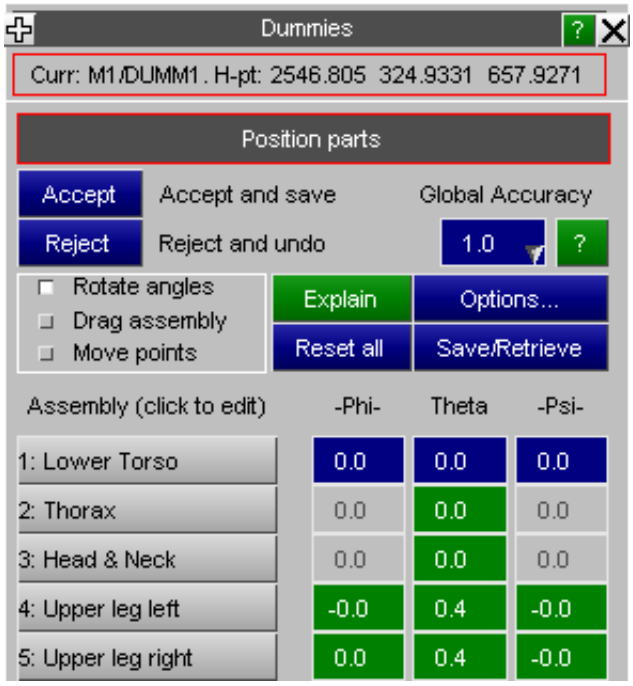
The rest of this section describes the process of positioning the dummy assemblies, i.e.***with*** articulation of its limbs.

When you enter the dummy positioner with the **Move Parts** command several operations are performed:

- Correctness of the dummy definition is checked. Parts and nodes should not appear in more than one assembly, and you are warned if they do and given some options for diagnosing and correcting these errors.
- You cannot have both **Dummy** and **Mechanism** positioning active at the same time in the same model. (This is because of the way positioning data is stored: the two processes would conflict.) If you attempt this you will be forced to shut down one operation before you can start the other.
- The current dummy position is saved as an "initial position". If things go wrong in the positioner you can return

to this as any time by using **Reset all**, and if you abort positioning using **Reject** the dummy will automatically be restored to this position.

The main positioning panel



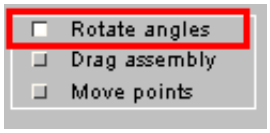
Assuming that these checks pass you then drop into the positioning panel. For dummies this operates in one of three modes:

Rotate angles	In this mode explicit rotation of assemblies about their parent connection node takes place.
Drag assembly	In this mode "free" dragging of the dummy takes place, combining translation and rotation.
Move points	In this mode points can be defined and edited, and "free" movement performed by giving updated coordinates for them.

A dummy is positioned by any combination of these modes, and when it is satisfactory the user must **Accept** it to make the geometrical changes permanent, or **Reject** it to abandon positioning and restore the original geometry.

Further positioning commands below describes these and other options in more detail.

Rotate Angles: Explicit rotation of assemblies about their connection nodes.



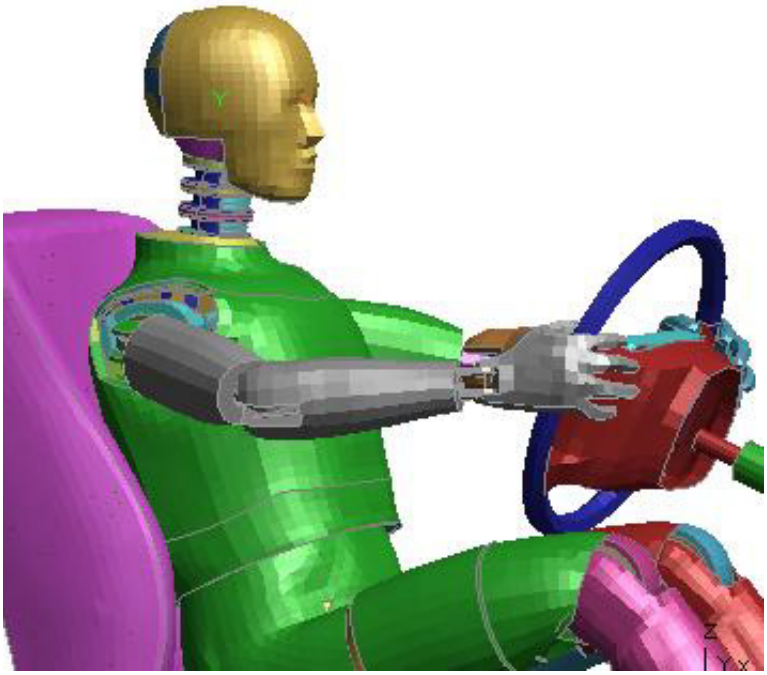
For each assembly a row showing the current joint angles is shown. Angles on a blue background are in the main dummy axes system, those on a green background are in the local system of the joint stiffness connecting this assembly to its parent. Greyed out angles are locked against rotation by the <DoF code> of the assembly.

Assembly (click to edit)	-Phi-	Theta	-Psi-
1: Lower Torso	0.0	0.0	0.0
2: Thorax	0.0	0.0	0.0
3: Head & Neck	0.0	0.0	0.0
4: Upper leg left	-0.2	13.8	-0.2
5: Upper leg right	0.2	14.1	0.2
6: Lower leg left	0.0	0.1	0.0
7: Lower leg right	0.0	0.1	0.0

In this mode an assembly is selected and rotated about one of the three axes. Both the selected assembly ***and its children*** are rotated as a rigid unit, and rotation only takes place about a single axis at a time.

"Euler" angles are used to specify and compute angles in this mode. These are described in more detail under "Euler angles in PRIMER" below.

In the example here the user has selected the upper right arm of the dummy (coloured grey to denote selection), and it can be seen that the elbow, lower arm, wrist and hand have all been selected too.

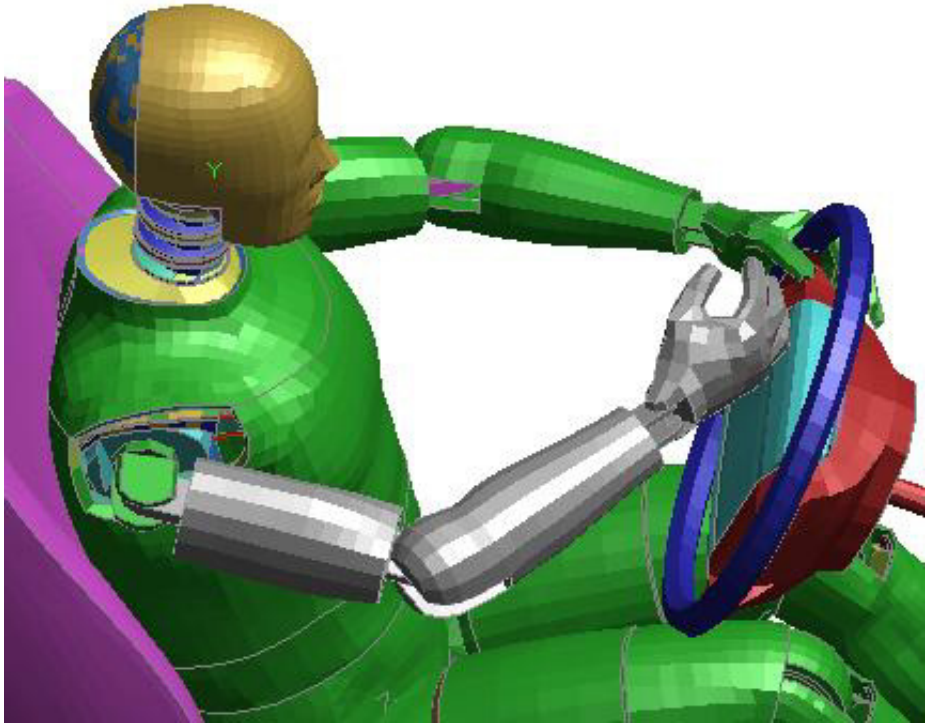


Dragging with the mouse is the easiest method:

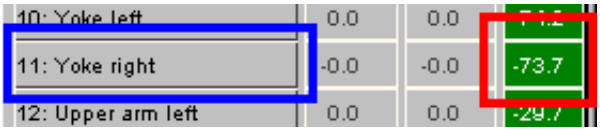
Mouse button	Drags axis
Left	Local X (Phi)
Middle	Local Y (Theta)
Right	Local Z (Psi)

Dragging is only permitted about the axes implied by the DoF Code specified on the parent assembly. In addition rotation will be limited to the stop angles specified on any Joint Stiffness definition for the joint.

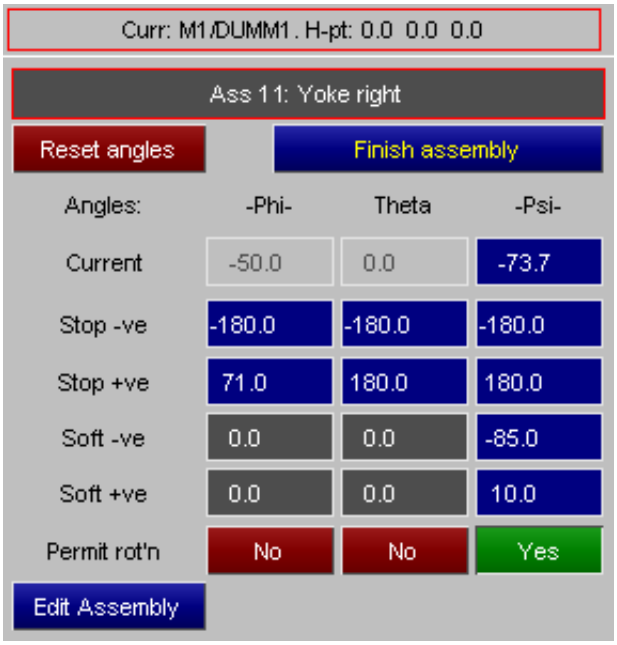
This image shows the arm after some movement, demonstrating how the assemblies below the upper arm in the hierarchy all move as a rigid combination, rotating about the (nearly) vertical axis at the right hand shoulder yoke joint.



Assembly angles can also be set explicitly by typing angles into the appropriate row text entry box (red outline):



Or the full editing panel for an assembly can be mapped by clicking on the "name" button (blue outline). The editing panel allows all the angle attributes of the assembly to be adjusted.



Articulation of each degree of freedom is limited by stop angles that defined maximum +ve and -ve articulation in degrees. Two sets of angles are considered:

- "Hard" stop angles (top two rows, designated **Stop -ve** and **Stop +ve**)
- "Soft" stop angles (lower two rows, designated **Soft -ve** and **Soft +ve**)

If no soft stop angles are defined both -ve and +ve fields will be zero and grey as shown here for the Phi and Theta degrees of freedom, meaning that only the hard angles apply.

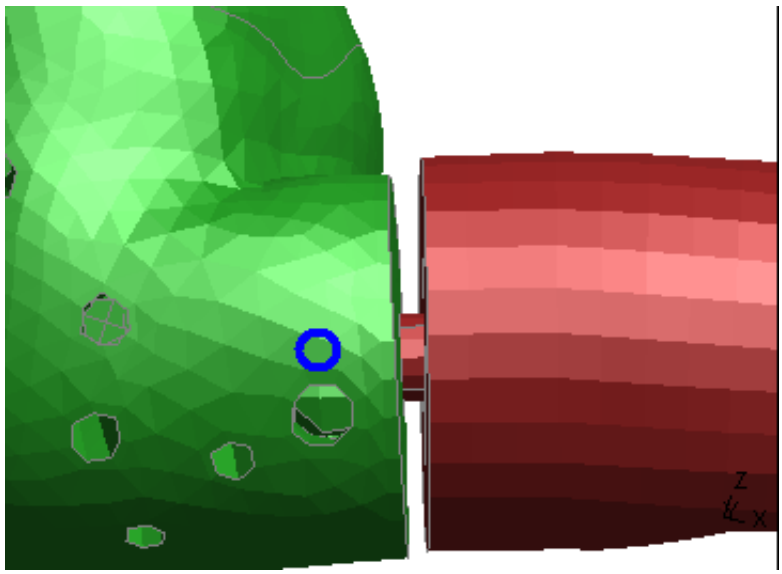
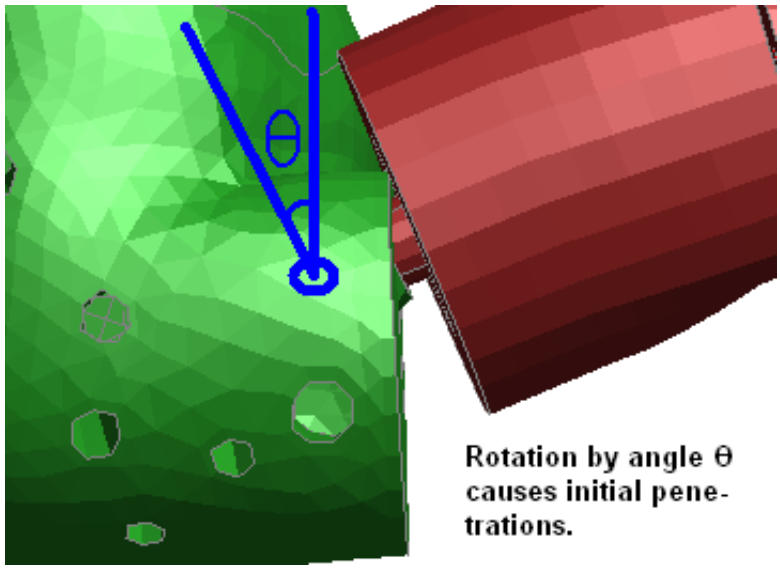
"Hard" vs "Soft" stop angles.

From PRIMER 12 onwards both "Hard" and "Soft" stop angles are supported, earlier versions did not make this distinction and simply implemented what are now described as "Hard" stop angles.

Meaning of the two types of stop angle	
"Hard" angles	Have always existed in all versions of PRIMER as plain "stop angles". • Are defined on the LS-DYNA *CONSTRAINED_JOINT_STIFFNESS card. • Act as limiting angles during analyses, with stiffness rising sharply once reached. • Will be used by PRIMER during positioning if no "soft" angles are defined for that degree of freedom.

"Soft" angles	Were introduced in PRIMER 12. • Are optional, and need not be defined. • Are defined in PRIMER's dummy tree file, under the dummy *ASSEMBLY keyword after *END. • If defined they act as a limit during positioning only, and are not considered during LS-DYNA analysis.

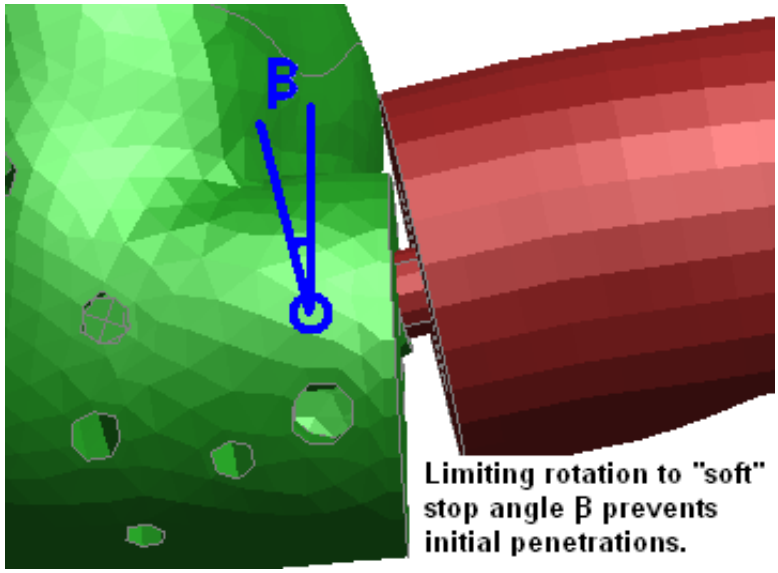
Soft stop angles have been introduced at the request of users as a simple way of preventing initial penetrations during positioning. The following sequence of images uses the junction between the upper leg and lower torso of a typical dummy to illustrate the problem.

Initial state Here is the leg (red) and lower torso (green) in their neutral, unpositioned state. Rotation of the leg takes place about the pelvis spherical joint.	
Positioned *without* Soft Stop angles The *CONSTRAINED_JOINT_STIFFNESS card used to define the local axis system and associated "hard" stop angles for the upper leg permits rotations about all axes, and also permits the leg to rotate up and down by an amount θ that can cause inter-penetration between leg and torso, as shown in this example. Of course in a LS-DYNA analysis there would be a contact surface between leg and torso which would generate forces resisting penetration, and the soft material itself would deform, so the "hard" stop angles are quite adequate. However positioning in PRIMER is totally rigid, and no compliance of the penetrating materials is considered.	

Positioned *with* Soft Stop angles

Here additional soft stop angles have been defined, limiting the upwards angle of rotation of the leg to angle β, which is set to prevent penetration.

Since the soft stop angles are only on PRIMER's dummy *ASSEMBLY card, after *END, they are not "seen" by LS-DYNA meaning that the normal hard angles are used as limits during the analysis.



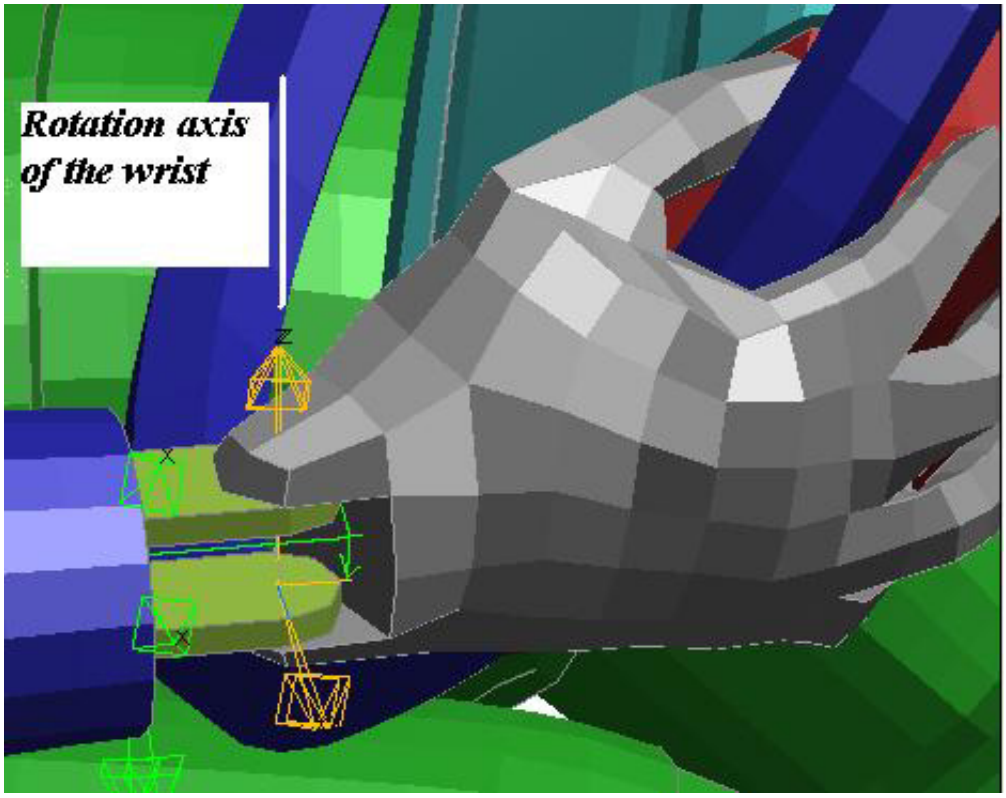
Documentation of the syntax used for soft stop angles can be found in the section on the Dummy tree file *ASSEMBLY card in Appendix II

This image shows the axes of explicit dragging more clearly.

Here only the right hand is being dragged, and the display of *Constrained Joint Stiffnesses has been turned on to show the local axis systems at the wrist to hand joint.

Rotation is only permitted about the joint's local Z axis, shown here with a white line superimposed on the image. There are two local axis systems at the joint: one on the wrist (parent) and one on the hand (child). Initially they were coincident.

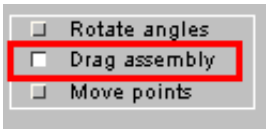
It can be seen from the divergence of the local X axes (coming out of the joint towards the observer) that the hand has been rotated a little.



The characteristics of **Rotate Angles** mode:

- **It is precise**: movement is calculated using trigonometry about the rotation point on the parent assembly. This preserves the accuracy of the dummy model and, in particular, preserves the coincidence of nodes at joints.
- **It takes no notice of** restraints on assemblies, or restrained points within them, or contacts: these are considered in "dragging" modes only.
- **It can be frustrating** to use since positioning of a limb requires a set of rotations about various joints back up its "tree", making precise positioning awkward.

Drag Assembly: Free dragging of limbs using mechanism analysis



In **Drag Assembly** mode the positioning panel changes.

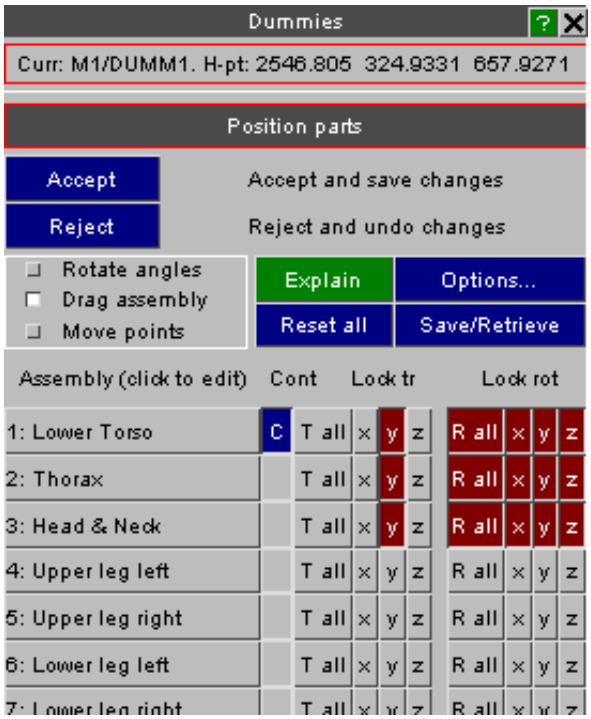
Each assembly is still shown as a row, but now:

- Clicking on the "name" button brings up the assembly editing panel as above.
- You can select the degrees of freedom to be restrained (locked) during positioning for each assembly. Restraint acts in the coordinate system of the assembly (if defined), otherwise in the global system.

In this example the lower torso, thorax and head are restrained in Ty, and against all rotations. Contact [C] is switched on between Lower Torso and structure.

As before you click on an assembly to drag it, but now the dummy is treated as a mechanism, and it will follow the mouse movement in a natural way, subject to any restraints placed upon it, and also the properties of the joints between assemblies.

Joint rotation axes and stop angles are honoured as in **Rotate Angles** mode above, but otherwise the dummy is treated as a pin-jointed set of rigid assemblies, and will respond to dragging using rigid body mechanics.



Mouse motion following picking on an assembly works as follows:

Mouse button	Resulting action
Left	All limbs attached to this limb, in both "parent" and "child" directions, that are not fully restrained become draggable, and will follow to where the motion of this limb drags them.
Middle	Only this assembly and its children will move.
Right	Only this assembly, its immediate parent and its children will move.

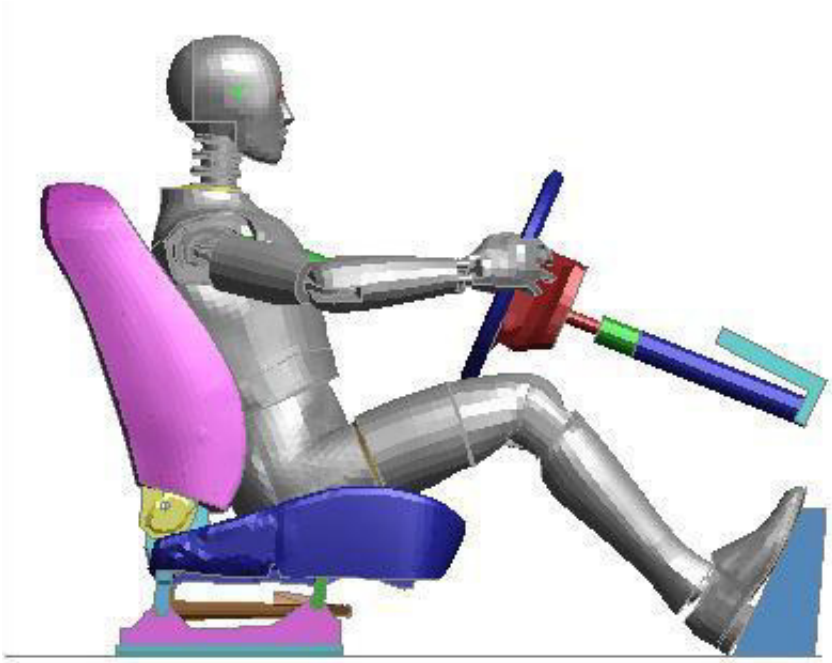
An example of Drag Assembly free dragging.

The following sequence of images shows how this might be used in practice. In this example the dummy has been positioned in the seat, with hands attached to the steering wheel and feet to the pedals. Both hands and feet are fully restrained in all degrees of freedom, the torso, thorax and head are restrained against all rotations and also Y (out of plane) translation.

The user has clicked on the lower torso with the left mouse button, so the whole dummy is selected for movement, and drags it progressively further forwards. This sequence would be carried out in a single operation, and for this dummy the drag occurs in near real-time on a modern desktop computer.

Initial condition.

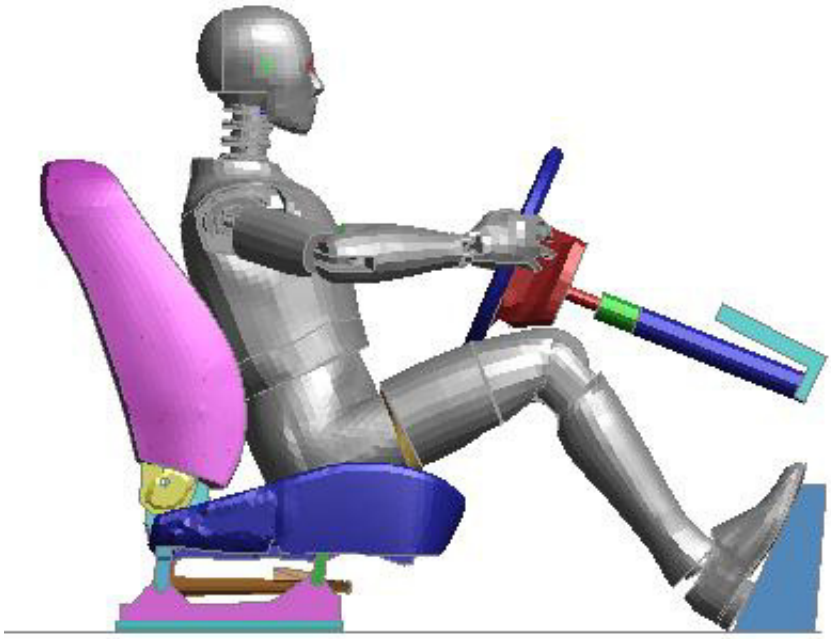
The user has clicked on the lower torso, which selects the whole dummy, and is about to drag from left to right



After about 100mm movement to the right.

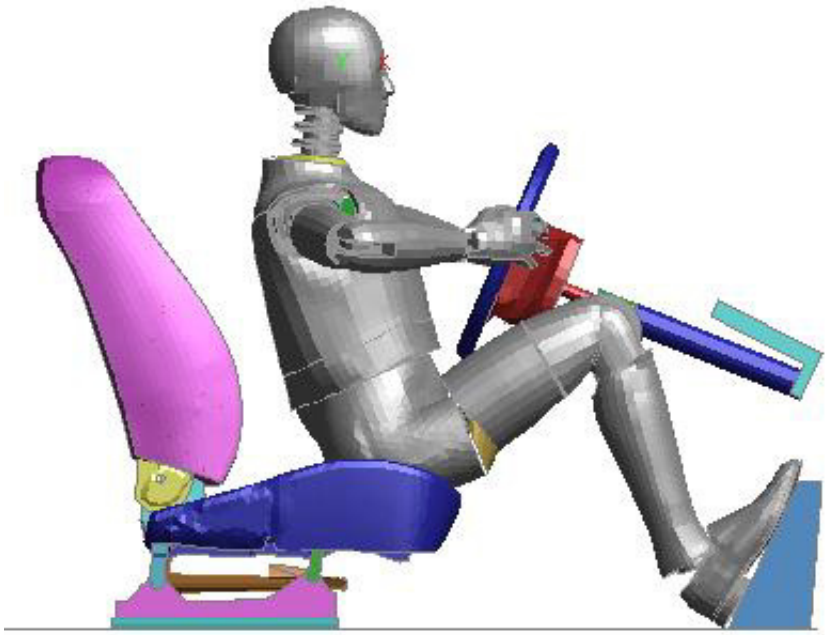
Notice that the hands and feet have remained fixed, the knees have moved up and the elbows have moved out.

Because of their rotational restraints the head, torso and pelvis regions have remained upright.



Final position.

The elbows have moved up and outwards, and the knees have moved up.



Here is the final position in an isometric view.

Arm and leg movement is very obvious!

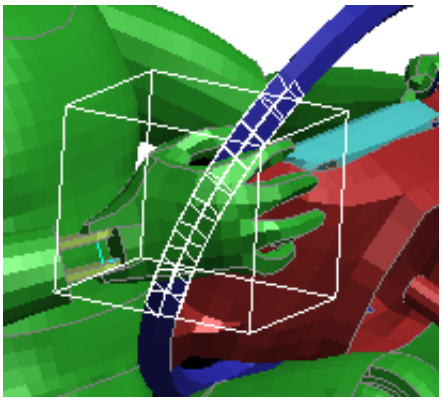
The characteristics of **Drag Assembly** mode:

- **It is approximate**: movement is calculated using rigid body mechanics in an iterative scheme, and some small errors are inevitably generated. Using the default **Options** errors will be of the order of 1 part in 10,000, or around 0.2mm for a typical dummy model which, in engineering terms, is not significant.

  However ls-dyna requires node pairs at joints to be coincident to a very tight tolerance, and the actions taken to achieve this when you **Accept** the positioned dummy are described below.
- **It considers** restraints on assemblies, or restrained points within them. You can switch restraints on/off at will during the positioning process, and indeed the "move to position, then clamp in place" process is the obvious way to work.
- **It is intuitive**: movement is a reasonably natural mixture of translation and rotation, more or less what one would get in real life from grabbing a limb and pulling it.

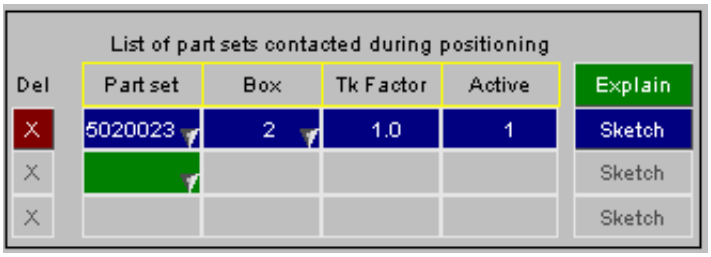
Using assembly to structure contact

In the example above the hands are fixed rigidly to the steering wheel, which prevents them from rotating and therefore forces the elbows out at an unrealistic angle.

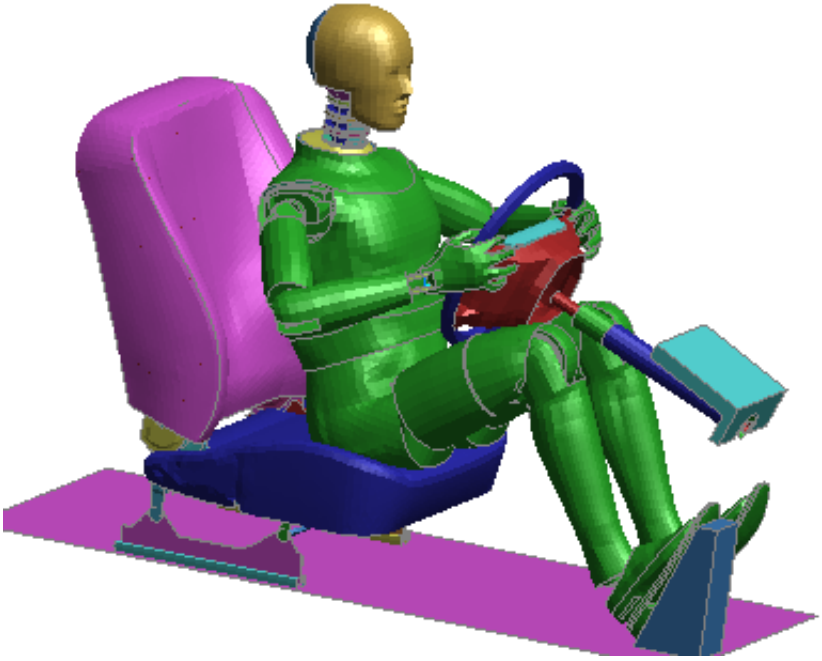


An alternative way of modelling the connection of the hands to the wheel is to define a contact between them and to turn off the fixity. This allows the hands to rotate on the wheel in a more realistic fashion and gives an altogether better final shape.

Contact for dummy positioning is not a "true" contact using the *CONTACT card, but rather a simplified version defined on the assembly editing panel as a "list of part sets contacted during positioning". Here the part set includes the steering wheel, and a box has been used to limit contact to just the section of the wheel near the right hand. A similar contact has also been set up for the right hand.



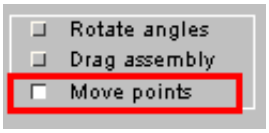
Compare the result with the final image from the example above. The positions of the arms and hands are more natural as they have been able to rotate on the wheel.



The disadvantage is that movement is much slower because of the need to compute contact, making it much harder to drag the dummy interactively when contact is used since response is so slow. For this reason contacts can be turned on/off via their [C] buttons in the "cont" column of the positioning panel.

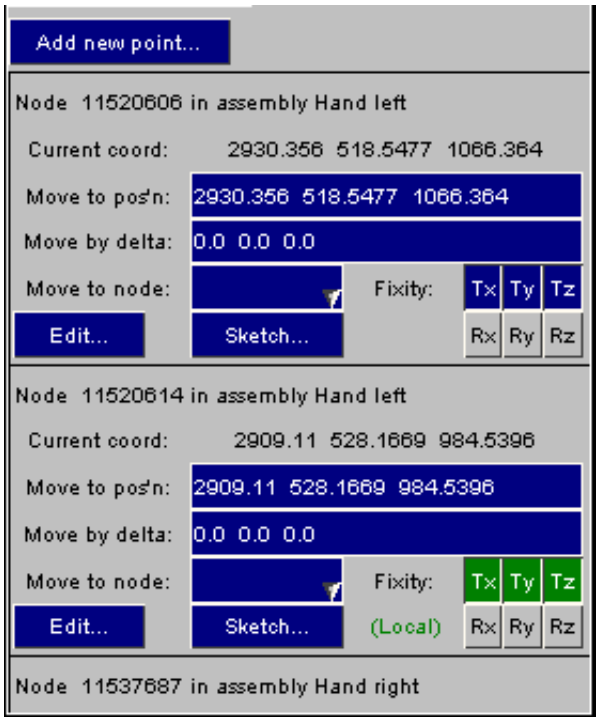
However when positioning a dummy by specifying displacement at a point motion is driven by PRIMER itself and the result is acceptable.

Move Points: "free" movement driven by updated point positions.



An alternative way of using the "free" dragging mode is to set new target positions for points. As described above any number of points can be defined in an assembly, and used both to apply localised restraint and to drive movement.

In this example two points have been created on the left hand, providing an alternative way of fixing it to the steering wheel which permits rotation about the axis through the two points. (Note that the 2nd points has a local coordinate system, so the fixity buttons are in green and "local" appears as a reminder that they are acting in the local system.)



Move to pos'n	Will move the point to the new coordinate specified
Move by delta	Will move the point by the specified [dx.dy,dz]
Move to node	Will move the point to the position of the chosen node.

In all cases the effect is similar to dragging with a mouse, with the difference that PRIMER will drive the iterative scheme for you to try to achieve the new position.

Iteration will continue either until the target point is reached, or the changes between successive iterations become insignificantly small. The latter is necessary since, obviously, it is possible to set a target position for a point that cannot be achieved because of restraints.

Using wild-card coordinates for positions.

It is sometimes the case that you want to move a point a certain distance along one axis, but not to constrain its movement along other axes. To allow for this PRIMER permits the following "wild-card" as opposed to"explicit" coordinate entry syntax.

Values entered explicitly as zero mean exactly that. Therefore

[*Move by delta*] 100.0 0.0 0.0 Means "move by 100 in X, but try not to have any movement in Y or Z."

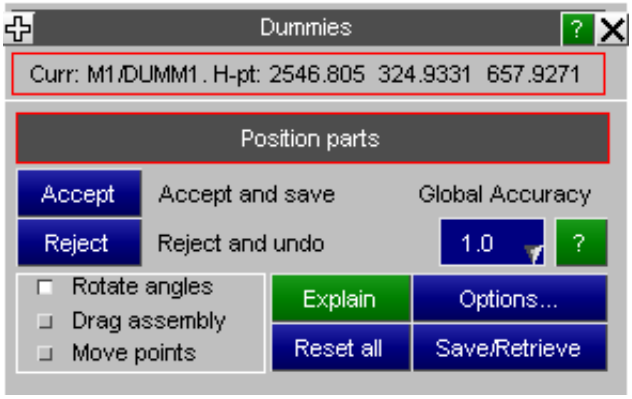
Omitted trailing values, or values entered as an asterisk "*" are taken to mean "not constrained". Therefore

[*Move by delta*] 100.0 Means "move by 100 in X, but don't care about movement in Y or Z"

Means "move by 100 in Z, but don't care about movement in X or Y"

The same syntax may be used for absolute [*Move to pos'n*] coordinate entry.

Further positioning commands



The following commands are common to all three positioning methods described above.

Accept	Accept position and save changes.
Reject	Abandon positioning, and restore initial position
Global Accuracy	Controls the accuracy of the positioning process.
Reset all	Restores the dummy to its initial position
Options...	Further positioning options
Save/Retrieve	Save and retrieve positions

Accept: accepts the current position and saves the changes you have made.

Once you are happy with the current position use **Accept** to save it and finish positioning.

Before it saves the position PRIMER examines all the nodal pairs at joints in the dummy to check that positioning has not pulled them apart. It applies a twin tolerance:

- An absolute value of 1.0e-3. This is the value hard-wired into the LS-DYNA keyword reader.
- A distance of 1.0e-6 times the model longest diagonal

Any joints at which separation of nodal pairs exceeds this figure will be listed and you will be given the option of **Autofixing** them.

This is performed by moving each pair of nodes to their average position and, so long as the errors are small, this is an acceptable distortion of the model. This is an iterative process since if a node is on more than one joint then correcting for joint A may move it out of position for joint B. If there are still errors after 5 passes the operation is abandoned and it is left to the user to sort out.

WARNING: You should avoid repeated [Position, Accept, Autofix] cycles on a model.

This is because each **Autofix** operation changes the geometry of your dummy slightly, and while a single such change may be insignificant repeated use of this feature will build up cumulative errors.

It is better to achieve a position in a single operation from an unmodified dummy model. If you are planning to generate a series of positions in succession you should use **Model, Copy** to create a new model from an original one each time, and create the position in the copy.

Reject: rejects the current position, restores the initial one and exits the positioner.

Use **Reject** if you want to abandon positioning and restore the initial position. All changes made during positioning will be lost, and you will return to the main **Position** menu with the model unchanged.

Reset all: restores the intial position.

Sometimes positioning goes horribly wrong and the best thing is to start again. **Reset all** restores the initial position that was saved when you entered the positioner, cancelling all changes made since then. You can use this at any time.

Global Accuracy: sets the precision of the mechanism calculation

By default the precision with which the mechanism positioning process calculation is performed is based on the diagonal of the box bounding the mechanism times a "convergence factor" of 1.0e-4. Therefore a mechanism fitting into a box with a diagonal of one metre will be solved to a precision of approximately 0.1mm, this being the maximum permitted error at connection points.

Note that this does not apply to* Rotate Angles *mode, in which orientation of assemblies is via explicit rotations.

Detailed convergence parameters can be set in Options, described below, but these are rather opaque and a simpler value to use is the **Global accuracy** parameter which can be in the range 0.1 to 100.0, default 1.0.

The effect of this factor is to divide the default value of the following three convergence parameters thus:

```
Convergence factor   =  1.0e-4 / Global Accuracy

End movement factor  =  1.0e-8 / Global Accuracy

Step size factor     =  1.0e-4 / Global Accuracy
```

You can still set these individually in Options below, this is just a simpler and easier way of doing that.

Global Accuracy is a floating point number that may have any value in the range 0.1 to 100.0, and typical values are:

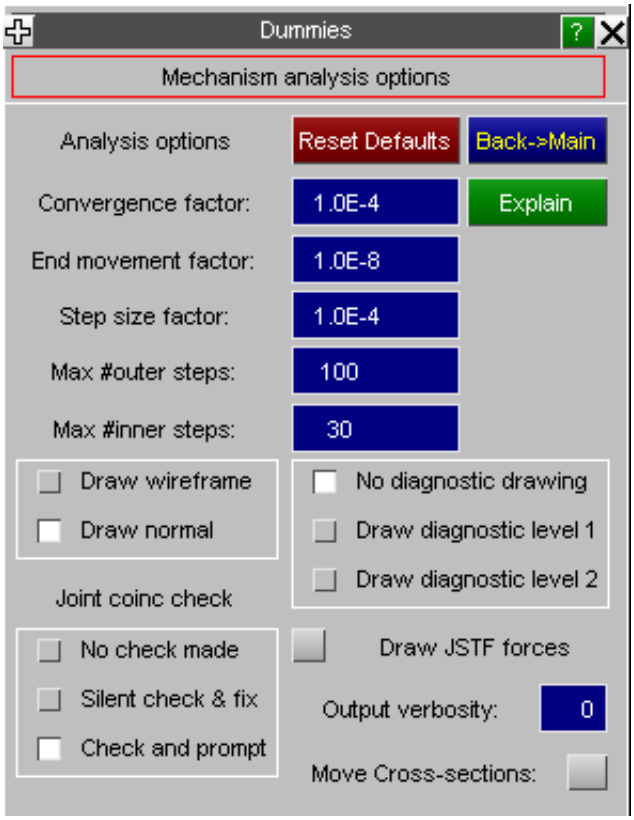
Global Accuracy typical values and their meanings.	
0.1	Very loose tolerance, ***not recommended***
1.0	**Default value**, suitable for dragging with the mouse
10.0	Tighter value, practical limit for dragging with the mouse
20.0	Tighter still, suitable for "specified" motion
50.0	Very tight, practical limit for "specified" motion
100.0	Extremely tight: will probably lock up.

You can change the value at any time, and some experimenting may be required with very complex mechanisms to find a value that gives a reasonably compromise between precision and speed of positioning.

The default value may be changed by the preference:

```
primer*mechanism_accuracy: value
```

Options... Setting positioning options.

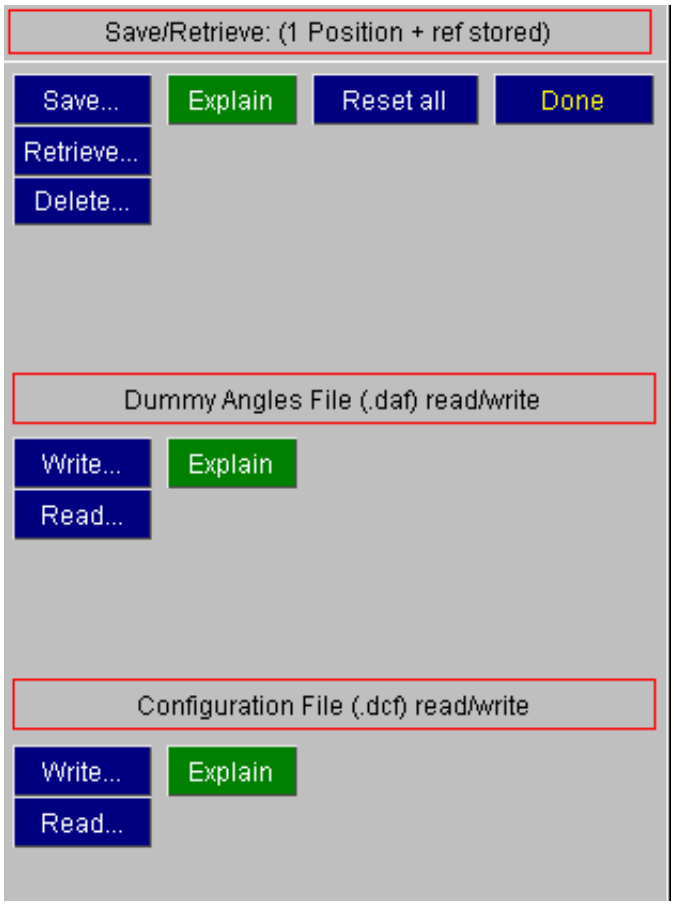


The following options affect the positioner:

Convergence factor End movement factor Step size factor Max #outer steps Max #inner steps	These options all affect the free dragging positioner only. They should not normally need to be changed, but if the dummy moves very slowly, or "gets stuck", then increasing the Convergence and Step size factors may help. Avoid much larger values as the solution will become inaccurate.
Draw wireframe Draw normal	Sets the graphics mode to be used when dragging assemblies. "Draw normal" shows the assembly in grey using normal graphics, but this demands quite a lot of cpu time and only slower computers "Draw wireframe" may be necessary to get acceptable dragging speed.
Joint coinc check	Is the post **Accept** joint node coincidence check. By default it will check and report any results, asking you what action to take. You can it work silently, which will fix any errors without further input, or turn it off altogether.
Move Cross-sections	Whether **`*DATABASE_CROSS_SECTIONS`** "belonging to" assemblies are moved with them during positioning.

Diagnostic drawing, output verbosity and JSTF force display are for programmer debugging purposes and - hopefully - can be ignored!

Save/Retrieve: Saving and restoring dummy positions.



You can store any number of dummy positions, and retrieve them at any time into the positioner.

A stored position contains a [centre of gravity] plus [3x3 direction cosines] for each assembly in the dummy, making it possible to return to a given configuration without any further calculation. Position data is stored in the dummy tree file and is saved when the keyword file is written out.

Save... Saves the current position. You only need to give a unique name.

Retrieve.. Restores a saved position. This becomes the current position and the dummy geometry is updated immediately causing it to "jump" to the new position.

Delete... Deletes the selected positions. Deletion is permanent!

A more detailed explanation of saved positions, including card formats, is given in Appendix IIc.

Dummy angle files: saving dummy positions in a model independent file

While saved positions are a powerful feature they have the disadvantage that they are specific to the current model, and also (because they contain explicit assembly centres of gravity) to the current dummy's geometry.

To make it possible to transfer positioning information between similar, but not identical dummies - for example a new version of an existing dummy - PRIMER also supports a model-independent "dummy angles file" (.daf extension).

This file contains:
- The dummy H-Point position.
- The rotation angles of the dummy.
- The rotation angles of each assembly.

On reading this file back in:
- Each assembly angle is reset to the specified angles, applied in the order [Rx, Ry, Rz].
- If the user chooses to move the H-Point then:
  - The dummy as a whole is translated to the specified H-Point position
  - If "whole dummy" rotation angles are present these are applied as an absolute orientation

This file is likely to work well when transferring positions between successive versions of similar dummies but, obviously, it will not be suitable where the target dummy geometry is significantly different to that of the source. However used intelligently it should be a useful tool.

See The Dummy Angles File in Appendix IId for a full description of the .daf file format, and the section on Euler angles in PRIMER below for a precise description of what "rotation angles" actually mean and how they are applied.

Dummy Configuration file: saving the current positioning settings only

In some situations you may wish to save and retrieve only the restraints (and any local coordinate systems) applied to the dummy during mechanism-style positioning, and a "configuration file" (.dcf) performs this function.

It contains the fully set of cards normally written between **`*DUMMY_START`** and **`*DUMMY_END`** in the keyword file, but when read back in ***only*** the following information is processed and applied to the current dummy:

Assemblies	Any restraints, and any local coordinate systems used for these.	(Assemblies are matched by label)
Points	Any restraints, and any local coordinate systems used for these.	(Points are matched by name)

Note that geometry, coordinates, connectivity and the like are ignored when this file is reread.

6.14.4 Batch (command line) positioning

A subset of the interactive positioning commands described above are also available in command-line line form. While these can be used interactively the main purpose of them is to enable positioning to be performed in batch mode. These commands will provide visual feedback if the graphical user interface is running, but if it is not (PRIMER started with "-d=batch" command line option) they will still function. A full listing of command-line commands is given in Appendix X11.

The positioning commands are invoked by the [Primer >] **`DUMMY`** command, and occupy a hierarchy as follows:

At **DUMMY >** level		
ASSEMBLY	Select an assembly by name or number, then perform one of the following operations upon it:	**FIX** *dof code* — Restrain the assembly in degrees of freedom *dof code* **TRANSLATE** *dx, dy, dz* — Translate assembly ***by*** amount *dx,dy,dz* **RX** or **RY** or **RZ** *theta* — Rotate assembly ***to*** angle *theta* degrees about x/y/z **RESET** — Undo all dummy transformations and return to initial state **DONE** — Finish with assembly and return to **DUMMY >** prompt **CONTACT** — **OFF** or **ON**. Turns assembly contact (if defined) on/off during positioning.
POINT	Select a point by name or number, then perform one of the following operations upon it: (Note: moving the point implicitly moves its "owner" assembly.)	**FIX** *dof code* — Restrain the point in degrees of freedom *dof code* **TRANSLATE** *dx, dy, dz* — Translate point assembly ***by*** amount *dx,dy,dz* **POSITION** *x, y, z* — Translate point assembly ***to*** coord *x, y, z* **RESET** — Undo all dummy transformations and return to initial state **DONE** — Finish with point and return to **DUMMY >** prompt
CONNECTION	Select a connection by name or number	**SLIDE** *distance* — Applies to **LINE** connections only, and will slide the joint by *distance* down its AB axis. **ANGLE** *theta* — Applies to **LINE** and **HINGE** connections only, and rotations the assemblies to achieve angle *theta* (in degrees) about the AB axis.
POSITION	Specify a position *name* or *id*	Retrieves and applies the stored position *name* or *id*
SAVE	Specify a position id and (optional) *name*	Saves the current configuration as a saved position id, with optional *name*.
H_POINT	Specify coordinate *x, y, z*	Will move the Dummy H-Point ***to*** coord *x, y, z*
READ_CONFIG	Specify a *filename*	Retrieves a free-standing dummy configuration file (the keywords and data between ***DUMMY_START** and ***DUMMY_END**). *Filename* will usually have the extension .dcf
READ_DUMMY_ANGLE	Specify a *filename*	Retrieves and applies the overall orientation, H-Point and joint angles stored in a Dummy Angles File (usually extension .daf).
ACCURACY	Specify a *value*	Global factor on the accuracy of the mechanism positioning process. Value must lie in the range 0.1 to 100.0
ACCEPT	Accept the current dummy position, save its updated geometry and return to the main [Primer >] prompt.	
RESET	Undo all transformations and restore the initial geometry of the dummy, remaining at this prompt level.	
QUIT	Undo all transformations and restore the initial geometry of the dummy, then return to the main [Primer >] prompt.	

Meanings of terms in the table above

dof code Is a numeric Degree of Freedom code made up of any permutation of 123456, where

1 = Tx, 2 = Ty, 3 = Tz, 4 = Rx, 5 = Ry, 6 = Rz

For example code **136** means restraint in TX, Tz, Rz

Code **0** may also be used, meaning "free all restraints"

dx, dy, dz Is a translation vector, ie a relative movement from the current position, made up of three numbers.

For example `10.0 20.0 30.0` means translate 10.0 in X, 20.0 in Y, 30.0 in Z.

"Wildcard" syntax is permitted: any number entered as an asterisk ("*"), and omitted trailing digits, are treated as "free" values. For example:

`10.0`	means translate 10.0 in X, but permit Y and Z to adopt any value.
`* * 20.0`	means translate 20.0 in Z, but permit X and Y to adopt any value

x, y, z Is an absolute coordinate.

For example `10.0 20.0 30.0` means coordinate X=10, Y=20, Z=30.

Wildcards as for translations above are permitted

theta Is an angle in degrees for the given degree of freedom.

In a dummy model angles are absolute values expressed in the coordinate system of the connection between this assembly and its parent. In most cases this will mean the system implied by the local axes of the joint stiffness definition at the joint.

6.14.5 Using dummies as "children" of mechanisms

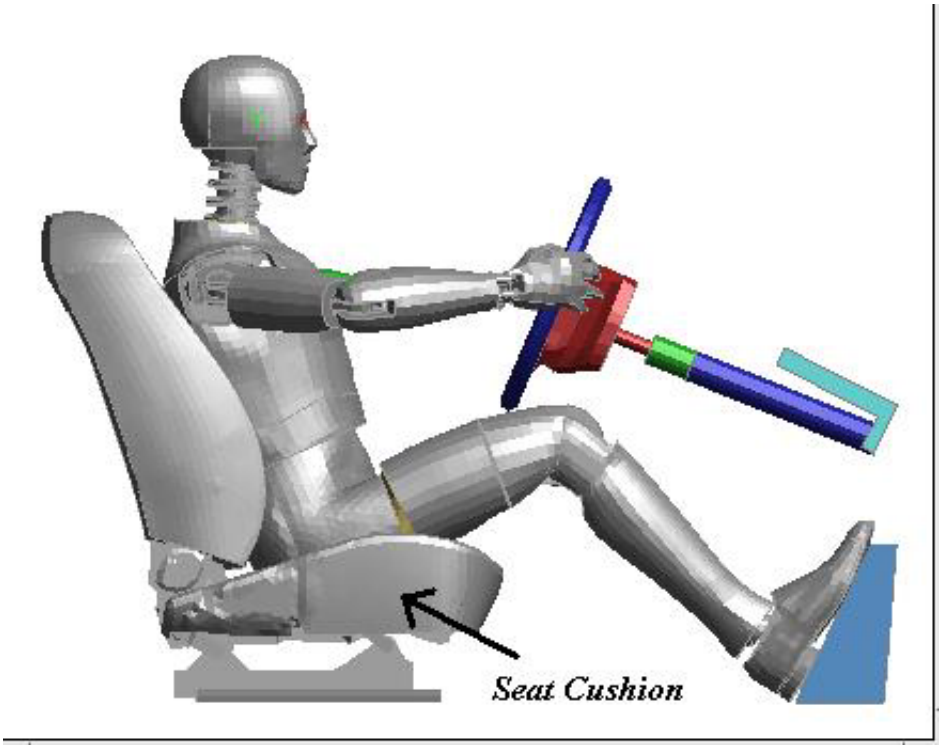
Dummies work as components of larger models, and it is usually the case that they are positioned on a seat with their feet in the floor or pedals, and their hands may be placed on a steering wheel. These "cockpit" components may themselves be capable of articulation, and the Mechanism capabilities of PRIMER may, for example, have been used to set the position of the seat.

Clearly when part of the cockpit moves it is likely that the dummy will need to be repositioned, and to make this easier PRIMER permits a dummy model to be made a "child" of a mechanism, and to move with it. When operating as a child a dummy is moved in the "free dragging" mode described above, with all the positioning capabilities and settings still active. The main difference is that the motion of the dummy is driven by the controlling assembly of the parent mechanism.

In the example below the seat assembly has been defined as a mechanism, and the lower torso of the dummy has been slaved to the motion of the seat cushion in degrees of freedom TX, Ty, Tz.

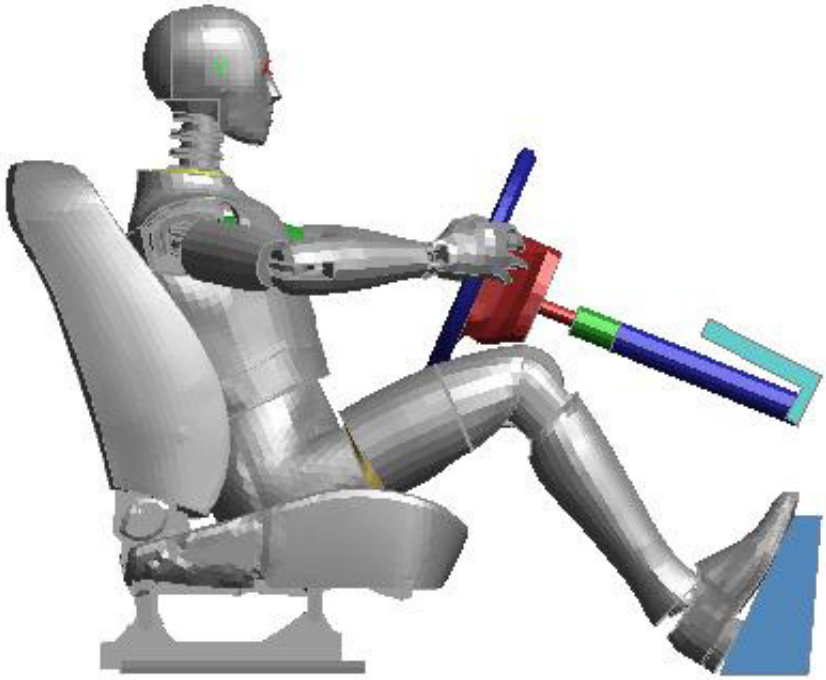
Initial state.

User has clicked on the seat cushion and the whole mechanism (seat) plus slaved dummy turn grey to denote that they are being dragged.



Intermediate state.

The seat has moved forward and risen up on its links, taking the dummy with it.



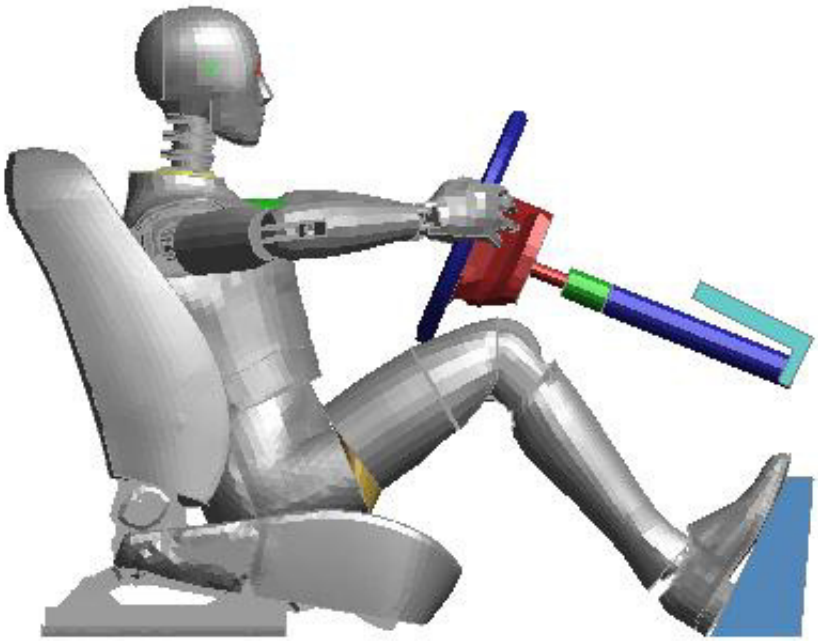
Final (very uncomfortable!) position.

In this example the seat has been moved forward and down to a ridiculous degree, but this demonstrates two things clearly:

(1) The dummy motion has remained linked to that of the seat.

(2) Connection between seat and dummy is in translation (TX, Ty, Tz) only with no rotational connection.

This is made clear by the way that the seat cushion has tilted down but the pelvis, torso and head of the dummy have not rotated.



The use of a dummy as a child of a mechanism is controlled entirely on the Mechanism panel, described in section 6.30.1.

6.14.6 Notes on using dummy angles.

There are some problems with the way GENERALIZED_STIFFNESSES have been programmed into LS-DYNA. These all arise from the method of defining joint rotations as angles about the three Cartesian axes on the "parent" side of the joint, often referred to in literature as "Euler angles".

- Where rotations take place only about one axis then there is no problem: the current angle as reported by PRIMER will be the simple cumulative sum of all rotations to date about that axis.

- However where rotations are permitted about more than one axis then life becomes more difficult since the order in which Euler angles are applied matters in two related ways:

1. A rotation about Phi, followed by one about Theta and then one about Psi will not give the same result as applying the same rotations in a different order

2. Once the current rotation about any one angle is non-zero then rotations about any other axis will result in some compound set of rotations that may not be the expected numerical sum of rotations about the individual axes.

   (To demonstrate this try the following: set the initial view in PRIMER to a plan on XY [**SXY**], then compare rotations of 90 degrees about screen X, then Y, then Z [RS 90 0 0], [RS 0 90 0], [RS 0 0 90] against the same rotations in the order Z,Y,X.)

- Therefore the reported angles for assemblies free to rotate about all "parent" axes may not be the simple cumulative sum of the incremental rotations about each axis.

To understand this requires some explanation of how "Euler angles" are used inside PRIMER, and also of how it computes and maintains the current orientation of dummy assemblies.

Euler angles in PRIMER

In the following discussion the [Phi, Theta, Psi] angle notation used by the *CONSTRAINED_JOINT_STIFFNESS card in LS-DYNA will be referred to as [X,Y,Z]. This easier to write and also to understand!

The order in which PRIMER applies Euler angles.

PRIMER treats a [X,Y,Z] compound angle definition as a set of rotations that it applies in the order X, Y, Z. To be more specific:

If the rotation matrix about X is written [Rx], and those about Y and Z as [Ry] and [Rz]

then a single compound matrix [Rc] is assembled from [Rz] . [Ry] . [Rx]

This looks counter-intuitive, but in fact when concatenating rotation matrices the effective order of rotations is right to left, ie the most recently applied rotation matrix (here [Rx]) is effectively the first rotation; thus the matrix above does indeed give rotations in the order X, Y, Z.

If you read up about Euler angles and robotics you will find that there are other possible application orders, but this is in many ways the simplest and most intuitive, so it is what PRIMER uses!

The current orientation, and computing updated Euler angles from this

Internally PRIMER keeps track of each assembly’s orientation using "direction cosines", which are effectively a local coordinate systems expressed by three vectors at right angles. When a Dummy is first read into Primer these direction cosines are initialised for each joint, taking into account any initial angular differences implicit in the *CONSTRAINED_JOINT_STIFFNESS definition.

When an assembly is rotated the compound rotation matrix [Rc] described above is applied about the "parent" node, resulting in some new orientation, and these direction cosines are updated accordingly, so that they always maintain an accurate description of the assembly’s orientation with respect to its parent.

The angles reported in the Dummy assembly rotations panel, and those used in the Dummy Angles File, are calculated from these direction cosines and ***not*** from some cumulative sum of applied angular rotations. There are two reasons for this:

1. Using a "cumulative sum" only works for rotations about a single axis; once rotations about all three axes are permitted then rotations quickly get jumbled up together. Therefore such an approach would not work for those assemblies (typically head and leg components).

2. Dummy assemblies may be moved arbitrarily during the "drag" mode positioning process, resulting in a large number of small incremental displacements and rotations. Not only would it be expensive to keep track of these, but it would also lead to a considerable cumulative error due to adding small increments to a (relatively) large running total.

In cases where rotation is about only one axis then the angles derived from the direction cosines will match those that would be computed from a "cumulative sum", but once rotation becomes significant about 2 or more axes then the values will differ. To see why this should be consider the rotation matrices required to build up the full matrix of direction cosines.

[Rx]: Rotation about the X axis: **[Ry]: Rotation about the Y axis:** **[Rz]: Rotation about the Z axis:**

Sx = Sin(theta X) **Sy** = Sin(theta Y) **Sz** = Sin(theta Z)
Cx = Cos(theta X) **Cy** = COs(theta Y) **Cz** = COs(theta Z)

```
[  1   0    0    ]      [  Cy   0   Sy  ]      [  CZ   -Sz   0  ]
[  0   Cx   -Sx  ]      [  0    1   0   ]      [  Sz   CZ    0  ]
[  0   Sx   Cx   ]      [  -Sy  0   Cy  ]      [  0    0     1  ]
```

Concatenating these together in the order [X, Y, Z], ie [Rz] . [Ry] . [Rx] gives the compound matrix [Rc]:

```
[   Cy.Cz      Sx.Sy.Cz - Cx.Sz      Cx.Sy.Cz + Sz.Sz   ]

[   Cy.Sz      Sx.Sy.Sz + Cx.Cz      Cx.Sy.Sz - Sx.Cz   ]

[   -Sy        Sx.Cy                Cx.Cy              ]
```

From which it can be seen that a set of Euler angles can be extracted as follows (using the notation <**ij**> is row <**i**>, column <**j**>)

Theta X = arctan(32/33) Since (Sx.Cy / Cx.Cy) = (Sx / Cx)
Theta Y = arcsin(-31)
Theta Z = arctan(21/11) Since (Cy.Sz / Cy.Cz) = (Sz / CZ)

However there are four well known problems with this calculation method:

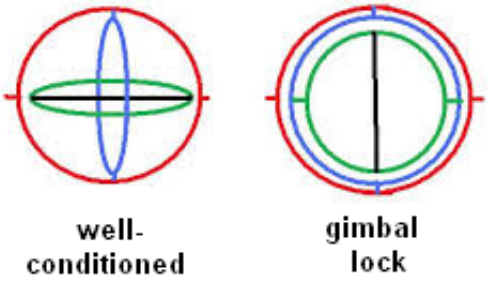
1. The rotation about the Y axis, theta Y, can only be obtained in the range +/-90 degrees from the arcsin() operation.

2. At the special case of Theta Y very close to +/-90 degrees, ie Cy = 0, the calculation of the rotations about the other two axes is ill-conditioned. To see why, here is the [Rc] matrix above with Cy = 0:

```
[  0    Sx.Sy.Cz - Cx.Sz   Cx.Sy.Cz + Sz.Sz ]
[  0    Sx.Sy.Sz + Cx.Cz   Cx.Sy.Sz - Sx.Cz ]
[ -Sy          0                   0        ]
```

Clearly the arctan() operations will be upon (0/0) for both theta X and theta Z, ie undefined.

This situation is analagous to "gimbal lock" in a 3 axis gyro-compass: the special case when the outer (X, red) and inner (Z, green) gimbal axes become co-planar with the middle (Y, blue) axis.



3. If rotation has taken place about more than one axis then the angles returned from this calculation will not necessarily be the same as those input, although the result of multiplying through by them to achieve a new orientation will be correct.

4. As mentioned above the rotation order [X, Y, Z] is implicit in this calculation, and combined rotations about 2 or more axes in a different order will give a different result. (Although, again, it will give a consistent result when used to orient a dummy or limb.)

PRIMER deals with these problems as follows:

1. If rotation takes place about the Y axis only, or nearly so, (ie theta X and theta Z both less than +/- 10 degrees) then special exception logic is used to calculate theta Y in the full range of +/-180 degrees. Therefore dummy limbs which are locked in X and Z rotation may be rotated safely in the full Y range.

2. The ill-conditioning problem is treated by using double-precision arithmetic, and relying on the (sensible) behaviour of the standard **atan2()** function near these singularities. In practice this solves the problem in virtually all cases, although rotations of exactly +/-90 degrees about the Y axis should still be avoided if at all possible if stop angles are to be used.

3. Compound rotations about multiple angles are calculated as described above, and a rational set of angles that matches these cosines is returned, even if it is not what you input in order to create them. This does not normally matter unless "stop angles" have been defined, in which case these will become increasingly unreliable as the angular movement of the assembly departs from its initial orientation.

4. The "order of combined rotation" problem is really the same as (3) above, and is treated in the same way.

In all cases the angles reported are "correct", in that if they were applied to the initial reference position of the assembly

they would give the current orientation, however they may not be what you expect. Perhaps a good way of thinking about this is to consider a journey across the earth's surface defined by increments of both latitude and longitude: your input would be a series of "rhumb line" increments, whereas your position would in fact be reported as the "great circle" angles required to get there. In addition if you travelled around the world, and approached your start position again, the angles returned would be those of the "shorter" distance, possibly negative, rather than the "longer" ones travelling around the globe.

Realistically "stop angles" will only work properly for joints that are only permitted to rotate about a single axis, in which case they can be computed unambiguously in the full +/-180 degree range.

Achieving explicit Euler angles in the positioning panel.

Once again, if rotation is only to be about a single axis then what you type in will be what you get reported back. However if you want to type in an explicit set of angles about multiple axes you may find that you don't get what you expect. The best solution to this "composite angle" case is to proceed as follows:

1. "Undo" any current angles in the order Z, Y, X. In other words set Theta Z to zero, then Theta Y, then ThetaX; in this way you will get back to angles (0.0, 0.0, 0.0).

2. Set the new angles in the order X, Y, Z. In this way they will not "interfere" with each other.

It follows from this that if you want to modify just one angle of an existing set you will be able to do so directly if:

- It is theta Z (ie the last cumulative one to be applied).

- Or the rotations applied "after" it are zero.

  So if you want to rotate about Y you will be able to get away with simple typing if theta Z is zero. If it isn't then it will be necessary to reset theta Z to zero, apply the new Y angle, then restore the original theta Z value.

6.14.7 Editing and Moving Dummies outside PRIMER

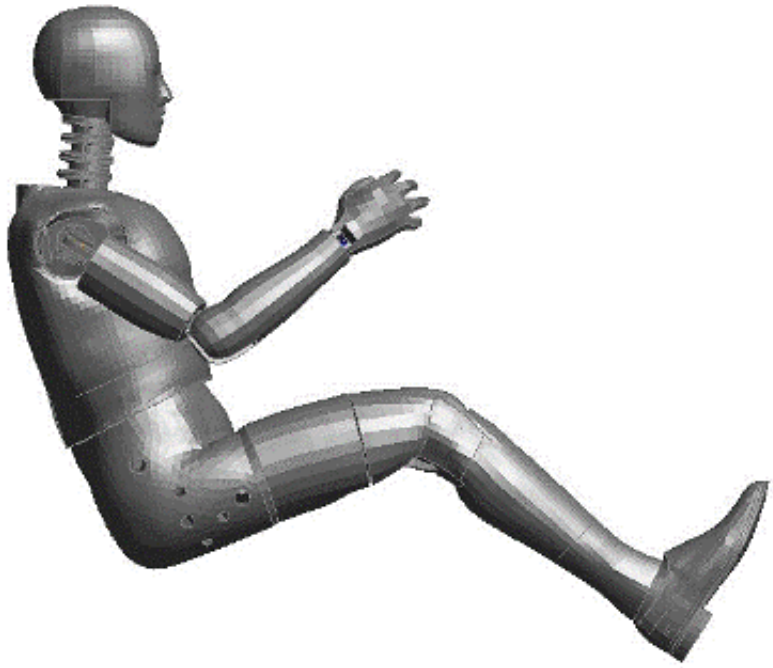
As a general rule it is recommended that Dummies (and mechanisms) are only edited and oriented inside PRIMER. It "knows about" their internal storage and will update obscure things such as point coordinates that might otherwise get left behind.

However it is recognised that model assembly, and in particular manipulation of components such as dummies and seats, may take place outside PRIMER and if this happens the guidelines in section 6.30.7 Editing and Moving Mechanisms (and Dummies) outside PRIMER should be observed.

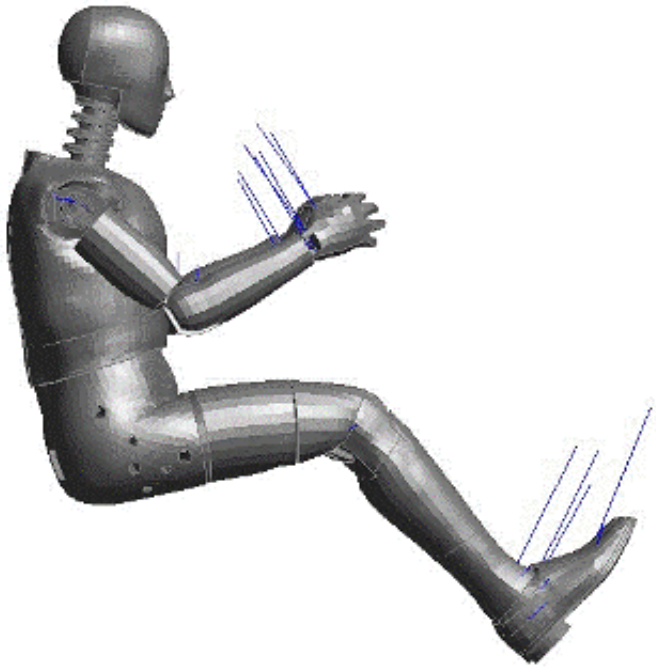
6.14.8 Dummy positioning using LS-DYNA

The LS-DYNA dummy positioning tool in PRIMER allows the user to create an LS-DYNA analysis automatically to position a dummy. This works in a similar way to the PRIMER seatsquash LS-DYNA method. This method of positioning is appropriate for those who wish to capture deformations in the foam/rubber parts of the dummy that occur due to the positioning, that are not captured by PRIMER's traditional dummy positioning methods. The LS-DYNA positioning method is illustrated below:

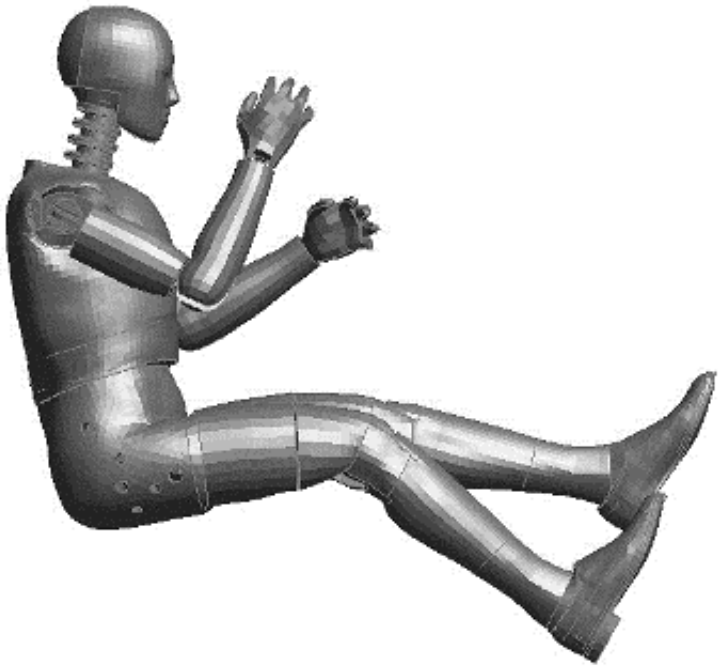
Starting point - dummy in original position.



Through using the PRIMER LS-DYNA positioning tool, an LS-DYNA analysis is created that will "pull" the dummy into the desired position.



A DYNAIN file is produced by the LS-DYNA analysis. PRIMER can now import the DYNAIN data back into the original model. This will mean updated coordinates capturing the deformation of the dummy foam/rubber parts will be updated in the original model.

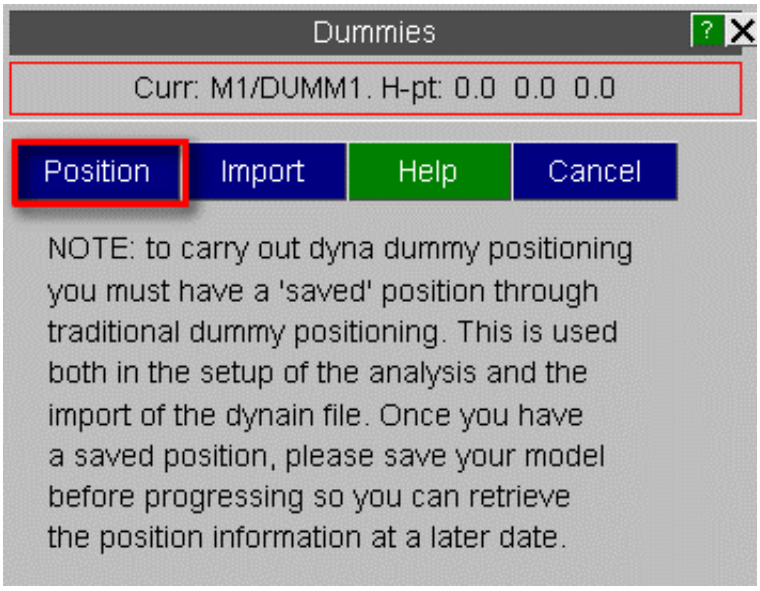


The tool works through a series of steps to create the LS-DYNA analysis. Before using this tool, the user must save the target position using Primer's traditional dummy positioning methods, as the saved position is used to create the entities required to "pull" the dummy into position. The initial model should contain just the dummy.

Step 1

Step 1 is a reminder to ensure you have a saved position using the traditional PRIMER positioning methods. See section 6.12.2 for information on saving positions.

Once you have a saved position, press **Position**.

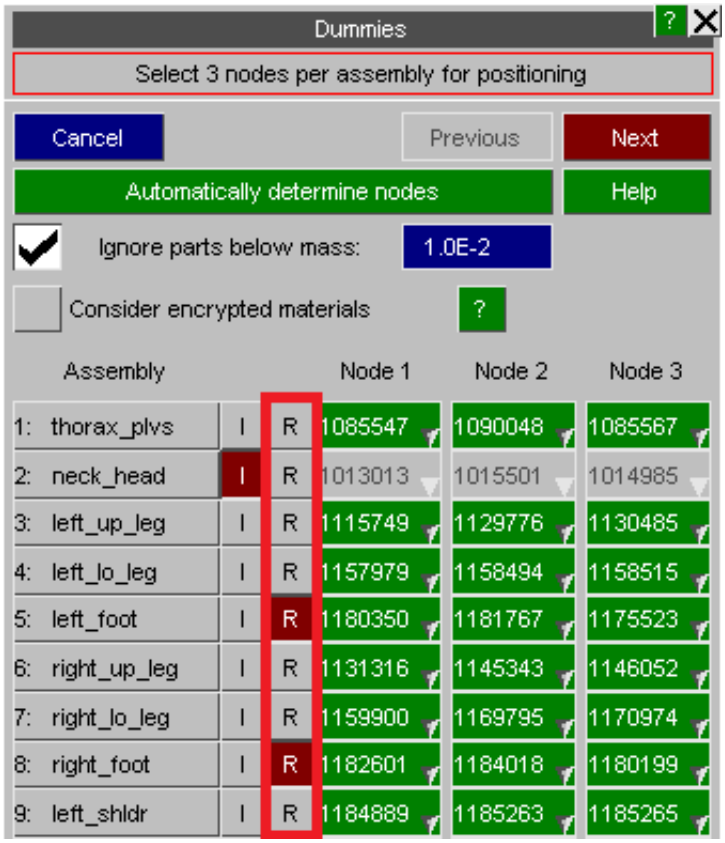


Step 2

To position the dummy, PRIMER requires three nodes per dummy assembly. These nodes are used to "pull" each assembly into their final position in a LS-DYNA analysis. The nodes selected should be on a rigid or stiff part of the assembly. The nodes can be set manually, however it is recommended to use **Automatically determine nodes**. This will find rigid nodes in each assembly to use during positioning.

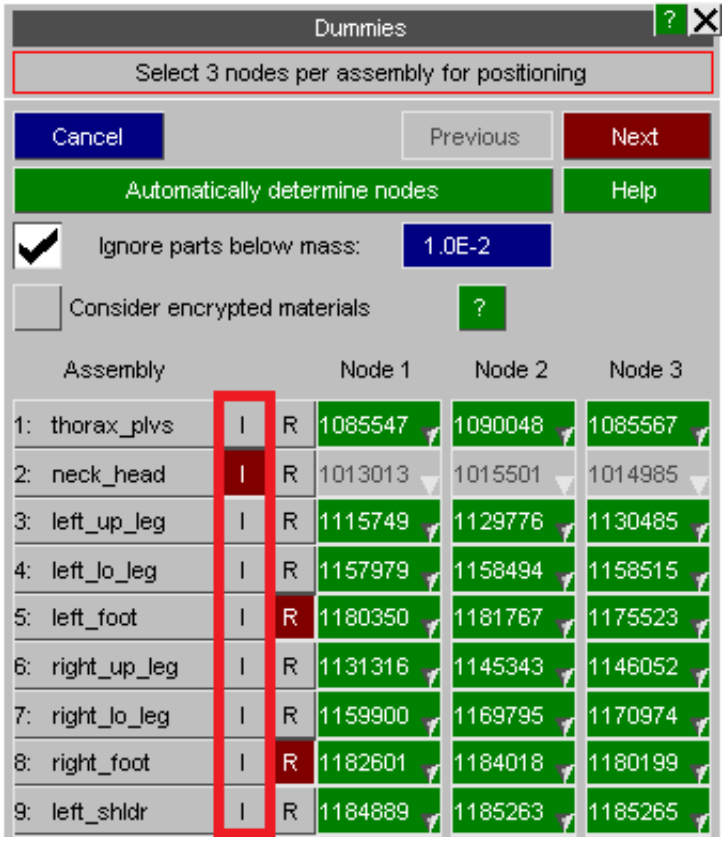
Note you can specify to ignore parts below a certain mass when automatically determining the nodes. This is so that small rigid parts that you do not want to be "pulled" (for example, the dummy may contain small rigid target marker parts) are not considered.

You also have the option to rigidify any assembly for the LS-DYNA analysis. This may be beneficial for soft extremities, such as hands and feet, which may become excessively deformed during the LS-DYNA positioning. If the **R** switch next to the assembly is toggled on, PRIMER will rigidify the assembly when setting up the analysis in the final step. Note that when toggling the **R** switch, Primer will automatically recalculate the 3 nodes for that assembly, as there will now be more/less "rigid" nodes in the assembly.



You can also chose to ignore any assembly so that cables are not attached. This may be beneficial if the model is over constrained. Removing cables from certain assemblies may help to solve this. Press the **I** switch next to the assembly to ignore it.

Once you have selected the nodes for each assembly, chosen which assemblies you wish to rigidify and/or ignore, click **Next**.



Step 3

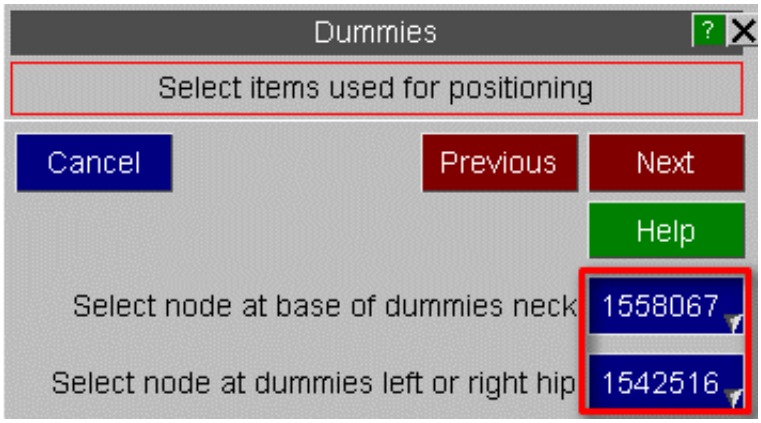
The next step is optional and allows you to select any part from the assemblies you have chosen to rigidify that you may wish to remain deformable.

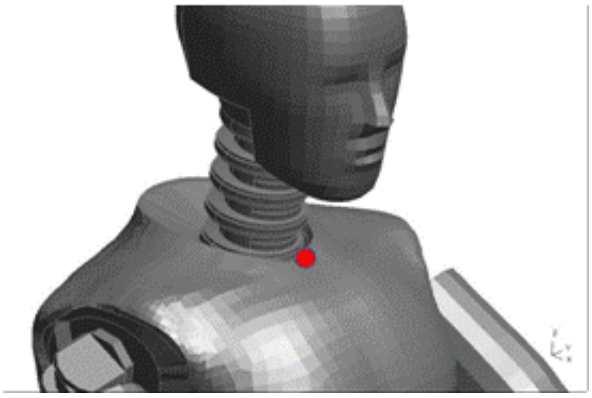
Once you have made any selection, or to skip this step, click **Next**.

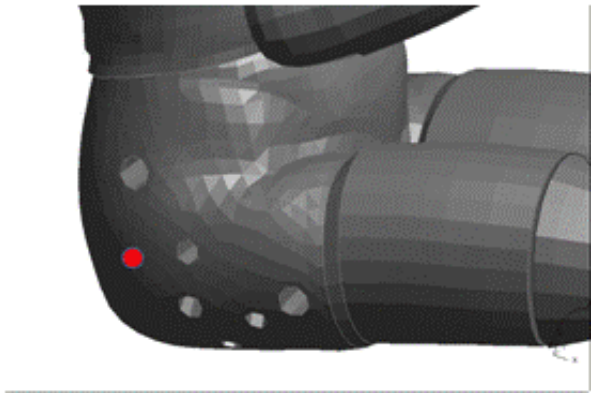
Step 4

When the analysis model is created, PRIMER will move the dummy to the desired H-Point. PRIMER will then use the neck base node to rotate the dummy around it's local Y axis to the approximate final position. This helps to reduce the overall movement of the assemblies during the analysis and hence can reduce run times. It is recommended that the neck node be set to a node at the top of the torso/base of the neck. Primer will also use the hip node to rotate the dummy around it's local Z axis to the approximate final position. It is recommended that this be set to a node on the right or left hip (outer surface). See images below for examples of this.

Once you have selected the two nodes, click **Next**.



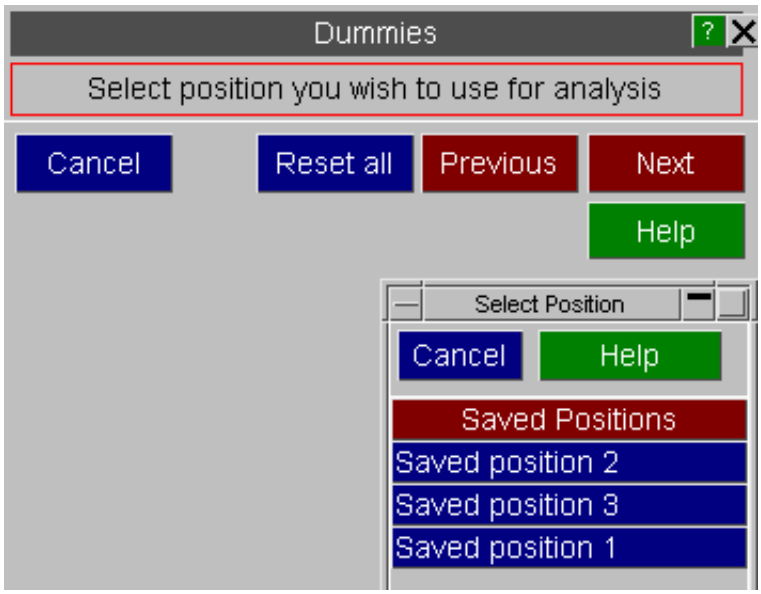




Step 5

The next step is to choose the position saved earlier. This is the position you want to achieve through the LS-DYNA analysis.

Once you have chosen the position, click **Next**.

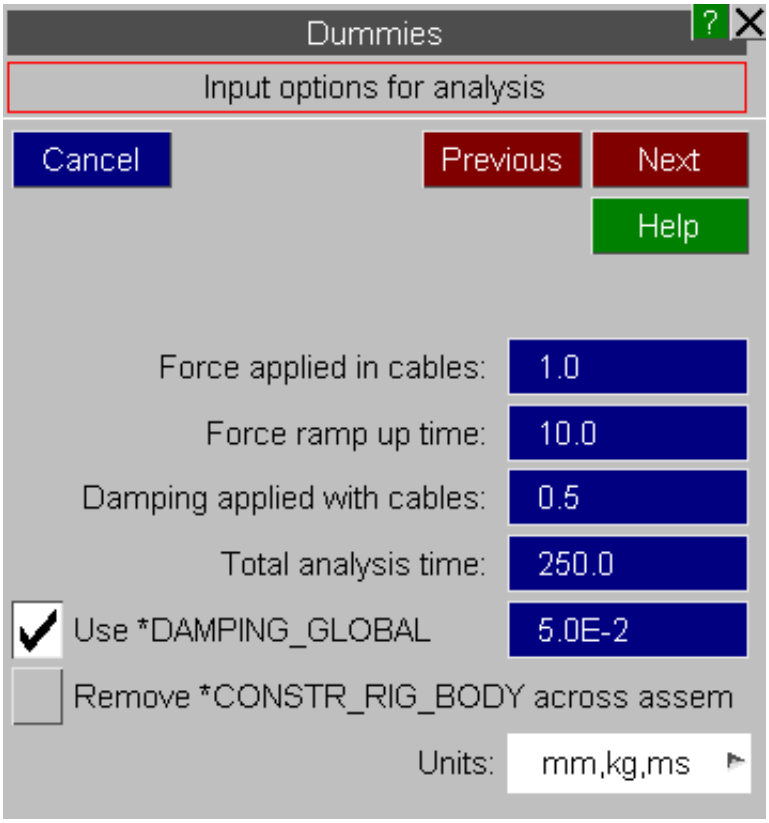


Step 6

At this point you may wish to save the model. The nodes chosen in the previous steps can be saved to the dummy tree in the keyword file. Leave the **Dummies** tab open, and click on **Model->Write** to save the model.

The LS-DYNA method works by creating cables and dampers that pull the assemblies into the final position. A number of inputs relating to the analysis can be set on the final panel:

Force applied in cables - The force that is applied to the cables to pull the assemblies into position.
Force ramp up time - The time taken to ramp the force up from zero.
Damping applied with cables - This is the damping applied in-line with the cables used to pull the assemblies into position.
Total analysis time - Termination time for the analysis. This is also used during creation of a *DEFINE_CONSTRUCTION_STAGES card which is used to create a DYNAIN file to import back into PRIMER.
Global damping - Global damping applied in the analysis (can also be turned off).
Remove *CONSTR_RIG_BODY across assem - turn this on if your dummy model has rigid body constraints across assemblies. Sometimes rigid body constraints are created in the model across assemblies to simulate the locking of assemblies relative to each other during a crash test. During positioning these would not be locked so should be removed during the setup of the pre-simulation analysis model.
Units - Specify the units system you are working in to update the above values accordingly.



It is recommended to use the default settings to start with. These can then be modified with subsequent analysis is required.

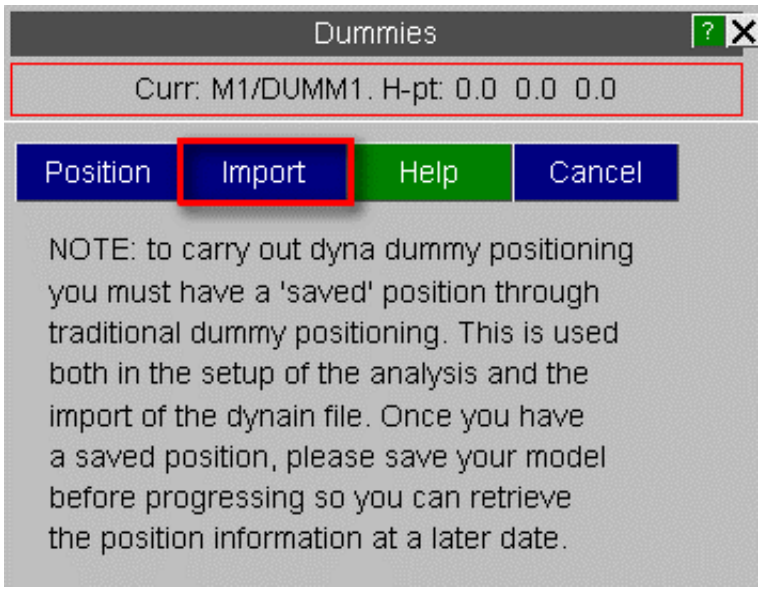
Once you have set the values you want, click **Next**, the **Apply**.

The analysis is now ready to run. Once complete, the analysis will produce a DYNAIN file, which will have a name similar to "end_stage001_dynain". This contains coordinate and initial stress information from the analysis which can be imported back into the original model using PRIMER. To import the DYNAIN file back into the original model, use the following steps:

Step 1

Step 1 is a reminder to ensure you have a saved position using the traditional PRIMER positioning methods. You need the original saved position for the import tool should you wish to further modify the position of the dummy using Primer's traditional rigid body dummy positioning methods. See section 6.12.2 for information on saving positions.

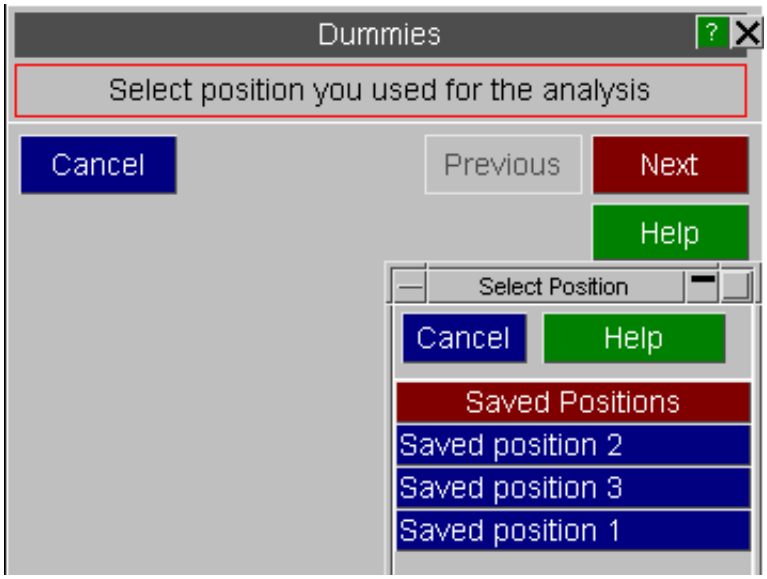
To continue with the import procedure, press **Import**.



Step 2

Step 2 is to select the position you used when creating the LS-DYNA analysis. This is the position saved using Primer's traditional rigid body positioning methods. The position needs to be chosen here to ensure any further positioning is possible.

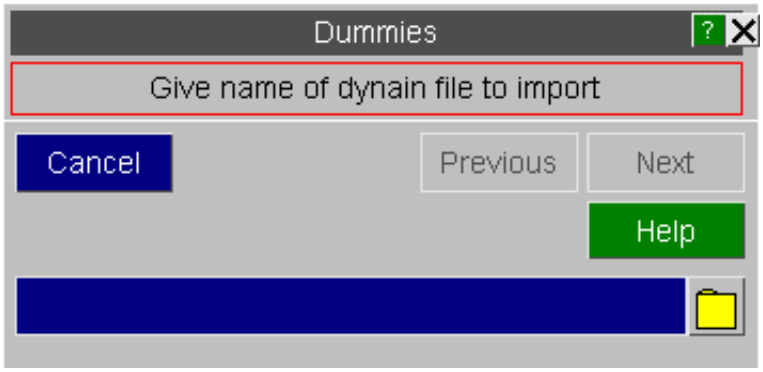
Once you have chosen the position, click **Next**.



Step 3

Step 3 is where you select the DYNAIN file produced by the LS-DYNA analysis.

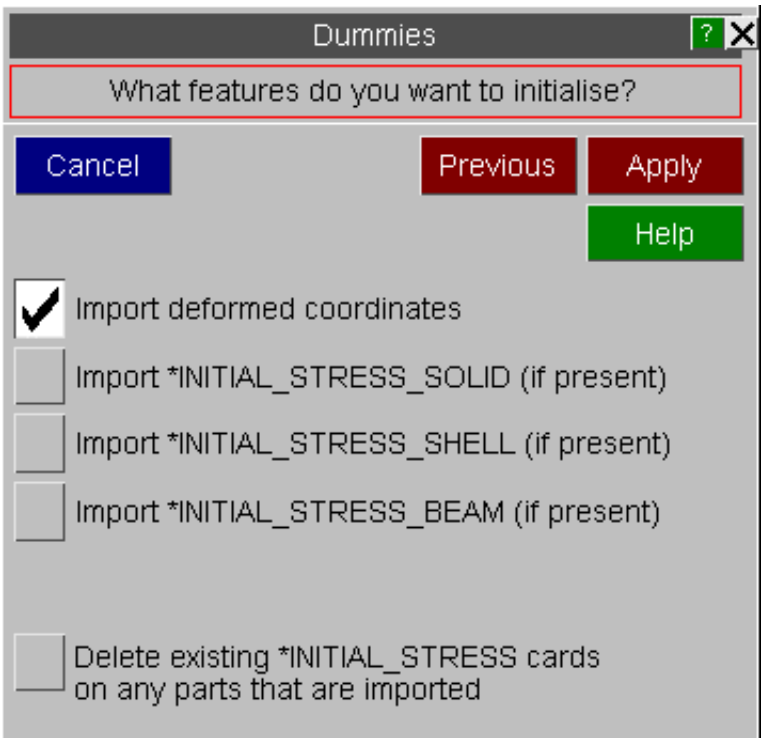
Once you have chosen the DYNAIN file, click **Next**.



Step 4

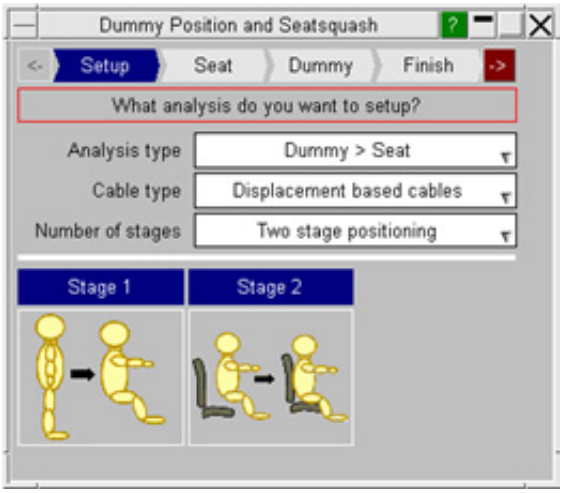
Step 4 is where you select what data you wish to import from the DYNAIN file.

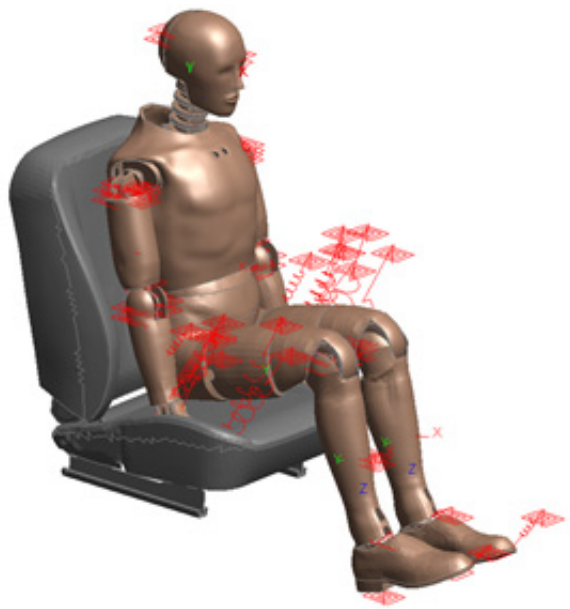
Once you have made your selection of what you want to import, click **Apply** to import the data into your model.



6.15 DUMMY AND SEATSQUASH

From v15.0 onwards PRIMER can create models to position a dummy and squash the seat foam in a single LS-DYNA analysis. It does this by creating cables to pull the dummy into position and means that interactions between the various parts are taken into accout.

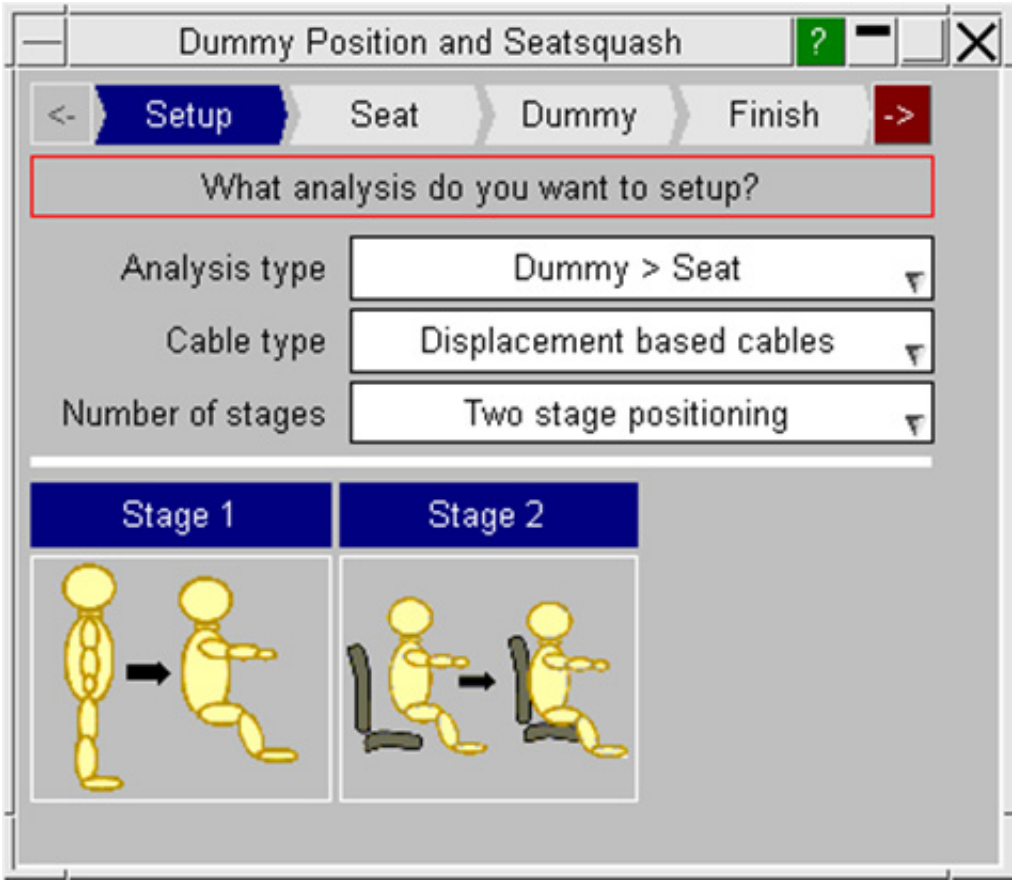




6.15 .1 Dummy and Seatsquash Setup

A floating window guides you through the process of creating the model.

The top of the window highlights in blue the step in the process you are currently at. You can move back and forward through the steps by pressing the '<-' and '-> buttons.

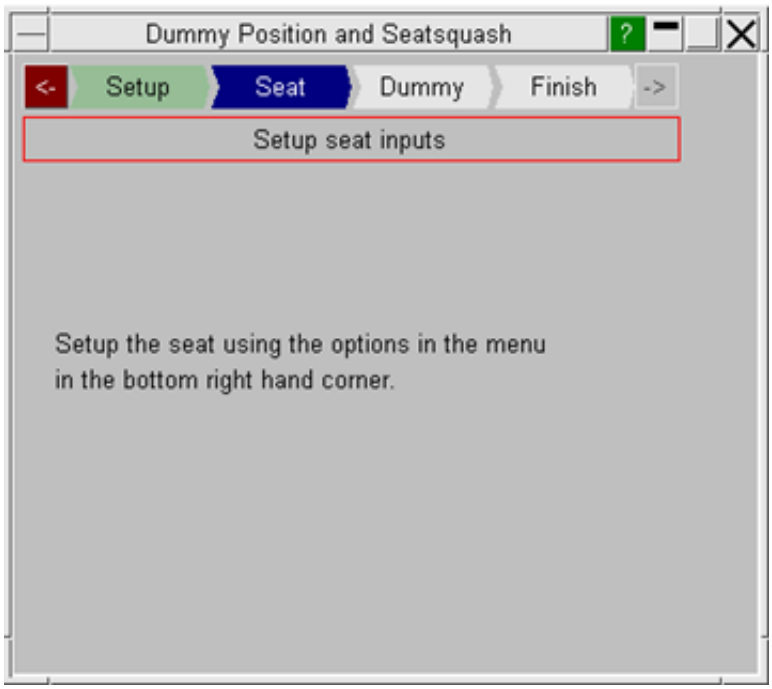


The first stage is to select what type of analysis you want to setup. The options are:

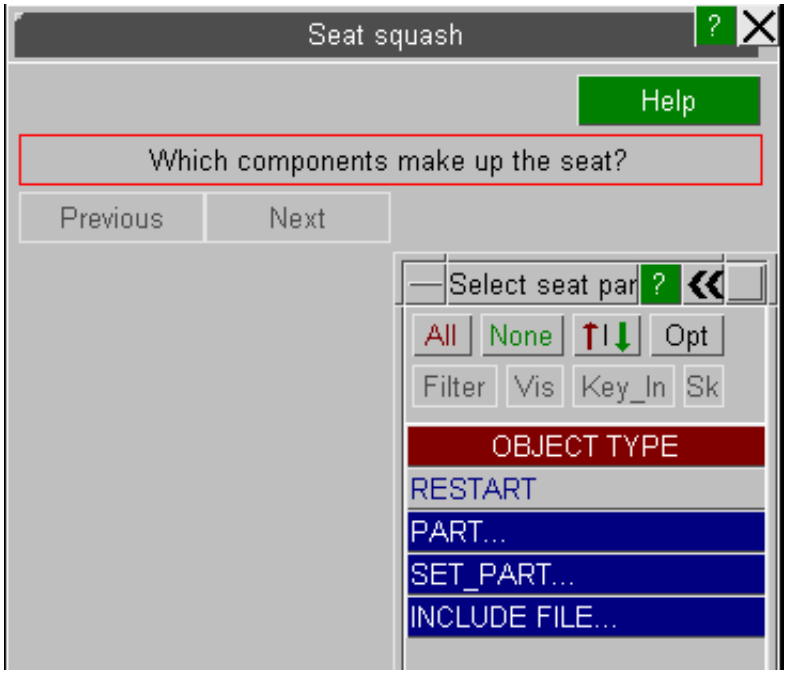
- Analysis type
  - Dummy > Seat: Setup a combined dummy positioning and seatsquash analysis
  - Dummy: Setup a dummy positioning analysis only
- Cable type
  - Displacement based cables: A displacement is applied to the cables to ensure the dummy will end in the final position by the end of the analysis.
  - Force based cables: A constant force is applied to the cables. Using this method there is no guarantee that the dummy will end in the final position before the analysis has finished.
- Number of stages
  - One stage: Position the dummy and squash it into the seat in one stage
  - Two stage: Position the dummy in one stage and then squash it into the seat in a seconde stage
  - Multi stage: Position the dummy in more than one stages. You define the multiple intermediate stages for positioning the dummy, like move hands in one stage, move legs in second stage and so on. This gives you more control over the positions. In multi stage, only displacement based cables is supported.

6.15 .2 Seat Setup

If a combined dummy positioning and seat squash analysis type was selected, the next stage will ask for information required for the seat squash analysis (part IDs, dummy to seat contact ID, etc):

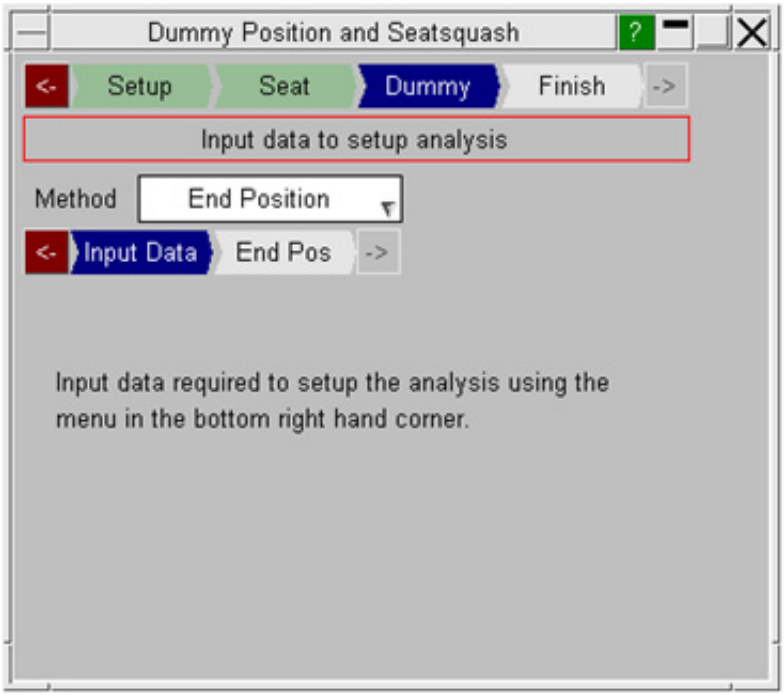


This data is entered in the menu in the bottom right hand corner of PRIMER. The steps to enter the data are similar to those described here.

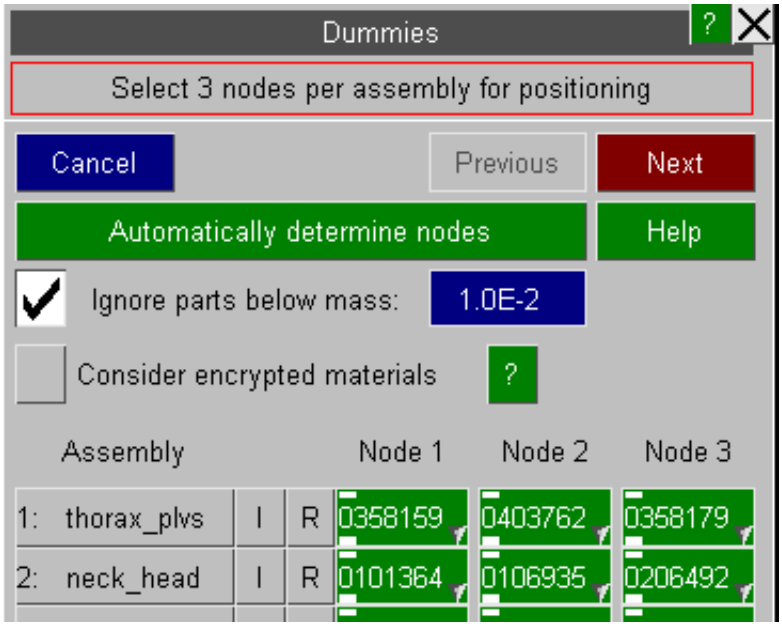


6.15 .3 Dummy Setup

The next stage will ask for data required for the dummy positioning. This includes things like which nodes the cables should be attached to, the analysis time, the stiffness for the cables, etc.



This data is entered in the menu in the bottom right hand corner of PRIMER. The steps to enter the data are similar to those described here.



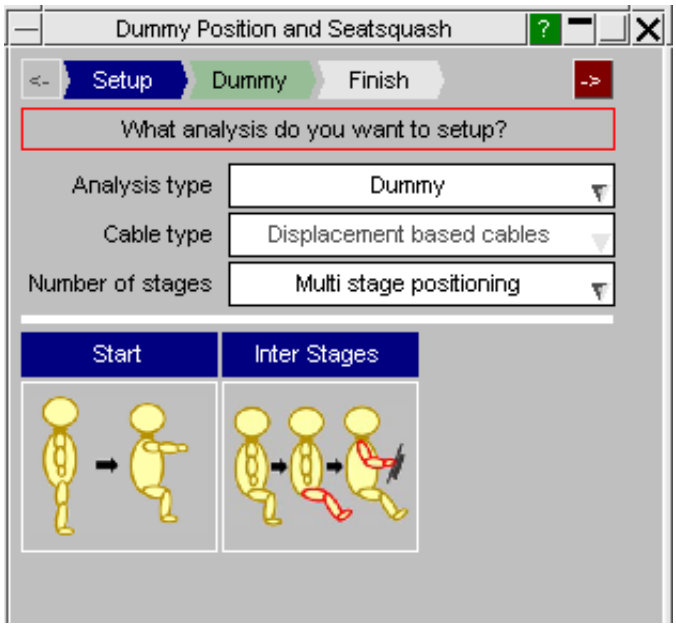
To set up the analysis you need to define a starting position for the dummy, an intermediate position (if the two stage option was selected) and a final position. There are two methods for defining these positions:

- End Position: You define the final position and PRIMER will calculate the starting and intermediate positions. This is the quickest way to define the positions.
- Sequential: You define the start, intermediate and final positions. This takes more time to setup, but gives you more control over the positions.

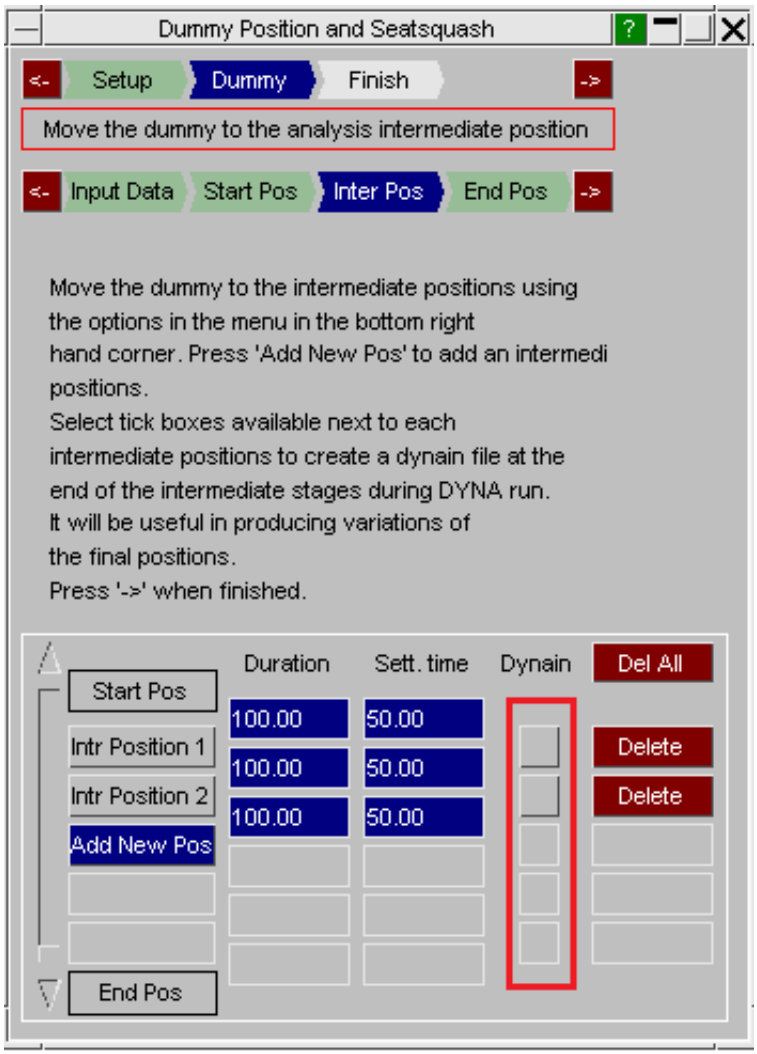
You can define multiple end positions. PRIMER will create a model for each one.

6.15 .3.1 Mutli Stage Setup

If multi stage option was selected during the analysis setup, you need to define multiple intermediate positions for the dummy.



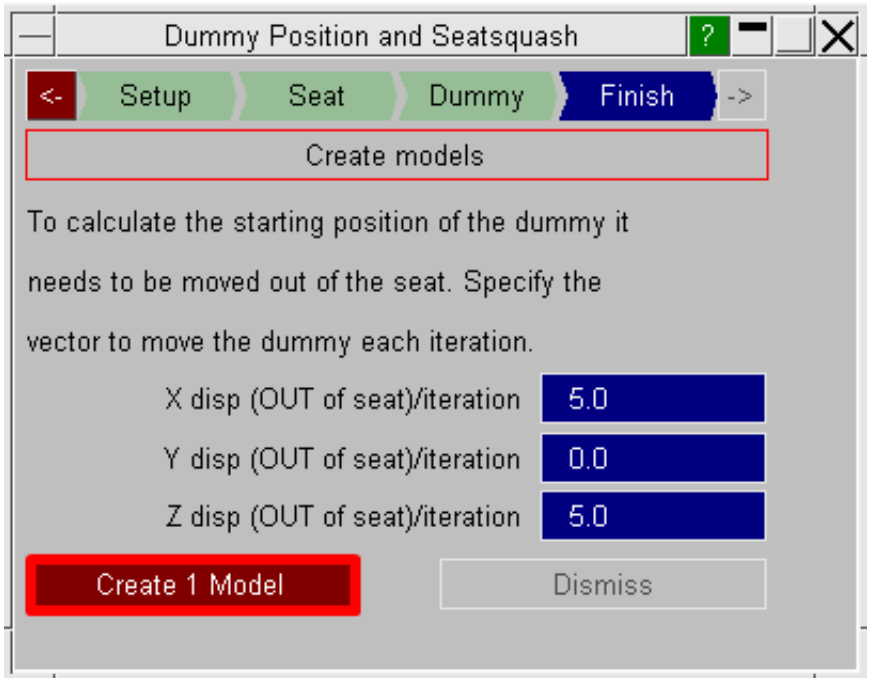
Move the dummy to desired position using the options in the menu in the bottom right hand corner and hit **Add New Pos** to add the position as an intermediate position. To update an existing intermediate position, select the desired **Intr Position** and move the dummy. Select tick boxes in each row to write out the dynain files at the end of the selected stages during DYNA run. It will allow you to produce variations of the final position.



6.15 .4 Finish Setup

Finally, if you selected a combined dummy positioning and seat squash analysis with the 'end position' method you will need to define a vector to depenetrate the dummy out of the seat so that PRIMER can calculate the start and intermediate dummy positions.

You can then create the model(s) to run in LS-DYNA. This will output a dynain file containing the final coordinates which can then be imported back into the original model.



Once the model(s) have been created you will need to write them out from PRIMER and then run them in LS-DYNA. This should produce a DYNAIN file for each model containing the coordinates and initial stress information from the analysis.

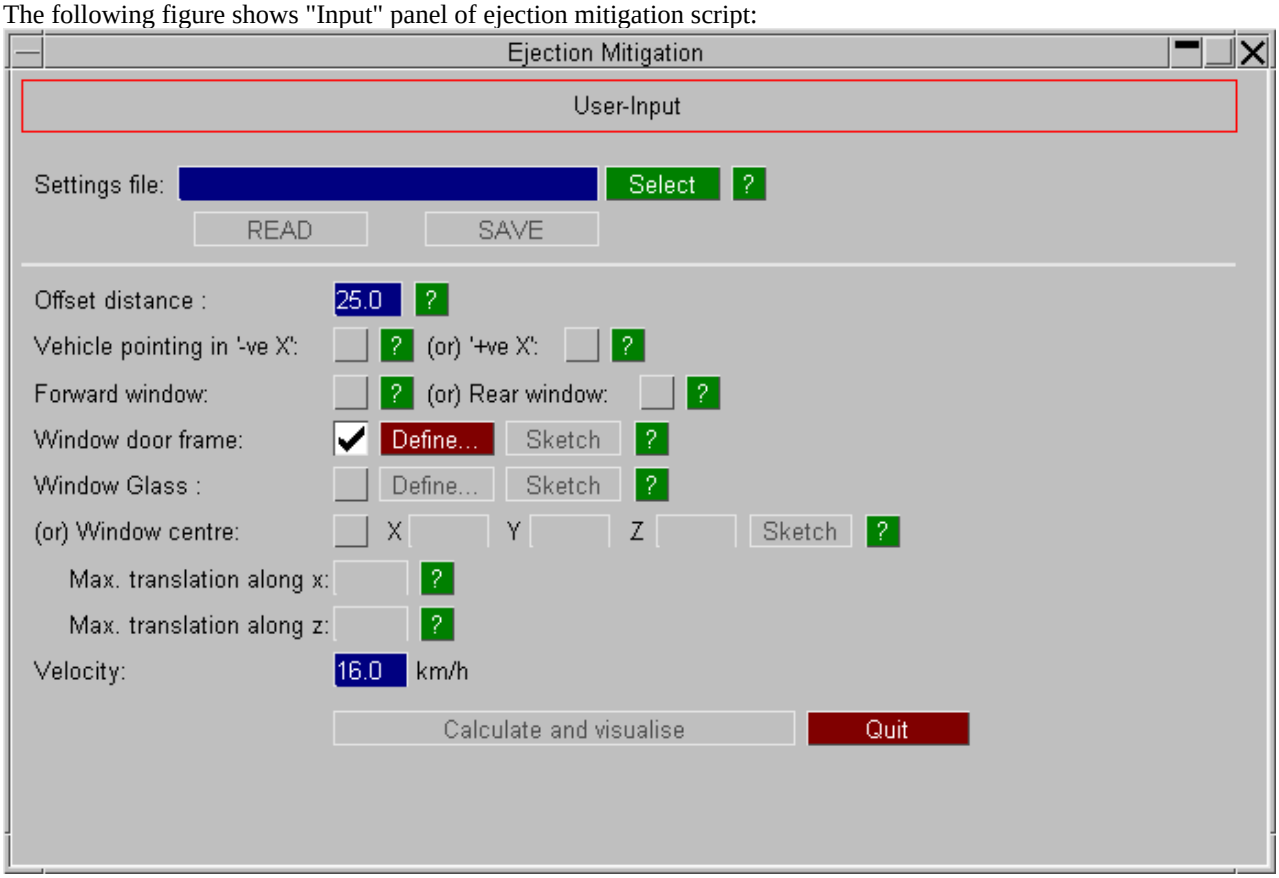
They can be imported back into the original model using PRIMER following the steps described here.

6.16 EJECTION MITIGATION SCRIPT

Introduction

The script calculates impact points on a vehicle model for the FMVSS226 specification.

The following figure shows "Input" panel of ejection mitigation script:

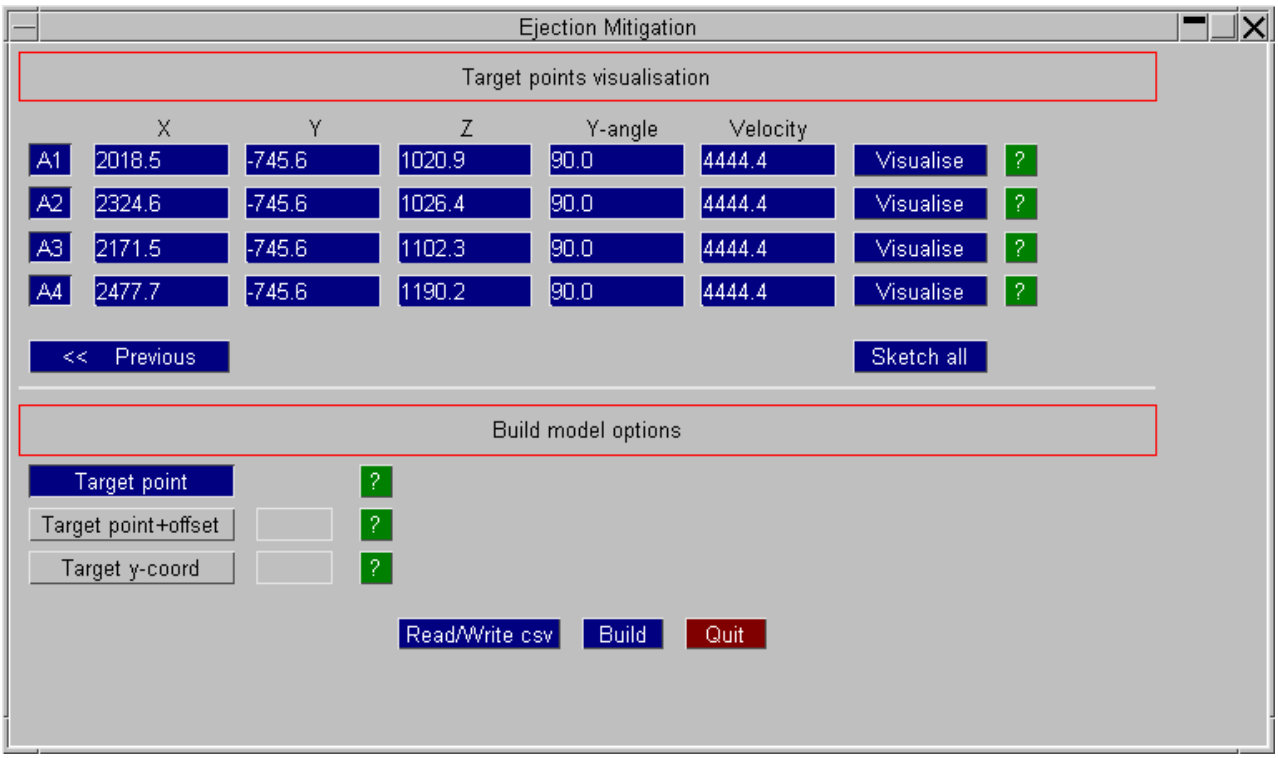


The following options are available on the input panel:

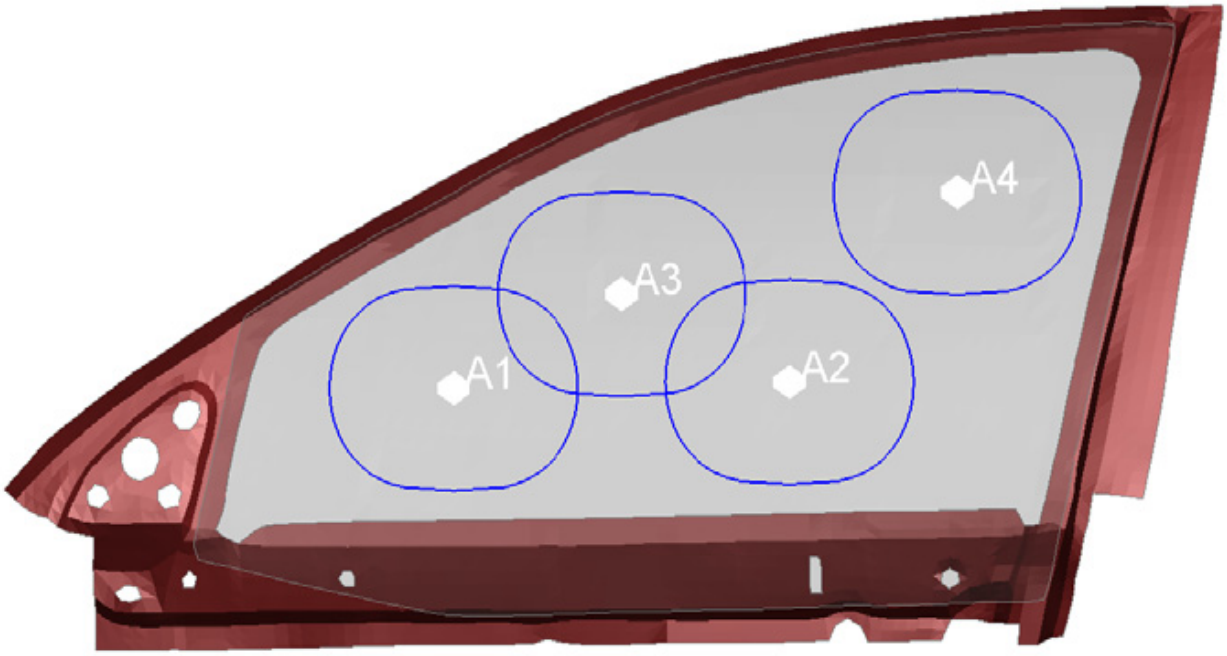
Settings file	A settings file may be read or written which may include mandatory information and other data required to compute standard impact points. The reading/writing of a settings file is optional.
Offset distance	The offset distance is the distance away from the edge of the daylight opening used when positioning. By default this is set to 25mm.
Vehicle direction	Specify the vehicle direction (either +ve X or -ve X).
Window selection	Specify impact points to be calculated for (either the forward or rear window).
Window glass or centre	Either specify a PART/PARTs that surround the daylight opening or the approximate centre point of the daylight opening and maximum translations in X, Y and Z (set the max values to approximately the size of the daylight opening).
Velocity	Velocity is set to 16 km/h by default.

Impact points visualisation

Once all the input conditions have been defined, click **Calculate and visualise** to start the calculation. After the calculation, impact points (or target points) are listed as shown below:



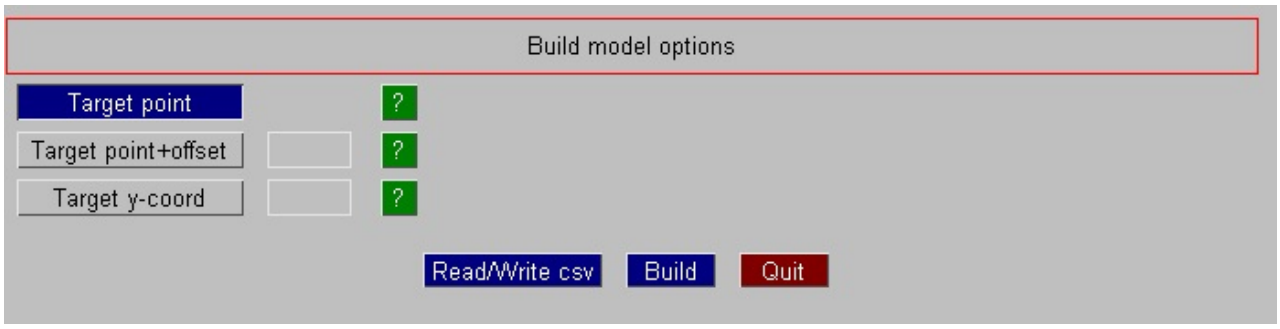
They can be visualised individually using **Visualise**, or all at the same time using **Sketch all** (see below) .The help **?** buttons will give you more information on the target points.



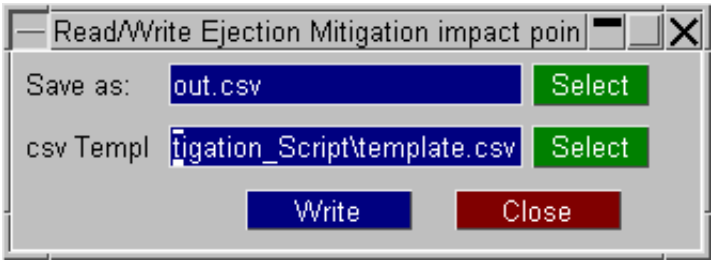
Export impact points to csv file

There are a number of options for writing the coordinate to a CSV file:

1. Use the Y-coordinate of the target point found (centre of the daylight opening) .
2. As above but with an offset towards the centre of the vehicle
3. Specify the Y-coordinate to use



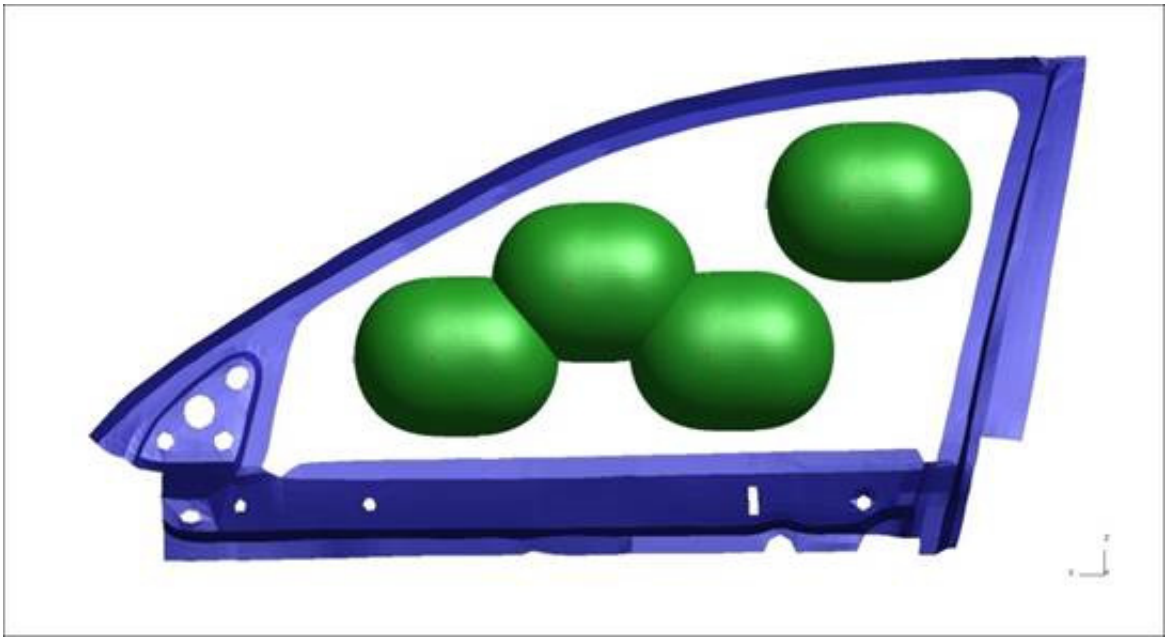
Clicking the **Read/Write** csv button opens a new window (shown below) that allows users to export impact point information to csv files.



A template file may be specified that permits specification of additional information such as model, impactor, and contact.

Setting up impact case

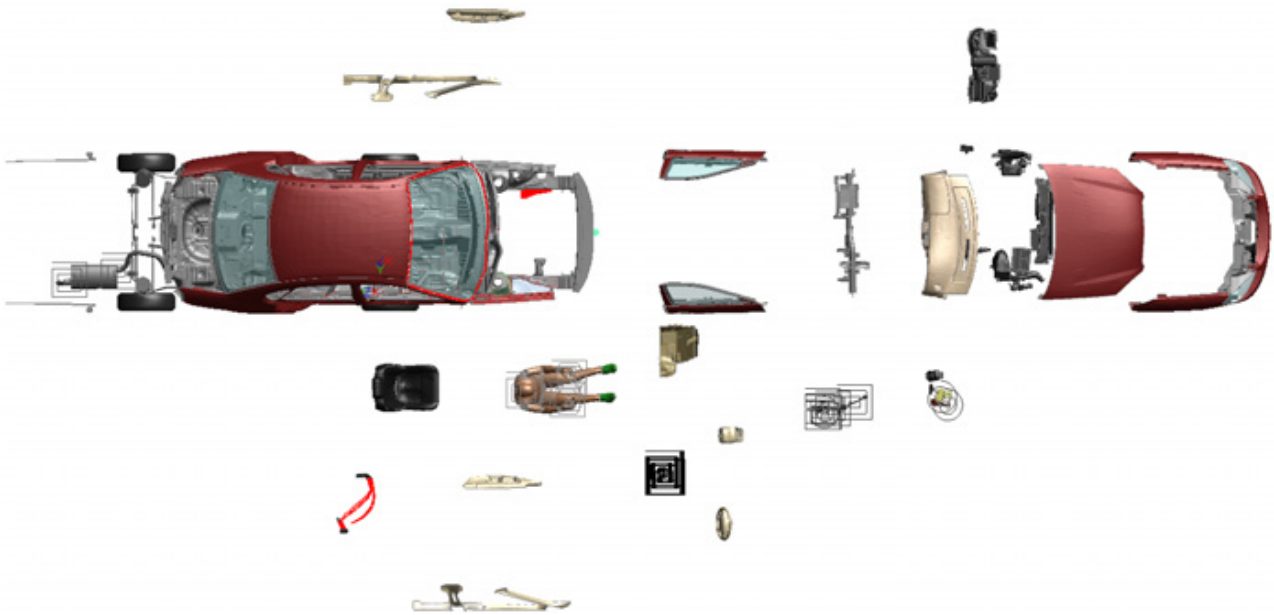
PRIMER's automatic build process can be used to position an impactor at various impact points. This can be done by clicking the 'Build' button at the bottom of the script window. Alternatively, the same can be done by selecting the 'Build from csv targetting file' option under Build in Primer's Model tab. Additional information may be obtained from Appendix XIV.



6.17 EXPLODE

Explode

The Exploded View function allows chosen entities of a user-selected type to be moved away from other entities of that type while being maintained as choate blocks. Supported entity types include parts, part sets, includes (seen below), and part tree assemblies.

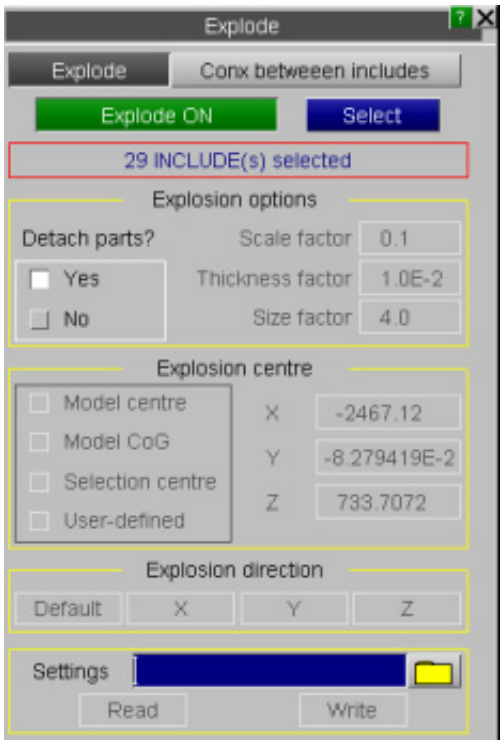


Two modes - part-based and node-based - are available:

Part-based explosion will cause parts that are meshed together to move apart while ignoring shared nodes.

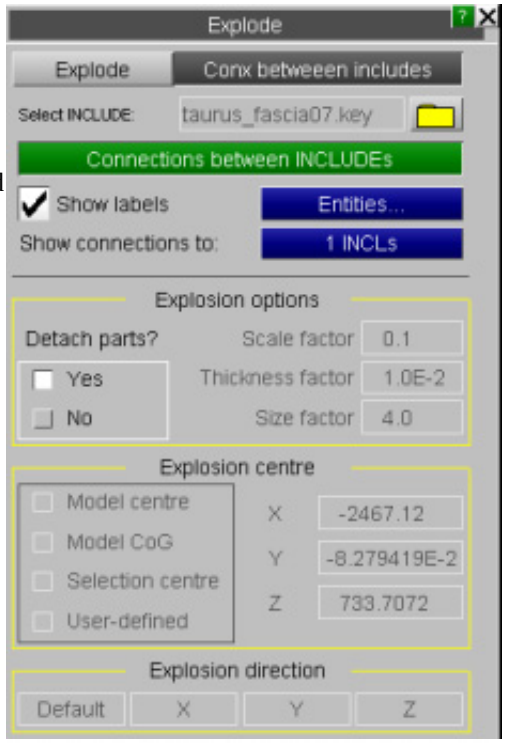
Node-based explosion may cause parts that are meshed together to deform to account for the shared nodes.

Options to control the Explosion volume, centre, and direction are available.



Connections between includes

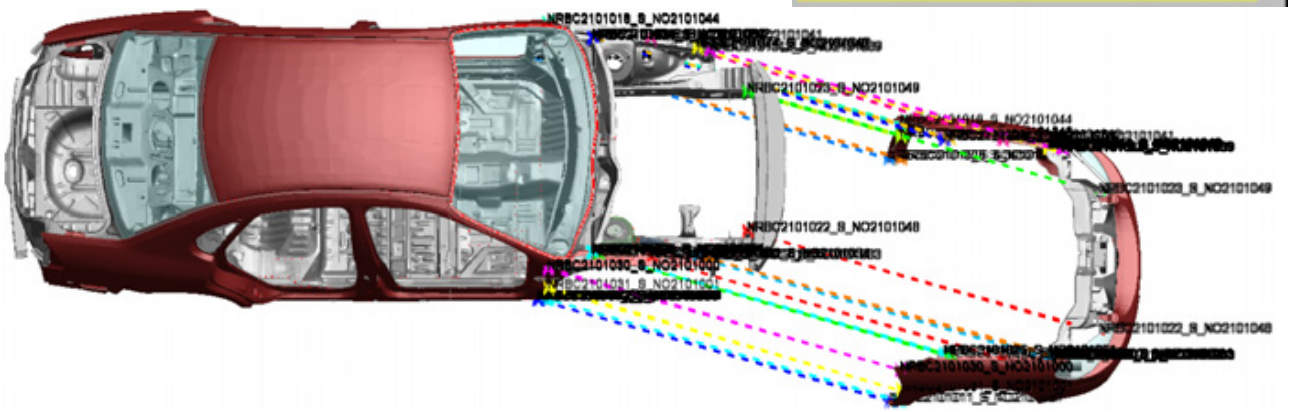
The **Conx between includes** tab in the Explode menu can be used to visualise connections between a selected includes and other includes in a model. This can be achieved by selecting an include and clicking **Connections between INCLUDEs** in this tab. Connections, in this context, include constrained items, elements, and tied contacts.



Relevant includes containing said connections are automatically exploded away from other includes, which are then hidden from view.

Specific includes may be chosen as candidates for said connections.

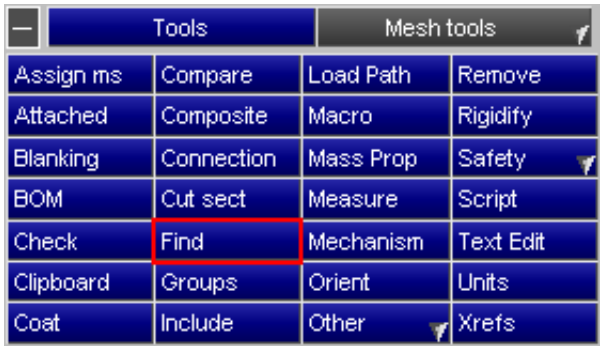
Connection labels may also be turned on/off.



6.18 FIND AND SKETCH

The **Find** tool allows you to locate an item in your model.

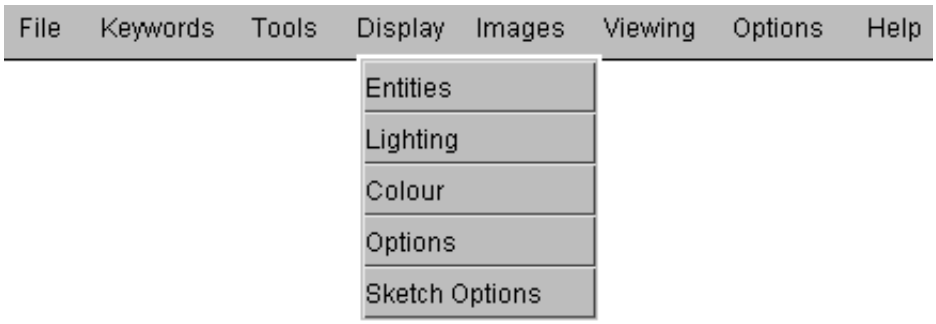
The features developed for this function have been generically incorporated into PRIMER's sketch function when it is applied to a single item



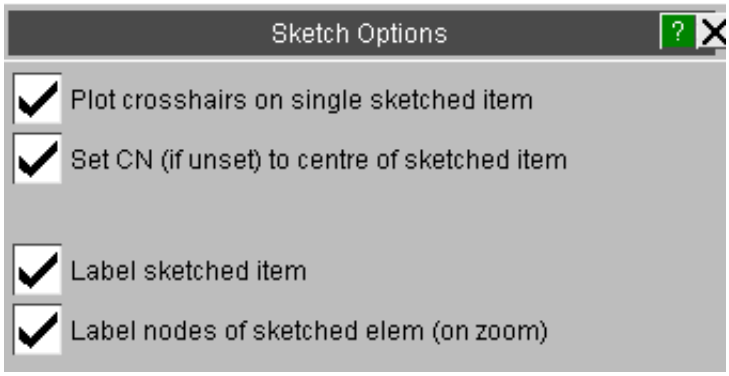
6.18.1 Sketching a single item

There are many ways in PRIMER to sketch items. For example
- sketch function available to each keyword
- sketch off object menu
- sketch on drop-down available in many contexts
- quick-pick sketch using id key in

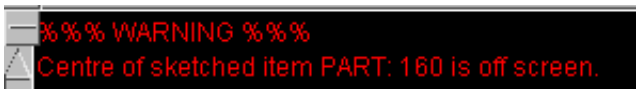
If a **single item** is being sketched this will be done according to the sketch options under Display.



By default the following actions will be taken in addition to the previous sketch function. These options may be switched off.



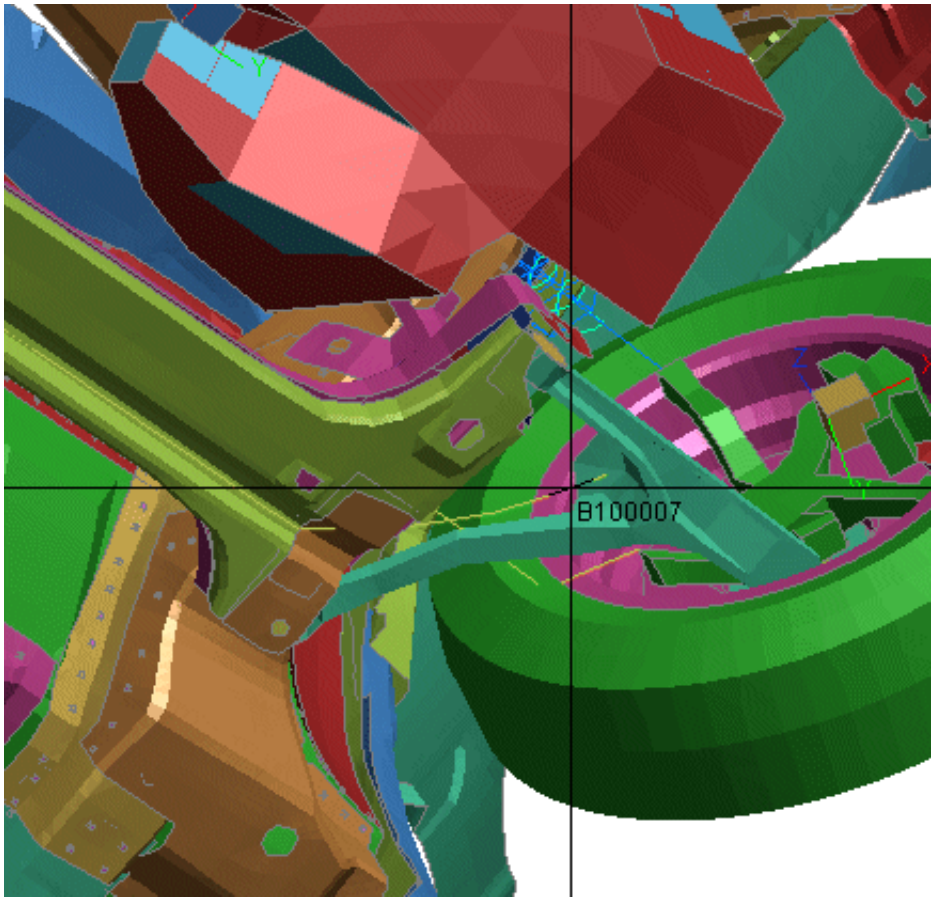
Plot crosshairs Draws crosshairs centred on the sketched item. This is particularly useful if the item is very small and the normal sketch cannot easily be seen. If the item is off screen a warning is given in the dialogue box.

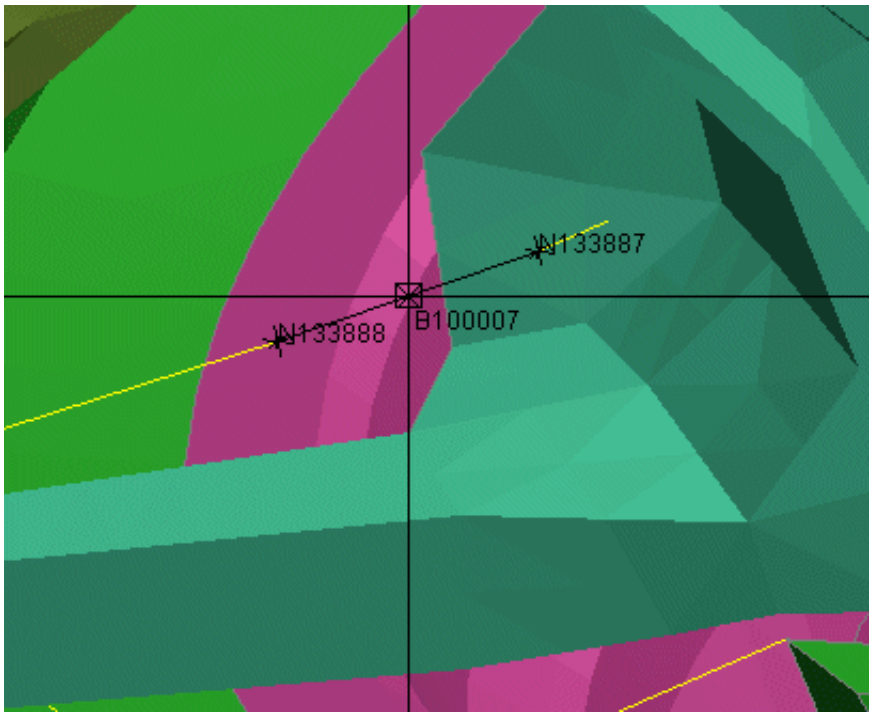


Set CN This option will not apply if the user has pressed CN in the view control panel. Otherwise, the centre of rotation/zoom will be set to match the cross-hairs if the sketched item is on screen. The option will automatically activate if an off screen item is brought into view and similarly de-activate if an on screen item is moved out of view. Pressing **CN** will unset the centre.

Label sketched This will label the sketched item in the form Mn/Sn.

Label nodes of sketched elem This will label the nodes for some items if the zoom is deemed sufficient to show them.



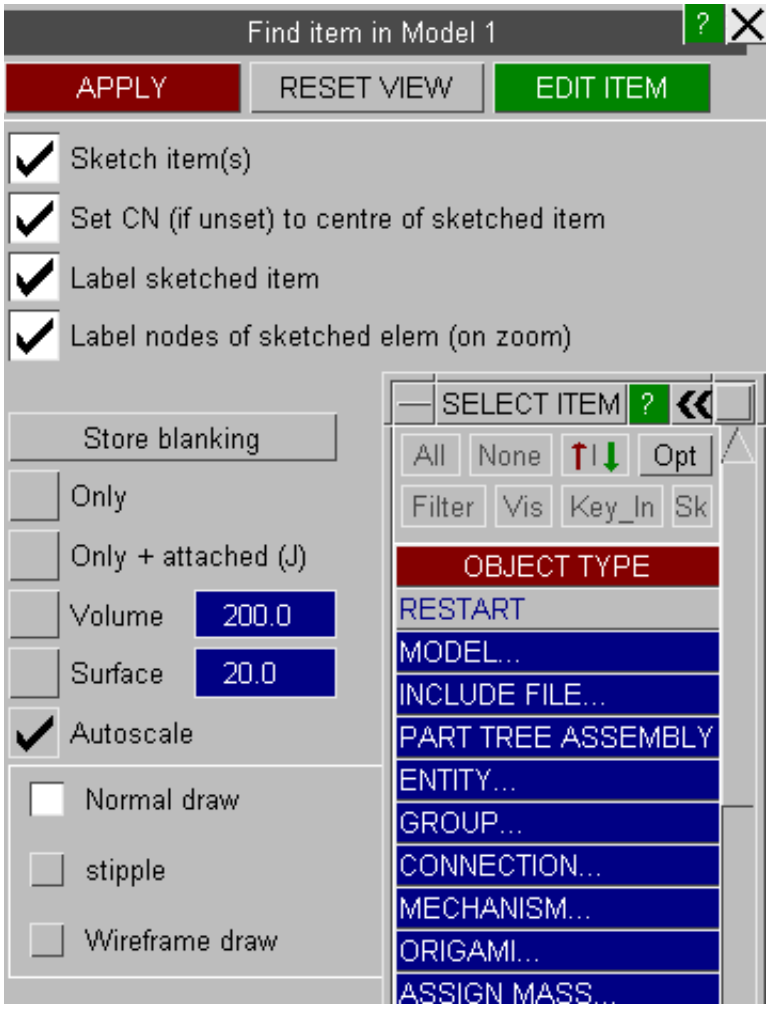


Press of Delete key or redraw by using Li/Hi/Sh will remove the cross-hair, clear the sketching and unset the CN.

If you sketch multiple items simultaneously, Primer will work as before and only apply a sketch.

If a cross-hair unexpectedly does not appear when sketching from object menu, it is most likely that more than one item is selected. Try clearing the selection with **None** and then reselecting.

6.18.2 Finding an item

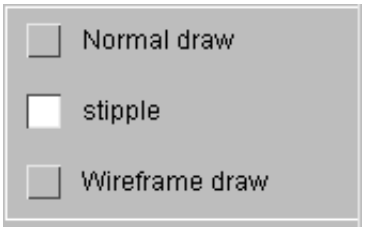


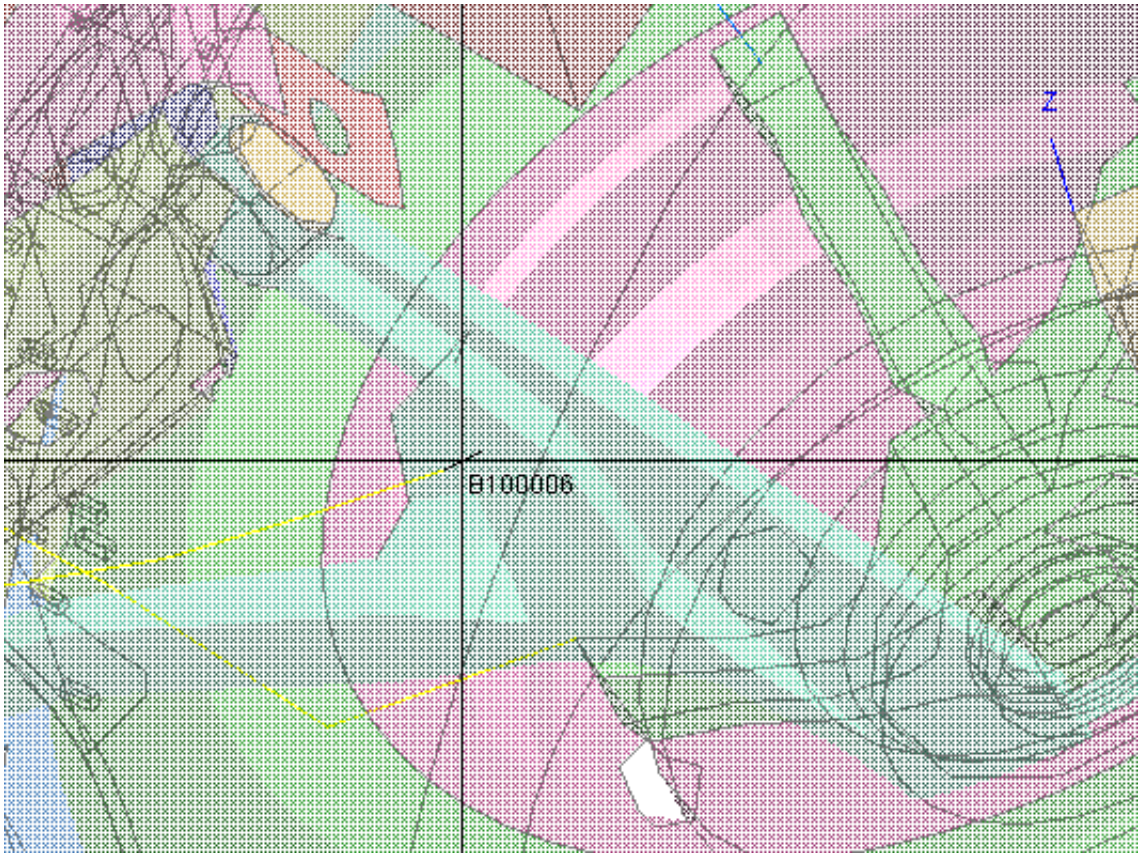
In its default mode, **FIND** is a sketch function for a single item with a generic object menu. The sketch options may be set exactly as described above.

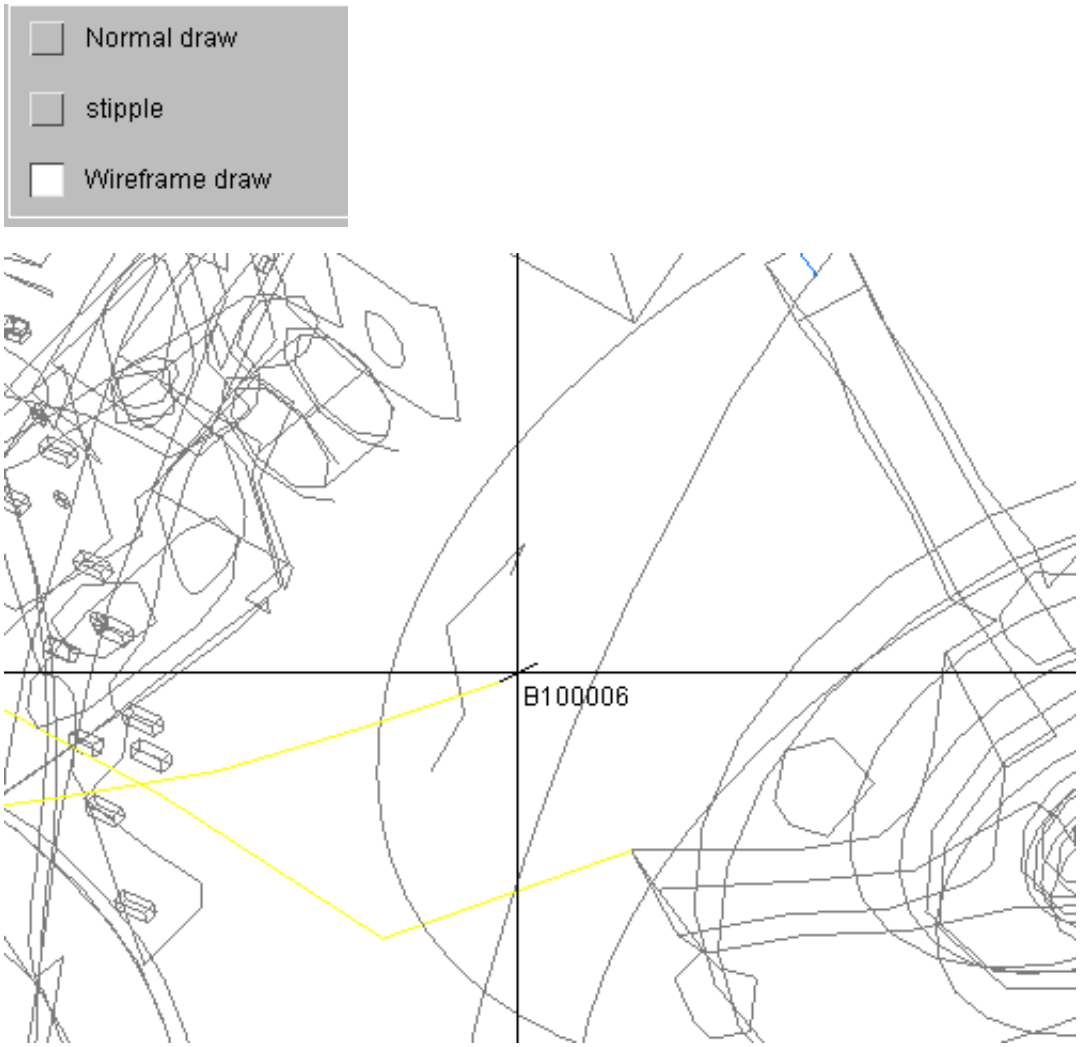
FIND also offers a number of additional features.

Stipple and wireframe draw

The drawing mode, normal draw by default, can be set to use stippled draw (a form of transparency) or wireframe draw to enable the sketched item to be seen. This is useful if the item is enclosed.





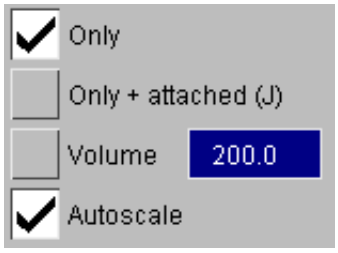

Normal draw
stipple
Wireframe draw
B100006

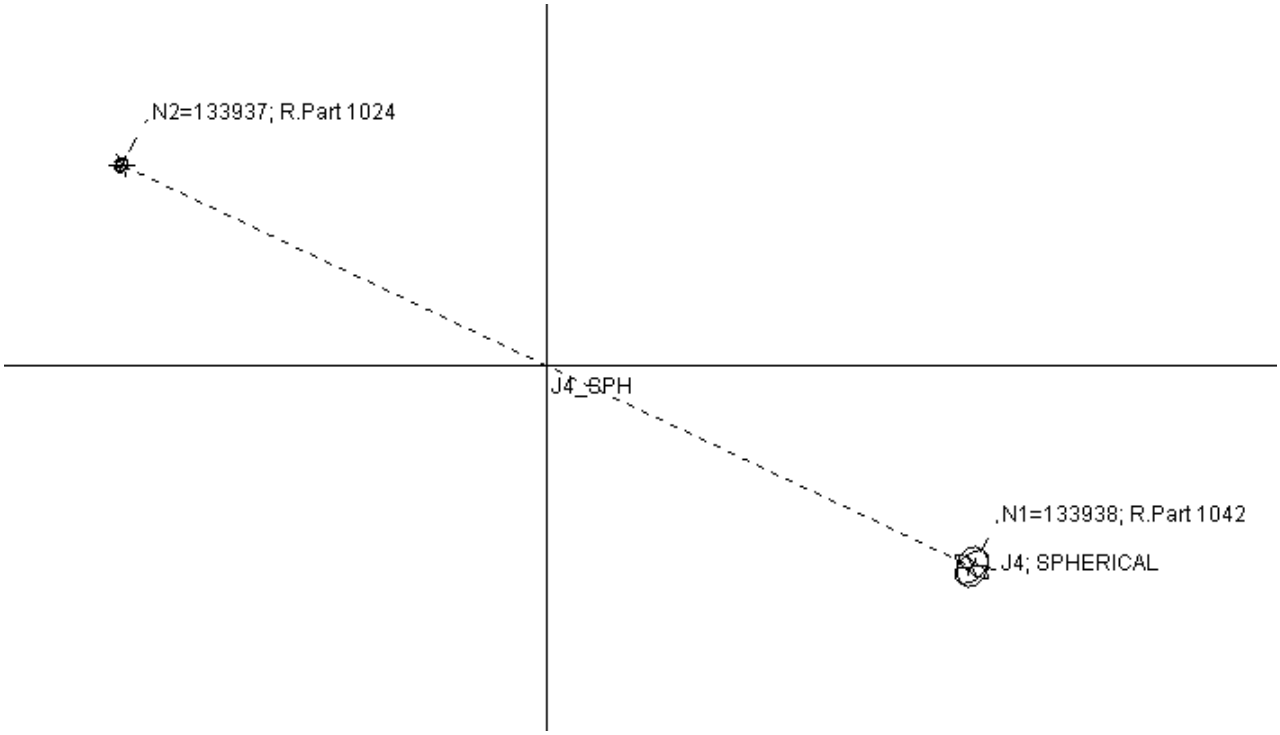
Find with Only

Before applying automatic blanking you may wish to store the current blanking status of the model by a press on **Store blanking.** On completion of the Find operation you can use **RESET VIEW** to restore the image.

3 methods are available Only, Only with attached and a Volume clipped view. By default autoscale will be applied but this can be de-activated.

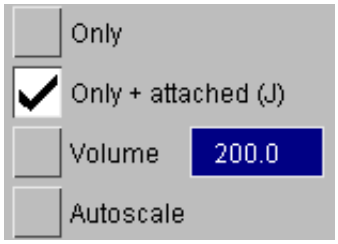
Only on an ill-conditioned spherical joint gives the following image.

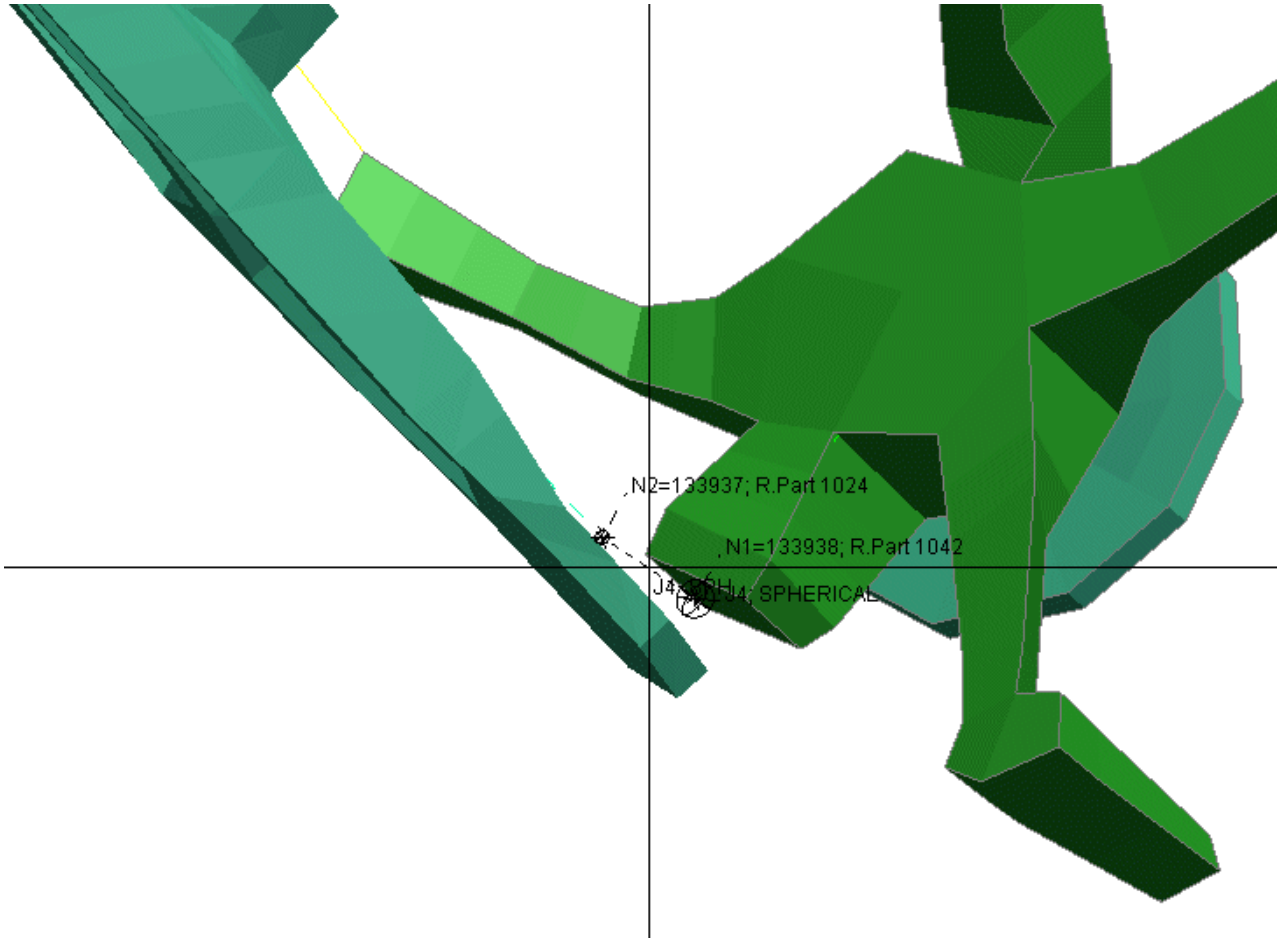




Find with Only & Attached

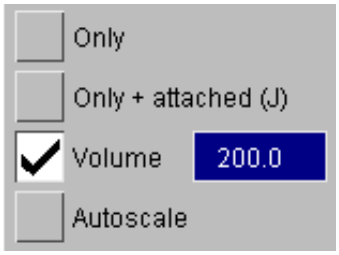
Switch to Only + attached to find the rigid bodies on the joint

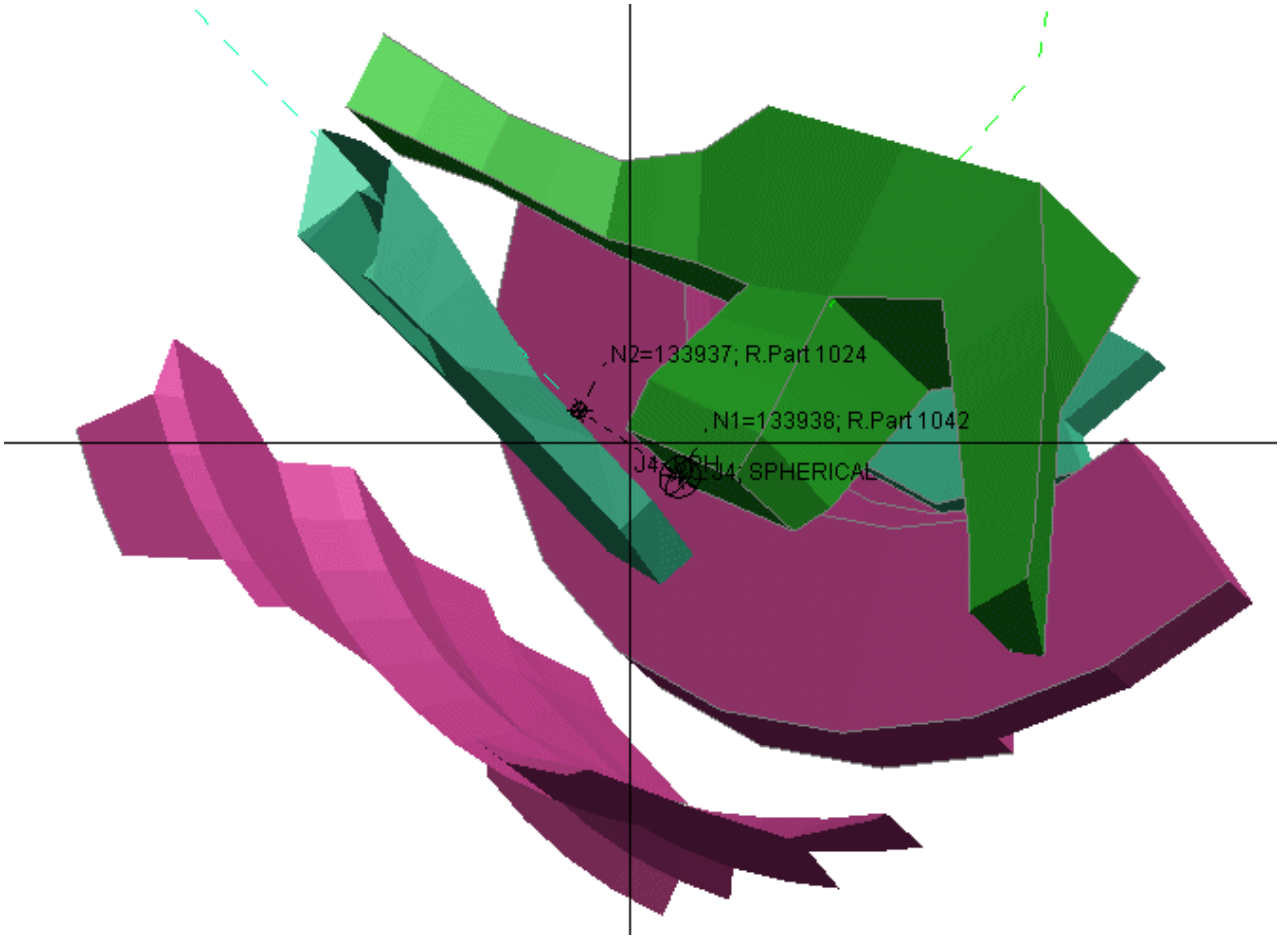




Find with Volume Clipping

Volume clip view will show everything that is attached to nodes that lie in the spatial volume NxNxN centred on the selected object, where the value of N is controlled by the user. This is useful when the selection consists of an NRB, a single beam element or a small part. If, for example, all wheels of vehicle shared the same part, selecting that may not be helpful.

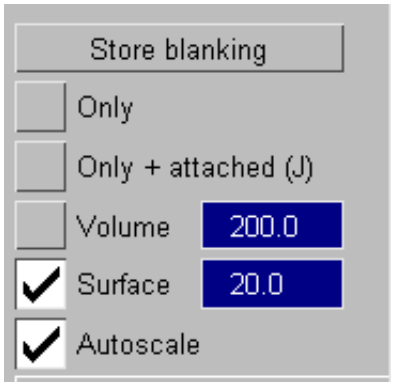


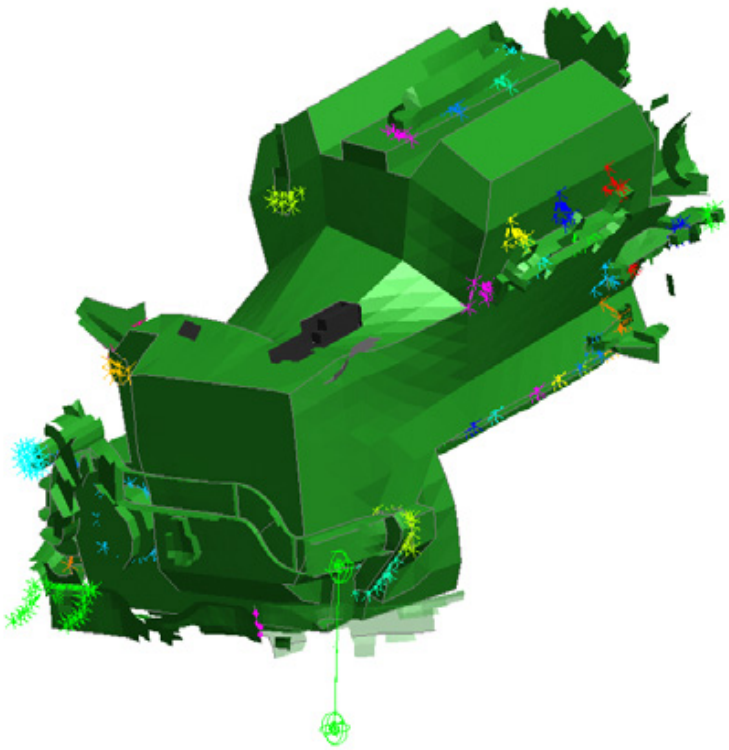


Find with Surface Search

Surface search can be applied to solid or shell part and will find all items which lie withing the search tolerance.

Comparison with results of attached search is useful for finding items which should attach but do not as yet..





Editing with Find

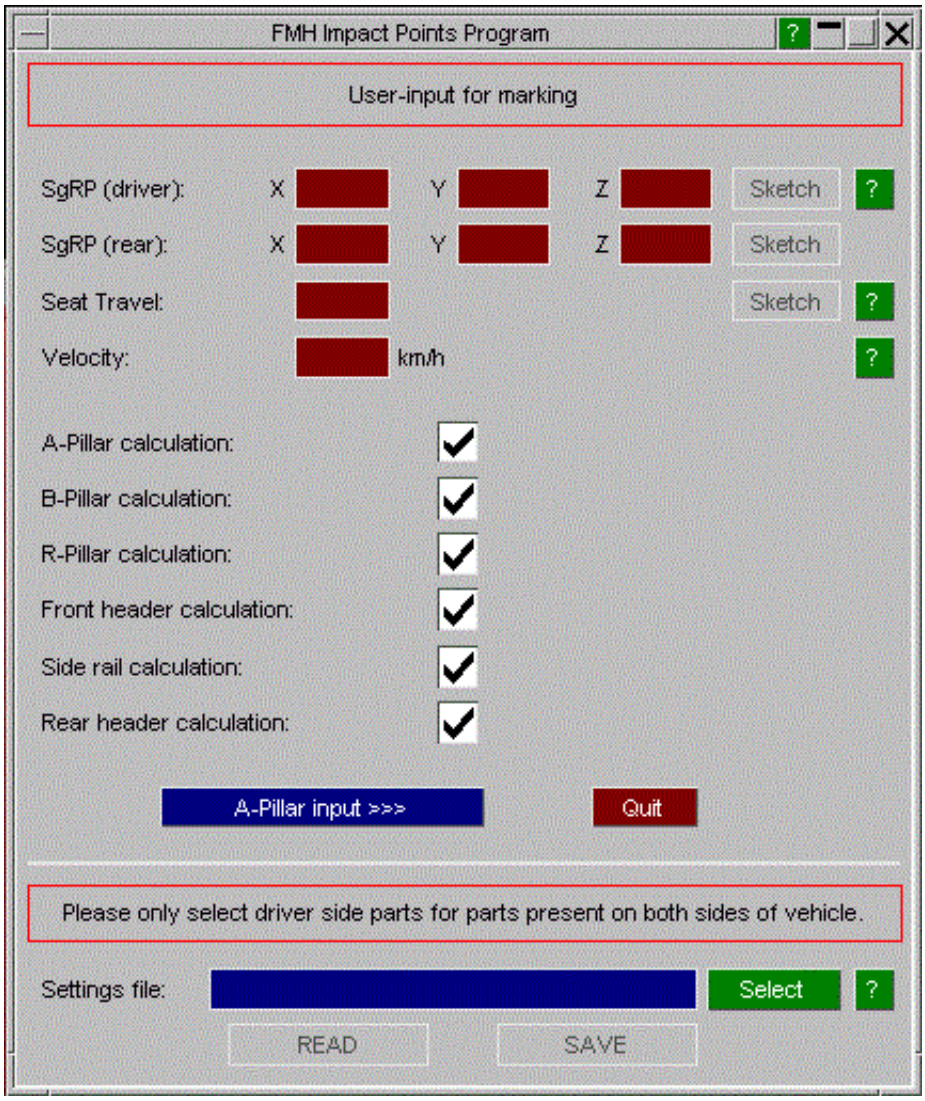
If there is an active item as displayed on the find header, EDIT ITEM may be used to open an edit panel or keyword editor.

6.19 FMH Free Motion Headform

- FMH Markup Script
- FMH Manual Setup

6.19.1 FMH Markup Script

This feature computes interior impact points on a vehicle model for the FMVSS201 specification. It also helps create multiple models corresponding to these points. Additional user-defined points may also be specified.



The model must be aligned with the global co-ordinate system as follows:
- +X from car front to back (vehicle points towards -X)
- +Y from car left to right
- +Z from the ground up

A settings file may be read or written which may include mandatory information and other data required to compute standard impact points. Mandatory information includes:
- Seating reference points
- Seat travel
- Impact velocity

The settings file is useful as it means the information required for the calculations need only be selected once. It is recommended that after specifying all required information, but before calculating the target points you return to the first panel and save a settings file.

Specific impact point categories such as A-Pillar may be selected/deselected. Information specific to these categories may be specified manually or imported via a settings file.

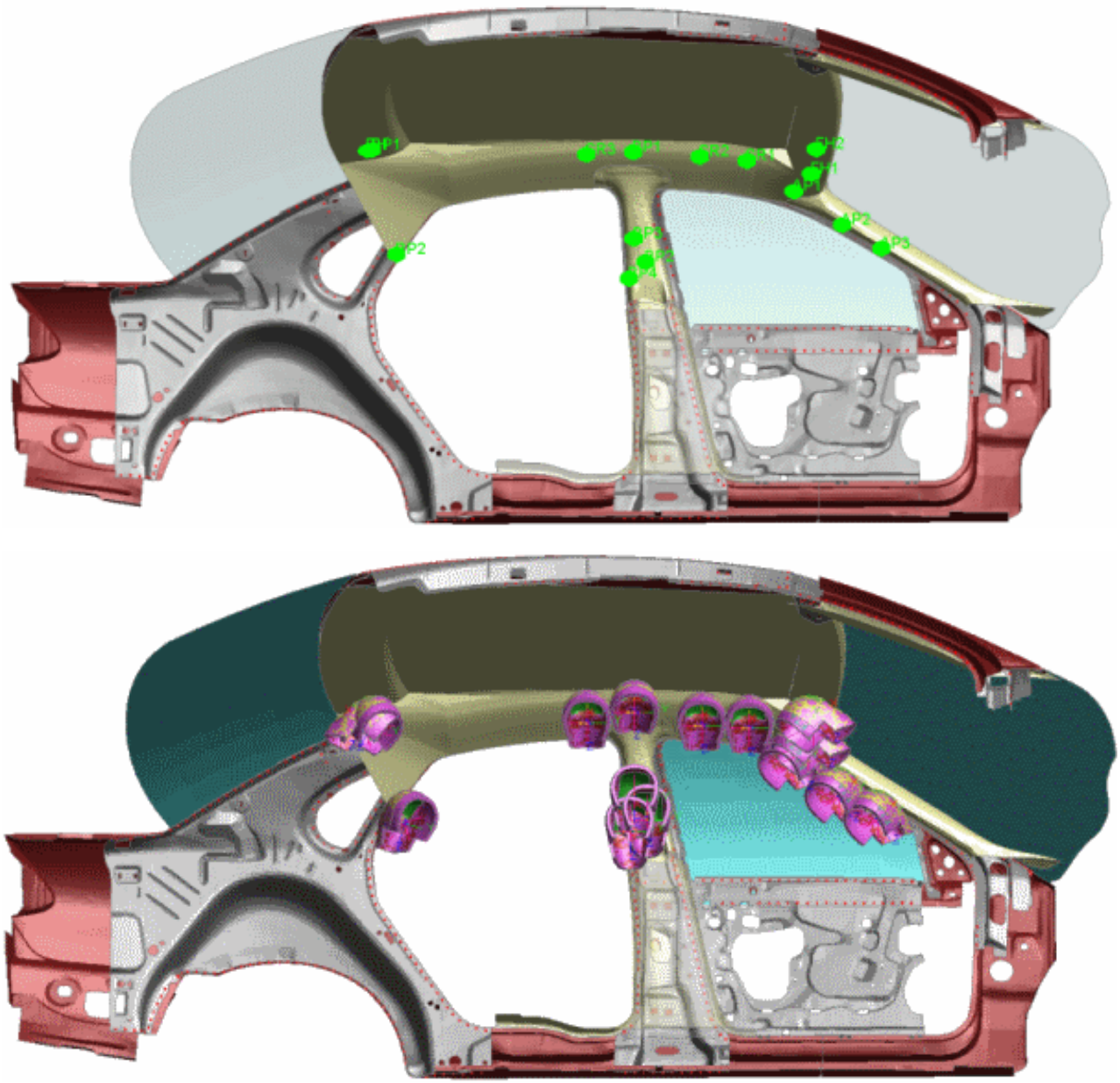
The ’Impact point visualisation’ screen provides information about standard impact points including position, approach angles and velocity. This information may be modified using appropriate text boxes. Additional information regarding this calculation may be obtained using appropriate ’Visualise’ and ’?’ buttons.

By default the vertical angle is set to AUTO. With this setting, PRIMER will use an iterative process to determine what the maximum vertical angle is, by rotating the headform until the chin of the head touches the vehicle trim, and then rotating back. This process can take some time, so it is recommended to change the AUTO to a specified vertical angle if you already know this information.

User-defined points may be created by clicking the **Create manually** button.

A csv file may be written by clicking the **Read/Write csv** button.

Finally, multiple models may be created at the selected points by clicking the **Build ...** button.



6.19.2 FMH Manual Setup

This feature has been written in order to position the freemotion headform according to FMVSS 201. Firstly at least one ***HEADFORM** definition must have been read in from file. The ***HEADFORM** card is similar to the ***DUMMY** definition and contains a number of keywords (described below) with information required by the positioner. These appear after the ***END** card and are ignored by LS-Dyna but used by PRIMER. An example of a headform tree file is given in appendix X. The corresponding target and position tree file example is available in appendix XI.

***HEADFORM_START**

The headform label and title.

***REF_POINT**

This is a node label, already exiting in the model, about which the headform will be rotated.

***UNITS**

The mass, length and time units used in the model (same options available as for a *DUMMY definition).

***COMPONENTS**

The part set which makes up the headform definition, the part on the headform to be used in the contact definition and the label of this contact definition.

***TARGET**

The target definition at which the headform is currently positioned. Blank if no target definitions exist in the model or the headform is not currently in position.

***AXES**

The label of a ***DEFINE_COORDINATE_NODES** definition already existing in the model to define the headform local co-ordinate system.

***HEADFORM_END**

The end of the headform definition.

Along with the ***HEADFORM** definition another keyword has been included to store information regarding the target points in the model.

***TARGET_POINT_START**

The first line contains a label, an acronym (as defined in FMVSS201) and an optional title. The second line contains the co-ordinates of the target. The third line contains the minimum and maximum horizontal angles. The fourth line contains the impact velocity for this target point, the part set to be used as the slave side in the contact definition and the current headform position number (see below).

***HEAD_POSITION**

At a given target point a number of different angles are normally investigated. Any number of unique positions can be stored with each target point to facilitate moving the headform about in the model. This keyword contains a label and a title, the co-ordinates of the headform reference point, the horizontal and vertical angles, a flag to indicate where the horizontal angle is in the allowable range and a positional node ID.

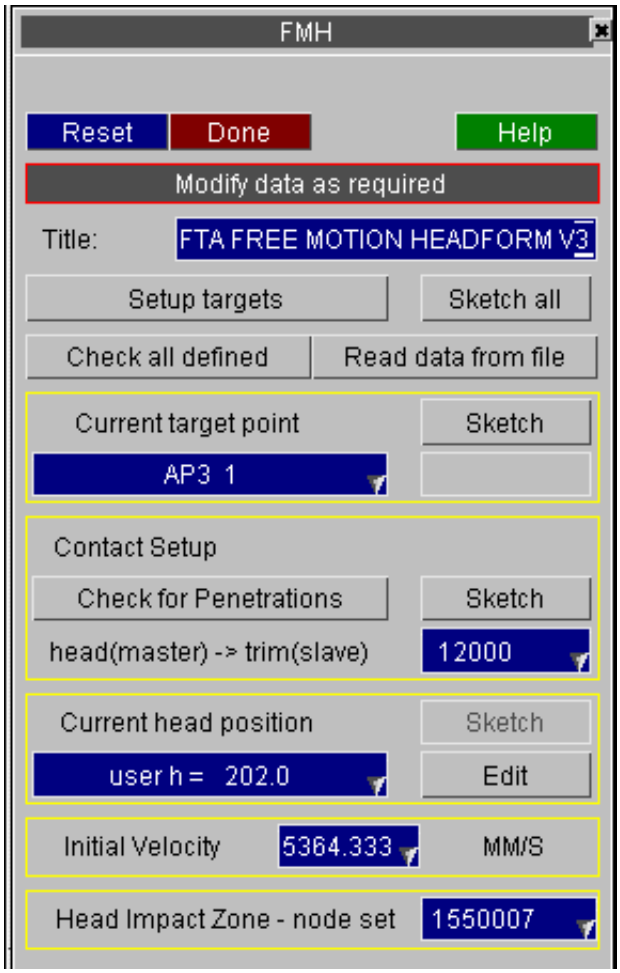
***TARGET_POINT_END**

Ends the target point definition.

6.19.2.1 Positioning the headform

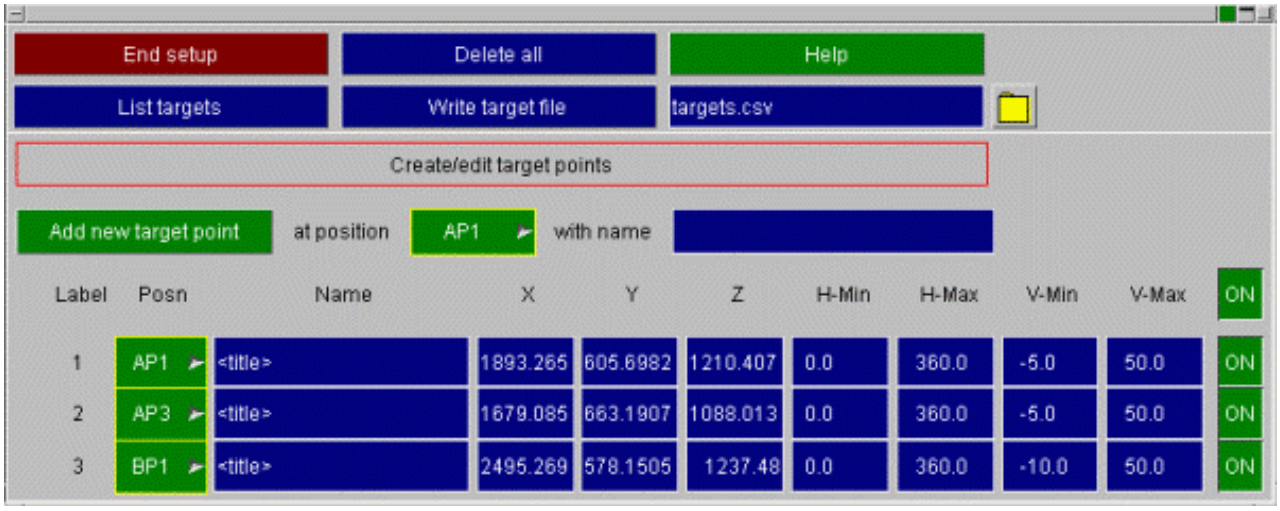
The figure below shows the main headform-positioning panel. This has been designed for use from top to bottom to create a run-ready input deck. A target definition must exist in the model in order to be able to position the headform. This can be created with the **SETUP_TARGETS** button.

If multiple headform definitions exist in the model the definition to be positioned must first be selected.



6.19.2.2 Setting up Targets

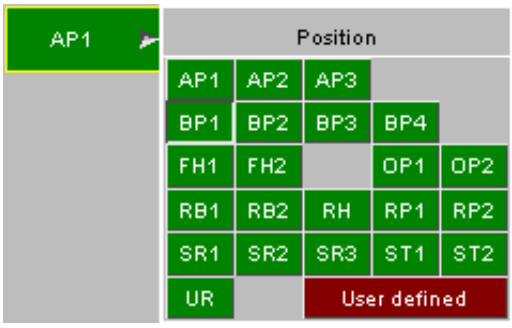
The **SETUP_TARGETS** button accesses the model target database.



You can have as many target points in your model as you want. A scrolling list shows all of the points. Additionally you can have more than one point at the same position.

When a target point has been assigned a head position it's target file button (on rhs) will become active. You can then write a csv targetting file which can be used by the command line build models from csv file function (see Appendix).

Adding a new target point



To add a new target point:

1. select the position name you want the point at using the popup menu shown on the right
2. enter a description/name for this point
3. Press **ADD new target point**.

The point will be added to the list of available target points. You can then modify the point coordinates and min and max angles.

Removing a target point

To remove a target point use the popup on the required target point and press **Remove target point**. If the target point is currently in use by a headform you will be asked to confirm removal of the point

Sketching a target point

To temporarily sketch/draw a target point on the screen use the popup on the required target point and press **Sketch target point**. If you want to see the target points at all times then you want to turn target point drawing on instead of sketching them. See drawing and labelling target points below.

Changing the position of a target point

To change the location/position of a target point either:

- type the new X, Y and Z coordinates into the test boxes.
- use the popup on the required target point and press **Pick new location**. You can then select a node from the screen. The coordinates will be taken from that node

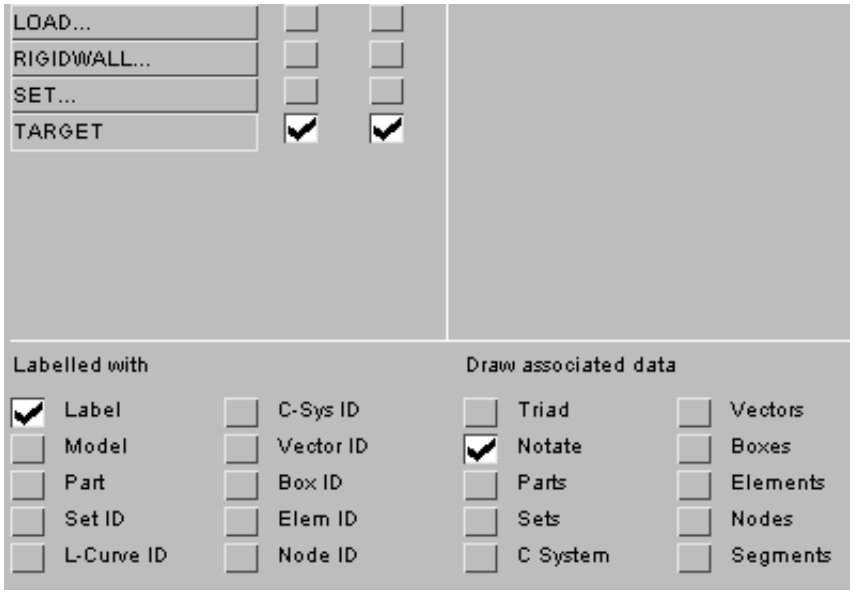
Changing the size of target points

The size of target points can be changed in the main **OPTIONS** panel.

Drawing and labelling target points

Target points can be drawn and labelled just like nodes and elements. They are turned on in the **ENT**ity Viewing .

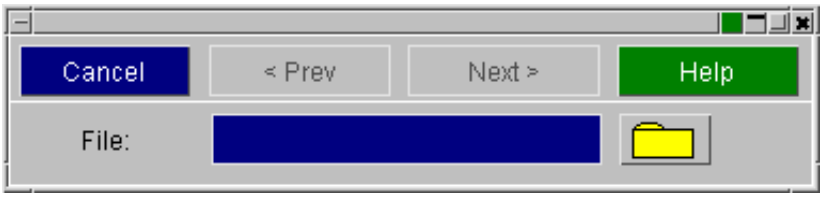
In addition to drawing and labelling target points the **NOTATE** function can be used. If this is turned on then the name and position of each target point is written on the screen as well as the target point number.



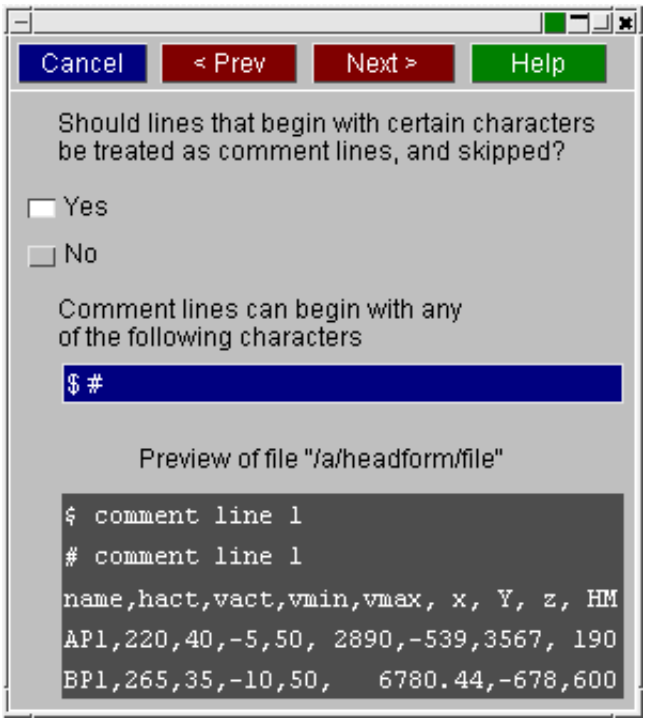
6.19.2.3 Reading headform position data from a file

You can read in headform position data from an external delimited file by clicking on the **Read data from file** button. This will guide you through a series of panels where you can specify the file type and what data you wish to read from the file. The file would generally be of a CSV format with each row containing information for a target point/head position. The sequence of panels is:

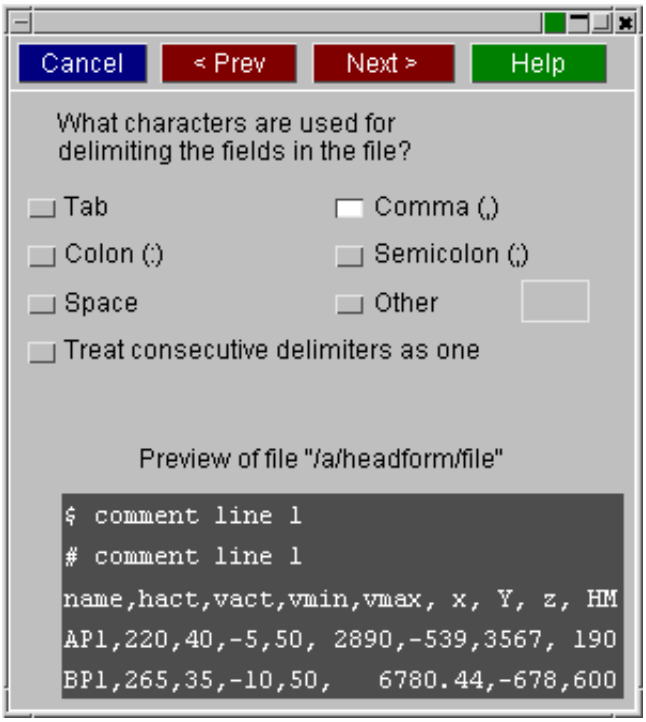
Select the file to read.



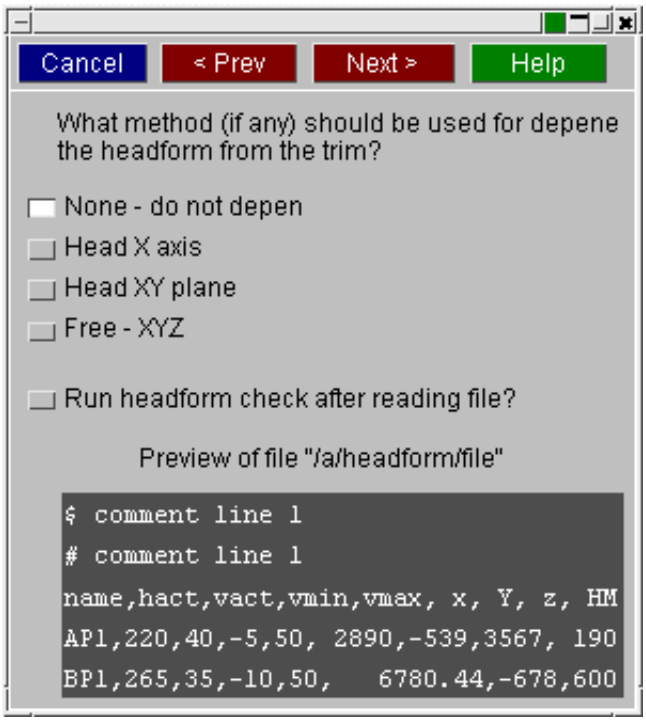
Specify any lines to ignore by defining characters at the start of the line that signify a comment. Note that a preview of the file is shown at the bottom of the panel.



Specify the delimiter for the data in the file.



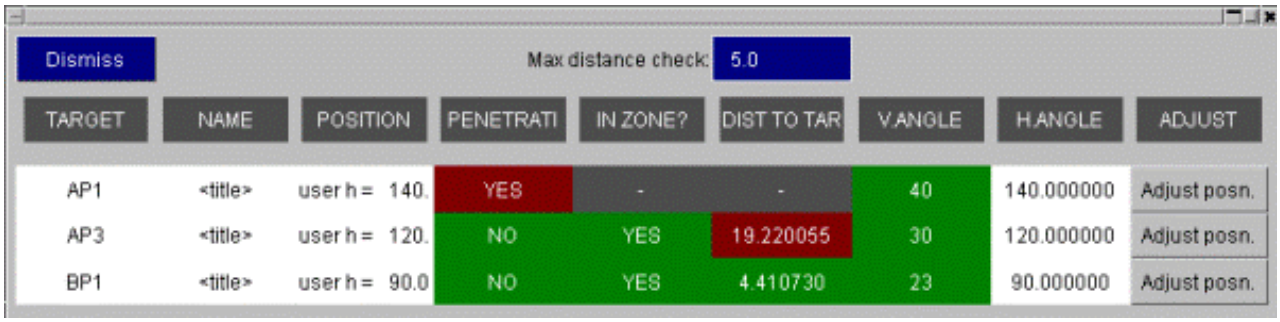
Choose depenetation options for when the data is read in. You can choose to automatically depenetrate the headform from it's starting position in different dergrees of freedom. You can also choose to run the headform checks after reading in the data (see section 6.13.4).



Finally the data is presented to you in a table format. If there where suitable titles in the input file, Primer will have attempted to guess the type of data in each column. If not, you can specify this on the panel by right clicking on the column headers and choosing the type of data from the resulting popup. After the columns have been assigned, click on **Apply** to read in the data and setup headform position information from the data. Note that the minimum that has to be contained in the file is the x, y, z coordinates of the target point.

6.19.2.4 Checking defined targets

You can check the status of all currently defined target points/positions by clicking on the **Check all defined** button. This will check each position in turn and then report the results to the screen in a table.



The check panel reports a number of things:

TARGET - The target point.

NAME - The target point name.

POSITION - The position within the target point definition.

PENETRATIONS? - Reports whether the current position of the headform means there are penetrations between the headform and the trim.

IN ZONE? - Reports whether the initial contact point of the headform to the trim is within a user defined head impact zone (see below).

DIST TO TARG - Reports the distance from the initial contact point to the defined target. Note the **Max distance check** value above this column determines whether the value in the table is shown in red or green.

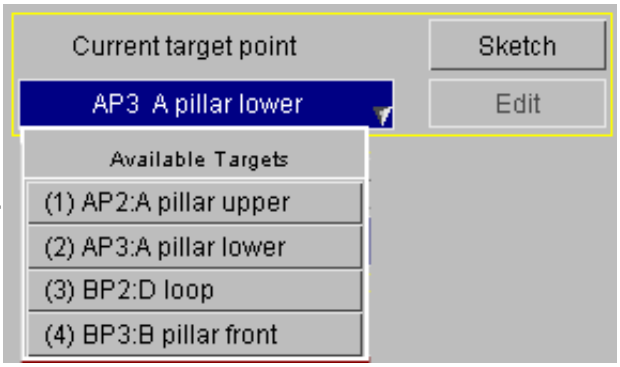
V-ANGLE - Current vertical angle of the headform. This is checked against defaults or user defined values on the target setup panel.

H-ANGLE - Current horizontal angle of the headform.

ADJUST - Opens up the position editing panel specific for this position on the table. This allows you to modify the position. After clicking **Done** on this panel you are returned to the check table and the details for this row are updated.

6.19.2.5 Selecting a target

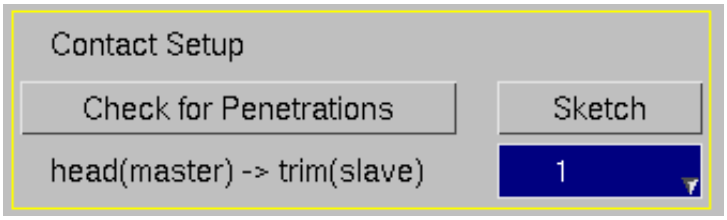
Clicking on the red button on the left-hand side activates a target point. Clicking on the button when it is green will remove the target point. Once active the required information can then be input and the target selected for positioning the headform via the popup on the main positioning panel.



The **NODE** buttons on the target database panel allow the user to select a node in the model from which the co-ordinates are taken for use by the target.

When a target point is selected for use the headform is moved to that target point with the headform reference node located at the target co-ordinates. If a position definition (see below) exists and has been previously selected, the headform angles will be determined by that otherwise it will default to 0º vertical angle and the target minimum horizontal angle.

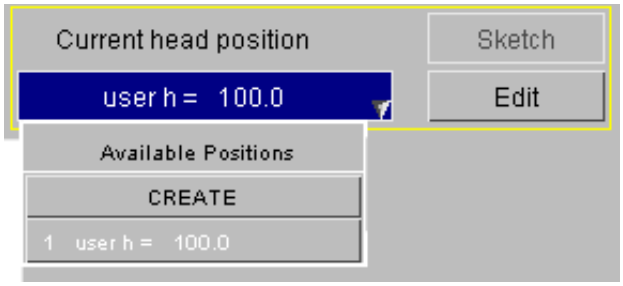
Next the contact needs to be set-up. The contact definition exists in the headform tree file, under HEADFORM_START -> COMPONENTS. This contact must be created or edited to suit the headform. It is best to use an automatic surface-surface contact defined by part set, with the master side defined on the headform.



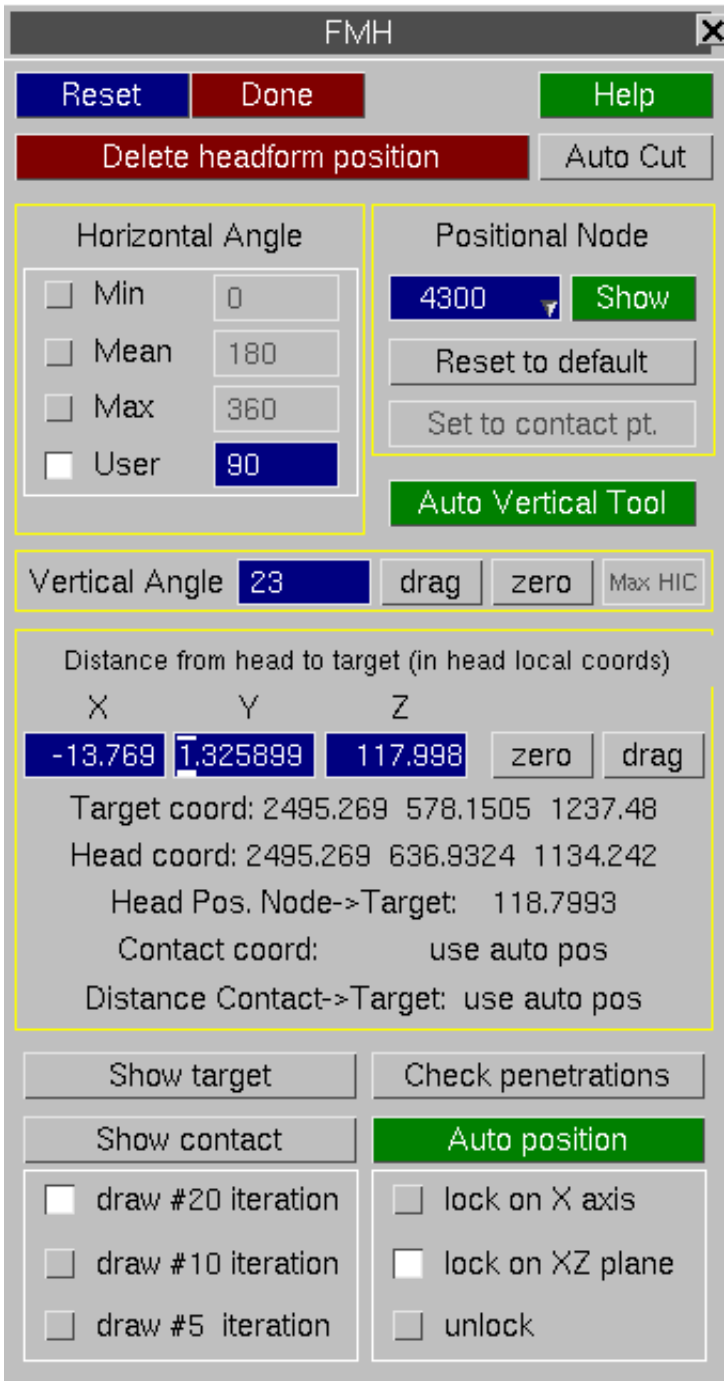
The slave side of the contact may require editing so that it contains the correct parts for each position. Alternatively a set can be defined with all the relevant parts and a box added to the contact. As the head is moved to different positions the box will be translated with it. In FMH mode the function which enlarges boxes as they are rotated is suppressed as this was found to enlarge the contact box excessively.

The contact can be edited and sketched in the usual way and also checked for initial penetrations. Those elements that are within a shell thickness apart or actually crossing over are sketched in white.

The headform is now positioned relative to the selected target point. This requires a position definition to be created. If position definitions already exist within the model for the selected target point then they can be selected via the popup.



Creating or editing a position definition will bring up the panel shown in the panel below. This will create a unique position for the head form that is stored in the keyword deck. This can then be re-used at a later stage without having to re-create it.



Delete headform position will remove the current position and return to the main panel. **Auto cut** will turn on PRIMER's cutsection feature and automatically orient the section through the centreline of the headform. This can aid in headform positioning.

The horizontal angle is the angle between the global X axis and the headform local X axis in the global XY plane. This can be set to the minimum, mean or maximum angle as defined on the selected target. If none of these angles are required a user defined angle can be entered which is required to be between the minimum and maximum.

The positional node is used for positioning and rotating. By default, when setting the vertical angle, Primer will rotate the headform around the head reference node. This can now be changed to any node on the headform. The default can be reset by clicking on **Reset to default**. After positioning the headform automatically, the initial contact point is known. The positional node can then be set to the initial contact point by clciking on **Set to contact pt**. With an alternative to the headform reference node chosen as the positional node, the X,Y,Z distance from the target point shown on this panel will now use the new node. Also, clicking **zero** will zero the headform to the target point at this new node.

The Auto vertical tool can be used to automatically position the headform to its maximum vertical angle by simulating the rolling of the headform on the trim. More information on this can be found below.

The vertical angle is the angle between the headform local X axis and the global X axis in the local XZ plane. This can either be typed in or dragged into position using the **DRAG** button and clicking and dragging the headform in the graphics area.

Max HIC This will rotate the head from +25 to -25 of its current position and leave it at the angle which minimizes the spread of penetrations in the XZ cutting plane of the head. This is theangle at which the head is least able to roll on impact and consequently should give the highest HIC value. It is recommended that the head be positioned at <0 0 0> first (note the positional node is reset to the head reference node for this operation).

AUTO POSITION This function will attempt to position the head to minimize the distance between the initial contact point and the target point. This distance (contact->target) is measured in the head local YZ plane, i.e as viewed along the X axis (line of flight).

There are 3 modes for controlling the corrective motion of the head.

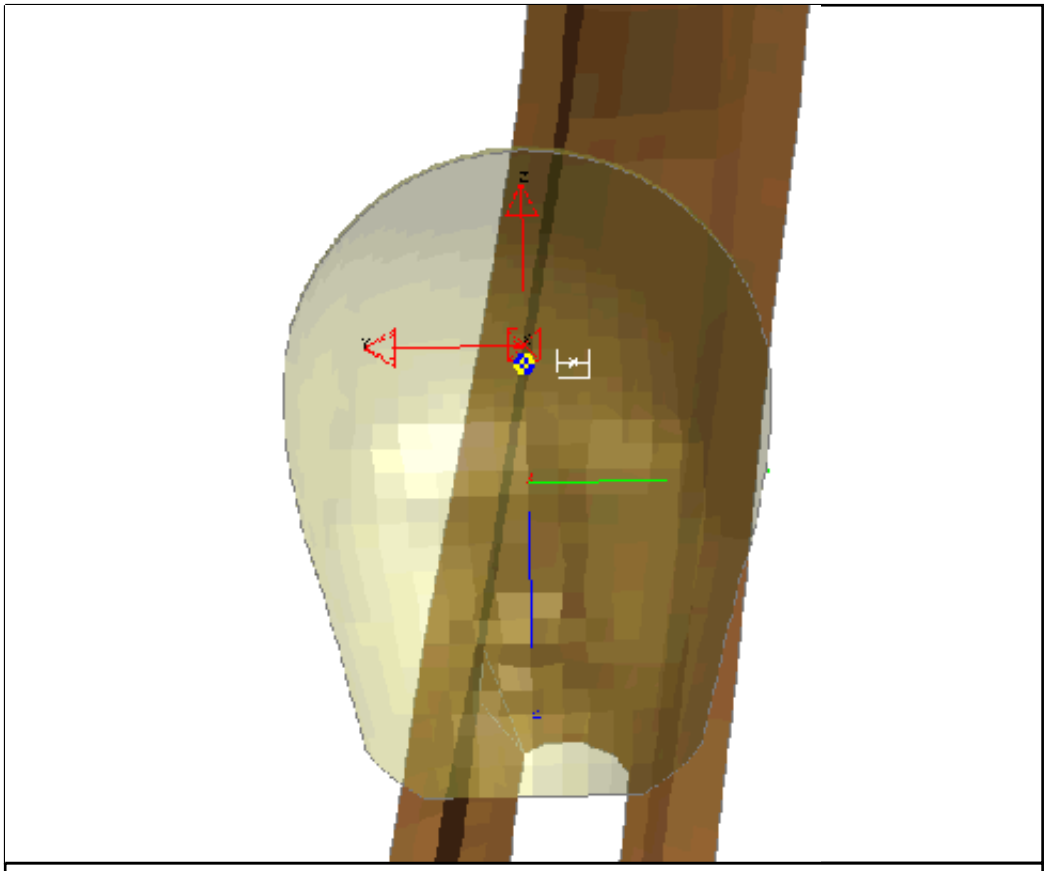
Lock on axis - in this mode the head will only be moved along its line of flight (local X), i.e.without varying local Y or Z coordinates

Lock on XZ plane - the head can move axially (local X) and up or down (local Z) but the local Y coordinate does not vary

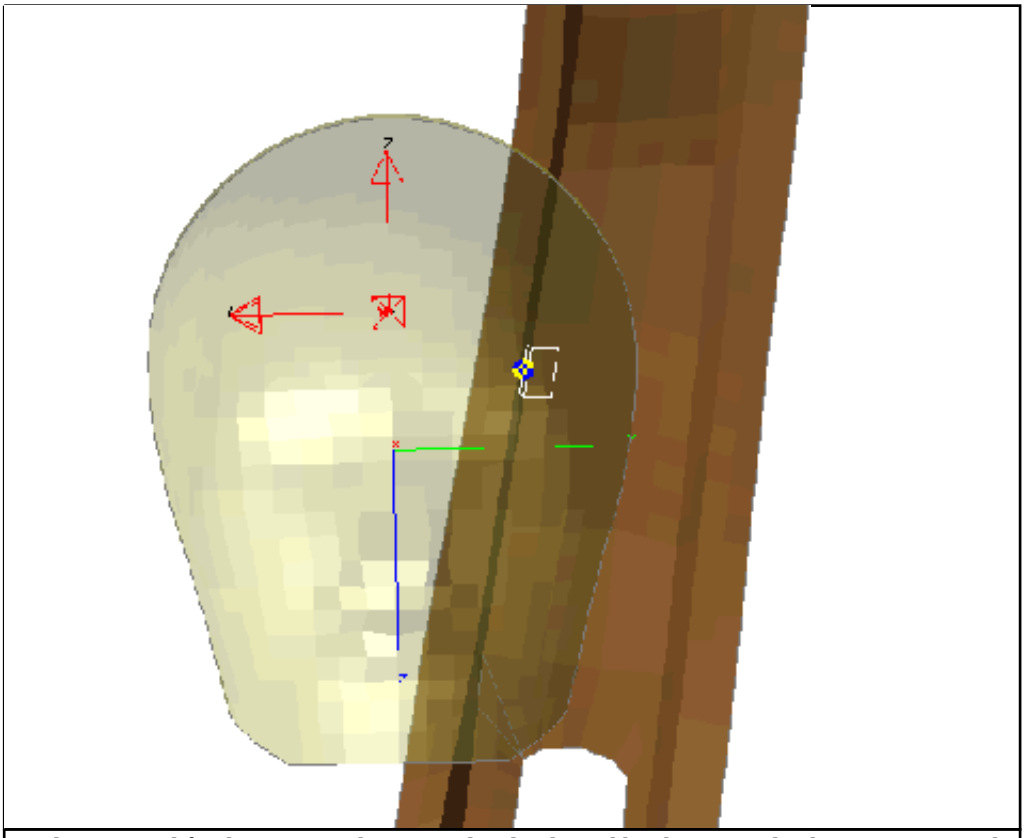
Unlock - the head can move freely

The example below shows how the different modes will position the head when faced with a "difficult" target point, which has been positioned in a re-entrant corner of the trim. In such cases, keeping the head locked to the XZ plane limits how close the initial contact point and target point can become. However, freeing the motion may result in excessive sideways movement and even an initial contact point outside the defined perimeter.

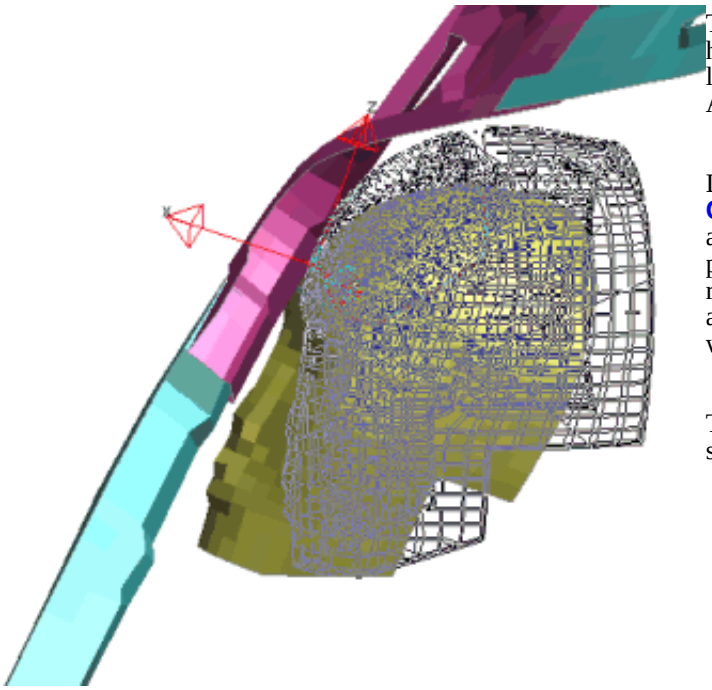
The automated positioning system is designed to assist in headform positioning, however, the user must use his own judgement about whether the iterative process has actually achieved the position most suited for test.



Head positioned with lock on XZ plane cannot get initial contact point very close to target point.



Head positioned freely gets initial contact closely aligned but has moved sideways excessively.

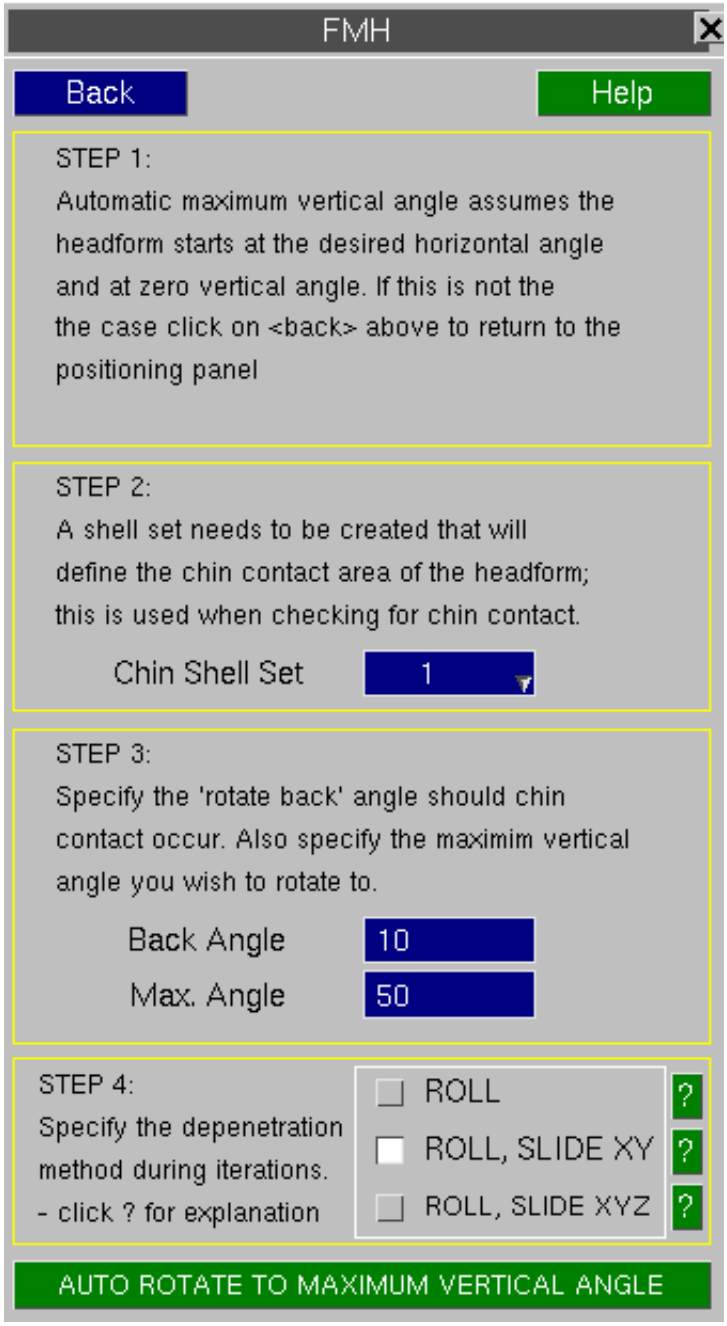


The headform can also be dragged in the headform local X, Y and Z directions using the left, middle and right mouse buttons respectively. Also its vertical angle can be dragged.

During the drag operations if the **CHECK_PENETRATIONS** button is made active then the contact is checked for penetrations each time a mouse button is released. The penetrations are sketched in white and the **CHECK_PENETRATIONS** is updated with a feedback message.

The contact and the target point can also be sketched at any time during these operations.

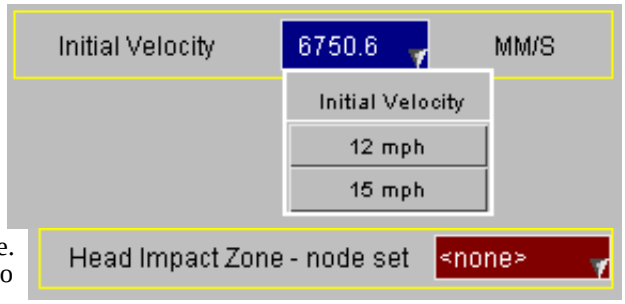
Auto vertical tool



The auto vertical tool can be used to automatically position the headform to its maximum vertical angle by simulating the rolling of the headform on the trim. The user can set a maximum angle and a ’back angle’ which is used when contact occurs between the chin and the trim. This process works by starting off with the headform in contact with the target point. It will then slowly rotate the headform (depenetrating along the way) until it reaches its maximum vertical angle or the chin touches the trim (whichever comes first). The headform will then rotate back by the desired back angle (usually 5 or 10 degrees), again depenetating the headform along the way.
Ensure the headform is positioned at the desired horizontal angle before using this tool. A shell set to define the chin area also needs to be created. The headform should also be positioned at the point of first contact as well, although Primer will run the automatic position process before entering this panel to find this point. As the headform rotates the contact point may change, and therefore the rotation centre point will change automatically. Also, the user can choose the method of head depenetration when rotating. With the depenetration method set to ’ROLL’, Primer will depenetrate the headform from the trim along the x-axis of the headform coordinate system. This simulates the rolling of the headform off the target point during rotation. With the ’ROLL, SLIDE XY’ setting, Primer will depenetrate the headform from the trim on the XY plane of the headform coordinate system. This simulates the headform rolling off the target point during rotation and then sliding back towards the target along the XY plane. With the ’ROLL, SLIDE XYZ setting, Primer will depenetrate the headform freely in all directions of the headform coordinate system. This simulates

the headform rolling off the target point during rotation and then being free to slide back towards the target point in X, Y and Z directions. **Auto rotate to maximum vertcial angle** will apply the process.

The headform initial velocity is automatically created in the headform local X axis direction. The popup gives the option of selecting the two common impact velocities (in miles per hour) or the velocity can be typed in (in model units).



You can specify a node set to define the headform impact zone. This is used in the headform checking panel described above to ensure that the point of first contact is within the impact zone

6.20 GROUPS

Groups can be used in PRIMER to collect things together. Anything that has a label can be put into a group, for example `*PART`, `*NODE`. Things that do not have labels such as `*CONSTRAINED` and `*BOUNDARY` cannot be grouped.
At present the only use for groups is for assigning mass. In future releases of PRIMER groups may have other uses. When a group is used (for example in assigning mass all we are really interested in is structural items to add mass to) PRIMER will automatically calculate the contents of the group. If the group contained part 1, PRIMER will automatically find the elements that are in part 1 and then the nodes that are on those elements.

6.20.1 Group I/O

All Groups in the model may be written to an Ascii file using the **EXPORT** function. Similarly an ascii groups file may be read in using **IMPORT**. The **PART_LIST** function writes an explicit list of all parts contained in groups to ascii file: group_parts.asc.

You can make groups that you create in PRIMER available in D3Plot by writing a groups file.

6.20.2 Group format

Groups are written to the keyword deck after the `*END` keyword. As with any keyword lines beginning with a `$` are treated as comments and skipped.
It is strongly recommended that you do not edit the groups by hand. Use the group creating/editing capabilities of PRIMER.
Entities can be added to a group by one of three methods;

- Selecting all entities (optionally in a box)
- Selecting a list of entities (optionally in a box)
- Selecting a range of entities (optionally in a box)

The box is used for selecting a subset of the entities that are in the box. For example if the group contained *'all elements inside box 1'* then only the elements that are in box 1 will be used, not all the elements.

Entities can be added or removed from a group. For example you may want a group to contain *'all elements except those in box 1'*. You could do this by first *'adding all elements'* and then *'removing elements in box 1'*.

Example

```
*GROUP
$<title>
This is the example group title
$  <label> < visual attributes of the group>
         1 R255G000B000 50 CURRENT * * * * UNBLANKED
$ The following lines give examples of a group
$
$ Adding all parts to a group
PART ALL
$ Adding all parts to a group in box 1
PART ALL BOX 1
$ Removing all parts from a group in box 2. Note -PART
-PART ALL BOX 2
$ Adding a list of shells to a group
SHELL 1 20 23 100 200
$ Adding a list of shells to a group in box 2
SHELL 1 20 23 100 200 BOX 2
$ Adding a range of nodes (100 to 1000) to a group
NODE 100:1000
$ Adding a range of nodes (100 to 1000) to a group in box 2
NODE 100:1000 BOX 2
$ Removing a range of beams from a group. Note -BEAM
-BEAM 50:60
```

Order of calculation of Groups

The order in which things are added to/removed from groups is important. To stop any ambiguity the following order is used.

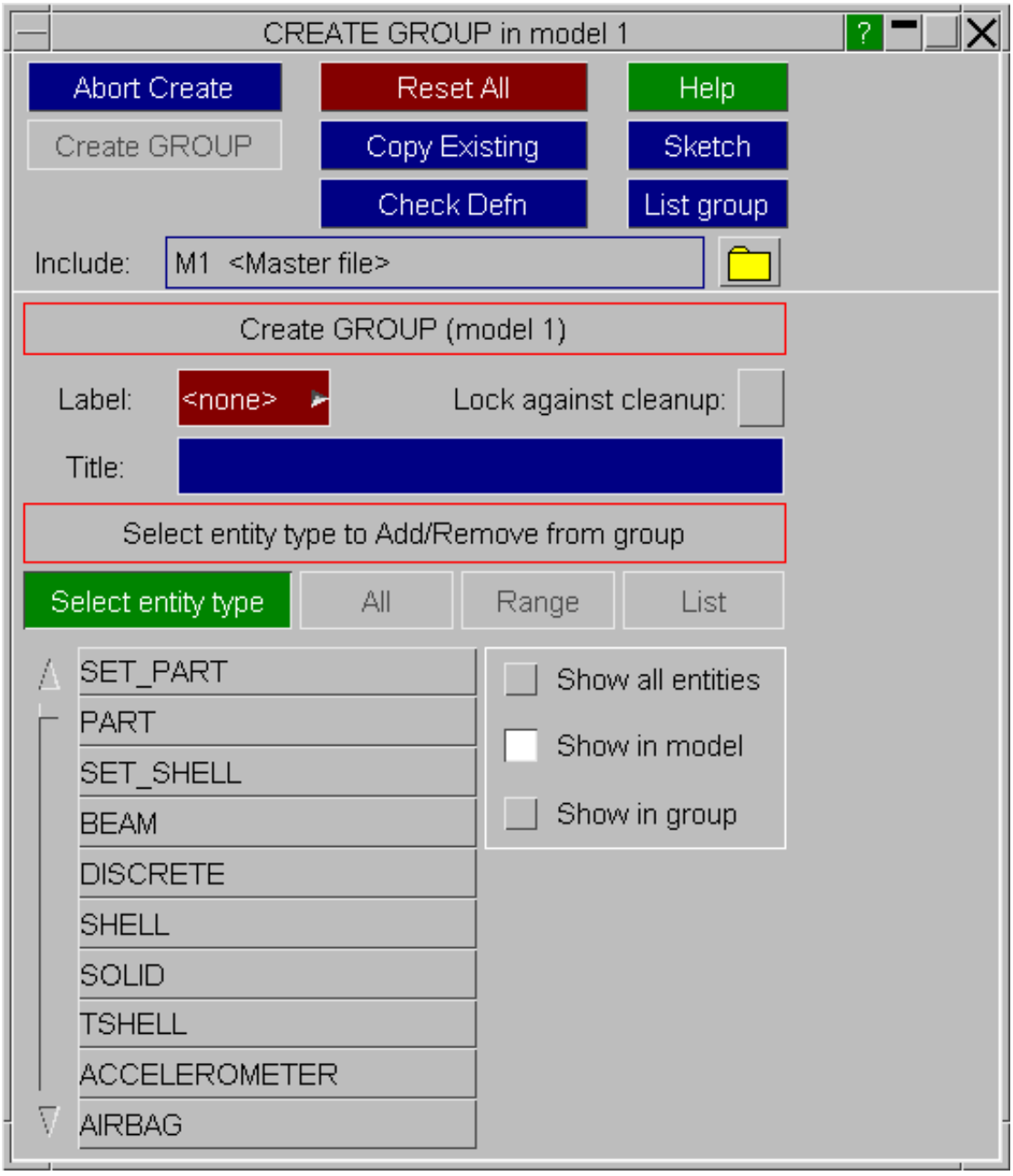
First, entities are added to groups in the following order:

- **SET_PART**
- **PART**
- other set types
- elements
- other entities alphabetically

Then, entities are removed from groups (if needed) in the same order.

6.20.3 Creating/Editing Groups

The initial screen for creating/editing groups is shown below. Before a group can be created a **Label** needs to be given. A **Title** can also be given if needed.



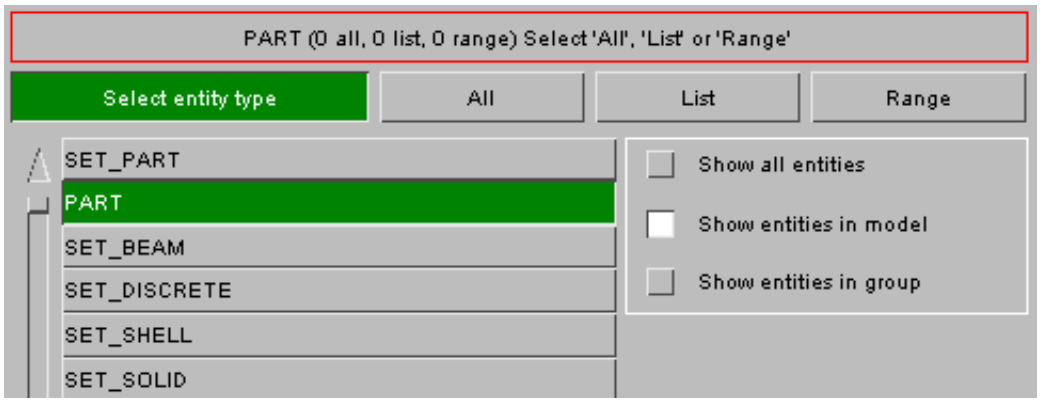
Once the label for the group is given the **CREATE_GROUP** button becomes active to create the group.

Locking the contents of a group against clean up

By default an entity that is not used in a model will be removed from a group in a model 'clean up unused' operation. The **Lock against cleanup** checkbox will prevent the contents of a group from being cleaned up. This is saved in the `*GROUP` keyword written after `*END` by PRIMER. For example, this could be useful if you want to make some sets in your model which you know will be needed at some time in the future but are currently not being used. If the sets are added to a group they will not be deleted by PRIMER.

Selecting entity type

Any entity which has a label can be added to a group. Before anything can be added to the group you have to choose the entity type you want to add. This is done with the list on the left hand side of the menu. By default all the entity types that are present in the model that you are editing are shown. This can be changed by using the radio buttons on the right. You can see all entity types (even if they are not present in your model), the entities in your model, or just the entities that are present in the group. Once the entity type is chosen the type is highlighted and the **All**, **List** and **Range** buttons become active to enable you to edit that type. For example if **PART** is selected:



The feedback box (shown at the top of the figure) changes to show what is defined in the group by **`PART`**. In this example above there are no entries by all, list or range so all are zero. As PARTs are added this will change.

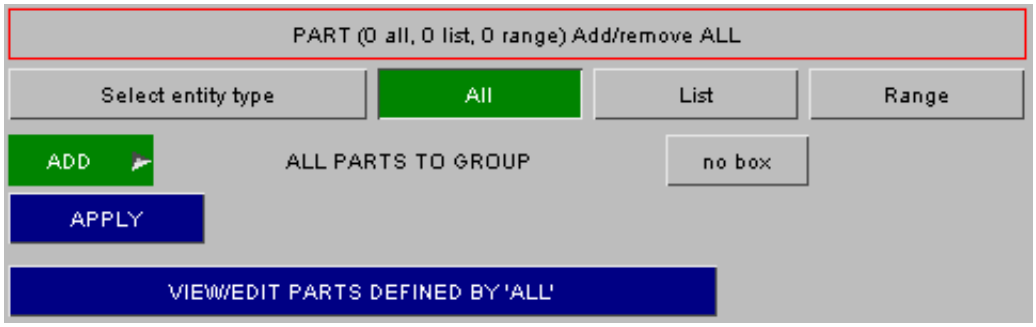
6.20.4 Adding, Editing and Deleting Entities in Groups

The following sections explain various ways of adding and manipulating entities within groups.

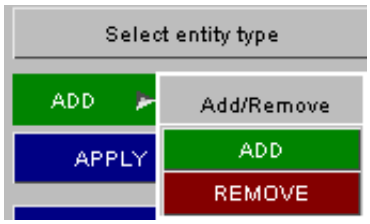
Adding entities by **ALL**
Adding entities by **LIST**
Adding entities by **RANGE**

6.20.5 Adding entities by **All**

Press the **All** button.
The screen changes to:



Selecting **ADD** or **REMOVE**



The green **ADD** button indicates that PARTs will be added to the group. If you want to remove PARTs instead of adding PARTs to the group, then the **Add/Remove** popup button can be used to change the action. If **REMOVE** is selected the button will change to red.

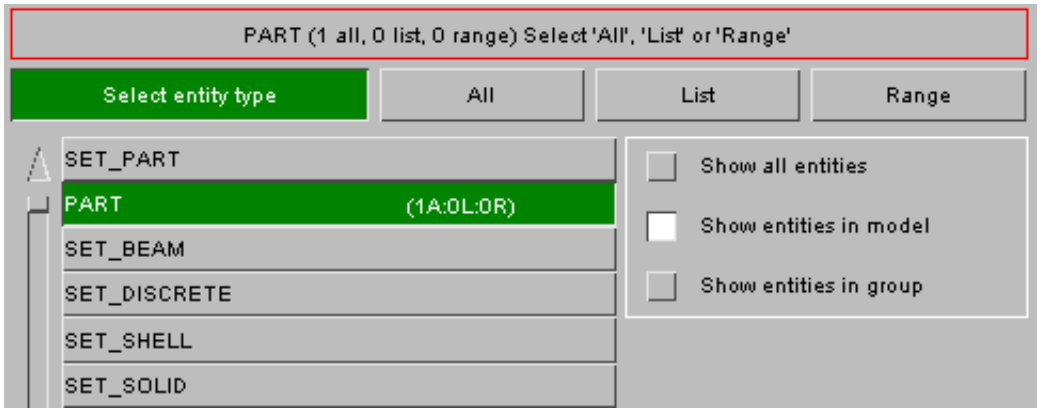
Using a box

In the above example we are adding all PARTs to the group. We are not using a box to limit the selection as **no box** is selected, If instead we wanted to add the elements from all PARTs that are in box 1 this can be done by clicking on the **no box** button. The display will change as shown on the left.

Until a box is selected the **APPLY** button will be inactive. You can turn off the box selection again by pressing the **in box** button. Once a box is selected by either typing in the box number or using the popup menu it will turn blue and the **APPLY** button will be activated.

Saving into the group

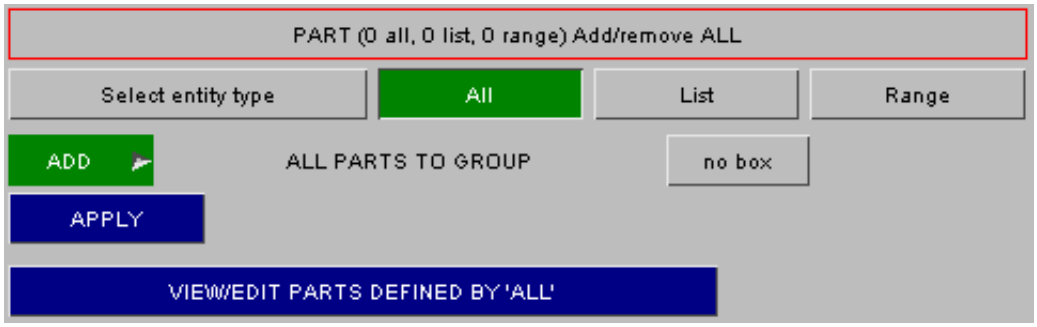
Once you have chosen to Add/Remove and if you want to use a box or not the selection can be saved into the group by pressing the **APPLY** button.



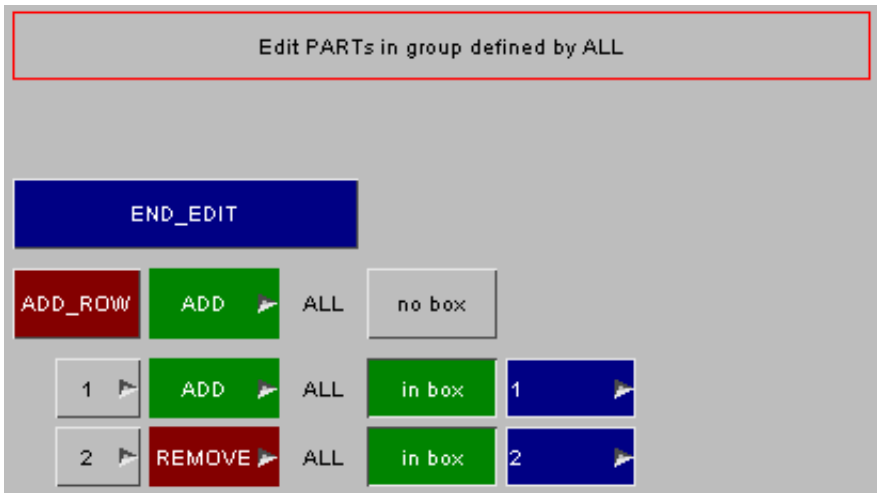
After the selection is saved to the group the screen refreshes back to the main screen (shown above). The feedback button (at the top of the image) changes to show that 1 selection of PARTs by all has been added. Additionally this is shown on the **PART** entity button. In this way you can easily see what entity types are present in the group.

6.20.6 Editing/deleting entities by **All**

Press the **All** button. The screen changes to:

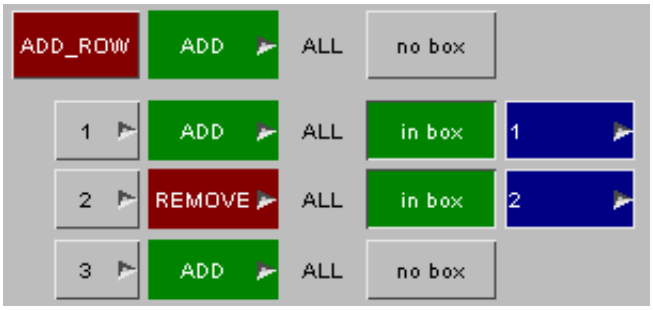


Press the **VIEW/EDIT PARTS DEFINED BY 'ALL'** button. The screen changes to the panel on the right. In this example there are 2 entries. Firstly ADD all PARTs in box 1. Secondly REMOVE all PARTs in box 2.



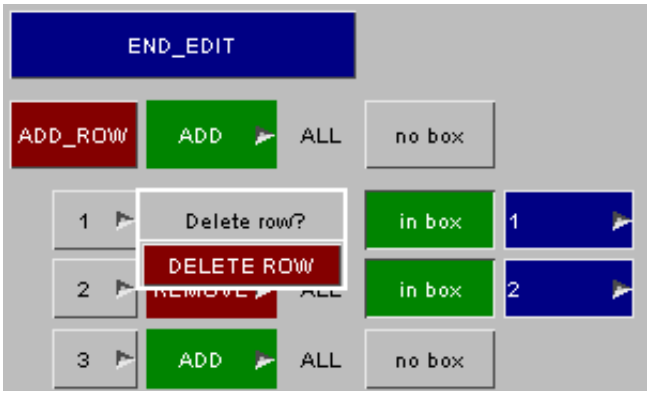
Adding an 'all' row using the group editor

A new entry can be added to the group by using the **ADD_ROW** button. The **ADD**/**REMOVE** and **box** buttons work in the same way as the panel to add entities by all. In the example above a new row 'ADD all parts' would be added to the group. A box would not be used as it is turned off. Adding the row would result in:

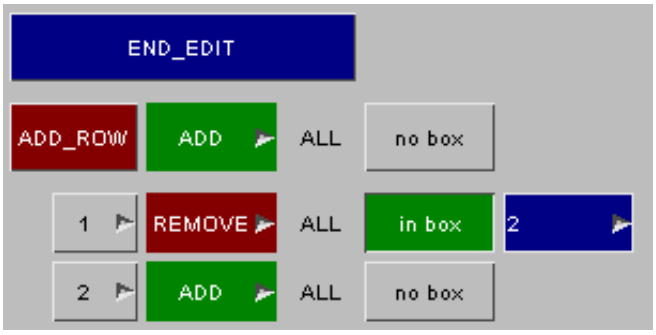


Deleting an 'all' row using the group editor

To delete an entry from a group use the **Delete row?** popup. For example, to remove entry/row 1 right click on the **1** button and select **DELETE ROW** from the popup menu.



The row is deleted and the remaining 2 entries are moved up the list.

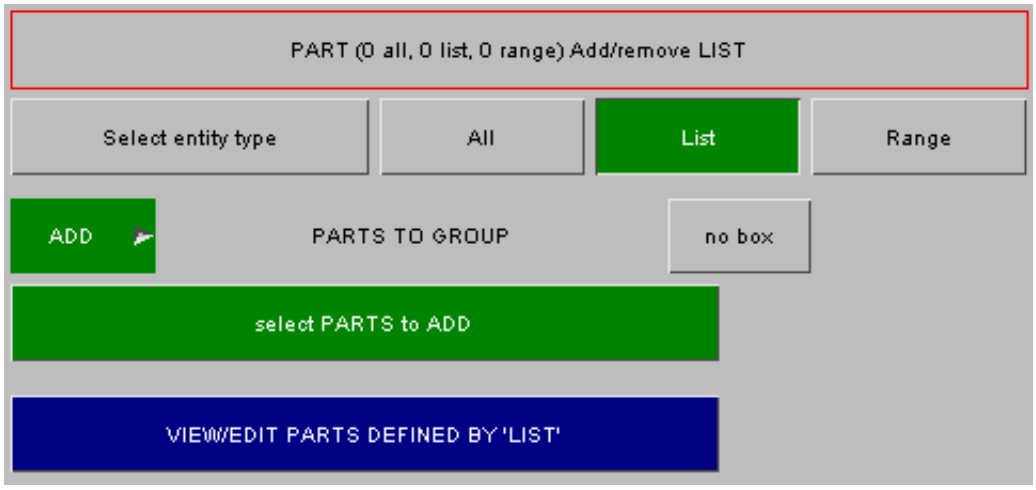


Editing an ’all’ row using the group editor

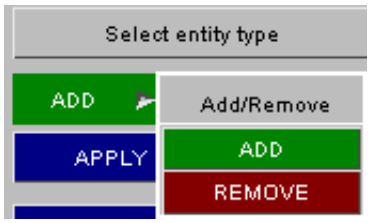
Each of the existing rows can be modified if needed. For each row the **ADD/REMOVE** and **box** buttons work in the same way as the panel to add entities by all.

6.20.7 Adding entities by **List**

Press the **List** button.
The screen changes to:



Selecting **ADD** or **REMOVE**



The green **ADD** button indicates that PARTs will be added to the group. If you want to remove PARTs instead of adding PARTs to the group, then the **Add/Remove** popup button can be used to change the action. If **REMOVE** is selected the button will change to red.

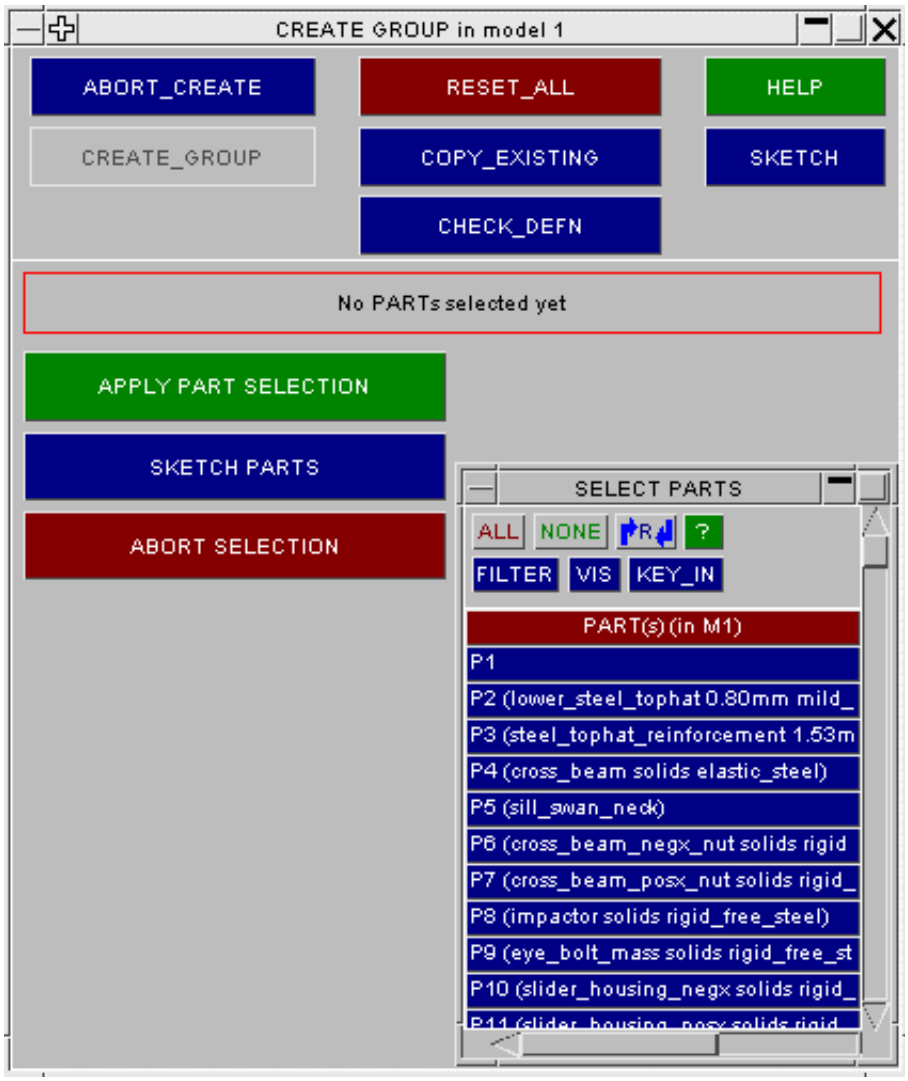
Using a box

In the above example we are adding PARTs to the group. We are not using a box to limit the selection as **no box** is selected, If instead we wanted to add the elements from PARTs that are in box 1 this can be done by clicking on the **no box** button. The display will change as shown on the left.

Until a box is selected the **APPLY** button will be inactive. You can turn off the box selection again by pressing the **in box** button. Once a box is selected by either typing in the box number or using the popup menu it will turn blue and the APPLY button will be activated.

Selecting entities to add

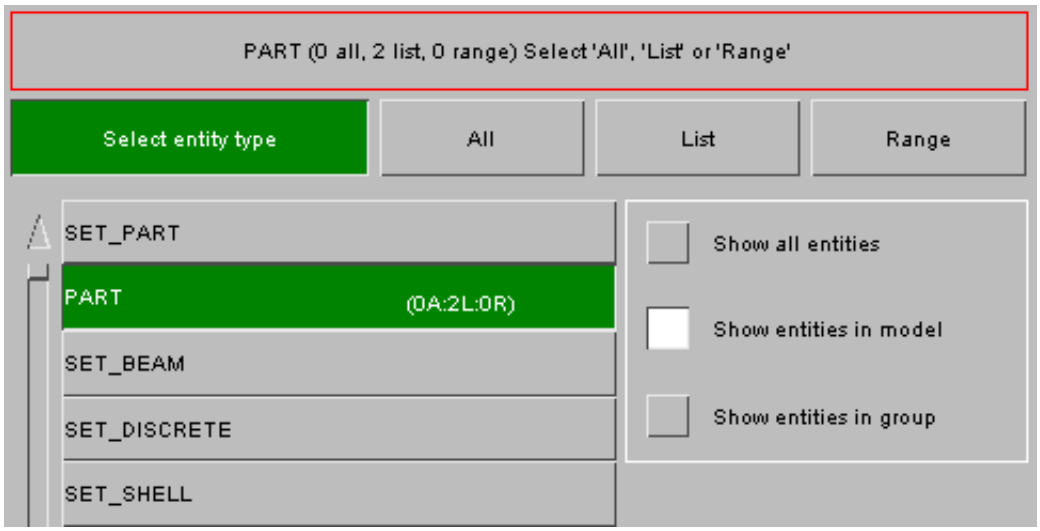
To start selecting the entities you want to add by list press the **Select PARTS to ADD** (eg if entity type is **PART**). The screen changes to:



The standard PRIMER object menus appear which allow you to select PARTs. Select the PARTs that you want to add by either selecting them from the list, picking visible parts etc. You can abort adding the entities at any time by pressing **ABORT SELECTION**. You can sketch the entities that you currently have selected to add by pressing the **SKETCH PARTS** button.

Saving into the group

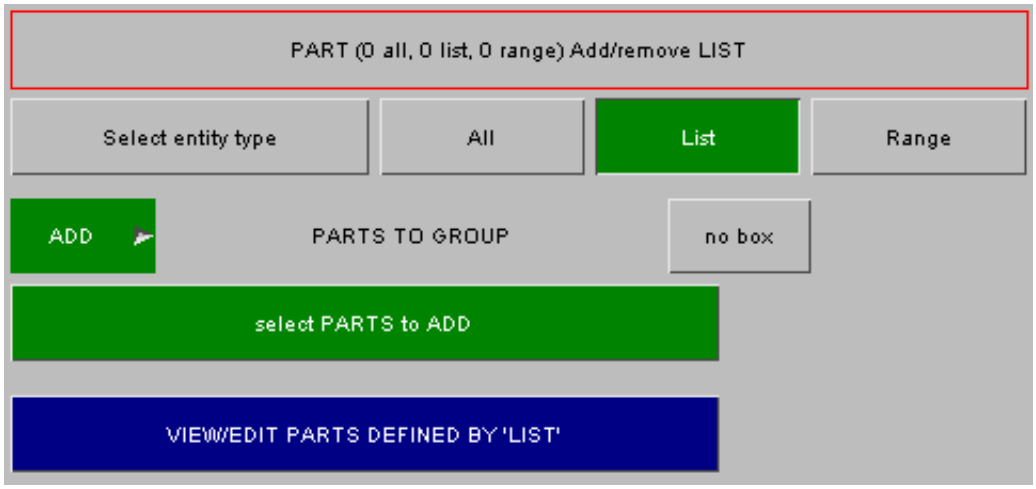
Once you have chosen the parts that you want to add using the object menus you can save them into the group by pressing the **APPLY PART SELECTION** button.



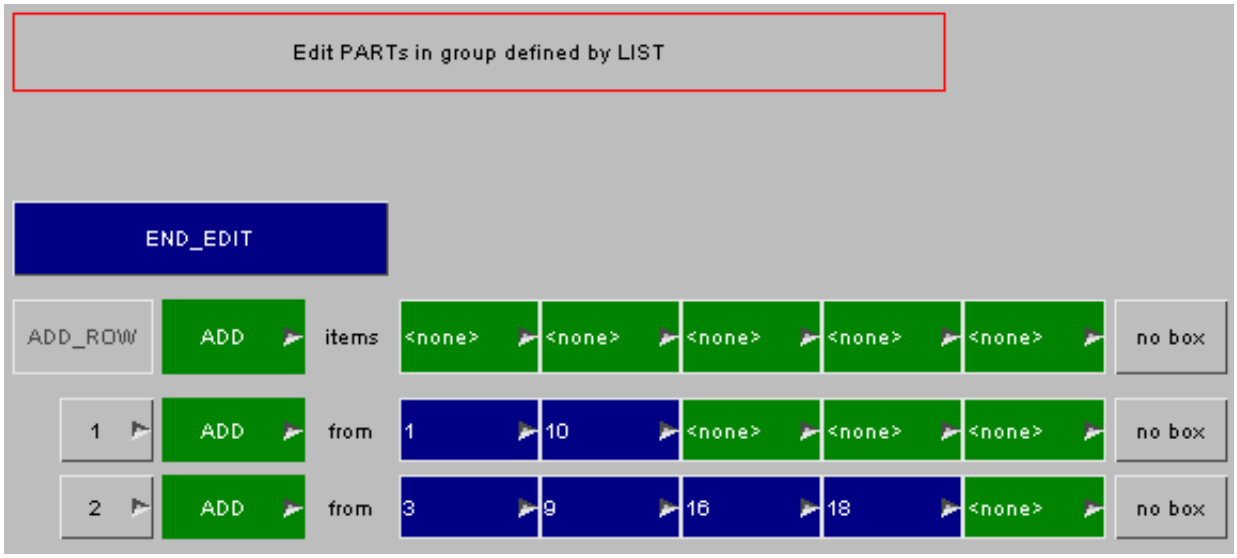
After the selection is saved to the group the screen refreshes back to the main screen (shown above). The feedback button (at the top of the image) changes to show that 2 selections of PARTs by list have been added in this example. Additionally this is shown on the **PART** entity button. In this way you can easily see what entity types are present in the group.

6.20.8 Editing/deleting entities by **List**

Press the **List** button.
The screen changes to:

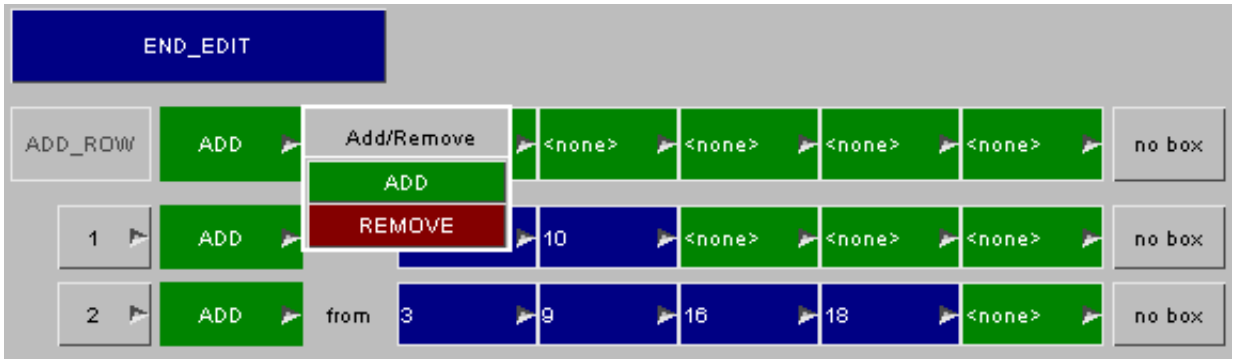


Press the **VIEW/EDIT PARTS DEFINED BY 'LIST'** button. The screen changes to the panel below. In this example there are 2 rows which have a total of 6 entries.

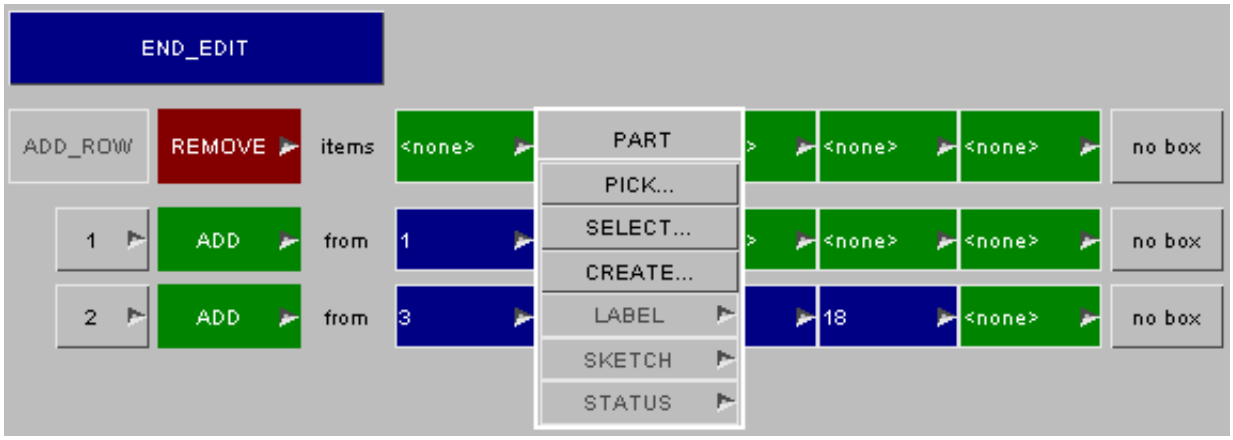


Adding a 'list' row using the group editor

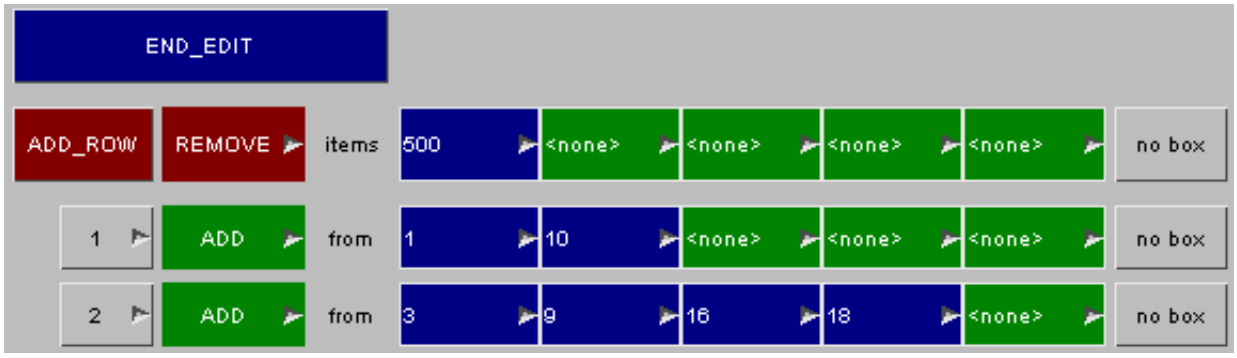
A new entry can be added to the group by using the **ADD_ROW** button. The **ADD/REMOVE** and **box** buttons work in the same way as the panel to add entities by list. As an example we will add the row to the group 'Remove **PART** 500'. At present the **ADD_ROW** button is inactive as there is nothing to add. Firstly, the Add/Remove popup must be used to change the **ADD** button to **REMOVE**.



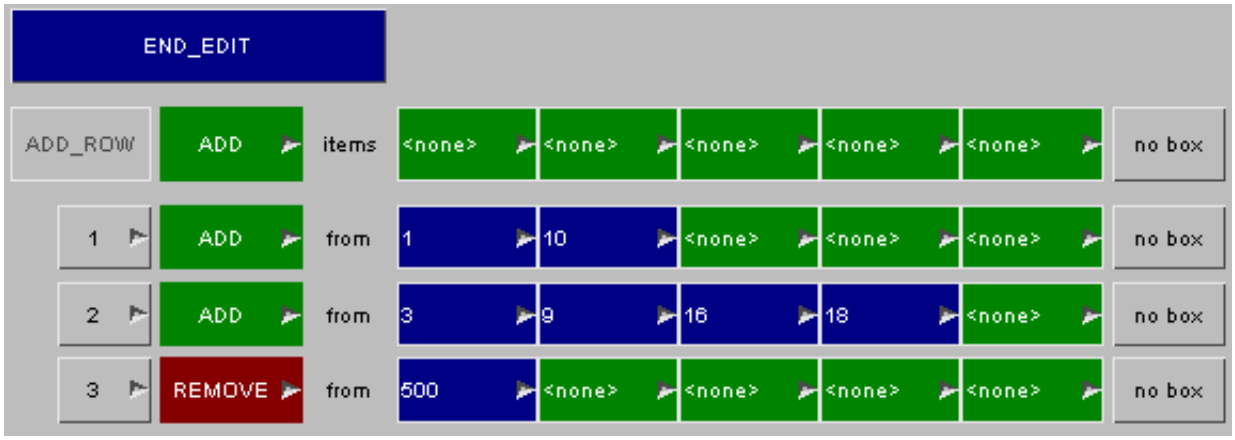
Next the part needs to be selected. You can either type the part number into the box or (as shown) use the popup menu to select the part from a list or pick it from the screen.



Now a part has been selected the **ADD_ROW** button becomes live.

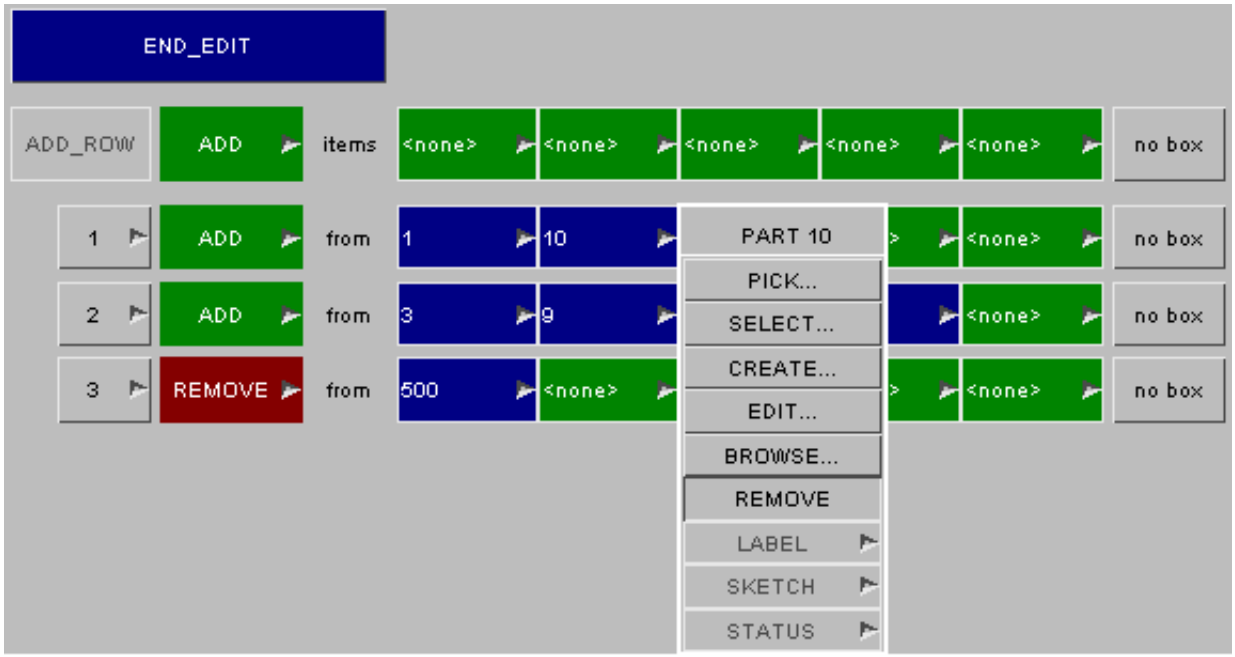


Now to add the row to the group simply press the **ADD_ROW** button. A new row '3' will be added to the list. The add row will be reset back to the default values.

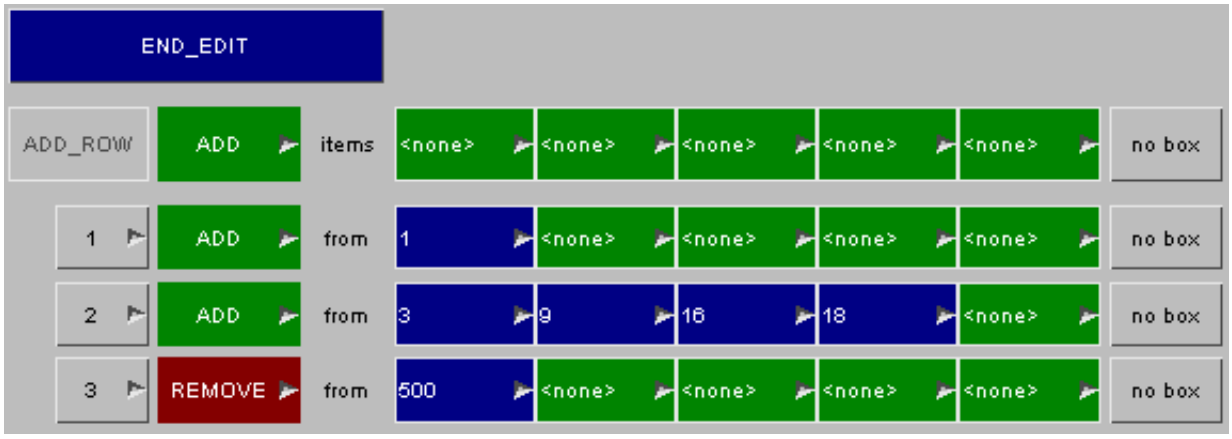


Editing a 'list' row using the group editor

Each of the existing rows can be modified if needed. For each row the **ADD/REMOVE** and **box** buttons work in the same way as the panel to add entities by list. An entity can be removed from a row by either deleting the number in the text box or by using the **REMOVE** option on the popup. For example to remove part 10, right click on 10 to bring up the popup menu and select **REMOVE**.

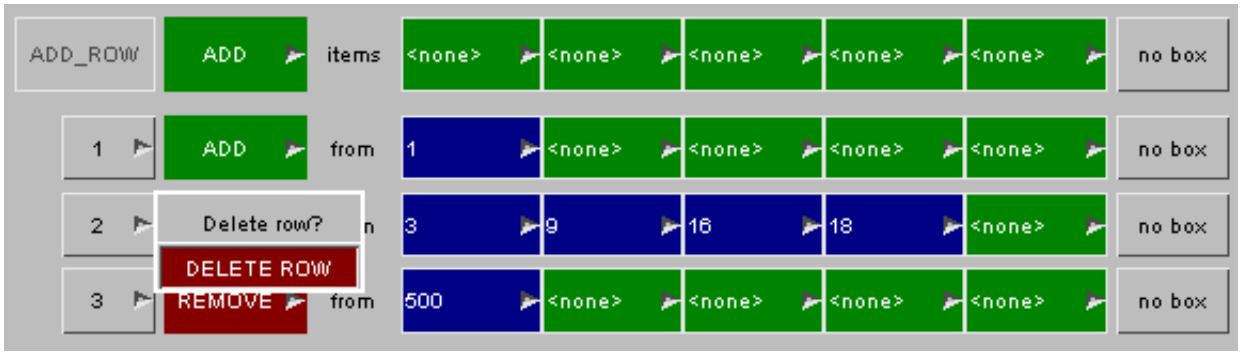


The part will be removed from the row (see image below). To save the change into the group press **END_EDIT**.

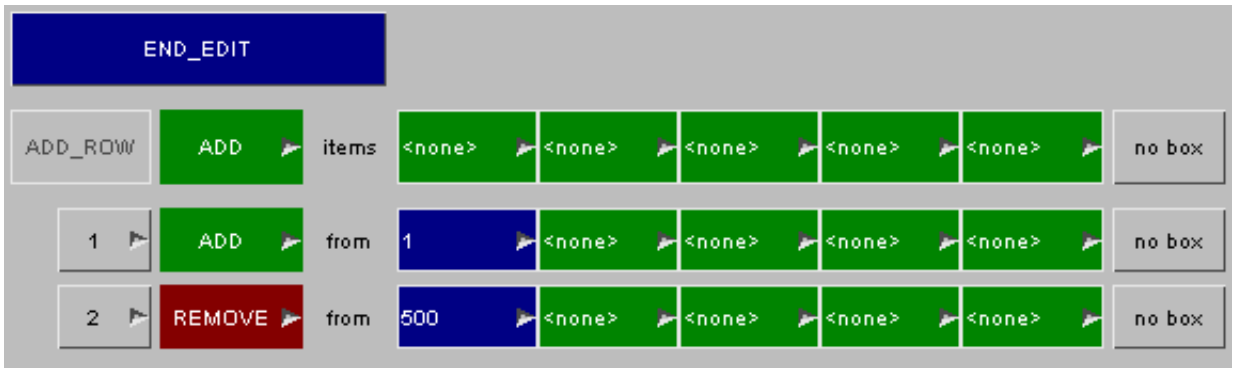


Deleting a 'list' row using the group editor

To delete an entry from a group use the **Delete row?** popup. For example, to remove entry/row 2right click on the **2** button and select **DELETE ROW** from the popup menu.

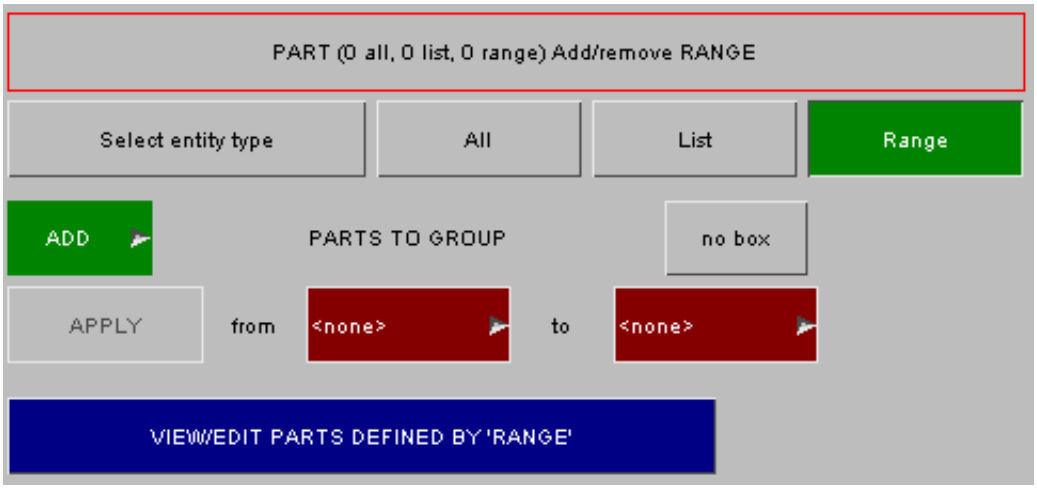


The row is deleted and the remaining entries are moved up the list.

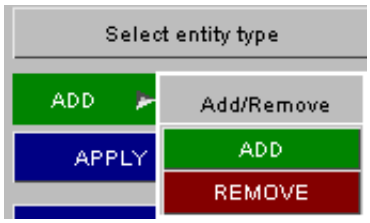


6.20.9 Adding entities by **Range**

Press the **Range** button. The screen changes to:



Selecting **ADD** or **REMOVE**



The green **ADD** button indicates that PARTs will be added to the group. If you want to remove PARTs instead of adding PARTs to the group, then the **Add/Remove** popup button can be used to change the action. If **REMOVE** is selected the button will change to red.

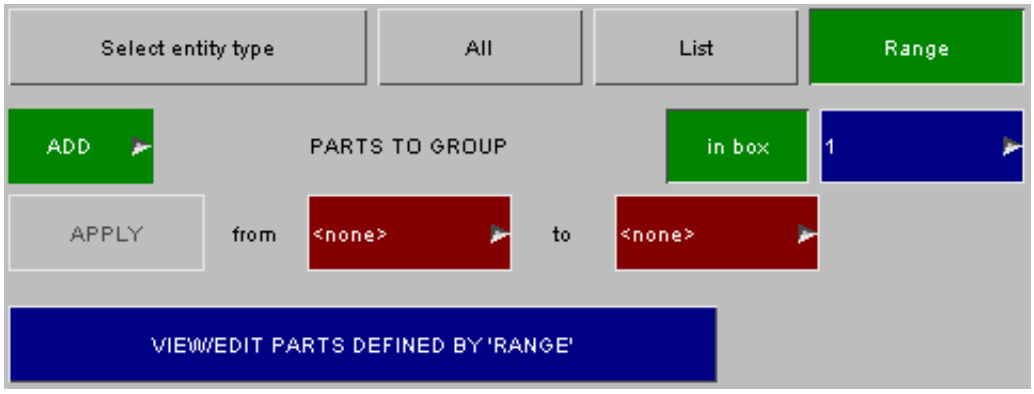
Using a box

In the above example we are adding a range of PARTs to the group. We are not using a box to limit the selection as **no box** is selected, If instead we wanted to add the elements from a range of PARTs that are in box 1 this can be done by clicking on the **no box** button. The display will change as shown on the left.

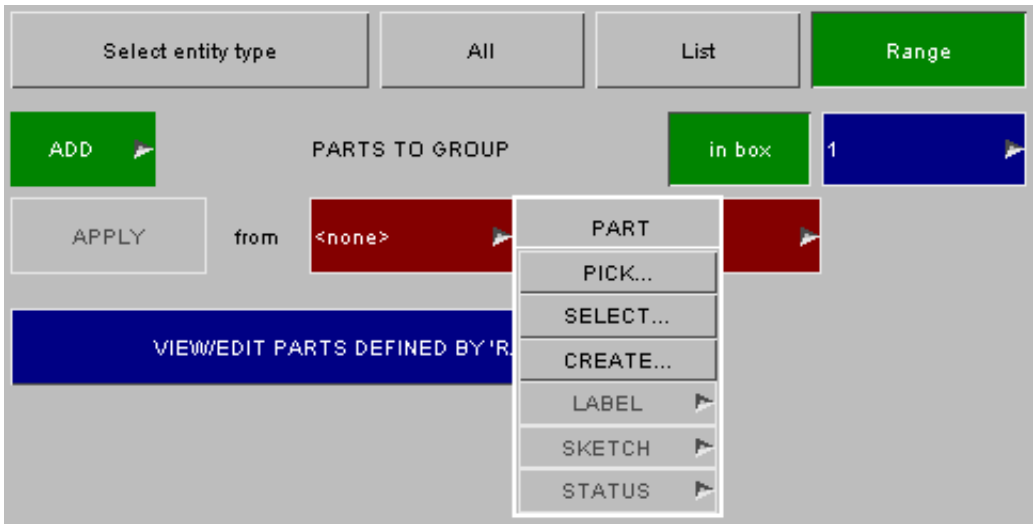
Until a box is selected the **APPLY** button will be inactive. You can turn off the box selection again by pressing the **in box** button. Once a box is selected by either typing in the box number or using the popup menu it will turn blue and the **APPLY** button will be activated.

Selecting entities to add

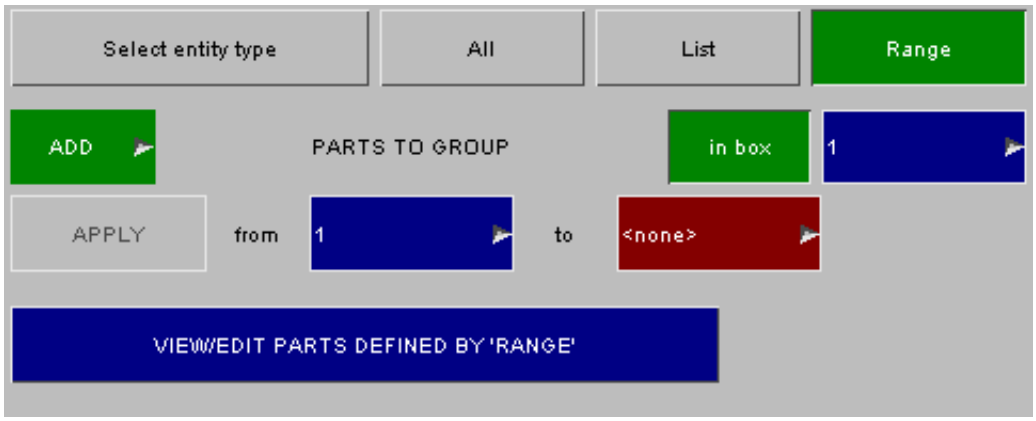
The following example shows how to add a range of entities to a group. We want to add 'Parts between 1 and 45 that are in box 1'.
First select the box as described above and choose **ADD or REMOVE** (in this example we want to add parts).



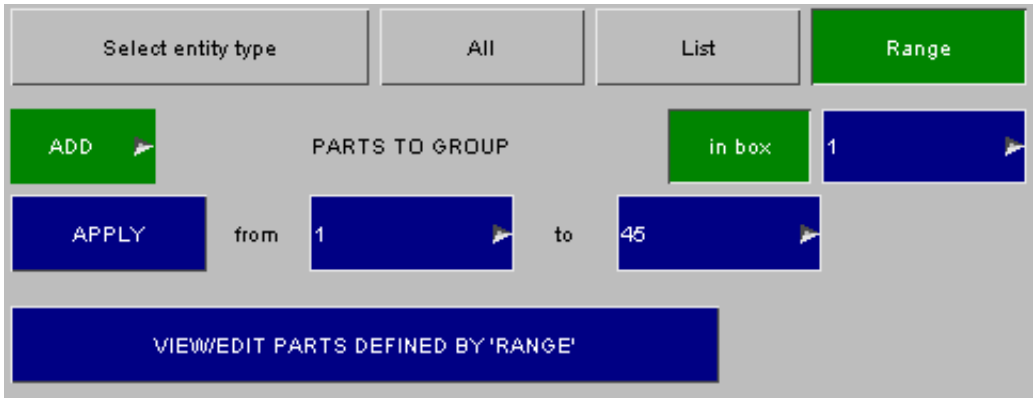
A start and end range must be given. You can either type the numbers into the boxes or use the standard PRIMER object menus to select the part or pick it from the screen. For example right clicking on the **From** field:



When the part is selected/picked (in this example part 1 is picked) the **from** field is filled in.

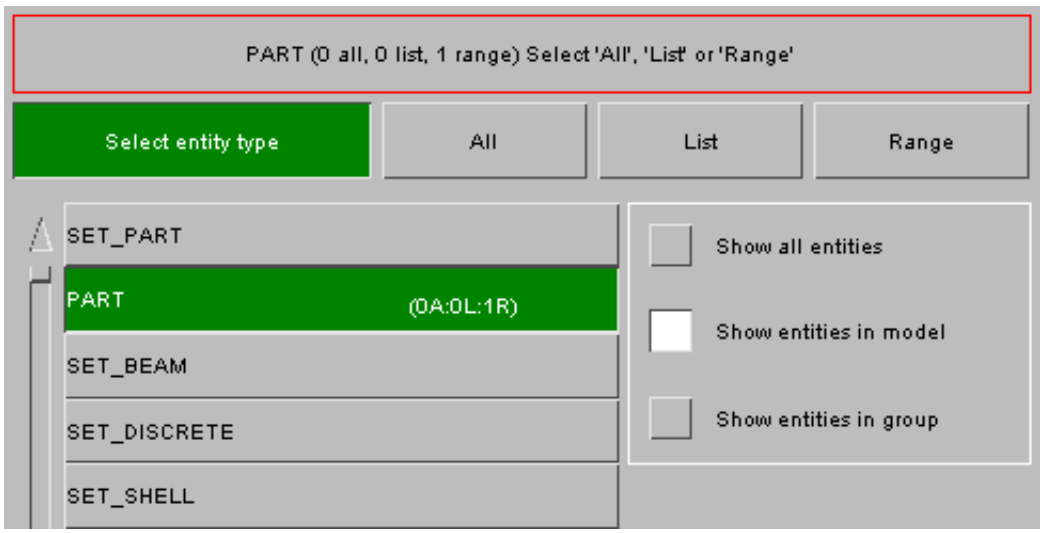


Similarly, the **to** field can be selected.



Saving into the group

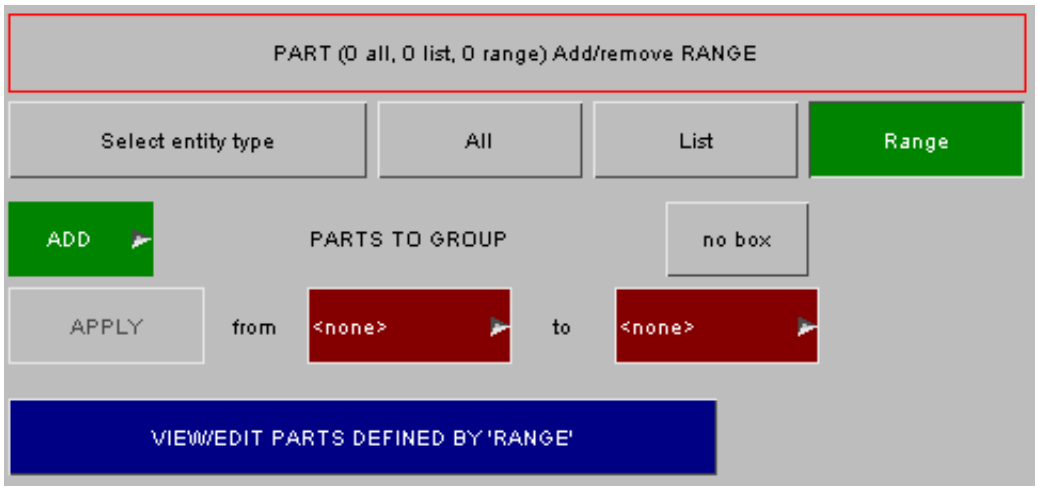
Once you have chosen the **to** and **from** fields for the parts that you want to add the **APPLY** button becomes active. You can save them into the group by pressing the **APPLY** button.



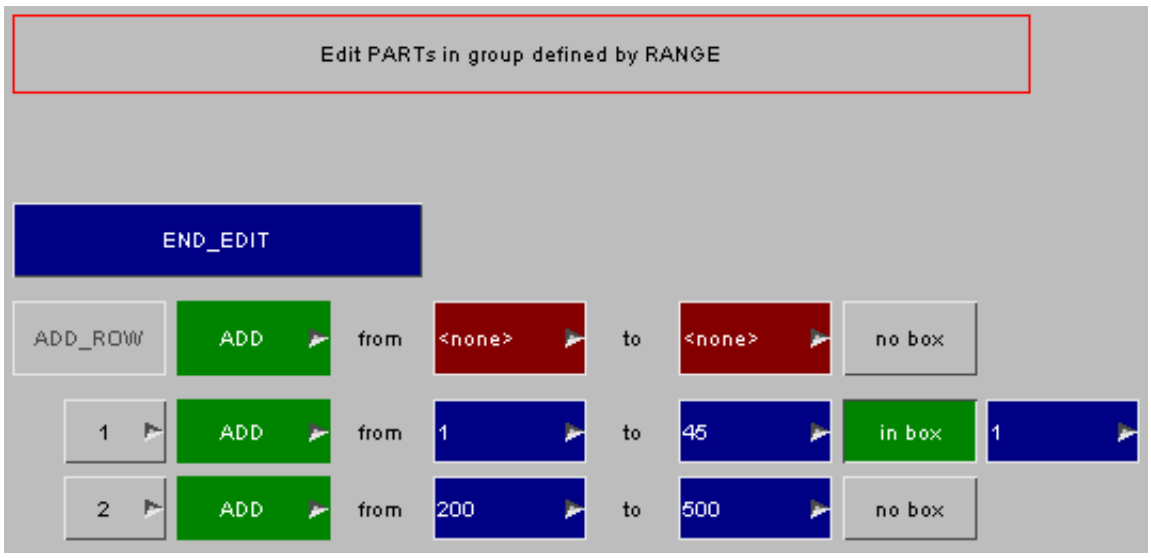
After the selection is saved to the group the screen refreshes back to the main screen (shown above). The feedback button (at the top of the image) changes to show that 1selection of PARTs by range has been added in this example. Additionally this is shown on the **PART** entity button. In this way you can easily see what entity types are present in the group.

6.20.10 Editing/deleting entities by **Range**

Press the **Range** button. The screen changes to:

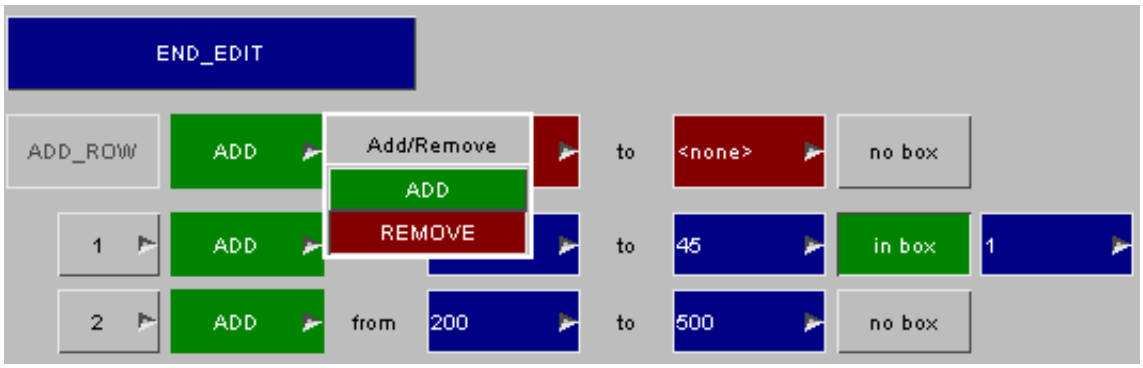


Press the **VIEW/EDIT PARTS DEFINED BY 'RANGE'** button. The screen changes to the panel below. In this example there are 2 rows. Row 1 adds parts 1 to 45 in box 1. Row 2 adds parts 200 to 500.

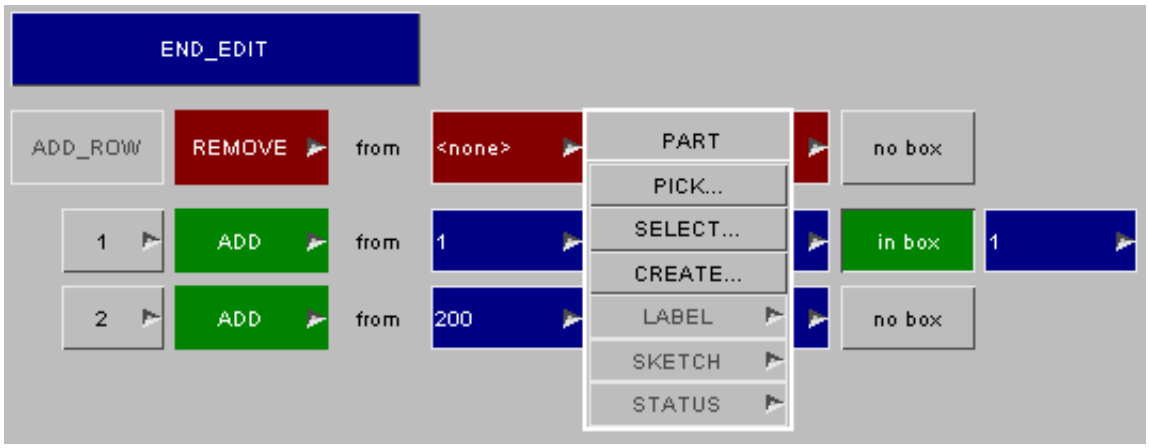


Adding a 'range' row using the group editor

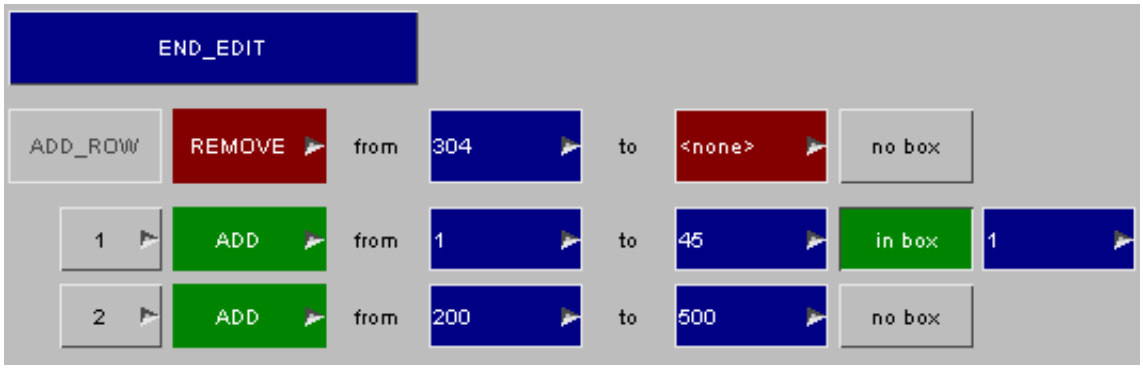
A new entry can be added to the group by using the **ADD_ROW** button. The **ADD/REMOVE** and **box** buttons work in the same way as the panel to add entities by list. As an example we will add the row to the group 'Remove **PARTs** 304 to 306 in box 2'. At present the **ADD_ROW** button is inactive as there is nothing to add. Firstly, the Add/Remove popup must be used to change the **ADD** button to **REMOVE**.



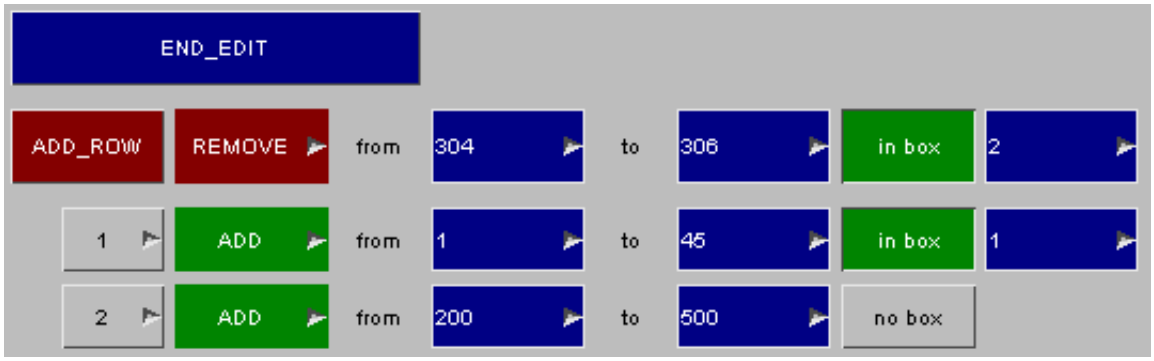
Next the to and from parts needs to be selected. You can either type the part number into the box or (as shown) use the popup menu to select the part from a list or pick it from the screen.



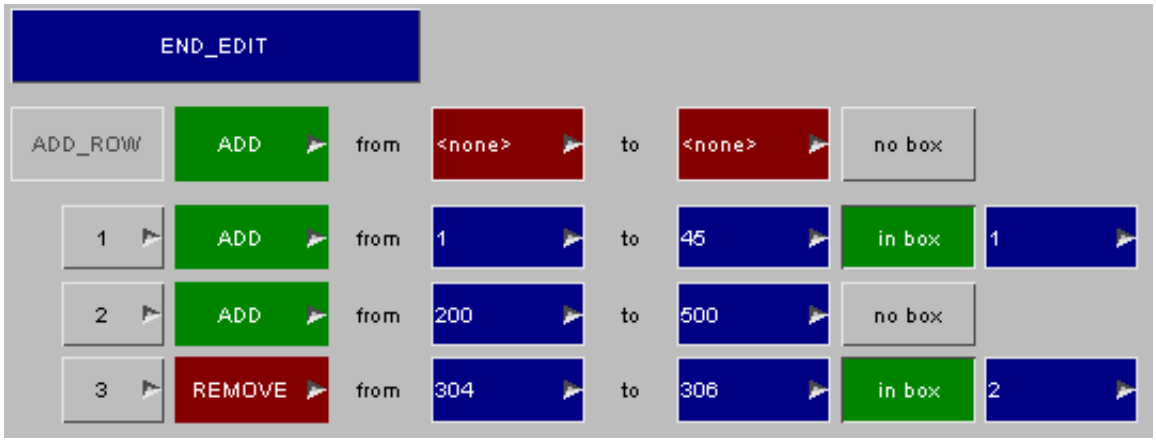
Once the part has been selected the **from** field will be filled in:



A similar process can be done to select the **to** and **box** fields. When the **to** and **from** fields are selected the **ADD_ROW** button becomes live.



Now to add the row to the group simply press the **ADD_ROW** button. A new row '3' will be added to the list. The add row will be reset back to the default values.



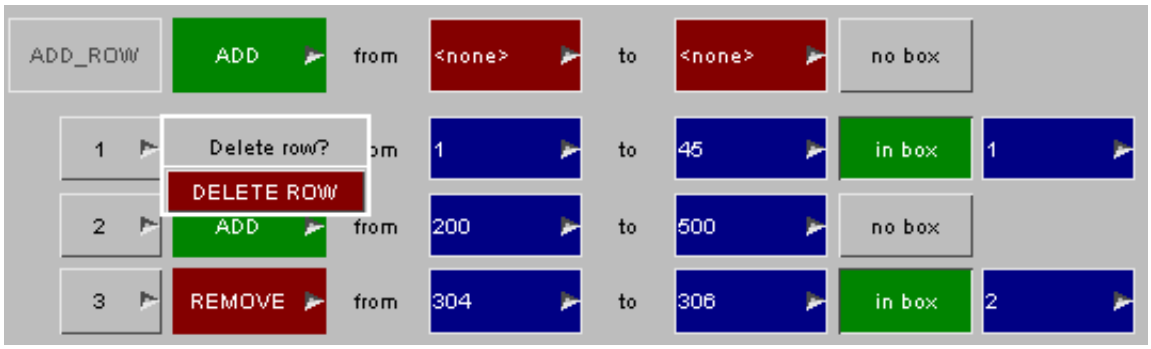
Editing a 'range' row using the group editor

Each of the existing rows can be modified if needed. For each row the **ADD/REMOVE** and **box** buttons work in the same way as the panel to add entities by list. The **from** and **to** entities can be modified by either changing the number in the text box or by using the popup menu.

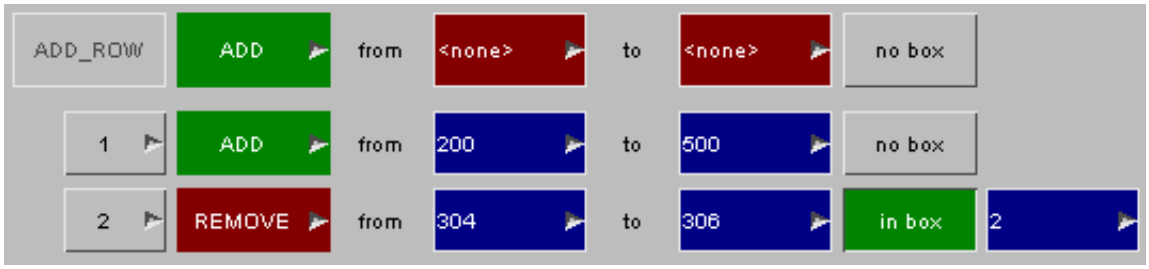
To save the change into the group press **END_EDIT**.

Deleting a 'range' row using the group editor

To delete an entry from a group use the **Delete row?** popup. For example, to remove entry/row 1right click on the **1** button and select **DELETE ROW** from the popup menu.



The row is deleted and the remaining entries are moved up the list.

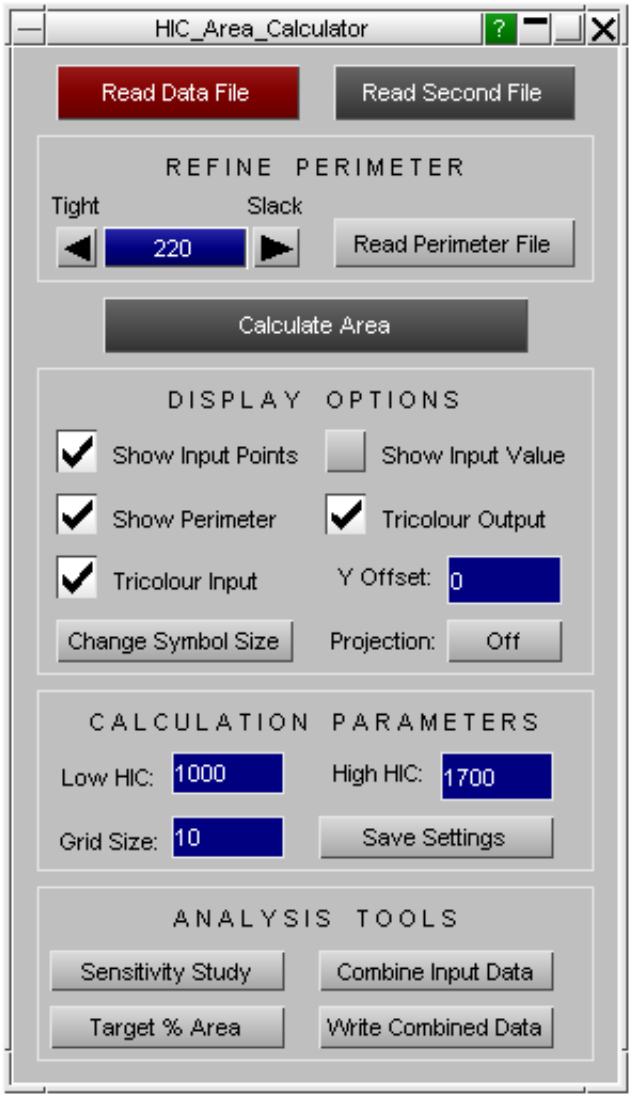


6.21 HIC Area Calculator

6.21 .1 Introduction

The HIC Area Calculator can be used to read external data files into PRIMER to plot the results from pedestrian head impact analyses. Once read the tool can then calculate the area of high/low HIC – a requirement of some pedestrian impact regulations.

The tool also contains additional functionality to help the user interpret the results and identify areas of sensitivity.



6.21 .2 Input Data

The ‘Read Data File’ button allows the user to select a text file containing the pedestrian head impact results.

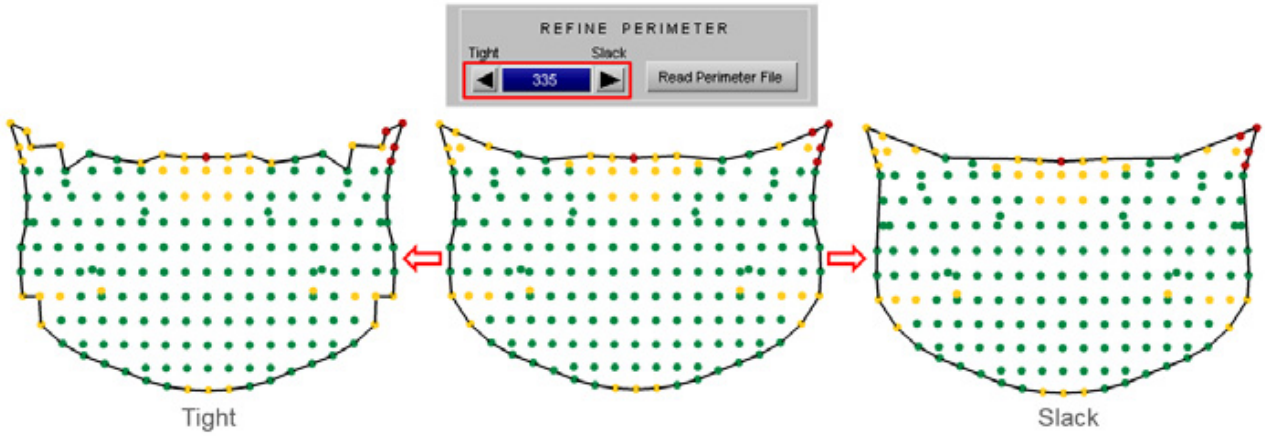
The format of the file is:

X coordinate, Y coordinate, Z coordinate, HIC value, [Name]

The data can be comma separated, tab separated or space separated and may also be prefixed with ‘data’ for compatibility with d3Plot. The name field is optional.

Once read, PRIMER will display the data on screen. PRIMER calculates a perimeter that encloses all of the data points.

The user can modify the shape of the perimeter using the arrow buttons (the number is the maximum allowable length of perimeter section and can be modified by clicking the blue button).



Tighter perimeters are more accurate and have shorter calculation times.

Should the user wish, a custom perimeter can be input using the 'Read Perimeter File' button. The format for this file is:

X coordinate, Y coordinate, (Z coordinate)

The data can be comma separated, tab separated or space separated. The order of the points is important and should represent the sequence of the perimeter. As the calculation is 2D it is not essential for the perimeter to have z coordinated defined.

For best results user-input perimeters should use the same coordinates as the data file.

Warning: ill-conditioned perimeters may result in error.

6.21 .3 **Calculation of Area**

Once the data has been read the area can be calculated using the 'Calculate Area Button'.

When complete the area plot will be displayed on screen, at the Z=0 plane. The calculated area value is shown at the bottom of the contour bar and is also echoed to the command line:

```
Beginning Calculation using a tolerance of: 185 and grid size of: 10
Area Calculation Complete.
Low HIC Area = 68.45 %    |    734434    |    Input: HIC_AREA_EXAMPLE.blob
```

The number in the centre is the actual low HIC area as calculated in model units.

The contour bar also contains a 'By Input Points' measurement. This is simply the ratio of high/low HIC points as contained in the input file. It may be of interest but is not as accurate as the calculated value.

The tool only interpolates, it does not extrapolate.

6.21 .4 Calculation Method

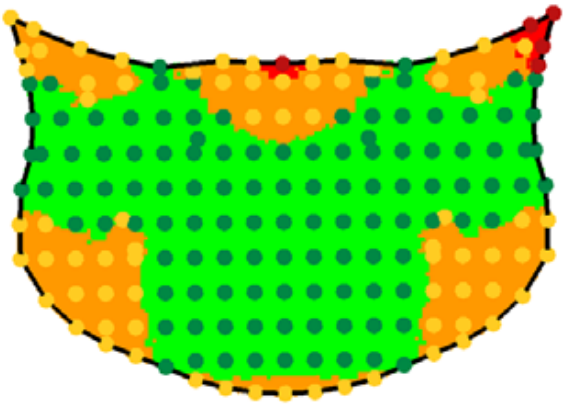
The steps for calculating the low HIC area % are as follows:

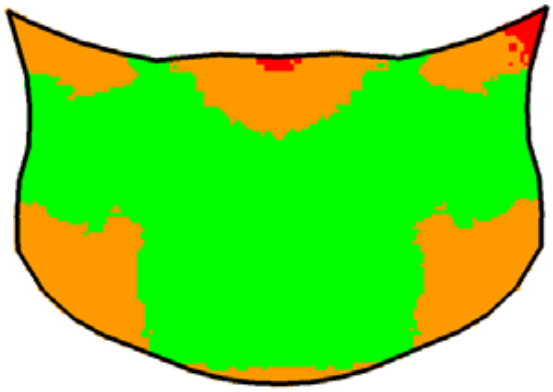
1. Create a 'fine' grid of points based on the grid size input parameter. This is typically 10% of the input point spacing.
2. Determine which of the fine points are within the perimeter.
3. Calculate a HIC value for each of the fine grid points by interpolating from the coarse spaced input points.
4. Calculated the ratio of fine low HIC values to the total.

6.21 .5 Display Options

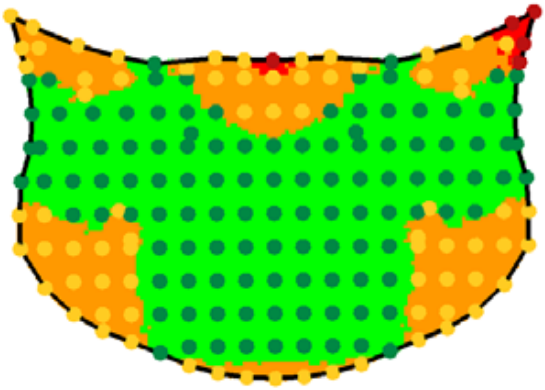
Several options are available for changing the display:

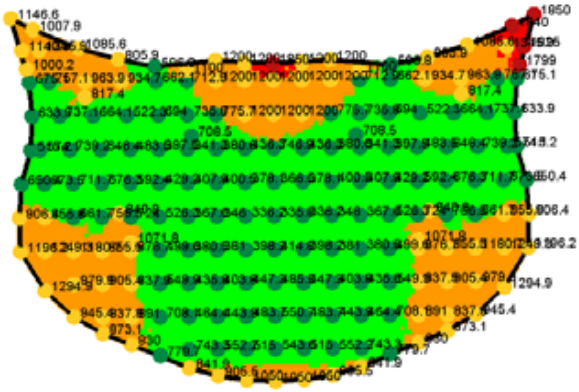
Show Input Points - Toggles whether the inputs are drawn:



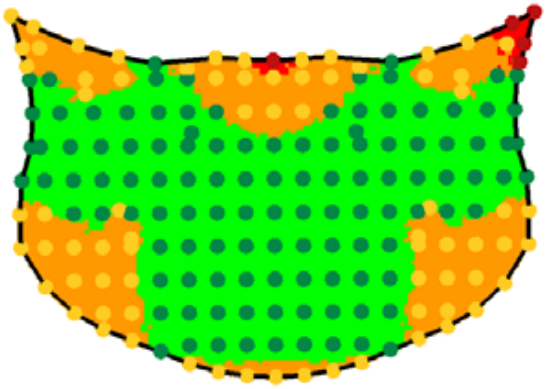


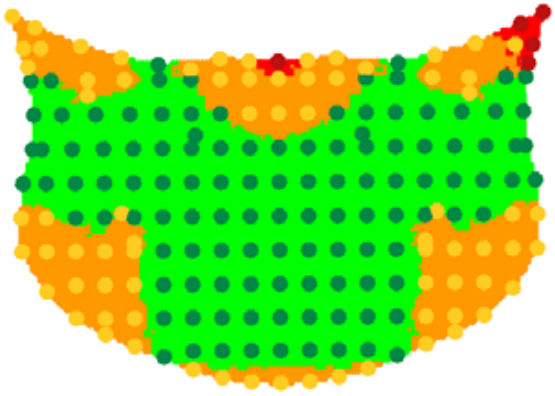
Show Input Value - Allows the user to select whether the input values/names are displayed.



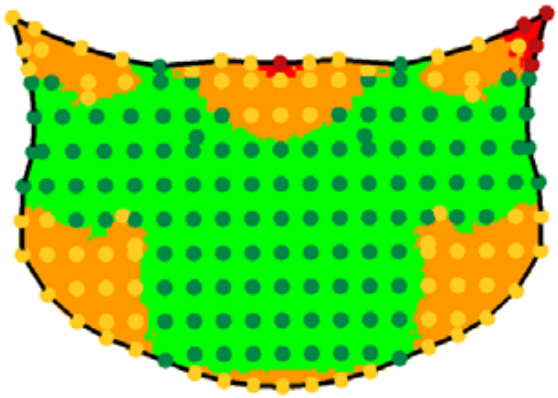


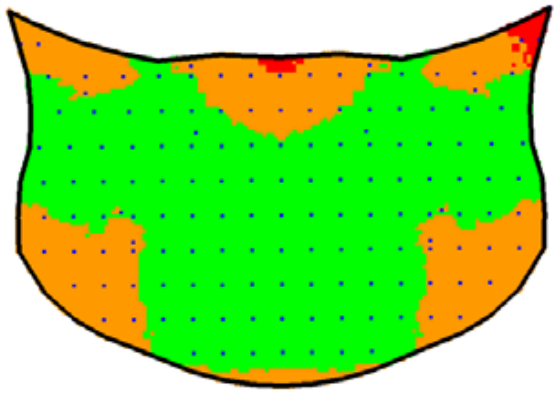
Show Perimeter - Toggles whether the perimeter is drawn:



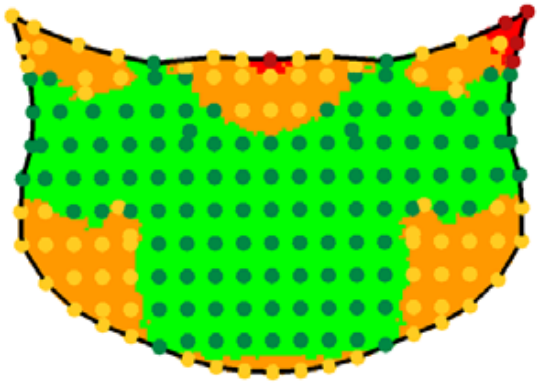


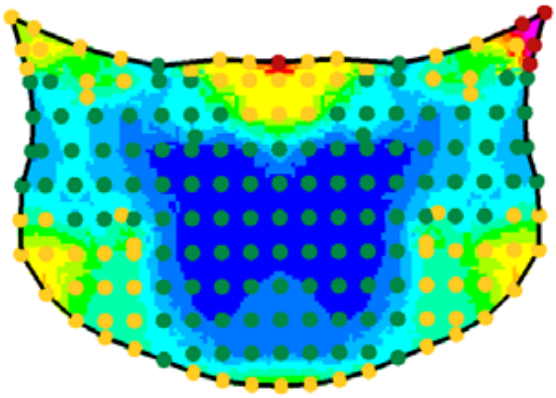
Tricolour Input - Toggles whether the input values are displayed as large tri-colour 'blobs' or small blue squares:



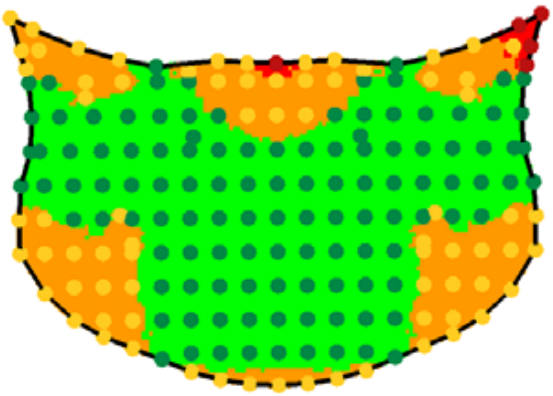


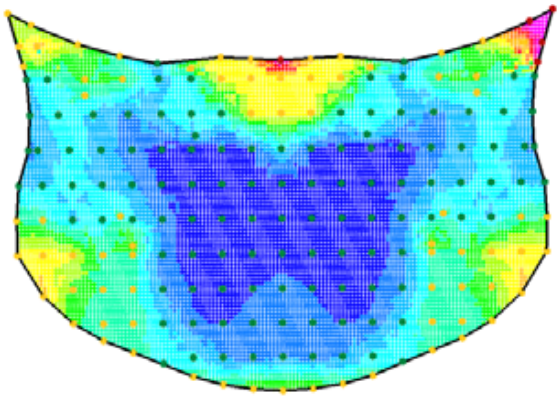
Tricolour Output - Toggles whether the area calculation is displayed in tri-colour or as a regular contour plot:





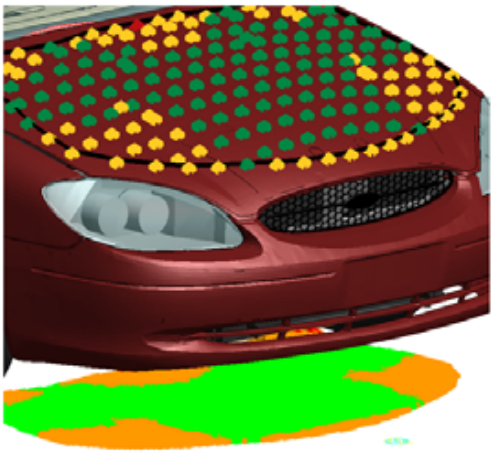
Change Symbol Size - Changes the size of the drawn data (input/output and perimeter):

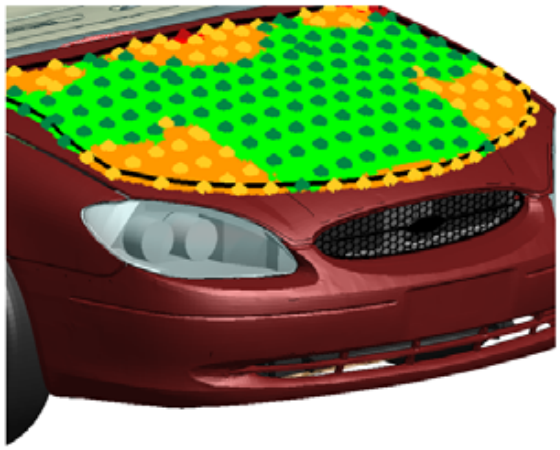




Y-Offset - This applies an offset to the y-coordinate of the displayed data it is useful when displaying multiple plots in the same session.

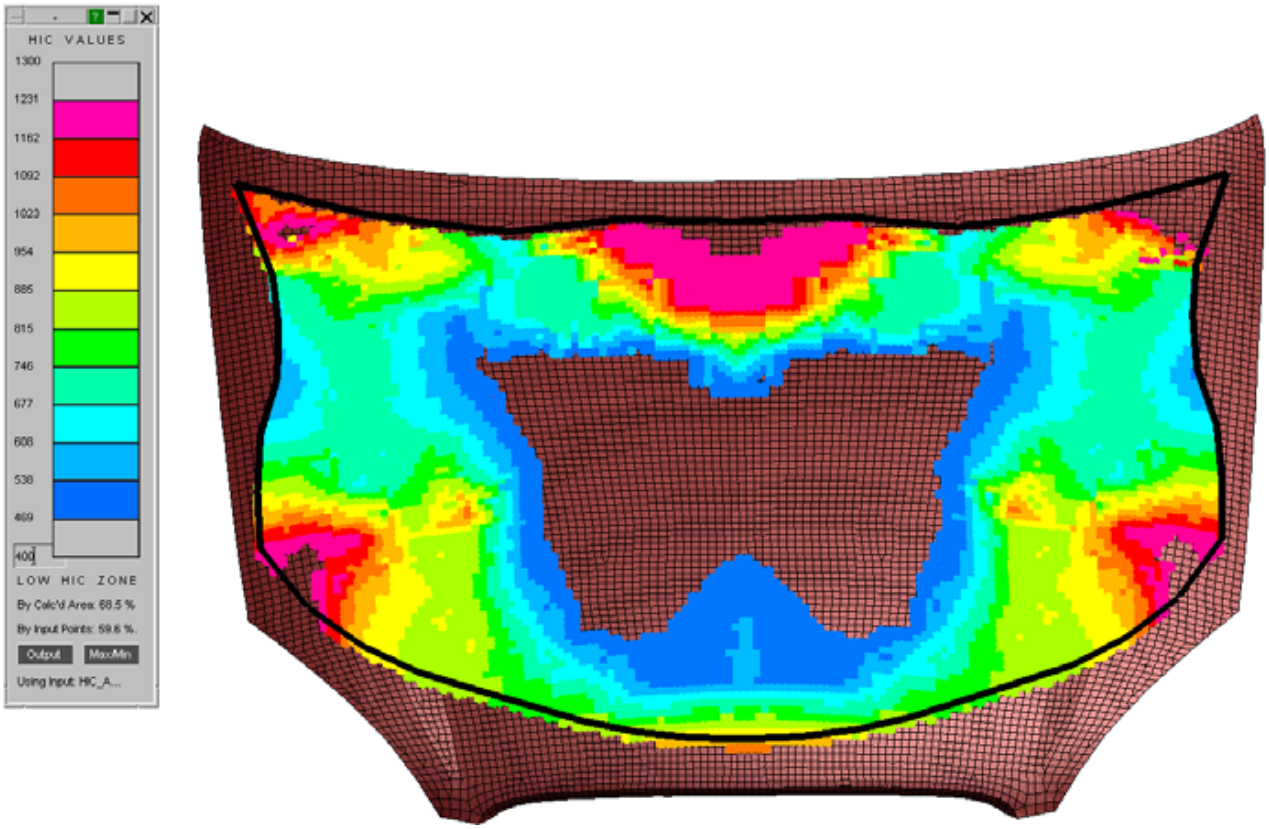
Projection - Can be used to project the calculated values from the z=0 plane to a mesh:





Note on output display mode:

When showing the calculated values the colourful contour plot, it is possible to turn on/off contour bands by clicking the colours in the contour bar. It is also possible to change the upper and lower bounds of the plot by typing over the values in the contour bar.



6.21 .6 Calculation Parameters

Users must select whether to use a GTR area based calculation or EuroNCAP (v8) based scoring.

For GTR:

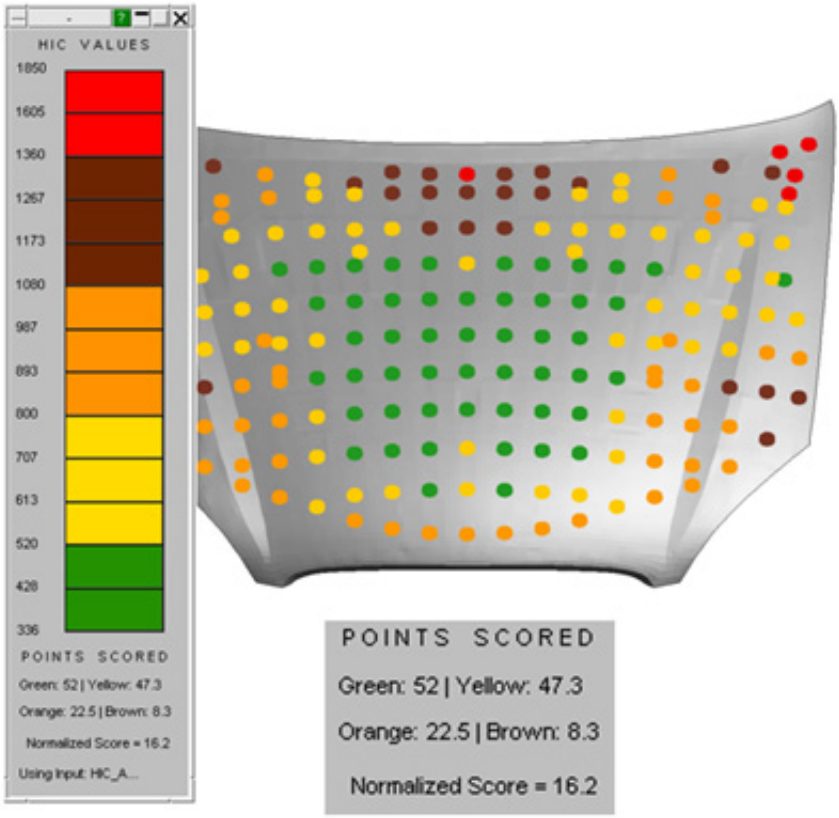
The low HIC area % is calculated using the given low HIC value. The high HIC value is not used in the calculation but is used to identify the ‘red’ areas in the tri-colour plots. Both HIC values can be changed by the user and the plot/result will update automatically. The fine grid size, as explained, in Calculation Method can also be changed. Smaller values of grid size will result in longer computation time and are not necessarily more accurate.

All calculation parameters can be saved in the oa_pref file using the ‘Save Settings’ button.

For EuroNCAP:

Yellow, orange, brown and red banding values are shown and can be edited by the user. HIC points are scored based on the band they belong to as per EuroNCAP v8 regulation.

For ENCAP no area calculation is required. Instead the output is a simple blob plot corresponding to the colour bands.



All calculation parameters can be saved in the oa_pref file using the ‘Save Settings’ button.

6.21 .7 Analysis Tools

In addition to calculating the low HIC area % there are several other tools available to the user:

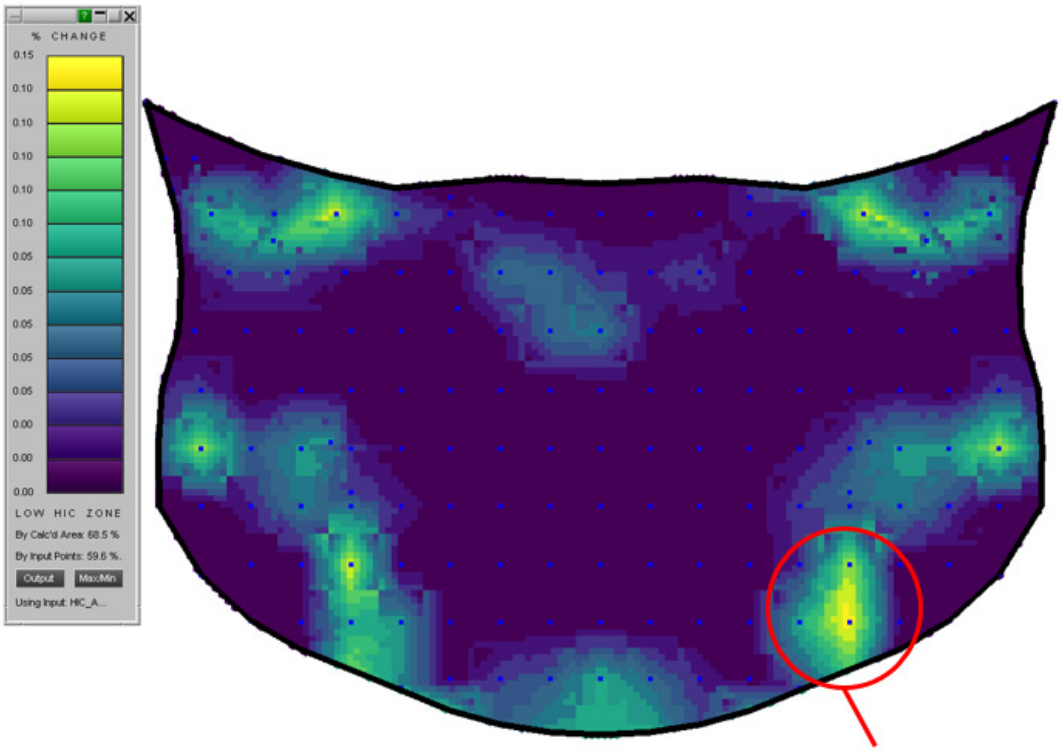
6.21 .7.1 **Area Sensitivity Study (GTR Only)**

It can sometimes be difficult for the pedestrian impact engineer to identify which areas should be targeted to improve the overall low HIC % area. Some points may be close to the low HIC boundary and deemed to be ‘easy wins’ but have little impact to the total % area. Whereas other points may be well above the low HIC boundary but have a greater influence.

The ‘Sensitivity Button’ allows the user to input a HIC ‘delta’. PRIMER will then:

1. Take input point 1 and add/subtract delta.
2. Recalculate low HIC area %.
3. Store change in area from baseline.
4. Reset input point back to original value.
5. Take next input point and return to step 1).

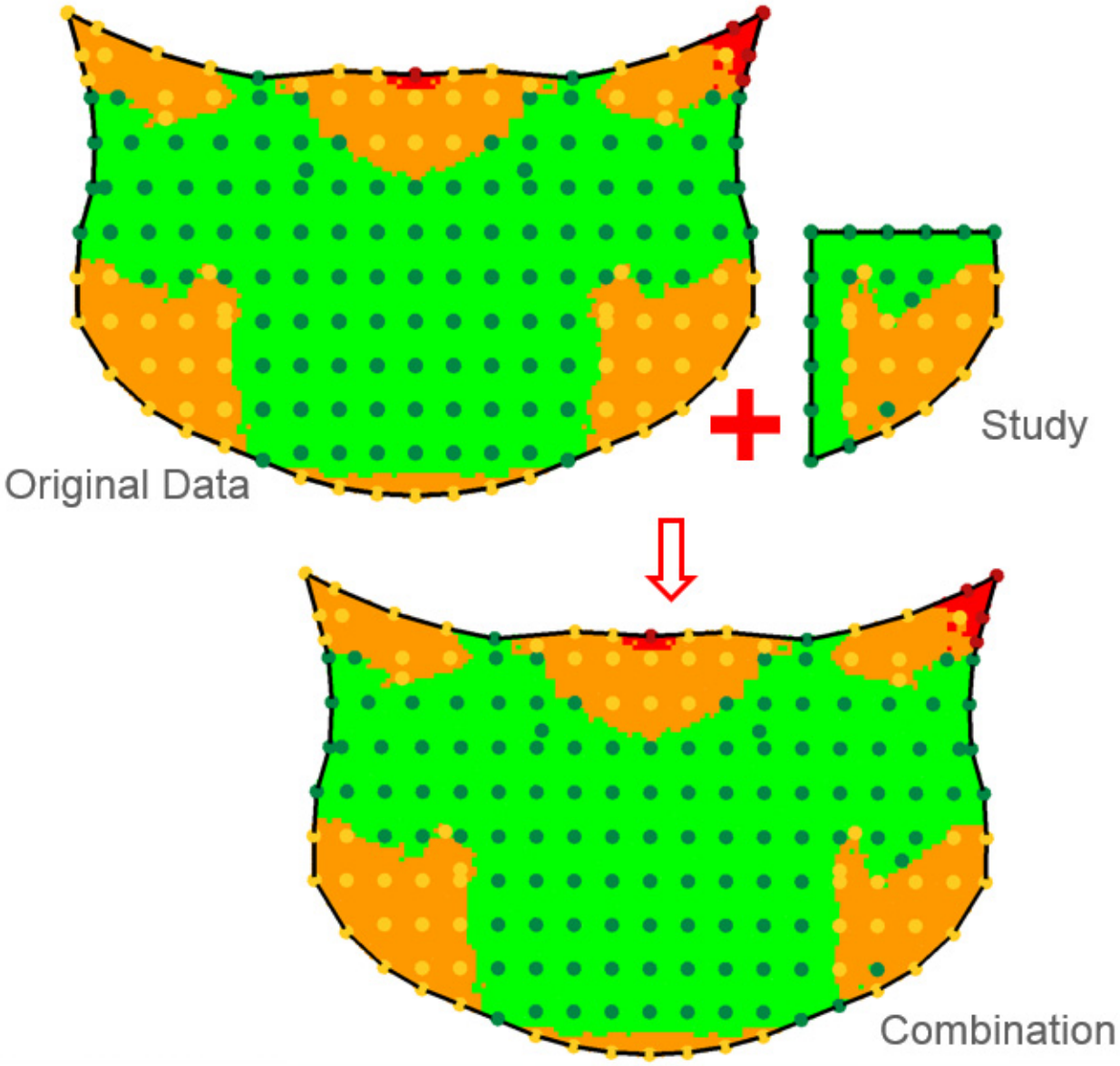
Once all input points have been analysed PRIMER will display a contour plot highlighting the areas which showed the greatest sensitivity to the given delta. These areas are those that will give the ‘greatest bang for your buck’ and should be prioritised in order to reduce low HIC % area.



Region most sensitive to improvement.

6.21 .7.2 **Combine Input Data**

Often pedestrian impact engineers will run small localised models over a specific part of the bonnet, rather than rerun the full sweep of impact points. The ‘Combine Input Data’ button allows the user to replace values from a full sweep with those from a sub-sweep; allowing the user to calculate a predicted full sweep result.



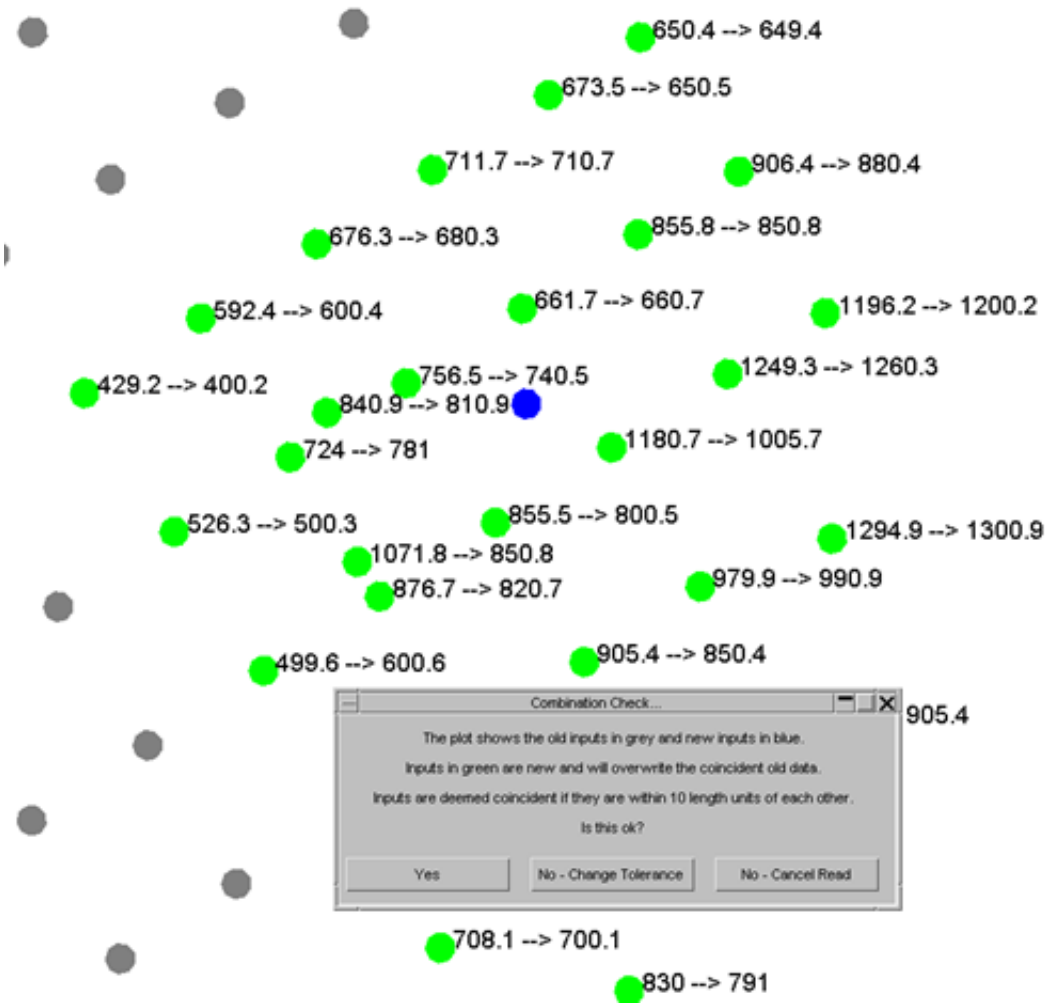
Prior to combining, PRIMER will display the values being changed:

Grey = Unchanged.

Green = Changed.

Blue = New.

An existing value is changed when a new value is found to be within a given tolerance (default is 10 length units).



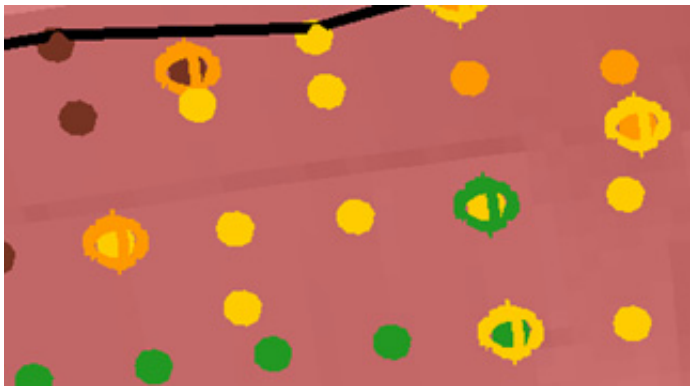
It is possible to write the combined data for future use using the ‘Write Combined Data’ button.

6.21 .7.3 **Target % Area**

The ‘Target % Area’ button allows the user to input the low HIC area % they are targeting. PRIMER will adjust the low HIC value so that the exact target is achieved. This provides the user with a rough estimate of how far their data is above/below from achieving the target.

6.21 .7.4 **Band Sensitivity**

The ‘Band Sensitivity’ button allows the user to input a HIC value. PRIMER will check each impact point to check whether a point is close to changing band (GTR or ENCAP). The points will be ringed by the colour band they are close to.



6.21 .7.5 **Edit Individual Point**

The ‘Edit Individual Point, allows user to edit the HIC value of a point by simply clicking it and typing in a new value. Area and scores can then be recalculated using the updated data value. This is useful for analysing ‘what if’ type scenarios.

6.21 .8 **Reading Additional Data**

The ‘Read Second File’ button allows the user to input an additional data file. This option is often combined with the *Y Offset* display setting to show the new data side by side. When using this function PRIMER will give the following message which can be continued:

Please note: this function is experimental and may be temperamental.

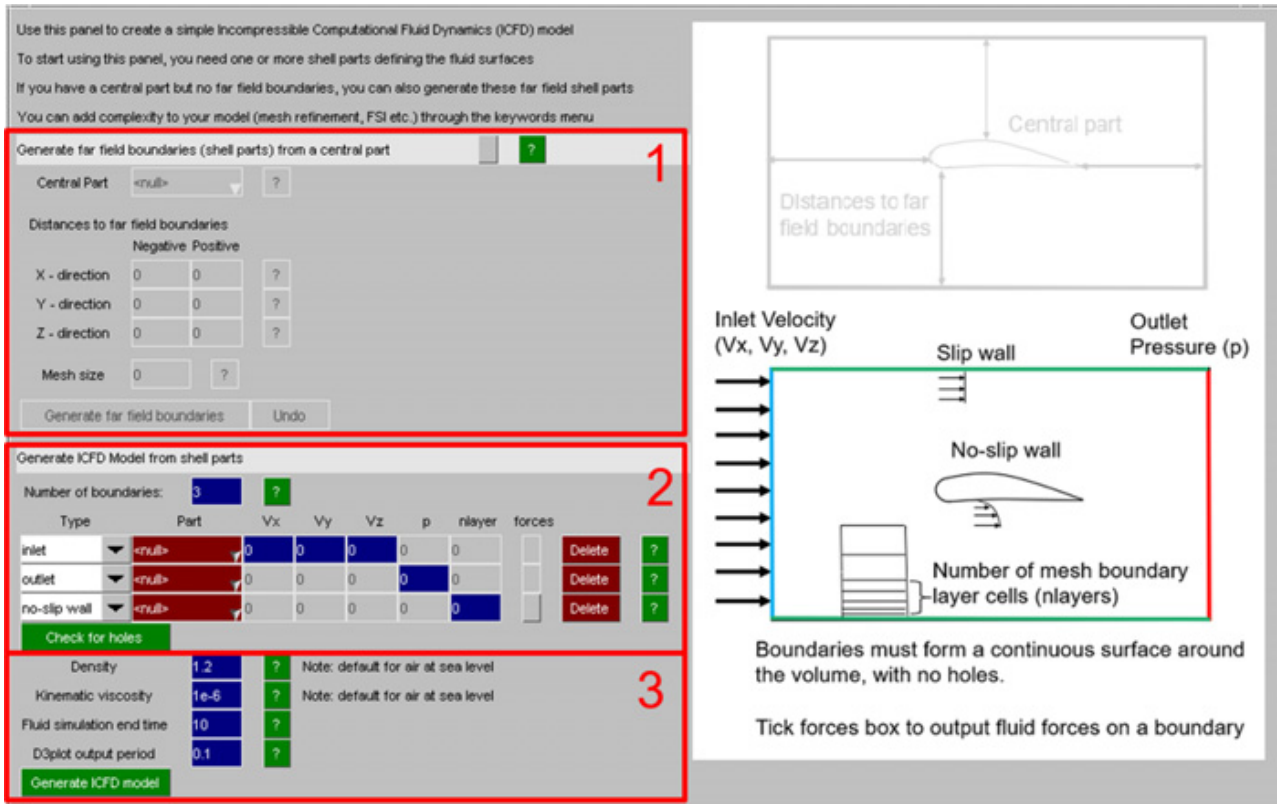
6.22 ICFD Setup script

The Incompressible Computational Fluid Dynamics (ICFD) solver can be used to simulate fluid flow problems where the fluid can be assumed to be incompressible. The solver can be used on its own, or it can be coupled to the solid mechanics or thermal solvers.

This ICFD model setup script enables the user to create a simple ICFD model from selected shell parts which represent the fluid boundaries. The user also has the option to generate far field boundaries around a part by selecting a central part and specifying the distances to the boundaries from this central part.

The model generated by this panel is a simple one and only a limited number of the ICFD features are used. However, a user wishing to create a more complex model may want to use this script to create a simple model first before adding complexity through the PRIMER keyword menu.

The following figure shows the input panel of the ICFD model setup script:



The first section of the input panel (highlighted in box 1 above) creates shell parts which are to be used as far field boundaries. Six shell parts are created, one for each direction, and the parts are made up of shell elements. The creation of far field boundaries may be required if a user has a part over which they wish to simulate the external flow (e.g. the airfoil shown in the figure). If the user already has all the shell parts defining the fluid boundary, then this section can be skipped.

The far field boundary generation can be activated by clicking the tick button at the top of section 1 in the figure above. The following options are available in the first section:

Central Part	This is a part which is used as a reference for creating the far field boundaries.
Distances to far field boundaries	These are the distances between the central reference part and the far field boundaries. Six values are required in this table, with one distance for each direction (negative x, positive x, negative y, positive y, negative z, positive z). The distances are measured from the closest node on the central reference part e.g. the minimum x coordinate of the part is used for the distance to the negative x boundary far field.
Mesh size	This is the mesh size on the far field boundaries. Select a large mesh size initially before reducing it if required.

The second section of the input panel (highlighted in box 2 above) enables the user to create an ICFD model from shell parts. These parts can be those created in section 1 of the input panel, or pre-existing shell parts. The ICFD model will

be generated as a separate new model.

The input table is used to input the numbers of shell parts and the ICFD boundary information. The following information must be specified on each row:

Type	Select one of the following: **Inlet** A boundary with a prescribed velocity (*ICFD_BOUNDARY_PRESCRIBED_VEL). **Outlet** A boundary with a prescribed pressure (*ICFD_BOUNDARY_PRESCRIBED_PRE). **No-slip wall** A boundary with zero normal velocity and zero tangential velocity (*ICFD_BOUNDARY_NONSLIP). **Slip wall** A boundary with zero normal velocity and no conditions on tangential velocity (*ICFD_BOUNDARY_FREESLIP).
Part	The ID of the shell part to be used as the boundary. An *ICFD_PART is created for each part, each shell element is translated into a *MESH_SURFACE_ELEMENT and each node is translated into a *MESH_NODE.
Vx, Vy, Vz	For **Inlet** type boundaries, these are the velocity components in the x, y and z directions.
p	For **Outlet** type boundaries, this is the prescribed pressure.
nlayer	For **No-slip wall** and **Slip wall** type boundaries, this is the number of mesh boundary layer cells. *MESH_BL_SYM cards are automatically added to inlet and outlet boundaries. This ensures that the boundary layer cells continue all the way to the inlet or outlet boundaries.
forces	For **No-slip wall** and **Slip wall** type boundaries, this box can be ticked to output the fluid force time history *ICFD_DATABASE_DRAG.

The number of rows in the table can be changed by either clicking the “Delete” button at the right-hand side of each row or by editing the “Number of boundaries” field just above the table.

To create an ICFD model, the ICFD fluid volume must be entirely enclosed by the boundaries, with no holes and no duplicate nodes at the edges. The “Check for holes” button can be used to check if this is the case.

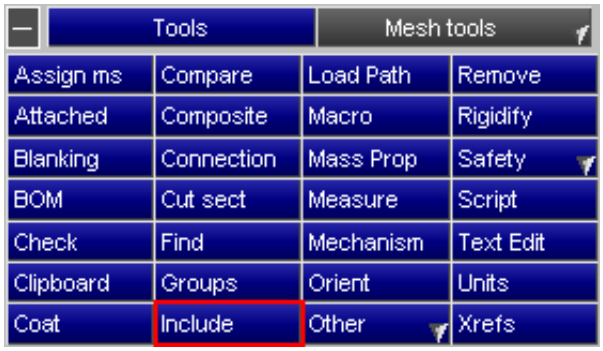
Once the boundaries have been written to this table, and checked for holes, some essential parameters must be entered before the ICFD model can be generated (section 3 in the figure above):

Density	This is the fluid density to be used in the *ICFD_MAT card.
Kinematic viscosity	This is the fluid kinematic viscosity to be used in the *ICFD_MAT card.
Fluid simulation	This is the ICFD simulation end time to be used in the *ICFD_CONTROL_TIME card.
D3plot output period	This is the time interval between states to be used in the *DATABASE_BINARY_D3PLOT card.

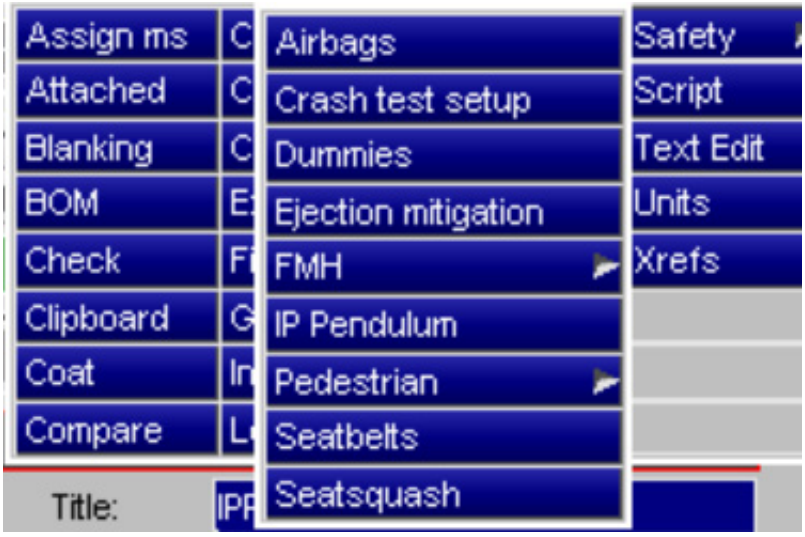
Once values have been inputted for these parameters, click “Generate ICFD model”. This will create a new model with all the cards required to run a simple ICFD simulation. ICFD models must be run in a double precision solver.

6.23 **INCLUDE** Controlling *INCLUDE files.

The **Include** menu in the **Tools** panel is covered in a separate section of the manual - see section 3.13 for details.
From version 12 of PRIMER the **Include** tool can also be started from **INCLUDE** in the keywords menu.



6.24 INSTRUMENT PANEL PENDULUM



The IP Pendulum function can be used to specify multiple Instrument Panel Pendulum impact models for ECE R21 or FMVSS202. It is accessed under **SAFETY** > **IP Pendulum**.

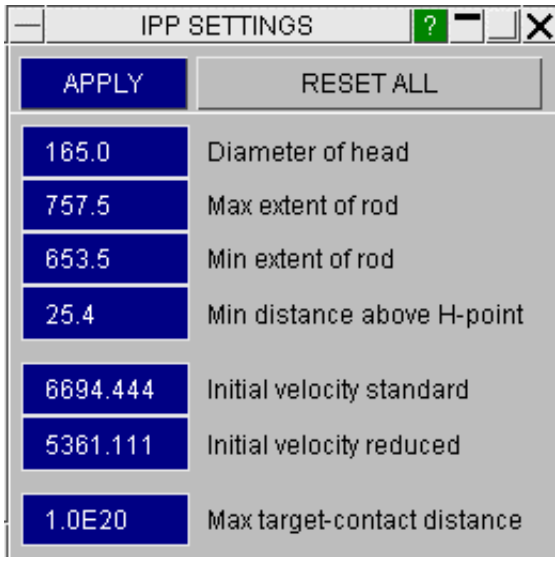
This function supports interactive and batch model processing. Automated positioning and depenetration is available.

The initial screen for setting up IP Pendulum parameters is shown below. Post *END data, if available, is read in by default. Pendulum to IP contact can be created by clicking on the Create button.

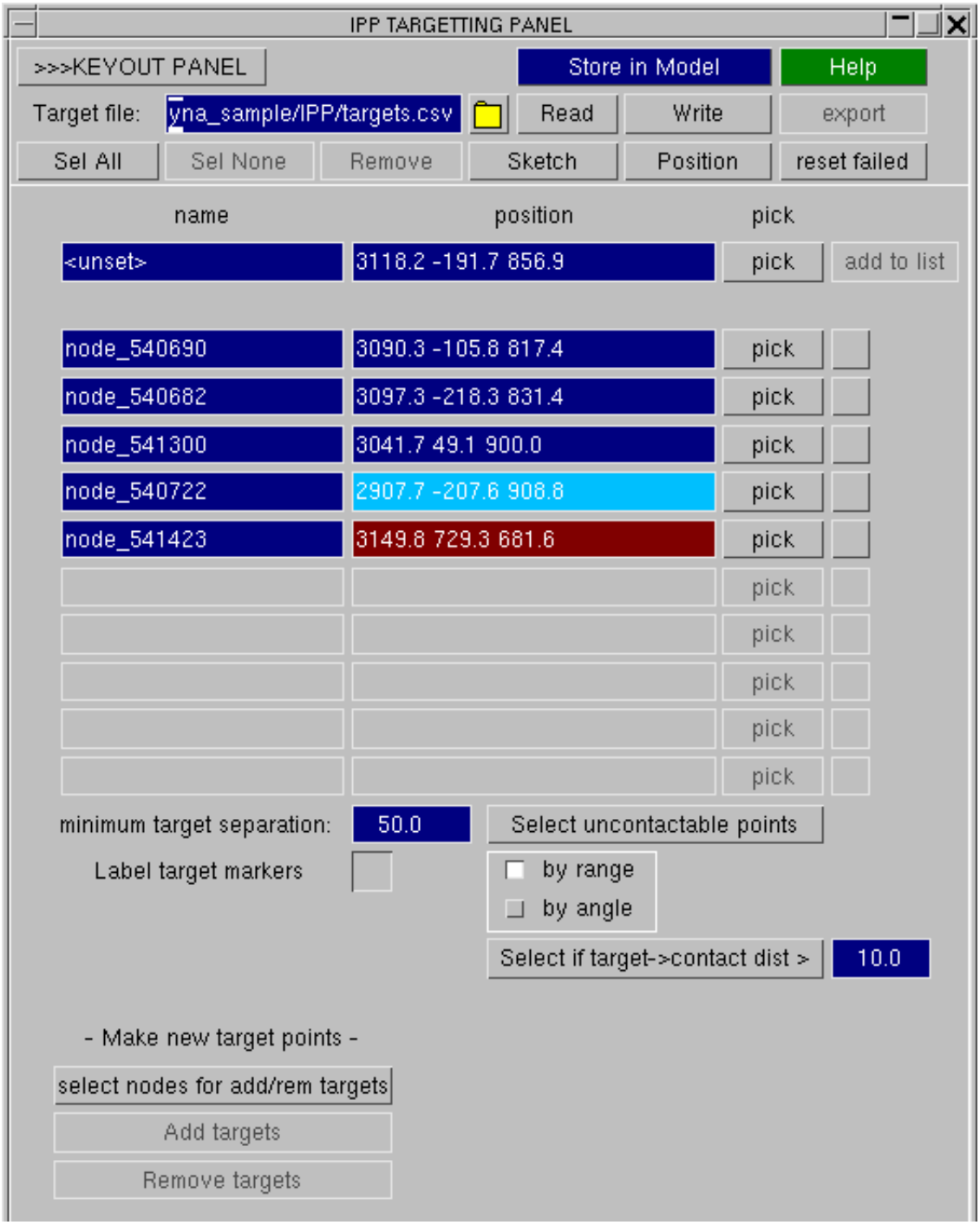
Standard and reduced impact airbag velocities can be specified using the appropriate text box and popup. Base and forward H-point coordinates can also be specified in this panel. The **IPP targetting panel** can be reached by clicking on the appropriate button.

The setting panel enables the user to change the default settings which are defined as per regulation. These values are used by the positioner to determine whether or not a target point is legitimate. The head diameter should be consistent with the model used.

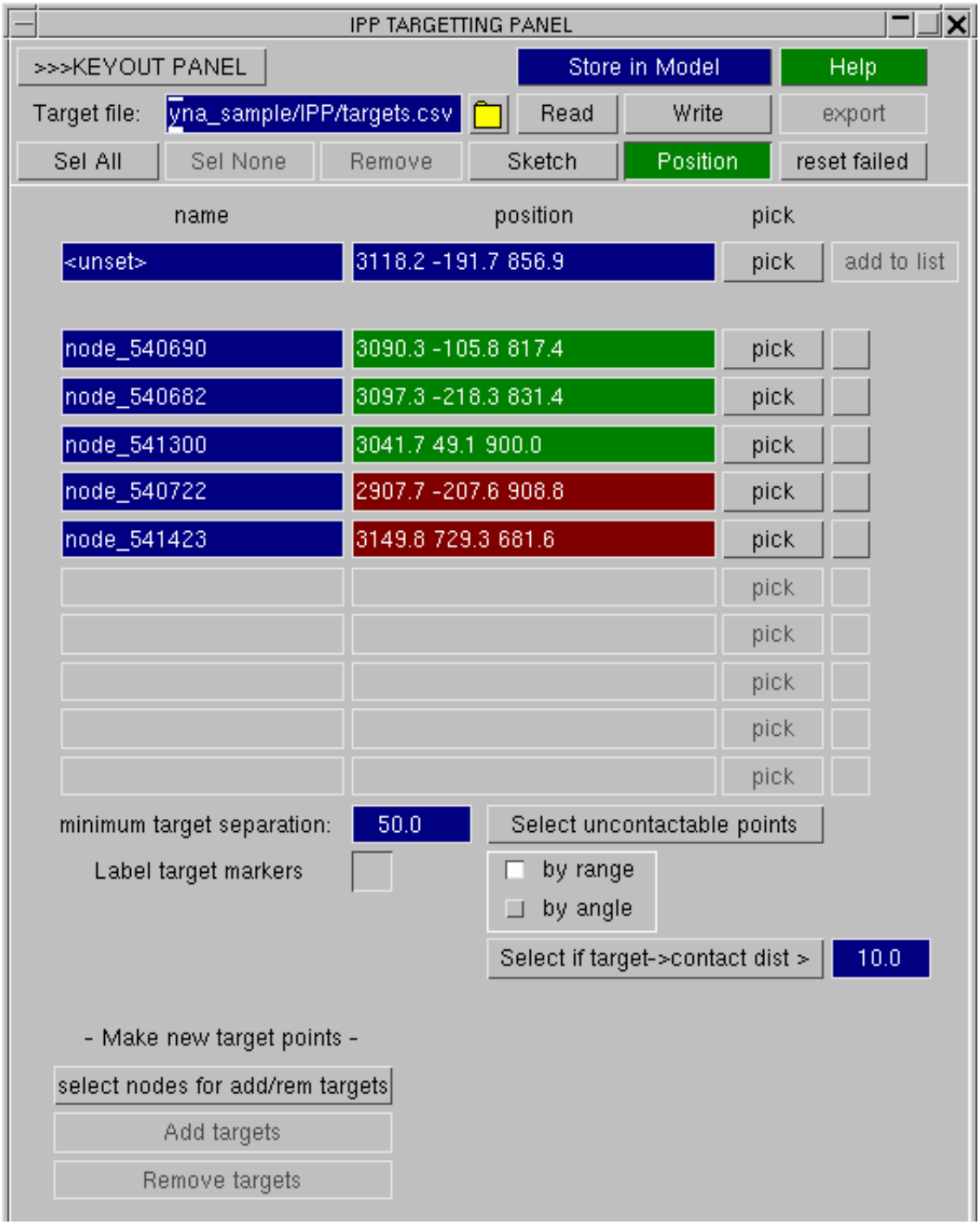
The user setting **Max target-contact distance** (accessed under **settings**) will exclude from the list of succesfully positioned points, any which fall outside this limit.



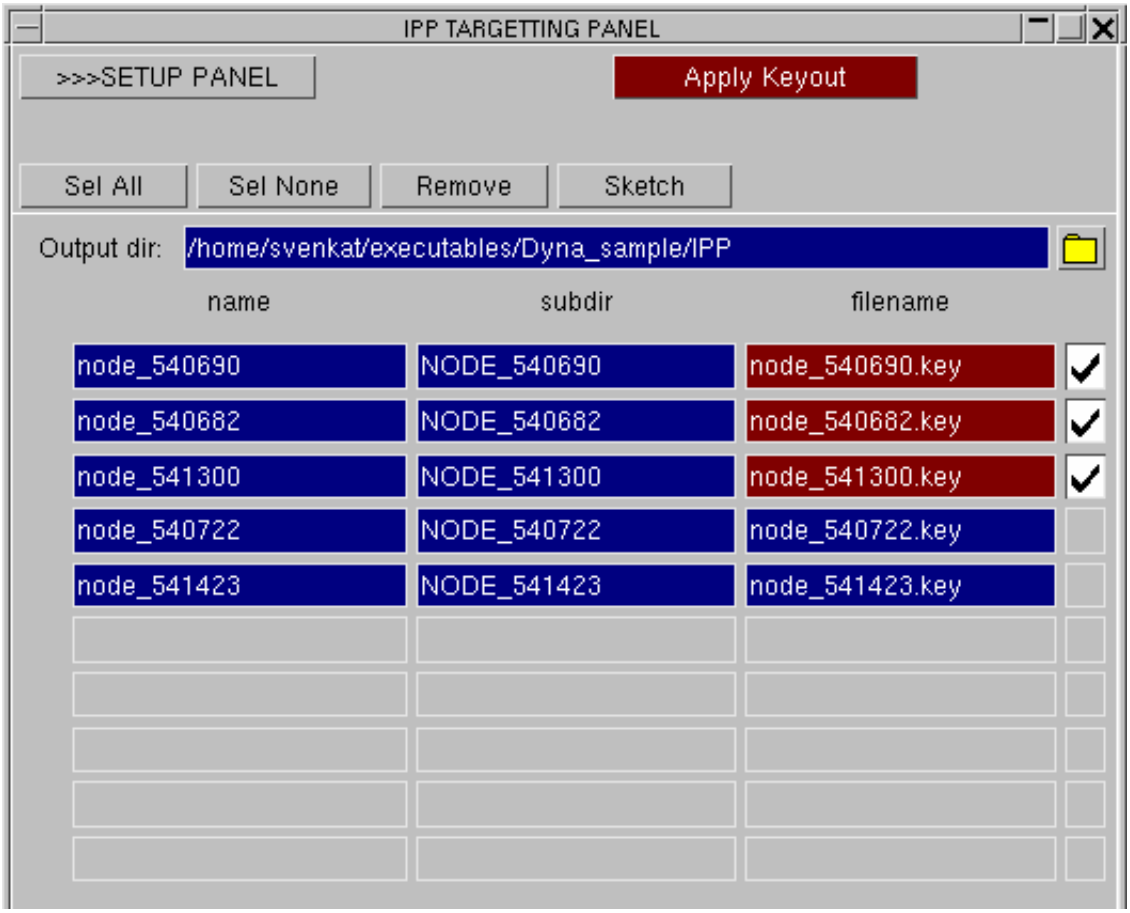
The IPP targetting panel, as shown below, displays existing targets in the model. A csv file can be read in using the **Read** button to load new points. New target points can be also added using the **sel nodes for add/rem targets** button. Prior to positioning, target points appear on a light blue background if they are misaligned with the trim normal. This indicates that they probably cannot be contacted by the pendulum. A red background indicates that they definitely cannot be reached by the pendulum. A dark blue background suggests that points are not yet positioned. When the **Position** button is clicked, selected targets are positioned and de-penetrated.



Target points that have been successfully positioned are shaded green. Failed points are shaded red.

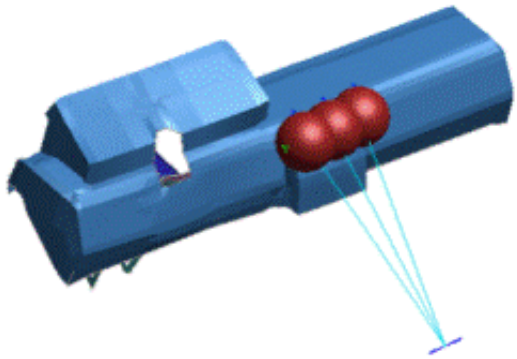


The list of target points can be stored in a model or written to a csv targetting file for batch processing. This can be done by switching to the **KEYOUT PANEL**. File path, sub-directory names, and file names can be modified by users prior to keyout.

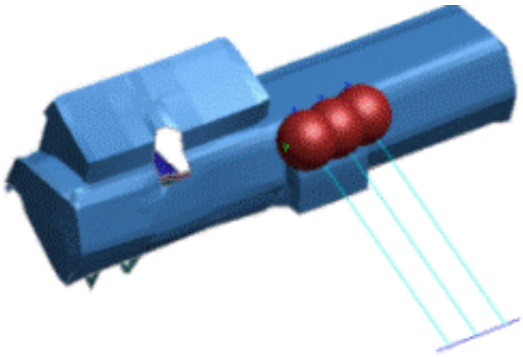


In ECE R21 mode the base of the pendulum is located at the H-point.

In rotated mode a subsequent DEFINE_TRANSFORM is applied to rotated the line of flight onto the trim normal.



In FMVSS201 mode after achieving a position, the base is translated in Y to align with each impact point.



IPP parameters can be specified in the input csv file in the following manner when the IPP build is run interactively:

```
IPP
target_point_start, px, py, pz, nx, ny, nz, h_flag, r_flag, v_flag
node_540690, 3090.3, -105.8, 817.4, 0.0, 0.0, 0.0, 0, 0, 0
node_540682, 3097.3, -218.3, 831.4, 0.0, 0.0, 0.0, 0, 0, 0
node_541300, 3041.7, 49.1, 900.0, 0.0, 0.0, 0.0, 0, 0, 0
node_540722, 3157.7, -247.6, 908.8, 0.0, 0.0, 0.0, 0, 0, 0
```

px, py, pz are the target points

h_flag = 0 indicates standard H-point, h_flag=1 forward H-point

v_flag = 0 means use standard velocity, v_flag=1 means use reduced velocity

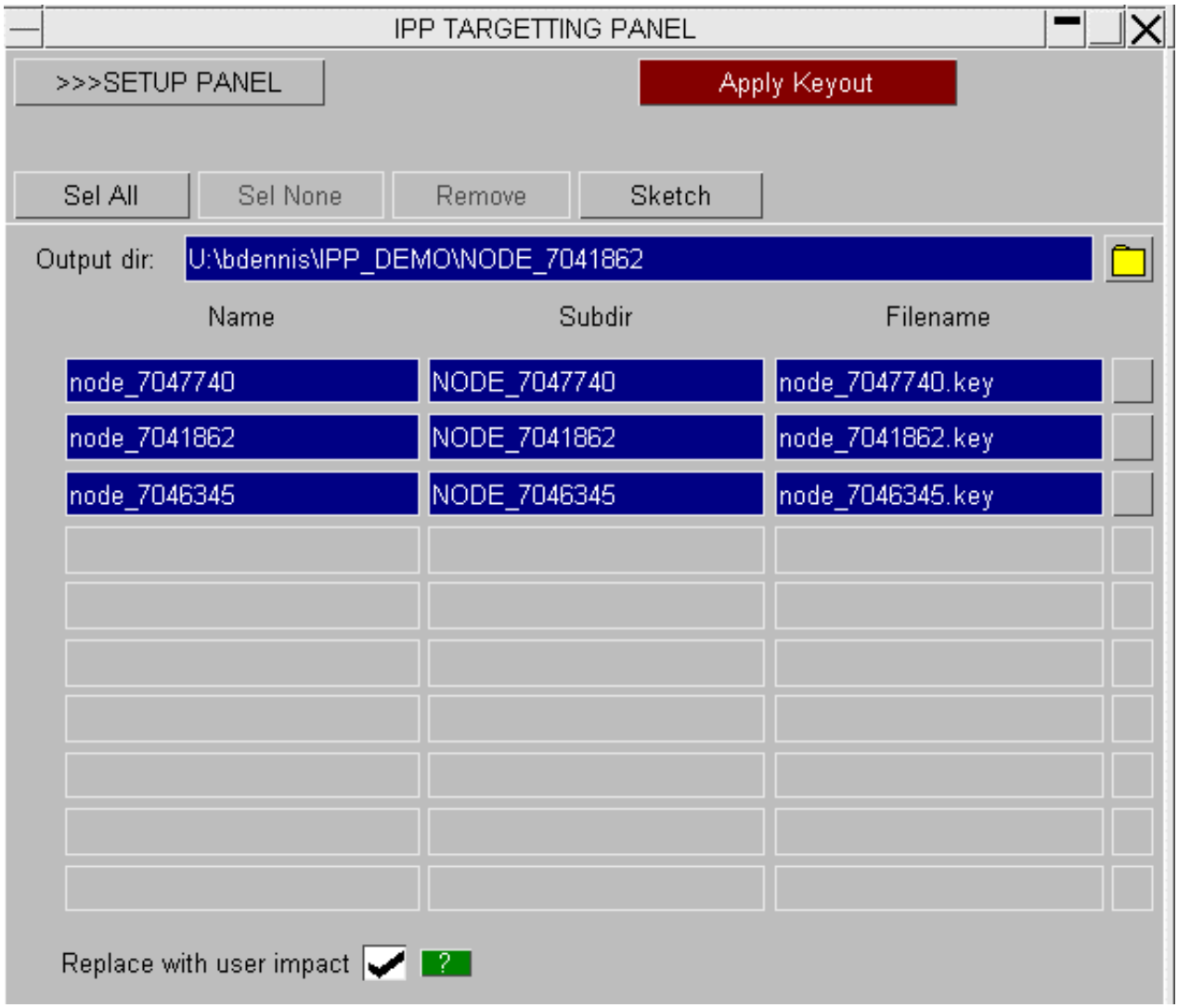
r_flag = 0 use ECE R21 position without rotation

r_flag = 1 use ECE R21 position with rotation so that line of flight aligns with trim normal

r_flag = 2 use FMVSS201 so that hinge is translated in Y to align with target point

Additional information would be required if the IPP build is run in batch mode. In this case, the csv file can be specified as follows:

```
IPP
INCLUDE, C:\test\include.key
IMPACTOR, C:\test\pendulum.key
CONTACT, pendulum to trim
HPOINT,     3400.0, 394.0, 220.0
HPOINT_F,   3445.0, 394.0, 239.0
VELOCITY,   6694.1
VELOCITY_R, 5361.1
OUTPUTDIR, C:\test
target_point_start, px, py, pz, nx, ny, nz, h_flag, r_flag, v_flag
node_540690, 3090.3, -105.8, 817.4, 0.0, 0.0, 0.0, 0, 0, 0
node_540682, 3097.3, -218.3, 831.4, 0.0, 0.0, 0.0, 0, 0, 0
node_541300, 3041.7, 49.1, 900.0, 0.0, 0.0, 0.0, 0, 0, 0
node_540722, 3157.7, -247.6, 908.8, 0.0, 0.0, 0.0, 0, 0, 0
```



A new feature has been added which enables the substitution of a user's own impactor model (presumably a linear impactor) before the model is keyed out.

This requires the preference setting **primer*replacement_ipp_impactor:<name>**

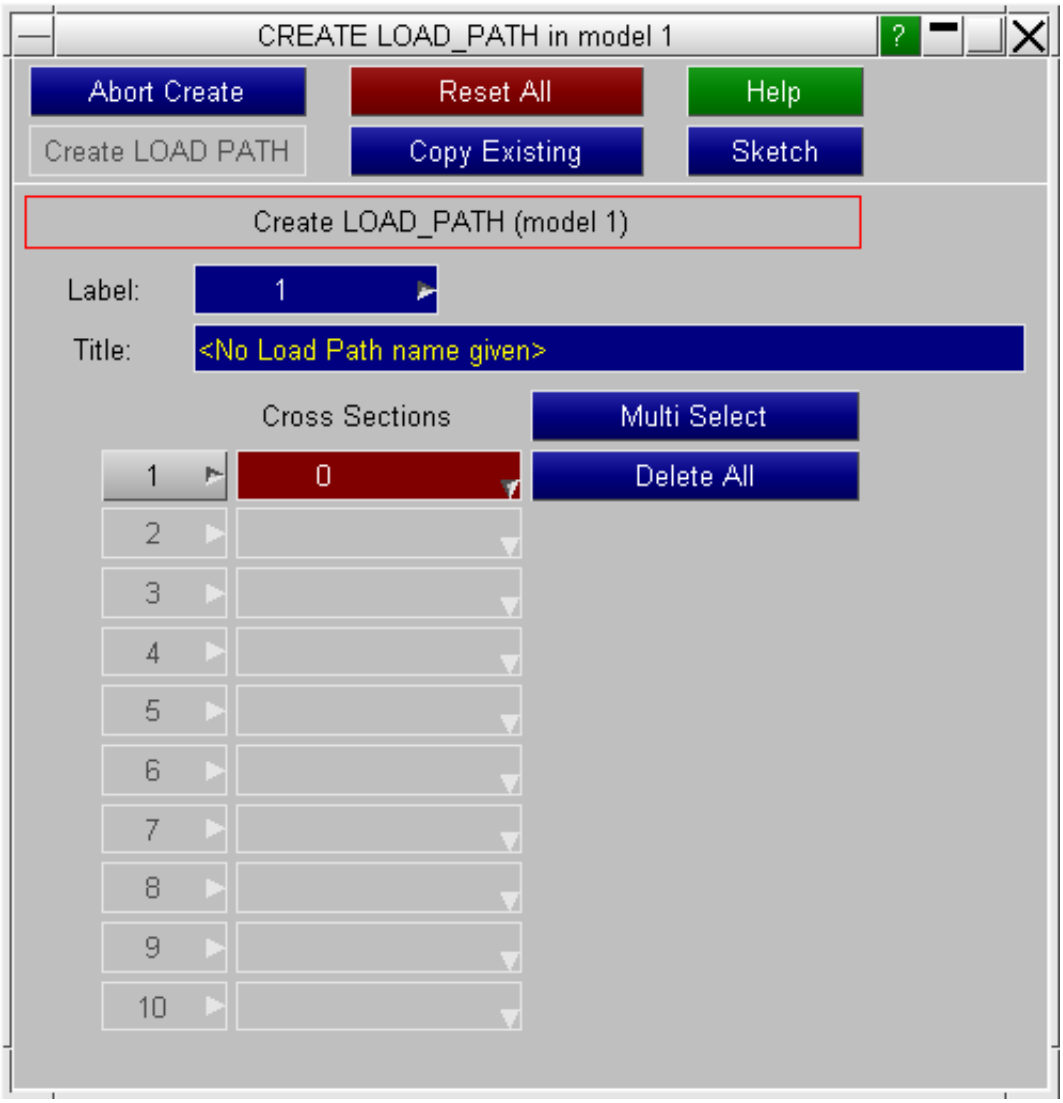
The replacement impactor is expected to reside at the origin and be aligned with global XYZ axes. Assuming position has been achieved, it will translated so the origin moves to the reference node (on the coordinate system) of the soon to be removed Arup impactor and rotated so the x-axis points along the line of flight. It will then be linearly depenetrated or moved to point of contact.

6.25 LOAD PATHS

The LOAD PATH panel allows you to create LOADPATHs joining the centres of *DATABASE_CROSS_SECTION definitions. They can then be viewed in D3PLOT if a ZTF file is generated. Forces calculated from the *DATABASE_CROSS_SECTIONs can be plotted on the LOADPATHs to make visualisation of loads through a structure easier.

6.25.1 Create a loadpath

To create a load path press the **Create** button to bring up a new floaing menu.



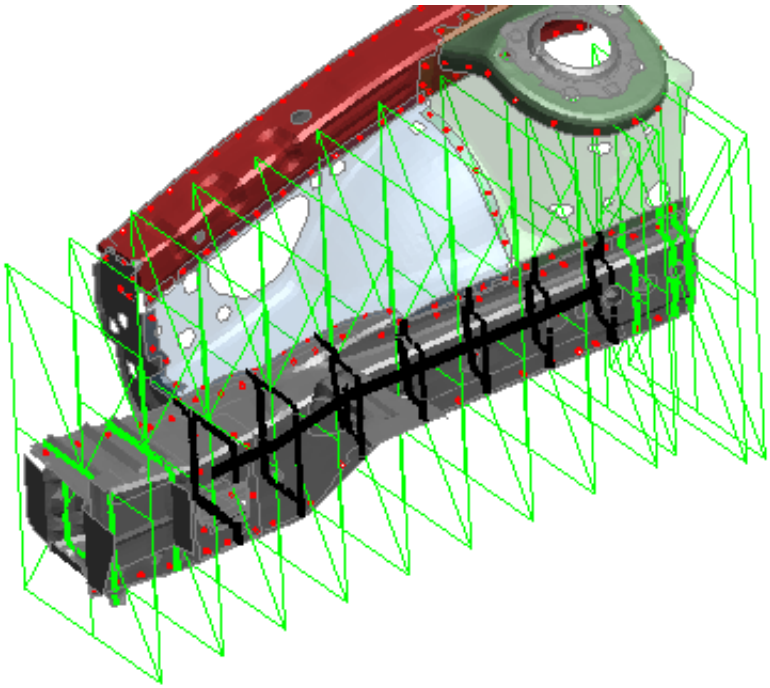
Press **Multi Select** and select the *DATABASE_CROSS_SECTIONs you want to join together. PRIMER will attempt to put them in a sensible order.

Press **Sketch** to view the order that PRIMER has used. If it is not what you want you can edit it using the dropdowns by each row to add and remove the cross sections.

The **Label** ID and and **Title** can also be modified.

When you're happy with the order, press **Create LOAD PATH** to create it.

When you save the model, the load path definitions will be written after *END keyword so you will not need to recreate them in new sessions of PRIMER.



6.25.2 Write a ZTF file

To view the load paths in D3PLOT you will need to write out a ZTF file. See Section 3.8.2 for how to do that.

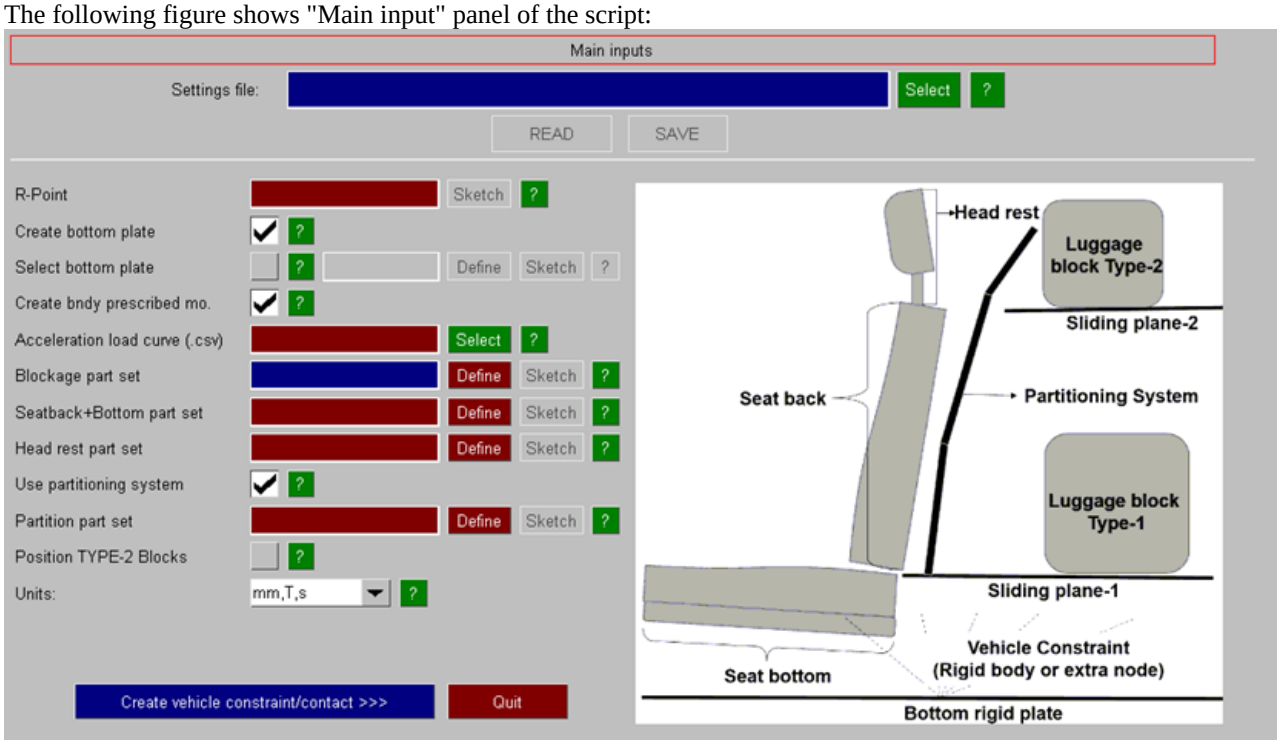
6.26 Luggage Retention Automation Script (ECE-R17)

Introduction

Strength of rear seat needs to be tested to protect the occupants against displacement of luggage during vehicle impact. The rear seat must be able to withstand a static test load simulating luggage impact. ECE-R17 is one of the tests that provide the assurance of sufficient strength resistance of the rear seat.

The luggae retention script creates plane for luggage masses to slide on and positions the luggage block of type-1 and type-2 if the seat-back is fitted with a head restraintis according to ECE-R17 specification. The script will also set up the analysis as per the regulation.

The following figure shows "Main input" panel of the script:



The following options are available on the main input panel:

Settings file	A settings file may be read or written which may include mandatory information and other data required to start calculation. The reading/writing of a settings file is optional.
R-point	R-point is the relative location of the seated dummy's hip point when the seat is set in the rearmost and lowermost seating position.
Create bottom plate	Select if you wish to create a bottom rigid plate part on which vehicle will be mounted. A longitudinal horizontal load shall be applied on bottom rigid plate.
Select bottom plate	Use this option to select an already existing bottom rigid plate in the model on which vehicle is mounted. A longitudinal horizontal load shall be applied on bottom rigid plate.
Ceate bndy prescribed motion	Select if you wish to create a *BOUNDARY_PRESCRIBED_MOTION on bottom rigid plate.
Blockage part set	Select part set that luggage blocks should not impinge on
Seatback+Bottom part set	Select part set consisting of PART(s) that makes up the rear seat excluding head restraint components
Head rest part set	Select part set comprising of head restraint parts.
Use partitioning system	Select if the vehicle has partitioning system(like net) which prevents luggage from sliding.
Partition part set	Select part set consist of partitioning system including net part.
Position TYPE-2 Blocks	Select if you wish to position TYPE-2 test blocks. By default, it will be OFF.
Units	Specify unit system of the selected model.

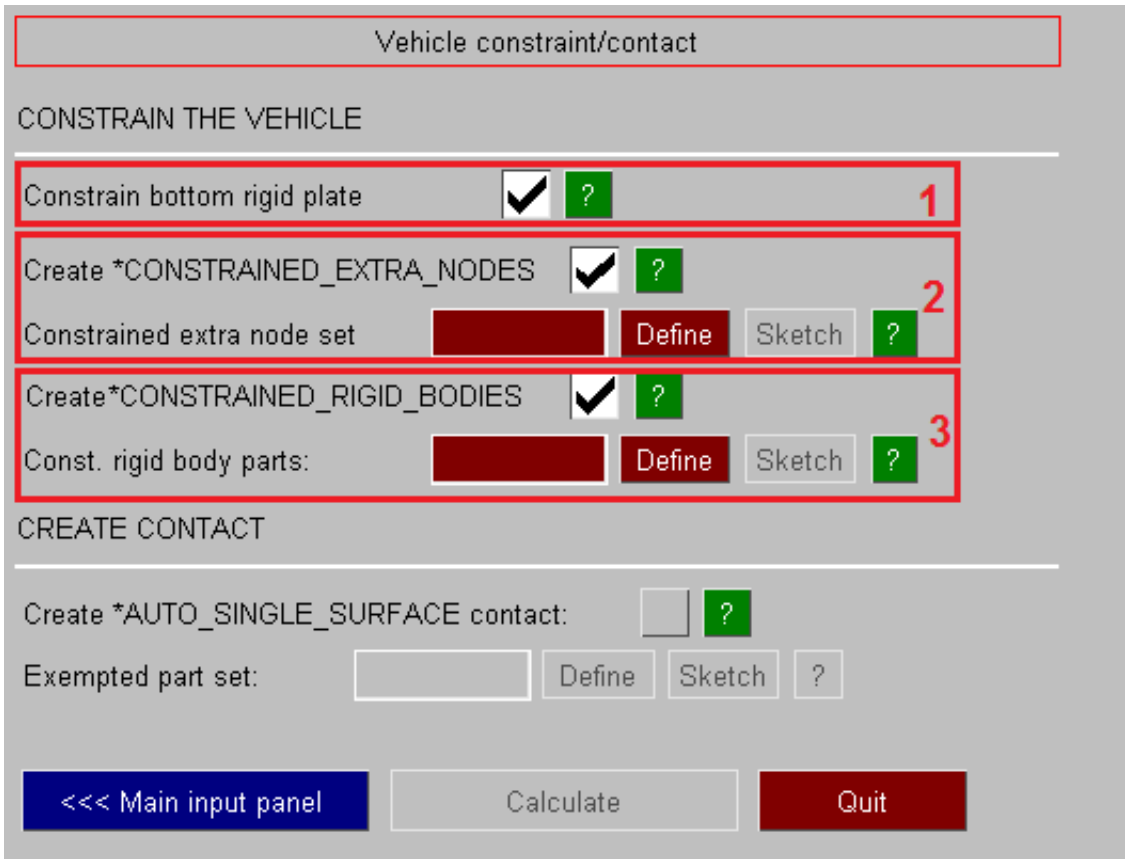
Constrain the vehicle

Click **Create vehicle constraint/contact** to create vehicle constraint and contact. All the options available on "Vehicle constraint/contact" input panel are optional.

Select "Constrain bottom rigid plate" option as shown below in red highlighted box-1 to constrain the bottom rigid plate in all DOFs (including rotations) except X-translations.

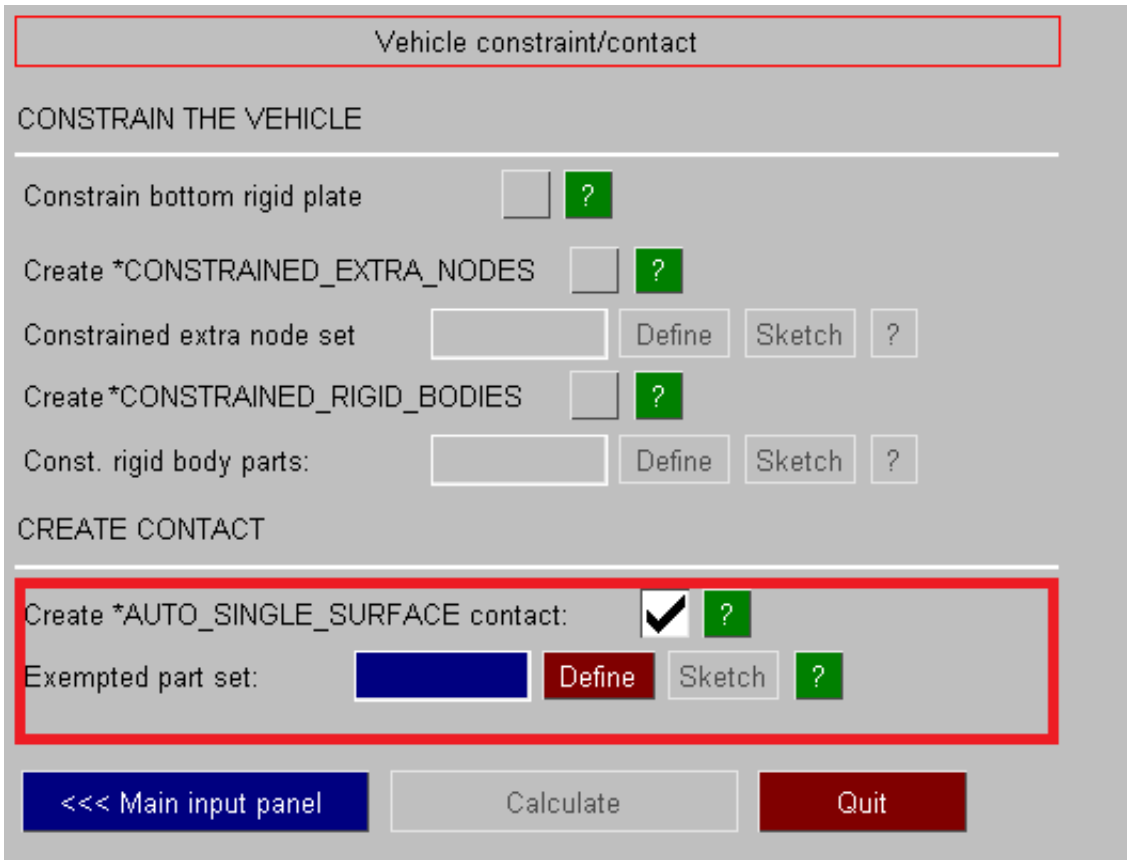
Select "Create *CONSTRAINED_EXTRA_NODE" option as shown below in red highlighted box-2 to constrain the vehicle with rigid plate using *CONSTRAINED_EXTRA_NODES. Use "Constrained extra node set" to provide node set for *CONSTRAINED_EXTRA_NODES.

Alternatively, "Create *CONSTRAINED_RIGID_BODIES" option as shown below in red highlighted box-3 can be used to constrain the vehicle on rigid plate. If this option is selected, use "Const. rigid body parts" to provide parts for *CONSTRAINED_RIGID_BODIES.



Create Contact

Select "create *AUTO_SINGLE_SURFACE contact"option as shown below to create a *CONTACT_AUTOMATIC_SINGLE_SURFACE with exempted part set in slave side. Impactor parts will get added automatically to the exempted part set. Use "Exempted part set" option to add any other part sets in the exempted set.



Calculate

Once all the input conditions have been defined, **Calculate** will become active. Press **Calculate** to start the calculation. After the calculation, the sliding planes/luggage blocks will be positioned in the vehicle as shown below:

6.27 MACROS

The **Macro** panel allows you to record and playback a sequence of commands in PRIMER similar to using visual basic macros in Excel etc. They are deliberately made to be human readable so that a macro can be edited by hand. The easiest way to see the format of a macro file is to record one and see what commands that PRIMER uses.

6.27.1 Recording a macro

To record a macro press the **Record** button at the top of the panel. Give a filename for the macro in the textbox or press the folder icon to select a new file location. Macro files in PRIMER should have the extension prm (**PR**imer **M**acro).

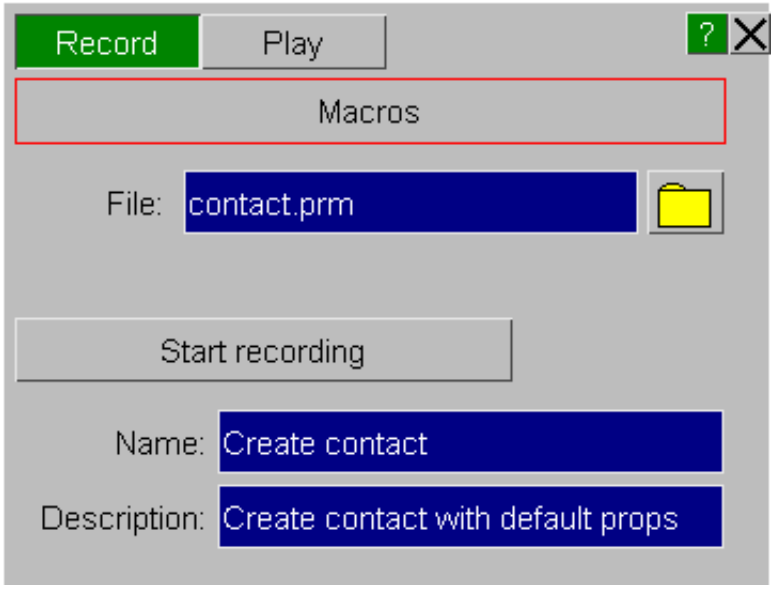
Once a filename has been given the **Start recording** button will become active.

A name and a description for the macro can also be given for the macro. These will be saved in the macro as `MacroName()` and `MacroDescription()` commands and are then used as the name and hover text for a button in the macro panel.

Press **Start recording**. Any commands that you now do in PRIMER will be written to the macro file.

The button will turn red and read **Stop recording**.

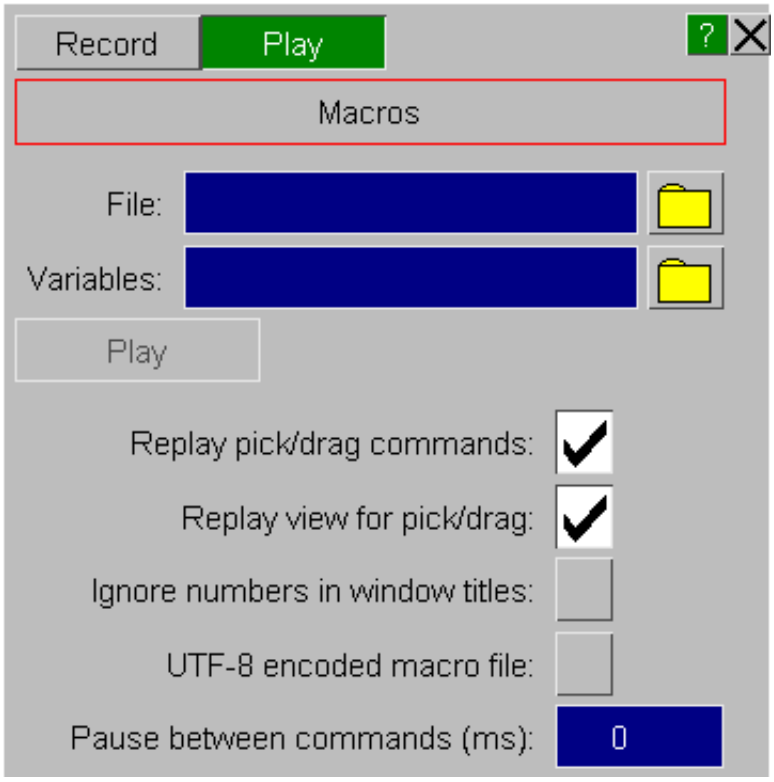
When you have finished recording the commands you want press the **Stop recording** button. The file will be closed and PRIMER will stop recording commands.



6.27.2 Playing a macro

To play a macro press the **Play** button at the top of the panel. Give a filename for the macro in the **File:** textbox or press the folder icon to select a macro file (macro files in PRIMER should have the extension prm (**PR**imer **M**acro)).

Once a filename has been given the **Play** button will become active.



Variables

A macro can be edited so that instead of using fixed values, it can use variables for certain key values and/or numbers. The user can then change these variables as required. For an example consider the following simple macro example which selects some shells to orient and applies the translation 0.000 0.000 100.000.

```
MacroName("Orient")
MacroDescription("Test macro variables work in orient")
Window("Tools/Keywords").Button("Orient")
Window("Orient").Menu("ORIENT ITEMS").Select1("SHELL...")
Pause("Select shells to translate")
In Window("Orient")
   .Textbox("Translation distance") = "0.000 0.000 100.000"
   .Button("Apply")
End In
Window("Orient").Window("CONFIRM ORIENT").Button("Accept")
```

We want to change the translation to use variables so the user can change the values. We need to do 2 things.

1. Define the variables in the macro
2. Change the numbers in the translate to be variables.

To define the variables in the macro we have to add `MacroVariable()` commands at the top of the file. each command defines one variable. Variable names must only contain letters, numbers and the underscore character. They cannot contain spaces. To refer to a variable we use the variable name preceeded by a dollar. e.g. for variable X_TRANS we use $X_TRANS. Alternatively the syntax ${X_TRANS} can be used (i.e. dollar followed by variable name in curly brackets). Each MacroVariable() command defines a name for the variable (e.g. "X_TRANS") a description (e.g. "X translation distance") and the default value for the variable (e.g. "0.0").

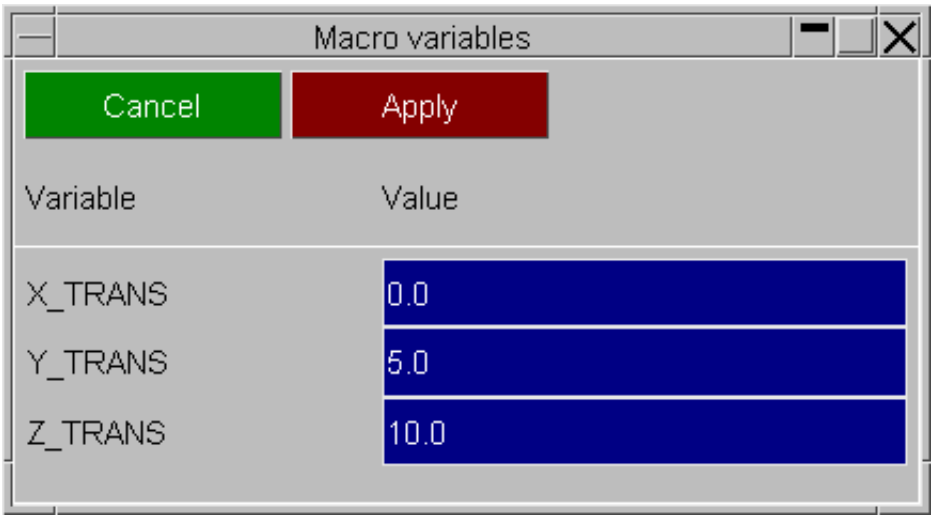
Adding these lines to the macro gives (added/changed lines shown in bold):

```
MacroName("Orient")
MacroDescription("Test macro variables work in orient")
MacroVariable("X_TRANS", "X translation distance", "0.0")
MacroVariable("Y_TRANS", "Y translation distance", "5.0")
MacroVariable("Z_TRANS", "Z translation distance", "10.0")
Window("Tools/Keywords").Button("Orient")
Window("Orient").Menu("ORIENT ITEMS").Select1("SHELL...")
Pause("Select shells to translate")
```

```
In Window("Orient")
   .Textbox("Translation distance") = "$X_TRANS  ${Y_TRANS}  $Z_TRANS"
   .Button("Apply")
End In
Window("Orient").Window("CONFIRM ORIENT").Button("Accept")
```

Using variables interactively

When you play a macro PRIMER scans the top of the macro to see if there are any `MacroVariable()` commands. If any are found then PRIMER shows a window with all of the variables. This allows you to change the variable values. Hovering over a variable name shows the description for each variable as hover text.



When the correct variable values are chosen the macro can be run by pressing **Apply**.

Using variables with CSV files

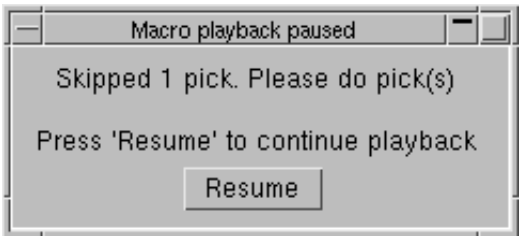
Instead of having to type the values for all variables interactively the values can be read from a CSV file. Give the name of the CSV file in the **Variables** textbox. The file should contain one variable and its value per line. Lines beginning with $ are treated as comments. e.g. the variables from the above example would look like.

```
$ Example macro variables CSV file.
X_TRANS,0.0
Y_TRANS,5.0
Z_TRANS,10.0
```

Playback options

There are some options that alter the way that macros are played back

Replay pick/drag commands: If a macro file contains any pick or drag commands then by default when the macro is played back the picks or drags will be played back exactly as they were recorded (i.e. the same position of the pick/drag on the screen will be replayed). If the **Replay pick/drag commands** option is selected this is what will happen.
If the option is unselected then the pick/drag command will be skipped and the macro playback will pause to allow you to replace the pick with whatever you want. A window will be mapped on the screen.



Once you have replaced the pick(s) or drag(s) then press **Resume** and the macro playback will restart.

Replay view for pick/drag

Whenever a pick or drag command is recorded PRIMER saves the current view in the graphics window to the macro with a `ViewMatrix` command. If this option is selected then on playback the view will be restored before picking. If it is not selected then the command will be skipped and the view will not be updated.

Ignore numbers in window titles

Ignore any numbers in window titles. See section on making macros work with different models for more details.

UTF-8 encoded macro file

Indicates that this macro contains Unicode text for Pause or MacroVariable commands and is UTF-8 encoded. If the macro contains a `MacroUTF8Encoded()` command this option will automatically be selected.

Pause between commands

Sometimes it is useful to have a pause between commands when playing the macro (e.g. if you are debugging a macro). This textbox allows you to give a pause (in milliseconds) between commands on playback.

Unicode

The text shown for `Pause()` commands and the descriptions in `MacroVariable()` commands can contain unicode text (e.g. Japanese or Chinese Kanji). If you want to use unicode then the macro file **MUST** be UTF-8 encoded and the **UTF-8 encoded macro file** playback option must be selected. If the `MacroUTF8Encoded()` command is added to the top of the macro then the option will automatically be selected.

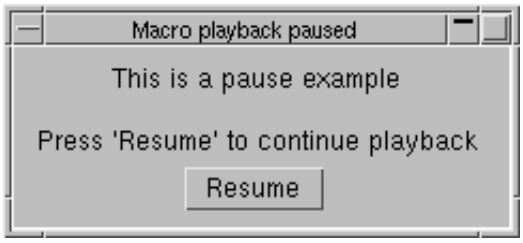
To show the unicode text the appropriate font must be used. This can be set using the preferences `primer*cjk_unix_font` and `primer*cjk_windows_font`.

Temporarily suspending macro files

It may be useful to pause playback of a macro so that the user can do certain operations. This can be done by manually adding a `Pause()` command to the macro. e.g. adding the command

```
Pause("This is a pause example")
```

will map the window shown on the right and then pause playback of the macro. The user can then do whatever operations are necessary and then press **Resume** to continue playback of the macro



This method could also be used to replace a sequence of picks in a macro. e.g. if a macro was recorded to create a contact and some parts were picked for the slave side of the contact there would be commands like

```
In GraphicsWindow("GRAPHICS 1")
   .ViewMatrix(1, 0, 0, 0, 1, 0, 0, 0, 1, 50, 72.5, 11.5, 34.8075, 316.838)
   .Pick1(1709, 1905)
   .ViewMatrix(1, 0, 0, 0, 1, 0, 0, 0, 1, 50, 72.5, 11.5, 34.8075, 316.838)
   .Pick1(2016, 1888)
   .ViewMatrix(1, 0, 0, 0, 1, 0, 0, 0, 1, 50, 72.5, 11.5, 34.8075, 316.838)
   .Pick1(2024, 1461)
   .ViewMatrix(1, 0, 0, 0, 1, 0, 0, 0, 1, 50, 72.5, 11.5, 34.8075, 316.838)
   .Pick1(1700, 1440)
```

```
End In
```

recorded in the macro. These could all be deleted and replaced with a command

```
Pause("Select parts for slave side of contact")
```

which would prompt the user to pick the appropriate parts.

Making macros work with different models

When PRIMER writes a button press to a macro it uses the title of the window you clicked in and the text on the button to create a human readable command in the macro. e.g. in the following macro:

```
Window("Keywords").Button("PART")
Window("Part").Menu("SELECT PART").Select1("M1/P152     (MC-A-ARM-BUSH1-L)")
In Window("MODIFY PART M1/P152")
     .Textbox("SECID") = "10"
     .Textbox("MID") = "10"
End In
```

the user has:

1. Pressed button **`PART`** in the **`Keywords`** window *[which maps the part modify panel]*
2. In the menu **`SELECT PART`** in the **`Part`** window selected the entry **`M1/P152 (MC-A-ARM-BUSH1-L)`** *[which modifies part 152]*
3. In window **`MODIFY PART M1/P152`** changes the **`SECID`** and the **`MID`** values to 10.

This is fine if the macro is going to be replayed on exactly the same model but it will fail if the macro is played back on a similar (but not identical) model if the title of part 152 is different (or if the ID of the part with title "MC-A-ARM-BUSH1-L" is changed. This is because in step 2 the text on the menu button that is recorded includes the part ID and the part name.

When PRIMER replays a macro and it finds a window or menu title or a menu entry it first tries to find a unique match for the text in the macro command. If it finds one then that window is used. If that fails then it looks for a window or menu title or a menu entry which contains the text somewhere in the title. You can use this fact to make macros more portable across models.
For example if the above macro is always going to be used to edit part 152 you can modify the second line to

```
Window("Part").Menu("SELECT PART").Select1("M1/P152")
```

and the macro would still work if the title of the part is different. However you must be careful when doing this. If part 152 exists in the model then the above will work. If part 152 does not exist but part 152000 does exist then this would still match and so may edit the wrong part by mistake.

Alternatively, if the part number may change but the part title will always be the same you could make the macro more portable by modifying lines 2 and 3 to:

```
Window("Part").Menu("SELECT PART").Select1("MC-A-ARM-BUSH1-L")
In Window("MODIFY PART")
```

This will work as long as the part name is unique (and is the first match).

Occasionally it may be useful to play a macro in a completely different window to the one it was recorded in. For example you could record a macro which sets the necessary defaults for a contact that you modify. In this case if you were modifying contact 1000 in model 1 when you were recording the macro the macro could look like:

```
In Window("MODIFY CONTACT M1/CONT1000")
      .Textbox("sfs") = "0.1"
      .Textbox("sst") = "0.5"
      lots more commands
 End In
```

Now you may want to set the defaults for contact 1001. You could not use the above macro to modify contact 1001 as the title for this would be `"MODIFY CONTACT M1/CONT1000"`. If you ignored any numbers in the title then you could use the macro as the title for both windows would reduce to `"MODIFY CONTACT M/CONT"`. The **Ignore numbers in window titles** option does this.

Grouping commands in the same window together

To try to make macros easier to read and to make hand editing them easier PRIMER will group commands that are in the same window together using `In` and `End In` commands. e.g. the following macro:

```
Window("MODIFY PART M1/P152").Textbox("SECID") = "10"
Window("MODIFY PART M1/P152").Textbox("MID") = "10"
Window("MODIFY PART M1/P152").Textbox("EOSID") = "10"
Window("MODIFY PART M1/P152").Textbox("HGID") = "10"
```

is identical to

```
In Window("MODIFY PART M1/P152")
    .Textbox("SECID") = "10"
    .Textbox("MID") = "10"
    .Textbox("EOSID") = "10"
    .Textbox("HGID") = "10"
End In
```

but the second form is more concise and easier to read and edit (e.g. if you wanted to change the ID of the part being modified you would only have to do it on one line).

6.27.3 List of macro commands

Below is a full list of all of the commands available in macros. Comments can be put anywhere in a macro file. Any line that begins with $ or # is treated as a comment and will be skipped.

Commands in windows

A window is identified in PRIMER with the commands. PRIMER will try to find a window exactly matching *name*. If that fails a partial match will be tried. See the section on making macros work with different models for more details.

Command	Description
`DialogWindow( "name" )`	A window that is shown asking the user to confirm an action or answer a question.
`GraphicsWindow( "name" )`	The main graphics window
`Menu( "name" )`	An object menu
`PopupWindow<number>()`	A popup window. `<number>` identifies the popup. i.e. `PopupWindow1` is a normal popup. `PopupWindow2` is a popup mapped from a popup etc.
`Window( "name" )`	A 'normal' window

The following commands are used in conjuction with a 'window' command. Some commands have various alternatives. Where this is possible the alternatives are separated by `|` and are in square brackets `[]`. e.g. the command

```
.[|Ctrl|Shift]Click( "name" )
```

can be

```
Click( "name" )
CtrlClick( "name" )
ShiftClick( "name" )
```

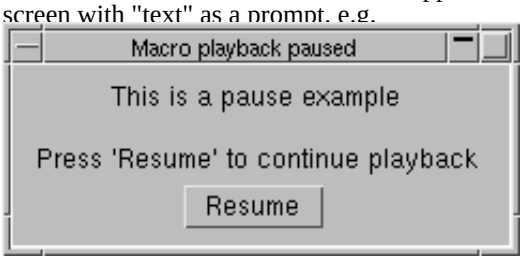
Command	Description
`.Actions()`	Equivalent to pressing the top left 'actions' button for a window
`.[Left|Right|Up|Down]Arrow()`	Equivalent to using arrow keys to scroll a list

`.Bitmap()`	Equivalent to pressing the 'SAVE->BITMAP' button in the top left actions popup for a window
`.[|Ctrl|Shift]Button( "name" )`	Press, Ctrl Press or Shift Press on button *name*.
`.[|Ctrl|Shift]Button( "name" ) = [on|off]`	Press, Ctrl Press or Shift Press on toggle button *name*. The state is set by using `= on` or `= off`.
`.[|Ctrl|Shift]Click( "name" )`	Click, Ctrl click or Shift click on item *name* in a table or tree.
`.Collapse( "name" )`	Collapse branch *name* in a tree.
`.Dismiss()`	Dismiss a window. Equivalent to pressing the top right button for a window
`.[|End|Start]Drag<number>( <x>, <y> )`	Drag at location *x*, *y* using mouse button *number*.
`.End()`	Equivalent to pressing End key to go to the bottom of a list
`.Expand( "name" )`	Expand branch *name* in a tree.
`.Feedback( "name" ) = "value"`	Perform feedback function on button *name* with *value* (e.g. when typing in parameter, show the list of matching parameters)
`.Help()`	Show help for a window. Equivalent to pressing the top right button for a window
`.Home()`	Equivalent to pressing Home key to go to the top of a list
`.Hover( "name" )`	Perform hover function on button *name* (e.g. when hovering over a button with a parameter, the parameter data is shown)
`.Lower()`	Lower a window in the stacking order.
`.Maximise()`	Maximise a window. Equivalent to pressing the top right button for a window
`.Minimise()`	Minimise a window. Equivalent to pressing the top right button for a window
`.Page[Down|Up]()`	Equivalent to pressing PageUp or PageDown to move up/down a page in a list
`.Pick<number>( <x>, <y> )`	Pick at location *x*, *y* using mouse button *number*.
`.Picking()`	Restart picking in a window. Equivalent to pressing the top left button for a window
`.Popup( "name" )`	Map popup window for button *name* (equivalent to right clicking on button)
`.Radio( "name" ) = "value"`	Set radio button *name* to *value*
`.Resize( <data> )`	Resize/move window
`.Restore()`	Restore a minimised window.
`.Select<number>( "name" )`	Select *name* from object menu at depth *number*. PRIMER will try to find a menu entry exactly matching *name*. If that fails a partial match will be tried. See the section on making macros work with different models for more details.

`.Slider( "name").Move[Up|Down|Right|Left]()`	Move slider *name* up, down, right or left.
`.Tab( "name" )`	Press tab *name* in a window.
`.Textbox( "name" ) = "value"`	Set textbox *name* to *value*
`.ViewMatrix( <view data> )`	Records the current view data to the command file so that when replaying the command file the view can be restored before doing a pick/drag

Other commands

Command	**Description**
`$ comment`	Any line beginning with `$` is a comment
`# comment`	Any line beginning with `#` is a comment
`Dialog( "command" )`	Iss *command* in the dialogue window
`Dynamic[Rotate|RotateZ|Scale|Translate]( <data> )`	Dynamic viewing command
`End In`	End a group of commands in the same window.
`FunctionKey( <key> )`	Equivalent of pressing function key
`In <window>`	Start a group of commands in the same window.
`Justify( "[Top|Bottom|Left|Right|Centre| TopRight|TopLeft|BottomRight|BottomLeft]" )`	By default `Pause` commands are shown at the top right of the PRIMER window. The `Justify` command changes the position that they are shown in. For example if you want them to be shown in the centre of the screen add the command `Justify("Centre")` to the macro **before** the `Pause` command.
`MacroDescription( "description" )`	Hover text that will be shown for the macro on the button in the macro panel. This must be in the first 10 lines of the macro.
`MacroName( "name" )`	Name that will be shown for the macro on the button in the macro panel. This must be in the first 10 lines of the macro.
`MacroUTF8Encoded()`	Indicates that the macro contains Unicode text for Pause or MacroVariable descriptions and is UTF-8 encoded. See the Unicode section for more details. This must be in the first 10 lines of the macro.
`MacroVariable( "name", "description", "value")`	Adds a variable definition to the macro. See the Variables section for more details. These must be defined near the top of the file before the variable(s) are used.

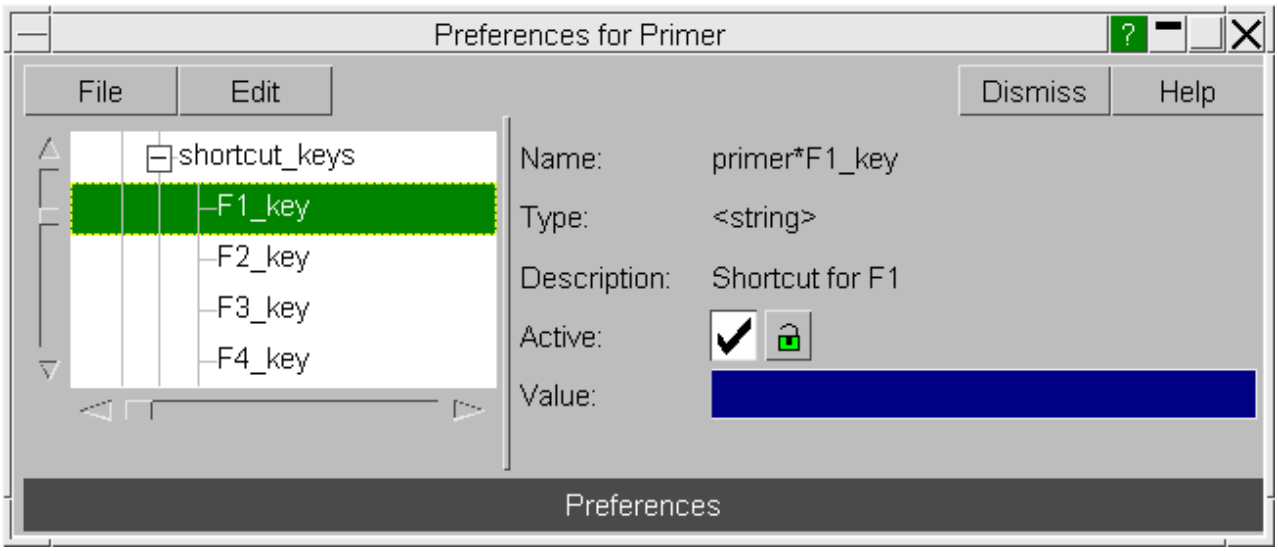
`Pause( "text" )`	Temporarily suspend command file playback to allow user interaction. A window is mapped on the screen with "text" as a prompt, e.g.  Press Resume to resume playback.
`Promptln( "message" )`	Write message to the dialogue box.
`Promptln( "message" )`	Write message to the dialogue box, adding a new line
`SelectFile( "filter" ) = "name"`	Select file *name* from the File selection window.
`ShortcutKey( <key> )`	Equivalent of doing shortcut *key*
`Version( "version", <build> )`	Records the version and build of PRIMER into the macro. this may be used in the future to help playback of macros recorded in earlier versions of PRIMER.
`WindowSize( <x>, <y> )`	Records the size of the graphics window to the macro so that when replaying the window can be resized to the correct proportions (if different) so that picks and/or drags work correctly.

6.27.4 Assigning macros to shortcut keys

Macros can be assigned to shortcut keys to make them quick and easy to run. To do this use the **Assign Macros to shortcut key** button (or press the shortcut key '**?**')

Using the above button will only assign the macro to the shortcut key for this session of PRIMER. If you want to always assign the macro to a shortcut key use the **`primer*F1_key`** preferences etc in the **`shortcut_keys`** branch for PRIMER in the oa_pref preference file. Note that the preference can be used to run either a shortcut, a script or a macro. For PRIMER to know that the preference should run a macro, the macro **MUST** have the extension **prm**.

The image below shows the preferences editor with the **`primer*F1_key`** open.



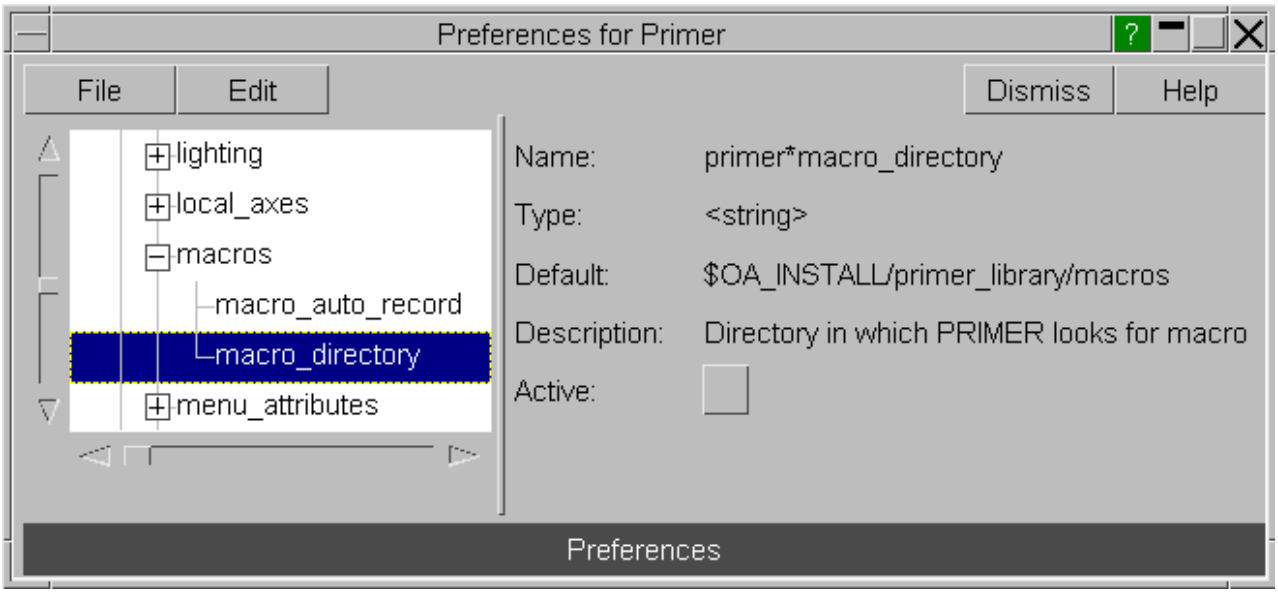
6.27.5 Assigning macros to buttons

When PRIMER starts it automatically looks for macros in the directories:

- $OA_ADMIN/primer_library/macros (if $OA_ADMIN is defined)
- $OA_INSTALL/primer_library/macros
- $OA_HOME/primer_library/macros

Each macro that is found is assigned to a button in the macros panel. The text that is shown on the button is read from the `MacroName()` command at the top of the macro. This is automatically added by Primer when you record a macro if you enter some text in the **Name** textbox. Additionally hover text for the button is read from the `MacroDescription()` command at the top of the macro. This is automatically added by Primer when you record a macro if you enter some text in the **Description** textbox.

The directory that PRIMER looks in for macro files can be changed in the oa_pref files in $OA_ADMIN, $OA_INSTALL and $OA_HOME by using the `macro_directory` preference.



For example if you change the **`macro_directory`** preference in the oa_pref file in the $OA_INSTALL directory to /test/primer_macros then PRIMER will look for macro files in the directories:

- $OA_ADMIN/primer_library/macros (if $OA_ADMIN is defined)
- /test/primer_macros
- $OA_HOME/primer_library/macros

6.27.6 Limitations

The biggest limitation with macros is that windows are identified by their title. This is fine if the titles are unique but causes problems if there are 2 windows with the same title. e.g. If you record a macro that modifies 2 different parts the windows will have different titles and the macro will be replay correctly. However, if you create 2 parts at the same time then both windows will have the same title and so on playback PRIMER will not be able to identify which of the 2 parts the button presses etc should be replayed in and the playback will fail.

In reality this is not a big limitation as the above situation is rare. You just need to be careful to only have one creation window for a particular type open at one time.

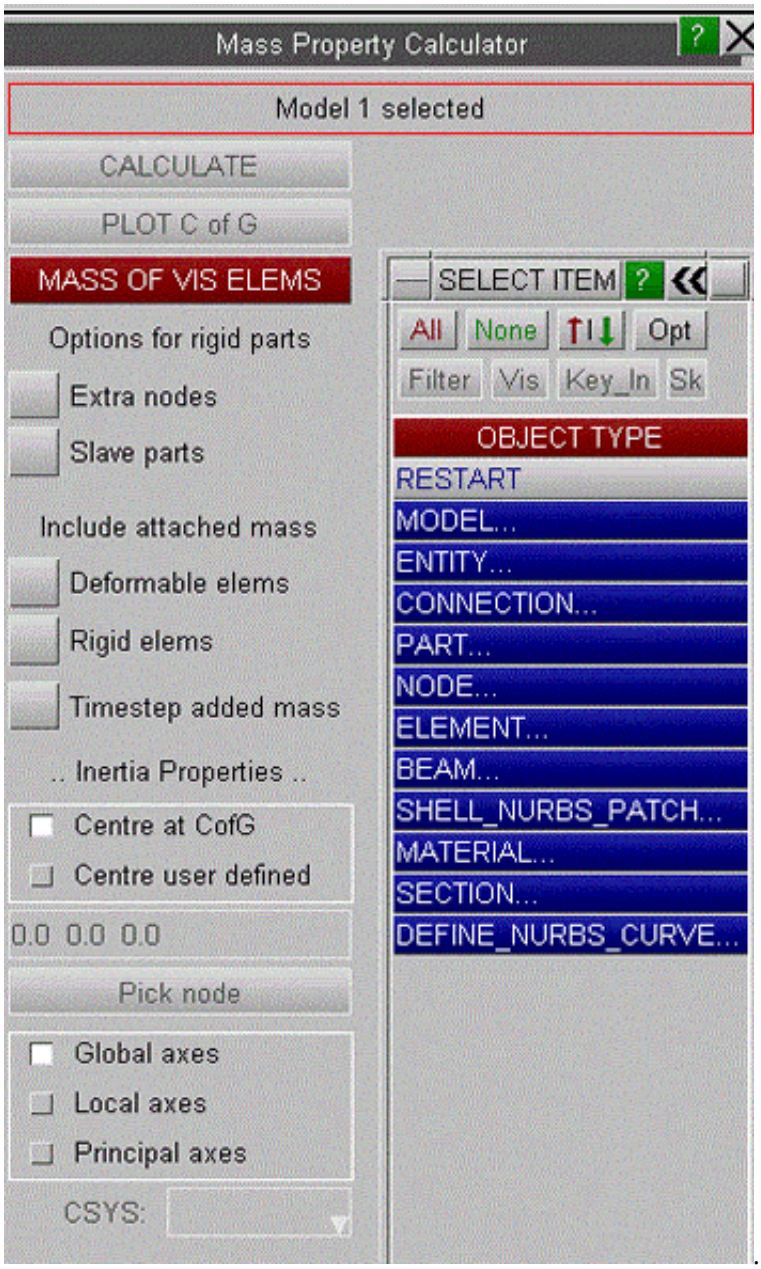
Dragging items in the part tree is not supported by macros. As a workaround you should right click on the selected parts and choose 'Cut' and then right click on the destination and choose 'Paste'. These actions will be recorded correctly in the macro.

6.28 MASS PROPERTY CALCULATOR

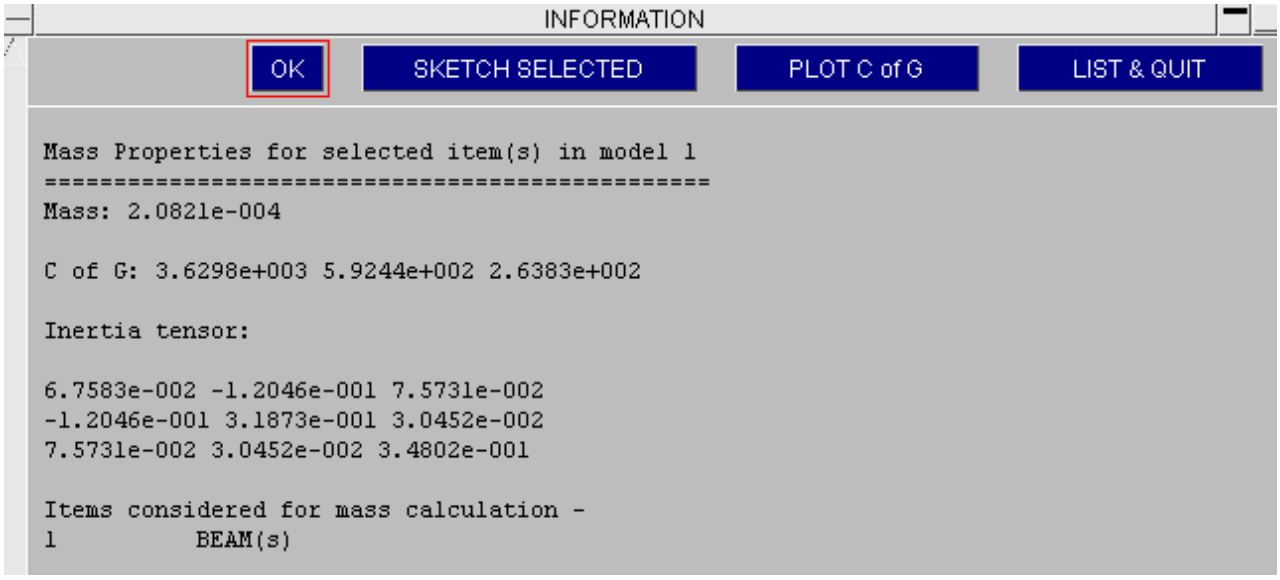
The **Mass Prop** tool allows you to calculate the mass properties of a selection of items from the model. It is accessed from the main Tools menu

6.28.1 Selecting an item

Select the chosen item from the object menu and press **CALCULATE.** The mass properties will be reported in an information panel



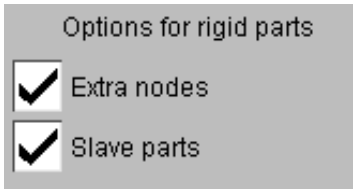
6.28.2 Calculating properties



In most cases, Primer will devolve the selection to the element level and sum the nodal masses derived from these elements as appropriate.

For a rigid element this will include the mass share of any deformable element directly attached.

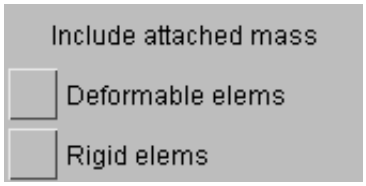
If a rigid part is selected, by default the mass of *CONSTRAINED_EXTRA_NODES and elements of any parts slaved to this one by *CONSTRAINED_RIGID_BODIES will be not be included in the calculation. These options can be switched ON.



For deformable elements mass at nodes attached to rigid parts/nrbs will not be subtracted. Consequently for a part you may get a slightly higher mass from this function than the part table gives.

If you make a selection which does not devolve to elements (such as a constrained joint) Primer will sum the masses of the nodes involved (devolved from elements that may not be selected) and report that.

Options may be set to include lumped mass attached to the nodes of selected elements. These are off by default.

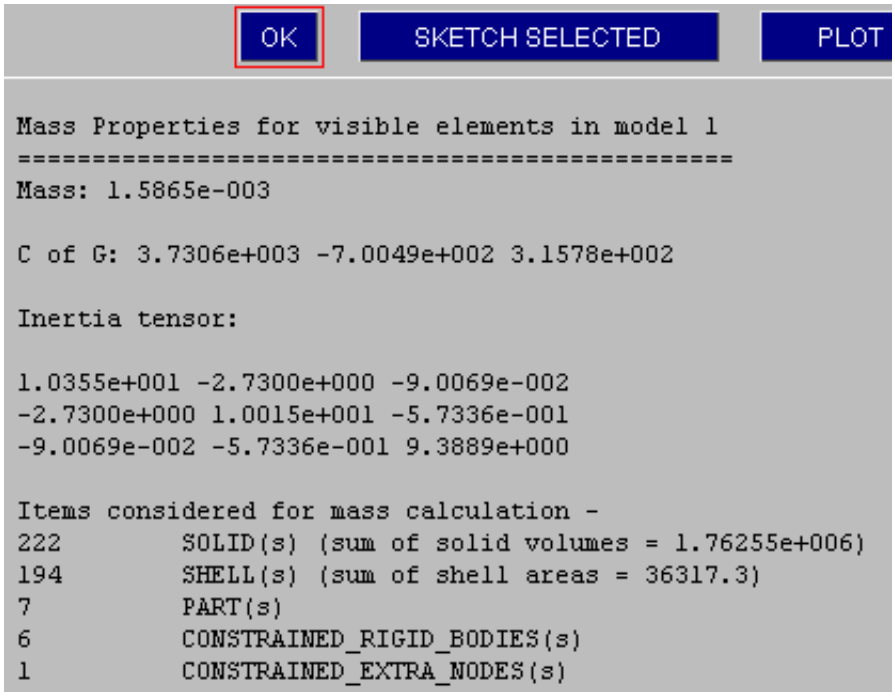


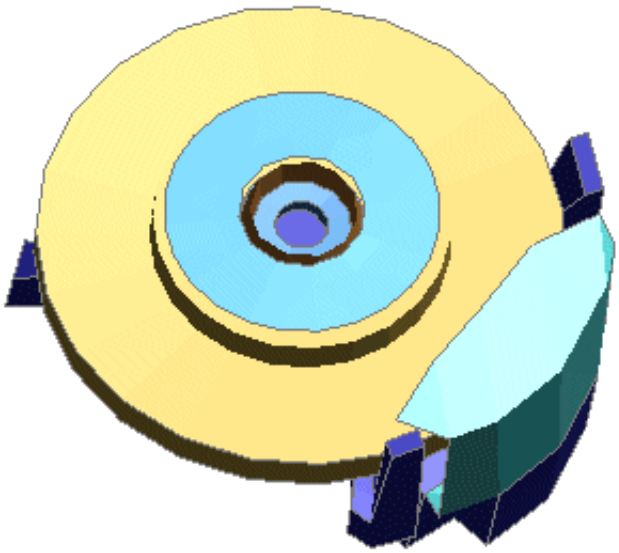
For a deformable part the value of lumped mass is shared equally amongst all the parts that attach to the node, so the calculation will only include that share that applies to selected elements.

Timestep Added Mass will include timestep mass scaled mass for deformable elements

6.28.3 Mass of what is visible

MASS OF VIS ELEMS is useful to determine mass properties for a set of displayed components when blanking has been applied.



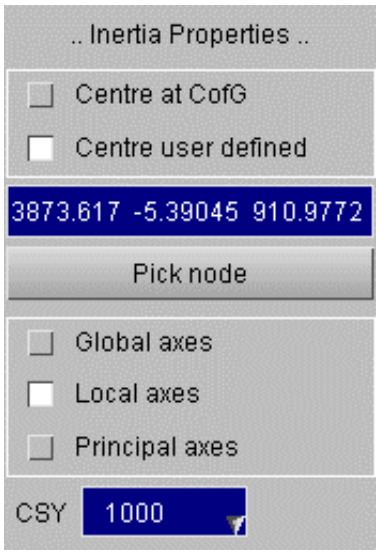


6.28.4 Inertia in local system

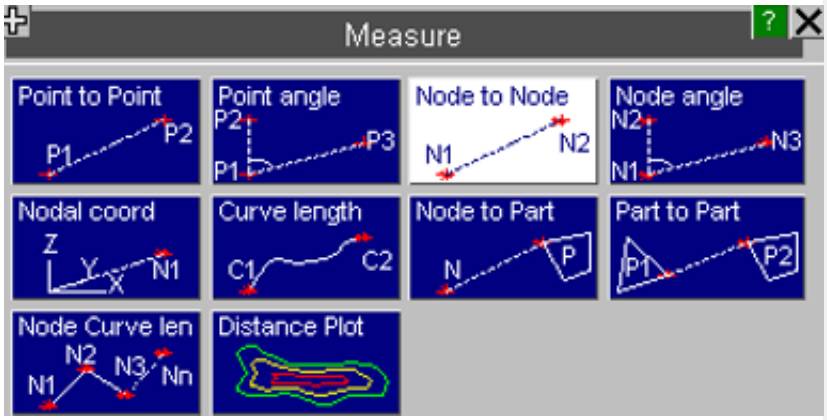
By default the inertia tensor is reported in the global axis system and about the CofG of the selected mass.

The tensor may also be expressed about the principle axes and the CofG by taking the ’Principal axes’ option.

In the most general case it may be expressed about an arbitrary centre and in a local axis system defined by the user by selecting ’Centre user defined’ and ’Local axes’.


.. Inertia Properties ..
Centre at CofG
Centre user defined
3873.617 -5.39045 910.9772
Pick node
Global axes
Local axes
Principal axes
CSY
1000

6.29 MEASURE Measuring the distance and angles between nodes and points on the screen.



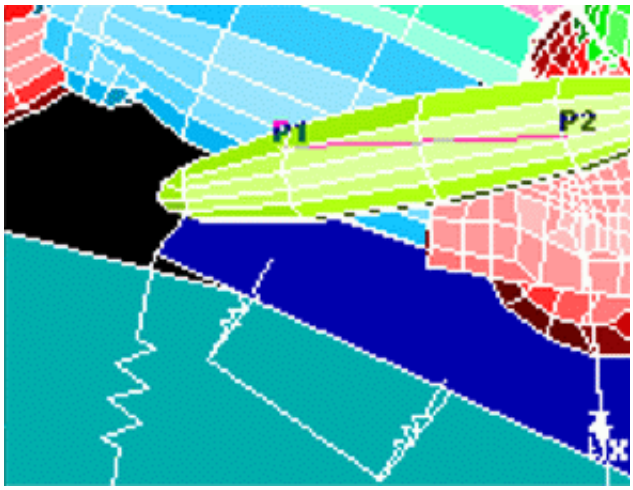
The **MEASURE** command is invoked from the **Tools** panel at the top of the screen or from the shortcut key M. There are five options, 3 using nodes and 2 using screen points, which measure:

Point to Point	The (x,y) distance between two screen points P1 and P2.
Point angle	The angle between vectors P1P2 and P1P3
Node to Node	The (x,y,z) distance between N1 and N2, where N1 and N2 are either nodes or geometry points.
Node angle	The angle between vectors N1N2 and N1N3, where N1, N2, N3 are either nodes or geometry points.
Nodal coord	The coordinate of node/geometry point N1.
Curve length	The length of the selected geometry curve.
Node to Part	The (x,y,z) distance between N and P, where N is either node or geometry point and P a part.
Part to Part	The (x,y,z) distance between P1 and P2, where P1 and P2 are both parts.
Node Curve length	Cumulative length for node selections
Distance Plot	The distance between two groups of parts

Screen "points" are simply transient 2d locations on the display picked with the cursor. They do not have any structural significance and are not part of any model. Distances and angles computed from them are in the 2d screen (x,y) space system.
In this example the user has selected two points, labelled `P1` and `P2` on the screen, and the 2D projected vector between them is to be computed.

The reported distance, and its orientation with respect to the model, will be a function of the current transformation matrix.

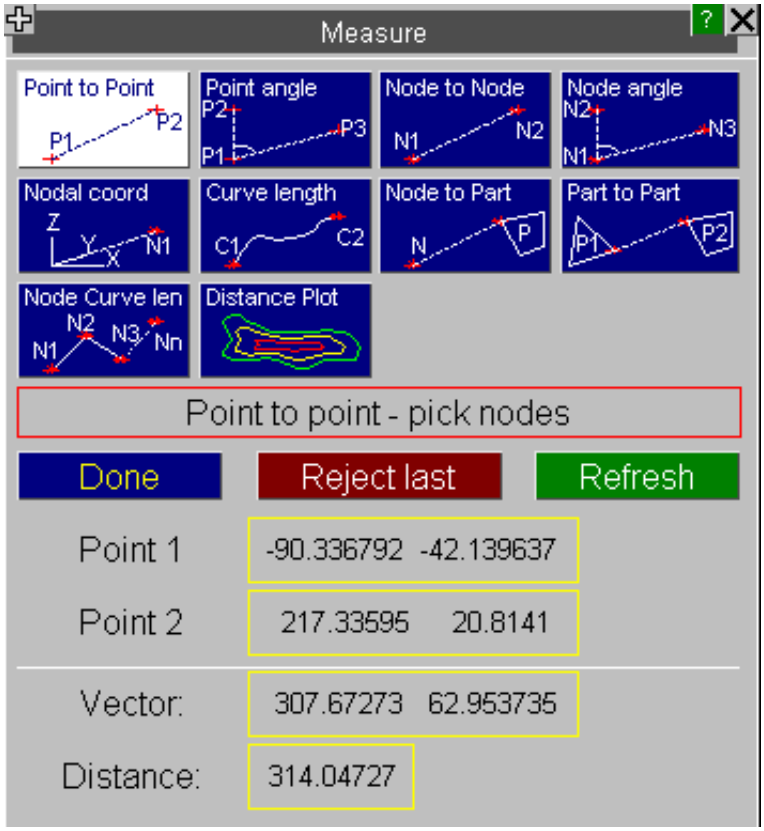
Therefore point-based measurement should be used when projected distances are required. For true 3D model space measurement it is better to use nodes.



6.29.1 Point to Point

Measures the projected 2D distance between two screen points.

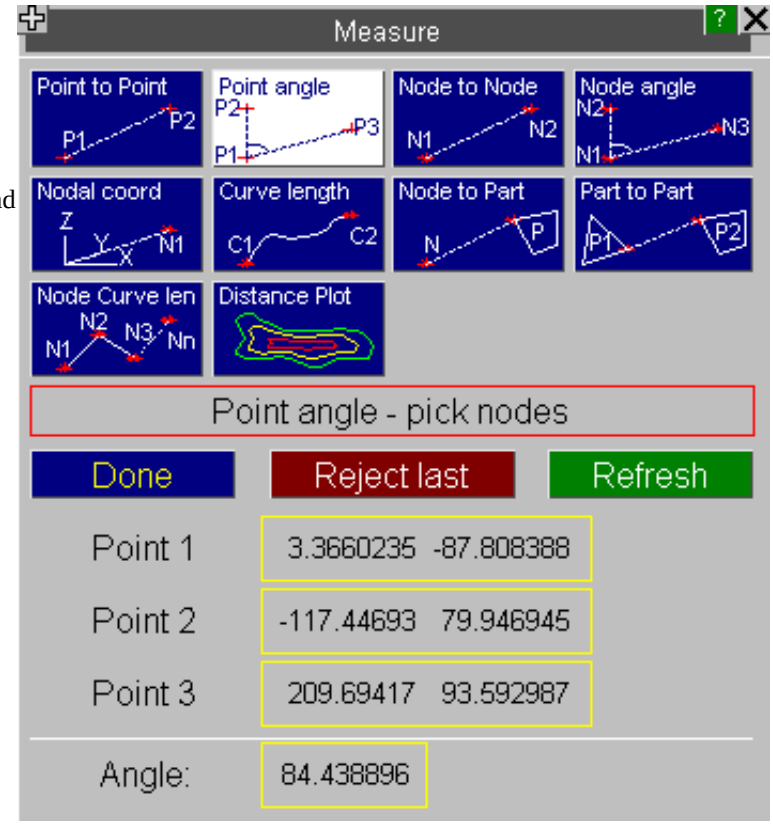
Select two points with the mouse, and the vector and (x,y) components of the distance between them is reported in screen space units.



6.29.2 Point angle

Point angle measurement computes the angle between the vectors P1P2 and P1P3.

The angle is computed in the 2D screen plane and reported in degrees.



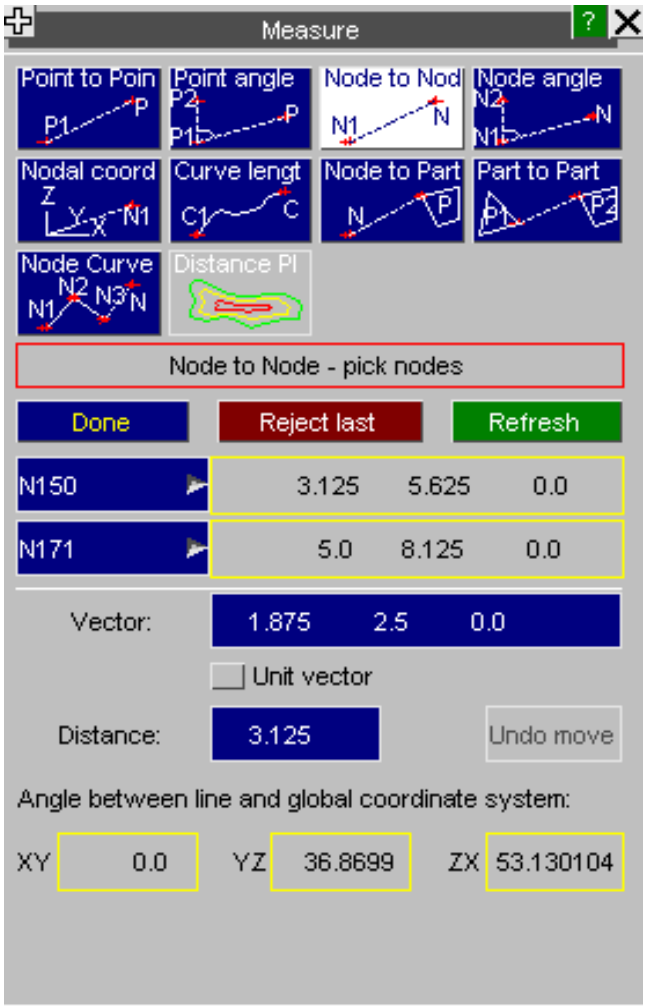
6.29.3 Node to Node

Node to node measurement computes the vector between nodes/geometry points N1 and N2, and reports it as model space (x,y,z) and magnitude components.

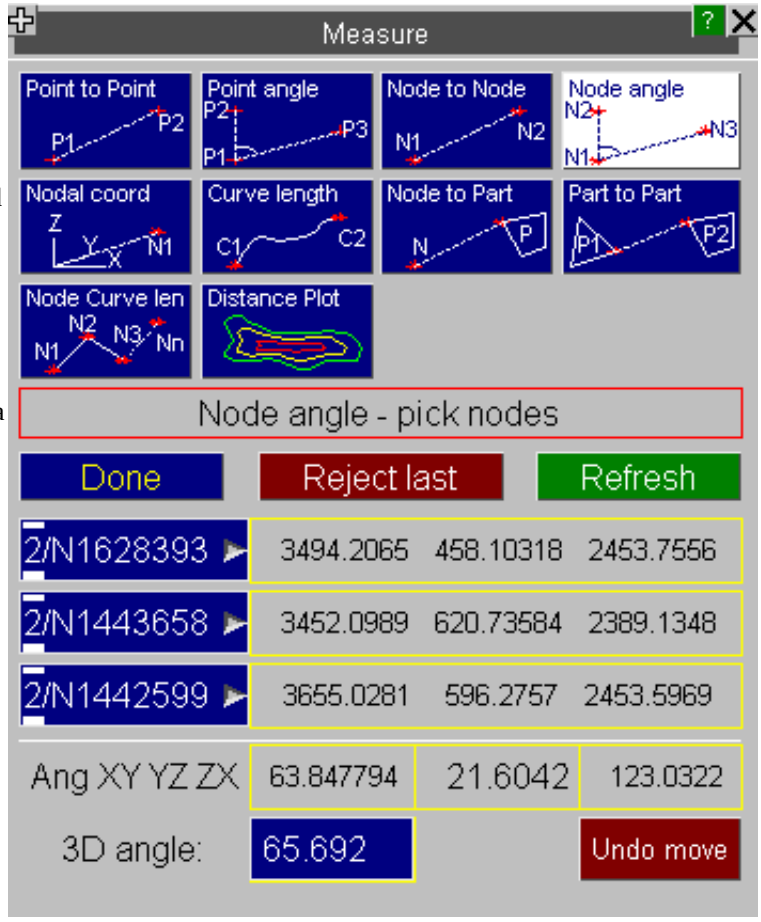
Nodes/geometry points may either be screen-picked, or have their label typed in, or use the standard popup options. As in this example nodes need not be in the same model.

Nodal distance is editable and if edited will result into a movement of the second node picked for measure (N2). The node will be moved along the line formed by N1N2 to reach the distance typed in.

In the same way, nodal vector is also editable. In this case, the user has the option of setting the unit vector option ON or OFF. If the unit vector option is ON, then the second node (N2) will be moved such that the vector N1-N2 will lie along the input unit vector keeping the original distance constant. If this option is OFF, then N2 will be set based on the direction and magnitude of the given vector.



6.29.4 Node angle



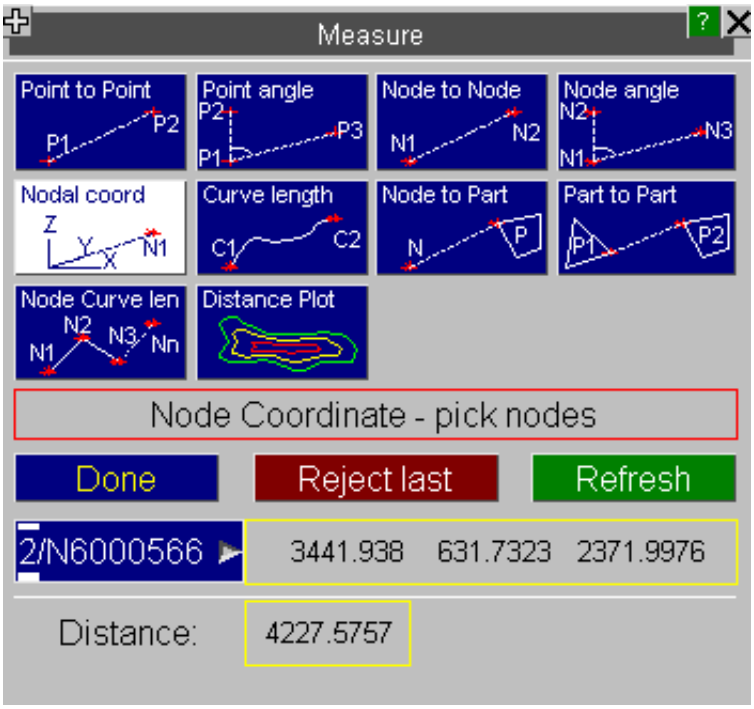
Node angle measure computes the angle between vectors N1N2 and N1N3.

These are reported in model space (x,y), (y,z) and (z,x) planes as well as in 3D (x,y,z) space. Units are degrees.

Nodes/geometry points are screen-picked or otherwise selected as in 6.5.3 above, and may be in different models.

3D angle is editable and if edited will result into a movement of the third node picked for measure (N3). N3 will move to reach the 3D angle typed in, staying in the same N1N2N3 plan, distance N2N3 will stay the same.

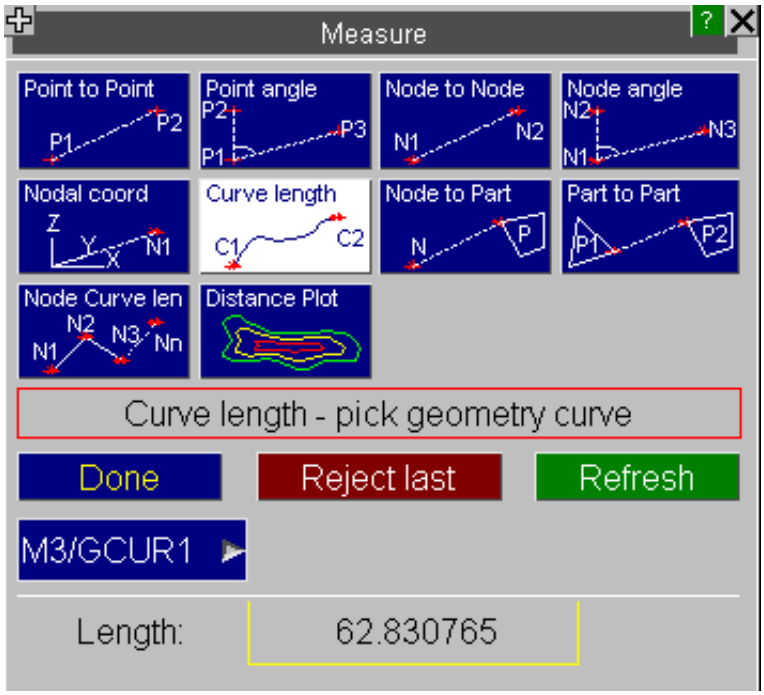
6.29.5 Nodal coord



Nodal coordinate gives the coordinates of the selected node/geometry point in model space units, and also its distance from the origin.

6.29.6 Curve length

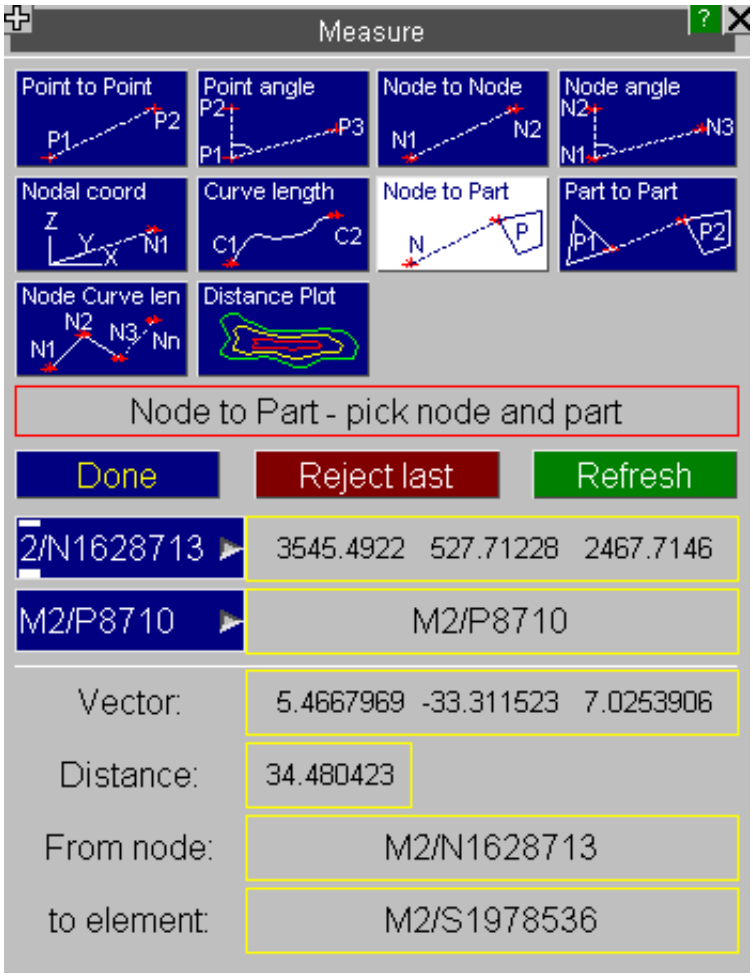
Curve length gives the length of the selected geometry curve in model space units.



6.29.7 Node to Part

Node to part measurement computes the vector between nodes/geometry points N and part P, and reports it as model space (x,y,z) and magnitude components.

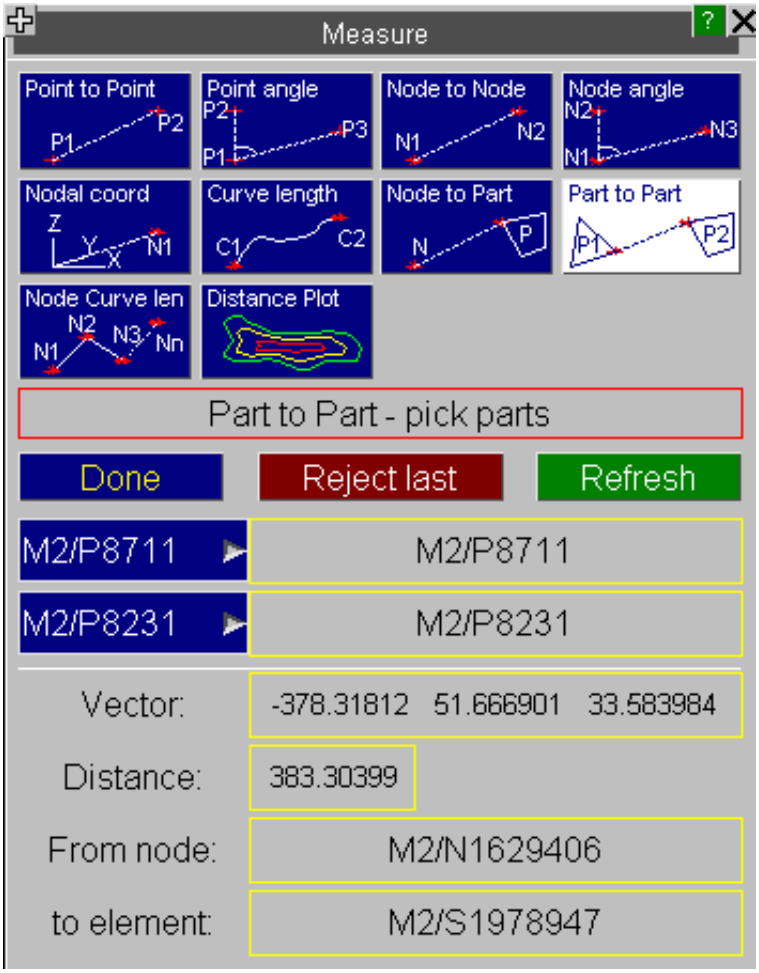
Nodes/parts may either be screen-picked, or have their label typed in, or use the standard popup options. As in this example nodes and parts need not be in the same model.



6.29.8 Part to Part

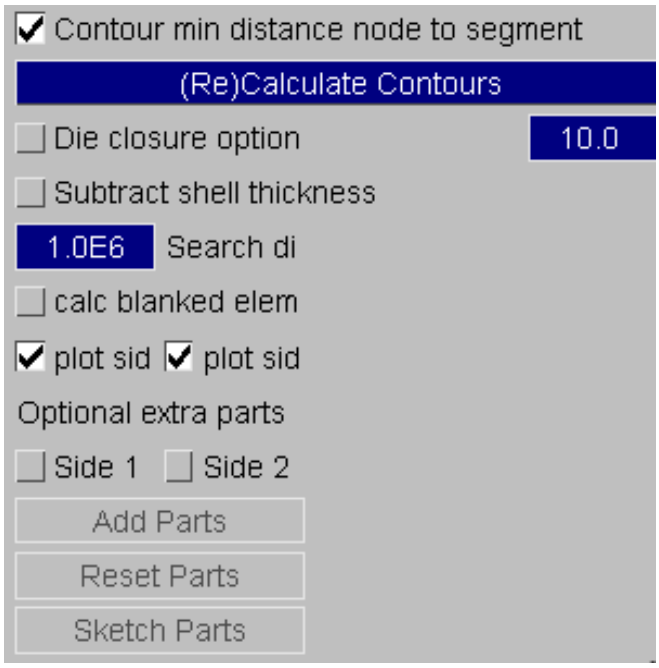
Part to part measurement computes the vector between parts P1 and P2, and reports it as model space (x,y,z) and magnitude components.

Parts may either be screen-picked, or have their label typed in, or use the standard popup options. Parts need not be in the same model.



6.29.9 Contour Part(s) to Part(s)

Select 2 parts to start the normal measure. You will be offered the option to contour min distance node to segment.

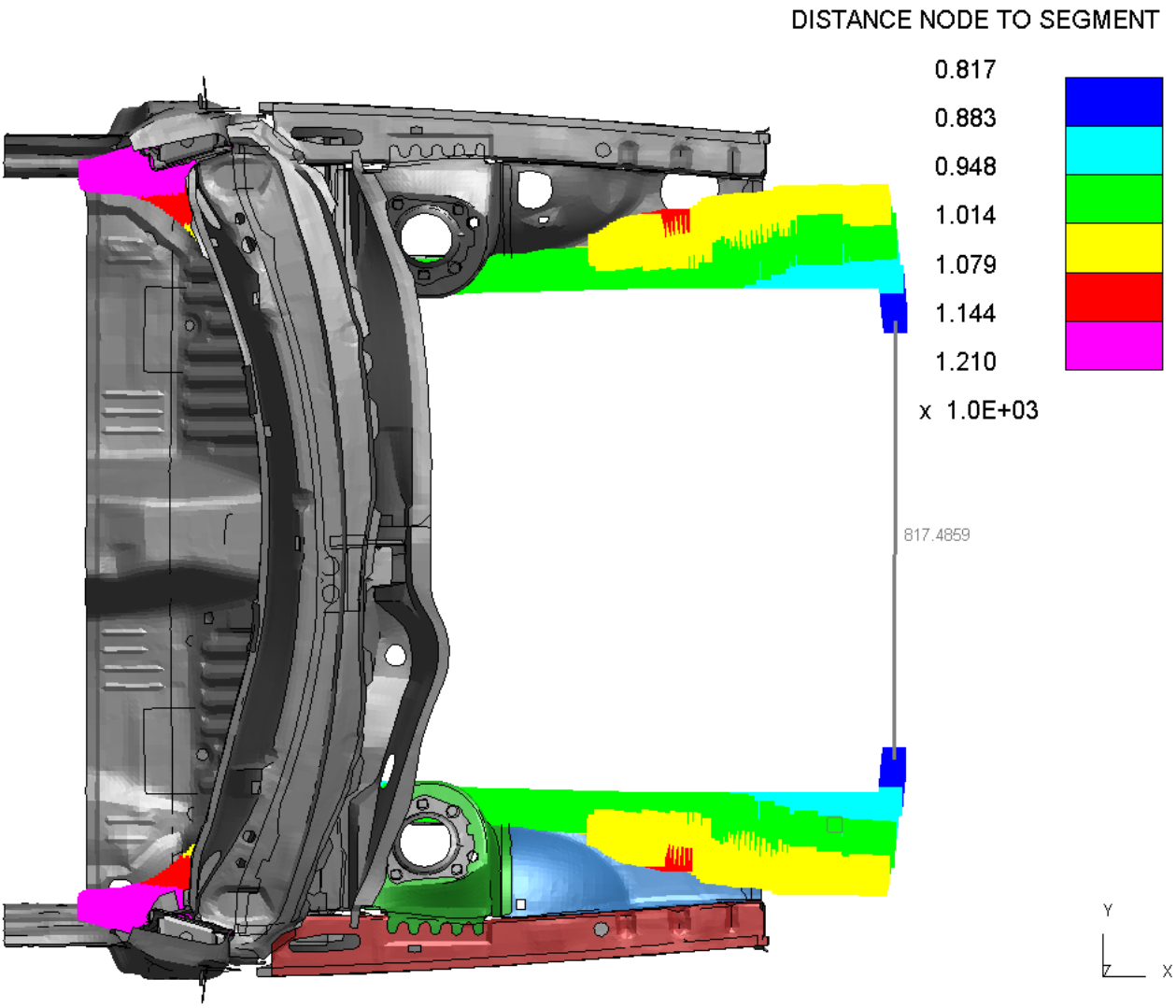


Selecting this will give contoured measure across the parts. The closest distance reported by standard measure will be the lowest contour.

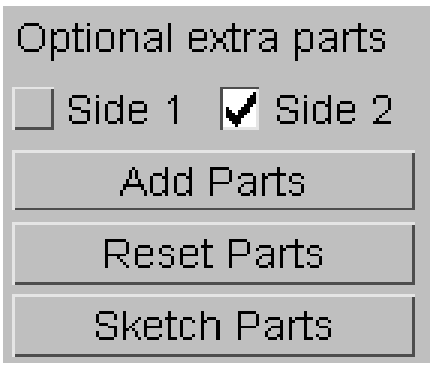
Additional options are

node normal distance - in this case we only plot values where the node to segment vectors coincides with the node local normal to the required tolerance (default 10 deg)

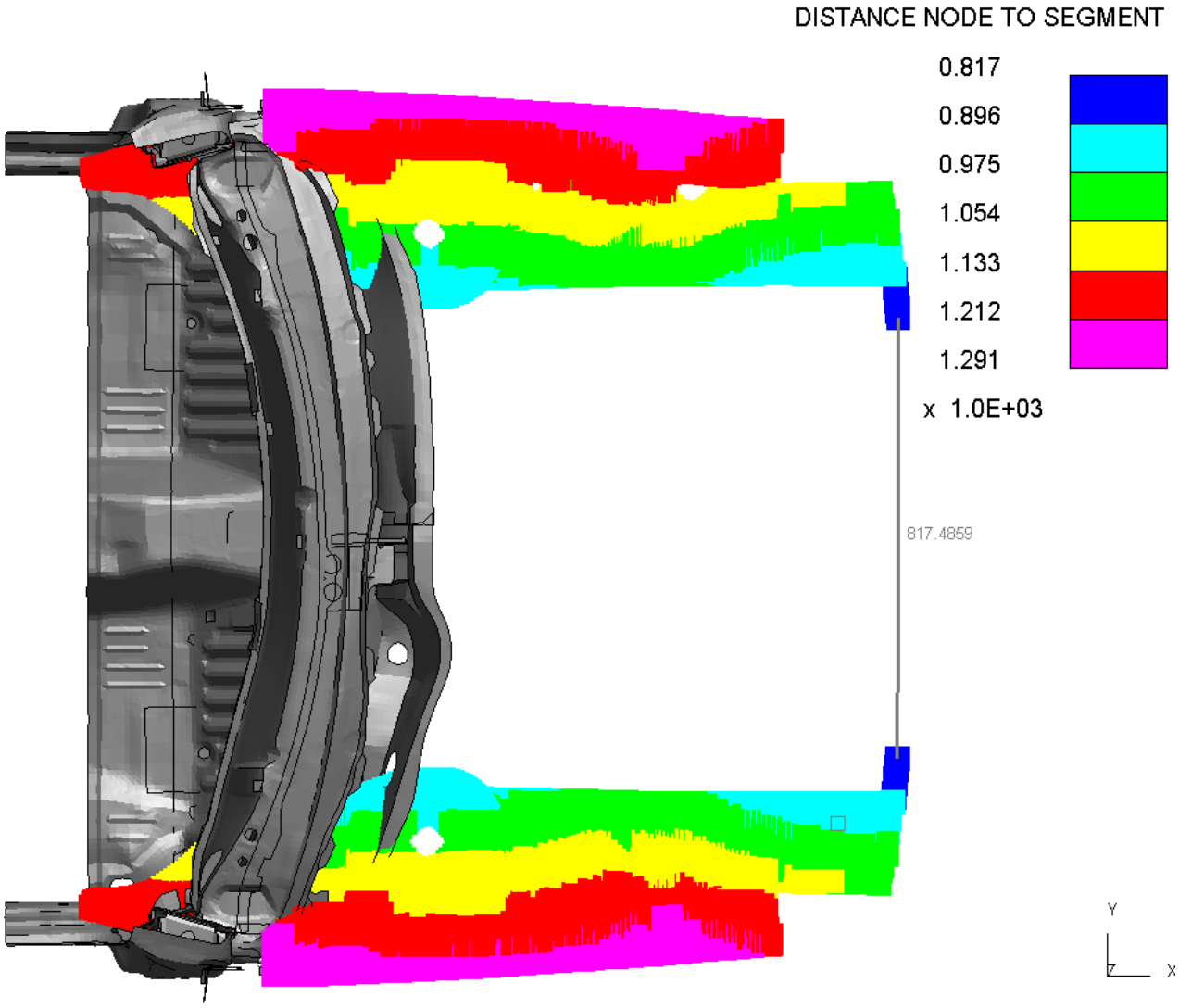
subtract shell thickness - the shell thickness at each end is subtracted from the measure



Additional parts can be added to each side

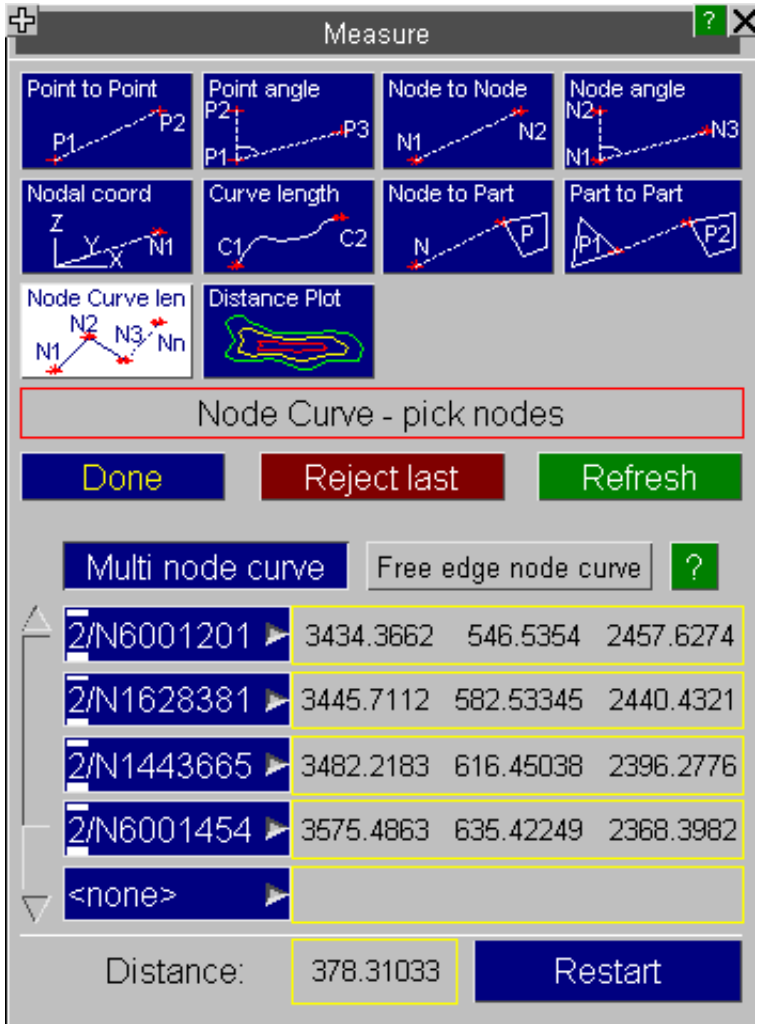


the contour plot will automatically update to include them. If either of the two main parts are changed contour mode will be de-activated.


DISTANCE NODE TO SEGMENT
0.817
0.896
0.975
1.054
1.133
1.212
1.291
x 1.0E+03
817.4859
Y
X

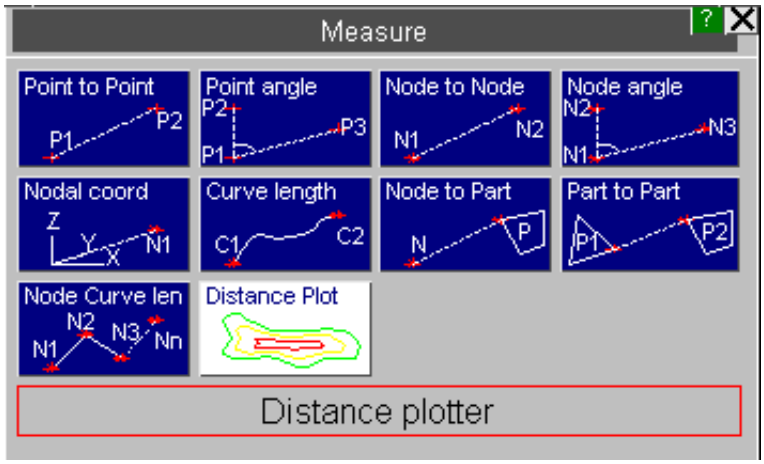
6.29.10 Node Curve Length

Node curve length calculates cumulative length for a number of node selections.

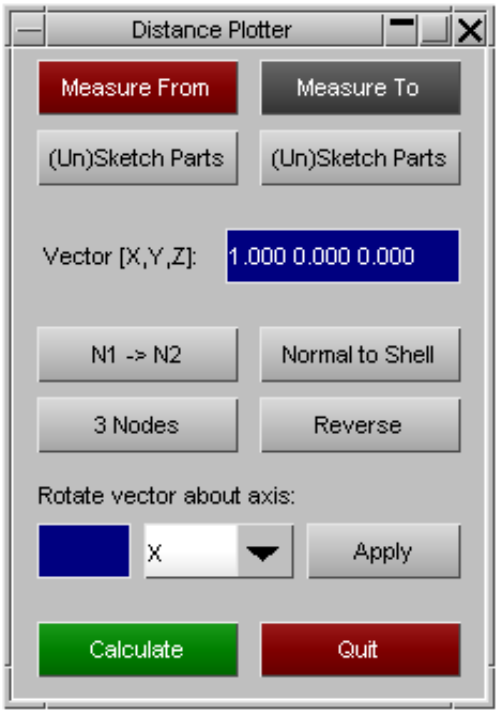


6.29.11 Distance Plot

Distance plot generates a contour plot displaying the distance between two groups of parts.

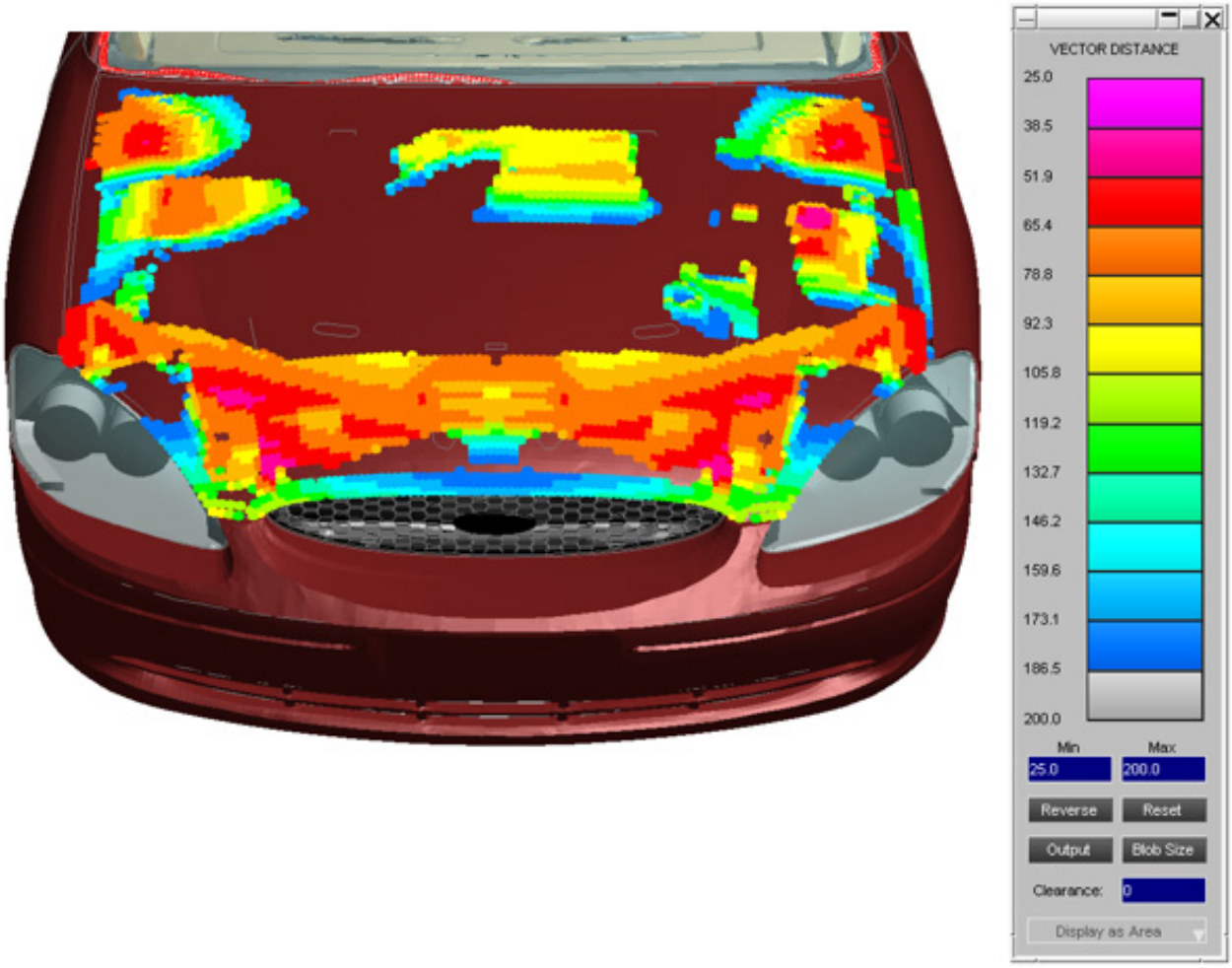


Distance plot generates a contour plot displaying the distance between two groups of parts. The user must select parts to measure from and (optionally) parts to measure to. Not selecting parts to measure to results in the entire model being considered, which may be slow to compute.
After selecting the parts for consideration, the user must define the measurement vector. There are several ways to define the vector: node to node, shell normal and three nodes. It is also possible to flip the vector direction and rotate it about the global axes.



The distance calculated will be from the nodes of the ‘from’ parts in the direction of the vector to the elements/segments of the ‘to’ parts. The measurement is one way, in the positive vector direction.

Once calculated the plot will be displayed as coloured ‘blobs’ at each of the ‘from’ nodes.



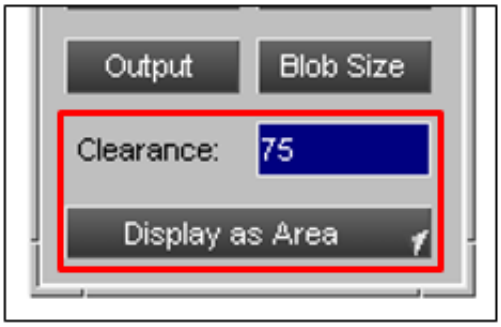
Contour Manipulation

The contour plot popup is interactive:

- Clicking the colour bands will toggle the visibility of that range.
- The max and min values can be changed using the text boxes.
- The colours can be reversed and the max/min reset.
- The plot can be output to a csv or a d3plot external data (blob) file.
- The size of the coloured ‘blobs’ can be changed.

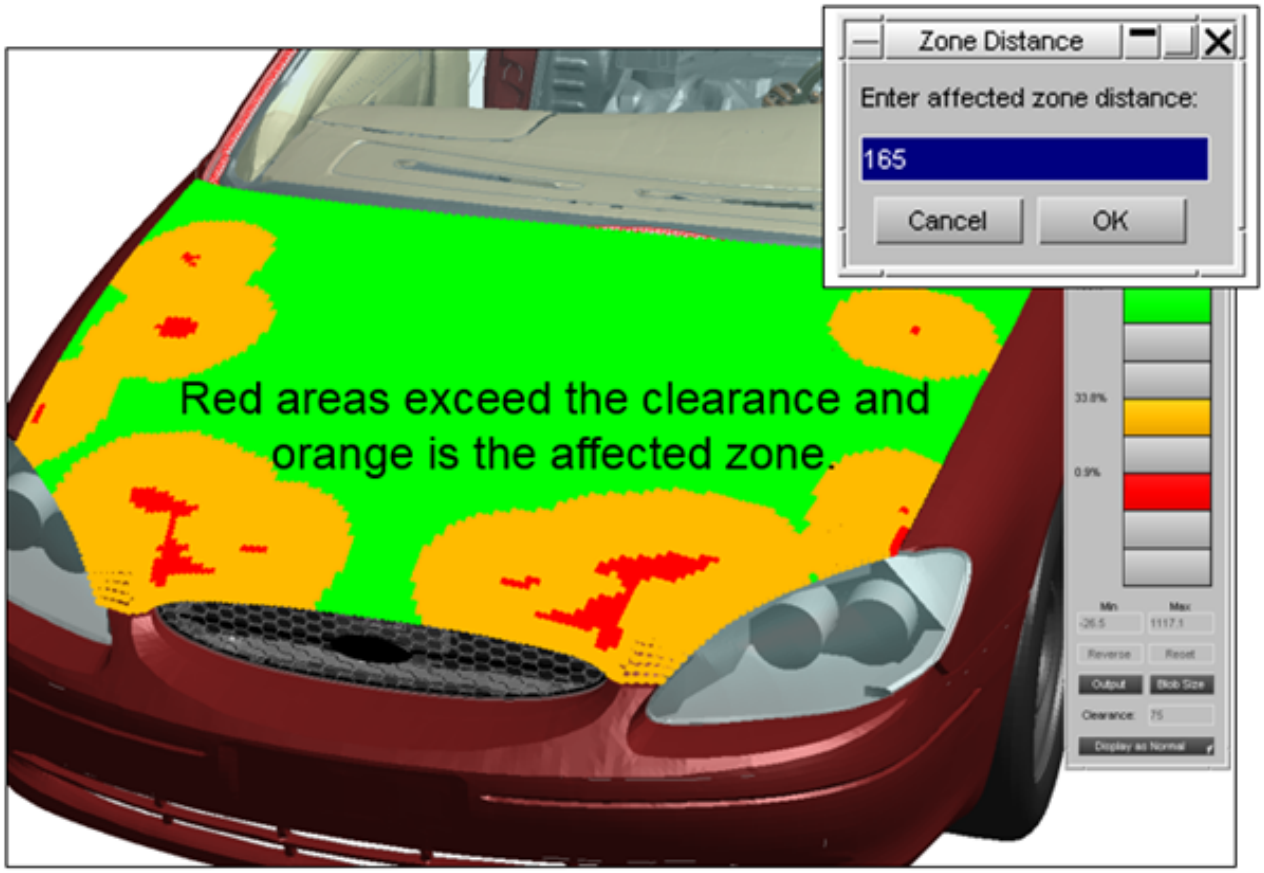
Clearance Plotting

It is also possible to input a clearance value which represents a ‘keep out’ zone.



PRIMER applies this as an offset to the measured values so that any negative values represent those that are exceeding the clearance.

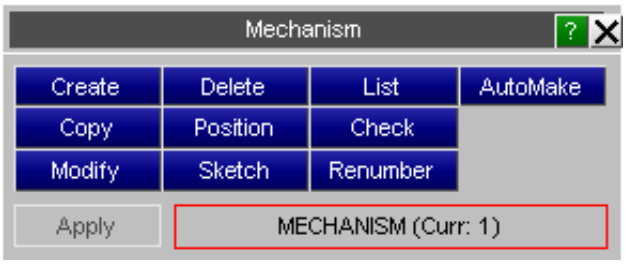
It is sometimes useful to understand what % of the measured area has exceeded the clearance. This is possible using the ‘Display as Area’ button, which becomes active once a clearance is defined. The user can also specify an affected zone distance, representing areas which are influenced by the exceeded clearance.


Zone Distance
Enter affected zone distance:
165
Cancel
OK
Red areas exceed the clearance and
orange is the affected zone.

6.30 MECHANISM Creating and analysing mechanisms

Mechanism analysis is a new capability in PRIMER release 9.3

It enables you to define a mechanism from components of your model and to analyse how they move as a collection of rigid bodies attached by joints of various types.



What is a "mechanism"?

A mechanism is a collection of two or more "assemblies" joined together by "connections".

- **Assemblies** are made up of any number of parts (deformable and/or rigid) and/or node sets, and for the purposes of mechanism analysis they are treated as totally rigid entities. There is currently a limit of 100 assemblies per mechanism.

- **Connections** may be one of a range of simple joints: currently pin, hinge, sliding (line) and coupler. (These are not the same as the *Constrained Joint type in ls-dyna, although they have similar characteristics.)

The mechanism definition is stored in **`*MECHANISM`** cards following **`*END`**, in a format described in Appendix IIb. A model may contain any number of mechanism definitions.

Mechanism analysis is performed using an iterative scheme, so the results are approximate and will result in small distortions of the model. The default error tolerance is 1.0e-4 times the bounding box dimension around the mechanism, or ~0.1mm for a typical 1 metre sized mechanism. This amount of error is usually insignificant, but the convergence tolerance can be tightened, at the expense of longer calculation times, to achieve greater accuracy where required. A special check is included for failure to preserve coincidence of pairs of nodes defining ls-dyna joints, and automatic fixing of this can be carried out.

Once a mechanism has been created its motion can be analysed using rigid body kinematics: an assembly can be dragged using the mouse and the motion of the whole mechanism is calculated and displayed interactively. The new position can be saved and the coordinates of the assemblies updated to provide a new structural configuration for subsequent analysis.

Mechanisms may also have "points" defined within them. These provide both a means of creating restraints and also of driving motion.

Mechanisms may also be organised in a hierarchy where a "parent" mechanism drives the motion of a "child" via connected degrees of freedom. The "child" may also be a Dummy definition, permitting the dummy's position to be driven by the motion of its parent mechanism.

> (There are many similarities between Mechanisms and Dummies in PRIMER , and indeed when positioning a Dummy in "free dragging" mode the dummy is in fact turned into a mechanism during the drag operation.)

Mechanism Keywords in the input deck.

The information describing the mechanism is saved in special keywords following the *END card.

The formats of these cards (which are similar to those of a Dummy tree file) are given in Appendix IIb

Primer also reads mechanism information from ANSA comments in the input deck.

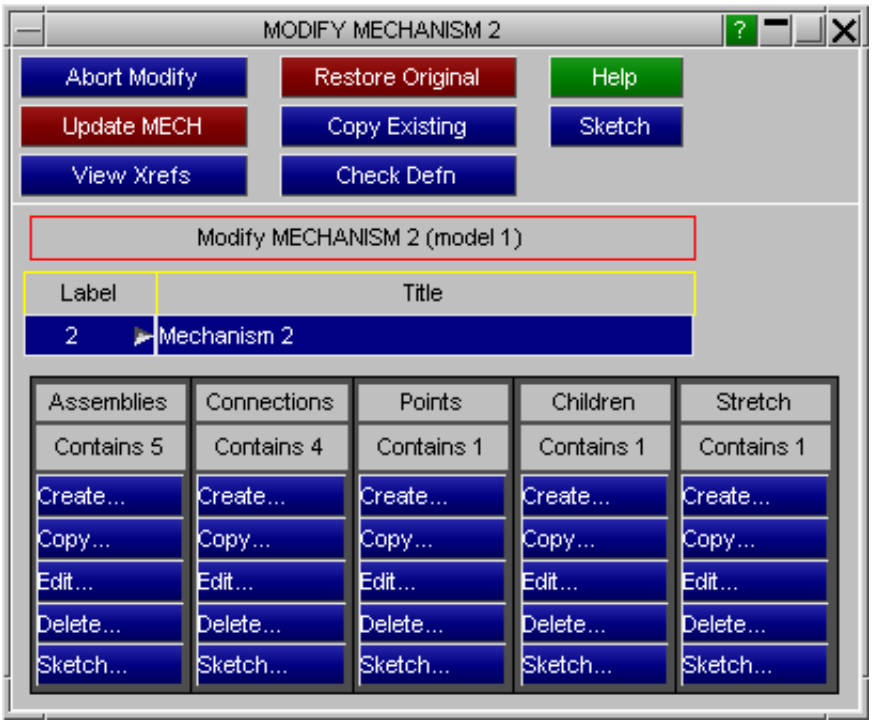
6.30.1 Creating and Editing mechanisms

Mechanism definitions contain Assemblies, Connections, Points and Children. They may also specify Stretches.

Assemblies are collections of one or more parts and/or node sets. The parts may be any permutation of rigid or deformable.

Connections join assemblies together. At present PRIMER contains four connection types: pin, hinge, line and coupler.

Points are optional, and any number may be defined. They are coordinates in space, "tied to" and a property of their parent assembly, that may have restraints in any combination degrees of freedom. If a local coordinate system is defined for a point then any restraints act in that system. Points may also be used to drive movement.



Children are optional. They may be other assemblies or Dummy definitions, and their motion is driven by their parent mechanism. Motion is transmitted in selected degrees of freedom from parent to child.

Stretch definitions are also optional. They allow you to define parts of the structure that are not included in the mechanism itself, but which will be "stretched" by the mechanism's movement. Typical examples might be coil springs in a vehicle suspension.

In addition from V12 onwards PRIMER will automatically determine **`*DATABASE_CROSS_SECTION`** definitions that "belong to" assemblies and update their motion with those assemblies, see Applying motion to Database Cross Sections below for more details.

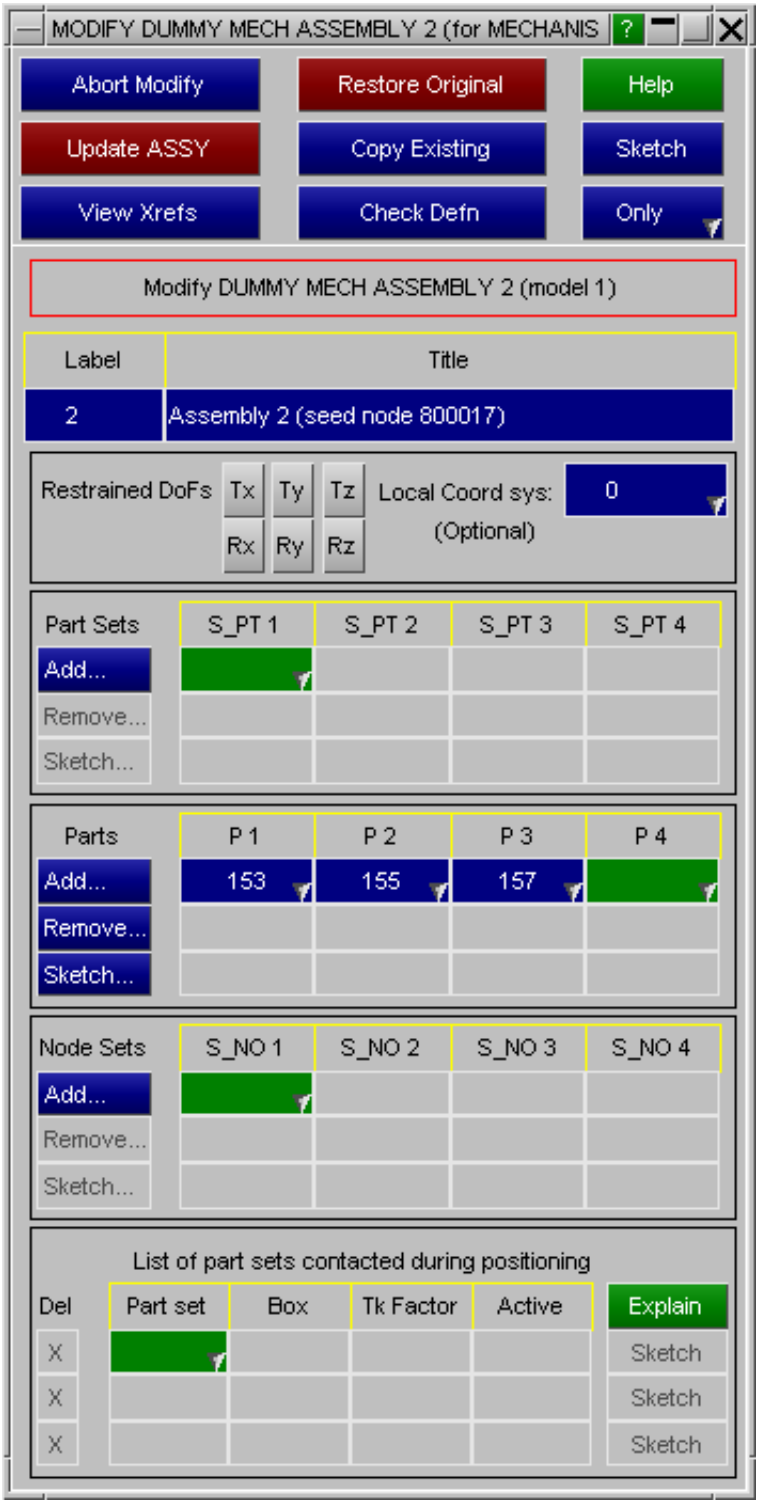
Use **Create...**, **Copy..., Edit...**, etc to select and operate on the relevant items.

When the definition is correct use **UPDATE_MECH** to save it.

Automake: Automatic creation of mechanisms

From PRIMER 11 onwards you can also create mechanisms automatically using the kinematic properties of the LS-DYNA model, this process is described below. An automatically created mechanism is exactly the same as one created manually, and can be edited and processed in same way.

Assembly Creation and Editing



Label and title — An assembly must have a label. Labels are "private" to this dummy, thus in a model with 2 dummies each may have assembly label 1 and they will not clash.

A title is optional but is recommended.

Restraints and coord system — Assemblies may have optional restraints. These are used during "free drag" positioning to constrain movement.

If a coordinate system is defined restraints are in that system, otherwise they are in the global system.

Part sets and parts — Any number of parts may be used in any permutation of explicit parts and part sets. It is legal to define a part more than once (eg in a set and explicitly): it will only be used once.

Node sets — Node sets may be added to the assembly. All nodes in these sets will move with the assembly.

Contact — You can choose to define a list of part sets that will be contacted during positioning.

If part sets are defined then contact between them and the assembly will be calculated during positioning. This will slow down the positioning process considerably because of the extra work required to perform the contact calculations.

Part sets for contact are assumed to remain stationary during positioning, and implicitly thus to be "fixed structure". If you define part sets on another assembly, and that assembly undergoes any significant movement during positioning then the contact may fail.

To visualise what is in an assembly you can use the standard **Only** button functionality to show just its contents on the screen.

(A significant difference between Dummy and Mechanism assemblies is that mechanism assemblies do not have any sense of hierarchy: there is no concept of a "root assembly" in a mechanism, nor is there a "tree" of connectivity. All assemblies have the same seniority, and the connections between assemblies can be completely arbitrary.)

Mechanism assemblies no longer "lock" their contents against deletion

Prior to release 10.2 of PRIMER usage of a Part, Part set or Node set by a mechanism assembly "locked" these against deletion. Several users requested that this logic be changed, so from release 10.2 (April 2011) onwards Parts, Part sets and Node sets used in assemblies can be deleted using **REMOVE, DELETE_UNWANTED** at will.

Therefore it is possible to remove the entire contents of a mechanism assembly via deletion. An empty assembly may become invalid if an attempt is made to position a mechanism that contains it, but this is checked prior to positioning and you will be warned if it is the case.

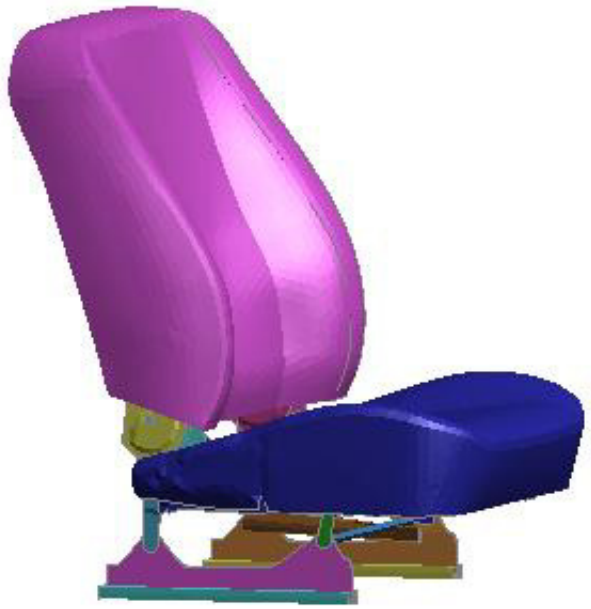
Note that, despite their similarity, ***Dummy*** assemblies ***do*** still lock their contents against deletion. This is because Dummies and their associated positioning data (tree files) are usually built by specialists as separate models, and it is likely that an attempt by a user to delete their contents will be unintended.

Connection creation and editing

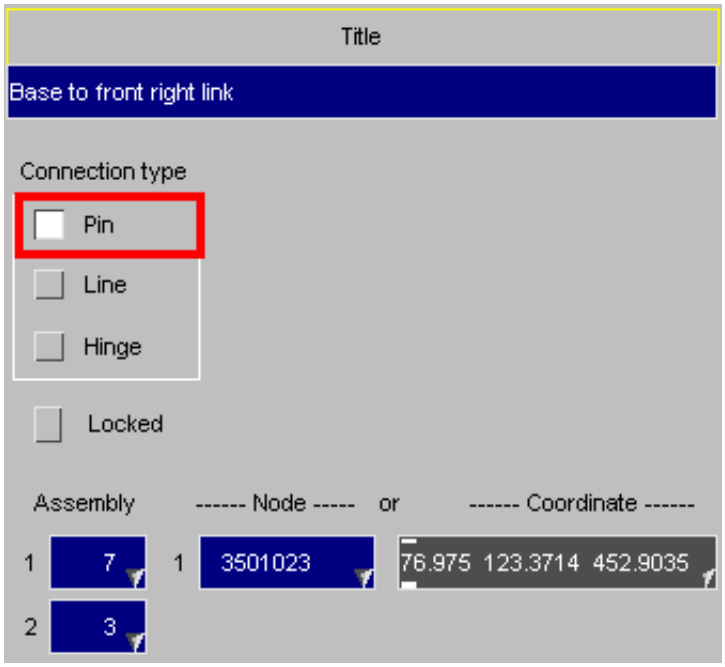
There are three types of connection available, demonstrated here by their usage in a seat mechanism.

The examples below show how the various assemblies of a seat mechanism have been joined together with connections.

This seat has runners attached to the floor in which sliders travel fore and aft. The base is attached to the sliders by links, and the seat back to the base by a hinge at its base.



PIN joint



Two assemblies are specified (the order doesn't matter).

Either a node or a coordinate is also specified at the pin location point. The coordinates of the node, or the specified coordinate, are used to define the connection point (a parametric coordinate) on both assemblies.

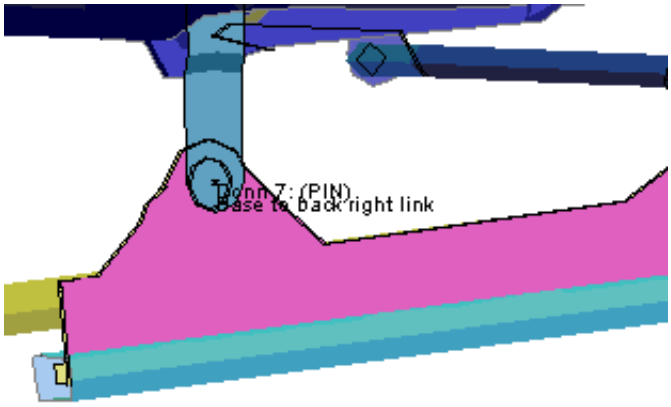
- If a node is specified it does not have to be on either assembly, but it is recommended that it is on one of them since, if its motion is not updated as the assemblies move, the connection point will move relative to the assemblies.

- If a coordinate is specified it is implicitly tied to assembly #1, and its motion will be updated with that assembly.

This joint, like all types, may be locked: see below for an explanation of this.

The pin acts like a spherical joint, providing connectivity in Tx, Ty, Tz; but no rotational constraint in its default unlocked condition.

In this example a pin joint has been used to connect the link between the sliding base of the a seat and the cushion frame.



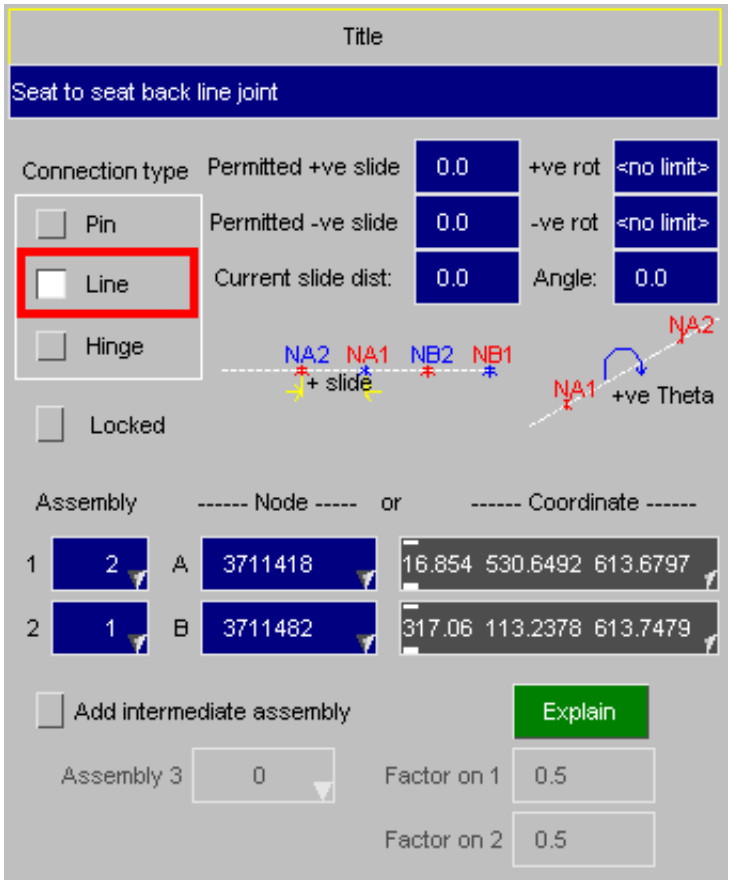
A PIN connection joins two assemblies at a point, acting like a spherical joint.

LINE joint

A LINE joint connects two assemblies 1 and 2 along the line between points A and B. Either or boths points may be defined by nodes (NA and NB), or by explicit coordinates.

- If nodes are used neither node has to be on either assembly, but it is best to put both on one or the other so that they move as the assemblies move.
- If coordinates are used then their position is "tied" to that of assembly A only. A single assembly is used to prevent build-up of rounding errors during positioning.

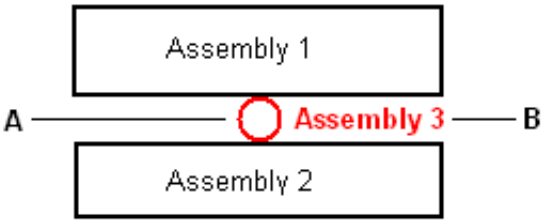
In this example a LINE joint has been used to model the sliding of the runners fore and aft in the guide rails attached to the floor.



Optional assembly 3

There is also the option of adding a 3rd "intermediate" assembly between the first two, with its motion specified as some proportion P of assembly #1, and implicitly (1-P) of assembly 2.

The most common usage of this would be when defining a roller between two assemblies (here assembly 3 in red), where the roller motion is geared to that of assemblies 1 and 2.



There is no "master/slave" relationship: all assemblies are equal, and any assembly can drive the motion of the other two.Motion is constrained to axis A-B, the separation shown here is artificial.

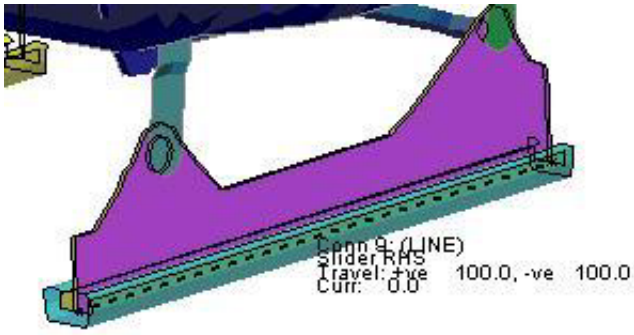
Motion is constrained to:

Sliding along the axis A - B

Travel is limited in both +ve and -ve directions to the permitted distances given, with the sign convention being:

+ve travel is assembly 1 moving along the vector A -> B

The "current slide distance" is set to zero when the connection is first defined, and thereafter is updated as the mechanism is analysed. The permitted limits and current value can be reset manually at any time in this panel.



A LINE joint allows assemblies 1 and 2 to slide along and rotate about axis NA-NB within the limits specified.

Rotation about axis A - B

By default there are no limits to this rotation, as shown in this example.

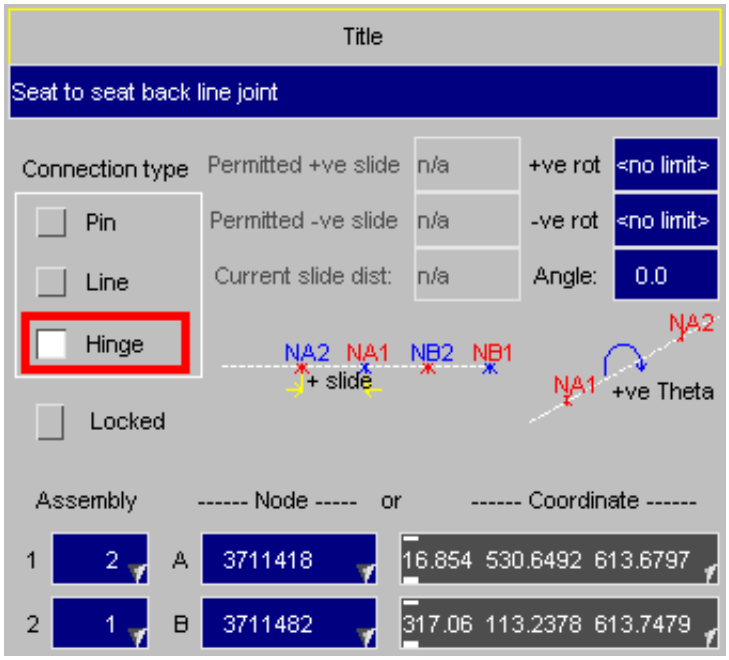
If limits are defined they should be expressed in degrees, the +ve rotation in the range 0 to +180 and the -ve rotation in the range 0 to -180. Rotation will be clamped to these limits.

+ve rotation is clockwise about vector A -> B using a right hand rule.

The "current rotation angle" is set to zero when the connection is first defined, and is updated and can be reset manually in exactly the same way as the translation distance - its actual value is notional.

If the "current rotation angle" value is changed it is important that any limiting rotation angles are also changed if necessary so that the -ve limit is less than or equal to the current value, and the +ve limit is greater than or equal to it.

HINGE joint



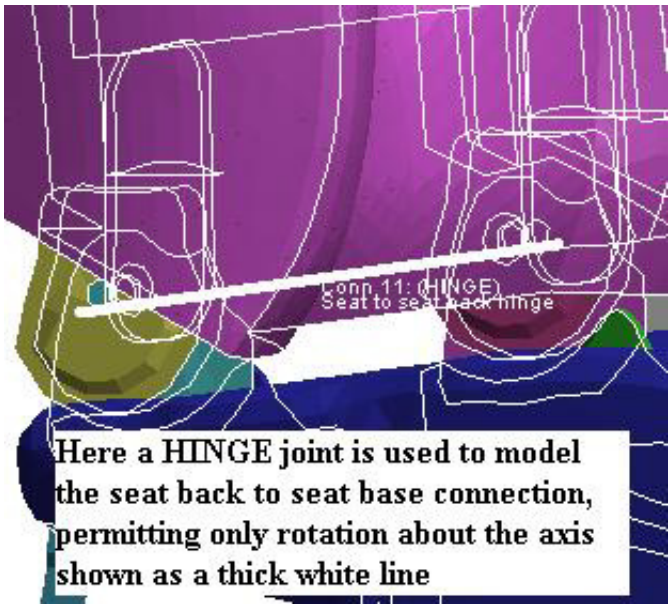
A HINGE joint also connects two assemblies 1 and 2 along the line between points A and B, each point being defined either by a node or an explicit coordinate. It is exactly like the LINE joint above except that translation along the axis A-B is not permitted.

- If nodes are used neither node has to be on either assembly, but it is best to put both on one or the other so that they move as the assemblies move.
- If coordinates are used then their position is "tied" to that of assembly A only. A single assembly is used to prevent build-up of rounding errors during positioning.

In this example a HINGE joint has been used to model the seat back to seat base connection, permitting only tilting backwards and forwards about the transverse axis.

Only rotation about the axis A-B is permitted, within the limits specified, translation along that axis being restrained.

(A LINE joint with its permitted translation distances set to zero is exactly the same as a HINGE joint.)



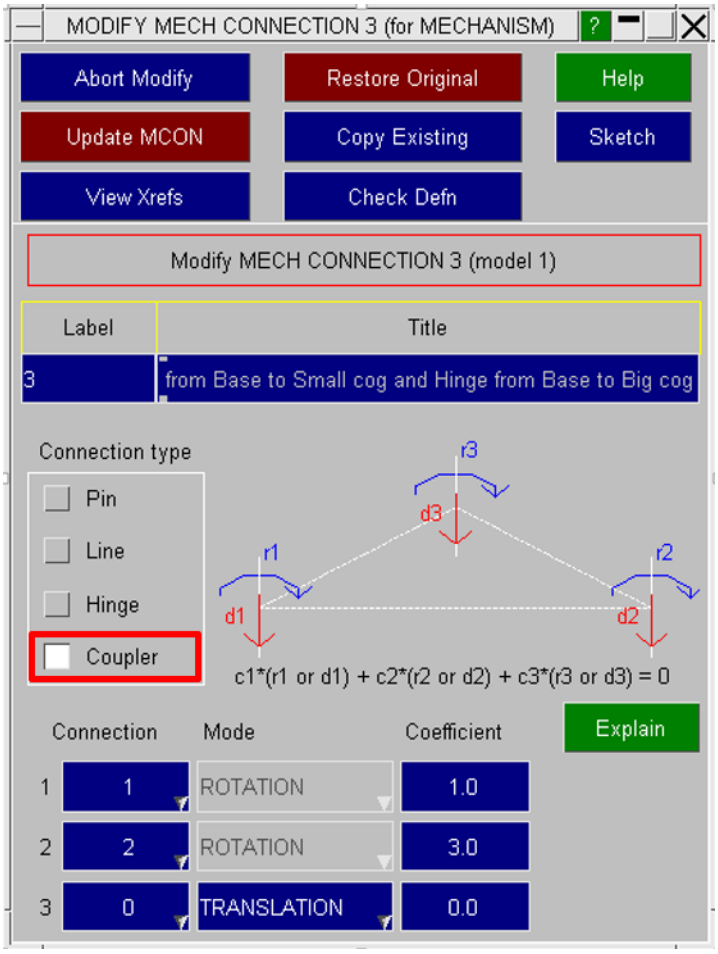
COUPLER joint

A COUPLER joint defines a linear equation

c1*(r1 or d1) + c2*(r2 or d2) = 0 or

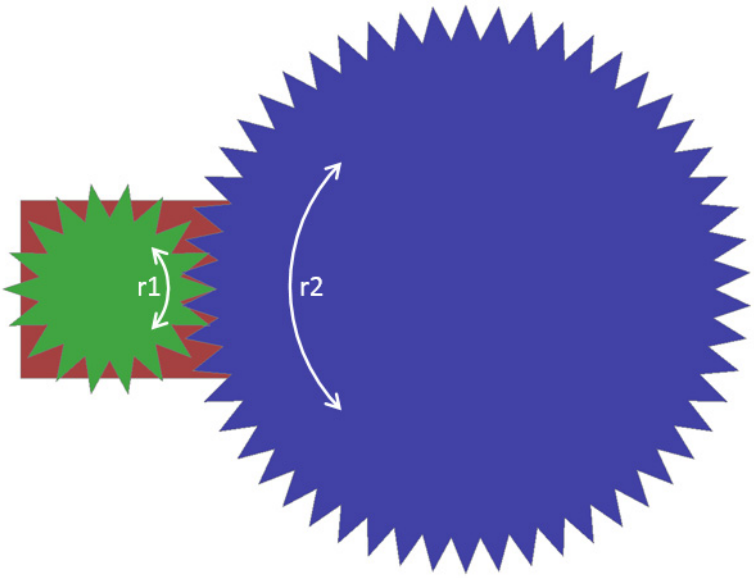
c1*(r1 or d1) + c2*(r2 or d2) + c3*(r3 or d3) = 0

between rotation angles r1, r2, r3 (in radians) and/or slide distances d1, d2, d3 of two or three LINE and/or HINGE connections. For each of the connections the mode on the edit panel specifies whether it is translation or rotation in the linear equation. Only for line connections both options are available, whereas for hinge connection only rotation can be coupled. The coefficients c1, c2 and c3 are those defining the equation. In the most common case with only two connections the third connection should be left as 0 on the edit panel, and Primer will ignore the settings for the mode and coefficient.



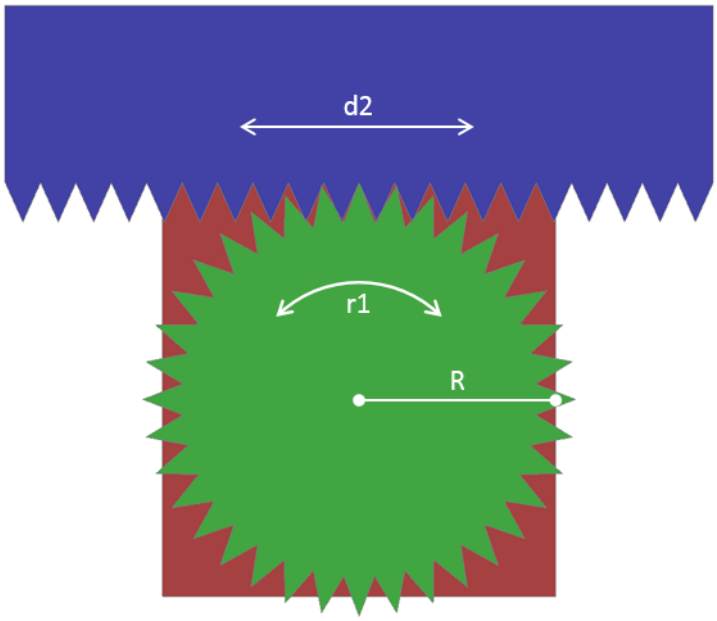
Example of coupler between rotation and rotation

Here the two gears are attached by hinge connections to the red base assembly, where the axes of the hinges are orthogonal to the plane of the red assembly. In the picture on the right the radius of the blue gear is three times the radius of the green gear. Therefore the rotation angle r1 of the green gear should always be three times the rotation angle r2 of the blue gear, but in opposite direction. This can be defined as a coupler with equation c1*r1 + c2*r2 = 0 by setting the coefficient c1 for the green gear axis to 1.0 and c2 for the blue gear axis to 3.0. For more general radii, the coefficients need to be proportional to the respective radii of the gears. Note that scaling all coupler coefficients by the same non-zero constant does not have any effect on the meaning of the coupler.



Example of coupler between rotation and translation

Here the green gear is again attached to the red base assembly by a hinge with axis orthogonal to the base. The blue rack can slide horizontally, which can be defined as a line connection between the blue and the red assemblies with horizontal axis and zero stop angles. As the gear rotates by the angle r1 in radians, the blue rack should translate by a distance d2 which the teeth of the gear move by its rotation. Since angles are measured in radians here, we have got d2 = R*r1, where R is the radius of the green gear. This can be defined as a coupler wit h equation c1*r1 + c2*d2 = 0, where c1 = R and c2 = 1.0 or c2 = -1.0. The correct sign will depend on the orientations when defining the hinge and line connections.



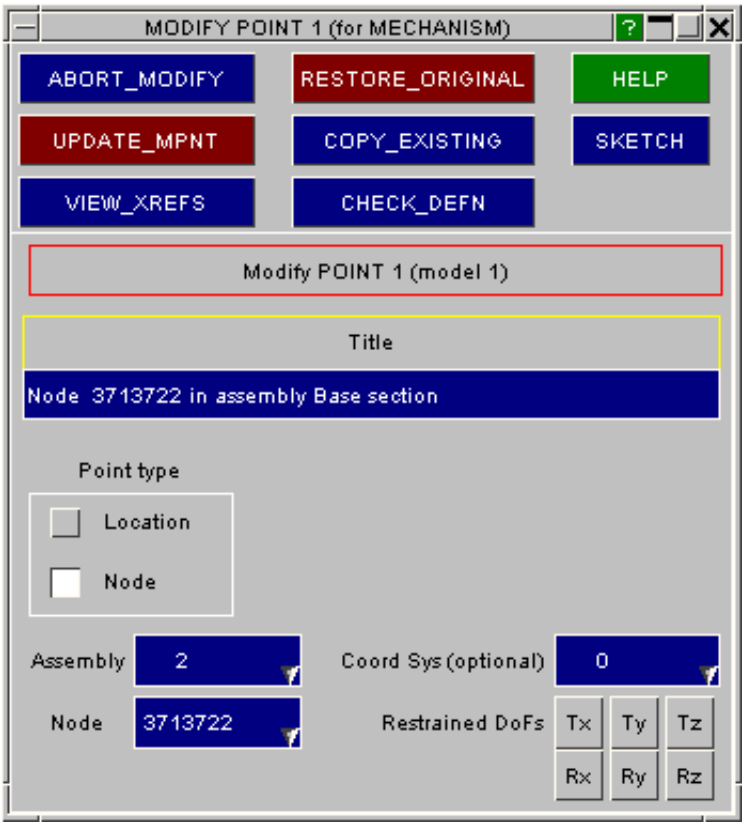
Locking connections.

It is normally the case that connections will be required to articulate in the expected degrees of freedom while analysing a mechanism, but there are also times when it is useful to be able to lock a connection completely. For example in the seat example above you might move the fore/aft slider in its base runner to some position, and then lock it there. Another example might be a headrest which must be slid to a particular height and then locked into position.

When a connection is locked all six degrees of freedom are constrained, and the effect is as if the two assemblies on either side have been merged into a single rigid assembly.

Connections may be locked and unlocked at any time, either on the connection editing panel as shown above or on the main positioning panel when it is in "Connection list" mode.

Point creation and editing



Title	A title will be generated automatically, but you can supersede this with your own. Points do not have labels
Point type	A point defined by **Location** is a coordinate in space that is attached to, and moves with, its parent assembly. A point defined by **Node** is essentially the same: it obtains it current coordinate from the node. (However if the node is not on a part or node set of the assembly it will not move with the assembly.) The node should normally be part of the parent assembly, but this is not mandatory.

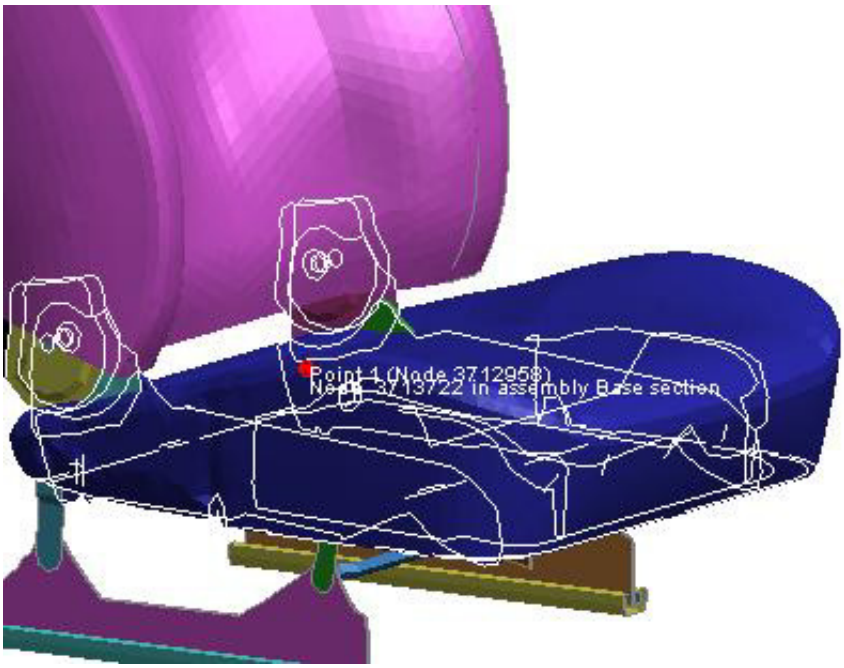
Restraints and coordinate systems	A point's movement may be restrained in any combination of degrees of freedom (or none). If a local coordinate system is defined restraints act in that system, otherwise they are global.

Visualising Points

Points may be visualised by using the **Sketch** options both on the parent dummy panel and on their edit/create panels.

Here is a picture of the point in the example above: it is a reference point in the seat base.

It is shown as a circular red symbol, labelled with its title, with the free edges of the parent part also displayed.



CHILD mechanisms

It is possible to define a mechanism or dummy that is a "child" to this mechanism. You define the following:

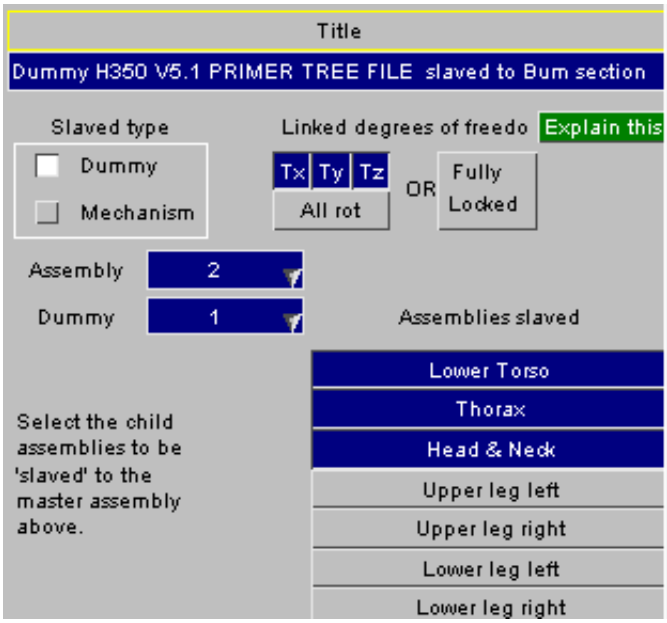
The child type (here a Dummy) and the mechanism or dummy label.

The master assembly on this mechanism.

The degrees of freedom to be linked.

The assemblies on the "child" to be linked via these degrees of freedom to the master assembly. Here the Lower Torso, Thorax and Head & Neck have been linked.

Child mechanisms may be nested to any level (child has child has child ...). Dummies may not have children.



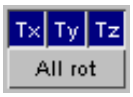
Warning: Mechanisms may not be recursive.

This means that a mechanism may not refer to itself as a child either directly (mechanism A has child mechanism A) or indirectly (mechanism A has child B which itself has child A). A moment's thought will reveal why this should be so: a mechanism cannot "drive" its own motion!

Primer will detect any attempts to create recursive mechanisms and report this an error; the positioner will also reject recursive mechanisms.

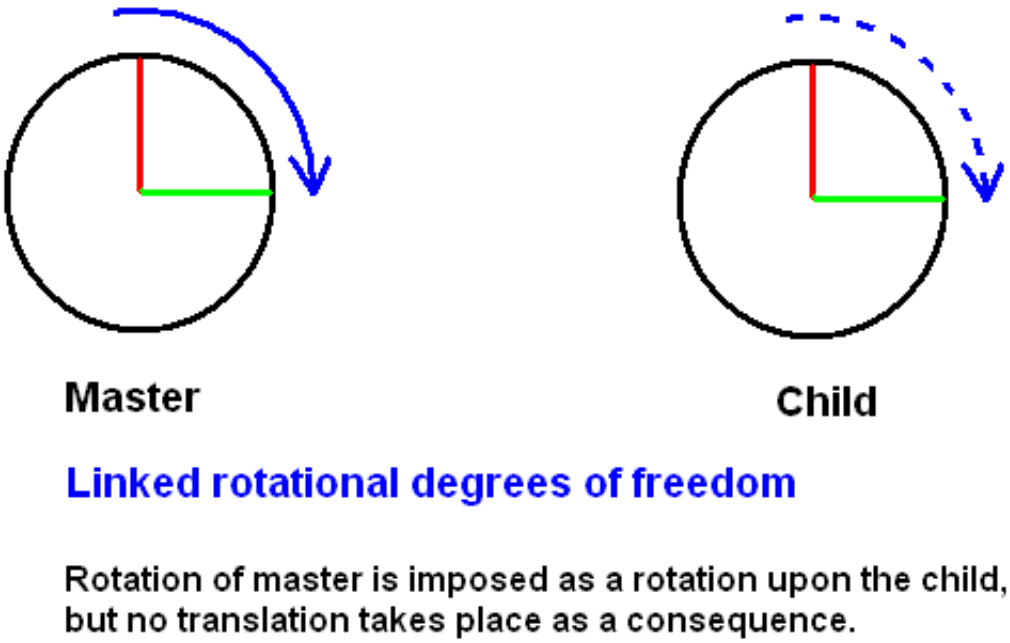
Linked degrees of freedom of children

PRIMER offers two related but different ways of slaving the motion of the child to its master.

	Linked degrees of freedom: **Tx, Ty, Tz** and **All rot**. The effect of these is similar to **`*CONSTRAINED_NODE_SET`** in that the chosen degrees of freedom of the master assembly are imposed on the child. Any permutation of the translational DoFs (Tx/y/z) and/or all rotational DoFs may be chosen. (Linking of individual rotational DoFs is not supported.)
	Fully locked The effect of this is like **`*CONSTRAINED_RIGID_BODY`**: the slave assemblies are merged into the master one to form a single rigid body.

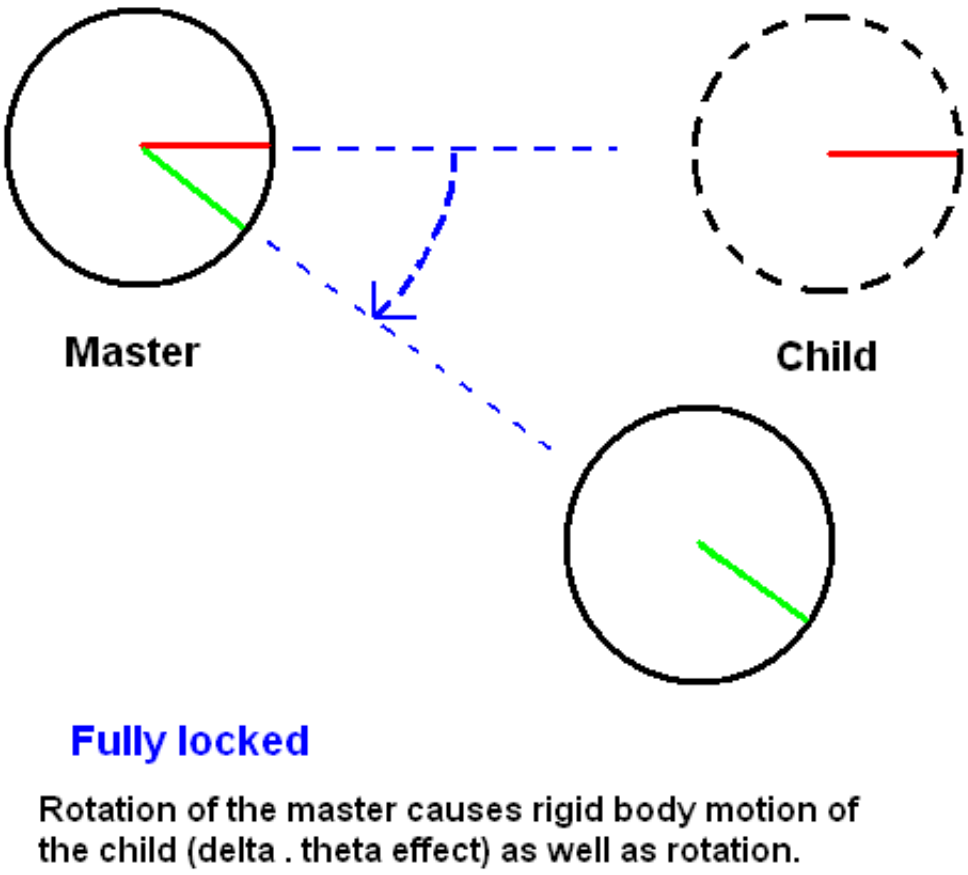
The effect of translation is easy enough to understand, and the two methods have the same effect in pure translation, but there are important differences between these two methods where rotations are concerned. In particular:

Selecting all linked degrees of freedom (**Tx**, **Ty**, **Tz** and **All rot**) is ***not the same as*** using **Fully locked**. The following figures explain why.



When rotational degrees of freedom are linked the rotation of the master is imposed on the child, but no translation arises from this.

A good way of thinking about this is to consider the master and child assemblies to be connected by a chain, like the pedals and back wheel of a bicycle. Rotating the pedals causes the back wheel to rotate, but has no tendency to try to lift it into the air.



When the master and child are fully locked then the child is both rotated and translated by the motion of the master since they are effectively a single rigid body.

Why have the two alternative linking methods?

Although **Fully locked** might at first sight appear to be the logical choice, experience has shown that when slaving dummies to seats the most natural behaviour is obtained if only the translational degrees of freedom (**Tx**, **Ty**, **Tz**) are linked. This is because any rotation of the seat cushion is not transferred to the dummy, which can remain in its upright position looking straight ahead even if the seat tilts underneath it.

How Child Mechanisms & Dummies work

When a child is slaved to a master mechanism the motion of the master assembly is imposed on the child assemblies in the degrees of freedom specified as described above.

During analysis the motion of the master assembly is computed and then applied to the child assemblies. There is feedback of force from the child to the master, so movement of the master will be constrained if it tries to push the child against a restraint. However in other respects it is a one-way treatment: moving child assemblies will not cause the master mechanism to move.

This is best demonstrated by example. Here a Dummy has been positioned in a cockpit, on a seat, and the dummy is a child of the seat linked in Tx, Ty, Tz.

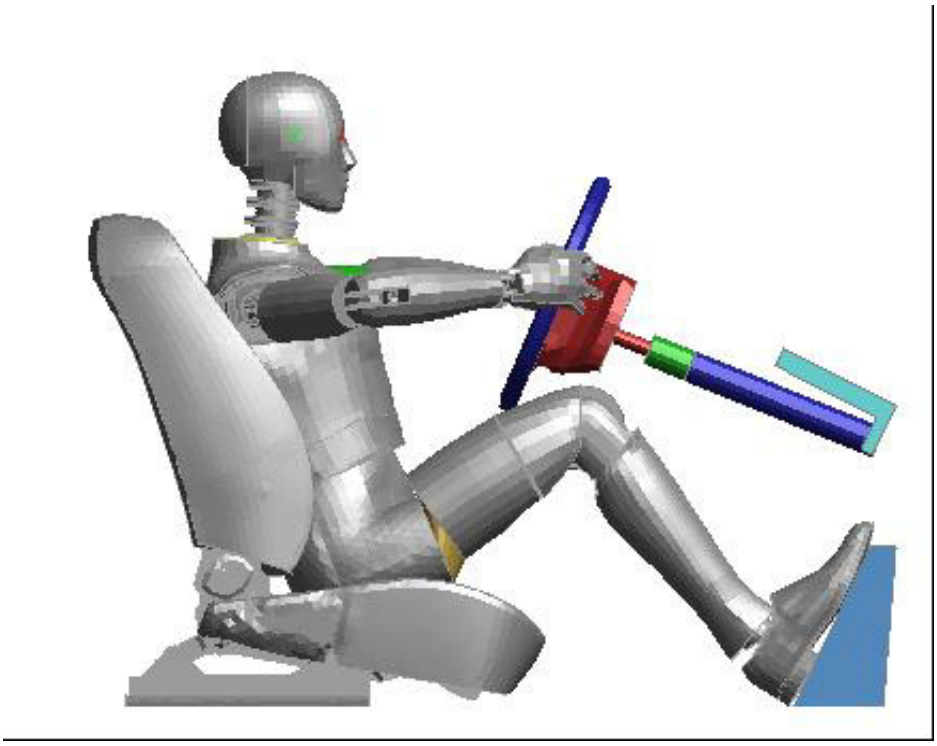
In this example the seat has been moved forward and down to a ridiculous degree, but this demonstrates two things clearly:

(1) The dummy motion has remained linked to that of the seat.

(2) Connection between seat and dummy is in translation (Tx, Ty, Tz) only.

This is made clear by the way that the seat cushion has tilted down but the pelvis, torso and head of the dummy have not rotated.

A more detailed exposition of the use of a Dummy as a Child of a mechanism is given in section 6.14.4 Using dummies as "children" of mechanisms, from which this image is taken.



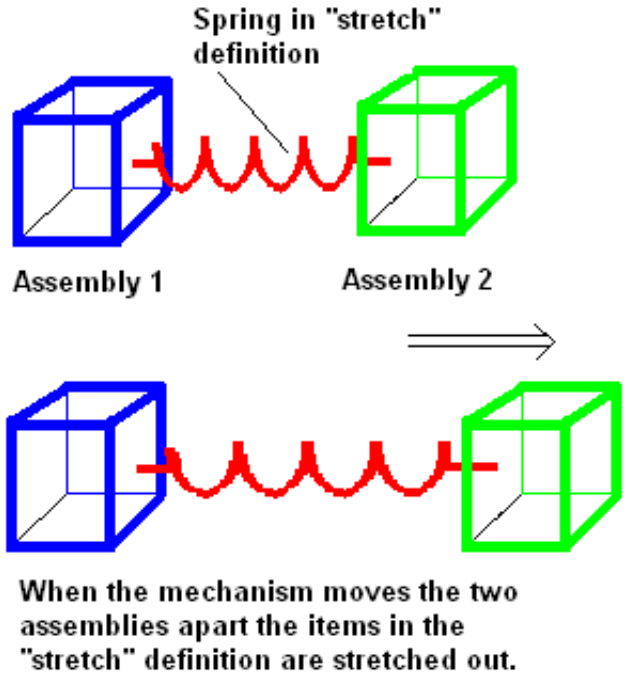
STRETCH definitions (From V11 onwards)

These allow you to define parts of the structure which are not part of the mechanism, but which will be "stretched" by mechanism movement.

Typical examples might be a coil spring modelled explicitly in a suspension tower which will be contracted or expanded by movement of the suspension below.

In the diagram here the blue and green blocks are mechanism assemblies, while the red spring is in a **stretch** definition (but not in any assembly). When the blocks are pulled apart the mechanism positioner stretches the red spring using the relative displacement of the two assemblies to define how it should be modified.

The structure being stretched does not have to be a simple spring, it can be of arbitrary complexity.



Defining a **Stretch** definition.

The structure to be stretched is defined, like an assembly, by any permutation of Parts, Part sets or Node sets.

Stretching is defined by the relative motion of two nodes on ends 1 and 2 respectively. At least one node should be on a mechanism assembly otherwise no motion will occur during positioning. Nodes do not have to be on the structure being stretched.

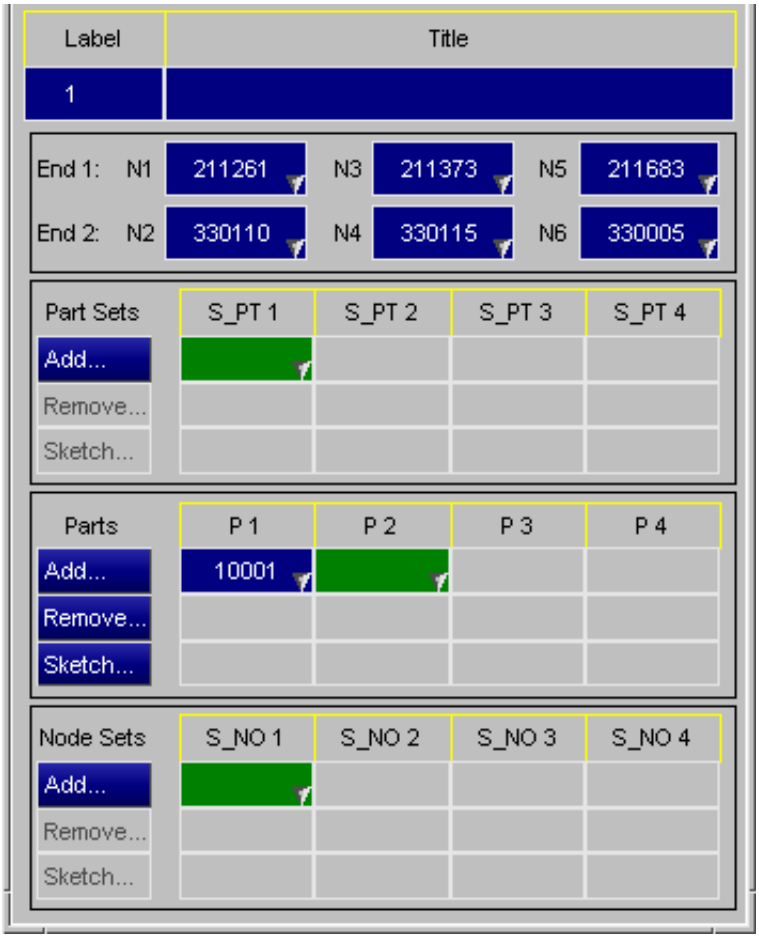
Two possibilities exist for end fixity:

(1) Pinned ends: Simple linear stretching, translation only.

In this case only nodes N1 and N2 are defined, and only linear displacement between the two will be used giving a "simple" translational stretch as in the example above.

(2) Encastre ends: Complex stretching, including rotation.

In this case all three nodes at an end must be defined, and should form a well-conditioned triad capable of forming a local coordinate system with its origin, the point of fixity, at the first node. For example at end 1 the three nodes N1, N3, and N5 must be defined:

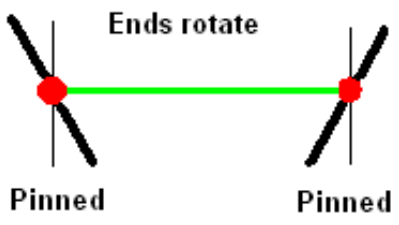
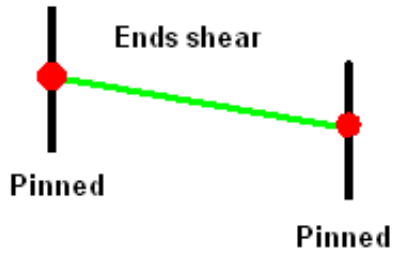
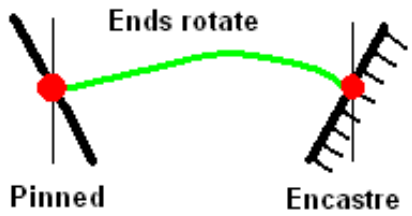
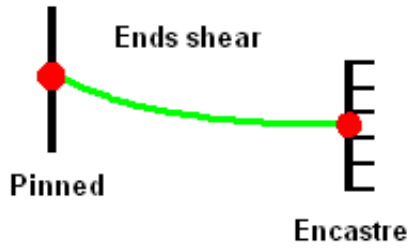
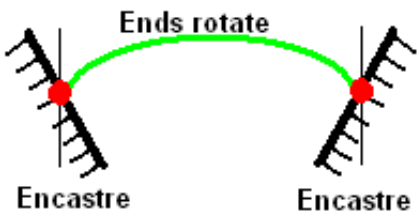
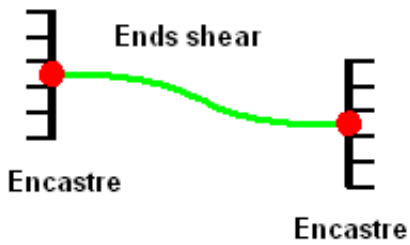


If encastre fixity is required at end 2 then nodes N2, N4, N6 must form a similar local system, with its origin at N2.

(Although the 3 nodes at an encastre end form a local coordinate system, and this is used internally, the orientation of this system is not important. However you may find it helpful to align it with the approximate directions of expected movement, as this may make it easier to visualise motion.)

Stretching is interpolated from the relative movements of ends 1 and 2. Translation is always included, but if an end is encastre then rotation and bending due to "rotation times lever arm" effects will also be included. You can have any permutation of fixity methods at ends 1 and 2, and the following diagrams show how end fixity will influence the shape of the structure between the ends. (Those who have studied bending of beams should find these images familiar.)

Examples of how end fixity affects stretched shape.

Both ends pinned		
One end pinned, one end encastre		
Both ends encastre		

Defining the structure to be stretched.

This consists of any permutation of parts, part sets and node sets and is defined in exactly the same way as for assemblies above. There are no restrictions upon membership of a stretch definition, but it is recommended that parts and nodes already in an assembly in this mechanism should not be used since this would lead to their position being "driven" by two possibly conflicting methods.

As with assemblies membership of a **Mechanism Stretch** definition does not "lock" parts etc against deletion, but membership of a **Dummy Stretch** definition does lock them.

How stretch is applied

Stretch is applied by updating nodal coordinates, and in the case of rigid parts using _INERTIA definitions the Centre of Gravity of the part is also updated.

Generally the structure to be stretched should ie in the region between nodes 1 and 2 because of the way motion is interpolated:

- Structure between N1 and N2 gets moved by (P1 x motion of N1) plus (P2 x motion of N2), where proportions P1 and P2 are calculated as follows:

  P1 = projection of vector between node to be moved N and N1 onto the vector N1N2 divided by the length of vector N1N2
  P2 = (1.0 - P1)

- Structure "behind" N1, ie P1 is < 0.0, gets 100% of the motion of N1, and none from N2.

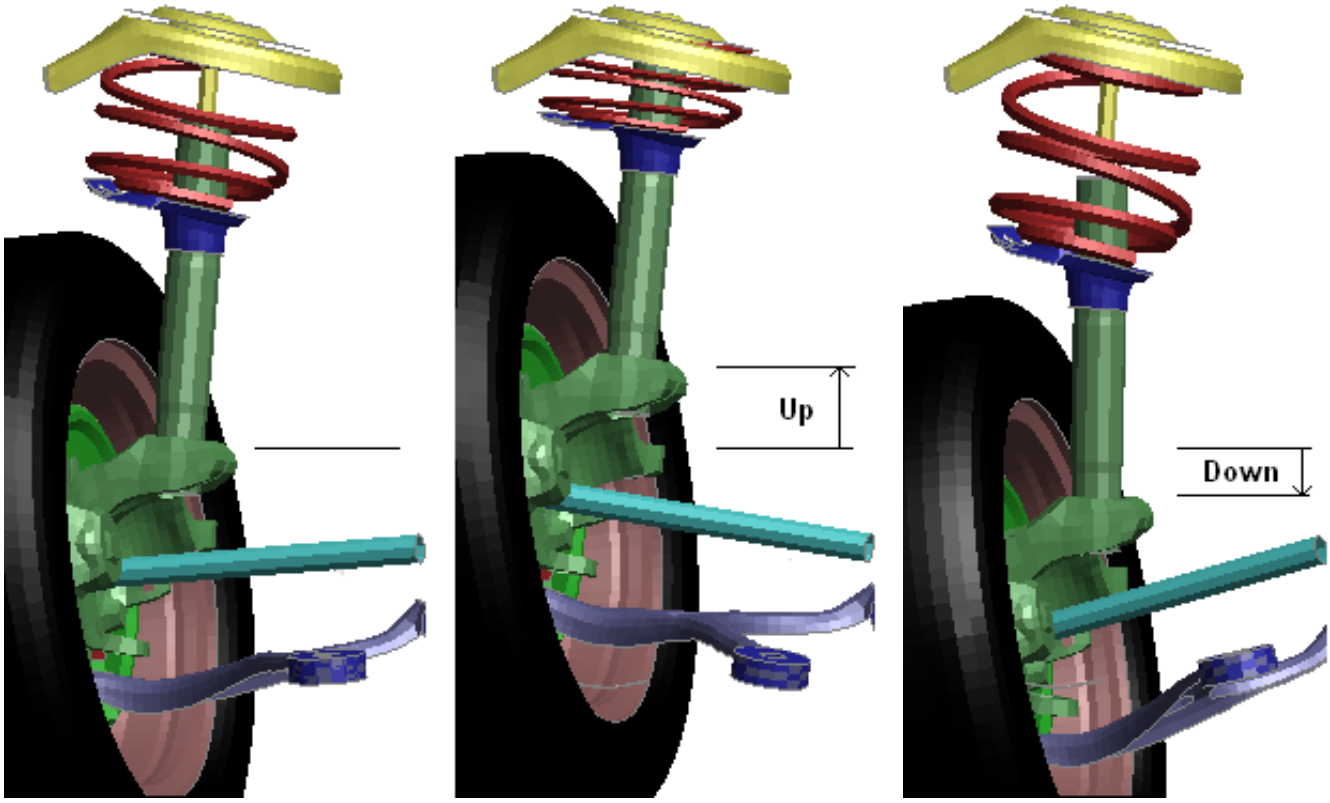
- Structure "behind" N2, ie P1 is > 1.0, gets 100% of the motion of N2, and none from N1.

When an end is encastre then an additional motion based on the (rotation of that end) x (the distance from that end's node) x (the proportion P) is applied. This gives a "delta x theta" motion.

Example of Stretch in action

Here is an example from a real model showing how an explicitly modelled spring at the top of a shock absorber tower is compressed and expanded. Here the wheel, brake assembly, wishbone and drive-shaft are all assemblies in a mechanism that allows them to move up and down. The top of the shock absorber tower (yellow) is fixed.

A **Stretch** definition with pinned ends has been made between the top of the shock absorber column and the top of the tower. The structure to be stretched is the spring (red) and its bracket (blue)



Neutral position Wheel moved up Wheel pulled down

This example illustrates two limitations of this approach:

1. It can be seen that interpolation has thinned the spring material in the centre image, and thickened it in the right hand one. This will clearly affect both the mass and the stiffness of the spring, and indeed any structure made of solid or thick shell elements that is stretched in this way.

2. In the centre image it can also be seen that the spring is just "poking through" the top of the tower. A better choice of node at the top of the tower (a bit lower down) would fix this problem, but it has been left to illustrate the point that stretching material may give rise to penetrations.

WARNING: **Stretch** is a purely geometrical transformation that distorts elements.

When structure is stretched it is likely that volume, and hence mass, will be added to or removed from the model when element sizes are changed. In the case of solid elements it is likely that structural thicknesses will also be changed.

It is your responsibility to ensure that such distortions of your model geometry are not excessive and do not invalidate it in any way.

Applying motion to *DATABASE_CROSS_SECTION definitions.

From release 12 onwards PRIMER will automatically find **`*DATABASE_CROSS_SECTION`** definitions in a model that "belong to" a mechanism assembly, and will update its motion as the assembly moves. In order to belong to and assembly a cross section must obey the following rules:

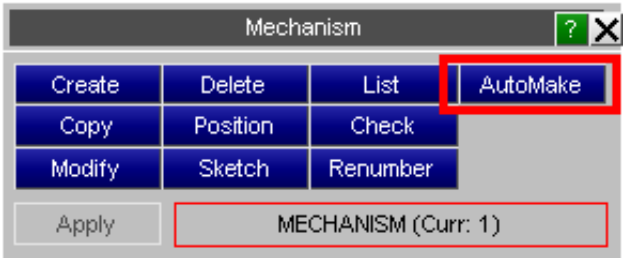
Rules that a *DATABASE_CROSS_SECTION must satisfy to "Belong to" a Mechanism assembly

It must be of type **_PLANE**	Sections of type **_SET** are ignored by the mechanism positioner because they don;t have an explicit location. (You can still use sections of this type and they will be carried through to the analysis, but because their location is determined by the nodes and elements that define them the positioner does not have to worry about them. In fact they may be a good solution if you want a cross section to include nodes and elements in multiple assemblies.)
It must have part set **PSID** defined.	Sections with **PSID = 0**, ie all parts in the model, are ignored.
At least one part in set **PSID** must be in this assembly, and other parts must either be in this assembly or not in any assembly in this mechanism. Shell parts referencing ***MAT_NULL** are omitted from this check.	If no parts are in this assembly it is ignored. If parts are in both this assembly and also one or more other assemblies in this mechanism then motion is ambiguous so the section will be ignored.

If a section has parts in more than one assembly in this mechanism then a warning is issued prior to positioning, and you are given the option of "cloning" the section into as many definitions as necessary to create sections that are unique to each assembly. Each "clone" is a new section definition that is geometrically identical to the original, but in which **PSID** only contains the subset of parts present in a given assembly. These clones can be positioned with their respective assemblies since their motion is no longer ambiguous, but the original section definition (which is left unchanged) will not be moved.

By default "move cross-section with assembly" is on, but you can turn it off in the Options panel of the positioner. You also have the option of turning it off if the pre-positioning warning detects sections spanning multiple assemblies. This on/off status is recorded in the mechanism section of the keyout file - see Appendix IIb for details.

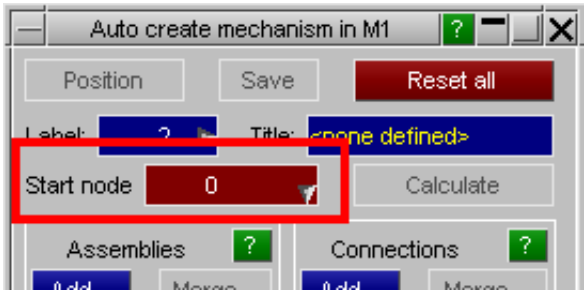
6.30.2 **Automake**: creating mechanisms automatically



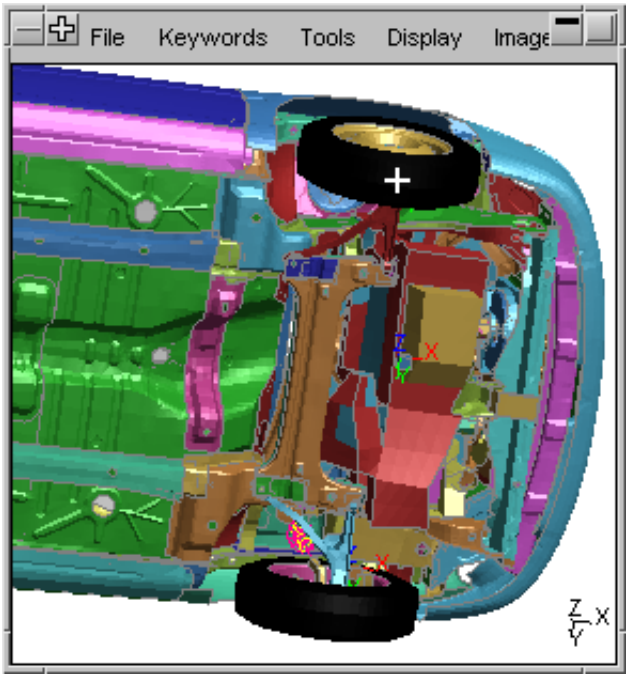
As well as creating mechanisms manually PRIMER 11 onwards can calculate mechanisms automatically from the structure of your model.

How it works

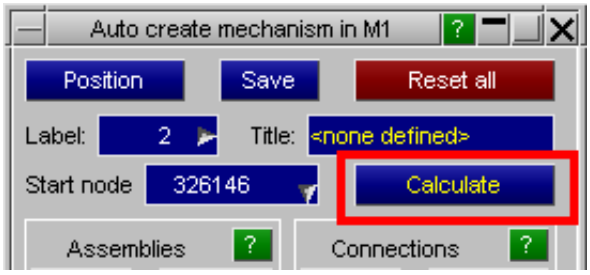
(1) Select an initial "seed" node, the **Start node**

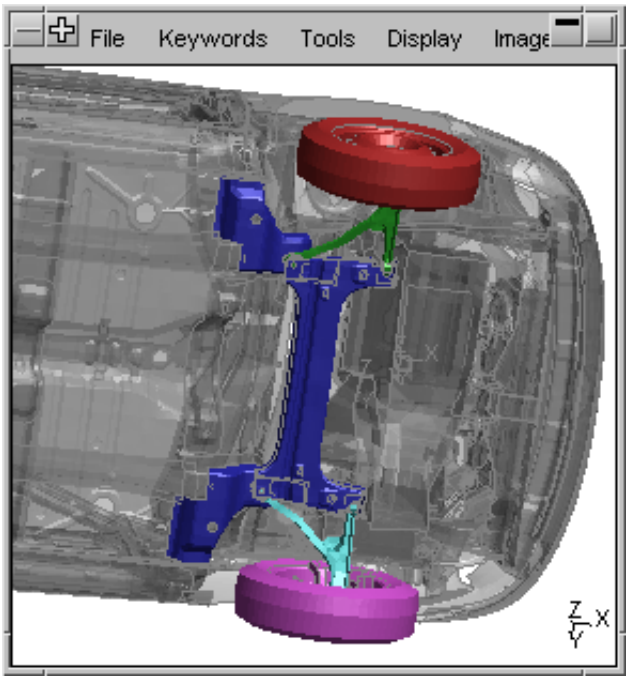


In this example, designed to create a mechanism from the front suspension of a vehicle model, the user has picked a node on the front tyre, shown here with a cross.



(2) Press **Calculate**





PRIMER will "grow" the first assembly, here the red wheel, outwards from that seed node until no more connected structure can be found, at which point the first assembly has been defined.

It then looks for potential connections, in the form of joints (*CONSTRAINED JOINT) or discrete elements (*ELEMENT DISCRETE, or *ELEMENT BEAM type 6) on that assembly.

If any are found then the 2nd node on these forms a seed node on a new assembly, and the growth processes is repeated. If a valid assembly can be created then PRIMER uses the joint or discrete element to make a connection.

This "grow assembly", "look for connections", "make new assembly on the other side" process is repeated until no more eligible structure can be found, at which point the mechanism is complete. In this example it can be seen that five assemblies have been created:

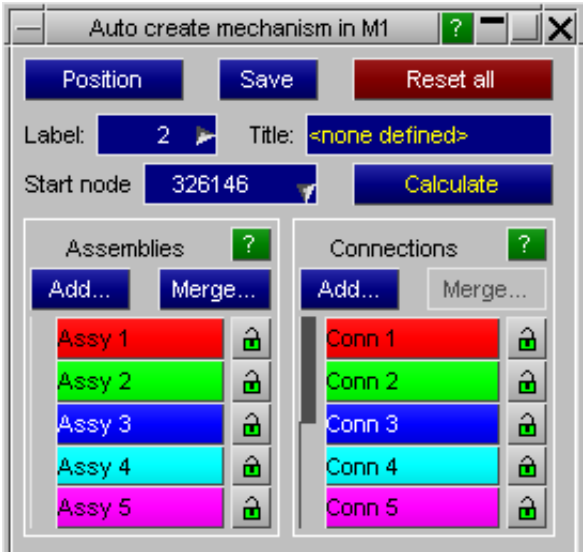
- Initially selected wheel (red)
- Wishbone (green)
- Subframe (blue)
- Other wishbone (cyan)
- Other wheel (magenta)

During this process PRIMER provides visual feedback by colouring each assembly as it is created, with the rest of the model being made transparent, as shown in the example here. (These are the default settings, to alter them see "Controlling how things are drawn" below.)

You can use **Position** to try out positioning this mechanism, and if you are happy with it you can **Save** it in the database.

In an ideal world this will produce a mechanism that models exactly what the model will do during the analysis, and sometimes it does, but in most cases you will need to adjust the "growth" procedure to achieve the result you want. The various controls available to you are described below.

Once you have used **Calculate** for the first time the **Automake** panel will become populated with the details of the mechanism you have created. You can adjust the settings here and repeat the **Calculate** process as many times as you like until you achieve the results you require.



Assemblies and **Connections** list what has been created.

These buttons are colour coded to match the display in the graphics window, and if you hover the mouse over a button the relevant item will be highlighted and labelled graphically, making it easy to identify items.

To edit an assembly or connection click on its row button, and this will map the relevant editing panel as shown above. Automatically calculated mechanisms are always "scratch" definitions until they are saved, so you can edit items without fear of affecting any other mechanisms saved in your model.

"Locking" assemblies and connections using

Normally each new **Calculate** operation will delete all assemblies and connections made previously and restart from scratch, meaning that any edits you have made will be lost. If you want to keep any item during this process then "lock" it using this button, and it will be retained. A locked assembly or connection will show a closed padlock coloured in red, in addition if you lock a connection then it also implicitly locks any assemblies to which it is attached, and their padlock symbols will turn blue to reflect this.

Note:

- "Locking" an assembly in this context does ***not*** restrain it in any way during positioning. To do this edit it by clicking on its row, and select the combination of restraints that you require.
- In the same way "Locking" a connection here does ***not*** "clamp together" its assemblies in any way. To do this edit the connection and select the "Locked" button on that panel.

Merge... "Merging" assemblies

Sometimes pieces of structure are split up into more assemblies than necessary, and while it would be possible to adjust the attributes of connections to correct this it is easier just to merge the offending assemblies directly. **Merge...** does this for you in the following way:

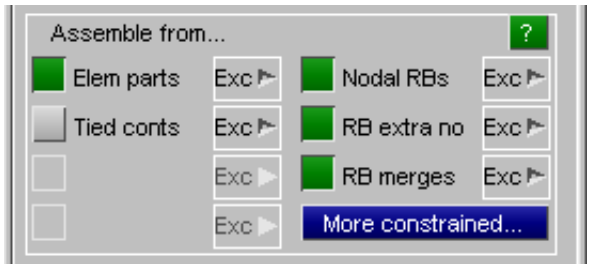
- You are given a list of all the assemblies in the current mechanism-to-be.
- You must select two or more of these
- All the selected assemblies will be merged into the first one.

The merge process does the following:

- All part sets, parts and node sets in assemblies #2 to #n are added to assembly 1.
- All assemblies #2 to #n are deleted.
- All connections between a pair of assemblies both in the list #1 to #n are deleted.
- All connections referencing an assembly #2 to #n has this connection reassigned to #1.
- Assembly #1 is locked to prevent any later passes deleting and/or modifying it.

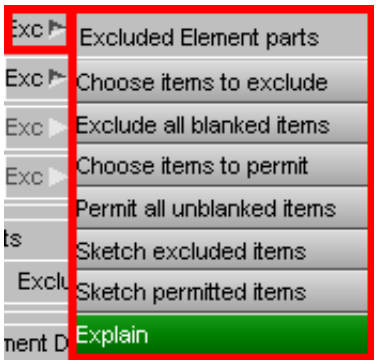
Assemble from... controls the items used to create assemblies.

Assembly growth will only occur across the item types selected below.



Elem parts. (Solid, shell, beam and thick shell parts)	Will grow via nodal connectivity only for deformable parts, ie growth only occurs across elements connected by common nodes. For rigid parts growth extends to all elements in the part regardless of connectivity.
Nodal RBs	Nodal Rigid Bodies grow across all nodes in the body, which add their node set to the assembly.
RB Extra no	Extra nodes on rigid bodies grow to all such nodes. Their nodes are not added to the assembly since these will be picked up anyway by virtue of the parent part being used.
RB Merges	Rigid body merges (*CONSTRAINED RIGID BODIES) grow to the "other" part, either slave or master, including it in the assembly
More constrained	Maps a sub panel listing other *CONSTRAINED items that can cause connectivity, for example welds. The relevant parts, nodes or elements get added to the assembly.
Tied conts	Tied contacts, which includes spotweld contacts, can be used to track across tied nodes. This is off by default since it can result in "too much growth" unless controlled.

Controlling what is used for assemblies via "Exclusion".



By default all these categories (except tied contacts) are turned on, and hence eligible for assembly growth. However you can turn any category off at two levels:

1. Turn off the whole category by deselecting its button.
2. Turn off individual coponents within it by using the [**Exc**] "exclusion" button.

Exclusion allows you to deselect individual parts, constrained items and contacts so that they do not contribute to growth in a **Calculate** operation, and this can be used to limit what is considered.

Selective exclusion can be changed at any time, and used in combination with "locking" of assemblies you might exclude some items, eg contacts, to grow initial assemblies, then lock these and re-enable growth across the excluded items for subsequent **Calculate** operations.

Connect via joints

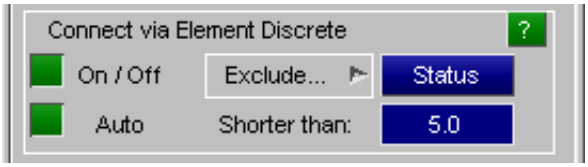
Controlling which joints make connections

The folowing subset of joints (*CONSTRAINED JOINT) types are used to make connections between assemblies

Joint type	Mechanism connection type created.
SPHERICAL	Pin joint at the location of N1
REVOLUTE	Hinge joint from N2 to N4
CYLINDRICAL	Line joint from N2 to N4, no restraint on sliding or rotation
UNIVERSAL	Pin joint at the location of N1
TRANSLATIONAL	Line joint from N2 to N4, sliding unlimited but rotation forbidden.
Other types	*Are currently not considered*

Exclude... may be used as above to exclude any joint from being eligible to make a connection.

Connect via Element Discrete



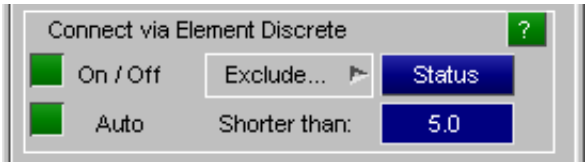
Controlling which discrete elements (springs and dampers defined by *ELEMENT DISCRETE) make connections.

"Short" discrete elements, usually springs, are sometimes used to model flexible connections, so this option allows you to make mechanism connections using them.

Discrete element type	Length	Mechanism type created
2 noded springs and dampers	<= **Shorter than** value	Pin joint at average location of N1 and N2
1 noded springs and dampers	n/a	Fully restrained point on parent assembly at location of N1

Exclude... may be used as above to exclude any element from being eligible to make a connection.

Connect via Element Beam



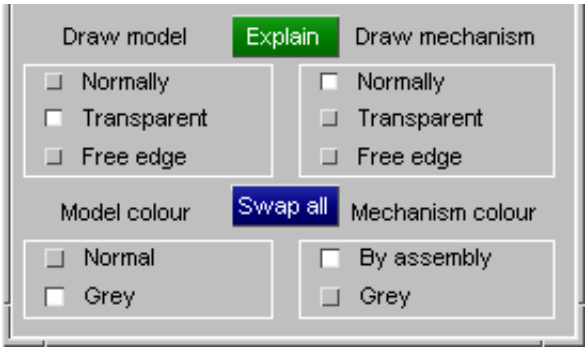
Controlling which discrete beam elements (*ELEMENT BEAM using section type ELFORM = 6) make connections.

"Short" discrete beam elements using section formulation ELFORM = 6, often of zero length, are also often used to make flexible connections between components in a model.

Beam element type	Length	Mechanism type created
2 noded beams of element formulation (ELFORM) 6	<= **Shorter than** value	Pin joint at average location of N1 and N2
other beam types	*n/a*	*Are not considered*

Exclude... may be used as above to exclude any element from being eligible to make a connection.

Controlling how things are drawn



Generally a mechanism forms only a small part of a larger model, and it can sometimes be hard to see what is happening. To try to relieve this problem the **Automake** function temporarily changes the visual attributes of the plot, breaking the model down into two categories:

"Model"	Everything in the model that is ***not*** in the mechanism By default this is drawn in transparent grey.
"Mechanism"	Those parts of the model in the mechanism. By default this is drawn "normally" (ie opaque) in the colours of the various mechanism assemblies.

You can change these settings at any time, and the graphics will update immediately. **Swap all** swaps over all model and mechanism settings, and can be useful when trying to see what is not in the mechanism.

When you finally exit from Automake the original model colours and other graphics attributes will be restored automatically.

Other commands

Position performing a test mechanism positioning operation.

Once you have created some assemblies and connections you can use **Position** at any time to try positioning them. This will launch the mechanism positioner on you current definition.

Warning: Within **Automake** your mechanism is always a "scratch" definition, so anything you do with it will not affect items stored permanently in the database. However an exception to this is that if you perform a positioning operation and then **Accept** rather than **Reject** the results any changes you have made to nodal coordinates and rigid body attributes ***will become permanent***.

Reset All resets all settings in the Automake panel.

If your mechanism has gone horribly wrong, or you simply want to start again with a new one, then **Reset all** will reset to default the following:

- All existing mechanisms and connections are deleted, whether or not they are "locked".
- All exclusion lists for assemblies and connections are deleted.

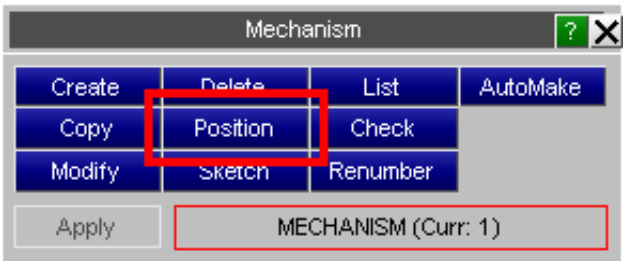
Save stores your mechanism in the database.

If you are happy with your mechanism you can store it permanently in the database with **Save**. This will also exit from the **Automake** panel, and restore all model colours and visual attributes.

[X] To exit without saving.

If you want to abandon **Automake** without saving your results simply use the [X] button at the top right of the panel. Colours and visual attributes will be restored, but no mechanism data will be saved.

6.30.3 **Position**: Analysing mechanisms



Once a mechanism has been defined, or read in from file, it can be analysed (positioned) in a variety of ways.

When you enter the mechanism positioner several operations are performed:

- Correctness of the mechanism definition is checked. Parts and nodes should not appear in more than one assembly, and you are warned if they do and given some options for diagnosing and correcting these errors. A more detailed treatment of potential errors and good mechanism modelling practice is given under Modelling Rules in Appendix II. (Although these rules refer to Dummies they apply equally to mechanisms.)

- You cannot have both **Dummy** and **Mechanism** positioning active at the same time in the same model. (This is because of the way positioning data is stored: the two processes would conflict.) If you attempt this you will be forced to shut down one operation before you can start the other.

- The current mechanism position is saved as an "initial position". If things go wrong in the positioner you can return to this as any time by using **Reset all**, and if you abort positioning using **Reject** the mechanism will automatically be restored to this position.

The main positioning panel

Assuming that these checks pass you then drop into the positioning panel. For mechanisms this operates in one of four modes:

Rotate angles	In this mode explicit rotation of assemblies about their parent connection node takes place.
Drag assembly	In this mode "free" dragging of the mechanism takes place, combining translation and rotation.
Position points	In this mode points can be defined and edited, and "free" movement performed by giving updated coordinates for them.
Connection list	Lists all connections (in mechanisms only) allowing you to lock them. Also to edit and sketch them.

A mechanism is positioned by any combination of these modes, and when it is satisfactory the user must **Accept** it to make the geometrical changes permanent, or **Reject** it to abandon positioning and restore the original geometry.

Global Accuracy determines the precision with which the mechanism is positioned.

Set colour by assembly temporarily changes the colours used in the graphics so that this mechanism's assemblies are drawn in the standard PRIMER colour sequence, and the rest of the model in light grey. This is automatically switched off when you leave mechanism positioning.

Note that this example contains a Dummy model slaved to the parent seat mechanism. The Assembly names are indented to the right a little to emphasise that they are "children". The "(R)" against "Lower Torso" denotes that it is the Dummy's root assembly.

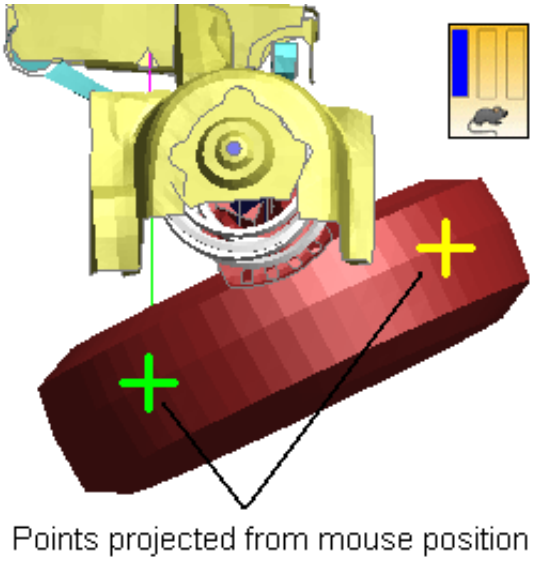
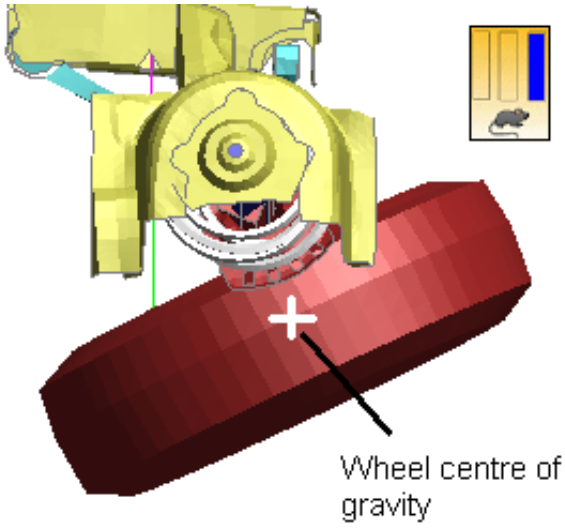
Further positioning commands below describes these and other options in more detail.

Dragging the mechanism with the mouse

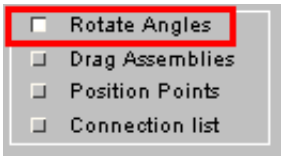
During Mechanism positioning the cursor is always active for picking and dragging assemblies, and operates in the same way (translational drag) in all four positioning modes above. **Note that the method of application of the left mouse button in this context has changed from PRIMER V11.1.**

Mouse button	Action	Mode
Left	Select assembly and apply motion to **virtual point at projected mouse position**.	Motion is always translational drag in the direction implied by the current mouse motion in screen space as projected onto the model. (Note that this cursor behaviour is different to that in Dummy positioning: mouse-driven rotation is not available for mechanisms.)
Middle	Not used in this context	
Right	Select assembly and apply motion to its **Centre of Gravity** *(This was the pre-V11.1 left mouse action)*	

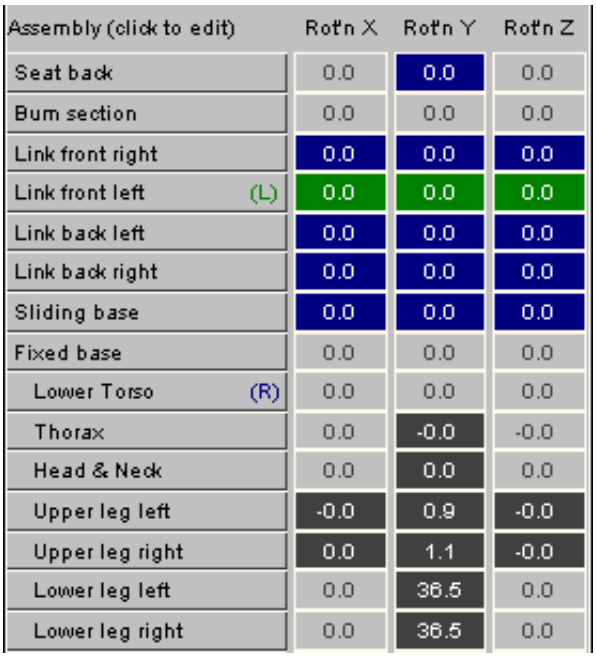
The difference between these two dragging modes is illustrated using the steering and suspension example in the images below.

	
Left mouse: a virtual point is created at the projection of the mouse position onto the assembly, and motion is applied at that point. Therefore in this example motion up and down at the green point will swivel the wheel in the opposite way to similar motion at the yellow point. This intuitive since it is like grabbing and pushing the assembly at the cursor location, and means that both translational and rotational motion can be applied, but it does make the result more variable and - obviously - sensitive to the initial mouse location.	**Right mouse**: motion is always applied at the assembly C of G, regardless of the position of the mouse inside the assembly. Therefore something which swivels, such as a wheel, may move in a direction that is counter-intuitive given the current mouse location, as motion will always be the same regardless of where on the wheel the mouse is located. This is the pre V11.1 behaviour and in most mechanisms, where motion is constrained by connectivity, it gives a satisfactory result. It is retained to permit users to continue to use the "old" method.

Rotate Angles: Rotation of assemblies



Rotating *Mechanism* assemblies:



Assembly (click to edit)	Rot'n X	Rot'n Y	Rot'n Z
Seat back	0.0	0.0	0.0
Bum section	0.0	0.0	0.0
Link front right	0.0	0.0	0.0
Link front left (L)	0.0	0.0	0.0
Link back left	0.0	0.0	0.0
Link back right	0.0	0.0	0.0
Sliding base	0.0	0.0	0.0
Fixed base	0.0	0.0	0.0
Lower Torso (R)	0.0	0.0	0.0
Thorax	0.0	-0.0	-0.0
Head & Neck	0.0	0.0	0.0
Upper leg left	-0.0	0.9	-0.0
Upper leg right	0.0	1.1	-0.0
Lower leg left	0.0	36.5	0.0
Lower leg right	0.0	36.5	0.0

For each assembly a row showing the current joint angles is shown. Angles on a blue background are in the mechanism's native system, those on a green background are in the local axes of the assembly's coordinate system where this has been defined.

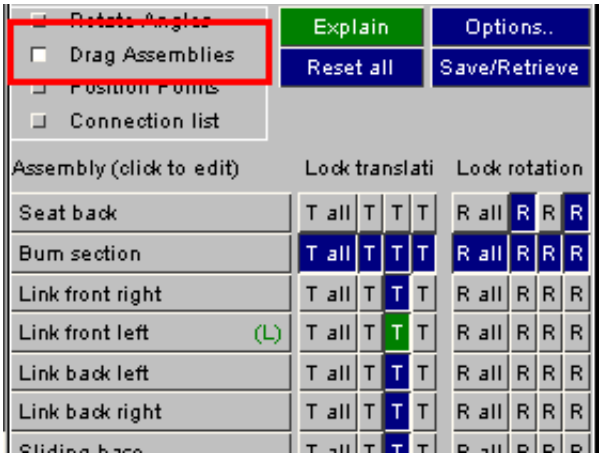
Typing in a new angle will cause the assembly to rotate to the new value using the iterative analysis (free drag) calculation method.

Mechanism assemblies are initially set to angle zero degrees (as shown here) when first defined, and thereafter their rotation angles are computed from their rotation relative to that initial orientation.

Angles are expressed in the mechanism's native system

In order to give consistent calculation and reporting of angles for assemblies, those with no local coordinate system are reported in the Mechanism's native system (on a blue background).

This is assigned automatically and is initially aligned with the global cartesian system, however if the mechanism as a whole is rotated with the **Orient** command then this system also rotates, thus reported angles will not change.



Rotation of mechanism assemblies is only permitted about axes that are not restrained.

You can view the status of these restraints by swapping to the "Drag Assemblies" mode of the positioning panel.

You can see here that the seat base ("Bum section" - a technical term!) is fully restrained in all degrees of freedom, while the seat back is restrained only against rotations Rx and Rz in anticipation of the example rotation below.

Rotating *Dummy* assemblies

Rotation works somewhat differently. For compatibility with the Dummy angle positioning process:

- Angles are computed relative to the coordinate system of the joint stiffness coordinate system at the parent side of the assembly's attachment node.
- Angular changes are applied explicitly via trigonometrical calculation, not via the iterative dragging scheme, and take no notice of assembly or point restraints.
- Both the selected assembly ***and its children*** are rotated as a rigid unit, and rotation only takes place about a single axis at a time.

This behaviour is very different to that of rotations of mechanism assemblies, and the angle entry buttons have a dark grey background to reinforce this point. It is usually best to position a Dummy in the occupant positioner, and then let its behaviour as a "child" of a mechanism be driven by the motion of that parent mechanism.

Please see **Rotate Angles** in the dummy documentation for more details.

Example of positioning a mechanism assembly by rotation.

Remember that the seat back in this demonstration model is connected to the seat base by a HINGE connection permitting rotation only.

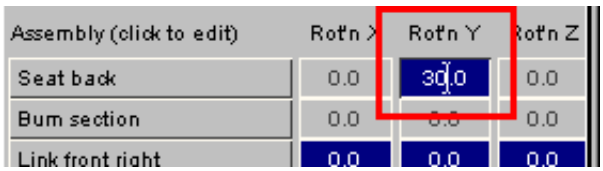
This image shows the seat back in its initial position.

This becomes the initial orientation of 0.0 degrees about all axes.

We propose to rotate by 30 degrees clockwise about the global Y axis, which is coming towards the observer out of the plane of the page.

(*Because the seat has not been rotated as a whole using* **Orient** *the mechanism axes are aligned with the global axes. If the mechanism as a whole is rotated then rotations will take place about the rotated axes - see the notes on the mechanism's native coordinate system above.*)

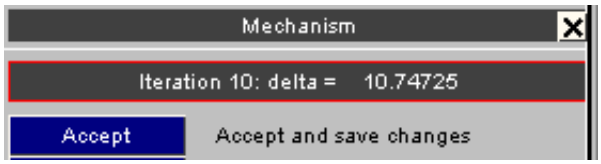
The user types the new angle into the Y rotation column



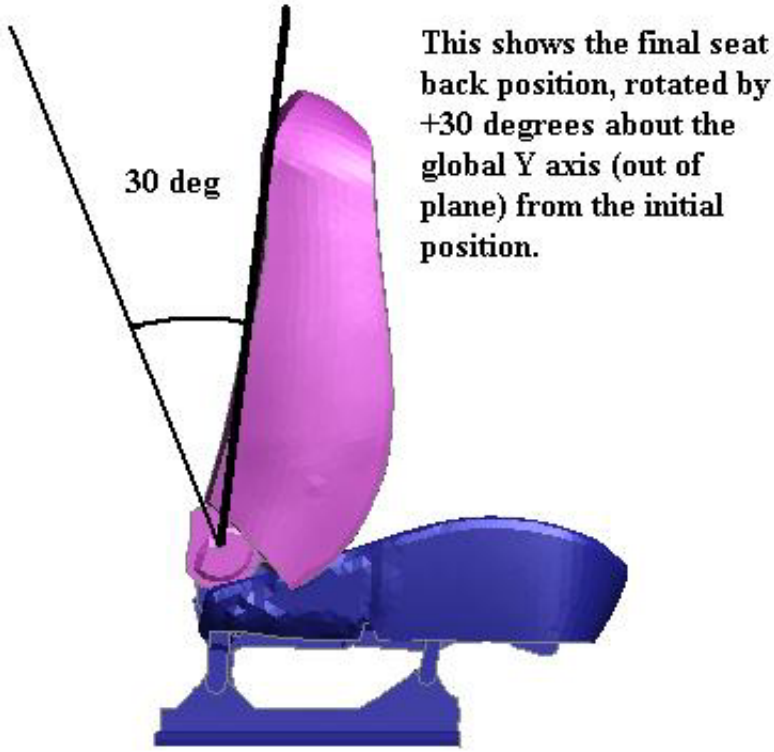
Assembly (click to edit)	Rot'n X	Rot'n Y	Rot'n Z
Seat back	0.0	30.0	0.0
Bum section	0.0	0.0	0.0
Link front right	0.0	0.0	0.0

PRIMER drives the rotation automatically, using the iterative calculation method.

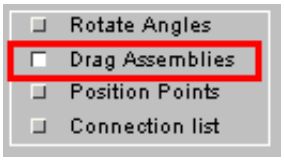
Feedback on progress achieved is given every 10 iterations.



Here is the final position, achieved after about 40 iterations.



Drag Assemblies: Free dragging of assemblies



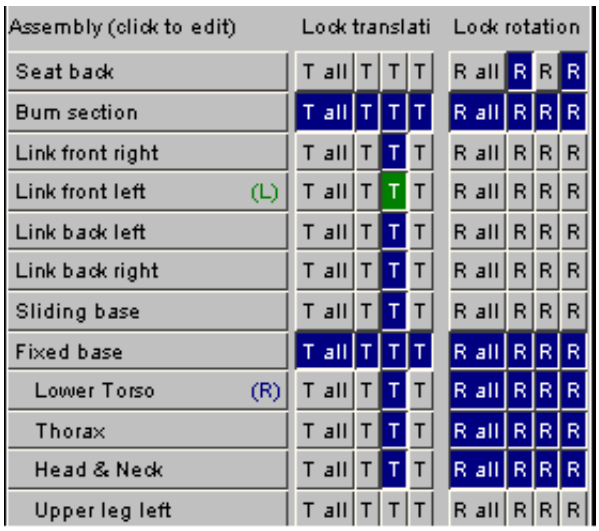
In **Drag Assemblies** mode the positioning panel changes.

Each assembly is still shown as a row, but now:

- Clicking on the "name" button brings up the assembly editing panel as above.
- You can select the degrees of freedom to be restrained (locked) during positioning for each assembly. Restraint acts in the coordinate system of the assembly (if defined), otherwise in the global system.

Restraints shown in blue are in the global system, those in green (here "Link front left") are in the local system of the assembly.

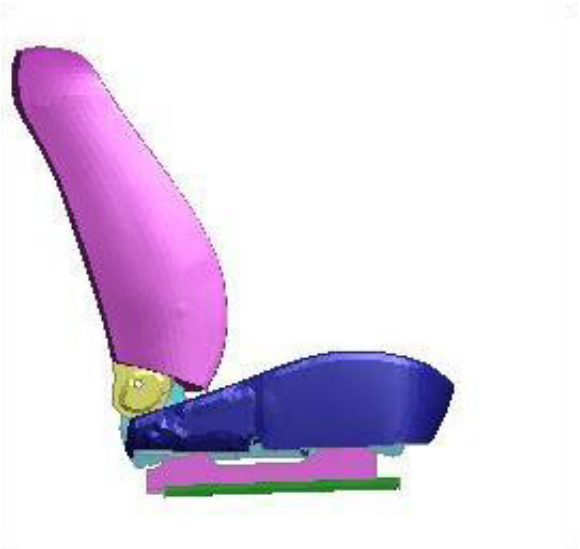
Restraints can be set and unset at any time during positioning.



An example of **Drag Assemblies** free dragging.

The following sequence of images shows how this might be used in practice. In this example the dummy has been positioned in the seat, with hands attached to the steering wheel and feet to the pedals. Both hands and feet are fully restrained in all degrees of freedom, the torso, thorax and head are restrained against all rotations and also Y (out of plane) translation.

The user has clicked on the lower torso with the left mouse button, so the whole dummy is selected for movement, and drags it progressively further forwards. This sequence would be carried out in a single operation, and for this dummy the drag occurs in near real-time on a modern desktop computer.



Initial condition.

The user is about to click on the seat cushion (blue) and move it up and forward

In progress.

The seat has to rise upwards on its links in order to move forwards

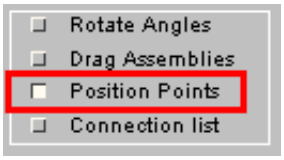
Further progress.

The seat has made progress forwards, rotating on its fore and aft links.

Final position.

The seat has come back down again to achieve its final forwards position.

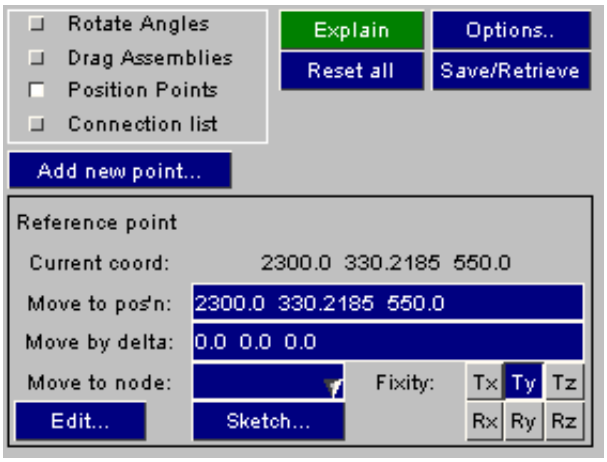
Move Points: movement driven by updated point positions.



An alternative to dragging with the mouse is to set new target positions for points. As described above any number of points can be defined in an assembly, and used both to apply localised restraint and to drive movement.

In this example a point has been created in the seat base, acting as a reference point for movement.

It can be moved by any combination of the following three methods:



Move to pos'n Will move the point ***to*** the new coordinate specified

Move by delta Will move the point ***by*** the specified distance [dx.dy,dz]

Move to node Will move the point ***to*** the position of the chosen node.

In all cases the effect is similar to dragging with a mouse, with the difference that PRIMER will drive the iterative scheme for you to try to achieve the new position.

Iteration will continue either until the target point is reached, or the changes between successive iterations become insignificantly small. The latter is necessary since, obviously, it is possible to set a target position for a point that cannot be achieved because of restraints.

Using wild-card coordinates for movement of positions.

It is sometimes the case that you want to move a point a certain distance along one axis, but not to constrain its movement along other axes. To allow for this PRIMER permits the following "wild-card" as opposed to"explicit" coordinate entry syntax.

Values entered explicitly as zero mean exactly that. Therefore

[*Move by delta*] 100.0 0.0 0.0 Means "move by 100 in X, but try not to have any movement in Y or Z."

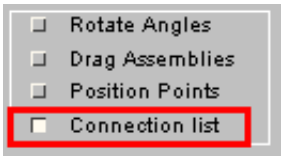
Omitted trailing values, or values entered as an asterisk "*" are taken to mean "not constrained". Therefore

[*Move by delta*] 100.0 Means "move by 100 in X, but don't care about movement in Y or Z"

Means "move by 100 in Z, but don't care about movement in X or Y"

The same syntax may be used for absolute [*Move to pos'n*] coordinate entry.

Connection list: editing, locking and moving connections



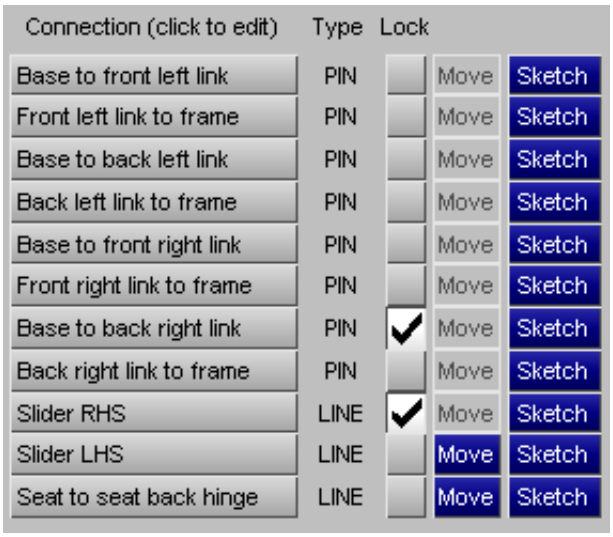
This panel lists all the connections in the mechanism (but not in any child dummies),

Connections may be edited by clicking on their name button, and also sketched.

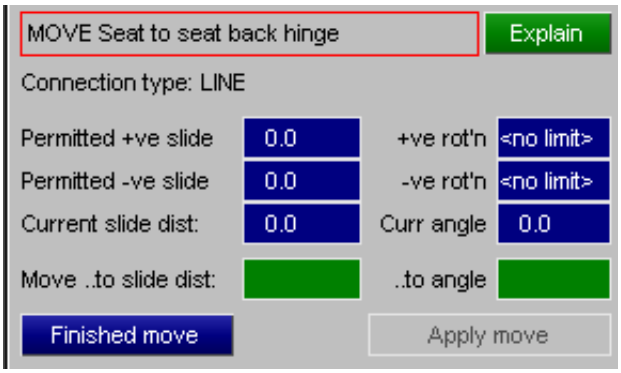
More usefully they can also be locked and unlocked: in this example the pin connection between "base and back right link" has been locked, as has the line connection along the "slider RHS". (Un)locking can take place at any time so, for example, two assemblies can be moved relative to one another and then locked in that configuration for subsequent positioning.

"Locking" makes the connection totally rigid in all 6 degrees of freedom, with the effect that the two assemblies on either side become merged into a single rigid combination.

You may also "move" a line or hinge connection, which means driving it to some new orientation.



Move: forcing a connection to adopt a new orientation.



Another way of positioning a mechanism is to "drive" a line or hinge connection to a new orientation by specifying a new slide distance or rotation angle.

The "current" slide distance or angle are really just arbitrary values that are used to limit motion against prescribed limits, and can be reset to new values at any time. Changing these values does ***not*** move the mechanism, it is more like moving the ruler or protractor used to measure its orientation.

To move the mechanism you need to specify a new "**`Move to ...`**" slide distance or angle in the green boxes shown here, and then to **Apply** the motion. This will cause the joint to slide or rotate as required in an attempt to achieve the new position or angle, driving the motion of the mechanism.

Moving a connection is very similar to the dragging and rotation of assemblies described above, the difference being that you are applying motion across a connection rather than to the assemblies to which it is connected. Internally the method is nearly identical: the attached assemblies have translations or rotations applied to them calculated to achieve the desired joint orientation.

Further positioning commands

The following commands are common to all four positioning methods described above.

Accept	Accept position and save changes.
Reject	Abandon positioning, and restore initial position
Reset all	Restores the mechanism to its initial position
Global Accuracy	Sets the accuracy of the mechanism calculation
Options...	Further positioning options
Save/Retrieve	Save and retrieve positions

Accept: accepts the current position and saves the changes you have made.

Once you are happy with the current position use **Accept** to save it and finish positioning.

Before it saves the position PRIMER checks all the nodes at (ls-dyna) joints in the mechanism to check that positioning has not pulled them apart. It applies a twin tolerance:

- An absolute value of 1.0e-3. This is the value hard-wired into the LS-DYNA keyword reader.
- A distance of 1.0e-6 times the model longest diagonal

Any joints at which separation of nodal pairs exceeds this figure will be listed and you will be given the option of **Autofixing** them.

This is performed by moving each pair of nodes to their average position and, so long as the errors are small, this is an acceptable distortion of the model. This is an iterative process since if a node is on more than one joint then correcting for joint A may move it out of position for joint B. If there are still errors after 5 passes the operation is abandoned and it is left to the user to sort out.

Warning: **Avoid repeated [Position, Accept, Autofix] cycles on a model.**

This is because each **Autofix** operation changes the geometry of your mechanism slightly, and while a single such change may be insignificant repeated use of this feature will build up cumulative errors. In particular repeated **Autofixing** of joints, which moves pairs of nodes to their average coordinate, may introduce alignment errors.

It is better to achieve a position in a single operation from an unmodified mechanism model. If you are planning to generate a series of positions in succession you should use **Model, Copy** to create a new model from an original one each time, and create the position in the copy.

Reject: rejects the current position, restores the initial one and exits the positioner.

Use **Reject** if you want to abandon positioning and restore the initial position. All changes made during positioning will be lost, and you will return to the main **Position** menu with the model unchanged.

Reset all: restores the intial position.

Sometimes positioning goes horribly wrong and the best thing is to start again. **Reset all** restores the initial position that was saved when you entered the positioner, cancelling all changes made since then. You can use this at any time.

Global Accuracy: sets the precision of the mechanism calculation

By default the precision with which the mechanism positioning process calculation is performed is based on the diagonal of the box bounding the mechanism times a "convergence factor" of 1.0e-4. Therefore a mechanism fitting into a box with a diagonal of one metre will be solved to a precision of approximately 0.1mm, this being the maximum permitted error at mechanism connection points.

This error manifests itself as small differences in the coordinates of theoretically coincident connection points on assemblies A and B that are joined by a mechanism connection. As part of "accepting" a mechanism PRIMER checks for LS-DYNA joints (**`*CONSTRAINED_JOINT`**) at such points and moves points A and B to an identical average position so as not to exceed the joint coincidence tolerance permitted by LS-DYNA. Points not connected by an LS-DYNA joint are left in their slightly different positions.

Therefore the result of mechanism positioning is a series of small distortions of your model, and possibly also small changes to joint geometry. You must make an engineering decision about what constitutes an acceptable tolerance for your particular model, and you may wish to use **Global accuracy** to make mechanism positioning more precise. It will not usually be sensible to make it less precise than the default value of 1.0, although doing so will make dragging faster - possibly an acceptable setting for performing an interactive demonstration in front of an impatient audience!

Detailed convergence parameters can be set in Options, described below, but these are rather opaque and a simpler value to use is the **Global accuracy** parameter which can be in the range 0.1 to 100.0, default 1.0.

The effect of this factor is to divide the default value of the following three convergence parameters thus:

```
Convergence factor   =  1.0e-4 / Global Accuracy

End movement factor  =  1.0e-8 / Global Accuracy

Step size factor     =  1.0e-4 / Global Accuracy
```

You can still set these individually in Options below, this is just a simpler and easier way of doing that.

Global Accuracy is a floating point number that may have any value in the range 0.1 to 100.0, and typical values are:

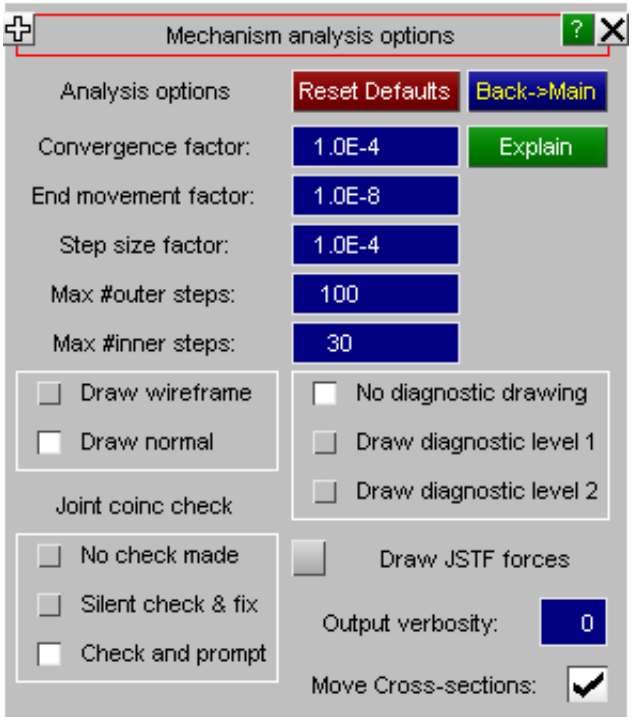
Global Accuracy typical values and their meanings.	
0.1	Very loose tolerance, ***not recommended***
1.0	**Default value**, suitable for dragging with the mouse
10.0	Tighter value, practical limit for dragging with the mouse
20.0	Tighter still, suitable for "specified" motion
50.0	Very tight, practical limit for "specified" motion
100.0	Extremely tight: will probably lock up.

You can change the value at any time, and some experimenting may be required with very complex mechanisms to find a value that gives a reasonably compromise between precision and speed of positioning.

The default value may be changed by the preference:

```
primer*mechanism_accuracy: value
```

Options... Setting positioning options.

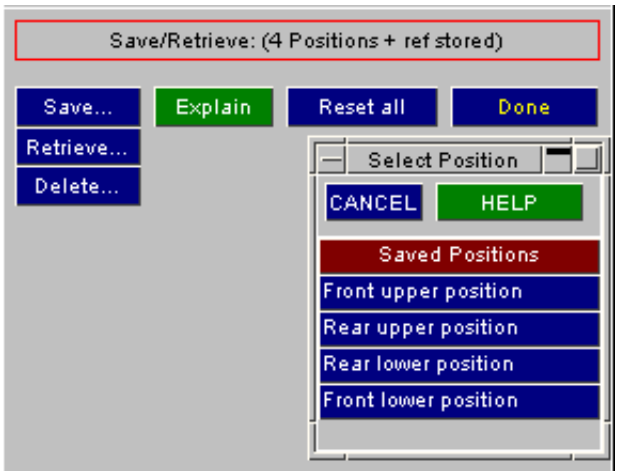


The following options affect the positioner:

Convergence factor End movement factor Step size factor Max #outer steps Max #inner steps	These options all affect the free dragging positioner only. They should not normally need to be changed, but if the mechanism moves very slowly, or "gets stuck", then increasing the Convergence and Step size factors may help. Avoid much larger values as the solution will become inaccurate.
Draw wireframe Draw normal	Sets the graphics mode to be used when dragging assemblies. "Draw normal" shows the assembly in grey using normal graphics, but this demands quite a lot of cpu time and only slower computers "Draw wireframe" may be necessary to get acceptable dragging speed.
Joint coinc check	Is the post **Accept** joint node coincidence check. By default it will check and report any results, asking you what action to take. You can it work silently, which will fix any errors without further input, or turn it off altogether.
Move Cross-sections	Whether or not **`*DATABASE_CROSS_SECTION_PLANE`** definitions that have parts wholly or partially in a single assembly are moved with that assembly.

Diagnostic drawing, output verbosity and JSTF force display are for programmer debugging purposes and - hopefully - can be ignored!

Save/Retrieve: Saving and restoring mechanism positions.



You can store any number of mechanism positions, and retrieve them at any time into the positioner.

A stored position contains a [centre of gravity] plus [3x3 direction cosines] for each assembly in the mechanism, making it possible to return to a given configuration without any further calculation.

Position data is stored with the mechanism definition and is saved when the keyword file is written out.

This information can also be saved into a seperate Mechanism Configuration File which can be read back into PRIMER.

Save... Saves the current position. You only need to give a unique name.

Retrieve.. Restores a saved position. This becomes the current position and the dummy geometry is updated immediately causing it to "jump" to the new position.

Delete... Deletes the selected positions. Deletion is permanent!

Where a mechanism has "child" mechanisms or dummies positions are saved and retrieved for these as well. Since position selection is by name retrieving a position which does not exist in a child will leave the child unmoved.

A more detailed explanation of saved positions, including card formats, is given in Appendix IIc.

6.30.4 Batch (command line) positioning

A subset of the interactive positioning commands described above are also available in command-line line form. While these can be used interactively the main purpose of them is to enable positioning to be performed in batch mode. These commands will provide visual feedback if the graphical user interface is running, but if it is not (PRIMER started with "-d=batch" command line option) they will still function. A full listing of command-line commands is given in Appendix X11.

The positioning commands are invoked by the [Primer >] **MECHANISM** command, and occupy a hierarchy as follows:

At **MECHANISM>** level		
ASSEMBLY	Select an assembly by name or number, then perform one of the following operations upon it:	**FIX** *dof code* — Restrain the assembly in degrees of freedom *dof code* **TRANSLATE** *dx, dy, dz* — Translate assembly ***by*** amount *dx,dy,dz* **RX** or **RY** or **RZ** *theta* — Rotate assembly ***to*** angle *theta* degrees about x/y/z **RESET** — Undo all dummy transformations and return to initial state **DONE** — Finish with assembly and return to **MECHANISM >** prompt **CONTACT** — **OFF** or **ON**. Turns assembly contact (if defined) on/off during positioning.
POINT	Select a point by name or number, then perform one of the following operations upon it: (Note: moving the point implicitly moves its "owner" assembly.)	**FIX** *dof code* — Restrain the point in degrees of freedom *dof code* **TRANSLATE** *dx, dy, dz* — Translate point assembly ***by*** amount *dx,dy,dz* **POSITION** *x, y, z* — Translate point assembly ***to*** coord *x, y, z* **RESET** — Undo all dummy transformations and return to initial state **DONE** — Finish with point and return to **MECHANISM >** prompt
CONNECTION	Select a connection by name or number	**SLIDE** *distance* — Applies to **LINE** connections only, and will slide the joint by *distance* down its AB axis. **ANGLE** *theta* — Applies to **LINE** and **HINGE** connections only, and rotations the assemblies to achieve angle *theta* (in degrees) about the AB axis.
POSITION	Specify a position *name* or *id*	Retrieves and applies the stored position *name* or *id*
SAVE	Specify a position id and (optional) *name*	Saves the current configuration as a saved position id, with optional *name*.
READ_CONFIG	Specify a *filename*	Retrieves a free-standing dummy configuration file (the keywords and data between ***MECHANISM_START** and ***MECHANISM_END**). *Filename* will usually have the extension .mcf

ACCURACY	Specify a *value*	Global factor on the accuracy of the mechanism positioning process. Value must lie in the range 0.1 to 100.0
ACCEPT	Accept the current mechanism position, save its updated geometry and return to the main [Primer >] prompt.	
RESET	Undo all transformations and restore the initial geometry of the mechanism, remaining at this prompt level.	
QUIT	Undo all transformations and restore the initial geometry of the mechanism, then return to the main [Primer >] prompt.	

Meanings of terms in the table above

dof code Is a numeric Degree of Freedom code made up of any permutation of 123456, where

1 = Tx, 2 = Ty, 3 = Tz, 4 = Rx, 5 = Ry, 6 = Rz

For example code **136** means restraint in Tx, Tz, Rz

Code **0** may also be used, meaning "free all restraints"

dx, dy, dz Is a translation vector, ie a relative movement from the current position, made up of three numbers.

For example **10.0 20.0 30.0** means translate 10.0 in X, 20.0 in Y, 30.0 in Z.

"Wildcard" syntax is permitted: any number entered as an asterisk ("*"), and omitted trailing digits, are treated as "free" values. For example:

```
    10.0      means translate 10.0 in X, but permit Y and Z to adopt any value.
* * 20.0        means translate 20.0 in Z, but permit X and Y to adopt any value
```

x, y, z Is an absolute coordinate.

For example **10.0 20.0 30.0** means coordinate X=10, Y=20, Z=30.

Wildcards as for translations above are permitted

theta Is an angle in degrees for the given degree of freedom.

In a dummy model angles are absolute values expressed in the coordinate system of the connection between this assembly and its parent. In most cases this will mean the system implied by the local axes of the joint stiffness definition at the joint.

6.30.5 Using mechanisms and dummies as "children" of mechanisms

Both mechanisms and dummies may be defined as "children" of a mechanism.

A "child" is a separate Dummy or Mechanism in which one or more assemblies are linked to an assembly in the master in any combination of Tx, Ty and Tz degrees of freedom. When the master mechanism is positioned then its motion also drives the motion of the linked assemblies in the child causing it to be positioned too.

Force feedback from child to master takes place, so the master will feel resistance if it tries to push the child where it doesn’t want to go. However this is a one-way treatment: while child assemblies can be moved within their own child mechanism this will not transmit force to their master, so dragging child assemblies will not move the master mechanism.

Child ***mechanisms*** may be nested to any depth: a child mechanism may itself have children.

Child ***dummies*** may not be nested: dummies cannot have children.

The principal use of this capability is to position a dummy on a seat mechanism, and this is described in section 6.14.4: Using dummies as "children" of mechanisms.

The process of defining and linking together master and child is described in CHILD mechanisms above.

6.30.6 Mechanisms and *INCLUDE_TRANSFORM

It is sometimes useful to place mechanism assemblies inside include files which use **`*INCLUDE_TRANSFORM`** plus **`*DEFINE_TRANSFORMATION`** to define their orientation and position.
PRIMER detects this automatically, and when such mechanisms are positioned an extra 4 lines are added to the relevant **`*DEFINE_TRANSFORMATION`** keywords which are equivalent to the change in each assembly's position.

Lines added by PRIMER to **`*DEFINE_TRANSFORMATION`**
`TRANSL dx dy dz` **`ROTATE 1 0 0 cx cy cz tx`** **`ROTATE 0 1 0 cx cy cz ty`** **`ROTATE 0 0 1 cx cy cz tz`**

<cx cy cz> is the centre of rotation, tx/y/z the angle in degrees.

Note that all four lines are always written, even when a given transformation is in fact zero.

This is to enable PRIMER to "know" what is there and overwrite it if the assembly is re-positioned during a session, thus avoiding a build-up of many lines.

A consequence of this is that you must not mix mechanism positioning and manual editing of these **`*DEFINE_TRANSFORMATION`** definitions within a single session, otherwise you may confuse the logic which updates these lines.

The effect on output from PRIMER is that such assemblies will revert to their untransformed position, and on subsequent reading into LS-DYNA (or back into PRIMER) will revert to their "as positioned" state when the transforms are applied.

Within a single PRIMER session these transformations are overwritten if the assembly is repositioned, so that a session will only ever add an extra 4 lines. However if you exit and restart PRIMER (or reread or copy the model) then any subsequent positioning will create a new block of 4 lines.

Therefore if a mechanism is to be generated in several different positions these should be created within a single PRIMER session or, which would be better because it would also reduce the build-up of small residual errors, by starting each time from the original (untransformed) file.

Rules when using *INCLUDE_TRANSFORM with mechanisms:

- PRIMER only peforms these modifications if an assembly is in one or more **`*INCLUDE_TRANSFORM`** files ***and*** these each refer to an existing **`*DEFINE_TRANSFORMATION`** definition.
- PRIMER assumes that a given **`*DEFINE_TRANSFORMATION`** definition is used ***only*** by the **`*INCLUDE_TRANSFORM`** file(s) that contains this assembly. (But see the change for 9.3.1 described below.)
- PRIMER permits an assembly to be spread over several **`*INCLUDE_TRANSFORM`** files (although this is not recommended) and each such file must either refer to the same **`*DEFINE_TRANSFORMATION`** definition, or to separate definitions that are used only by these files.
- PRIMER will create a new block of the four **`*DEFINE_TRANSFORMATION`** lines described above when an assembly in such an include file is first positioned, and it assumes that it "owns" these last four definitions for the duration of the session. If you edit these cards manually in a session in which you have performed mechanism positioning the results may go wrong.

Recommended modelling practice when using *INCLUDE_TRANSFORM with mechanisms:

It is ***strongly*** recommended that if **`*INCLUDE_TRANSFORM`** files positioned by **`*DEFINE_TRANSFORMATION`**s are to be used with mechanisms (and dummies) then the following practice be adopted:

1. **Use a "one include file per assembly" policy.**

   Put the nodes, parts, elements, sets etc that make up an assembly into a single file, do not split them over multiple include files.

2. **Also use a "one assembly per include file" policy.**

Do not put multiple assemblies in an include file, as a single **`*DEFINE_TRANSFORMATION`** cannot then be applied to them.

3. **Do not mix "assembly" and "non-assembly" data in include files.**

   This will avoid transformations being applied to "non-assembly" structure that should not be moved.

4. **Use a unique transformation for each such include file.**

   Create a single **`*DEFINE_TRANSFORMATION`** for each such include file, and use it only for that file. This will avoid possible conflicts.

If this practice is adopted then it is highly unlikely that things will get confused.

Rule (1) above is not strictly necessary but it is good practice, and it should avoid problems arising from small dimensional errors where part of an assembly is moved en-bloc by a **`*DEFINE_TRANSFORMATION`** and part by explicit updates of nodal coordinates. Remember that joints in particular are sensitive to even very small initial misalignments!

Change of behaviour in release 9.3.1 to handle multiple assemblies in *INCLUDE_TRANSFORM file

In the original release 9.3 software it was assumed that users would adhere to the rule that "**`*INCLUDE_TRANSFORM`** could only apply to a single assembly" as described above.

However experience showed that some users would import a complete mechanism or dummy (containing multiple assemblies) in a single include file subject to a single transform. This caused problems because every assembly added its own 4 lines of transforms, leading to a ridiculous result.

Therefore from release 9.3.1 onwards a check has been added as follows:

Each assembly is checked for membership of **`*INCLUDE_TRANSFORM`**s and the extra positioning lines are only added to the associated **`*DEFINE_TRANSFORMATION`** card if the following criteria are satisfied:

1. The **`*INCLUDE_TRANSFORM`** file ***must*** contain ***only*** items in that assembly. It may contain the whole assembly, or only a subset of it, but it may not refer to anything else in a different assembly or to unrelated structure not in the mechanism/dummy definition.

2. An assembly may be made up of components from multiple include files (although this would be unusual), and transformations will only be applied to those include files which satisfy rule #1 above.

This means that - for example - a dummy may be imported in an **`*INCLUDE_TRANSFORM`** file and located in the model using a **`*DEFINE_TRANSFORMATION`** associated with this. Any subsequent mechanism or dummy positioning will work normally, but extra transformation lines will no longer be added to the **`*DEFINE_TRANSFORMATION`**, meaning that the nodal coordinates will be updated when that include file is written out.

Extension of the above logic in release 10.0

It turned out that the check added above, which was based only on which nodes were in which include files, was defeated by a case where a modeller placed nodes for all assemblies in include file A, then parts and elements for all assemblies in include file B, and subjected both files to the same transform. Since the test above assumed implicitly that nodes and elements would be in the same include file PRIMER wrongly created multiple transformations for the second file B.

Therefore the check for multiple assemblies in an include file has been extended to look at parts, elements and nodes. This is still bad practice, and the modelling practice described above is recommended for all cases where mechanism and dummy assemblies are placed in include files subject to transformations.

6.30.7 Editing and Moving Mechanisms (and Dummies) outside PRIMER

It is best to move mechanisms and dummies bodily (meaning as a whole) using the Orient function in PRIMER since this will update not only nodal coordinates and other part-related data, but also information in any saved *POSITION data.

However it is recognised that some workflow practices will require sub-models containing mechanisms and dummies to be moved outside PRIMER, and the resulting merged deck then taken into PRIMER for detailed positioning. This will work so long as the following rules are obeyed:

1. Don’t try to "position" individual assemblies within a dummy or mechanism outside PRIMER. It is unlikely that you will update all the required geometry and *POSITION data, and this may lead to internal inconsistencies when the model is re-imported back into PRIMER. Positioning by dragging assemblies will probably still work after such changes, but retrieval of any saved *POSITION information is likely to give the wrong result.

2. Minor editing of assembly contents, for example remeshing of a part or addition of nodes and elements, is acceptable so long as any nodes used for mechanism connections or points are not disturbed. Significant edits should be avoided, or if they are necessary then you will probably need to remake the assembly in question once back inside PRIMER

3. Translating and/or rotating the mechanism or dummy as a single quasi-rigid block, ie keeping the relative positions and orientations of assemblies the same, is OK. Scaling or "morphing" may result in any saved *POSITION information giving the wrong result.

   However if you do this make sure that all definitions of connections and points within assemblies use nodes rather than explicit coordinates, as external software almost certainly will not know how to change some of the more obscure data fields on *MECHANISM and *DUMMY cards.

Perhaps surprisingly translating, rotating or reflecting a mechanism or dummy as a whole does not have any effect on stored *POSITION data or subsequent positioning operations. *POSITION data creates its own space system, and so long as all assemblies and stored positions use the same space system it doesn’t matter where these are located in space or how they are oriented. This is because retrieving a saved position effectively extracts only the translations and rotations to get from orientation A to orientation B, and these transformations can be applied successfully to any object located anywhere.

However the key words in the first sentence are "***as a whole***". If you have moved individual assemblies relative to one another your saved *POSITION data may become invalid. If you have done this accidentally, which may become obvious when retrieval of saved positions gives the wrong answer, then you should delete all the *POSITION data inside the *MECHANISM or *DUMMY blocks in the keyword file in order to force PRIMER to recreate this information.

6.31 MESHING

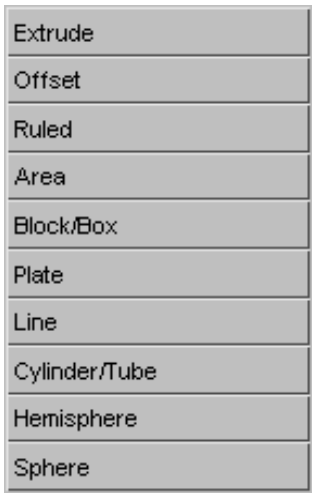
PRIMER has some simple meshing facilities. The options available under the Meshing tool are:

Mesh	Simple meshing facilities such as extrude, offset, ruled surfaces, plates etc
Split	Split shells
Create hole	Create a hole in an existing mesh
Remove hole	Remove a hole from an existing mesh
Remesh area	Remesh an area of existing mesh
Cobweb mesh	Create a cobweb mesh on an existing mesh
Morph	Morph a mesh
Tet mesh	Create a tetrahedron mesh inside a closed volume of shells
Geometry	Mesh geometry surfaces

6.31.1 Simple meshing operations

Primer has simple meshing capabilities. Currently you can:

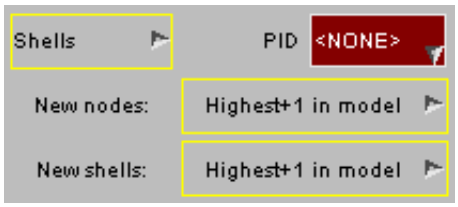
- Extrude nodes to create beams or shells, or extrude shells to create solids or thick shells.
- Offset shells to create shells or solids.
- Create a ruled mesh between two lines of nodes.
- Mesh a quadrilateral or triangular area by giving the corner nodes.
- Create a cuboidal solid block.
- Create a rectangular shell plate.
- Create a line of beams between 2 nodes.
- Create a cylinder/tube made of shells.
- Create a hemisphere made of shells.
- Create a sphere made of shells.



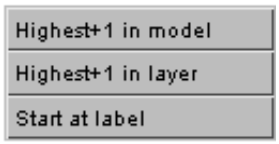
To change the mode use the popup at the top left of the meshing panel.

For all of the meshing modes you must give a part ID for the new beams/shells/solids/thick shells to be created in.

Additionally you have control of the numbering that is used for new nodes and new beams/shells/solids/thick shells.

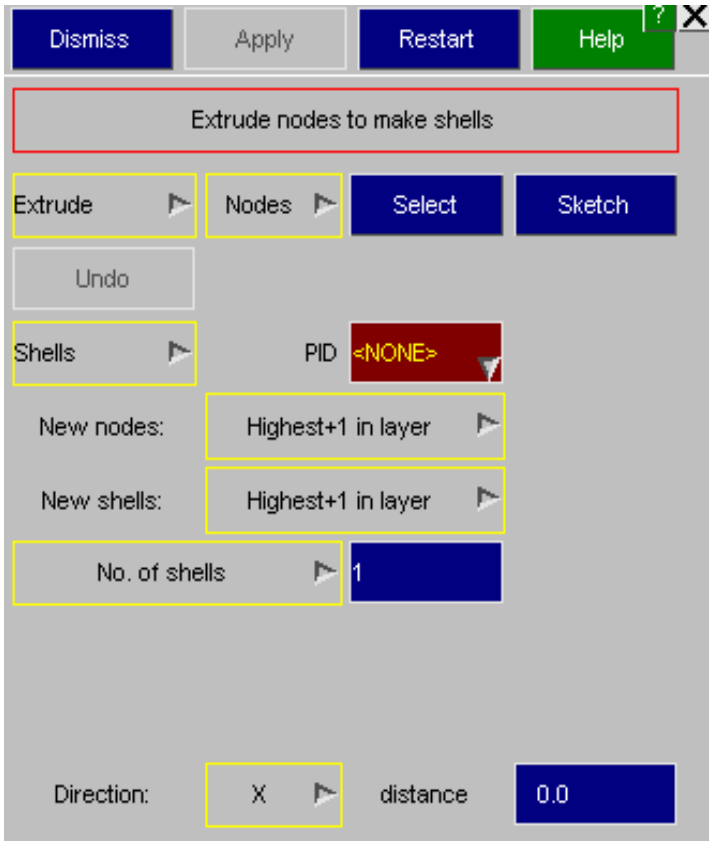


Use the popup to select which option you require.



If you choose **Start at label** then give a label number to start from. Primer will try to use that number. If a node or beam/shell/solid already exists with that label it will revert to **Highest+1 in model**.

Extrude



Extrude allows you to:

- Extrude nodes to create beams.
- Extrude nodes to create shells.
- Extrude shells to create solids or thick shells.

The popups allow to choose the mode.

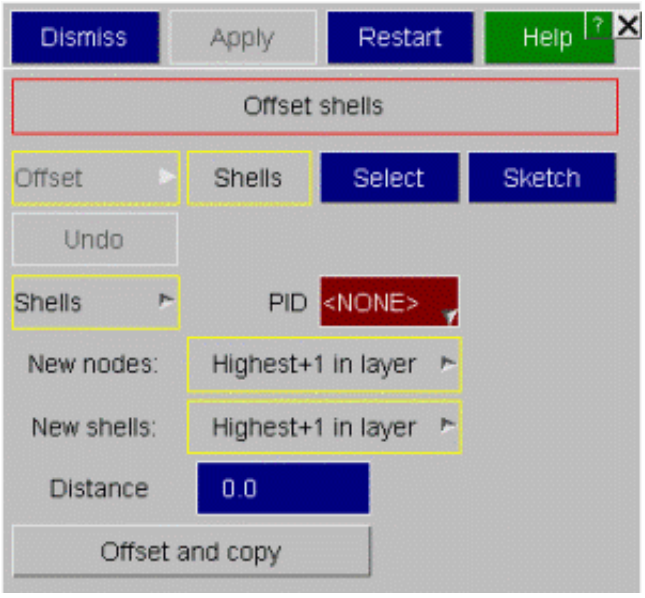
The number of beams, shells, solids or thick shells to create in the extrude direction can either be given or you can give the element size, in which case Primer will determine the number required.

The extrude direction can be given by:

- The global X, Y, or Z axes. Give the distance
- A vector given by 2 nodes. Either give a distance or use the length N2-N1.
- A vector given by X, Y and Z components. Either give the distance or use the length of the vector.
- In case of extruding shells to Solid/Thick shells, we can also extrude in shell normal direction.

When extruding nodes to create shells, if the last node chosen is the same as the first node then the shells created will wrap round.

Offset



Offset allows you to:

- Move shells or create shells offset a specific distance from existing shells.
- Create solids by offsetting existing shells.

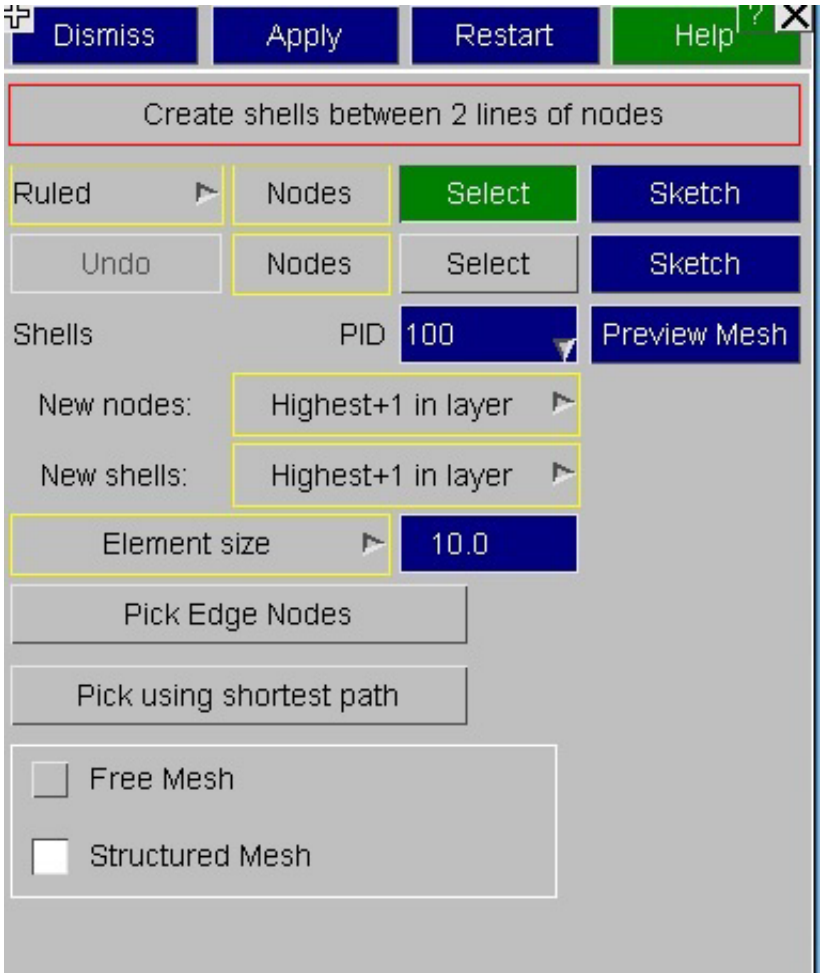
For both options ensure that the normals of the shells you want to offset are consistent, otherwise different directions could be used.

For shells give the distance that the new shell should be offset from the existing shells.

For solids give how many solids should be created while offsetting and the distance to offset.

For shells, to move the original shells rather than create new ones toggle the **Offset and Copy** button to **Offset - no copy**.

Ruled



Ruled allows you to create a mesh of shells between 2 lines of nodes.

You can toggle the line of nodes that you are picking nodes for by pressing the **Select** button.

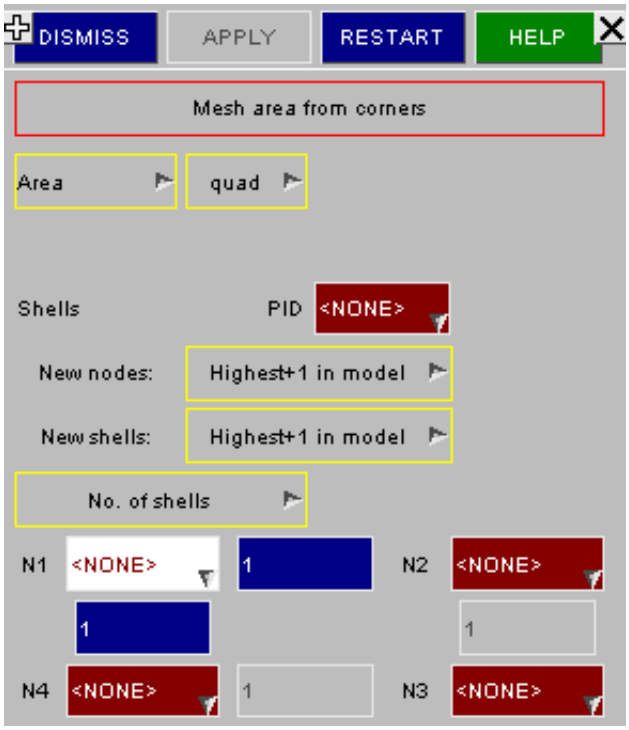
When the number of nodes on both ends is not equal - two options are present a) Free meshing b) Structured Meshing.

You can pick nodes by selecting "Normal screen pick" or "Pick Edge Nodes" or "Pick using shortest path" methods.

Give either the number of rows of shells to create or the size of shell element size, in which case Primer will determine the number of rows of shells required.

The preview of the to be created mesh can be seen with the **Preview Mesh** button.

Area



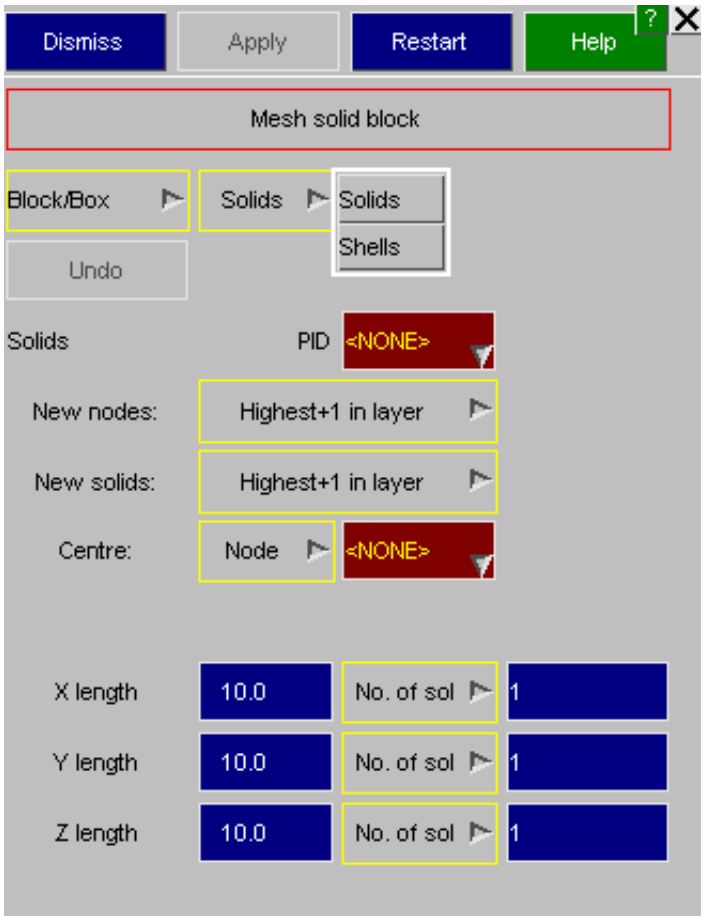
Area allows you to mesh a quadrilateral or triangular area. Use the popup to choose which type to mesh.

Pick the 3/4 corner nodes for the mesh. Currently, for a quadrilateral mesh the number of shells between N1and N2 must be the same as between N3 and N4, and the number of shells between N1and N4 must be the same as between N2 and N3. For a triangular mesh the same number must be used for all sides.

Give either the number of shells to create or the size of shells to create, in which case Primer will determine the number required.

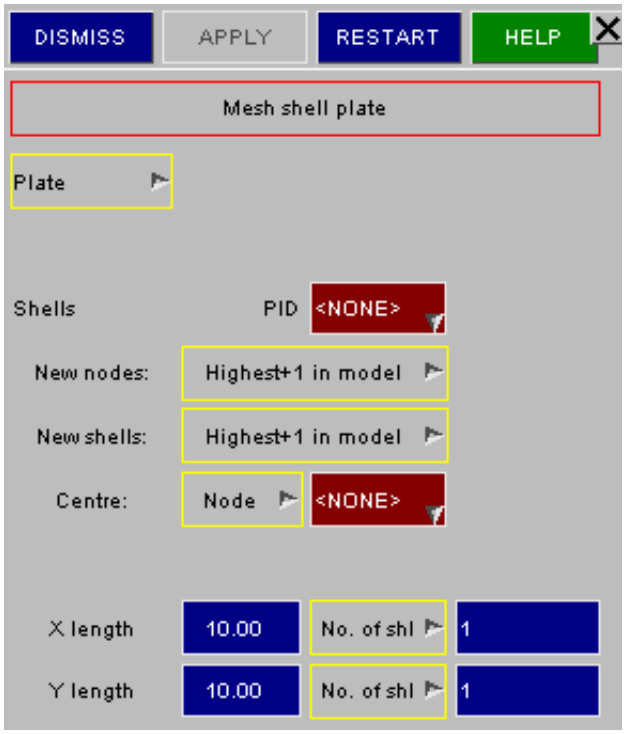
Block/Box

Block/Box allows you to create a simple block of solid elements or an empty box of shell elements. Give a node or coordinate for the centre of the block/box, then for the X, Y and Z directions give the length of the block/box and either the number of solids/shells or the element size.

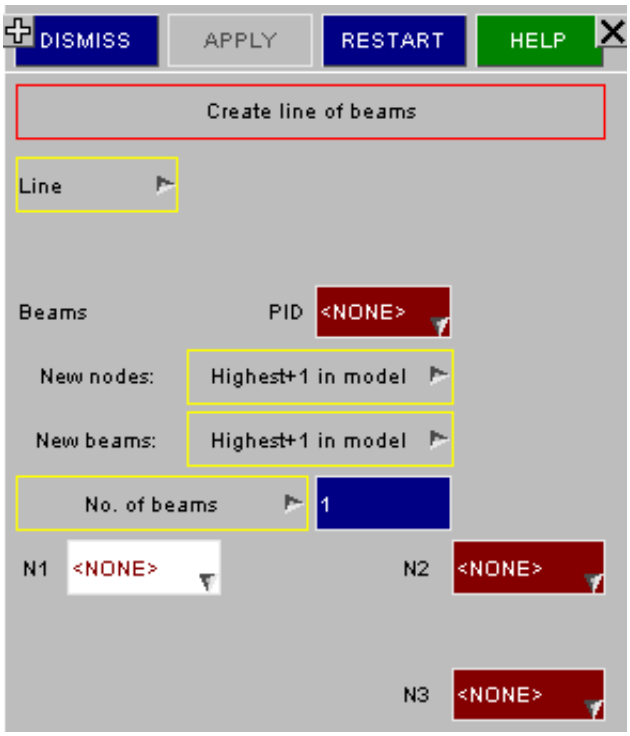


Plate

Plate allows you to create a simple plate of shell elements in the xy plane. Give a node or coordinate for the centre of the plane, then for the X and Y directions give the length of the plate and either the number of shells or the shell size.



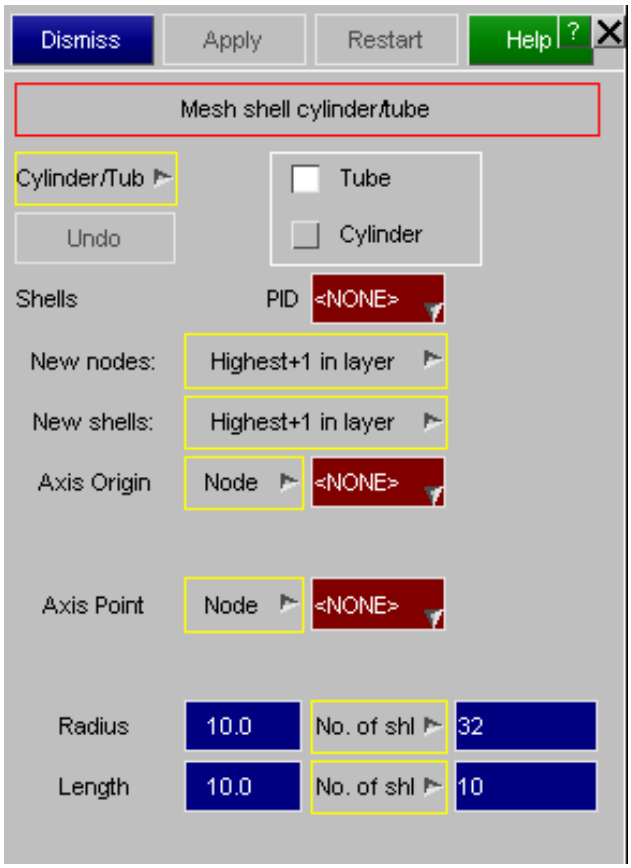
Line



Line allows you to create a line of beams between N1 and N2. N3 must also be given and will be used as the 3rd node for all of the beams that are created.

Either give the number of beams to create or the size of a beam, in which case Primer will determine how many to create.

Cylinder/Tube



Cylinder/Tube allows you to create a cylinder/tube mesh of shells.

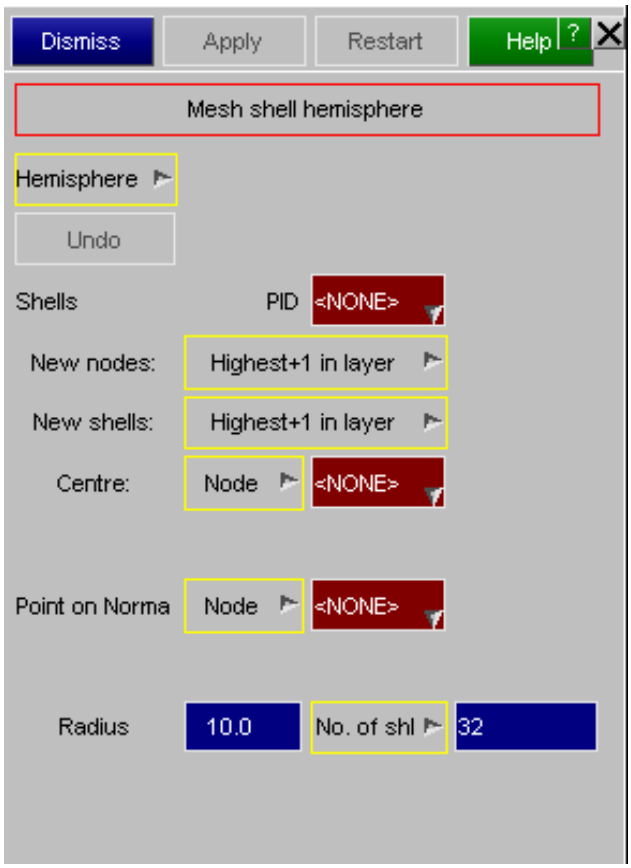
The axis of the cylinder/tube is defined by Axis origin and Axis point, which can be specified by a NODE id or by entering the coordinates.

After setting the Axis Origin and Axis Points, give radius, height and either the number of shells or the size of shells to create in radial and axial directions and hit **Apply** to create.

Hemisphere

Hemisphere allows you to create a hemispherical mesh of shells.

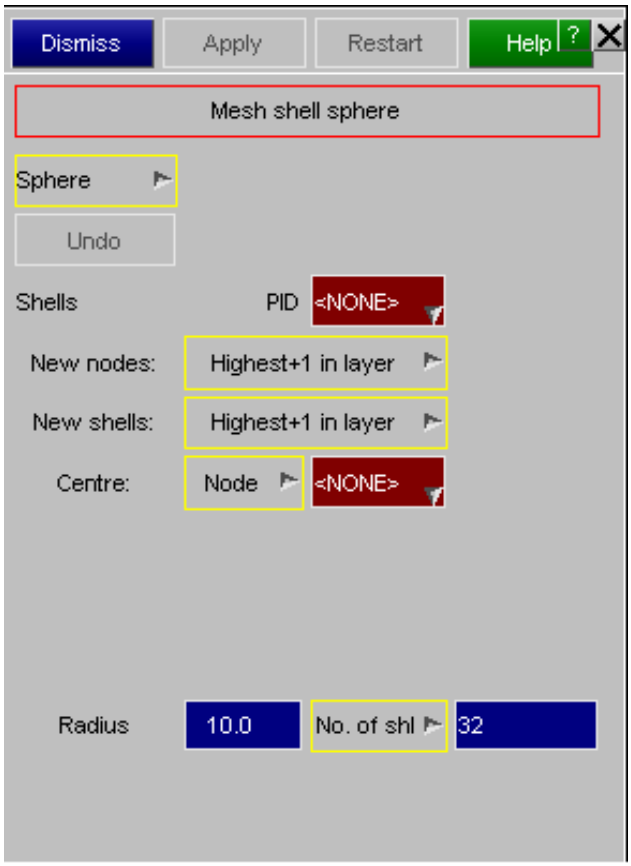
After defining the Centre and a Point on normal, give radius and either the number of shells around circumference or the size of shells and hit **Apply** to create.



Sphere

Sphere allows you to create a spherical mesh of shells.

After defining the Centre, give radius and either the number of shells around circumference or the size of shells and hit **Apply** to create.



6.31.2 Swage Mesh

The swage mesh allows you to easily create swages/beads in shell meshes by selecting points on a Shell or a Shell Node.

The swage panel shows the section view of the intended swage mesh and explans the basic parameters required to create the swage mesh.

You can also adjust any of the parameters if required. When you are happy with the properties pressing **Sketch** will see a preview of the mesh that PRIMER will create.
Outline will only show the outline of the proposed mesh.
Auto Update is a toggle button that automatically updates the sketch of the swage mesh whenever any of the swage mesh parameters or the options values are changed.
To actually create the swage and remesh press **Apply**.

Pick pts. is the toggle button to switch ON/OFF the points selection to define the path of the swage on the shell mesh.

Auto PID automatically takes the PART ID of the new swage mesh from the shell element on which the first swage path point is selected.

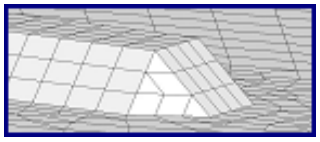
The swage mesh is created along the NORMAL direction of the shell elements on which the swage path points are selected.
Reverse button reverses the direction of the swage.

Element Pitch is the element size along the length of the swage path.

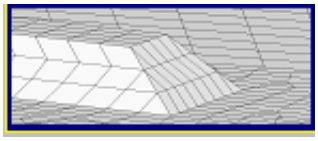
Auto El. Size button automatically sets the mesh element sizes for the swage mesh as the "average" element size of the original shell mesh.

The **SIDE ends** of the swage mesh can be three different types of shapes:

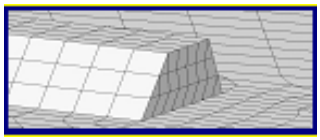
CHAMFERED



SLANTED



VERTICAL



Also the **SIDE** surfaces for the swage mesh can be created with these two methods:
- FREE: Creates a free shell mesh using average element sizes specified for the side edges.
- STRUCTURED: Creates a regular shell mesh with well defined edges along the side surfaces.

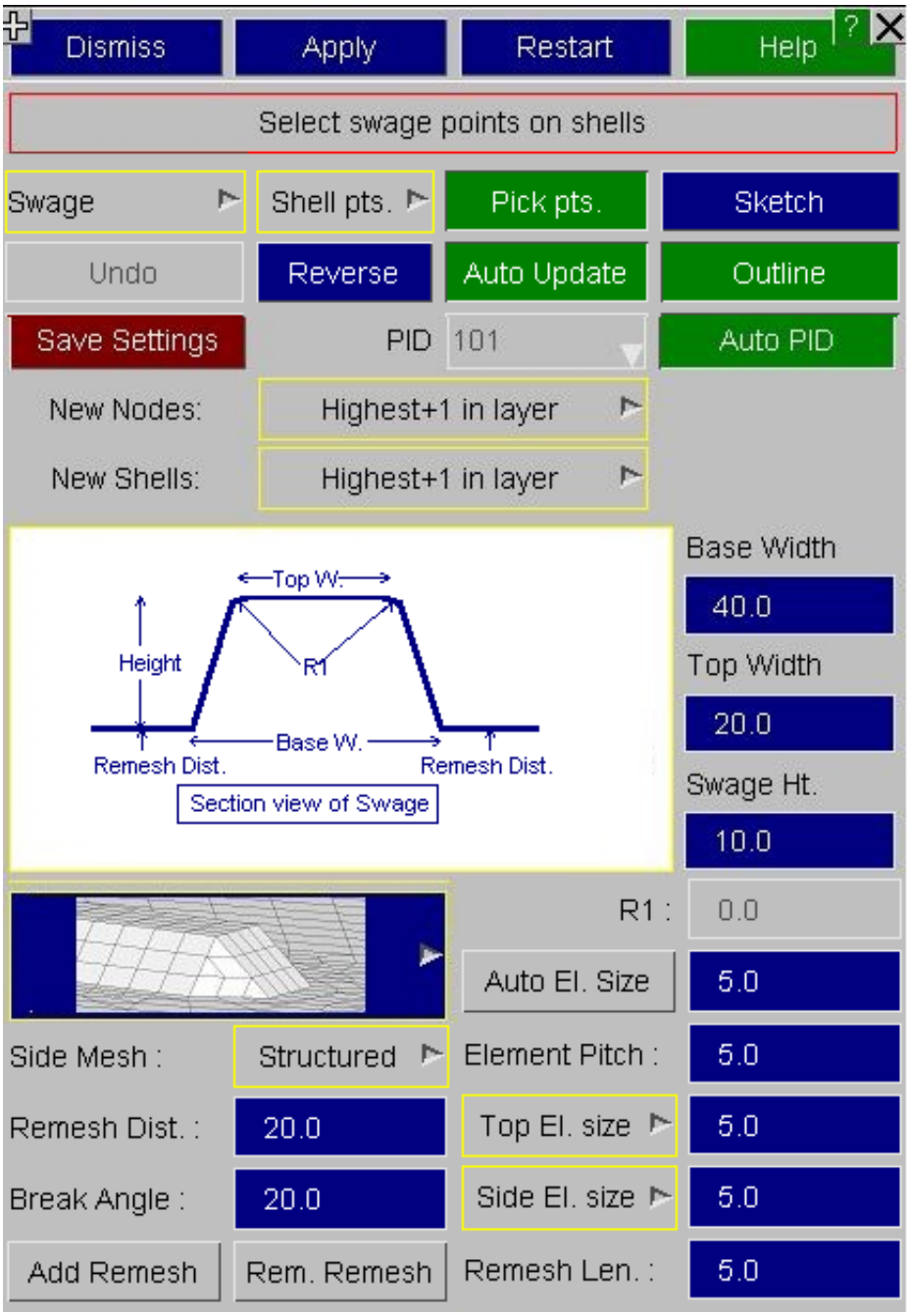
The original shell mesh is remeshed up to the distance from the swage mesh defined by **Remesh Distance.**

The remeshed shells selection is restricted by break angle between normals of the adjacent shells defined by the **Break Angle**.

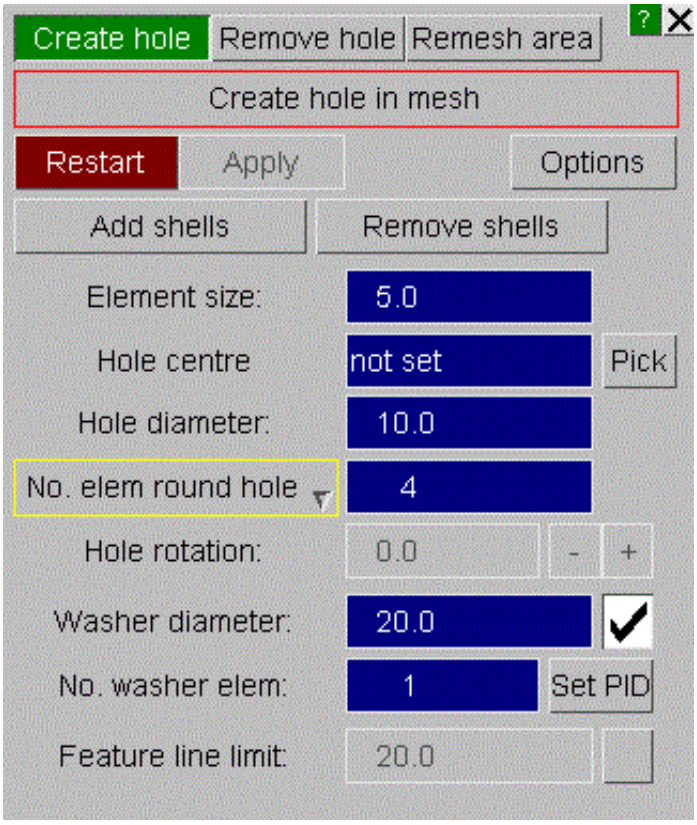
Add Remesh and **Remove Remesh** can be used to alter the selection of shells to be remeshed on the original shell mesh if required.

The element size for the remeshed shell mesh is defined by **Remesh Length**.

Save Settings button saves the current parameters and options values defined in the swage mesh panel to the preferences file.


Dismiss
Apply
Restart
Help
Select swage points on shells
Swage
Shell pts.
Pick pts.
Sketch
Undo
Reverse
Auto Update
Outline
Save Settings
PID
101
Auto PID
New Nodes:
Highest+1 in layer
New Shells:
Highest+1 in layer
Top W.
Height
R1
Base W.
Remesh Dist.
Remesh Dist.
Section view of Swage
Base Width
40.0
Top Width
20.0
Swage Ht.
10.0
R1 :
0.0
Auto El. Size
5.0
Side Mesh :
Structured
Element Pitch :
5.0
Remesh Dist. :
20.0
Top El. size
5.0
Break Angle :
20.0
Side El. size
5.0
Add Remesh
Rem. Remesh
Remesh Len. :
5.0

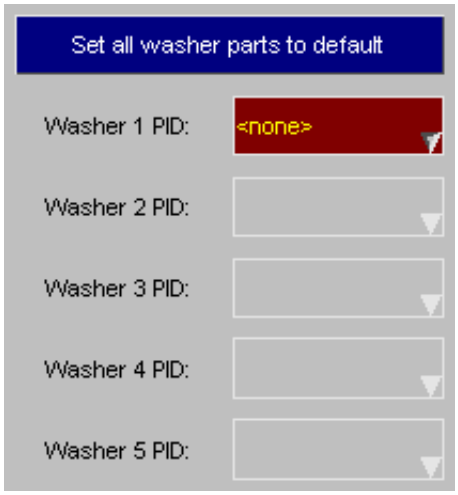
6.31.3 Create hole in mesh



Create hole allows you to make a hole in an existing mesh without any geometry. There are various options to control the mesh and hole created.

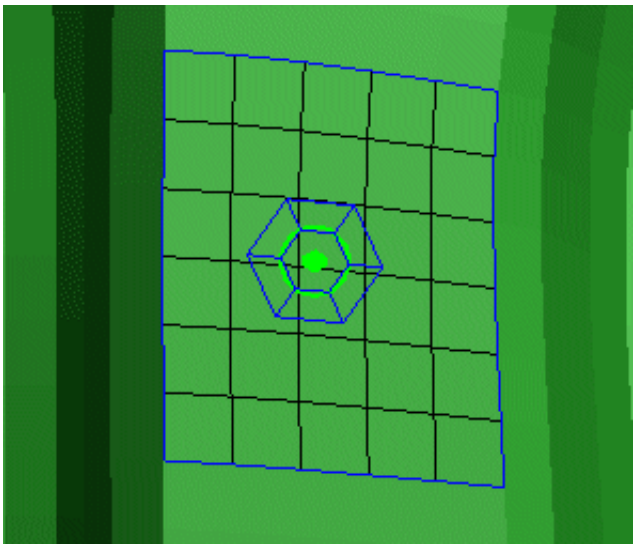
Element size allows you to change the size of elements that will be created.
Hole diameter changes the diameter of the hole created.
The number of elements that will be created around the hole can either be specified by using **No. elem round hole** or the size of the elements can be given instead by using the popup and changing to **Elem size round hole**.
The hole can be created with or without 'washer' elements around the hole. This can be turned on or off by using the checkbox. If it is on then the **Washer diameter** and **No. washer elem** can be used to control how the washer is created.

The **Set PID** button can be used to assign part ids to the to the individual washer layers. If these washer parts are not set, then PRIMER will default back to using the same part as that of the surrounding hole elements for the washer layers.



You can make a hole at a specific coordinate by using the **Hole centre** textbox or you can pick a point on a shell by using the **Pick** button.

Once you have selected the hole location PRIMER will draw a preview of the hole that will be created (here shown with 1 washer element and 6 elements around the hole). The elements which PRIMER proposes to remesh are also sketched. You can now update the hole properties if required and the preview will update. If the hole is made bigger then PRIMER will automatically select more elements to remesh. If you want to manually add or remove elements to the selection you can use **Add shells** and **Remove shells**.



The **Hole rotation** can be used to change the orientation of the elements for the hole. Either use the **+** and **-** buttons or type in a new angle.
Feature line limit can be used to stop automatic selection of shells which are beyond a certain angle. e.g. in this example the shell selection on the left and right of the hole stops at the change in angle.

Its meaning is the same as selection in object menus with feature lines.

When you are happy with the hole properties pressing **Apply** will create a preview of the mesh that PRIMER will create. You can still change the hole properties and PRIMER will update the mesh interactively. To actually create the mesh press **Confirm**.

6.31.4 Remove hole from mesh

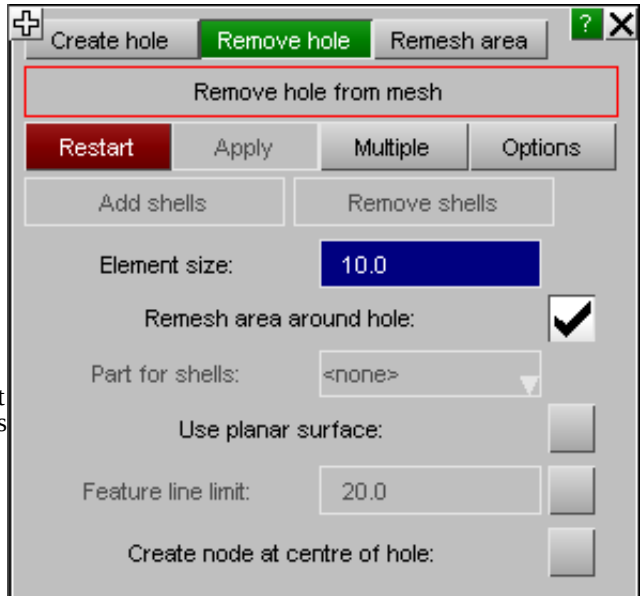
The **Remove hole** function allows you to either remove a single hole or multiple holes in parts or shells.

Removing a single hole

To remove a single hole toggle the **Multiple** button off (it is off by default)

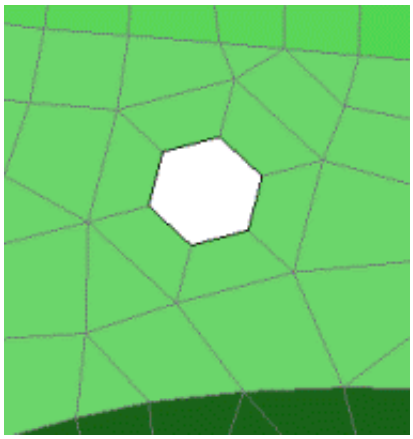
To remove a hole from a mesh pick a shell element next to the hole. PRIMER will then automatically select shells around the hole to remesh similarly to creating a hole. **Add shells** and **Remove shells** can be used to alter the selection.

There are two ways that PRIMER can remove the hole. It can either just fill in the hole with new elements or it can completely remesh the area around the hole to remove the hole completely. This is controlled by the **Remesh area around hole** checkbox. If selected then the result is shown of the right hand image below. If not selected then by default the shells created in the hole are in the same part as the shells around the hole, but this can be altered with the **Part for shells** textbox.

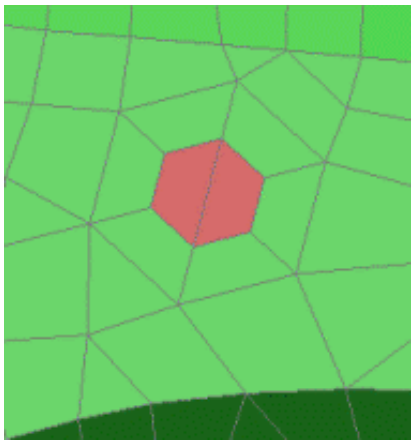


Feature line limit can be used to stop automatic selection of shells which are beyond a certain angle. Its meaning is the same as selection in object menus with feature lines.

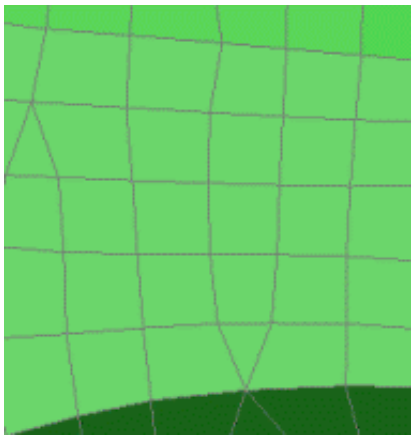
Original hole



Remesh area unset



Remesh area selected



If the surface that you are remeshing is flat then PRIMER will create the new shells on the flat surface. If the surface is curved then PRIMER will attempt to use the normals of the shells around the hole to make sure that the new shells that are created follow the curvature of the surface. This isn't always what you want. Selecting **Use planar surface** will force PRIMER to try to use a planar surface when creating shells. The planar surface will be the best 'fit' that can be created from the nodes on the edge of the hole.

When removing the hole PRIMER can optionally make a node at the centre of a hole. This may be useful if you want to make a beam or connection at the hole position. This is controlled by the **Create node at centre of hole** checkbox.

When you are happy with the properties pressing **Apply** will create a preview of the mesh that PRIMER will create. You can still change the properties and PRIMER will update the mesh interactively. To actually create the mesh press **Confirm**.

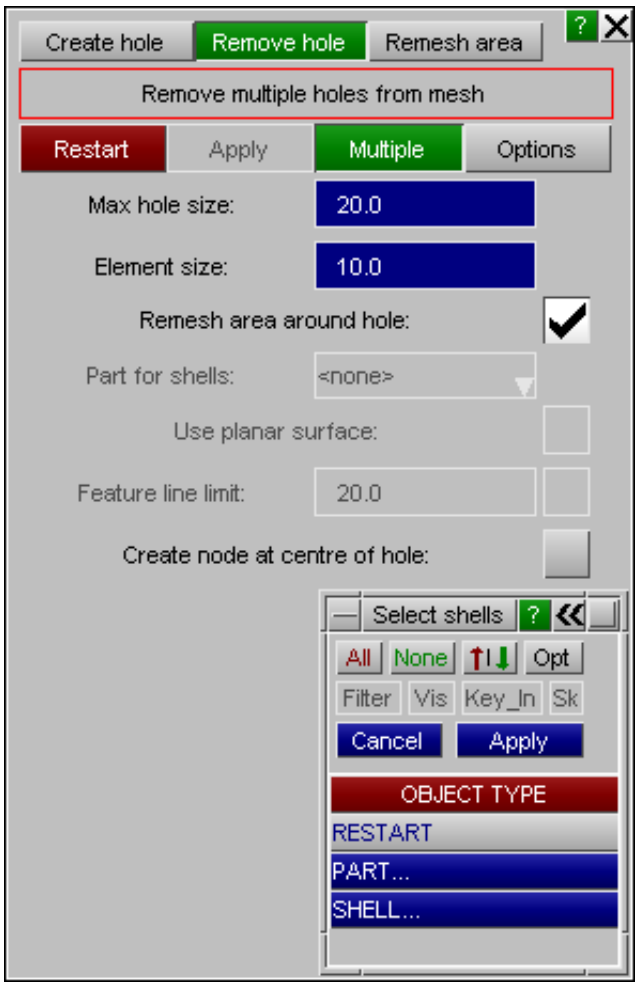
Removing multiple holes

To remove multiple holes toggle the **Multiple** button on

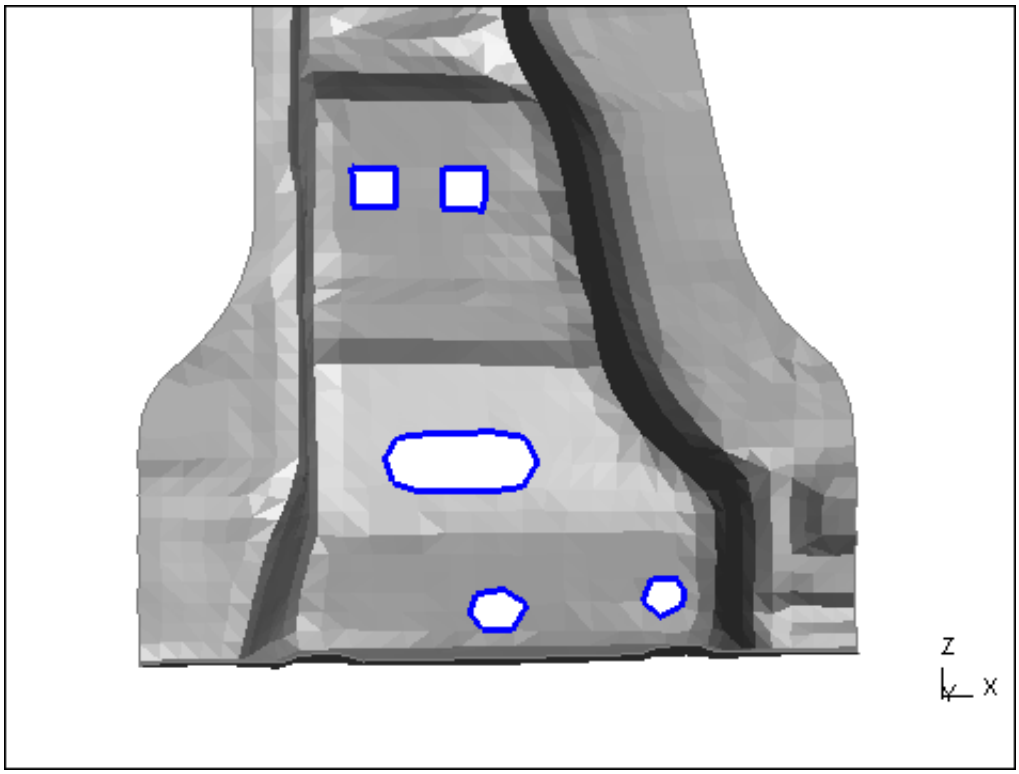
PRIMER allows you to choose the maximum hole size that you want to remove. Use **Max hole size** to change the maximum size hole that PRIMER will remove.

As when removing single holes there are two ways that PRIMER can remove the hole. It can either just fill in the hole with new elements or it can completely remesh the area around the hole to remove the hole completely. This is controlled by the **Remesh area around hole** checkbox.

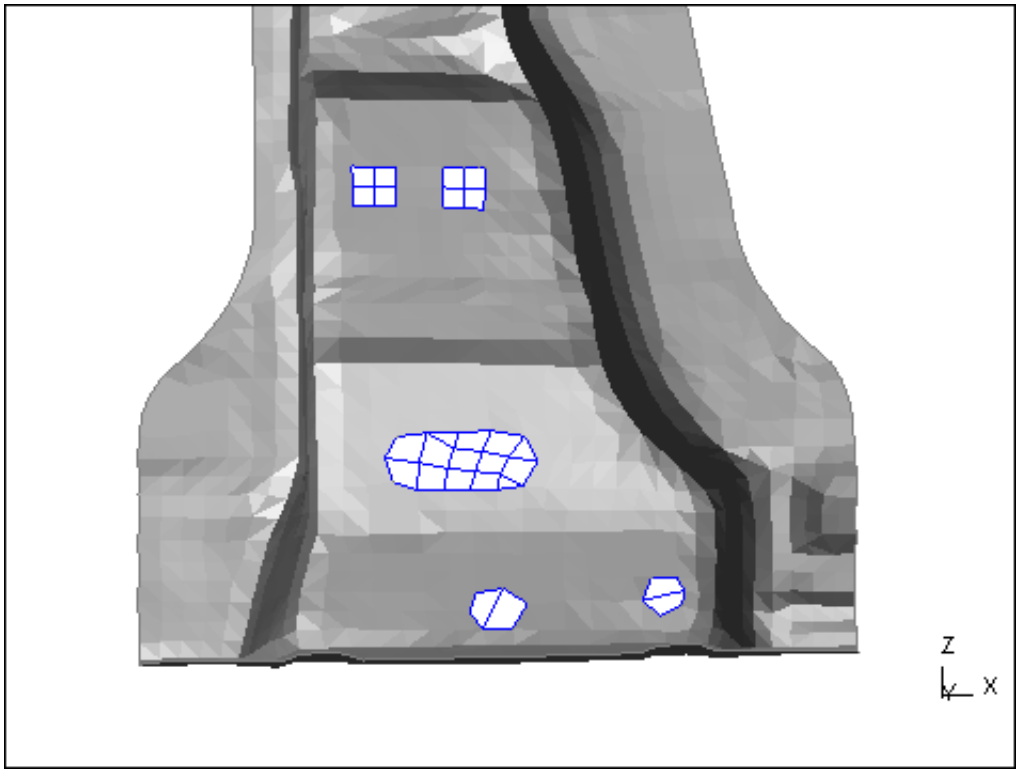
Once you have chosen the maximum hole size and the remeshing option use the object menu to choose the part(s) or shells that contain the holes you want to remove. Once you have selected what you want press **Apply** in the object menu.



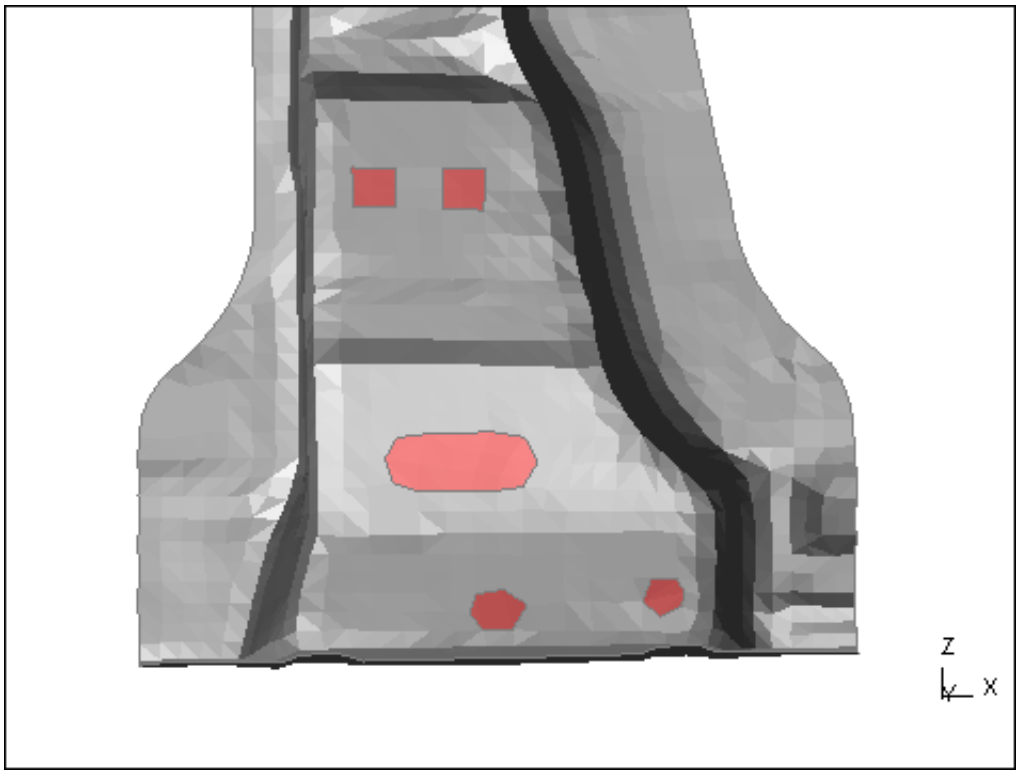
Once some shells are selected PRIMER will look for any holes automatically and highlight them with blue lines. In the image below PRIMER has found five holes to remove.



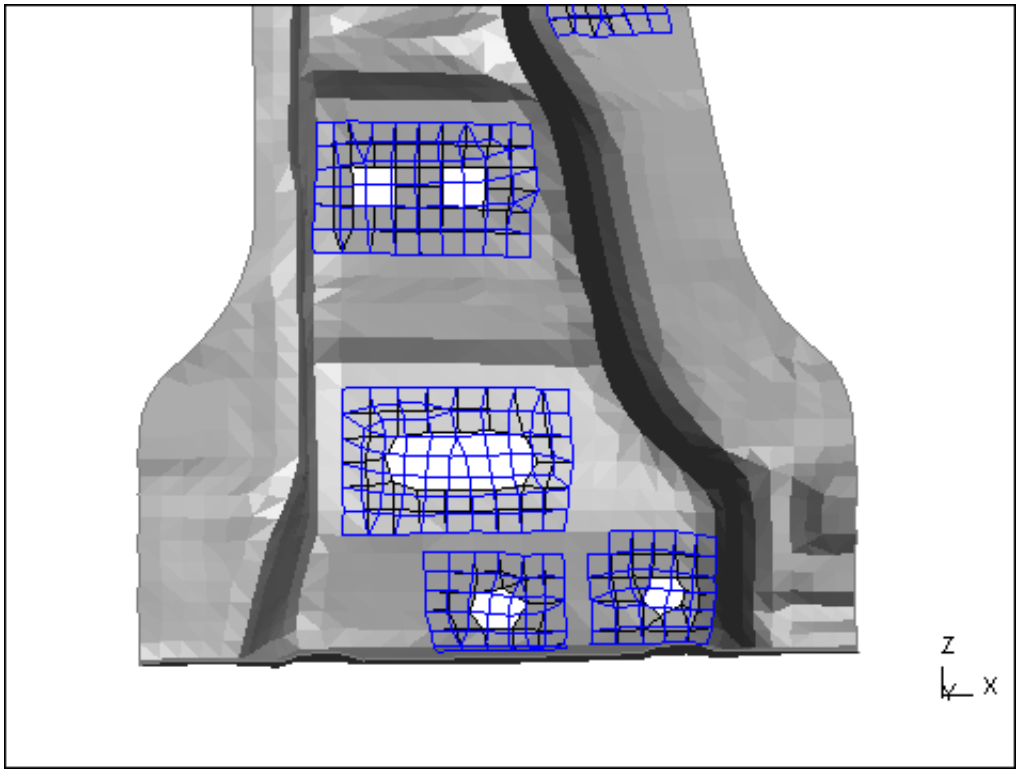
Once the holes you want to remove are highlighted press **Apply**.
If **Remesh area round hole** is unset then PRIMER will sketch the shells it will create in the hole



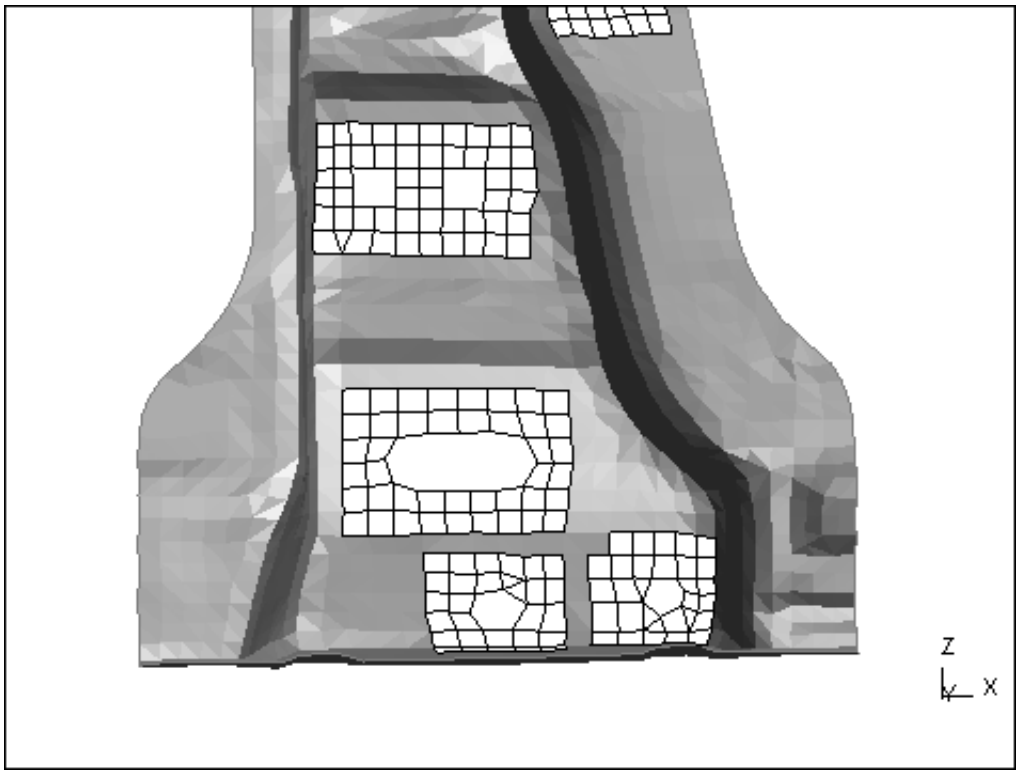
Press **Confirrm** and PRIMER will actually create the elements.



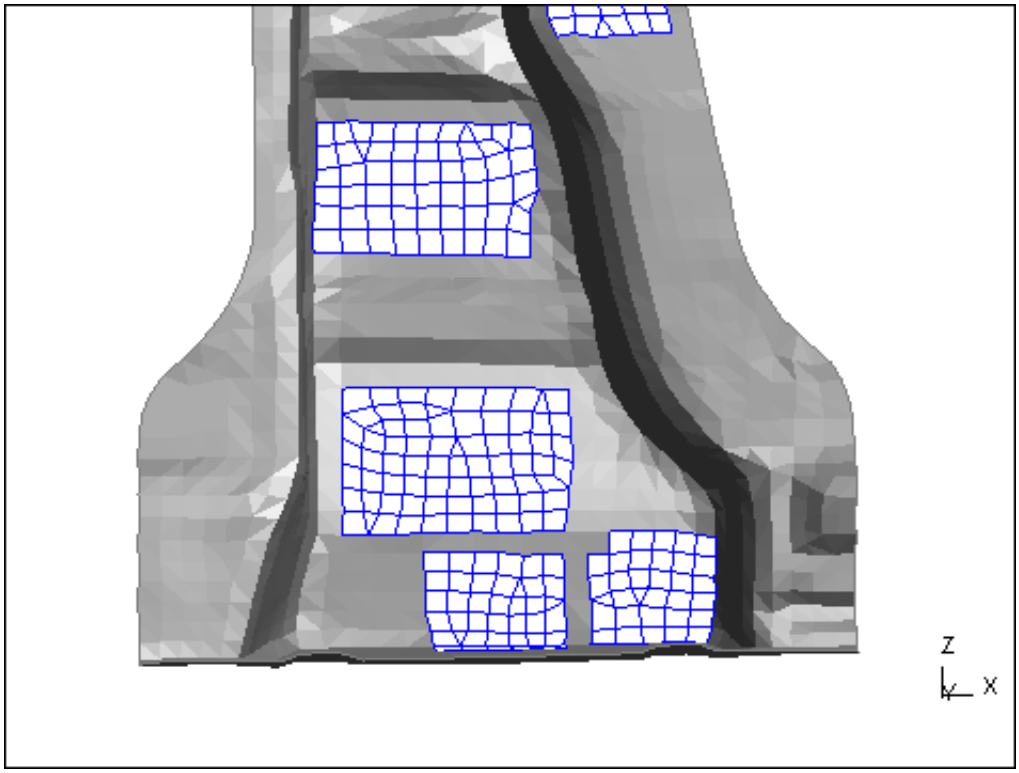
If instead **Remesh area round hole** is set then when **Apply** is pressed PRIMER will automatically select some of the shells around the hole to remesh and try to remesh the area(s). The shells PRIMER will remesh are shown in the sketch colour (black in the image below) and the shells PRIMER will create are sketched in blue



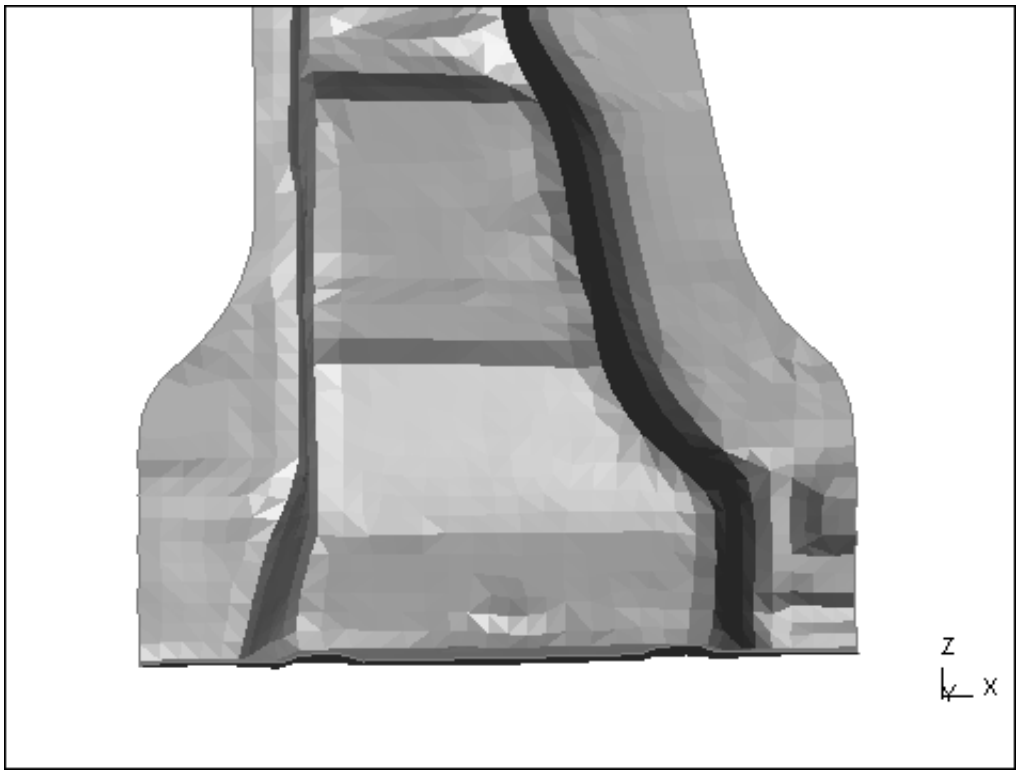
As this can be confusing the **Toggle mesh visibility** button can be used to alter what is shown. Pressing once will show the original mesh only



Pressing again will show the proposed new mesh only



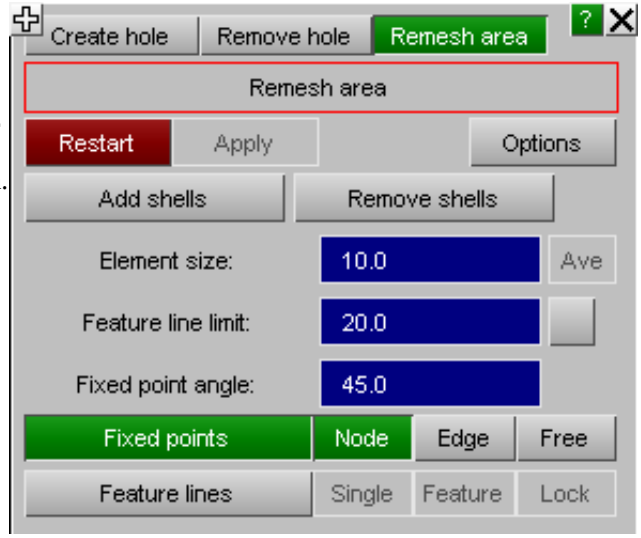
Pressing again will return to the original view.
If you are happy with the proposed mesh press **Confirrm** and PRIMER will actually create the elements.



6.31.5 Remesh area

The **Remesh area** function was completely rewritten for version 13 of PRIMER.

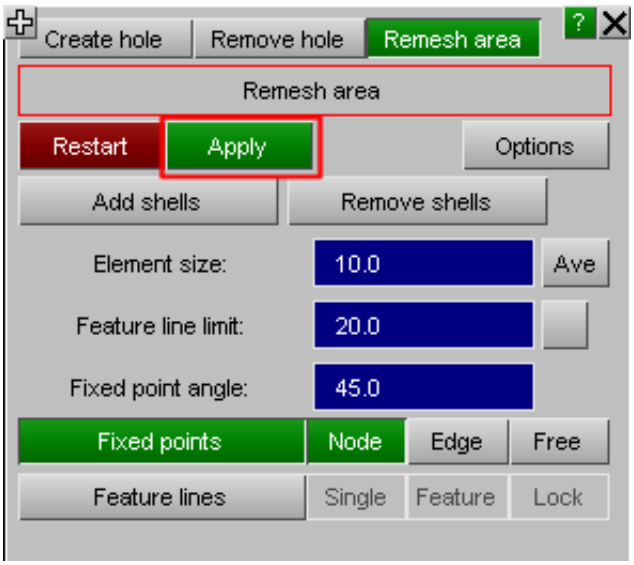
To start the process PRIMER needs some shells to remesh. Use the **Add shells** and **Remove shells** buttons to add or remove shells to remesh. Both of these will map an object menu which will allow you to choose which shells to remesh. Use the **Apply** button in the object menu to add or remove them.



Once there are some shells to remesh the **Apply** button will become active.

The target size for new shells is given with the **Element size** textbox. You can either type in the necessary size in the textbox or pressing **Ave** will use the average size of the selected shells as the target size.

Shells can be added or removed as much as you like. Note that the shells do not have to form a single attached area. There can be several different areas of shells. When you are happy with the selection pressing **Apply** will take you to the next step.

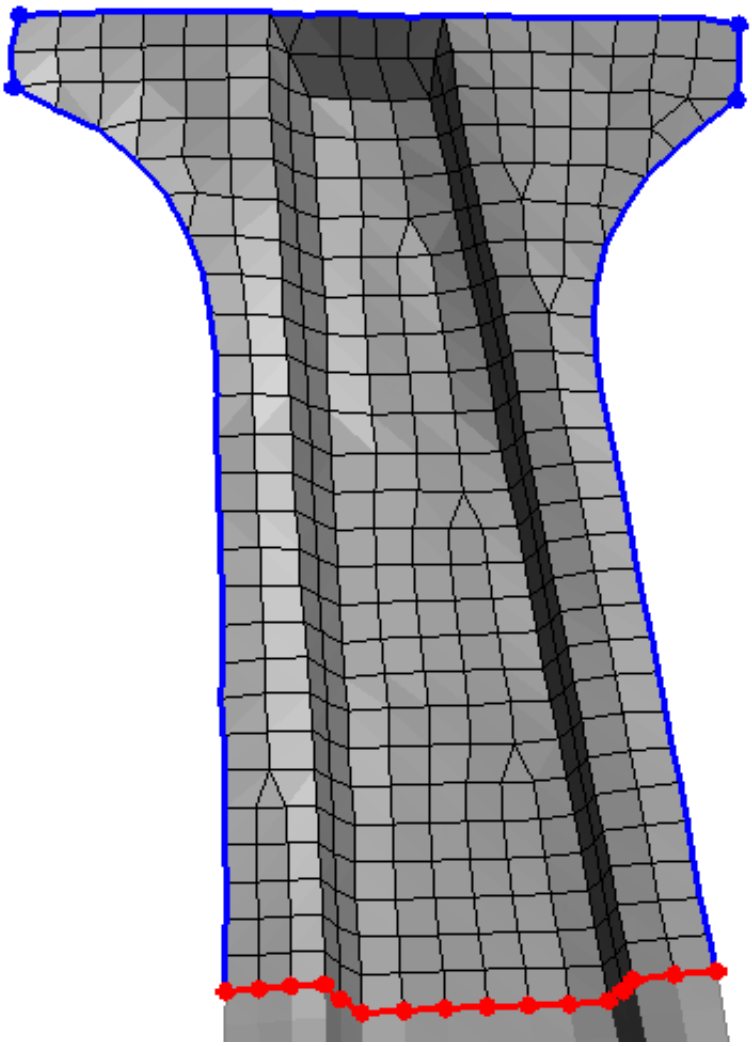


To illustrate some of the remeshing features we will show an example of remeshing the top part of a component with a finer mesh. The mesh on the right has an average size of about 10mm. We want to remesh it with a size of 6mm.
To start the meshing process PRIMER looks at the selected shells to find any free edges and edges which are shared by other elements.

The selected shells are sketched (shown in black in the image on the right).

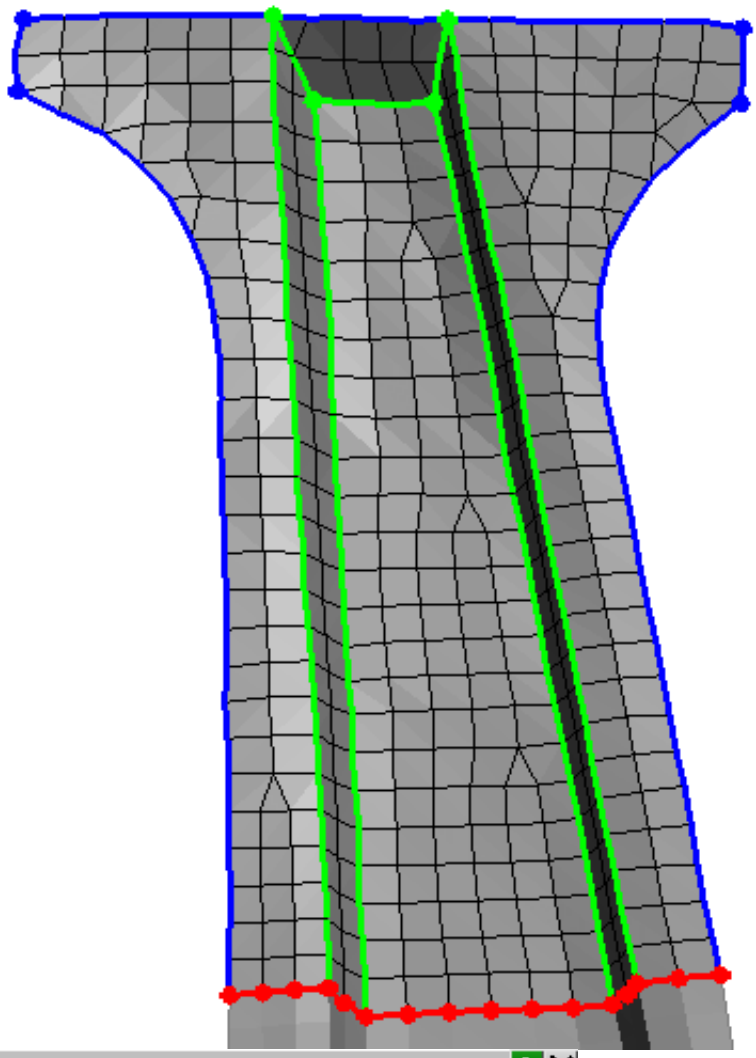
Edges that are shared with other elements are shown in red and a red blob is drawn for each node that PRIMER must keep to maintain the mesh connectivity.

Free edges are shown in blue. PRIMER will look along the free edges and anwhere where two adjacent free edges have an angle greater than **Fixed point angle** a fixed node will also be added to ensure that any sharp edge edges are maintained. In the image on the right these are shown as blue blobs.
If a new value is given for **Fixed point angle** then PRIMER will recalculate and redraw as required.

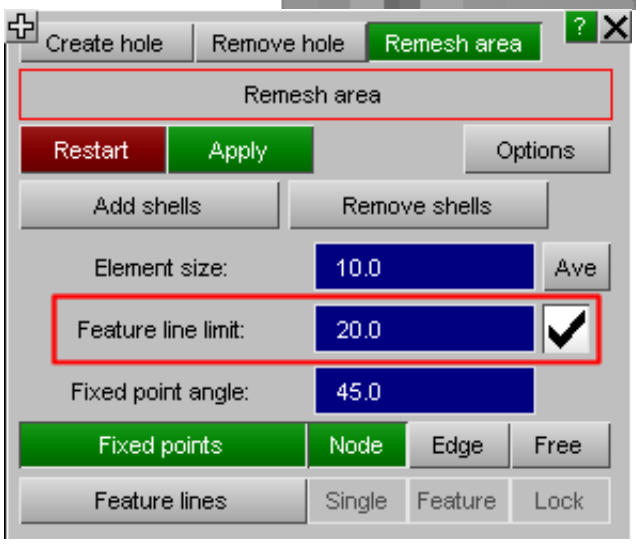


By default PRIMER will only preserve the edges of the mesh. There may be feature lines on the mesh where you want to ensure that nodes are created.

The **Feature line limit** tells PRIMER to look for any feature lines in the selected shells that have an angle greater than feature line limit. If it is turned on by ticking the checkbox then PRIMER will show the feature lines in green. Where feature lines intersect PRIMER will also add fixed points (shown as green blobs). The image on the right shows the feature lines found with an angle of 20 degrees.

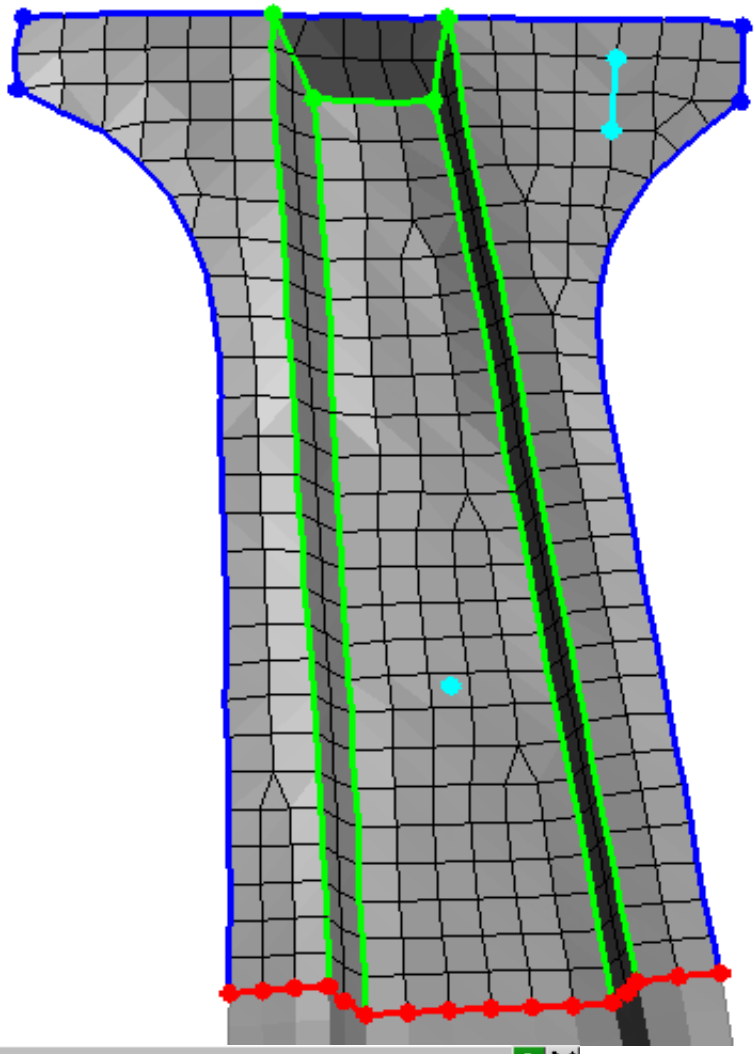


If a new value is given for **Feature line limit** then PRIMER will recalculate and redraw as required.

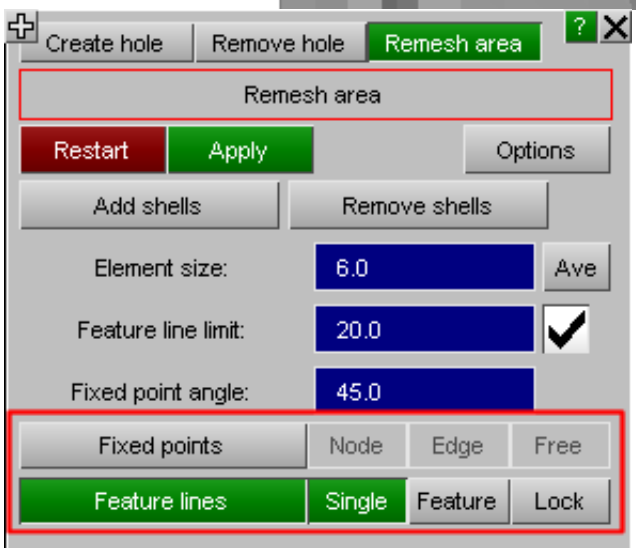


It is also possible to add extra fixed points and free edges to the mesh that should be preserved. To do this use the **Fixed points** and **Feature lines** tools. The tool which is currently selected is shown in green.

The **Fixed points** tool has **Node**, **Edge** and **Free** options. In the image above the **Node** tool is active.
Node allows a fixed point to be added/removed on an existing node.
Edge allows a fixed point to be added/removed on an existing element edge.
Free allows a fixed point to be added at a free location on an existing shell.
Fixed points that have been added manually are shown as cyan blobs.



The **Feature lines** tool has **Single** and **Feature** options.
Single allows a feature line to be created on a single element edge.
Feature allows the free edge to propagate until the end of the feature is found.



To add a fixed point left click with the mouse where you want the fixed point to be created.
To remove a fixed point right click with the mouse on the fixed point you want to remove. Only manually added fixed points (shown in cyan) and fixed points added by PRIMER on free edges (shown in blue) can be removed.

In the image above there is a **Free** fixed point in the middle of the component and there are two **Node** fixed points. PRIMER will ensure that nodes are created where there are fixed points.

To add a feature line left click with the mouse on an element edge where you want the feature line to be created.
To remove a feature line right click with the mouse on the feature line you want to remove. Only manually added feature lines (shown in cyan) can be removed.

Lock allows you to lock a mesh on a free edge/feature line. This will ensure that the original nodes are maintained. This may be useful to keep nodes on hole boundaries that are used in nodal rigid bodies or connections. A locked edge will be drawn in red with red blobs for the fixed nodes.

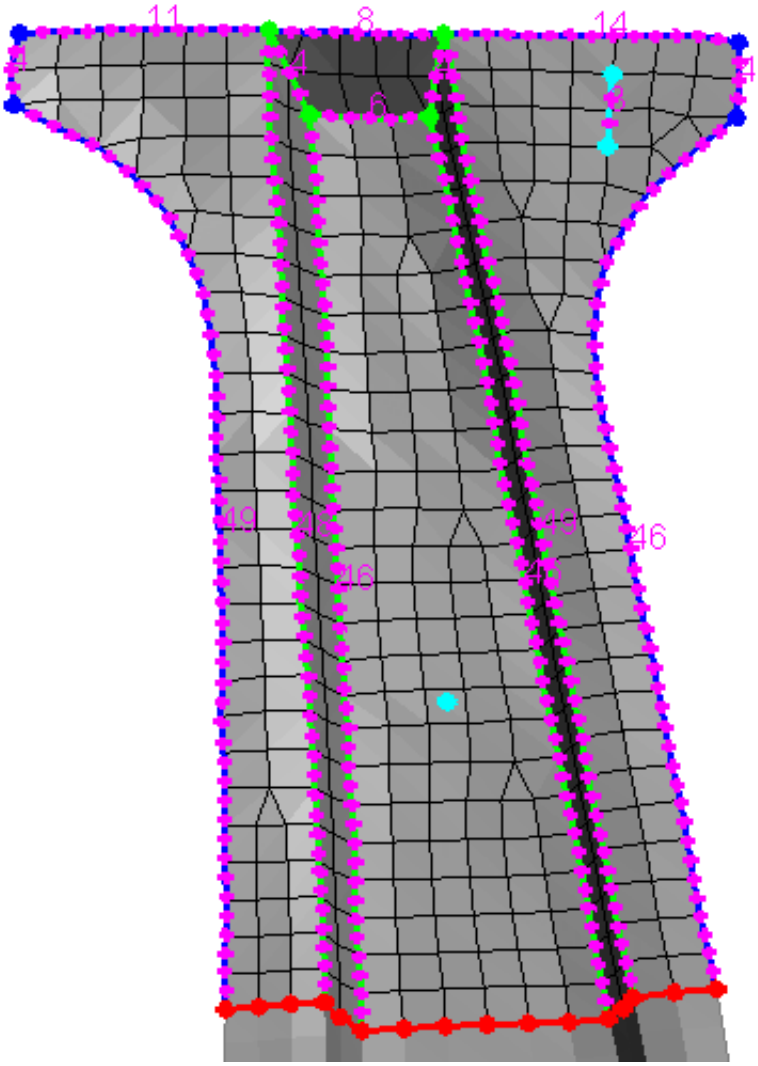
In the image above there are feature lines on 2 element edges. PRIMER will ensure that nodes are created along feature lines.

Once you have created any **Fixed points** and **Feature lines** that you need in the mesh and set the Element size to the target mesh size (6mm in this example) press **Apply** and PRIMER will move onto the next stage.

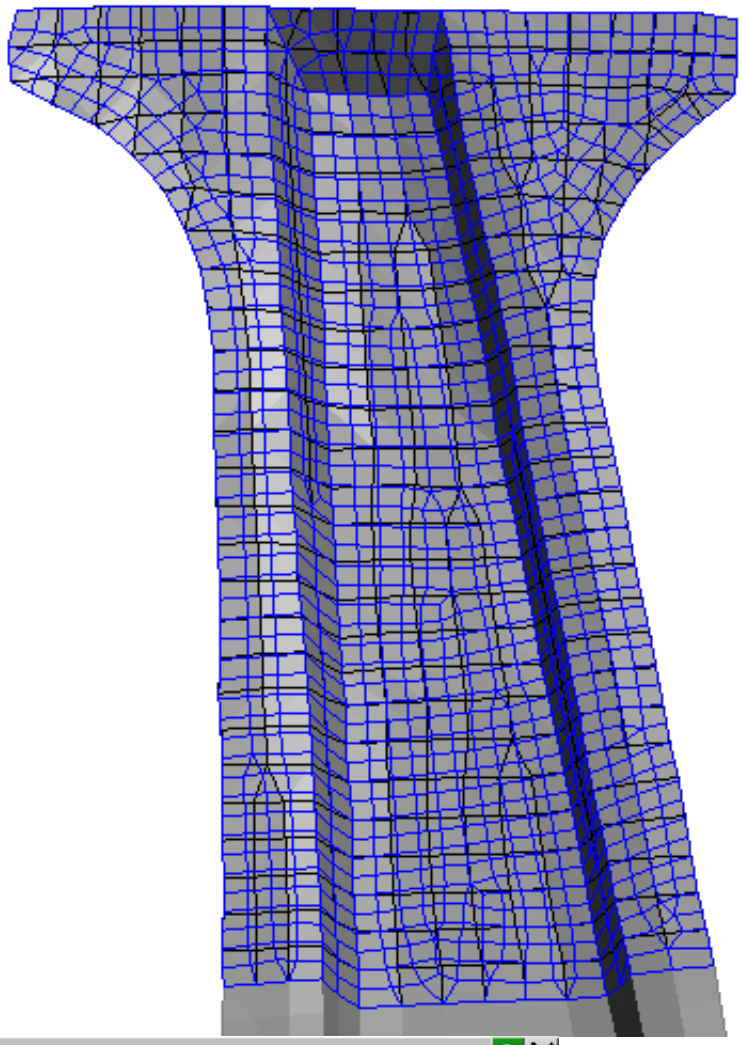
PRIMER will look at the free edges and feature lines that you have selected and choose how many nodes to create on them to achieve the target mesh size. A magenta + symbol will be shown where each node will be created and then number of elements that will be created between fixed points on a free edge is shown as a magenta number.

To increase the number of elements created along a free edge left click with the mouse on the number.
To decrease the number of elements created along a free edge right click with the mouse on the number.
PRIMER will adjust the mesh density along that free edge.

If you want to go back and add some more fixed points and/or feature lines press **Reject**.
Once you are happy with the mesh density then press **Apply** to preview the mesh.



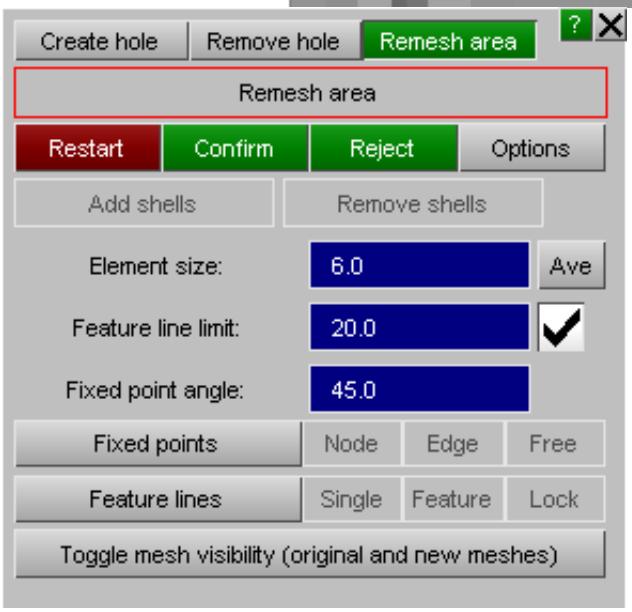
The preview of the mesh is shown in blue.



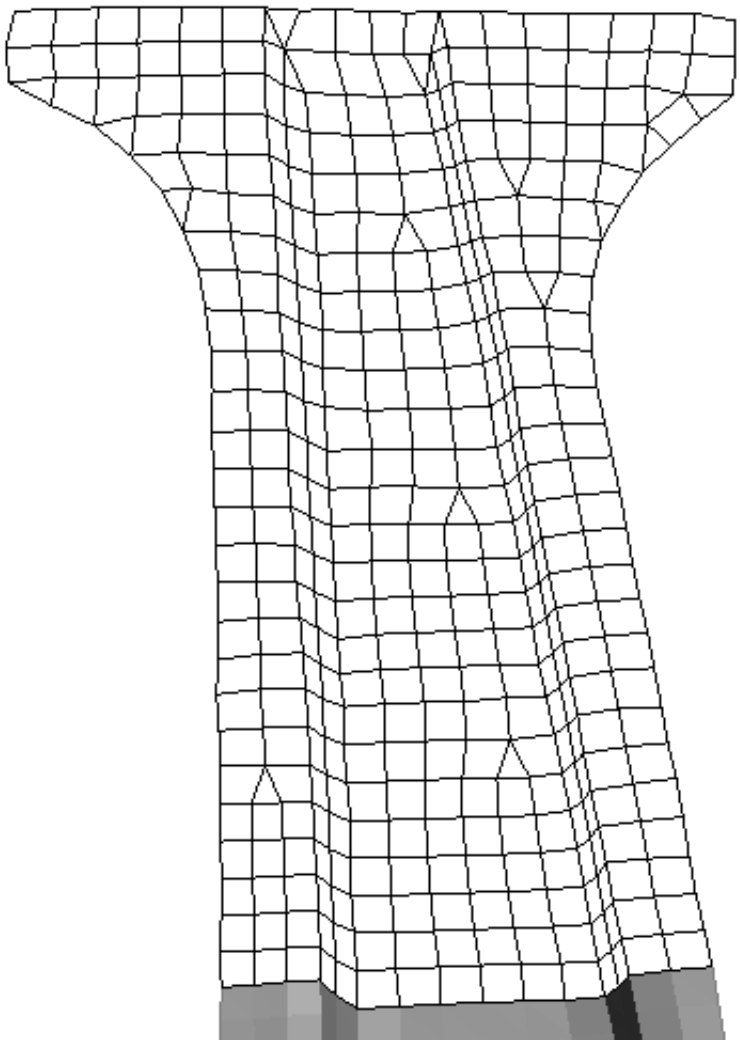
If you are happy with the mesh press **Confirm** to finalise the mesh.
If you want to go back and change the mesh density or add fixed points/feature lines press **Reject**.

It can be difficult to see the new mesh so the **Toggle mesh visibility** button can be used to toggle between both meshes, the original mesh and the new mesh.

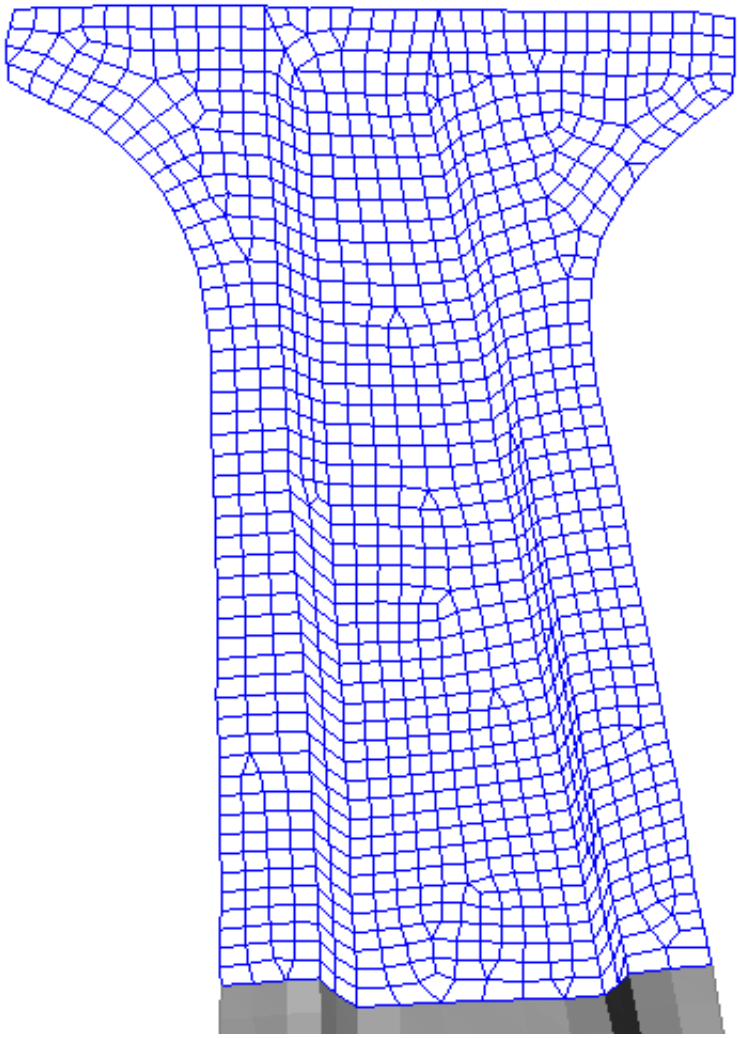
It can be difficult to see the new mesh so the **Toggle mesh visibility** button can be used to toggle between both meshes, the original mesh and the new mesh.



Original mesh shown after pressing **Toggle mesh visibility** button once



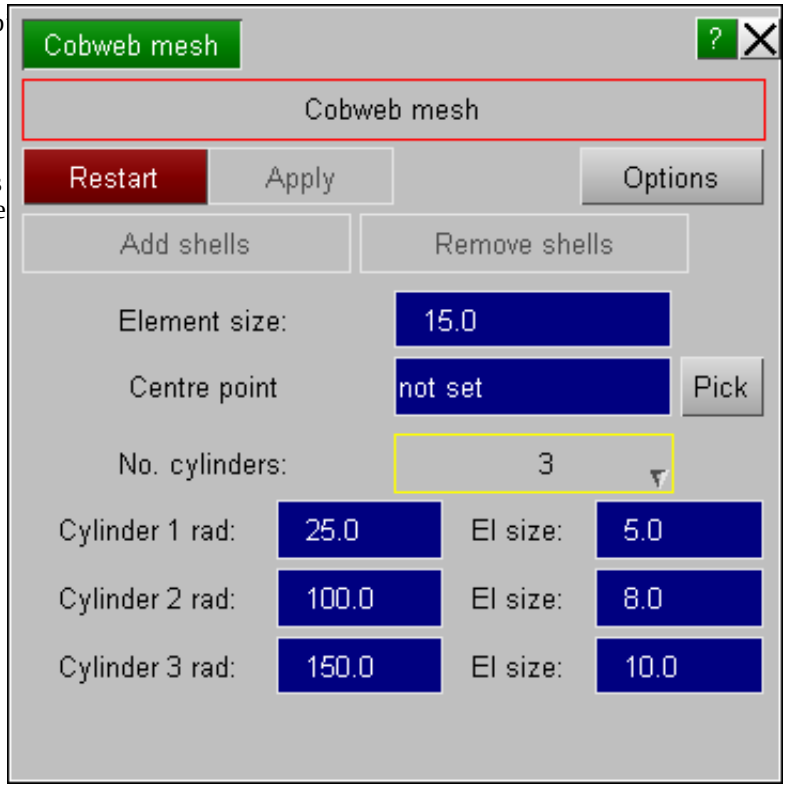
New mesh shown after pressing **Toggle mesh visibility** button again.



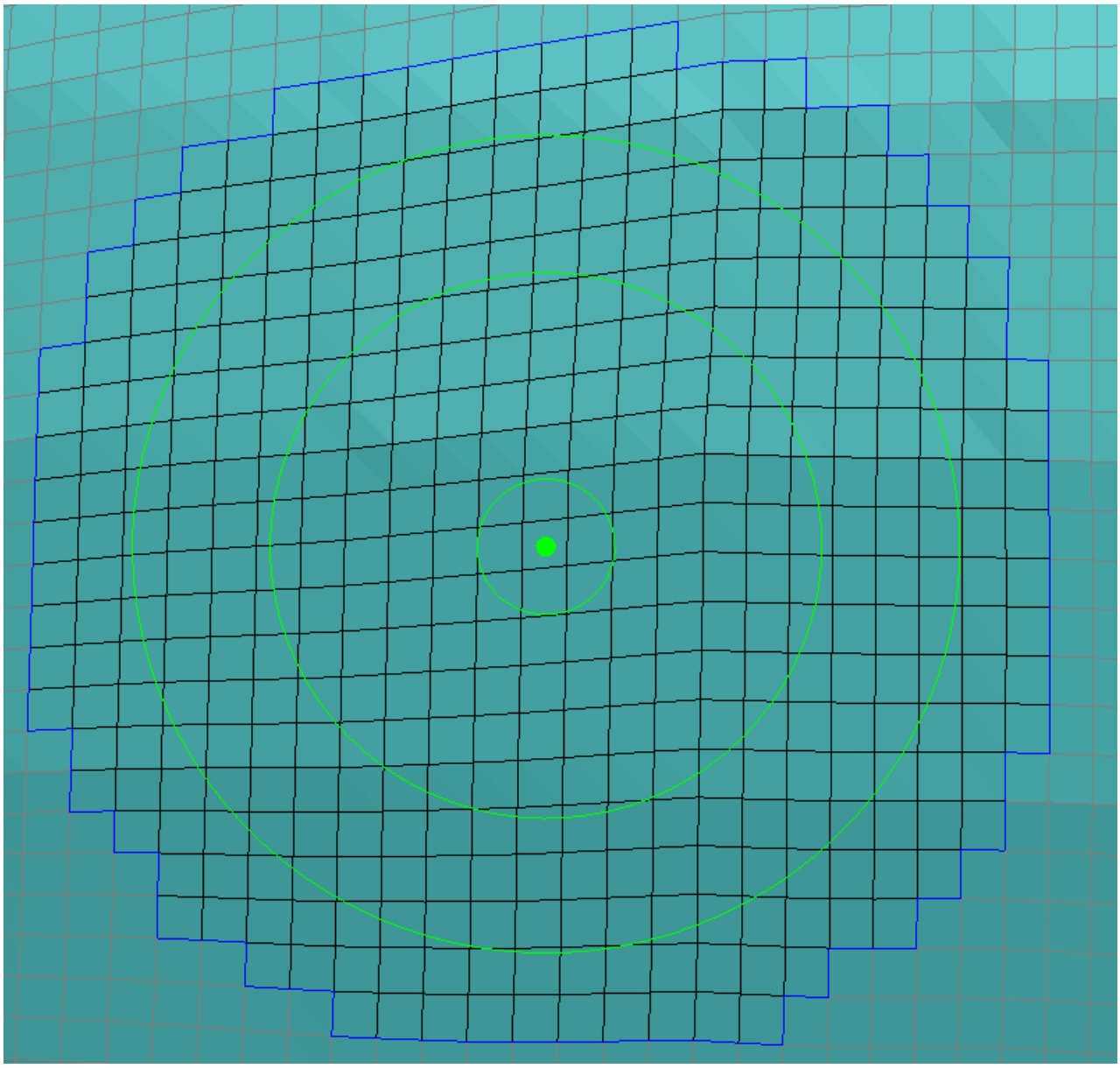
6.31.6 Cobweb mesh

The **Cobweb mesh** feature creates a 'cobweb mesh pattern. This is sometimes used in areas where crack propagation needs to be modelled at impact points (for example a head impact on a windscreen).

The input parameters are a number of cylinders and their radii, the element size you want to use for each cylinder and the element size used outside the cylinders. In the menu on the right the element size gets smaller towards the centre.

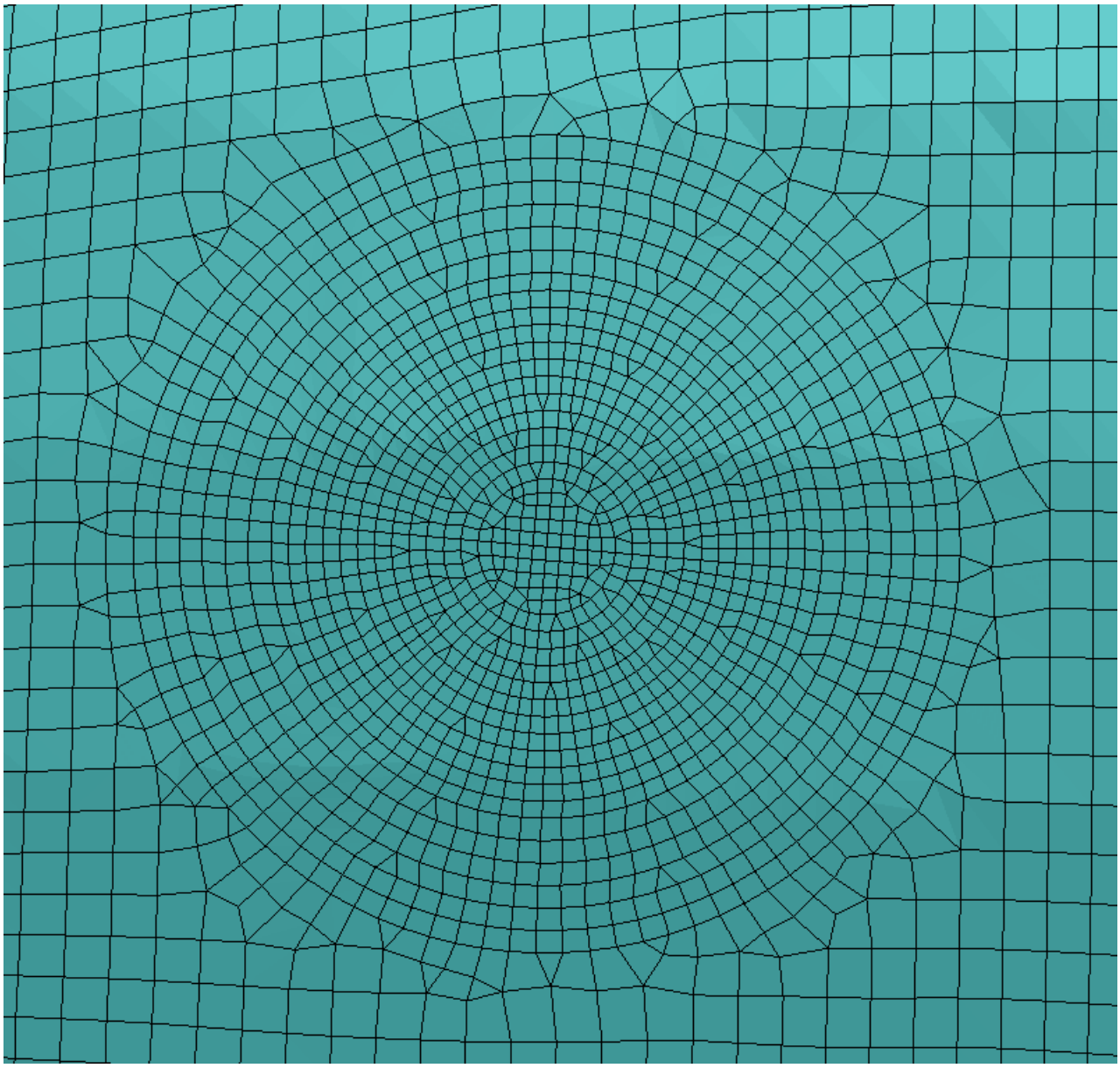


When you have the correct parameters use **Pick** to choose a centre point or type in a centre point into the **Centre point** textbox. PRIMER will then automatically select shells around the hole to remesh similarly to creating a hole and show the cylinders and centre point on the mesh. See the following image for an example.



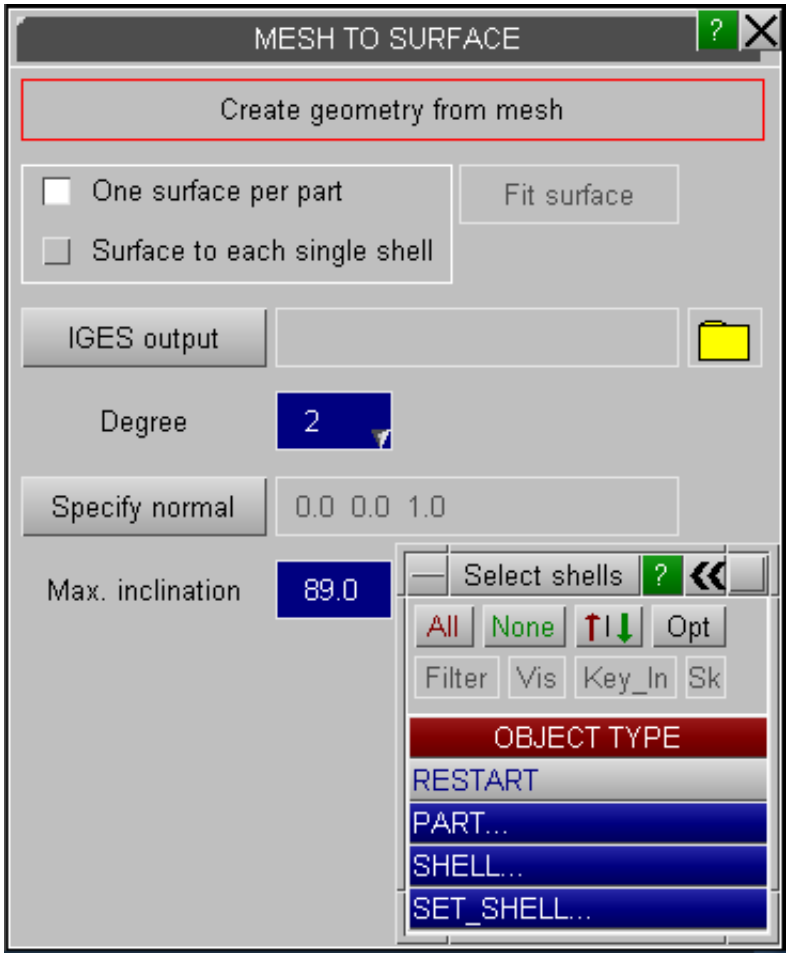
Add shells and **Remove shells** can be used to alter the selection if required. You can also adjust any of the parameters if required. When you are happy with the properties pressing **Apply** will create a preview of the mesh that PRIMER will create. To actually create the mesh press **Confirm**.

An example of a cobweb mesh (with the parameters from the above example) is shown in the following image.



6.31.7 Create geometry from mesh

To fit a NURB surface to a mesh, **Mesh->surf** under Mesh Tools opens a panel as shown on the right. This allows you to fit geometry surfaces to a selection of shells, which can be done in the object menu by parts, shell sets or shells directly. If the **IGES output** option is toggled before clicking **Fit surface**, an IGES file containing all surfaces created in this operation is written.



Surfaces can be fitted in two different modes, which are illustrated in the figures below. You can choose to create **One surface per part** of the selection (left) or a **Surface to each single shell** (right), where different colours indicate different surfaces. The latter method creates a linear NURB surface with 2 by 2 control points on the corners of each shell in the selection. This typically creates a large number of small surface pieces, but this is robust for all types of shell selections.

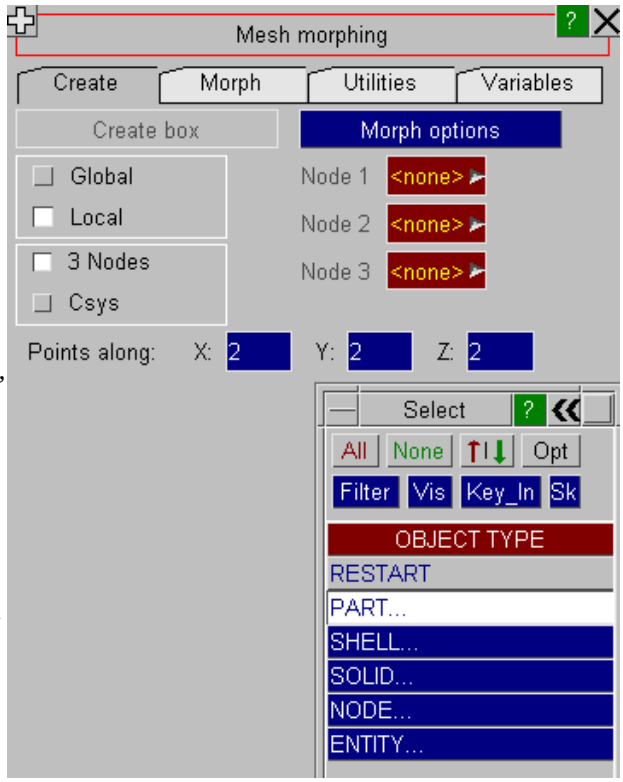
When fitting one surface to a part, a reference plane is fitted to the part using a principal component analysis on the nodal coordinates. The method requires that the linear projection of the part to its reference plane hasn't got any overlaps. This condition is given for "flat" parts like windows or doors of a car. Whenever you attempt to fit a surface to a part not satisfying this restriction, e.g. to a tyre or a bumper, this part is skipped and you will get a warning that the selection of shells is too curved. There is a threshold for the inclination of a shell to the reference plane, which is set to 89.0 degrees by default. The closer the angle specified for **Max. inclination** is to 90 degrees, the fewer parts are rejected as too curved, but this may compromise the quality of the surfaces.

It is also possible to split up too curved parts such that Primer creates several surfaces for them. Unless fitting a surface to each single shell, you will have to specify the shells for each surface piece manually. This can be done by selecting shells or shell sets in the object menu.

In exceptional cases the reference plane fitted by the aforementioned principal component analysis is not optimal for the algorithm. If the result is unsatisfactory but you think that the selected shells of the part do project onto a plane without overlap, you can use the **Specify normal** option and type the normal vector of the desired plane into the text box.

All surfaces created by these functions will be NURB surfaces whose **Degree** can be set to 1 (linear), 2 (quadratic) or 3 (cubic) for both parametric coordinate directions.

6.31.8 Mesh morphing



The **Morph** under Mesh Tools allows you to morph a mesh. This is done with morph boxes around the nodes whose coordinates can change, and morph points on the box can be moved to control how these nodes are morphed. Morph points and morph boxes will be written after *END in keyword files, so they can continue to be used in later PRIMER sessions. Details of the format for these keywords can be found in appendix 18.

When the Morph panel is first opened, it appears as shown on the right. This is the mode to create new morph boxes in the model.

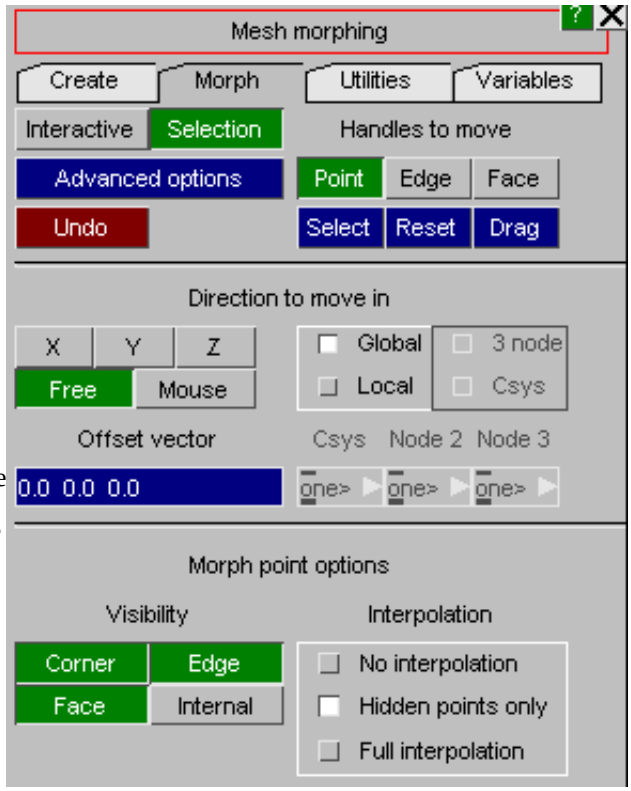
The mesh to be morphed can be specified by Part, Shell, Solid, Node or any entity type. The actual morphing is then performed on the underlying nodes in all cases: The coordinates of the nodes will change in general, but the topology of the elements will remain the same during morphing.

Morph boxes will be created as bounding boxes around the selected entities. Only when the selection is too close to one plane, then a specified minimum size will be applied. This can be controlled on the **Morph options** button.

The morph boxes can be created as bounding boxes in either Global coordinates or Local coordinates defined by one of the following:

- **3 Nodes**: local coordinate system defined by three nodes.
- **Csys**: selecting a *DEFINE_COORDINATE_SYSTEM.

By default morph boxes have got eight morph points at the corners of the box. In that case the nodal coordinates inside the box will be recalculated linearly in each coordinate direction when morph points have been moved. It is also possible to specify more than two points in a coordinate direction for the box creation, in which case nodal coordinates are calculated as higher degree polynomial (a combination of Bernstein polynomials in each coordinate direction) when coordinates of morph points change in the morphing panel. These high order morph boxes therefore give you more control over smoothness of the morphing deformations.



The following items can be dragged to morph a mesh:

- Morph points: These are drawn as little green cubes at the corners of a morph box for linear boxes (with 2 by 2 by 2 morph points) or in a 3-dimensional grid on morph boxes of any order. These can be turned on or off using the **Point** button under **Handles to move**.
- Edge drag handles: They are by default in the middle of an edge segment between two adjacent morph points. Visibility of these handles is controlled by the **Edge** button
- Face drag handles: They are by default in the middle of a rectangle of four adjacent morph points (a face in the case of a 2 by 2 by 2 box). Here visibility can be switched with the **Face** button.

When the dragging operation for the morphing is active, hovering the mouse over a drag handle highlights all morph points which will move as soon as you start dragging from the current mouse position. The result of the morphing on the nodal coordinates after moving an edge or face handle will be the same as after moving all underlying morph points individually, but using the drag handle makes it easier to move the morph points all by the same vector.

There are two different ways of selecting drag handles:

- **Interactive** : allows to select just one handle interactively by dragging with a mouse button.
- **Selection** : allows to select multiple handles by interactively picking on them one at a time. They can then be dragged all at the same time by the same vector.

PRIMER allows you to morph in any direction in both the global coordinate system and in a user-defined local coordinate system.

- **X** : drag along X direction of the global coordinate system or in a user-defined local coordinate system.
- **Y** : drag along Y direction of the global coordinate system or in a user-defined local coordinate system.
- **Z** : drag along Z direction of the global coordinate system or in a user-defined local coordinate system.
- **FREE** : drag along any direction in the plane of view.
- **MOUSE** : drag along X, Y or Z direction in both global coordinate system or in a user-defined local coordinate system, using left mouse button for X, middle mouse button for Y and right mouse button for Z.

When dragging morph points, the **Offset vector** text box dynamically shows the vector by which the morph points have been moved. It is shown in either global or user-defined local coordinates depending on the setting for the dragging. It is also possible to type in a vector (with its components separated by spaces) to this text box to move the selected morph points by that given vector.

PRIMER allows you to revert the last mesh morphing operation using **Undo** button.

When there are high order morph boxes in PRIMER, there are buttons to filter visibility of morph points by their type **Corner**, **Edge**, **Face** or **Internal**. By default the internal morph points are not drawn. These morph point types should not be confused with the types of drag handles at the midpoints of edges and faces controlling several morph points at the same time. The midpoint drag handles can also be turned on or off for high order boxes, but by default they are not drawn to make the view clearer.

When dragging or applying an offset vector to selected morph points on a high order box, the movements of unselected morph points on the same box(es) can be interpolated in the following ways:

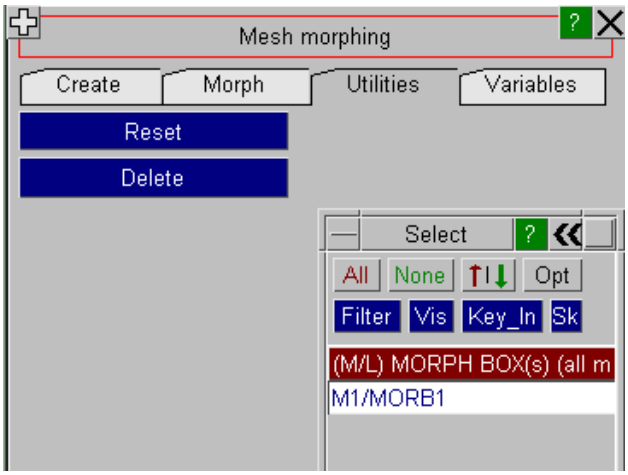
- **No interpolation**: Only the selected morph points move as you drag, and all other morph points remain at the position where they are.
- **Hidden points only**: The selected morph points move by the vector given by dragging or offset vector. The other visible morph points remain where they are. Morph points whose type is turned off at the visibility switches are interpolated. Therefore the interpolation depends on the visibility switches of morph point types here. For example, when corner, edge and face points are visible but internal points are not visible, then only the movement of internal points will be interpolated. While dragging, you will not see the interpolated morph pints moving. This behaviour is motivated by the situation when all visibile morph points on a box are selected and moved by the same vector, in which case it will ensure that the contents of the morph box will translate by that vector without the invisible (usually internal) morph points staying behind.
- **Full interpolation**: Here also the movement of visible morph points is interpolated.

There are also different settings how the interpolations can be calculated. They are on the Program options panel, which can also be accessed with the **Advanced options** button at the top of the panel.

To bring back any morph box to its initial configuration, it is possible to use the **Reset** button in the **Utilities** tab. The initial configuration is written in the *MORPH_BOX keyword after *END, so morph boxes can also be reset to their initial position in future PRIMER sessions when you write the model.

There is also a **Delete** button, which allows to delete morph boxes with a confirmation menu similar to that in the **Remove** tool.

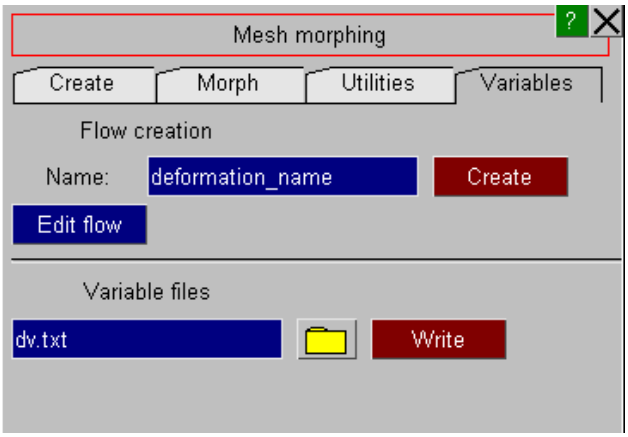
For both Reset and Delete morph boxes can be selected from any model.



The **Variables** tab controls so-called *MORPH_FLOW definitions and design variable files, which can be used to parametrise morphing.

A morph flow is a list of morph points assigning each point a vector. This describes how morph points move when a design variable is applied: For each morph point the vector is multiplied with a variable value (specified later) and then the morph point is displaced by that vector. Furthermore, the morph flow definition contains the following explained below in more detail:

- a unique name used to identify it from the command line
- informaton about variable ranges, which has been added for future development only, but it is not currently used.



The **Edit flow** button allows you to edit the existing morph flow definitions. This opens an edit panel with the morph flow name, variable ranges and the list of morph points with their vectors. Details of the meaning of these keyword fields are described in appendix 18.

Interactive morph flow creation

When you create a morph flow, the following steps are recommended:

- Morph the box(es) how you wish them to look when a variable is applied. This can be done with the interactive functions on the **Morph** tab. Also the numerical offset vector input can be useful to ensure symmetry.
- Go to the **Variables** tab.
- Fill in the unique name for the morph flow.
- Click the **Create** button. This will create a morph flow with the given name containing all morph points which have been moved from their original position. The vector for each point will be that from the original position to the current position. All these vectors will be scaled such that the longest one has got unit length.
- Reset all morph boxes in the **Utilities** tab. If you create multiple morph flows, this avoids unexpected results with the same morph points appearing on multiple flows. Even when the same points appear delibeartely on different flows, resetting boxes ensures that you do not get unexpected vectors from the second morph flow onwards. After creating all morph flows, it is also recommended to reset all morph boxes before writing the model because the current position of morph points in the keyword file will be the initial position from which design variables will be applied.

Design variable files

The morph flow definitions can also be written in a design variable file, which can be passed to PRIMER as a command line argument to apply the morphing.

The design variable file is a JSON text file with the following format:

```
[
 {
  "name": "variable_with_continuous",
  "type": "CONTINUOUS",
  "value": 0,
  "minimum": -10,
  "maximum": 10
```

```
 },
 {
  "name": "variable_with_step",
  "type": "STEP",
  "value": 0,
  "minimum": -10,
  "maximum": 10,
  "step": 2
 },
 {
  "name": "variable_with_discrete",
  "type": "DISCRETE",
  "value": 0,
  "values": [-6, -5, -4, -3, -2, -1, 0, 1, 2, 3, 4, 5, 6]
 }
]
```

This can be passed to the command line as follows:

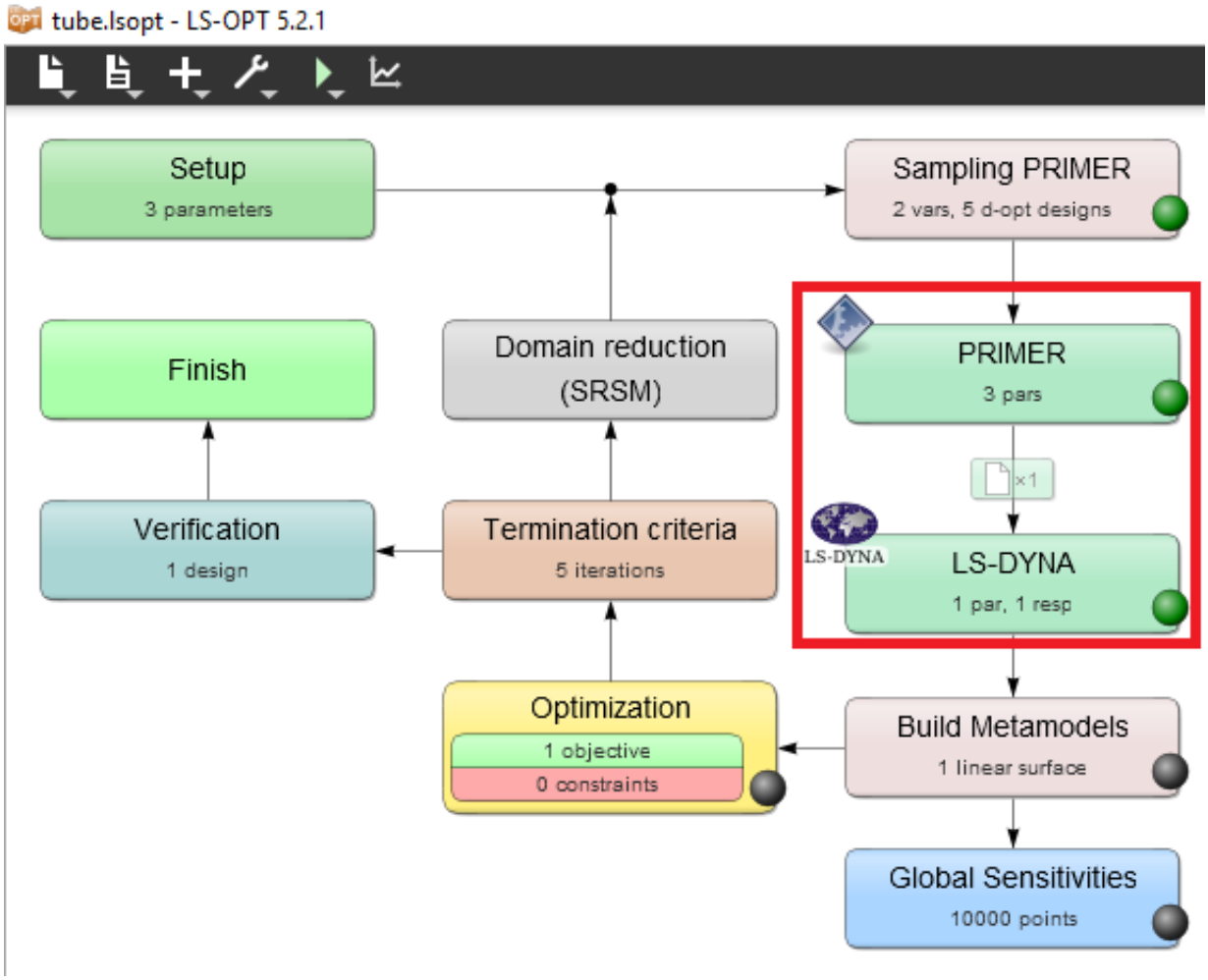
primer.exe input.key -ls_opt=*dv.txt* -js_arg=*output.key* -exit

This command moves each morph point on each morph flow by the specified vector multiplied by the variable value. When the design variable file is written from PRIMER, then the variable values will always be written as 0, but when other values are inserted either manually or by another program like LS-OPT, then the morph points will move as described. Also, the nodes in the relevant morph boxes will move to their position calculated from the morph point positions in the same way as in interactive morphing.

The morphing applied from the command line will only depend on the "name" and "value" properties in the JSON design variable file. The other properties have only been added for future LS-OPT development.

If no output .key file is specified with -js_arg, the input file will be overwritten with the model where the morphing has been applied.

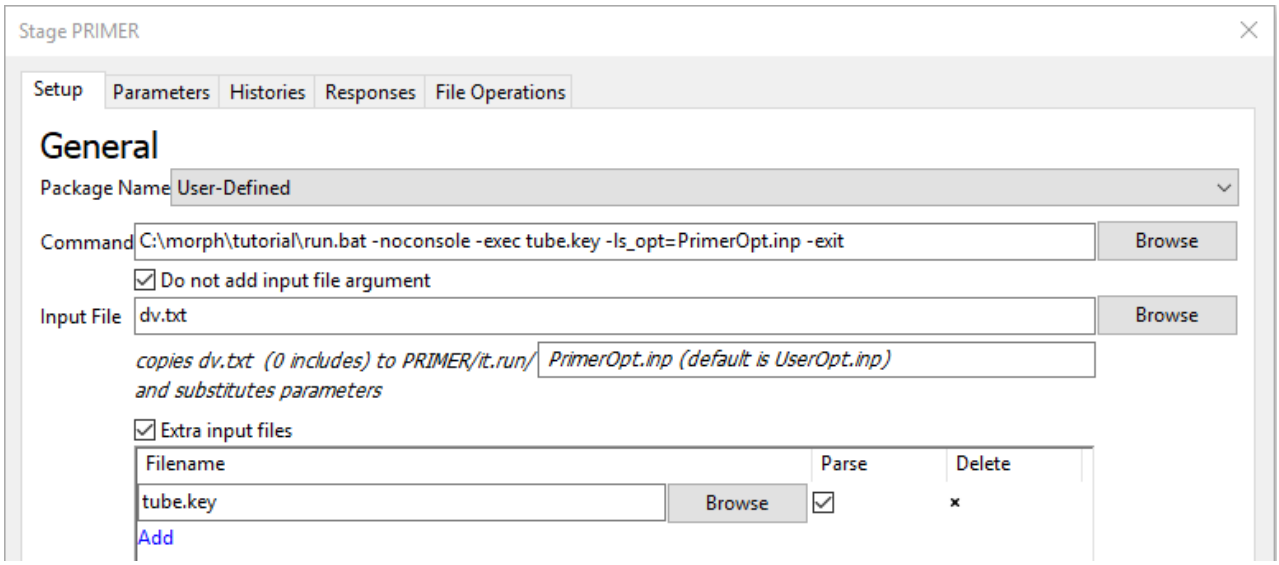
Optimising morphing in LS-OPT



- The flowchart above shows a possible layout of stages in LS-OPT.
- There should be two stages PRIMER and LS-DYNA after each other. When the pre-processor stage PRIMER is defined, LS-DYNA can be added as a new stage in PRIMER, and the stage dependency arrow can then be dragged.
- Below we will see more detail about the setup of the individual stages.

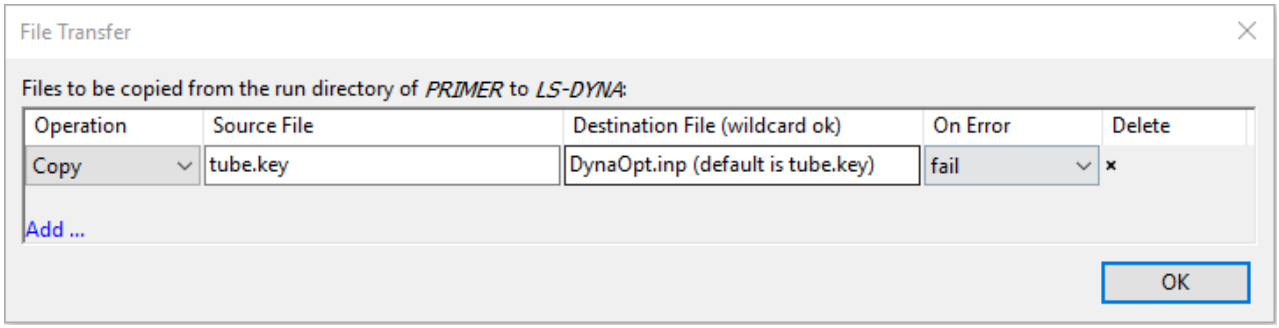
```
[
 {
  "name": "depth_x",
  "type": "CONTINUOUS",
  "value": <<depth_x>>,
  "minimum": -20,
  "maximum": 40
 },
 {
  "name": "depth_y",
  "type": "CONTINUOUS",
  "value": <<depth_y>>,
  "minimum": -20,
  "maximum": 40
 }
]
```

- Edit dv.txt in your working directory such that the "value" properties are <<variable_name>>, i.e. the LS-OPT variable name in <<>> brackets. This variable name can be identical to the morph flow name. Make sure that the line in the JSON file is still terminated by a comma, but there are no quotes except for those around "value".

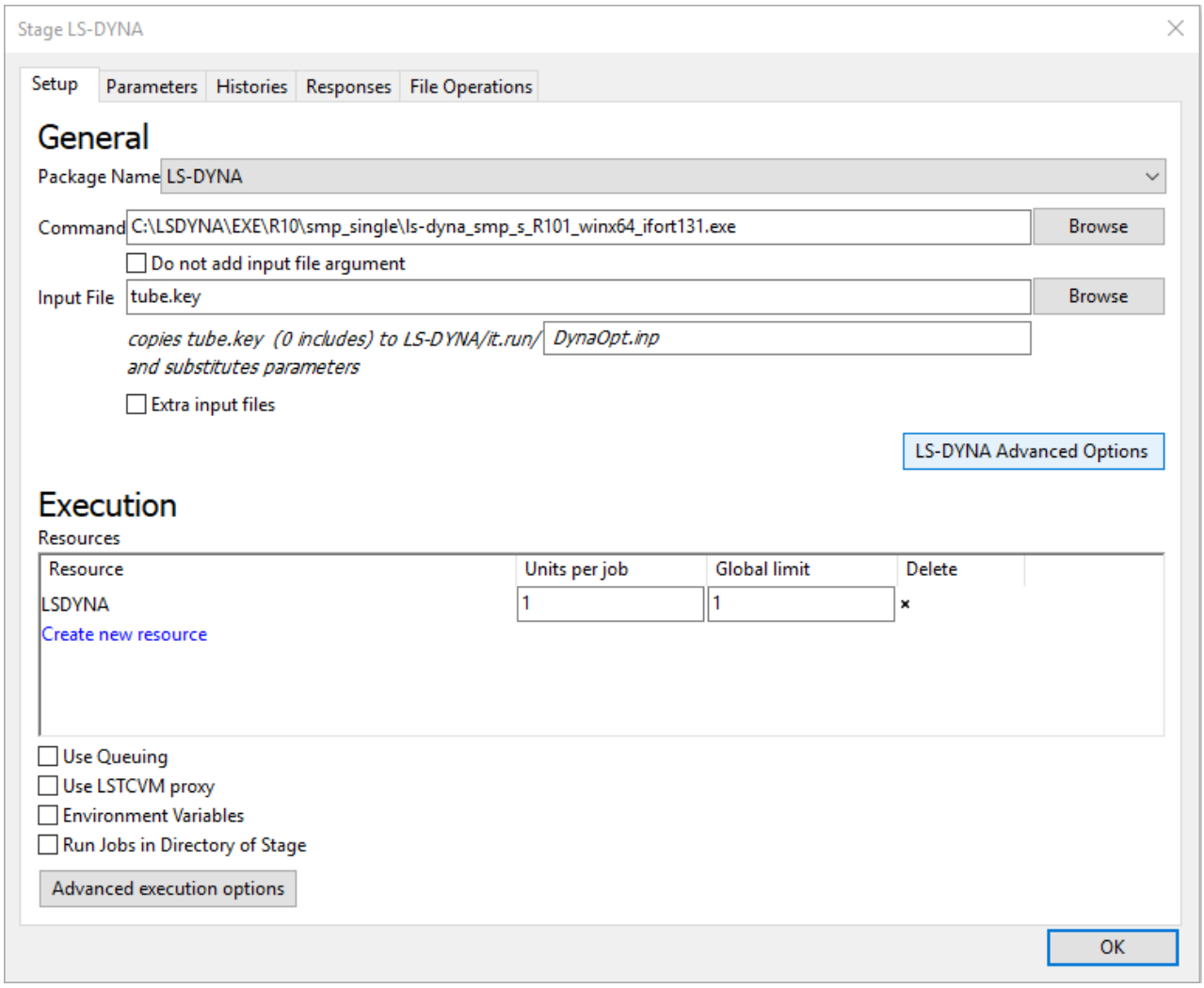



```
@"executable_path\primer16_x64.exe" %*
working_directory\run.bat -noconsole -exec your_input_file.key
-ls_opt=PrimerOpt.inp -exit
```

- When running on Linux, specify the PRIMER executable path directly instead of run.bat and without -noconsole
  `executable_path\primer16_x64.exe your_input_file.key -ls_opt=PrimerOpt.inp -exit`
- Optionally use -batch instead of the -exit for the PRIMER command if you wish to avoid PRIMER windows appearing on the screen.
- Tick "Do not add input file argument".
- As input file specify dv.txt.
- As target on the line "copies dv.txt ...", which by default says UserOpt.inp, specify the file name PrimerOpt.inp.
- Tick "Extra input files" and add your_input_file.key.
- Tick the box to parse that extra input file.



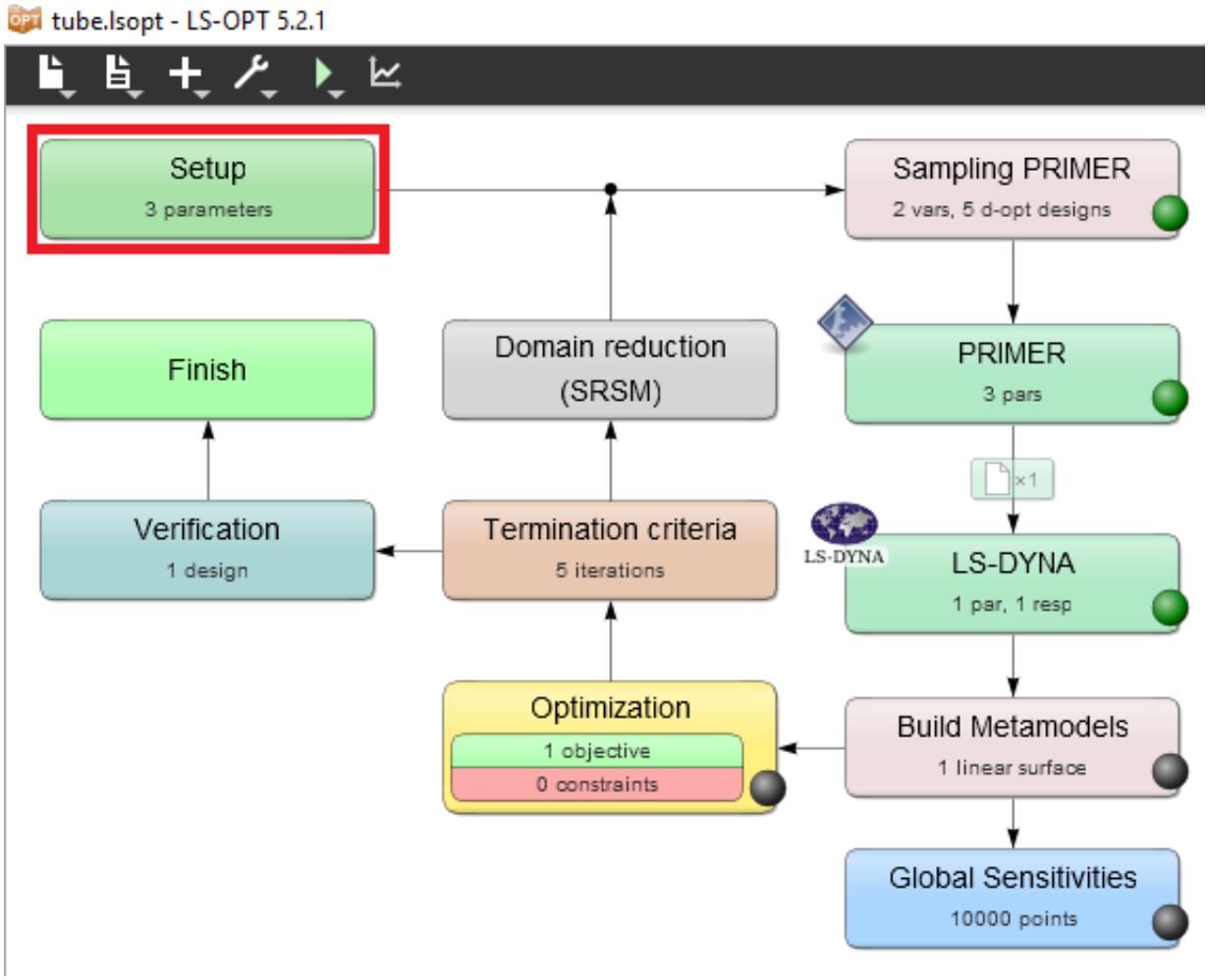
- In the file transfer between the PRIMER and LS-DYNA stages add a file copy action.
- As source file fill in the original .key file.
- As destination specify the file name DynaOpt.inp.
- Make sure that the "On Error" action is set to "fail".

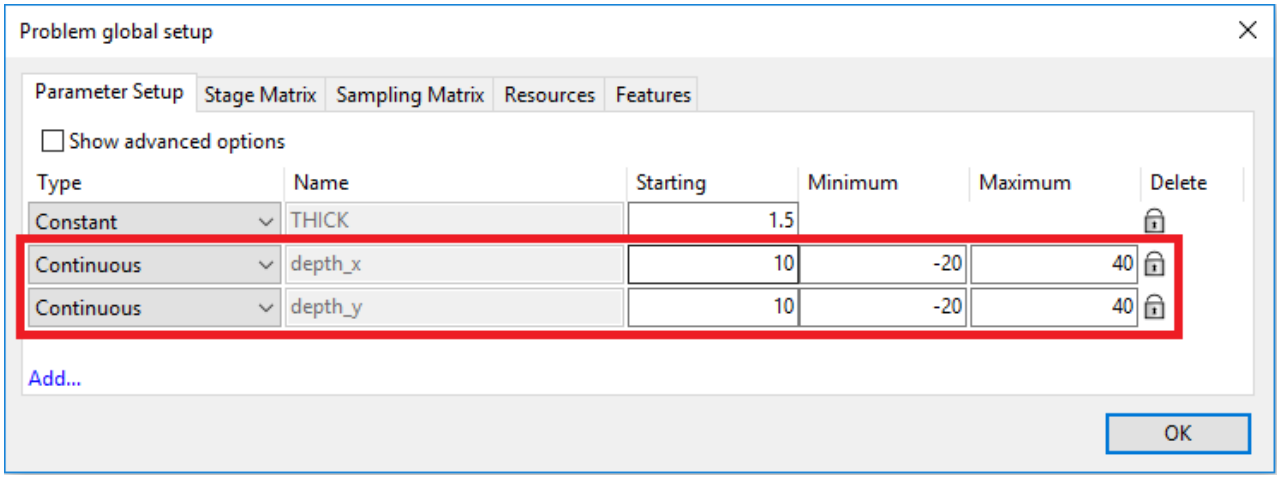


- In the LS-DYNA stage fill in the LS-DYNA executable path.
- As input file specify your .key file without directory path.
- Set the text box to copy the .key file to DynaOpt.inp in each run directory.

In future LS-OPT versions the setup might become easier:

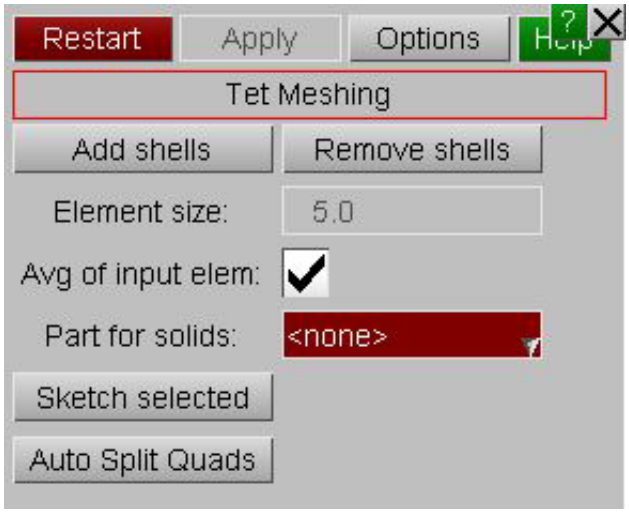
- It will be possible to specify PRIMER directly in LS-OPT in the pre-processor stage definition. This will make the setup of the PRIMER command easier, and on Windows the run.bat files will no longer need to be created manually.
- The variable range information might be picked up automatically by the LS-OPT Setup stage from the JSON files as shown below.


tube.lsopt - LS-OPT 5.2.1
Setup
3 parameters
Sampling PRIMER
2 vars, 5 d-opt designs
Finish
Domain reduction
(SRSM)
PRIMER
3 pars
×1
Verification
1 design
Termination criteria
5 iterations
LS-DYNA
LS-DYNA
1 par, 1 resp
Optimization
1 objective
0 constraints
Build Metamodels
1 linear surface
Global Sensitivities
10000 points


Problem global setup
Parameter Setup
Stage Matrix
Sampling Matrix
Resources
Features
Show advanced options
Type
Name
Starting
Minimum
Maximum
Delete
Constant
THICK
1.5
Continuous
depth_x
10
-20
40
Continuous
depth_y
10
-20
40
Add...
OK

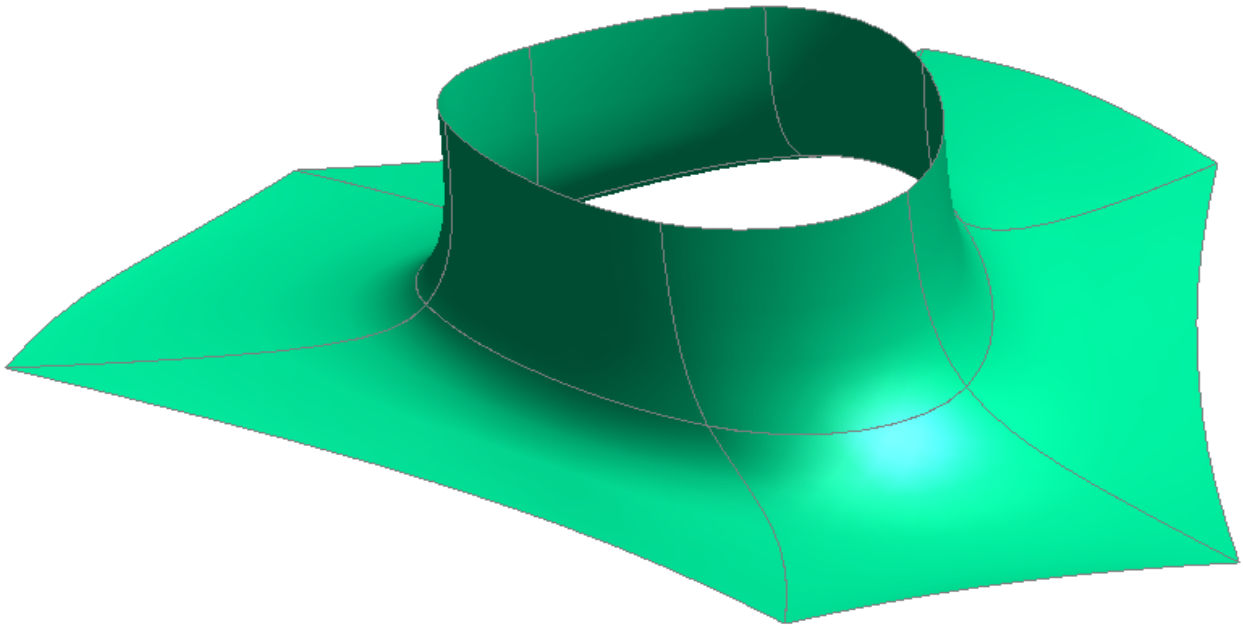
6.31.9 Create tetrahedron mesh

To create a tet mesh, **TetMesh** under Mesh Tools opens a panel as shown on the right. This allows to create tet mesh inside a closed volume of shells. The shells can be selected with **Add shells**. If they are triangular, Primer will create a tet mesh. If some shells are quadrilateral, Primer attempts to split these into triangles automatically if the option **Auto Split Quads** is active.



6.31.10 Mesh geometry

Now in version 16 PRIMER has ability to mesh the selected surface into a single meshed part. For example the image below shows 12 surfaces that we want to mesh. Each of the 12 surfaces will be meshed together so there will be no nodes on all of the shared surface edges.

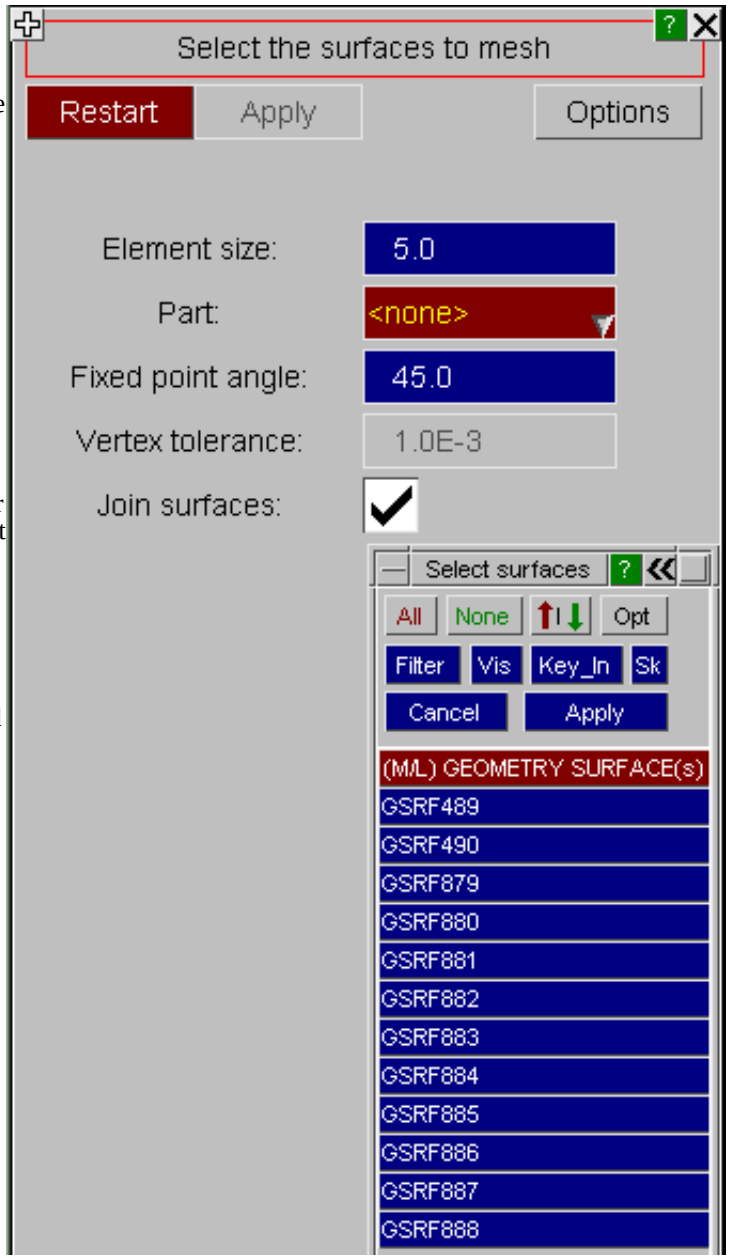


The geometry meshing panel is available in **Mesh tools** under the **Mesh** popup button.

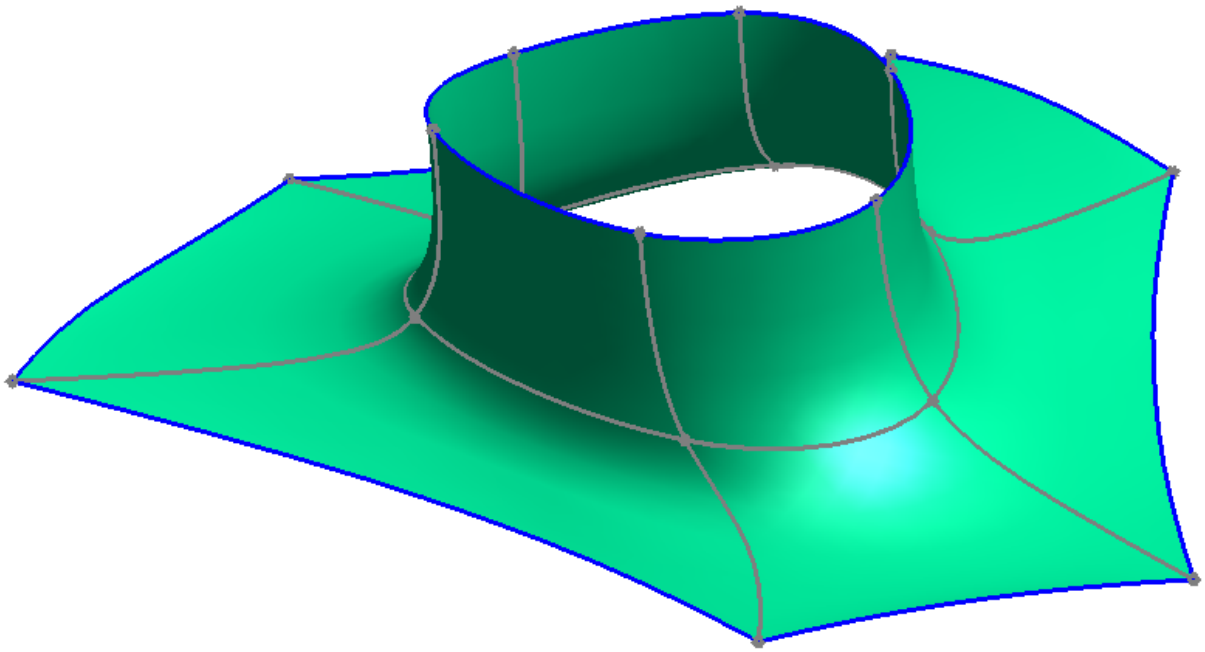
Element size will be the target element size for the generated mesh.
Part is the part ID for elements to be created in.
Fixed point angle specifies the angle at which fixed points will be created on the boundary. For example a flat rectangular surface has 4 corners and the angles at these corners are 90 degrees. Fixed points (PRIMER will ensure than a node is created at that point) on the mesh will be created at these corners if the **Fixed point angle** is less than this.

Vertex tolerance specifies the tolerance to find matching vertices in the selected surfaces. It is not available at first. After selecting the surfaces and followed by "Apply" on the selection panel, vertex tolerance will become active. In case there are minor gaps between the selected surface, provide a relevant **Vertex tolerance** value to perform matching. The maximum tolerance value which can be specified is controlled by the minimum surface length in the selected surfaces.

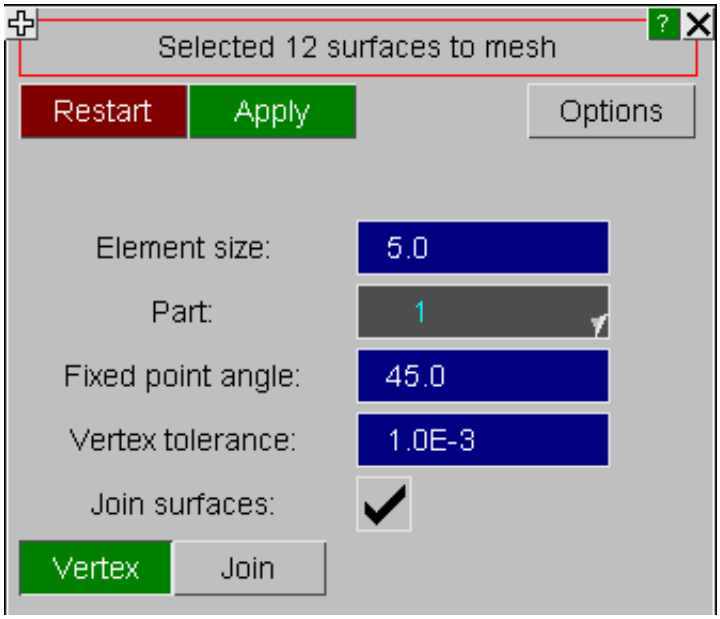
Join surfaces is ticked on by default. This will allow to mesh selected surfaces into a single meshed part. If this option is turned off, each of the selected surfaces will be meshed independently from the others so there will be nodes on all of the surface edges.

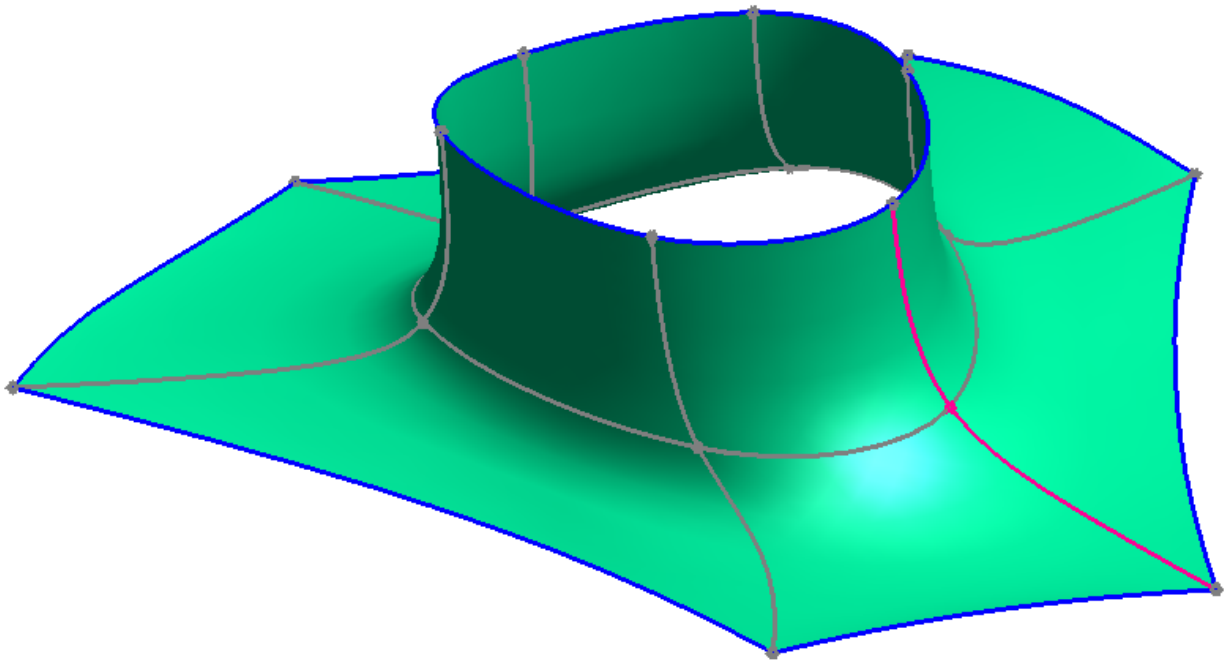


Select the surface(s) that you want to mesh in the object menu (by either picking the surfaces or selecting in the object menu) and press Apply in the object menu. PRIMER will then show the surfaces that will be meshed together with Grey line boundaries and fixed points with Grey circles. Blue line edges between surfaces will indicate that the surfaces will not be joined due to gap/intersection between the selected surfaces. Free edges are also indicated by Blue line.

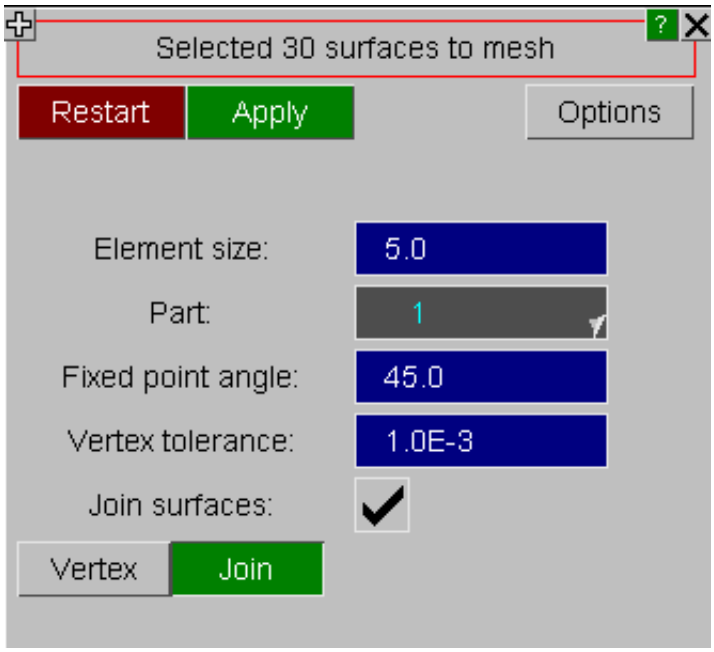


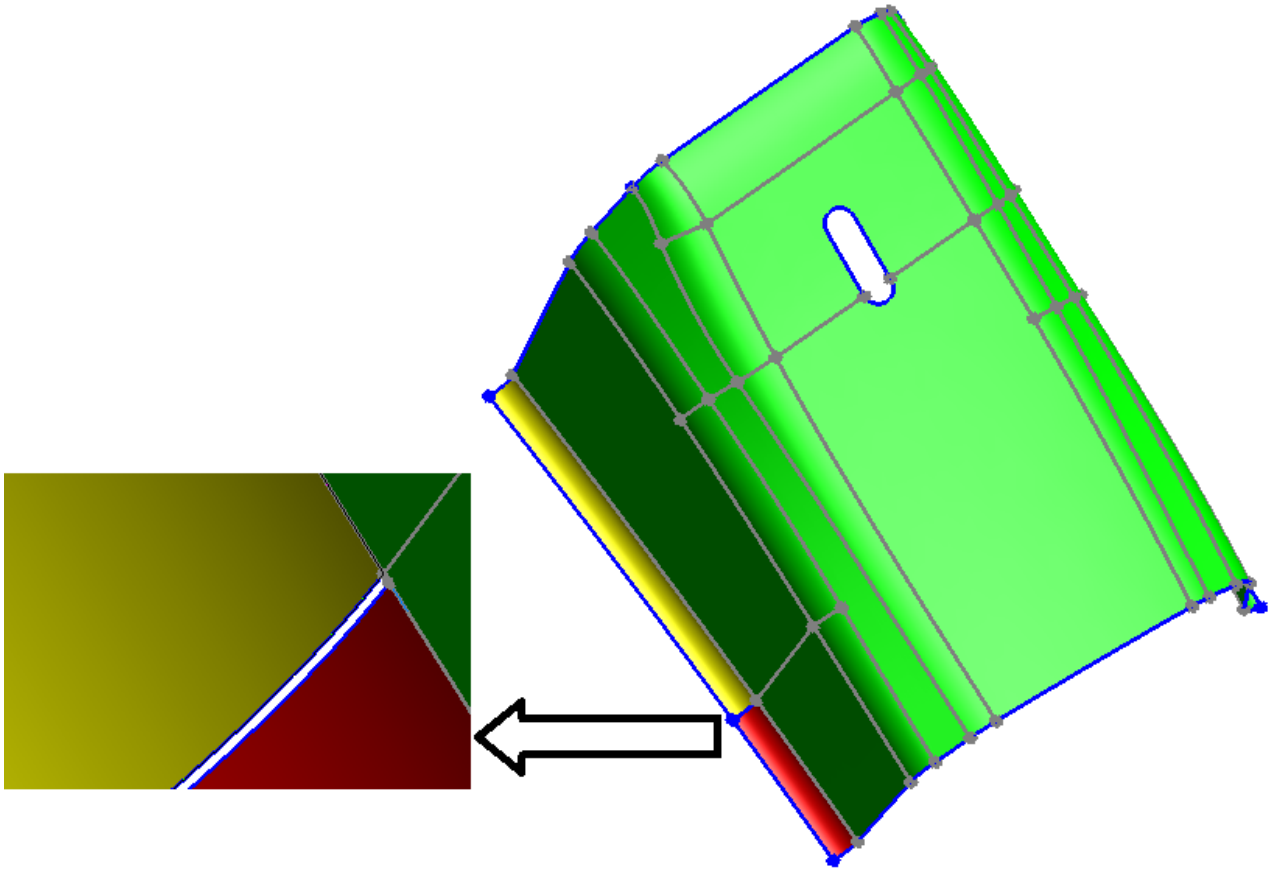
After selecting the surfaces, feature line/points and vertex tolerance can be defined. Feature line and points insure that node will be created on the specified line and points while meshing. Select **Join** to select feature line and **Vertex** to define points. Selection is made using left click on the mouse and right click on mouse is used to deselect. Selected feature lines and points are displayed in Magenta.

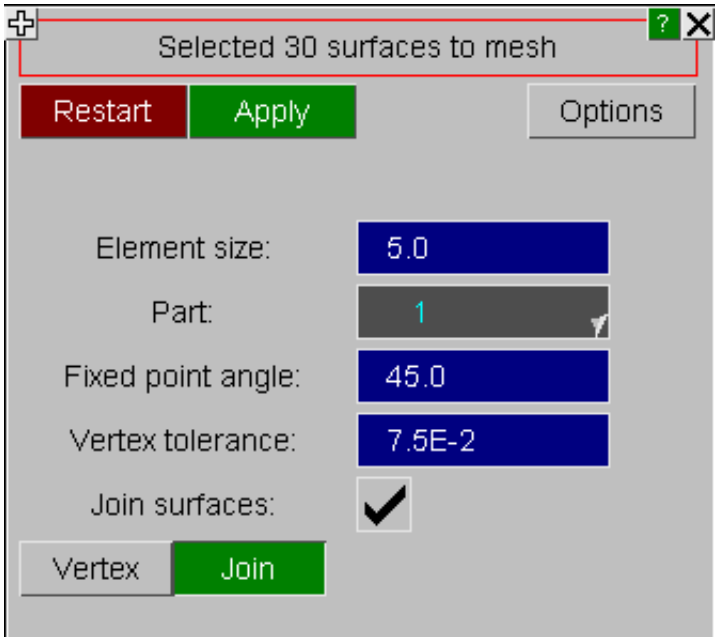


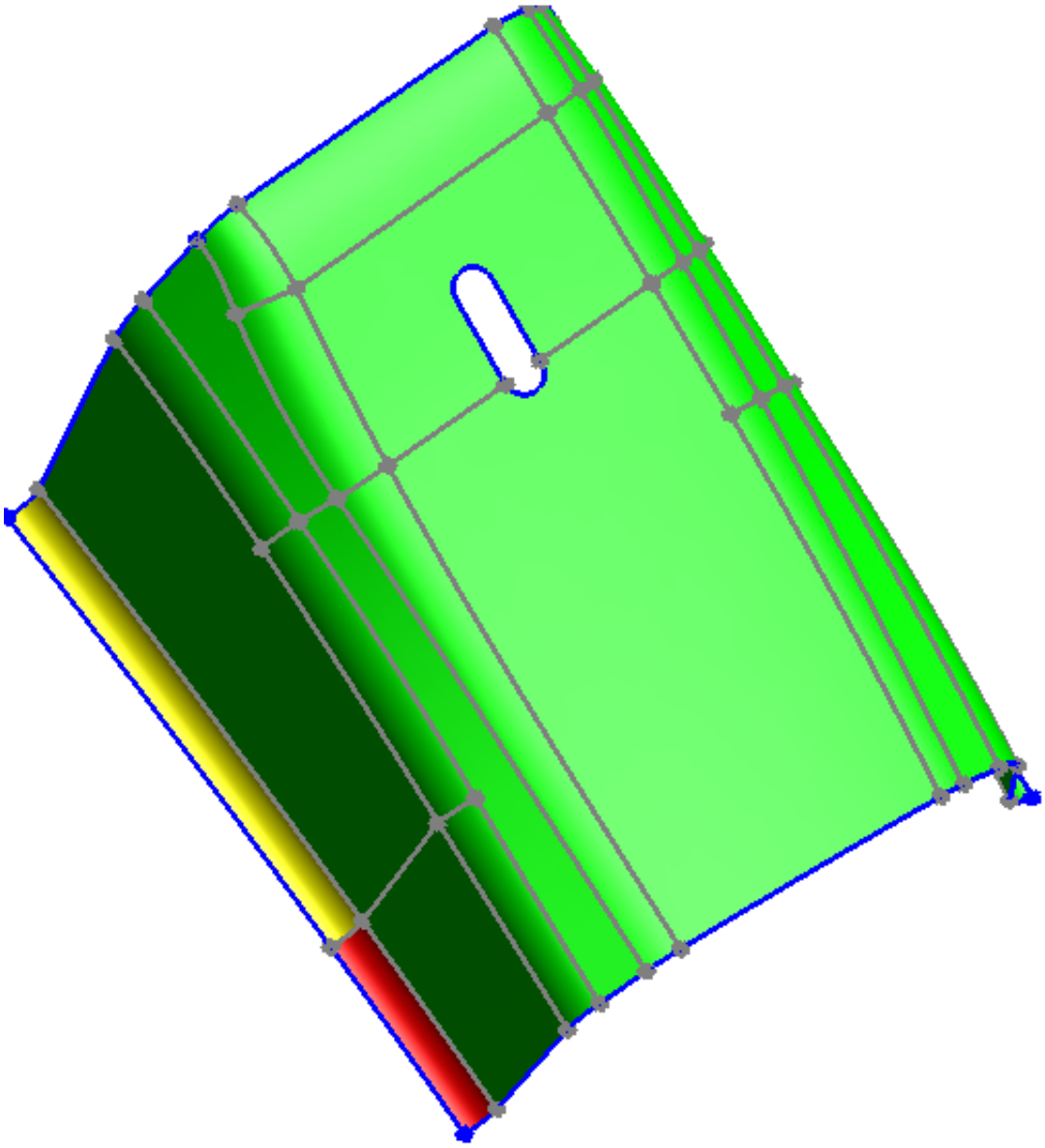


In case there are minor gaps between the selected surface, provide a relevant **Vertex tolerance** value to perform matching. For example the image below shows 30 surfaces that we have selected. A default tolerance of 1.0E-3 is used to do matching. The surfaces which are matched have Grey shared edges. The one which are not matched due to gap like Red and Yellow surfaces have Blue boundaries. There is a gap of 7.48E-2 between Red and Yellow surface. Enter 7.5E-2 as a **Vertex tolerance** and hit Restart. Now these surfaces are matched properly.



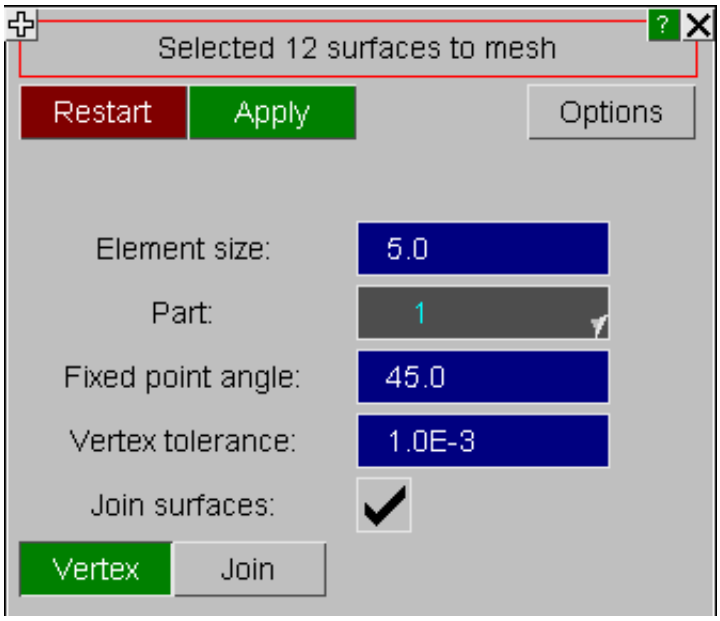


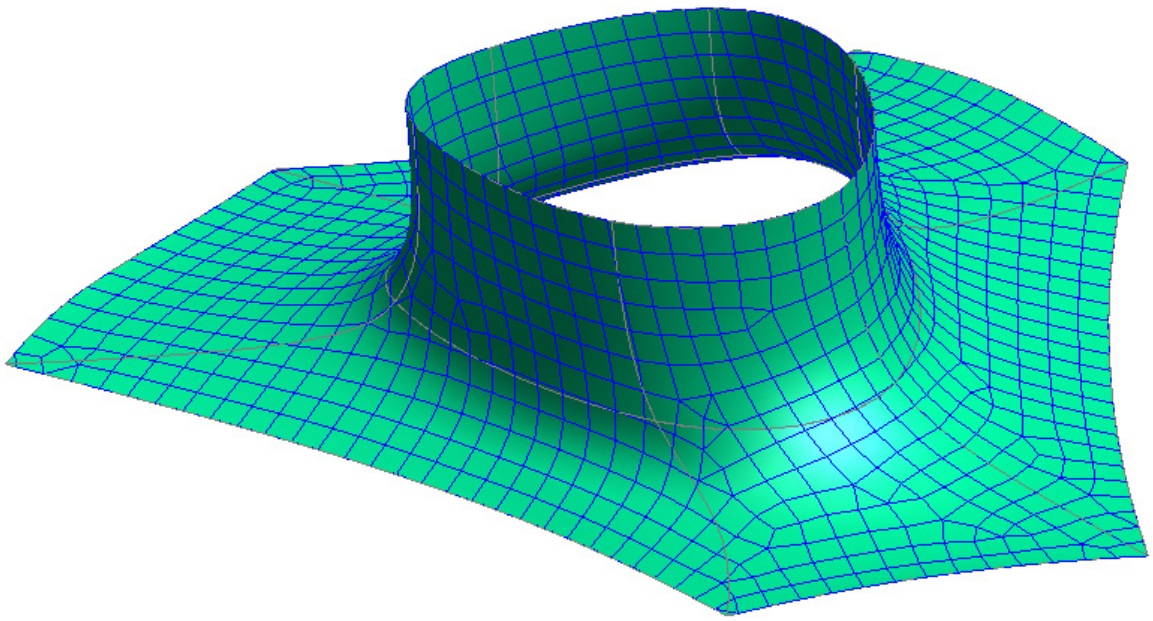

Selected 30 surfaces to mesh
Restart
Apply
Options
Element size:
5.0
Part:
1
Fixed point angle:
45.0
Vertex tolerance:
7.5E-2
Join surfaces:
Vertex
Join



Once the surfaces have been selected and a part ID is given the **Apply** button will become active.

If you press it then PRIMER will show a preview of the mesh that will be generated.

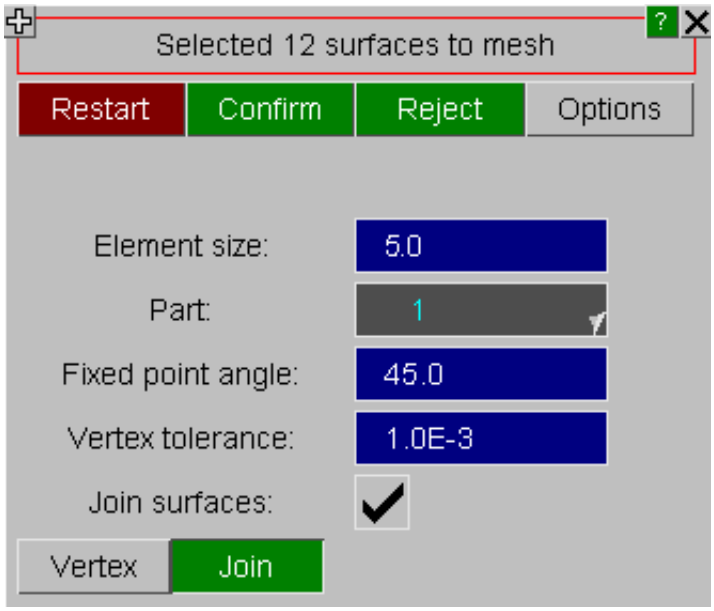


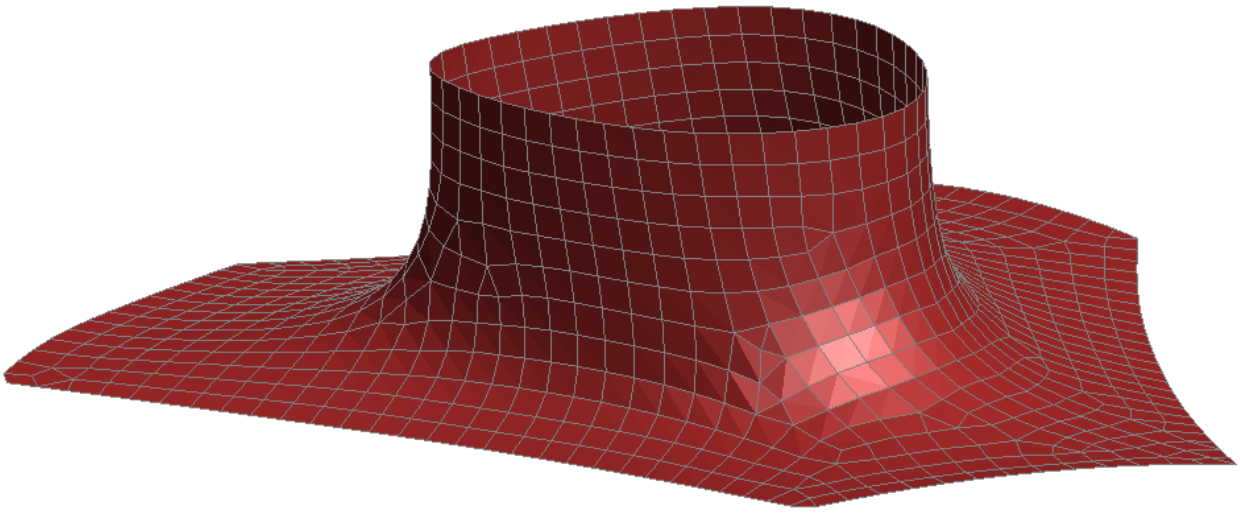


At this point you are free to change the element size, part ID or fixed point angle.

The mesh can be cancelled by pressing **Reject**.

Once you are happy with the mesh press **Confirm** to create it.





6.31.11 Meshing limitations

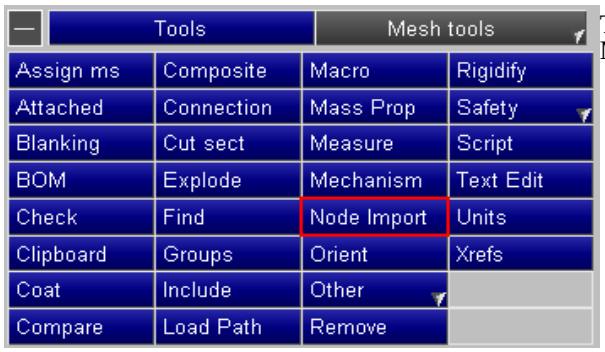
The hole creation, removal and area mesh features in PRIMER were new for version 10.0 and the cobweb mesh was added in version 11.0 . They were an initial attempt to add some meshing capabilities to PRIMER and were quite limited.

In version 13.0 the remesh area feature was completely rewritten and multiple hole removal was added.

The tetrahedral and basic geometry meshing were added in version 14.0

It is expected that the mesh quality and meshing functionality will improve over time with subsequent releases.

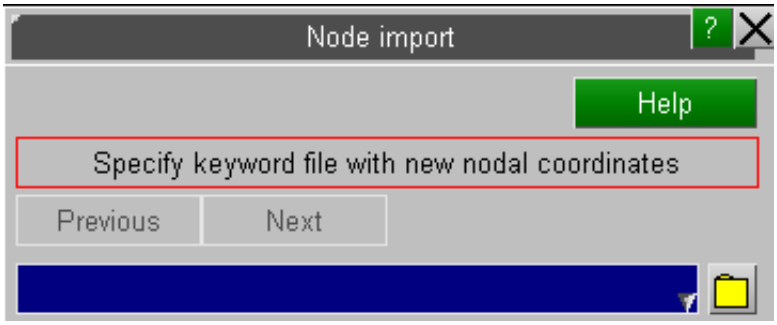
6.32 NODE IMPORT



The **NODE IMPORT** function is a tool available from the MAIN top box, to give the master panel shown in this figure.

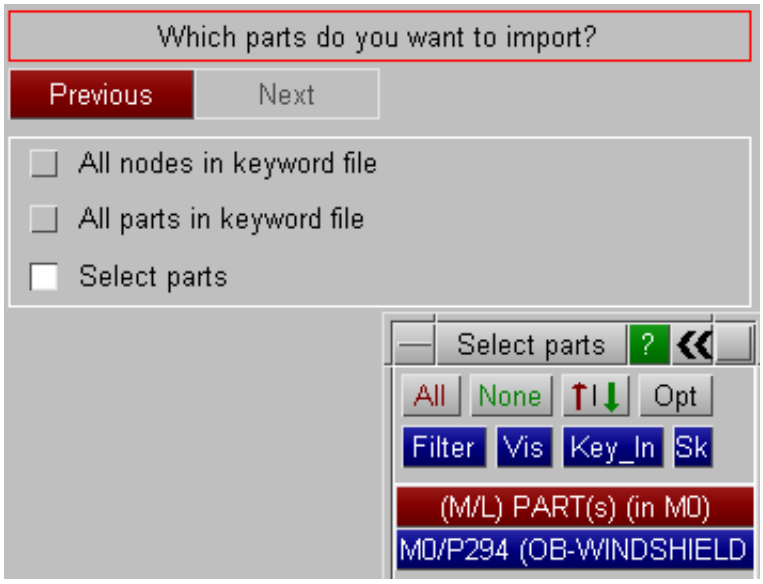
Step 1

The nodal coordinates of the model will be replaced with those from another keyword file, which is selected in this step. After the file selection press **Next**.



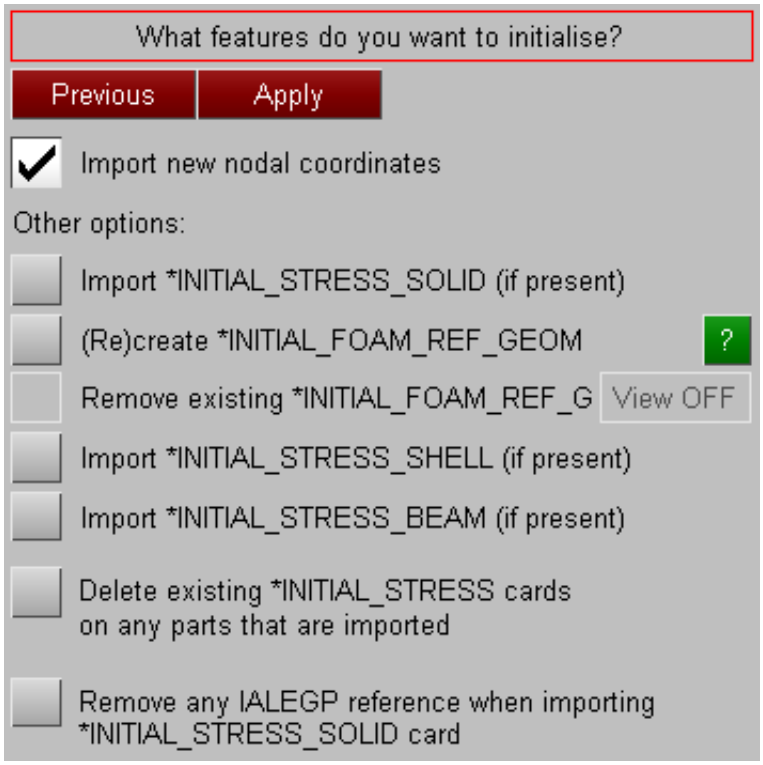
Step 2

You can now select to import the new nodal coordinates from all nodes in the keyword file, all parts, or just a selection of parts. Press **Next**.

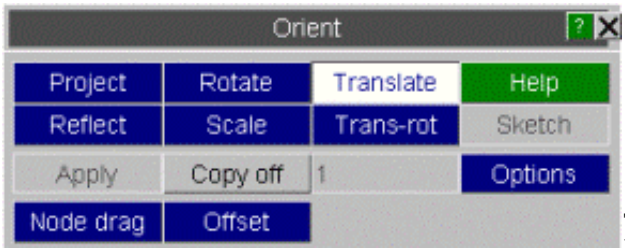


Step 3

By default only the new nodal coordinates will be imported. There are other options which data can be updated in addition to the new coordinates. Make your selection and click **Apply**.



6.33 ORIENT Translating, rotating, scaling, reflecting, projecting



The **ORIENT** command is invoked from the MAIN top box, to give the master panel shown in this figure.
There are currently six types of orientation available:

TRANSLATE	shift by global vector, n1->n2, or normal to plane
ROTATE	rotate about global or local axes
REFLECT	reflect about a distance [d] down a given axis.
SCALE	factor nodal and other coordinates by [Sx,Sy,Sz]
PROJECT	project nodes to line or plane or mesh surface
TRANS-ROT	translate and rotate in one operation

There are also links to two other orient related features in PRIMER:

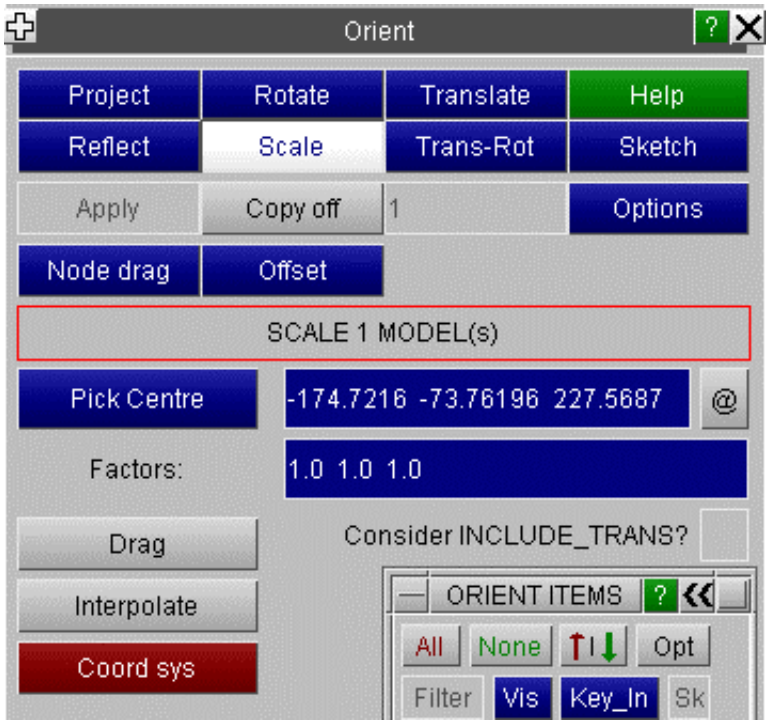
NODE DRAG	drag node interactively and mesh optimisation
OFFSET	offset shells

Normally just the selected items are oriented by the amount specified. It is also possible to **INTERPOLATE** movement to achieve other effects.

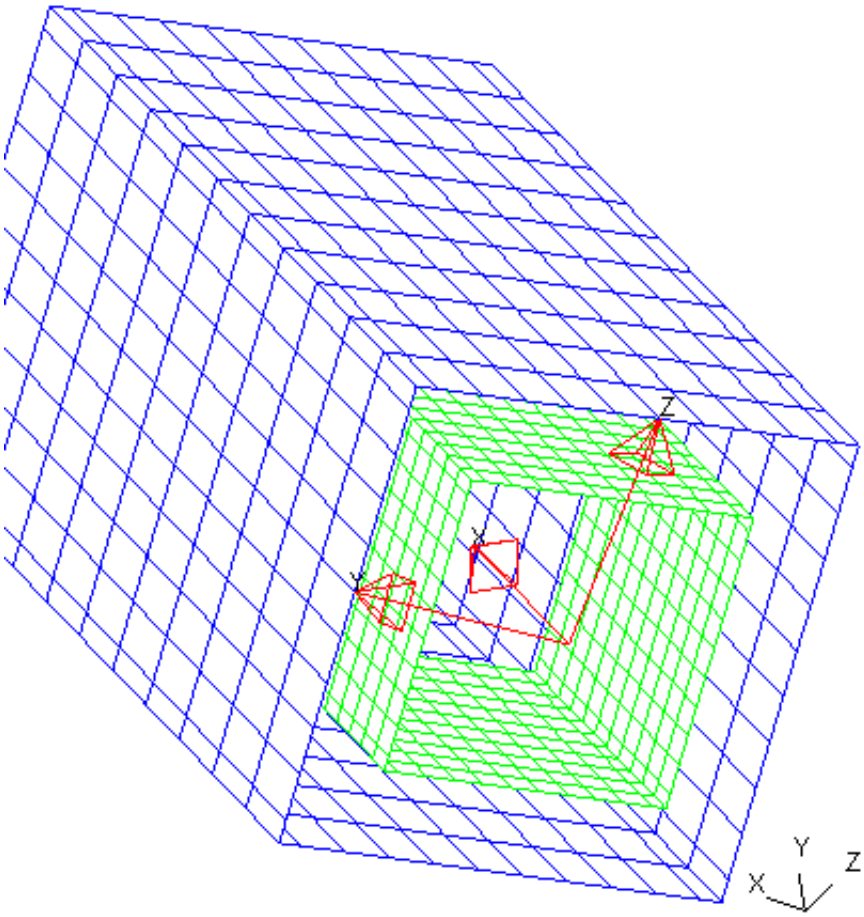
Orient in a local system

Orients of types translate, rotate, reflect and scale may be performed in a local coordinate system by selecting a *DEFINE_COORDINATE_SYSTEM that exists in the model. Some of the older functionality for local rotation and translation is now obsolete, but has been left for those that are used to it.

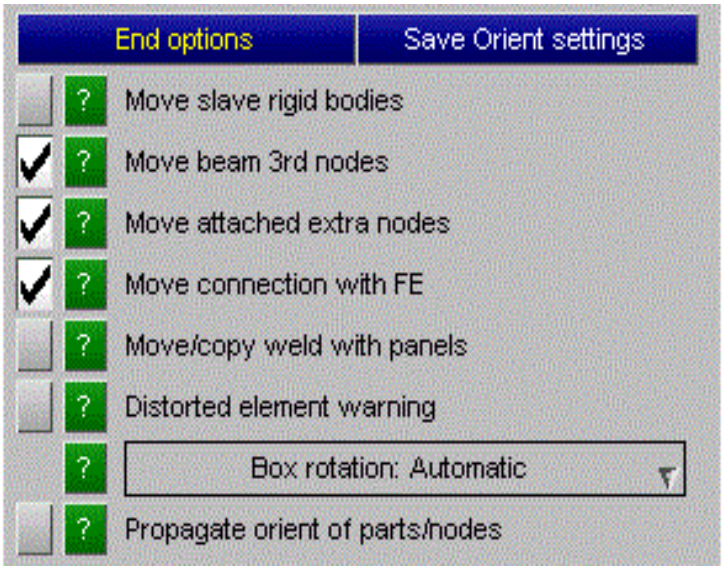
Pressing **Coord sys** button will put up the object menu. Once selected the button will be red and local orientation will persist until the same button is pressed to de-activate it or the orient panel is dismissed. The local system will be applied to orient drag operations where DoF filter may be used to control local motion.



In this example a rectangular tube has been scaled in its local YZ axes to enlarge it and then along its local X axis to lengthen it.



Options for Orient



The **Save Orient settings** button saves the following options to the oa_pref file.

Slave rigid bodies may be implicitly oriented when their master part is. The default is not to orient.

Beam 3rd nodes (which define the beam section) by default will be oriented when the beam itself is.

When a rigid part is oriented, by default the constrained extra nodes will also be oriented. This may distort deformable elements.

When all FE entities of a connection are oriented, by default the connection itself will also be oriented.

There is an option to move (or copy) welds that attach to panels which are being oriented.

There is an option to report that nodes of unselected elements have been moved as a result of the orientation as this may have caused distorted elements.

Box rotation by default is automatic, which means that if dyna output version is set to 971R6 or above a rotated box will be converted to _LOCAL. Otherwise the old method of enlarging rotated boxes will apply.

Propagation of orient is no longer applied unconditionally to all "junior" items that are cross-referenced by the items selected for orient. This gave inconsistent results when parts or nodes were selected as such items as *BOUNDARY may or may not be oriented depending on their configuration. By default, Primer will limit propagation from parts and nodes.

6.33.1 TRANSLATE Shifting by [dx,dy,dz]

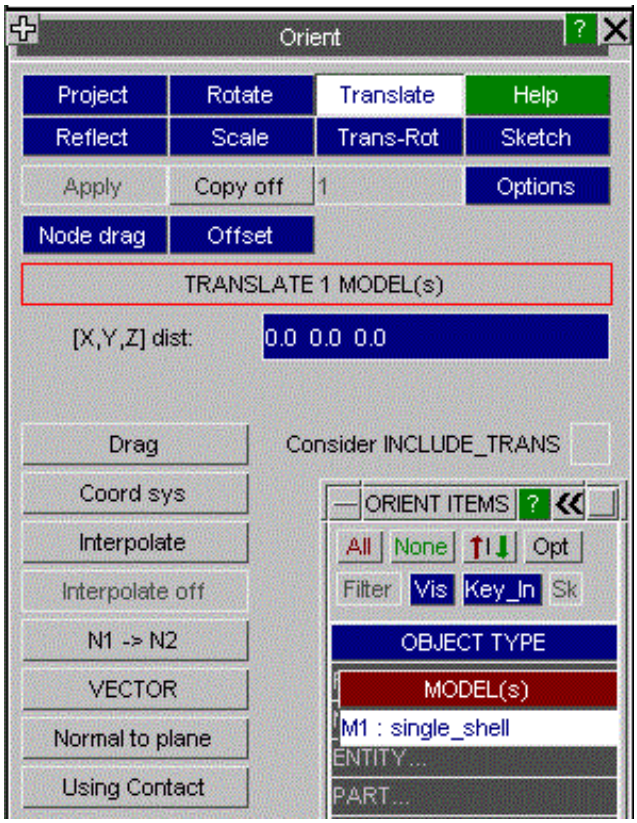
On entering TRANSLATE you must first select the objects to be moved, then

- enter a global translation vector [X, Y, Z]
- define a vector n1->n2 and a distance
- define a plane with 3 nodes (or pick a shell) and a distance for normal projection.

APPLY will update the nodal coordinates.

When a transformation is applied the image is redrawn so that you can see what the result looks like, and you are given the options of accepting, rejecting or repeating the transformation before it becomes permanent.

If you reject an orient, the nodal coordinates are restored from a backup cache, so no rounding error is incurred.



INTERPOLATE is described in section below.

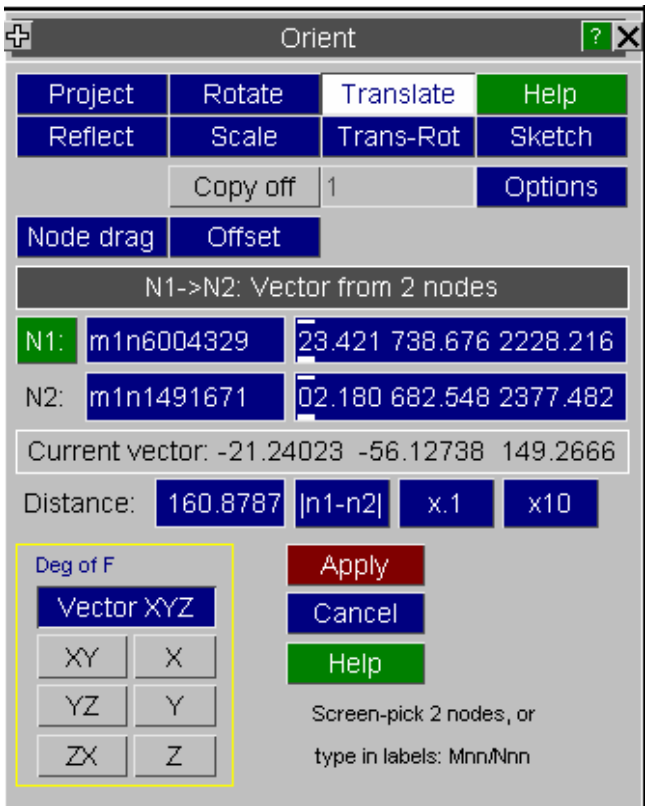
Alternative ways of defining a translation distance.

N1 -> N2 Using the vector between two nodes.

In this method you select two nodes: either by screen-picking them or by typing their labels into the relevant boxes in format M<model number>/N<node label>. Alternatively, nodal coordinates may also be specified directly. The coordinates of the second node may optionally be kept as the default value of (0, 0, 0) thereby permitting easy translation of entities to the origin. The vector is computed from the coordinates of N2 - N1 and the distance set.

You can choose the degrees of freedom of this vector to use. By default **VECTOR_XYZ** is in force, meaning all of the [x,y,z] components, but you can reduce this to two or one component only using XY, ... Z.

When you have obtained the desired vector use OK to return to the main TRANSLATE panel, where you can then **APPLY** it.

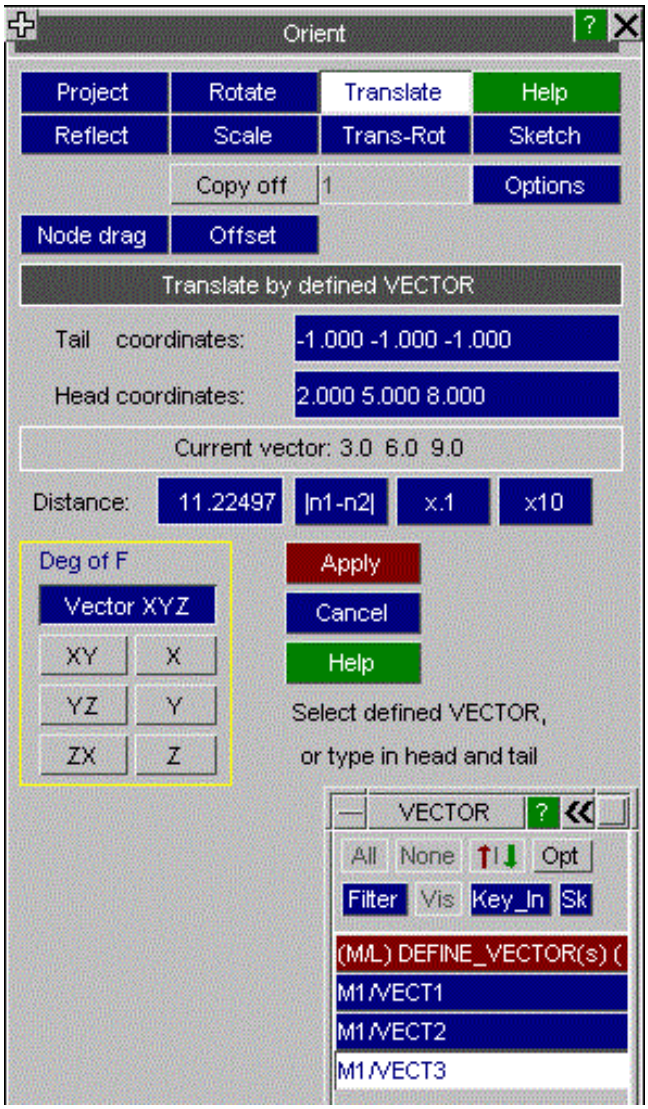


VECTOR Using DEFINE_VECTOR(s)

In this method you select a defined VECTOR from the object menu. Alternatively, the head and tail coordinates may also be specified directly. The head coordinates may optionally be kept as the default value of (0, 0, 0) thereby permitting easy translation of entities to the origin. The vector is computed from the head and tail coordinates as well as the distance set.

You can choose the degrees of freedom of this vector to use. By default **VECTOR_XYZ** is in force, meaning all of the [x,y,z] components, but you can reduce this to two or one component only using XY, ... Z.

When you have obtained the desired vector use OK to return to the main TRANSLATE panel, where you can then **APPLY** it.

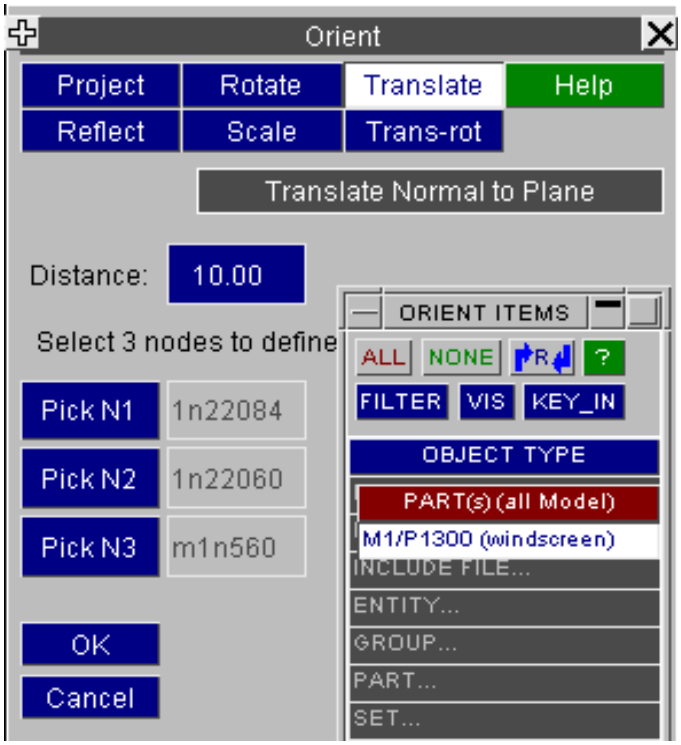

Orient
Project
Rotate
Translate
Help
Reflect
Scale
Trans-Rot
Sketch
Copy off
1
Options
Node drag
Offset
Translate by defined VECTOR
Tail coordinates:
-1.000 -1.000 -1.000
Head coordinates:
2.000 5.000 8.000
Current vector: 3.0 6.0 9.0
Distance:
11.22497
|n1-n2|
x.1
x10
Deg of F
Vector XYZ
XY
X
YZ
Y
ZX
Z
Apply
Cancel
Help
Select defined VECTOR,
or type in head and tail
VECTOR
All
None
Opt
Filter
Vis
Key_In
Sk
(M/L) DEFINE_VECTOR(s) (
M1/VECT1
M1/VECT2
M1/VECT3

NORMAL TO PLANE

Select the items to translate, and click on the **NORMAL_TO_PLANE** option.

Define the plane by picking on 3 nodes and set the translation distance.

Use **OK** to return to the main TRANSLATE panel, where you can then **APPLY** the orient.



Translate Using Contact

With this function selected parts (typically an impactor or barrier model) may be translated along a defined slide vector until they are brought into position or depenetrated (if initially penetrating).

Contact part(s) on the main model (typically the vehicle) must be selected from the object menu of parts/part sets (the target side). The orientee items and the part do **not** need to be in the same model as this function has special logic to create contacts across models.

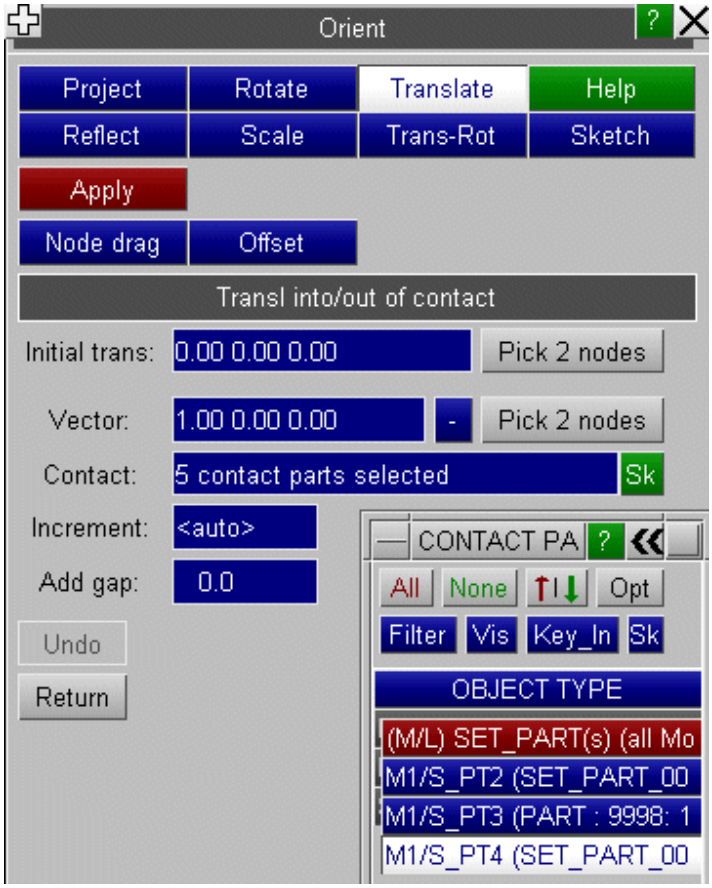
If initial penetration is detected, items will be moved *against the direction of slide vector* until depenetrated. If not, they will be moved *in the direction of the slide vector*.

The increment for each iteration may be set by the user, although the automatic method should work for most models.

Add Gap option will impose a final translation against the direction of the slide vector.

The default contact method is AUTOMATIC_SURFACE_TO_SURFACE. However if the target (master) side contains beam parts (excluding spotwelds) an AUTOMATIC_GENERAL single surface contact will be used with additional logic to exclude self contact between parts of the target side.

Green **Sk** button will provide red (orientee slave parts) / green (target master parts) plot to show the contact that will used.

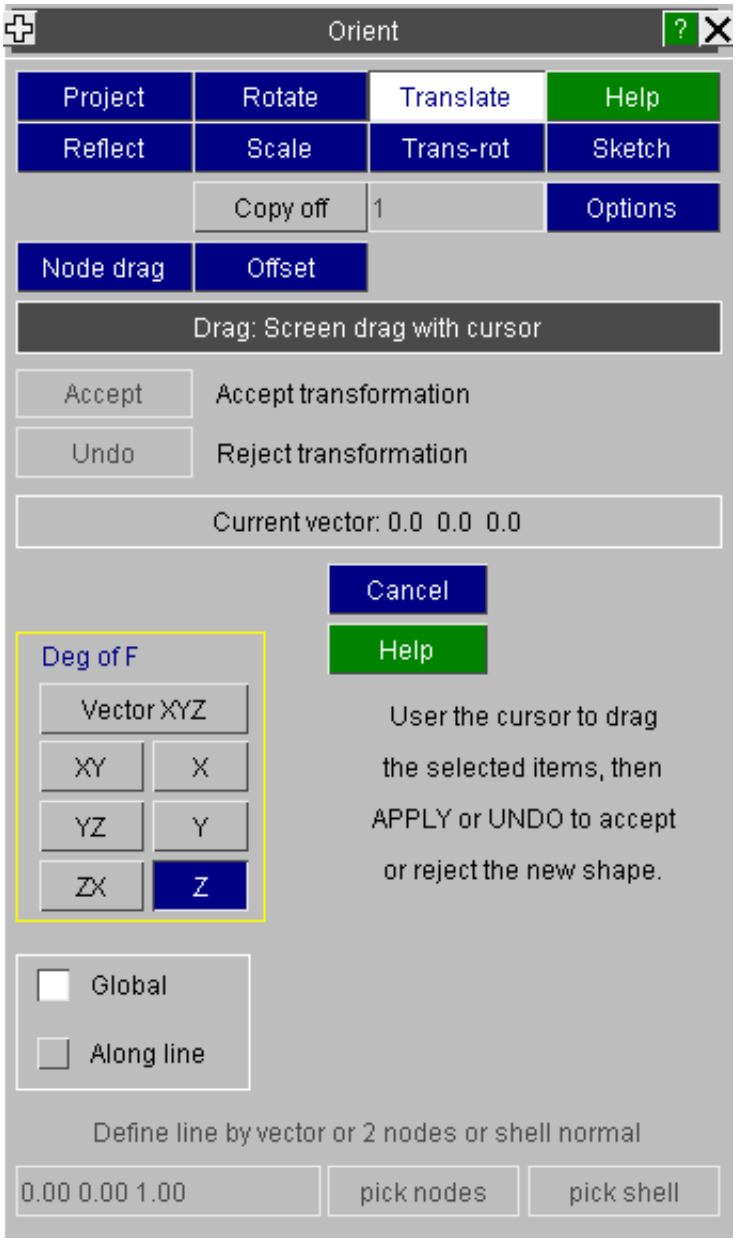


DRAG TRANSLATE Using the cursor to "drag" objects.

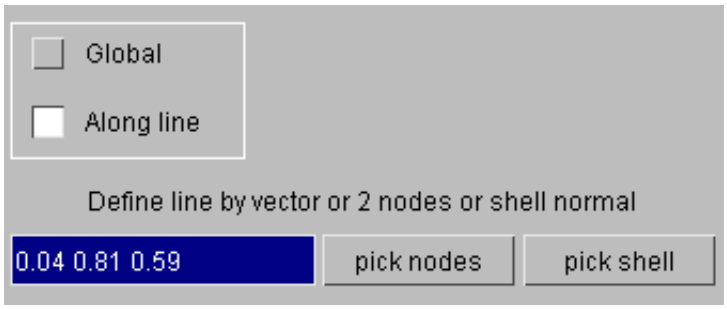
Click down the left mouse button at any point on the screen (it's not related to the object) and drag it in the desired direction. The object, as a reduced set of vectors if it is large, will follow the mouse across the screen, stopping when you release the mouse button.

Then use **APPLY** to accept the transformation, or **UNDO** to reject it and restore the status quo ante.

Drags take place in the plane of the screen, so the actual [x,y,z] vector will depend on the current view. It is strongly recommended that you use one of the XY ... Z options to limit object motion to either a plane or a single vector.



The motion may be limited to an arbitrary vector by switching to **Along line** mode. The vector may be typed in, defined by 2 node picks or the normal of a picked shell.



On completion of drag, the new position may be accepted or rejected.

6.33.2 ROTATE Rotating by x,y,z

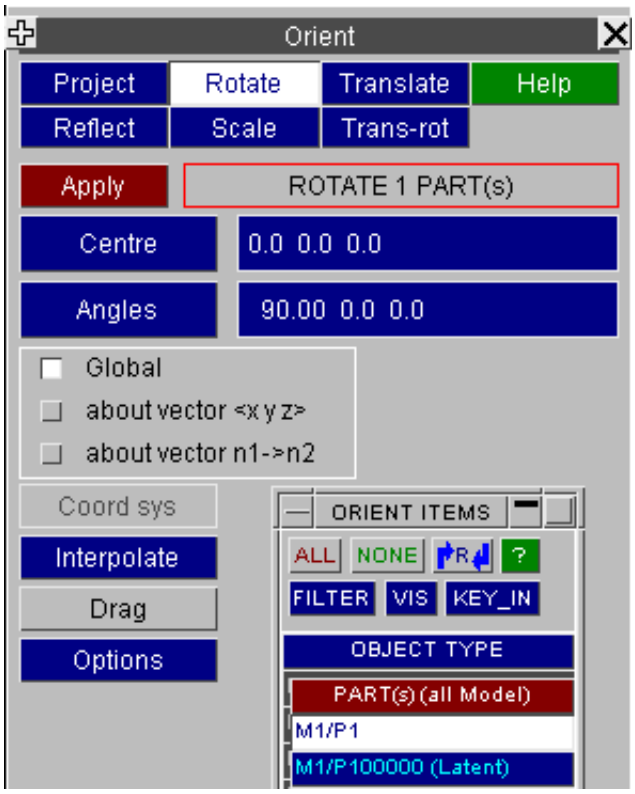
On entering ROTATE you must define:

- The objects to be rotated;
- The centre about which to rotate;
- either the rotation angles (**GLOBAL**_option) or a single angle and a local axis (**about_vector**_option)

Local axis of rotation may be defined by a vector <x y z> or two nodes.

When these have been entered press **APPLY** to make them take effect.

As with TRANSLATE when the rotation is applied the image is redrawn so that you can see what the result looks like, and you are given the options of accepting, rejecting or repeating the transformation before it becomes permanent.



CENTRE: Defining the rotation centre

Instead of typing in an [x,y,z] coordinate you can use **CENTRE** to pick a node to be used as the rotation centre. The node's coordinate will be placed in the "centre" box for you.

For rotate and for scale, if a single part is selected the useful option **centre @ part CofG** will become active.

ANGLES: Defining rotation angles

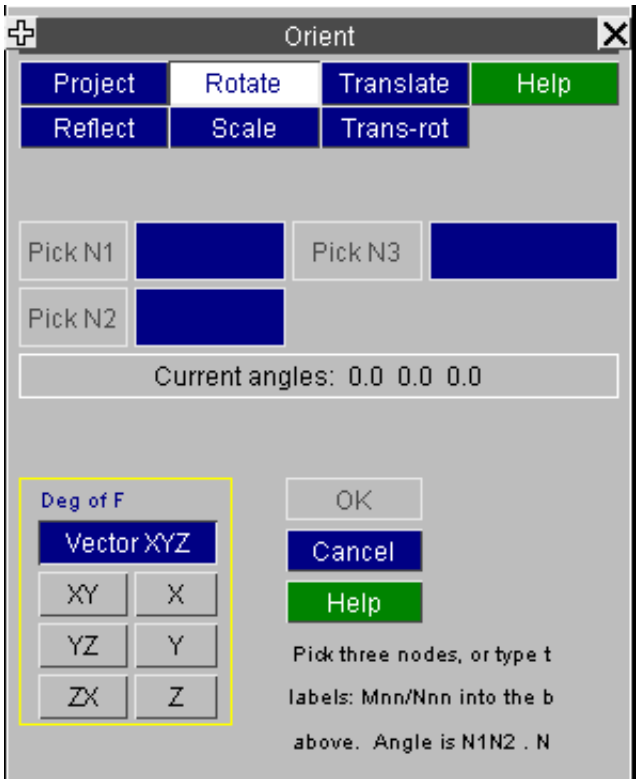
As an alternative to typing in angles about the [x,y,z] axes you can use the **ANGLES** command to calculate the angle between 3 nodes, as shown in this figure.

Pick, or type in the labels of, 3 nodes.

The vectors N1N2 and N1N3 are computed, and then the angle between them.

You can choose the 3D angle, or the projected value about any axes using **XY, ...Z** as before.

When the angle is satisfactory use **OK** to return to the main ROTATE box where you can then apply it.



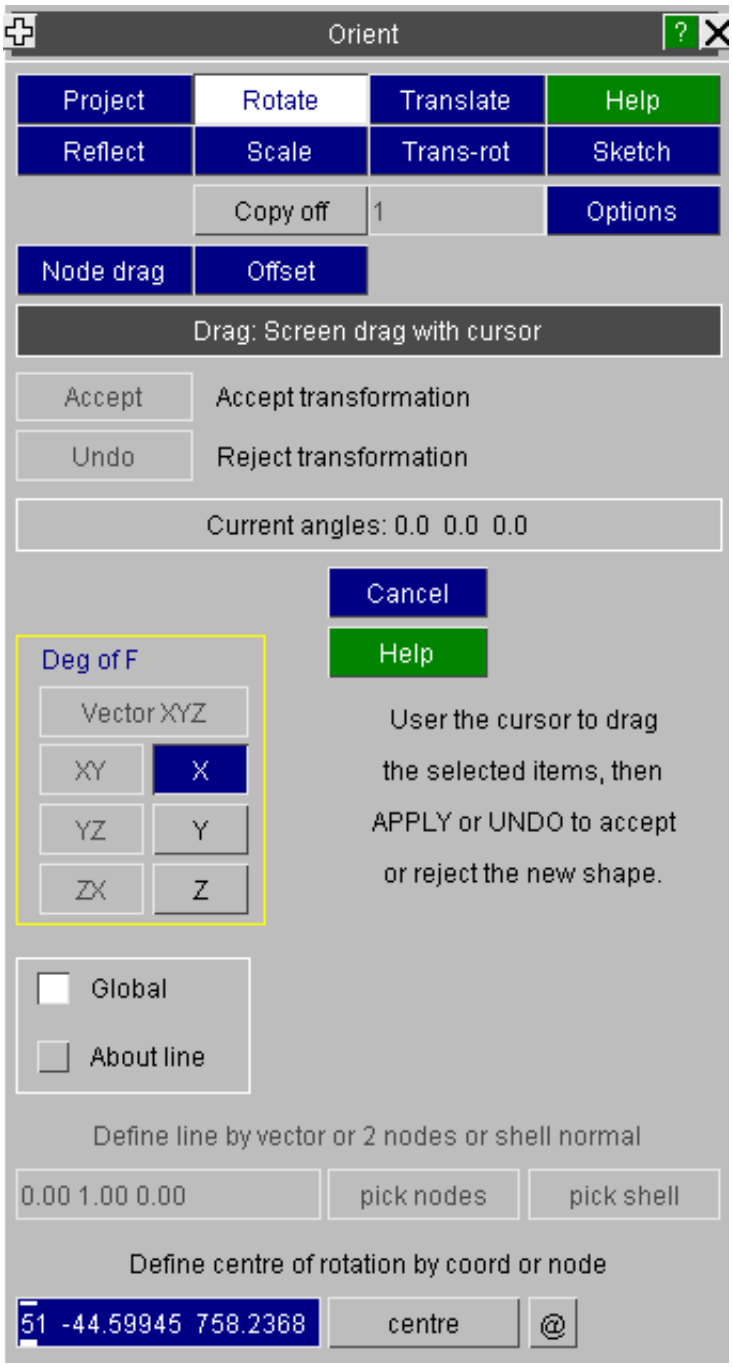
DRAG ROTATE Using the cursor

As an alternative to specifying rotation angles you can "drag" the objects, about the defined centre, using the **DRAG** option shown in this figure.

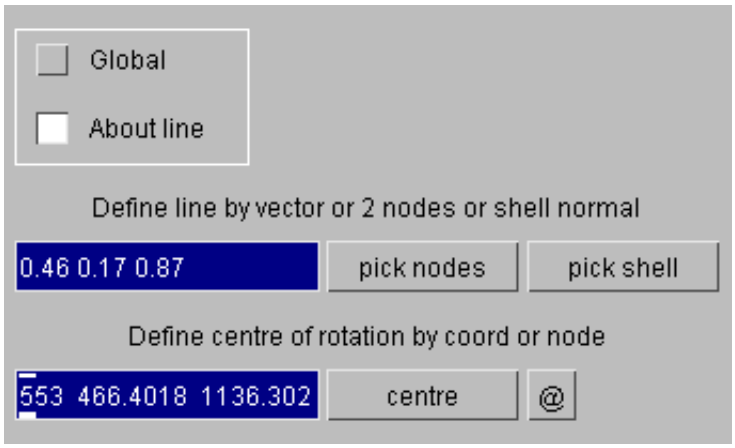
Place the cursor anywhere on the graphics screen, press the left mouse button and, keeping it depressed, move it until the desired position is reached, then release it. The image (or a subset of it if it is large) will move across the screen, and then be redrawn at the new position.

Use **APPLY** to make the change permanent, or **UNDO** to reject it and return it to how it was before. (Using **CANCEL** also implicitly rejects any dragged rotations.)

Dragging can only take place about one axis at a time, the default being global **X** as shown here. The centre of rotation may be typed in or set by a node pick. The @ button will sketch the current centre.



The **About line** option will allow a rotational drag about an arbitrary line.

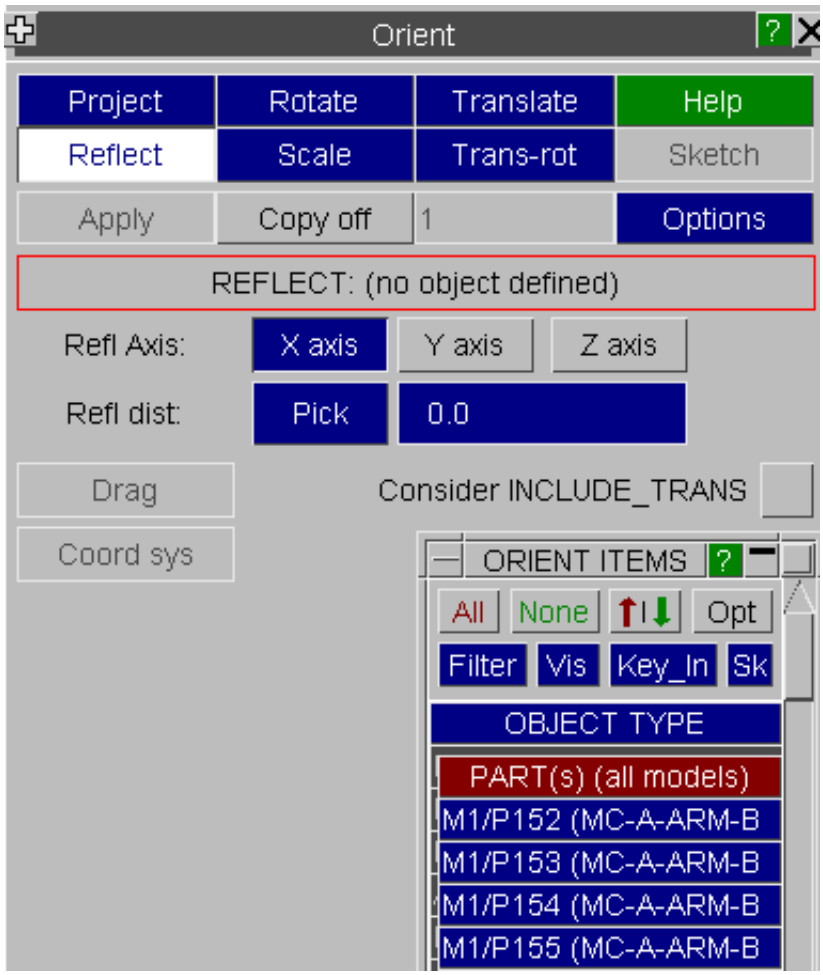


6.33.3 REFLECT: Reflect about an axis.

Reflections take place about one of the global X, Y or Z axes, about a plane at a specified distance down that axis.

To use it:
- Select the objects to reflect.
- Pick an axis: X Y or Z
- Define a distance, or use **PICK** to use a nodal coordinate to define the reflection plane distance.
- Use **APPLY** to make the reflection happen.

As before the image is drawn showing the new configuration, and you can accept, reject or repeat the transformation.



Notes on REFLECT:

- At present reflection may only take place about global axes.
- ***IMPORTANT***: A reflection is ***NOT*** the same as a rotation by 180 degrees!

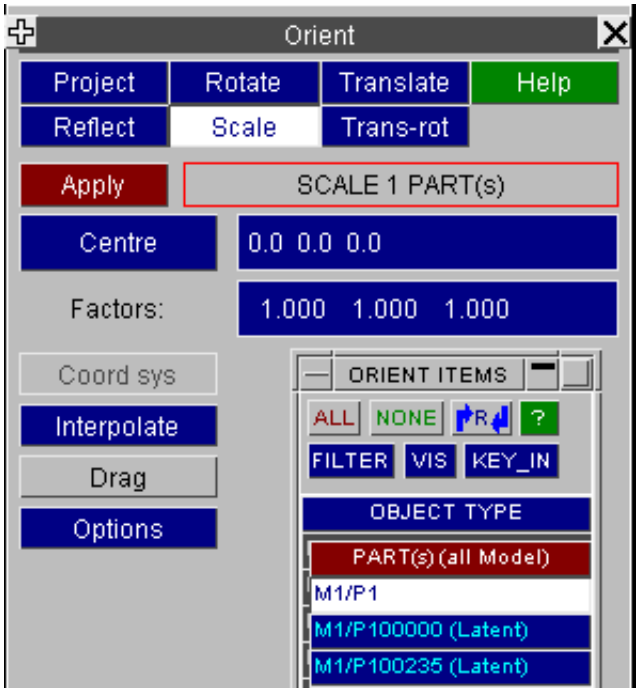
  Reflection not only moves coordinates, but also reverses the topology ordering of elements with 3 or more nodes so as to preserve their local axis systems (and +ve volume in the case of 3D elements). Whereas rotation by 180 degrees just moves the coordinates. So although the results may look similar they can have different properties.
- ***NEGATIVE SCALE:*** Primer treats negative scale (SCX * SCY * SCZ < 0) as reflection i.e. the topology of elements **is reversed**. This change was necessitated by the fact that *INCLUDE_TRANSFORM can only encode a reflection as a -ve scale and LS-Dyna does reverse the topology in this case (otherwise it would not work with 3d elements).

6.33.4 SCALE Scale by [Sx,Sy,Sz]

You can scale objects by independent factors about each of the [x,y,z] axes. To do this:
- Select objects as before;
- Define the coordinate to scale about.
- Define the factors for each of the [x,y,z] axes.
- Use **APPLY** to make it happen.

As before the image is drawn showing the new configuration, and you can accept, reject or repeat the transformation.

Negative factors are allowed, see the notes above on **REFLECT**.



CENTRE Defining a central coordinate to scale about.

Instead of typing in a coordinate you can use **CENTRE** to define a node whose coordinate will be used as the centre of scaling.

For rotate and for scale, if a single part is selected the useful option **centre @ part CofG** will become active.

What is and is not scaled

Scale factors can be different in each of [X,Y,Z] directions, which implicitly ties scaling to the global coordinate system and restricts it to spatial and directional quantities. There is no ambiguity where coordinates and vectors are to be scaled, but the issue of scalar length values is more difficult because they do not usually have an orientation.

For example scaling the thickness values T1 - T4 on a *SECTION_SHELL card would not be appropriate.

Therefore the general rule in PRIMER is that:

- Coordinates and vectors expressed in the global system ***are*** scaled
- Single ("scalar") length values are ***not*** scaled, however there are some exceptions:
    - The "finite" lengths on *RIGIDWALL cards (lenl, lenm, etc) ***are*** scaled. See Section 5, Rigidwalls for details.

DRAG: Dragging with the cursor.

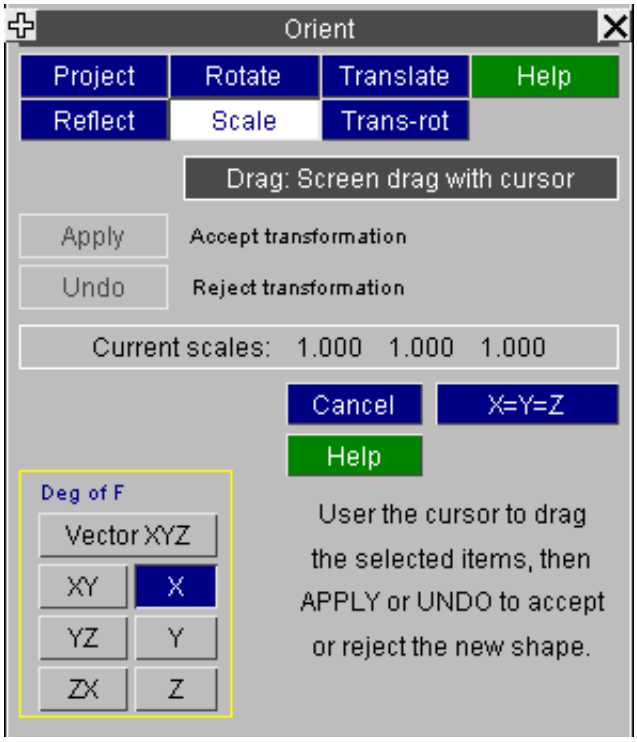
As an alternative to specifying scale factors you can "drag" the objects' scales, with respect to the defined centre, using the **DRAG** option shown in this figure.

Place the cursor anywhere on the graphics screen, press the left mouse button and, keeping it depressed, move it until the desired size is reached, then release it. The image (or a subset of it if it is large) will expand or contract on the screen, and then be redrawn at the new position.

Use **APPLY** to make the change permanent, or **UNDO** to reject it and return it to how it was before. (Using **CANCEL** also implicitly rejects any dragged rotations.)

Dragged scaling can take place using any combination of **XYZ ... Z** axis factors, and by default the factors about each axis are computed independently from the cursor movement in that axis as projected from the current view.

This tends to give unsymmetrical scaling if more than one axis is in use, so you can "clamp" all the factors to have the same values using the **X=Y=Z** button: particularly useful when combined with **VECTOR_XYZ** to expand or contract by a uniform amount in all three directions.

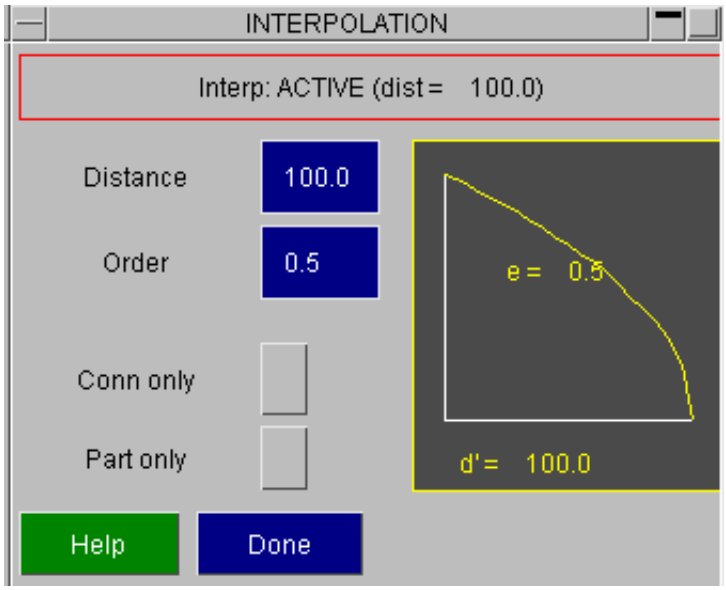


6.33.5 The **Interpolate** command.

Applying interpolation to **ORIENT** functions.

The **TRANSLATE, ROTATE** and **SCALE** functions all act by default only on the chosen items. However if **INTERPOLATE** is used the effect can be spread over a wider area by applying "interpolated" values to adjacent nodes

This is made active by setting a **Distance** value (in this example 100) in the interpolate box. This value remains current until changed, setting this value to 0 turns interpolation off again.



The **Interpolate** button is shown in red when it is active.

Once a **Distance** value has been set the coordinates of unselected nodes within a radius <**Distance**> of any explicitly selected nodes will have their coordinates updated as follows:

$$C_{new} = C_{old} + \delta * \left(\frac{d}{D}\right)^{Order}$$

$$\text{Where } \left(\frac{d}{D}\right) > 0.0$$

Where:

C	= Coordinate of this node
δ	= Coordinate change due to Translation / Rotation / Scale
d	= Distance from this node to nearest explicitly selected node
D	= The specified Distance value
Order	= The specified Order value

The **Order** value defaults to 1.0, giving a linear interpolation, but any positive value > 0.001 is permissible, and a sketch of the factor vs. distance is shown in the box. (In the figure above 0.5 has been used.)

Conn only Restricting movement to "connected" nodes only

The **Conn only** switch limits nodes that are eligible for transformation by **INTERPOLATE** still further: if it is switched on only those nodes which are connected via element mesh to explicitly chosen nodes, (as well as being within Distance), are eligible for movement.

"Connected" in this context means that it is possible to get from the node in question to any explicitly selected node via a continuous mesh of structural elements. The connection path does not have to be direct, PRIMER will follow mesh of any complexity, but there must not be any breaks to cross.

This is intended for use within very crowded areas of mesh where a purely geometrical selection of nodes for movement could lead to undesirable results by including unrelated items.

Part only Restricting movement to nodes of parts

In this mode, Primer will identify parts of which nodes have been selected for orient and only allow interpolation on nodes of these parts.

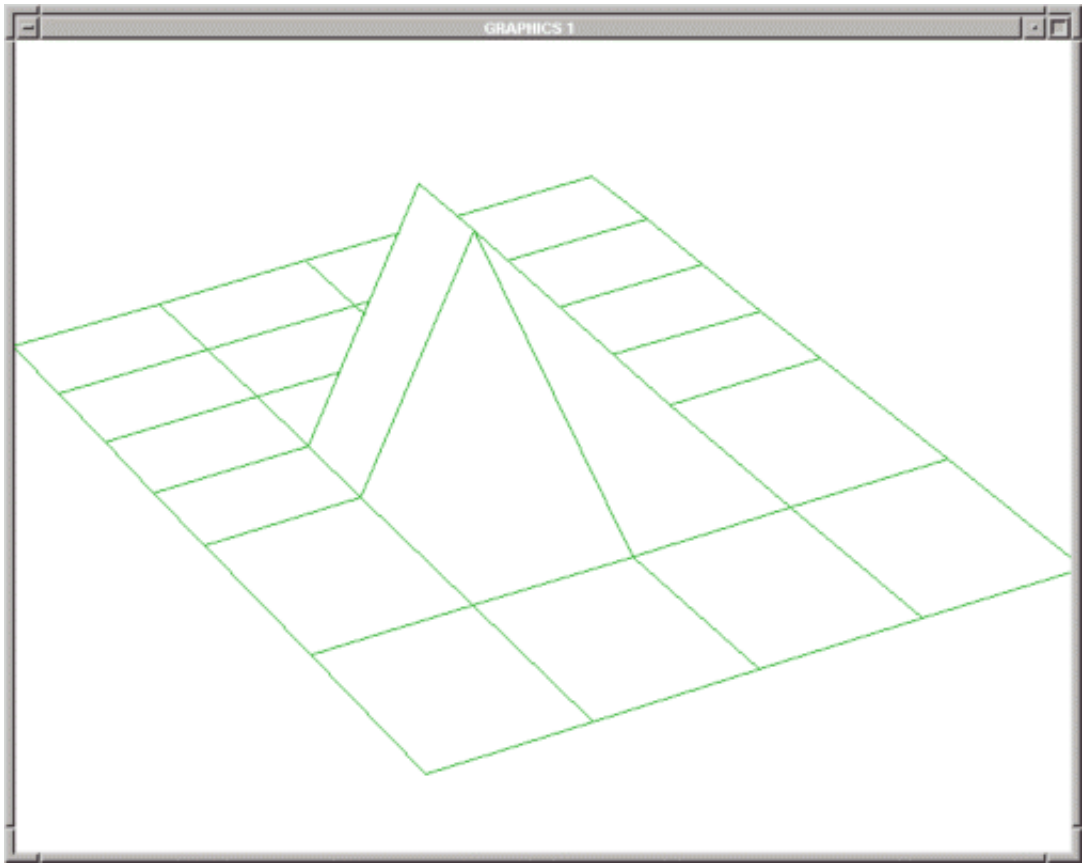
Warning about speed penalties

Using **INTERPOLATE** can slow down transformations significantly, particularly if large **Distance** values are used. This is because a bucket sort for the "nearest explicitly selected node" is required for every candidate node within **Distance** of selected nodes, and the number of nodes in the sort increases as the cube of **Distance**.

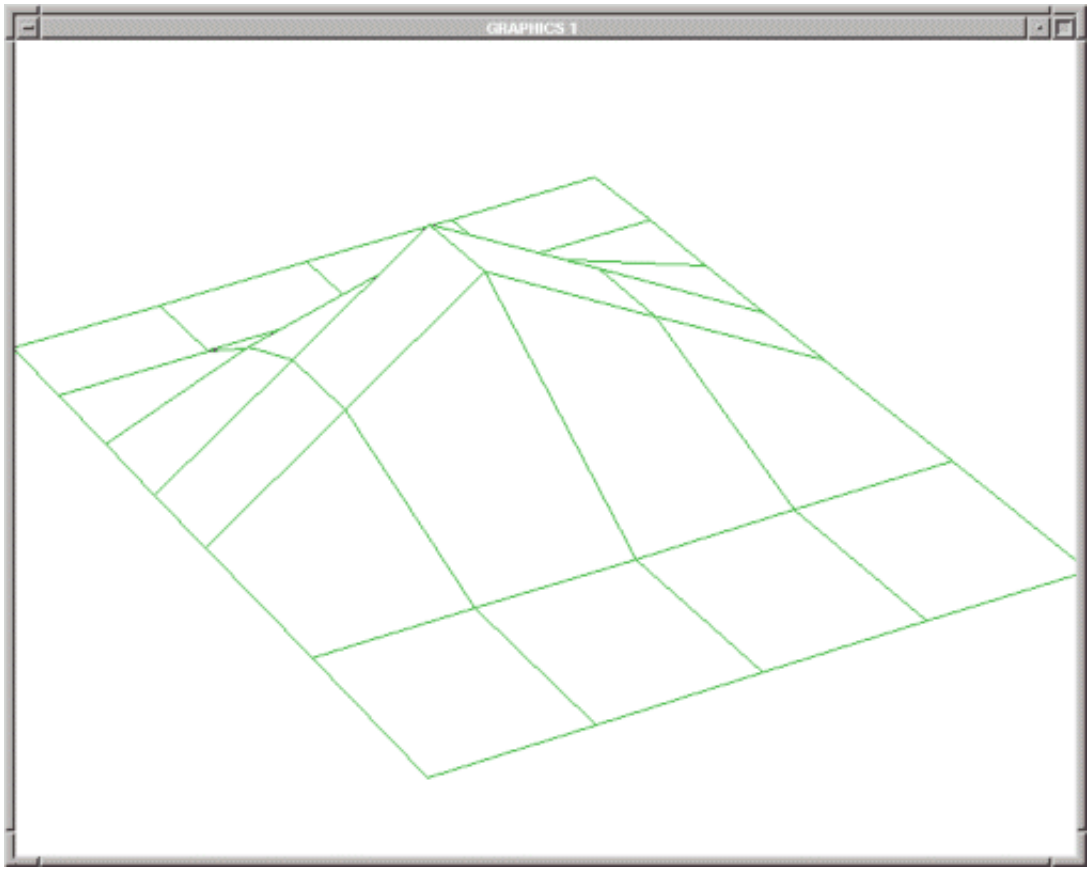
So don't be surprised if there is a significant delay when interpolation is used with large **Distance** values in big models. Using **Conn only** can speed matters up as it usually reduces the number of candidate nodes for movement.

Example use of INTERPOLATE:

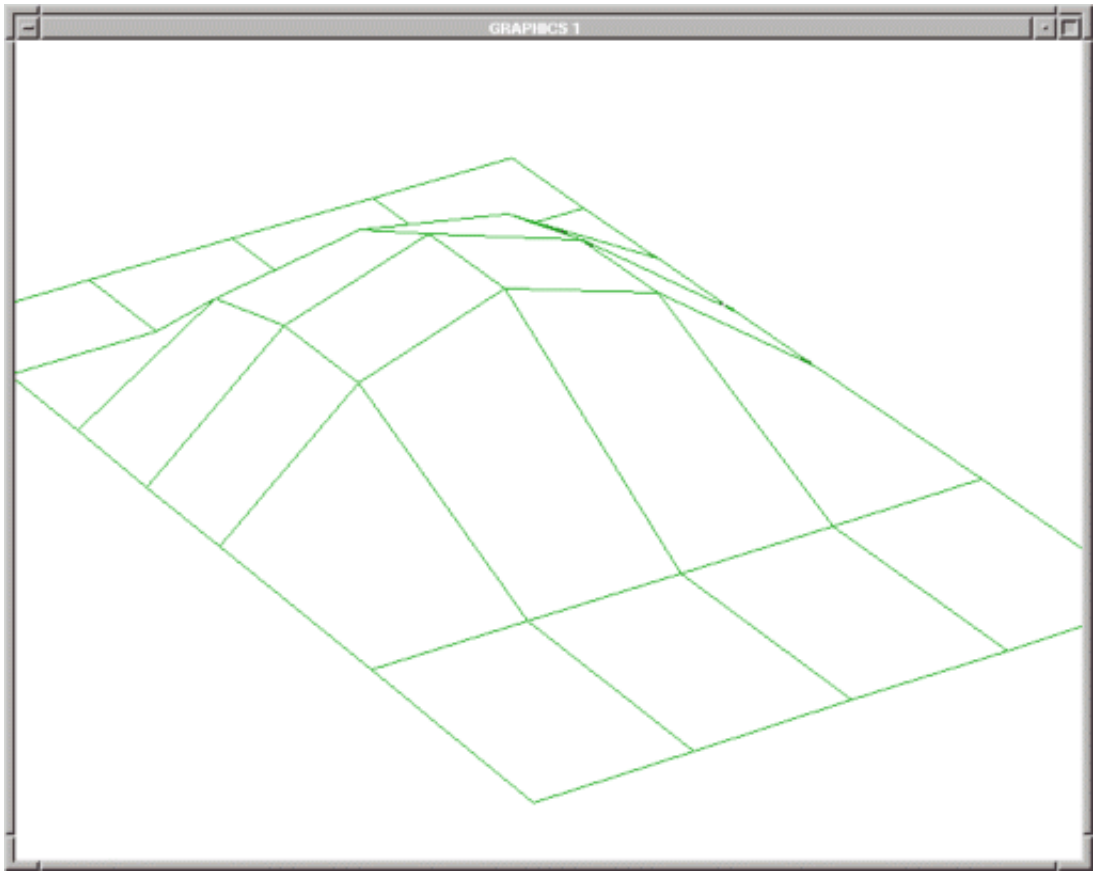
Distance = 0
In this figure two nodes in the centre of a flat plate have been raised, with no **INTERPOLATE** value set.
It is clear that only they have moved, and adjacent nodes are unaffected.



Distance = 20, **Order** = 1.0
In this figure the same nodes have been moved, but now **INTERPOLATE** has been switched on. The **Distance** chosen is equal to half the smaller mesh dimension.
Here **Order** = 1.0, so there is a linear interpolation between the selected nodes and the edge of the mesh.



Distance = 20, **Order** = 0.5
In this final figure the **Order** value has been reset to 0.5, giving a curved variation from centre to edge of the mesh. This shows how a non-linear effect can be achieved.



6.33.6. PROJECT: project to a line, plane, or surface

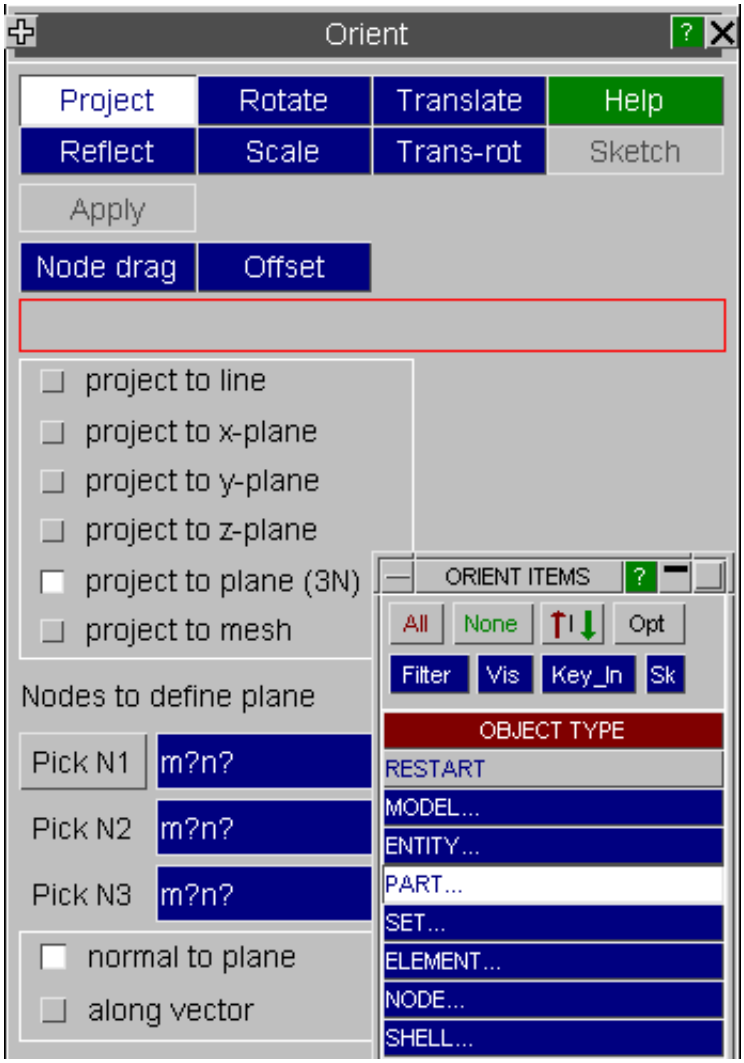
The projection option operates on the nodes of the items selected through the object menu.

The project-to-line operation will move them to the nearest point on the defined line.

Projection-to-plane can use either a global plane, defined by a single coordinate or a node pick, or an arbitrary plane defined by 3 nodes. Projection may be done normal to the plane or along a vector.

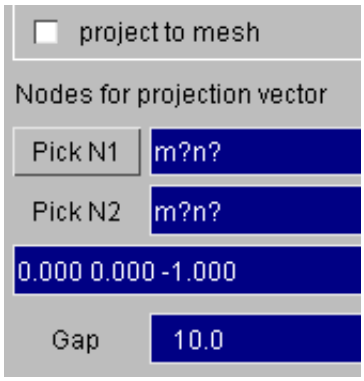
Projection-to-mesh requires a direction vector and the mesh can be defined using shells, shell sets, or shell parts.

To apply the orient press **APPLY**. You will then have the option to **UNDO_ALL**, if the orient is not as you wanted it.

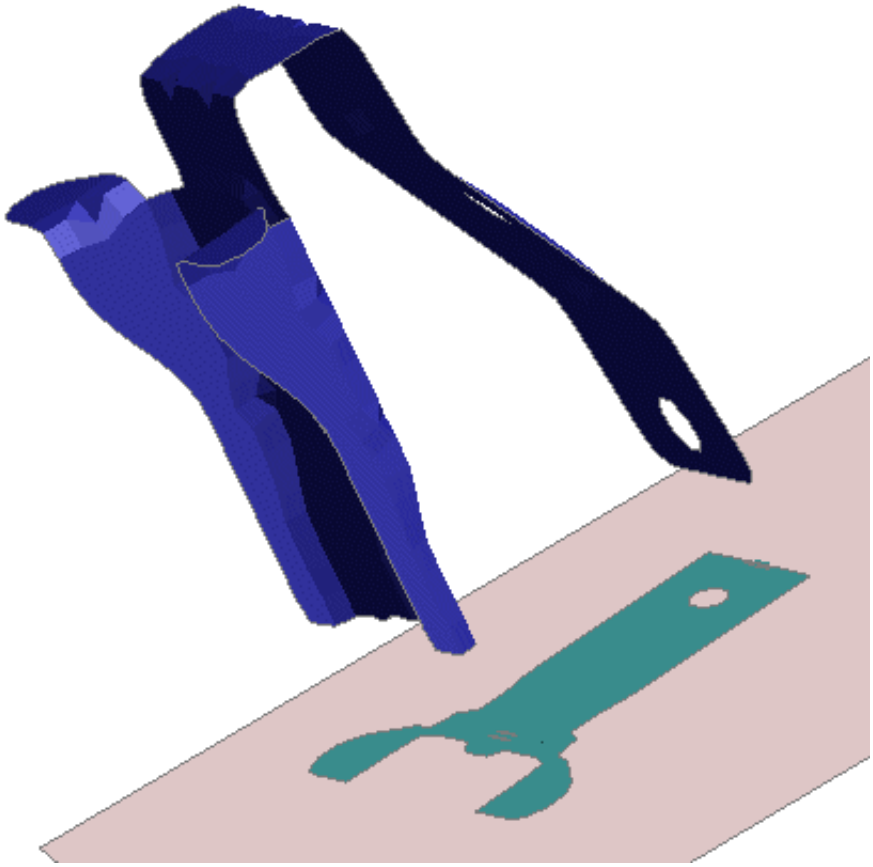


Projecting across models

Project to mesh can be used to project across models, i.e. the projected items and the mesh to which they project do not need to be in the same model.



- a gap value may be set to offset the projection
- COPY may be used with project



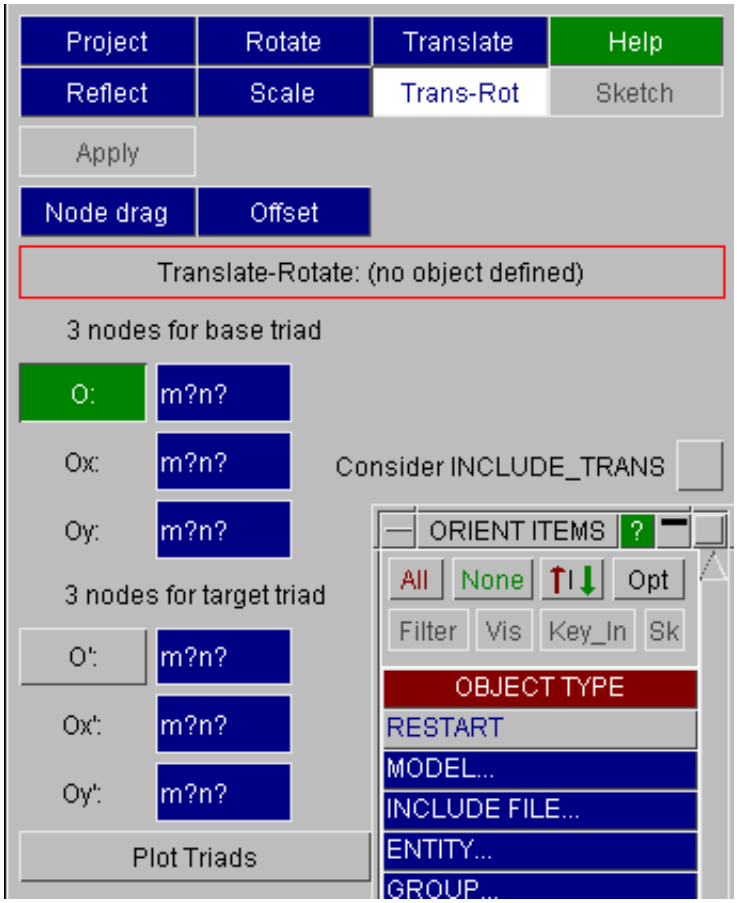
6.33.7. TRANS-ROT: translate and rotate

The Translate-rotate function requires the user to define a base triad and a target triad.

Each is defined by an origin node (O) a second node prescribing the X vector (Ox) and a third node lying in the XY plane (Oy). The appropriate Y and Z vectors are then found.

A translation vector of base origin node to target origin node and a set of rotations (base triad to target triad) have now been established.

This translation and rotation will then be applied to the select items.

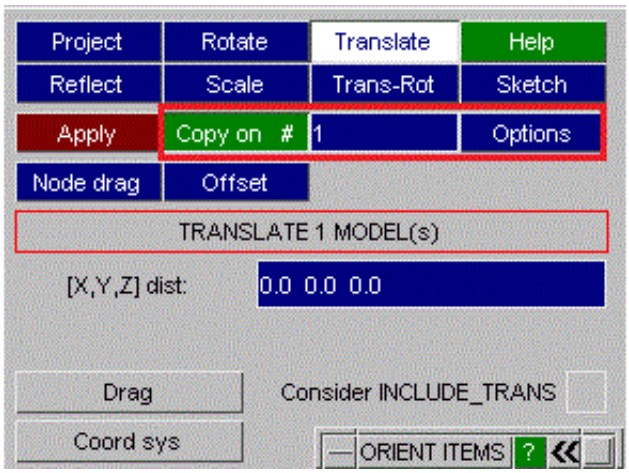


6.33.8. Copy and Orient

The **COPY ON** option is available for REFLECT, ROTATE, TRANSLATE, SCALE and PROJECT.

The copy function is **not** available for translate by **Contact Orient** or for **Trans-Rot** (triad to triad orientation).

The copy can be applied once or multiple times, the orient being incremented each time. The initial orient may be defined by a DRAG operation or explicitly. The copy function can be turned on at the top of the orient panel. The copy options can be opened by clicking on **Options**.

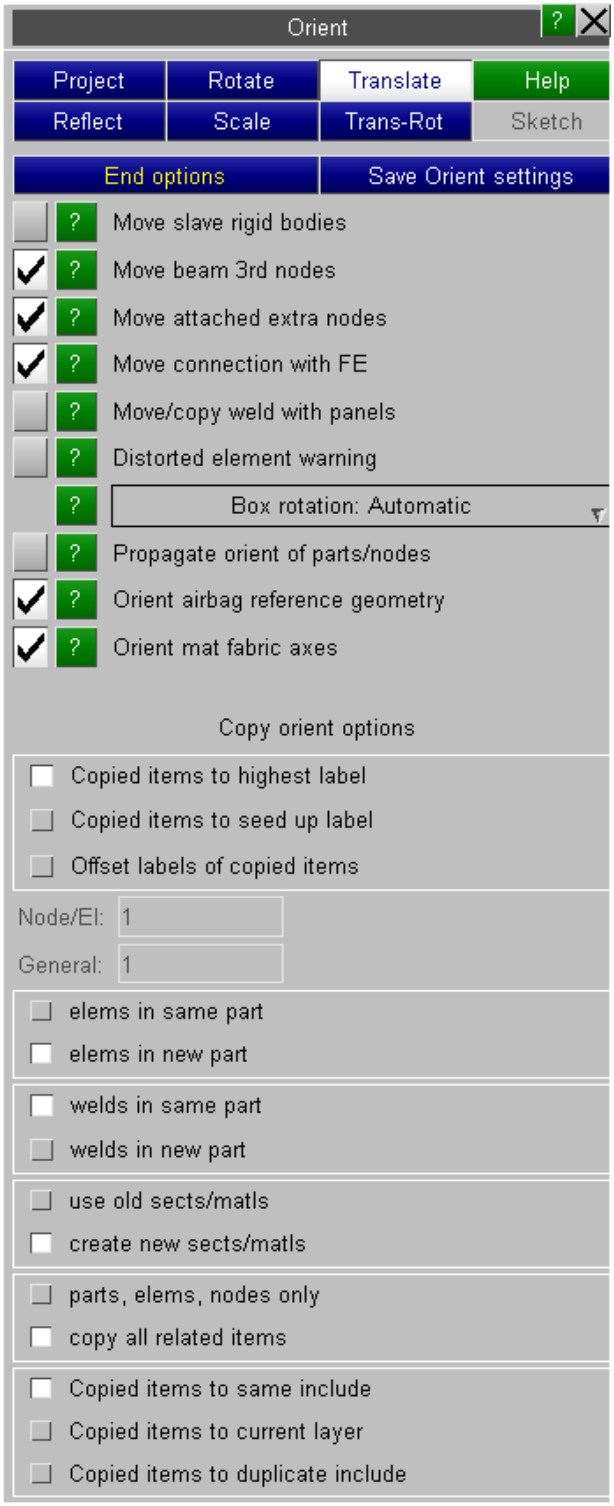


When you select **COPY ON** the orient option are temporarily pre-configured as follows:

- Move slave rigid body OFF
- Move beam 3rd node ON
- Move attached extra nodes ON
- Move connection entities ON

Move/copy welds with panel is available for user to set.

When copy is deselected (or the orient panel dismissed) Primer will restore the settings to their previous value.



Labels for new items By default the new items will be labelled starting with the highest current label + 1 for each item type. Alternately, the user may specify a pair of seed labels (this or the next available label will be used) or a pair of offsets. One label is for the more populous type of item (nodes, elements, node sets, nrbs), the other for all other types. In the offset case, Primer will check that all the offset labels are available.

If include label ranges are defined for the model, Primer will always try to correct any out of range labels once the orient operation is completed.

Part for new items For copied element the user nay choose to

- **elems in same part** put copied elements in the original part
- **elems in new part** put copied elements into a newly created part

If the user selects **elems in new part**, items referenced by the original part (e.g. material, section) will also be copied if and only if the radio button **create new sects/matls** is selected. Otherwise the old ones are used. If the part is

referenced directly by something (e.g. Boundary Prescribed Motion Rigid or contact by part) this card will also be copied if the option **copy all related items** is ticked.

Part for welds If the options **copy welds with panels** and **elems in new part** are set, by default the option is to keep welds in their original part. If you want a new part for the welds you may switch from **welds in same part** to **welds in new part.**

Include files There are three options for the include file which copied items are put into:

- **Copied items to same include** will put the items into the same include where the original item was in.
- **Copied items to current layer** uses the current include.
- **Copied items to duplicate include** creates a duplicate include (named to xxx_1, etc) for the new items.

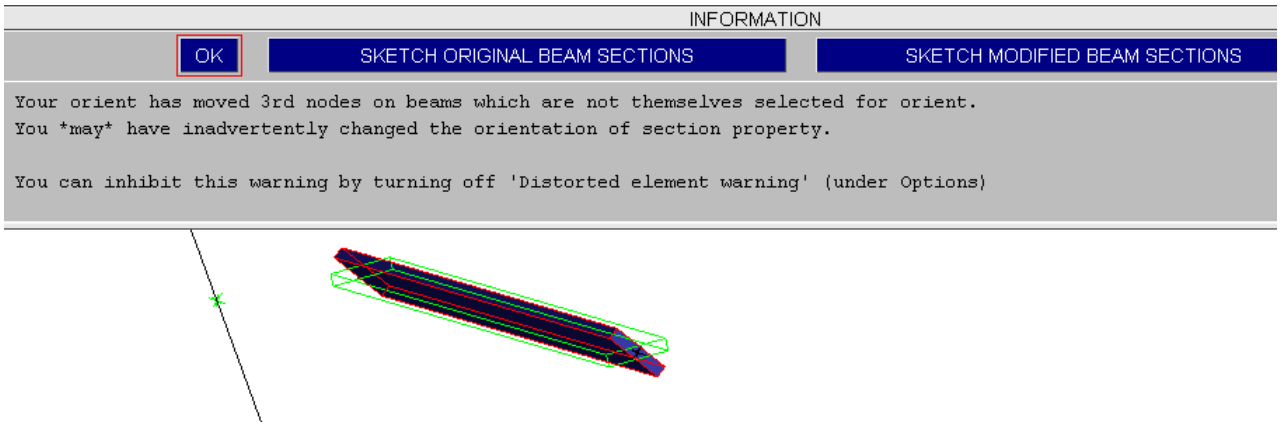
6.33.9. Check for element distortion

With the option **Distorted Element Warning** active, PRIMER will detect when an orient operation moves some but not all the nodes of an element.

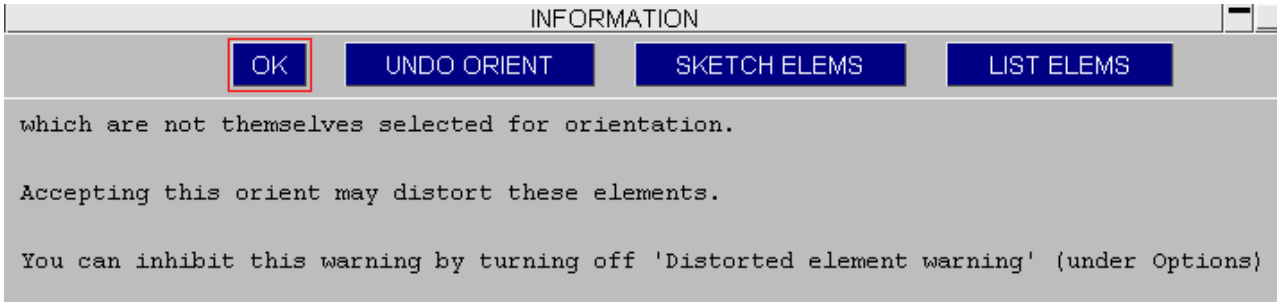
This typically occurs when orienting a rigid part with the option to **Move attached extra nodes** active. These nodes attach to deformable elements which will get distorted.

The function will allow you to sketch the problematic elements and undo the entire orient or just undo the orient of extra nodes if applicable.

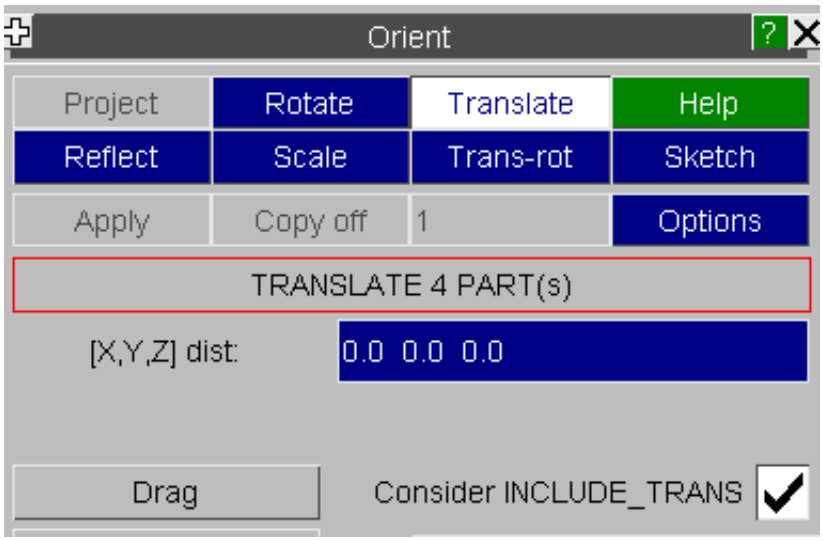
Additionally the 3rd nodes of non-circular beam sections are checked. If the orientation of the section changes, you will get a warning message which allows you to sketch the original and the current section.



Subsequently you will be offered the option to undo the orient.

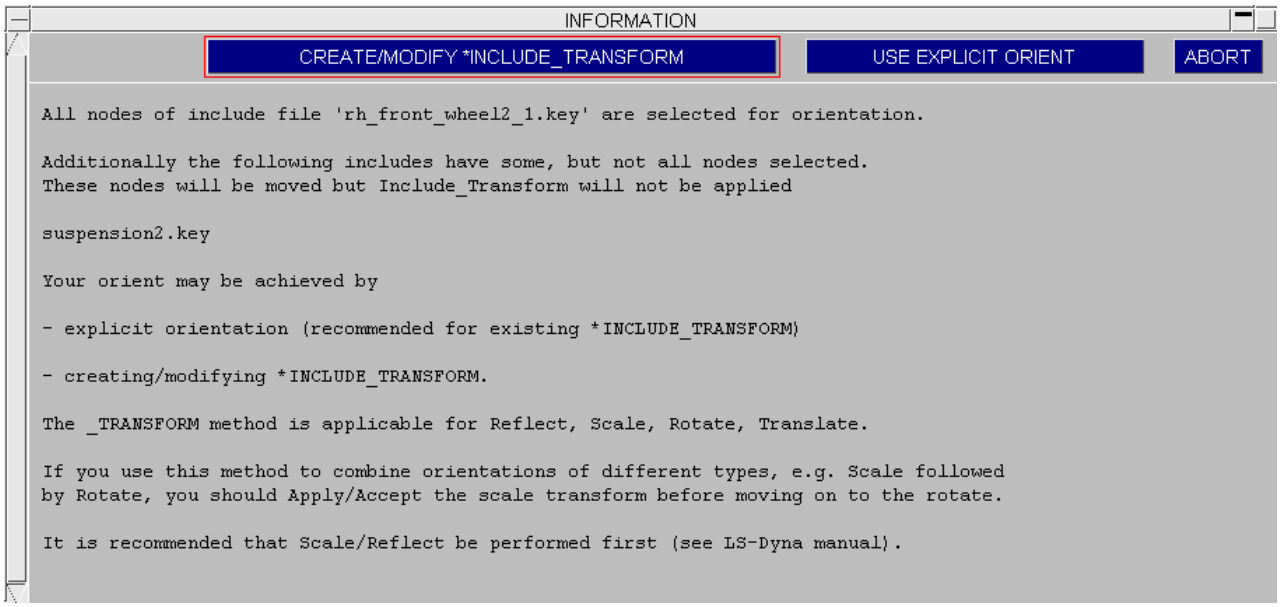


6.33.10. Orient and Include Transform

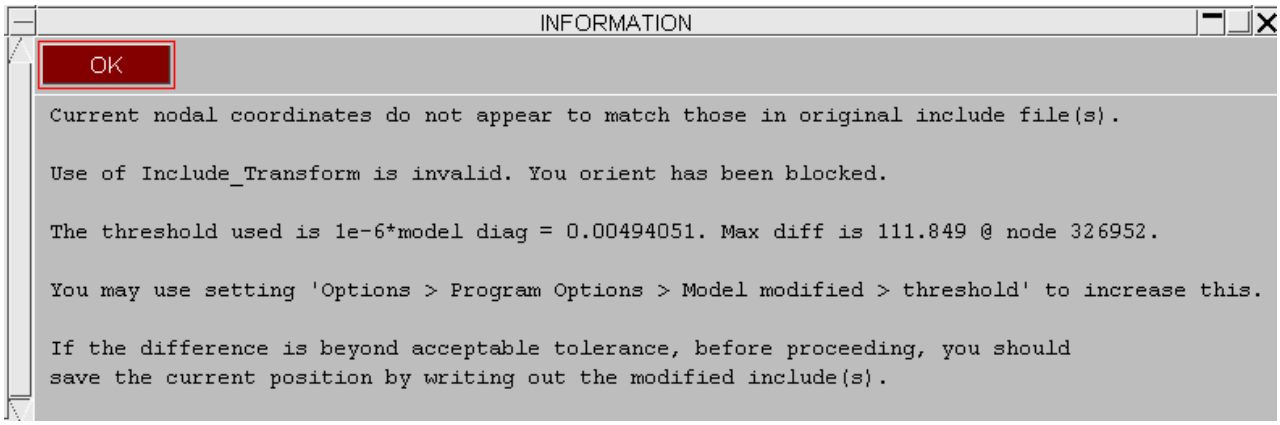


Consider INCLUDE_TRANSFORM option is available for Scale, Reflect, Rotate, Translate, Trans-Rot orients.

In this case, before orient is performed, Primer will inspect what is to be oriented looking for include files where all nodes are selected. If found, you will be offered the opportunity to create/modify Include_transform (as an alternative to explicit orient). You will also be warned of any includes where a subset of nodes have been selected - these obviously cannot use the transform method and will be oriented explicitly.



If you select the transform method Primer will then check that the nodes are in their original as read position, i.e .that an explicit orient has not been already applied. If the nodes fall outside the given tolerance the orient will be blocked. If the max diff reported seems small enough you may wish to increase the tolerance under Options > Program Options > Model Modified > Threshold. Otherwise, you need to save the include in its current position, before you can implement the include tranform method.



6.34 OTHER

Other lesser-used options are grouped into this popup. These are:

6.34.1 Clones

6.34.2 Forming

6.34.3 Target marker

6.34.4 Transfer

6.34.1 CLONES

How PRIMER handles duplicate definitions using Clones.

Theoretically an LS-DYNA input deck should only have a single instance of any labelled item A, for example ***NODE** label 1 should only ever appear once. "Unique" keywords such as ***CONTROL** should also only occur once. However LS-DYNA tolerates duplicate definitions in a range of ways:

- ***NODE** definitions may be repeated so long as they obey certain rules about coincidence. See **[NODE] Duplicates** for more information on how these are processed in PRIMER.
- ***PARAMETER** definitions can be repeated by using suffices such as **_LOCAL** and **_MUTABLE**, and handling of repeated definitions is further controlled via the ***PARAMETER_DUPLICATION** card. See **PARAMETER**s for more information on how these are processed in PRIMER.
- LS-DYNA also tolerates repetition of some other keywords, usually by ignoring all but the last instance found in the input deck. PRIMER tolerates this subject to the [Options] Permit duplicates keyword input setting.

Unlike LS-DYNA, which can ignore all but the solitary definition it chooses to use, PRIMER must "remember" these duplicate definitions so that they can be written out again correctly, otherwise the sequence "read and write model" will end up removing entries from keyword files, and it does this by creating "clones".

What is a clone?

A cloned definition is not a copy of a keyword, rather it is a ***reference*** to the original "true" definition, and is subject to the following rules. In the definitions below "include file" can also mean the master file.

- There is only ever a single true definition of an entity inside PRIMER. This will normally be the first such definition read from the input deck, and this will dictate which include file it exists in during the PRIMER session. However if differences are detected in duplicate definitions you can choose which to use, see below.
- Any number of "clones" may refer to the true definition A, subject to the limitation that an include file may only contain a single definition of item A. This means that:

  - A clone of A may not exist in the same include file as the true definition of A.
  - An include file may only contain a single clone of item A.
- During keyword output the external keyword format of the true definition is written out in its "true" include file and also in every include file in which a clone of it exists.

Inside PRIMER each model has a list of clones, which can be thought of as pseudo-keywords. They show up in cross-reference lists of true items, but otherwise they are mostly invisible and can be ignored.

What happens if duplicate definitions contain differences?

Special rules apply to the processing of ***NODE** and ***PARAMETER** cards, but for all other duplicates PRIMER applies the following rules during keyword input.

- Each duplicate definition is stored temporarily until keyword input is complete.
- Then each duplicate definition is compared with the original "true" definition, searching for any differences.
- If no differences are found then the first definition is used as the "true" one, and subsequent definitions become clones. This process is silent.
- If differences ***are*** found then these are listed and the user is asked how to handle them. The options available are:

USE_LAST	Typical LS_DYNA behaviour. The **last** read definition becomes the true one, and all others become clones.
USE_FIRST	The **first** read definition is the true one, the remainder become clones.
MASTER_OR_LAST	If a definition exists in the master file this becomes the true one, otherwise the last read is used. The rest become clones.

MASTER_OR_FIRST	If a definition exists in the master file this becomes the true one, otherwise the first read is used. The rest become clones.
ABORT	Keyword input is aborted and the model is deleted from PRIMER's memory.

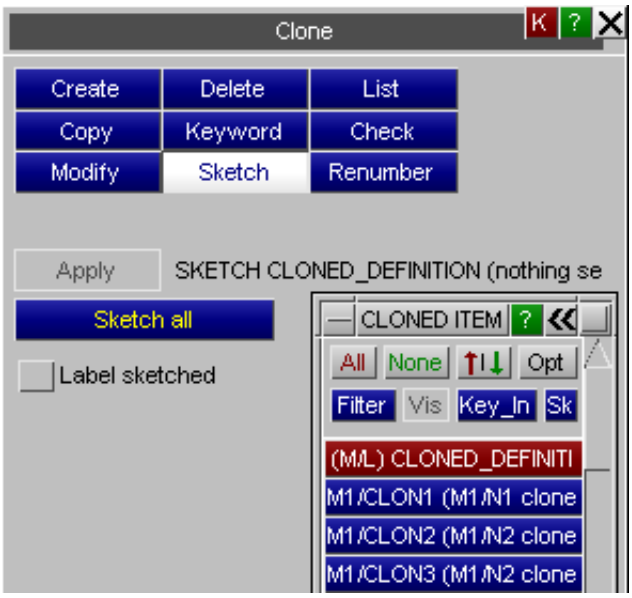
It must be emphasised that inside PRIMER only a single true definition exists, and on keyword output this true definition is simply repeated in each clone location.

Therefore any differences in the content of duplicate definitions will be lost when the deck is read into PRIMER.

This may seem like a limitation, but in fact it is a safety measure. If your deck contains duplicate definitions that have different contents this may well be an error, and even if it is not an error then having differences in these definitions may lead to subtle and hard to find errors during the analysis since the values used will depend on the order in which include files are read. In addition LS-DYNA does not state explicitly how it processes duplicate definitions, so any change in its behaviour in future versions may also affect your results.

It is ***strongly*** recommended that you treat any differences that PRIMER detects as an error, and sort them out manually.

The top level **CLONE** panel



Although the Clone panel is accessed from **[Tools] Other**, it is really a pseudo-keyword and as such its functionality mimics that of conventional keywords.

Create, **Modify** and **Keyword** functions are all performed using the keyword editor, see Keyword editing below. **Delete**, **List**, **Check** and **Renumber** all work as for other keywords.

Drawing Clones

Clones are not drawn separately on plots because they have no separate existence, and there is no separate `CLONE` entry in the **ENTITY** panel for the same reason, so the only way of visualising them explicitly is to the **Sketch** options in this panel, as shown here. You can select a subset of definitions from the menu for sketching, perhaps using filtering options to narrow down what is shown and then using **Apply** to draw them. Or **Sketch all** will draw everything.

By default only the items themselves are sketched, however if you turn on **Label Sketched** then the original item labels, the clone label and also the incoude file in which the clone lives will all be added to the plot.

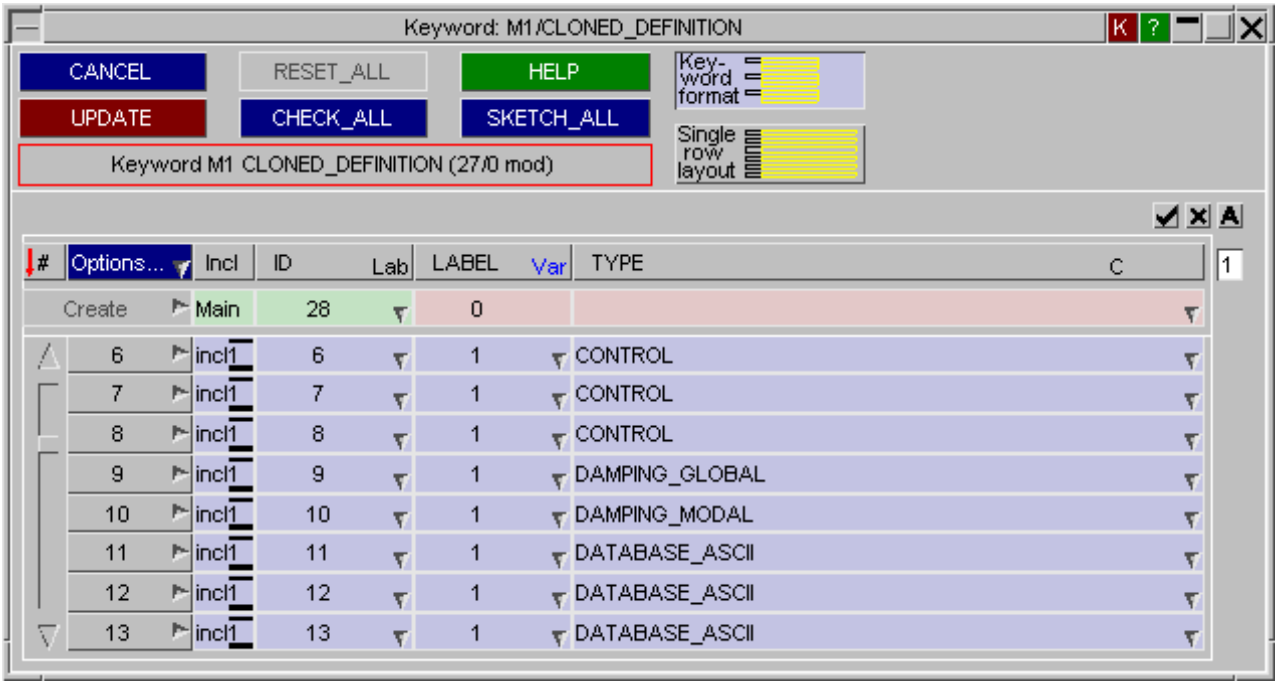
Keyword editing clones

Since clones are pseudo keywords you can think of them as having the keyword format:

ID (clone label)	**TYPE** (keyword type)	**ITEM** (label of duplicate item)
label 1 .. n	eg NODE, SOLID, MAT	label 1 .. n

The clone ID is purely for reference inside PRIMER, and clones will be created sequentially from 1. It has no relationship with the actual duplicate **ITEM**.

Here is an example of the clone keyword editor in a model containing duplicate items of various types.



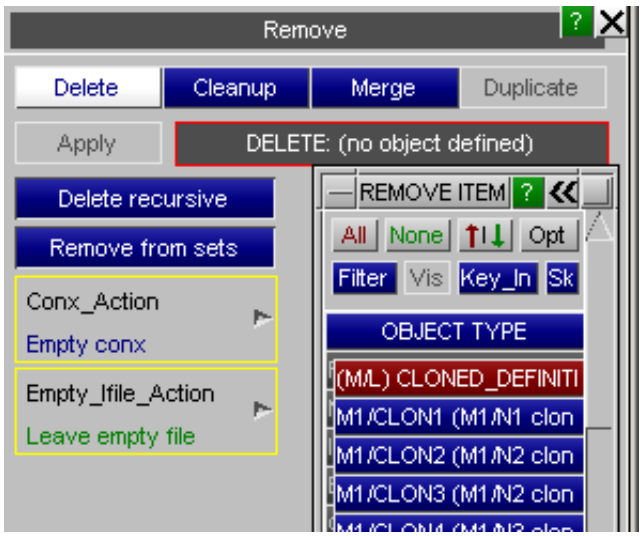
Definitions can be created, edited and deleted here. The **`Incl`** column shows the include file in which each clone exists, ***not*** the include file of the true definition.

Deleting clones

Although clones can be deleted in the keyword editor above this is a cumbersome method, and they can more easily be deleted using the normal **Remove** logic.

Simply select the category **Cloned Items** and then select clones for removal in the normal way.

Deleting a clone has no effect on the "true" definition of the item, it only removes the clone definition itself.



.

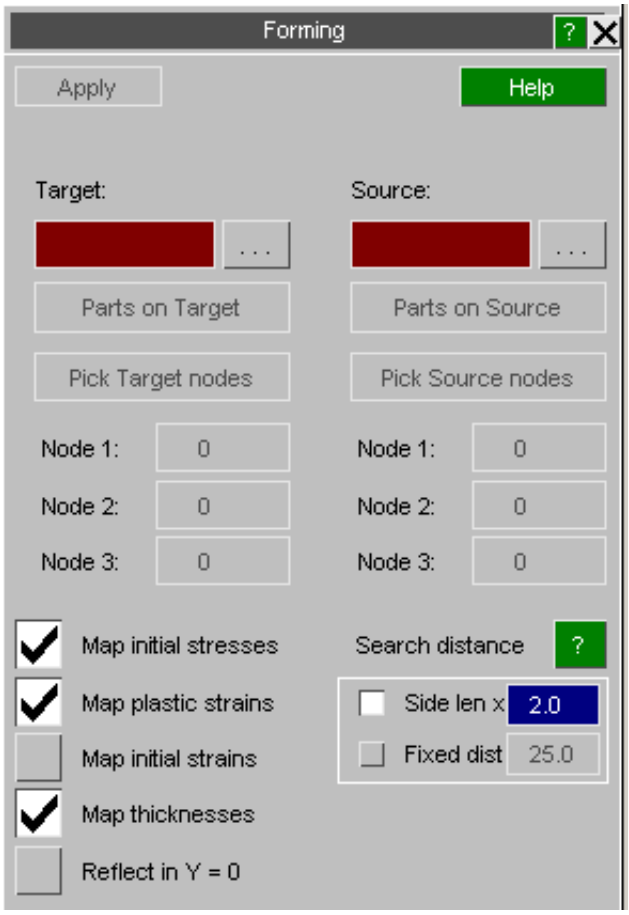
6.34.2 METAL-FORMING

The **METAL-FORMING** specialist function allows you to include the effects of metal forming on parts in crash models by giving the part in the crash model the initial thicknesses and plastic strains from the metal-forming analysis. The models can have different meshes and different orientations. PRIMER will map the results from the forming model mesh onto the crash model mesh.

Select the **Other** pop-up menu from **Tools** and then press the **FORMING** button to start the process.

Main panel

The main metal-forming panel is shown in the adjacent figure.



The panel allows you to map the results from **Part B** in the forming model (**Model B**) onto **Part A** in the crash model (**Model A**).

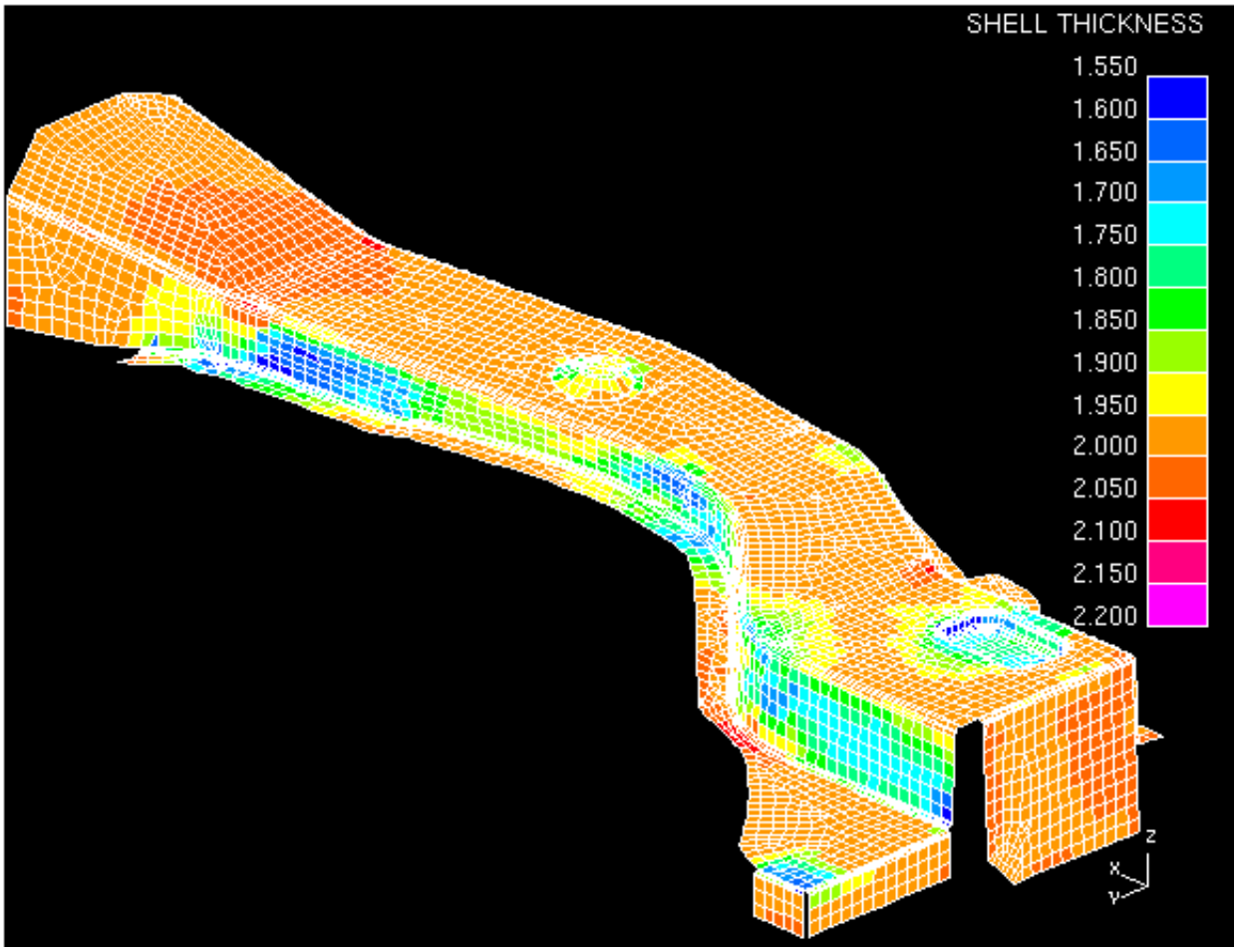
Model A must be the crash model.
Model B must be the forming model.

The process to map the results from the forming model onto the crash model is:

- Read the crash model (target) and the forming model (source) into PRIMER.
- Type in the model numbers of the crash model into **Model A** and the forming model into **Model B** (e.g. 1 and 2).
- Pick, select or type the part in the crash model you want to modify into **Part A**.
- Pick, select or type the equivalent part in the forming model into **Part B**.
- Give 3 pairs of equivalent nodes in **Model A** and **Model B** for orientation - i.e. **Node 1** in **Model A** is equivalent to **Node 1** in **Model B**. Each triplet of nodes forms a right-handed coordinate system with its origin at **Node 1**.
- Select the data to be copied from source to target model
- Optionally use **Reflect in Y = 0** to reflect the source model and its data prior to mapping.
- Press **APPLY** to map the results from the forming model to the crash model.

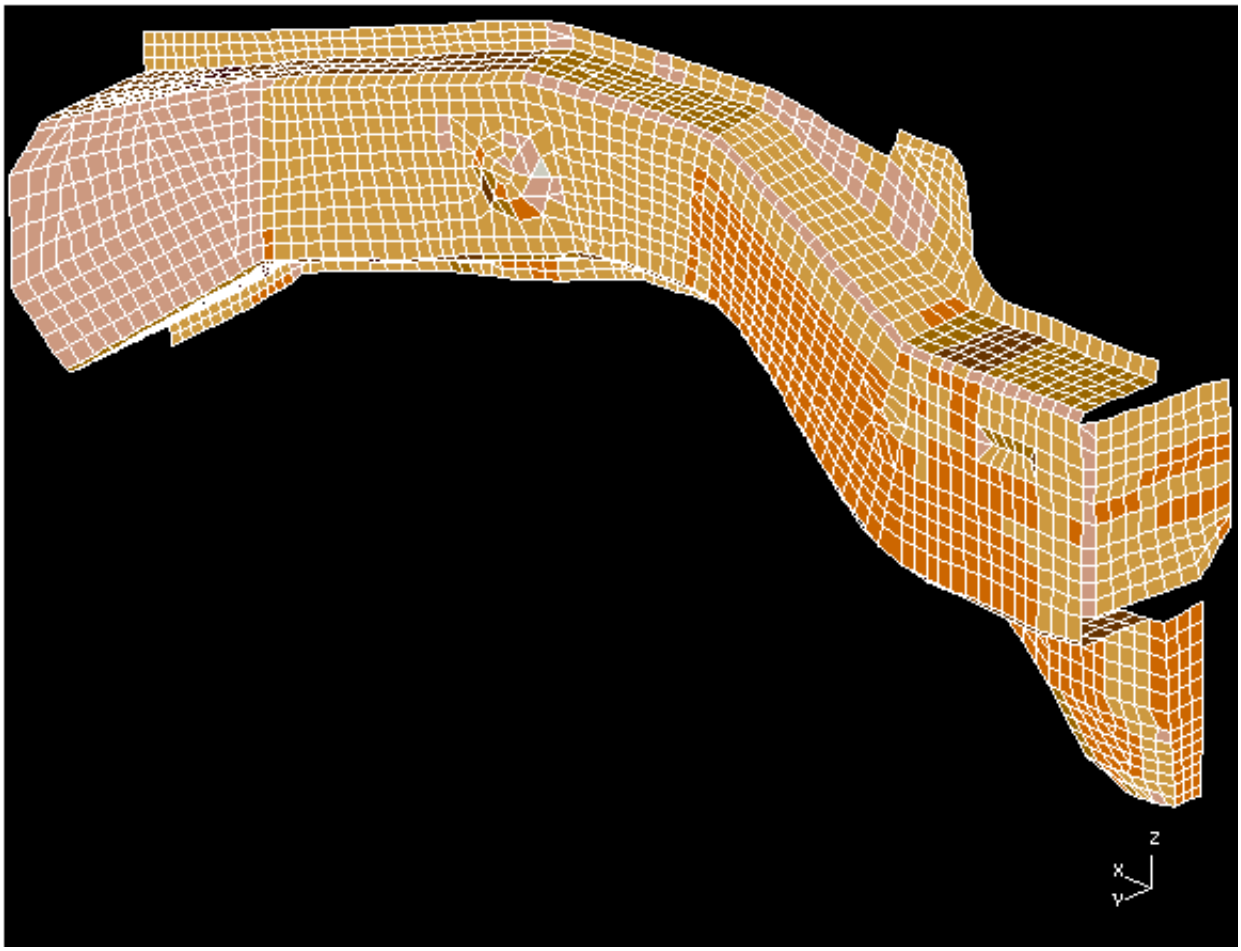
Example

The image below shows the thickness distribution from the forming analysis.



Thickness distribution in forming model

The crash model has a uniform initial thickness and a different mesh to the forming model

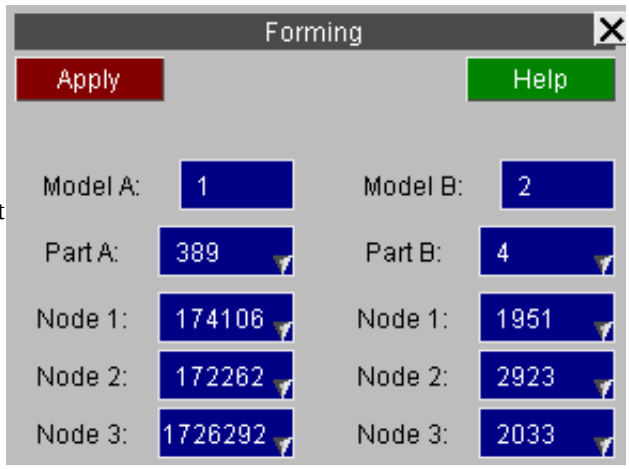


Mesh of crash model panel

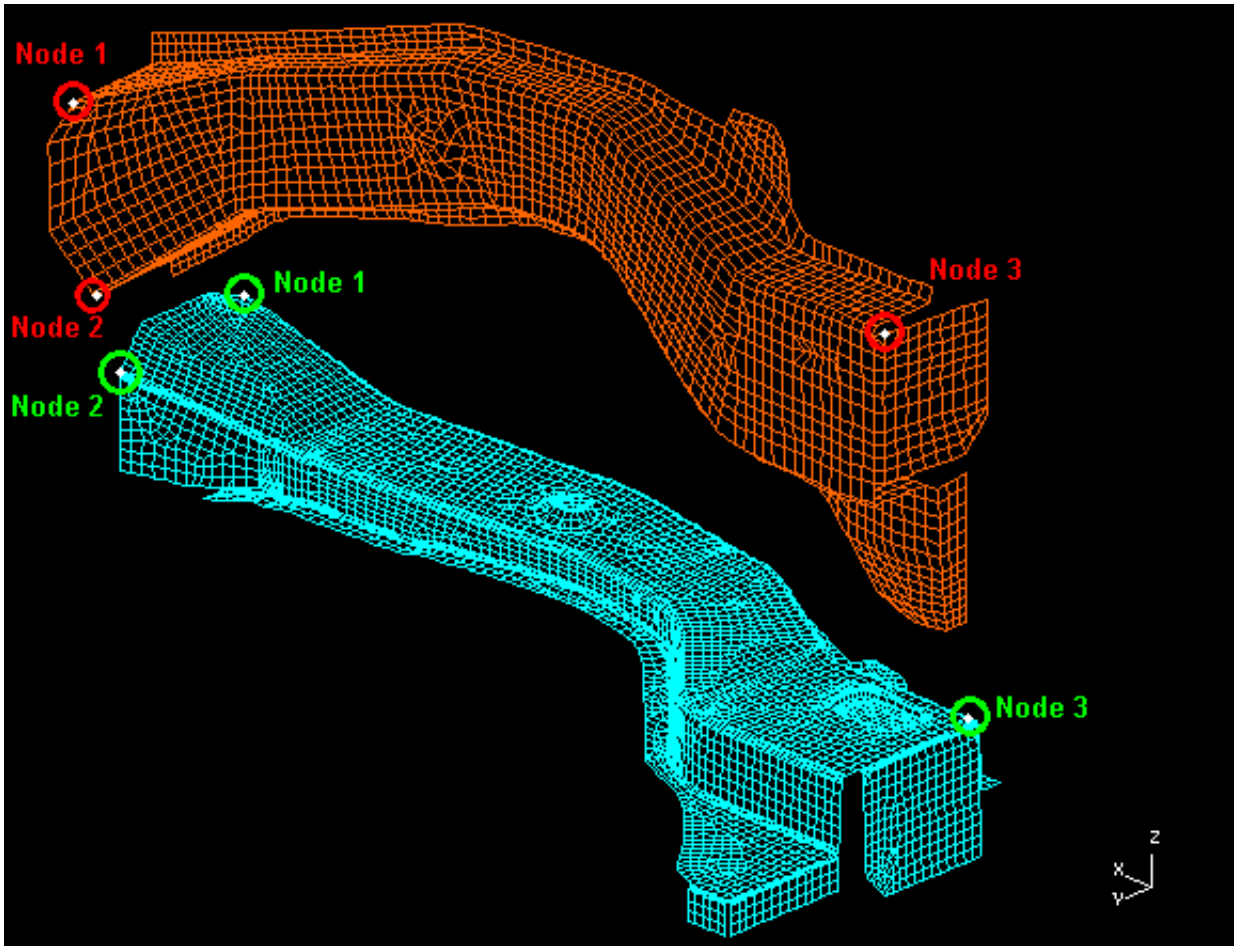
To map the results from the forming model panel onto the crash model panel we give the model number part and 3 nodes for each model.

The Crash model details are given in **model A** on the left side of the panel.

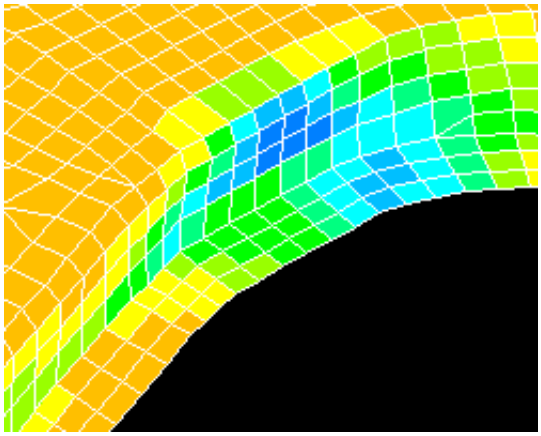
The Forming model details are given in **model B** on the right side of the panel.



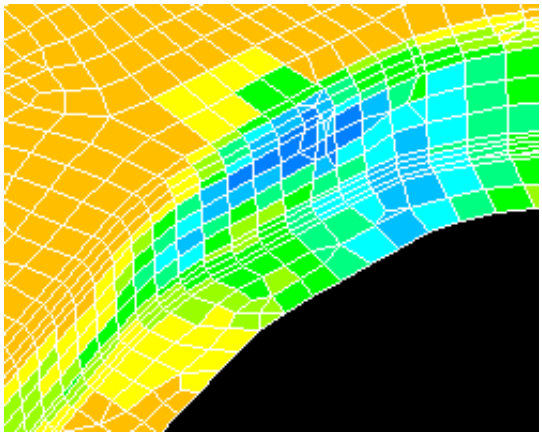
The figure below shows the locations of nodes 1, 2 and 3 in both models. These nodes **must** be at equivalent points on the panel. It is essential to make the 3 nodes as far apart as possible and not colinear so that PRIMER can map the results as accurately as possible.



When the **APPLY** button is pressed Primer takes the results from the forming model and maps them onto the crash model. For example plotting shell thickness gives:



Crash model



Forming model

How this process works.

The process goes through the following stages:

1. The local coordinate systems for Source (forming) and Target (crash) models, based on nodes 1 to 3 in each, are calculated. Node 1 is the origin, vector N1N2 gives the local X axis, and N3 lies on the local XY plane. If no nodes are defined then the global coordinate system is used.

2. In both Source and Target models the average coordinate of each element, calculated from the average of its nodal coordinates in its local system, is calculated and stored. The longest element side length in each model is

also calculated for use in defining the default "search distance".

3. For each element in the Target model a search is made for the nearest element in the Source model. This is based on the vector distance between the average element coordinates as calculated in step (1). If no Source element is found within the "search distance" of a Target element then that element is unmatched, and no data will be transferred for it. The matching process will report how many elements were unmatched, and if this unacceptable you may need to correct the local coordinate systems, or change the search distance (the default distance is 2x the longest element side length in the models).

4. For each matched Target element the selected data from the nearest element in the Source model is copied. In more detail:

   - Thickness is taken from the source element's *ELEMENT_SHELL_THICKNESS definition if it exists, otherwise from its *SECTION_SHELL card. The element in the target model is always converted to a *ELEMENT_SHELL_THICKNESS definition, with thicknesses being defined for all nodes.

   - Stresses and strains from the *INITIAL_STRESS_SHELL and *INITIAL_STRAIN_SHELL cards are copied over verbatim. Any existing data for this element in the target model is deleted and replaced by the incoming data.

     Note: No check is made for consistency of integration point locations in *INITIAL data, in other words PRIMER does not check that the number and location of integration points in the source element matches those in the target element, it simply copies the data from source to target.

     This is not an error since LS-DYNA will interpolate the initial data onto the underlying element definition, giving a consistent result. However if the detailed distribution of stress in the target element is important you should be aware that PRIMER will not issue any warnings if source and target elements have different formulations or integration point distributions.

   - Elements in the target model that are not matched are not changed in any way: they will keep their original thickness and any initial stress and/or strain definitions they may have.

If the "Reflect in Y = 0" option is used then the following modifications to the process above are made:

- Before step (1) above all nodal coordinates in the Source model have their signs reversed, and all initial stress and strain off-diagonal tensor terms with a Y in them (σxy, σyz, εxy, εyz) also have their signs reversed.

- The mapping process in steps (1) to (4) then proceeds as normal.

- Finally the Y coordinates and tensor terms in the source model are restored to their original values.

Note that this means that reflection about the Y axis explicitly means the **Global** Y axis at Y = 0. If you need to reflect source model results about some other local axis it will be necessary to **Orient** the model first so that the reflection plane becomes parallel to global Y, and is located at Y = 0.

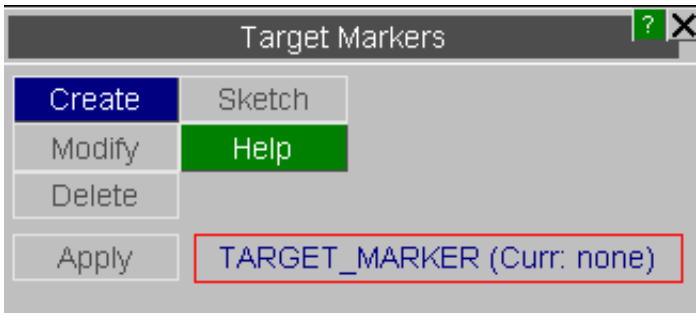
6.34.3 TARGET MARKER

These are circular symbols with four quadrants that are attached to nodes (shown below). If applied, they are drawn whenever the node is visible. They are drawn in the plane or the screen,so if you rotate a model will see them appear to rotate in space.

These can be edited through their own specific editing panel (see below).

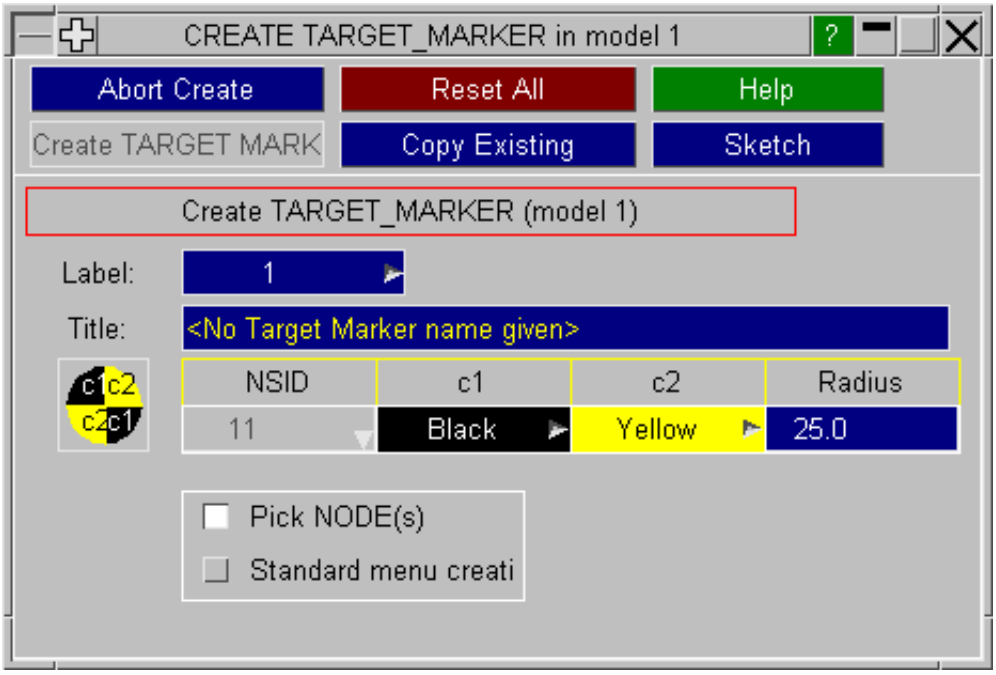
- Main Menu
- Creation
- Editing
- Deletion
- Sketching

The following figure shows the main menu for the editing of target marker definitions.



CREATE Making a new target marker definition.

The create panel will allow the user to select a set of nodes to create target markers on. Click "Pick NODE(s)" to interactively select NODE(s) to create target markers on (using this method will automatically create a node set for the selected nodes.). c1 and c2 are quadrant colours for the target marker. By default c1, c2 are set to black and yellow colour respectively. Radius controls marker size in either model or screen space units. To change between model and screen space display go to "Option-> Program->Options->Category->Target Marker".



MODIFY Modifying the attributes of an existing target marker.

MODIFY functions in the same way as **CREATE**, except that an initial definition will be present. Any modifications made to the target marker definition will not be made permanent until the **UPDATE** button is pressed. At this point the local copy which has been updated is used to overwrite the version in the model.

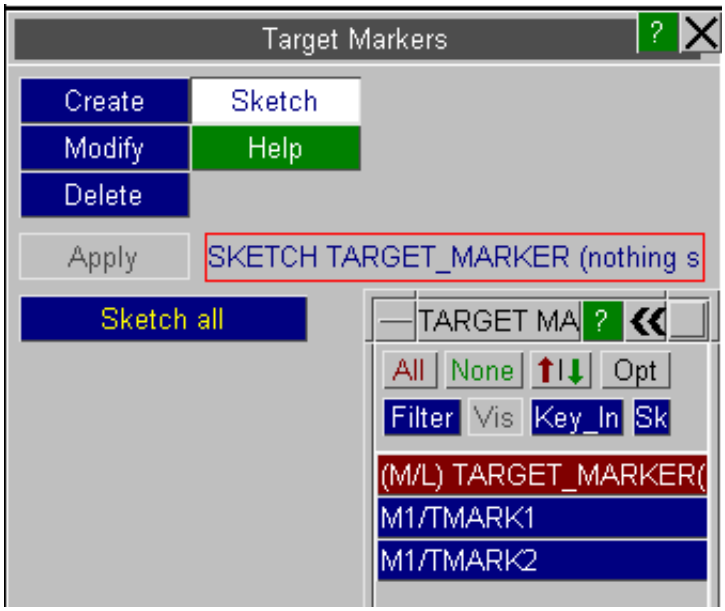
DELETE Delete existing target markers.

The selected target markers are deleted.

Target markers do not "own" anything, so the concept of recursive deletion does not apply, however target marker that is referred to (ie "owned") by some higher order item will not be deletable unless that item is deleted too, or its reference to the target marker.

SKETCH Sketch target marker.

SKETCH draws the target marker on top of the current graphics image.



6.34.4 TRANSFER DATA

Transferring data between "source" (reference) and "target" models.

The **TRANSFER_DATA** function is designed to copy properties into model A (the "target" model) from model B (the "source"). The intention is that a model which has had some of its content stripped by a journey through an external piece of software, for example during remeshing, can be re-united with its original properties in a simple operation. Alternatively models from diverse sources can be "married up" with standard definitions held in read-only files.

- Most of the parameters in this panel can be preset in the "oa_pref" file - see below.
- In addition this operation can be performed in dialogue-only mode, and hence in batch - see Appendix X11.

The function is invoked from the pop-up menu **Other** in the **Tools** panel.

This figure shows the main **TRANSFER_DATA** panel in its initial state.

It is mandatory that you define the following parameters:

- **Target model** (into which it is transferred)
- **Source model** (from which data is extracted)

The source and target models must be different.

You may then define your **Data type to transfer** from the options below. Any permutation of these can be selected.

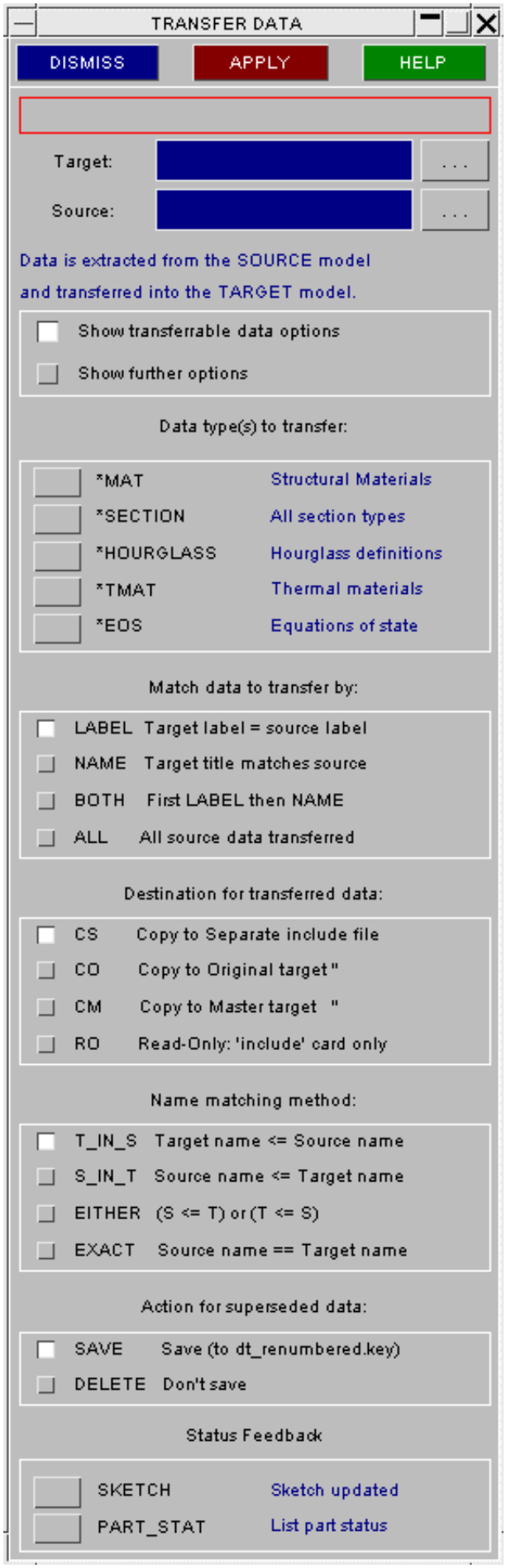
***MAT**	Structural materials
***SECTION**	Element sections
***HOURGLASS**	Hourglass definitions
***TMAT**	Thermal materials
***EOS**	Equations of state

Under further options, you may define one or more types for re-population of missing data from the following list.

DISCRETE	Spring/damper
JOINT	Joint props
WELD/RIVET	weld/rivet data
NODE_SET	node set data
NODAL_RB	nrb data

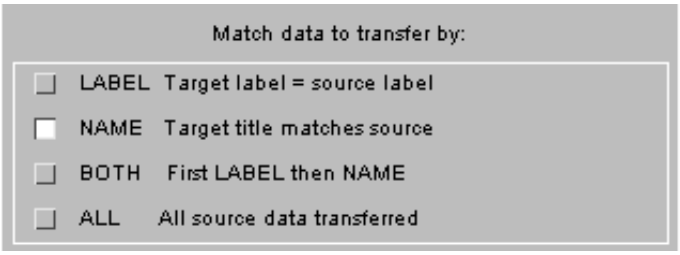
Once you have defined any one of these the **APPLY** button will become "live" and you will be able to proceed. However you should also consider the following settings:

Match data	Determines how data in the source model is selected for transfer
Destination	Determines how the transferred data is treated and stored in the target model
Name Matching	If you are matching data by **NAME** (or **BOTH**) then you should also consider which Name Matching Method to use. Name matching rules Explains the rules for case sensitivity, white space handling, etc when objects are matched by **NAME**.
Superseded data	Controls whether superseded (overwritten) objects in the target model are **SAVE**d or **DELETE**d.
Status Feedback	Controls how much visual and listing feedback about the transfer operation is given to the user.



Match Data to transfer by...

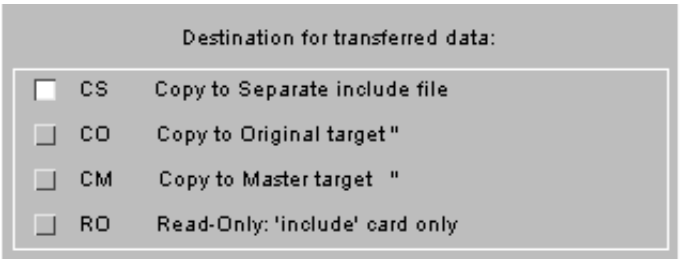
There is a range of ways in which data can be matched for transfer:



LABEL	If the labels in source and target models match
NAME	If the name in the target model is equal to, or a subset of, the name in the source model. This requires that the objects to be matched have titles, which is possible using: • The _TITLE keyword suffix in LS960 and above • The $PR_TITLE comment line in earlier versions
BOTH	Objects are matched first by LABEL, then by NAME
ALL	All data in the source model is transferred into the target.

Destination for transferred data...

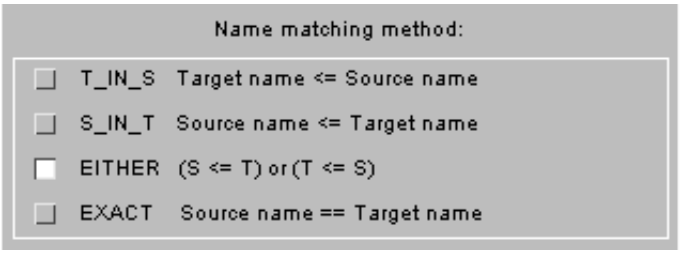
When data is transferred from source to target you can control how it is treated and stored.



CS (Copy to Separate include file)	Moves all transferred data to a new include file called **`dt_transfer_from_<source>.key`**
CO (Copy to Original include file)	All transferred data remains in the same include (or master) file as the original target data that has been overwritten.
CM (Copy to Master file)	All transferred data is moved into the master target file, regardless of its original location.
RO (Copy as read-only)	This assumes that the source file will be included verbatim in the ultimate keyword file that is analysed, so the following actions are taken: • Transferred data is placed into a special include file in the target model, marked as "read only". • When the target model is eventually written out this file is not included. • However the original <source> filename is referred to in an *INCLUDE statement via its full pathname. This means that the <source> file must be available on the computer that ultimately runs the analysis, using the same pathname.

Name Matching Method...

When objects are matched by **NAME** (or **BOTH**) the methods used to determine whether or not names match must be set. This option only becomes "live" when that is the case.



T_IN_S (Target IN Source)	The ***target*** name must equal the ***source*** name, or be a subset of it. For example: • Source name = "**MAT_123**" • Target name = "**123**" Would be matched, since "**123**" is a subset of "**MAT_123**"
S_IN_T (Source IN Target)	The ***source*** name must equal the ***target*** name, or be a subset of it. Therefore the example above would not be matched since "**MAT_123**" is not equal to or a subset of "**123**"
EITHER (T_IN_S or S_IN_T)	***Source*** may be a subset of ***target***, or ***target*** of ***source***, or they may be equal. Therefore the example above would be matched on the **T_IN_S** basis.
EXACT (Source == Target)	The ***source*** and ***target*** names must be identical. (No subset matching).

More rules for name matching...

When names are matched, by whatever method, the following rules also apply:

Case sensitivity	The matching process is always case-***in***sensitive (case is ignored). Therefore "**ABC**" matches "**abc**" matches "**AbC**" matches "**aBc**" etc
Leading and trailing spaces	Leading and trailing "white space" is always removed before matching takes place. Therefore " **ABC** " and "**ABC**" will match. (But "**ABC DEF**" and "**ABCDEF**" will not - the spaces are embedded.)
Empty names	Names that are empty are always ignored. So unnamed objects will never be matched.

Superseded Data: what happens when target objects are to be overwritten.

Transferring data from source to target models means implicitly that the target data will be overwritten. You can choose to:

SAVE The original target data is copied to a new, unused label and is transferred to include file "dt_renumbered.key". This new copy is not referenced by anything and can be removed by a **CLEANUP_UNUSED** operation.

DELETE The original target data is overwritten and lost permanently.

Status Feedback: controlling visual and tabular feedback

A summary of what has been overwritten with what (by label) is always printed, together with totals, but you can choose to have further feedback:

SKETCH Sketches on the current image all objects that have been modified by the data transfer operation

PART_STATUS Produces a table (by label) of all parts that have been modified (or have had some attribute, eg material, modified); and also a table of those that are unchanged.

How data transferred by **NAME** gets special treatment

When data is transferred by **NAME** it is possible that more than one object in the target model will match an object in the source model, in which case multiple copies of the source data will be transferred into the target model. For example consider the following case for materials where **EITHER** name matching is used:

Original target model contents	Source model contents	Resulting output in target model
`MAT #1 "Steel 316s12"`	`MAT #1 "Aluminium"`	`MAT #2 "Generic steel"`
`MAT #2 "Steel to BS4360"`	`MAT #2 "Generic steel"`	`MAT #2 "Generic steel"`
`MAT #3 "Special steel for supports"`	`MAT #3 "Concrete"`	`MAT #2 "Generic steel"`

In this case all three target materials contain the word "**`steel`**", so they will all be overwritten by material #2 (**`Generic steel`**) from the source model. This would leave three materials in the target model with the same label, so how is this resolved? What happens is this:

Every existing target object that is about to be overwritten has its properties copied to the highest object label + 1.

In this case since superseded data is to be **SAVE**d, before overwriting the "old" source material properties:

- Original target material #1 is copied to #4.
- Material #2 is copied to #5
- Material #3 is copied to #6

These new (#4 .. #6) definitions are not referenced by anything, and are placed into a separate include file. If they are not required a **CLEANUP_UNUSED** operation will delete them.

Source object data is copied in verbatim

- Source material #2 is copied verbatim into target materials #1, #2 and #3
- At this stage there will be 3 materials in the target model all labelled #2 - clearly illegal

A post-transfer check and sorting out takes place

Once all transfers by **NAME** have taken place then any clashes or illegalities in the target model are sorted out.

- Where any incoming label from a source object clashes with a pre-existing (but not overwritten) target object of the same label, then that original target object is re-labelled.
- Then a check for multiple instances of transferred in objects is made (which will all share the same label), and these are "collapsed" into a single definition.

In the example above the three target materials, all labelled #2, would be collapsed into a single material definition and all references to them (eg from *PART cards) are implicitly also rationalised to reference this single material definition.

How are objects such as loadcurves which might be referenced by transferred data handled?

For example you might transfer a material which contains references to loadcurves in the source model.

In this situation these "supporting" objects are also transferred, although the logic for handling label clashes in the target model differs from the data above:

- If there is no label clash they are copied in "as is".
- If there is a label clash the original definition in the target model is renumbered to a free slot, and the new data is copied in.

This is different because although existing objects in the target model might be relabelled, they are not superseded.

Setting **TRANSFER_DATA** defaults in the "oa_pref" file.

Most of the settings in the **TRANSFER_DATA** panel can be pre-defined in the "oa_pref" file. The following table shows the default settings, and the ways in which they can be modified in that file:

Setting	Default value	"oa_pref" file keyword	"oa_pref" options
Target model	*Undefined*. But if only one model is present then this is assumed	n/a	n/a
Source model	*Undefined*	**`primer*transfer_source_file:`**	***`<filename>`***
Data Type	*Undefined*	**`primer*transfer_data_type:`**	• MAT • SECTION • EOS • HOURGLASS • TMAT
Match By method	**NAME**	**`primer*transfer_match_by:`**	• ID • NAME • BOTH • ALL
Destination Action	**CS** (Copy to Separate)	**`primer*transfer_action:`**	• CS • CO • CM • RO
Name Matching method	**EITHER**	**`primer*transfer_name_match:`**	• T_IN_S • S_IN_T • EITHER • EXACT
Superseded data action	**SAVE**	**`primer*transfer_superseded:`**	• SAVE • DELETE
Feedback options	**SKETCH** (only in graphical mode) **PART_STATUS**	n/a	n/a

Performing **TRANSFER_DATA** operations in command-line mode.

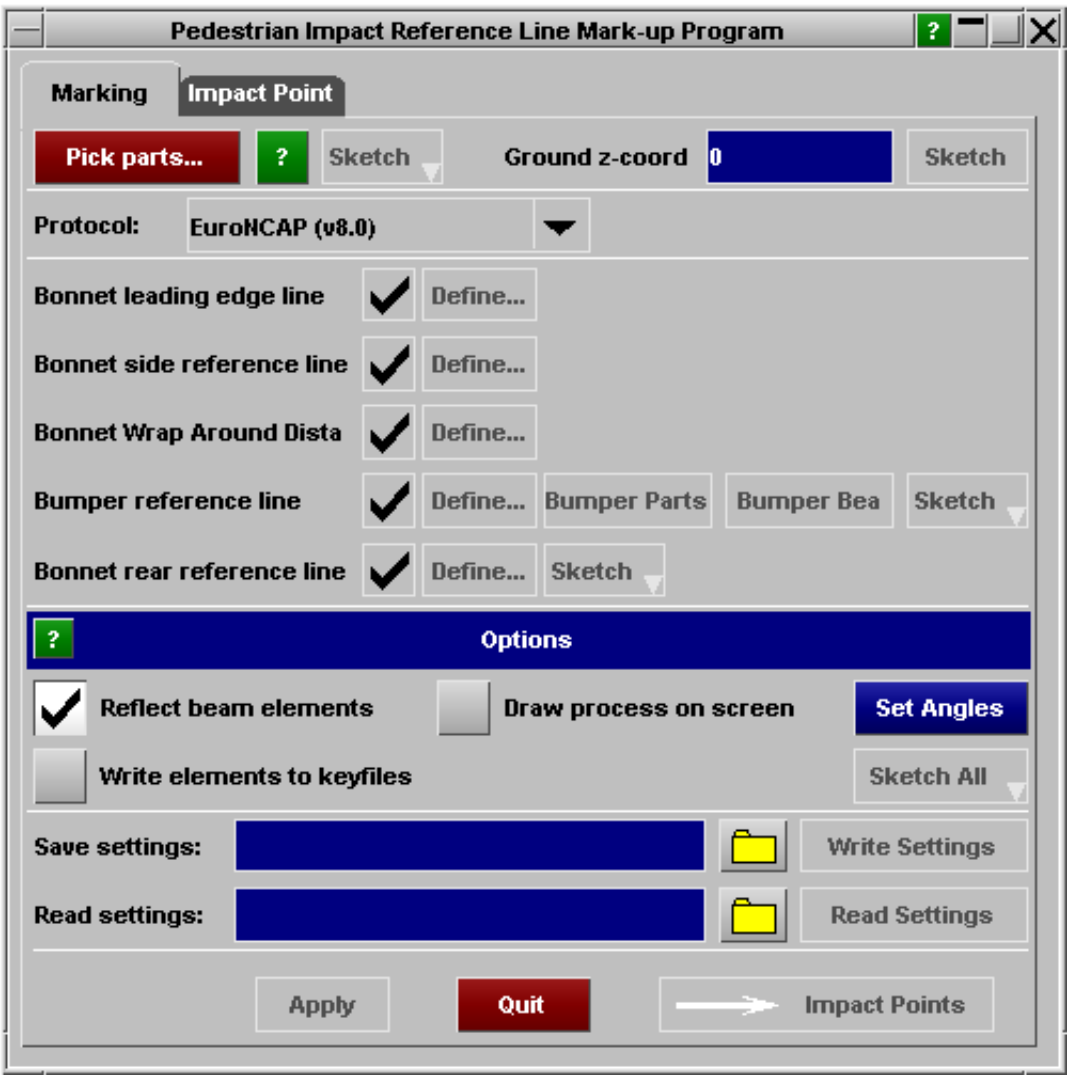
All the operations above can also be performed in command-line mode, and therefore also in batch. See Appendix X11 for more information about this.

6.35 PEDESTRIAN MARKUP

6.35.1 Introduction

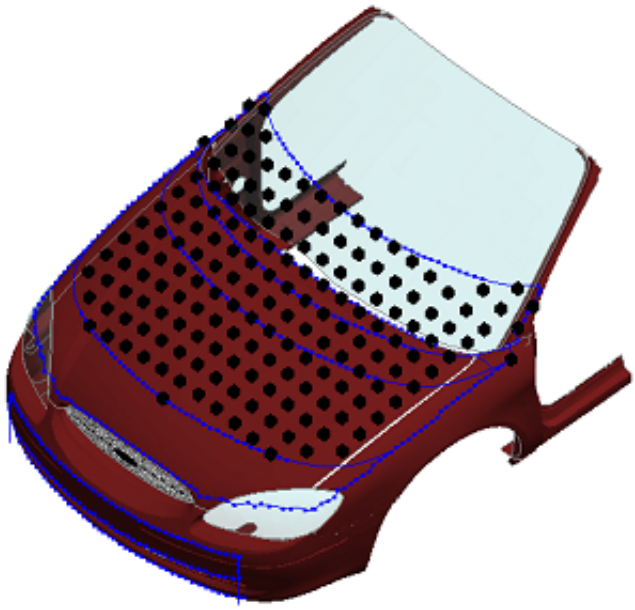
The pedestrian markup tool can be used to markup vehicles and create multiple pedestrian impact models according to rules defined in different pedestrian safety protocols.

Currently the following protocols are supported:
- EuroNCAP v5 - v8.2
- GTR

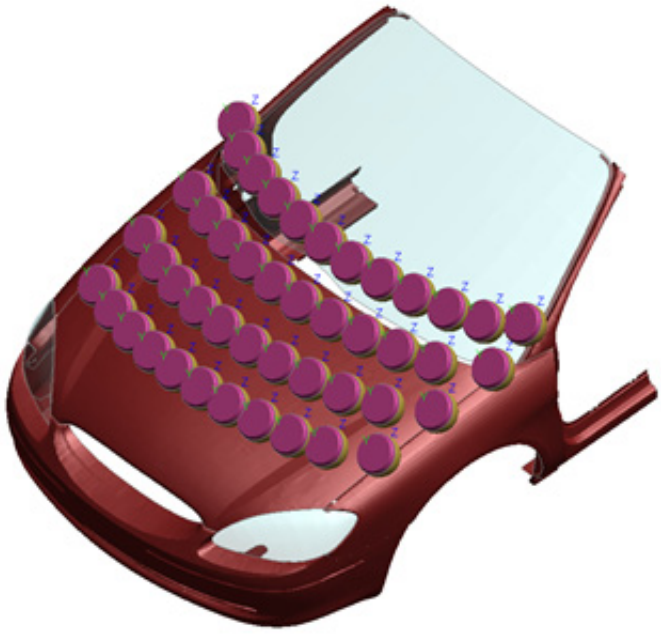


The vehicle model is marked up according to the selected protocol

Impact points can then be automatically created for head, upper leg and lower leg impacts.

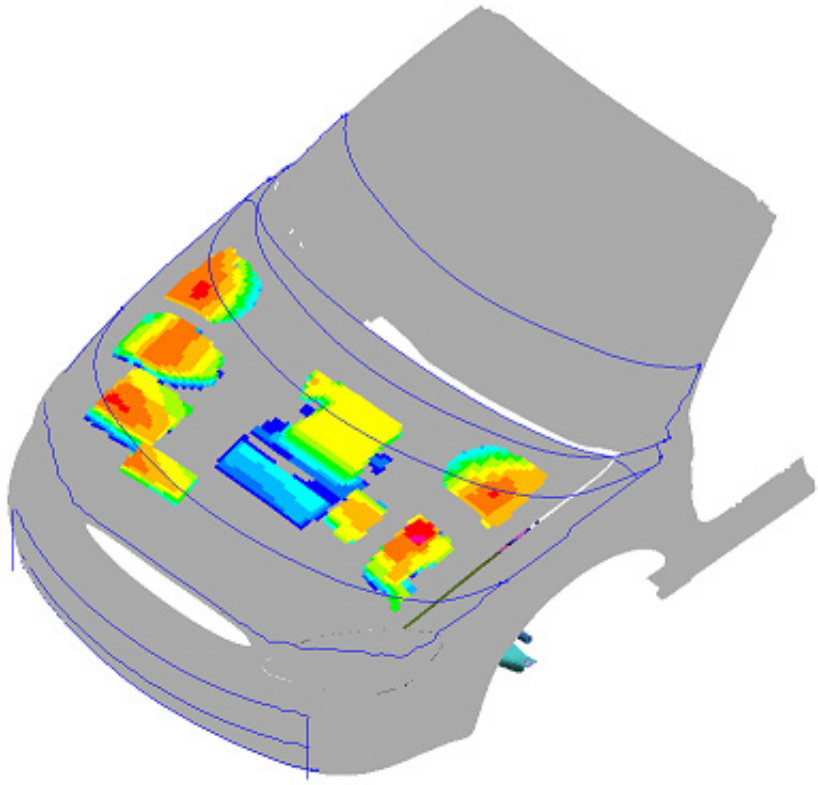


Multiple models can be built ready for analysis.

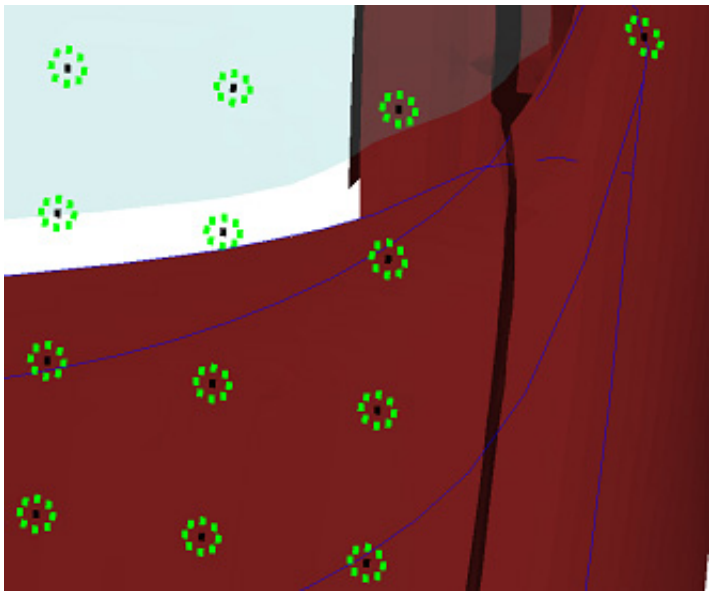


For head impacts PRIMER can automatically detect points close to hard parts under the surface of the bonnet. Points can also be created manually.

The distance from the outer surface to the hard parts can also be CT plotted.



For robustness studies points can be created in a ring around other points.



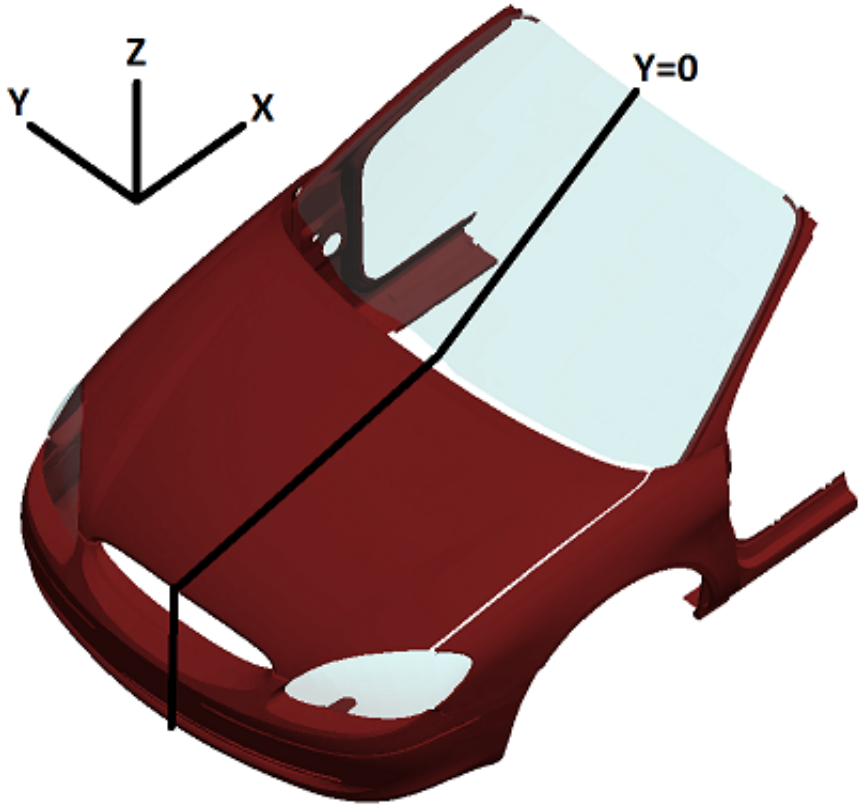
6.35.2 How to create markup lines

The markup procedure assumes the model is in mm. If not you should convert it so that it is.

First read in the vehicle model you want to markup. It needs to be oriented correctly for the script to work. It must be aligned with the global co-ordinate system:

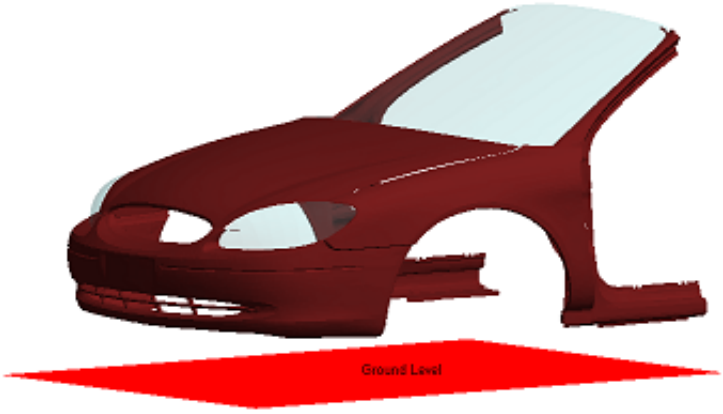
- X from car front to back
- Y from car left to right
- Z from the ground up

The centre line along the vehicle needs to be at y=0.



With the model in the correct orientation, press the **PICK PARTS...** button and select the **OUTER** parts of the vehicle.

Next, set the ground Z-coordinate to the correct height (By default it's at 0). Press **SKETCH** to view it relative to the vehicle.



Now chose the protocol you want to use to markup the vehicle.

If you have selected a protocol that requires a Bonnet Rear Reference Line then you will need to press the **DEFINE...** button for that line (it should be red, to indicate that something needs to be done).

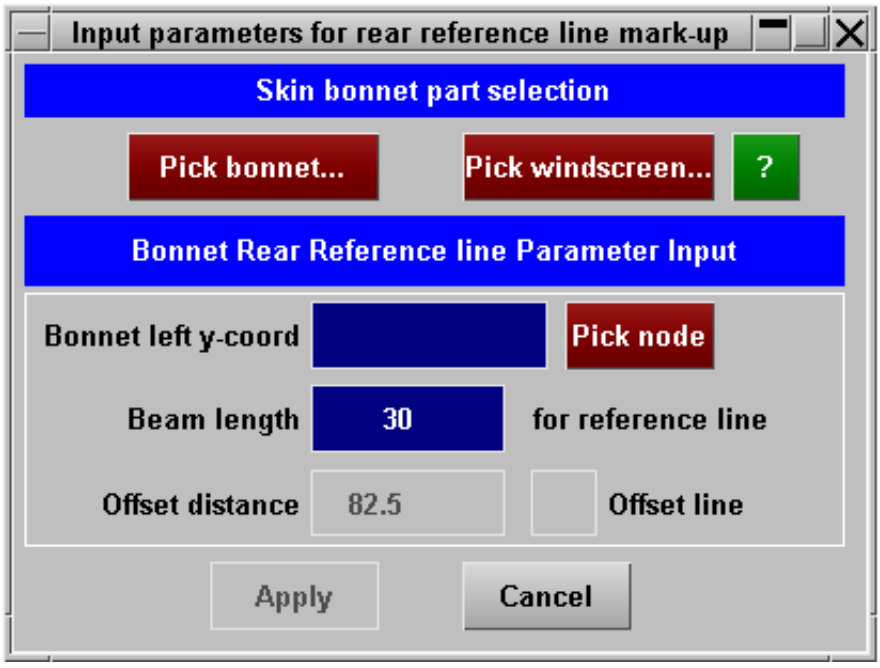
All the other **DEFINE...** buttons should be green to indicate that they have enough information to create the lines.

This will open a new window.

Press the **PICK BONNET...** button to select the **OUTER** bonnet part(s) (pick the cowl if you want to include this).

Press the **PICK WINDSCREEN...** button to select the **OUTER** windscreen part(s).

Once this is done press **APPLY**. The **DEFINE...** button for the Rear Reference Line should now be green.



The protocols state that for the upper bumper line, if the bumper is identifiable the reference line should be marked on those parts. Use the **BUMPER PARTS** button to select the parts that make up the bumper. If you don't select any parts the script will just use the parts selected at the top of the menu (i.e. it assumes the bumper is unidentifiable).

From v6 of the EuroNCAP protocol the extent of the bumper test zone is defined as the area limited by the bumper corners or the outermost ends of the bumper beam. Use the **BUMPER BEAM** button to select the parts that make up the bumper beam. If you don't select any parts the script will just define the test zone via the bumper corners. From v8 of the EuroNCAP protocol you must select the bumper beam parts as they are used to calculate the Internal Beam Reference Line.

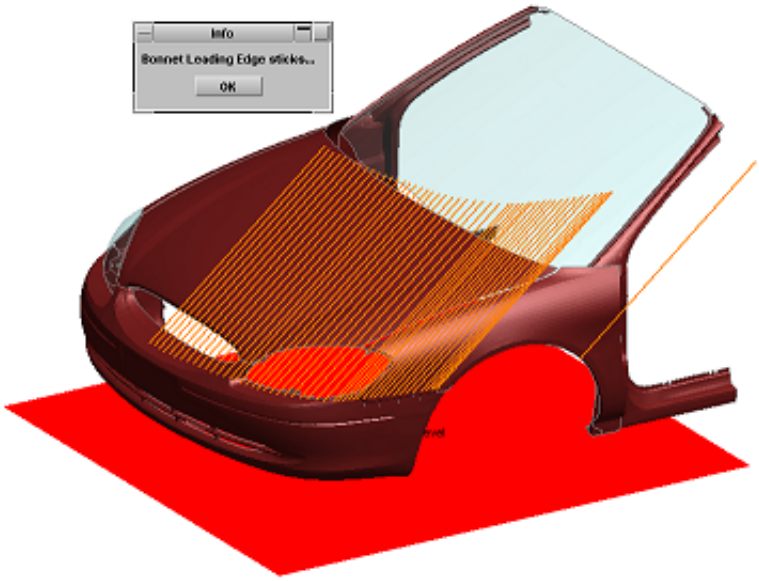
The **APPLY** button at the bottom of the main menu should now be active. Press this to create the markup the lines. They are created as beam elements in a new model.

6.35.2.1 Additional options

There are some settings that can be changed which control how the lines are created. Normally the default settings will be acceptable and you will not need to modify them. However, below is a description of each setting.

The markup lines are only calculated on the left hand side of the vehicle and reflected to the right hand side. You can turn off the reflection with this option.

To help with debugging you can turn this option on to visualise what the markup process is doing.



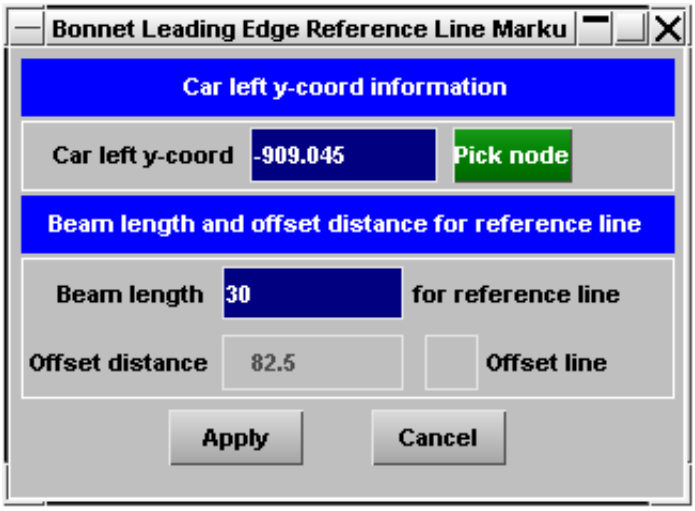
The markup lines are created as beam elements. Turning this option on will write out the elements to keyword files.

For each reference line there is a **DEFINE...** button where settings for that line can be changed. Normally you will not need to change any of the settings.

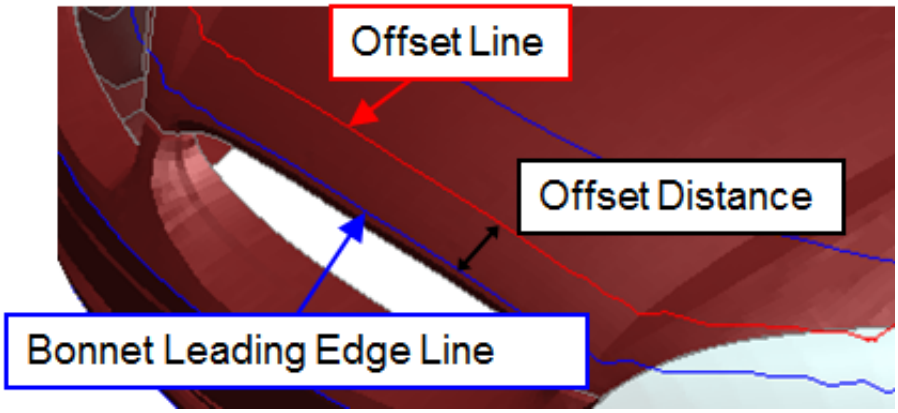
Bonnet Leading Edge:

The **Car left y-coord** is the coordinate of the left hand side of the vehicle. By default it will be set to the extreme left hand side of the selected outer parts, but you can change it if you wish. The line will be calculated up to this coordinate.

The **Beam Length** is the length the beams will be when creating the lines. A smaller length will give a more accurate line, but will take longer to calculate.



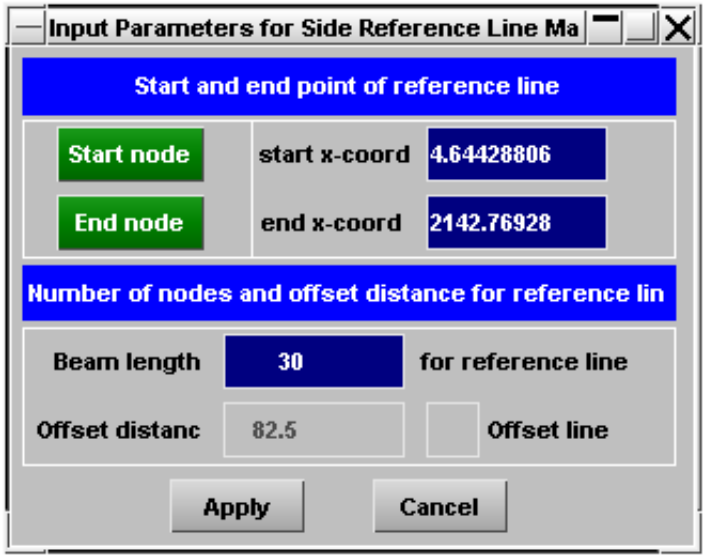
The **Offset distance** is only valid for some protocols. If selected an additional line will be created, offset from the main line and head impact points will only be created up to the offset line.



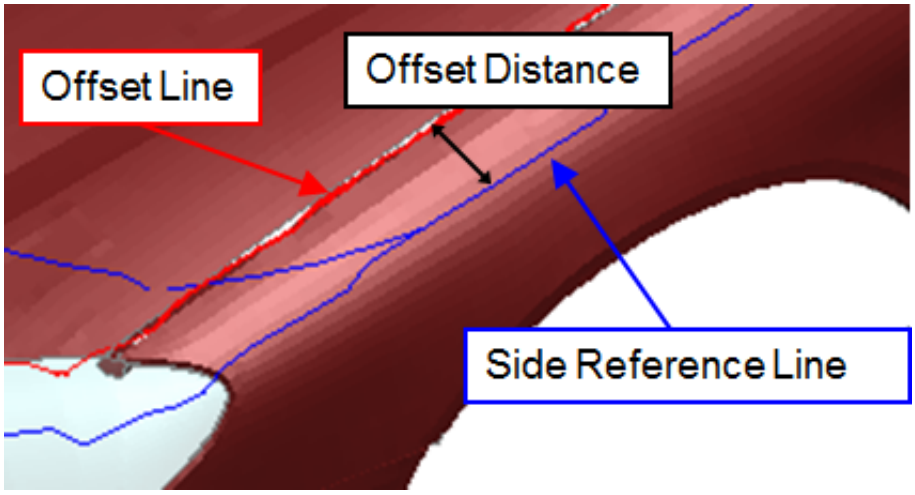
Side Reference Line:

The **Start x-coord** and **End X-coord** are the coordinates of the front and back of the vehicle. By default they will be set to the extreme front and back of the selected outer parts, but you can change it if you wish. The line will be calculated between to these coordinates.

The **Beam Length** is the length the beams will be when creating the lines. A smaller length will give a more accurate line, but will take longer to calculate.

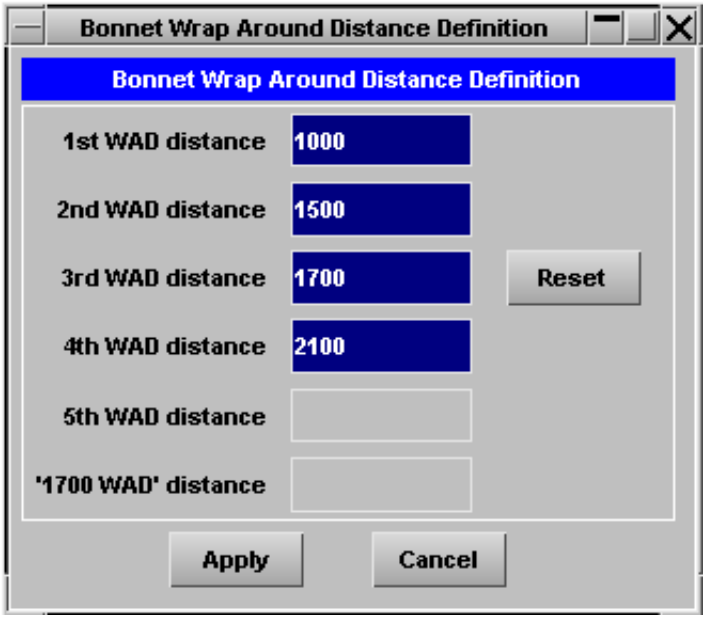


The **Offset distance** is only valid for some protocols. If selected an additional line will be created, offset from the main line and head impact points will only be created up to the offset line.



Wrap Around Distances:

The wrap around distances can be modified here. The number of WADs will depend on the protocol selected.

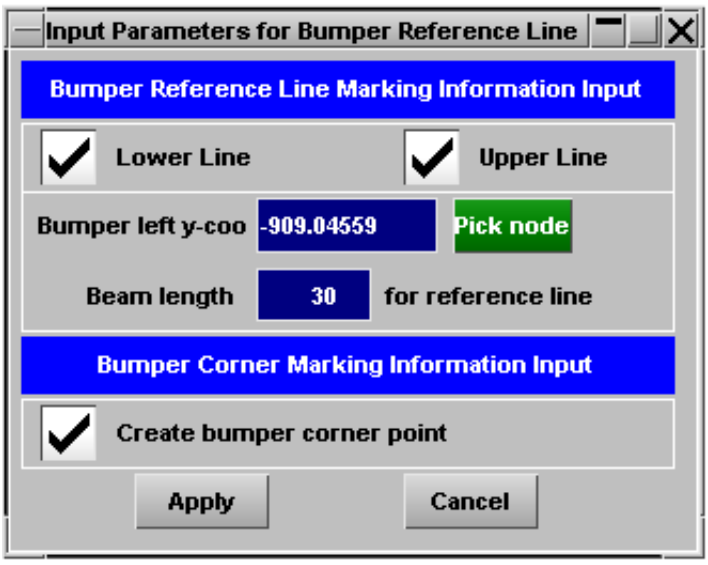


Bumper Reference Lines:

The **Bumper left y-coord** is the coordinate of the left hand side of the vehicle. By default it will be set to the extreme left hand side of the selected outer parts, but you can change it if you wish. The line will be calculated up to this coordinate.

The **Beam Length** is the length the beams will be when creating the lines. A smaller length will give a more accurate line, but will take longer to calculate.

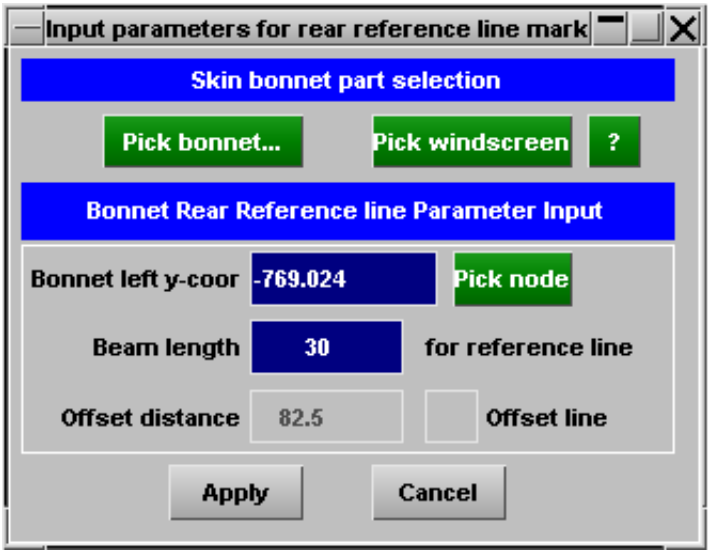
The calculation of the **Lower Line**, **Upper Line** and **Corner point** can be turned off by unticking the checkboxes.



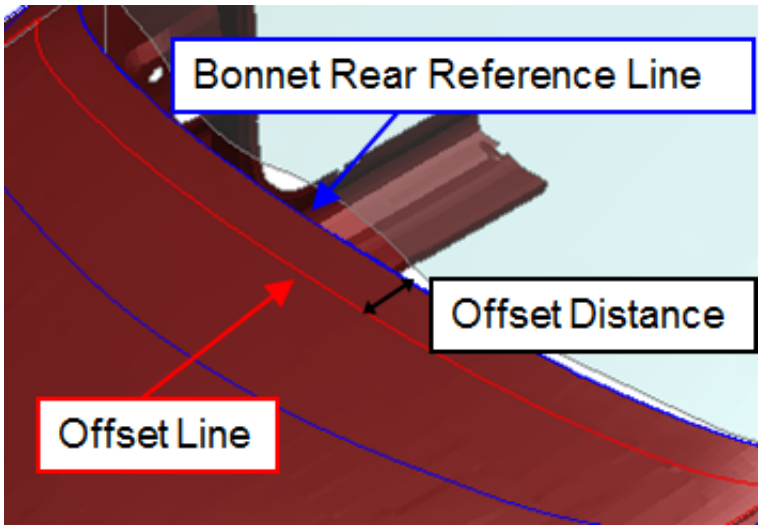
Rear Reference Line:

The **Bonnet left y-coord** is the coordinate of the left hand side of the bonnet. By default it will be set to the extreme left hand side of the selected bonnet parts, but you can change it if you wish. The line will be calculated up to this coordinate.

The **Beam Length** is the length the beams will be when creating the lines. A smaller length will give a more accurate line, but will take longer to calculate.



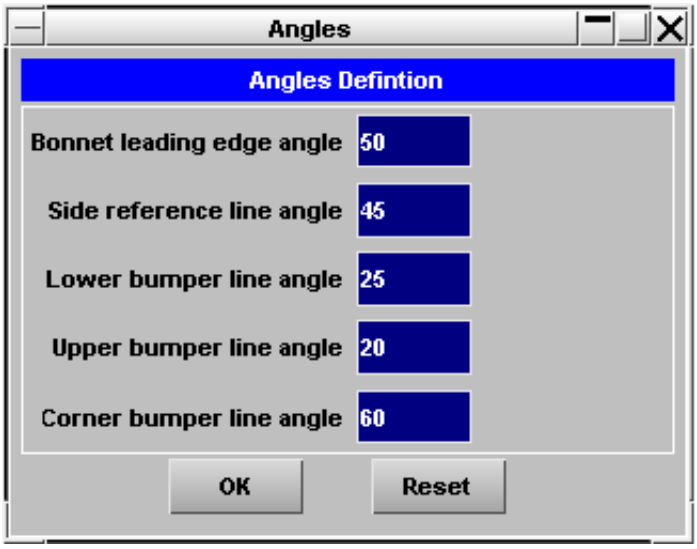
The **Offset distance** is only valid for some protocols. If selected an additional line will be created, offset from the main line and head impact points will only be created up to the offset line.



Angles

If you press the **SET ANGLES** button you can change the angles of the sticks used to markup the lines.

This can be used for to see how sensitive your results are to changes in how the lines are marked up.



Read/Write Settings

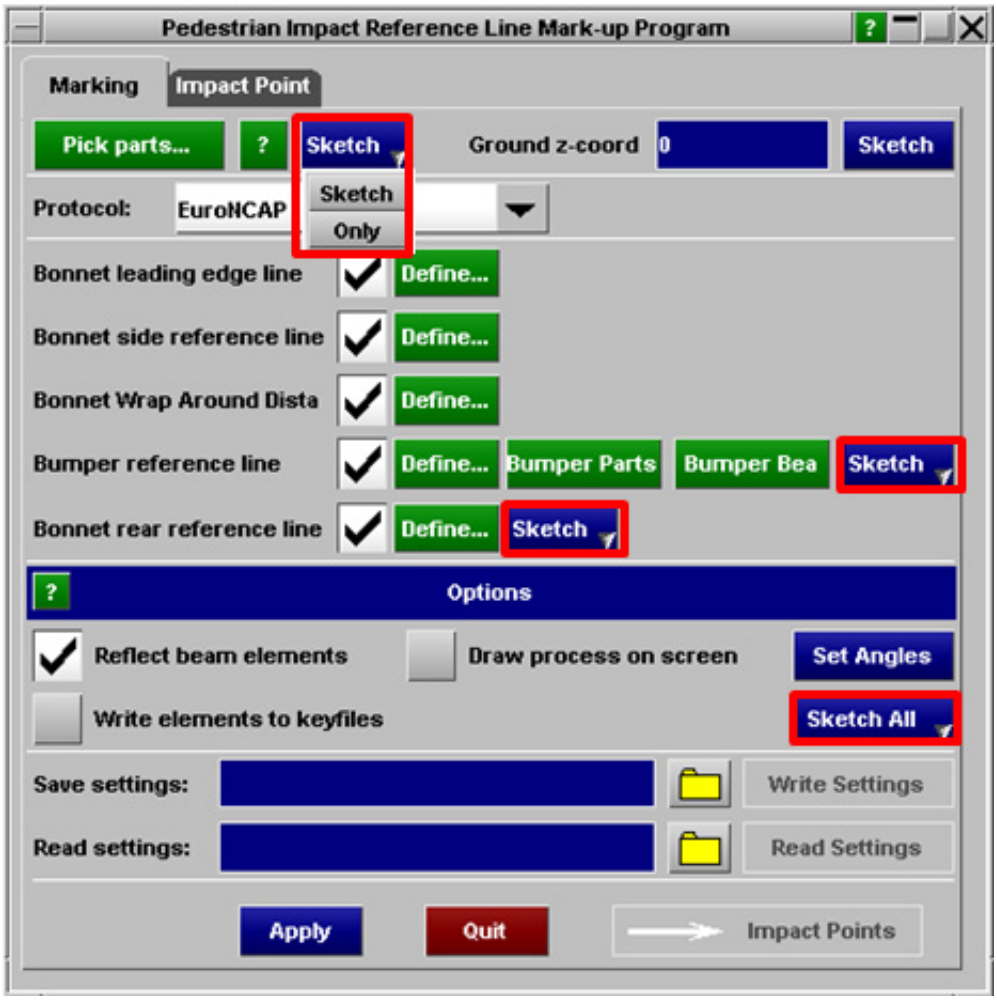
The settings used to create the lines can be saved to a file and read in later sessions of PRIMER.

All the settings described above are saved to the file, e.g. protocol, outer part IDs, Z ground coordinate, beam lengths, angles etc.

It is a simple CSV file so could be created from an external source if you would like.

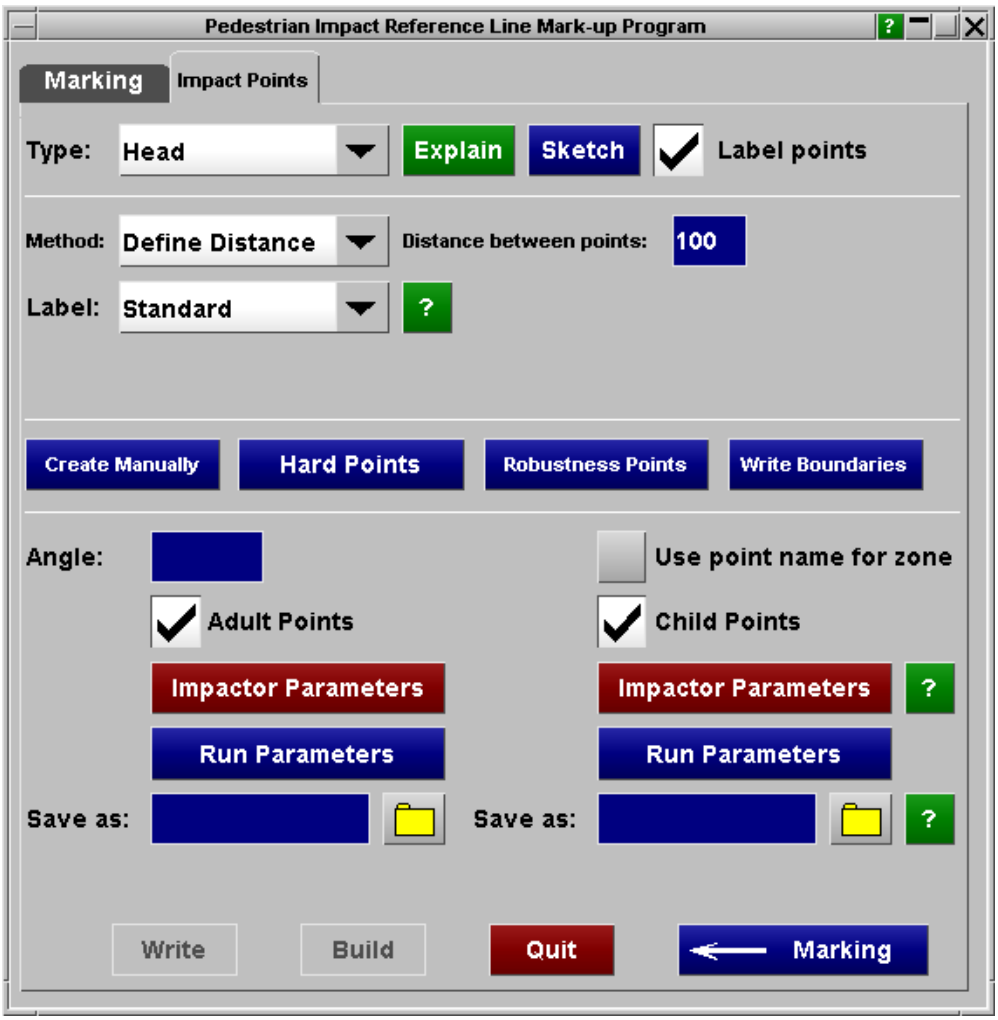
Sketch/Only Parts

The selected parts can Sketched or Only'd to view what has been selected.



6.35.3 How to create models

Once the markup lines have been created, the main menu will update to a new tab where you can create impact points for head, upper leg and lower leg impactors and build models.



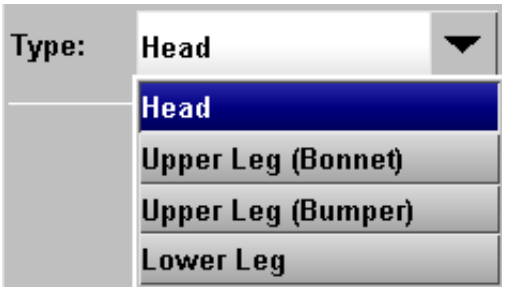
Impact points can be created automatically and manually.

For head impacts PRIMER can automatically detect points close to hard parts under the surface of the bonnet.

For robustness studies points can also be created in a ring around other points.

6.35.3.1 Create head impact models

First set the **Type** to Head Impacts.



Automatically create points

You then need to set how you want to automatically create impact points. There are a number of ways to do this:

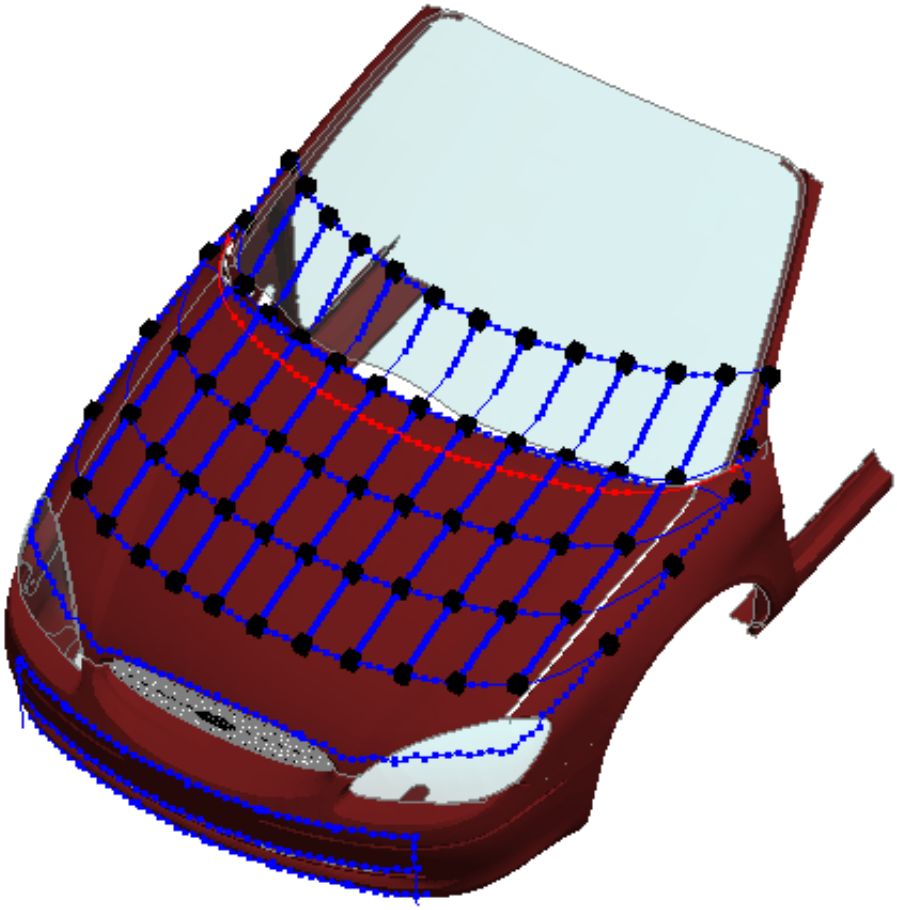
- Define a distance between points to create a grid
- Define NxN points per zone (EuroNCAP v5 only)
- Define NxM points per zone (EuroNCAP v5 only)
- Nothing

If you set the **Method** to Define Distance, you will need to select the distance between points.

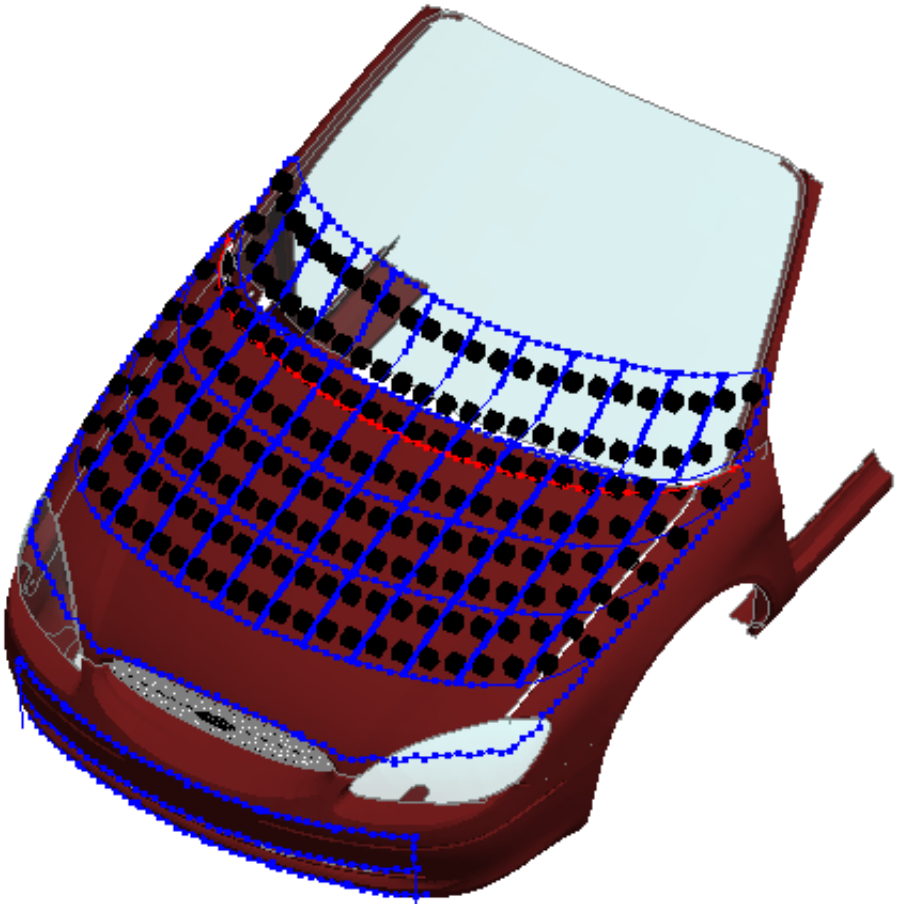
If you set the **Method** to N points, you will need to select the number of points and a spacing factor.

The spacing factor can be in the range 0-N, where 0 will put the impact points on the edge of the zone and N will put them all at the centre. Set it to 1 to space the points equally.

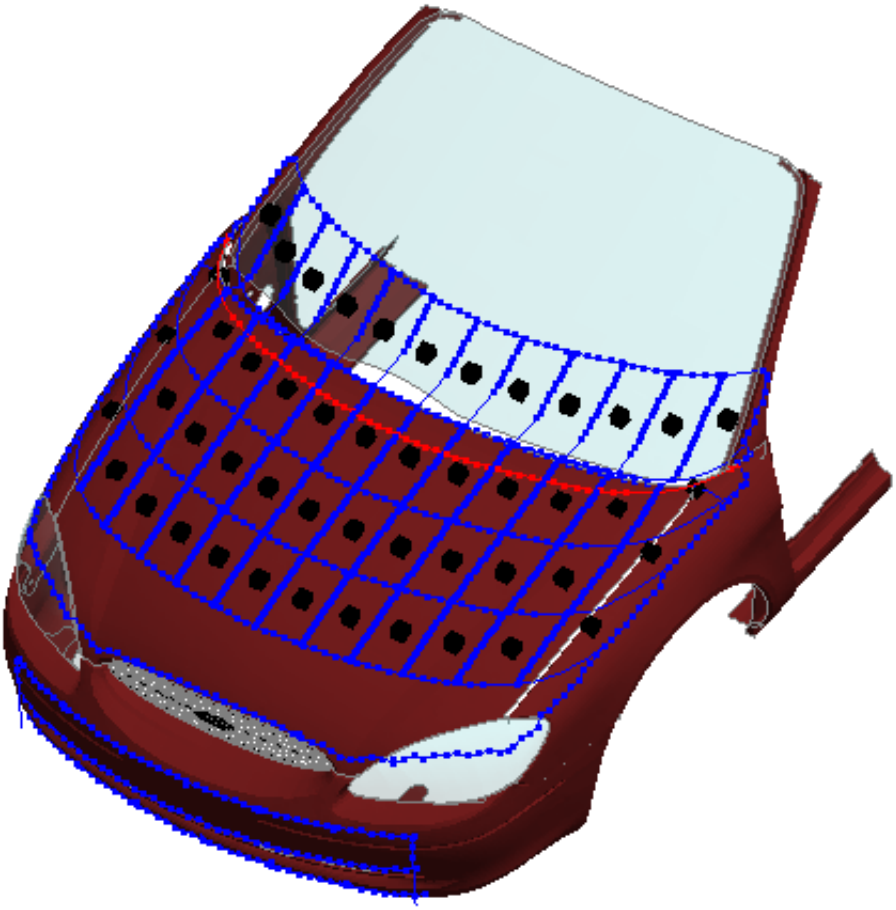
N=2, Spacing factor = 0



N=2, Spacing factor = 1



N=2, Spacing factor = 2



If you set the **Method** to NxM points, you will need to select the number of points and a spacing factor for both the rows and columns. The same rules as above apply for the spacing factor.

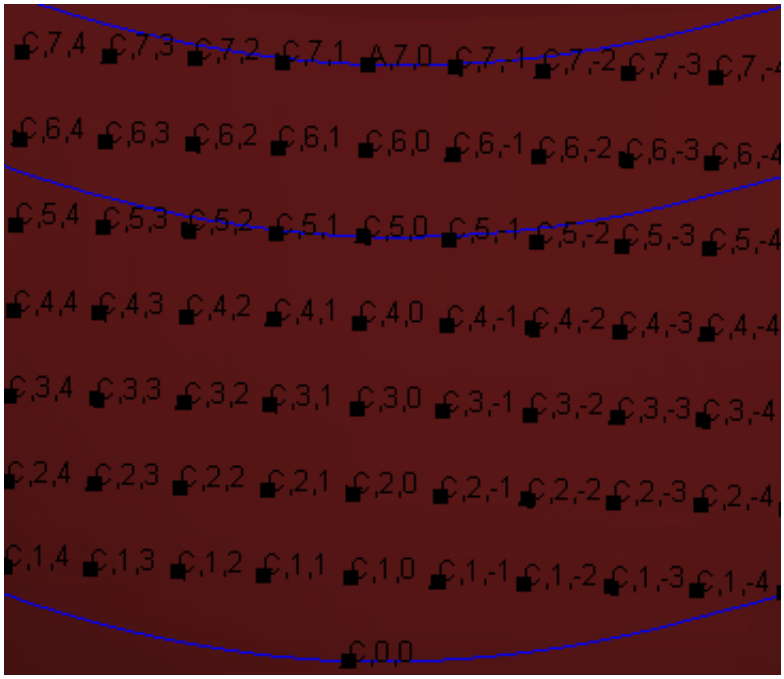
You can set the **Method** to Nothing, in which case no points will be created automatically.

To create models you will need to create points manually or detect points close to hard parts.

Labelling impact points

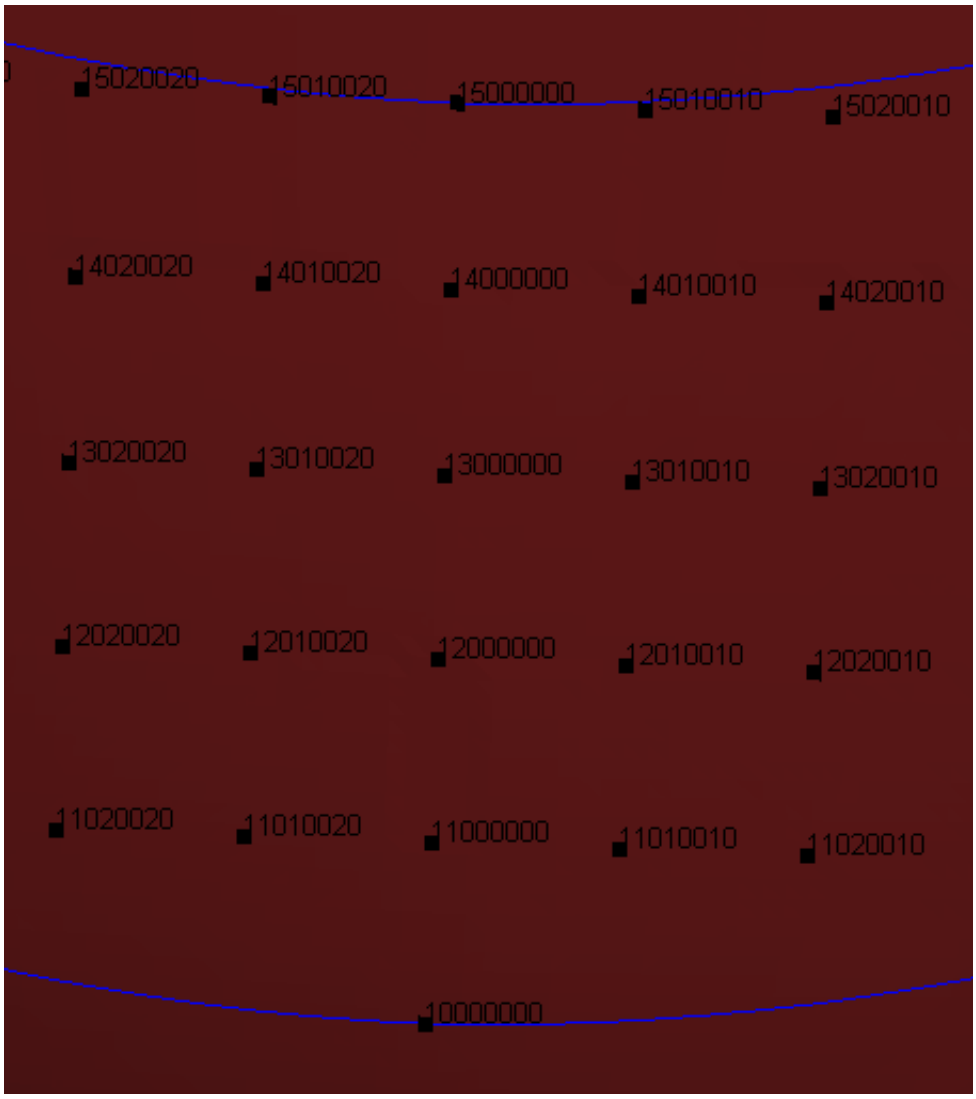
There are two methods for labelling head impact points:
- Standard
- GM Style

The Standard method uses the labelling rules defined in the selected protocol, e.g. for EuroNCAP v7



GM Style uses an 8 digit number [ABCDEFGH] to encode the location of the point:
- ABC = the first 3 numbers of the WAD
- DEF = the absolute value of the Y co-ordinate
- G = 0 if on the centre, 1 if on the left, 2 if on the right
- H = 0 for a target point

e.g. 12050010 is a target point at the 1200 WAD, 500mm from the centre on the left hand side.



If GM Style is used then all the *DEFINE_TRANSFORMATION definitions for each head impactor position are written to a common file, which needs to be specified here. Each *DEFINE_TRANSFORMATION is given the same label as the impact point.

In addition, the final transformation is the impact point coordinates.

A title for each *DEFINE_TRANSFORMATION can also be specified and is a combination of the impact point label, the specified string and the depenetration type (’aim point’ or ’target point’), e.g.

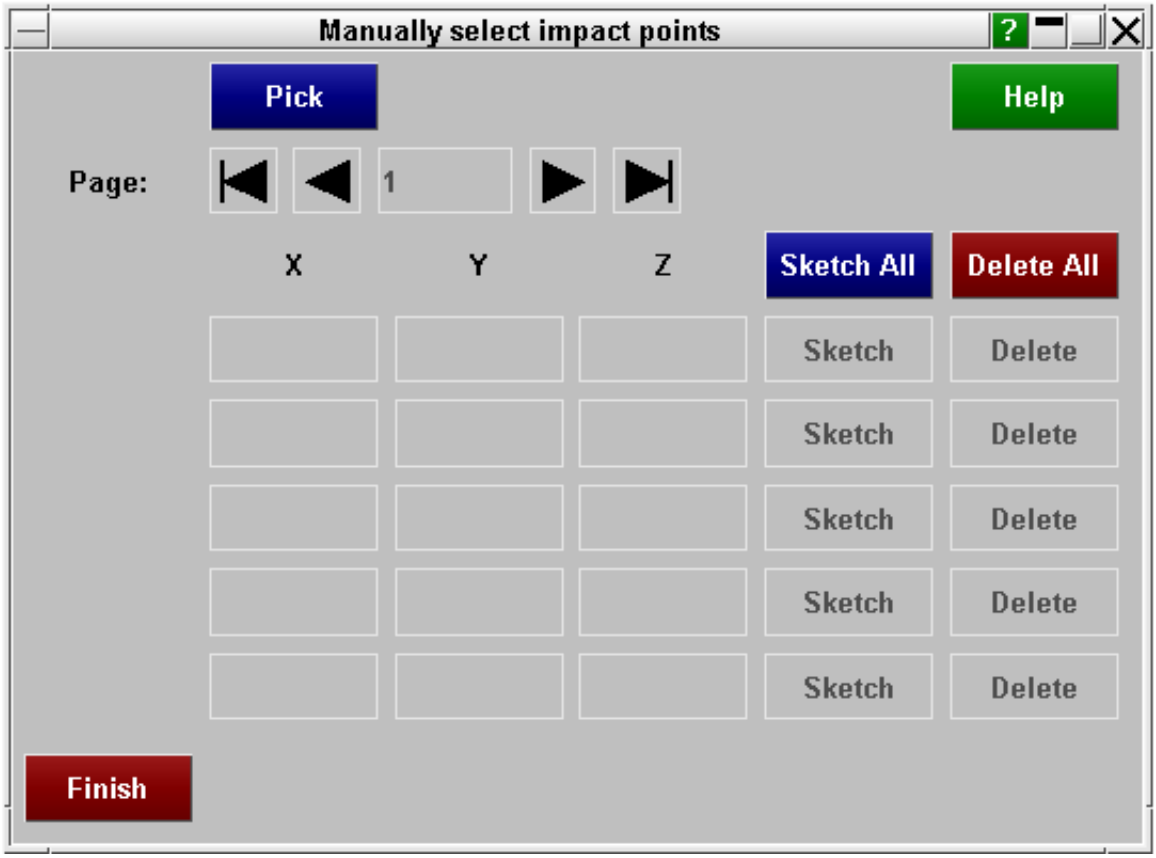
12050010 **project_x**: aim point

Some protocols specify an area on the windscreen where points can be defaulted to green (meaning they get the maximum score) without having to test them.

If you don’t want to create models for the points in this area you need to select some part(s) that define the area by pressing the **PICK DEFAULT GREEN PARTS** button.

Manually create points

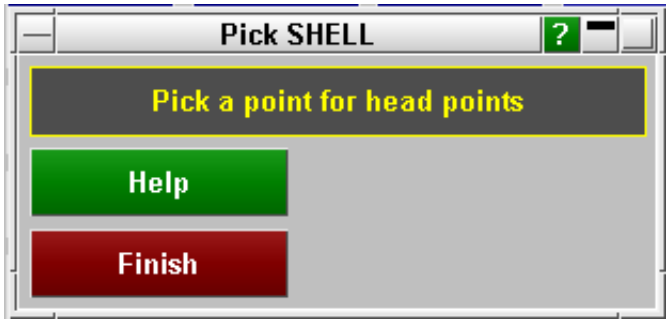
Pressing the **CREATE MANUALLY** button will bring up a menu where you can manually create impact points.

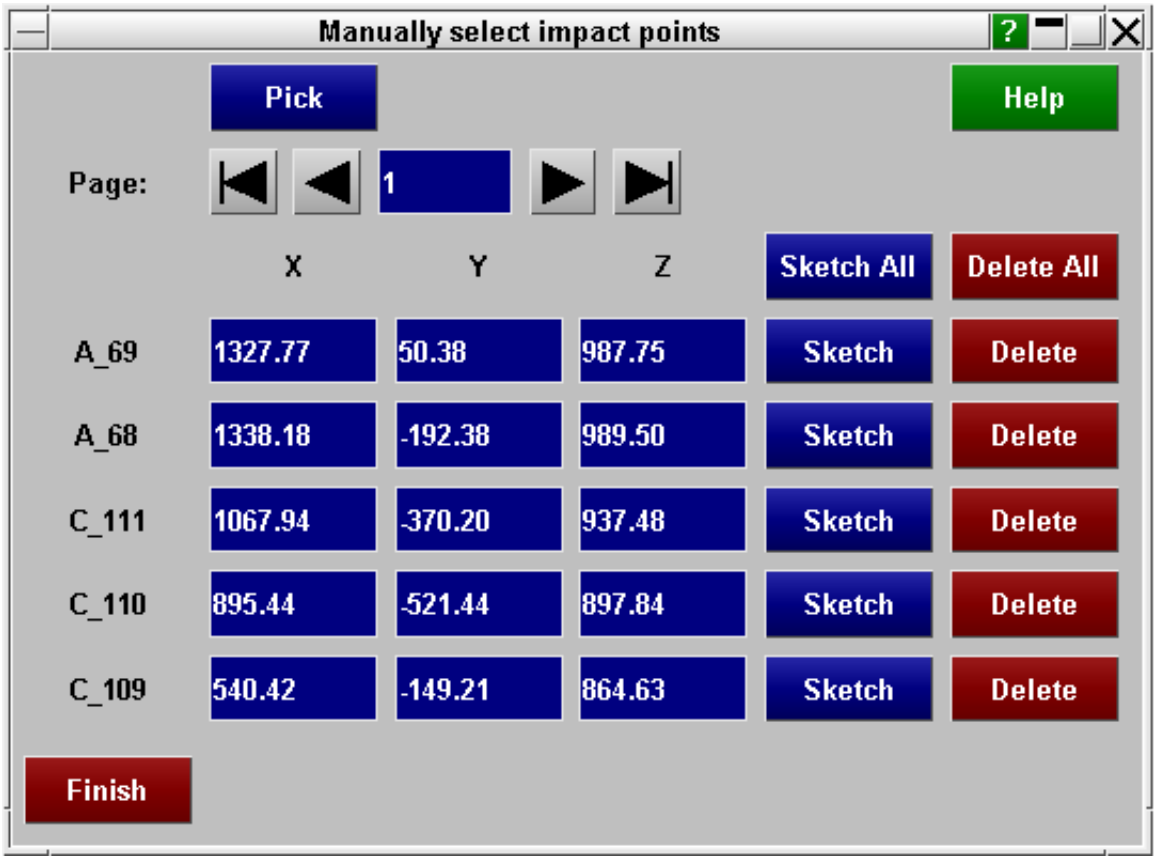


Press the **PICK** button to start picking points on the vehicle. If you select outside the markup line boundaries then a point will not be created.

Once you have selected all the points you want, press the **FINISH** button in the window that has popped up.

The menu will be filled with the labels and coordinates of the points created.



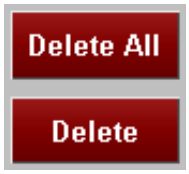


If you have created enough points you can move to different pages in the menu to view them.

You can also edit the coordinates by typing in the appropriate textboxes.

To view the points press **SKETCH ALL** or **SKETCH**

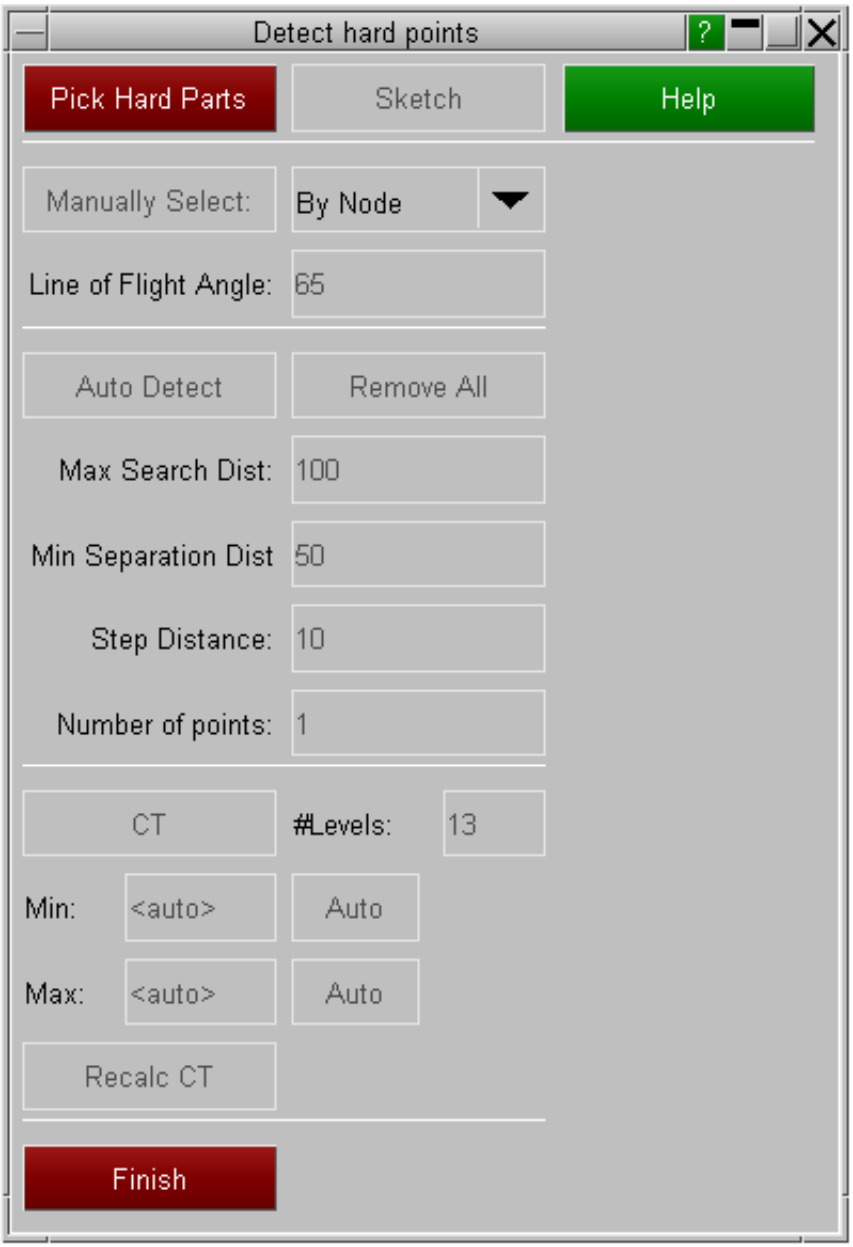
To delete points press **DELETE ALL** or **DELETE**



Once you have finished press the **FINISH** button to close the menu.

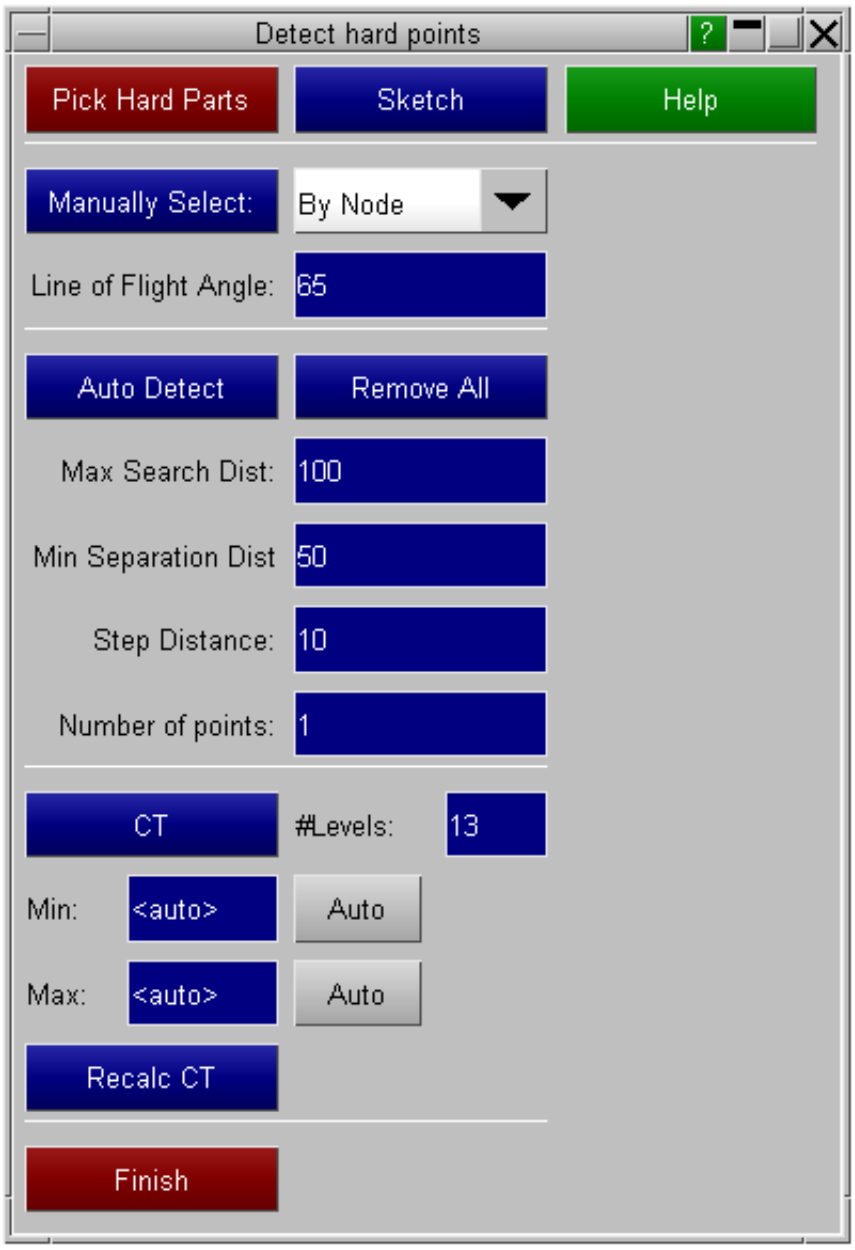
Create hard points

Pressing the **HARD POINTS** button will bring up a menu where you can create impact points close to hard parts under the bonnet surface.



Press the **PICK HARD PARTS** button to start picking parts under the bonnet surface. They can be Shell or Solid parts.

Once selected, the buttons in the menu should all become active.

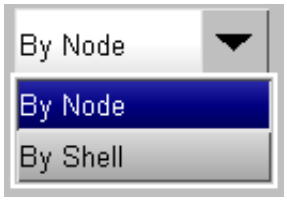


The menu is split into three sections

- At the top you can manually create points
- In the centre you can get PRIMER to automatically detect points
- At the bottom you can do a CT plot of the distance from the vehicle surface to hard parts

Manually find hard points

To manually create points first chose whether you want to select points on nodes or shells using the dropdown menu.



You should also set the line of flight angle of the head impactor. By default it is set to 65 degrees.

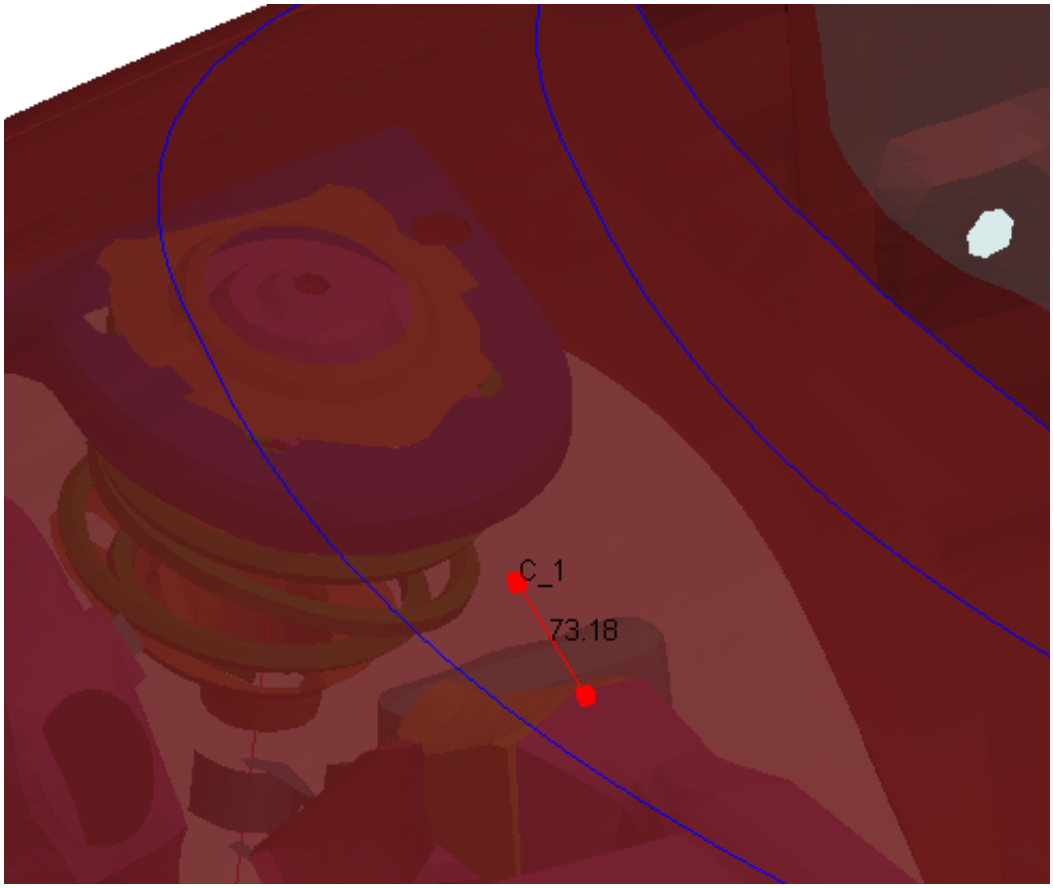
Now press the **MANUALLY SELECT** button.

You will now be able to select points on the surface on the vehicle or on the hard parts.

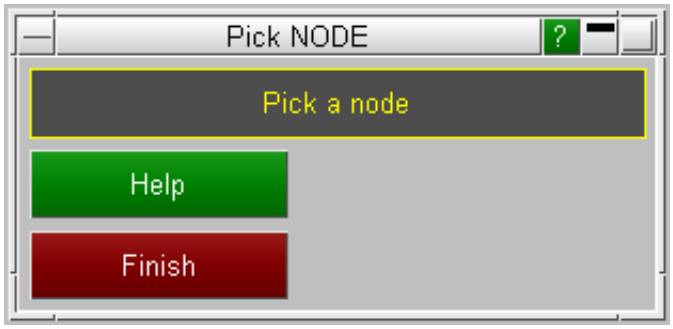
If you select a point on the vehicle PRIMER will project a line along the line of flight to calculate whether it will intersect with any of the hard parts selected. If it does it will create an impact point.

If you select a point on a hard part PRIMER will project a line along the line of flight to calculate whether it will intersect the vehicle surface. If it does it will create an impact point.

If a point is created it will be drawn on the screen along with the distance from the vehicle surface to the hard part along the line of flight:



Once you have selected all the points you want, press the **FINISH** button in the window that has popped up.



Automatically find hard points

To automatically find hard points press the **AUTO DETECT** button.

The following parameters define how the detection is carried out:

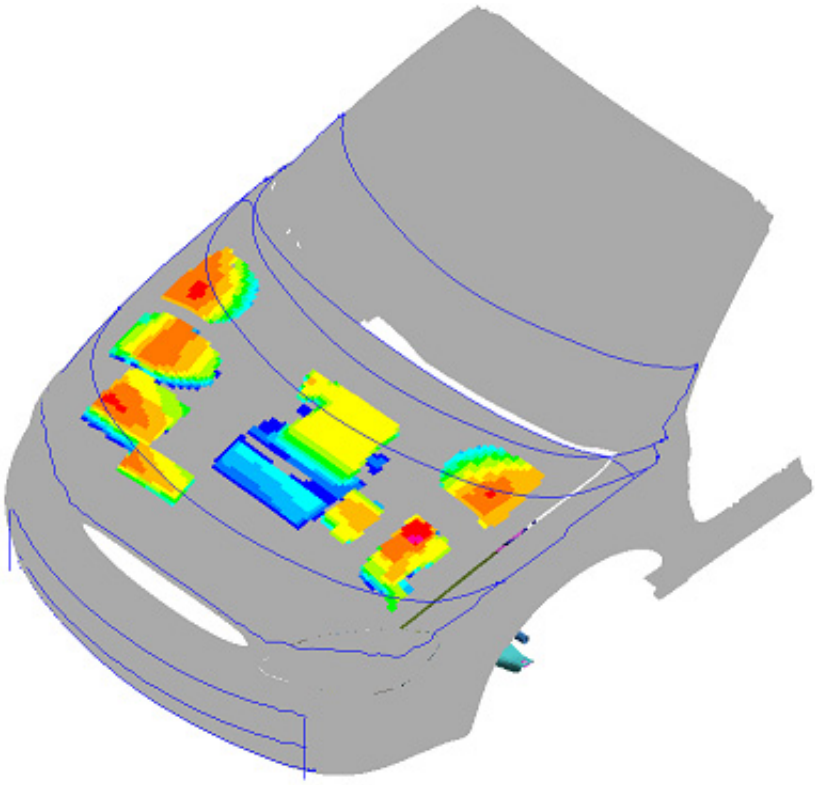
- **Max Search Dist**: The maximum distance to search for hard parts from the outer surface.
- **Min Separation Dist**: The minimum distance between successive points.
- **Step Distance**: The search is carried out along a grid. This is the distance between each grid point.
- **Number of points**: The number of hard points to look for.
- **Angle**: The line of flight angle.

If point(s) are successfully found they will be drawn on the screen along with the distance(s) from the vehicle surface to the hard part along the line of flight:

To delete any hard points created, press the REMOVE ALL button. This will delete both manually and automatically created points.

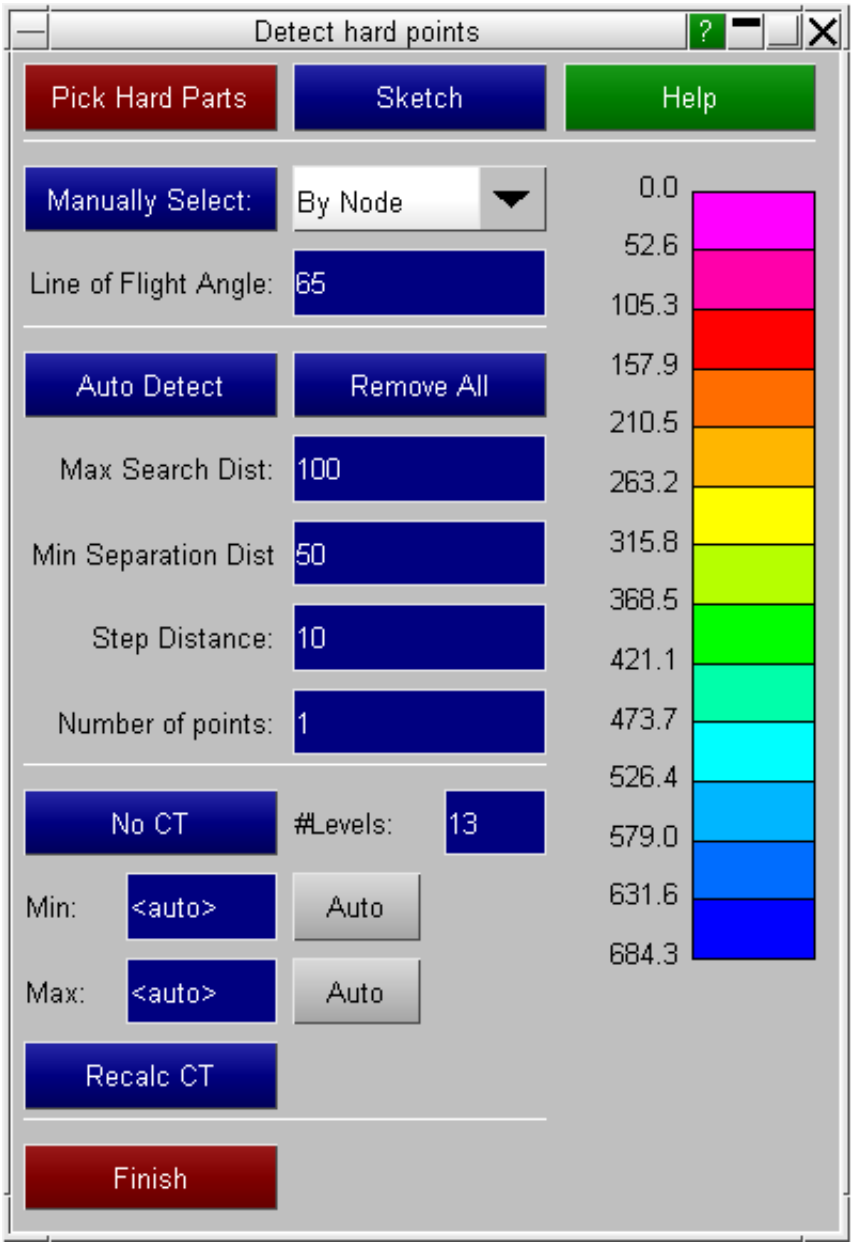
CT plot distance from vehicle to hard parts

Press the **CT** button to do a CT plot of the distance from the vehicle outer surface to the hard parts.



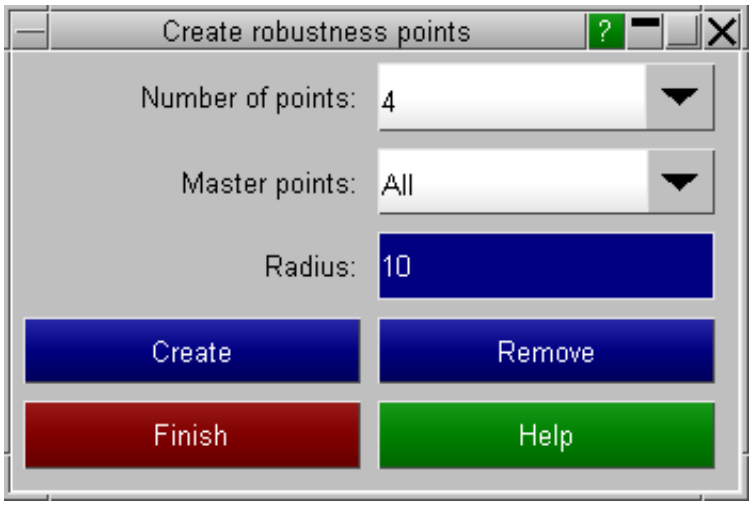
You can turn off the CT plot by pressing the **NO CT** button.

The contour bar is displayed in the hard points menu. By default it is set to 13 levels and the min and max values are automatically set. These can be changed using the **#Levels** and **Min** and **Max** textboxes.

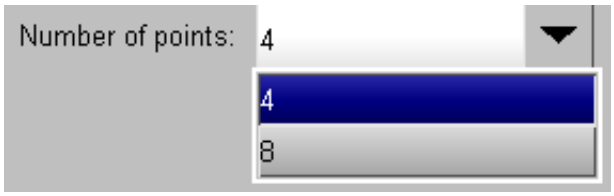


Create robustness points

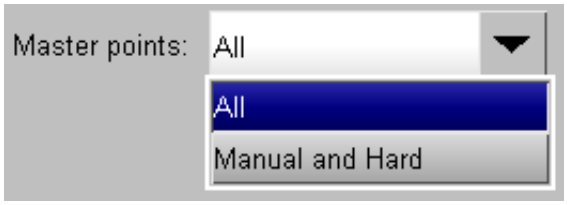
Pressing the **ROBUSTNESS POINTS** button will bring up a menu where you can create a ring of impact points around other points (all points or just manual and hard points).



First select the number of robustness points you want to create around each master point (4 or 8) from the dropdown.

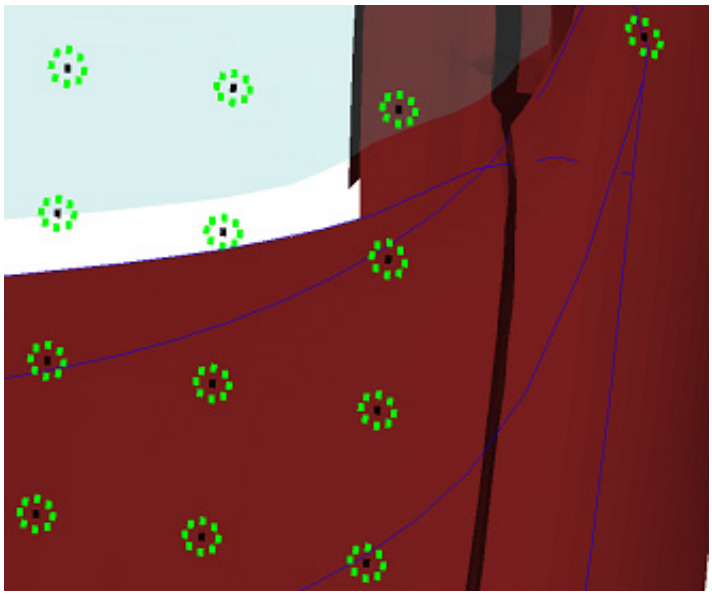


Next select which points you want to create the robustness points around. **All** will create them around all automatic, manual and hard points. **Manual and Hard** will only create them around manual and hard points.



Then set the radius of the ring of robustness points around the master points.

Press **CREATE** to create the robustness points.

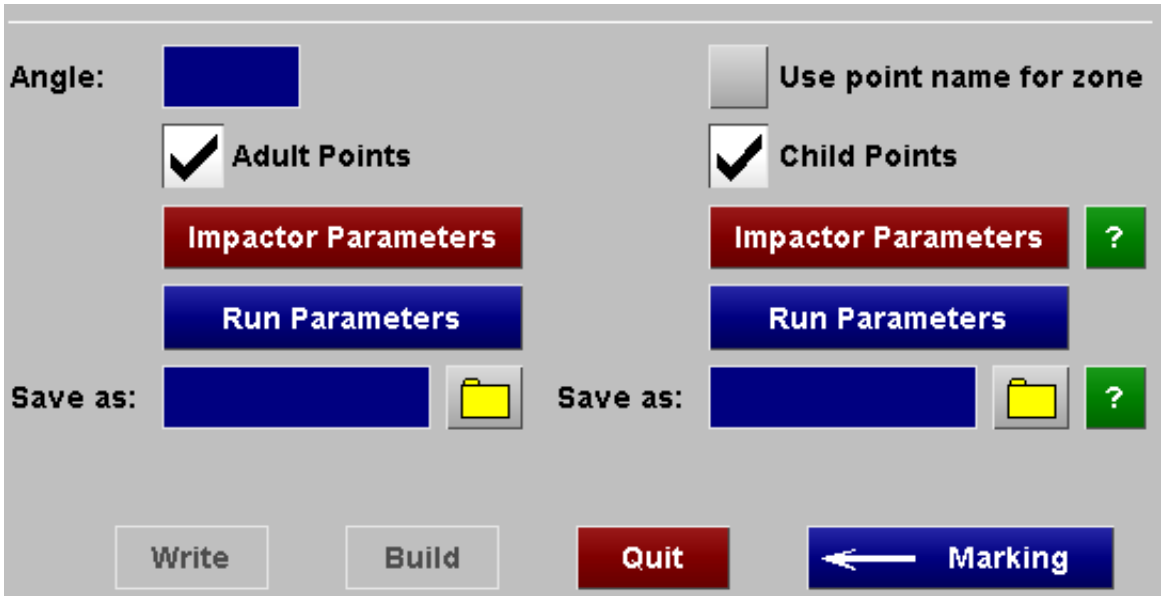


To delete any robustness points, press **REMOVE**.

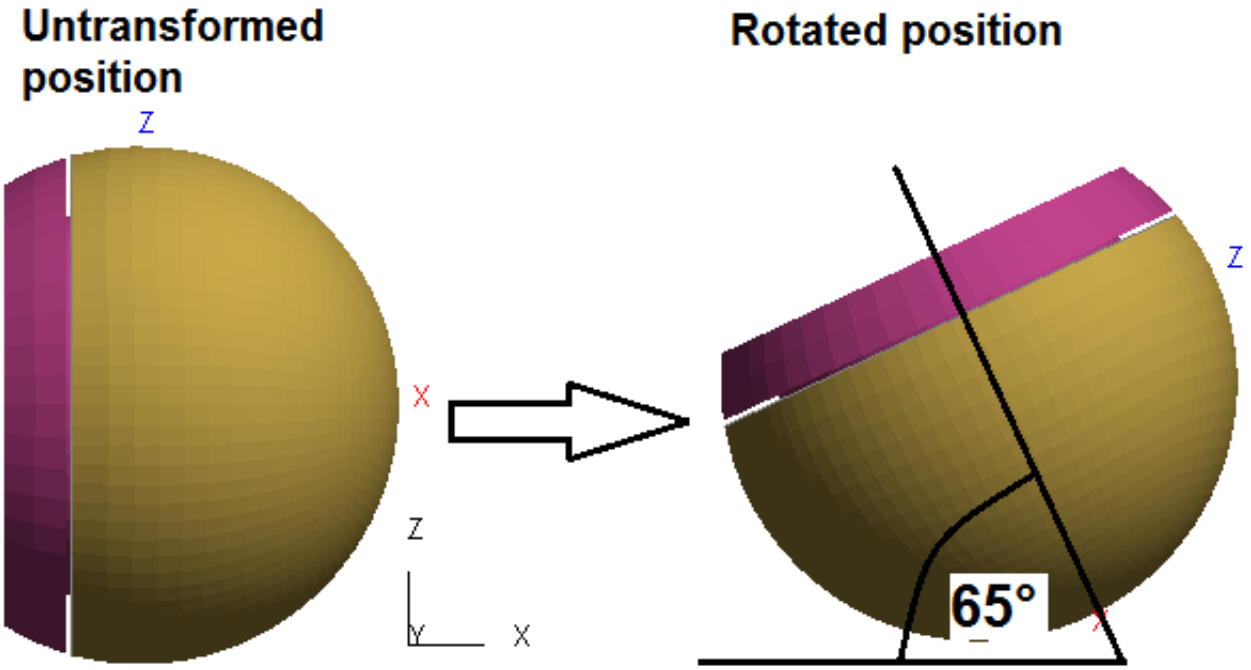
Build Models

To build models, the pedestrian markup tool uses PRIMERs model build from a CSV file functionality. You can either write out a CSV file and then use the model build menu or you can build the models directly in the pedestrian markup tool.

For head impacts you can chose to build only the adult head impacts, only the child head impacts or both.



First of all you need to select the angle to rotate the head impactor. If it is already in the correct orienation you can leave this blank, e.g.



Next select which models you want to build.

Now select where you want to save the CSV file(s) that will be used by PRIMERs model build function (one each for the adult and child head impacts).

The **WRITE** button should now become active, meaning you can write out the CSV file. However, this will only write out data for each impact point (name, coordinates and angle if set). To be able to build the models more information is required (see Appendix XIV), e.g.

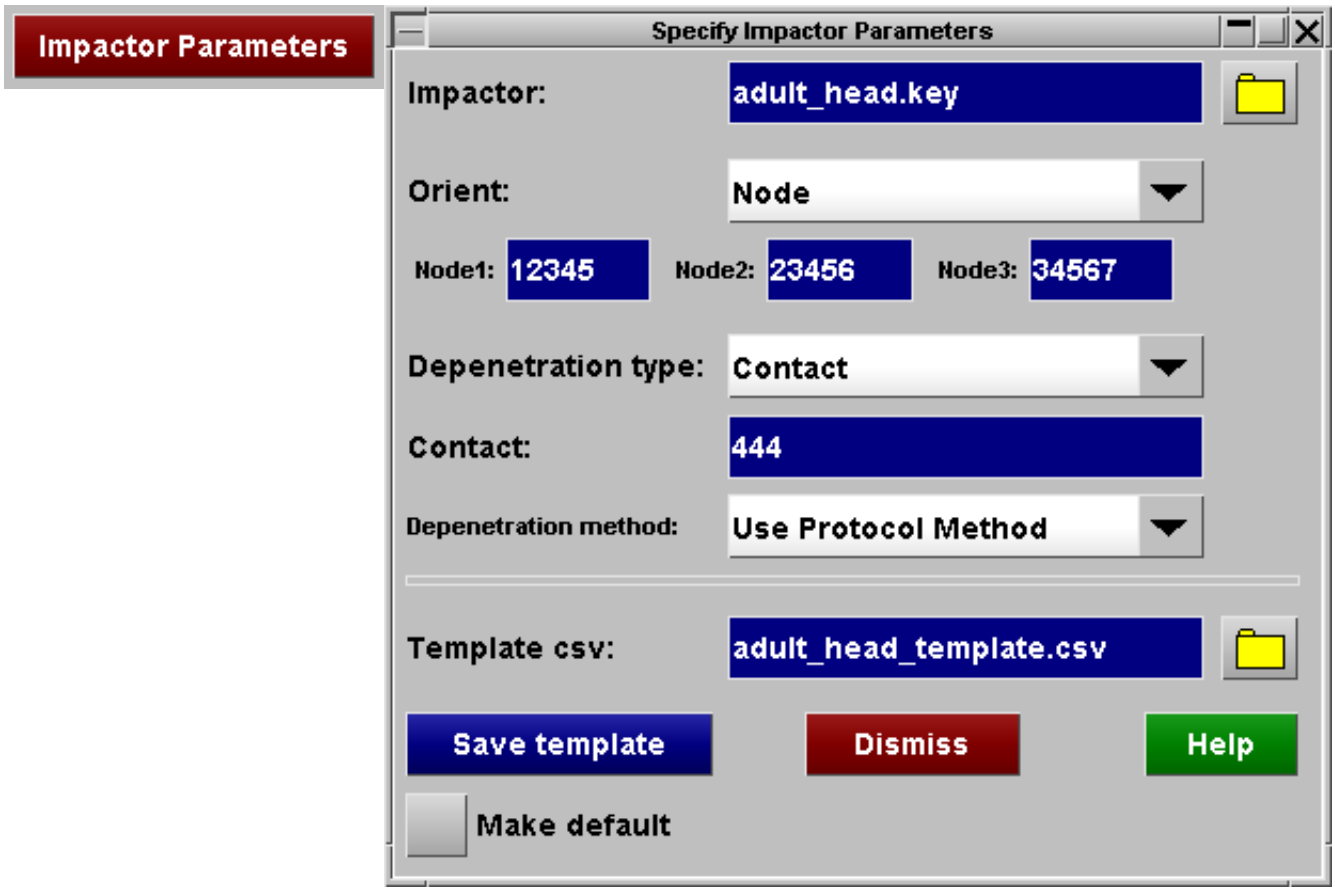
- The location of the vehicle keyword file
- The location of the impactor keyword file
- Information on the impactor orientation
- How to depenetrate the impactor from the vehicle

This information needs to be written to the top of the CSV file and can be specified here via templates and additional options. Adult and child head templates can be automatically selected using the following Primer preferences:

```
primer*pm_adult_head_template: <adult head template file with full path>

primer*pm_child_head_template: <child head template file with full path>
```

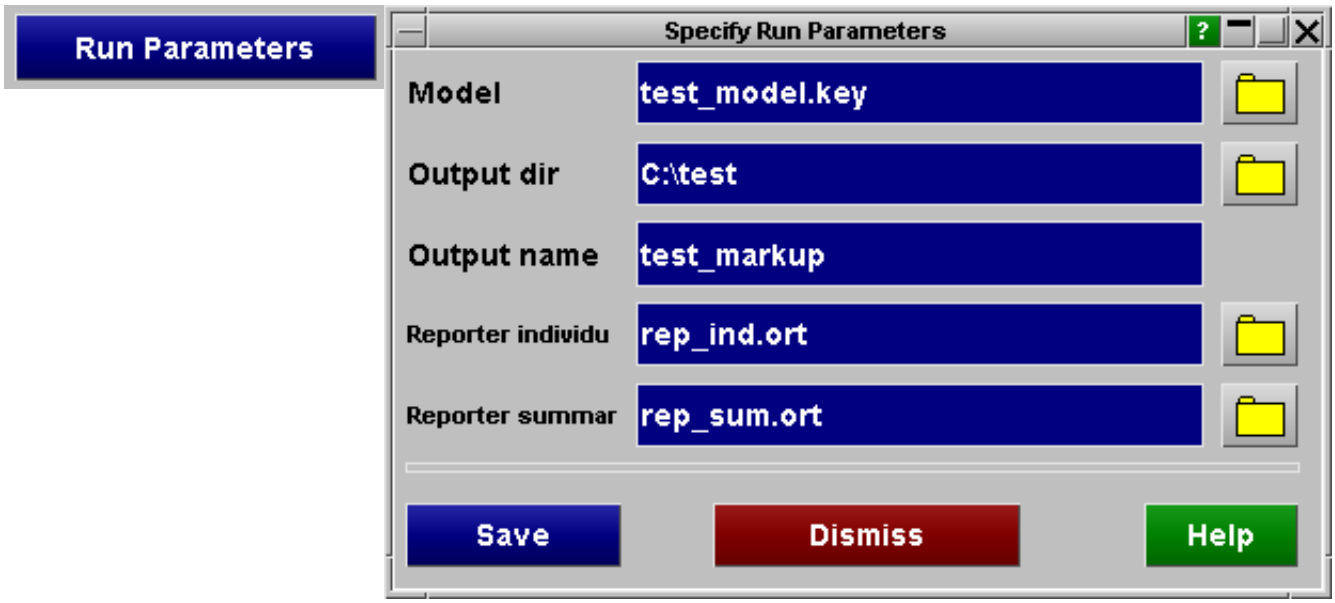
The **Impactor Parameters** menu may be used to create a template file if one doesn't exist. The template file should only need to be created once for each vehicle-impactor combination.



If you don't select a template file comments are written at the top of the CSV file showing what needs to be in the template file:

```
$ ----------------------------------------------------------------------
$ THE FOLLOWING NEEDS TO BE HAND EDITED
$
$ Remove the '$$'s from the following lines and put in the correct values
$
$ MODELS
$ ------
$$ model, <model_filename>
$$ impactor, <impactor_filename>
$
$ IMPACTOR ORIENTATION
$ --------------------
$$ orient, define, <coord system name/id>
$$ or
$$ orient, nodes, <base node name/id>, <x node name/id>, <y node name/id>
$
$ DEPENETRATION METHOD
$ --------------------
$
$$ depenetrate, contact, <contact name/id>, <dof (X, XZ or XYZ)>
$$ or
$$ depenetrate, partset, <partset name/id>, <dof (X, XZ or XYZ)>
$
$ END OF SECTION THAT NEEDS TO BE HAND EDITED
$ ---------------------------------------------
```

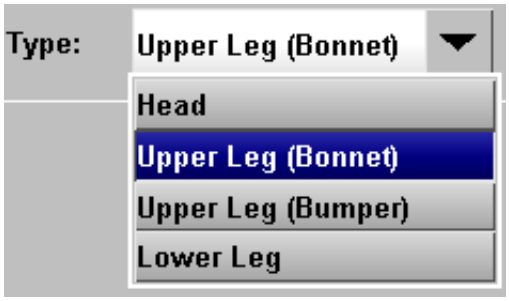
Additional optional information that can be used by the model build (from csv) function may be specified using the **Run Parameters** menu.



If you do select a template file the **BUILD** button should become active. If you press this the CSV file should get written out with the information from the template file copied into the top. Assuming the information is correct, the models should get built.

6.35.3.2 Create upper leg impact models

First set the **Type** to Upper Leg Impacts. You can chose to create impacts on either the Bonnet or the Bumper.



Automatically create points

You then need to set how you want to automatically create impact points. There are a number of ways to do this:

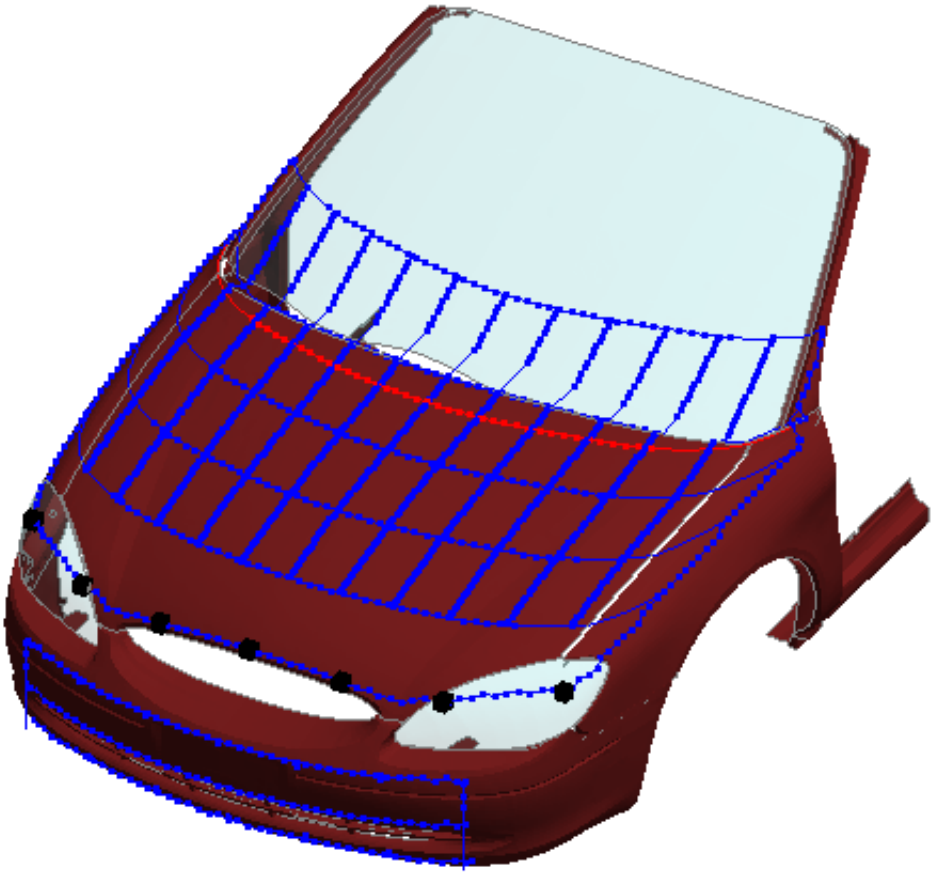
- Define a distance between points
- Define N points per zone (EuroNCAP v5 only)
- Nothing

If you set the **Method** to Define Distance, you will need to select the distance between points.

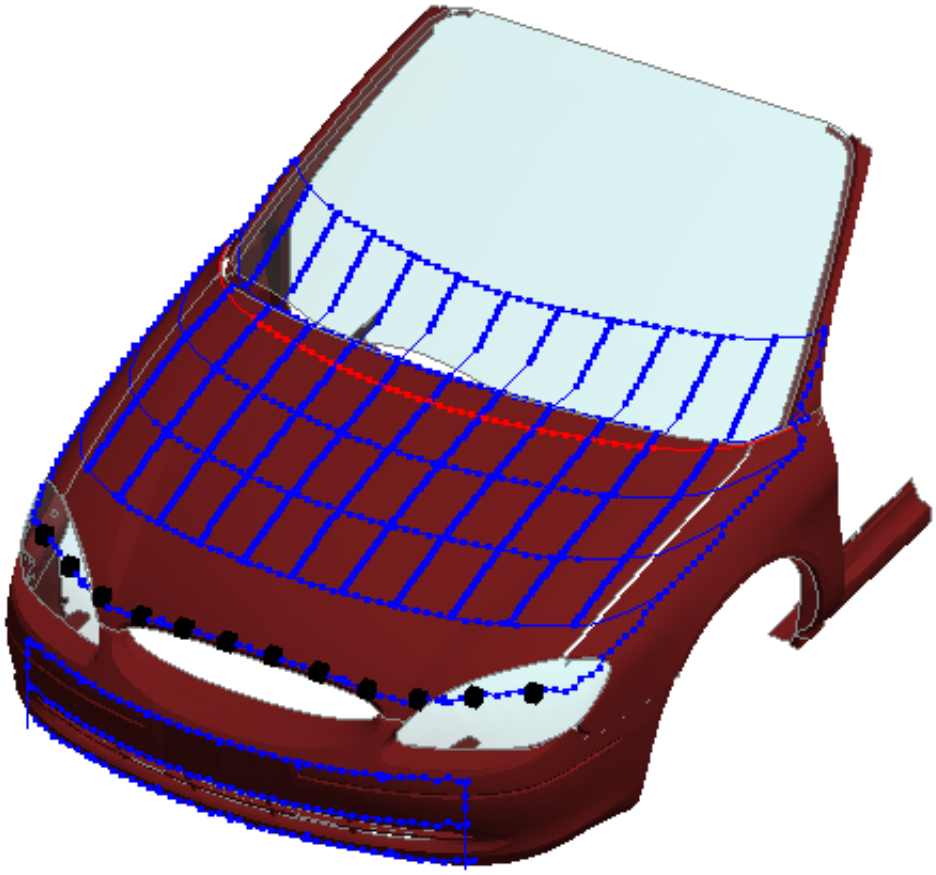
If you set the **Method** to N points, you will need to select the number of points and a spacing factor.

The spacing factor can be in the range 0-N, where 0 will put the impact points on the edge of the zone and N will put them all at the centre. Set it to 1 to space the points equally.

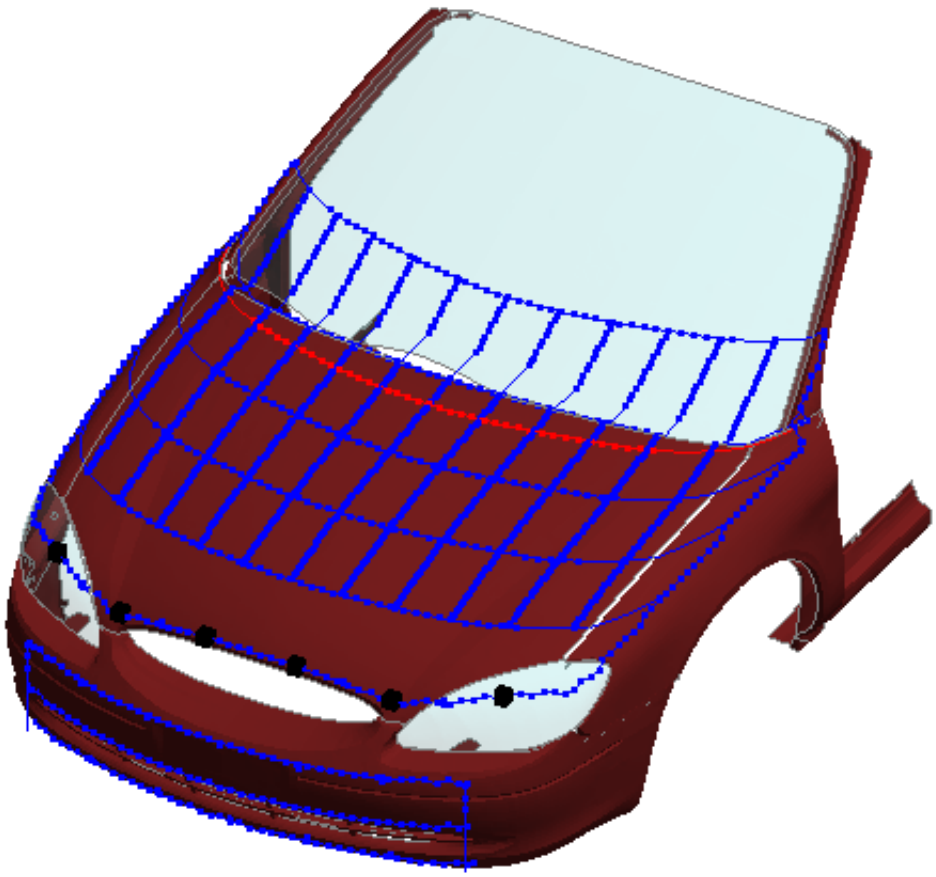
N=2, Spacing factor = 0



N=2, Spacing factor = 1



N=2, Spacing factor = 2

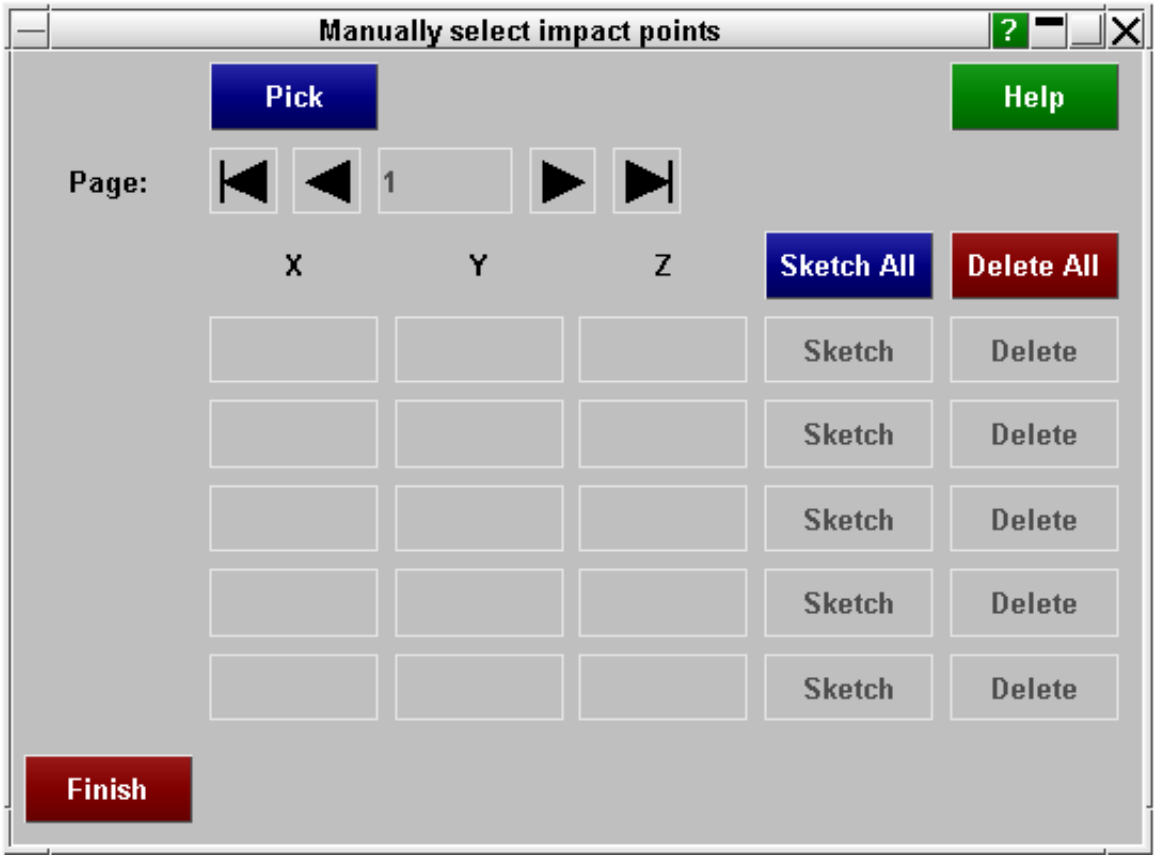


You can set the **Method** to Nothing, in which case no points will be created automatically.

To create models you will need to create points manually.

Manually create points

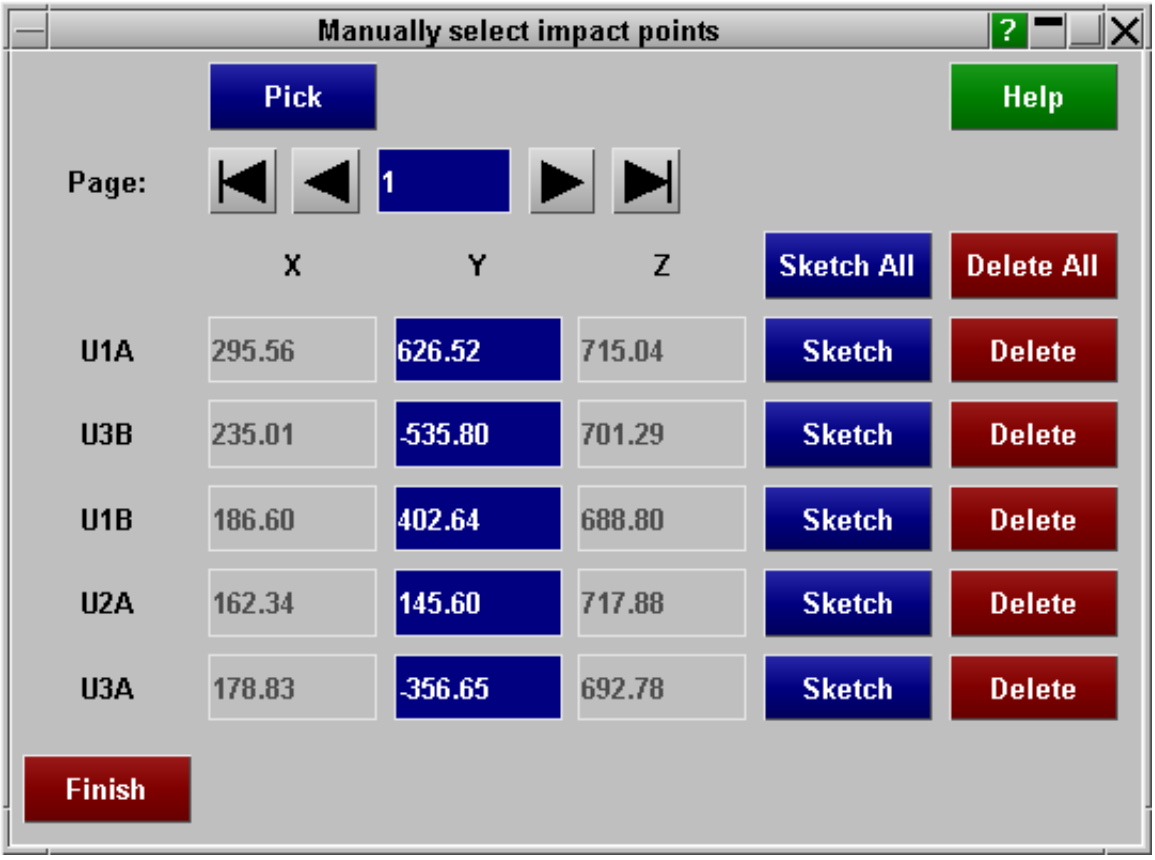
Pressing the **CREATE MANUALLY** button will bring up a menu where you can manually create impact points.



Press the **PICK** button to start picking points on the vehicle. If you select outside the markup line boundaries then a point will not be created.

Once you have selected all the points you want, press the **FINISH** button in the window that has popped up.

The menu will be filled with the labels and coordinates of the points created.

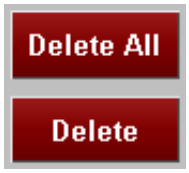


If you have created enough points you can move to different pages in the menu to view them.

You can also edit the coordinates by typing in the appropriate textboxes. Only the Y cordinate can be modified as the X and Z coordinates are constrained to the Bonnet Leading Edge line (for bonnet impacts) or Bumper Line (for bumper impacts) and will be calculated automatically.

To view the points press **SKETCH ALL** or **SKETCH**

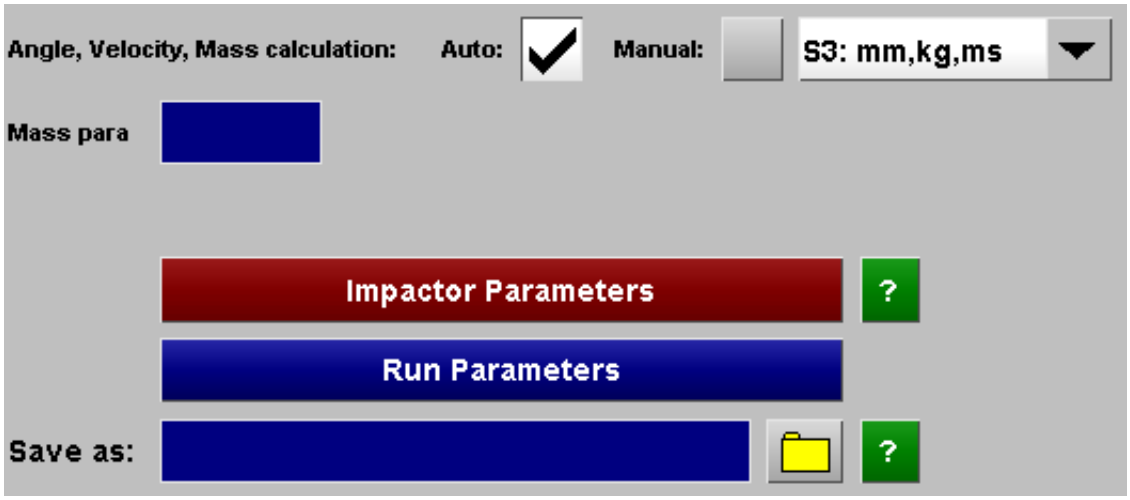
To delete points press **DELETE ALL** or **DELETE**



Once you have finished press the **FINISH** button to close the menu.

Build Models

To build models, the pedestrian markup tool uses PRIMERs model build from a CSV file functionality. You can either write out a CSV file and then use the model build menu or you can build the models directly in the pedestrian markup tool.



For upper leg impacts the angle, velocity and mass the impactor should be set to is dependent on the location of the impact point on the vehicle and will most likely be different for each one. PRIMER can use the rules specified in the protocol to automatically calculate the correct values and use them to build the models. Alternatively you can specify these values manually, however they will be applied to **all** the impact points.

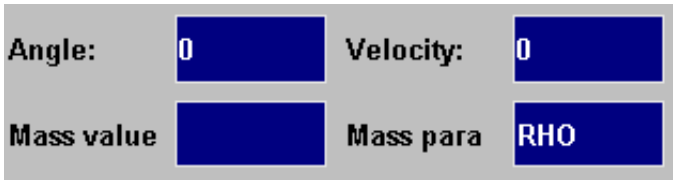
If you select the Auto option, you will also need to select the unit system of your model so that PRIMER knows how it should output the velocity and mass values.

The impactor model should contain a reference to a parameter that when modified will change it's mass, e.g. the density field of a material card:

*MAT_ELASTIC
3,&RHO,210.0,0.3,0.0,0.0,0.0

The name of this parameter should be specified here.

For the manual method you need to set the angle. velocity, mass parameter value and mass parameter name. These will be applied to all the impact points.



Now select where you want to save the CSV file that will be used by PRIMERs model build function.

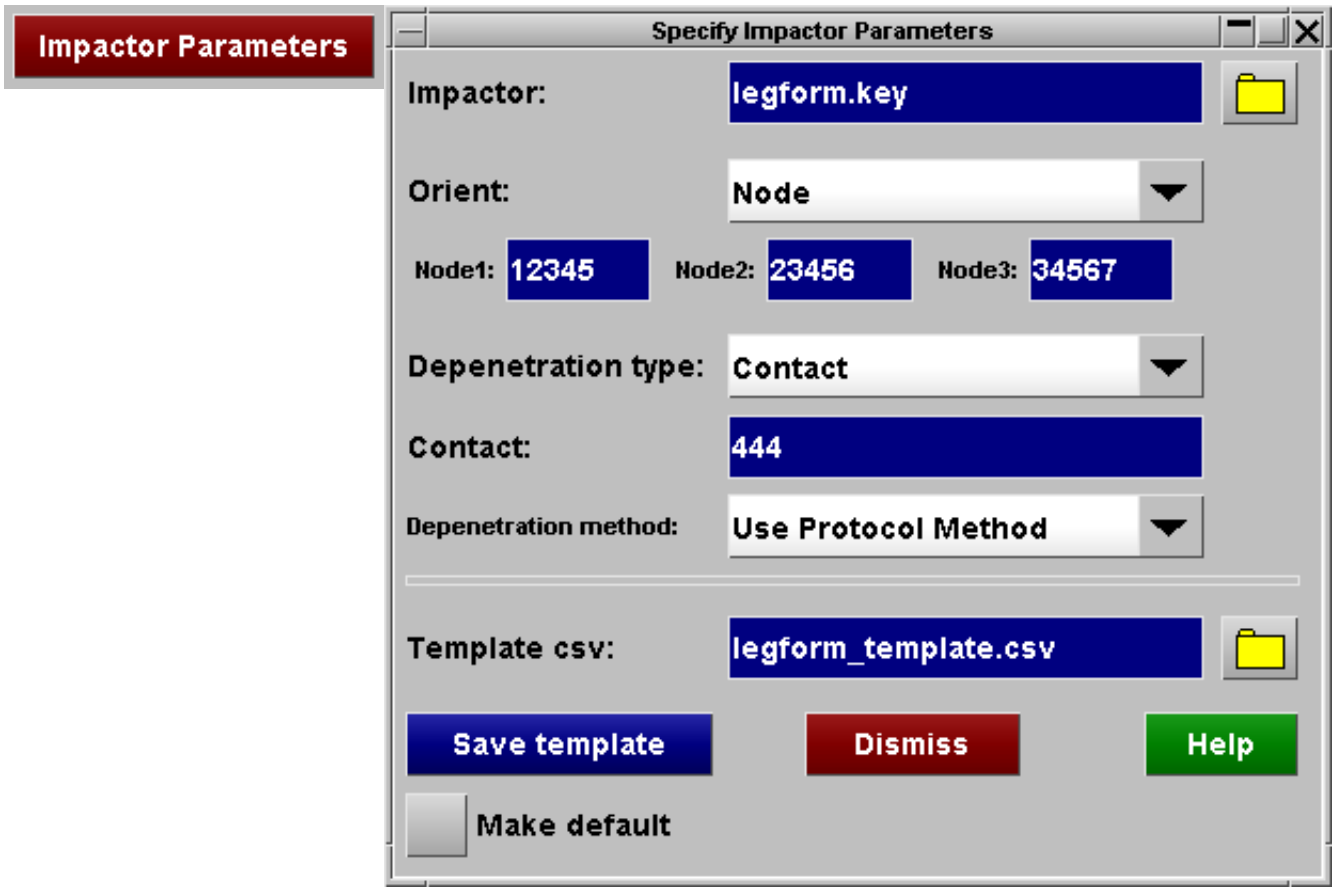
The **WRITE** button should now become active, meaning you can write out the CSV file. However, this will only write out data for each impact point (name, coordinates, angle, velocity and mass parameter value). To be able to build the models more information is required (see Appendix XIV), e.g.

- The location of the vehicle keyword file
- The location of the impactor keyword file
- Information on the impactor orientation
- How to depenetrate the impactor from the vehicle

This information needs to be written to the top of the CSV file and can be specified here via templates and additional options. The upper legform template can be automatically selected using the following Primer preference:

```
primer*pm_upper_leg_template: <upper legform template file with full path>
```

The **Impactor Parameters** menu may be used to create a template file if one doesn't exist. The template file should only need to be created once for each vehicle-impactor combination.

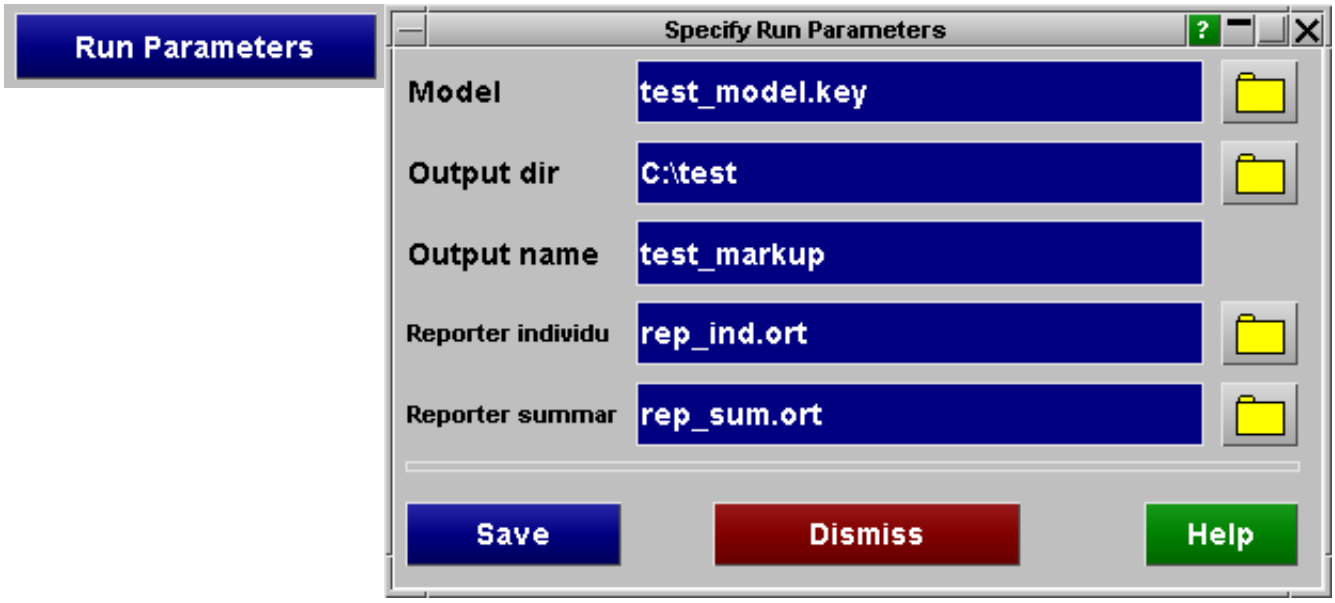


If you don't select a template file comments are written at the top of the CSV file showing what needs to be in the template file:

```
$ ----------------------------------------------------------------------
$ THE FOLLOWING NEEDS TO BE HAND EDITED
$
$ Remove the '$$'s from the following lines and put in the correct values
$
$ MODELS
$ ------
$$ model, <model_filename>
$$ impactor, <impactor_filename>
$
$ IMPACTOR ORIENTATION
$ --------------------
$$ orient, define, <coord system name/id>
$$ or
$$ orient, nodes, <base node name/id>, <x node name/id>, <y node name/id>
```

```
$
$ DEPENETRATION METHOD
$ --------------------
$
$$ depenetrate, contact, <contact name/id>, <dof (X, XZ or XYZ)>
$$ or
$$ depenetrate, partset, <partset name/id>, <dof (X, XZ or XYZ)>
$
$ END OF SECTION THAT NEEDS TO BE HAND EDITED
$ ---------------------------------------------
```

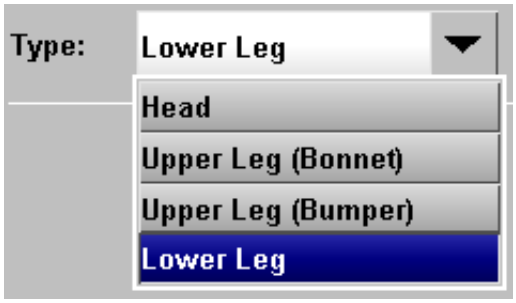
Additional optional information that can be used by the model build (from csv) function may be specified using the **Run Parameters** menu.



If you do select a template file the **BUILD** button should become active. If you press this the CSV file should get written out with the information from the template file copied into the top. Assuming the information is correct, the models should get built.

6.35.3.3 Create lower leg impact models

First set the **Type** to Lower Leg Impacts.



Automatically create points

You then need to set how you want to automatically create impact points. There are a number of ways to do this:

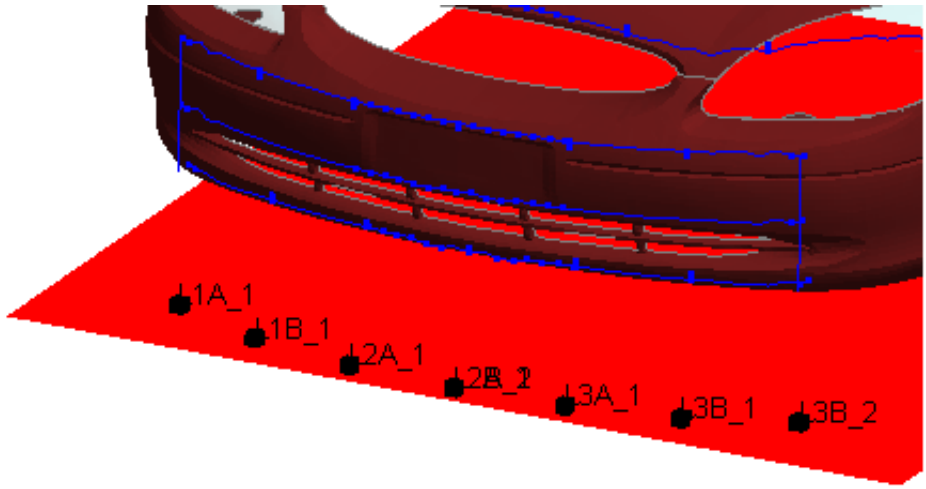
- Define a distance between points
- Define N points per zone (EuroNCAP v5 only)
- Nothing

If you set the **Method** to Define Distance, you will need to select the distance between points.

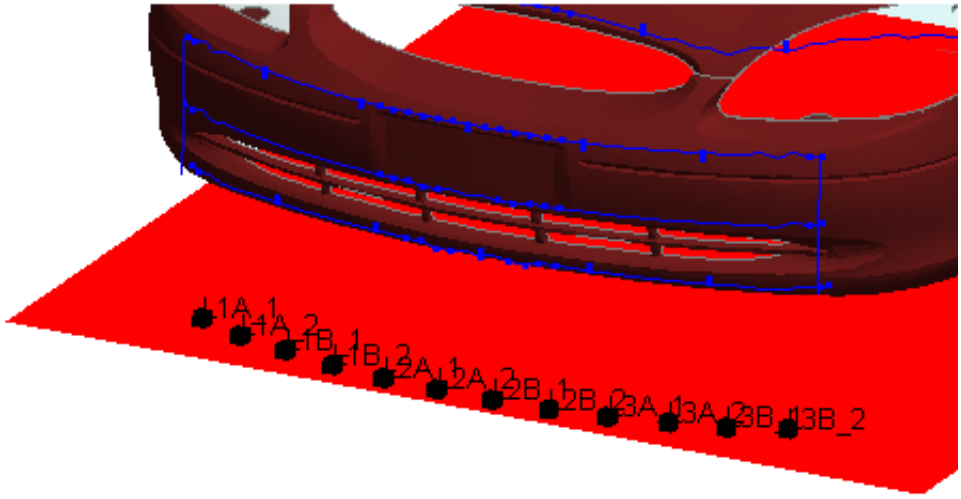
If you set the **Method** to N points, you will need to select the number of points and a spacing factor.

The spacing factor can be in the range 0-N, where 0 will put the impact points on the edge of the zone and N will put them all at the centre. Set it to 1 to space the points equally.

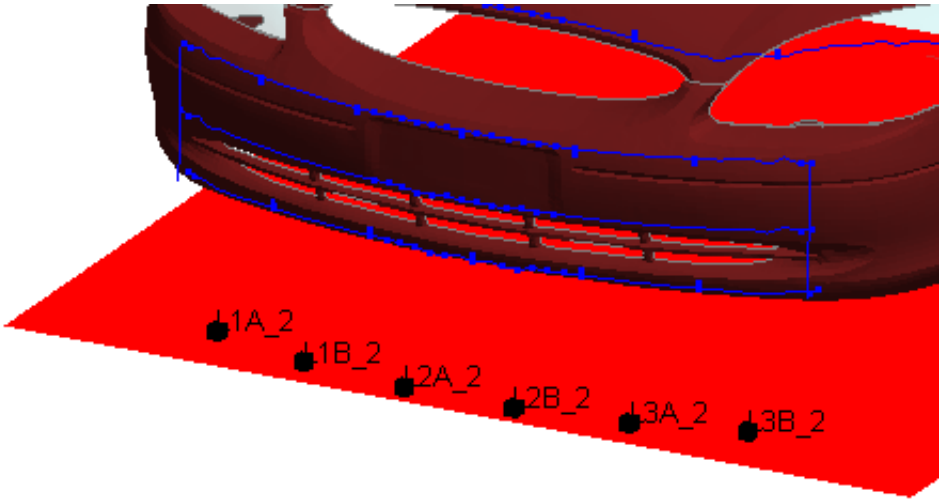
N=2, Spacing factor = 0



N=2, Spacing factor = 1



N=2, Spacing factor = 2

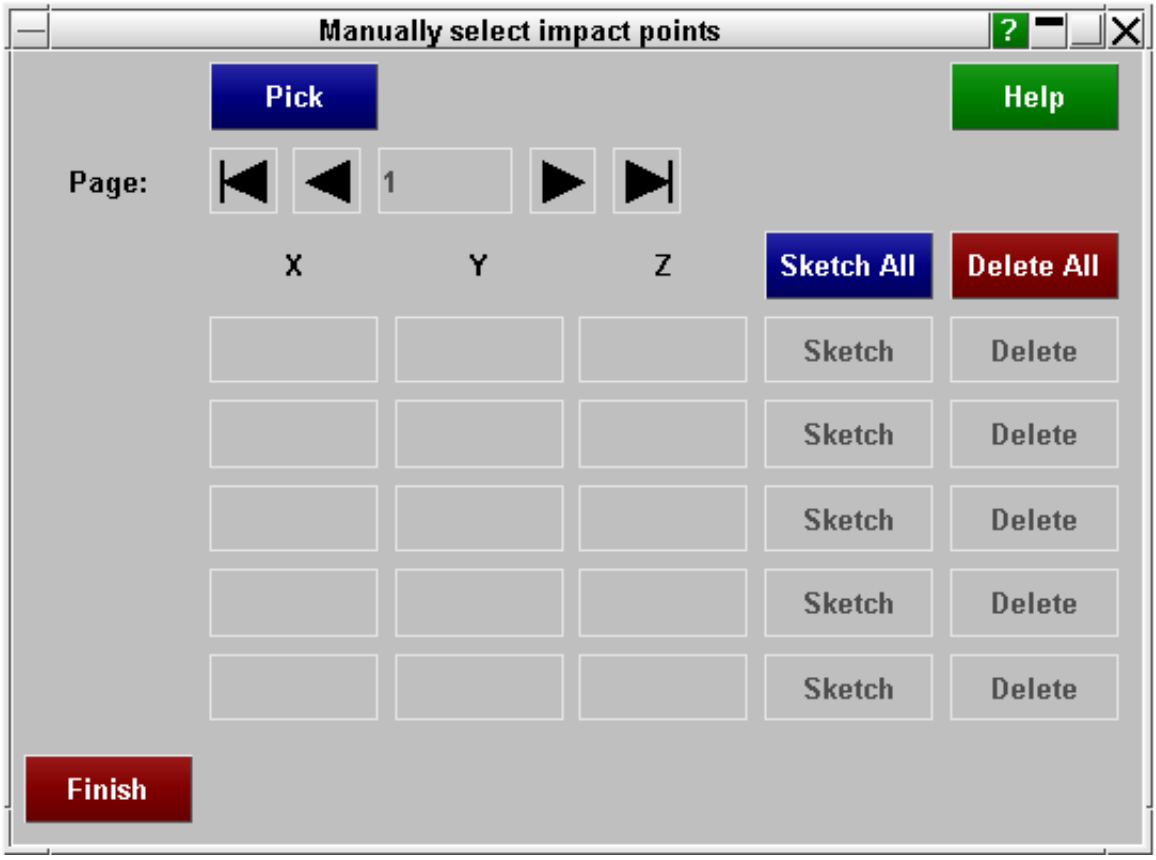


You can set the **Method** to Nothing, in which case no points will be created automatically.

To create models you will need to create points manually.

Manually create points

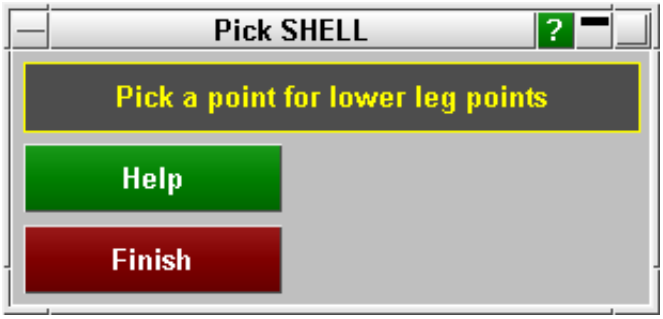
Pressing the **CREATE MANUALLY** button will bring up a menu where you can manually create impact points.

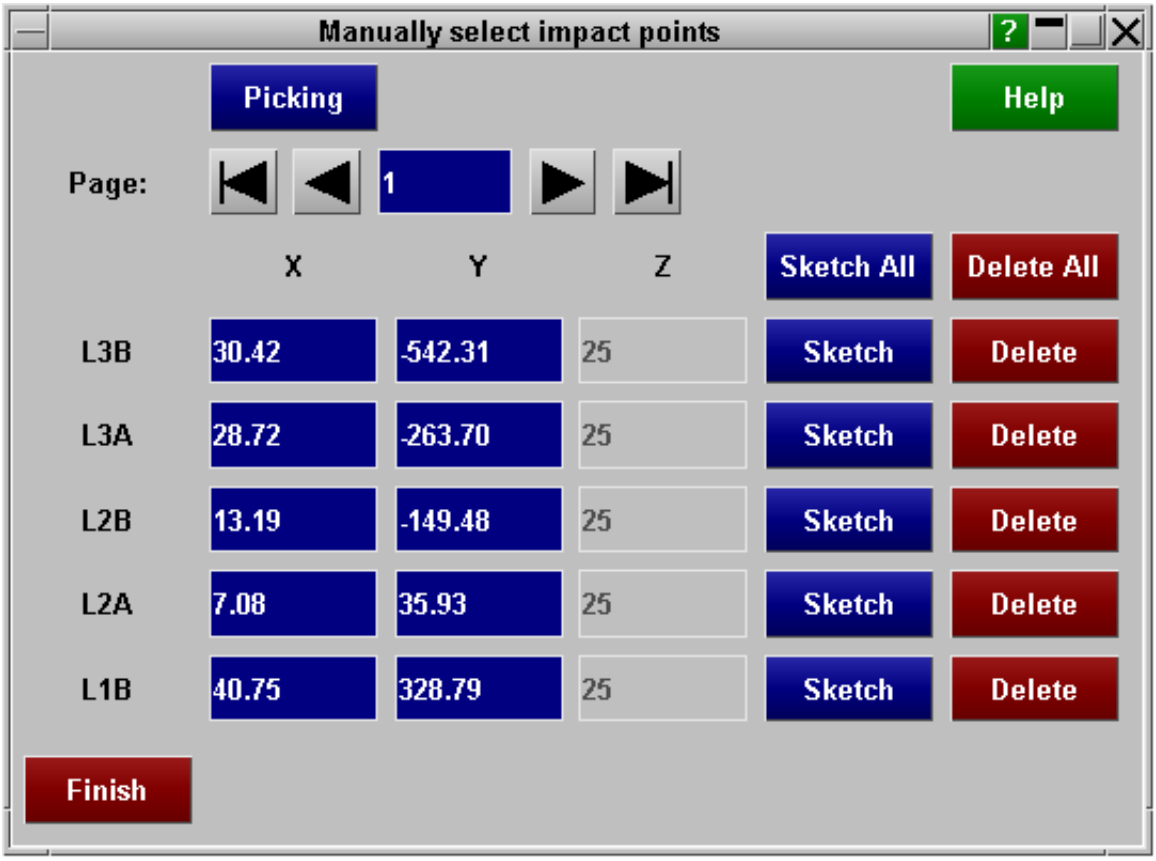


Press the **PICK** button to start picking points on the vehicle. If you select outside the markup line boundaries then a point will not be created.

Once you have selected all the points you want, press the **FINISH** button in the window that has popped up.

The menu will be filled with the labels and coordinates of the points created.



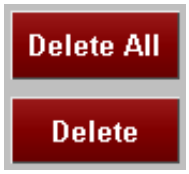


If you have created enough points you can move to different pages in the menu to view them.

You can also edit the coordinates by typing in the appropriate textboxes. Only the X and Y cordinate can be modified as the Z coordinate is specified below.

To view the points press **SKETCH ALL** or **SKETCH**

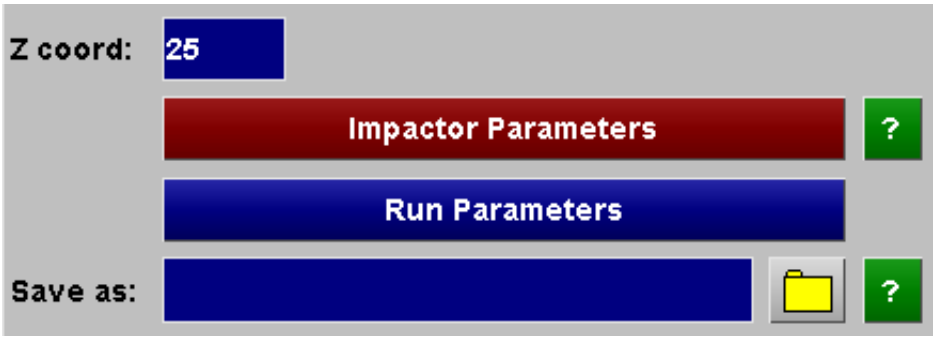
To delete points press **DELETE ALL** or **DELETE**



Once you have finished press the **FINISH** button to close the menu.

Build Models

To build models, the pedestrian markup tool uses PRIMERs model build from a CSV file functionality. You can either write out a CSV file and then use the model build menu or you can build the models directly in the pedestrian markup tool.



For lower leg impacts you need to specify the Z coordinate of the bottom of the impactor. By default this is set to 25mm above the ground level.

Now select where you want to save the CSV file that will be used by PRIMERs model build function.

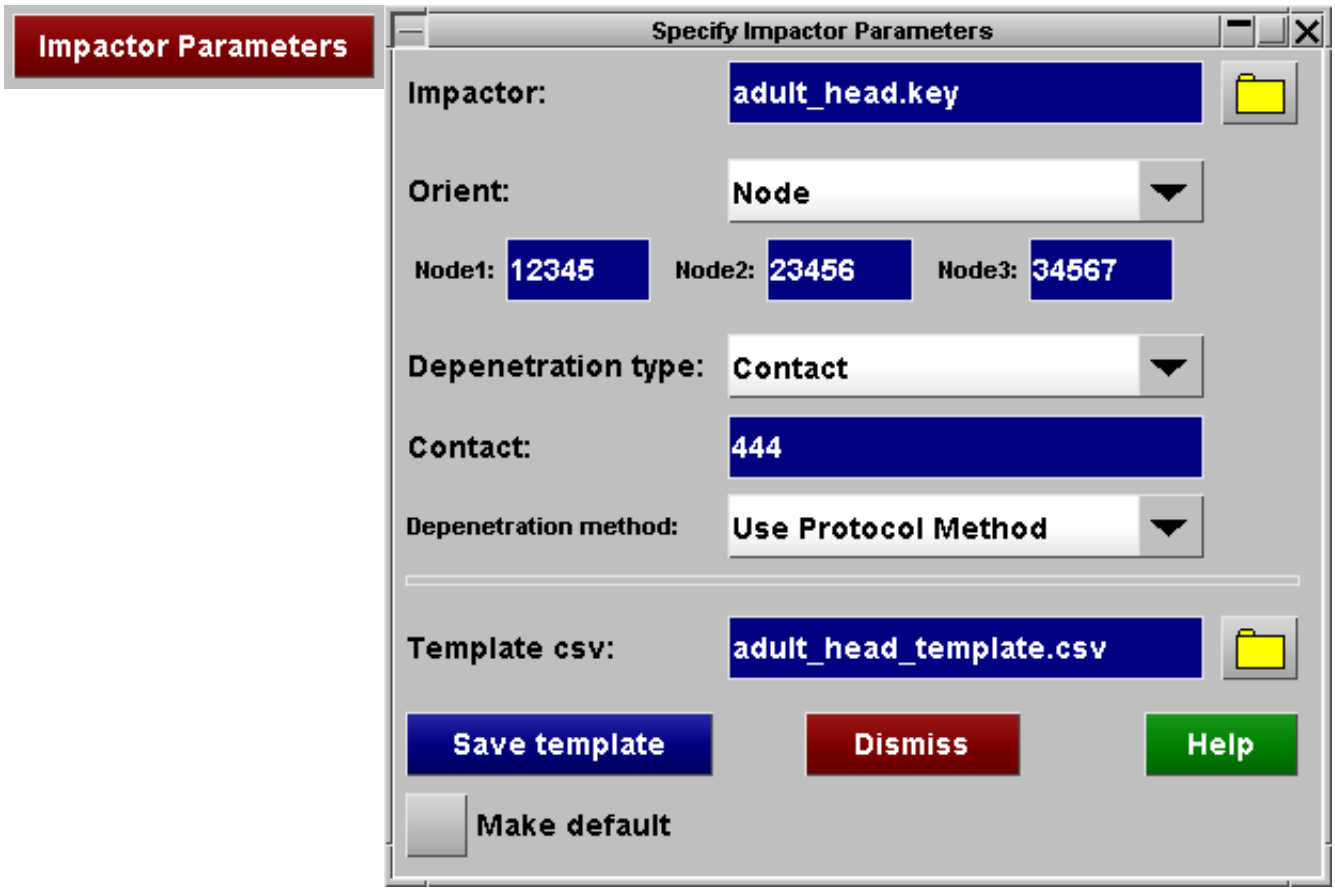
The **WRITE** button should now become active, meaning you can write out the CSV file. However, this will only write out data for each impact point (name and coordinates). To be able to build the models more information is required (see Appendix XIV), e.g.

- The location of the vehicle keyword file
- The location of the impactor keyword file
- Information on the impactor orientation
- How to depenetrate the impactor from the vehicle

This information needs to be written to the top of the CSV file and can be specified here via a template and additional options. The lower legform template can be automatically selected using the following Primer preference:

```
primer*pm_lower_leg_template: <lower legform template file with full path>
```

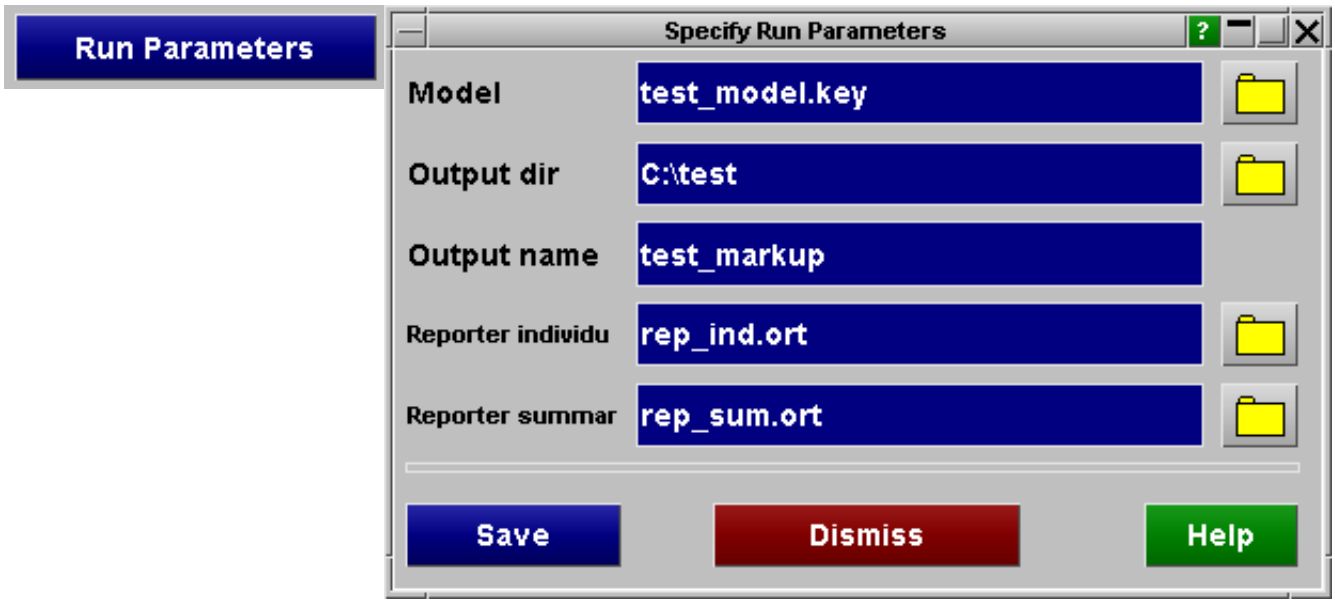
The **Impactor Parameters** menu may be used to create a template file if one doesn't exist. The template file should only need to be created once for each vehicle-impactor combination.



If you don't select a template file comments are written at the top of the CSV file showing what needs to be in the template file:

```
$ ----------------------------------------------------------------------
$ THE FOLLOWING NEEDS TO BE HAND EDITED
$
$ Remove the '$$'s from the following lines and put in the correct values
$
$ MODELS
$ ------
$$ model, <model_filename>
$$ impactor, <impactor_filename>
$
$ IMPACTOR ORIENTATION
$ --------------------
$$ orient, define, <coord system name/id>
$$ or
$$ orient, nodes, <base node name/id>, <x node name/id>, <y node name/id>
$
$ DEPENETRATION METHOD
$ --------------------
$
$$ depenetrate, contact, <contact name/id>, <dof (X, XZ or XYZ)>
$$ or
$$ depenetrate, partset, <partset name/id>, <dof (X, XZ or XYZ)>
$
$ END OF SECTION THAT NEEDS TO BE HAND EDITED
$ ---------------------------------------------
```

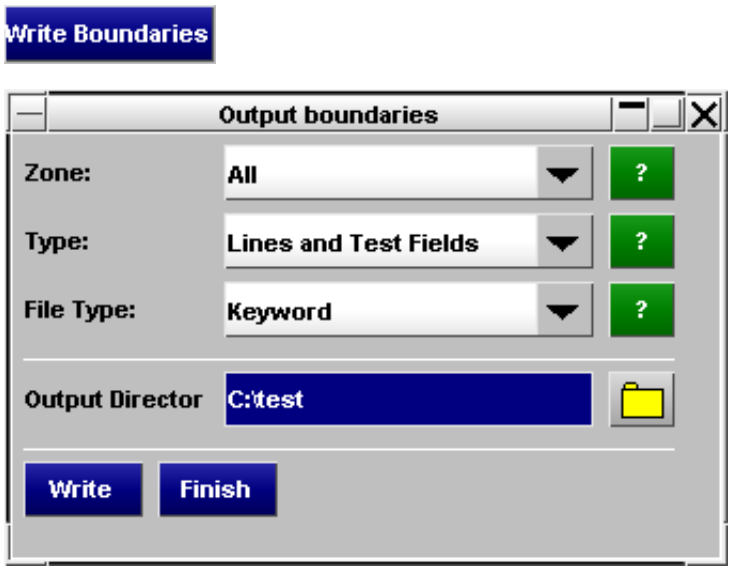
Additional optional information that can be used by the model build (from csv) function may be specified using the **Run Parameters** menu.



If you do select a template file the **BUILD** button should become active. If you press this the CSV file should get written out with the information from the template file copied into the top. Assuming the information is correct, the models should get built.

6.35.4 Output Lines to File

The lines and test fields can be written out as keyword or IGES files so they can be viewed in other programs. Press the **WRITE BOUNDARIES** button to open a window to select what to output.



Select the Zone [All, Head, Upper Leg or Lower Leg], Type [Lines and/or Test Fields] and File Type [Keyword or IGES], then select the directory to output the files and press **WRITE**. The files are given appropriate names to indicate what they are, e.g. 'adult_head_boundary.key' and 'WAD1000.key'.

6.35.5 Command Line

The pedestrian markup tool can also be run from the command line:

primer.exe -js=*pedestrian_impact_marking_program.js* -js_arg=*arguments_file vehicle.key*

Where:

primer.exe	Is the full pathname of the PRIMER executable to run.

pedestrian_impact_marking_program.js	Is the full pathname of the pedestrian markup script. This can be found in the `primer_library/scripts` directory where the PRIMER executable has been installed.
arguments_file	Is the full pathname to a file containing arguments for the script. The format is described below.
vehicle.key	Is the full pathname to the vehicle model to markup.

The *arguments_file* is a comma separated file of *variable name,variable value* pairs that pass the information required for the script to markup the vehicle and build models.

Below is a list of all the variables that can be passed to the script. Some of them have to be set for the script to do anything, but most of them do not in which case default values will are used.

Name	**Description**
adult_head_csv_filename	CSV filename for ADULT Head impacts
adult_head_impactor	Filename of ADULT Head impactor to use in the model build. This will overwrite the impactor defined in the template file.
adult_head_template_filename	Template filename for ADULT Head impacts
ble_angle	Bonnet Leading Edge stick angle
ble_beam_length	Bonnet Leading Edge beam length
ble_offset_distance	Bonnet Leading Edge offset distance
ble_offset_line	Switch for Bonnet Leading Edge offset line (set to true or false)
ble_line	Switch for Bonnet Leading Edge line (set to true or false)
bonnet_parts	List of bonnet parts separated by commas, e.g. bonnet_parts,100,200,300
bumper_beam_length	Bumper lines beam length
bumper_beam_parts	List of bumper beam parts separated by commas, e.g. bumper_beam_parts,100,200,300
bumper_lines	Switch for Bumper lines (set to true or false)
bumper_corner_point	Switch for Bumper Corner Point (set to true or false)
bumper_lower_line	Switch for Bumper Lower Line (set to true or false)
bumper_upper_line	Switch for Bumper Upper Line (set to true or false)
bumper_parts	List of bumper parts separated by commas, e.g. bumper_parts,100,200,300
child_head_csv_filename	CSV filename for CHILD Head impacts
child_head_impactor	Filename of CHILD Head impactor to use in the model build. This will overwrite the impactor defined in the template file.
child_head_template_filename	Template filename for CHILD Head impacts
corner_bumper_angle	Bumper Corner stick angle
default_green_parts	List of default green parts separated by commas, e.g. default_green_parts,100,200,300
distance_between_points	Distance between points
exit	Exit from the script if set to true
error_filename	Filename for writing any errors
gm_transform_filename	Transform filename if using the "gm" head label style
gm_transform_title	Transform title if using the "gm" head label style
ground_z	Ground z coordinate
head_angle	Angle for head impacts
head_label_style	Label style for head impacts (gm or standard)

impact_auto_method	Automatic method for calculating impact points (N_Points, NxM_Points, Define_Distance or Nothing)
impact_m_points	Number of ’m’ points for automatic method NxM_Points
impact_m_spacing	Spacing between ’m’ points for automatic method NxM_Points
impact_n_points	Number of ’n’ points for automatic methods N_Points and NxM_Points
impact_n_spacing	Spacing between ’n’ points for automatic methods N_Points and NxM_Points
impact_type	Impact type (Head, Upper_Leg_Bonnet, Upper_Leg_Bumper or Lower_Leg)
lower_bumper_angle	Lower Bumper stick angle
lower_leg_impactor	Filename of Lower Leg impactor to use in the model build. This will overwrite the impactor defined in the template file.
lower_leg_z	Lower Leg Z coordinate
lower_leg_csv_filename	CSV filename for Lower Leg impacts
lower_leg_template_filename	Template filename for Lower Leg impacts
master_model	Filename of Vehicle model to use in the model build. This will overwrite the model defined in the template file.
outer_parts	List of outer vehicle parts separated by commas, e.g. outer_parts,100,200,300
protocol	Protocol type (EuroNCAP, EuroNCAP_Grid, EuroNCAP_7_0 or GTR)
rootdir	Directory where models should be built. This will overwrite the directory defined in the template file
rrl_beam_length	Rear Reference Line beam length
rrl_offset_distance	Rear Reference Line offset distance
rrl_offset_line	Switch for Rear Reference offset line (set to true or false)
rrl_line	Switch for Rear Reference Line (set to true or false)
srl_angle	Side Reference Line stick angle
srl_beam_length	Side Reference Line beam length
srl_offset_distance	Side Reference Line offset distance
srl_offset_line	Switch for Side Reference offset line (set to true or false)
srl_line	Switch for Side Reference Line (set to true or false)
upper_bumper_angle	Upper Bumper stick angle
upper_leg_angle	Angle for Upper Leg impacts (used if upper_leg_calculation is set to manual)
upper_leg_calculation	Angle, Velocity and Mass calculation method for Upper Leg impacts (auto or manual)
upper_leg_impactor_filename	Filename of Upper Leg impactor to use for auto calculation of angle, velocity and mass.
upper_leg_mass_value	Mass value for Upper Leg impacts (used if upper_leg_calculation is set to manual)
upper_leg_mass_parameter	Mass parameter for Upper Leg impacts (used if upper_leg_calculation is set to manual)
upper_leg_csv_filename	CSV filename for Upper Leg impacts
upper_leg_impactor	Filename of Upper Leg impactor to use in the model build. This will overwrite the impactor defined in the template file.
upper_leg_template_filename	Template filename for Upper Leg impacts
upper_leg_units	Unit system for Upper Leg auto method calculation (S2 or S3)
upper_leg_velocity	Velocity value for Upper Leg impacts (used if upper_leg_calculation is set to manual)

wad_lines	Switch for WAD Lines (set to true or false)
wad1	1st WAD value
wad2	2nd WAD value
wad3	3rd WAD value
wad4	4th WAD value
wad5	5th WAD value
wad6	6th WAD value
windscreen_parts	List of windscreen parts separated by commas, e.g. windscreen_parts,100,200,300
write_boundary_lines	Write boundary lines to file
write_boundary_zone	Write boundary zone (All, Head, Upper_Leg_Bonnet, Upper_Leg_Bumper or Lower_Leg)
write_boundary_type	Write boundary type (All, Lines or Test_Fields)
write_boundary_file_type	Write boundary file type (Keyword, IGES or CSV)
write_boundary_directory	Write boundary directory

To markup the vehicle the folllowing need to be defined:

- protocol
- outer_parts
- bonnet_parts - (only for EuroNCAP_Grid, EuroNCAP_7_0 and GTR protocols)
- windscreen_parts - (only for EuroNCAP_Grid, EuroNCAP_7_0 and GTR protocols)

e.g.

```
protocol,EuroNCAP_7_0

outer_parts,100,200,300,400,500,600

bonnet_parts,100

windscreen_parts,200
```

If you want to write out CSV files of the impact points you also need to specify at least one of the following (depending on the impact type selected):

- adult_head_csv_filename
- child_head_csv_filename
- lower_leg_csv_filename
- upper_leg_csv_filename

e.g.

```
impact_type,Lower_Leg

lower_leg_csv_filename,C:\lower_leg.csv
```

To build models you also need to specify at least one of the following (depending on the impact type selected):

- adult_head_template_filename
- child_head_template_filename
- lower_leg_template_filename
- upper_leg_template_filename

e.g.

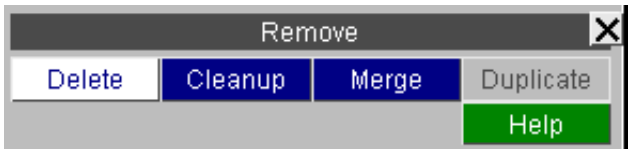
```
impact_type,Lower_Leg

lower_leg_template_filename,C:\lower_leg_template.csv
```

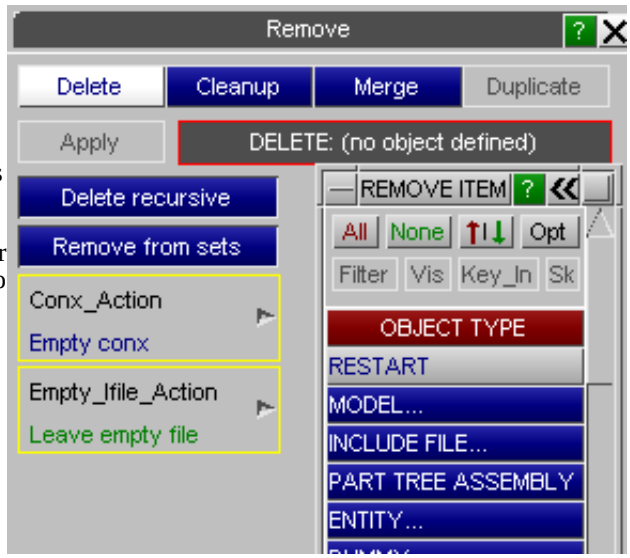
6.36 REMOVE Delete unwanted, model clean-up, node merging and duplicate elimination.

The **Remove** command is invoked from the **Tools** panel at the top of the screen.
It has three options:

Delete	Delete unwanted entities from any model;
Cleanup	Remove unused non-structural model entities;
Merge	Combine together nodes within a certain tolerance



6.36.1 DELETE UNWANTED



To delete model entities use **Delete**.

This option can be used to delete anything from entire models to individual items, using the standard PRIMER hierarchy.

(Note, however, that deleting a complete model is much faster if the main **MODEL> DELETE** command is used because no cross-checking for dependencies is required.)

To delete entities:

- Select items to be deleted.
- On pressing **Apply** PRIMER searches through the model hierarchy (subject to the recursion settings, see rule 2 below).
- You are presented with a list of everything marked for deletion, and given the opportunity to modify it.
- When you confirm the deletion list those items are removed.

6.36.1.1 Deletion Rules

There are some strict rules governing how deletion is applied which are intended to prevent you accidentally leaving something "hanging" by deleting an item upon which it depends. These rules are:

- No entity may be deleted if it is "locked" because it is referenced by another entity which is not itself being deleted. For example a part definition cannot be deleted if it is referenced by an element which is to remain. The entity hierarchy is described in section 2.12.

  However if "Forced deletion", added in Primer 10.0, is used then items which are "locked" can be deleted. However they will be replaced with Latent definitions, so this is not a good general solution to the "why can't I delete this?" problem. Forced deletion is described in more detail below.
- Switching **Delete recursive** on or off (see figure above) before pressing **Apply** affects the selection of entities to be deleted. For example if parts are selected for deletion and recursive deletion is switched on then all entities

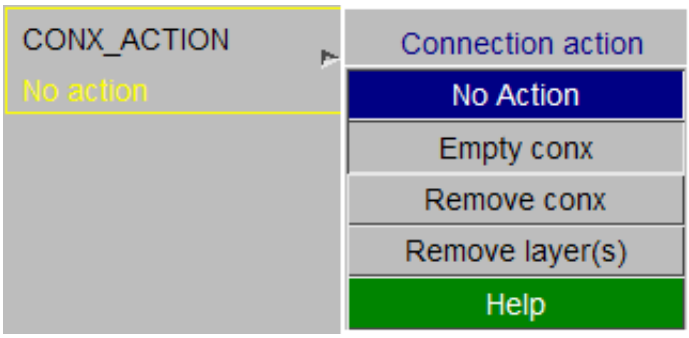
(constraints, elements and nodes etc.) associated with that part will also be deleted. Without recursion only the chosen parts are selected for deletion, and only those which are not referenced by any entities will in fact be deleted. (see rule 1).

- If the **REMOVE from sets** button is selected then PRIMER removes deleted entities from sets. With this option any deleted part (or node etc) will also have its reference removed from any part sets (or node sets etc.).Without this option a part (or node etc.) referenced by a part set (or node set etc.) will not be deleted when it is referenced by another entity (ie the set) which is to remain.

  From V11 onwards set contents can be "locked" against deletion on a per-set basis. This was introduced in order to protect against accidental deletion of set contents where doing so would invalidate the item using the set. This topic is described more fully under Locking Sets against Deletion in section 5, SET.

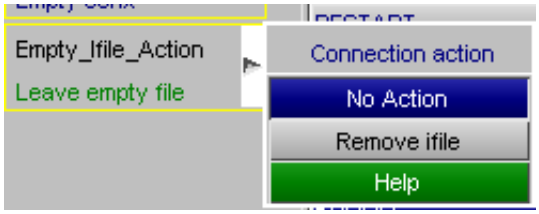
Connections may need some special treatment during deletion. The **CONX_ACTION** popup allows you to set what action to do when PRIMER needs to look at connections that are attached/tied to parts or shells that you select for deletion. The options are:

- **No Action**. PRIMER will not take any special action. This could cause problems if the connection uses `*ELEMENT_BEAM_PID` as the element refers to the part which will stop the part being deleted.
- **Empty conx**. PRIMER will delete the entities (e.g. nodes and solids) that make up the connection and leave the connection latent.
- **Remove conx**. PRIMER will delete the connection and it's entities.
- **Remove layer(s)**. PRIMER will delete the entities (e.g. nodes and solids) that make up the connection and then remove the layer containing the part from the connection. If the connection then only has one layer the connection will be deleted.



Include files also get special treatment.

If you have selected category **Include File** for deletion then the contents of the include file will be selected for deletion, but historically PRIMER behaviour is that the include file itself will not be deleted, even if it is totally empty. This can be a nuisance since if it is an **`*INCLUDE_TRANSFORM`** then any **`*DEFINE_TRANSFORMATION`** that it references will also be locked against deletion, as will any parameters used on either of these cards.



From release 12 onwards there is now an "**Empty Include file action**" option:

- By default "**No action**" is taken, meaning that behaviour is exactly the same as before and deletion of an include file will remove its contents but leave the empty file and its associated *INCLUDE cards in the model.

- Choosing "**Remove file**" will, if the include file itself has been selected for deletion, also delete the **`*INCLUDE`** entry and associated cards if the file is empty.

Notes:

1. An Include file will only be deleted in **Remove File** mode if the file itself has been selected for deletion, either explicitly via category **Include File** or by selecting **Model**. Just deleting the contents of an include file will not delete the file itself, even if this switch is set.

2. If "forcible deletion" is used this will leave latent entries behind. The most common reason for using this option is to replace the contents of an include file, so deleting the include file as well would rather defeat the purpose of the exercise! Therefore in this context the fact that the include file has no explicit contents, only latent ones, does ***not*** qualify it as "empty" and it will not be deleted.

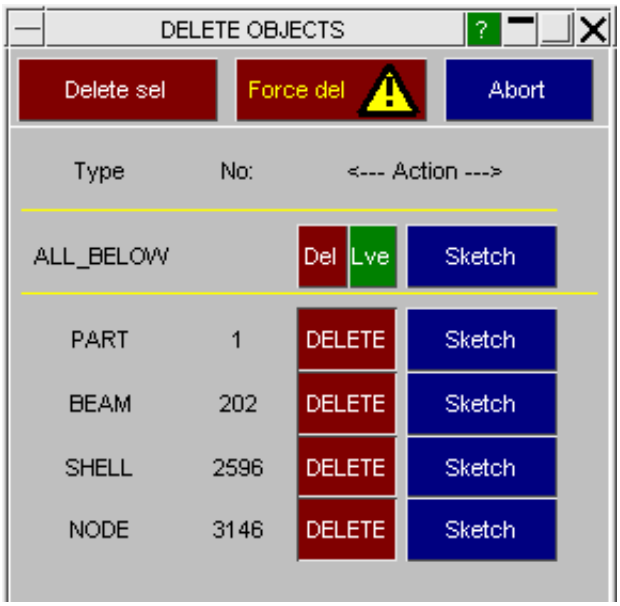
6.36.1.2 Example

The figures below show the effect of recursive deletion. The entities selected by PRIMER with (left column) and without (right column) recursion are shown. In this example the user selected one part to be deleted.

- **SKETCH** will sketch each of the entities that are to be deleted. (**SKETCH** may not be available for all entity types.)

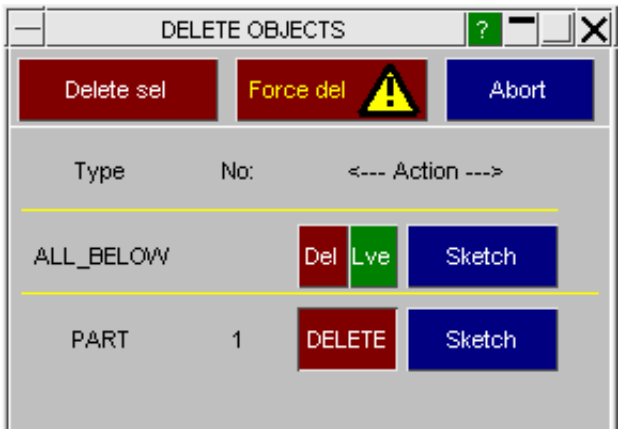
- **ABORT_SEL** will quit the delete operation making no changes to the model.
- The user can toggle each of the entities using the **DELETE/LEAVE** and **DEL/LVE** toggles so that they will be deleted or left as required.
- **DELETE_SEL** executes the deletion using the above rules and reports to the user the number of entities that have been deleted - see the lower pair of figures.

DELETE_RECURSIVE ON

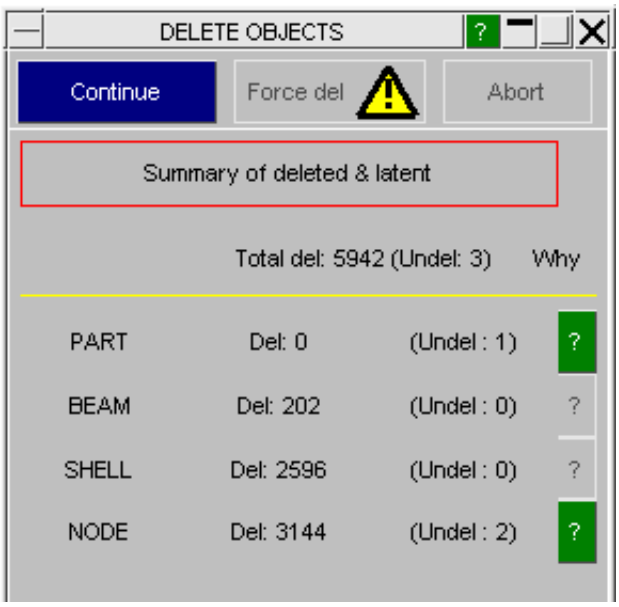


The left figure shows that recursive deletion has found a number of elements, nodes etc.

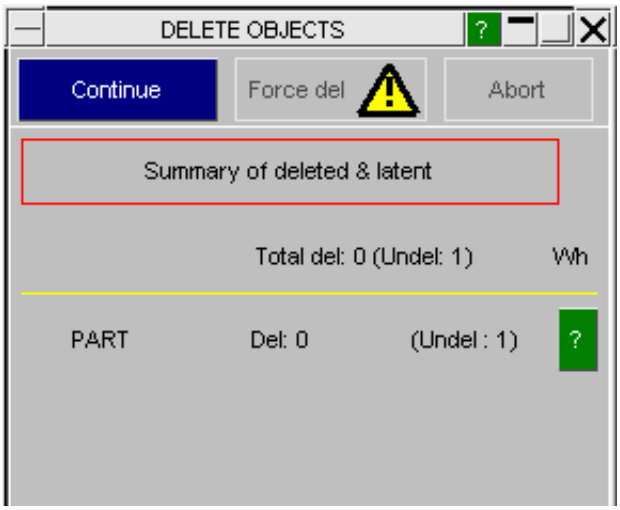
DELETE_RECURSIVE OFF



Without recursion PRIMER finds the entities shown on the right(ie the part only).



Following **DELETE_SEL** the part and its associated data have been deleted on the left, since recursive deletion has picked up all the subordinate items.



On the right nothing has been deleted since the part is"locked" by its elements.

6.36.1.3 Finding out why an item is locked against deletion

You can find out why items have not been deleted by clicking on the relevant **[?]** button in the "Why" column.

For example the PART definition above was not deleted, and the "Why" button produces this panel which shows that it is referenced by:

- Two *SET_PART definitions. (These would not in themselves lock the part if **Remove from Sets** is active)
- Two connections. It is these which are truly locking the part.

You can delve deeper into exactly what is locking the item by using the **X-Refs** button to map the standard Cross-references panel.

6.36.1.4 **Force Del** Using forcible deletion

Deleting items whether they are "locked" or not.

At a first glance forcible deletion, which deletes items even when they are "locked" by a reference from elsewhere, appears to be a simple solution to the problem of removing things which refuse to be deleted. However before using it you should understand what it does and how this can affect your model.

If you select **Force del** rather than the normal **Delete sel** the following happens:

- Items which can be deleted using the normal rules, ie which are not locked, are removed as above.
- Items which remain, but which are marked for deletion are then also forcibly removed ...

  ... but they are replaced with Latent (empty) definitions of the same type.

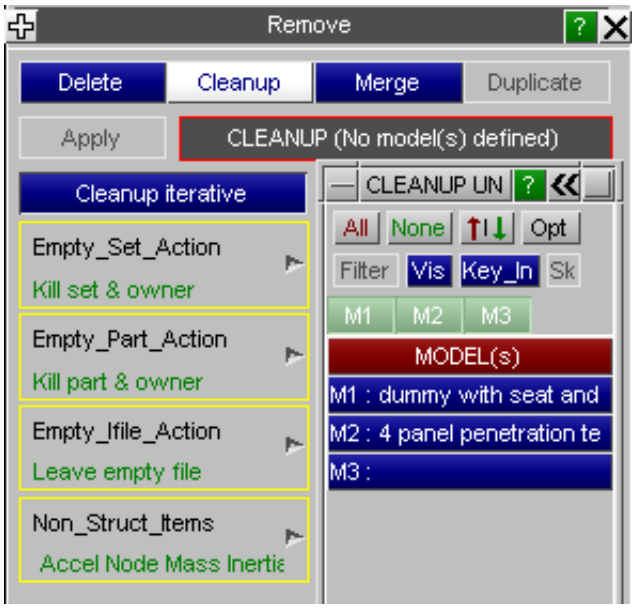
If you run a **Model > Check** on a model that has undergone forcible deletion you will find that all these Latent definitions generate "Referenced but undefined" errors, and the deck is very unlikely to initialise in LS-DYNA in that state.

So if it leaves all these errors what is **Force Del** for? Two special cases:

1. When replacing include files it is useful to be able to delete ***all*** existing file contents, regardless of references from elsewhere in the model, since we know that importing the new include file will re-populate the latent items with new definitions. PRIMER uses this during the [Tools] **Include, Replace contents** process.
2. A very careful user can also perform a similar operation manually. If you know that you are going to replace what you have removed, or deal with the latent definitions in some way, you can use this feature manually. However it will be your responsibility to sort out all the Latent definitions.

What you should ***not*** do is use **Force Del** as a quick fix to the problem that normal deletion refuses to remove locked items. Instead you should use the **[?]** "why" buttons to find out why items are locked, and deal with the problems at source.

6.36.2 Clean-up Unused



CLEAN-UP UNUSED removes redundant entities from models. For example loadcurves that are not referenced, non-structural nodes, etc.

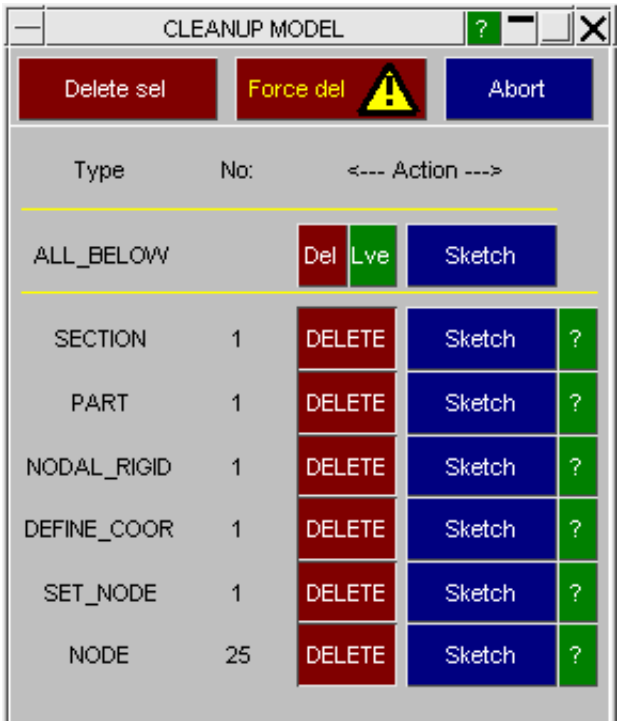
On entering **CLEAN-UP UNUSED** from the **REMOVE** menu the user is prompted to define which model(s) to cleanup. More than one model can be cleaned up at once but the user may find it safer to clean up a single model at a time.

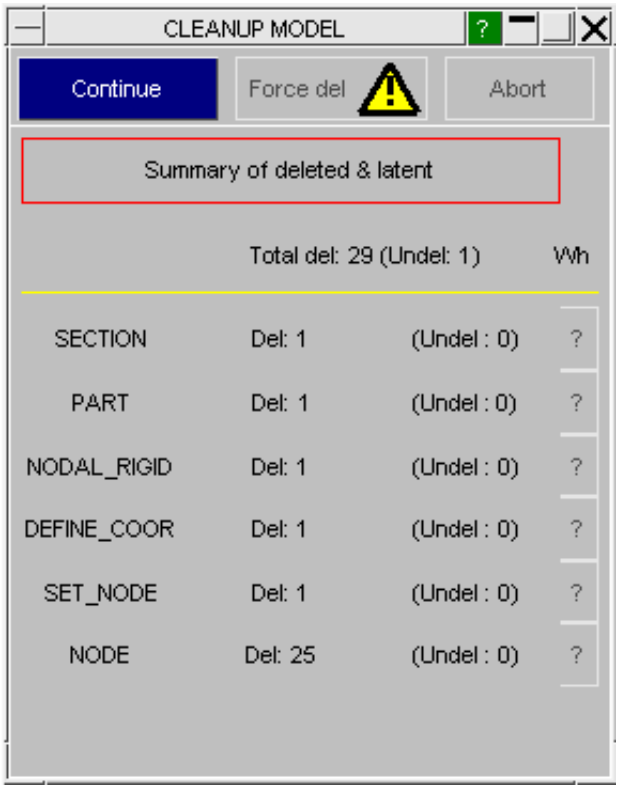
Once model(s) have been chosen **APPLY** starts the clean-up process.

From PRIMER 12 onwards **CLEANUP** can also be applied to a subset of a model, defined by selecting include files or part tree assemblies, see Cleaning up a subset of a model below.

6.36.2.1 CLEANUP UNUSED example

In the same way as for **Delete** you are presented with a list of items identified for removal, and you must confirm these. In the example above pressing **Apply** gives:





This figure shows the confirmation panel of items identified as unwanted.

This figure shows the resulting deletion echo.

Normally everything marked as unwanted should indeed be deleted, but if anything remains you can use the **[?]** button in the Why? column as above to find out what is locking it.

6.36.2.2 Switches controlling CLEANUP UNUSED

Clearly PRIMER must search through the model(s) chosen to identify things that are no longer needed, and there are several switches which may be used to control this process.

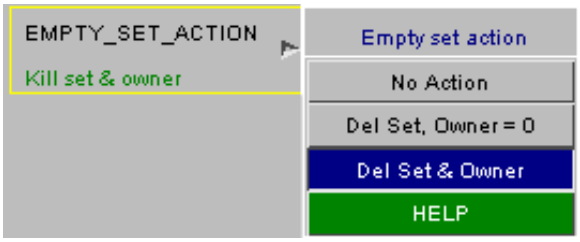
CLEANUP_ITERATIVE Whether to use iterative searching for items.

Sometimes when an item is found to be redundant removing it can lead to other items becoming redundant. It may require multiple passes through the model to identify all these consequential deletions.

For example, if a model contains a part with no elements then in the first iteration the part will be flagged for removal. Iteration 2 will find that the section and material properties etc that this part referenced are also no longer required and will flag them for removal (unless other parts reference these). Iteration 3 will find any loadcurves etc used by the materials that have been flagged for removal (if these aren't used by other materials). And so on until nothing remains to be found.

By default this iterative process will be used, but you can turn it off in order to limit the extent of a clean-up operation to a single pass. This can give more control over what is removed in each **CLEANUP** operation.

EMPTY_SET_ACTION Dealing with empty SETS.



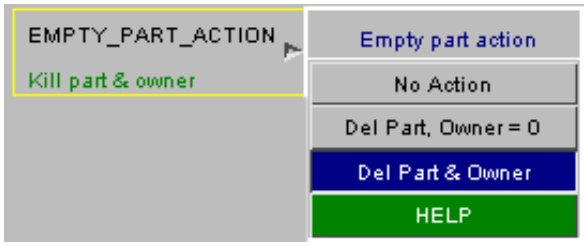
When all the contents of a SET have been removed (following a **REMOVE** operation) the empty SET definition itself may remain.

This is not strictly illegal, but it can cause problems in the analysis code at run-time since LS-DYNA may crash if sets with no contents are found.
Therefore PRIMER treats it as an error, and provides the following options for dealing with it:

No action	The set is not removed, and references to it remain
Del Set, Owner = 0	The set is marked for deletion, and any references to it are replaced with a zero. This can cause unexpected outcomes when <`set id = 0`> implies "use the whole model", as is the case in some contexts - use with care!
Del Set & Owner	Both the set ***and the item referring to it*** are marked for deletion. This is the default setting, and generally the most useful.

.There is an exception in the last (**Del Set & Owner**) case in that where a reference to a set is optional, for example "set of nodes exempted from ..." where replacing the reference with a zero would be harmless, that solution is adopted instead and the "owner" definition is not marked for deletion.

EMPTY_PART_ACTION Dealing with empty Parts



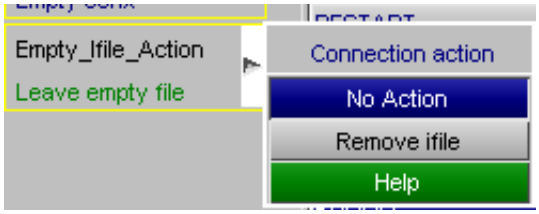
If all the elements have been deleted from (or transferred out of) a part then it will be empty.

As with empty sets this is not strictly illegal, but it can cause problems in LS-DYNA.

Therefore PRIMER treats it as an error and provides the following options:

No action	The part is not removed, and references to it remain.
Del Part, Owner = 0	The part is marked for deletion, and any references to it are replaced with a zero. This can cause unexpected outcomes when <`part id = 0`> implies "use the whole model", as is the case in some contexts - use with care!
Del Part & Owner	Both the part ***and the item referring to it*** are marked for deletion. This is the default setting, and generally the most useful.

EMPTY_IFILE_ACTION Dealing with empty include files..



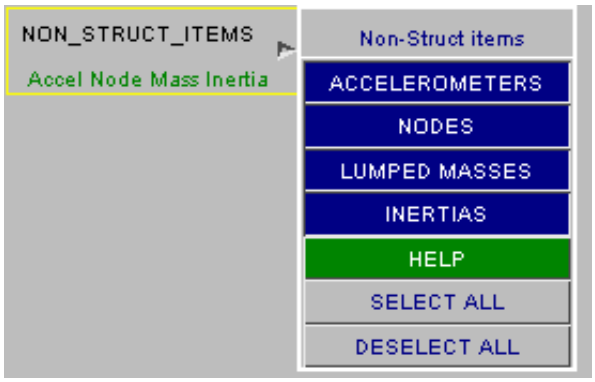
If cleaning up of the model will leave empty include files the historic behaviour has been not to remove these, but rather to leave the the **`*INCLUDE`** card referencing the empty file.

This can be a nuisance since if it is an **`*INCLUDE_TRANSFORM`** then any **`*DEFINE_TRANSFORMATION`** that it references will also be locked against deletion, as will any parameters used on either of these cards, even if these are legitimate subjects for a "clean up".

From release 12 onwards there is now an "**Empty Include file action**" option:

- By default "**No action**" is taken, meaning that behaviour is exactly the same as before and an empty file and its associated **`*INCLUDE`** cards will be left in the model.
- Choosing "**Remove file**" will permit the include file to be cleaned up if it is empty. This will, in turn, also permit any **`*DEFINE_TRANSFORMATION`** and **`*PARAMETER`** cards referenced by that file to be cleaned up

NON_STRUCT_ITEMS Dealing with items that have no structural purpose



Following the removal of other things you can be left with valid and legal objects which are nevertheless "non-structural", meaning that they will not play any part in an analysis.

PRIMER can detect and mark for deletion the following:

ACCELEROMETERS	Accelerometers which exist in isolation
NODES	Nodes which are not attached to elements, not extra nodes on a rigid part or some other constraint, and not useful in any other context.
LUMPED MASSES	Lumped mass elements attached to non-structural nodes.
INERTIAS	Inertia elements attached to non-structural nodes.

Some of the seatbelt-related elements (sliprings, pretensioners, etc) can also be non-structural by the definitions above. However they may often be imported as part of pre-meshed dummy models, and will become structural when attached to a vehicle, thus it would be unfortunate if they were accidentally deleted. Therefore they are not included in these checks and will need to be deleted manually if required, but the overhead of leaving them is minimal.

Other things marked for deletion during a cleanup operation.

As well as items which are unused, plus those which meet the criteria above, the following things are also automatically checked and marked for deletion as required:

CONSTRAINED definitions which have become redundant or invalid:
- Generalized welds referencing sets containing fewer than 2 nodes;
- Linear constraints ditto
- Node sets ditto
- Shell to solid defns ditto
- Tied nodes ditto
- Joints for which attached parts are absent, no longer rigid, or non-structural;

SEGMENTS that are no longer valid:
- Because their parent set, load or other definition has been removed;
- Because they no longer lie on a shell element, or the face of a 3D element.

"Latent" definitions serving no useful purpose plus their referees:
- Items referenced in sets, boundary conditions, database cards, initial definitions, etc that have never been explicitly defined or referenced in other contexts. These can be deleted along with the references to them since they do not serve any independent purpose in isolation. (For example a restraint on a node not referred to anywhere else is redundant.)

6.36.2.3 When multiple calls to CLEANUP UNUSED may be required

Usually turning the "iterative" switch on will identify all items that are to be removed in a single operation, but if a large number of different item types are to be removed it may require multiple clean-ups to remove absolutely everything unwanted from a model.

The reason is that in iterative cleanup mode PRIMER has to ask the question "if I remove this then can I remove that?" to many levels of complexity, and if a lot of different entity types are removed there comes a stage at which this gets too difficult to resolve in a single operation.

Therefore if a long list of different item types is identified for removal it is usually worth repeating the **CLEANUP UNUSED** operation until the message **`"No items found to delete"`** is returned. This guarantees that no further unwanted items remain.

6.36.2.4 More aggressive treatment of Latent items from V11.1 onwards

Prior to release 11.1 the treatment of latent (referred to but not explicitly defined) items during cleanup was very conservative. Such items were not marked for deletion if anything explicitly defined referred to them, and this could lead to annoying situations. For example:
- Some latent **`*NODE`** cards are found
- These exist only because they are referred to by some explicit **`*LOAD_NODE`** cards

By inspection both the NODE and LOAD cards are redundant in this example and should be cleaned up, but this would not happen.

Therefore from V11.1 onwards a more aggressive treatment of latent items has been implemented.
- Latent items are inspected as before.
- If all references to them are "junior" in the PRIMER hierarchy then this no longer locks them against deletion.
- In this situation both the latent item itself (NODE in the example above) and the referring items (LOAD in this example) are marked for cleanup.
- There is an exception to the "installed junior item no longer locks latent senior item" logic: if the junior item is "structural" then it still locks both the latent item and itself against cleanup.

"Junior"and "senior" refer to the standard PRIMER hierarchical order which, for example, makes *PART senior to *ELEMENT, and *ELEMENT senior to *NODE. More information about this can be found in Section 2.12 on Operational Hierarchy in PRIMER.

"Structural" items are, broadly, items that can usefully exist on their own (eg shells and solids); and also PARTs that contain such elements. A node is structural if attached to such an element, but non-structural if it has nothing useful attached to it. Consider the following two examples:

Examples of the new, more aggressive cleanup treatment of latent items			
Example 1	Some Latent *NODE cards exist.	Only Installed *LOAD_NODEs refer to them	Both *NODE and *LOAD cards will now be marked for cleanup because LOAD is junior to NODE, and LOAD is non-structural
Example 2	Some Latent *PART cards exist.	These are referred to by Installed *ELEMENT_BEAM cards	Neither *PART not *ELEMENT cards will be marked for cleanup because while BEAM is junior to PART, it is "structural".
Example 3	Some latent *SECTION cards exist.	These are referred to by Installed *PART cards	PART is junior to SECTION, however a PART containing elements is "structural". Therefore only empty parts with no elements will be marked for cleanup, and only section cards referred to solely by such empty parts will also be marked for cleanup.
Example 4	Some latent *ELEMENT_SOLID cards exist.	These are not referred to by anything.	Because nothing refers to them these will be marked for cleanup. (This behaviour is unchanged.)

6.36.2.5 Cleaning up a subset of a model

From PRIMER 12 onwards it is possible to clean up only a subset of a model by selecting either include files or part tree assemblies. These options will be presented for selection in the main **Remove, Cleanup** menu if present in the model(s), and you can also perform the selective **Cleanup** operation from:

The **Part Tree**	Right click on either **Assembly** or **Include file** rows and select **Cleanup**
The **Include file Tree**	Right click on an Include file and select **Cleanup**

6.36.2.6 Cleaning up a subset of a model

By default, when deleting entities, PRIMER will write UNDO information to disk, so that any deletion operation can be undone. If deleting large amounts of information, the UNDO writing can be slow, so it can be turned of by unchecking the "save undo information" checkbox:

6.36.3 Merge nodes

This operation will permit coincident (and/or close) nodes to be merged.

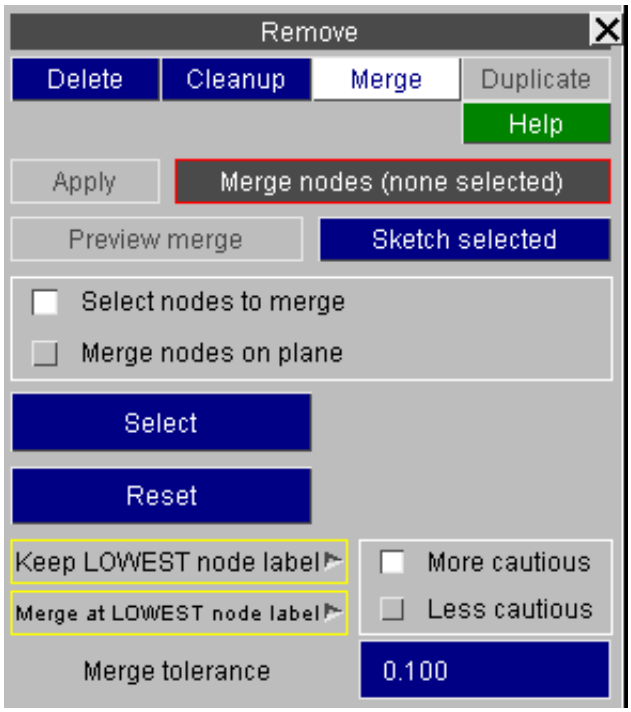
The **MERGE NODES** function is accessed either through the **Remove** button on the top panel or through **Volumes I & II**->**NODE**->**Merge**.

Using Merge nodes

Merge applied to a selection of nodes or on a global plane, using user defined tolerance.

- retain lowest/highest node label
- merge at lowest/highest node or at average position

PREVIEW MERGE will highlight the nodes to be merged.



More cautious and Less cautious modes

Merge nodes has two different modes: **More cautious** and **Less cautious**.

For simple cases where only 2 nodes will be merged and the nodes are only on elements such as shells or solids there is no difference but for cases where there are more than 2 nodes to merge or where there is a constrained item such as a nodal
rigid body or extra nodes on a rigid body using the 2 nodes to merge you will get a different result depending on the mode.

To understand the difference between the 2 modes consider the following examples:

Example 1

During a merge PRIMER has found 3 potential nodes to merge together. The 2 slave nodes are on a nodal rigid body.
In the more cautious mode PRIMER will not merge any nodes.
In the less cautious mode PRIMER will merge all 3 nodes together (removing the duplicate node from the NRB).

Example 2

During a merge PRIMER has found 3 potential nodes to merge together. The 2 slave nodes have a beam between them.
In the more cautious mode PRIMER will not merge any nodes.
In the less cautious mode PRIMER will choose the node on the beam which is closest to the master and merge it to the master node.

Example 3

During a merge PRIMER has found 2 potential nodes to merge together. The master and slave nodes are on the same nodal
rigid body.

In the more cautious mode PRIMER will not merge any nodes.
In the less cautious mode PRIMER will merge the slave node to the master node.

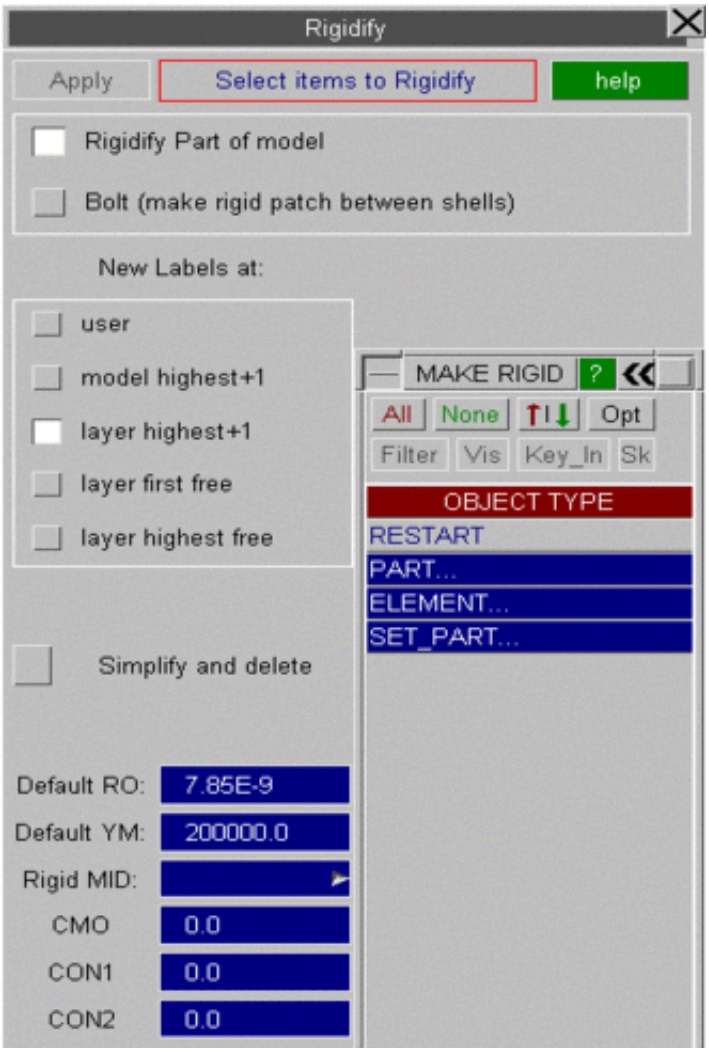
Collapsing shells

By default Merge nodes will not collapse quad shells to trias. This can be turned on by using the **Collapse quad shells to trias** option in the **Node replace/merge** section in **Program options** (in the **Options** menu). This can also be changed with the **`replace_collapse_shell`** preference.

6.37 RIGIDIFY

The **RIGIDIFY** function has 2 modes.

6.37.1 Rigidify Part of Model



Rigidify Part of Model is designed to make rigid a selection of Parts, Elements or Part Sets in a model. If some elements of a part are selected, it does this by creating new rigid parts and moving the selected elements into them. If a part is selected or all elements of a part, a rigid material card is substituted. All rigidified parts are slaved to a dummy master part.

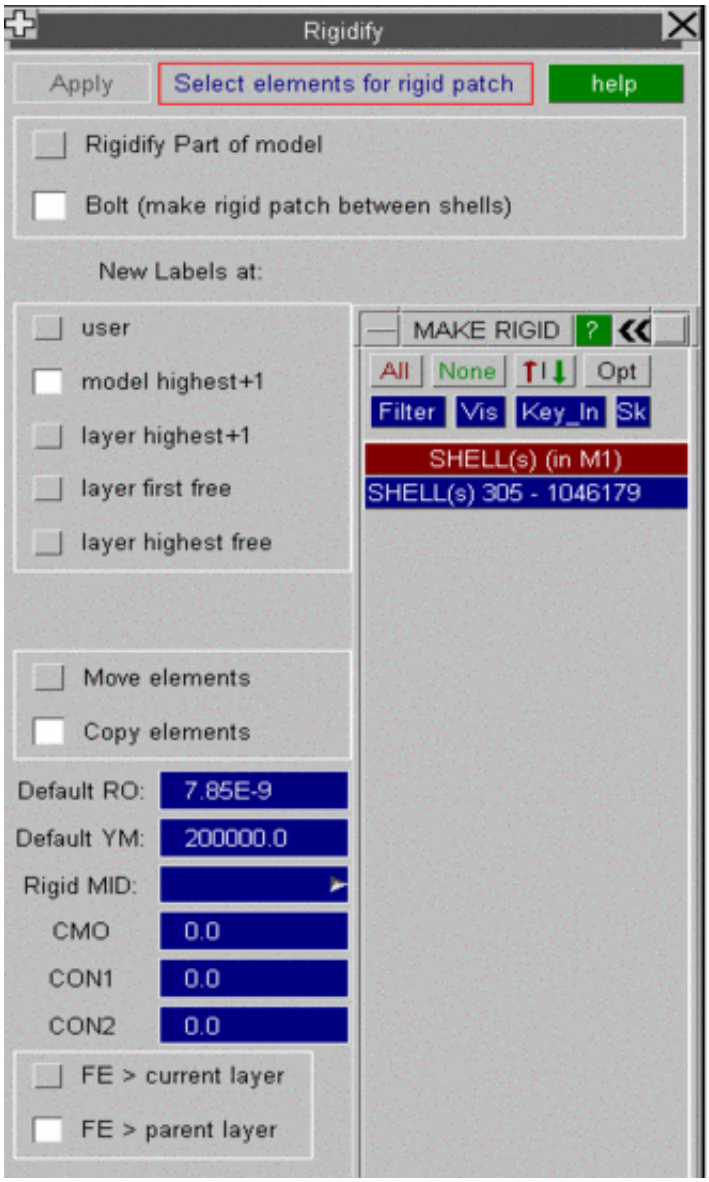
The density and modulus of newly created materials will normally match those of the deformable ones they replace, to preserve the part mass. If these values cannot be determined, the default values will be used.

The user simply selects items in the usual way with the object menu and presses **APPLY**.

However, the consequences of applying **RIGIDIFY** to the model are **considerable and irreversible**. It is therefore recommended that the user always saves the model before he/she presses **APPLY**.

The user can choose to delete the original elements chosen to rigidify if they wish. In this case, the mass and inertia properties of the original elements are applied to the dummy master part in the for of a *PART_INERTIA. This option is activated by checking the **Simplify and delete** check box.

6.37.2 Create Rigid Patch (bolt)



Create Rigid Patch (bolt) will rigidify a selection of elements and (if they belong to different parts) create a merge onto the first part selected. The option is available to copy the selected elements rather than just move them into the new rigid part (default).

New Part Labels: These may be created starting with a user defined label (which will be automatically incremented to first free) or at the highest in layer + 1 for the part or at model highest + 1. If highest in layer + 1 is not available, model highest + 1 is used. There are also options for first free and highest free in a layer.

6.37.3 How RIGIDIFY will change the model

All the structural elements (shells, thick shells, solids and beams) that have been selected will end up in a rigid part slaved to the dummy master rigid part.

Selection of a deformable Part: if a part is selected directly or all the elements of a part are selected, a new material will be generated (keeping the same values of E and rho) and the part->material reference updated. The part will be merged to the dummy master part.

Selection of deformable elements: if some elements of a deformable part are selected, a new part and material will be created and the elements will be moved into this part. The part will be merged to the dummy master part.

Selection of a rigid part or an element of a rigid part: in this case only the merge to the dummy master is made.

The rigidified parts will all be written to a group, so the user can easily determine the mass properties.

6.37.4 How RIGIDIFY will flag items for deletion

To ensure that the rigidification of the model does not incur errors Primer will flag selected items for deletion.

Certain nodal constraints within the rigidified area will require removal, these include joints, constrained welds, boundary prescribed motions, boundary spcs, constrained extra nodes and nodal rigid bodies. In these cases connection to nodes located outside the rigidified area will be retained by the creation of additional constrained extra nodes.

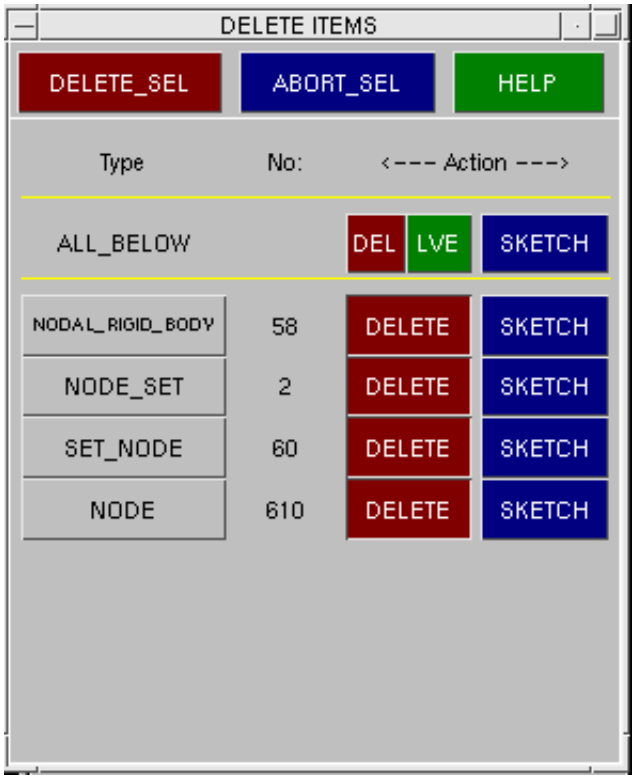
Rigid body merges, where the part to be rigidified is the slave, will be removed to avoid clashing with the merge onto the dummy master.

Spotweld beams will be flagged for removal where both sides of the weld are rigidified. Where one side of the weld remains deformable, the spotwelds will be retained, but a warning will be issued if the contact is not suitable for rigid parts.

Contacts where all parts have been rigidified will be flagged for removal as they are superfluous.

The user has control over the deletion panel and may choose to leave the offending items. If deletion is aborted, a Primer model check may contain errors.

If **Simplify and delete** is selected, the elements will also be flagged for deletion.



6.38 SCRIPT Using JavaScript in PRIMER

Scripting is covered in section 10 of this manual, see:

Using JavaScript in PRIMER

A Brief Tutorial

The Javascript API Manual

JaDe: The JavaScript Debugger

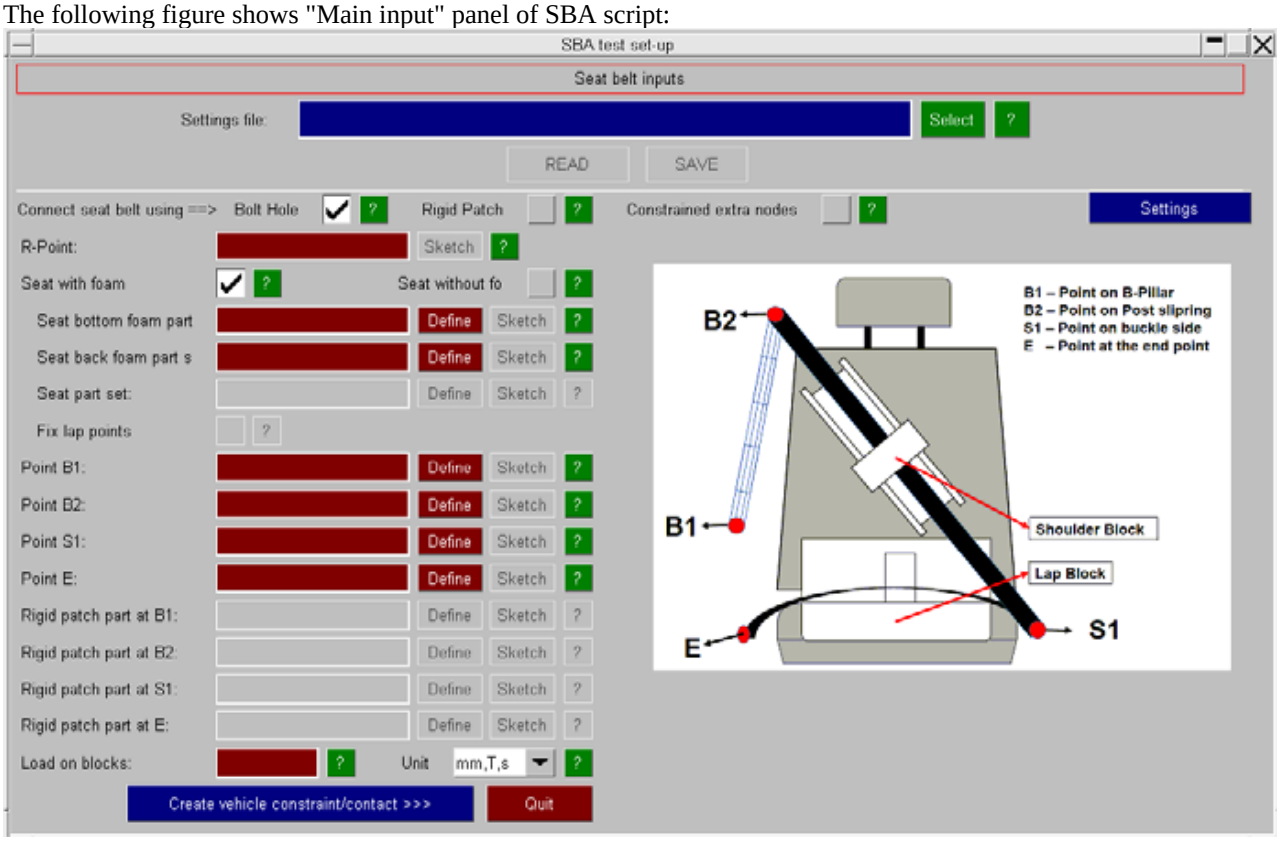
6.39 Seat Belt Anchorage Automation Script (ECE-R14)

Introduction

Seat belt system need to be tested to make sure its proper functioning in case of vehicle impact. The belt anchorages must be able to withstand a static test load simulating vehicle impact.

ECE-R14 is one of the tests that provide the assurance of sufficient strength resistance of all anchorage points. The SBA (seat belt anchorage analysis) script positions the loading devices (lap block and shoulder block) attached with seatbelt system at R-point in the vehicle and setup the analysis according to ECE R14 specification.

The following figure shows "Main input" panel of SBA script:



The following options are available on the main input panel:

Seat belt connection	Select an option from the below to connect the end of the belt to the body: Bolt hole: Create bolt for each anchorage to join the end of the belt to the seat/body. Rigid Patch: Use rigid patches on the BIW for each anchorage to join the end of the belt to the seat/body. Constrained extra nodes: Create *CONSTRAINED_EXTRA _NODES for each anchorage to join the end of the belt to the seat/body.
Settings file	A settings file may be read or written which may include mandatory information and other data required to start calculation. The reading/writing of a settings file is optional.
R-point	R-point is the relative location of the seated dummy's hip point when the seat is set in the rearmost and lowermost seating position.
Seat data input	Select seat with foam or seat without foam. For seat with foam, provide part set for bottom and back foam separately. For seat without foam, provide part set comprising of all the parts that might effect positioning of the impactor.
Fix lap points	This is applicable to seat without foam. If this option is ON, seat belt points will be fixed on lap block.
B1	B1 is the point on B-Pillar.
B2	B2 is the point on B-Post.

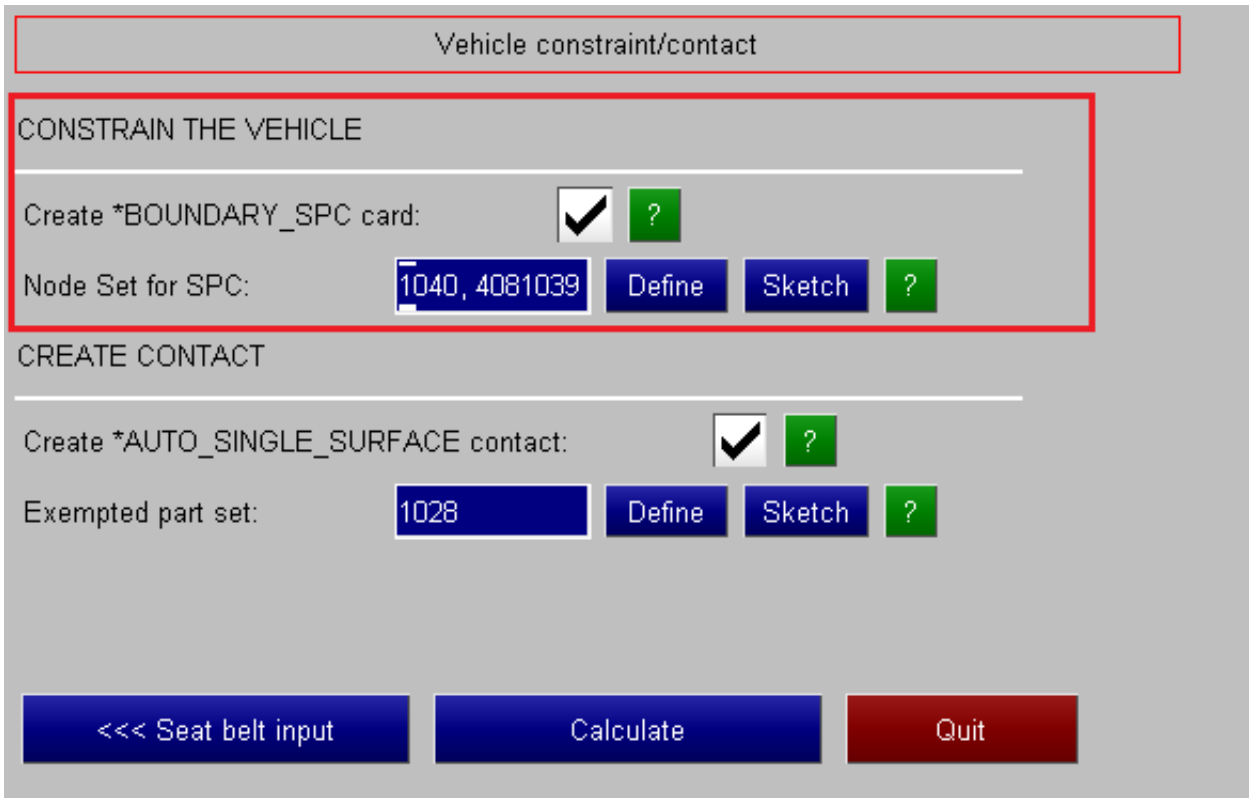
S1	S1 is the point on buckle side
E	E is the point at the end
rigid patch/node set at B1	Either specify rigid patch part(if rigid patch is selected to connect seat belt) or node set at B1 (if extra_node is used for connection).
rigid patch/node set at B2	Either specify rigid patch part(if rigid patch is selected to connect seat belt) or node set at B2 (if extra_node is used for connection).
rigid patch/node set at S1	Either specify rigid patch part(if rigid patch is selected to connect seat belt) or node set at S1 (if extra_node is used for connection).
rigid patch/node set at E	Either specify rigid patch part(if rigid patch is selected to connect seat belt) or node set at E (if extra_node is used for connection).
Load on blocks	Load input on lap and shoulder block.
Units	Specify unit system of the selected model.

Click **Setting** to open the setting panel to select seat type, set maximum iteration for seat belt fitting, bolt angle tolerance, translate shoulder block in global axis, rotate lap block, and select middle seatbelt type.

Constrain the vehicle

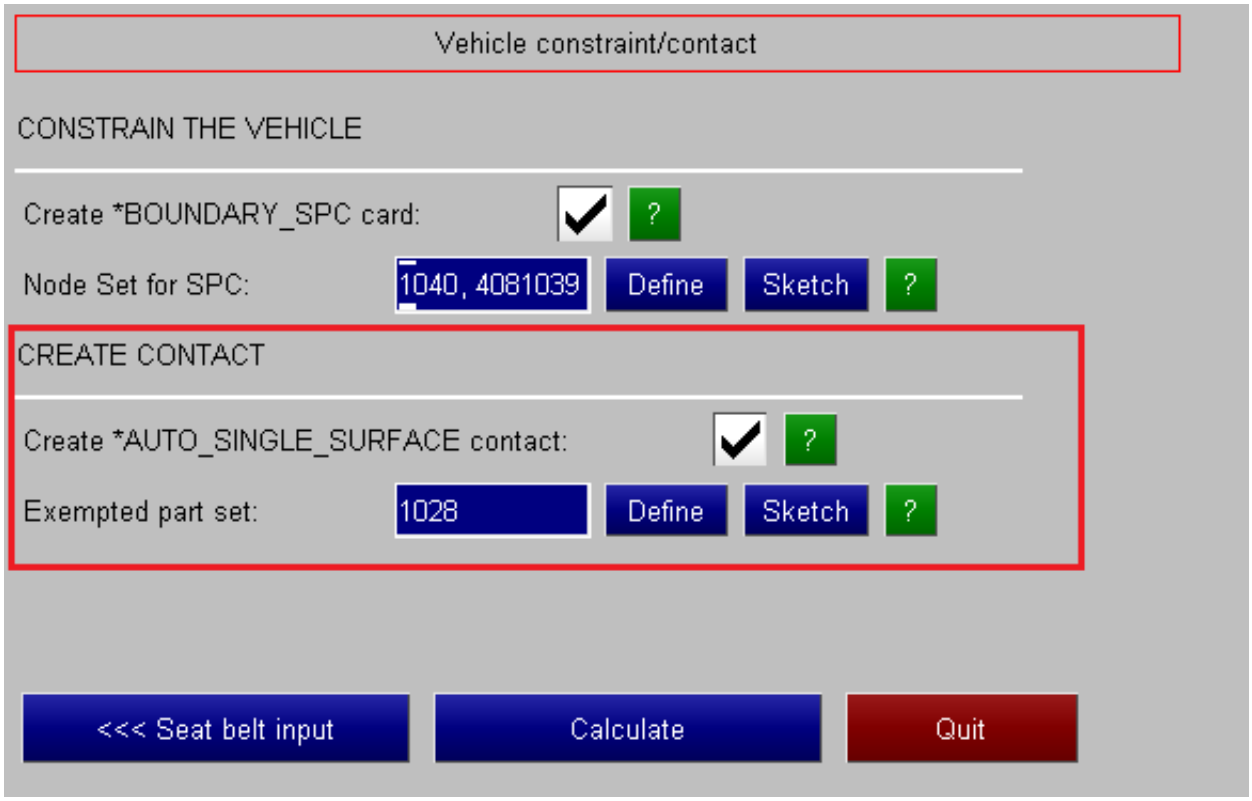
Click **Create vehicle constraint/contact** to create vehicle constraint and contact. All the options available on "Vehicle constraint/contact" input panel are optional.

Select "create *BOUNDARY_SPC card" option as shown below to constrain the vehicle by creating *BOUNDARY_SPC card. Use "Node set for SPC" text box to provide set id for *BOUNDARY_SPC card



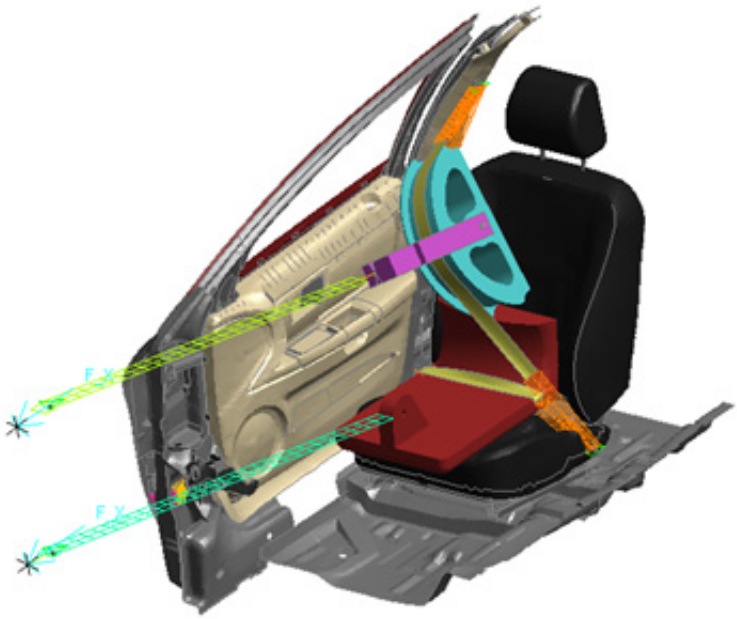
Create Contact

Select "create *AUTO_SINGLE_SURFACE contact"option as shown below to create a *CONTACT_AUTOMATIC_SINGLE_SURFACE with exempted part set in slave side. Impactor parts will get added automatically to the exempted part set. Use "Exempted part set" option to add any other part sets in the exempted set.



Calculate

Once all the input conditions have been defined, **Calculate** will become active. Hit **Calculate** to start the calculation. After the calculation, the loading device will be positioned at R-point in the vehicle as shown below:



6.40 SEAT-BELTS Fitting seatbelts and related elements.

Links to particular sections

If you know what you want then please jump to the relevant section using the links below, but if you are new to belt fitting in PRIMER please read the tutorial below first.

Description	Description of belt fitting
Define structure	Creating a "structure" definition onto which to fit the belt
Create path	Creating an initial belt "path"
Fitting	Fitting the belt "path" to the structure
Meshing	Creating a belt &/or shell element mesh on the fitted belt path
Contact	Creating a contact between belt and dummy (optional)
Related items	Creating retractors, sliprings, etc
Auto Refit	Automatically refitting the belt when the dummy moves
New Dummy	Replacing one dummy with another inside a belt definition
Settings...	General settings for seatbelt fitting.
Batch fitting	Using command-line (batch) commands to refit belts.
Saved belt data	What is saved in a "Belt" definition. (See also Appendix V for the detailed card formats)

A mini-tutorial on PRIMER seatbelt fitting.

The seatbelt fitting routines are designed to fit one or more belts to a structure, using **`shell`** and **`1d and/or 2d seatbelt`** elements. It is also possible to define the belt-related items such as **`retractors`**, **`sliprings, sensors`**, etc, and to specify database cross-sections down the belt.

While intended primarily for Occupant analyses, the "structure" used for belt fitting does not have to be a dummy: it can be any combination of solid, shell and thick shell elements, and belt fitting may be used in any context where a line of elements needs to be fitted around an arbitrary shape. In order to create and fit a seatbelt you need to go through the following process:

(T1) Define the components to which the belt will be fitted.

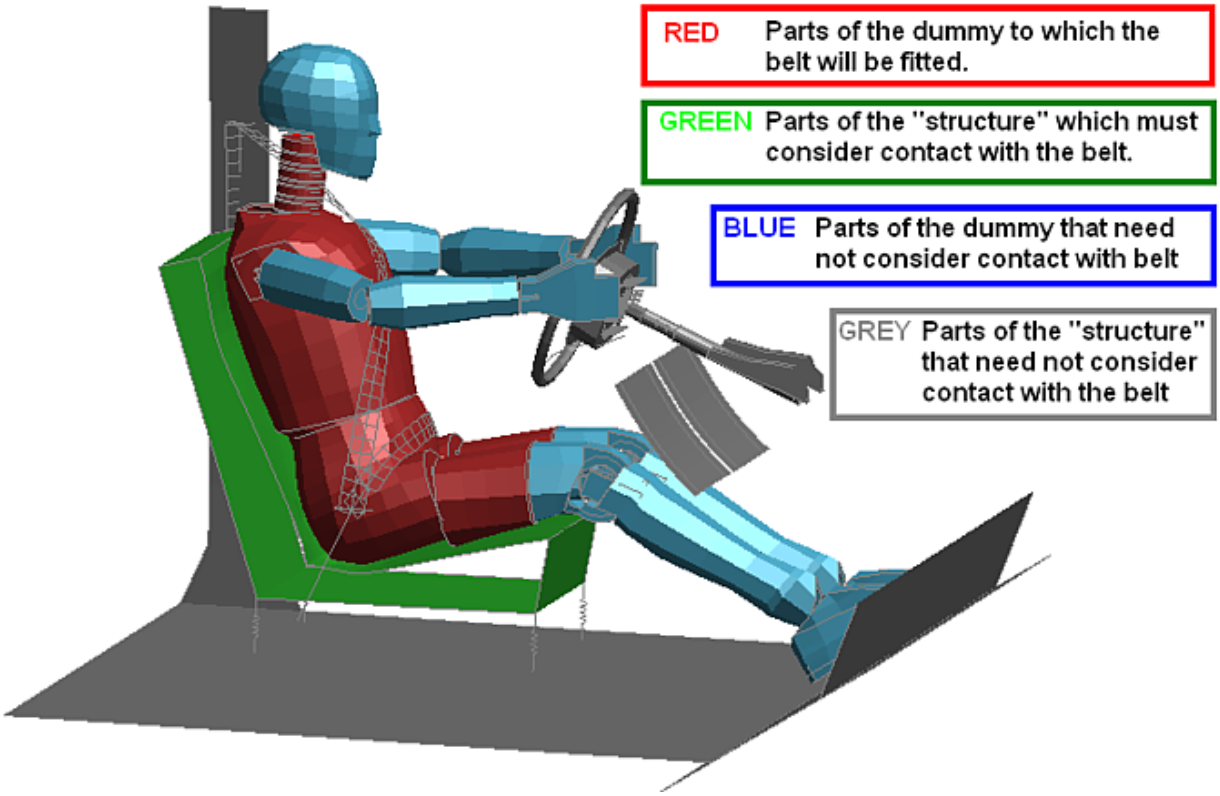
The first step is to define those elements of dummy and nearby structure against which the belt will be fitted. Since fitting (form-finding) is an iterative process that involves contact calculations it is worthwhile trying to minimise the number of elements to be considered, since this will speed up the process.

The image below shows a very simple sled model in which those elements which need to be considered for belt fitting, ie those which the belt will contact, have been coloured red (dummy) and green (nearby structure). Dummy elements that need not be considered are blue, and structure that need not be considered is grey.

During the actual analysis the belt may come into contact with airbag, B-Post, steering wheel and other components, and the contact surfaces used for the LS-DYNA analysis itself should consider this. However ***during belt fitting*** only contact with the red and green elements needs to be considered, and the process will be faster if only these are used.

(It might be reasonable also to exclude the dummy's upper leg elements in this case, however experience shows that when seat positions are moved back and forth it is possible for the upper legs of dummies to contact the lap section of the belt during fitting, even if the final shape does not pass over them, so it is usually sensible to include the upper legs in the contact as shown here.)

For a more detailed description see Section 6.40.1 **Define:** Defining "Structure" for seatbelt fitting



(T2) Define the basic path of the belt.

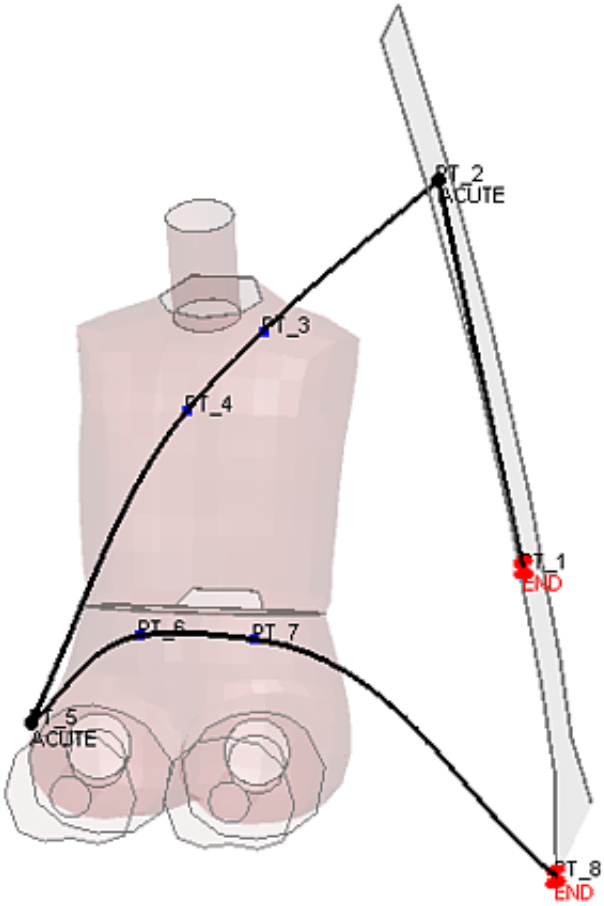
The image on the left below shows the basic path, made up of 8 points located at nodes on the structure or dummy.

- Points 1 and 2 are on the B post
- Points 3 and 4 are on the dummy's chest
- Point 5 is at the pelvis buckle position
- Points 6 and 7 are on the dummy's pelvis
- Point 8 is the final attachment point on the floor-pan.

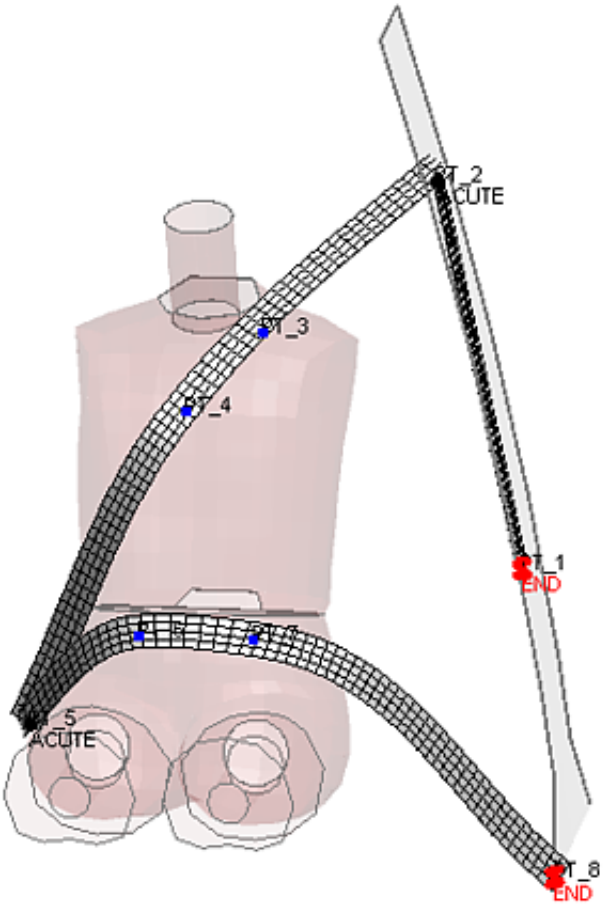
The belt fitter creates a curved path through these points, with breaks where the angle is acute, the black line in the left hand image.

Then it projects this path forwards from the dummy, calculates the plane of the belt elements, and creates an initial meshing path as shown in the right hand image. At this stage the meshing path is some way forward from the dummy, and not in its final shape, for example the lap section is far too high above the pelvis. This does not matter at this stage, the important thing is that the meshing path should be topologically "outside" the dummy and not have any initial penetrations into it.

For a more detailed description see section 6.40.2 Creating a belt "path"



Basic path, here using 8 points



Meshing path derived from this.

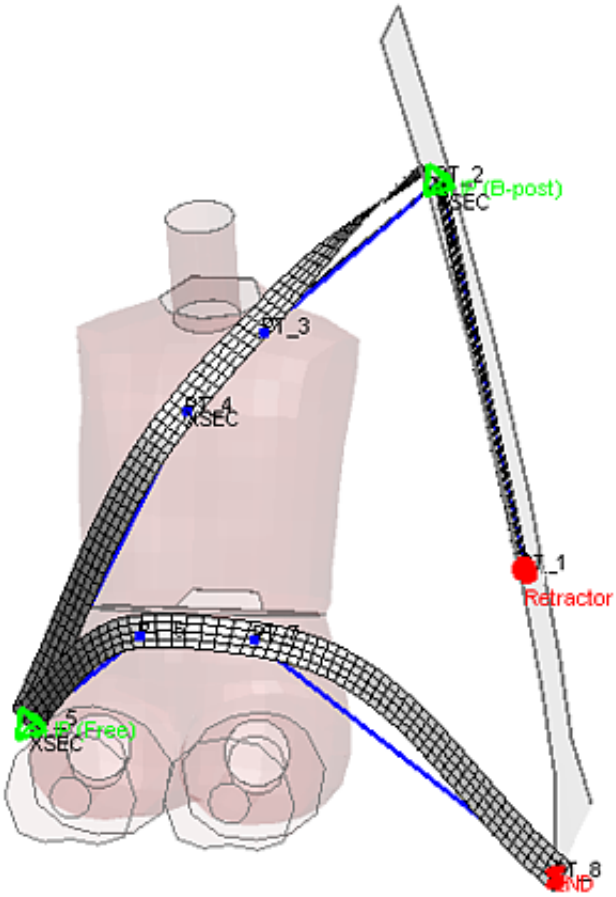
(T3) Add detail to and adjust the meshing path.

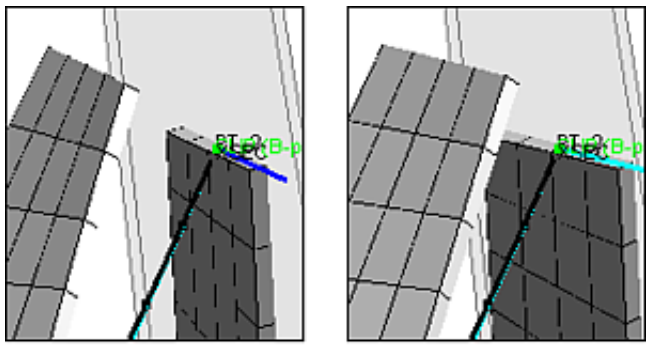
The left hand image shows typical additions to the basic belt path which add detail.

- A retractor has been defined at point 1 on the B-Post
- Sliprings have been defined at points 2 (top of B-Post) and 5 (pelvis buckle)
- Database cross-sections to obtain cut forces during the analysis have been added at points 2, 4 and 5

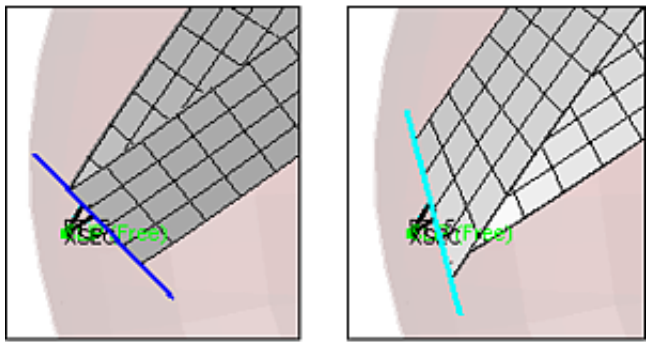
The right hand image shows how the angle and position of both sliprings has been adjusted to give an accurate match with the geometry at those points. (Points can be moved and the twist of the belt can be adjusted to obtain a better initial shape.)

For more information see Adding detail to the belt path.





Adjust orientation of B-Post Slipring to the correct angle (point 2).



Adjust position and orientation of Pelvis slipring to line up with belt buckle (point 5).

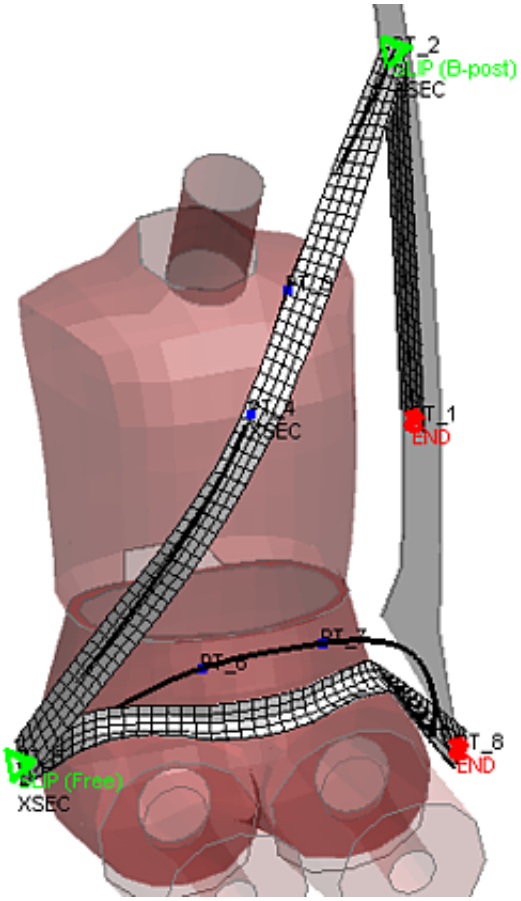
The fully configured meshing path before fitting

(T4) "Fit" and then mesh the final shape.

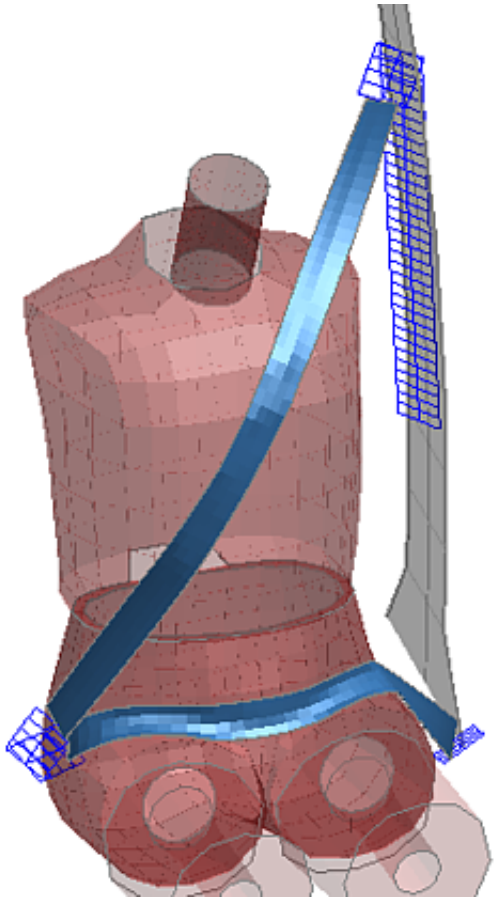
The left hand image shows the final shape of the meshing path after "fitting". this is a form-finding process in which the belt is pulled inwards onto the dummy until it makes contact, and it can also slide across the dummy in order to try to find the shortest path between ends. In this example the shape of the chest section has not changed very much, but observe how the pelvis section has pulled down onto the lower torso to achieve a realistic shape.

The right hand image shows the final part of the process: meshing the fitted belt. In this example the mesh consists of 1d belt elements between retractor and slipring, and also a short stretch of 1d belt elements either side of both sliprings. The chest and pelvis sections are meshed with fabric shell elements.

For more information see FIT: Commencing the form-finding operation



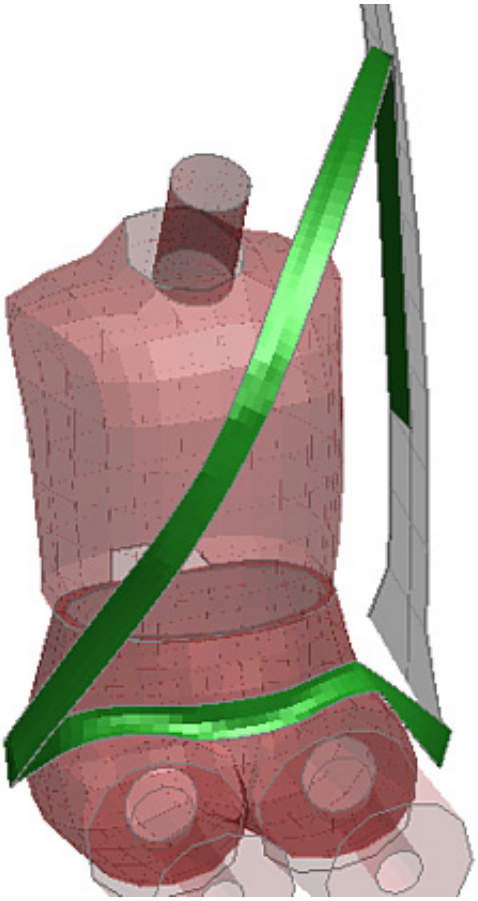
The fitted meshing path after form-finding



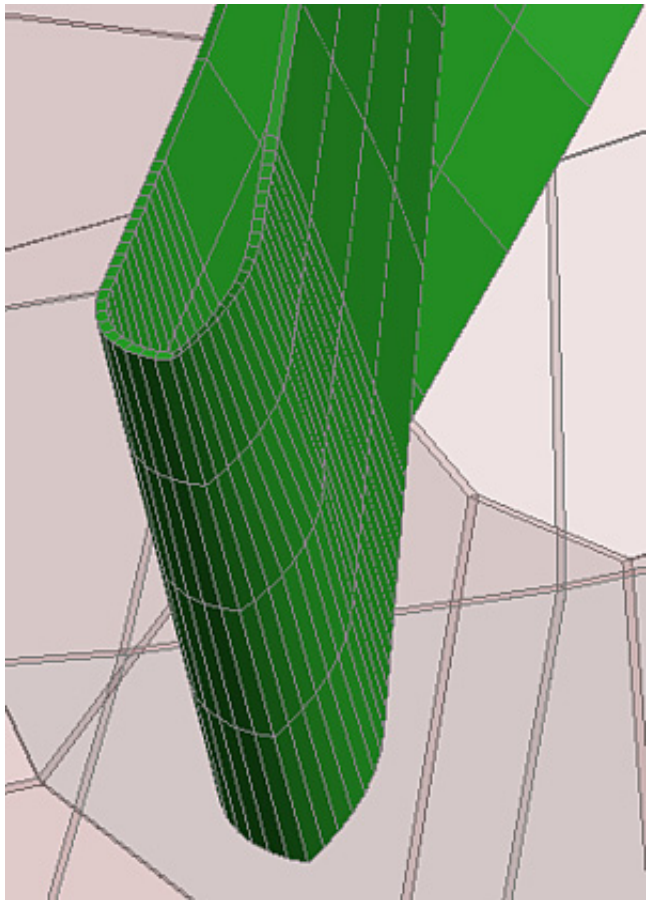
The final meshed result, here 1d belt elements and shells

PRIMER can mesh witha mixture of 1d belt elements, 2d belt elements and traditional shells, and it is easy to remesh a fitted path at will with a different combination of element types. For example the left hand image below shows the same fitted meshing path remeshed with 2d belt elements for the whole belt. The mesh is now continuous through the sliprings, with all the various node and element sets required by 2d sliprings and retractors created as required.

It is also possible to replace discrete slipring elements (*ELEMENT_SEATBELT_SLIPRING) with explicitly meshed shells or 2d seatbelt elements wrapped around the specified radius. The right hand image below shows such a mesh at the pelvis buckle location in the model above.



Remeshing the result above entirely in 2d belt elements



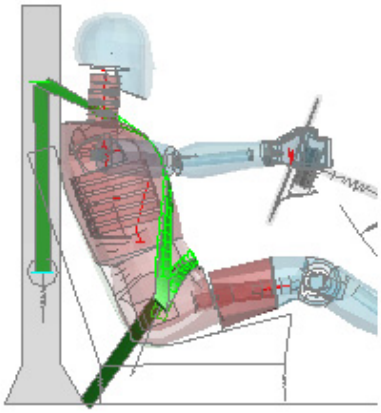
Detail of an explicitly meshed (radiused) slipring

PRIMER also allows you to define the attributes of retractors, sliprings, sensors, pre-tensioners, etc and these can be "remembered" when the belt is remeshed so that they don’t have to be repeatedly redefined.

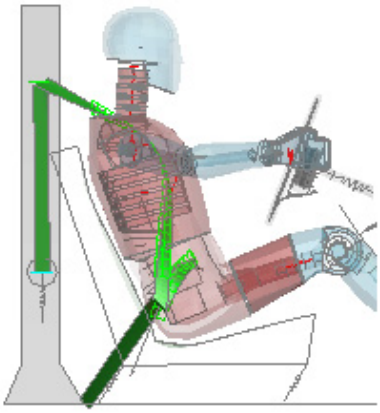
(T5) Refitting belts automatically.

The first time you fit a belt to a dummy you will need to define the path and mesh the result manually.

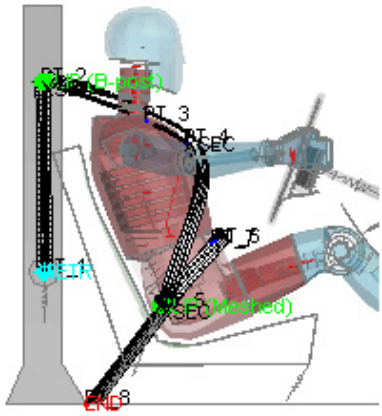
However it is often the case that a dummy needs to be moved, typically when adjusting the seat position, and then remeshed in its new location. Because the basic path points in section (2) above are located at nodes on the dummy and structure PRIMER "knows" when these nodes move, so it can update the basic path shape using their new coordinates. From this revised basic path it can generate a new meshing path and refit this, then using the existing meshing information it can remesh the revised belt path.



Belt fitted to dummy in original position



Seat and dummy have moved forwards, so belt now penetrates the dummy and must be refitted



However if path points are on nodes the fitter "knows" about the movement and the path is adjusted.

The **Auto-Refit** function performs the following tasks:

- Adjusts the fitting path according to the movement of nodes at which path points are defined.
- Refits the belt to the new dummy position using this revised path.
- Deletes the old belt and remeshes it reusing the original properties, part ids, sliprings, retractors, cross-sections etc

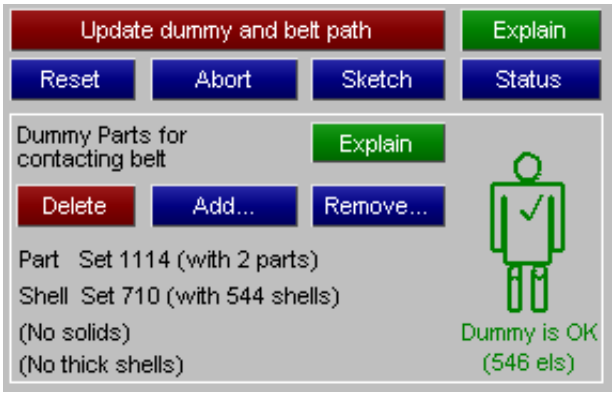
So long as the dummy has not moved such a long way that the original path can now longer fit successfully this process can be performed automatically in a single operation.

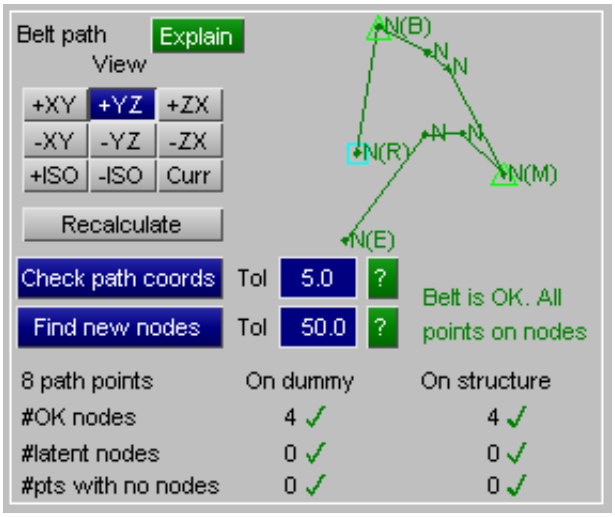
For more information see section 6.40.7 **Auto-Refit**: Refitting a belt automatically when the dummy moves

(T6) Replacing one dummy with another.

Another common operation is to replace dummy A with dummy B, perhaps replacing a male dummy with a female one. The two dummies will have roughly the same shape and organisation, but the details of their meshing will be different and it is unlikely that a node number used to locate a belt path point on dummy A will be in the same place on dummy B.

PRIMER cannot perform this process automatically, but it can make the manual task easier





The first step is to define the part and / or element sets that make up the new dummy structure against which the belt will be fitted.

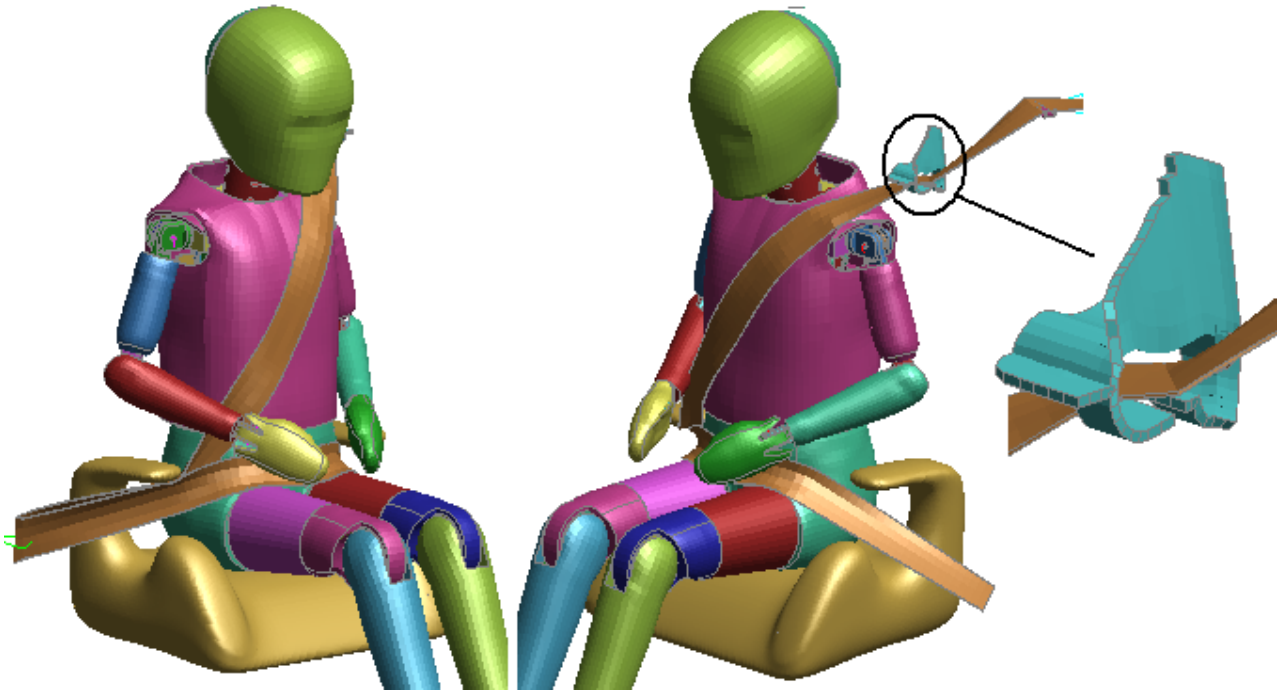
PRIMER will check these (left hand image above) then will attempt to identify nodes on the new dummy that are close to path points of the existing basic path (right hand image). Once a sufficient number of path points have been found **Update dummy and belt path** will replace the old dummy definition with the new one.

For more information see section 6.40.8 **New Dummy**: Replacing one dummy with another

(T7) Fitting belts to more complex shapes.

So far this tutorial has explained the basic process of setting up, fitting and meshing an adult dummy model since this is the simplest case.

However PRIMER can also fit belts to more complex models such as a child dummy in a seat with guides and "wings" through which the belt must pass. The procedure is similar to the above, but the definition of the basic belt path is more complex. The basic path editor has various preset modes, one of which is "child dummy in seat", which helps with this process.



This example shows two views of a typical child dummy in a seat, with the belt guide detail at the shoulder enlarged.

Child dummies such as the example above do not conform to the simple "rubber band around an egg" shape that can be used for adult dummies. The belt path can undergo reverse curvature, it may have to thread through holes and guides, and in the region of the child seat "wings" it may have to navigate very tight geometry. There are several consequences arising from this:

1. Defining a simple path at nodes, projecting it forwards and then pulling it back is not going to work.

   This means that while path points may start of being located at nodes it is likely that they will need to be moved away manually into "thin air", breaking the relationship with the original node. This does not affect manual fitting, but it does mean that "auto refit" after adjusting the dummy's position will not work if the dummy has moved any significant distance.

2. Trying to calculate belt path twist according to nearby structure will tend to give the wrong answer near complex geometry

   The adult belt fitter tries to align the initial belt path "twist" with the nearby structure, but for child dummies this method can be replaced with a simpler approach based purely on the basic path curvature. This does mean that more manual intervention will be required to control the belt twist in critical regions.

Fitting a belt to a child dummy in a seat, and to other complex geometries, is more labour-intensive than fitting to the simple adult case, but PRIMER has various tools which help with the process. The Fitting options panel contains many feature that will help with this process.

(T8) Saving results in the keyword output file.

```
$
*BELT_START
         1Seatbelt definition 1
         0       700         0         0      1113
         4      40.0      15.0       3.0     200.0     150.0       0.0
0.0
        25    1.0E-5         0       1.0       5.0    2.0E-3      25.0
30.0
       0.0      90.0
         2         2         0         0         4         0
         1       1.0       500       5.0      15.0      20.0
$
*BELT_MESH
      1053         0       709  2349.002         1         3
         0         0         1         1         1         1         1
5
      1045      1051
      3000       0.0
```

PRIMER has its own set of keywords, prefaced *BELT, which describe each seatbelt definition in a model. Since these are not LS-DYNA keywords they are written after *END in the relevant file.

They save all the information necessary to define, modify, refit and remesh a belt path. Details of the card formats are given in Appendix V.

Manual versus Automatic fitting

The documentation below describes how to create a belt path, to fit it to the structure, then mesh it and finally to create a contact between belt and dummy. When a new belt definition is created this process needs to be followed in that sequence, and is referred to as "manual" fitting since it is totally interactive.

Once you have a belt definition in a model you can change the geometry of the model and use the **Auto-refit** function to repeat the fit / mesh / contact process above in a single automatic step. This can be performed both interactively and via batch (command-line) commands. This process is commonly used to perform stochastic analyses where the dummy and seat are moved through a range of positions, refitting the belt automatically each time.

It is also possible to combine these two processes: you can adjust the shape of an existing belt path in the path editor or by external means, then use **Auto-refit** to perform the fit / mesh / contact process automatically rather than manually. Some users have created processes that update belt path points as well as modifying structure nodes.

Generally speaking:

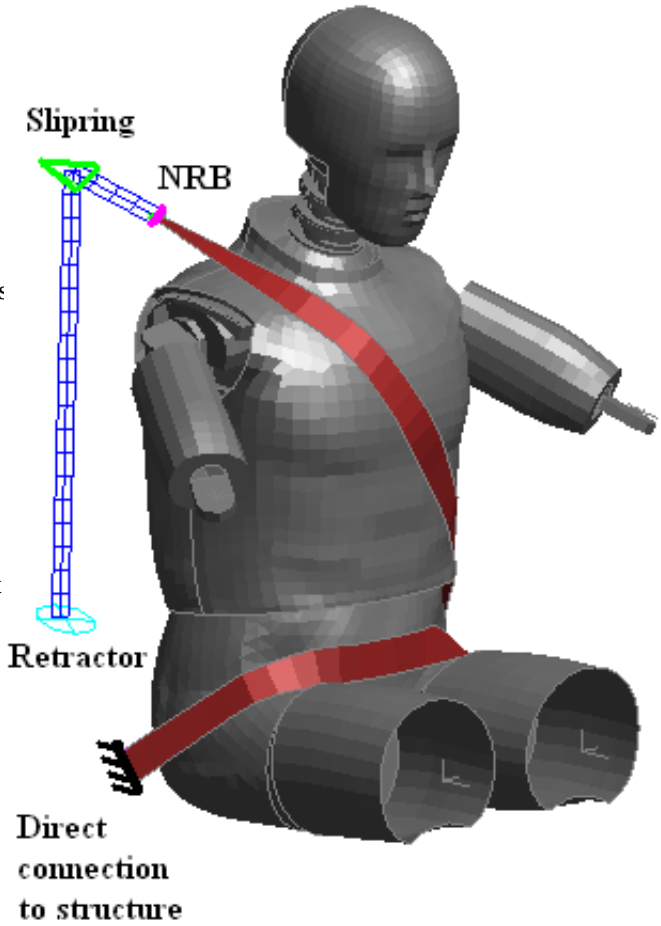
Manual fitting:	• Will delete any existing belt path definitions, sets, contacts, etc • Will create a new set of definitions
Automatic refitting:	• Will preserve as much of the original definition as possible • Will reuse and repopulate existing sets, contacts as much as it can

The choice of processes is up to you, but most users will find that automatic refitting is preferable once a *BELT definition has been created.

6.40.0 Description of Seatbelt Fitting

Traditional 1D Seatbelts + shells method.

A "belt definition" in PRIMER has traditionally been made up out of a combination of **SHELL** and **SEATBELT** element types. Typically **SHELL** elements will be used where the belt touches the dummy, so as to give a realistic area of contact, and **SEATBELT** elements will be attached at the ends of each section so that they can interact with **RETRACTOR** and **SLIPRING** elements.



This figure shows a typical seatbelt definition using a traditional mixture of 1D seatbelt elements and shells

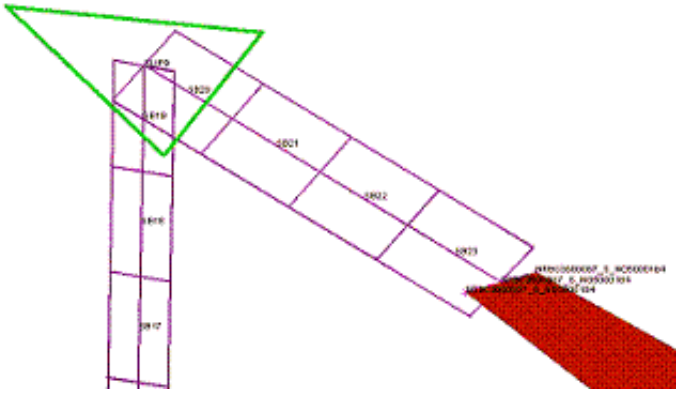
- **SHELL** elements (red) cover the regions where the belt crosses the dummy.
- A **SLIPRING** has been created behind the right shoulder. (There is another one at the left pelvis, not visible here.)
- A **RETRACTOR** has been created at the base of the B-Pillar behind the right pelvis.
- **1D SEATBELT** elements (blue) are used to connect to the slipring at the shoulder, (and at the left pelvis), and also for the whole section from slipring to retractor.
- Nodal rigid bodies (**NRB**) are required to connect the 1D seatbelt elements to the shells.
- At the belt end a direct connection to structure via a NRB if deformable, or extra nodes on rigid part if rigid.

PRIMER would permit the whole belt to be made of **1D SEATBELT** elements, but this would only be suitable for rigid dummies as these elements only give a "line" (zero width) contact with the dummy structure elements.

All stretches of **SHELL** elements are terminated with nodal rigid bodies or, where a shell connects directly to a rigid part, by making the shell end nodes "extra" on that rigid part.

This is to achieve **SHELL** to **SEATBELT** connections, as shown in this figure, and also to stabilise the end element if it connected directly to the structure.

This leads to the end shell being artificially stiff in a transverse direction, but this is acceptable in this context.



Four noded 2D seatbelt shell elements

LS-DYNA release 971 introduced the concept of 4 noded "seatbelt shell" elements which have the special property that they can pass through sliprings and retractors in their "shell" form.

These are implemented in LS-DYNA as follows:

- Despite being entered on a *ELEMENT_SEATBELT card these are in fact SHELL elements, sharing the same label range as conventional shells with whose labels they must not clash. They should use *SECTION_SHELL but *MAT_SEATBELT cards.

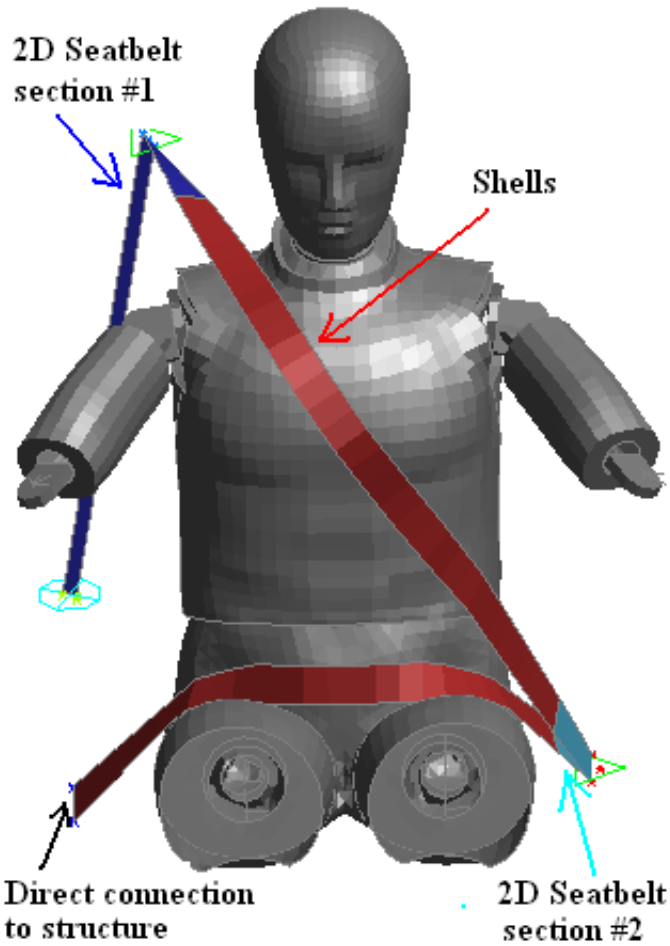
- Sliprings and retractors have to be given information in an alternate form, with careful numbering of node and element sets.

- On input LS-DYNA automatically creates parallel rows of conventional (1D) seatbelt elements, sliprings and retractors which act in the normal way, thus carrying the seatbelt shell elements through themselves as "passengers" on the relevant nodes. The shells are used mainly for contact, although they must also carry a certain amount of force in order to preserve the belt shape.

PRIMER s will create and fit belts using these 2D seatbelt shell elements, and is capable of building a belt definition from them alone or in combination with conventional shells. Mesh continuity through sliprings is preserved, and all the various sets that need to be defined in retractors, sliprings and section definitions are also created.

Getting the node and element numbering right for these 2d belt elements is not easy, and it is recommended that users do not attempt to create or modify the belt mesh manually.

Alternative Shells + 2D Seatbelt shells method.



It is also possible to mesh belts using the 2D Seatbelt Shell elements available in LS-DYNA 971.

This image shows the same example as above with the 1D seatbelt elements replaced with 2D Seatbelt shells, allowing existing shell element properties to be preserved but better geometry to be achieved at slipring and retractor locations.

- Shell elements across the dummy are still used, in exactly the same way as above.
- The seatbelt shell mesh is continuous through sliprings and into retractors.
- The belt mesh is realistic in shape and orientation where it passes through sliprings, especially at the shoulder, making sensible contact between belt, dummy and structure at these points achievable.
- Nodal rigid bodies between Shell and Seatbelt elements are no longer required since the mesh is continuous.
- Mesh ends, whether seatbelt or ordinary shells, which connect directly to structure are still given Nodal Rigid Bodies or, if the structure is rigid, made "extra" nodes on that part.

Note that PRIMER generates new Part, Section and Material properties for each isolated section of 2D Seatbelt shell mesh. (Here blue from retractor to shoulder, light blue at left hand side of pelvis through slipring.)

This is because LS-DYNA requires 2D Seatbelt shell meshes to be continuous and it does not "track across" any intervening stretches of ordinary shells, so two separate definitions are required in this example.

Alternative fully 2D Seatbelt Shell method.

PRIMER is also able to generate a mesh that is wholly made up of 2D Seatbelt shells.

Since 2D seatbelt shells are actually genuine shell elements the result is visually indistinguishable from the mixed shell / 2D seatbelt shell result above, except that only a single part / section / material definition is used because the stretch of elements is continuous.

Alternative fully conventional shell method.

PRIMER can also generate a belt that is wholly made up of conventional shell (ie not 2d belt) elements.

Prior to version 13 this has not been a practical proposition because there was no way of meshing sliprings with these elements, but now that Meshed (radiused) sliprings are available this meshing method may become useful. Retractors and pre-tensioners will have to be meshed by alternative means, perhaps by a short separate belt definition attached to the end of the shells by a nodal rigid body.

The full choice of meshing methods available

The choice of meshing method is up to the user. We anticipate that users with existing 1D seatbelt element + shell belt definitions wishing to achieve better contact behaviour at slipring and retractor locations will convert the 1D belt sections to 2D seatbelt shells, leaving the intervening shell sections unchanged in order to retain consistent behaviour. Users creating new belt definitions may wish to make them entirely from 2D belt elements.

PRIMER V14 onwards permits an arbitrary mixture of element types.

From Version 14 onwards you can mesh each segment of the seatbelt with any permutation of 1d belt and/or 2d belt and/or shell elements.

Prior to V14 combinations of elements in a segment were limited

PRIMER permits existing *BELT definitions previously created in PRIMER to be refitted and/or remeshed in any of the following:

1. 1D seatbelt elements only
2. 2D seatbelt elements only
3. Conventional shells only
4. Mixed 1D seatbelt elements + shells
5. Mixed 2D seatbelt elements + shells

Options (1) is unlikely to be used in practice, but is provided for completeness.

Contact between belt and dummy.

In modern modelling practice it is likely that the belt will be included in a larger contact surface that includes dummy, vehicle, seat, steering wheel and - possibly - airbag.

However PRIMER can generate a separate contact between belt and dummy if required.

Two possible contact surfaces are offered

`AUTOMATIC_SURFACE_TO_SURFACE` For contact between belt shell elements, 2d seatbelt shells and the underlying structure.

`AUTOMATIC_NODES_TO_SURFACE` For contact between 1D seatbelt element nodes and the underlying structure.

Either or both may be omitted if unnecessary, and all parameters may be altered as required.

In PRIMER the various fields on the contact all have their default values, with the exception of the "soft" option which is set to 1. Experience has shown that the relative stiffnesses of belt and dummy may be very different, and the use of this "soft constraint" option can improve contact behaviour.

Refitting a belt when the dummy moves.

If the dummy on which the belt has been fitted is subsequently moved, perhaps by Dummy or Mechanism positioning, then the belt can be automatically refitted to the new dummy position using the **AUTO_REFIT** function. In most cases no user intervention is required and the belt, in its current form, is remeshed to fit the new dummy position.

It can also be refitted manually using the conventional **FIT** process below.

In both cases any basic belt path points that are located at nodes will "know" about movement of those nodes, enabling the basic path itself to move in space so that it remains in much the same position relative to the repositioned dummy. Path points that are not at nodes will have their movement interpolated from adjacent points, so they too will move. However this only works if the number of path points not at nodes is not excessive, otherwise there is no information from which to interpolate.

In practice this means that belts can usually be refitted to adult dummies, where the "locate point at node and project it forwards" approach works well, but not on child-in-seat dummies where points usually have to be moved to locations in space in order to thread the path through obstacles.

Replacing one dummy with another

It is often the case that dummies need to be swapped around, for example replacing a 50th percentile dummy with a 95th percentile one.

To save having to delete the old belt definition and creating a new one from scratch PRIMER has a **New Dummy** capability. The process is not automatic, but it provides assistance with remapping the old belt path onto the new dummy, preserving as much information as possible.

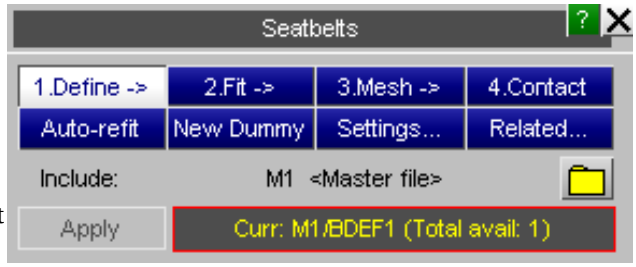
The process works by looking for "nearest nodes" on the new dummy which match existing belt path points, so as long as the dummies are broadly similar in shape, and not wildly different in size, this process should be reasonably helpful. However if there are gross differences in size or shape it is likely that restarting the belt definition from scratch will be more efficient.

Saving belt definitions: the *BELT keyword (after *END)

PRIMER permits any number of belt definitions to be defined in a model, (cf: Dummies and Airbag Folding definitions), and these are written out as a **`*BELT`** block of keywords after the **`*END`** card of an LS-DYNA deck so that they are not lost between PRIMER runs. This also means that if a dummy is moved it is possible to go back to the seatbelt definition and to refit the belt to the new dummy position.

These data are added automatically to your model when you create a belt fitting definition. Details of the card formats used are given in Appendix V.

6.40.1 1.Define: Defining "Structure" for seatbelt fitting



Belt elements are fitted directly onto "structure" elements, (it is not necessary to have predefined contact surfaces), so clearly you must have some structure to which to fit your belt definition!

In PRIMER this may be any combination of **`SHELL`**, **`SOLID`** and **`THICK SHELL`** elements., the only limitation being that they must be in a single model. (If you have separate models, for instance #1 is a car and #2 a dummy, you will have to merge them before fitting a seatbelt.)

Before fitting can take place you must **SELECT** an existing belt definition (implicitly containing structure) or **CREATE** a new one. Selecting **Define** from the main seatbelt menu as shown on the right brings up the menu shown below.

CREATE Manages the creation of a new seatbelt definition.

DELETE Deletes existing definitions.

LIST Lists existing definitions and their contents.

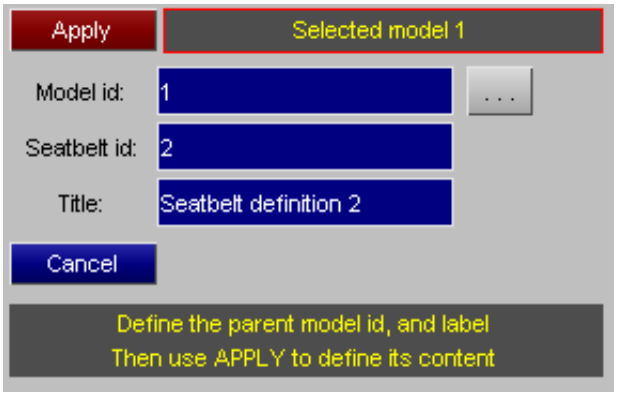
SELECT Selects an existing definition and makes it the current one.

ADD Edits the current definition by adding new elements.

REMOVE Edits it by removing elements.

DONE Exits this panel to return to the main seatbelt menu.

CREATE Creating a new belt definition.



You must first select the model in which the new definition will reside.

Then you must give a label and title for this new definition, and press **APPLY**.

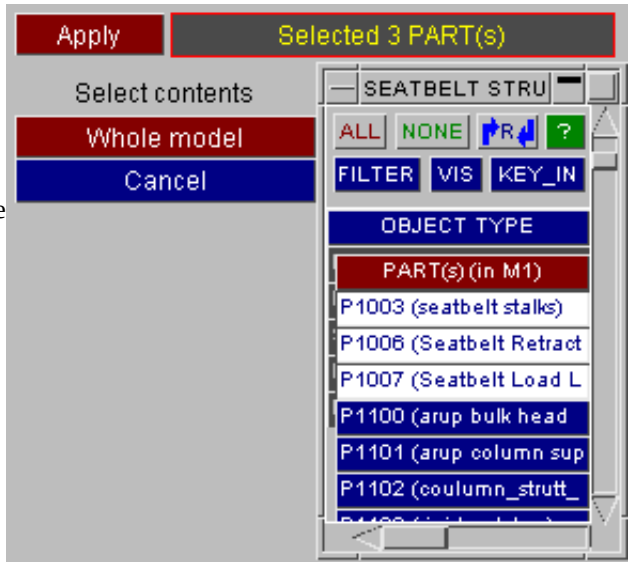
Labels are arbitrary, but must be unique within a model.

Then select the **PARTS** and/or **SHELL**, **SOLID** and **THICK SHELL** elements that will constitute the structure for this definition.

PRIMER will create sets of these element types (**SET_SHELL**, etc) when you press **APPLY**.

Then **SELECT** this new definition to make it current, and use **DONE** to return to the top seatbelt menu.

The most efficient, and usually also the most convenient, method is to select whole PARTs. This means that the belt fitter can amalgamate the "structure" definition into a ***SET_PART** definition, which is convenient for contact and other definitions. However you can choose any mixture of individual elements, including subsets of a part if you wish.



Efficient selection of structure elements

It is tempting, and quite legal, to select the whole model here; but you should consider the following:

1. These sets are used during the iterative belt form-finding process to create the pseudo-contacts against which the belt elements are fitted. The time taken in this process rises a linear function of the number of elements, so it is wasteful to select elements which will never be needed: for instance internal structure and elements on the rear facing sides.

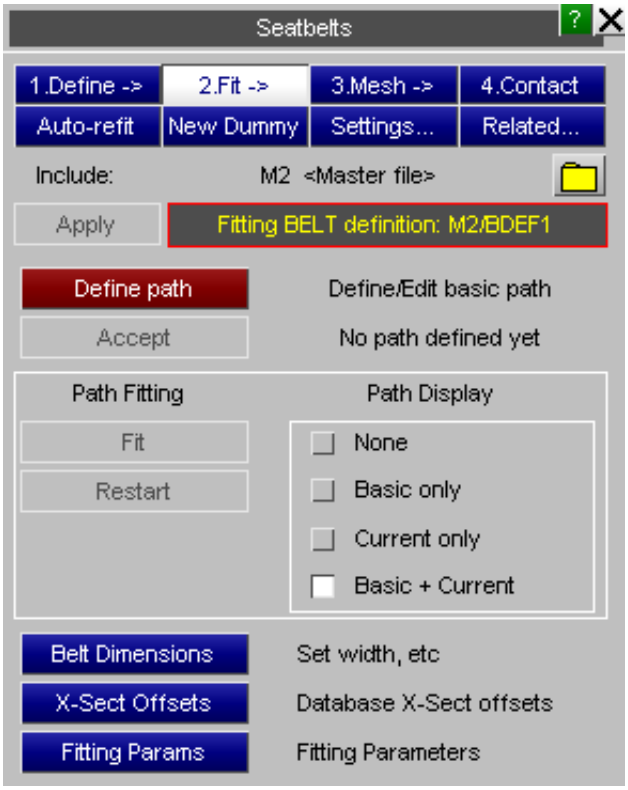
2. These sets may also be used to define the master side of the belt-to-dummy contacts for the actual analysis, so the same efficiency and speed considerations apply there.

3. Many dummies are "coated" with a layer of null shells in order to give a continuous contact surface between assemblies that will move during the analysis. Sometimes these shells are given very small thickness values, for example much less than 1mm, and if these shells are included in the dummy structure then belt fitting may be slow as the "quantum" of movement between successive iterations will be very small in order to avoid penetrating through to the wrong side of these elements. Try to avoid selecting such shells, or if they are selected make sure that their thickness has a sensible value (at least the same thickness as the belt elements themselves).

4. Dummies tend to have quite complex internal structure, and it can sometimes confuse the belt-fitting contact algorithms (especially automatic depenetration) if they have to try to distinguish between "surface" and "internal" elements very close to the outer surface. It is best to try to define only those elements that make up the outer surface against which the belt will be fitted.

The best structure definition is one which includes only those elements actually required for the fitting process and contact during subsequent analysis, and it is worth taking some trouble to achieve this - especially with large and complex dummies.

6.40.2 **2.Fit:** Creating a belt "path"



The second stage of seatbelt definition requires you to "fit" the belt by:

- defining a crude "path" for the belt;
- assigning geometric properties to that path (width, thickness, etc)
- assisting PRIMER in fitting the path to the structure using a form-finding process.

And hence obtaining an accurate geometry for the actual belt which can then be meshed. The main **FITTING** panel is shown here in its initial state with no path defined. To define one the **Define Path** command is used.

Entering "**2.Fit**" also maps the Fitting Options panel.

The options on here will be described in more detail under **Fitting Options** below.

What is a belt "path"?

A "path" is a connected set of 2 or more points which define an initial, crude line for the belt.

Each point is simply an [x,y,z] location in space, and has no structural significance: it is used solely to define an initial shape for the belt path. It is normal practice to locate path points at nodes, whereupon the points use the coordinates of those nodes, however points may also be located in "thin air" and not at a nodal location. A point may also be located initially at a node, then have its coordinate updated manually, which "breaks" its association with the node. Locating points at nodes has advantages when dummies are repositioned, since the path point coordinate is updated to the new nodal position. This is covered in more detail in **Auto-Refit** below.

Paths have the following rules:

- A path may contain any number of points, and by default all points except the two ends are free to move. Any

intermediate point may be fixed, meaning form-finding won't move it.

- A path consists of one or more segments. A segment is a path section of at least 2 points, and is demarcated by its end points being **`Fixed`**, **`Acute`**, **`Slipring`** or **`End`** points. Each segment is treated separately for both form-finding and subsequent meshing.
- If two adjacent points are fixed the path segment between them is assumed to be straight and will not move during fitting.
- At each unfixed point the adjacent lines must form an angle of > 90 degrees. If an acute angle is found the point will be designated automatically as **`Acute`**, forming a break between the segments on either side.

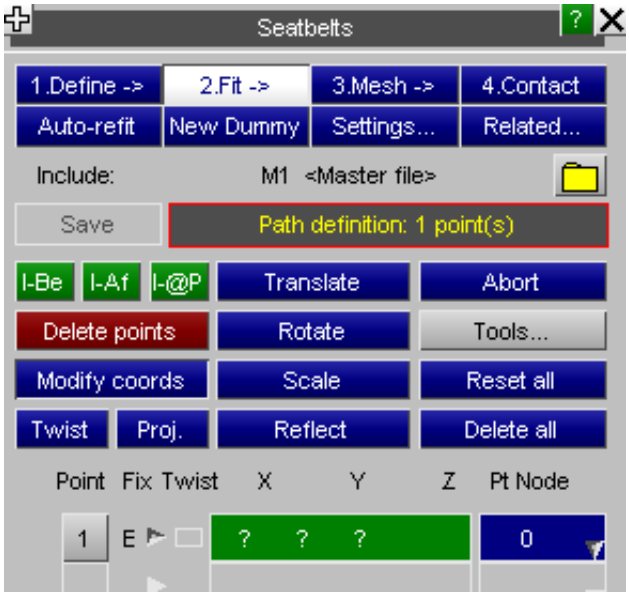
Path points may be have one or more of the following attributes:

<table>
<tr><td colspan="3">The following attributes may be defined by the user. (2nd column shows suffix used in plots)</td></tr>
<tr><td>Known</td><td>K</td><td>The belt is "known" to pass through this point, and this acts as a constraint upon form-finding which will make the centreline of the pass go through the point.</td></tr>
<tr><td>Fixed</td><td>F</td><td>A designated "fixed" point that demarcates the ends of any attached segments. The belt path is discontinuous at this point.</td></tr>
<tr><td>Acute</td><td>A</td><td>If an otherwise free intermediate path point has an acute angle (< 90 degrees) between adjacent segments then PRIMER automatically defines the point as "Acute". This implicitly fixes it and creates a break between the adjacent segments.</td></tr>
<tr><td>End</td><td>E</td><td>Both ends of the path unless a Retractor has been defined. "End" points are implicitly fixed.</td></tr>
<tr><td>Retractor</td><td>R</td><td>A retractor will be created at this point. This is only legal at path end points, and the presence of a retractor supersedes the automatic "End" definition.</td></tr>
<tr><td>Slipring</td><td></td><td>A slipring will be created at this point. This is only legal at intermediate path points, and a slipring implicitly "fixes" the point and supersedes any "Fixed" or "Acute" definition. PRIMER V13 onwards has three types of slipring available:
<table>
<tr><th>Type</th><th>Suffix</th><th>Description</th></tr>
<tr><td>Free</td><td>S</td><td>No constraint on orientation, which defaults to the average of the two path segments meeting at that point.</td></tr>
<tr><td>B-Post</td><td>B</td><td>Presumes that one side will be a straight stretch attached directly to a retractor, and constrains rotation to be about an axis normal to the notional B-Post</td></tr>
<tr><td>Meshed. explicitly meshed radius</td><td>M</td><td>As a free slipring, but instead of using *ELEMENT_SEATBELT_SLIPRING it meshes the geometry explicitly around a user-defined radius using short elements.</td></tr>
</table></td></tr>
<tr><td>Cross-section</td><td>X</td><td>A *DATABASE_CROSS_SECTION definition will be created at this point. This is legal at any point on the path including at sliprings, retractors and ends. In the case of intermediate fixed points, typically sliprings, two sections will be created: one on each side of the point.</td></tr>
<tr><td>Projection</td><td>P</td><td>The point has been assigned a non-default projection down the "outwards" vector. See Control Projection mode below.</td></tr>
<tr><td colspan="3">The following attributes are assigned automatically by PRIMER unless a user-defined attribute from the list above supersedes them.</td></tr>
<tr><td>Unfixed</td><td>U</td><td>This is the default for intermediate path points with obtuse angles, which are free to move during form-finding.</td></tr>
<tr><td>Acute</td><td>X</td><td>If an otherwise free intermediate path point has an acute angle (< 90 degrees) between adjacent segments then PRIMER automatically defines the point as "Acute". This implicitly fixes it and creates a break between the adjacent segments.</td></tr>
</table>

End	**E**	Both ends of the path unless a Retractor has been defined. "End" points are implicitly fixed.

Creating a belt path

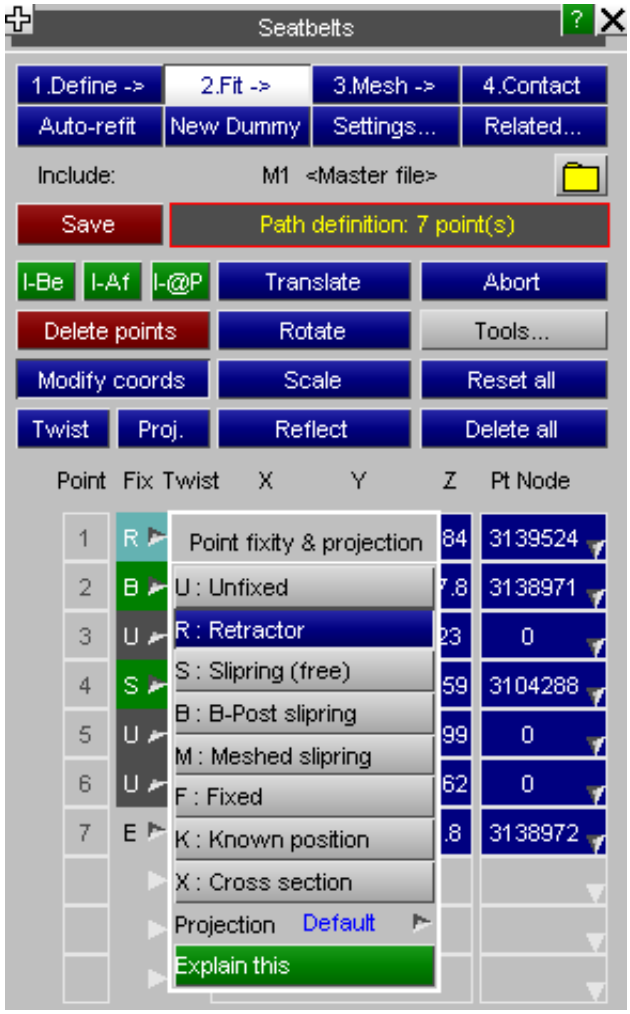
Belt paths are created in a simple text and visual editor, invoked by the **Define path** command in the **FITTING** menu.



The panel above shows the path editor in its initial, empty state before any path has been defined.

By far the easiest way to create a path is to click on nodes on the dummy and structure. The coordinates of the nodes are abstracted and become the path points.

Here some points have been defined, and the user has mapped the Point Fixity popup menu, which allows points to be set to one of:

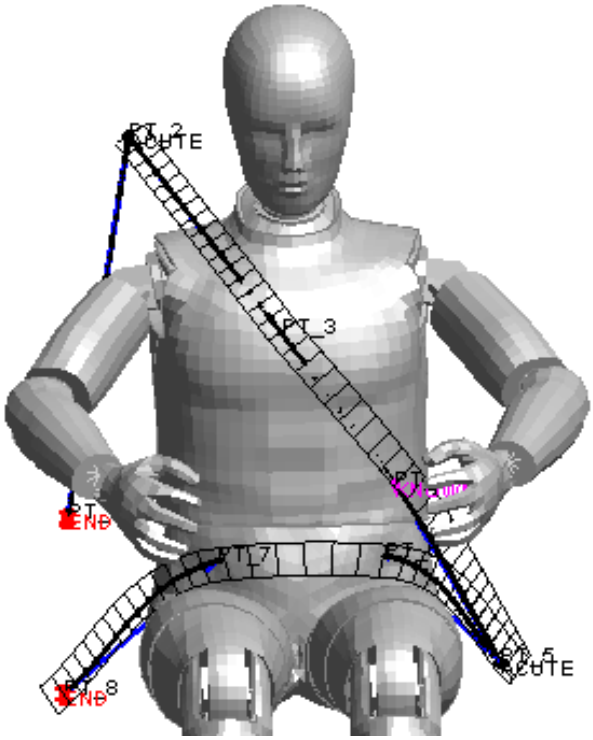


U : Unfixed
R : Retractor
S : Slipring (free)
B: B-Post slipring
M: Meshed slipring
F : Fixed
K : Known position

X : Cross-section is an additional attribute which can be added at any point.

Projection controls the outwards projection of individual base path points when creating the initial belt fitting path.

Defining the initial path



This figure shows a typical belt path round a dummy as initially defined. It has three segments:

1. Retractor to upper right shoulder slipring.
2. Upper right shoulder to left pelvis slipring.
3. Left pelvis slipring to anchor point behind right pelvis.

Note that:

- Only 8 points have been defined. As a general rule the best paths are those with the fewest points, and point 4 in this definition could be omitted . However the user has designated it as a "known" point.
- PRIMER has marked the "End" and "Acute" points automatically, and has inserted breaks into the path at the two acute points where sliprings will be defined later.

The "as projected" belt path is always shown at the basic path creation stage. Path projection takes place for any path segment with 3 or more points, and is in the outward normal direction determined from nearby structure or intrinsic path curvature, whichever is more relevant at that point.

The projected belt path also shows the approximate mesh that will be created. In this example only a single line of elements has been selected, but this can be altered at any stage.

Editor modes

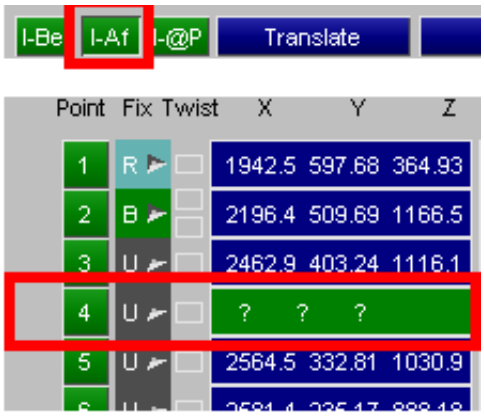
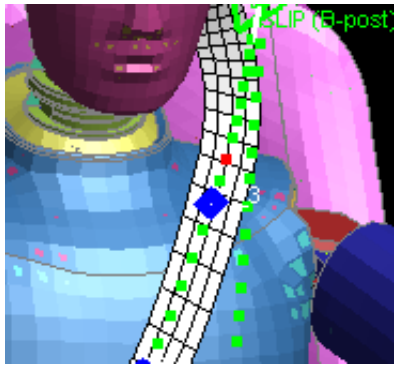
The editor allows you to type in coordinates for points at any time: just click on the relevant box and (over-)type the coordinate value. But this is seldom convenient, and it is usually much easier to define coordinates by picking nodes, and to adjust them by dragging them with the mouse.

The following table lists the colours used both in the path editor and for the graphics in the various path editing modes.

Mode	Path Colour	Cursor	Label buttons
INSERT	Green	Picks nodes for point coords,or "at" an intermediate point.	Choose point to insert before / after
MODIFY	Blue	Left: Modifies X coord Middle: " Y " Right: " Z "	<Not active>
CONTROL TWIST	Cyan	Left on point pick handle twists path Middle on pick handle skews the path (fixed points only)	<Not active>
CONTROL PROJECTION	Cyan	Left on point pick handle moves path along outwards vector Middle on pick handle resets that path point to default.	<Not active>
DELETE	Red	Picks points to delete	Choose point to delete

INSERT mode

This mode is used to create new points, and works in two ways:

I-Be or **I-Af**	Creates new points between existing path points either by screen-picking nodes or by typing in explicit coordinates. **I-Be** inserts a new point ***Before*** the point selected in the path editor. **I-Af** inserts a new point ***After*** the point selected in the path editor. You can insert any number of points, insertion being terminated by changing to a different editor mode or exiting the editor. This mode is usually best for defining the initial shape of the path by picking existing nodes. The coordinates of the picked node are used, and the node itself is "remembered" for future reference during meshing or refitting. In this example the user has selected **I-Af** and then point 3, meaning that he creating a new point 4 after point 3.  When picking points at nodes it doesn't matter that nodal coordinates lie on the neutral axis locations of shells: the path is projected outwards prior to fitting, then pulled back to the structure surface by the form-finding operation..
I-@P	Creates a new path point ***At*** an existing intermediate point. Once you have defined at least two points the current path will be drawn on the screen as either a straight line or a spline fitted between the basic points, split into smaller sections as defined by the belt element length. If you wish to create a new basic path point at an intermdiate position somewhere on this path you can use **I-@P** to do this: • Every intermediate point (at element length spacing) is shown as a green dot. • Hovering the mouse near a dot will change it to red to signify which will be chosen. • Clicking on the dot will create a new basic path point at that location.  This image shows that the basic path points are drawn as blue diamonds, and you will see the row of green dots that show the interpolated belt path. Here the mouse has been hovered over a point roughly in the centre of the image. Points created this way will not be associated with nodes, they are defined solely by their coordinates.

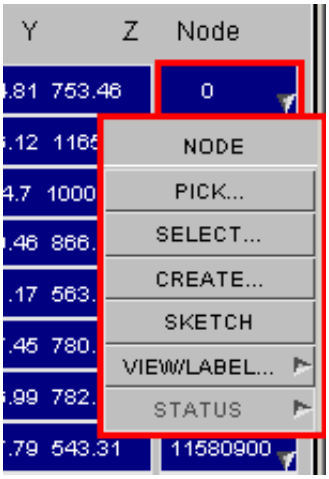
MODIFY mode

This mode permits point coordinates to be adjusted. You can do this in any permutation of the following three ways:

1. Use the popup menu in the "Node" column in the path editor to screen-pick or select a new node.

   This is the preferred method since the point will still have a node associated with it, which is valuable if a retractor or slipring are defined at that point since it determines clearly which (structural) node the element should be associated with. (Field SBRNID). It is also valuable at path end points where the belt is attached directly to structure since, again, it defines which node should be used.

   This is still true in the case of 2D belt meshes, as although sliprings, retractors and belt ends will all have at least two nodes nevertheless they can still be associated with the specific nodal location via a nodal rigid body or extra nodes on the parent rigid part.



1. Over-type the point coordinates with new values. The new coordinates will be applied immediately, but note that the associativity between the point and the node originally used to pick it will be lost.

2. Drag points visually. Click on a path point with the relevant button (see below) and, holding that button down, drag the point to its new location.

Left button	:	Drags in global	X
Middle button	:	Drags in global	Y
Right button	:	Drags in global	Z

As with method 2 the association between point and its original definition node will be lost once its coordinate is updated.

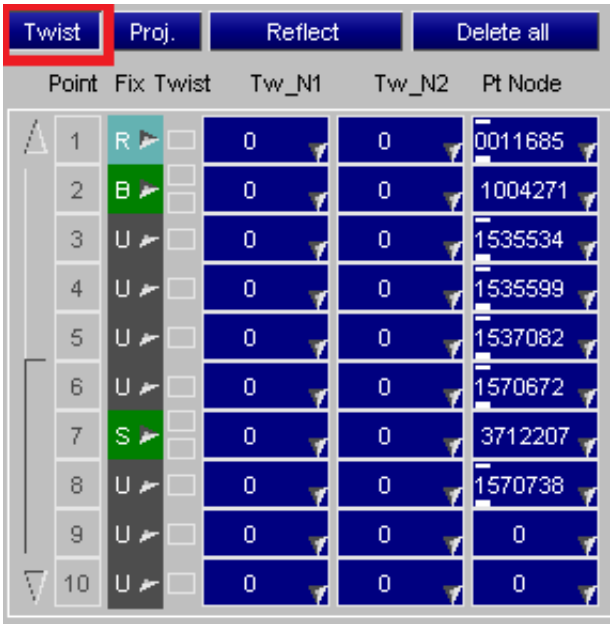
In all cases both the visible path and the relevant coordinate in the editor will be updated dynamically so that you can see what is happening.

Control Twist mode

The orientation of the belt path, its "twist", is chosen initially by PRIMER to give what it believes to be the best fit. This is computed in two possible ways:

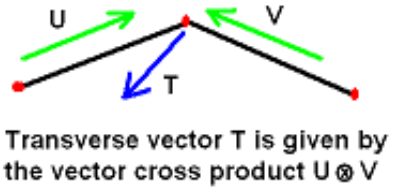
(1) "Local Normal" method.

- The outward normal of nearby structure elements is used at intermediate points, and at end points the most recent normal is extrapolated to give a consistent end section.
- If no nearby structure elements are found then the path is treated as lying approximately on the surface of a sphere with its centre at the average (central) coordinate of the "structure" nodes.
- Special rules apply to straight sections between two fixed points, where a "best guess" at the wanted path is adopted.



(2) "Path Twist" method

- The twist is based on the natural curvature of the path. In mathematical terms the transverse vector (the plane in which the belt elements lie) is given by the cross-product of the two vectors meeting at a point. Local structure is ignored.



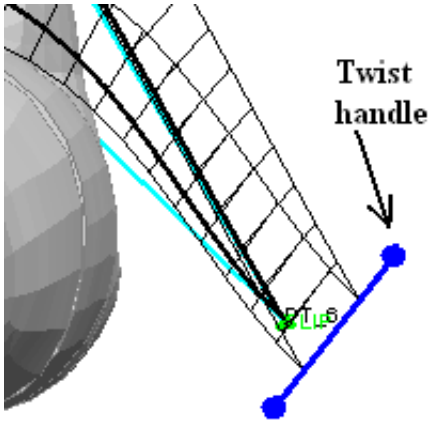
The method used to define the default twist is controlled in the **Fitting Options** panel.

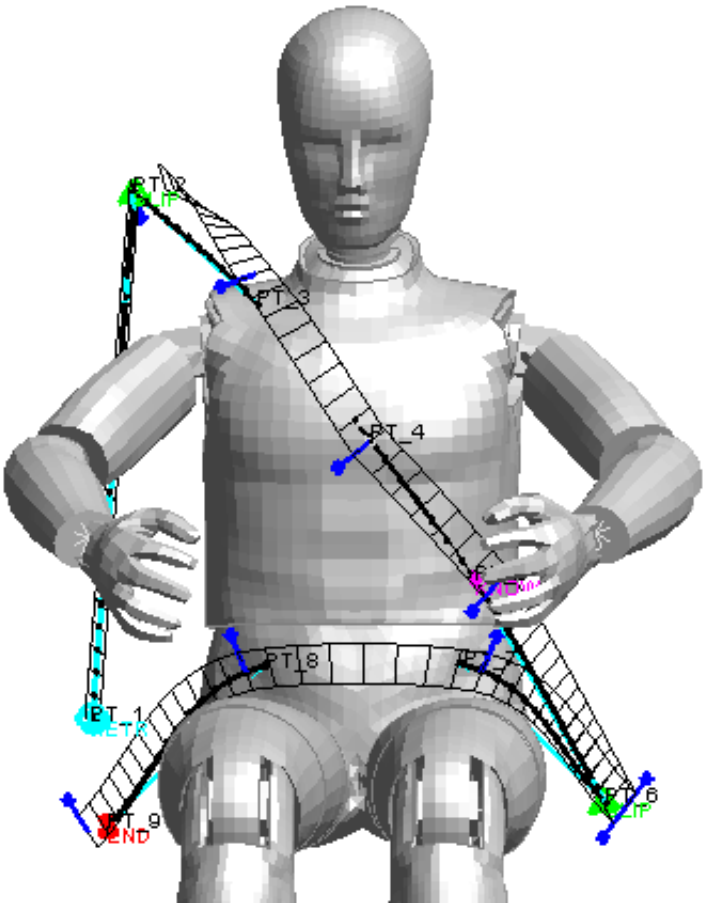
However in many cases these defaults are inadequate and it is necessary to adjust the twist of the belt path.

Twisting a point manually using the mouse

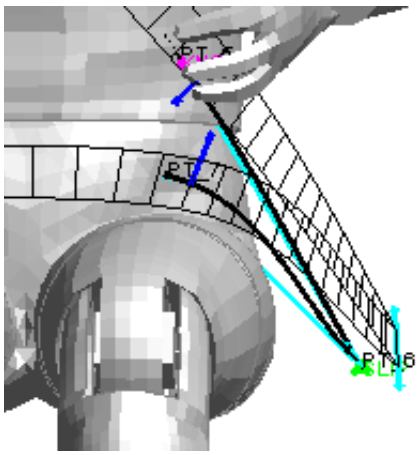
In "Control twist" mode each path point has a "twist handle" attached to it, and by clicking and dragging on that handle you can adjust the twist of the path at that point. The axis that is twisted depends on the mouse button used:

- The **Left** mouse button alters the **radial** (outwards) vector
- The **Middle** mouse button alters the **transverse** (rotation about radial) vector.
- (The right mouse button is not used in this context)

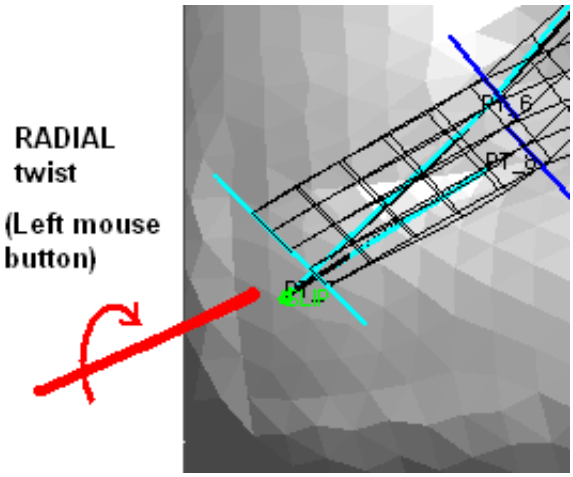




Here the path at the lower (pelvis) slipring has been rotated to
a more realistic angle by rotating its radial vector:

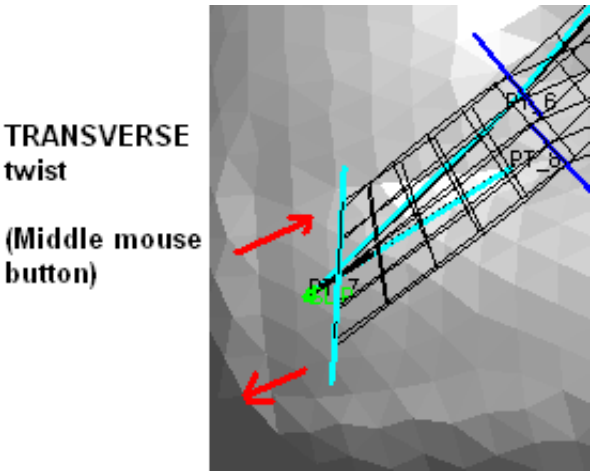


RADIAL twist, imposed by the left mouse button, changes the direction of the radial (outward) vector, effectively twisting it about the long axis of the belt at that point:



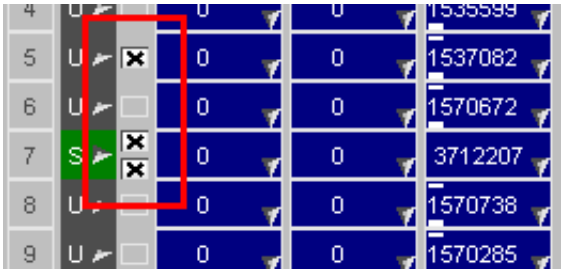
Radial twist can be applied at any path point.

TRANSVERSE twist, imposed by the middle mouse button, rotates the transverse mesh lines about the radial (outward) vector at that point.



Transverse twist can only be applied at end points, retractor and slipring locations, and fixed points.

Reporting and cancelling twist at a point.



Once a point has been twisted away from its "natural" orientation PRIMER remembers this and shows it in two ways:

1. The colour of the twist handle on the plot changes from dark blue to light blue (compare the lower and upper images above).

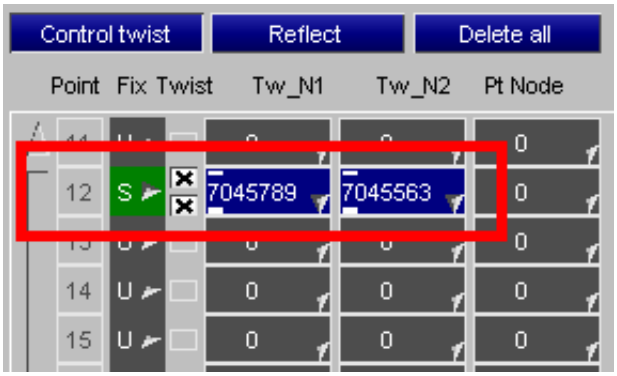
2. In the path editor a cross [X] is shown in the "Twist" column, see points 5 (intermediate) and 7 (slipring) in the image on the right.

The twist can be adjusted further at any time by dragging the handle to a new position.

To cancel twist at a point, resetting it to its default orientation, click on the [X] symbol in the path editor.

Fixed points and acute points will have two [X] buttons, see point #7 in this figure, and also two twist handles on the plot. This is because the path twist on either side is independently controllable at these points.

Twisting a point using 2 nodes to define a vector

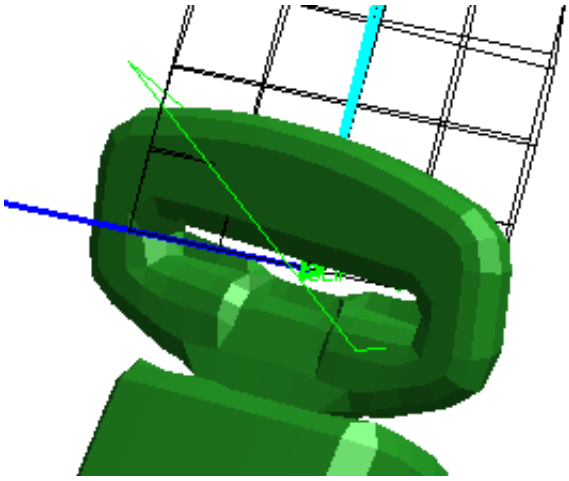


As an alternative to dragging the twist handles using the mouse, which is an interactive-only process, you can define the twist vector at a point by defining two "twist nodes", which define a vector N1N2. This has two advantages:

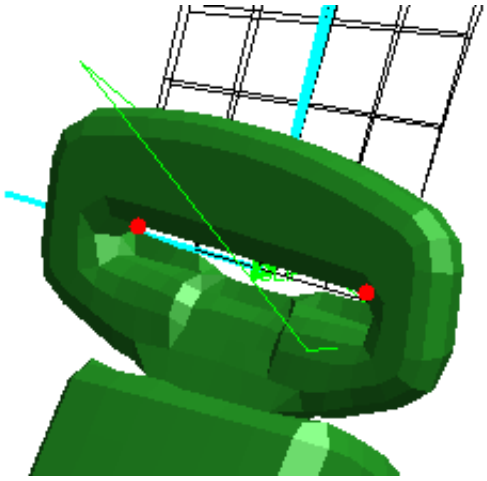
1. Where you are trying to align the belt with an existing piece of mesh, typically at a slipring buckle, you can simply pick two nodes and the path alignment will be done for you, avoiding the need to line things up "by eye".

2. The process need not be interactive, and therefore is suitable for batch fitting. The coordinates of the nodes at the time of fitting will be used, so a script could modify the twist node coordinates if required between repeated fitting operations on an updated model.

The following example illustrates how two twist nodes might be used to line up the belt path correctly at a slipring buckle prior to fitting a 2d belt.



Here it can be seen that the belt path is centred correctly in the buckle, but that its mesh is skewed.



Here is the result of setting the two red nodes shown as "twist nodes: it is now correctly aligned.

Note that the twist nodes do not locate the path point in space in any way, but just provide an orientation vector.

The following rules apply to twist nodes:

1. Both nodes must be defined in order for the vector to be calculated and applied. If only one node is present then no action will be taken.

2. The twist nodes do ***not*** define the path point position in any way, this is defined either by a position node **`nid`**, or by an explicit coordinate. The two twist nodes can be located anywhere in space (although they will usually be near the belt path) as they simply define an orientation vector. In addition the order in which the twist nodes is defined is not important as the direction of the vector is reversed automatically if required.

3. The way the vector is applied varies slightly depending upon whether the path point is a fixed end or an intermediate point.

   At fixed ends, including sliprings, the N1N2 vector is applied directly to become the path orientation at that point, permitting the belt path to be both twisted and skewed. This is equivalent to twisting the belt using a combination of both left mouse and middle mouse motions above.

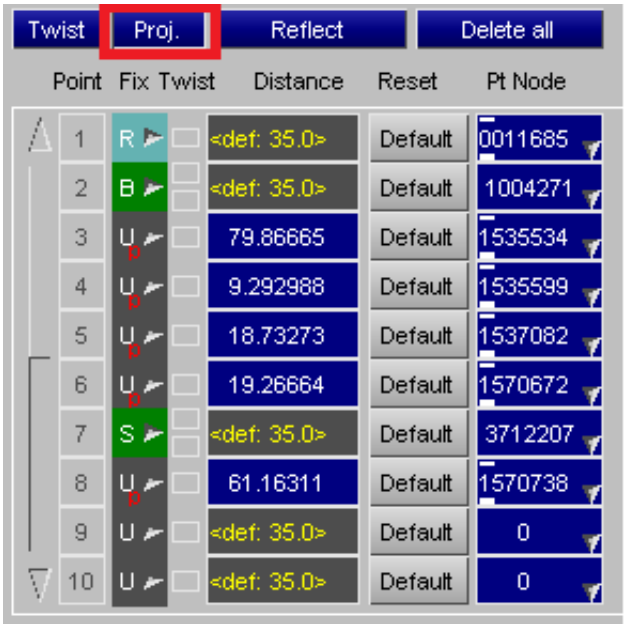
   At intermediate points only the belt twist is updated with no skew being applied, ie the transverse mesh lines remain approximately at right angles to the belt path and the N1N2 orientation vector just defines the plane in which the belt path lies. This is equivalent to twisting the belt using the left mouse only.

4. Cancelling belt twist using the **`[x]`** button(s) in the belt path editor cancels belt twist, and hence also removes any twist nodes at a point. Similarly applying a manual twist using left or middle mouse buttons supersedes any twist applied using twist nodes, and also deletes them.

5. Twist nodes are "remembered" in the **`*BELT`** cards written after **`*END`** (see Appendix V). Since the coordinates of the nodes ***at the time of fitting*** are used to (re)compute twist vectors this provides a mechanism for defining model-specific belt twist in different models.

6. Twist nodes may be used at a slipring position, in which case they will determine the orientation of the slipring. If the slipring has been defined as being of the "B-Post" type then there is a potential conflict between the orientation as defined by the special B-Post logic and that defined by the N1N2 vector, in this situation the N1N2 vector "wins" and the B-Post logic is ignored, effectively treating the slipring as being "free".

Control Projection mode



The way a belt is normally fitted is that a basic (line) path is defined using points located at nodes on the structure, these are projected outwards to give some clearance, and an initial 2d belt path is calculated .

By default all points are projected out by the same amount, in this example the default is 35mm but this can be changed using Point offsets in the Fitting Options panel, and this is generally successful for adult dummies. However when fitting to more complicated geometries it can be necessary to adjust the locations of individual points in order to fit the path through holes and around obstacles.

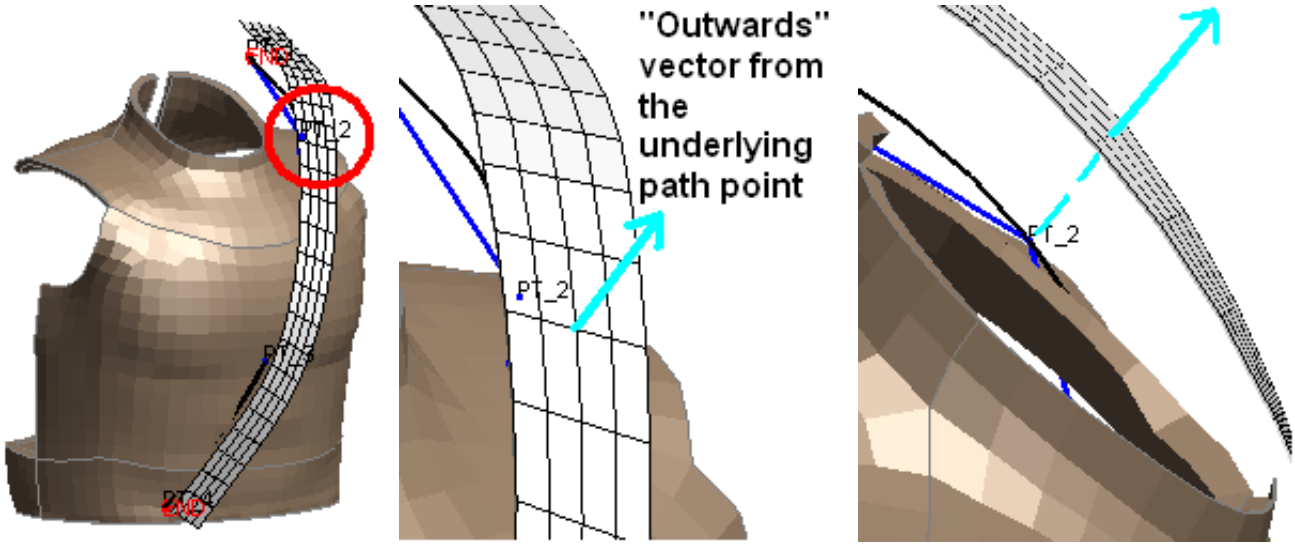
It is possible to change a point's position using the **Modify** mode above, but the disadvantages of that approach are two-fold:

1. Moving a path point away from its original node "breaks" the association with that node, and if remeshing is to be performed when the dummy has been moved this can cause problems since the belt fitter may not know how to move that path point if it no longer knows about its underlying node. (It can interpolate movement from adjacent points defined by nodes, but this becomes increasingly unsatisfactory as these become more distant.)

2. Modifying a path point involves either typing in new coordinates, or using mouse buttons to drag the point down some combination of global X, Y and Z vectors, and it is often cumbersome to achieve the desired position by this method.

Control Projection achieves the same result by different means. Instead of changing the coordinates of the basic path point instead the distance by which this point is projected outwards to create the initial 2d fitting path is modified.

This retains the original node if defined, and it is generally easier to perform since the "Outwards" vector at a path point is usually the direction in which you need to adjust the position, so a single mouse-driven dragging operation will usually suffice.



Typical chest section of dummy

Close-up of point #2 at shoulder showing the outward vector

View of the same point from the side

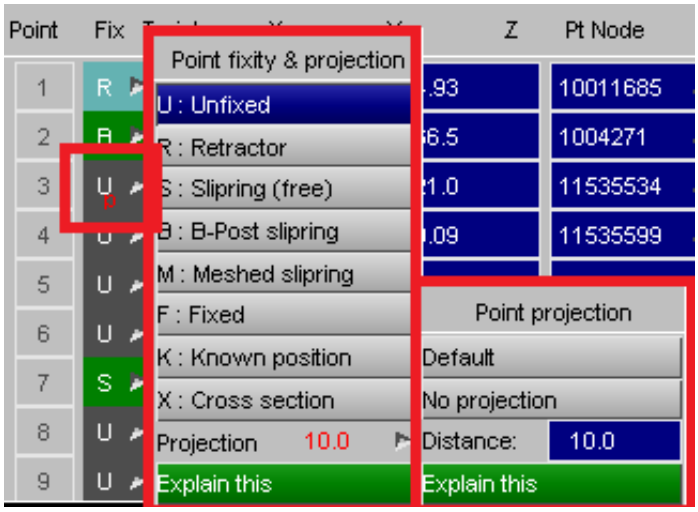
In control projection mode the mouse buttons work as follows:

- Left button drags the belt path along the outwards vector
- Middle button restores projection at the point to the default value.

You can also type a projection distance directly into each point's button, or restore it to the default projection.

In addition in all editor modes the popup menu for each point not only shows whether or not a point has a non-default projection set by add ing a red "P" symbol, but also allows you to set and restore this directly.

In this example point #3 has been given a non-default projection of 10mm.



Control of point projection is new in PRIMER V14, and the projection distance is stored in the *BELT cards written after *END. Therefore if point projection is defined at any point the resulting keyout file will not read into an earlier version of PRIMER. If it is necessary to export such a deck to an earlier PRIMER version it will be necessary to set the following preference to the appropriate version:

`primer*mdumm_keyout_format: V13 | V12 | V11`

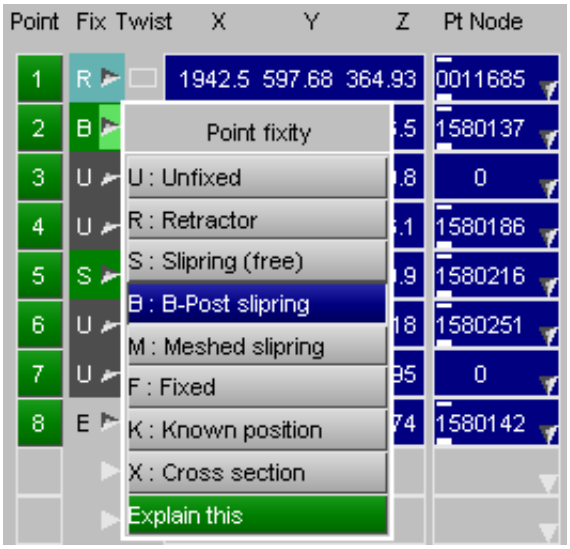
DELETE mode

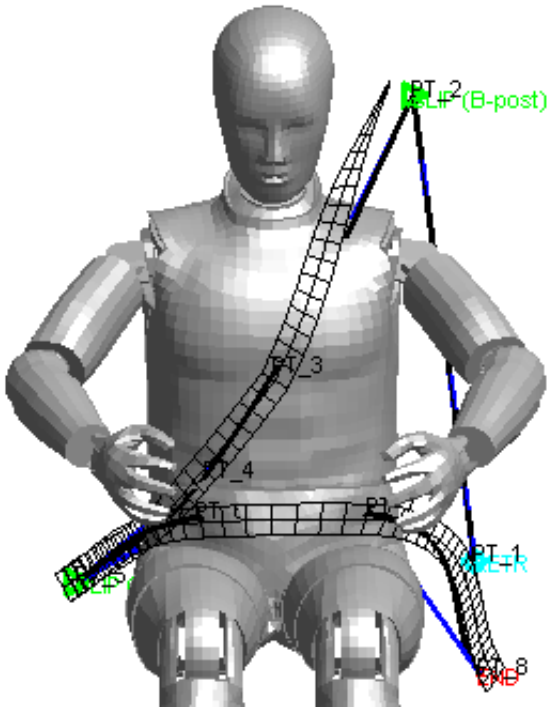
This mode lets you delete points. Either press the label button of the point to be deleted, or click on the point with the mouse (left button). With either method it is deleted immediately.

Adding detail to the basic path

Adding sliprings and retractors

Using the popup menus against the "Fix" column in the path editor:





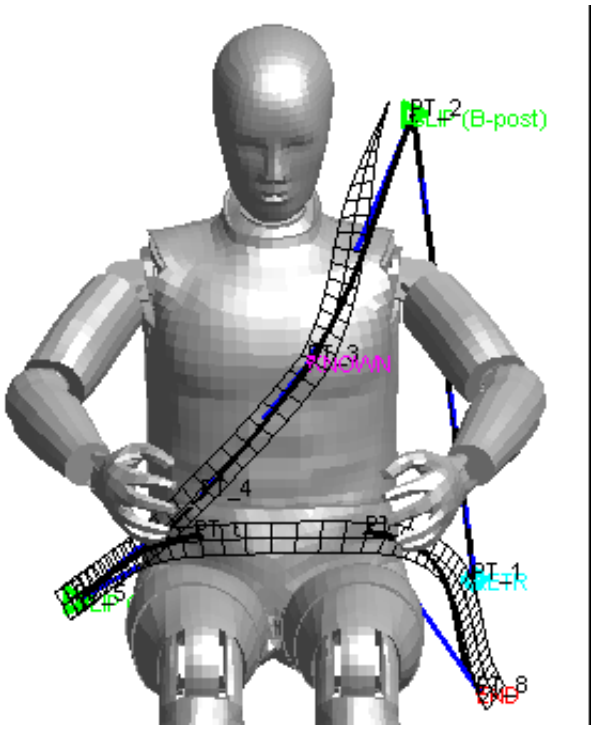
The user has now added:

- A Retractor at point one
- A B-Post Slipring at point 2
- A Free Slipring at point 5

Adding "Known" points

In this example the user knows exactly where the point will pass across the dummy's upper chest, so point #3 has been moved to this known position and marked as "known".

During fitting this will force the path to pass through this point, even though it is not on the "natural" shape of the belt.

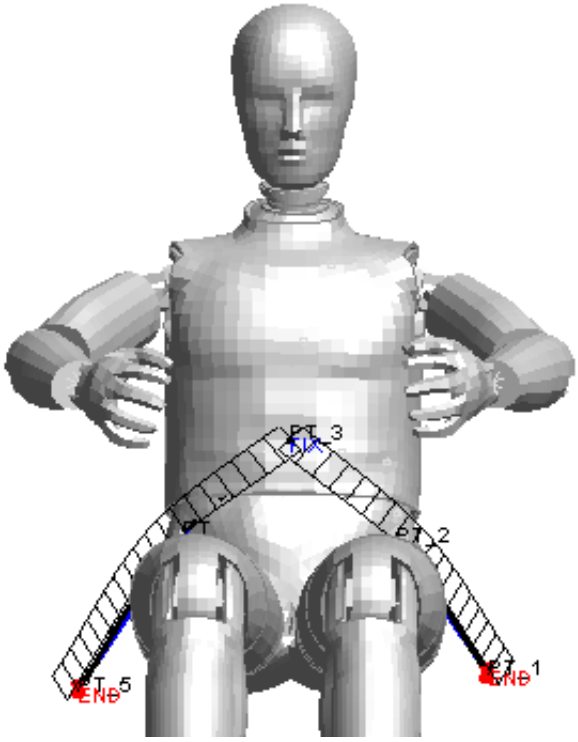


Adding "Fixed" points.

Fixed intermediate points effectively form a fixed end to the adjacent sections, and this would be unusual in the case of a normal lap and diagonal 3 point belt.

However, as this example shows, were you to create a full four or five point harness you would have to do this in two or three separate belt fitting operations, each meeting at a common point at the centre of the abdomen.

This picture shows how you might create the lap section with the fixed point #3 forming the buckle. The two shoulder belts would be added by a similar process, also meeting at the node used for point #3.



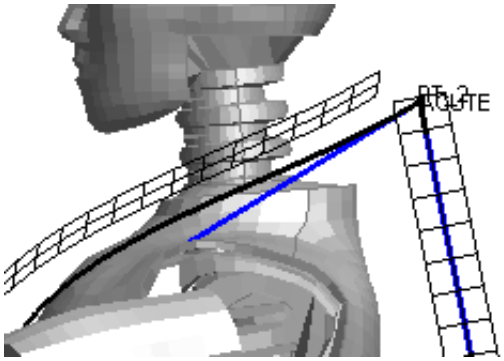
Handling "Acute" points.

By default the fitter assumes that it will be draping a curved belt path over a reasonably curved shape, and it also assumes that the belt path will curve reasonably smoothly without sharp changes of direction. It defines a "sharp change of direction" as an "acute" angle, and by default it assumes that acute means < 90 degrees.

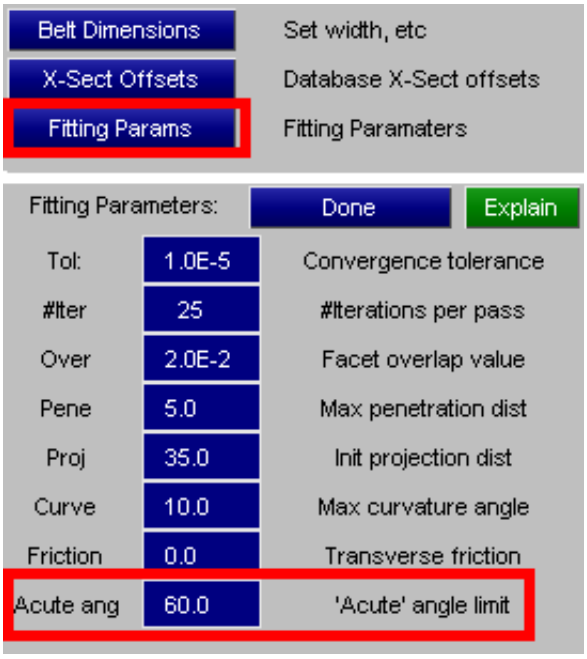
Acute angles at points are handled in one of two ways:

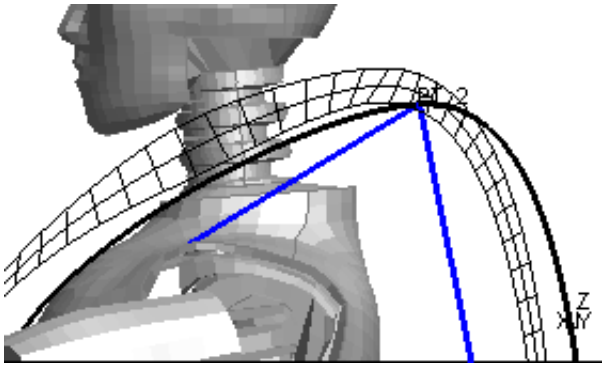
- If a slipring is defined at that point the belt is assumed to run continuously through the slipring, and will be meshed accordingly.
- Otherwise a break in the belt path is assumed, and the mesh will not be continuous.

In the upper image here the B-Post slipring at the dummy’s shoulder has been removed leaving an acute angle less than the default of 90 degrees, hence a break in the belt path occurs. When meshed the two belt paths would not be continuous, but instead each end would share a common nodal rigid body with a single common node at the path point.



In the lower image the definition of an "Acute angle" has been changed to 60 degrees meaning that the belt will run continuously through this point. The change is made in the "Fitting params" section as shown below:



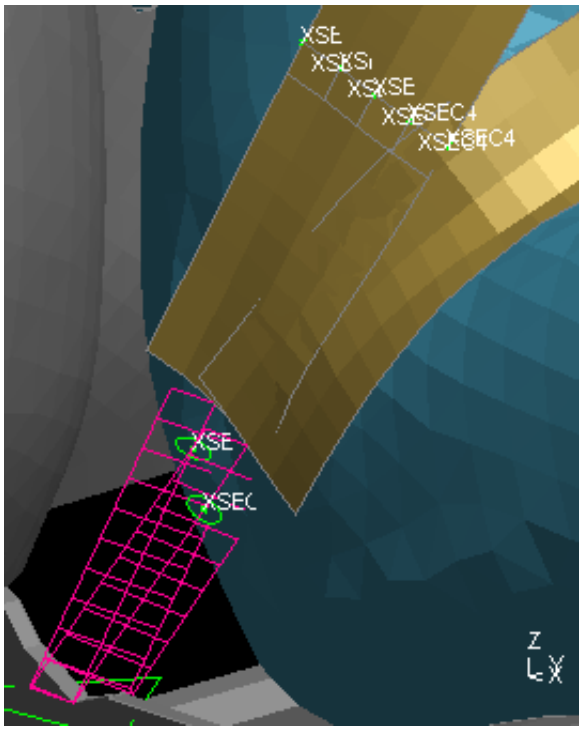


Adding Cross-Sections.

It is possible to add a **CROSS_SECTION** definition at any path point, including retractors, sliprings and ends. This does not affect the geometry of the path, but during the meshing stage a ***DATABASE_CROSS_SECTION** definition will be created at that point. Its attributes will be:

- It is required to have a label, therefore it will use the **_ID** suffix.
- The type of ***DATABASE_CROSS_SECTION** created depends on the type of belt elements it cuts.
  - Cutting shells or 2D seatbelt elements creates a **_SET** definition in which **SSID** is a set of elements (2D belt elements are really shells), and **NSID** is a set of nodes at one end of those elements.
  - Cutting 1D belt elements requires the **_PLANE** variant to be used since LS-DYNA does not contain the concept of a set of 1D belt elements, however the 1D seatbelt element has a part, therefore it can be placed in part set PSID.

    The origin of the plane will be at the centreline of the belt half way through the "next" element beyond the path point. This location is chosen in order to give a well-conditioned intersection, as cutting elements exactly at mesh lines can cause problems.

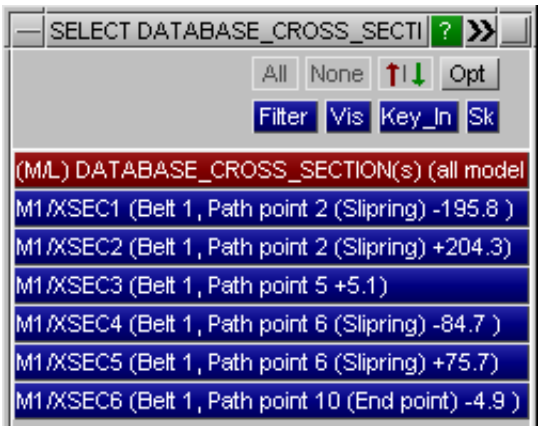


The image here shows cross-sections near the pelvis slipring on a belt meshed with a mixture of 1D belt elements and shells. It can be seen that in the 1D region the sections are circular planes, normal to the direction of the belt; in the shell region they are defined by sets of shells and nodes.

Database cross-sections meshed on belts are automatically given titles of the form:

[Belt id] [Path point id] [Path point type if relevant] [Distance from that point].

For example here is a menu of database cross-sections generated for the mesh in the picture above.



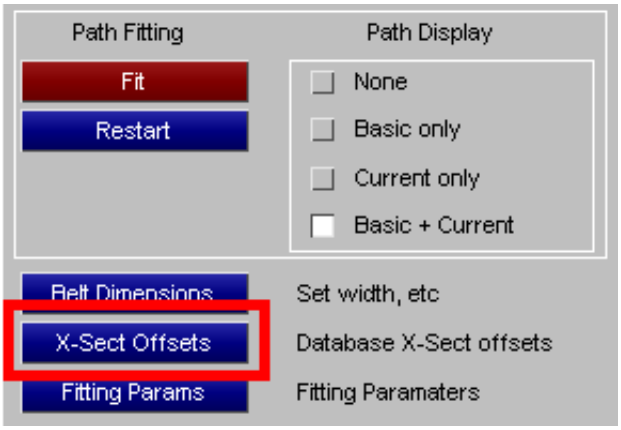
Special rules for cross-sections defined at end, fixed, retractor or slipring path points.

The belt path editor permits you to define a cross-section at a "cusp" point in the belt: end, fixed, retractor or slipring locations. Obviously locating a section exactly at such a point would not work since either it would not intersect an element (belt end point), or it would lie exactly on the border between two elements (intermediate / slipring point). Therefore in order to obtain a well-conditioned cut the following logic is used:

End point (Fixed end or retractor)	A single cut section is located, by default half an element length into the belt.
Intermediate point (Acute fixed point or slipring)	Two cut-sections are created, one on each side of the point. By default these points are offset by varying amounts as described below.

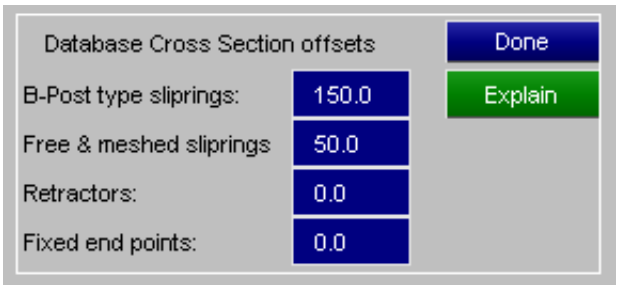
Offsets from points are necessary at slipring locations where locating the sections 1/2 an element either side of the slipring would result in the "upstream" one being pulled through to the other side as the belt tightened. Offsets may also be defined at other points.

Offsets are defined in the fitting panel using **X-Sect Offsets**



The default offsets are split into four categories.

B-Post slipring	Generates 2 sections, one each side, by default at approximately +/-200 from the point.
Free (pelvis) slipring	Generates 2 sections, one each side, by default at approximately +/-150 from the point.
Meshed slipring	
Retractor	Generates a single section with a default offset of zero.
End point	Generates a single section with a default offset of zero.



The actual locations of cross-sections will always be at the mid-point of the row of elements closest to the desired location. For example even though the offset it zero at an end point the section will be located 1/2 element length into the belt.

This also means that the location of sections with a finite offset will be at the 1/2 element location nearest to the requested distance from the point, and not exactly at the requested offset distance.

In the example above an offset of 80mm from the pelvis slipring was set, and it can be see that the actual locations of the sections as reported in their titles was 84.7mm before (-ve) and 75.7mm after (+ve) the slipring location.

The effect of Sliprings on the belt path.

There are two distinct geometries of slipring that need to be considered if 2d seatbelt elements are to be meshed correctly:

1. **B-Post** (shoulder) location, where the slipring is constrained to rotate about the transverse axis defined by its fixing bolt.

2. **Free** (typically, but not exclusively, pelvis) location, where the slipring is free to adopt the average orientation of the two belt segments meeting at that point. **Meshed** (radiused) sliprings are also effectively "free", in that their orientation is not constrained by adjacent path geomety.

Before V12 you simply defined a "slipring" at a point, and PRIMER would choose the type automatically based on the geometry of the adjacent belt path points. This caused problems when users wanted to mesh non-typical belts, resulting in B-Post geometry being used when Free geometry would have been more appropriate.

From V12 onwards the type of a slipring is explicitly defined by the user, who must choose either "B-Post" or "Free", and this solves the problem of using the wrong type. It also makes it much easier to fit a belt to shapes that are not a typical dummy.

From V13 onwards it is also possible to use a "Meshed" slipring which is explicitly meshed, using finer elements, around a stipulated radius.

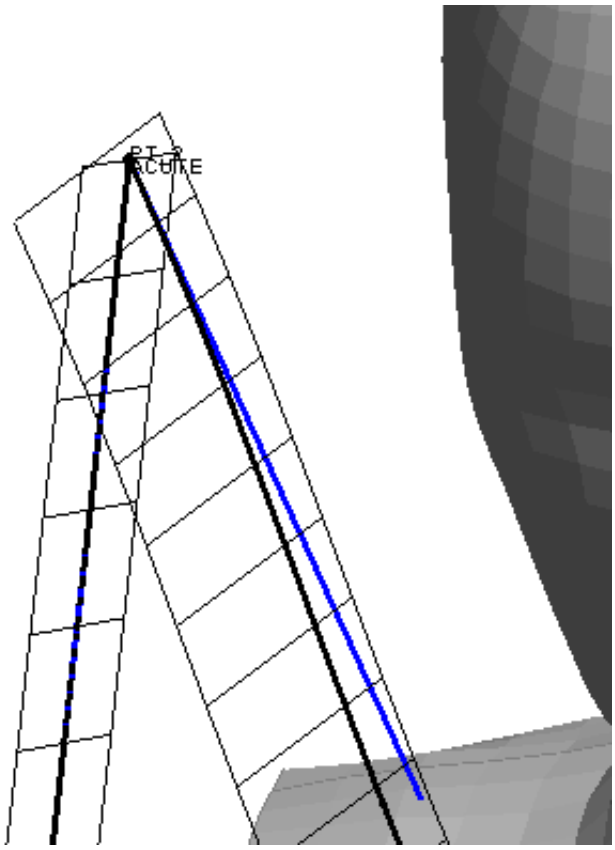
Backwards compatibility of slipring type from pre-V12 decks.

Input decks from versions ofPRIMER written before V12 do not contain this explicit distinction, and for backwards

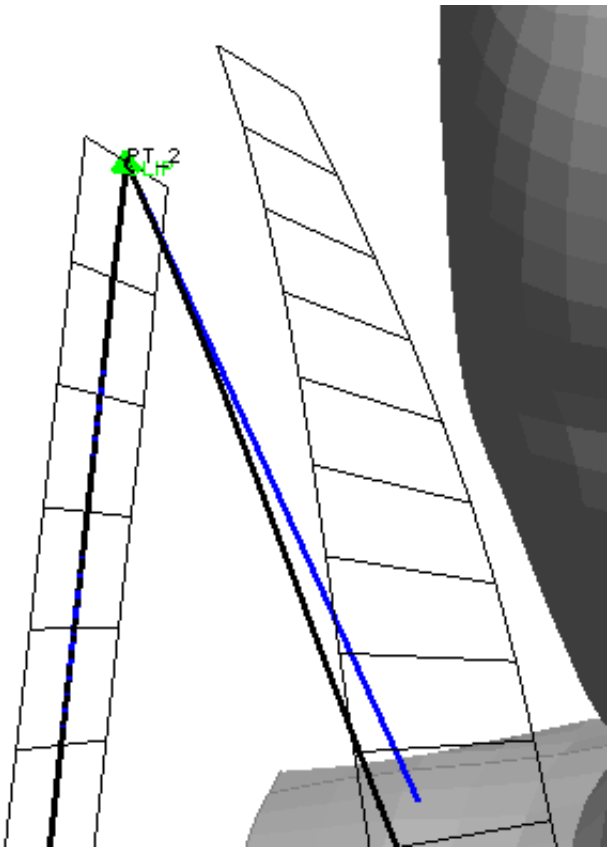
compatibility these are still read and processed as before, making the automatic choice of slipring geometry using the old rules. However the user can use the path editor of the belt fitter to change this to an explicit choice, and once such a deck has passed through the belt fitter it will write out this explicit choice in the *BELT cards. Therefore conversion to V12 format is a "once off" and permanent process.

Slipring Case 1 : B-Post type at shoulder location.

Note the difference between paths at the B-Post in the images above before and after a slipring was added, shown enlarged below.



Before slipring, simply an "acute" point.



After slipring defined, twist of free section has changed.

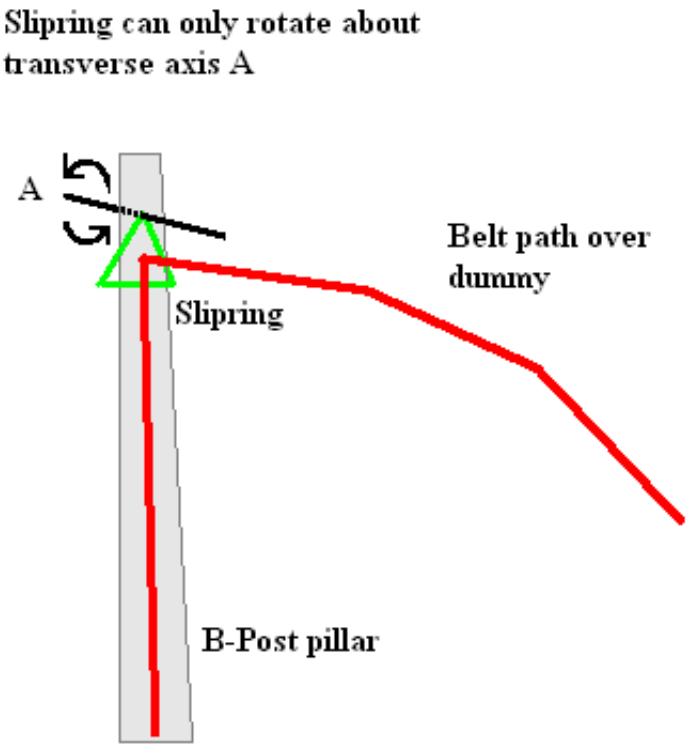
This example shows how the special case of a slipring at the B-Post location is treated. PRIMER assumes that the slipring is only free to rotate about (approximately) the horizontal axis across the vehicle, axis "A" in the diagram here, defined by the bolt attaching it to the B-Post

This requires that the section of the belt going back over the shoulder towards the slipring must adopt a reverse twist in order to give the correct belt path through the slipring.

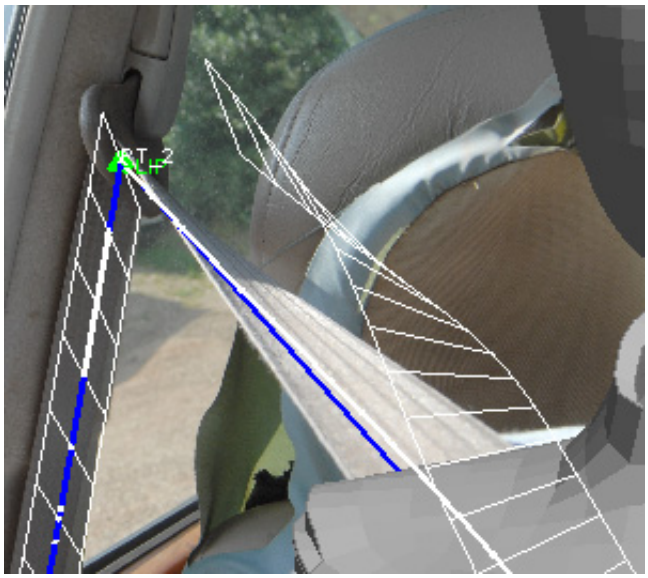
PRIMER treats axis "A" as being the outward normal of the straight section of belt between retractor and slipring, which poses a problem since it is just a line and lines have no orientation.

Therefore PRIMER estimates what it believes will be a credible orientation for that section of the belt path, but it may require adjustment to its twist angle to achieve the correct angle at the slipring.

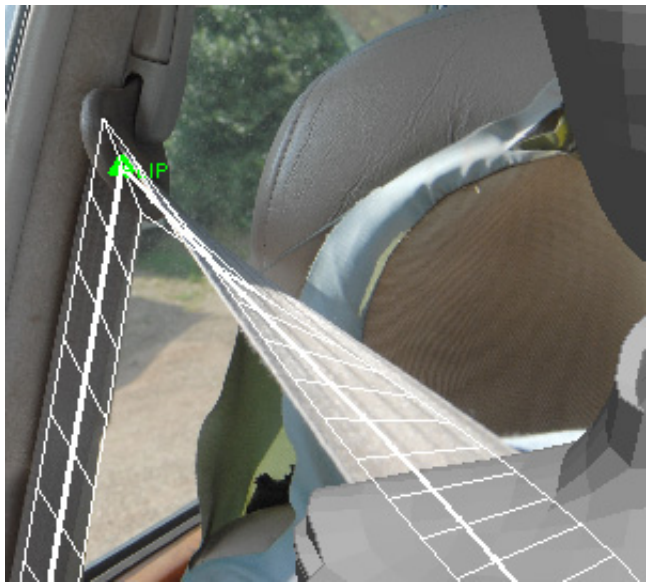
NOTE: If you define an orientation vector via "twist" nodes N1 and N2 at a B-Post slipring the N1N2 vector "wins" and the special B-Post logic is ignored, effectively treating the slipring as being "free".



The following two images show seatbelt path fitting at the shoulder slipring in PRIMER superimposed on top of photographs of the shoulder belt detail taken in a real vehicle, and they demonstrate how the twist of the belt passing over the shoulder has to rotate backwards to match the angle at the slipring imposed by its limited rotation axis.



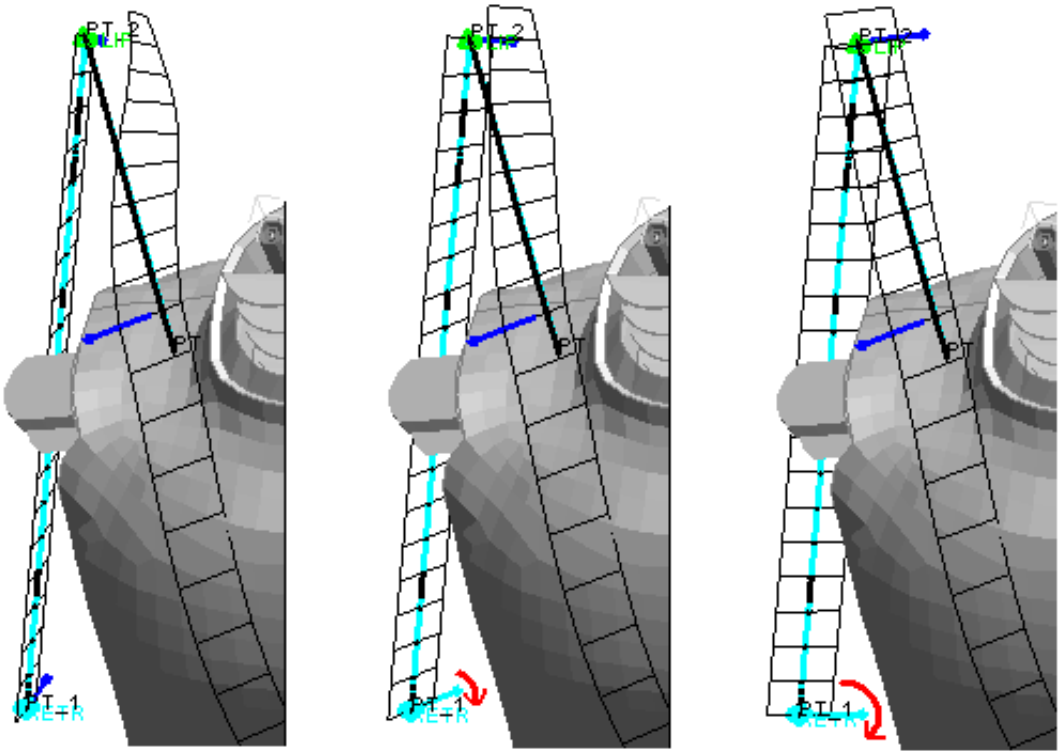
Before fitting, with free section of path projected above its final position



After fitting, showing the final shape of the belt path.

The orientation of the shoulder slipring is based on the outward normal of the straight vertical path from retractor to slipring, and since this is a straight line there is no "correct" value for this. PRIMER attempts to choose a default orientation that makes sense, but it may not always get it right making it necessary to adjust the twist of this straight section to achieve the correct geometry.

The following figure shows how altering the twist of the vertical section influences the shape of the belt at the shoulder slipring.



Default orientation, considerable twist at shoulder.

Belt rotated by about 30 degrees

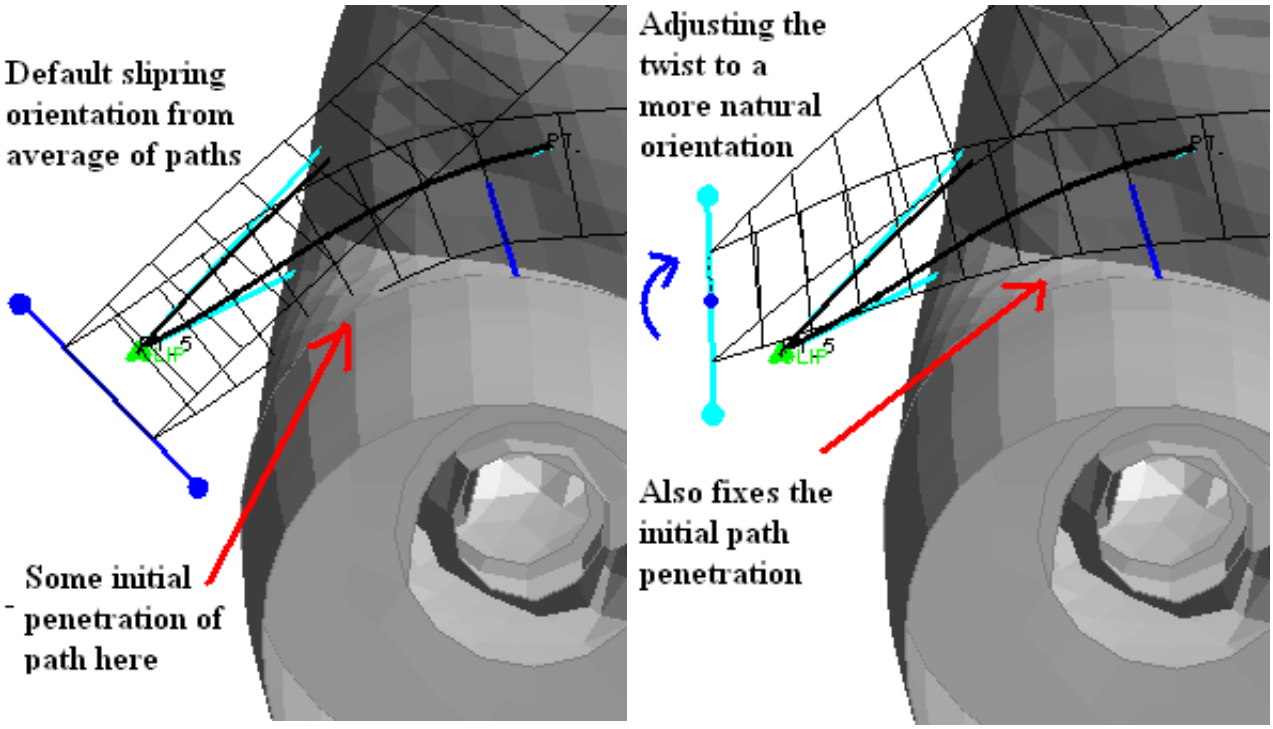
Belt rotated by about 80 degrees

In this example the belt has been rotated at the base (retractor) location, but it could equally well have been rotated at the top (slipring) location. If an explicit twist is applied only to one point on a straight section of path then the whole section will rotate as shown here, however separate twists may be applied at each end in which case the path will twist between them.

Slipring Case 2 : Free type at Pelvis location

At the pelvis the slipring lies between two sections of belt that are both curved and thus free to move during fitting. Its orientation is unknown, and cannot be deduced from its location or the belt geometry, so instead of attempting to control the twist of the belt path it simply adopts the average twist angle and orientation of the two sections attached to it.

The following example illustrates how this slipring may affect the belt path, and how it can be adjusted to correct this.

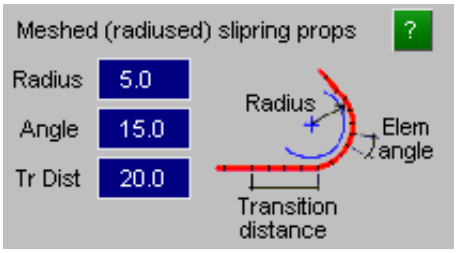


Slipring Case 3 : Meshed type at Pelvis location

A meshed (radiused) slipring can be used wherever you might otherwise consider using a free slipring.

Unlike the normal B-Post and Free sliprings, which use *ELEMENT_SEATBELT_SLIPRING to give a sharply angled transition in the belt path, meshed sliprings mesh the belt continuously around a user-defined radius. The user must define:

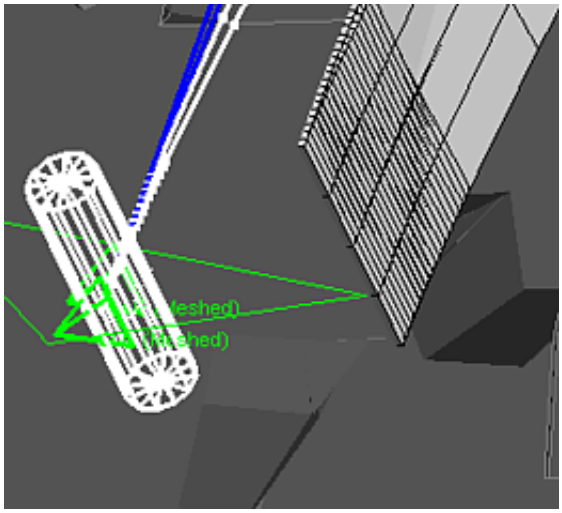
- The radius
- The angle subtended by the "short" elements near and around the slipring
- The transition distance over which short elements extend either side of the slipring.



Meshed slipring properties are defined in the **Fitting Options** panel.

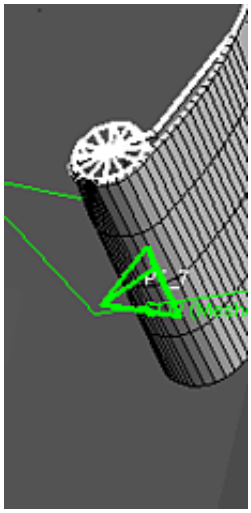
This example shows the pelvis slipring above using a Meshed slipring instead of a Free one (from a different viewpoint). Note that the white cylinder around which the slipring is formed is added by the path editor to show how the slipring will be located and oriented, it is not structural and has no physical existence.

When using a slipring like this you will need to have some genuine buckle structure around which the belt will fit, but since the belt path has to be pulled "through" this structure during the form-finding process the buckle should not be part of the "structure" used for fitting. However it will need to be part of the contact between itself and the belt during the analysis.



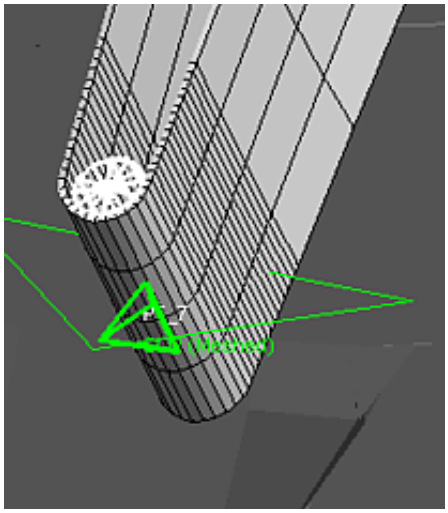
Before fitting.

The path is projected outwards and needs to be pulled back through and around the slipring in order to achieve the correct path.



During fitting.

The inner segment of the belt has to pass through the slipring cylinder to get to the correct side.

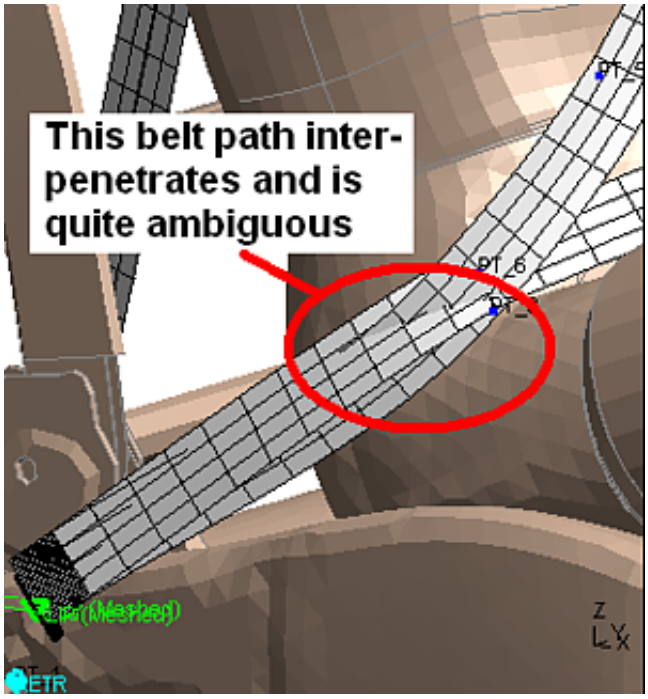


After fitting.

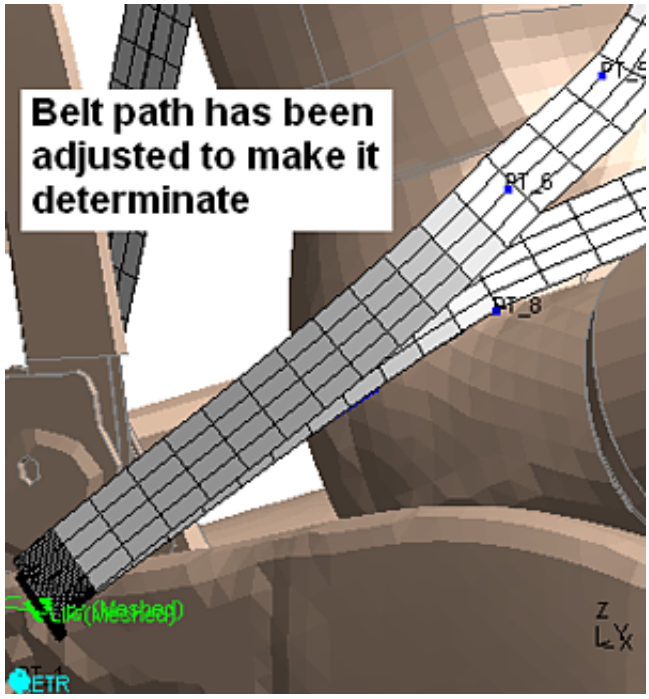
The path has now fitted around the slipring. Because it pulls through the slipring any buckle elements (not shown here) must not be part of the "structure" used for belt fitting.

It will also be evident from the images above that a meshed slipring requires an unambiguous definition of "in front" and "behind" belt paths. During fitting the "behind" segment must pass through the slipring while the "in front" segment stays on the outside.

Meshed sliprings use special logic to determine which belt segment is which in their immediate vicinity. It examines each segment of the belt path to see which has more points "in front of" the other, where the "in front" direction is the outwards vector along which the belt path is projected prior to fitting, and the one with more points in front is defined as the "in front" segment. Therefore you should aim to create an initial belt path that does not interpenetrate near these slipring locations, since this gives the algorithm the best chance of determining the belt shape correctly.



A poorly defined initial belt path



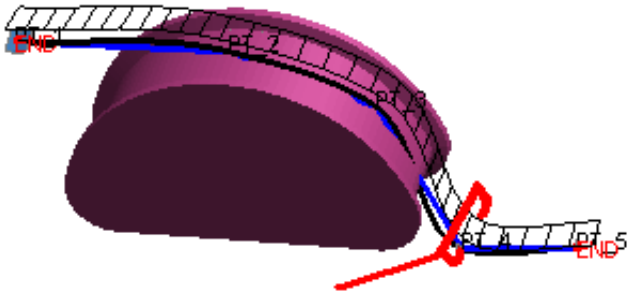
Path has been adjusted manually to make its shape clearer

In fact this clear separation of belt paths is good modelling practice for all slipring types, both meshed and explicit using *ELEMENT_SEATBELT_SLIPRING. You can use initial depenetration to try to do the job for you, but you are more likely to get a good result if you separate the belt paths manually.

Slipring Case 4: Platen test.

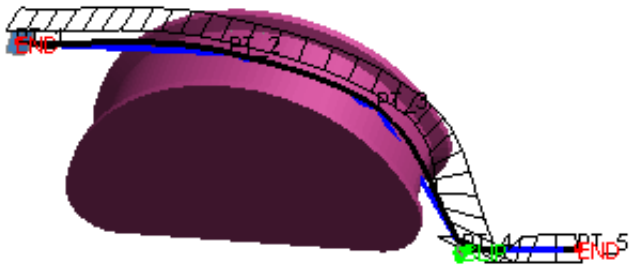
This is a simplified seatbelt pull-out test in which a belt passes over a platen and then through a slipring to a fixed anchorage point.

The user wants to put a slipring at point #4 to guide the belt pull over the platen



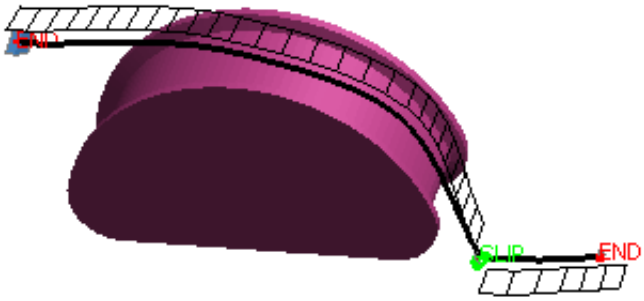
Slipring to go here

This image shows the initial path before the definition of any sliprings.



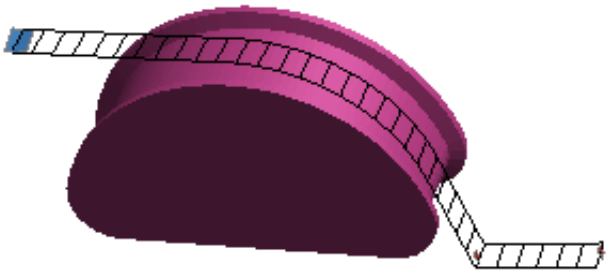
Making point 4 a B-Post slipring leads to the geometry becoming horribly twisted as shown here.

In versions of PRIMER before V12 the logic used to determine slipring types would have made this a B-Post slipring, with the horrible consequences shown above. Not the right answer!!



The correct solution: using a Free slipring at point 4.

The correct solution here is to use a "free" slipring as shown above. You may need to control the twist a little to obtain the required orientation, but the free logic that averages the two paths is the correct solution here.

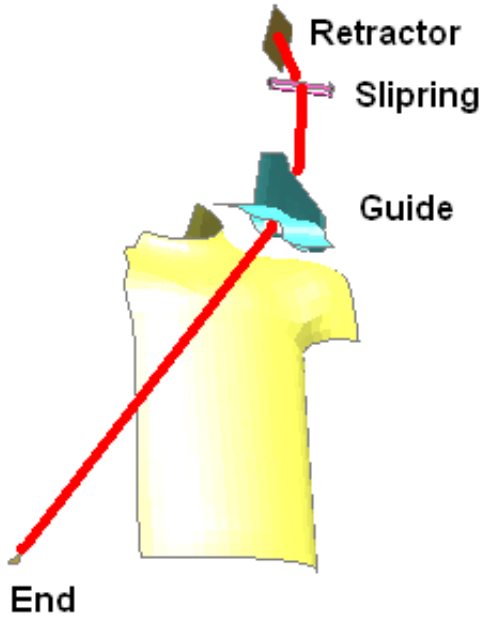


Belt path after fitting

And here is the full model with fitting complete. Note how the two belt sections have come together at the slipring location.

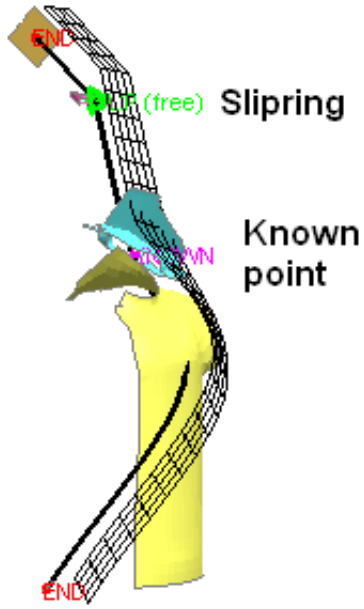
Slipring case 5: child dummy in seat.

This is an example that is becoming common as child dummies in booster seats are modelled.



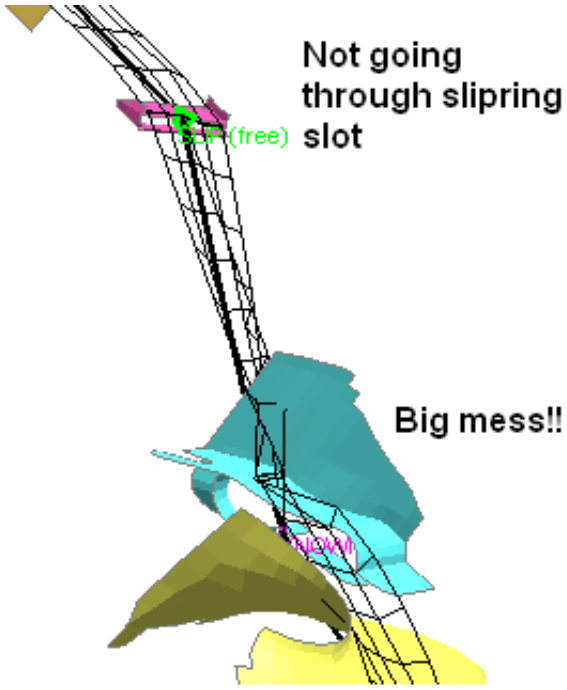
Child seat belt: the problem

The problem here is that the belt has to pass accurately through a slipring, then also through the slot in the guide before crossing the chest of the child.

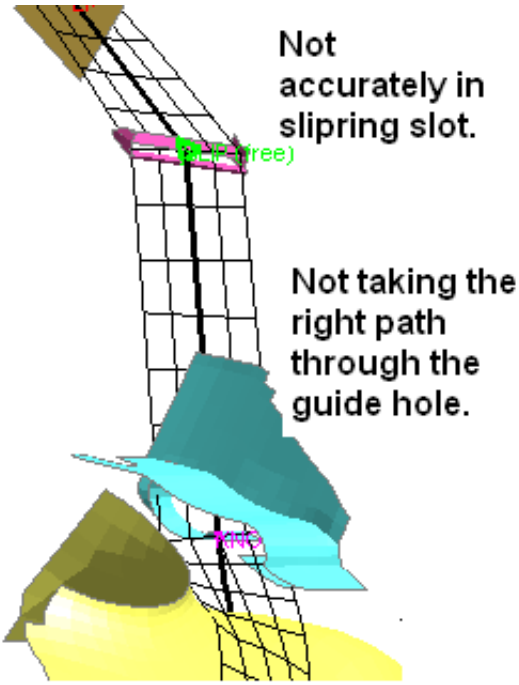


Simplistic solution

This is the over-simplistic solution to the problem. All components shown here have been made part of the "dummy"; a slipring has been located at the slipring slot, and a known point at the centre of the guide.



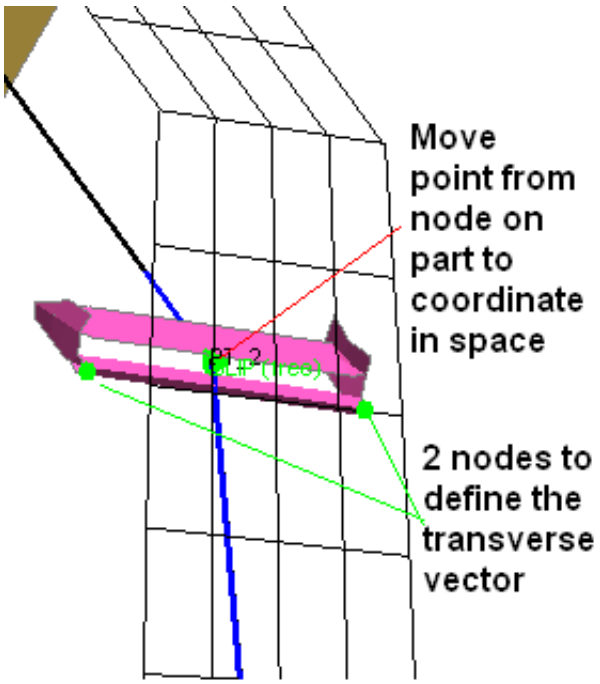
Going horribly wrong: part 1



Still wrong, but getting better...

And it all goes horribly wrong at the slipring and guide locations (but the rest of the belt is OK).

The problem here is that the material of the slipring (magenta) and the guide (blue) have been made part of the "dummy" and so are considered for contact with the belt. Since the belt path is "thrown outwards" prior to fitting it cannot then pull itself back through these parts.



Sorting out the slipring

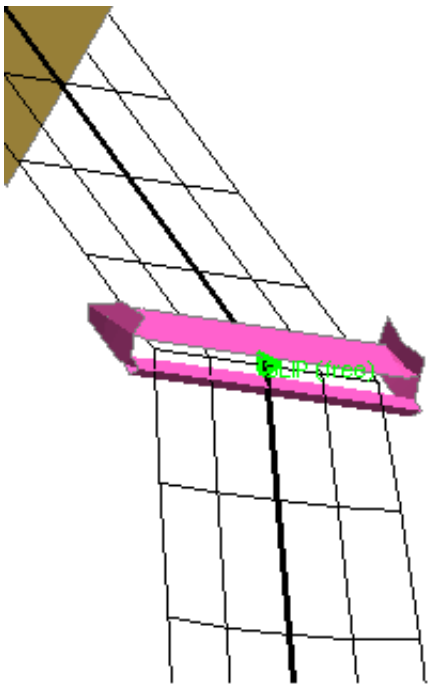
The slipring needs its twist adjusting. Also the use of a node on the magenta part to define the point #2 is wrong as this locates the belt path on the material itself rather than in the air gap.

So point #2 is moved down slightly to occupy a location in space in the middle of the slot. (It’s fine that it is not on a node.)

Also two nodes N1 and N2 (green dots) either side of the slot are used to define the transverse vector correctly. They don’t have to be co-linear with the path point, only the N1N2 direction vector is used.

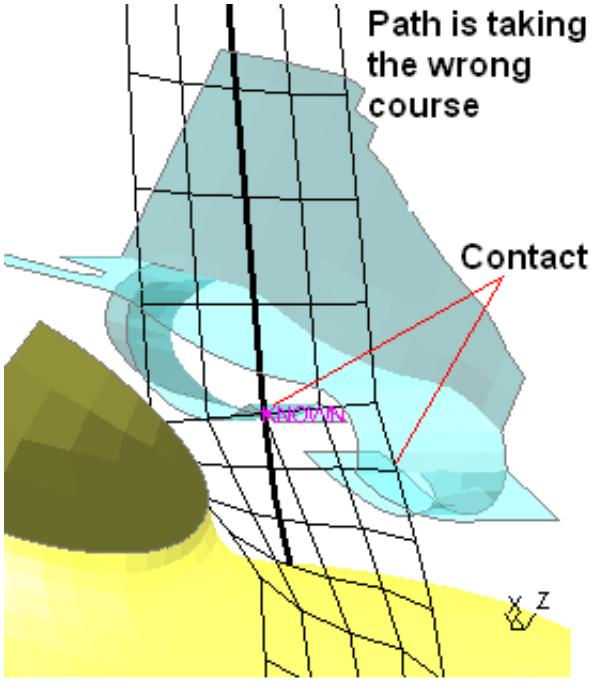
Here the magenta and blue parts have been removed from the "dummy" definition, and are thus ignored by belt contact. This is fine for the purposes of belt fitting, they can always be added back into the "true" contact used during the analysis.

This is better, but the belt path still needs some adjustment to get it accurately through the geometry it must navigate.

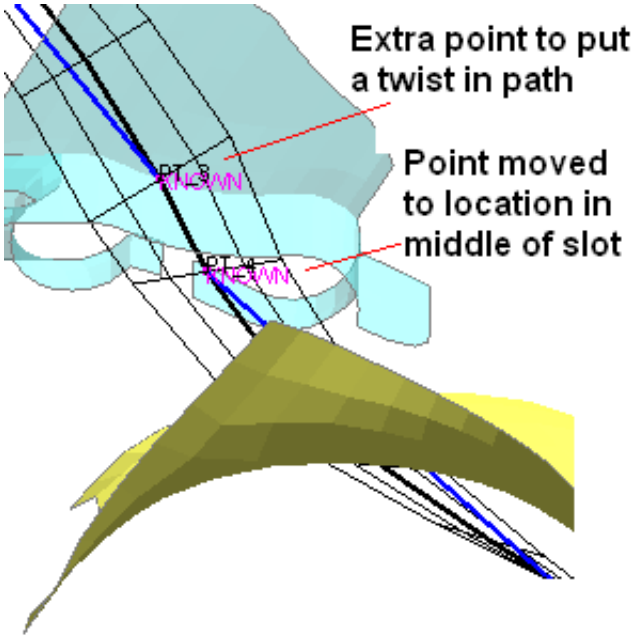


Now the belt lies neatly in the slipring slot after fitting.

This image, a close-up of the belt path after fitting, shows how it fits neatly through the slipring slot now that its position and alignment have been corrected.



Close-up of path errors at guide
(The guide has been made transparent here)



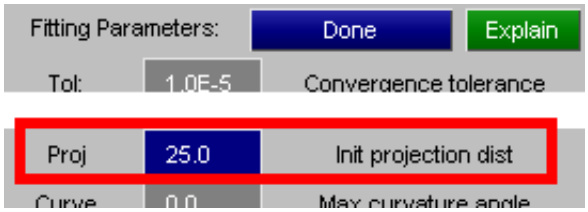
Correcting path through guide slot
(Projection distance reduced to 0.1 to show path that the fitter will aim for.)

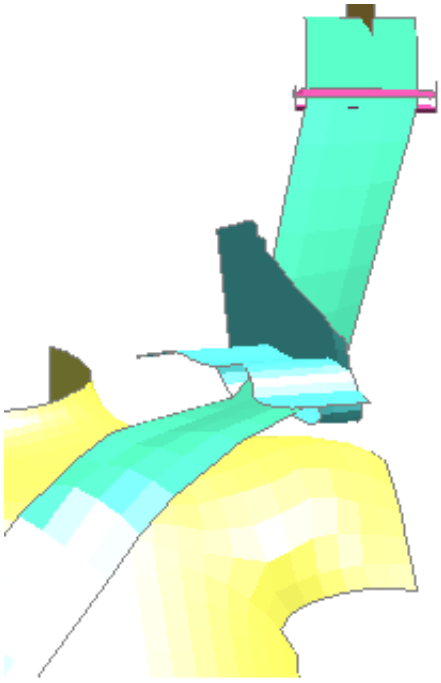
This image shows details of the problems at the guide slot.

The belt path is wrong, it needs to twist more, also the use of a node on the slot to locate it means that (like the slipring above) it will make contact with the material.

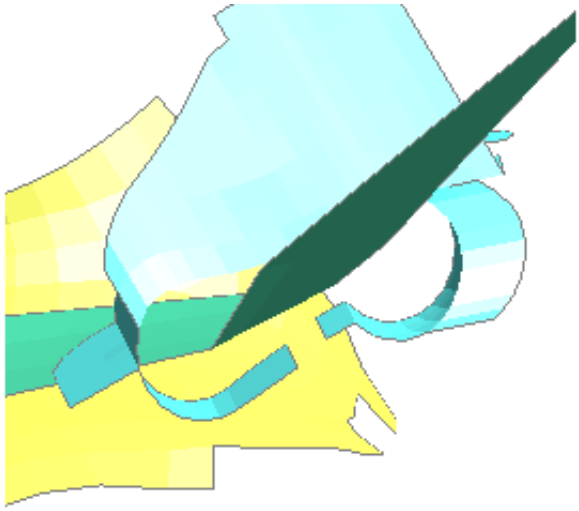
Hint: reducing the "projection distance" from its default of 25mm to 0.1mm will leave the fitter path near its "true" position, making it easier to work out how to align it. You need to reset it back to its default before commencing form fitting ("fit" operation).

An extra point has been added on the far side of the guide to put a curve into the path at that point, and the twist of these two points has been adjusted to get the belt path passing through the slot.





End result: neatly fitted belt.



View of belt guide from other side showing path is correct.

Should you use a slipring or simply a "known point" (as used here) where the belt passes through the guide? There are pros and cons to consider:

- Using "Known points" is simple, but you will have to model explicit contact between the guide and the belt if they are to make contact during the analysis and work correctly together.

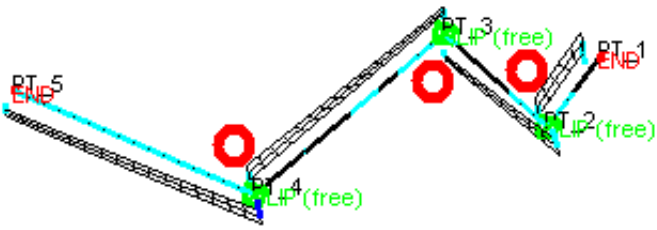
  Also if the belt and guide meshes are quite coarse you will get a "ratchet" effect as the belt elements pull through the guide slot, a bit like pulling a bicycle chain over an edge, which may overestimate the true friction of this arrangement.

- Using a slipring, or possibly two, is simple enough: just replace "Known point" with "Free slipring" points 3 and 4 here. However there will be some extra work to perform since you will need to locate the slipring(s) in the slot which, for a 2d slipring, will mean setting up a line of nodes in "thin air", attached to the guide material probably by a nodal rigid body.

  However once set up sliprings will give you a smoother transition of material from one side to the other, and will also provide instrumentation of pull-through distance and force.

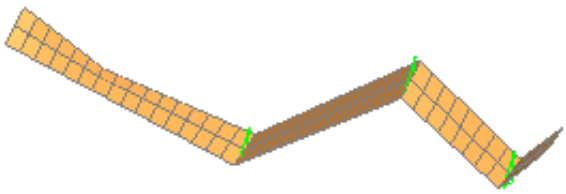
Slipring case 6: Belt passing over rollers.

This is a fictional example in which a belt passes over a series of rollers in a "zig-zag" fashion.



Belt passing over rollers, using "free" sliprings at roller locations.

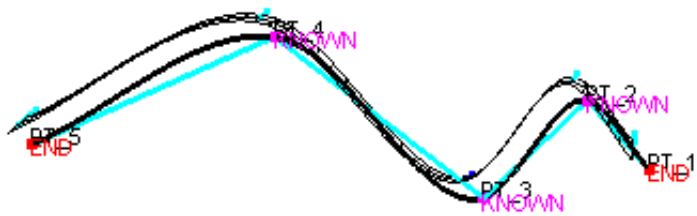
Free Sliprings have been used at the roller positions (red circles)



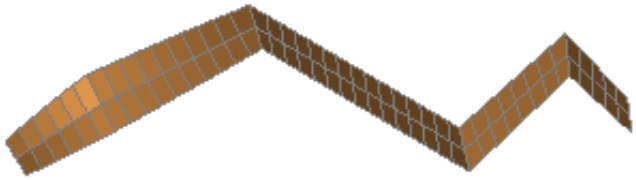
End result showing zig-zag shape with free sliprings at roller positions

How the mesh ends up once fitted.

An alternative solution using "known points" rather than sliprings



Alternative approach using "known" points rather than sliprings.



Meshed result. You will need to supply explicit contact at roller locations.

You don't have to use sliprings. If you define the roller points as "known" the fitter will put the path through them giving much the same shape as above.

However remember that if there are no sliprings then there is nothing to restrain the belt during the analysis, and you will need to provide some structure and contact to represent the rollers.

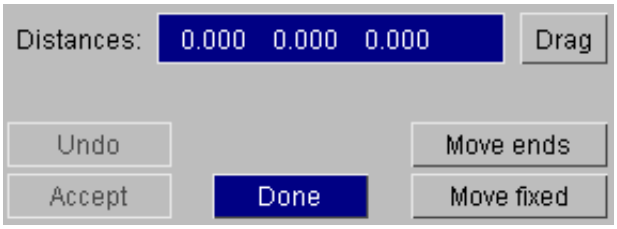
Orienting the whole path

It can be convenient to move the whole path as a single entity, for example when refitting an existing belt path to a dummy definition that has moved.

Therefore it is possible to apply the standard PRIMER orientation functions to the points that make up the path.

The transformations are self-explanatory, and usage is similar to that of the main **ORIENT** commands, with the exception of the treatment of fixed and end points. The **TRANSLATE** variant is used in this example, but it applies equally to the other modes.

By default only unfixed points are moved, but you can choose to move end and/or fixed points as well.



MOVE_ENDS Allows end points to move

MOVE_FIXED Allows all fixed points to move.

Paths can be oriented by typing in transformations, or **DRAG**ged with the mouse. Dragging uses the same <*mouse button*> vs <*orientation axis*> arrangement as modifying single points: left button is X axis, middle Y and right Z.

Aborting path operations

CANCEL Cancels the current editing operation leaving the external (saved) path unchanged.

DELETE_ALL Deletes the scratch path inside the editor.

RESET_ALL Deletes both saved and scratch paths.

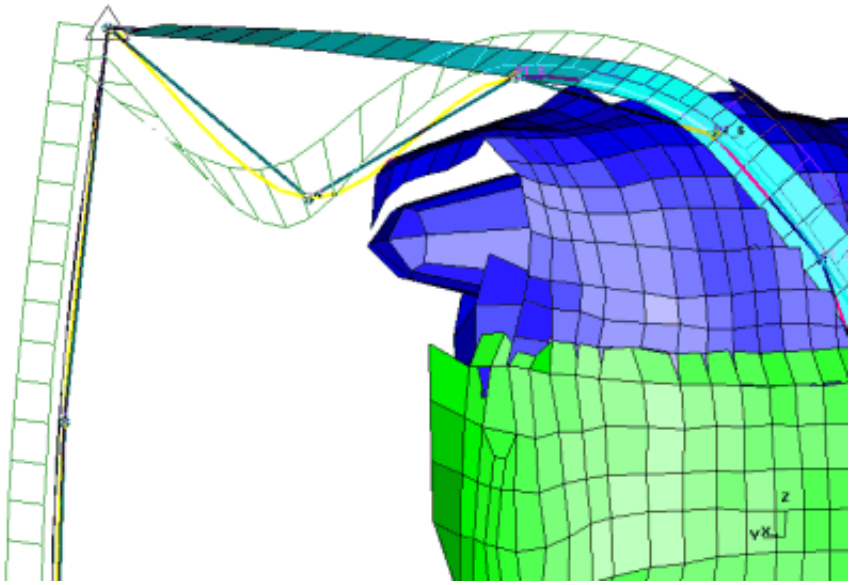
Tools... maps the **Fitting Options** panel.

Hints on basic path definition.

Avoid reverse curvature

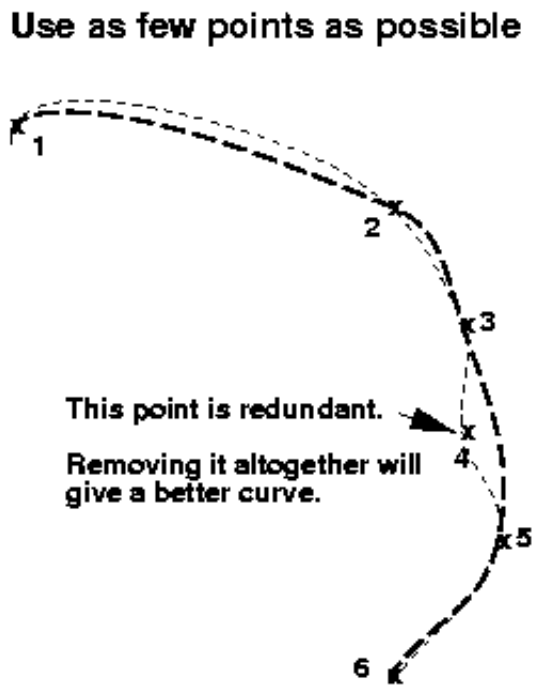
In this example an extra, redundant point has been added between the shoulder and the slipring, and it has been located such that it causes reverse curvature. (A correctly meshed belt is also shown for comparison.)

This shows that the chassis mesh gets twisted - almost through 180deg - which is clearly wrong, and while it could be corrected by adjusting the belt twist it is better to avoid it in the first place by sensible positioning of path points.

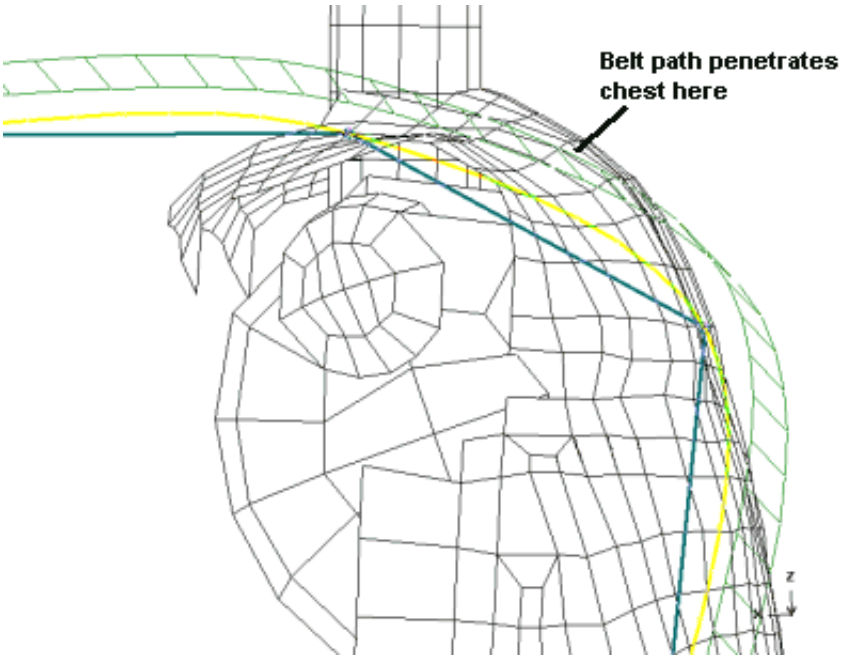


A common cause of this happening is the use of too many points when defining a path. In the example here point 4 is unnecessary: the dotted line shows how it causes reversal, whereas omitting it (long dashes) works OK.

Remember that an initial path does not have to follow dummy contours slavishly: it needs only to make a suitable initial shape for subsequent form-finding, thus a very simple shape will be adequate.



Choose as few points as possible for each path segment

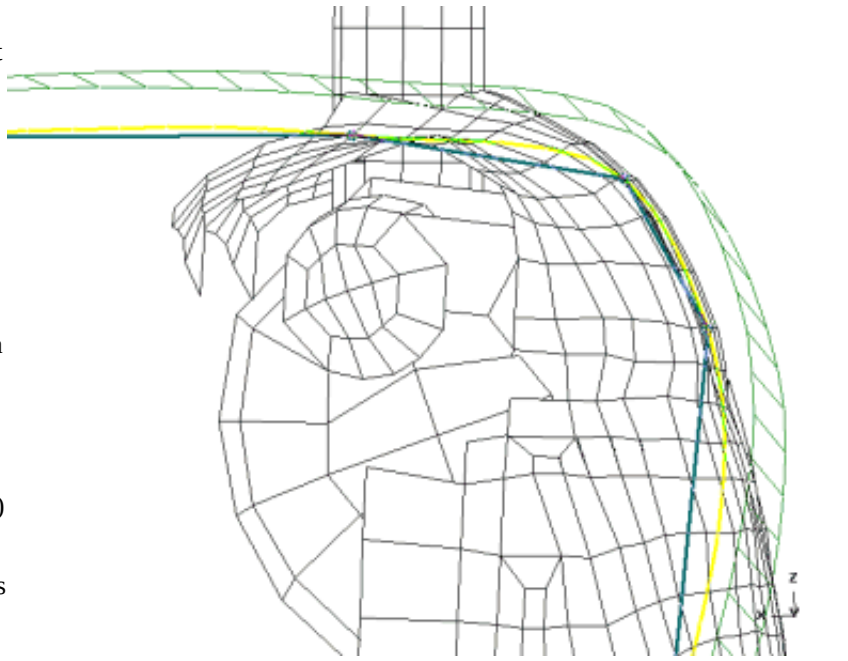


In this example the advice has been taken too literally!

A point has been chosen on the shoulder, and another on the abdomen, but the path between then fails to clear the bulge of the upper chest.

If this was fitted part of the belt would get stuck "inside" the chest shells.

The problem could be solved in two ways: either by increasing the initial distance by which the belt is projected outwards prior to fitting (which would be an inefficient solution), or by adding an extra point.



Here the problem has been solved by adding an extra point on the upper chest to force the belt path clear of the dummy.

(This is more efficient than increasing the projection distance since fewer form-finding iterations will be required to pull the path back into contact with the dummy.)

However the principle of using as few points as possible to define the belt path is correct as it will give a simpler and smoother path, as well as reducing the likelihood of reverse curvature. (As with approximating functions with polynomials, a lower order equation will give a smoother, if less precise, fit.)

Remember that the initial path does not have to adhere slavishly to the dummy's contours: a simpler and looser shape will work just as well as it will be pulled back onto the dummy surface by the form-finding process.

Good basic path creation practice

Use as few points as possible.	Extra points are usually unnecessary, and can cause curvature problems: 4 or 5 per segment is plenty.
Basic path shape is not critical	The basic path does not need to follow the dummy shape slavishly: a "loose" initial fit giving good curvature will be quite satisfactory.

6.40.3 Fitting the belt to the dummy

Once you have defined a basic "path", and hence an initial "chassis" mesh, you can proceed to fit it to the structure.

You may need to adjust both its **DIMENSIONS** (width, length, #rows, etc), and also the fitting **PARAMETERS** (convergence value, contact attributes, projection distance, etc) before you proceed.

Finally, once the fitting process is complete, you can **ACCEPT** the result and return to the main seatbelt definition menu to mesh it.

DIMENSIONS Setting width, length, etc

The dimensions of a belt are:

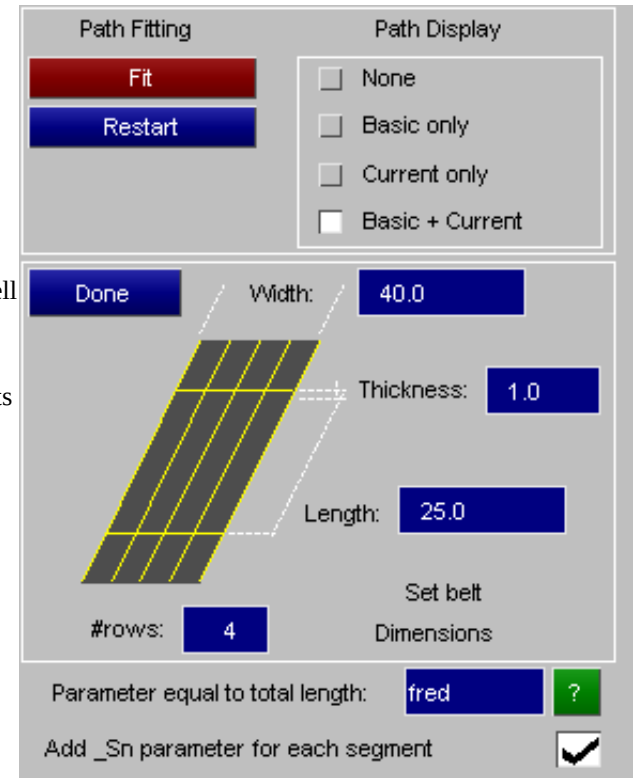
#rows: The number of (shell) elements across the width.

Width: The total width of the belt. (For **#rows**>0 each shell is **Width**/**#rows** wide.)

Thickness: Shell element thickness.

Length: Characteristic element length (both seatbelt and shell elements).

Clearly the first three refer only to belts that include shell elements. If **#rows**=0 then the belt becomes seatbelt elements only.



Parameter equal to total length

This is optional. If defined it gives a parameter name (as defined by the **`*PARAMETER`** keyword) that will be updated with the total length of the as-meshed belt each time it is generated. This can be referred to elsewhere, typically in **`*ELEMENT_SEATBELT_SENSOR`**, to update attributes in accordance with the revised belt length. See Parameter to contain the total seatbelt length for more details.

Add _S*n* parameter for each segment.

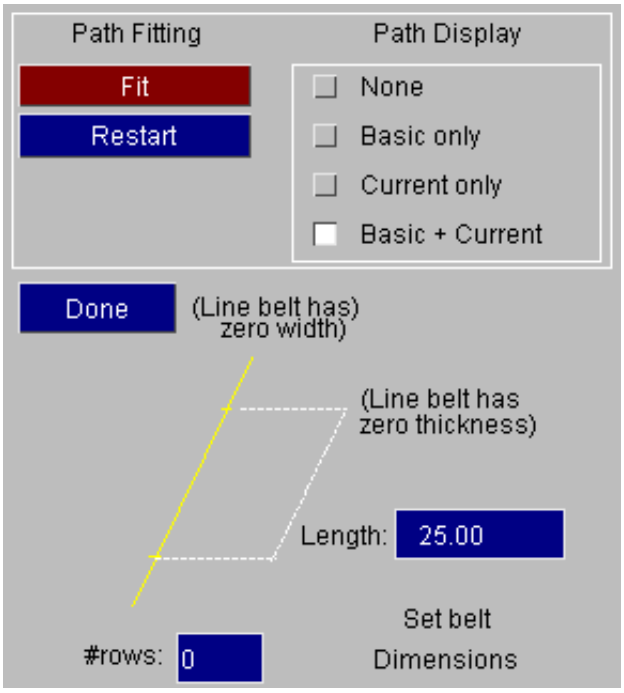
As an extension to the above if this option is selected each segment of the belt will have its length assigned to a separate parameter, which can be used in any context.

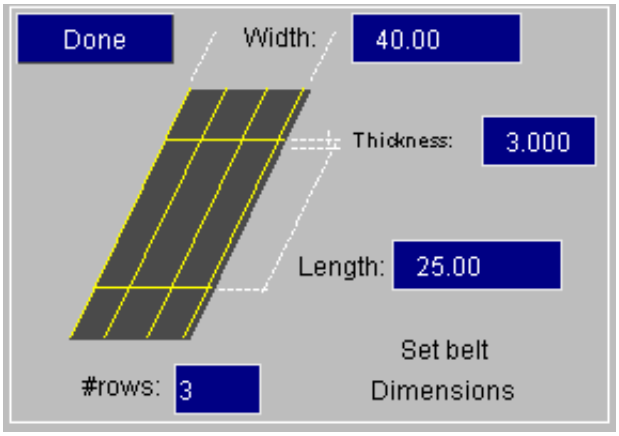
To save having to define multiple names these parameters are given the name of the total length one above with "_s1", "_s2", etc added for segments #1, #2, etc. For example:

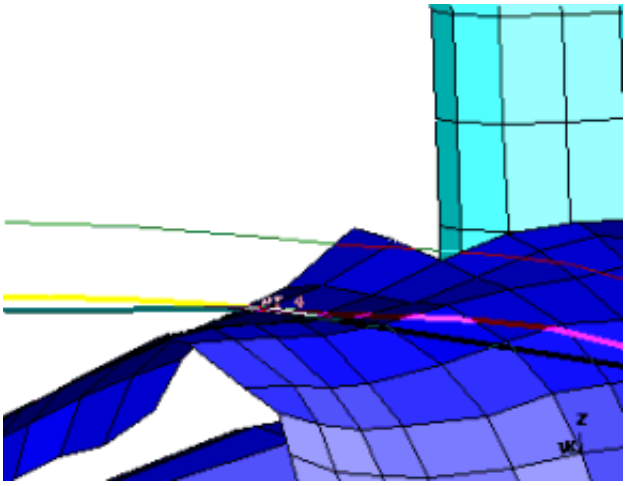
Nomenclature of per-segment parameters if parameter for total length is called **BLEN**	
Segment 1	**BLEN_S1**
Segment 2	**BLEN_S2**
... etc for all belt segments.	

Parameter names may not exceed 9 characters in normal LS-DYNA format, or 19 characters in long format. Therefore the name of the total length parameter will be truncated if necessary when adding "**_S*n***" suffices in order to stay within these limits. For example if the total length parameter is "**beltlen**" then the parameter for segment #1 will be called "**beltle_s1**" in normal format.

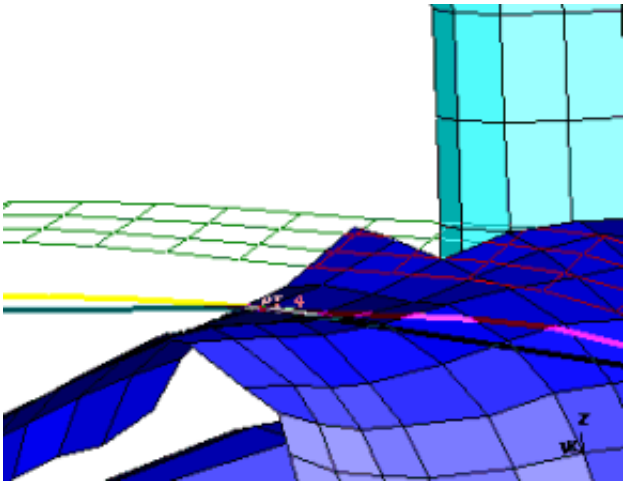
The following pairs of figures show alternative cases of no (seatbelt only) and 3 shell rows:





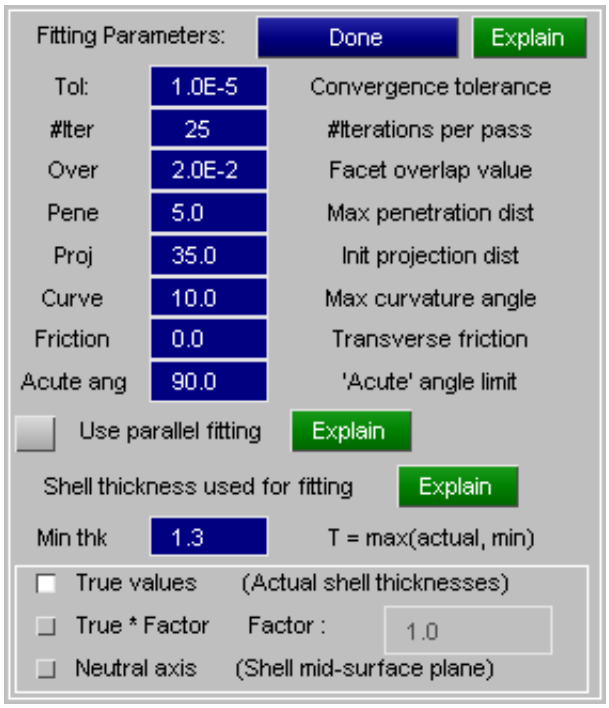


#rows=0: The seatbelt element only case



#rows=3: Three shells across width

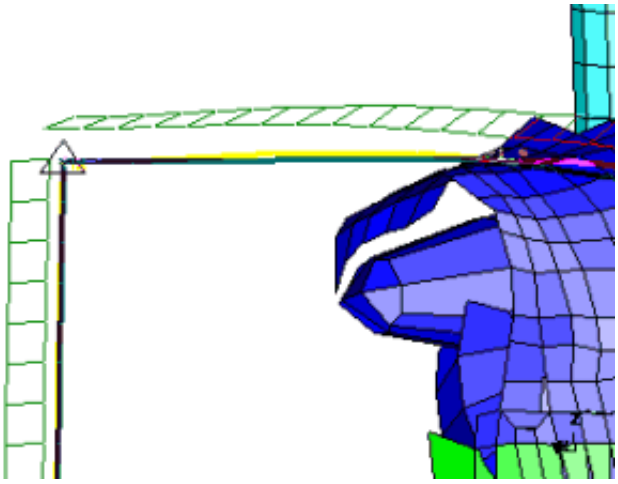
PARAMETERS #1 Basic control of the form-finding process.

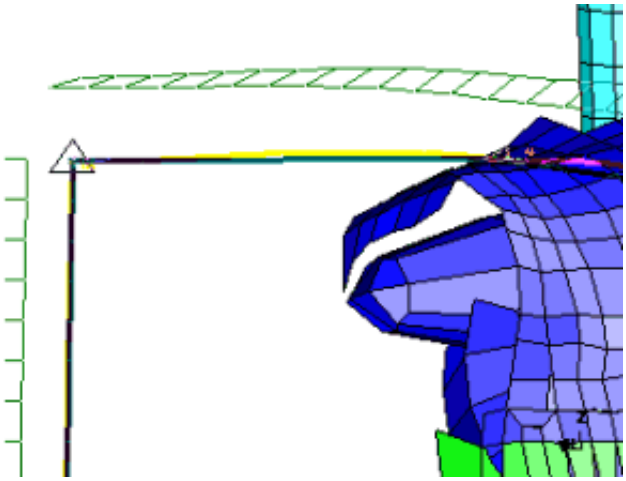


This panel contains controls for the form-finding process, and also the parameters for belt to structure contact that takes place during fitting. At this stage we will only consider:

Proj The "projection distance" which defines how far the chassis mesh is projected outwards prior to fitting.

The remaining controls will be discussed once the fitting process itself has been explained.





The **Proj**(ection) value is the distance by which the chassis mesh is lifted "outwards" from its defined position prior to fitting. The two examples above show the results of changing from the default of 25mm to 50mm.

There is no "correct" value to use, the criteria are:

- The distance must be sufficient to lift ***all*** chassis mesh segments clear of the underlying structure so that, especially in the case of shells, they start fitting on the correct side of the elements. (Remember that the basic path has almost certainly been defined using nodes on the element mid-planes.)
- The distance must not be so great that it causes the chassis mesh to interact with unrelated bits of structure, or distorts the initial shape so much that it doesn't get pulled back onto the structure correctly.
- It is uneconomical to use excessively large values since it will require many form-finding iterations just to close the gap with the structure.

Special treatment of fixed (segment end) points

Note that projection is applied to all points, *including* the fixed (segment end) ones and, as in the examples above, intermediate fixed points will have separate projections for each of the two adjacent segments.

Fixed points get exceptional treatment during form-finding: at these points the chassis mesh is only permitted to move along the "radius" vector back to the point so, ultimately, an end point should arrive back at its defined position - subject to any interaction with the structure.

If structure intervenes at an intermediate segment end point causing the two segment ends not to be coincident following form-finding, the first segment will "win" at the meshing stage and the common end node will be at its location causing potential distortion of the second segment's end element. You should correct this situation by altering

the segment end point to avoid such distortions.

Special treatment of "known" intermediate points

Intermediate "known" points are also treated specially. In effect there is a strong force pulling them back to their "known" position during the form-finding operation, and this is stronger than the force which tries to straighten the belt path at that point.

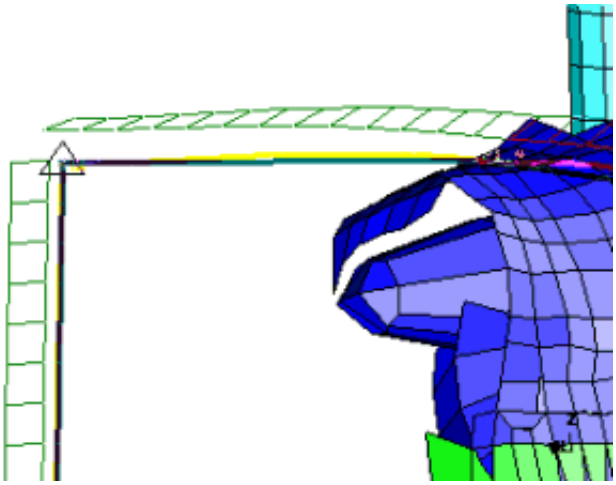
Movement of these points is not as constrained as that of "fixed" ones, but the effect is very similar.

FIT: Commencing the form-finding operation

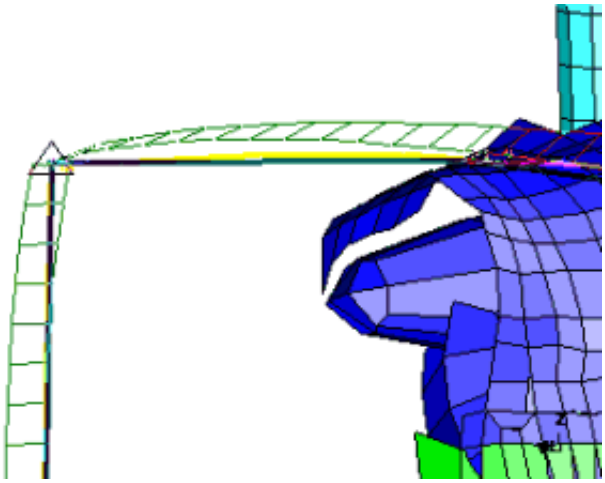
Once the chassis mesh, dimensions and basic parameters are defined you can initiate the form-finding process with the **FIT** command. This loops through steps (1) to (4) as follows:

1. For each point on the chassis mesh a new position is computed that is closer to the dummy structure, and the point is moved to that location.
2. Contact between each point and dummy is checked at this new location, and the point is pushed out again if necessary to prevent any penetrations.
3. Convergence is checked by comparing the current chassis mesh with that at the previous iteration, and form-finding halts if the change is less than the specified convergence tolerance.
4. Form-finding also halts after the user-defined number of iterations so that you can check progress periodically.

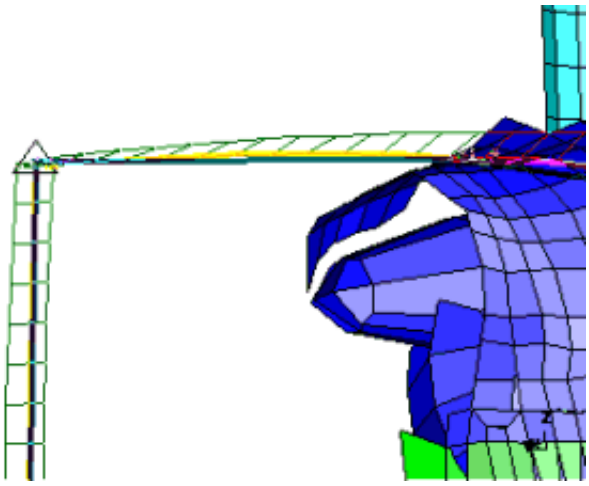
The following is a typical sequence showing fitting from initial position to final convergence: between 100 and 200 iterations is typical for most belts.



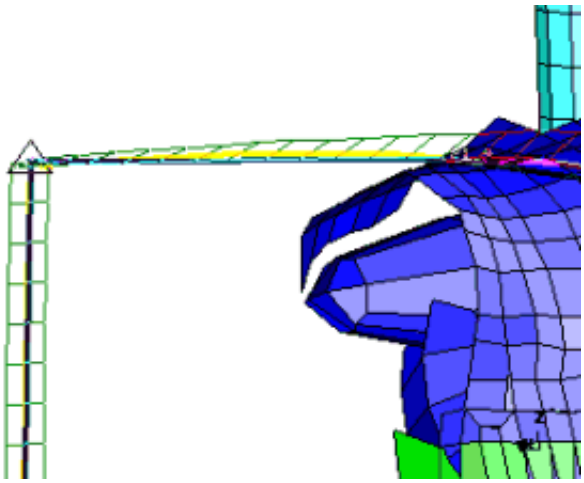
Fitting #0: At initial position



Fitting #1: After 50 iterations



Fitting #2: After 100 iterations



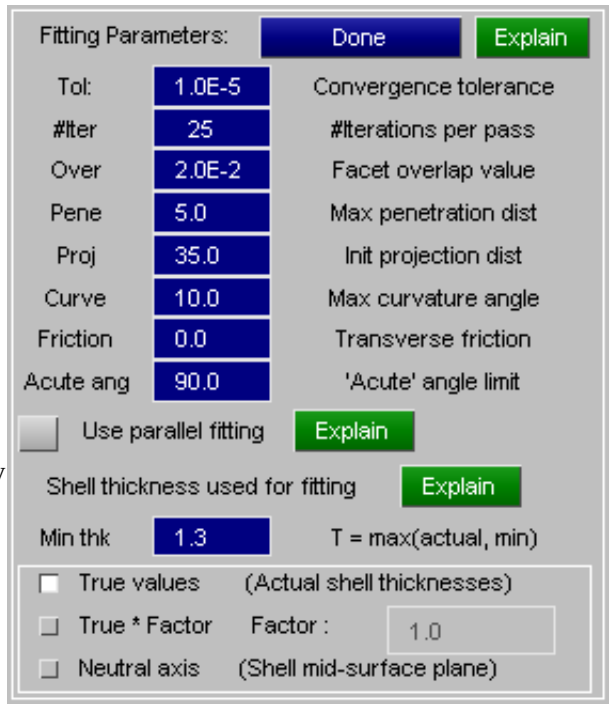
Fitting #1: Converged at 131 iterations

When fitting halts, due either to (3) or (4) above, you can do any of:

- Continue the process from where it stopped with no changes by pressing **FIT** again. It will be necessary to do this several times to get the belt to converge when the number of iterations required is larger than the periodic halt (`#iter`) value.
- Adjust some **PARAMETERS**, then continue from where it stopped by pressing **FIT** again. But make sure that the changed parameters are valid when applied to the current, partially fitted position. If they aren't start again using **RESTART**.
- Adjust **PARAMETERS** and/or **DIMENSIONS** and restart from the initial projected-out position, using the revised values, by pressing **RESTART**.

- Or if you are happy with the converged shape press **ACCEPT** which will:
  - Save the current path, dimensions and fitting parameters permanently in the current seatbelt definition. Prior to this point you were working with a scratch copy.
  - From the main seatbelt menu you can go ahead and mesh the result with actual Finite Elements.

PARAMETERS #2 More about controlling the form-finding process.



Now that the form-finding process itself has been described we can revisit the **PARAMETERS** panel to consider some of its more obscure options.

`Tol` Is the tolerance used to decide when form-finding has converged. The smaller this value the more precise the final shape will be.

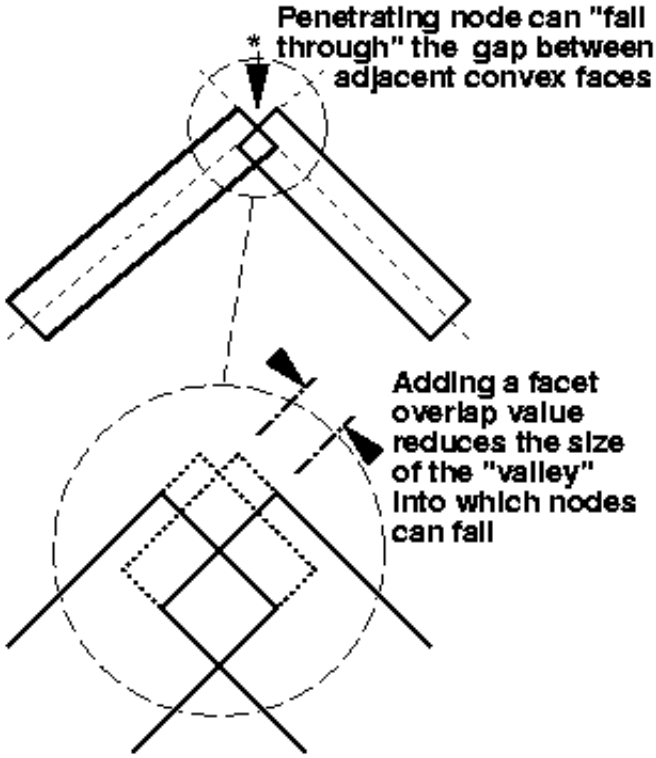
#iter Is the number of iterations that will take place before form-finding pauses to resort the "buckets" used to determine belt to structure contact. The default of 25 iterations is normally adequate, but if you have an extremely fine structure mesh and unwanted penetrations are occurring reducing this value may help.

Over Is the facet overlap value used during contact. It increases the size of structure element facets by (`1.0 +` **Over**) to try to stop nodes falling through the gaps of curved surfaces.

As this figure shows there is a "valley" between adjacent facets on a convex surface into which penetrating nodes can fall.

Artificially increasing the facet size for contact purposes, dotted area, reduces the size of this valley and therefore helps to prevent incorrect penetration.

The default value of 2e-3, (ie the facet dimension is factored by 1.002), works well for most cases; but larger values may be needed if penetration occurs.



Pene Is the maximum distance behind a 3D (solid or thick shell) facet at which penetration is considered.

In this figure the **Pene** distance is shown by the dotted line, and point (b) is therefore too far inside the solid elements to be considered for contact.

Problems can also occur in the shaded areas inside the element corners since a penetrating point can only be ejected from one face, and this may not be the one through which it penetrated.

There is no easy solution to this problem save adjusting the belt path to move the offending point away from the element corner - and this may prove to be a problem during the analysis too. Good (structure) meshing practice would avoid sharp mesh corners where contact occurs by chamfering the edge.

Usually the **Pene** value only needs adjusting when a belt is fitted to a region containing layers of thin 3D elements, when it may need setting to half the thinnest 3D element dimension.

Contact with facets on 3D elements

Point (a) is outside and will not be considered.

Point (b) is inside, and its distance from an external face is greater than *Pene*, so it too is not considered.

Point (c) is inside the *Pene* distance and will be moved to the external surface.

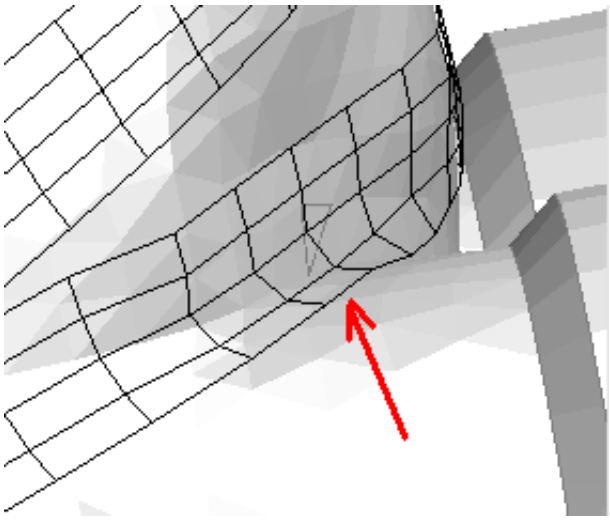
Curve is the maximum permitted transverse curvature of the belt

If this value is non-zero it sets the limiting angle (in degrees) between transversely adjacent belt facets. This can be useful to stop the belt "creasing" down into sharply concave areas of the dummy as shown in the images below.

Treatment of curvature angle in PRIMER versions		
Before V12	Default angle was zero, meaning unconstrained.	This change has been made because the change in default transverse friction coefficient (see below) means that the belt is more likely to slide, and this in turn means that it is more likely to "crease" over on itself.
V12 onwards	Default angle is now 30 degrees.	

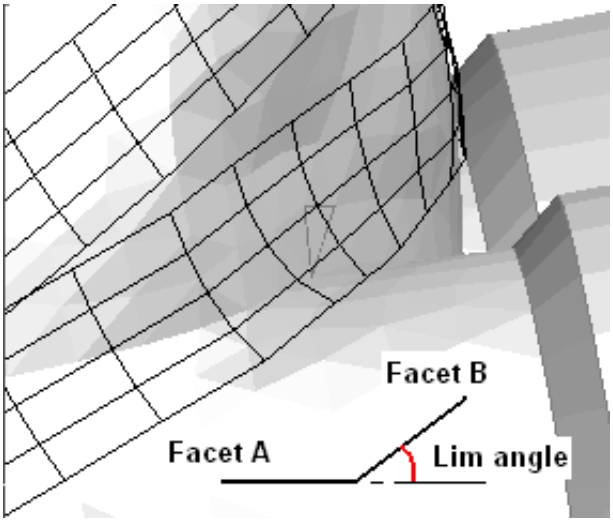
The examples below both show the belt mesh in the region where the lower torso meets the upper leg, in which typical dummies have a concave angle of approximately 90 degrees.

In this example no **Curve** value is set, and the belt creases sharply in the concave region (arrowed). This can sometimes give rise to poor belt element behaviour during the analysis.



Curve = 0 degrees (default)

This example is the same in all respects, except that a **Curve** value of 10 degrees has been used. It can be seen that the belt is now "stiff" in the transverse direction and has not creased down into the concave region of the dummy.



Curve = 10 degrees

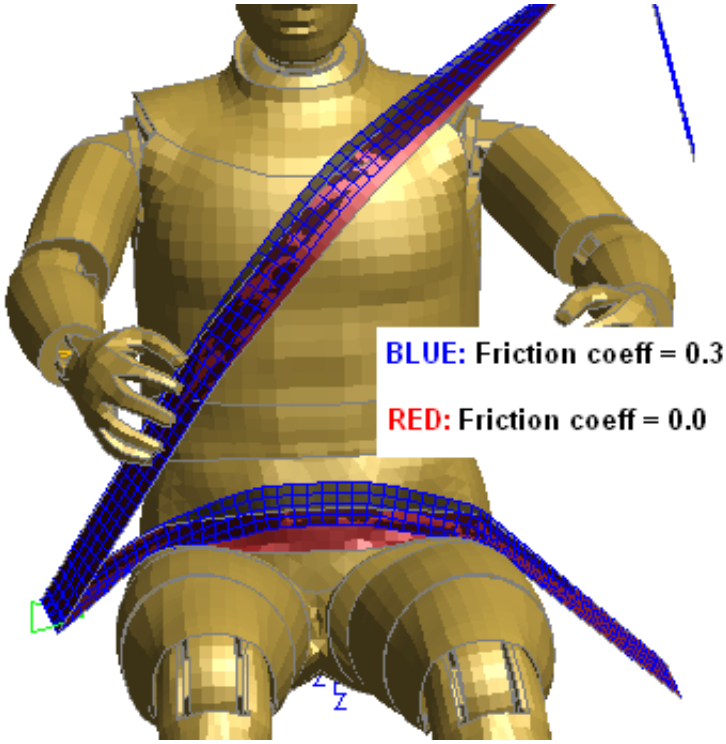
Friction is the transverse friction coefficient between belt and structure.

When the belt makes contact with the structure there will, in real life, be some resistance to sliding and this friction coefficient emulates that. Friction coefficients may lie in the range 0.0 "slippery" to 1.0 "sticky".

The ability to control the friction coefficient is new in PRIMER 12 and behaviour has changed as follows:

Treatment of friction coefficient in PRIMER versions	
Before V12	Coefficient was fixed at ~0.3
V12 onwards	Coefficient may lie in the range 0.0 - 1.0, default = 0.0

The image on the right shows the effect of his change. The blue belt was fitted in V11.1 using the default friction coefficient in that version of ~0.3, the red belt was fitted in V12 using the default coefficient of 0.0. It is clear that the belt has pulled further across the chest and down the pelvis in V12 due to the absence of friction.



This is a change in behaviour that has been requested by our clients, and to revert to approximate V11 fitting behaviour it is recommended that a friction coefficient of ~0.3 is used.

Acute Ang is the critical angle at which a change in direction of the fitting path is considered to be "acute". The path editor will make a break in the belt at acute points, resulting in the mesh across that point not being continuous. The default value is 90 degrees, but users wishing to run continuous mesher around tight bends may need to reduce this value.

The use of this variable is described in Handling "Acute" points.

Use parallel fitting

Prior to PRIMER release 11 belt fitting (form-finding) was a scalar process, and as belts have become more detailed this was taking an unacceptably long time. The bulk of the cpu time was expended in the contact algorithm between belt and structure, and this has been parallelised in release 11. Together with some other internal improvements this has resulted in the fitting time on a typical 4 core machine reducing by a factor of between 2x and 3x.

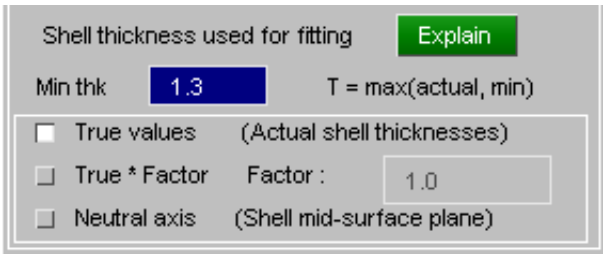
However parallelising the contact algorithm has introduced slight changes to the order in which contact segments are calculated, and since depenetration is based on the sum of a series of small movements this can result in subtly different belt shapes. In cases where the belt path is very sensitive to slight changes of geometry the differences are sometimes quite large.

In recognition of the fact that obtaining a different results from a new release of software may cause problems for some users for whom consistency is more important than speed it is possible to turn parallelised fitting off. This will revert to the "old" scalar algorithm that will produce the same results as earlier releases, albeit more slowly. The default fitting method can be set by the preference:

`primer*belt_parallel_fit:` **`true`** or **`false`**

Users fitting in batch who wish to use the old algorithm will need to set this to **`false`**.

Thickness used for fitting

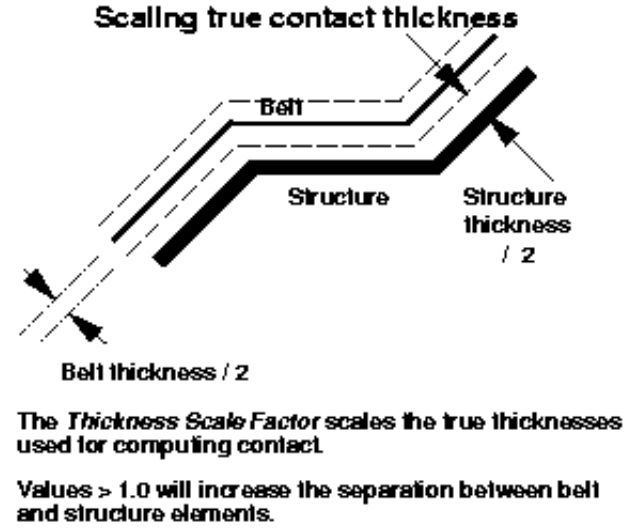


By default the chassis mesh is pulled back onto the structure such that the true external surfaces of belt and structure elements just meet.

The `Thickness` factor scales the true element thickness values allowing you to increase the separation between belt and structure.

This can be useful both for visual purposes and for optimising contact during the actual analysis.

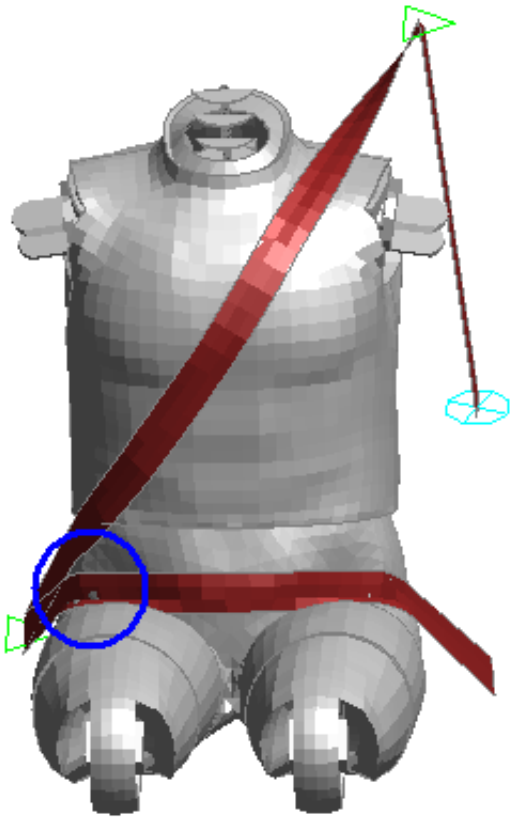
The images below (contrived artificially) show how scaling contact thickness might be used to prevent initial penetrations or - more likely - crossed edges during initialisation.



The **Min Thk** value imposes a lower-bound thickness value to be used for shells, such that the thickness used for contact penetration checking is:

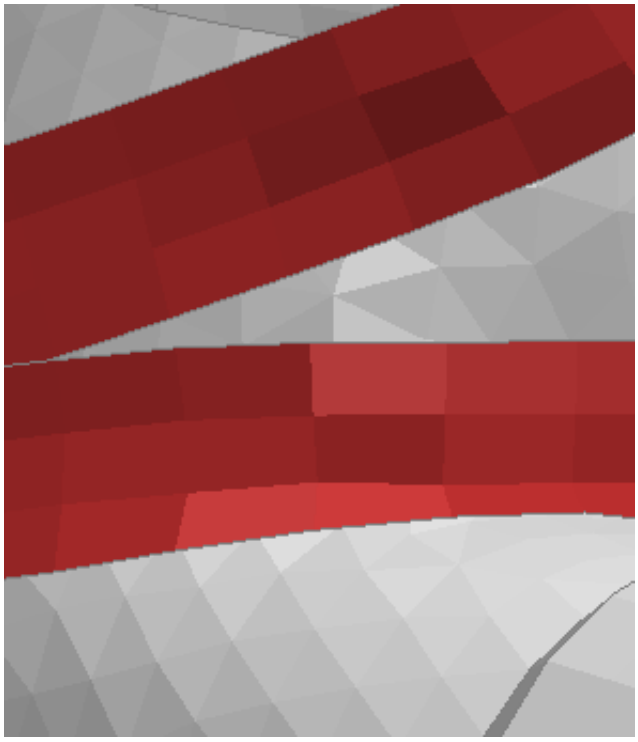
```
t = max(True thickness, Min thickness)
```

This can be useful when dummies have been coated in a layer of null shells, and those shells have a very small thickness, some model builders will set a value of 0.1mm or thinner. Using the true thickness will work, but in order to avoid penetrating to the "wrong" side of the shell neutral axis it imposes a very small "quantum" of movement during fitting, so using a minimum thickness (the default is 1mm) will give much faster convergence.



This image shows the regions of crossed edges in detail.

In this example the belt path has been fitted very tightly, using an unrealistically small contact thickness, leading to some crossed edges in regions of very "bumpy" mesh.



Here the belt has been refitted using a factor of 1.5x the true contact thickness, and the crossed edges in that region have been eliminated.

Detecting and solving problems during fitting

The path gets stuck on the "wrong" side of some structure.

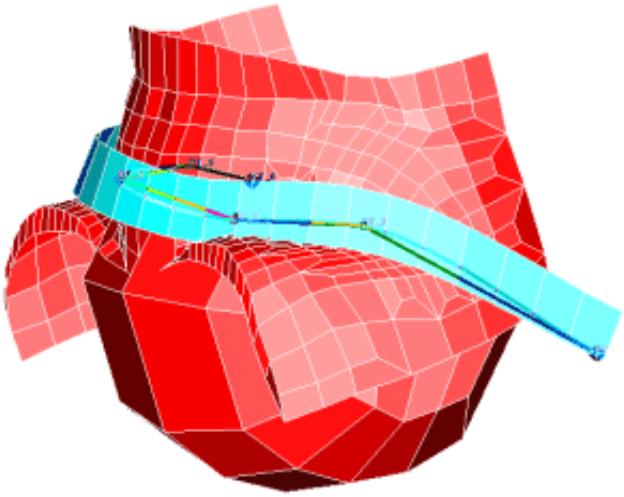
This is usually caused by the basic path not being defined correctly and having initial penetrations. It can be solved by moving path points or possibly adding new ones - see the section on "Defining the initial path" for more advice.

It is also sometimes caused because the route the belt takes when being pulled in is not the same as the outwards projection, and this leads to it getting snagged on intervening structure. Also it may be that not enough "structure" has been defined leaving a gap for the belt to "fall through".

In this (contrived) example of fitting to a pelvis the belt above the left leg penetrates the upper leg structure.

This is caused by the 3rd path point being too low down causing the lower edge of the belt path to be on the wrong side of the upper leg structure.

Moving this point up (**PATH...**, **MODIFY**), then re-fitting the belt will fix the problem.



I want to mesh chest and lap belts using two different element properties

If you define the whole belt, chest and lap, in one operation then it will only be possible to mesh it using a single property. Also the nodes at fixed point locations, eg sliprings, will be common to their adjacent segments giving continuity across sliprings.

To get separate properties it will be necessary to create two separate belt definitions: one for the chest section and one for the lap. These will then be able to have different properties defined during their meshing operations.

By default this will give separate nodes at end points, thus losing the continuity of the belt through sliprings. However the **MESHING** operation permits you to stipulate the end nodes to be used, so if continuity is required the second section to be meshed can still be made to join up with the end of the first one.

There is no limit to the number of belt definitions that can be defined in a model, and they may be joined up as required.

This example shows how a simple four-point harness could be generated using four separate belt definitions: left and right sides, upper and lower cross-sections. Two different section properties are used.

Fitting options.

In PRIMER V13 a new Fitting Options panel was added. This combines existing settings that were previously a bit hard to get at during belt fitting with some new capabilities.

It is mapped automatically when 2:Fit mode is entered, and it is recommended that it is moved to one side of the screen, or possibly onto the desktop,so that it does not obscure the dummy during belt path creating and subsequent fitting.

If it is dismissed it can be mapped again using the **Tools...** button in the path editor, or the **Fitting Options** button on the main panel.

This section goes through every capability on the panel, some of which has already been covered in the sections above. These will be repeated in brief with references to the earlier sections.

Quick links to the various sections:

Preset fitting modes

Path order | Path drawn as | Showing

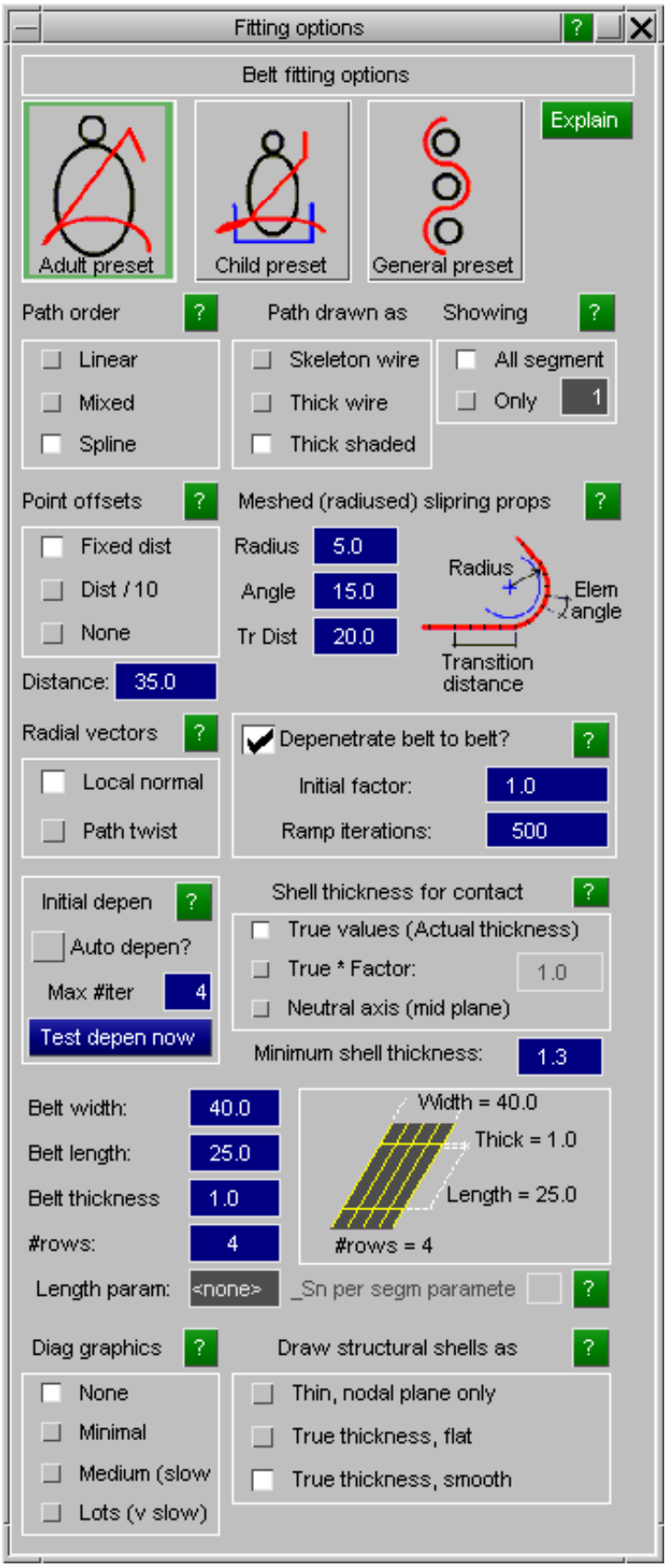
Point offsets | Meshed sliprings

Radial vectors | Depenetrate belt to belt

Initial depenetration | Shell thickness for contact

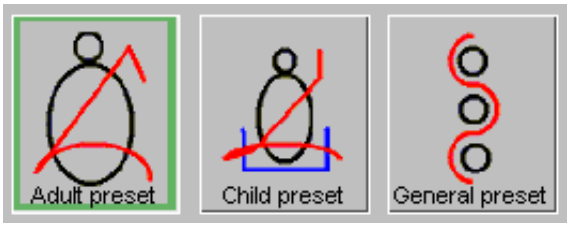
Belt dimensions | Length parameter

Diagnostic graphics | Draw structural shells as



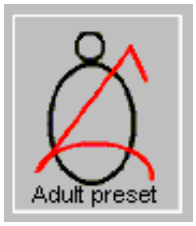
Fitting Modes:

Pre-configuring settings in this panel.



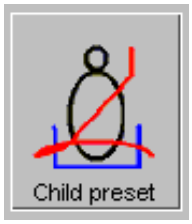
Important: These different modes do ***not*** change how belt fitting works, rather they preconfigure various settings in this options panel to defaults that are most likely to be appropriate to the task in hand. You do ***not*** need to use a particular mode for a particular purpose, use whatever works for you and your particular geometry - don't be afraid to experiment!

The currently selected mode is highlighted in green (adult mode in the image above) but changing any of the individual settings to something else will cancel highlighting.



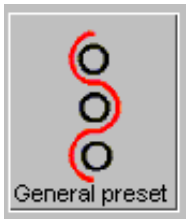
Adult preset mode is intended for the traditional case of "fitting a rubber band around an egg". The curvature of each segment of belt path will be simple, so it can be projected forwards and then pulled back onto the dummy structure. This presets the following:

Path order	Spline
Projection distance	Fixed distance, ie 100% of specified value
Radial vectors	Local normal, ie based on normal of nearby elements.
Initial depenetration	Not used
Belt-to-belt depenetration	Turned on



Child preset mode is intended for the more difficult case of fitting a belt to a child in a booster seat. The curvature of path segments can be complex, and they may be required to fit around obstacles and through gaps in a way that makes "Project forwards and then pull back" simply impossible. This preset the following:

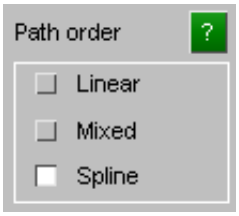
Path order	Spline
Projection distance	Fixed distance / 10, ie 10% of specified value
Radial vectors	Path twist, based solely on path shape
Initial depenetration	Turned on
Belt-to-belt depenetration	Turned on



General preset mode is intended for the entirely general case where no assumptions can be made about geometry. Fitting a belt between rollers might be one example, but really anything which is not a dummy. This presets the following:

Path order	Linear
Projection distance	None. No projection of path
Radial vectors	Path twist, based solely on path shape
Initial depenetration	Turned off
Belt-to-belt depenetration	Turned off

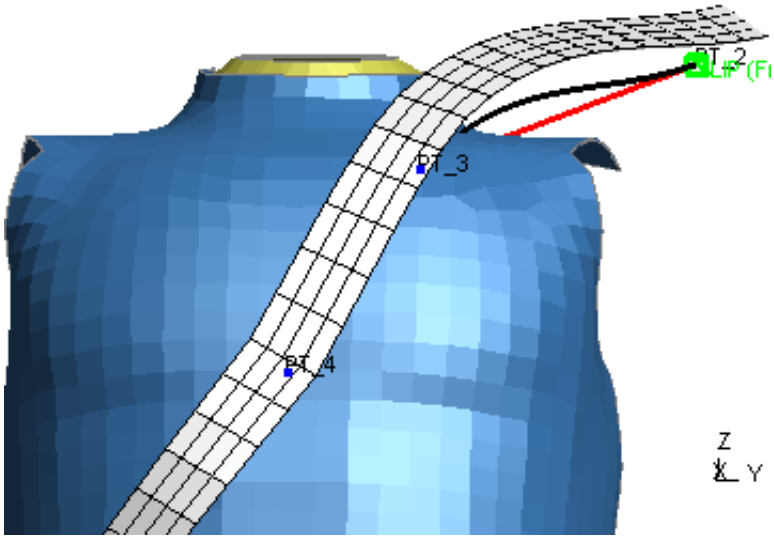
Path order



Controls the interpolation method used to generate the fitting path between the basic path points.

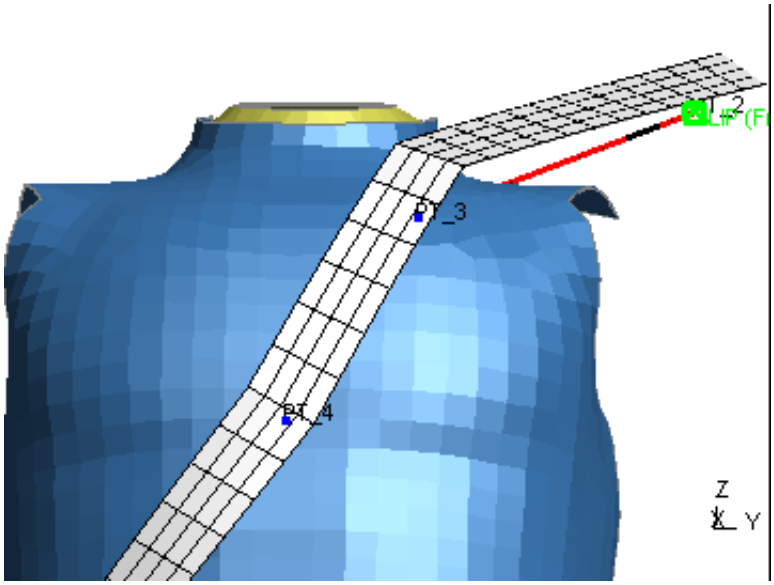
Cubic Spline fits a continuous curve through points.

This is generally best for curved geometry such as dummies, it also gives a clear indication of the "outwards" direction used when projecting the path forwards.



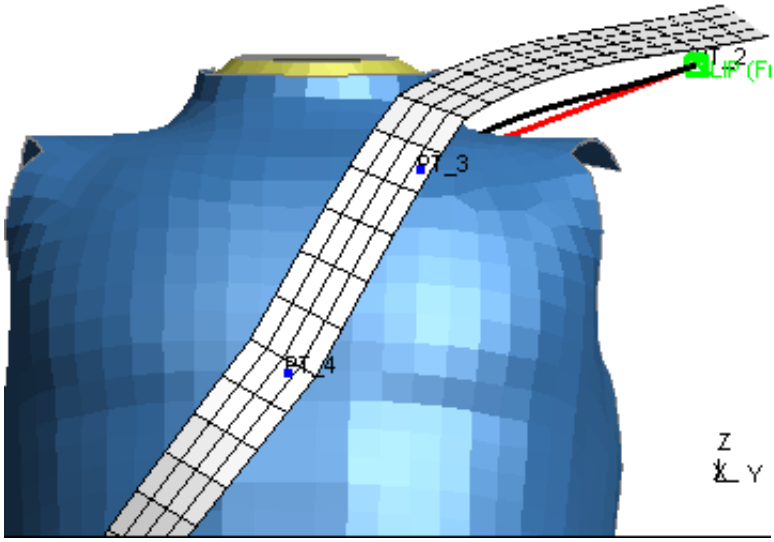
Linear fits straight lines between basic path points.

This is more likely to work well for arbitrary geometries made up of a series of straight sections, perhaps fitting a belt between rollers.

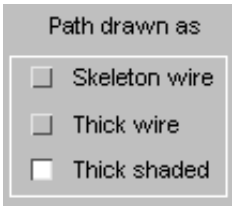


Mixed is the average of **Spline** and **Linear** shapes.

This may work well in mildly curved situations



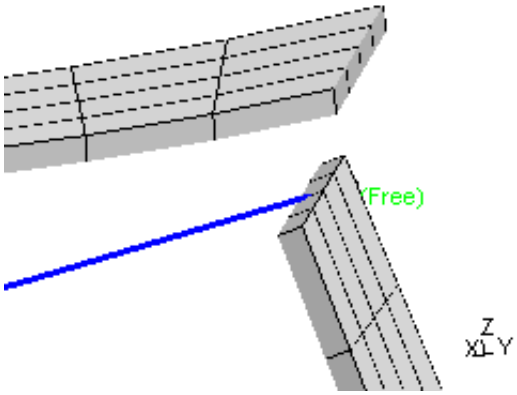
Path Drawn as



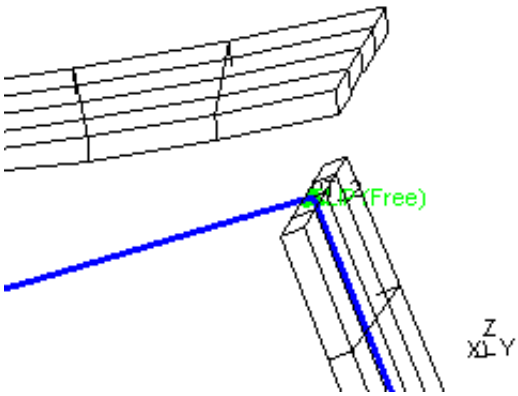
Controls how the fitting path is drawn.

All the images below show the belt path near the shoulder slipring. The belt thickness has been made artificially large to demonstrate the various modes more clearly.

Thick shaded draws the belt path in fully shaded and hidden modes, showing its actual thickness.

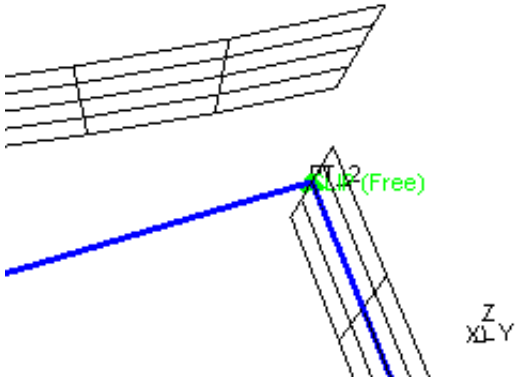


Thick Wireframe draws the belt path in wireframe (lines, see-through) mode, showing its actyual thickness

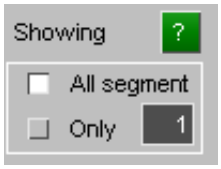


Skeleton Wireframe draws the belt path in wireframe, at its centreline location with no thickness.

(Prior to V13 this was the only display mode available).



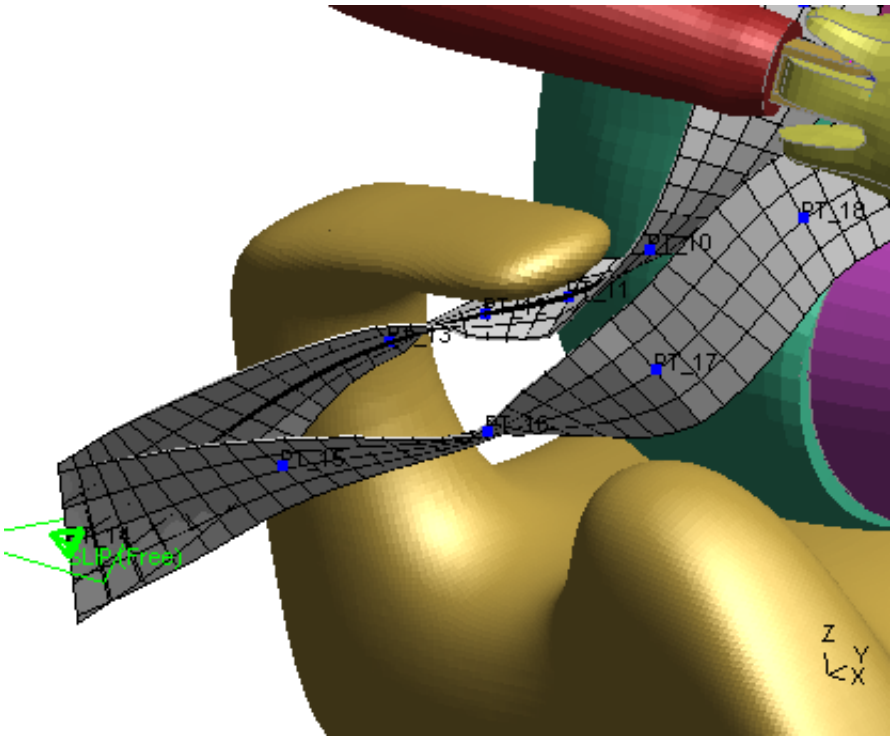
Showing (path segments)



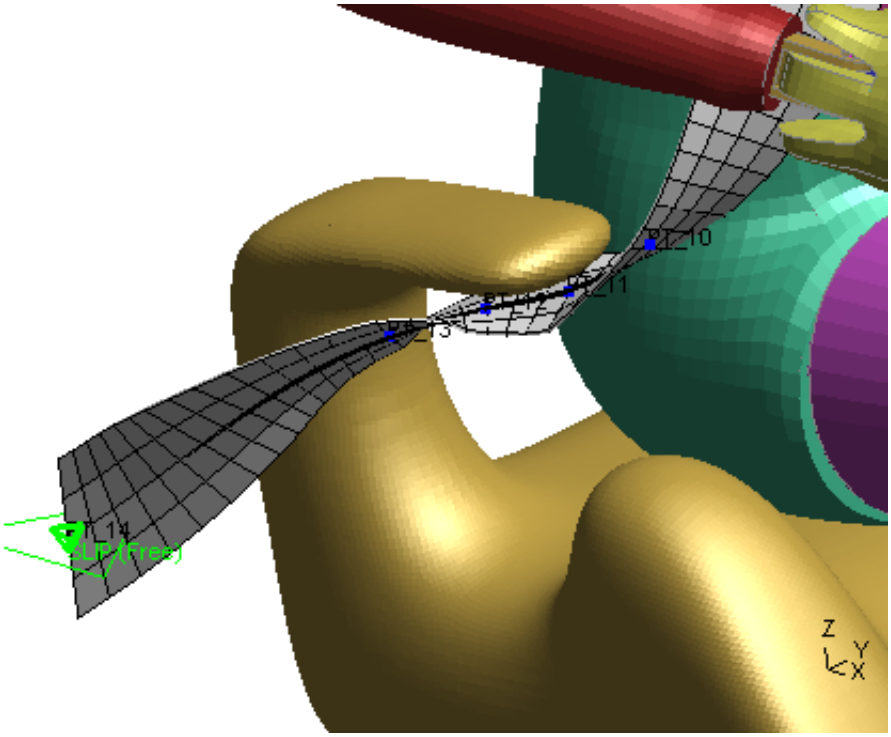
Controls whether the complete belt path is drawn, or only a single segment of it.

All segments is the default case, in which the whole belt path is drawn.

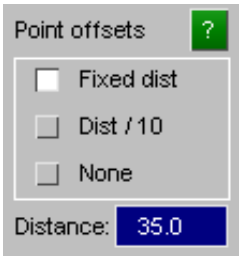
When defining a path for complex geometries, such as the detail around the "wing" of the child seat shown here, it can sometimes help if only the segment of interest is shown, since other segments may get in the way.



Showing **Only** one segment, here the first one, can make it easier to see what you are doing.



Point offsets



Controlling whether and by how much the belt path is projected forwards from the base path.

Note: From PRIMER V14 onwards it is possible to control offsets on a per-point basis. See **Control Projection mode**.

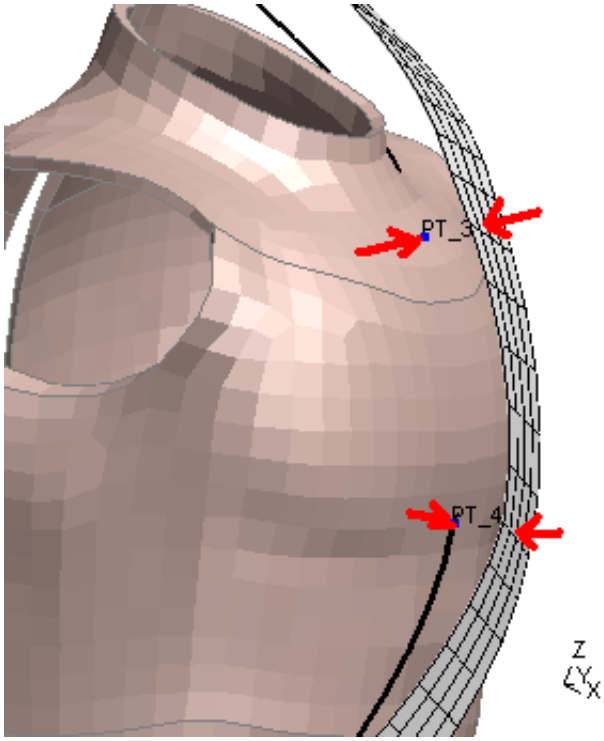
Fixed distance - suitable for adult dummies.

This example shows how fitting a belt to an adult dummy is like "fitting a rubber band around an egg".

The underlying path (black line) is projected outwards at each point by the projection distance when creating the meshing path, shown here by the distance between the red arrows, and then pulled back in during form-finding.

This permits basic path points to be located at nodes, despite the fact that these nodes, which are at the neutral axis of shells, are "inside" the dummy structure.

Not only does this make defining path points easy, but when the dummy is repositioned the path points can move with the nodes making it easy to refit the belt automatically.



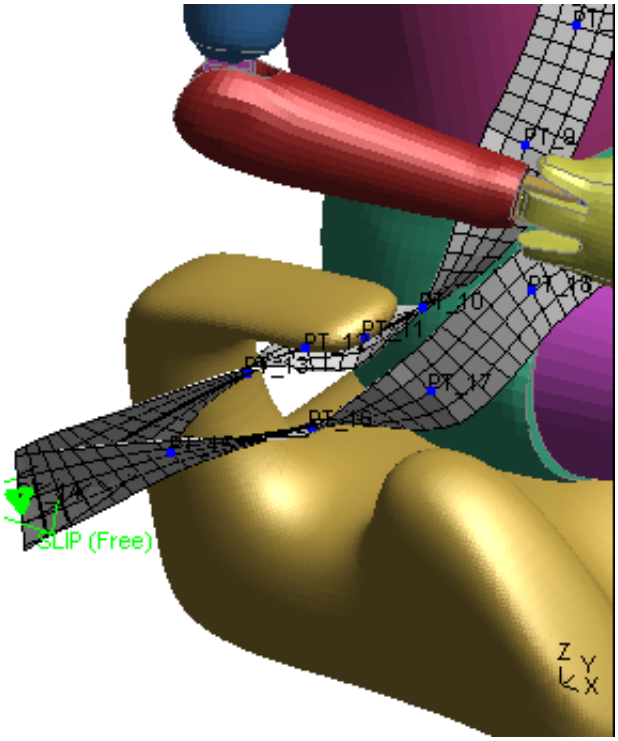
Distance / 10 - suitable for child dummies.
None - also suitable for child dummies.

For child dummy where the belt path is likely to have to thread through hole, may have reverse curvature, and will usually have a complex shape the "rubber band around an egg" modelling method no longer works.

In these cases you either want a very small projection, hence the "distance / 10", or perhaps no projection at all.

The choice will depend on the geometry, but generally the more complex the shape the less likely projection is to be a help when fitting a belt path.

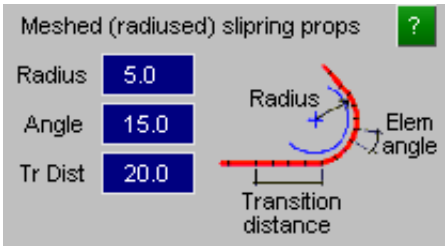
Fitting a belt path to a child will almost certainly require you to adjust most of the basic path points, breaking their association with any underlying node used to pick them originally. This makes projection less valuable since any movement of the dummy prior to a belt refit will not be "known" about at points which are not at nodes.



Distance - setting the projection distance.

This defaults to 35mm, but you can set any distance you like - including zero. This setting is the same as that in the Fitting Parameters panel, it makes no difference where you define it. It is duplicated on this options panel for convenience during fitting.

Meshed (radiused) sliprings



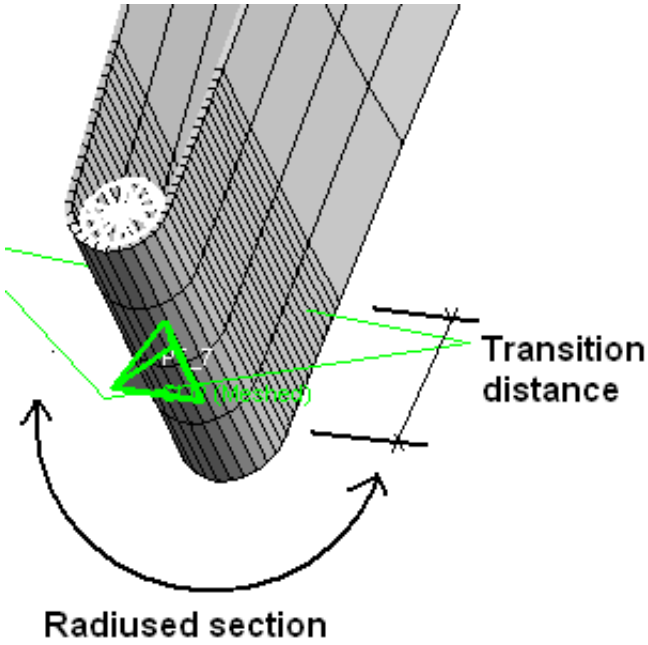
As an alternative to using the *ELEMENT_SEATBELT_SLIPRING element you can mesh the belt path explicitly around a tight radius at slipring locations.

This process has been described in some detail as example 3 in the creation of sliprings above, so this is just a summary.

A meshed slipring has three key attributes:

- A radius around which it must fit.
- A maximum angle which the "short" elements going around the radiused section are allowed to subtend. This is because the elements must be reasonably short in order to fit smoothly around a tight radius in order to avoid a "ratchet" effect (like pulling a bicycle chain around a rod).
- A transition distance either side of the slipring over which these short elements extend. This is so that material pulled around the slipring during the analysis also avoids any ratchet effect.

The image here shows a fitted slipring that exhibits these characteristics. It uses the default settings shown on the options panel.

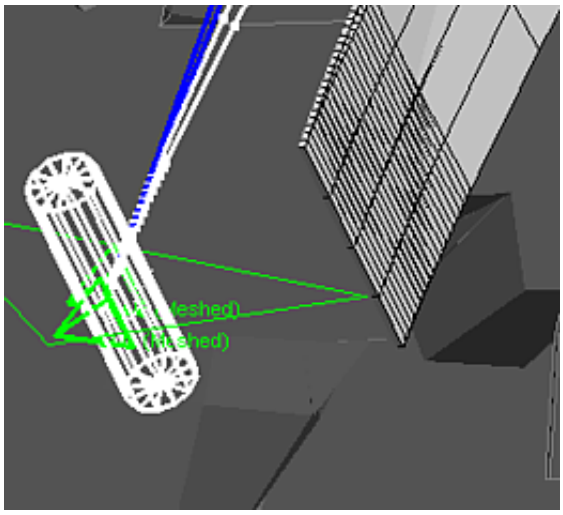


Things to consider when using meshed sliprings:

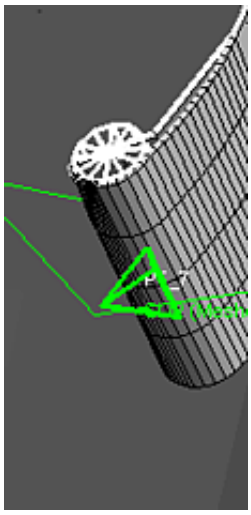
(1) The belt path will have to pull back from "in front of" the slipring (its offset) down onto it, and the inside segment of path will pull through it.

The centre image below shows the fitting process partially complete with the inside of the belt still pulling through.

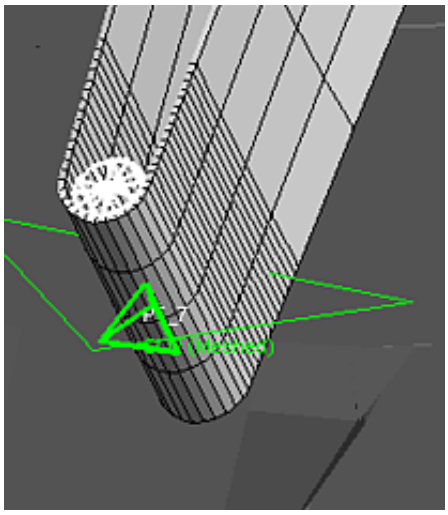
This also means that if the belt buckle structure has been meshed it must not be included in the "structure" definition for belt fitting, otherwise the belt path will not pull through it but rather get stuck in front of it. (The cylinder at the centre of the slipring in the images below is drawn by the belt fitter, it has no physical existence.)



The belt path will start off "outside" the slipring, being projected outwards just like the rest of the path.



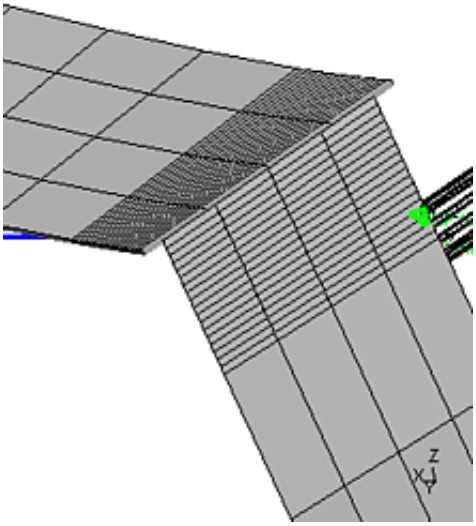
It will pull onto the slipring, and the "inside" will pull through it.



The final shape should wrap round the slipring correctly, but be careful that you have aligned it correctly - see below.

(2) The two sections of path meeting at the slipring need to be lined up correctly.

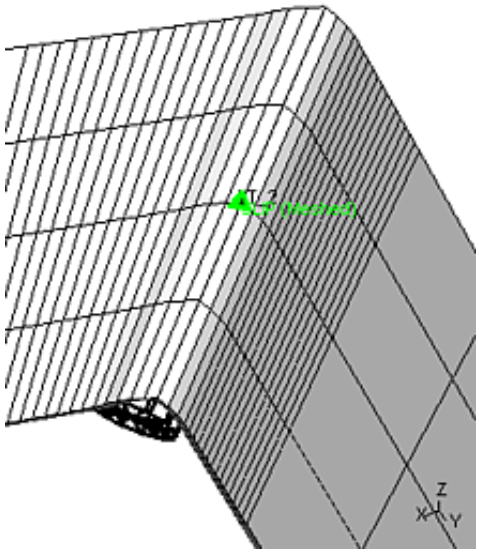
The left hand image below shows a shoulder slipring detail where the two stretches of belt have deliberately been twisted so that they do not line up, showing how the widths are different. The right hand image shows what happens if this detail is fitted without correction, and anyone who has ever performed any carpentry will recognise the problem.



Shoulder slipring detail with the two sections deliberately mis-aligned, giving different widths

When joining two pieces of wood at an angle the angle (red line) must be chosen so that it bisects the two vectors, otherwise the widths of the two pieces will not be the same at the join.

Seatbelts are the same!

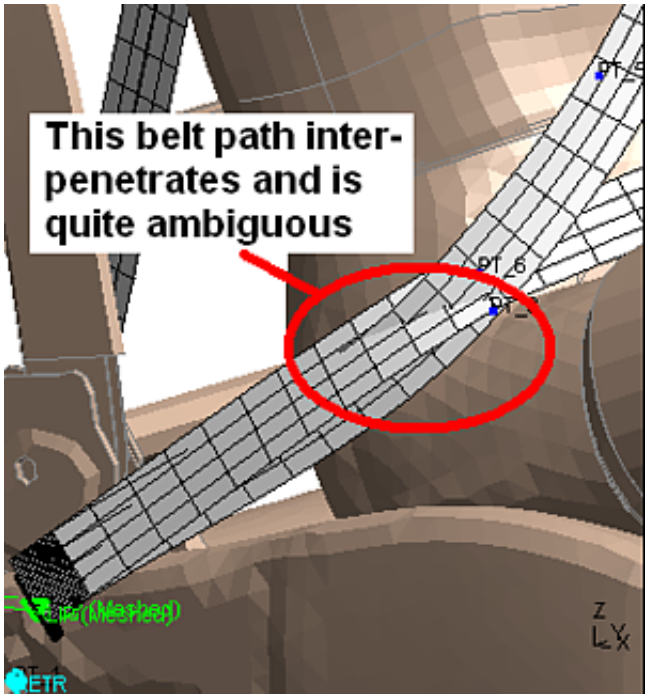


What happens when the detail on the left is "fitted". The right hand section is narrower than the left leading to a transition in width as it crosses the slipring.

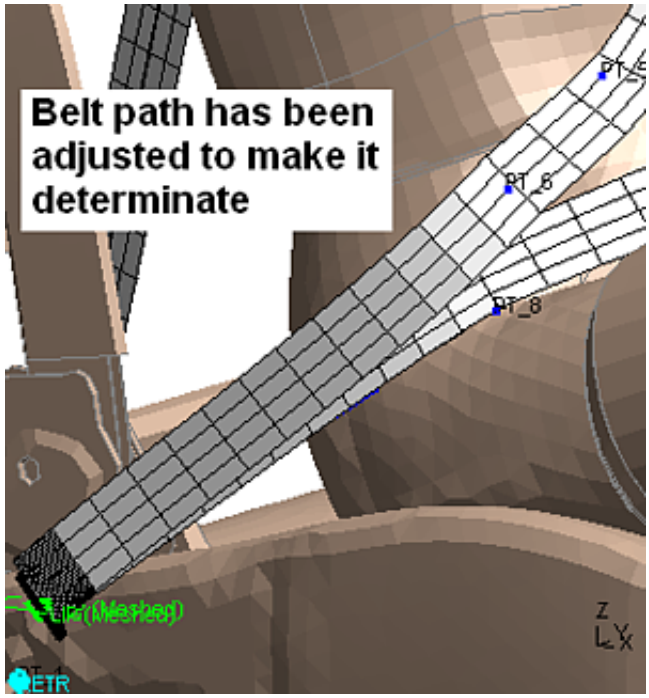
When using meshed sliprings (or free ones for that matter) it is important to make sure that they are lined up correctly so that the angle of the slipring bisects the two incoming paths properly. Sometimes this is a process of trial and error since fitting may pull the paths to different angles meaning that something which was correct in the unfitted shape no longer lines up in the as-fitted state.

(3) It will also be evident from the images above that a meshed slipring requires an unambiguous definition of "in front" and "behind" belt paths. During fitting the "behind" segment must pass through the slipring while the "in front" segment stays on the outside.

Meshed sliprings use special logic to determine which belt segment is which in their immediate vicinity. It examines each segment of the belt path to see which has more points "in front of" the other, where the "in front" direction is the outwards vector along which the belt path is projected prior to fitting, and the one with more points in front is defined as the "in front" segment. Therefore you should aim to create an initial belt path that does not interpenetrate near these slipring locations, since this gives the algorithm the best chance of determining the belt shape correctly.



A poorly defined initial belt path

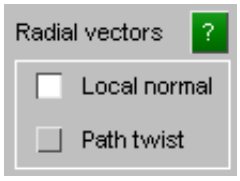


Path has been adjusted manually to make its shape clearer

This is good modelling practice for all sliprings, including those which use *ELEMENT_SEATBELT_SLIPRING, but it is especially important for meshed ones since they must separate the belt by a distance equal to the diameter of the buckle.

The situation on the left may also be resolved by the initial depenetration logic described below. This is still experimental, and may fail, therefore a better solution is to adjust the belt manually to give a clear result as in the right hand image above.

Radial vectors

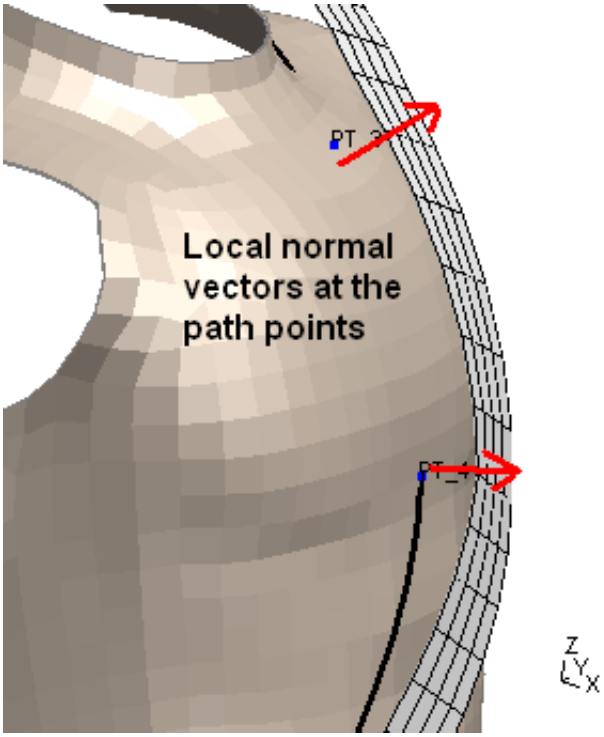


Controls how the default "twist" of the belt path is computed.

Local Normal is suitable for gently curved shapes where the belt path will not reverse curvature in a segment. This is the typical adult dummy case.

The outward normal of the elements at the belt path points is computed (red arrows here), and is used to define the outward normal of the belt path at that point.

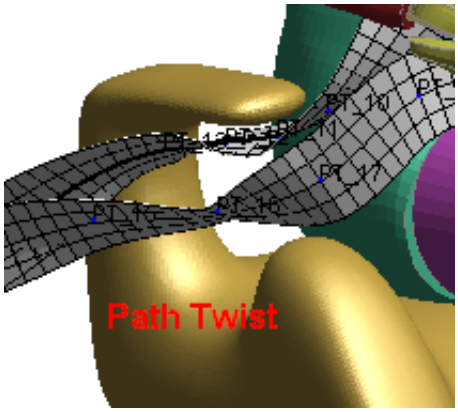
Path points not on a node find the nearest node, or if there is none they interpolate the normals from adjacent points.

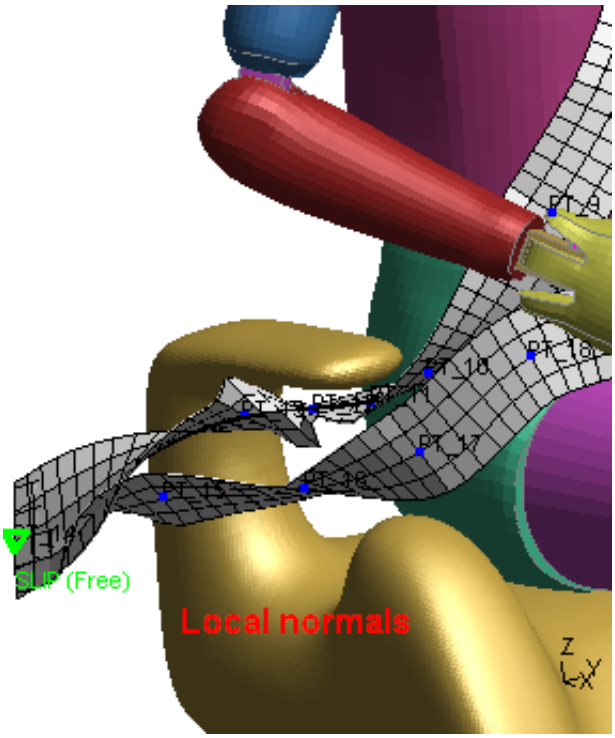


Attempting to use **Local normal** for cases where the belt path reverses curvature tend to be disastrous!

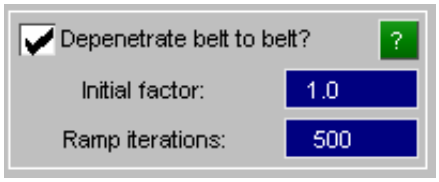
The section of belt path going under the wing of the seat in this image illustrates the problem, with sharp reversals of curvature. It is easy to see why it is going wrong.

Reverting to **Path twist**, which ignores local structure, gives the much better result below in that region.





Depenetrate belt to belt



Whether or not contact between adjacent belt paths is considered. Also an optional increase in belt thickness in the early stages of form-finding.

In most regions of the belt there is not likely to be any contact between adjacent segments of belt, but close to the pelvis slipring that may not be the case as this example shows. In these regions it may be necessary to consider belt-to-belt contact during form-finding, but that is not always the case as there are pros and cons to doing this.

Why you might turn on belt to belt contact during fitting:

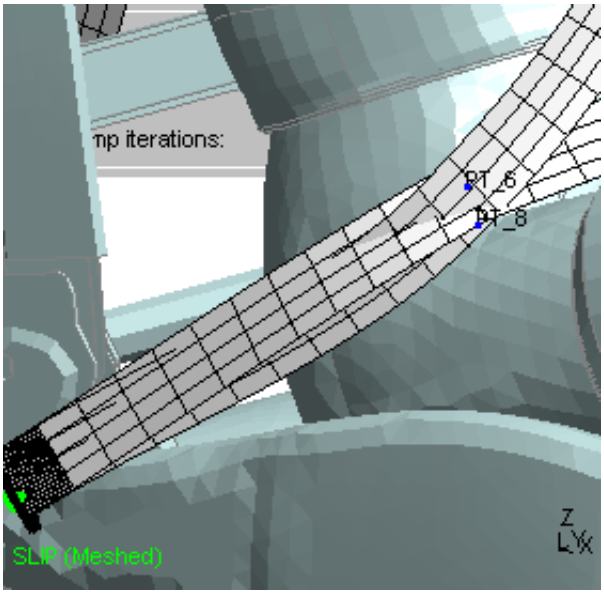
- The belt paths genuinely overlap in the as-fitted shape, and it is important to avoid initial penetrations at the start of the analysis.

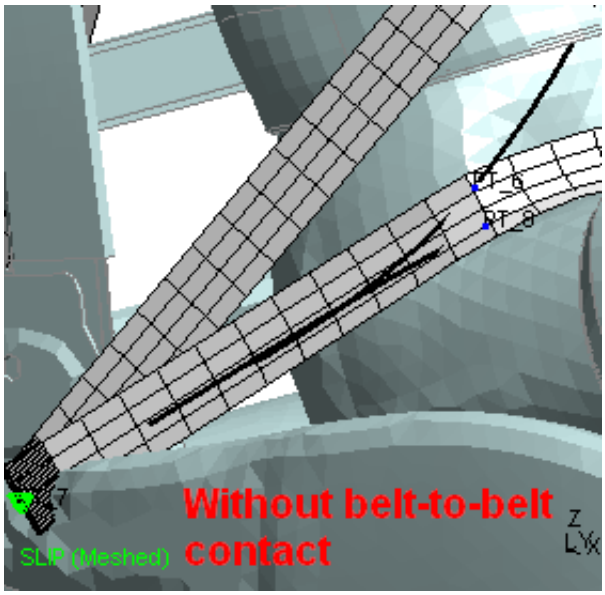
Why you might choose to ignore belt-to-belt contact during fitting:

- The initial shape has inter-penetrations (as in the example to the right here), but in the final as-fitted shape the two paths will not overlap. Trying to resolve an initial geometry like this may well cause the contact algorithms to go wrong.

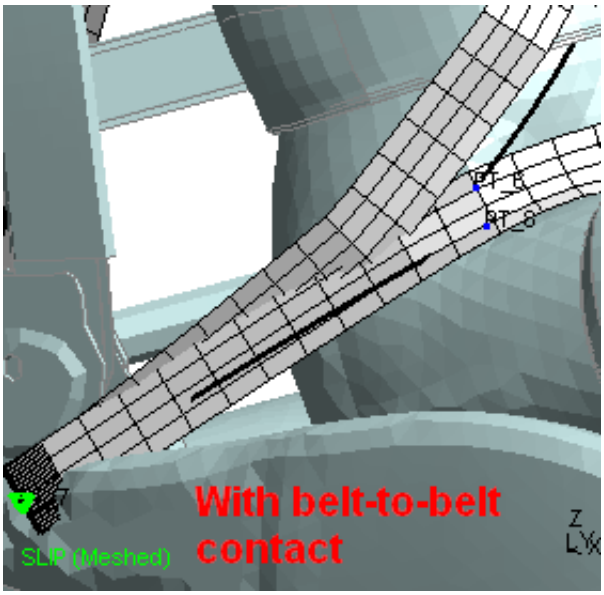
- Belt-to-belt contact is expensive to compute and slows down fitting. There is no point doing it if it is not needed to achieve the correct end result.

So was it necessary here? This is what happened when this geometry was fitted.





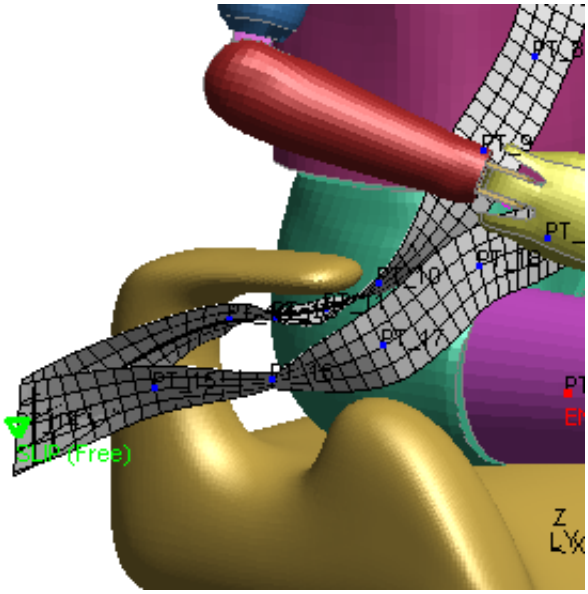
Result with ***no*** belt-to-belt contact, the two paths have pulled through one another because they didn't "see" one another for contact, and have reached a satisfactory final shape



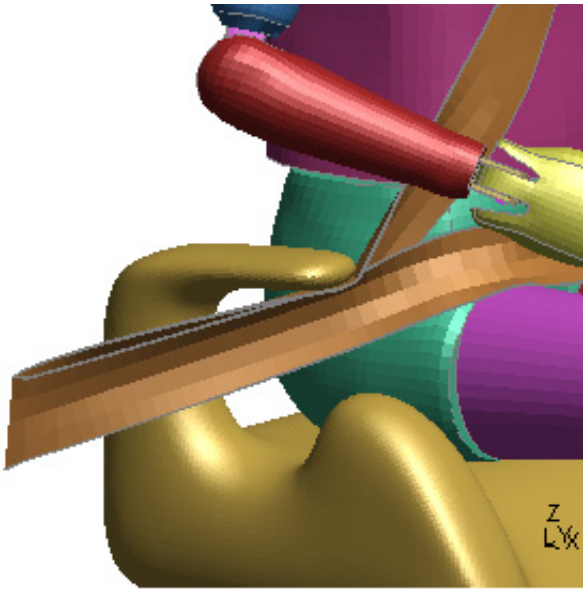
Result ***with*** belt-to-belt contact. The belt path was badly set up to start with, with the two segments overlapping and penetrating, so it has "got stuck" when trying to sort itself out.

Clearly this example is contrived: the initial shape is a poorly configured belt path that is simply asking for trouble. However it illustrates the point that belt-to-belt contact may not be necessary in many cases, but where it ****is**** necessary you need to make sure that your belt path has no initial penetrations, and also that it will "pull" (ie form-find) to the final position in a sensible way.

Here is another example where belt-to-belt contact is definitely required. It is clear that the chest and pelvis sections of belt will lie on top of one another in the final shape as they pass under the seat wing, so belt-to-belt contact cannot be ignored and it has been turned on.



Before fitting: note belt path is well sorted out, with no inter-penetrations and a clear route its final shape.



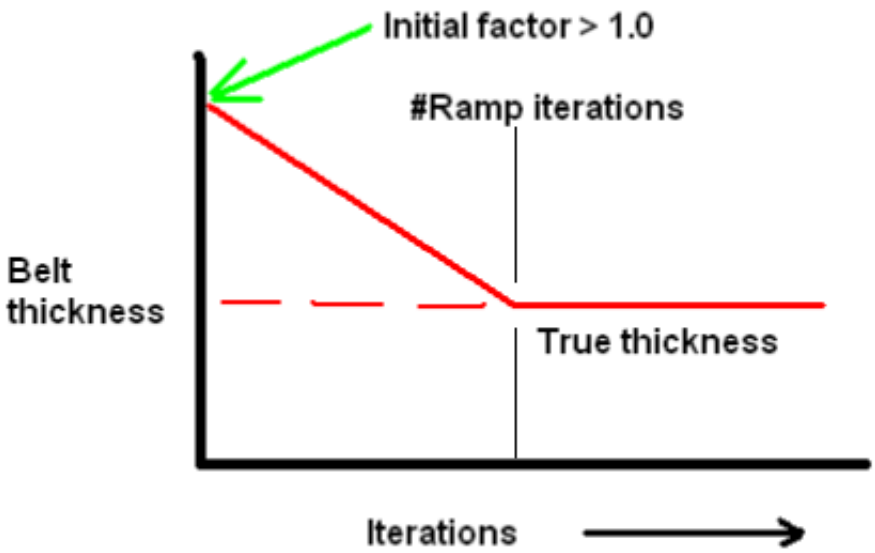
Final as-fitted belt path

Optional initial factor on belt thickness.

Normally the thickness used for belt-to-belt contact is the true thickness of the belt, however experience has shown that it is sometimes useful to increase that thickness in the early stages of fitting, since this may give a better result in areas of complicated geometry. Therefore you can define:

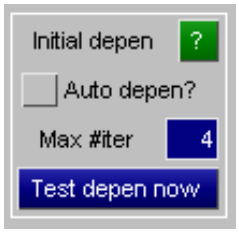
Initial factor	Factor on thickness
Ramp iterations	The number of fitting iterations over which this reverts linearly back to a factor of 1.0

This can be expressed graphically as follows



If you have problems with the belt sorting itself out in an area of complicated geometry this may provide a solution, it is certainly worth experimenting with. If this factor is not defined it defaults to 1.0, and no thickness scaling takes place.

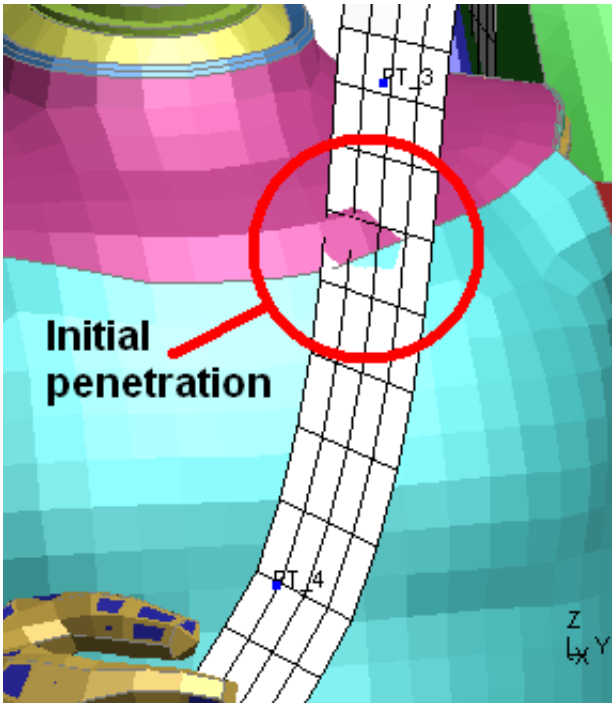
Initial depenetration.



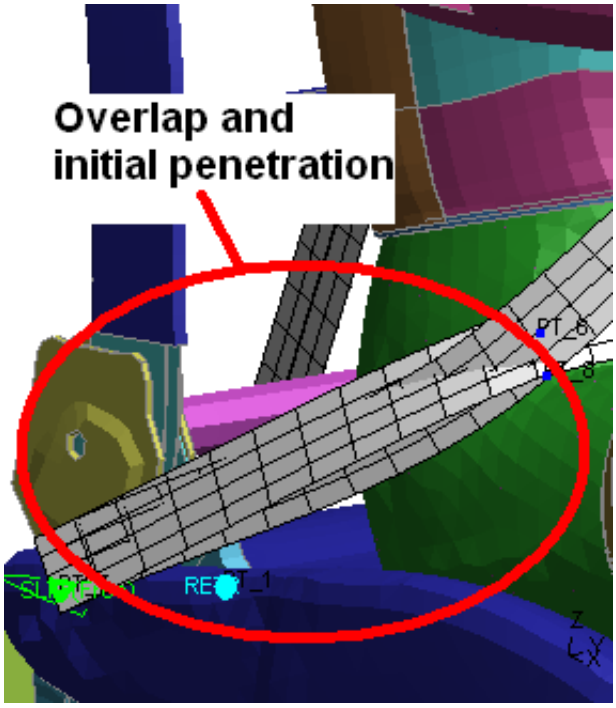
This feature is experimental, and still under development.

Initial depenetration can be used to sort out minor path definition errors, however it can go wrong (see How and why automatic depenetration fails, and what to do about it below) and it should not be relied on always to give a good answer. Adjusting the belt path manually to resolve errors is more likely to give a good end result. However in batch fitting, or in an automated process where manual intervention is not practical it may help to improve robustness.‘ We will continue to develop this feature.

One problem when defining a basic belt path is that the path may accidentally penetrate the structure resulting in it getting stuck on the "wrong" side of shells during fitting because of the way the contact algorithms work. Another problem is that in regions where belt paths converge, typically near a pelvis slipring, the segments of belt path may overlap and inter-penetrate each other. Examples of these problems are shown below:

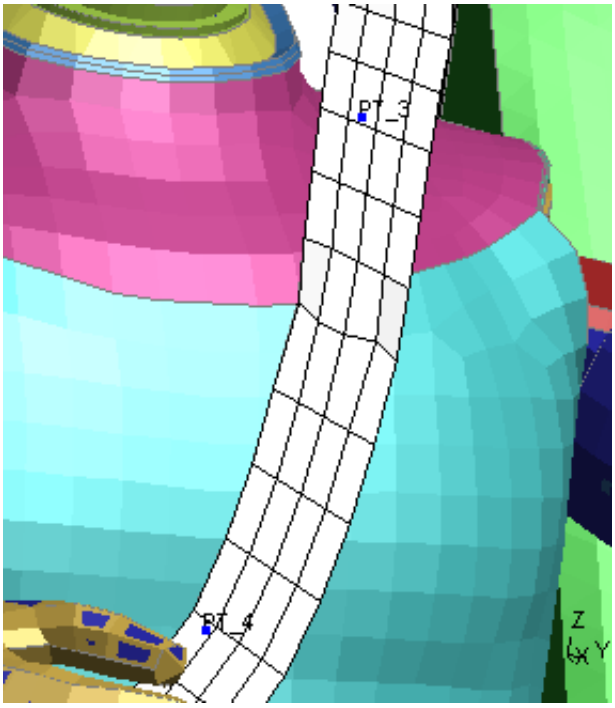


The belt path penetrates the chest of the dummy

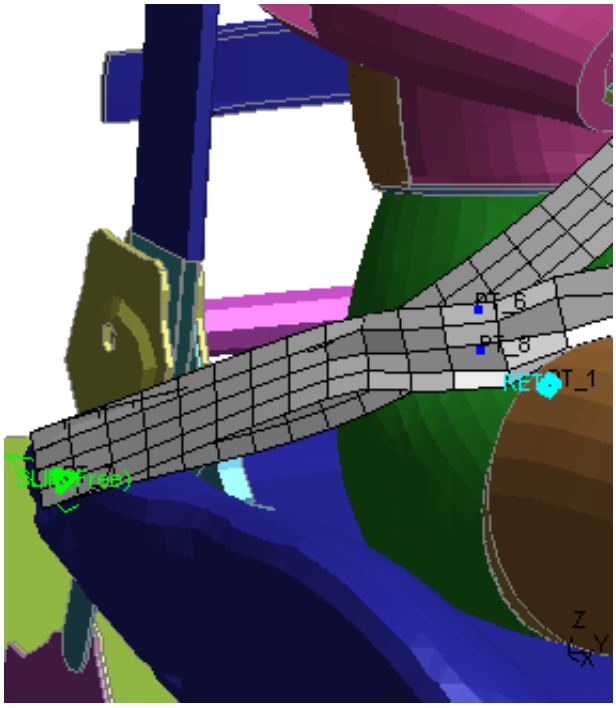


The belt path overlaps and self-intersects

Initial depenetration can - sometimes - sort out these problems. The images below show what happens when it is run on the two problems above.



The belt has been ejected from the chest.



The belt path has been sorted out.

In both cases the depenetrated result looks quite "bumpy" and has distorted the belt path. Don’t worry about this: depenetration does not consider the physics of the belt, but form-finding (fitting) does and it will pull the belt back into a reasonable shape.

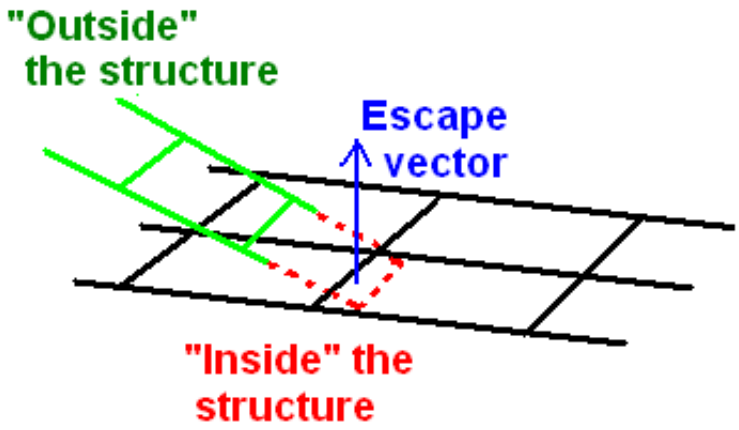
The two sorts of depenetration here are two separate problems that are tackled in two different but related ways:

(1) Depenetrating the belt from the structure.

This is achieved by finding a start point on the belt that is definitely "outside" the structure, green in this image, then walking down the belt in small increments of distance until the path moves to "inside" the structure (red).

Because the distinction between outside and inside is well defined it is possible to identify the depenetration "escape" vector (blue) along which it is necessary to move this region of belt path in order to depenetrate it.

This technique tends to work reasonably well where the belt "dips into" the structure as in the example above. It works less well for penetrations along the edge of the belt where it has "cut sideways" into the structure, because in these situations the escape vector is harder to define.



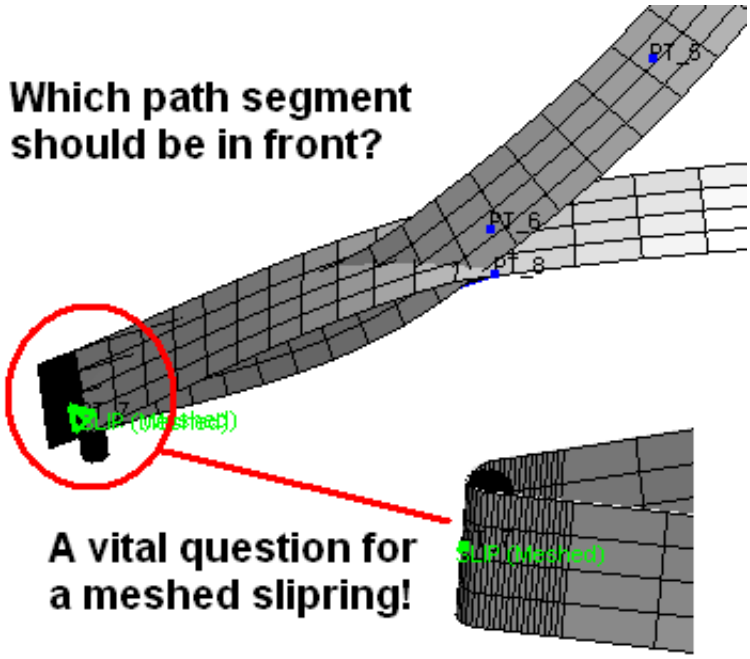
(2) Sorting out overlapping regions of belt.

This problem is not so much outside vs inside, but rather "in front" vs "behind".

The depenetration algorithms first have to work out which segment should be in front of the other, and they do this by majority voting in the region of interest. Each point is tested to see if it is behind or in front of the other segment of belt, and the segment that has most "in front" points is then required to be in front along the whole of its length, the other being forced to be all behind.

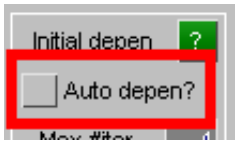
Once this priority has been determined then logic similar to the outside vs inside described above can be used to depenetrate the two segments.

This process is particularly necessary where meshed sliprings are used, since there cannot be any ambiguity about which belt segment is to be on which side of the slipring cylinder. Even if initial depenetration is not used the belt near these sliprings is still processed through the majority voting scheme described above in order to resolve this question.



How to use initial depenetration.

(1) Auto Depen: An initial pass before form-finding



If **Auto Depen** is selected then prior to a **Fit** operation depenetration is applied automatically.

(2) Test Depen Now: Immediate depenetration at any time.

You can run the depenetration test at any time during path creation and editing by using **Test Depen Now**. It will leave the path in a modified state, and to restore it to the original shape click on any Fitting Options control which affects its shape, the easiest is the **Point Offsets** one.

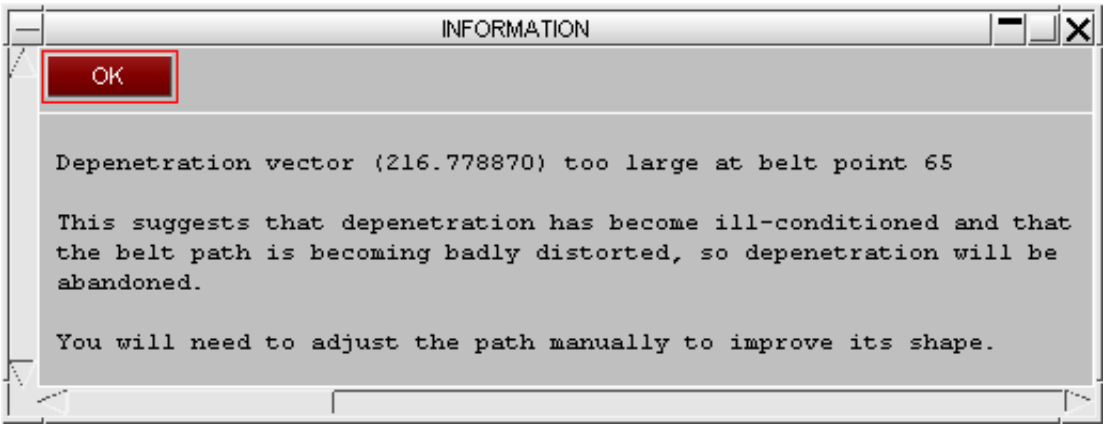
(3) Max iter: Setting the maximum number of iterations

Depenetration is an iterative operation that makes multiple passes down the belt. If it is going to work it usually does so in 2 or possibly 3 passes, if tit takes longer than that it is probably busy failing (see below). You can halt the process at any time with the general **Stop** button, so restricting the number of passes is really just an upper-bound limit on how long the process will take if you do not intervene manually.

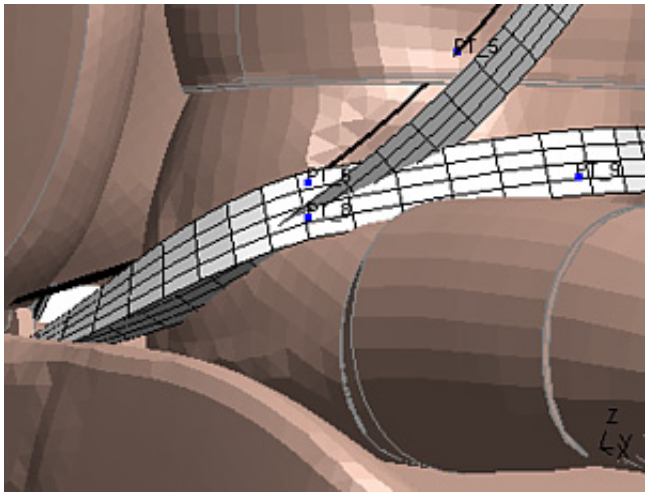
How and why automatic depenetration fails, and what to do about it.

The process is not always successful: try using it and you will find that it sometimes "blows up". This is because it not only has to use logic similar to contact algorithms but it also has to try to work out how a point should be depenetrated. If it chooses the wrong escape vector things rapidly get out of shape and beyond recovery.

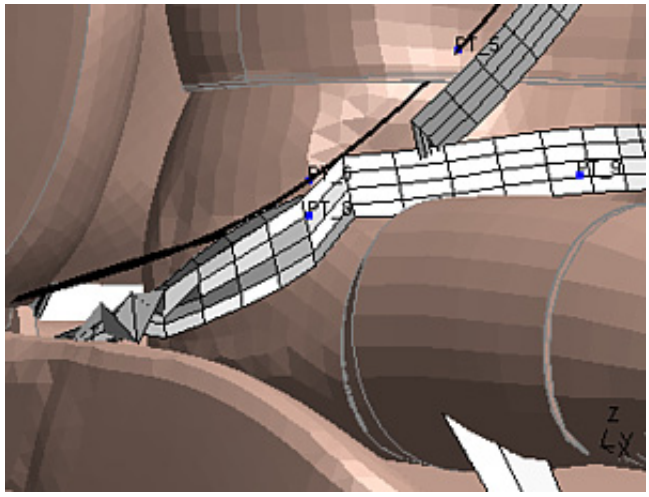
The process is iterative and monitors its own progress, so when it finds that a point on the belt is being moved by a ridiculous amount it gives up and reports failure. You will then see a message something like this:



To demonstrate this here is an example which fails. In the left hand image below the belt path around the chest has been twisted so that it intersects the pelvis section at a ridiculous angle, giving the "edge penetration" problem described above that the depenetration algorithms find hard. The right hand image shows what happens, and generated the message above.



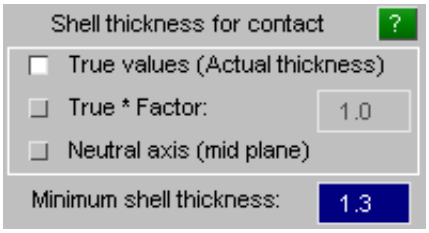
A really difficult depenetration problem of serious belt overlap and intersection. This is a deliberately contrived example, and is very poor modelling practice!



The result of attempting an automatic depentration of this: total failure! The belt path has become horribly distorted, and the only solution is to reposition the path points to give a better initial shape.

In conclusion automatic depenetration is still experimental and it is not totally satisfactory. It will resolve minor penetrations and simple belt overlaps, but it will not deal with really poor geometry. There is no substitute for creating a sensible and well-conditioned belt path in the first place!

Shell thickness for contact



Applying a factor to the thickness of structure shells in contact with the belt. (This is the same setting as that in Parameters 2 above, duplicated here to make it easier to adjust during path creation and editing.)

Normally contact between the belt and any shell elements in the structure uses the same parameters as an LS-DYNA contact would use, that is the shell's true thickness, or the thickness on a *PART_CONTACT card if this is defined.

This will give a tight fit of belt to dummy, but it can sometimes result in small initial penetrations because of differences between the contact algorithms in LS-DYNA and those in the belt fitter. Also the belt sometimes "pulls through to the wrong side" of structure shells during belt fitting due to the limitations of the belt contact algorithm, and artificially increasing the thickness of structure shells may help to prevent this happening.

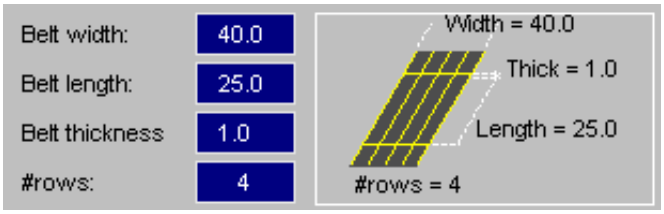
By default the true thickness is used, but by using **True * Factor** and supplying a factor greater than one the problems described here can often be avoided. Usually a factor in the range 1.5 to 2.0 will suffice, and you should use the smallest factor that works in order to avoid having an excessively loose fit between belt and dummy.

The Minimum Shell Thickness value can be used to set a lowerbound thickness, such that

```
t = max(True thickness, Minimum thickness)
```

This can be useful when a dummy has been coated with very thin null shells, and using their true thickness would impose a very small quantum of movement during fitting,.

Belt dimensions



The width, length, thickness and number of rows of elements across the belt can be set here. (This is the same setting as that in Dimensions above, duplicated here to make it easier to adjust during path creation and editing.)

Length Parameter

An optional parameter to receive the overall belt length, and further optional parameters to receive the length of each segment.

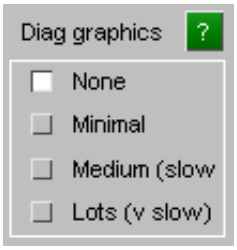
(This is the same setting as that in Parameter equal to total length above, duplicated here to make it easier to adjust during path creation and editing.)

If a **Length Parameter** ***name*** is defined then each time the belt is fitting PRIMER will update the parameter of that name with the overall length of the belt, creating it if it does not already exist.

If **_Sn per segment parameter** is ticked then PRIMER will also update parameters of ***name_1***, ***name_2***, etc for segments 1, 2, etc of the belt with the length of that segment, again creating these parameters of they do not already exist.

The purpose of this is to allow belt-related items such as sensors to define properties that are a function of belt length by referring to parameter ***name***. The most likely usage is to make ***name*** a component of a PARAMETER_EXPRESSION, permitting formulae to be defined.

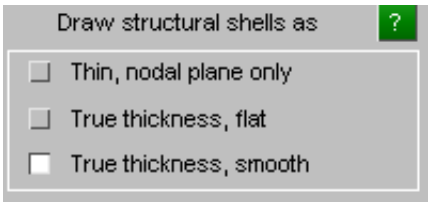
Diagnostic graphics



Adding further graphics during belt path creation, depenetration and fitting.

These options are mainly used during development and debugging, and while you are welcome to try them you should be warned that the results may be both confusing and likely to slow down operations. They have been left in the production version in case they can help with diagnosing problems at client sites, and Oasys Ltd may ask you to turn them on when capturing images for problem-solving.

Draw Structural Shells as



How shell elements that make up the structure are drawn during belt fitting.

PRIMER normally draws shell elements as "thin" at the plane of their nodes, ignoring any thickness or offsets. This is done for speed, and it is satisfactory in most contexts.

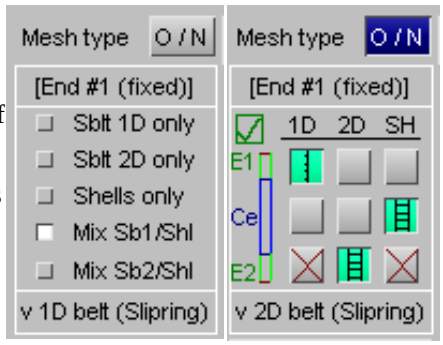
However during belt fitting, and especially when fitting a belt to tight geometries, it can be important to be able to visualise the true thickness of both the belt itself and any shells that make up the structure. The human eye is by far the best of whether and how closely things fit, but to see this it needs the correct information!

PRIMER has the option to draw shells using their true thickness, see Section 4.1.1 Display Options, Shells. However to save having to set this manually the belt fitter has a "local" setting for this which is active only when it is active.

6.40.4 **Mesh:** Meshing the fitted "chassis" mesh with structural Finite Elements

This is the final stage of seatbelt creation in which true Finite Element (FE) mesh elements are applied to the "chassis" mesh you have created and fitted.

"Old" versus "New" methods of defining elements



"Old" method "New" method

In PRIMER V14 the way in which belt elements are defined in each segment of the belt has been revised to make it more flexible.

Prior to V14, the "old" method, only certain combinations of belt element types were permitted in each segment of the belt:

- 1D seatbelt elements only
- 2D seatbelt elements only
- Conventional shells only
- Mixed 1D seatbelt elements + shells
- Mixed 2D seatbelt elements + shells

From V14 owards each segment of belt can be made up of any permutation of these element types, split into three spans:

- End 1 (E1)
- Centre (Ce)
- End 2 (E2)

You can swap between "Old" and "New" meshing methods via the **[O / N]** button.

Setting the default meshing method.

By default PRIMER V14 onwards will show the new style meshing method since it is much more flexible, however you can change this via two related preferences:

`primer*belt_definition_method: automatic | old | new`	Sets the meshing method button explicitly. The default is **automatic**, which means "new" unless the output format pre-dates V14.
`primer*mdumm_keyout_format: current | V14 | V13 | V12 | V11`	Sets the keyword output format for occupant-related cards written by PRIMER after *END.

It may not be possible to represent some "new" permutations of element types in "old" mode, in which case display will revert to the new mode.

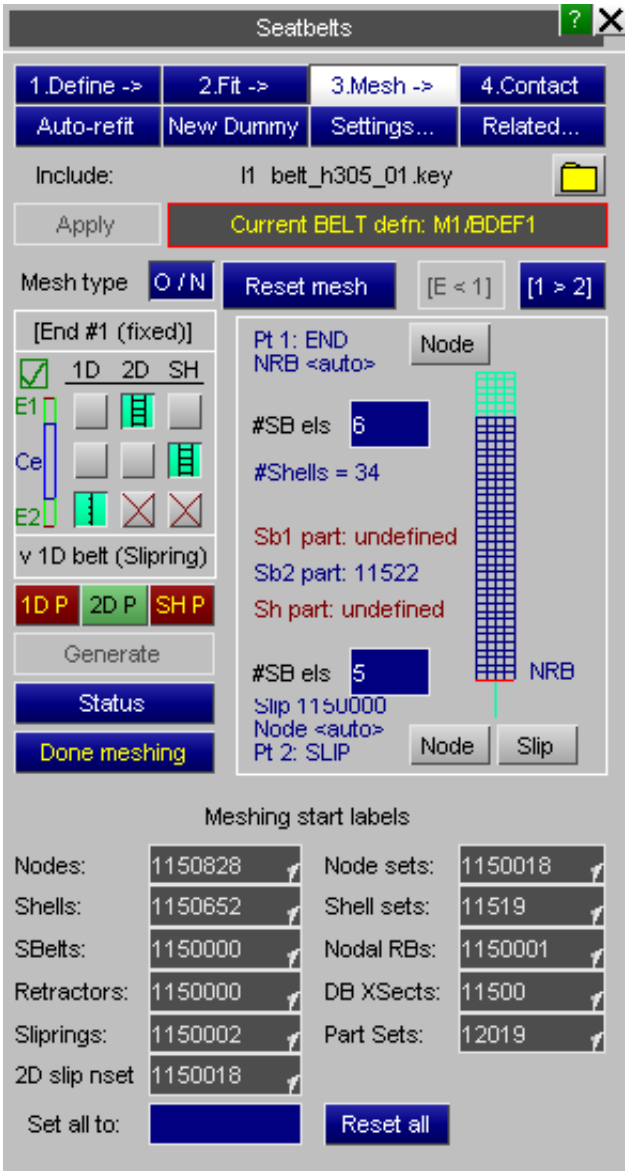
Meshing method compatibility between PRIMER versions

This change is backwards compatible: any belt meshed in an earlier version of PRIMER can be represented in the new method.

However going from V14 to an earlier version, ie representing the belt in "old" style, will only work if all segments of the belt are meshed in one of the 5 "old" layouts. This is because the meanings of the various "new" output data fields in the *BELT cards will not be understood by earlier versions of PRIMER.

(It is possible to get PRIMER to write out *BELT cards in an earlier file format using the mdumm_keyout_format preference described above, or interactively using the relevant controls in the **Settings...** panel.)

General belt mesh controls



You move between belt segments using the [>] (forwards) and [<] (backwards) buttons. A diagram of the current section is shown, and will also be sketched on the graphics image.

`Mesh type` determines how this section of belt will be meshed. From V14 onwards you have the options of "New style" (arbitrary mixture of element types) and "Old style") restricted to 5 arrangements. You can switch between old and new modes using the **[O / N]** button.

1D P, **2D P** and **SH P** define the "chassis" element properties for 1d belt elements, 2d belt elements and shell elements respectively. This means the Part, Section and Material definitions for each type plus any further optional data on the relevant Element cards.

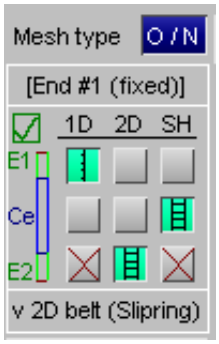
It is also possible to select explicit nodes and, if relevant, sliprings or retractors at each end of the belt segment. However this can usually be left in "automatic" mode.

In the case of a belt segment which mixes 1D or 2D seatbelt elements and shells it is also possible to define how many seatbelt elements should be used at each end.

Once the mesh definition is complete and all required properties have been given **Generate** may be used to create the mesh.

`Meshing start labels` allow you to control the labels from which each of the various belt components are created. These default to the "next free in layer", but may be set to any value so long as there is enough contiguous free space above them to hold all the required items.

Meshing type: Defining the mix of SEATBELT and SHELL elements.



Each segment of the belt between fixed ends or sliprings is divided into three spans:

- End 1 (E1)
- Centre (Ce)
- End (E2)

From V14 onwards, using "new style" meshing, each span may be any of the eligible element types: 1d belt, 2d belt or shell. In the example on the right this segment is meshed from element types:

[**1d belt** | **shells** | **2d belt**]

However there are some limitations imposed by the end conditions of the segment:

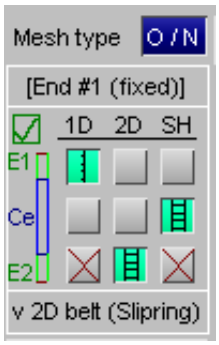
End condition	Limitations
Fixed end	No limitations, any element type may be used. Attachment will be directly by node, using a nodal rigid body terminate shell and 2d belt elements.
Retractor	Only 1d or 2d belt elements may attach to a retractor.
Slipring	At a slipring element (*ELEMENT_SEATBELT_SLIPRING) only 1d or 2d belt elements may be used. At a radiused slipring, where PRIMER meshes continuously around a finite radius, any element type may be used. However in both element and radiused cases the element types at both sides of the slipring must be the same. Thus if segment #2 / end #2 is a 1d belt element then segment #3 / end #1 must also be a 1d belt element, and so on.

Using the new style Mesh Type buttons.

This array of buttons forms a matrix:

[End condition of previous segment]			
Status	**1d belt**	**2d belt**	**Shell**
End 1	[ ]	[ ]	[ ]
Centre	[ ]	[ ]	[ ]
End 2	[ ]	[ ]	[ ]
[End condition of next segment]			

For each span E1, Ce, E2 you need to select an element type. The colours and symbols of the buttons have the following meanings:



If the background of the button is green that means that this selection is valid for that span of the belt.

If the background of the image is orange this means that there is an uncorrected error due to this selection. Hovering the mouse over an orange button will explain why that selection is invalid, for example element mismatch across a slipring.

An unselected option with a red X means that it would be an error to select this element type in this location. Again, hovering over the button will explain why this is the case.

The "status" button at the top left of this matrix shows the good / bad status of the belt as a whole, considering all of its segments.

Means that the meshing definition does not contain any errors.

Means that there are one or more errors somewhere in the belt mesh definition, and hovering over this symbol will list all such errors. It will not be possible to generate the belt mesh while there are errors in the mesh.

Using the old style Mesh Type buttons.

Each segment must be meshed with one of:

Sblt 1D only	**1D *ELEMENT_SEATBELT** elements only
Sblt 2D only	**2D *ELEMENT_SEATBELT** elements only
Shells only	Conventional **SHELL** elements only
Mixed Sb1/Sh	A mixture of **1D SEATBELT** and **SHELL** elements
Mixed SB2/Sh	A mixture of **2D SEATBELT** and **SHELL** elements.

Historically 1D Seatbelt elements have been used to attach to retractors and in stretches through sliprings, and Shell elements have been used where contact with the dummy is required. This is **Mixed Sb1/Sh** mode.

Now that 2D Seatbelt (shell) elements may be used then such meshes may be (re-)created in two possible ways:

1. Replacing the stretches of 1D seatbelt elements with 2D seatbelt shells, but preserving the stretches of conventional shells in between. This is **Mixed SB2/Sh** mode, and may be preferred by users who wish to retain belts meshed with shells and simply want to improve the contact between belt and surrounding structure in slipring and retractor regions.

2. Replacing the whole belt with 2D seatbelt shell elements, which is **Sblt 2D** only mode.

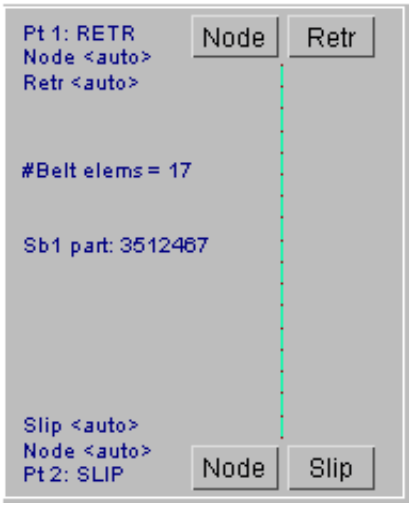
PRIMER will remesh an existing belt definition in any of the new types above, changing element, slipring and retractor properties as required. However certain permutations are not geometrically possible:

- Where a belt passes through an element slipring then the sections on both sides must be either 1D Seatbelt or 2D seatbelt elements as sliprings cannot mix 1D and 2D belt types. In addition retractors may only attach to 1D or 2D seatbelt types. Therefore at a Slipring or Retractor element only the following permutations are available:

Adjacent belt type	Permitted new style element type	Permitted old style adjacent meshing types
1D Seatbelts	**1d Belt**	**Sblt 1D only** or **Mixed Sb1/Sh**
2D Seatbelts	**2d Belt**	**Sblt 2D only** or **Mixed SB2/Sh**

Examples of each type of mesh

Sblt 1D only: Meshing with 1D Seatbelt elements only



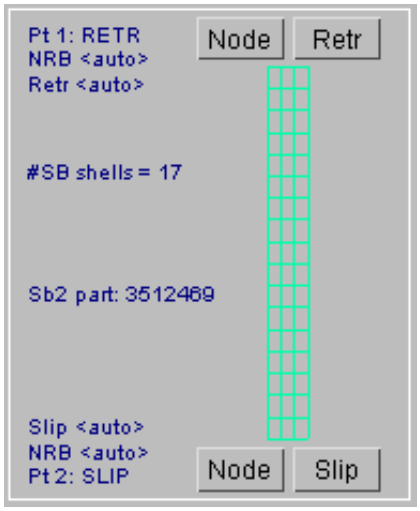
This mode is typically used in a mixed 1D Seatbelt + Shell mesh for the initial stretch between retractor and slipring at shoulder.

There is nothing further to define since the stretch will contain only a single type of element.

End details are handled as followed:

At a retractor or slipring	The belt connects directly to the retractor or slipring element
At structure	The last belt node uses the relevant node on the structure
At another 1D belt element	The belt elements share a common node at the point.

Sblt 2D only: Meshing with 2D Seatbelt elements only

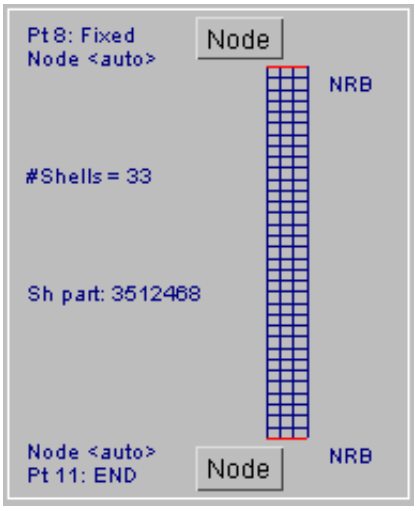


This mode may be used for any stretch of belt.

End details are handled as followed:

At a retractor or slipring	The belt connects directly to the retractor or slipring element
At deformable structure	A nodal rigid body is created from the line of nodes at the end of the belt, plus the relevant node on the structure.
At rigid structure	The nodes on the end of the belt are placed in a node set which is used to make an "extra nodes on rigid part" definition assigned to the rigid structure part.
At another 2D belt element	A nodal rigid body of the nodes at the ends of each section would be created. (This would he highly unusual in practice.)

Shells only: Meshing with conventional shells only

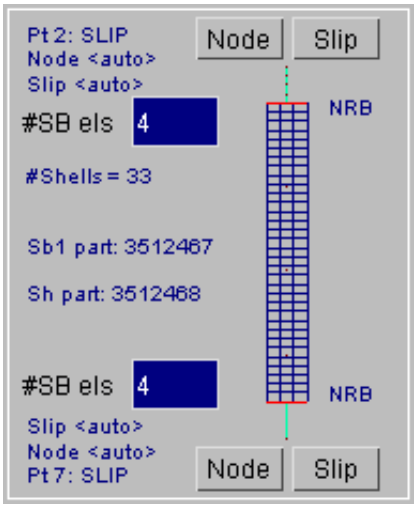


This would be unusual since "shell only" stretches of mesh cannot attach to slipring or retractor elements, so this would only be applicable to an isolated stretch of belt with fixed connections at each end. However is *is* possible to attach shells to a "meshed" slipring.

The end details are handled as follows:

At deformable structure	A nodal rigid body is created from the line of nodes at the end of the belt, plus the relevant node on the structure.
At rigid structure	The nodes on the end of the belt are placed in a node set which is used to make an "extra nodes on rigid part" definition assigned to the rigid structure part.
At another shell (belt) element	A nodal rigid body of the nodes at the ends of each section would be created. (This would he highly unusual in practice.)
At a meshed slipring	The shell elements continue around the slipring, using a reduced length per element, meeting the next segment’s elements half way round.

Mix Sb1/Sh: Meshing with 1D belt elements + shells



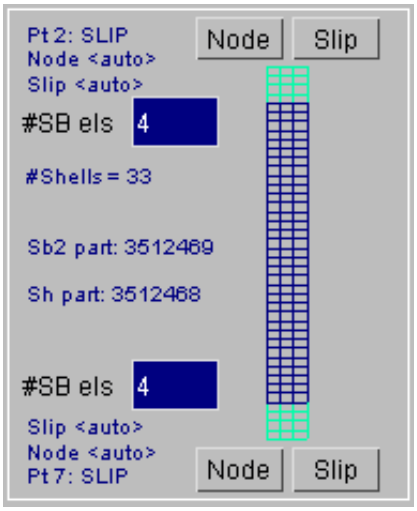
This is the traditional method of combining connectivity through sliprings and retractors (using 1D seatbelt elements) with good contact across the dummy (using shells). A typical stretch of belt will be mainly shells with a few seatbelt elements at each end.

#SB els defines how many 1D seatbelt elements are to be used at each end, which must be at least one if there is a slipring or retractor at that end.

The end details are handled as follows:

1D belt element at structure	The end node of the belt element is the specified node on the structure.
1D belt element at slipring / retractor	The belt connects directly to the slipring/retractor.
Shell to 1D belt connection	A nodal rigid body is formed from the end row of shell nodes, and the 1D belt element also has an end node in this node set.
Shell directly to deformable structure	A nodal rigid body is created from the line of nodes at the end of the shell, plus the relevant node on the structure.
Shell directly to rigid structure	The nodes on the end of the shell are placed in a node set which is used to make an "extra nodes on rigid part" definition assigned to the rigid structure part.
At another shell (belt) element	A nodal rigid body of the nodes at the ends of each section would be created. (This would he highly unusual in practice.)

Mix Sb2/Sh: Meshing with 2D belt elements + shells



This method could be used as a direct replacement for the mixed 1d belts + shells above. In this case **#SB els** defines how many 2D belt elements are used at each end. Since 2D seatbelt elements are really shells no special measures need to be taken at the transition between the two element types.

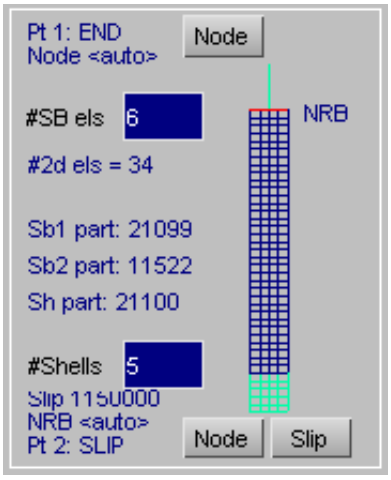
The end details are handled as follows:

2D belt element at slipring / retractor	The belt connects directly to the slipring/retractor.
2D belt or Shell element directly to deformable structure	A nodal rigid body is created from the line of nodes at the end of the belt/shell, plus the relevant node on the structure.
2D belt or Shell element directly to rigid structure	The nodes on the end of the belt/shell are placed in a node set which is used to make an "extra nodes on rigid part" definition assigned to the rigid structure part.
2D belt or Shell to another belt/shell (belt) element	A nodal rigid body of the nodes at the ends of each section would be created. (This would he highly unusual in practice.)

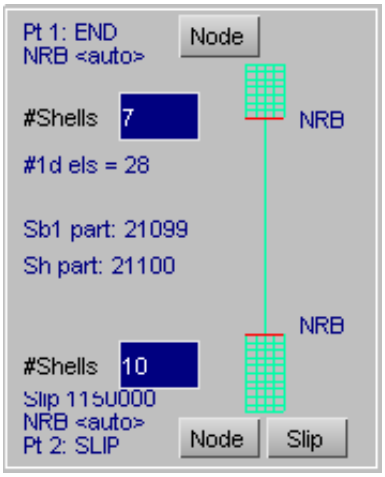
Mix Sb1/Sb2/Sh: Meshing with an arbitrary mixture of element types.

This is only possible in new style meshing mode, in PRIMER V14 onwards. Several examples of possible mesh layouts are shown here since the choice is compleely arbitrary. The end conditions of the various element types conform to the descriptions given above.

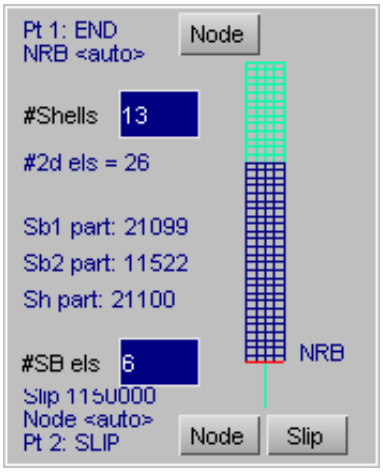
[**1d belt | shell | 2d belt**]



[**shell | 1d belt | shell**]



[**2d belt | shell | 1d belt**]



How the ends of stretches of shells or 2D seatbelts are handled

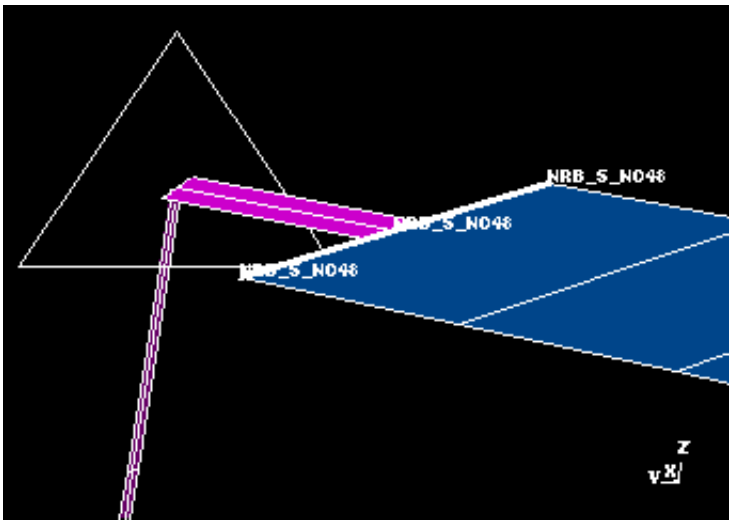
The ends of stretches of (seatbelt) shell elements need special treatment since they need to be attached at a single point either to a structural node or to the adjoining seatbelt element. This presents two problems:

- For odd numbers of shells across the belt (`#rows` = 1, 3) there is no central node to attach to.
- In all cases fixing at a single point will lead to excessive warping, hourglassing or other distortion of the end shells because of the lack of end constraint.

Therefore PRIMER usually creates a "Nodal Rigid Body" (NRB) at the ends of stretches of shell elements: both to constrain the ends against warping and to provide an attachment point. (Exceptions are given below.)

This example shows a shell (blue) to seatbelt (purple) connection next to a slipring. A Nodal Rigid Body has been created using the two shell end nodes, and an extra node has been created at the shell centreline to which the seatbelt element is attached.

There are some special cases that arise because of the limitation that nodes can only be constrained by one rigid body at a time.

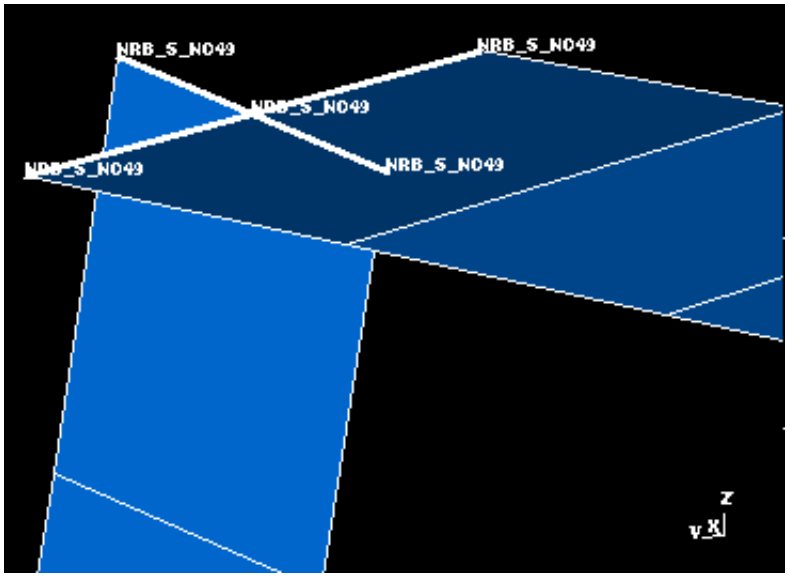


Shell mesh meets shell mesh at fixed point (no intervening seatbelt elements)

In this situation Nodal Rigid Bodies are required at each shell end, but because they share a common centre node they cannot be separate.

Therefore PRIMER creates a single NRB and places all the nodes into it, as shown in this example.

This has the effect of fixing the belt end geometries together, ie they can't twist relative to one another. If this is not acceptable it will be necessary to mesh the two segments as separate belt definitions, then to connect the central end nodes (which will be separate) manually with a joint, which may need lumped masses at the nodes for stability.

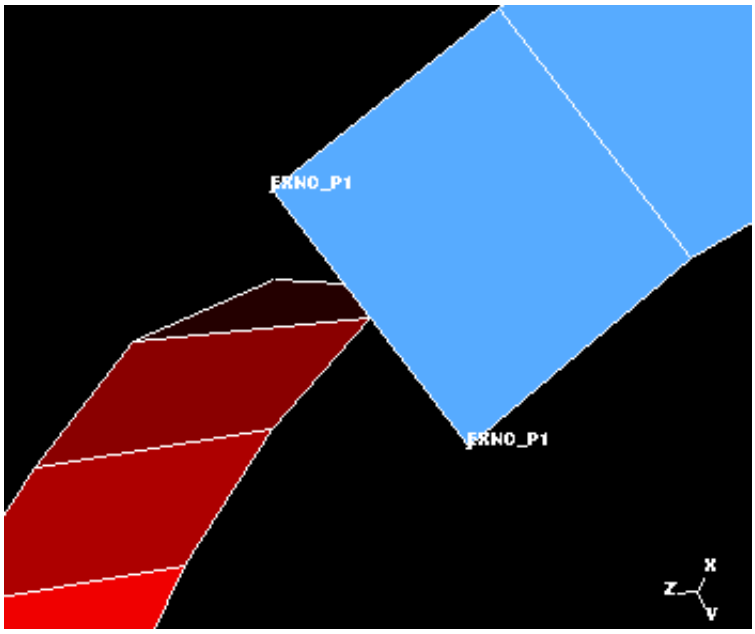


Shell mesh meets rigid structural node. (also no intervening seatbelt elements)

Where a shell mesh attaches directly to a node on rigid structure, PRIMER does not create a nodal rigid body: instead it makes the end nodes of the shell "Extra Nodes" on the rigid part of the structure.

In this example the (blue) belt shells are attached to (red) rigid part #1, and the two end nodes have been made Extra Nodes on part 1.

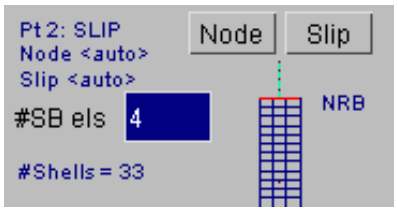
Again, this fixes the end shell relative to the rigid structure, inhibiting relative rotations. If this is unsatisfactory then let PRIMER create the shell centre node and NRB, then attach the structure manually to the central NRB node via a joint (maybe with a mass) as above.



Other illegal constraints on end nodes.

At present PRIMER only checks for the cases above when meshing belts. It would be possible to imagine some other very obscure illegal constraints, but these are so unlikely that they are ignored. You will have to fix such cases manually if they arise.

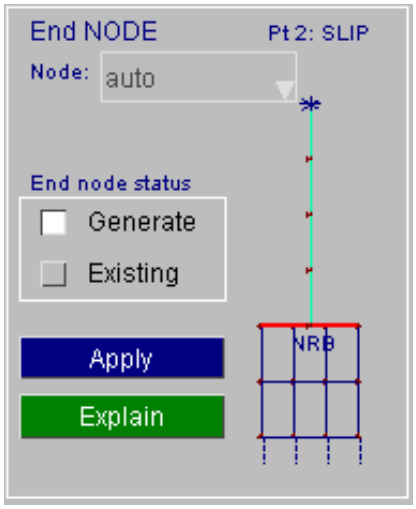
Controlling the end connectivity of each segment



By default PRIMER generates any nodes required at the end of each segment, ensuring that connectivity between stretches of belt is maintained. Also if a slipring or retractor are specified at the end of a segment this will also be created.

However it is possible to override this behaviour and to specify existing nodes, sliprings and retractors at the ends of segments, and this might be necessary when connecting up a belt meshed in PRIMER to one meshed elsewhere that the belt fitter does not "know" about.

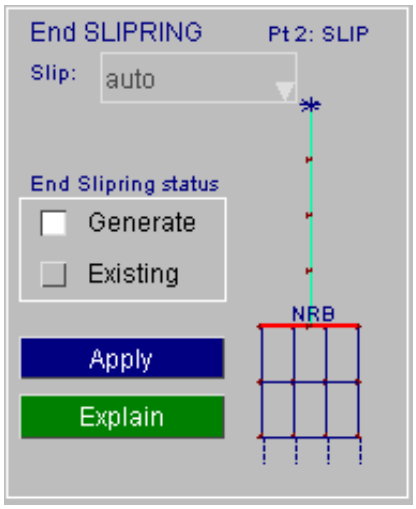
NODE: defining the end node for 1D seatbelt element connectivity



When a segment ends in a 1D seatbelt element the end node is normally created or selected automatically, but you can override this by choosing **Existing** and specifying an existing node.

In this case it is your responsibility to ensure that connectivity between adjacent stretches of belt is correct.

SLIP or **RETR**: defining a slipring or retractor (1D and 2D cases)



In a similar way if a slipring or retractor have been defined at the end of a stretch of belt PRIMER will normally create them automatically, however you can override this and select an existing element.

Note that:

- In the 1D case any existing seatbelt elements will be replaced as required with new elements created as part of this meshing operation, and will be left "floating" in the model. You will need to delete these manually.

  The slipring or retractor node (SBRNID) will be replaced with the correct node for this mesh. This may result in the slipring / retractor moving in space in order to be at the correct position.

- In the 2D case any existing element sets will be emptied of their contents, and new elements from this meshing operation inserted. Again you will need to tidy up any elements left "floating" in the model.

  Any existing node set (-SBRNID) will be emptied and repopulated with new nodes at the correct positions. As in the 1D case this may result in the slipring/retractor moving to a new location.

Meshing with "meshed" (radiused) sliprings

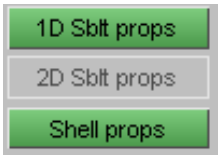
When a meshed (radiused) slipring has been specified PRIMER will show SLIP as being specified at that point, and also a label. However during the actual meshing stage no explicit slipring will be created, rather the belt elements will be continuous around the specified radius. Therefore no special action is required for these sliprings.

Navigating the meshing panel between segments.

When a belt definition has more than one segment the meshing panel only displays one segment at a time: by default the first. In addition the segment highlighted on the display will be the current one.

To move between segments use **[<]** to go backwards, and **[>]** to go forwards. You can revisit and modify segments as often as you like, they only get meshed when the **GENERATE** command is used.

Setting element properties.



Before elements can be generated PRIMER must know what properties (Part ids, etc) to assign to the elements.

Therefore it is necessary to select an existing property, or create a new one for the relevant class(es) of element. The following colour scheme is used for these buttons:

Red	The property has not yet been defined.
Orange	The property has been defined, but a check shows errors or warnings
Green	The property has been defined and checks cleanly.

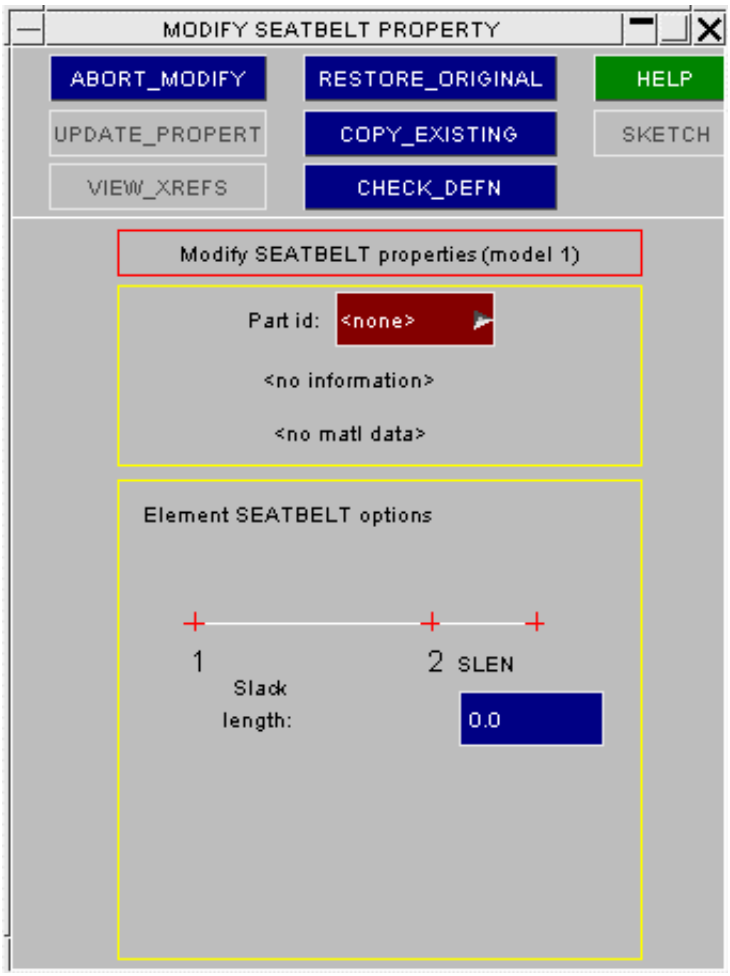
If a property is required but has yet to be defined that button will have a red background, and the **GENERATE** button will be greyed out until it has been selected or created.

Properties not required will have their button greyed out.

This panel shows the 1D seatbelt element create/ edit box.

The user is required to give a part ID, this can either be typed in directly or selected/created via the part pop-up box. Information about the section and material properties from the part to be used are shown below the part ID.

There is also the option of setting the element-specific "slack length" value.



This panel shows the shell property creation/editing box.

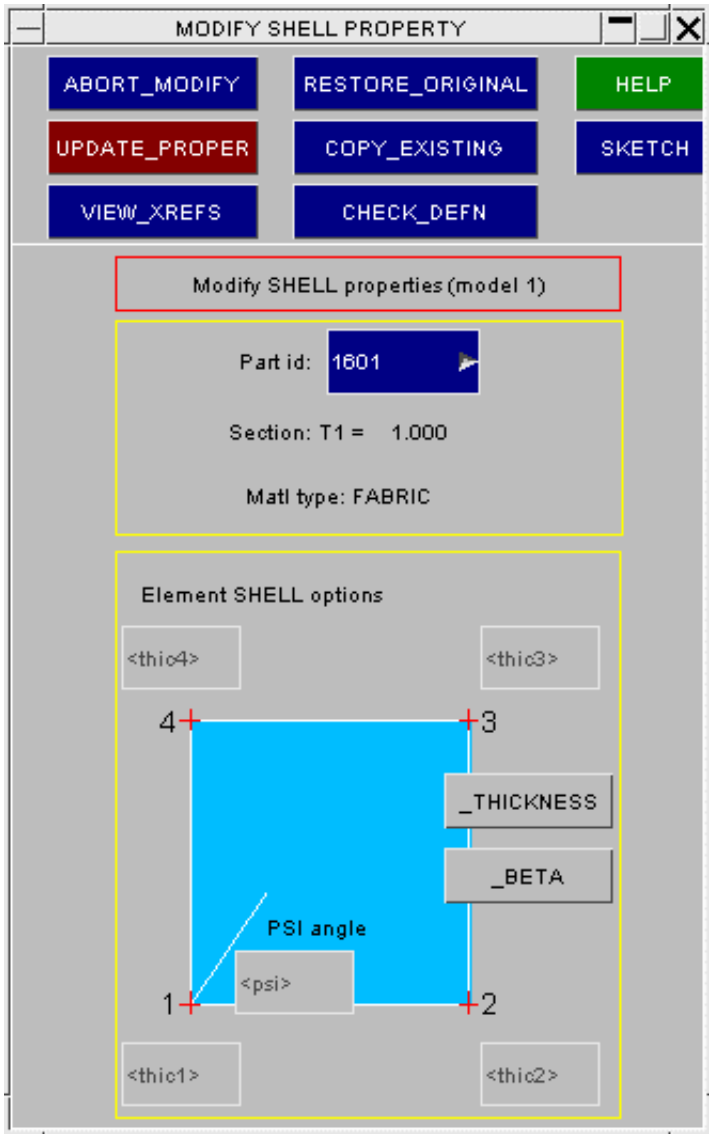
As above, a part ID is required to be set. Here one has been selected and the properties are displayed below the ID. The **update_property** button has now been made active.

The shell element specific data, (variable thickness at nodes and beta angle) is not used here.

This panel is also used for 2D Seatbelt Shell elements which, despite being defined as **`*ELEMENT_SEATBELT`** are in fact shells and must have the following attributes:

- A **`*PART`** card unique to this seatbelt shell definition.
- A **`*SECTION_SHELL`** (***not _SEATBELT***) definition that is unique to the part. Since the shells are used only for contact purposes the properties on this card are not structurally significant, however a sensible thickness should be used and a membrane formulation (type 5) is recommended.
- A **`*MAT_SEATBELT`** card. It is recommended that this too is unique to the **`*PART`** definition above.

Unique definitions are required because of the extra work performed in the LS-DYNA keyword reader to "track down" the belt and generate 1D belt, slipring and retractor elements. It gets confused if multiple belts used the same basic definitions.



Actually creating the Finite Element mesh

Once you have defined all the necessary information use **GENERATE** to go ahead and generate the finite elements themselves.

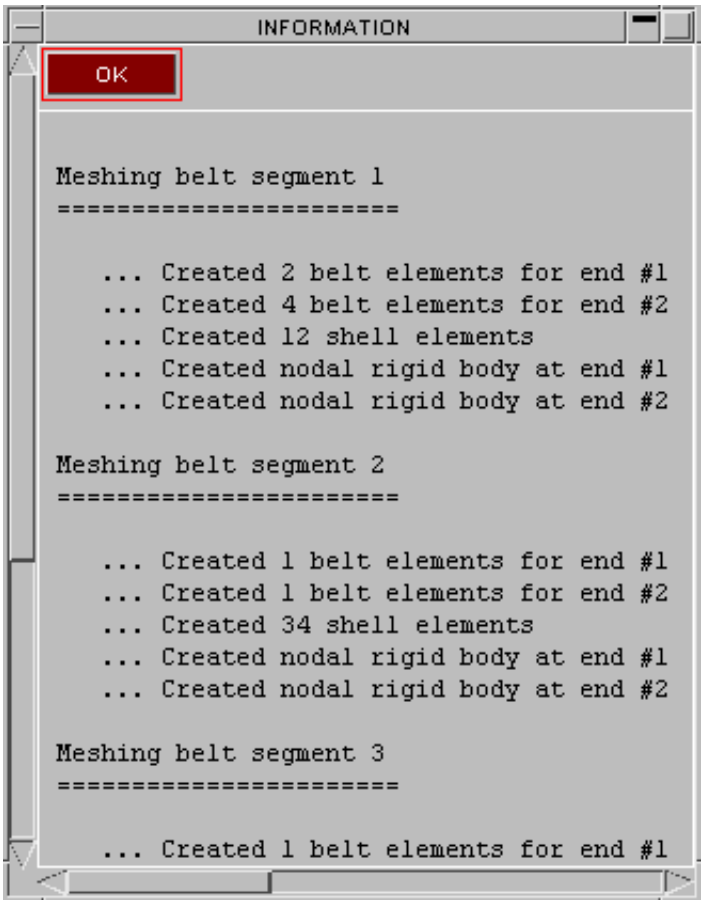
If you are modifying an existing seatbelt definition you will be asked first whether you want to delete the existing nodes, elements, etc. You have the choice of deleting them or leaving them - it doesn't matter which - as the new definition will supersede any previous one.

However if you have chosen start labels for the new mesh that you know will overlap with the existing ones then you must delete them!

PRIMER will generate a confirmation message listing what has been created.

Here a typical mixed seatbelt and shell belt, with nodal rigid bodies connected the two element types.

And that's it: you've created a seatbelt!



Parameter to contain the total seatbelt length.

As part of the post-meshing report the length of each belt segment and also the total length are reported.

There is an option to define the name of a ***PARAMETER** that will have its value updated with the total belt length whenever the belt is remeshed. This works as follows:

- The name of a parameter is defined in the belt fitting dimensions panel. No leading "&" should be used.
- If it does not already exist this parameter will be created. If it does already exist it should be a floating point parameter, type "real".
- Whenever the belt is remeshed the value of this parameter will be updated with the new total belt length. The model will then be checked for anything that uses this parameter, and if anything is found you will be shown the standard PRIMER panel that controls the update of the model following a parameter change. The effect is identical to editing the parameter manually and inserting the new length value by hand.
- When the model is written out to a keyword file a ***BELT_PARAMETER** card will be written that contains this parameter name.

The purpose of this is to permit items that might rely on the total belt length to be updated automatically as it changes. For example a ***ELEMENT_SEATBELT_SENSOR** card might use the length of the belt to control trigger parameters.

Further parameters to contain the length of each belt segment.

If the option to "add _Sn parameter" for each belt segment was selected in the in the belt fitting dimensions panel then each section of the belt will have a parameter generated which contains that segment's length. This will have the name of the total length parameter above with "_S1" appended for segment 1, "_2" for segment 2, and so on.

As with the overall length parameter the intention is that these can be used to control behaviour of individual belt segments.

Post-meshing review

Following a meshing operation PRIMER runs a check on the new definition and shows a summary of the results. It is not uncommon for the definition of retractors to be incomplete (PRIMER cannot know how they are to be activated) and for other errors to occur.

Items with errors are highlighted in red, as shown in this example, and you can click on them to open their editin panels and correct the problems.

What is stored following meshing

Internally PRIMER saves the following information about Seatbelt Definitions:

The sets of structure definitions. (See section 6.40.1 Defining structure for seatbelt fitting).

The basic path. (See 6.40.2 Defining a belt path).

The "chassis" mesh and associated fitting data. (See 6.40.3 Fitting the belt to the dummy).

The belt FE data: sets of shells, first and last seatbelt, slipring and retractor elements, and the sets of nodal rigid bodies used.

Any cross-section definitions that were requested.

Any parameter name defined to hold the belt total length.

Contact data if it exists (See section 6.40.5 **Contact**).

A **MESHING** operation operates on the information saved in (1) to (3) above without changing it, therefore if you are happy with the belt shape but don't like the mix of element types you have created, you can go back and change it at will.

What you *can't* do in **MESHING** is change the "chassis" path, and in particular you can't change the number of shells across the width: to do that you must go back to **FITTING** (section 6.40.3), fit a revised chassis mesh with the new number of rows, then **ACCEPT** and remesh it.

Changing and remeshing an existing belt definition.

Once a seatbelt has been created and meshed in PRIMER, (ie that there is a set of *BELT cards after *END) then you can do the following:

Remesh the existing path geometry using a different mixture of elements. For example change an existing Mixed 1D/Shell belt to a Mixed 2D/Shell belt.	Simply use **Mesh** to define a new arrangment of elements or element types. Note that simply remeshing does not, in itself, permit the following: • Changing the basic belt path • Changing the number of rows of elements across the belt, or any other aspect of the belt element geometry. The existing belt definition will be deleted and the new one will replace it. (If you have not previously "Fitted" the belt during this PRIMER session it will be necessary to do that first, since the detailed meshing path is not stored in the keyword file. Performing a "Fit" will repeat the form-finding operation using the saved basic path, which regenerates a detailed path suitable for remehsing.)
Modify the basic belt path, or change the belt geometry, and then remesh.	Firstly use **Fit**, where you can: • Define a new belt path by adding, subtracting or moving path points. • Change the belt dimensions or number of row Once the revised path geometry is correct **Save** it, re-**Fit** the belt to the dummy, **Accept** the result, and then re-**Mesh** it as above. The existing definition is deleted and a new one created as before.
If the dummy has moved, and you want to refit the existing belt definition to it in its new location.	The **Auto-refit** capability will perform this task if no other changes are required to belt organisation or topology. **Auto-refit** attempts to re-use all existing node, element and set labels so that the belt just appears to move to the new location. Otherwise the process may be performed manually as above.

More details about meshing 2D seatbelt elements.

2D seatbelt elements, introduced in LS-DYNA 971R4, can be meshed by PRIMER but they are quite intricate and the following explanation has been added at the request of users to try to explain in more details how this is done.

What is a 2D seatbelt element, and how does it work?

LS-DYNA imports these elements using the **`*ELEMENT_SEATBELT`** card which traditionally has been used to define 1D seatbelt elements, however if nodes 3 and 4 are defined on this card it becomes a 2D seatbelt element. Confusingly 2D seatbelt elements are actually shells, albeit special ones, and their label range forms part of that used by

***ELEMENT_SHELL**. Therefore PRIMER is also forced to treat them as shells to avoid conflict. The following summary may help:

- 2D seatbelt elements are defined on the ***ELEMENT_SEATBELT** card by defining nodes N3 and N4.
- Internally they are actually **SHELL** elements, and their labels must not clash with existing shells.
- They should refer to a conventional ***PART** card.
- The ***PART** card should refer to a ***SECTION_SHELL** card (***not *SECTION_SEATBELT***) which should be unique to each stretch of 2D belt elements.
- The ***PART** card should refer to a ***MAT_SEATBELT** definition.

- ***ELEMENT_SEATBELT_RETRACTOR** and ***ELEMENT_SEATBELT_SLIPRING** definitions must also obey special 2D syntax, using sets of nodes and elements.
- A node set (field <**edgset**>) on the ***SECTION_SHELL** card is used to tell LS-DYNA where the end of the belt is, and the ordering of nodes across it.

- The ordering of nodes, seatbelt elements and the sets of these used in retractors and sliprings must be topologically consistent throughout the belt.

Inside the LS-DYNA keyword reader the following happens when these elements are read in:

- The **N** parallel lines of 2D ***ELEMENT_SEATBELT** elements making up a belt are used to generate **N+1** parallel lines of conventional 1D seatbelt elements.

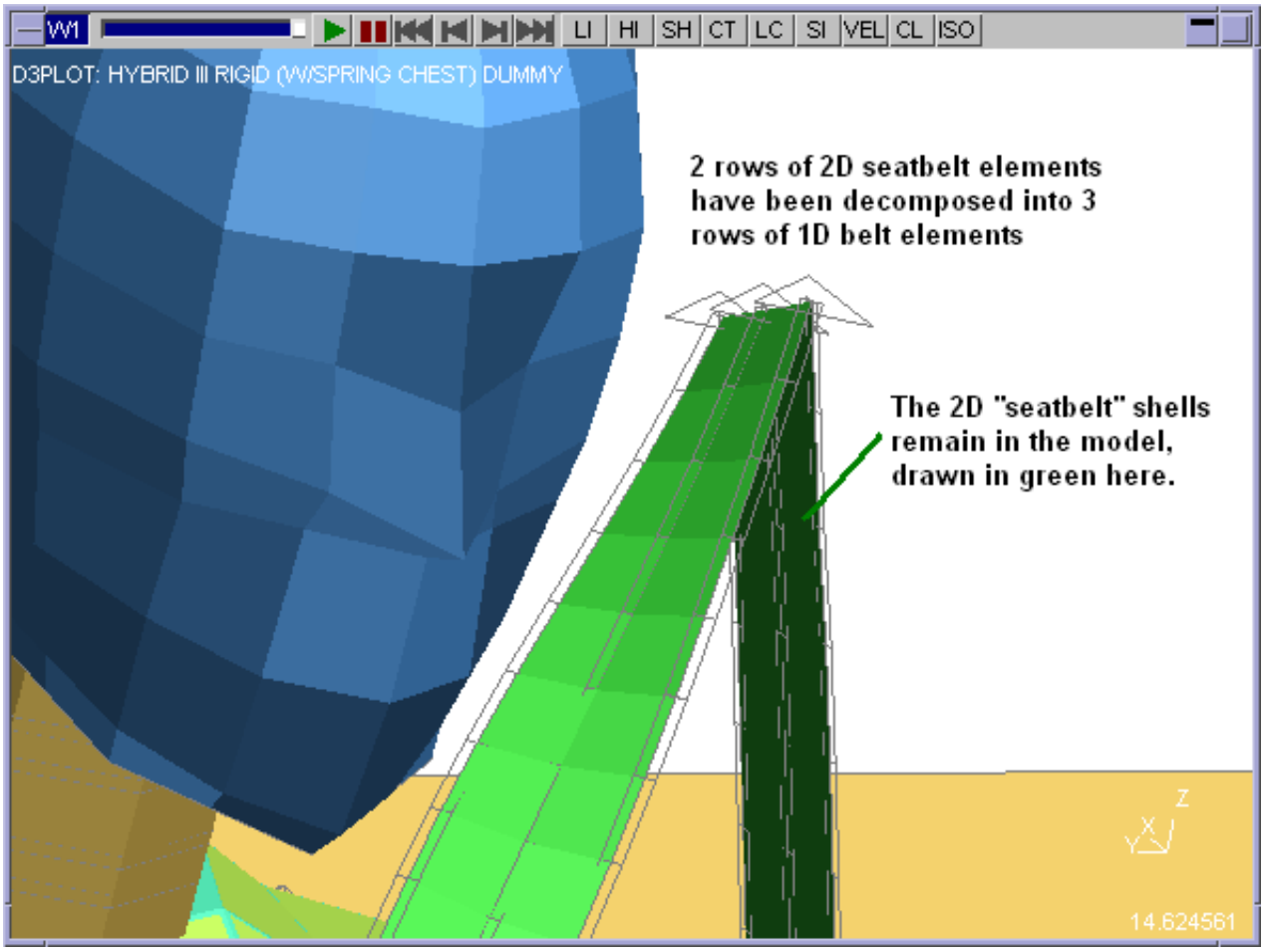
- The 2D seatbelt elements, which are really shells, are retained as shells and are given notional non-structural properties. The properties used are not known, but their job is to hold the parallel lines of 1D belt elements together during the analysis and also to provide a 2D contact between belt and dummy.

- 2D retractors and sliprings are also decomposed into **N+1** parallel rows of their 1D equivalents, and the rows of 1D seatbelt elements pass through these.

- During the analysis these rows of 1D belt elements carry the force in the seatbelt, although the details of how they work with the 2D shells are not known.

- Examining the output during post-processing reveals how this decomposition from 2D belt items to their 1D equivalents has taken place.

The following image, showing (during post-processing) the shoulder slipring detail of a model containing a 2D seatbelt with 2 rows of "belt" shells demonstrates how this decomposition has taken place. The three sliprings and parallel rows of 1D belt elements superimposed on the shell mesh (green) are clearly visible.



How 2D belts are meshed in PRIMER

During the fitting operation PRIMER maintains a "scratch" belt definition showing the number of parallel rows of elements that the user has defined. During 2D belt meshing these simply become 2D **`*ELEMENT_SEATBELT`** definitions, but the following extra work has to be performed:

- At any retractor and slipring positions a **`*SET_NODE`** is required to define the structure to which the retractor/slipring is attached. This is defined as field **`sbrnid`** on the retractor and slipring cards using -ve label to designate "set for 2D element".

- Sets of shell elements (**`*SET_SHELL`**) have to be created (data field **`sbid`** on the retractor card, **`sbid1`** and **`sbid2`** on the slipring card, all using -ve labels) to define the row of 2D seatbelt elements, which are really shells, adjacent the the retractor/slipring.

- At any free end of the belt PRIMER will create a nodal rigid body to act as a rigid fixing using the set of nodes at the free end, unless the end node location is already on a rigid part in which case these nodes become "extra nodes" on that rigid part instead.

- PRIMER will also place a node set at a free end of the belt into field **`edgset`** on the **`*SECTION_SHELL`** card of the belt elements. This is required to tell LS_DYNA both where the end of the belt is, and what its topology and orientation are. Since **`edgset`** can only refer to a single belt this explains why a separate **`*SECTION_SHELL`** card is required for every 2D belt in a model. If a retractor has been used PRIMER will use its node set **`sbrnid`**, otherwise it will use the node set of the free end.

Special care has to be taken to ensure that the topology of all elements and the ordering nodes in the belt, and in all sets, is consistent so that LS-DYNA can rely on it to determine the belt geometry.

Special geometrical rules also have to be applied at slipring locations to make sure that the 2D belt mesh is continuous through them and aligned sensibly. This has been described previously in "The effect of sliprings on the belt path".

Special considerations affecting node set numbering on 2d sliprings.

There is a bug in the LS-DYNA structured format which means that the label of the node set NSID used to define the location of a 2d slipring cannot be more than 7 digits long. This limitation will be removed in future versions of LS-DYNA, and is also not relevant if "wide" format is used. In the meantime PRIMER applies the following rules:

- It maintains separate internal label ranges for "node sets used for 2d sliprings" and "node sets used for everything else in the belt".
- In the case of remeshing a belt where node sets for sliprings are 7 or fewer digits, and all others 8 or more digits, the separation is maintained so that new sliprings retain their small node set labels.
- When meshing from scratch you are free to set separate initial labels for "node sets generally" and "node sets used for 2d sliprings".

In PRIMER V12, which can handle "wide" format these restrictions are removed if the model is using large (ie > 8 digit) labels, and they will also be made LS-DYNA version dependent once a version is released in which this bug has been fixed.

Output from 2D Seatbelt elements

Time-history output of 2D seatbelt data appears in the ascii file **`sbtout`**, generated by **`*DATABASE_SBTOUT`**, and has changed slightly over time:

- In LS971R4.2.1 and older output reveals how the internal representation of these belts is split up into parallel 1D rows as above.
- In later LS-DYNA versions output is merged back into single sliprings and retractors, as it appears in the input.
- There is an undocumented data field **`1p1dblt`** in row 2, column 7 of the **`*CONTROL_OUTPUT`** card which will cause output to revert back to the raw 1D output if set to 1.

- LSTC recommend that forces in 2D seatbelt elements are obtained by creating **`*DATABASE_CROSS_SECTION`** definitions cutting the belt. From release 11 PRIMER can generate these automatically as part of the belt fitting and meshing process.

Output of data for plotting is possible because LS-DYNA will generate beams on top of "discrete" elements, which includes 1D seatbelt elements, if the **`beam`** field in column 3 of the **`*DATABASE_D3PLOT`** card is set correctly. Therefore beam plotting can be used to visualise the forces in the belt elements.

6.40.5 Contact: Creating a contact between belt and dummy

Once you have created and meshed your seatbelt the final, stage is to create a contact between it and the dummy. In most models this will be done outside the seatbelt fitter since the contact will almost certainly have to include structure not explicitly included in the dummy and belt definition (seat, dashboard, airbag, steering wheel, etc).

However if a simple contact between belt and dummy will suffice PRIMER provides the option of generating these contact definitions automatically for you.

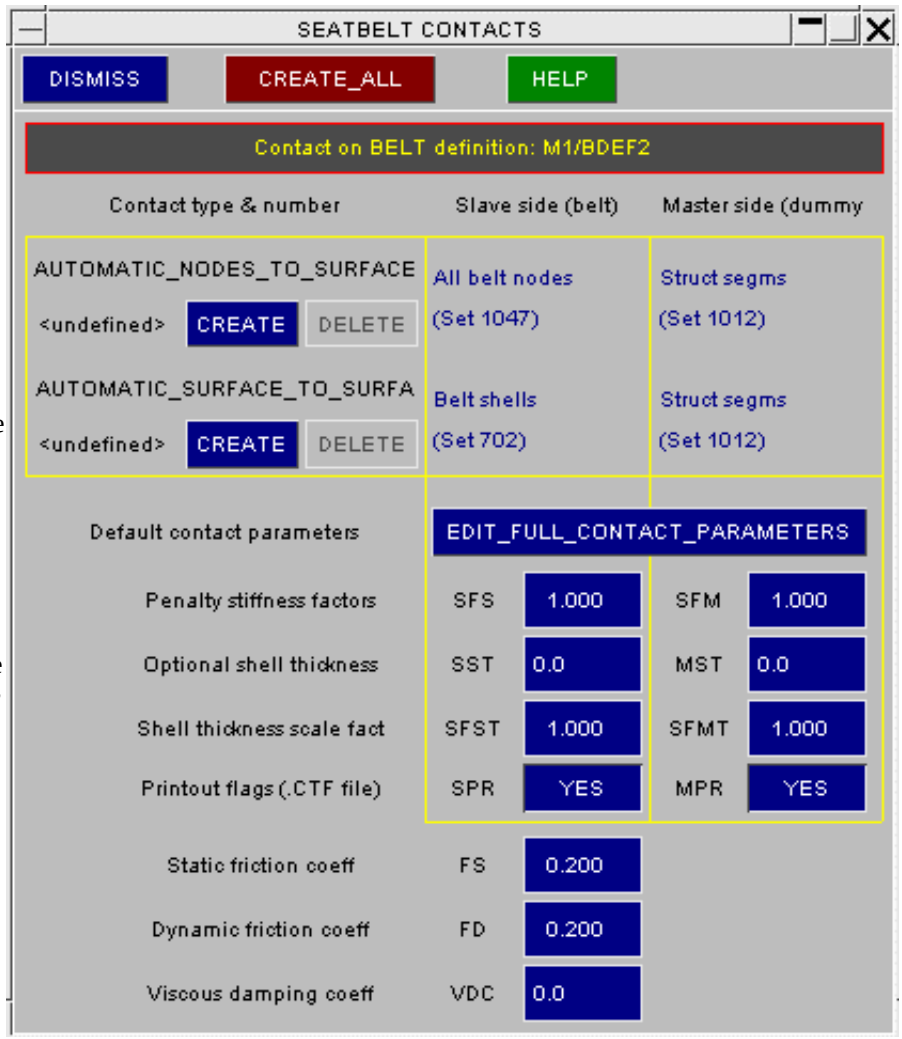
PRIMER works on the assumption that either or both of the following contact surfaces will be required:

`*CONTACT_AUTOMATIC_SURFACE_TO_SURFACE` Between belt shells and dummy.

`*CONTACT_AUTOMATIC_NODES_TO_SURFACE` Between belt nodes and dummy.

The need for the second, node-based contact arises when **`*ELEMENT_SEATBELT`** elements are used, since these are "line" elements with no effective surface.

PRIMER generates "chassis" contact definitions as shown in the adjacent figure.



Prototypes of the two contact types defined above are created by assuming that the dummy structure (as segments) form the master side of both, and creating two further sets:

A **`*SET_NODE`** of all nodes on the belt as the slave side of the **`NODES_TO_SURFACE`** contact.

A **`*SET_SHELL`** of all shells in the belt as the slave side of the **`SURFACE_TO_SURFACE`** contact.

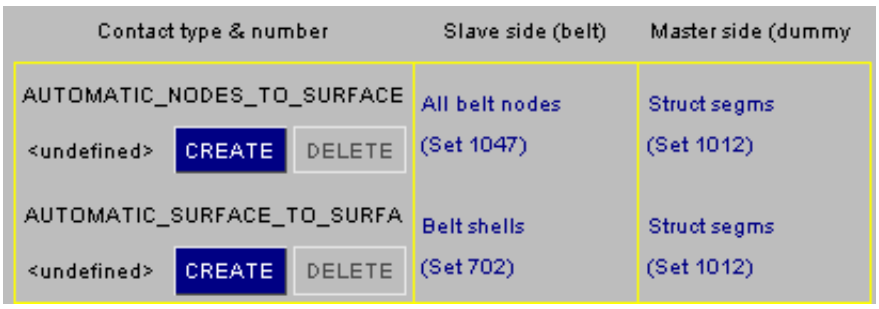
In both cases the "structure"side is defined by a **`*SET_SEGMENT`**, because this allows it to include an arbitrary mixture of element types.

Default parameters are set up for the complete contact, and the main ones are shown here.

To create both these contacts with the default settings use **CREATE_ALL**. This will turn these "chassis" definitions into actual contact surfaces which, while part of a belt definition, are normal contact definitions which may be viewed and edited just like any other.

The default settings will create two contacts, which implies some duplication since all the nodes on shell elements will be included in the **`*NODE_TO_SURFACE`** contact, which is a bit wasteful. In addition default parameters may not be suitable for all models.

To create/edit contacts selectively

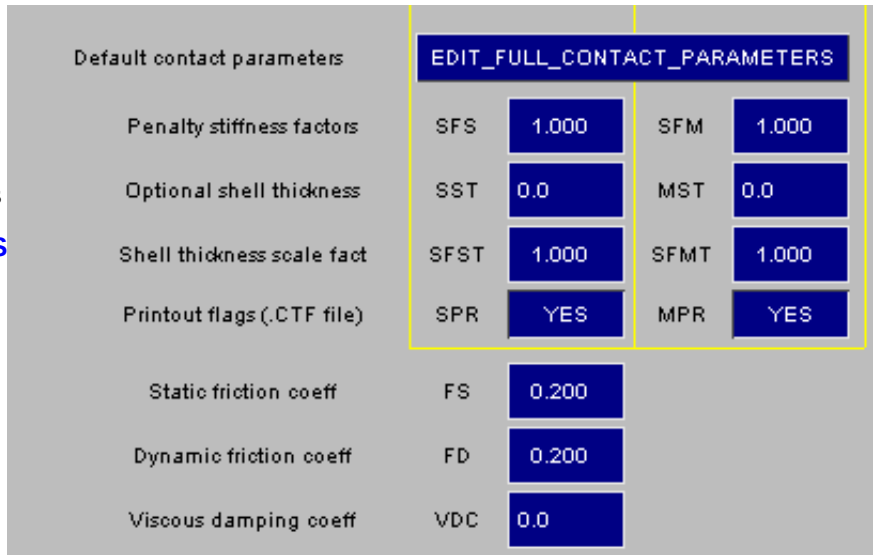


Instead of using **CREATE_ALL** you can create the surfaces selectively using the appropriate **CREATE** buttons. Once created they may be **EDIT**ed or **DELETE**d at will.

To change default settings

When contacts are created automatically the default parameters shown here will be used.

Only the most common ones are given in this box, and to gain access to the full set use **EDIT_FULL_CONTACT_PARAMETERS**



Editing seatbelt contacts once they have been defined

There is nothing special about seatbelt contacts and the sets used to define them: they can be changed, deleted and re-created just like any others, either from this panel or (at any time) from the main **CONTACT** keyword editing command.

For example to remove the nodes on shells from the **NODES_TO_SURFACE** contact in this example edit **SET_NODE** #52 to remove the nodes in **SET_SHELL** #6. And to define a contact between (say) and airbag and the belt it would make sense to reuse **SET_NODE** #52.

Saving belt contact information to file

Any contact surfaces made here will be saved in the seatbelt "tree" file structure, together with their sets. These are references to the surface and set definitions, not the definitions themselves, and are useful for reviewing and editing belt to dummy contact.

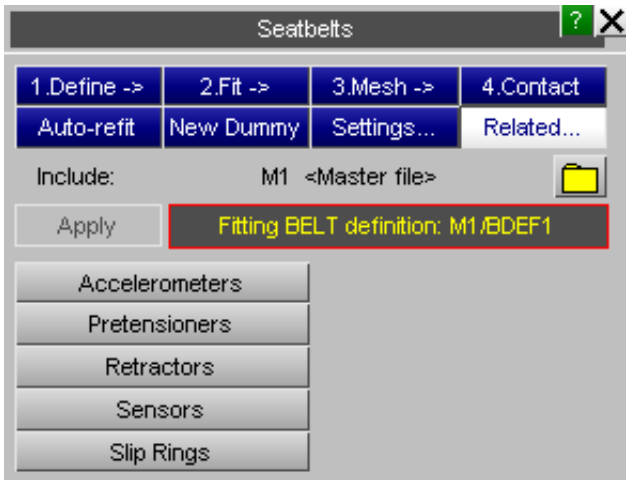
However if a seatbelt has been remeshed the "old" sets are invalid, so both they and the contact definitions are discarded and new ones must be created for the revised geometry.

6.40.6 **Related Items:** Creating Retractors, Sliprings, and other seatbelt-related elements

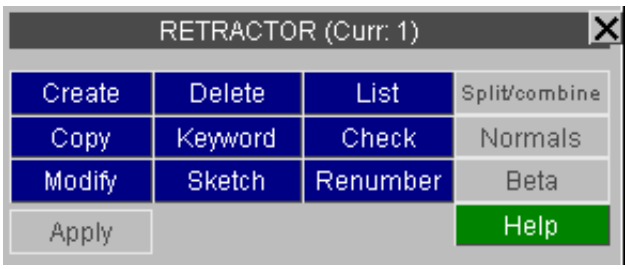
The main seatbelt control panel allows you to create and manipulate the other "seatbelt-related" items:

RETRACTORS	Spool in seatbelt elements, are triggered by sensors.
SLIPRINGS	Feed seatbelt elements through themselves to model material passing from one side to the other.
PRETENSIONERS	Pulls in material to tighten a belt, having been triggered by various means.
SENSORS	Provide a "trigger" for items above by detecting acceleration thresholds, relative movement, etc.
ACCELEROMETERS	Attach to a rigid body and provide accelerations in the frame of reference of that body for post-processing.

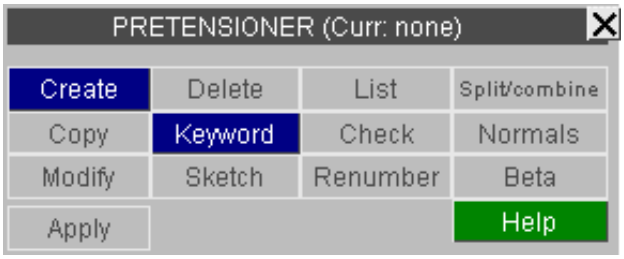
(There is nothing special about creating these elements from inside the seatbelt fitting panel, they may equally well be created from the normal **ELEMENT** keyword.)



Generic top level panel for all types

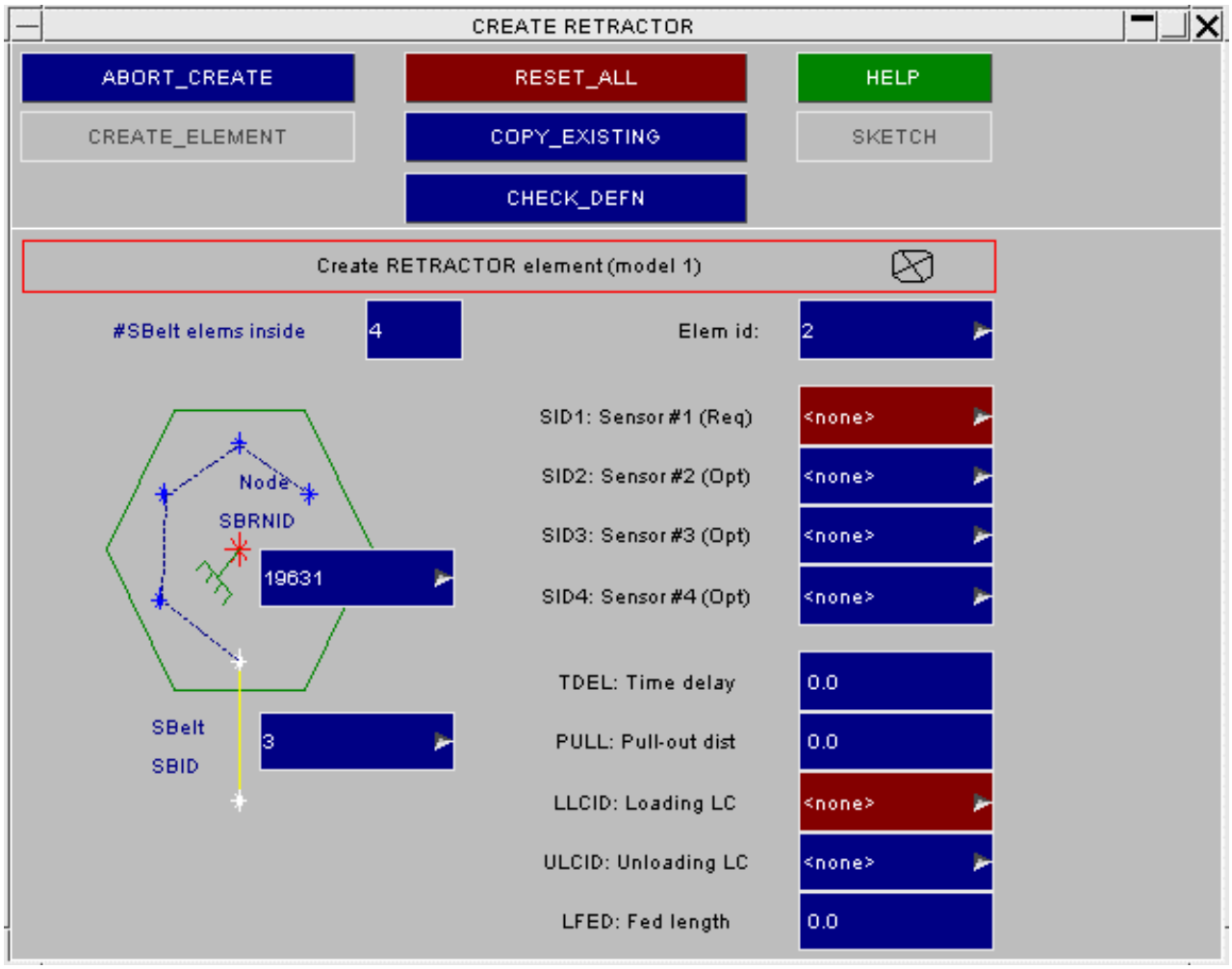


Top **RETRACTOR** panel



Top **Pretensioner** panel

All types have the same options and layout in their top panel, so only two examples are shown. In this example a retractor already exists, so the **MODIFY** and **DELETE** options are available, but no pretensioners have been defined yet, so only the **CREATE** and **KEYWORD** options are available.



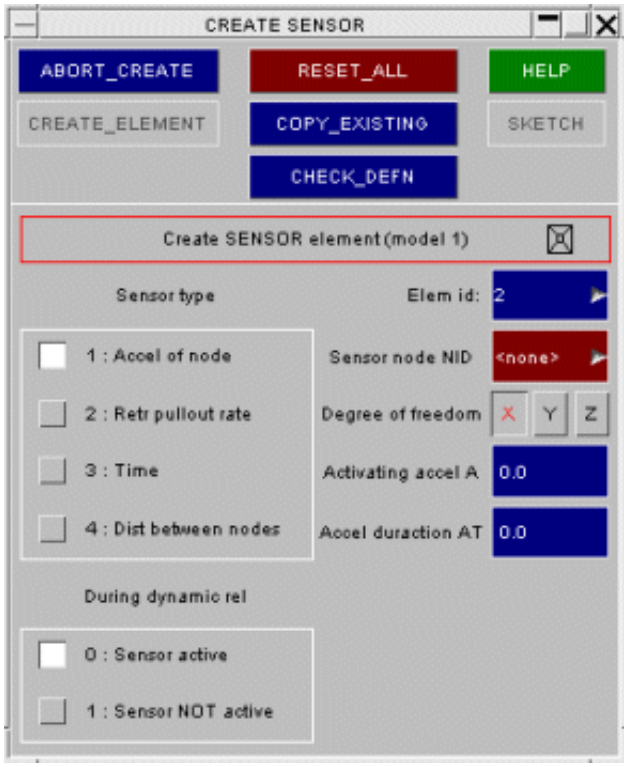
Creating a **RETRACTOR**

This panel shows the process of creating a retractor, with some items still to be defined. The layout and controls are standard for all seatbelt-related types:

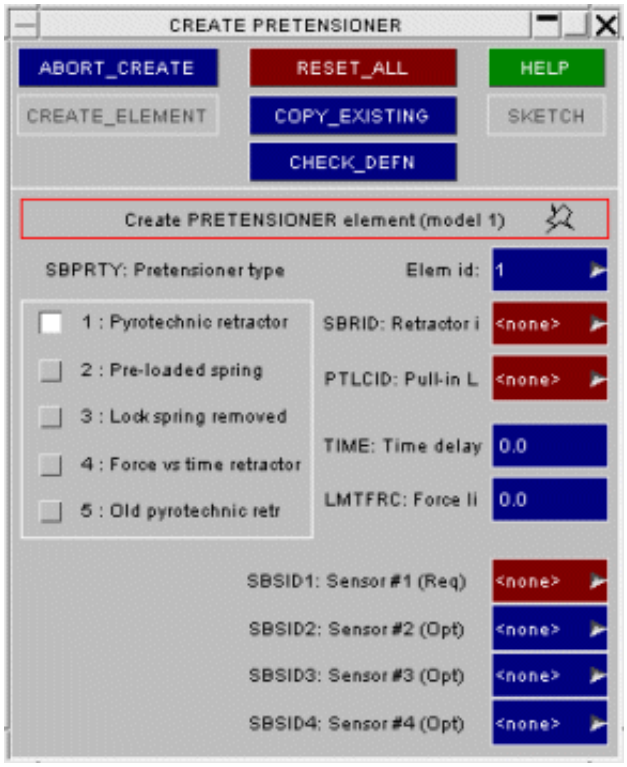
- Boxes with a red background are mandatory data that is missing (here the first sensor and a loadcurve id). The **CREATE**/**UPDATE** button will be "live" only when these missing items have been filled in.
- Boxes with a blue background have already been filled in or are optional data.

The top options

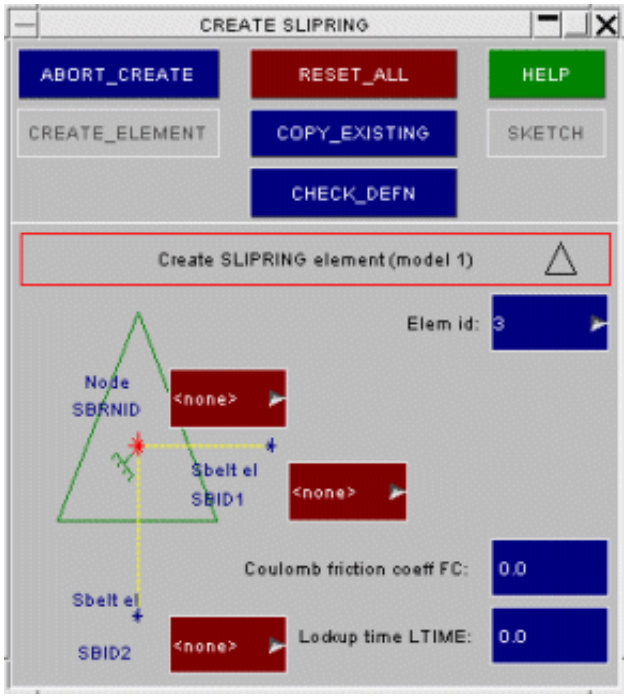
RESTORE/**RESET**	Resets the definition to its original state (modify) or zero (create).
COPY_EXISTING	Copies an existing definition into this one.
LIST_XREFS	Lists what (if anything) references this element.
CHECK_DEFN	Checks the definition so far, listing any errors found.
CREATE/**UPDATE**	Creates a new (create) or overwrites the existing (modify) definition.
ABORT	Abandons this operation leaving any original definition unchanged.



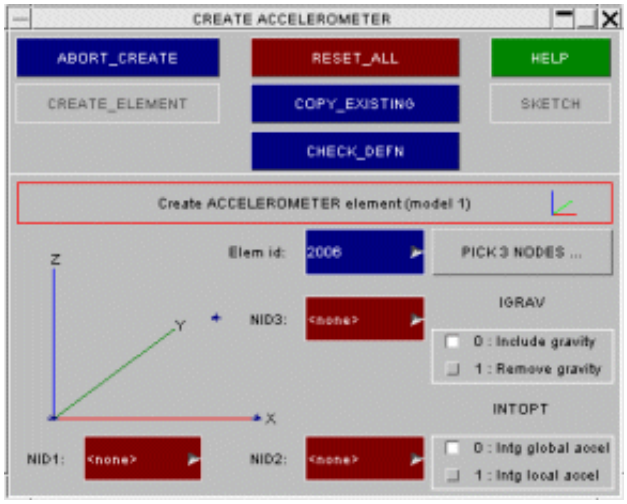
Create SENSORS



Create **PRETENSIONERS**



Create **SLIPRINGS**



Create **ACCELEROMETERS**

These figures show the create/modify panels for the remaining types. All follow the same standard layout, and use the same box colour and top options. With reference to the analysis code user manual the input required is self-explanatory.

The model in which elements are created

When only one model exists there is no ambiguity, but if more than one model is present in the database you will need to define the model in which the element(s) are to be created.

6.40.7 **Auto-Refit**: Refitting a belt automatically when the dummy moves

When a dummy is moved, for example by Occupant or Mechanism positioning, the old seatbelt definition will tend to get left behind, so it will become invalid and require repositioning and remeshing.

Primer can perform this task automatically so long as the belt was previously created in Primer, and the keyword input deck contains **`*BELT_xxx`** cards after **`*END`** as described in section 6.40.9.

How **Auto-Refit** works

When Primer reads in a keyword file containing **`*BELT_xxx`** data it automatically finds the nodes on the "structure" that are nearest to each belt path point, and stores their initial positions. During the refit operation the [dx,dy,dz] movement vectors at these "nearest nodes" are applied to the corresponding belt base path points, thus the path is updated to the new dummy position and the belt-fitting process can be repeated to find a new path around the dummy.

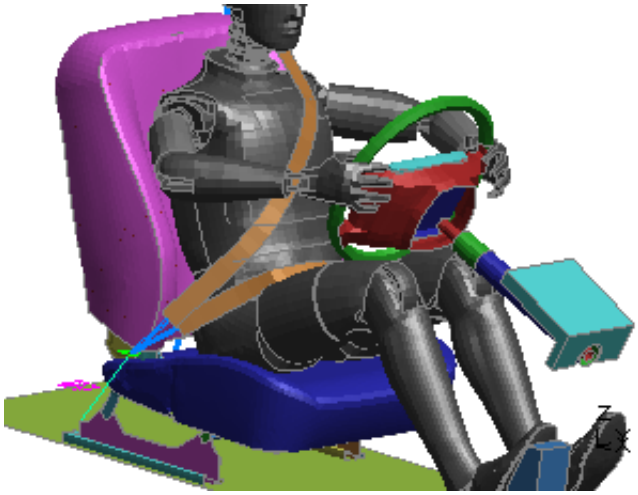
Using the information stored about the belt mesh Primer can then delete the old mesh and, using the same attributes, create a new mesh on the refitted path. This process is not limited to belt elements: it also tracks down each original belt section to find any retractors and sliprings, or direct connections to the structure. If found these are re-used and connected up to the revised belt mesh.

At present contact between the belt and the dummy is not recreated.

The whole process is automatic: provided the dummy shape is not dramatically different, and its movement from its original position is not excessive, a simple **APPLY** should refit and remesh the belt in a single operation.

Auto-Refit example

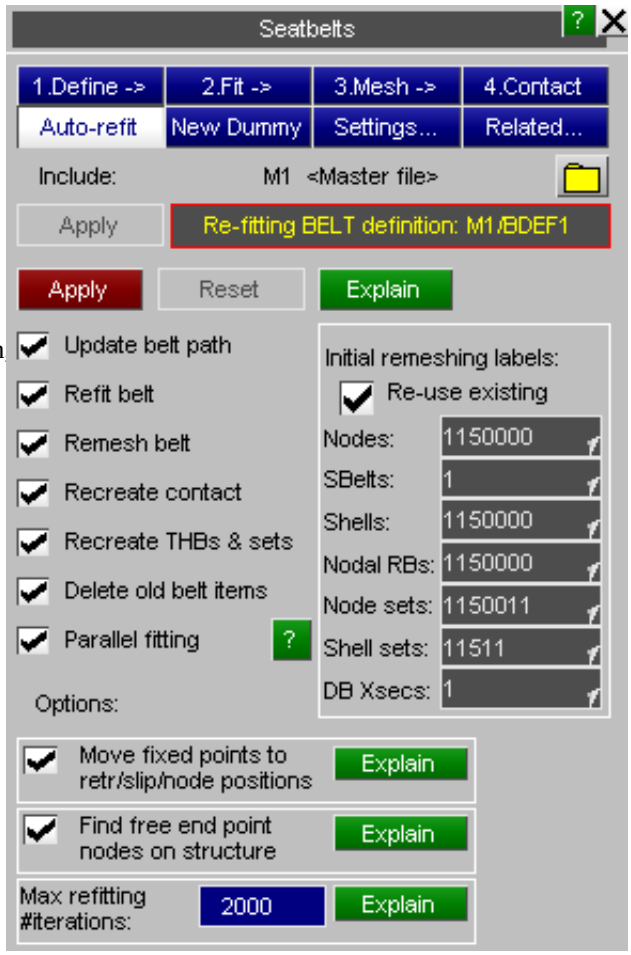
The following model shows a belted dummy both before and after it has been moved.



This image shows the occupant belted up in its mid position.

Here Mechanism positioning has been used to move the occupant and seat forward (by about 50mm) and down. However the belt has been "left behind" and needs to be repositioned.

The Auto-Refit panel:



This panel effectively automates a repeat of the normal belt fitting process using the following stages:

- **`Update belt path`** moves the initial path points to their new locations, computing offsets from the motion of their "nearest nodes".

- **`Refit belt`** performs a new form-finding operation using the revised path.

- **`Remesh belt`** creates a new mesh on the fitted path using the attributes, element types and properties of the original belt. Any existing retractors and sliprings are re-used, and direct connections of belt ends to structural nodes are retained.

- **`Recreate contact`** at present does nothing, and can be ignored. (Experience has shown that users prefer to manage their own belt to dummy contacts.)

- **`Delete old belt items`** will, if selected, automatically delete the old belt elements, nodal rigid bodies, etc once the new definition has been created.

As a general rule you should leave all these options selected. It is only when the automatic refitting goes wrong that it may be necessary to intervene and perform some stages manually.

The following options require a little explanation:

`Parallel fitting` .

This capability is new in PRIMER release 11 and dictates whether or not the fitting (form-finding) process should run in parallel.

- Parallel fitting is much faster, typically 2x to 3x on a 4 core machine. However because the order in which operations are performed is re-ordered slightly the final belt shape can be different to that obtained from earlier releases of PRIMER. The difference is usually small, but in a few cases where belt shape is very sensitive to geometry a significantlt different final shape may result.

- Scalar fitting, which uses the "old" pre-release 11 algorithm, is slower. However the shape obtained will be the same as that from earlier releases of PRIMER.

Users for whom consistency with results from older versions of PRIMER is important may wish to switch this off. Its default setting can be controlled by the preference:

`primer*belt_parallel_fit:  true` *or* **`false`**

`Move fixed points to retr/slip/node positions.`

Sometimes basic belt path fixed end points do not lie exactly at the "structural" position of the (old) as-meshed belt. Typically such points should be at slipring, retractor or structure node positions, but sometimes the end sections of belt meshes are adjusted by hand after fitting, leading to a small difference between "basic path" and "as-meshed" belt element end positions.

For auto-refitting to produce good shapes it is usually necessary in these cases to move the end point of the basic path onto the revised "as-meshed" position, and it is recommended that this option is left selected.

Find free end point nodes on structure.

When the original belt mesh is checked a search is made for any retractors or sliprings at the end of sections so that they can be reused. However it is normally the case that one end of the belt is bolted directly to the vehicle or seat structure, typically at the lower outside end, and this point will not have a retractor or slipring.

If this option is selected then a search will be made for a structural node near to the path end point, with preference being given to a node used in the topology of the end belt element, and this node will be used to terminate the new belt mesh. Again it is is recommended that this option is left selected.

Max refitting #iterations.

This simply acts as an upper-bound on the number of iterations the form-finding algorithm will go through when attempting to find a new shape. If the differences between old and new dummy positions are very great it is sometimes the case that the automatic update of the basic path will produce an initial shape that cannot give a solution, and this limit will stop it iterating for ever! The default value is 300, but it can be changed to any sensible value.

Special rules applying to *SET_NODE labels on 2d sliprings.

LS-DYNA has an input format limitation which means that the labels of node set **NSID** used on 2d ***ELEMENT_SEATBELT_RETRACTOR** cards must be 7 digits or less. (This is a known bug at the time of writing, August 2013, that will be fixed.)

As a consequence users who place Dummy and Seatbelt definitions in an 8 digit label range have to make an exception and use 7 digits for these node sets. This can have unfortunate consequences when refitting belts because the seatbelt fitter uses the following label logic for each category of item (node, element, set, etc):

- Find the first and last labels in the existing seatbelt definition
- Work out how many items of this type will be created in the new belt definition
- Assume that the new definition will start at the existing "first label"
- Check that there is space in the existing label range <first> to <last> for the new items starting at that "first label".
- If there is a potential clash then relabel the existing items in that range to move them out of the way.

This implicitly assumes that label ranges used in the seatbelt definition will be tightly packed, however if the "first" node set label is a 7 digit id used on a slipring, and "last" label is an 8 digit one used elsewhere, then the potential range can become very large with the result that any existing node sets in that range may be relabelled. This can play havoc with carefully thought out labelling ranges, especially for structural items such as ***CONSTRAINED_NODAL_RIGID_BODY**.

Therefore from V11.1 onwards PRIMER applies some special exception logic which works as follows. When the label range for node sets in theexisting belt definition is worked out it is divided into two separate groups:

Group A: All node set labels except those use for field **NSID** on 2d slipring elements

Group B: Only node set labels used for NSID on 2d slipring elements

The ranges of the two groups are then compared, and in the special case of

Group A uses labels of 8 or more digits
and
Group B uses labels of 7 or fewer digits

Then the two groups are kept separate and are treated as distinct ranges, with the "relabel existing out of the way" logic for each group applied independently.

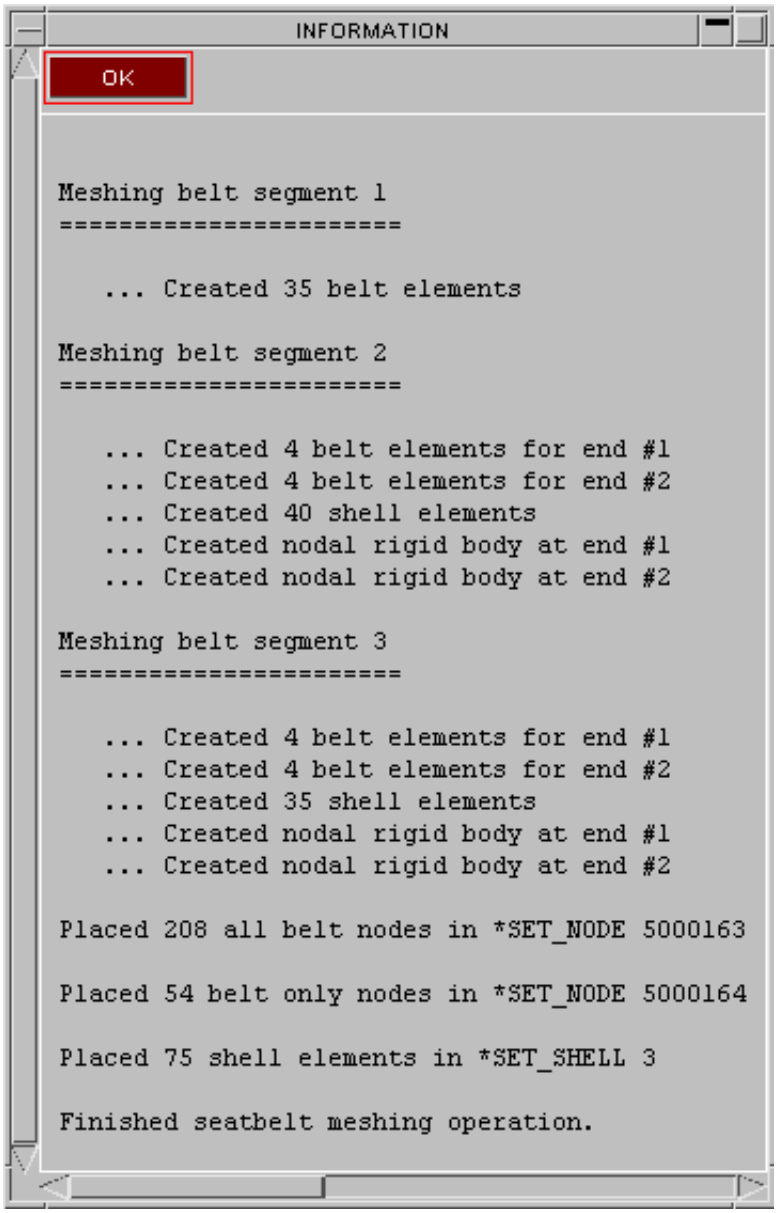
If this is not the case then the two groups are concatenated into a single start / end range, and the normal logic is applied.

APPLY - executing the auto-refit.

Once you have selected the appropriate options press **APPLY** and the refitter will do the rest.

Assuming that all phases work correctly it will fit and create a new mesh, delete the old one (if this option is selected), and the refit will be complete. The only task remaining is to sort out contact between the new belt and the dummy.

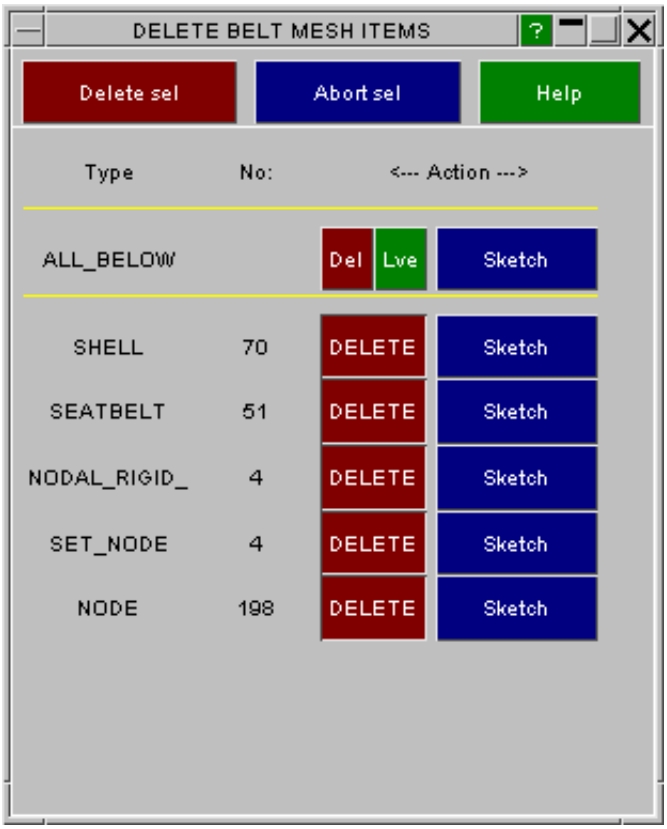
In the example above the following was produced.



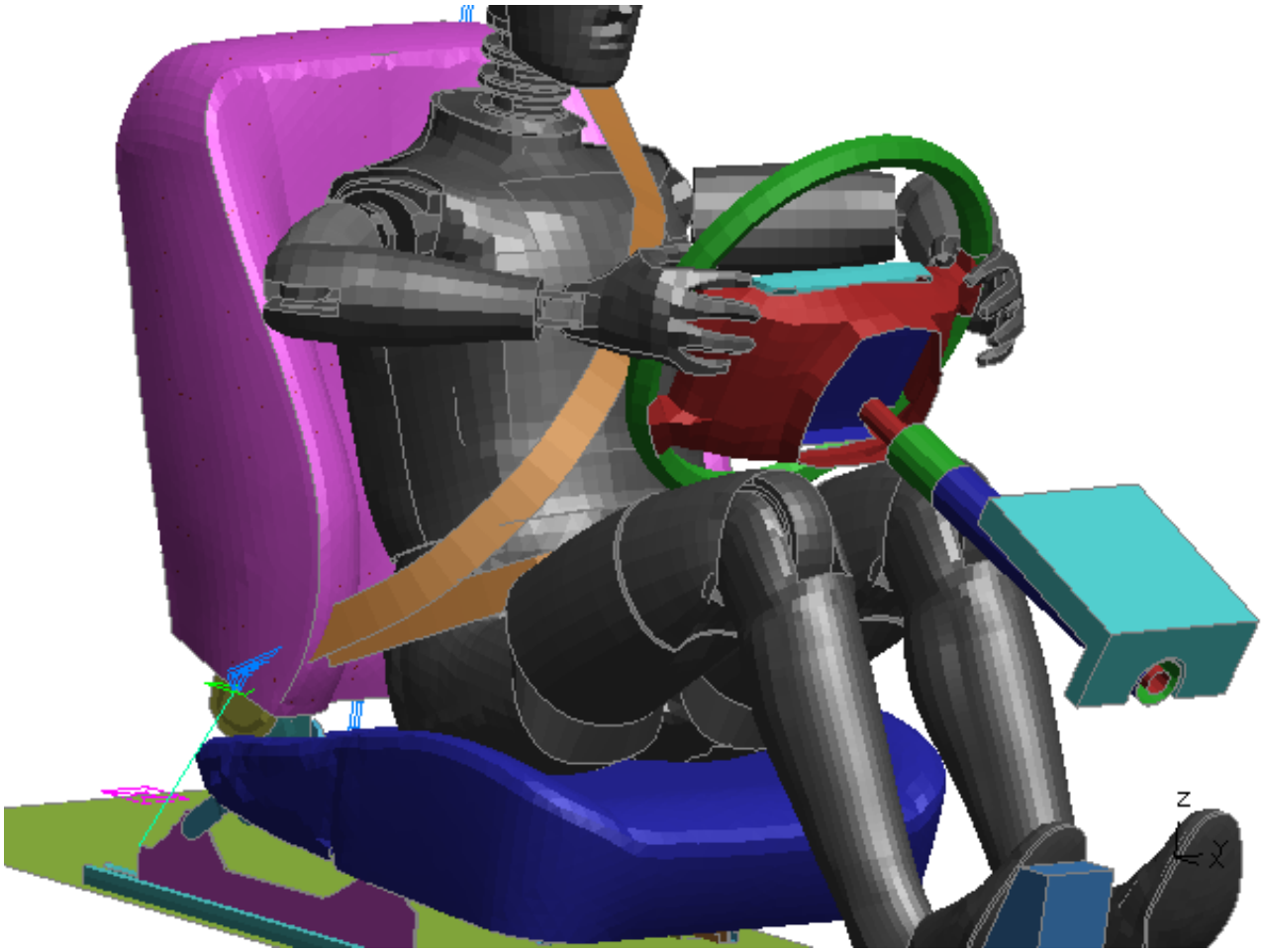
"Delete old belt items" was selected, so the original mesh was also marked for deletion.

Note that this goes through the normal PRIMER deletion logic, so you must confirm it and you can cancel deletion of the old mesh even at this stage if you wish to.

Note also that the 2 sliprings and 1 retractor used in the original belt have not been marked for deletion, since they have been reused.



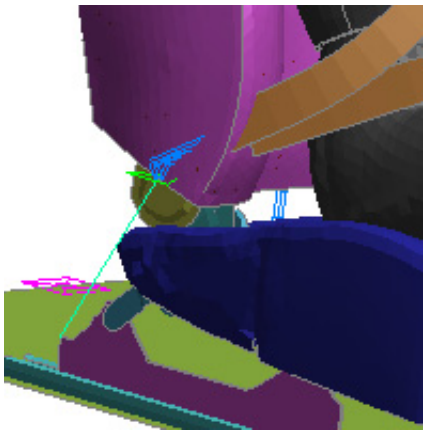
Finally the image is updated to show the new mesh:



If it goes wrong ...

Hopefully it won't, but common problems are:

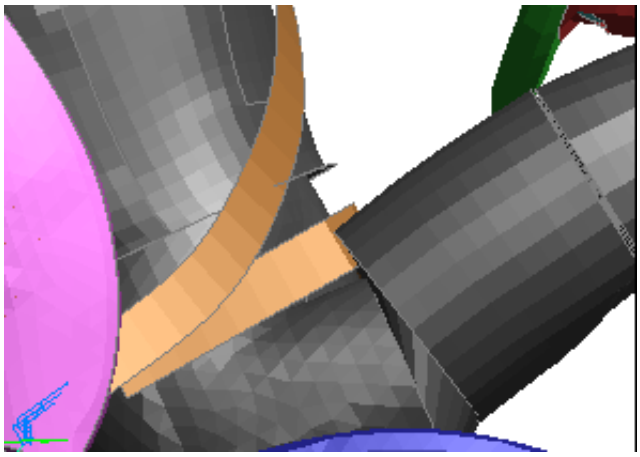
Attachment points are no longer in valid positions



This (artificial) example shows a common problem with belt remeshing following dummy movement: the stalk for the belt buckle has been moved with the seat assembly below it, but this has been insufficient to make the remeshed belt clear the seat back cushion and the belt penetrates it.
It has happened here because the seat base has moved down by a significant amount but the base slider, to which the belt stalk is attached, has only moved forwards and the stalk has not rotated forward to account for this. The vertical movement here is a bit excessive (the dummy looks very uncomfortable in the picture above) but it illustrates the point.

It is a good idea to try to visualise where the end points of a newly refitted belt will end up before refitting, in order to avoid this sort of problem, and some manual movement of belt end positions may be required to prevent it.

The dummy has moved so far from its original position that the "structure" definition needs updating.



A common problem when dummies are moved forwards is that their upper legs move upwards, and belt fitting may need to consider contact between base path and legs where before, in the original position, this was not necessary. This example shows the problem: the upper legs are not in the "structure" definition, so were not considered for contact during refitting and are penetrated by the belt.

Adding the parts of the upper legs to the belt "structure"and repeating the refit may solve this problem.

However very large movements from the initial position may simply not work, in which case it will be necessary to abandon automatic fitting and revert to manual refitting of the belt from scratch.

The belt was originally fitted, then the whole dummy and belt was moved to a new position leaving its "base path" definition behind.

In this situation attempting to reproject from the base path almost certainly will not work because its relationship to the dummy's new position is invalid. It might be possible to "Orient" the dummy base path to re-align it with the dummy's new position, but it will probably be quicker just to refit the belt manually.

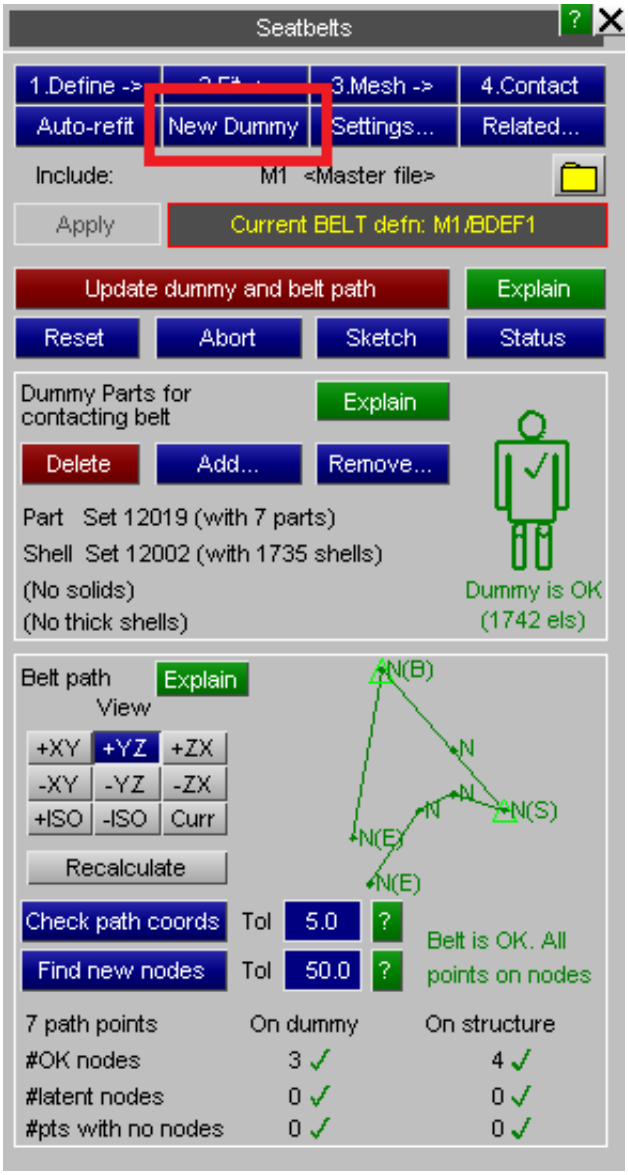
Trying repeated **Auto-Refit** operations

You can go through the cycle:

- Try a refit
- Adjust some part of the geometry or belt definition
- Try another refit

As many times as you like. Each cycle will delete the previous belt and repeat the fitting procedure with the revised definition, making it easy to adjust the solution to get the desired result.

6.40.8 **New Dummy**: Replacing one dummy with another



When an existing belt + dummy combination needs to have dummy A replaced by similar dummy B the following operations are required:

- Remove "structure" definition of dummy A, ie the parts of it to which the belt will fit.
- Insert structure for dummy B

These two operations will normally be done using **INCLUDE**, **Replace Include**. However any method which effectively removes one dummy from the model and replaces it will another will suffice.

Once the new dummy is in place the original belt path will probably fit it approximately, but there will be two outstanding problems:

1. References to nodes at path points will either refer to nodes that no longer exist, or if they do exist they will not be in the same locations.

2. The definition of the "dummy" in the seatbelt fitter, ie those parts of the dummy model that the seatbelt will contact, is likely to be out of date.

The **New Dummy** function helps to resolve these two problems, and in most cases where old and new dummies are approximately the same size and shape it should reduce the problem of fitting the old belt path to the new dummy to a few clicks.

The following sequence demonstrates how to use the facility via a worked example.

Example problem: new 95th %ile dummy replaces old 50th %ile

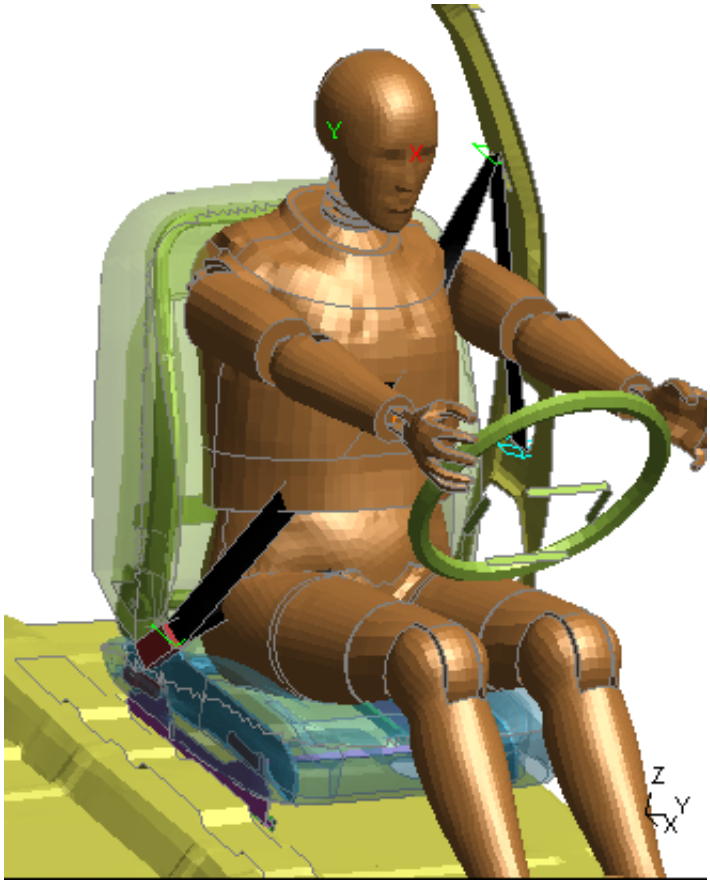
Here a belted 50th percentile dummy has been replaced (via **Include, Replace Include**) by a larger 95th percentile version.

It can be seen that the existing seatbelt (black) now penetrates the chest and the pelvis, so it clearly needs to be refitted.

However if you go directly to the seatbelt fitter and attempt a **Fit** or an **Auto-refit** you will get a message something like this:

```
Failed to find structure 'near'
to point 3 on path after 5
iterations. (Final search
distance = 800.000000.)

This suggests that all the
points on this section of belt
are a long way from the
'structure', making it
impossible to compute sensible
path attributes; or
alternatively the 'projection
distance', (which is used as a
tolerance when searching for
'near' structure), has been
made too small. Please
respecify the path &/or
structure, or increase the
projection distance, and try
again.
```



This has happened because the part sets used to define the belt-to-dummy contact do not exist in the new dummy, and the nodes used to define the path points are also either missing or in totally different locations.

It will be necessary to redefine the parts on the dummy contacted by the belt, and also to move the belt path points back out onto the dummy surface.

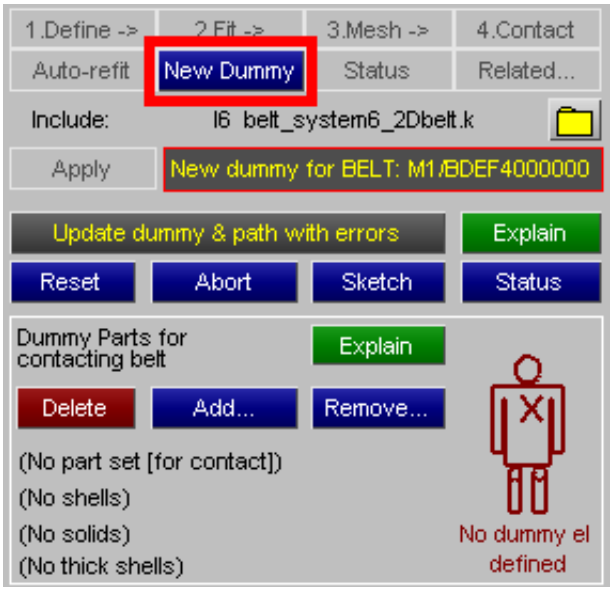
Solution part 1: redefine the "dummy" surface

When you first select **New Dummy** you will see that the "**`Dummy parts for contacting belt`**" section will be empty, and the picture of the dummy will be red with the message "**`No dummy els defined`**".

(***Note***: *If your new dummy is sufficiently similar to the old, for example if the relevant part set labels are the same in both, this section may be green and you will not need to redefine the dummy. However this example assumes the worst case with no useful information being carried over.*)

Use **Add...** to select parts or elements of the dummy to be contacted by the belt.

Points to consider when selecting parts for belt-to-dummy contact are:



- You will need to project the belt path onto the surface of the dummy, so it is best to use shells rather than solids. Most dummies are "skinned" with shells to form a continuous outer surface, and it is recommended that the parts of these shells are used. If solid parts are used there is a danger that the "nearest node" to an existing path path may be inside the dummy, especially if - as in this example - the new dummy is larger than the old one and the existing path is partially buried inside it.

- It is tempting to throw lots of parts into the "dummy" definition, or indeed just to select the whole dummy from the selection menu. However for efficiency and speed of belt form-finding it is best to use the minimum number of parts required to do the job. Typically all you need is a part defining the shoulders and torso, and another part defining the pelvis.

Once you have defined some parts the icon of the dummy should go green:

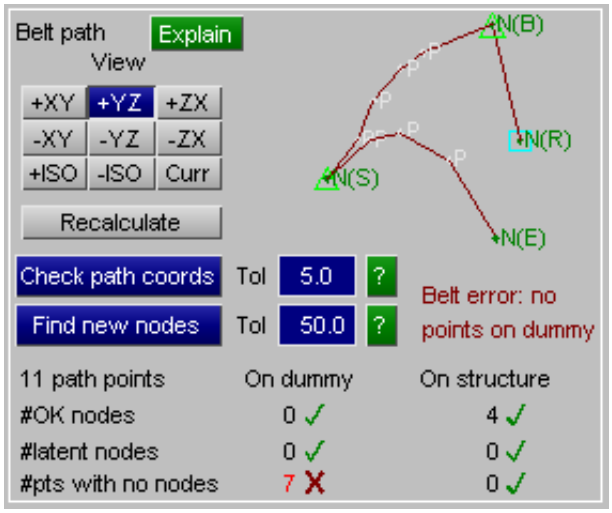
You can check that you have selected a sensible range of elements using **Sketch**, and you can change the selection at any time by using **Add...** and **Remove...** as required. To start again from scratch use **Delete**, which will delete all elements in the the "dummy" definition.

Solution part 2: correct the belt path

Initially it is likely that the Belt Path section of this panel will look like this:

- The end nodes of the path, located on the vehicle structure, will probably have survived the replacement of the dummy unchanged. In this example there are nodes at the Retractor (**R**), B-Post slipring (**B**), pelvis slipring (**S**) and end anchorage point (**E**).

- However all the intermediate points on the belt path have "lost" their associated nodes, since these are either missing or too far away on the new dummy, and have thus reverted to plain path points (**P**) defined only by their coordinates.



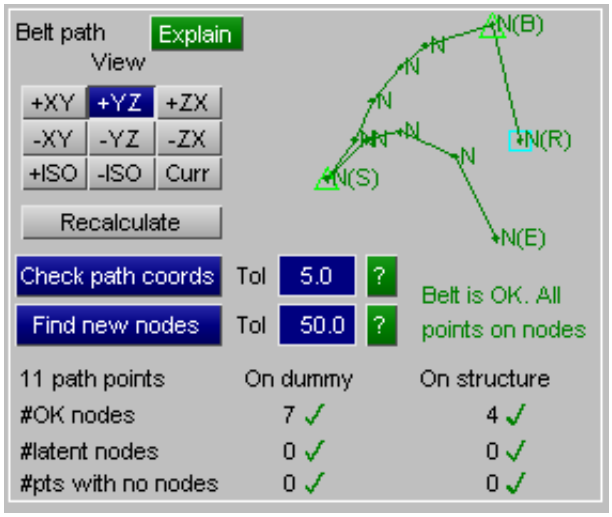
In this context having no path points on dummy nodes is considered to be an error. Technically you could still refit the belt in this situation, and you will be allowed to **Update dummy & path with errors**, but this is not recommended since further manual intervention will be required to fit the belt.

At present most of the intermediate belt points (**P**) are inside the dummy, because the new dummy is larger, and they need to be moved out onto its surface if the belt is to fit over rather than inside the dummy. In addition it is best of the majority of the belt points in the dummy region are defined by nodes on the dummy, rather than just plain points, since that way if the dummy is repositioned its nodes will move, and the path points will move with them.

(*Belt fitting after repositioning the dummy will still work if only some path points are defined by nodes. Those in between, defined only by coordinates, will have these coordinates interpolated from the motion of their their neighbouring points that are at nodes. However this process is more likely to be satisfactory if the majority of path points are at nodes.*)

Your goal is to achieve a result that looks like the picture on the right in which all path points have found nodes on the "dummy" structure as defined in part 1 above.

Each time you change the definition of the "dummy" the path is scanned automatically to try to find nodes near to each path point, and this panel will be updated accordingly. There are tolerances built into this process and you can perform these checks manually with different tolerances as follows:



Check Path Coords	Looks at each path point that has a node associated with it. If the node is latent, or is more than "Tol" distance away, default 5mm, the node is removed from that point.
Find new nodes	Looks for the node nearest to each path point, subject to being within the "Tol" distance, default 50mm. If found the path point is updated to that node and its coordinate.

The default tolerance for Check path coords is 5mm, and for Find new Nodes is 50mm. These defaults may be changed by the following preferences:

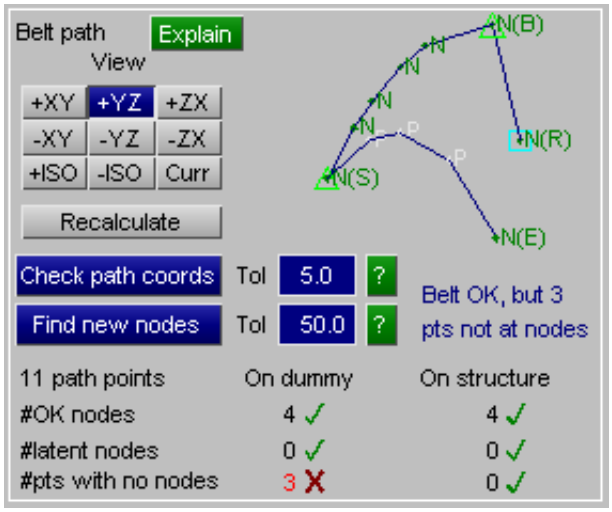
primer*belt_path_match_tol: ***tolerance*** (in mm)
primer*belt_path_nfind_tol: ***tolerance*** (in mm)

The status of what has been found is reported for nodes, and the ideal is that all points in the path find a node on the "dummy" or the vehicle "structure" and have a green tick against them.

Here is an artifical example in which only dummy structure in the torso region has had parts defined, and the pelvis region has been omitted.

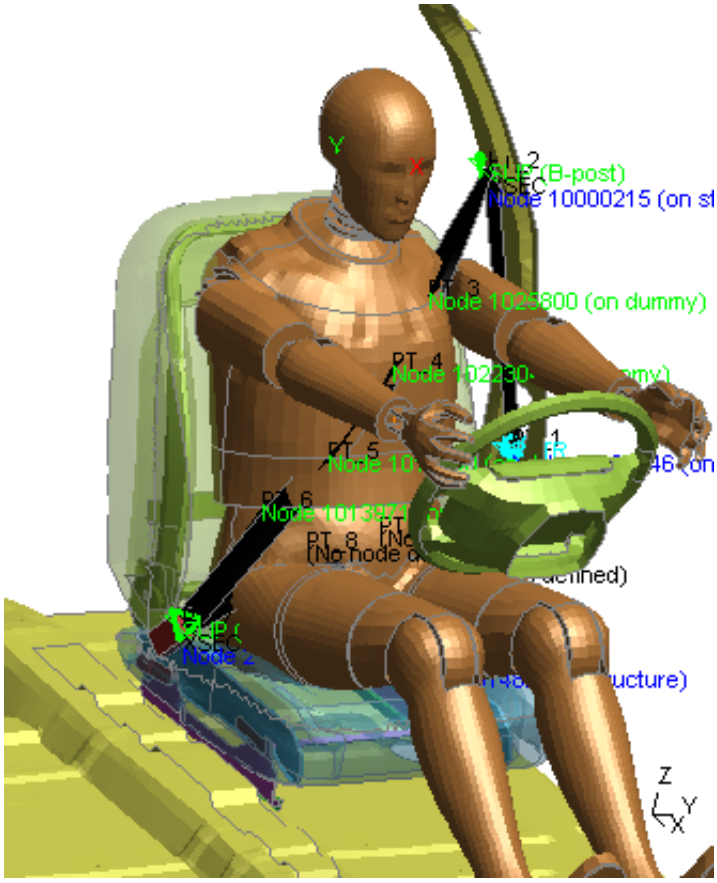
There are 11 path points in total, but only 8 have found nodes: 4 on the dummy and 4 on the vehicle structure. This is marked as "OK" but with the qualification that 3 path points are not on nodes, and these can be seen marked as (**P**) on the path diagram.

In this example this is plainly an error that is easily corrected by adding more "dummy" definition parts, but in other cases it may be acceptable to have isolated path points that are not located at nodes. If you have this situation you will have to examine your structure to determine whether or not the result is acceptable.



If you have trouble identifying where the path points are on the dummy you can use the different views (+XY, etc) to get a better picture.

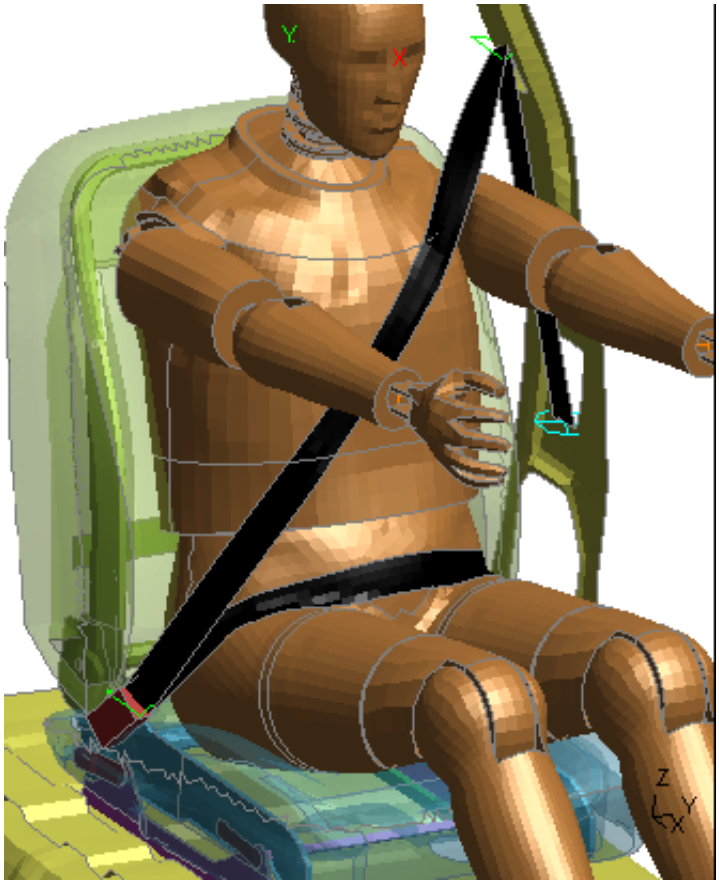
Also for the duration of the New Dummy process the plot in the graphics window will show the current belt path, labelled with point numbers and node status at each point.



Solution part 3: **Update** and **refit**

Once you have defined the dummy and sorted out the belt points you will be able to Update Dummy and Belt Path to save the corrected definition, after which an **Auto-refit** or possibly a manual **Fit** will refit the belt.

Here is the example above after an **Auto-refit**, showing that the belt path has been successfully projected out to the surface of the dummy, fitting round the larger 95th %ile chest.

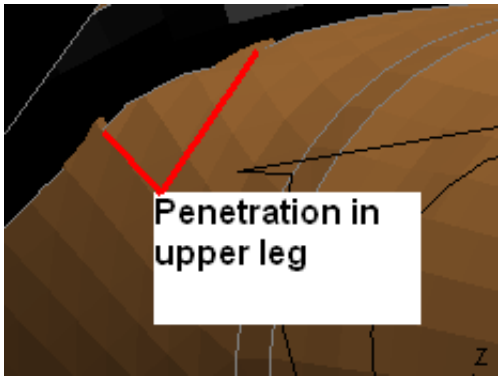


Possible problems, and how to fix them.

Fitting a belt from a smaller dummy onto a large one, as in this example, can result in problems that go beyond plain "matching path nodes to new points" as the following sequence shows.

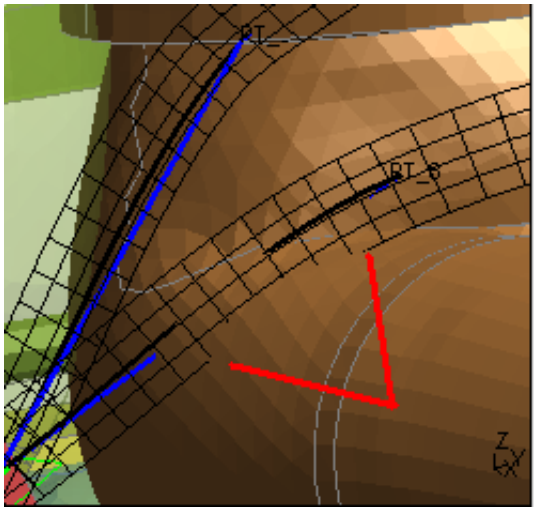
First appearances suggest that the process above has worked first time.

However closer inspection of the end result of this example reveals that the belt is penetrating the upper right hand leg as shown by the close-up here, and this needs to be fixed.



Going into the path editor (**Fit**, **Define Path**) reveals that the twist of the fitting path in this upper leg region is wrong, resulting in the initial path penetrating the upper leg.

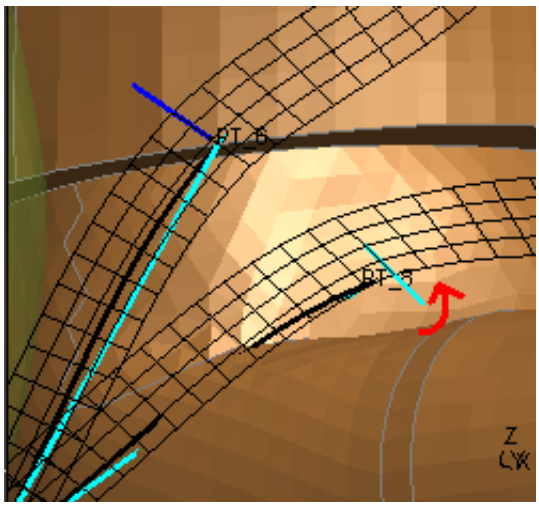
This means that this section of the belt starts off on the "wrong" (inside) surface of the leg and does not pull itself out again.



Using the path editor in **Control Twist** mode that point #6 is rotated up out of the leg. Once corrected another refit now positions the belt successfully around the dummy.

This is a particular case of the more general problem that moving a belt path from one dummy to another can pose problems that this **New Dummy** function cannot always solve.

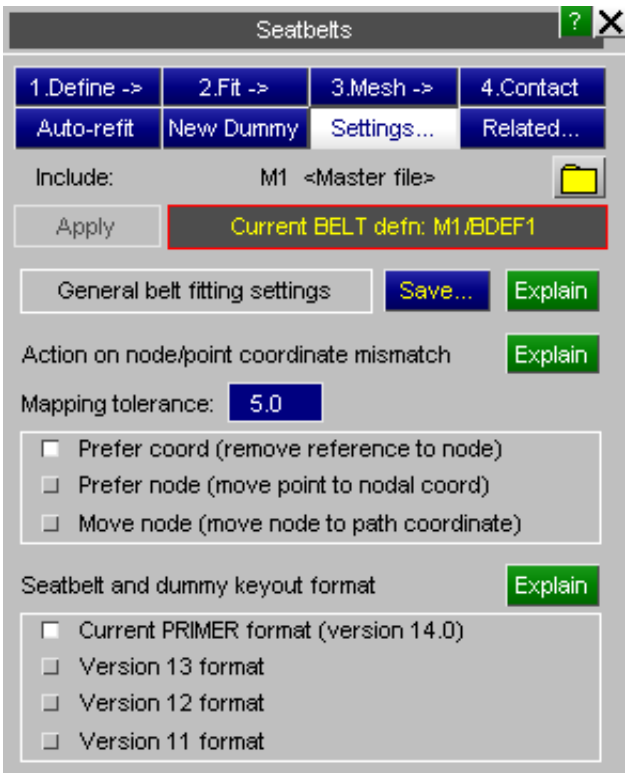
It is a help, but not a panacea - ***please always check the end result!*** And be prepared to pay a visit to the belt path editor to adjust the belt path if necessary.



6.40.9 Settings...

Some miscellaneous settings that control belt fitting are available here.

Preferences may also be saved automatically to you $home oa_pref file using **Save...**



Action on node/point coordinate mismatch

When the *BELT_PATH keyword defining the points that make up the basic belt path are read they contain both an explicit coordinate <x1, y1, z1> and an optional node <nid>:

***BELT_PATH**
<npts>
<bits> <x1> <y1> <z1> (<nid>)
(Optional row 1 of further data depending on <bits>)
(Optional row 2 of further data depending on <bits>)

Prior to V14 PRIMER applied a hard-wired tolerance of 5mm when comparing the coordinates of node <nid> with the explicit coordinates <x1, y1, z1>, and the action taken was as follows:

Pre-V14 behaviour when dealing with node / point coordinate differences.	
Point within 5mm of node	Move point silently to nodal coordinate
Point more than 5mm from node	Remove reference to node, use point coordinate

From PRIMER V14 onwards this behaviour is now much more controllable as follows:

The mapping tolerance can be controlled

The default of 5mm can be changed to any reasonable value. (Note that this value is the same as that used in New Dummy when looking for nodes on the new dummy onto which to map the existing belt path.)

This can also be set by the preference shown.

Preference: **`primer*belt_path_match_tol:`**

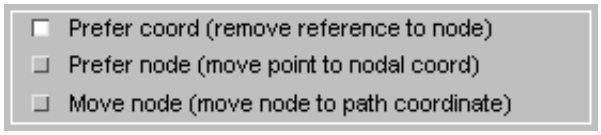
<value>

The way point / node coordinate mismatches are handled can be controlled

Three behaviours are available if the point and node coordinates do not match within the given tolerance:

Prefer coord	Pre V14 behaviour. The reference to the node is deleted and the point coordinate is used. For backwards compatibility this is the default.
Prefer node	The node is used regardless, and the path point is moved to the node's cordinate.
Move node	The node is used regardless, but the coordinate of the node is moved to the path point.

The behaviour if the node and point are within tolerance is unchanged, ie move point to node coordinate.

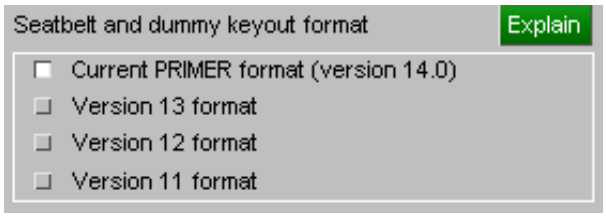


Preference:
`primer*belt_path_match_method:`

`prefer_coord`
`prefer_node`
`move_node`

Seatbelt and dummy keyout format

Controls the version of PRIMER for which the post *END cards written for ocupant related data are formatted.



This setting affects the following PRIMER-specific keywords:

***BELT_xxx**	Seatbelt fitting information. See appendix V.
***DUMMY_xxx**	Dummy positioning. See appendix IIa
***MECHANISM_xxx**	Mechanism positioning. See appendix IIb

Warning: Choosing a format for an earlier version of PRIMER will attempt to format the relevant cards for that version, but it will inevitably mean that some data are lost and as a consequence that behaviour may not be the same. In some cases, for example "new" versus "old" style belt meshing, it may not be possible to express the data from the current PRIMER version in the older format. As a consequence it is recommended that this option is only used as a last resort.

This output format may also be set by the preference:

primer*mdumm_keyout_format: current | V14 | V13 | V12 | V11

6.40.10 Saving Seatbelt Definition data to file, and its use for re-meshing

Seatbelt definitions are automatically saved in LS-DYNA format output files by appending extra keywords after the LS-DYNA *END card. The keywords are:

***BELT_START**	Giving label, title, structure sets and other key information.
***BELT_MESH**	Giving the dimensions and parameters used during meshing.
***BELT_PATH**	Giving the coordinates and attributes of all basic path points.
***BELT_PARAMETER**	Giving the name of the parameter updated with the total belt length
***BELT_END**	Terminates the definition.

Details of the format are given in Appendix V, but users should avoid editing these sections since errors may cause internal inconsistencies. See Seatbelt and dummy keyout format above for information on how to format these cards for earlier versions of PRIMER.

Deleting these sections from the end of a file is legal: the analysis will still run, but PRIMER will not "know" about the belt definitions when the file is reread.

Writing *BELT cards for earlier PRIMER versions

As PRIMER has evolved so more data fields and lines have been added to the *BELT cards, and this can mean that an output deck written from a later version of PRIMER may not read into an earlier version. If it is necessary to export such a deck to an earlier PRIMER version it will be necessary to set the following preference to the appropriate version:

primer*mdumm_keyout_format: V14 | V13 | V12 | V11 | CURRENT

If this preference is not defined the default will always be CURRENT, ie the current version.

Remeshing existing belt definitions

When a file containing this extra ***BELT_...** data is read back into PRIMER the belt definitions will be created automatically. This makes it possible to re-mesh a seatbelt either automatically or by hand.

Auto-Refit: Remeshing a belt automatically

This is the preferred method, and will recreate both belt and any sliprings and retractors in the new dummy position. It is covered in the section 6.40.7 .

Manual refit: Remeshing the belt by hand.

It is possible to repeat the refitting and remeshing process by hand as follows:

1. Only the basic path is stored in the file, so it will be necessary to repeat the **FITTING** operation to obtain a "chassis" mesh before remeshing can take place, see "Changing and remeshing an existing belt definition" for more details.

2. It is not possible to reposition a belt by "adjusting existing nodal coordinates", even if the amount they need to move is small (although it might be feasible to use **ORIENT**, **TRANSLATE** instead with suitable interpolation). You must generate a new mesh each time, usually deleting the old one.
   The reason is that since a characteristic element length is used, moving the chassis mesh by even a small amount may change the number of elements in a segment, and then mapping the new shape onto the old mesh would be impractical.

3. Manual intervention during remeshing will usually be required at retractor and slipring locations, since these element types will "lock" connected seatbelt elements preventing their deletion. The simplest solution is to create the new mesh anyway, then to edit the retractor locations replacing the "old" seatbelt elements with the "new" ones, and then to delete the redundant old ones.

Now that automatic refitting has been added to Primer it is recommended that this be used instead.

6.40.11 Command-line (batch) commands for seatbelt fitting

A limited subset of the operations described above can be performed in command-line mode. These are intended to permit batch mode refitting of an existing belt definition, and they also include the ability to modify some aspects of belt geometry and fitting.

BELT	Refit an existing seatbelt to a dummy. Primer will select a seatbelt definition in the model automatically. Available options are:

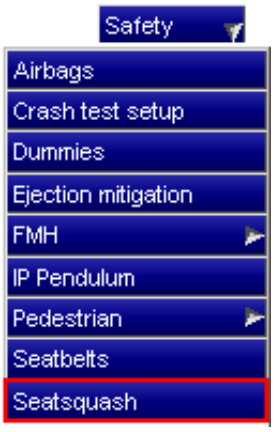
SELECT	Select a different belt definition for refitting
REFIT	Refits the current belt definition to its dummy
PR_BASIC	Retrieve and update basic belt dimensions and form-finding parameters
PR_REFIT	Retrieve and update belt refit parameters
DONE	To return to the main menu prompt

6.41 SEAT FOAM COMPRESSION

Positioning a dummy to its H-point position will generally lead to penetration with the seat foam components.

In PRIMER release 9.3, the user has the capability to deform the foam under the dummy and remove seat foam penetrations.

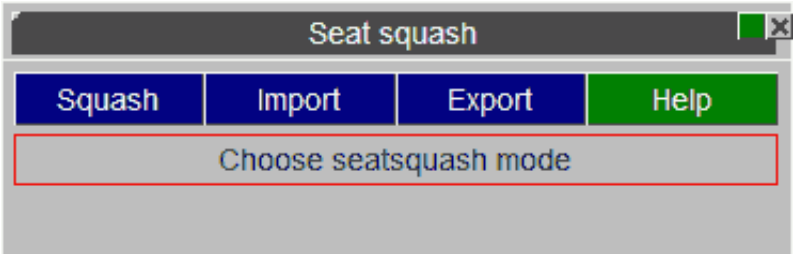
To access the Seat squash menu, click on the **Occupant** button and, in the drop-down menu, select the **Seatsquash** tool.



Three options are available in the seatsquash panel.

- **Squash**
- **Import**
- **Export**

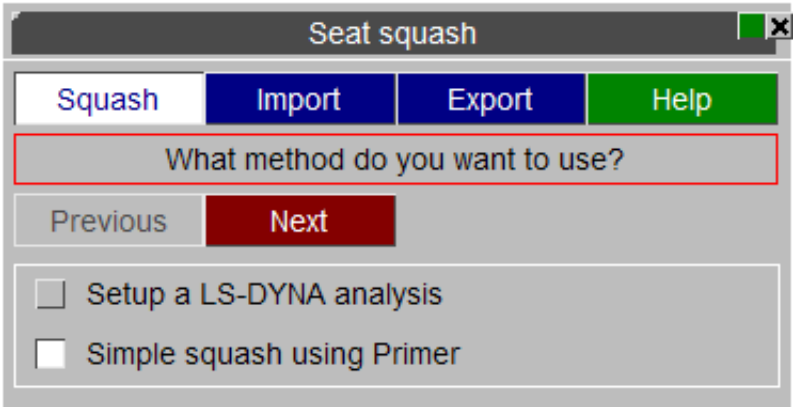
They are described in the following sections.



6.41.1 Undoing a seatquash operation

There is no 'undo' button in the seatsquash. However, if before you start doing a seatsquash you export the coordinates of the seatfoam (and other parts if required) you can undo the seatsquash by importing these coordinates back into PRIMER.

6.41.2 Squash options



Press the **Squash** button to start a seatsquash.
Two methods are available to deform the seat foam:

- **Setup a LS-DYNA analysis**
- **Simple squash using Primer**

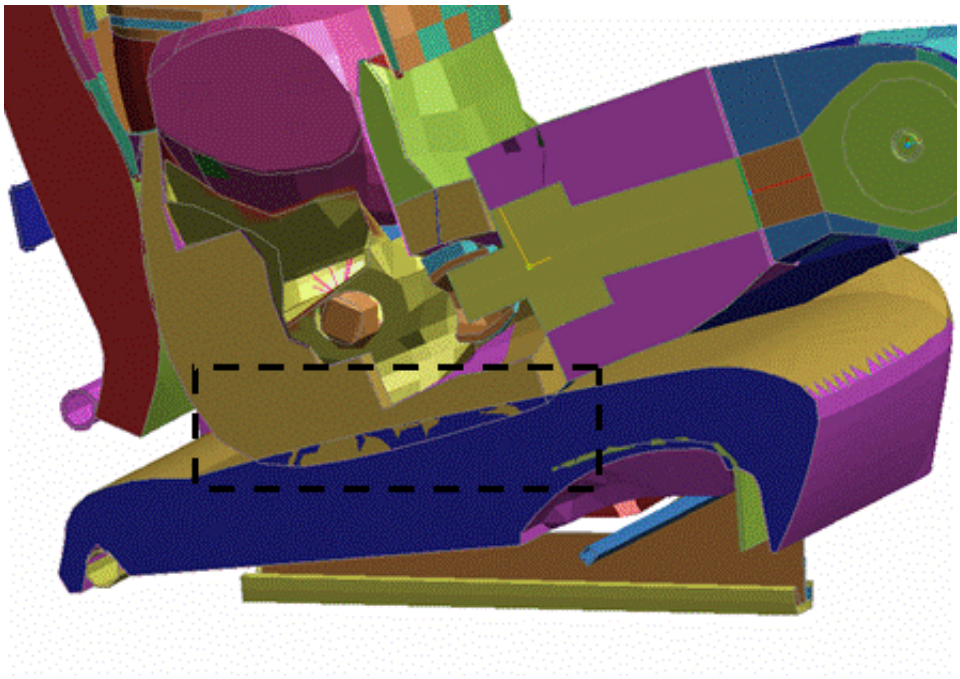
The PRIMER method uses the contact depenetrator in PRIMER to push the dummy into seat. The seat is deformed uniformly through its thickness. This obviously will not have the correct material response but it is meant as a quick method. If the seat deformation is critical then you should use the LS-DYNA method.

The LS-DYNA method will create an LS-DYNA import deck which will push the dummy into the seat. This should be run using LS-DYNA and the dynain file which it creates can be imported back into PRIMER to deform the seat.

Select the option you want and press **Next** to start the process.

Simple squash using PRIMER

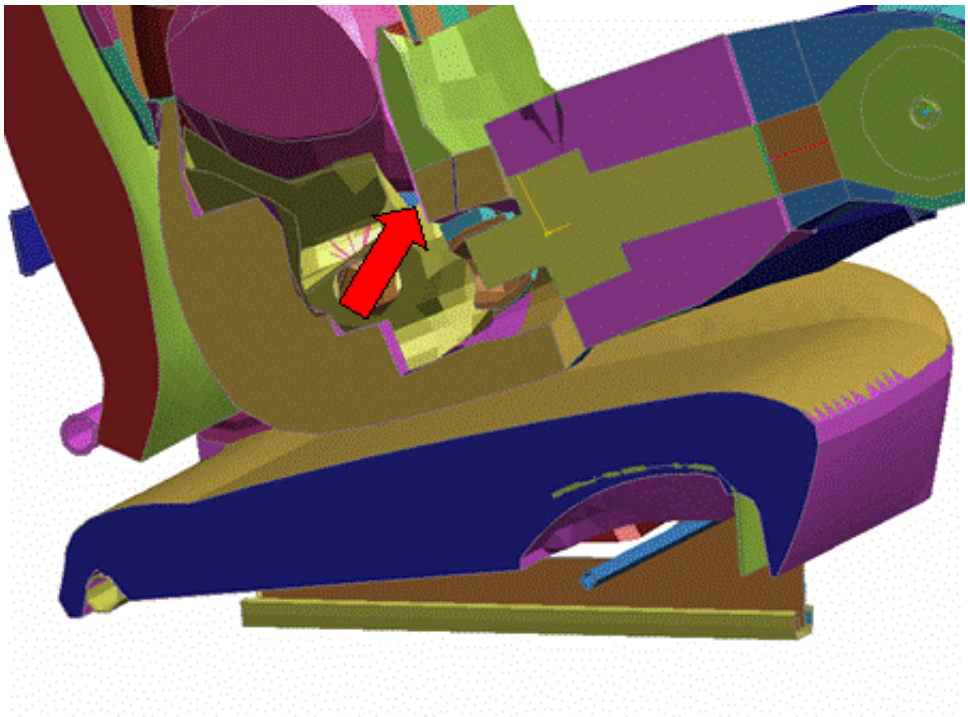
Process description



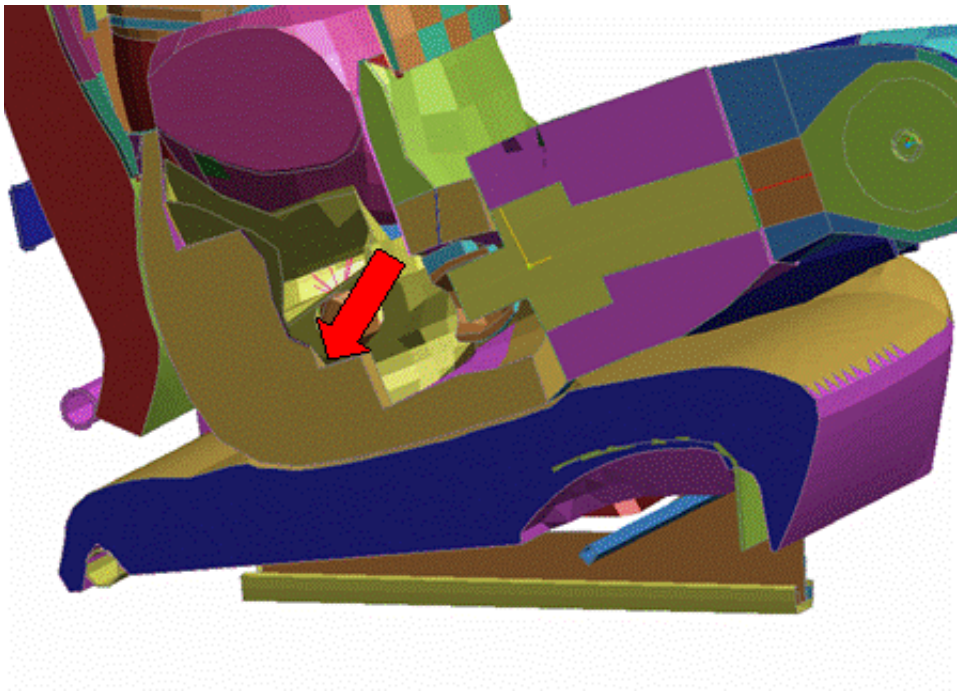
Initial state: model with penetration

Cut section through dummy and seat showing initial penetration between dummy components and seat foam.

The dummy is at the correct H-point but penetrates the seat.



The dummy is moved from its initial H-point position, following the direction prescribed by the user until the contact with the seat top shell part has no penetration.



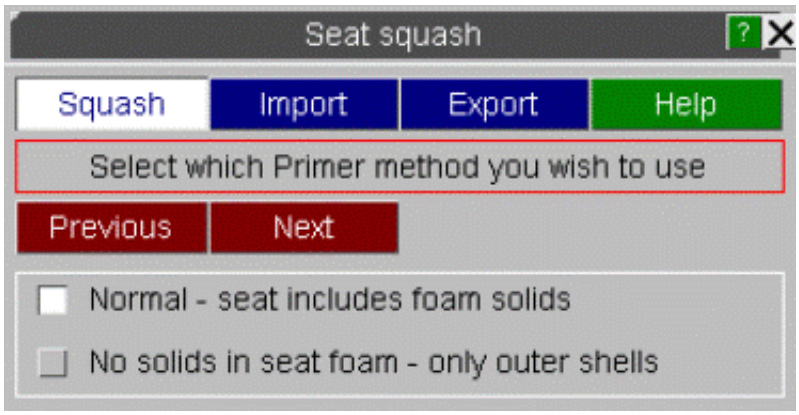
Final state: Seat foam compressed

The dummy is moved back to its initial H-point position by increments while using Primer contact depenetration option. The seat foam is progressively squashed under the dummy. Interior nodes within the foam components are also displaced to uniformly distribute strain.

Step 1

There are two types of simple seat squash. The first type is where you specify the solid elements in the seat, and the solids are deformed during the compression. The second type does not consider the seat solid elements. Use the second type if your model does not contain seat foam solids, and you just wish to deform the outer shells of the seat and mesh the solids after the deformation.

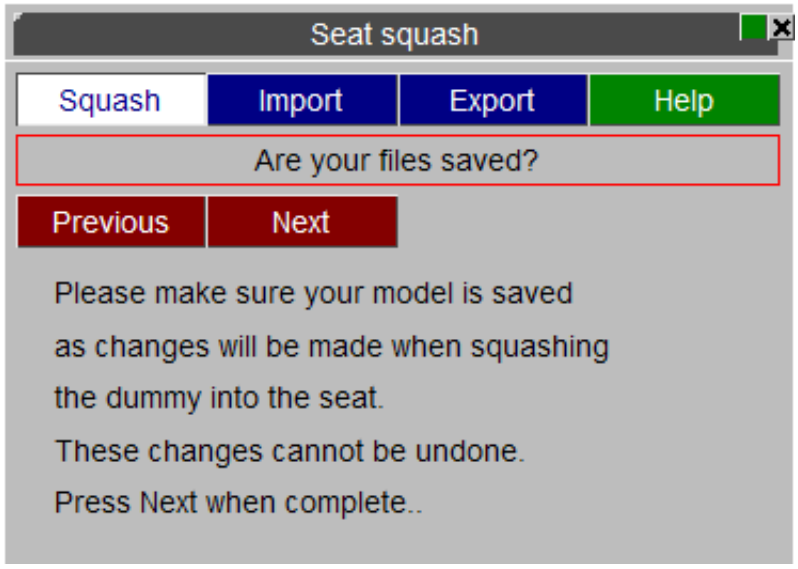
Once you have chosen the method you wish to use press **Next**



Step 2

Before you go any further you should save your model. PRIMER will prompt you to save your model as the seat squash changes are irreversible (although you can import coordinates from a dynain file to effectively do an 'undo' See section 6.41.1 for more details).

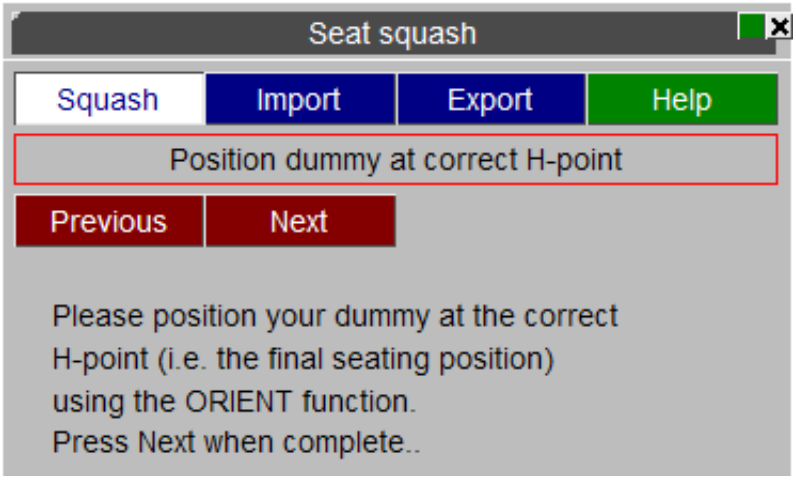
Once you have saved your model press **Next**



Step 3

You will be asked to move your dummy to the correct H-point location. Position the dummy by either using the orient menu or dummy positioning menu.

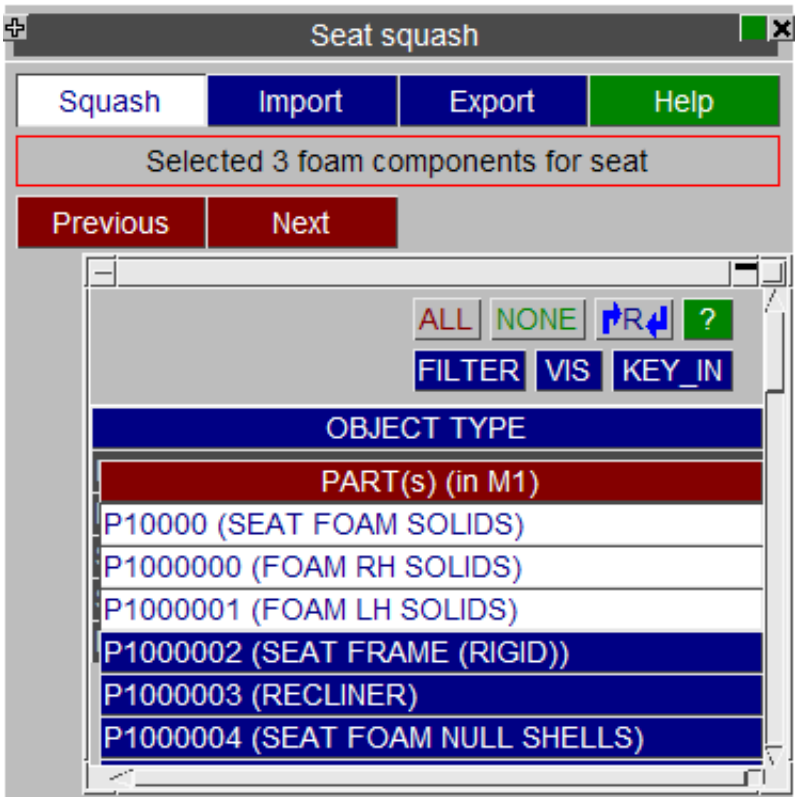
Once this is done press **Next**



Step 4

Select the foam parts of the seat using the standard object menu. These are the parts that PRIMER will squash the dummy into. This step is only available if you have chosen the "normal" method in step 1.

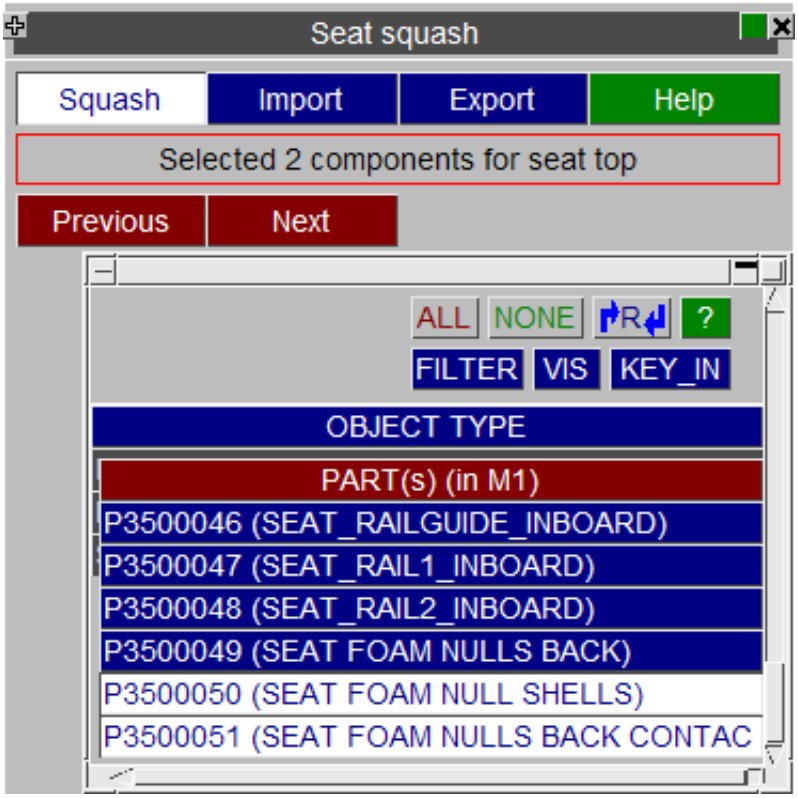
To continue press **Next**



Step 5

Select the coating shell parts on the top surface of the seats. These can either be defined by parts or by sets of parts.

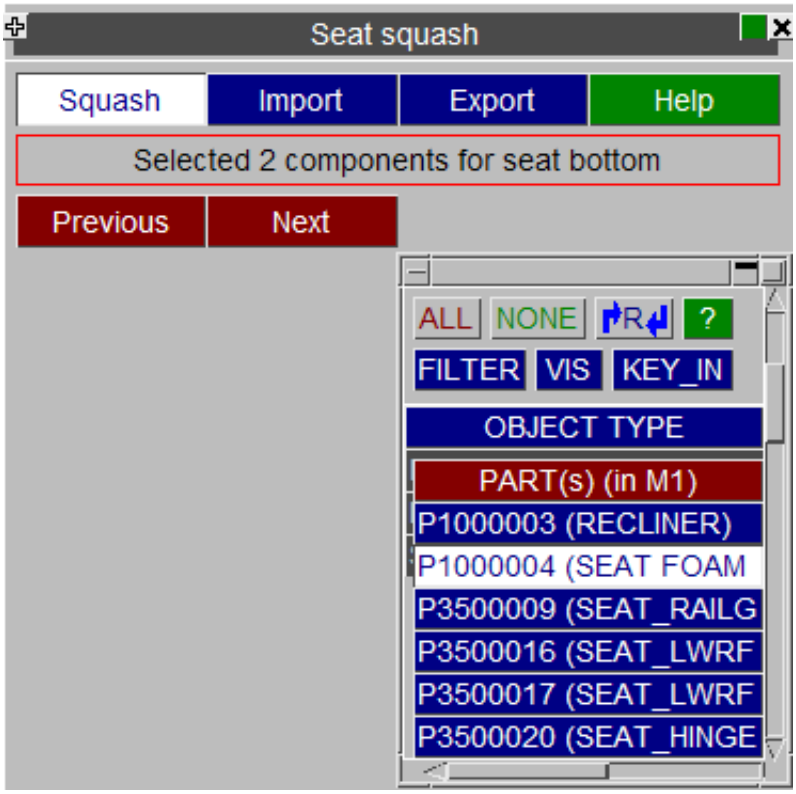
To continue press **Next**



Step 6

Select the coating shell parts on the bottom surface of the seats. These can either be defined by parts or by sets of parts. These parts will be fixed. PRIMER Will deform the seat evenly as required between the top and bottom surfaces of the seat.

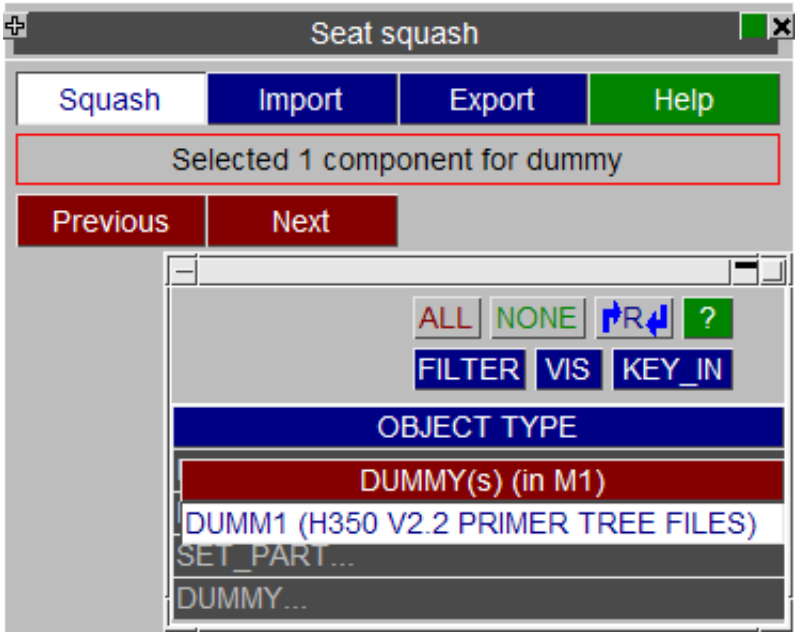
To continue press **Next**



Step 7

Select the dummy components. You can use the **DUMMY...** option in the standard object window if required..

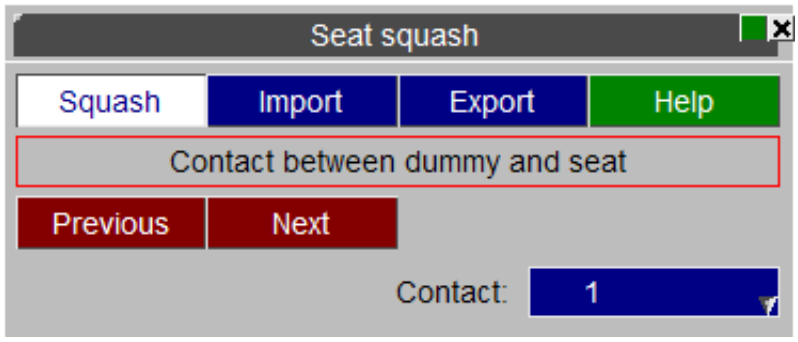
To continue press **Next**



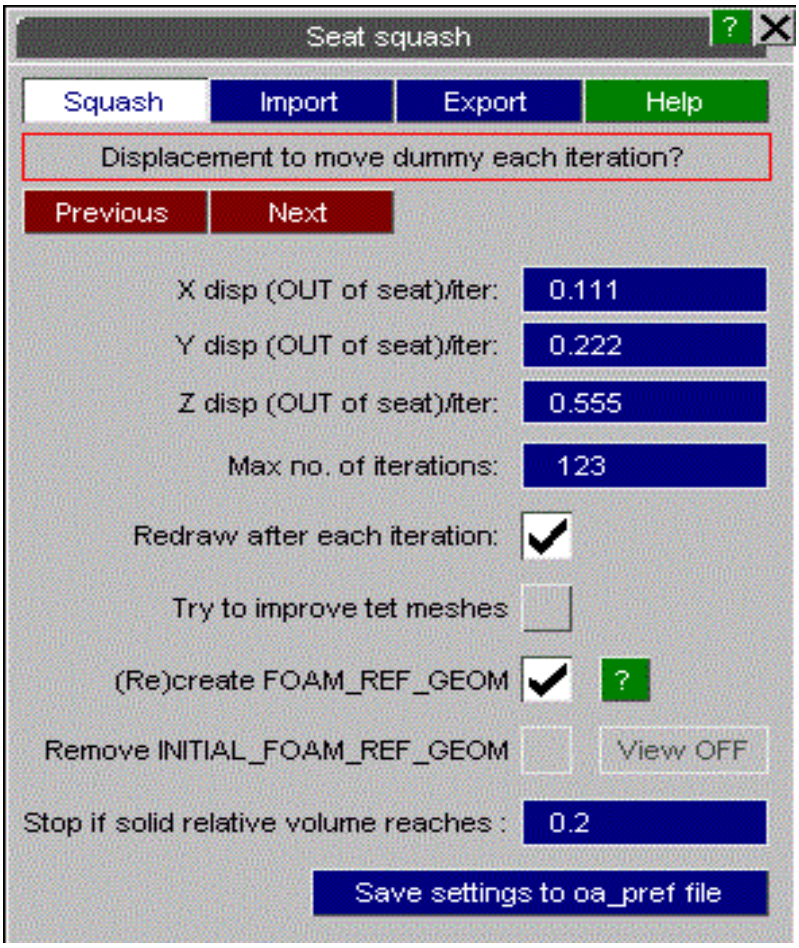
Step 8

To select the contact between the seat foam and the dummy. You can use the standard popup functions (right click). You can then Pick, Select an existing contact or Create a new one. Make sure that your contact uses a sensible thickness. This is what PRIMER will use when pushing the dummy into the seat. If the thickness is very small then you will have to have a small increment per iteration.

Once you have selected/created the contact press **Next**



Step 9



Define the dummy displacement increment at each de-penetration iteration. Give the X, Y, and Z displacements that the dummy will move per iteration to move the dummy out of the seat. Once PRIMER has moved the dummy out of the seat enough to eliminate any penetrations it will reverse the motion, squashing the dummy back into the seat to the original position.
If the displacement per iteration is bigger than the contact thickness chosen in the previous step, PRIMER will scale it down.

You can set the maximum number of iterations that PRIMER will try to do when moving the dummy out of the seat.

If you want to see the progress of the seatsquash then select the **Redraw after each iteration** checkbox. This will make the process much slower so if you want, you can turn it off.

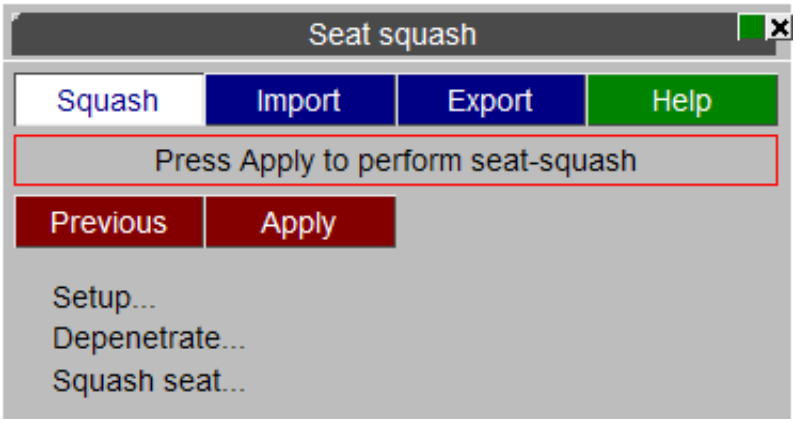
Some tet meshes can be very badly deformed making it very easy to make badly deformed elements. PRIMER can try to 'smooth' tet meshes to make them better. This may help if you are having problems squashing a dummy into a tet meshed seat.

You can opt to create *INITIAL_FOAM_REFERENCE_GEOMETRY cards for the nodes in the seat foam before the deformation. This is only available for hyperelastic materials and certain solid element formulations. Note the REF field on the appropriate material card will be set to 1.0 upon creation of the *INITIAL_FOAM_REFERENCE_GEOMERTY cards.

The minimum value of relative volume for the seat foam solid elements is by default set to 0.2. If any solid element becomes excessively deformed and reaches this threshold, the seat squash process will stop. You can modified this value if you wish. Finally, it is possible to save settings by using the button **Save settings to oa_pref file**.

Then press **Next**

Step 10



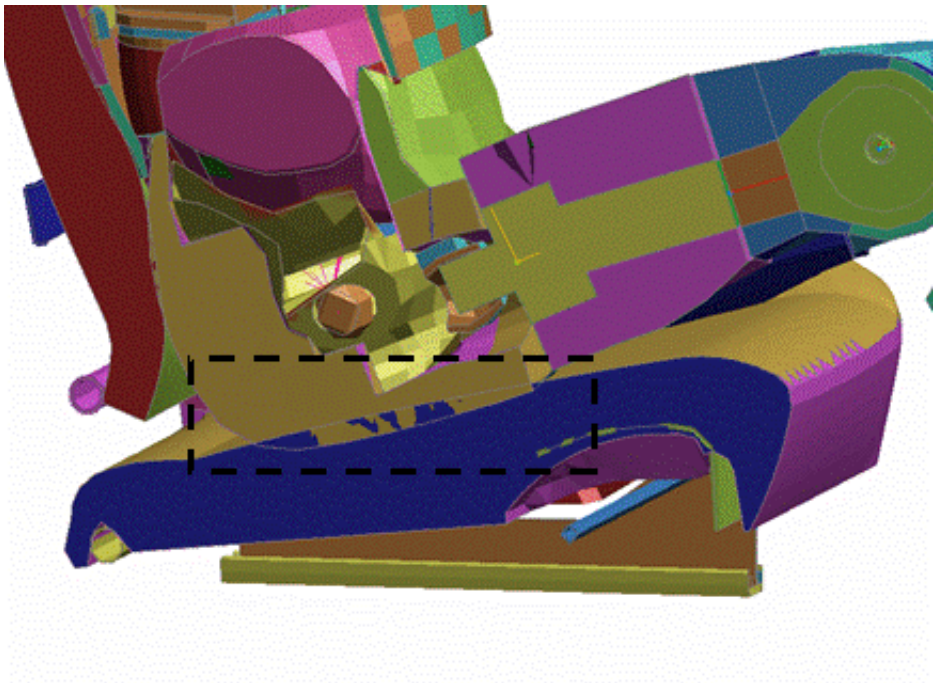
Press **Apply** to start the process. First, the dummy will move away from the seat according to the displacement you prescribed until the contact between seat and dummy is fully de-penetrated.

The dummy is then moved back to its original H-point position while compressing the seat foam.

Once finished you can save your model and/or export coordinates if required for use in other analyses.

LS-DYNA seat squash method

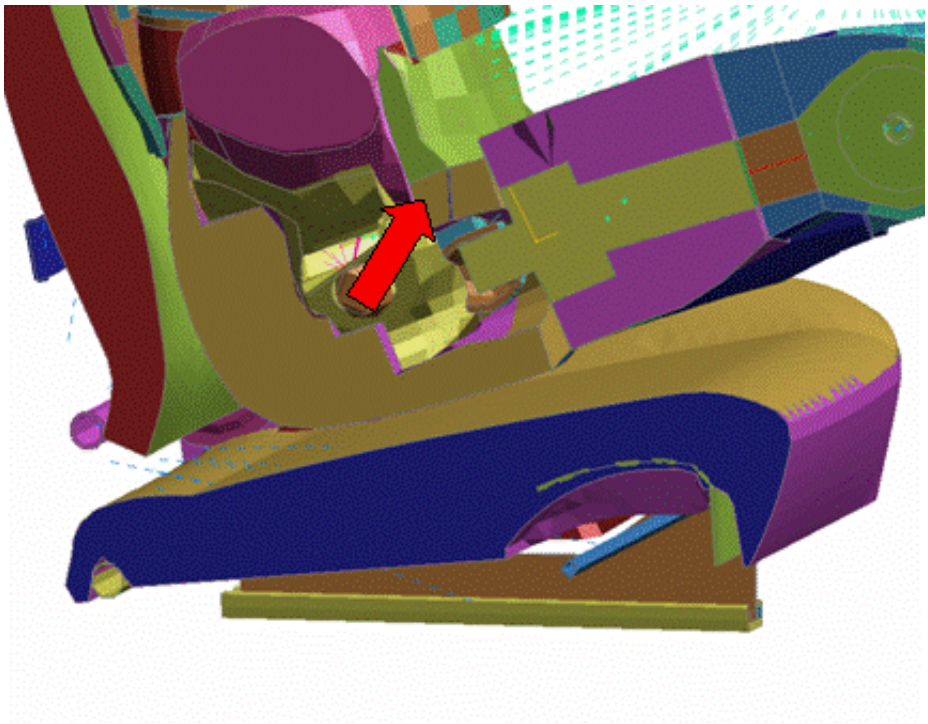
Process description



Initial state: model with penetration

Cut section through dummy and seat showing initial penetration between dummy components and seat foam.

The dummy is at the correct H-point but penetrates the seat.



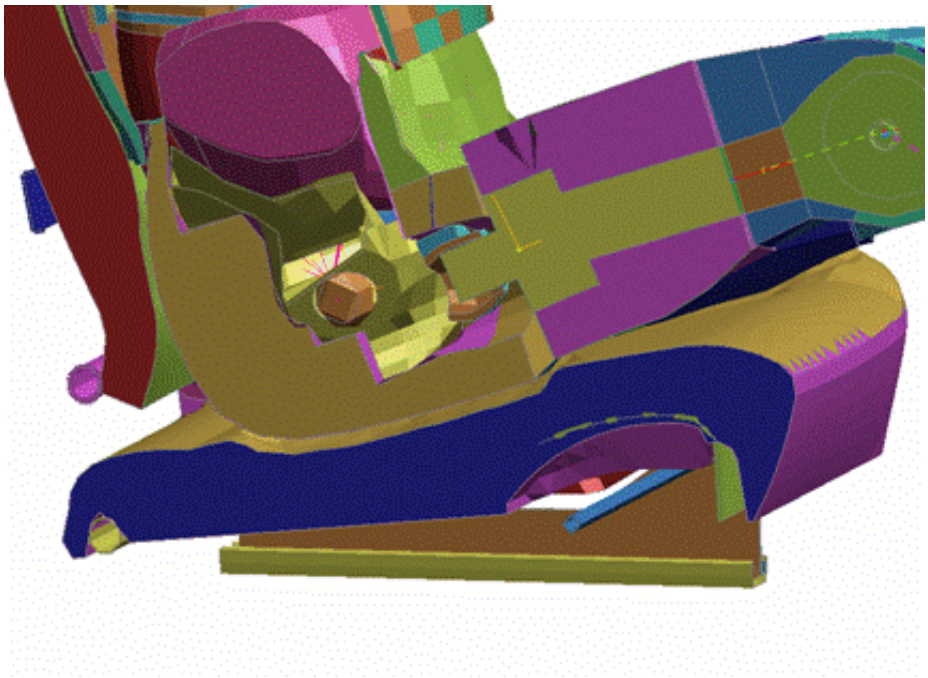
The dummy is moved from its initial H-point position, following the direction prescribed by the user until the contact with the seat top shell part has no penetration.

The dummy and the seat components that you defined as non-deformable are then rigidified.

Any contacts that you identify as redundant are then deleted.

The complete input deck for the LS-DYNA seat-squash analysis is set-up.

The user tidies, checks and modifies the LS-DYNA input file as required and then runs the analysis.



Final state: Seat foam compressed

When the LS-DYNA analysis terminates it will write a dynain file that contains the coordinates and initial stresses for the seat foam (and possibly other parts too)

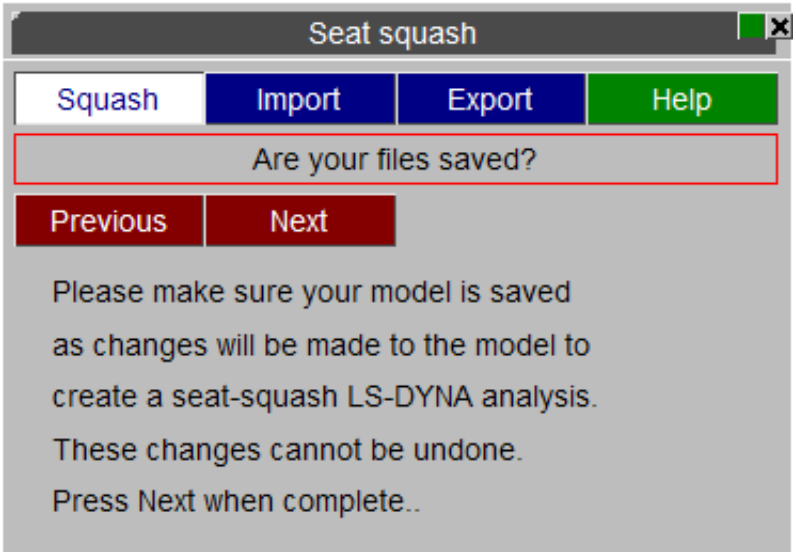
The user imports the data written out in the dynain file. The model now contains the deformed geometry and initial stress of all the **DEFORMABLE** parts

The seat foam components are now in their compressed state and the contact between seat and dummy is de-penetrated.

Step 1

Before you go any further you should save your model. PRIMER Will prompt you to save your model as the seat squash changes are irreversible (although you can import coordinates from a dynain file to effectively do an 'undo' See section 6.41.1 for more details).

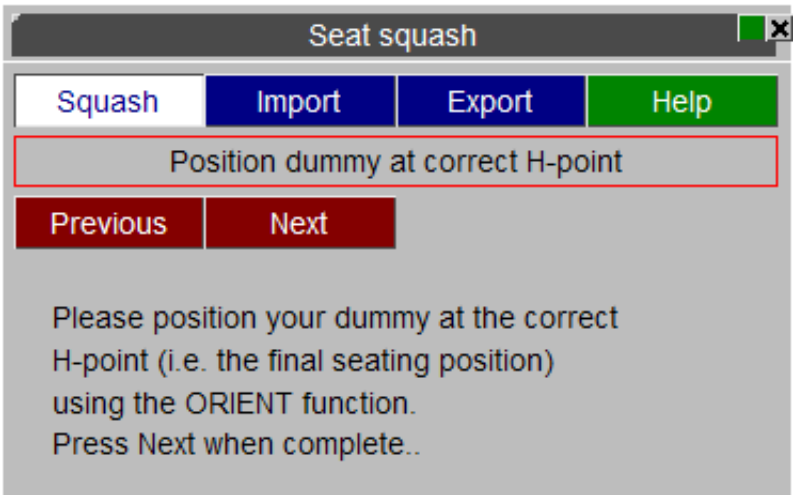
Once you have saved your model press **Next**



Step 2

You will be asked to move your dummy to the correct H-point location. Position the dummy by either using the orient menu or dummy positioning menu.

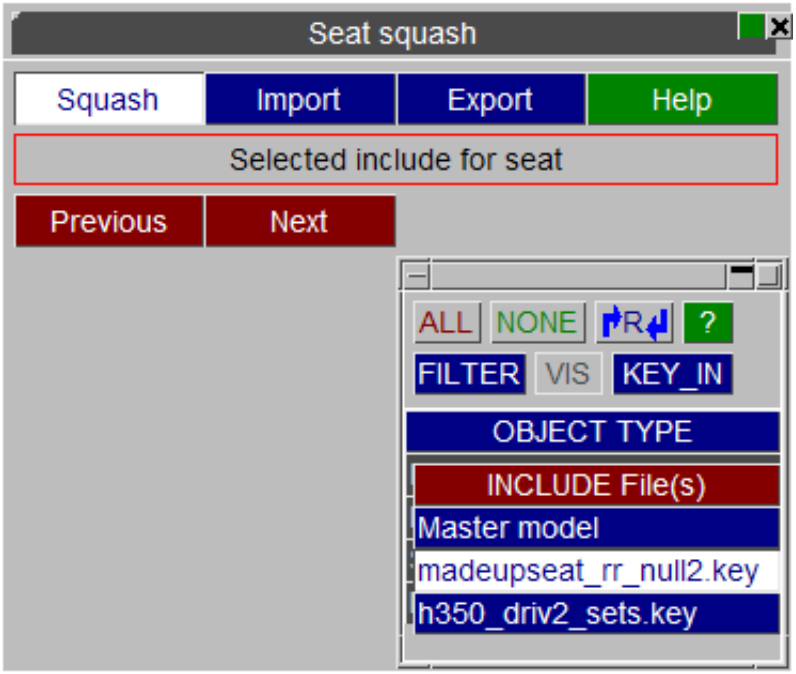
Once this is done press **Next**



Step 3

Select all of the parts that make up the seat using the standard object menu.

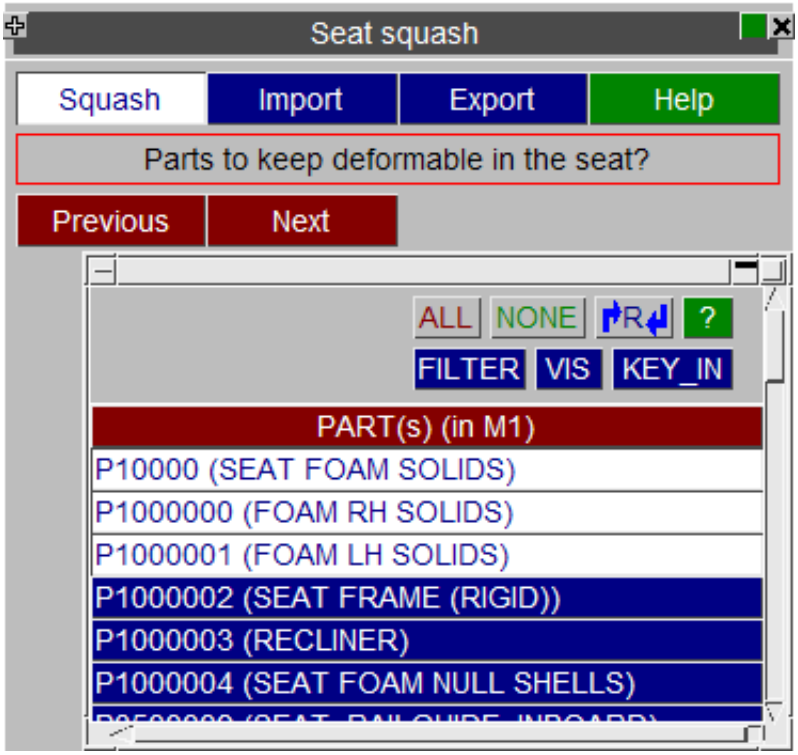
To continue press **Next**



Step 4

Select the all the **DEFORMABLE** parts of the seat structure using the standard object menu. PRIMER Will automatically select any parts that use a foam material. Typically, you should select the foam components and null shells on the surfaces. You can add and/or change this selection as required. PRIMER Will rigidify any parts that are not deformable to make the LS-DYNA analysis quicker.

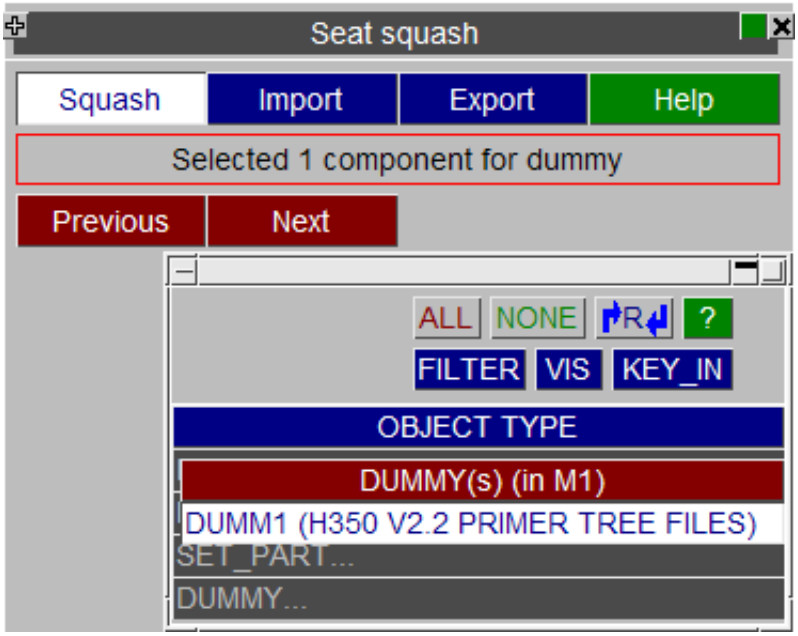
To continue press **Next**



Step 5

Select the dummy components. You can use the **DUMMY...** option in the standard object window.

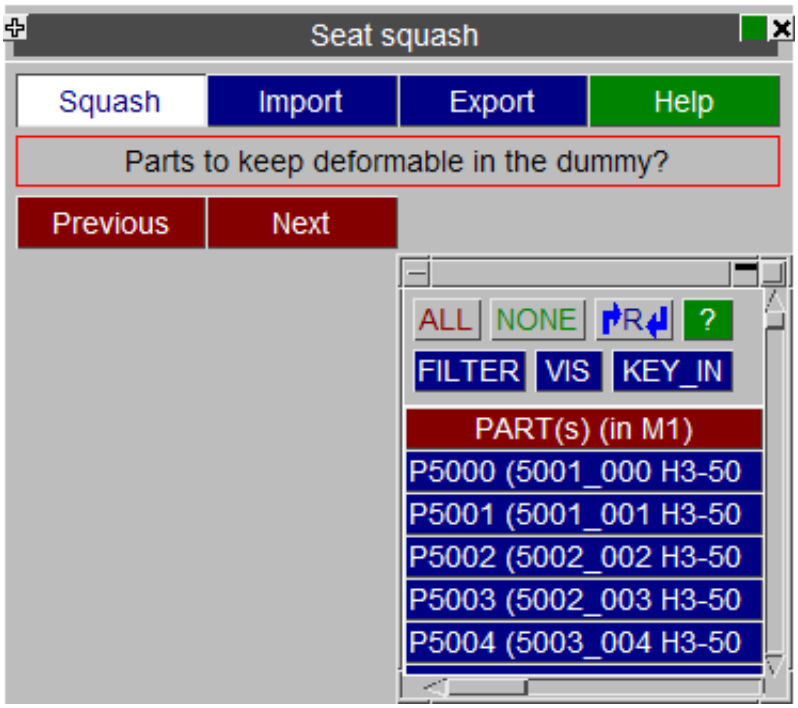
To continue press **Next**



Step 6

Select any parts of the dummy that you want to keep deformable using the standard object menu. Typically, you would not select any parts so the entire dummy is rigidified. However, you may want to keep some parts deformable so you can change this selection as required. PRIMER Will rigidify any parts that are not deformable to make the LS-DYNA analysis quicker.

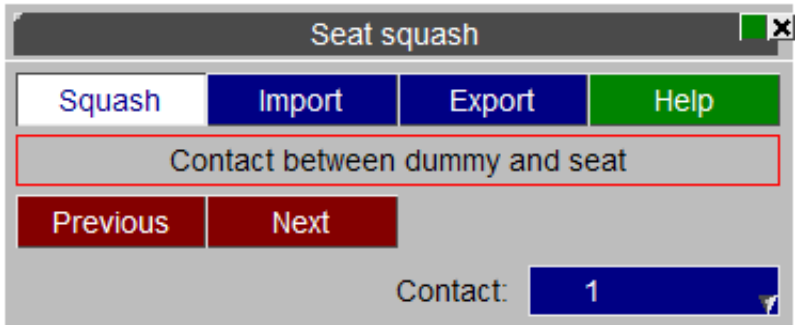
To continue press **Next**



Step 7

To select/create the contact between the seat foam and the dummy, you can use the standard popup functions (right click). You can then Pick, Select an existing contact or Create a new one.

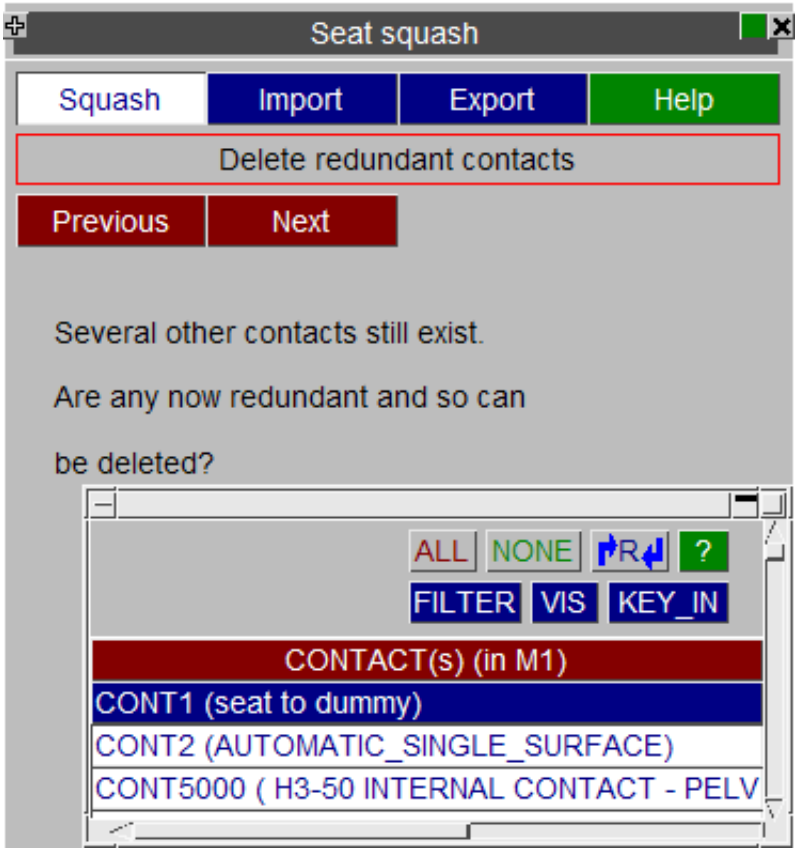
Once the contact has been selected/created press **Next**



Step 8

In the simplest seatsquash model the only contact that you will need is between the dummy and the seat. Any other contacts that are present in the model will just slow the analysis down. PRIMER Will prompt you to delete any contacts which it thinks are unnecessary. By default, all contacts except the contact defined in step 7 are chosen. Change this as required.

When the relevant contacts are selected press **Next**



Step 9

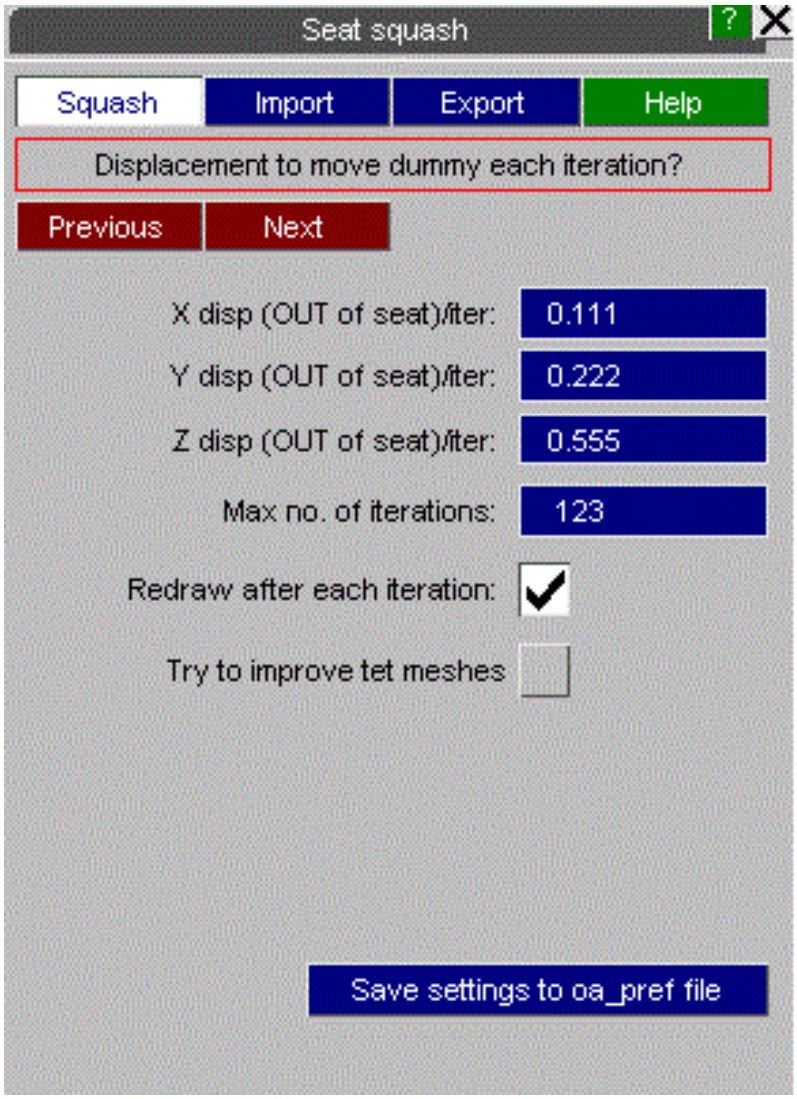
Define the dummy displacement increment at each de-penetration iteration. X, Y, and Z increments are given for each iteration to move the dummy out of the seat. As this is only going to be used to move the dummy out of the seat the iteration can be as large as you like.

You can set the maximum number of iterations that PRIMER will try to do when moving the dummy out of the seat.

If you want to see the progress of the seatsquash then select the **Redraw after each iteration** checkbox. This will make the process much slower so if you want, you can turn it off.

Some tet meshes can be very badly deformed making it very easy to make badly deformed elements. PRIMER Can try to 'smooth' tet meshes to make them better. This may help if you are having problems squashing a dummy into a tet meshed seat. Finally, it is possible to save settings by using the button **Save settings to oa_pref file**.

To continue press **Next**



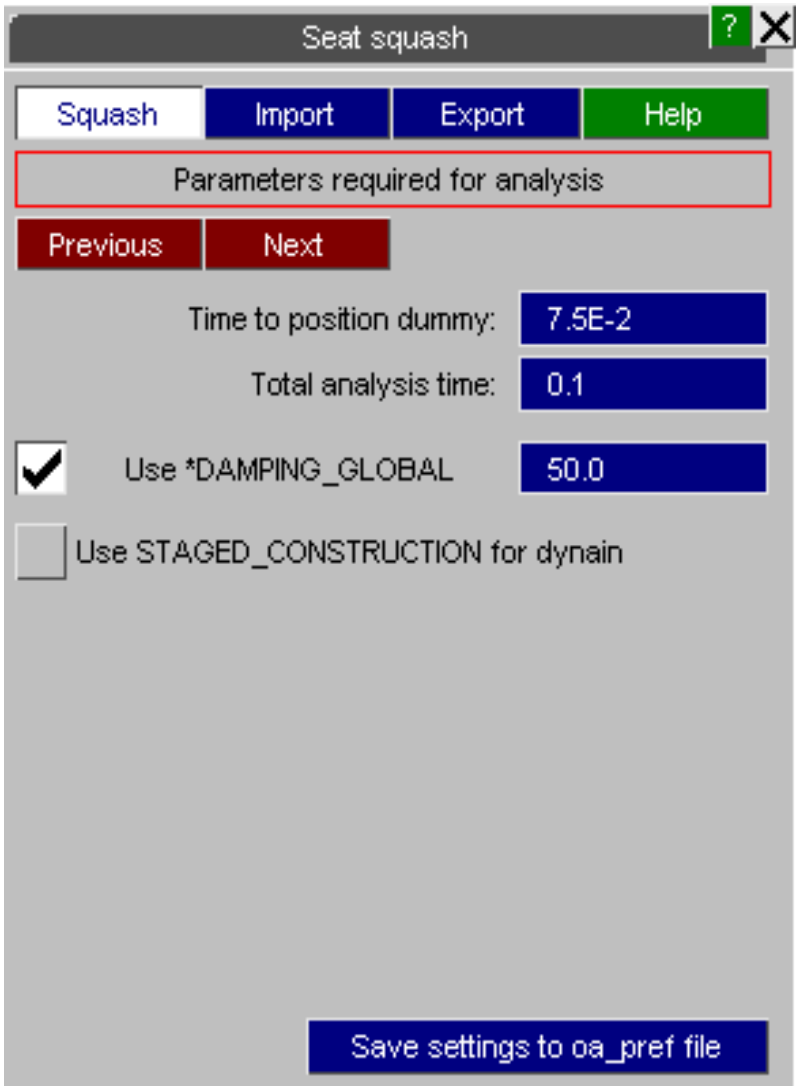
Step 10

Setup the parameters of the LS-DYNA seat-squash analysis. In most cases, the default values setup in Primer should be appropriate.

When the analysis is run it will output a dynain file which by default will be output using the *INTERFACE_SPRINGBACK card. In LS-DYNA R9 however, this does not output the coordinates of parts which use the *MAT_FABRIC material. An alternative method for outputting a dynain file is to use STAGED_CONSTRUCTION cards and this can be selected here if required.

Finally, it is possible to save settings by using the button **Save settings to oa_pref file**.

To continue press **Next**



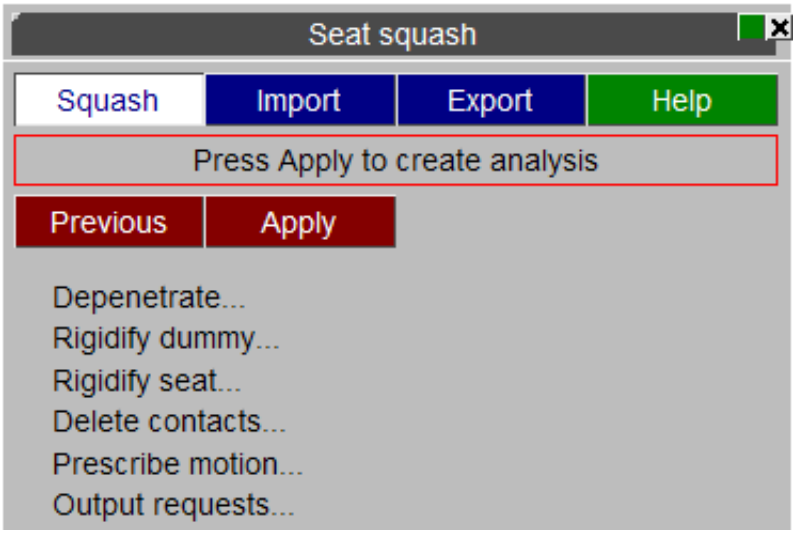
Step 11

Press **Apply** to start the process. First, the dummy will move away from the seat according to the displacement you prescribed until the contact between seat and dummy is fully de-penetrated.

The dummy and the seat components that you defined as non-deformable are then rigidified.

The contacts that you identified as redundant are then deleted.

The complete input deck for the LS-DYNA seat-squash analysis is setup

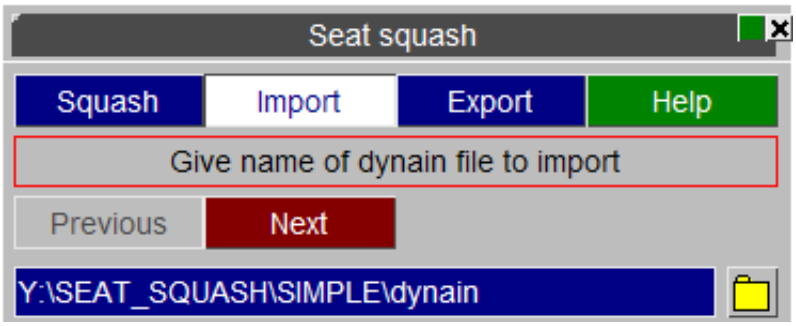


Step 12

Now review the LS-DYNA input deck that PRIMER has created, making any amendments you wish. Run the analysis using LS-DYNA and then import the required coordinates and initial stresses to your main model.

6.41.3 **Import** option

Step 1

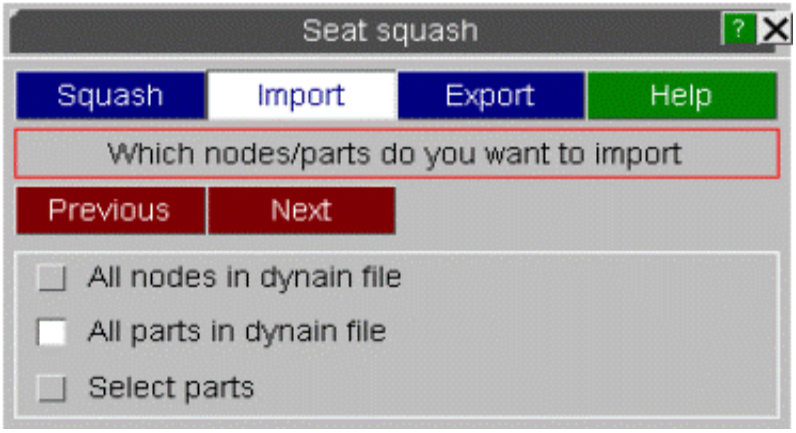


Once the analysis is finished, read your original model, in which dummy and seat foam components have penetration.

Go back to the Seat squash panel, click on the **Import** button and read in the dynain file that LS-DYNA has written out. This file contains the final geometry and initial stress data of all the **DEFORMABLE** parts that you define at **STEP 4.**

To continue press **Next**

Step 2



PRIMER will then read the file into a temporary model.

Either read the data for **All nodes in dynain file, All parts in dynain file** or **Select parts** to choose the parts in the dynain file to read data from

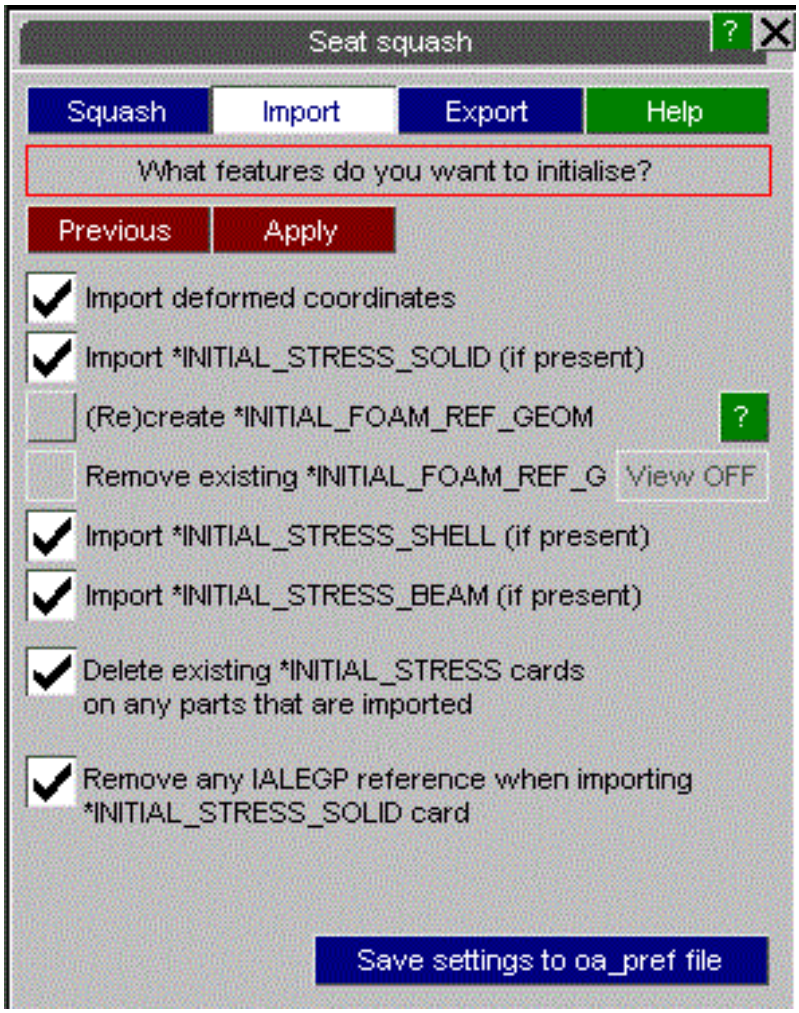
To continue press **Next**

Step 3

Select the features that you want to initialise. (Initial stress). Finally, it is possible to save settings by using the button **Save settings to oa_pref file**.

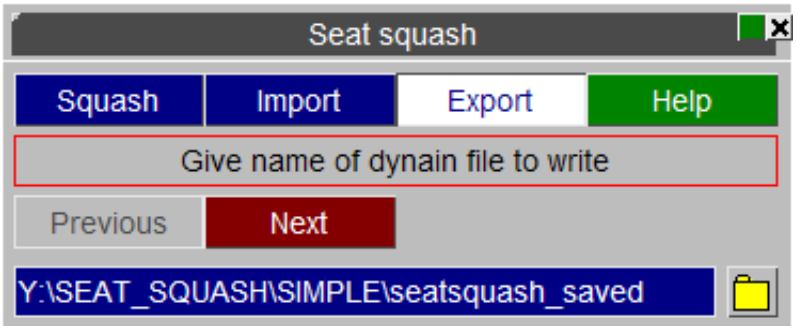
To continue press **Apply** and **Finish**.

The seat foam components are now in their compressed state and the contact between seat and dummy is de-penetrated.



6.41.4 **Export** option

Step 1



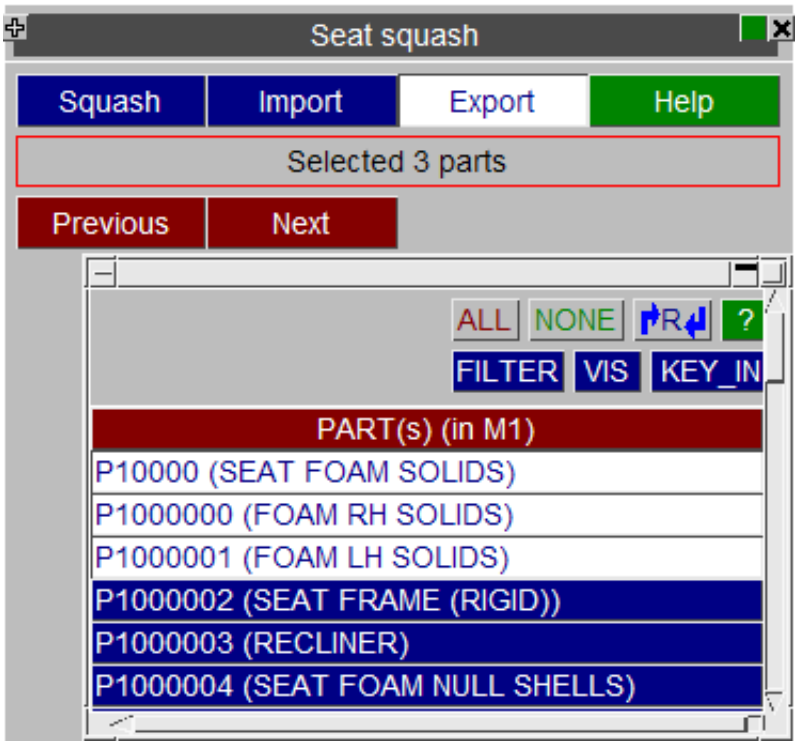
The **Export** option allows you to write out nodal coordinates and initial stress data for parts in the seat.

There are three steps to follow to write out the data:

First select the name of the file that you want to write by either typing the name in the textbox or using the file selector.

Once done, press **Next**.

Step 2



Second, select the parts that you want to write to the file.

PRIMER Will show an object menu allowing you to select the parts you want. All of the normal options for object menus will be available. For example, if the seat is in an include file it may be helpful to first filter by include file so that only the parts for the seat are shown. Then if you only want to write out the data for the solid parts you could then additionally filter by element type SOLID or by a material type to further limit what is shown in the object menu.

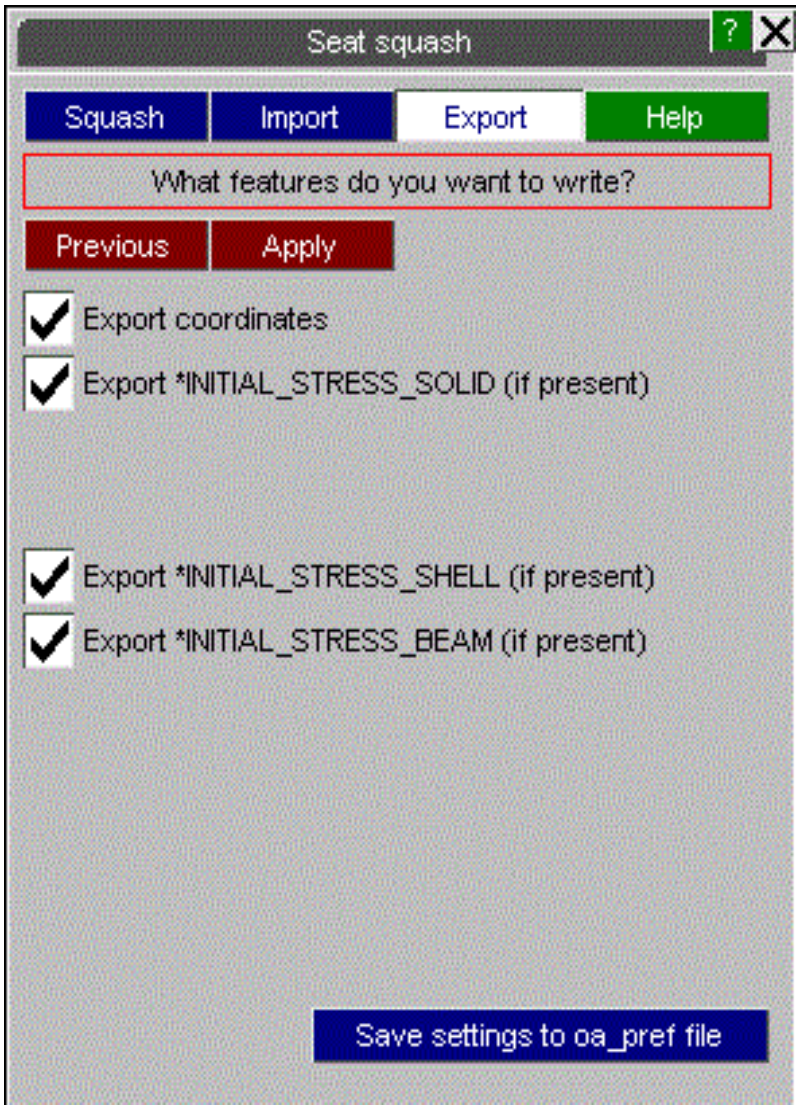
Once all of the required parts are selected press **Next**.

Step 3

Now select what you want to write to the file by using the checkboxes.
A basic file could just contain the nodal coordinates. This could also be used as a way of undoing a seatsquash. If you save the coordinates of the seat foam nodes before doing a seatsquash you can import them again if required to 'undo' the seatsquash.

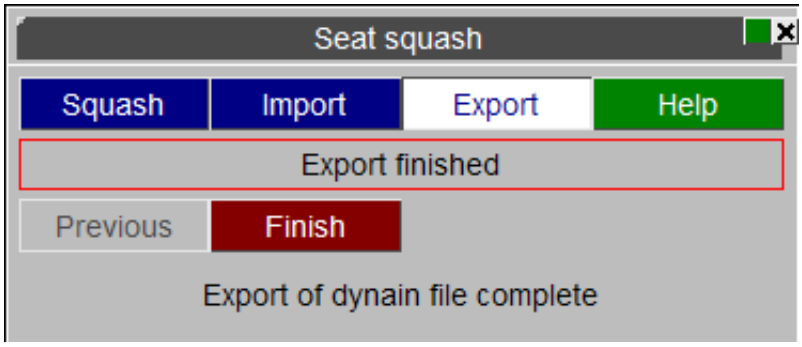
When you have the categories you want selected press **Apply** to write the file. Finally, it is possible to save settings by using the button **Save settings to oa_pref file**.

To continue press **Apply** and **Finish**.



Step 4

When the file has been written press **Finish**.



6.42 Sled Test Automation Script (ECE-R17)

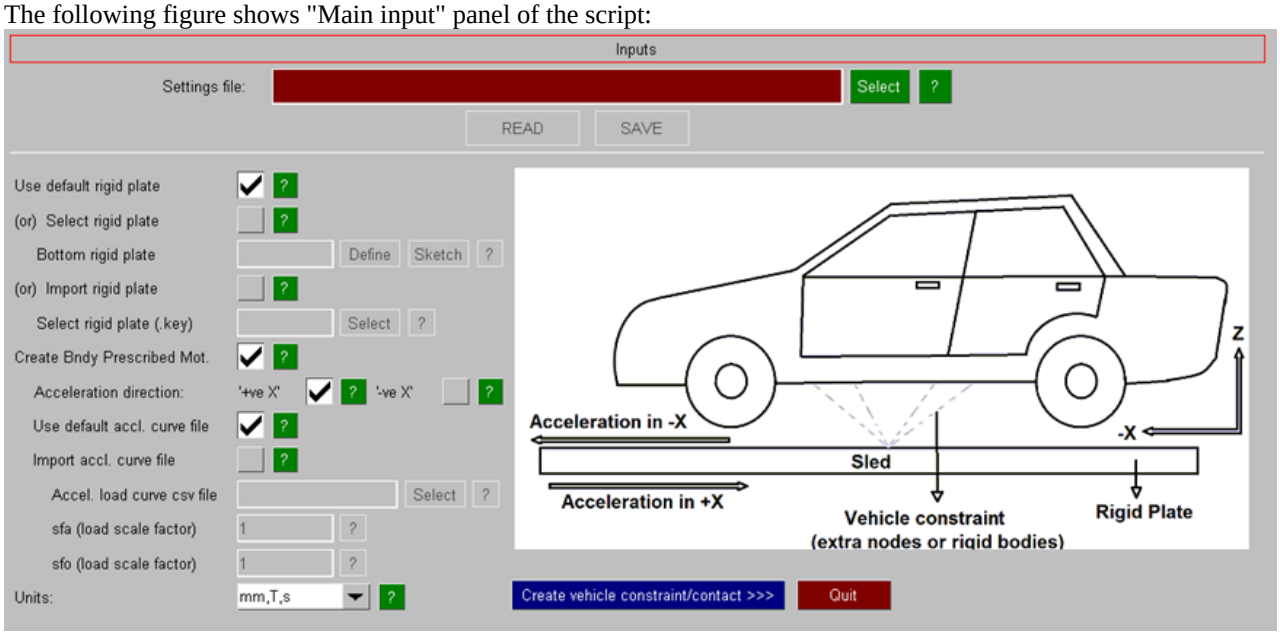
Introduction

This regulation deals with test of strength of the seat anchorage and the adjustment, locking and displacement systems of the vehicle.

The seat anchorage and the adjustment, locking and displacement systems must be able to withstand a longitudinal horizontal deceleration of not less than 20g applied for 30 milliseconds in the forward/rear direction to the whole shell of the vehicle.

The script creates a rigid plate on which vehicle will be mounted and set up the analysis as per the regulation ECE-R17.

The following figure shows "Main input" panel of the script:



The following options are available on the main input panel:

Settings file	A settings file may be read or written which may include mandatory information and other data required to start calculation. The reading/writing of a settings file is optional.
Use default rigid plate	Select this option to create a default rigid plate which will be positioned at bottom of the vehicle. A longitudinal horizontal deceleration shall be applied on bottom rigid plate in the forward/rear directions.
Select rigid plate	Alternatively, use this option to select an existing bottom rigid plate on which vehicle is mounted.
Bottom rigid plate	An existing bottom rigid part on which vehicle is mounted.
Import rigid plate	Or, you can use this option to import a rigid plate include file (.key) into the model.
Select rigid plate (.key)	The rigid plate include file(.key) will be imported into the model and plate will be positioned at bottom of the vehicle.
Create Bndy Prescribed Mot.	Select to create a *BOUNDARY_PRESCRIBED_MOTION card for bottom rigid plate.
Acceleration direction	Load direction.
Use default acc. curve file	Select if you wish to use default acceleration curve which will be used in *BOUNDARY_PRESCRIBED_MOTION card.
Impot accl. curve csv file	Select this option to import a acceleration curve file in CSV format which will be used in *BOUNDARY_PRESCRIBED_MOTION card.
sfa	Acceleration load curve scale factor for abscissa value. By default, it is set to 1.
sfo	Acceleration load curve scale factor for ordinate value. By default, it is set to 1
Units	Specify unit system of the selected model.

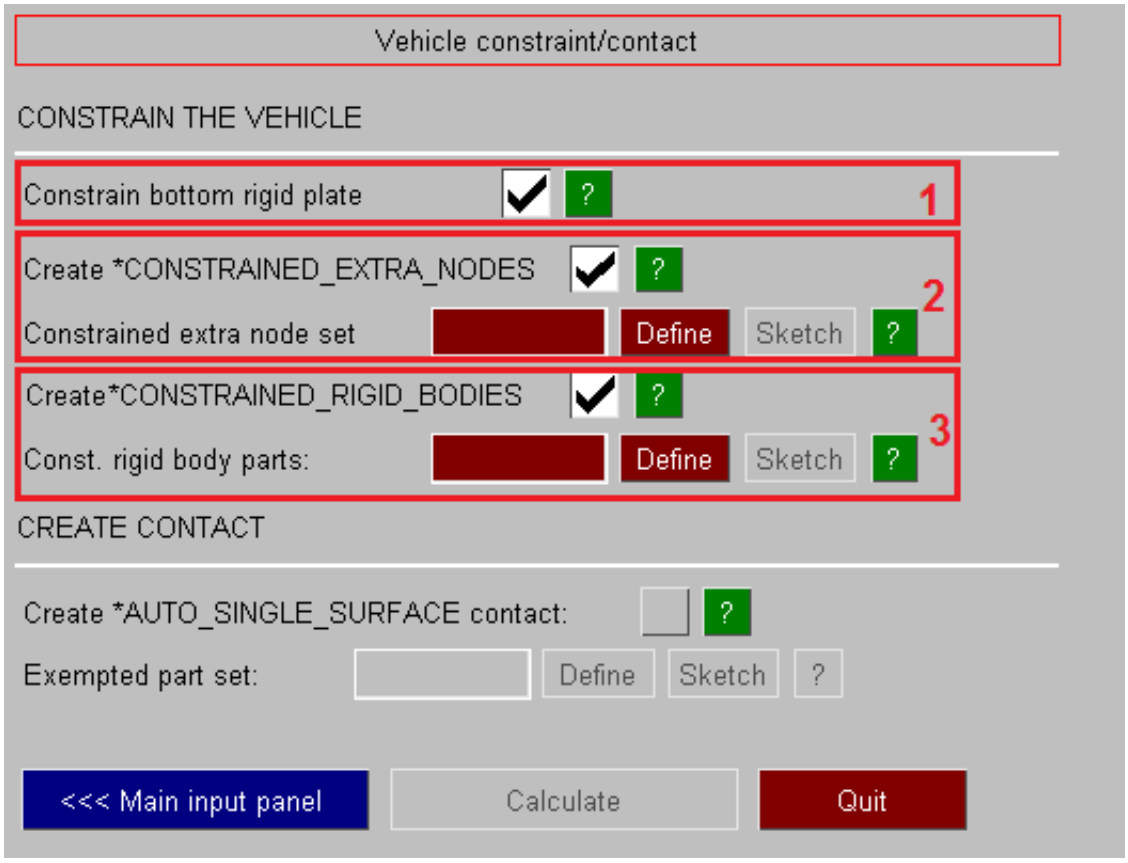
Constrain the vehicle

Click **Create vehicle constraint/contact** to create vehicle constraint and contact. All the options available on "Vehicle constraint/contact" input panel are optional.

Select "Constrain bottom rigid plate" option as shown below in red highlighted box-1 to constrain the bottom rigid plate in all DOFs (including rotations) except X-translations.

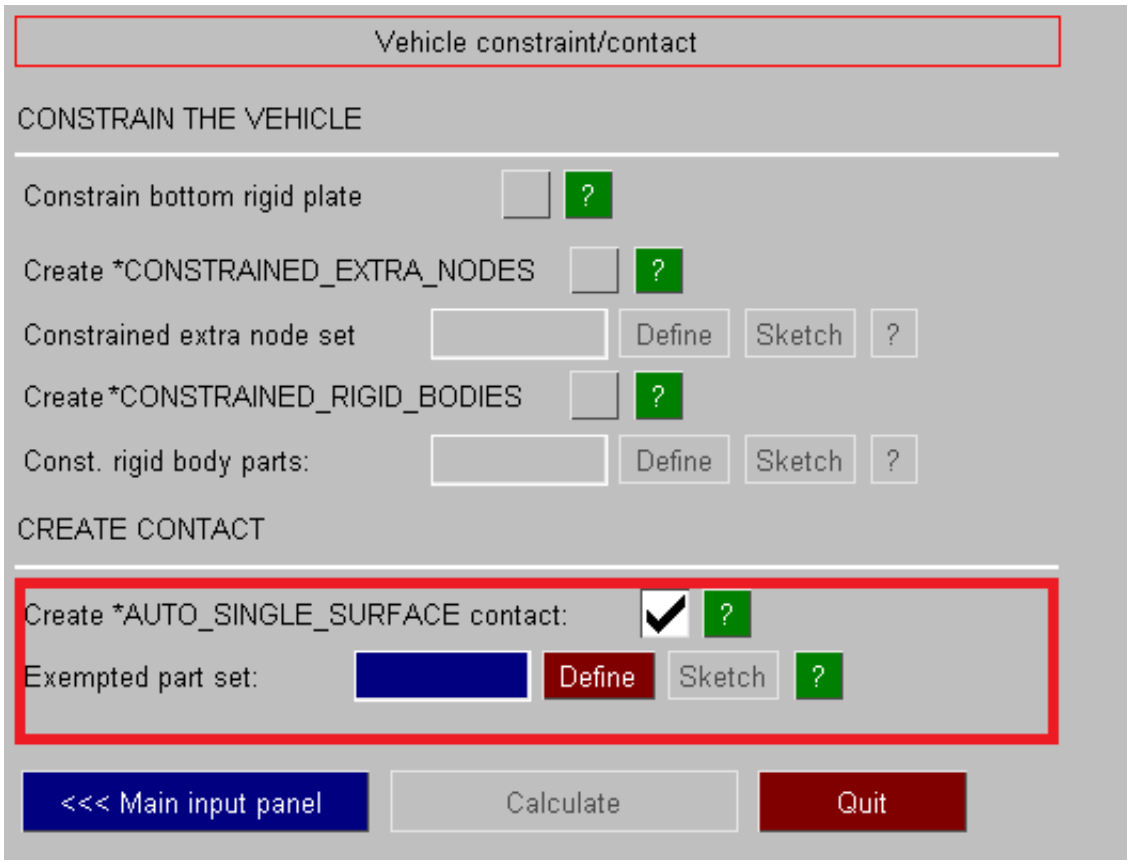
Select "Create *CONSTRAINED_EXTRA_NODE" option as shown below in red highlighted box-2 to constrain the vehicle with rigid plate using *CONSTRAINED_EXTRA_NODES. Use "Constrained extra node set" to provide node set for *CONSTRAINED_EXTRA_NODES.

Alternatively, "Create *CONSTRAINED_RIGID_BODIES" option as shown below in red highlighted box-3 can be used to constrain the vehicle on rigid plate. If this option is selected, use "Const. rigid body parts" to provide parts for *CONSTRAINED_RIGID_BODIES.



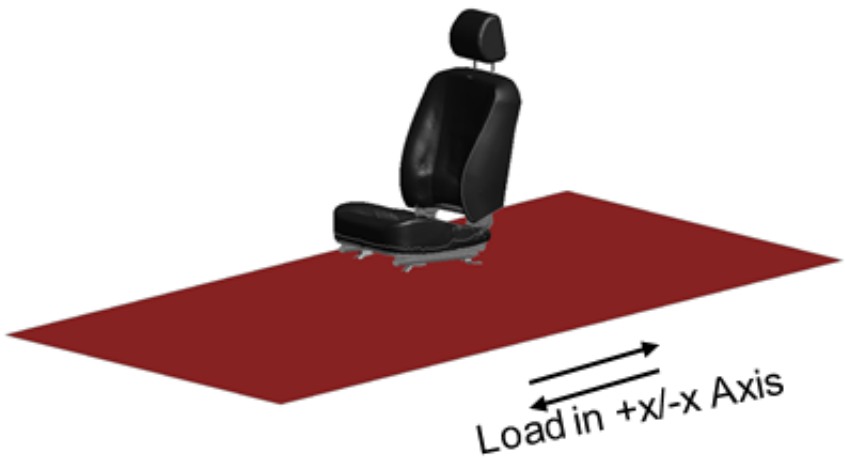
Create Contact

Select "create *AUTO_SINGLE_SURFACE contact"option as shown below to create a *CONTACT_AUTOMATIC_SINGLE_SURFACE with exempted part set in slave side. Impactor parts will get added automatically to the exempted part set. Use "Exempted part set" option to add any other part sets in the exempted set.



Calculate

Once all the input conditions have been defined, **Calculate** will become active. Press **Calculate** to start the calculation. After the calculation, vehicle will be mounted on the rigid plate as shown below:



6.43 **Text Edit**. External editing of any keyword

The generalised **Text Edit** tool allows any keyword, with two exceptions, to be edited in an external editor and then re-imported to the model in PRIMER. This process may also be used to perform a "mini keyword read" operation to read in a fragment of an input deck.

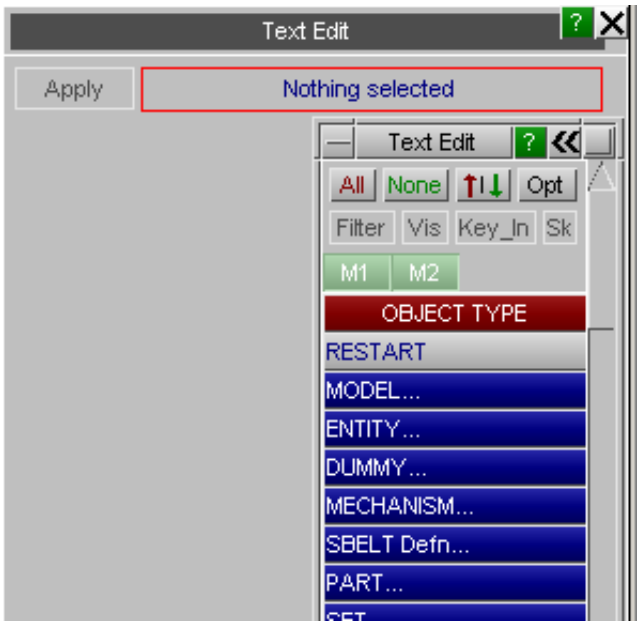
The editing process is started from the main **Tools** panel by the **Text Edit** button.

6.43.1 Selecting the items to edit

When first invoked the **Text Edit** panel will show a top level menu containing all keyword in all models.

You must first select a keyword type, and then select a single item or a range of items. These may be in a single model or spread over multiple models.

Once something has been selected the **Apply** button will become live, and clicking on this will launch the editing session(s).



6.43.2 How the editing process works

For each model that contains selected items PRIMER will write a "mini keyword file" and then launch a system editor session which opens this file.

The user can then inspect the keywords (ie read only) or update them, adding new keywords if desired. If the file is saved in its edited form then PRIMER detects this and reads it back into the model, updating any changed keywords and adding new ones.

This process is exactly the same as **Text Edit** when launched from an editing panel or the generic keyword editor, as described in section 5.1.11. To avoid repetition you are referred to that section which covers:

- How to control what comments appear in the text file. By default these are verbose, but can be controlled by preferences.
- How to control which system editor is used, again controllable by preferences.
- How the fact that editor sessions are asynchronous is handled, in particular what happens if the model is deleted from PRIMER.

6.43.3 Limitations

Unsupported keywords

The following keywords are not supported in this context:

`*INCLUDE` and **`*INCLUDE_TRANSFORM`**
`*CASE`

The reason is that these keywords are complex to process, with many consequence if they change, and the simple keyword read used in this process will not handle all the consequences of changing them correctly. These keywords will not be available in the list, even if they exist in the model.

Only existing keywords may be edited.

PRIMER does not permit you to select a keyword for **Text Edit** that is not already present in the model. The reason for this is that many keywords in LS-DYNA have a variable format that is dependent upon their content, so starting from nothing often requires some degree of knowledge of the possible card format, however the **Text Edit** capability is totally "dumb" in this respect.

Changing item labels has limitations..

Text Edit does not "know" or "remember" what it wrote out to a file to be edited, so if you change the label of an item then the effect when it is read back in will be to make a new item (or possibly overwrite some other existing one) rather than to relabel the original definition. This is best illustrated by example.

Model contains	User selects for **Text Edit**	Labels are changed to	Result when read back into PRIMER
PART 1	PART 1	*<no change>*	PART 1 *(no change)*
PART 2	PART 2	PART **10**	PART 2 *(no change)*
PART 3	PART 3	PART **5**	PART 3 *(no change)*
PART 4	*<not selected>*	*<no change>*	PART 4 *(no change)*
PART 5	*<not selected>*	*<no change>*	PART 5 ***contains what was PART 3***
			PART 10 ***contains what was PART 2***

Note that:

- The original definitions of PARTs 2 and 3 inside PRIMER are not changed, even though they have been changed in the file.

  This is because **Text Edit** does not "remember" that they were written to file, so when they are no longer there to be read in the original definitions are not updated.

- PART 5 insider PRIMER is overwritten by what used to be PART 3

- A new PART 10 is created inside PRIMER, from what used to be PART 2

Note that this behaviour is different to running **Text Edit** from an item editing panel. In that context it "knows" what is being edited, but here it does not.

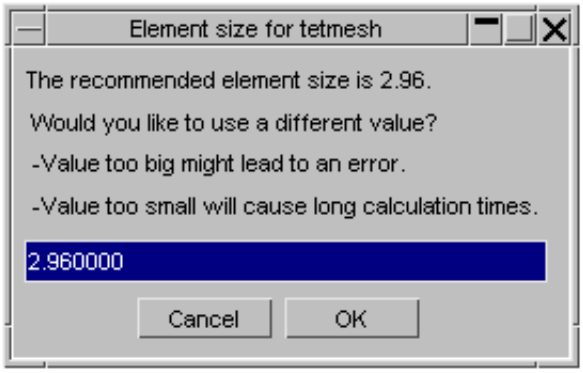
So be careful when changing item labels!!

6.44 VOLUME CALCULATOR

The Mesh Volume Analysis tool is designed to make analysing closed meshes, specially fuel tanks, easier for the user. The key features include liquid level visualisation, wet surface area and step-by-step volume calculation. In addition, the results can be written to an Excel file for further analysis.

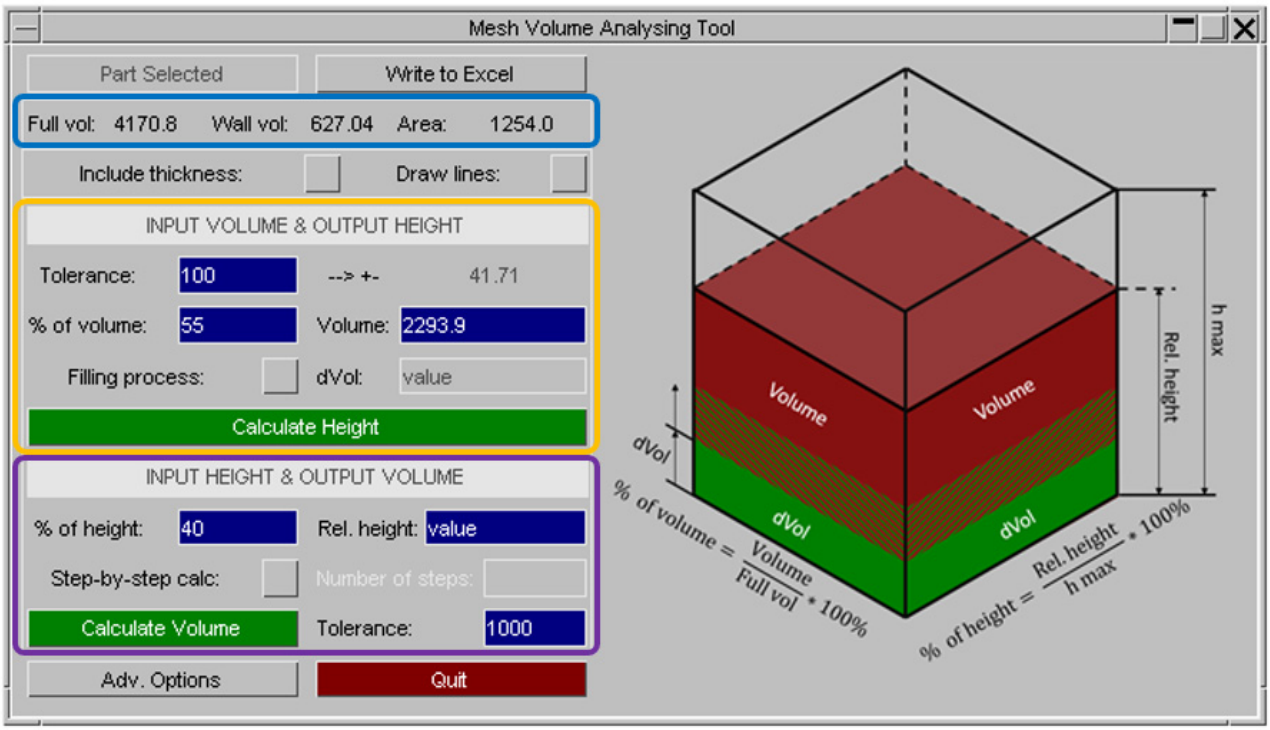
6.44 .1 Using the Mesh Volume Analysis tool

Firstly, the user has to select a part or parts. It can be either be a shell or a ready-made solid part. The part needs to have closed geometry. Mesh density has an impact on calculation times. If it is a fine mesh, then calculations will take longer.
If the user selects a shell part the part will be copied into a new model, then a tetmesh will be generated based on the element size user inputs, the suggested element size is based on the average minimum shell length*3.5. Calculation will be based on each of the solid tetrahedron in the solid mesh.



When a solid is selected, then a similar window will pop up but the user has to define shell thickness. Then the outer layer will be defined as a midsurface.

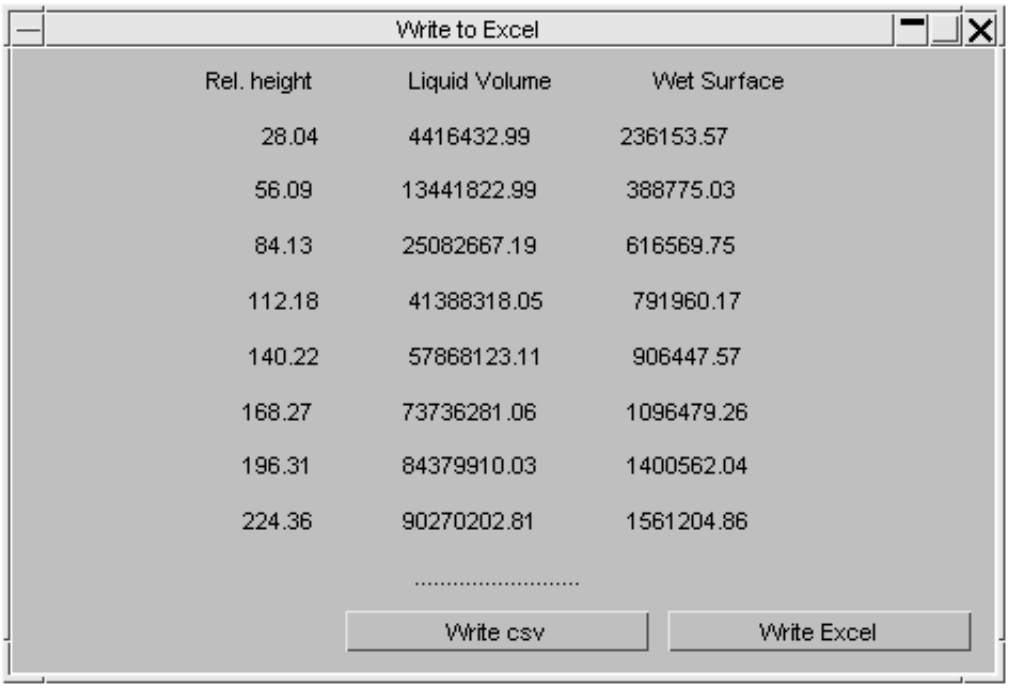
6.44 .2 Main Panel



IMPORTANT: If the user hovers over the labels in the panel, then some additional information will be displayed about their meaning.

The panel marked with blue shows full volume (maximum volume) of the fuel tank, wall volume (shell area * thickness) and total area of the shells.
The panel marked with orange is for volume based analysis – user inputs volume and height values with wetted surface area will be output.
The panel marked with purple is for height based analysis – user inputs height and corresponding volume with wetted surface area is output.

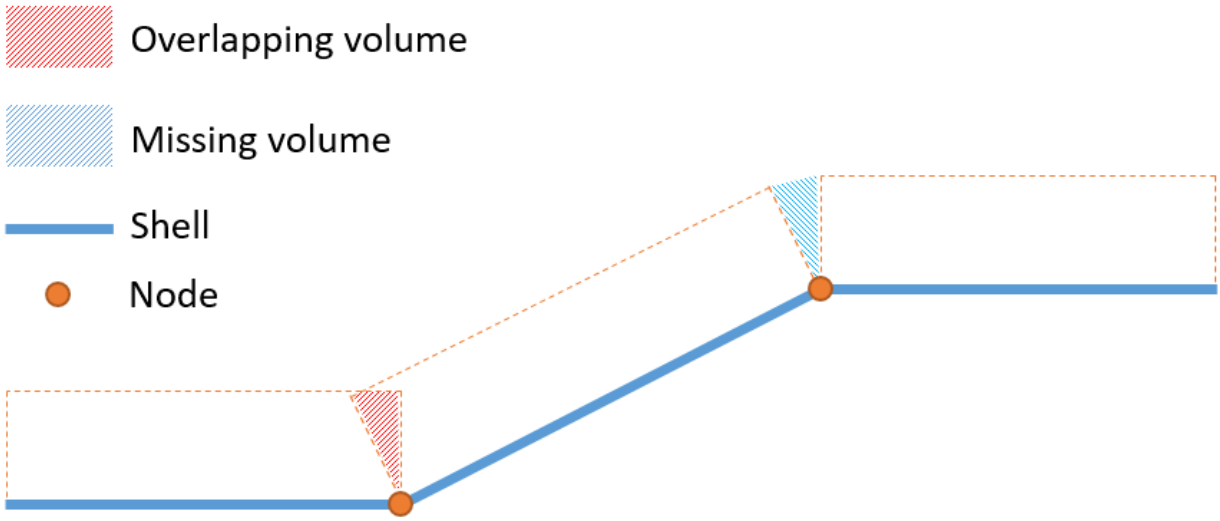
At the very top of the main menu, there is a button “Write to File”. The user has a chance to write results to an Excel or csv file. In order to do that, the user must run some type of calculation/analysis on the geometry.



In main window there are 2 checkboxes at the top of the panels:

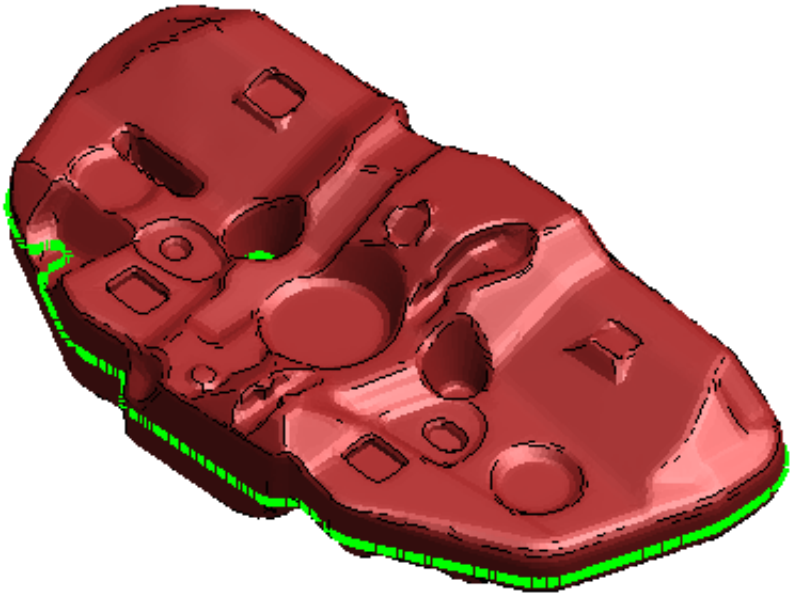
Include Thickness

Will include thickness of the shells in calculations. Furthermore, it understands whether the shell definition is outer, middle or inner surface. The thickness is based on the following formula: Shell area*Shell thickness. Note that there are some inaccuracies to this method as there might be some overlapping or inconsistencies of volumes:



Draw Lines

Will visualise resting liquid levels by drawing a line around the perimeter of the geometry and displaying information like volume and height.

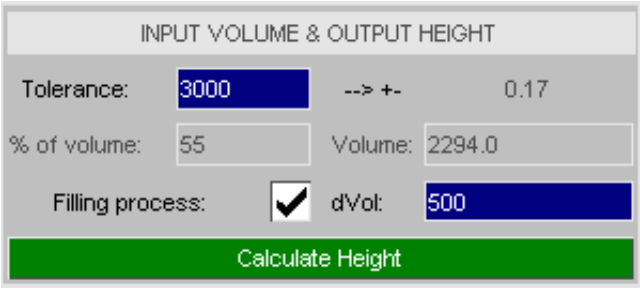


Volume:41388318.05 Height:112.18 (270.25)

6.44.3 How to use the “INPUT VOLUME & OUTPUT HEIGHT” panel

User has 2 options:
1. Enter single volume and get single height.
2. Simulate filling process by entering the amount of liquid added each step.

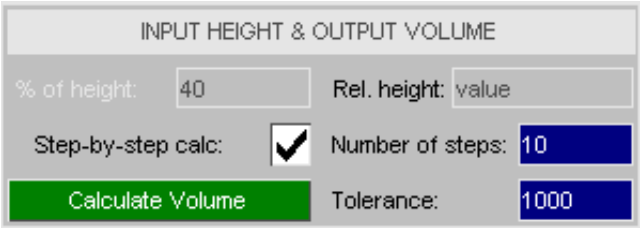
The calculation is an iterative process where volume is calculated at 50% of height of the geometry and then compared to the input value. This process is repeated until a volume within the tolerance is found.
The tolerance value is based on either Volume or dVol textbox. Tolerance determines the boundaries in which found volume is acceptable. For example, if dVol value is 500 and tolerance is 10, then output values can vary from +/-500/10 = +/-50. Therefore, any output between 450 and 550 is possible. Tighter tolerance will lead to longer calculation times and might mean increasing “Max iterations” under “Advanced Options” panel.



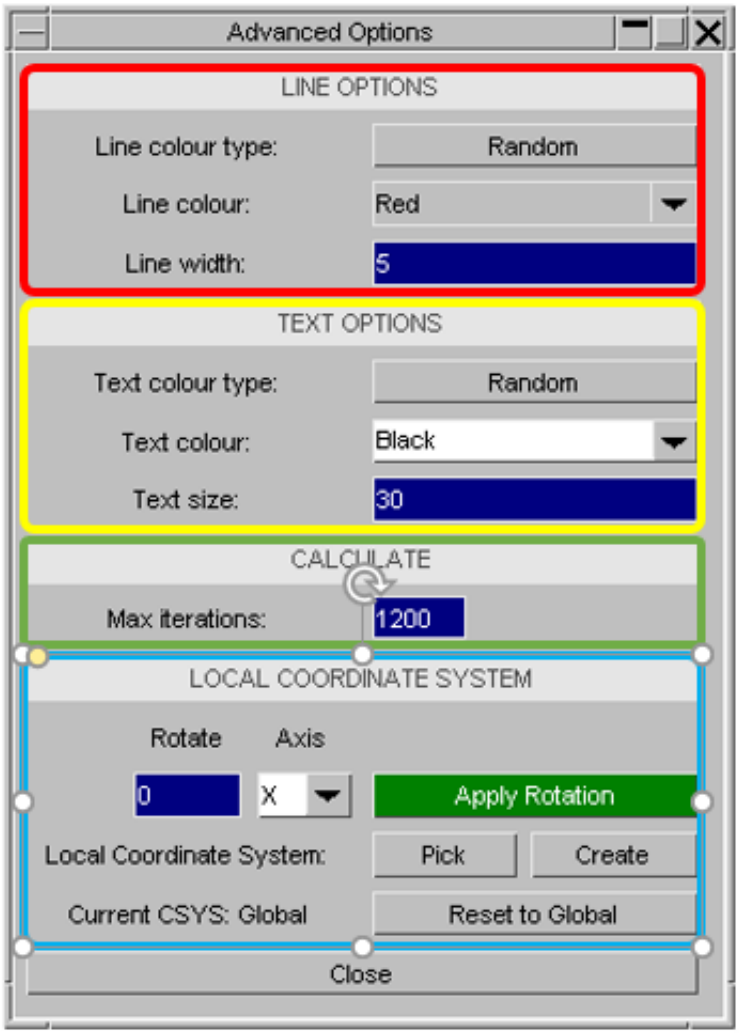
6.44.4 How to use the “INPUT HEIGHT & OUTPUT VOLUME” panel

User has 2 options:
1. Enter either relative height or a percentage of total height and output volume. (Single value)
2. Enter number of steps where the total height is divided into equal parts and volume is output after every (total length)/(number of steps) height units.

The tolerance value is used for boundary conditions where the entered height value is between tet’s max and min value. In most of the cases, this value should not be changed but if the script is outputting wrong answers or then changing tolerance might fix it. Furthermore, if lines are visualised and they are not around the perimeter of the geometry, then increasing the tolerance might help.



6.44.5 Advanced Options



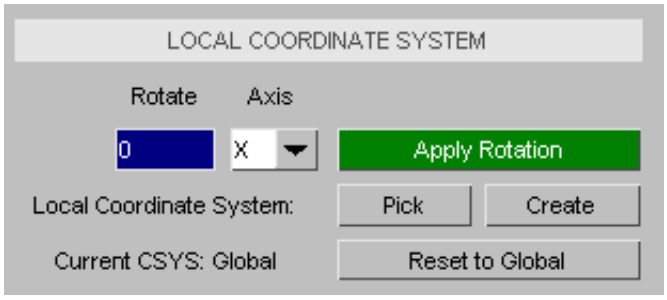
Line options panel (red) is for line visualisation when “draw lines” option is checked in the previous window.
Text option panel (yellow) is for label visualisation when “draw lines” checkbox is enabled in the previous window.

Line options and text options panels work in the same way. The user is able to choose colours and size of the text or width of the lines. Furthermore, the user can specify colours and whether the colours are random or constant. If “random” is chosen for both lines and text then the text and lines will be the same colour at each level.

Calculate panel (green) is for “Input volume & Output height” options. “Max iterations” determines the limit of the iterative process per dVol value.

Local Coordinate Systems

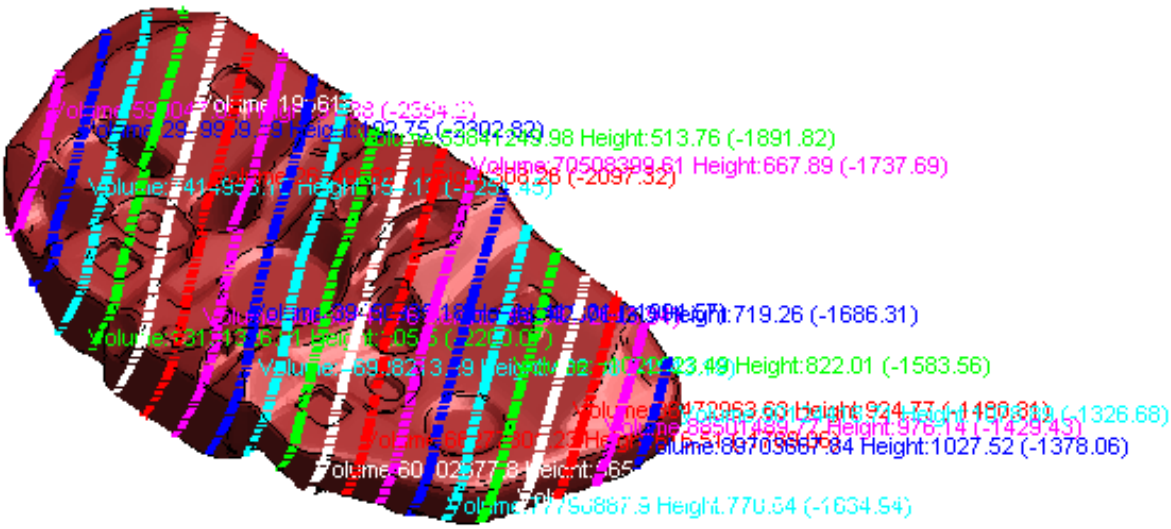
The local coordinate system section is for choosing or creating local coordinate systems or for rotating the global system which would only have an impact on the calculation (the actual global coordinate system will stay the same).



Note: Z axis will determine the direction of calculation and xy-plane will define the resting liquid surface.
User has two options:

1. “Rotate” the global system locally (has no effect on the model).
2. Choose or create their own defined local coordinate system.

First option will rotate the global coordinate system around the specified axis. The rotation is done locally. It will work only around one axis at a time. For example, if the user rotates X-axis 30 degrees and then decides to rotate the Y-axis, the X-axis rotation will go back to 0 degrees.
Second option of choosing or creating a local coordinate system means that the user can pick a coordinate system they have created. The coordinate system can be based on nodes, vectors or system. If a local coordinate system is selected, then the user can go back to global by pressing the “Reset to Global” button.

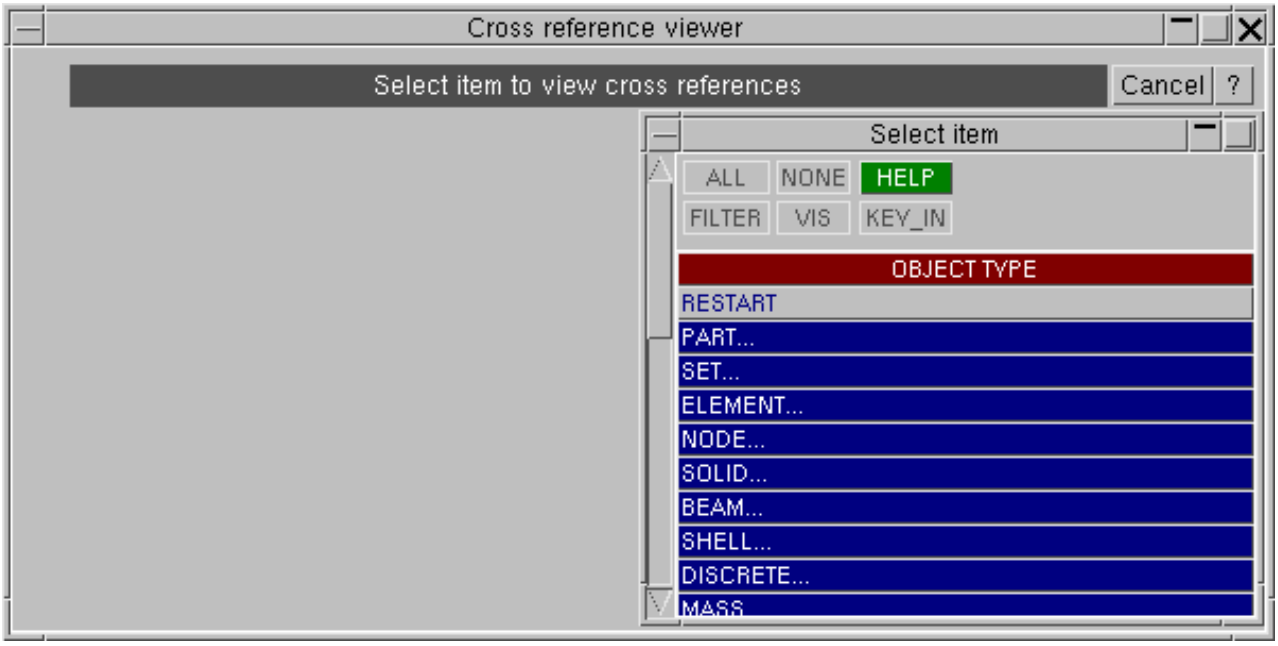


6.45 XREFS Cross references viewer

The cross reference viewer allows you to look at how a specific keyword is used. The viewer is started from the main **Tools** panel by the **Xrefs** button or in editing panels by the **VIEW_XREFS** button.

6.45 .1 Selecting an item in the viewer

If the viewer is started from the main **Tools** panel you need to select the item to display. An object menu allows you to choose an item using the normal methods (selecting from list in menu, picking, filtering etc.)

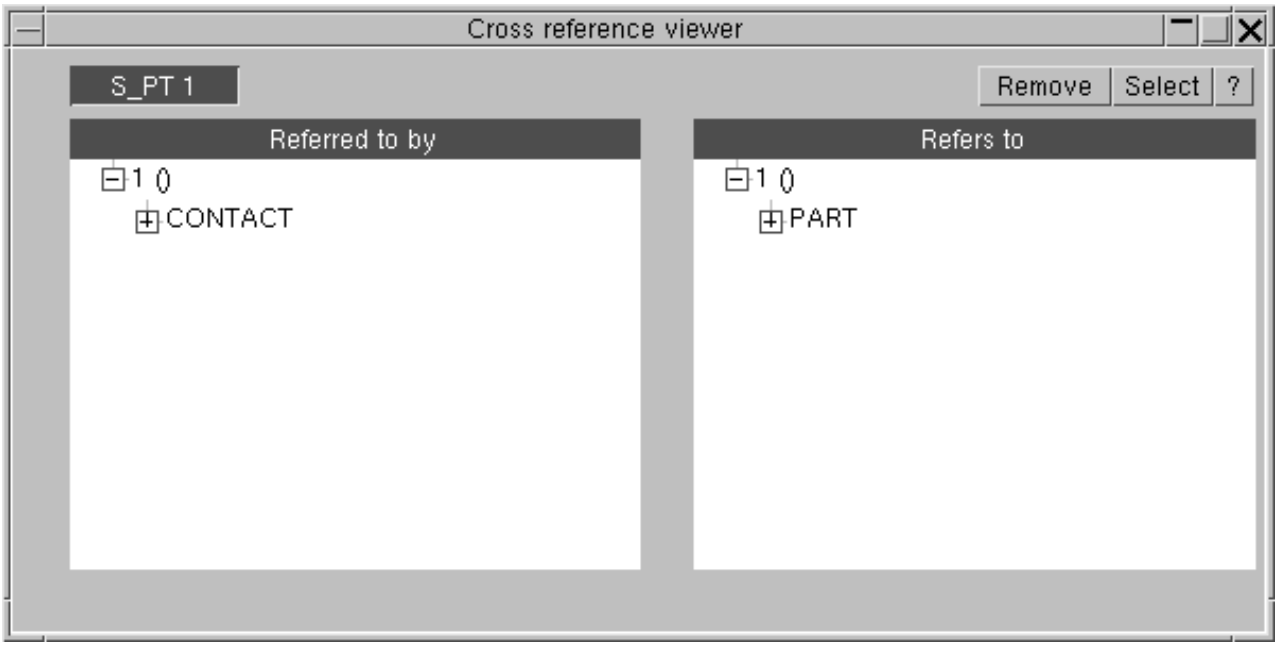


Once you have chosen an item the viewer will display the references for that item. **Cancel** quits the object menu and returns you to the normal cross reference display.

6.45 .2 How cross references are displayed

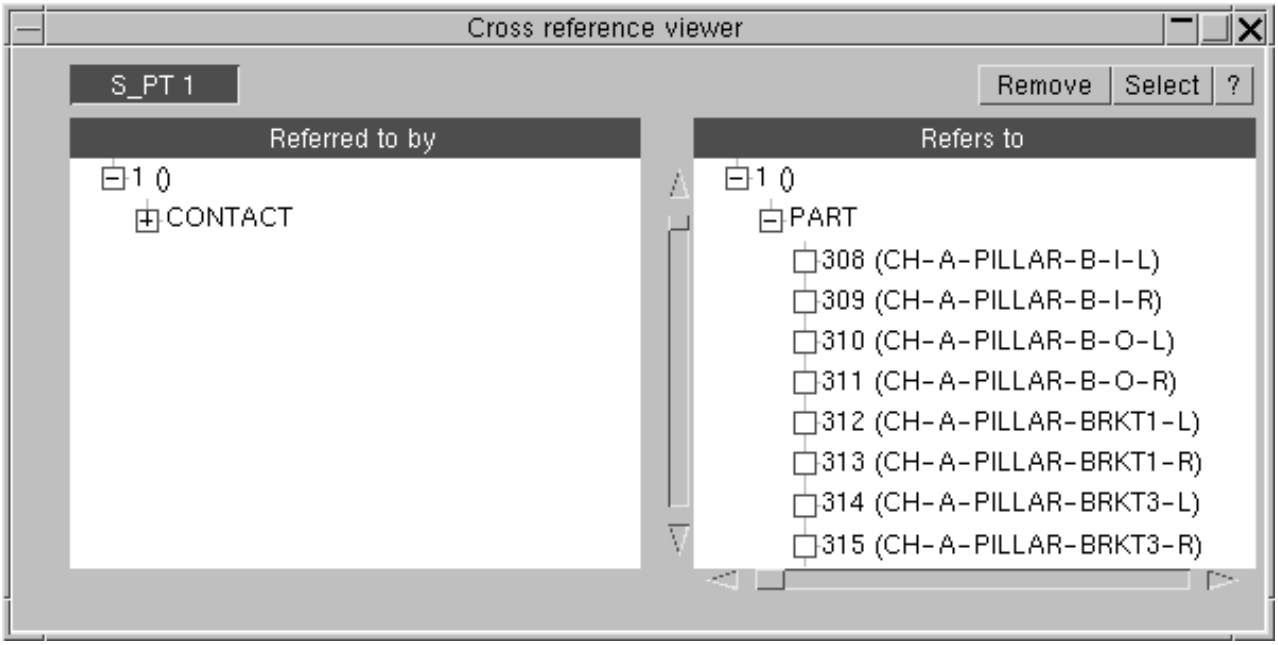
Cross references for an item are displayed using 2 ’tree’ views.
The left hand tree shows the items that are **referred to by the selected item**. For example below `*SET_PART 1` is referred to by `CONTACT` cards (as there is a `CONTACT` branch in the left hand tree).
The right hand tree shows the items that the **selected item refers to**. For example below `*SET_PART 1` refers to `PART` cards (as there is a `PART` branch in the right hand tree).

A tab is shown (**S_PT 1**) at the top of the panel for each item (only `*SET_PART 1` in this example). If multiple tabs are shown you can switch between them by pressing the tabs.

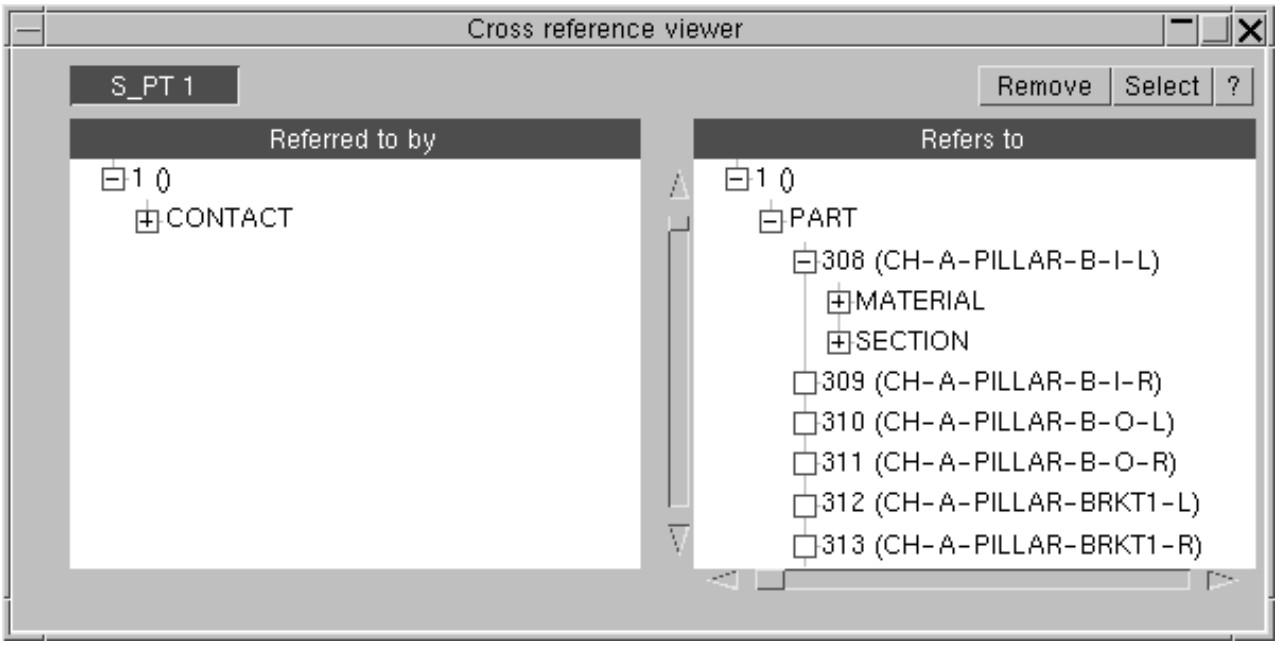


To expand a branch click on the ⊞ symbol. For example to see the parts that `*SET_PART 1` refers to click on the ⊞ on the **PART** branch. The branch is expanded and the parts are shown.

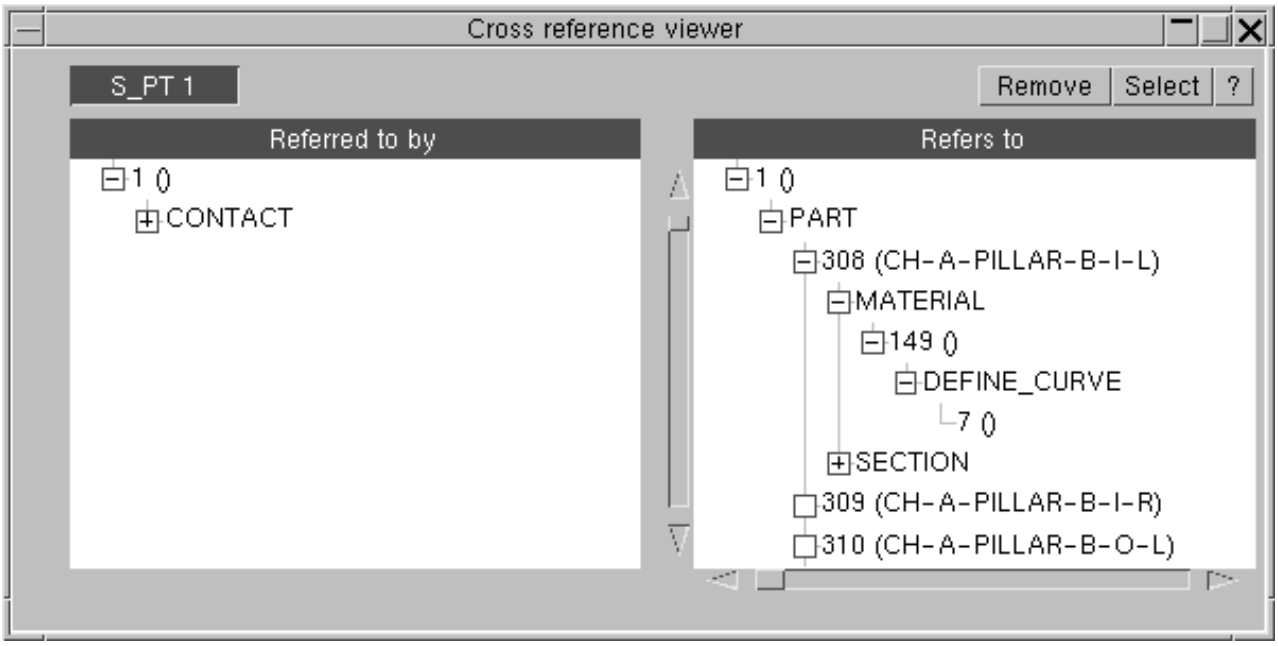
The cross reference viewer allows you to carry on looking for references, expanding branches as required. So, for example, we may want to look to see what PART 308 refers to. Does it have any references? At the moment we don’t know. Primer does not calculate all the references in the tree at the beginning as this could take a long time for a large model. Instead it looks for cross references when each branch is clicked on. When Primer does not know if a branch has cross references a ☐ symbol is shown instead of a ⊞ symbol. So, in the example below we do not know if any of the parts have cross references yet.



To see if part 308 has references click on the ☐ symbol. In this example part 308 does have references (`MATERIAL` and `SECTION` references) so the branch is expanded. If the branch did not have any references the ☐ symbol would just disappear.



We can continue along the branches as required. In the example below `SET_PART` 1 refers to `PART` 308 which refers to `MATERIAL` 149 which refers to `DEFINE_CURVE` 7. Curve 7 is the end of the branch as there is no ☐ symbol.

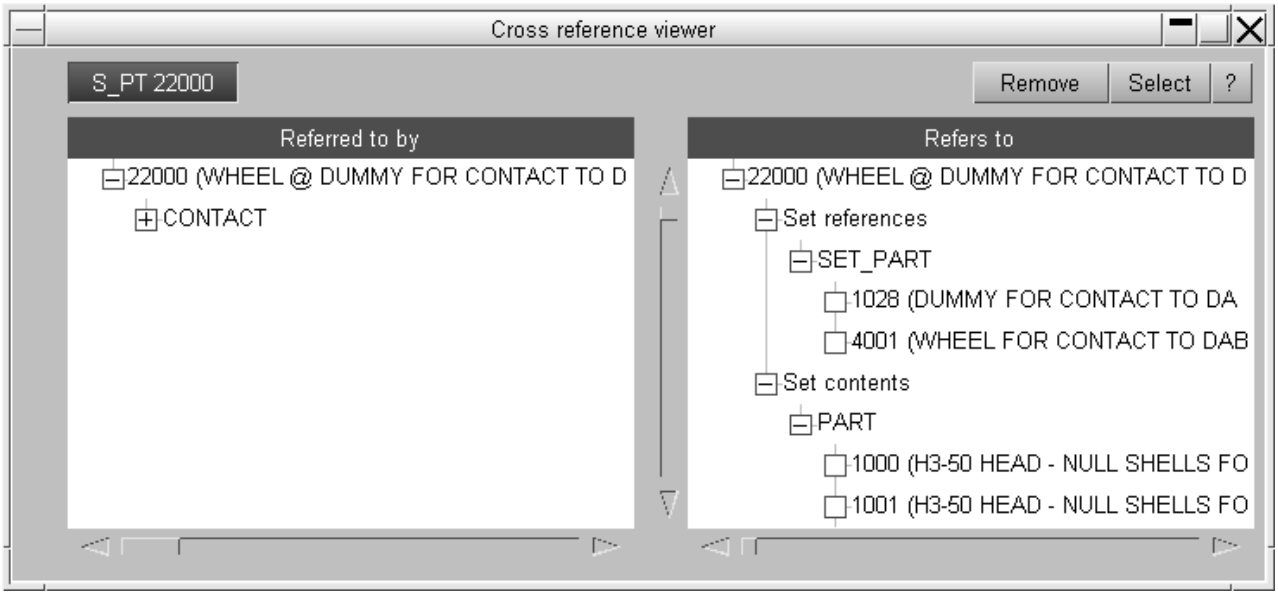


Displaying set cross references

Some of the newer set options in LS-DYNA such as *SET_xxxx_GENERAL, *SET_xxxx_ADD and *SET_xxxx_INTERSECT allow you do define the set contents by boxes, combinations of other sets etc. As well as seeing the items *referenced* in the set definition you also want to see the *actual* items that will end up in the set.

For these set types PRIMER shows 2 branches. The **Set references** branch shows the items referred to by the set. The **Set contents** branch shows the actual items that will be in the set when LS-DYNA is run.

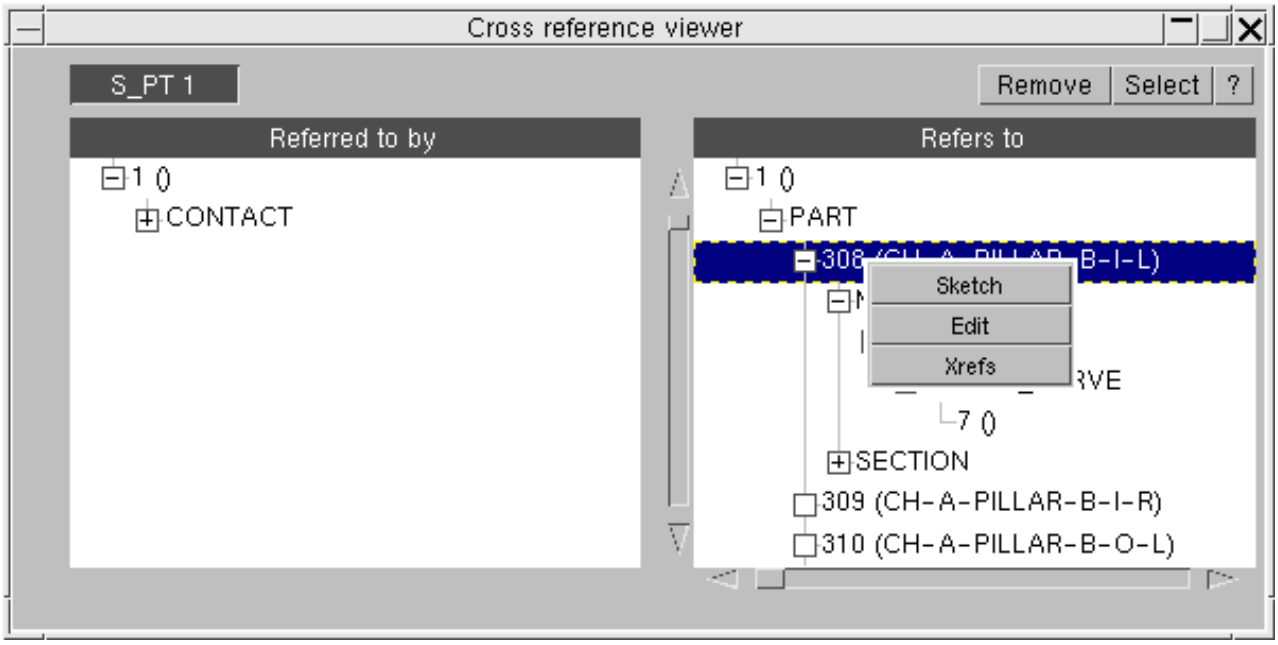
For example in the image below *SET_PART_ADD 22000 refers to part sets 1028 and 4001 in its definition. However the parts 1000, 1001, ... are the actual parts that will be in the set.



6.45 .3 Sketching, editing and references for items

Right clicking on an item in either tree brings up a popup menu. If the item is sketchable a **Sketch** button is available, and if there is an edit panel for the item an **Edit** button is available that will allow you to edit the item.

The **Xrefs** button in the popup creates another tab in the viewer for that item. For example, below selecting **Xrefs** for `PART` 308 will create another tab with `PART` 308 at the top of the tree.

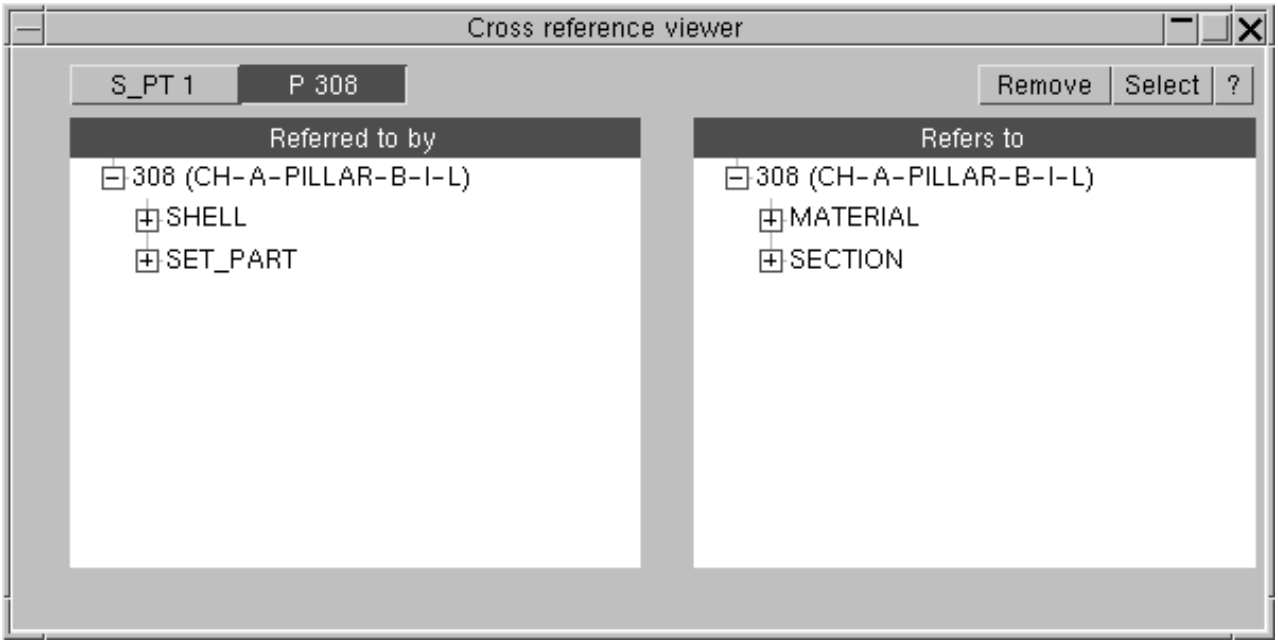


Other actions are also availabe in the right click popup panel. These include options for adding entities to the clipboard, visual options (blank/unblank/only) and deletion.

6.45 .4 Multiple tabs

The cross reference viewer enables you to have multiple tabs in the same window. Each tab can have a different item at the top of the tree.

For example, in the above images we have been looking at the references to `SET PART` 1 and we have found that it contains (**refers to**) `PART` 308. We now want to see what elements are in `PART` 308. i.e. we want to see which shells `PART` 308 is **referred to by**. This means we need to use the left hand (**referred to by**) tree. In the image above we right click on `PART` 308 and select **Xrefs**. This makes a new tab for `PART` 308 (image below) and we can now see the shells in the tree.



Remove will delete the current tab from the window. **Select** maps an object menu and allows you to select another item. Another tab will be created for the item.

7 Part Tree and Table

7.1 Part Tree

7.2 Part Table

7.3 Part Compare

7.0 Part tree and table.

The part table gives a clear and intuitive method for viewing part properties and manipulating part or parts quickly and efficiently see section 7.1 for more details

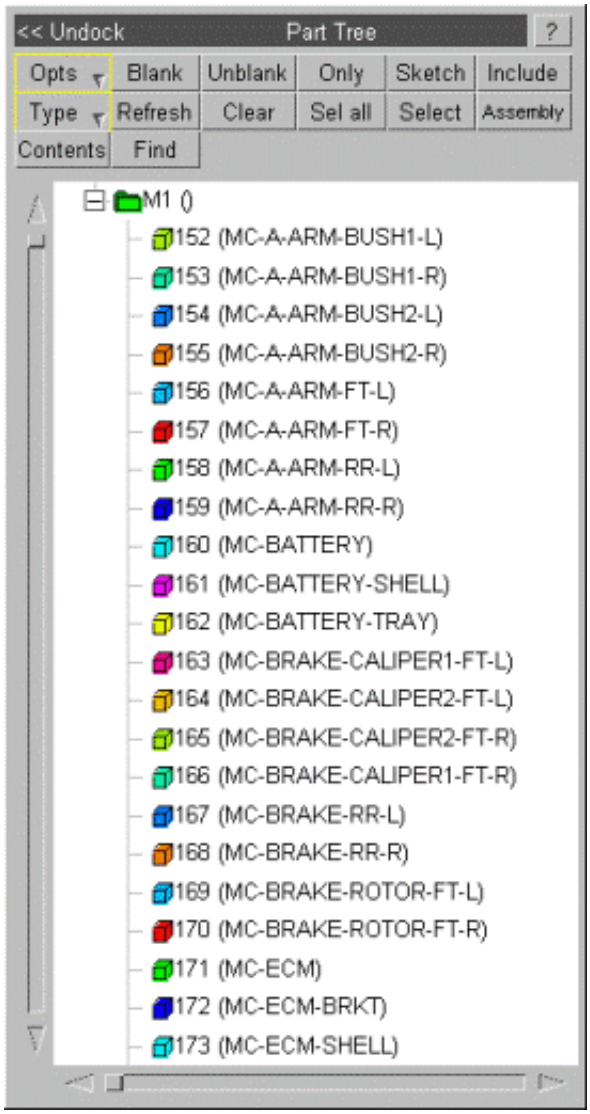
The part tree is a powerful visualisation and manipulation tool based on the organisation of parts within the model see section 7.2 for more details

Part compare is a function which drives the part table to find differences in properties between ostensibly matched parts, see section 7.3 for more details.

7.1 PART TREE

This enables quick navigation around a Model. It is possible to manipulate, view and edit entities in a quick and easy fashion. The part tree is focused around part manipulation, but it is also possible to view other types - e.g. materials, shells. There are various types of view expansion and contraction available, multiple selection and key viewing functions to hand. This makes the part tree a very powerful Primer tool and therefore it is constantly available through the tabbed area below the tools and keywords on the right hand side:

The part tree defaults to a view of the parts within the model, a typical display is shown here (with parts and materials displayed):

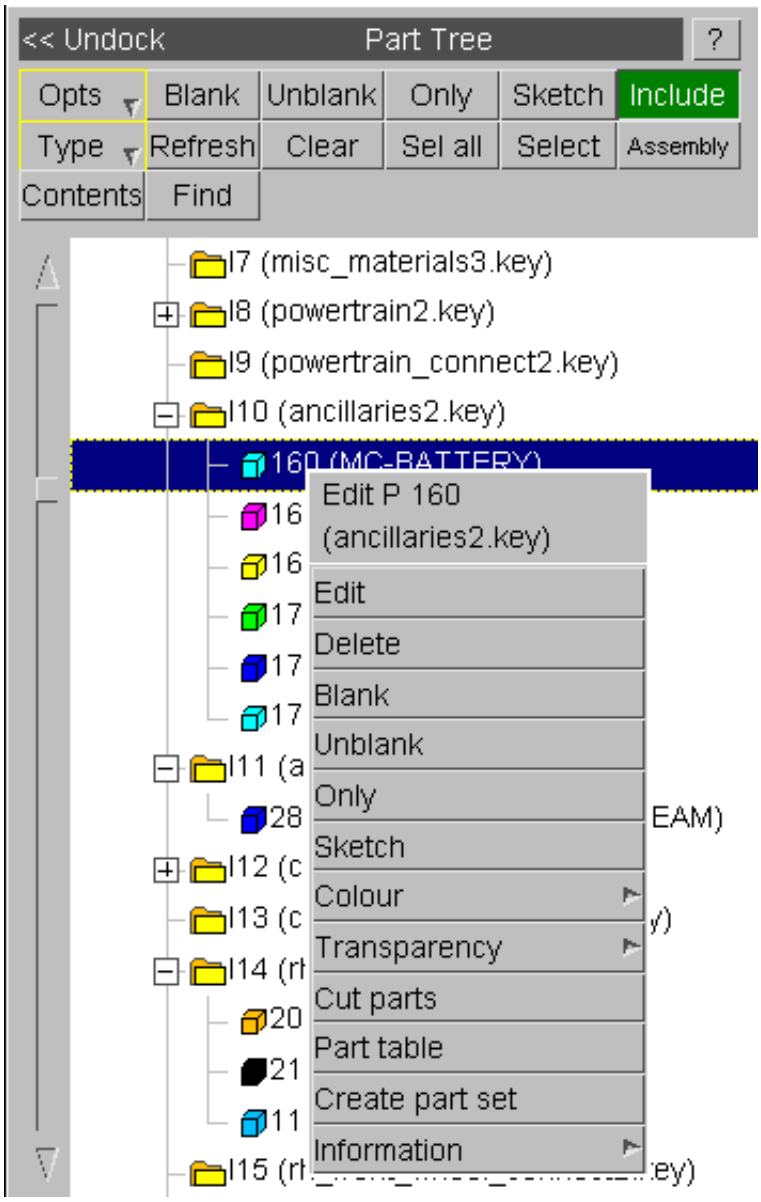


Part Tree Behaviour

Items can be selected by left-clicking anywhere on their row. Where selecting more than 1 item would be valid you can hold <ctrl> whilst clicking to select multiple items. Alternatively the <click> (start of range) .. <shift><click> (end of range) method (cf Windows) may be used.

Clicking on the [-] button next to models / include files / assemblies will collapse branches. Collapsed branches will have a [+] button which when clicked will expand the branch.

Right-clicking on an item or a selection of items produces a pop-up menu with the options shown on the right (not all of these options will be available for some selections).



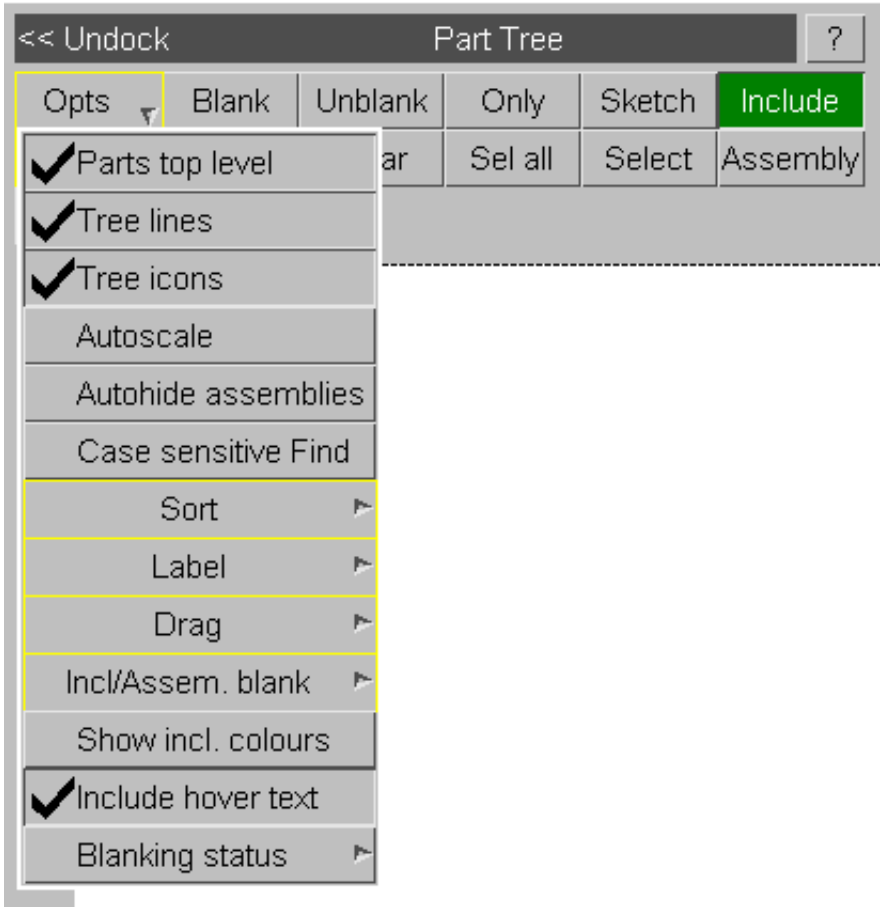
Edit	Brings up the standard editing panel for that item
Delete	Deletes the item
Blank	Blanks the item
Unblank	Unblanks the item
Only	Blanks all other items and unblanks the item
Sketch	Sketch the item
Colour	Colours the items (or elements associated with the item)as selected
Transparency	Sets the transparency the items (or elements associated with the item)as selected
Cut parts	Marks the part as those to be moved upon receiving a paste command
Part table	Brings up the Part table for the selected parts
Paste parts	Moves the last cut parts into the selected Include file or assembly

Create part set	Create a part set which the currently selected parts as the contents
Create part set and link	Available for assemblies. Creates a part set which the currently selected parts as the contents. If the assembly contents are updated, the part set is also updated
Information	Display information on the currently selected part
Make current layer	Makes the selected include file the current layer into which newly created entities will be put
Read Assembly file	Read an assembly file (applies to a model only)
Write Assembly file	Write out an assembly file (applies to a model only)
Create Assembly	Create an assembly in the the selected model or as a child of the selected assembly
Delete Assembly	Remove the assembly (but not the contents)
Rename Assembly	Rename the assembly
Assembly C of G	Give information on assembly C of G
Select parts to add	Opens an object menu to select parts to add to an assembly
Make current assembly	Makes the selected assembly the current layer into which newly created entities will be put
Clear current assembly	Clears the current assembly (newly created entities will not go into an assembly)
Add to Clipboard	Add assembly contents to the clipboard
Remove from Clipboard	Remove assembly contents from the clipboard
Replace Clipboard	Empty the clipboard contents and replace with assembly contents
Flatten all assemblies	Flatten all assemblies in the selected model (confirmation will be required)
Edit comments	Edit the header comments for a model or include file
Rename Include file	Changes an include file name
Suspend transformation	Temporarily suspends the transformation on an include transform. This can also be done from the Include tool.
Reinstate transformation	Reinstates the transform on a previously suspended include transform. This can also be done from the Include tool.

Part tree top menu bar

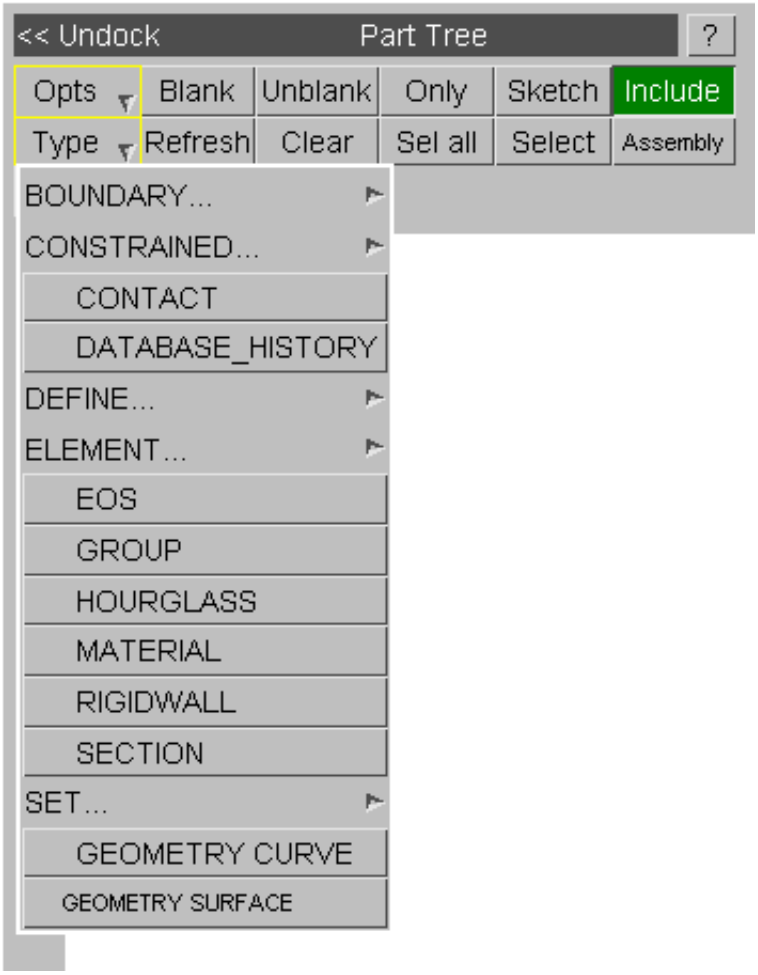
The top menu bar allows quick access top common Primer functions, as well as controlling how the part tree behaves and is displayed.

Opts



There is a range of options for controlling the part tree available via the **Opts** pop-up menu. These include how items are labeled and ordered as well as whether assemblies, tree lines and icons are drawn. The **Drag** option controls which items are moved when using the part tree to move items between include files/assemblies. For example when moving parts to a different include file you may wish to move the sections and materials as well. The **Incl/Assem blank** option allows you to control how items are treated when using blank/unblank/only on the part tree. For example, the elements within a part may be in a different include file to the part, but you may still wish to blank/unblank/only the elements. **Show incl colours** will display include colours if colours are set for each individual include file. **Include hover text** will display hover text as your cursor moves over the part tree contents. Turning on **Blanking status** will display in the part tree information for blanked/unblanked parts/entities. For example, if half the parts are blanked within one include file, a graphical bar will be shown for that include file displaying that half the contents are blanked. The blanking status can be changed from part based (default) to either element based or entity based.

Type



From the Type pop-up menu it possible to select a variety of different item types to be displayed in the tree in addition to parts. These appear below the parts in the tree, and most of the options (edit, blank etc.) are available through the "right-click" menu.

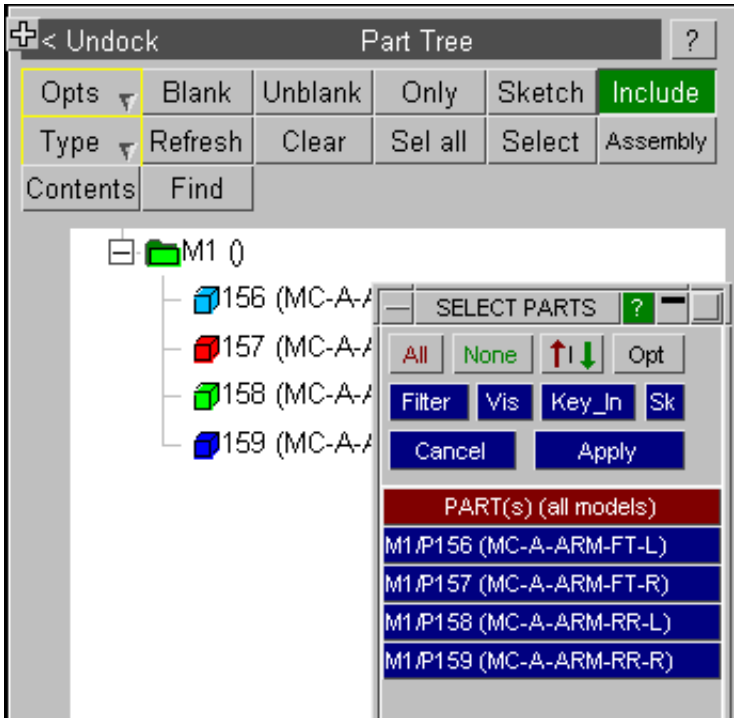
Blank / Unblank / Only / Sketch

One use of the part tree is as easy way access to blanking commands. **Blank**, **Unblank**, **Only** (blank all other items) and **Sketch** commands can be applied to the currently selected items.

Sel all / Clear

The **Sel all** and **Clear** buttons can be used to select all items and empty the selection respectively.

Select

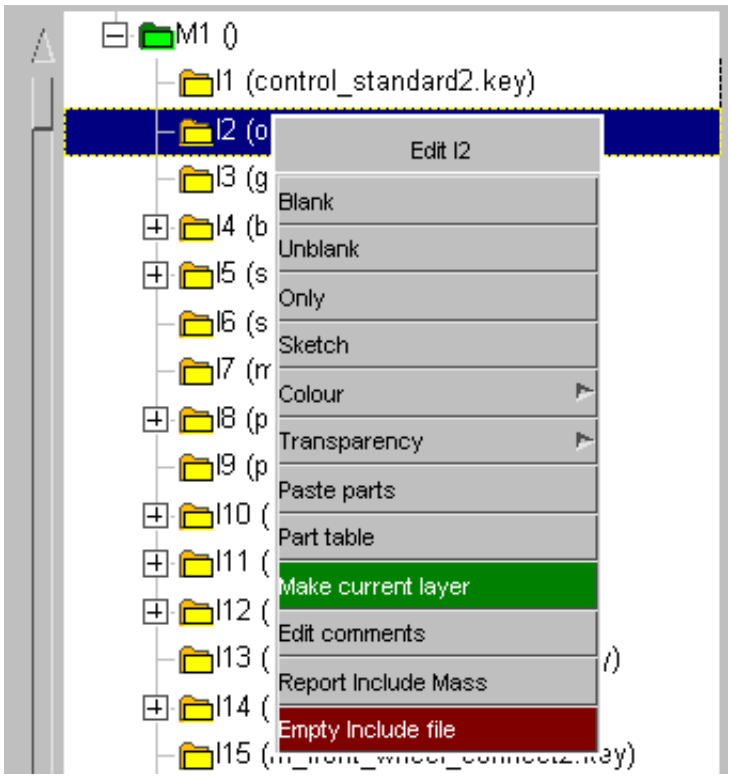


The **Select** button invokes an object menu for selecting parts. Selection can also be made via the Quick Pick option "Locate in Tree".Note that selections can be Include files, Assemblies or Models as well as parts. For example, click Only then an Include file to display only that Include file.

Include

The Include and Assembly buttons determine what type of hierarchy is displayed.

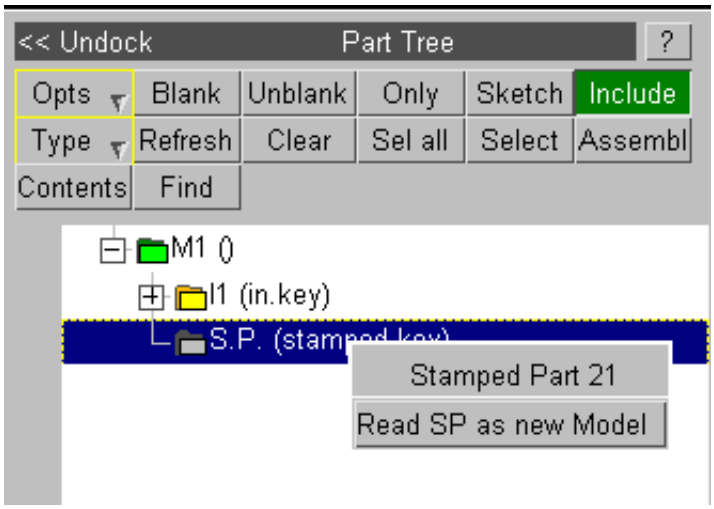
When in Include mode, the Include file structure of the model is shown. Parts can be dragged from one Include to another- this has the same effect as putting the parts on the Clipboard and moving to an include file with the "find referenced items" option (nodes and elements are moved in addition to the part cards). It is also possible to "Make current layer" which sets the current layer to the relevant include file (the layer is where Primer creates new entities).



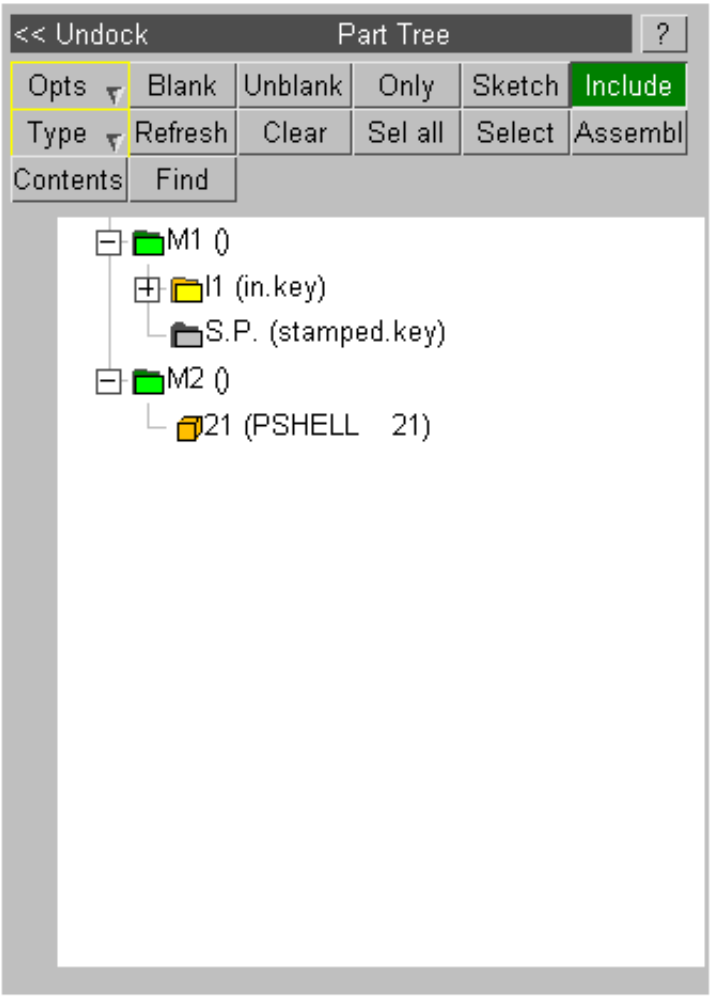
You can also cut and paste parts using the "right-click" menu. When a single or multiple selection of parts is shown, right click and cut parts. Then right-click over the include file and paste parts (shown above).

Read include stamped part

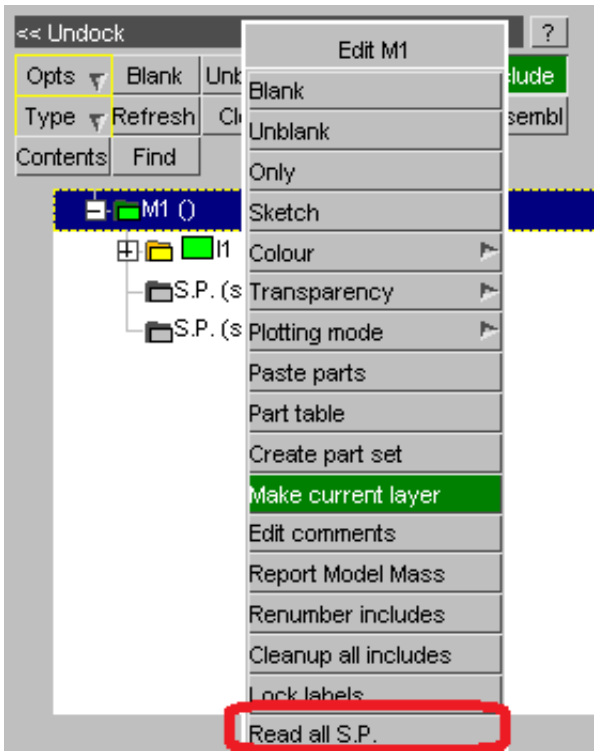
*INCLUDE_STAMPED_PART definitions are also shown in the part tree with **S.P.** shown before the filename and a grey coloured folder icon. The stamped part definitions cannot be dragged. An option "Read SP as new Model" is available when right clicking on them (see below).



When "Read SP as new Model" is clicked, a new model will be read into PRIMER and PLASTIC STRAIN contours will be automatically contoured.



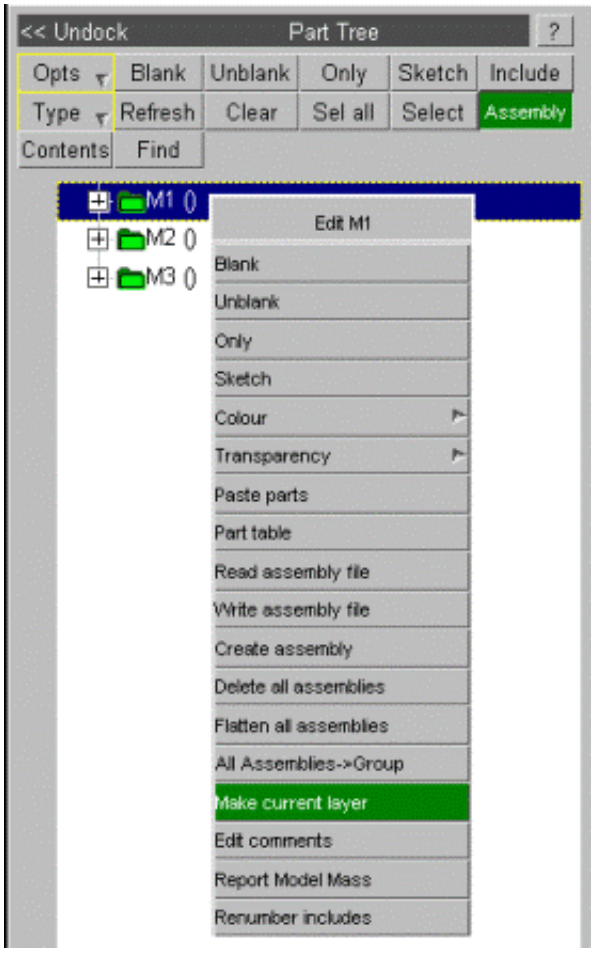
It is also possible to read all the *INCLUDE_STAMPED_PART definitions in the entire model by right clicking on the model icon and clicking on the ’Read all S.P." button. This read function automatically merges any nodes which may be present and a summary of the labels decashed/merged is presented to the user. Stamped parts can also be read in using the quick pick menu for *PART.

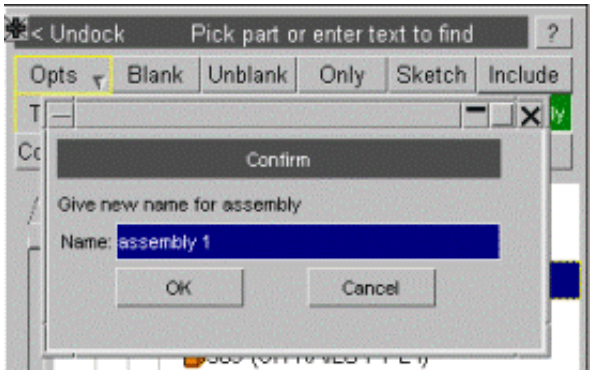

<< Undock
?
Opts
Blank
Type
Refresh
Contents
Find
M1 ()
l1
S.P. (s
S.P. (s
Edit M1
Blank
Unblank
Only
Sketch
Colour
Transparency
Plotting mode
Paste parts
Part table
Create part set
Make current layer
Edit comments
Report Model Mass
Renumber includes
Cleanup all includes
Lock labels
Read all S.P.

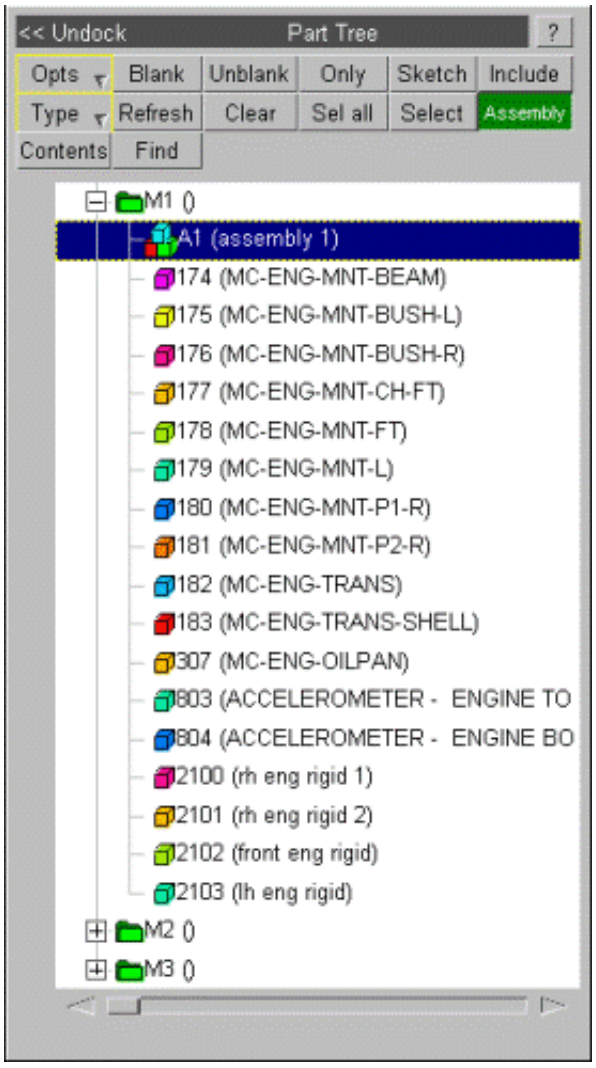
Assembly

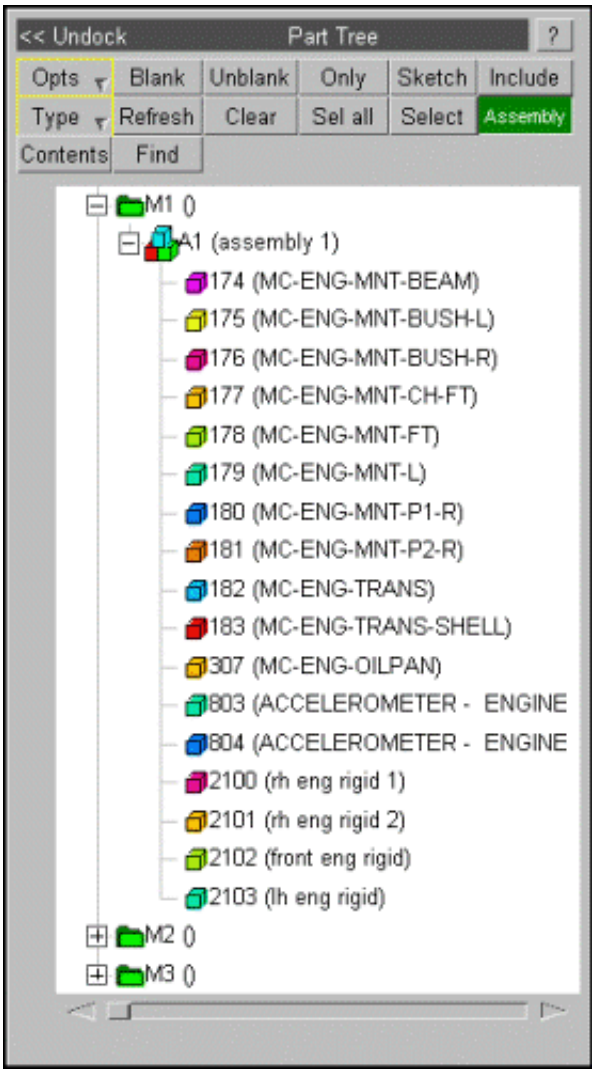
Assemblies are user-defined hierarchical groupings of entities. They exist only in Primer and do not effect the output analysis file (they are written in comment lines). Assemblies provide a way of grouping entities together to enable quick model manipulation and viewing. Entities from different include files can be grouped together, and the hierarchy is stored in what is known as an **assembly file**. Note in earlier versions of Primer, only parts could be placed in assemblies. Now, any entity type can be placed in an assembly.

The assemblies are created by right-clicking on a model (or existing assembly) in the part tree. Parts can then be dragged into the assembly as shown below. The clipboard can also be used to move entities into an assembly.









When the keyword file is written out, the assembly to which each entity belongs will be written with the entity data as a comment. The hierarchical structure can be written only as a separate assembly file (right-click on the model in the part tree) - this is to avoid duplicated or missing assembly hierarchies when working with the model in Include files. If the keyword file is written but not the assembly file, when the model is next read in, the part tree will show a list of any assemblies that contain entities but flattened into a 1-layer-deep structure.

Note that assembly hierarchies are automatically created from Hypermesh and ANSA hierarchy comment data should this exist in the input deck. Upon keyout you can choose if you wish to save the assembly hierarchy data in Primer, Hypermesh or ANSA format.

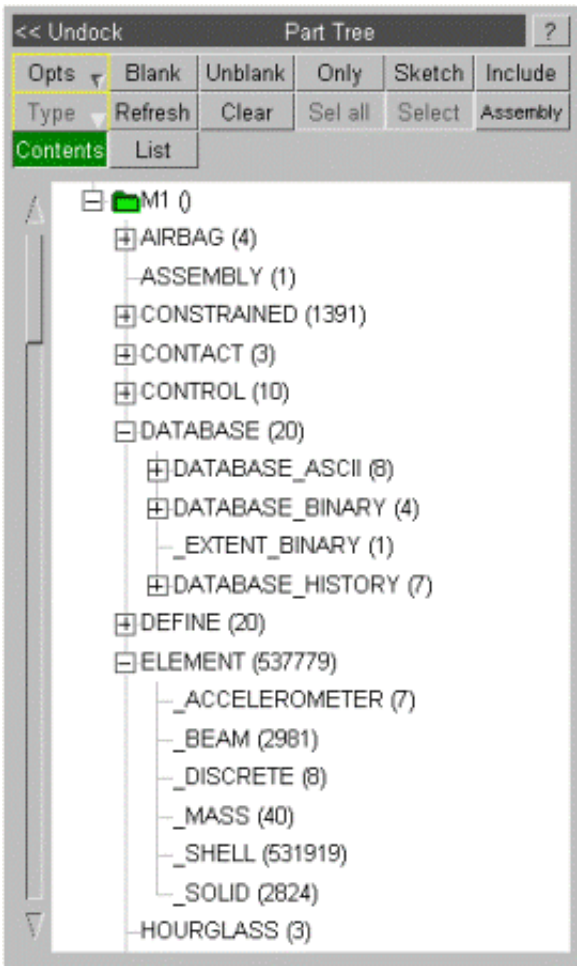
Find

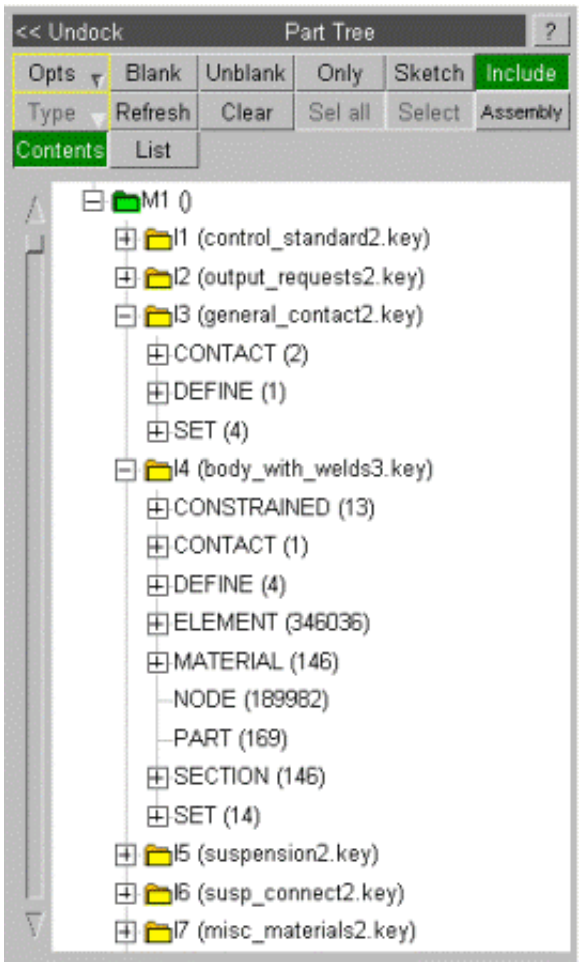
The **Find** button is available only when **Contents** is switched off, and it gives a search option. Text or an ID number is entered in the text field. Primer finds a part whose title contains the text, or a part with an ID matching the number. The arrows determine whether the search direction is up or down from the current selection. **Next** will find the next matching item. The search will only find matches for currently enabled options (i.e. if id is disabled and items are labeled by name only a search for part 15 will return no matches regardless of its presence in the tree).

Contents

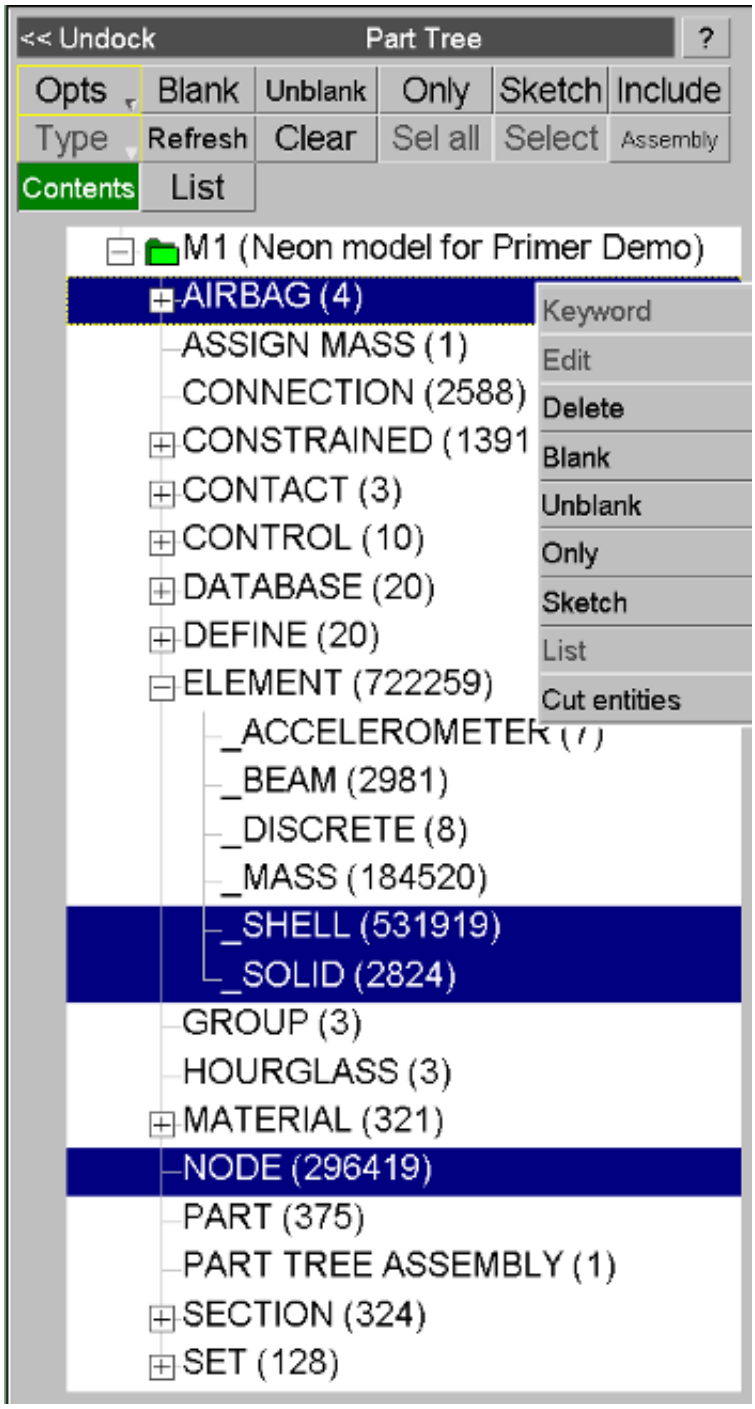
When **Contents** is switched on, the part tree displays each keyword in the model along with the number of entities of that particular keyword type arranged in alphabetical order of keywords. If **Include** is switched on in conjunction with **Contents**, the part tree displays the keywords and their numbers by include files. On the other hand, if **Include** is switched off, the keywords and their numbers are displayed for the entire model. The following illustrations depict the

part tree in **Contents** mode with **Include** switched off and on respectively:





Branches can be selected from the tree and operations such as blanking, unblanking, etc., can be performed. Not all operations are permitted for all keywords, and the buttons for the non-permitted operations are greyed-out in the popup box as seen below:

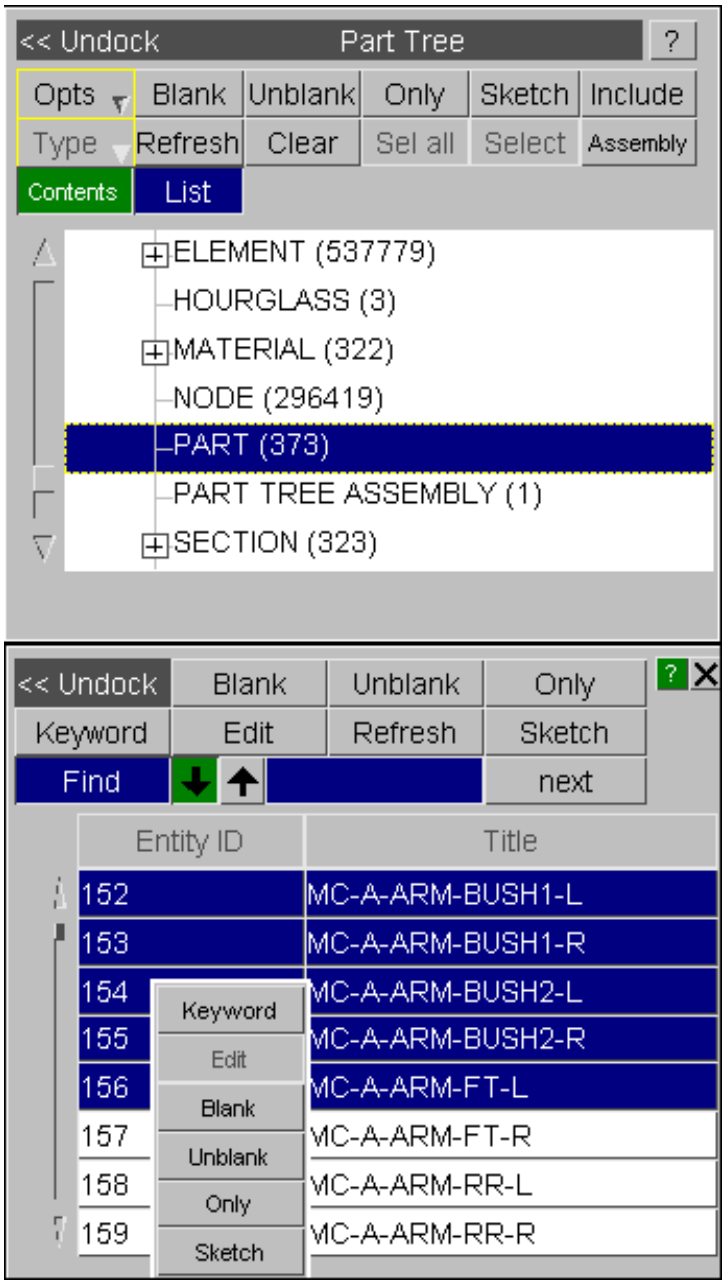


In addition to the options available at the top of the tree, the pop-up box offers the following new options when the part tree is in **Contents** mode:

- **Keyword**: This option is available if the entities that belong to a selected keyword can be manipulated via a generic keyword editor in Primer. When clicked, this option invokes the generic keyword editor containing all the entities that belong to the selected branch. This option is obviously not available when multiple keyword are selected in the part tree.
- **Create**: Create option is permitted for entities which have edit panels.
- **Edit**: If only one keyword is selected from the part tree, and if only one entity of the selected keyword type exists in the model, the editing panel for the selected entity can be directly invoked using this option.
- **Cut/Paste**: Allows you to move entities to different include files.
- **List**: This button invokes the **Contents List** window for the selected keyword. The Contents List window is described in the following section.
- **Delete**: This will bring up the standard deletion panel to allow you to delete the selected entities and associated entities.

List

This button replaces the **Find** button when **Contents** is switched on, and is therefore available only when the part tree is in **Contents** mode. If multiple entities of a keyword type are permitted in a model, the entities of that type can be listed in a single **Contents List** window by means of the **List** button. Keywords such as *CONTROL_ and *DATABASE_ASCII_ are exceptions as only one entity of each of such keyword types are allowed in a model. The **Contents List** window can alternatively be invoked by means of the **List** option in the pop-up box for the keyword selected in the part tree. The **Contents List** window can be seen in the illustration below:



The entries in the Contents List window can be selected and subjected to operations such **Keyword**, **Edit**, **Blank**, **Unblank**, **Only** and **Sketch**. Note that not all operations are available for every keyword type, and that the buttons for the non-permitted operations are greyed-out in the popup box as seen above.

Just as in the part tree itself, option **Keyword** invokes the generic keyword editor for the selection (if permitted by Primer), and option **Edit** invokes the editing panel for one single entity selected in the **Contents List** window. Right click **Delete** will allow you to delete the selected entities.

The **Contents List** window also has a **Find** button that looks and functions in a similar fashion to the one in the main **Part Tree**.

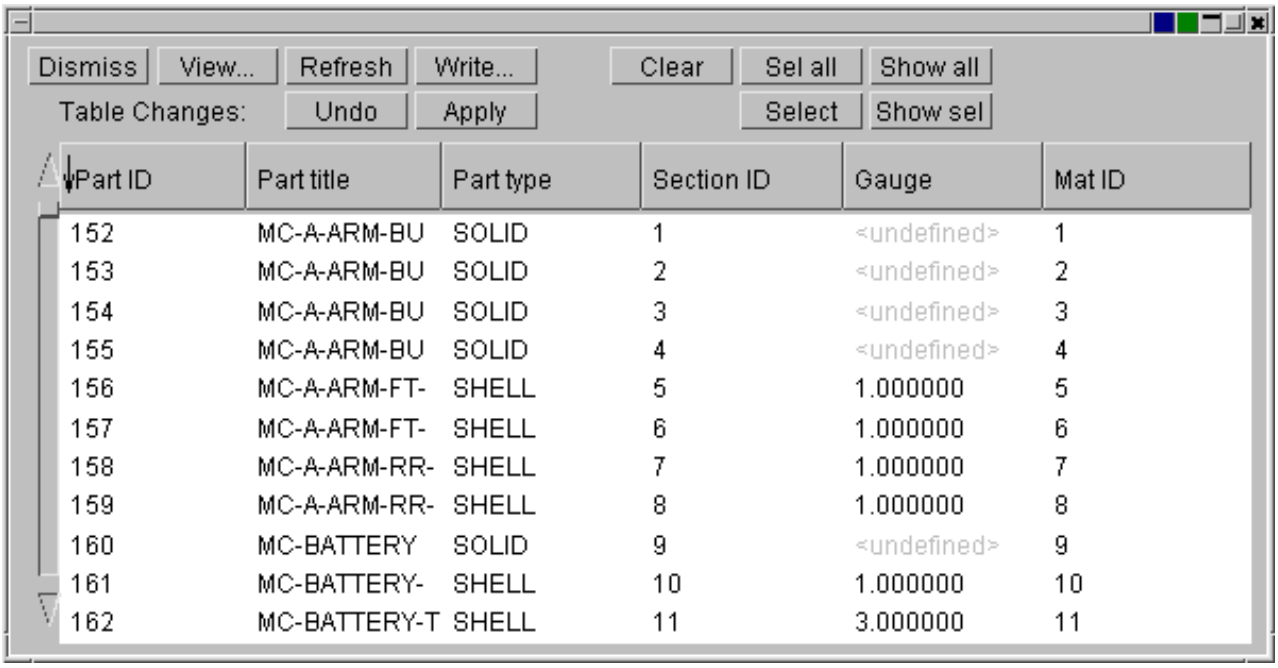
7.2 PART TABLE

The part table allows you to easily view and change information for parts in your model. The table can be started by either:

- Selecting parts/include files/assemblies/models in the **part tree**, right clicking and selecting **Part table** from the popup menu.
- Using the **Table** option in The **Part** menu.
- Using quick picking for PART with **Part Table** mode selected.

Each row in the table represents one part. By default the rows are sorted by ascending Part ID. The column used for sorting can be changed by clicking on the appropriate column header. Additionally clicking on a column header again will sort by descending order instead of ascending order. The column which is currently used for sorting has an arrow drawn on the header (in the example below it is Part ID).

If a column is invalid for a part **<undefined>** is shown. For example, below, **Gauge** is not valid for solid parts.

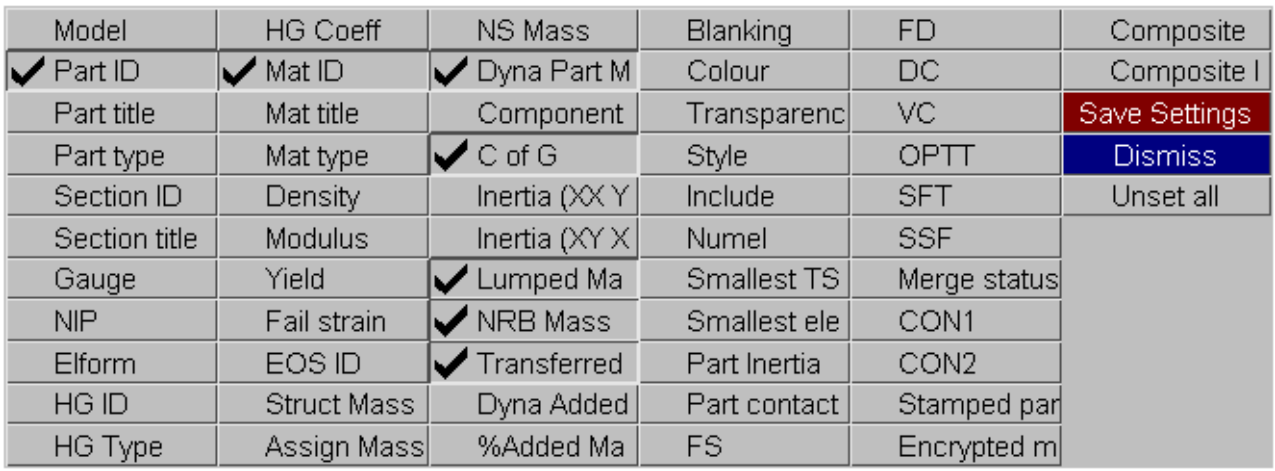


Part ID	Part title	Part type	Section ID	Gauge	Mat ID
152	MC-A-ARM-BU	SOLID	1	<undefined>	1
153	MC-A-ARM-BU	SOLID	2	<undefined>	2
154	MC-A-ARM-BU	SOLID	3	<undefined>	3
155	MC-A-ARM-BU	SOLID	4	<undefined>	4
156	MC-A-ARM-FT-	SHELL	5	1.000000	5
157	MC-A-ARM-FT-	SHELL	6	1.000000	6
158	MC-A-ARM-RR-	SHELL	7	1.000000	7
159	MC-A-ARM-RR-	SHELL	8	1.000000	8
160	MC-BATTERY	SOLID	9	<undefined>	9
161	MC-BATTERY-	SHELL	10	1.000000	10
162	MC-BATTERY-T	SHELL	11	3.000000	11

The part table will resize as required as the window size is changed.

Changing which columns are shown

There are many different fields that can be shown. To add or remove a column press **View...** which will bring up the list of field types as shown below. The fields which are currently shown will have a tick symbol next to them



Changing the table columns

The following columns are available for display in the part table under **View...**

Column	Explanation
Model	Model label
Part ID	Part label
Part title	Part title
Part type	Type of part (shell, solid etc.)
Section ID	Section ID part uses
Section title	Title of section part uses
Section SHRF	Section shear correction factor
Gauge	Gauge or thickness of part
NIP	Number of integration points
Elform	Element formulation
HG ID	Hourglass ID part uses
HG Type	Hourglass type
HG Coeff	Hourglass coefficient
Mat ID	Material ID part uses
Mat title	Title of material part uses
Mat type	Type of material part uses
Density	Density of material part uses
Modulus	Young's Modulus of material part uses
Yield	Yield stress of material part uses
Fail strain	Failure strain of material part uses
EOS ID	EOS ID part uses
Struct Mass	Structural mass (rho x vol)
Assign Mass	Mass added through lumped mass belonging assign mass
NS Mass	Non-structural mass (section card or *Element_mass_part)
Dyna Part Mass	Part mass (see below)
Component mass	Part mass (see below)
Lumped Mass	Mass applied through lumped masses on nodes of deformable part (including assign mass)
NRB Mass	Mass of nodes of deformable part that is attributed to NRB
Transferred Mass	Mass lost/gained where node is shared by both deformable & rigid element
Added Mass	Timestep Added mass (true value applicable for solid spotwelds see appendix 17)
Added Mass %	Percentage added mass
C of G	Centre of gravity
Inertia (XX YY ZZ)	Part inertia tensor
Inertia (XY XZ YZ)	Part inertia tensor
Blanking	Displays whether part is blanked or not
Colour	Colour of part
Transparency	Transparency status of part
Style	Current style of the part
Include	Include file the part resides in
Numel	Number of elements contained within the part

Smallest TimeStep	id & timestep of element with smallest timestep in this part
Smallest elem	id & characteristic length of element with smallest characteristic length
Part Inertia	Is _INERTIA applied to the part?
Part Contact	Is _CONTACTapplied to the part?
FS	_CONTACT field. Static coefficient of friction
FD	_CONTACT field. Dynamic coefficient of friction
DC	_CONTACT field. Exponential decay coefficient
VC	_CONTACT field. Coefficient of viscous friction
OPTT	_CONTACT field. Optional contact thickness
SFT	_CONTACT field. Option thickness scale factor.
SSF	_CONTACT field. Contact stiffness scale factor
Merge status	Rigid body merge status (master or slave)
CON1	CON1 field from any applicable rigid material card
CON2	CON2 field from any applicable rigid material card
Stamped part	Displays whether the part is in an INCLUDE_STAMPED_PART definition or not
Encrypted material	Displays YES if the part references an encrypted material card, otherwise displays NO
Composite	Displays YES if the part is of type PART_COMPOSITE, otherwise displays NO
Composite layers	Displays the number of layers, if the part is of type PART_COMPOSITE

Changing the default table columns

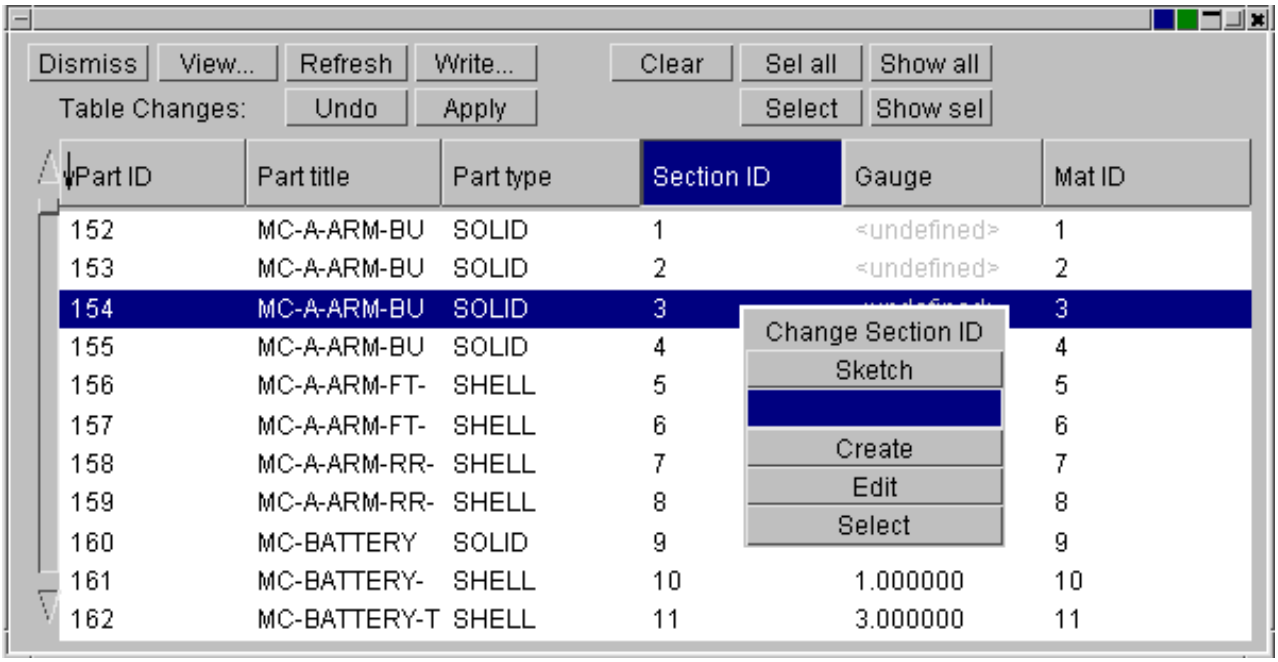
By default the part table will show the Part ID, Part title, Part type, Section ID, Gauge and Mat ID columns. If you want to change which columns are shown by default then change the columns shown to be the ones you want and press **Save Settings** in the **View...** popup. This will automatically add a preference `primer*part_table_columns` to your home oa_pref file with the appropriate columns.

Selecting rows in the table

Rows can be selected in the table by clicking with the mouse. To select multiple rows use the **`Ctrl`** key while clicking and to select a range of rows use the **`Shift`** key while clicking. There are also buttons at the top of the table to aid you in selecting and viewing parts in the table. The **Clear** button clears all current selections. The **Sel all** button selects all parts currently displayed on the table. **Select** will bring up an object menu and allow you to select parts using that method (i.e. being able to use various filters to select parts). **Show sel** will display in the table only those parts currently selected. **Show all** will bring back and display the original parts on the table.

Changing a value in the part table

To change the value for a field, select the parts that you want to change and then right click on the column that you want to change. This shows a popup menu allowing you to change the value. e.g. below Section ID is being changed for part 154.

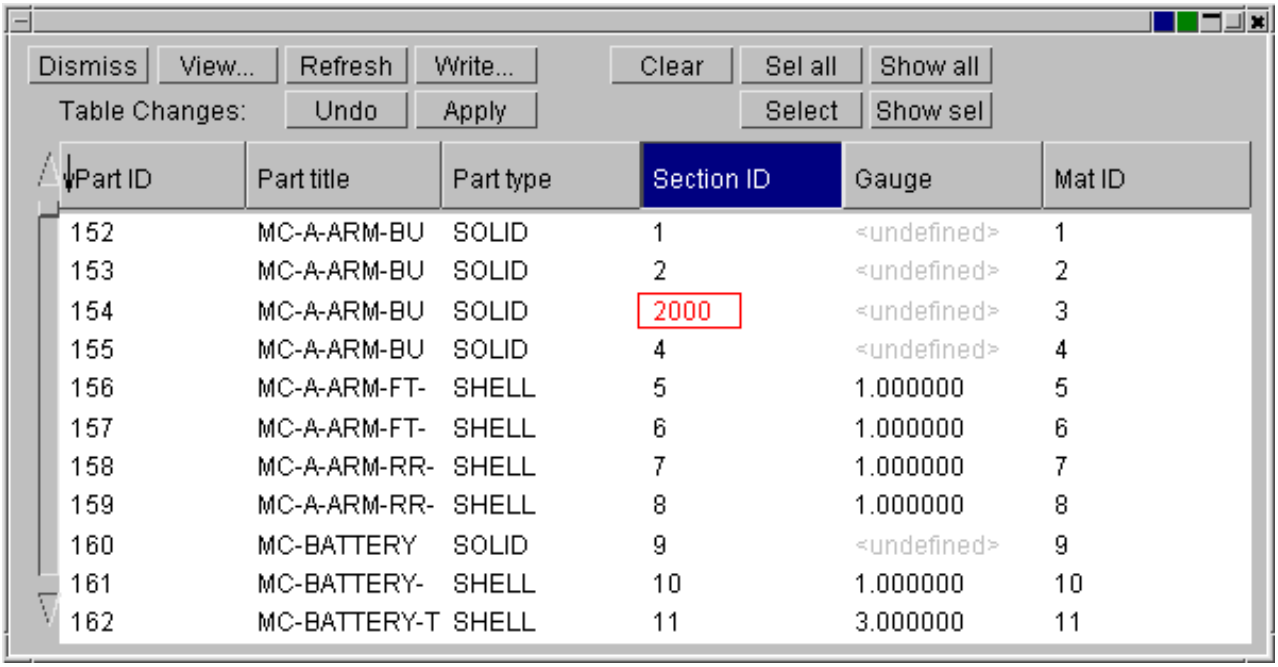


To change the value, type in the new value in the text box. Alternatively, if the field refers to an item (e.g. section ID, material ID) you can use **Select** to choose the item or **Create**/**Edit** to make a new item or edit the item respectively. Note for materials in the table there is the option to **Select from table**. This restricts the selection to only materials currently referenced by parts on the table.

Sketch will draw the currently selected parts.

If the table is used to change Part colour, transparency, blanking and drawing mode the new settings are applied immediately. For other values (e.g. title, gauge etc.) the values will not be updated for the parts until the **Apply** button is pressed. To indicate that a value has changed but not applied it will be shown in red. e.g. below, the section ID for part 154 has been changed to 1000 and so is shown in red. Changes can be undone by pressing **Undo**.

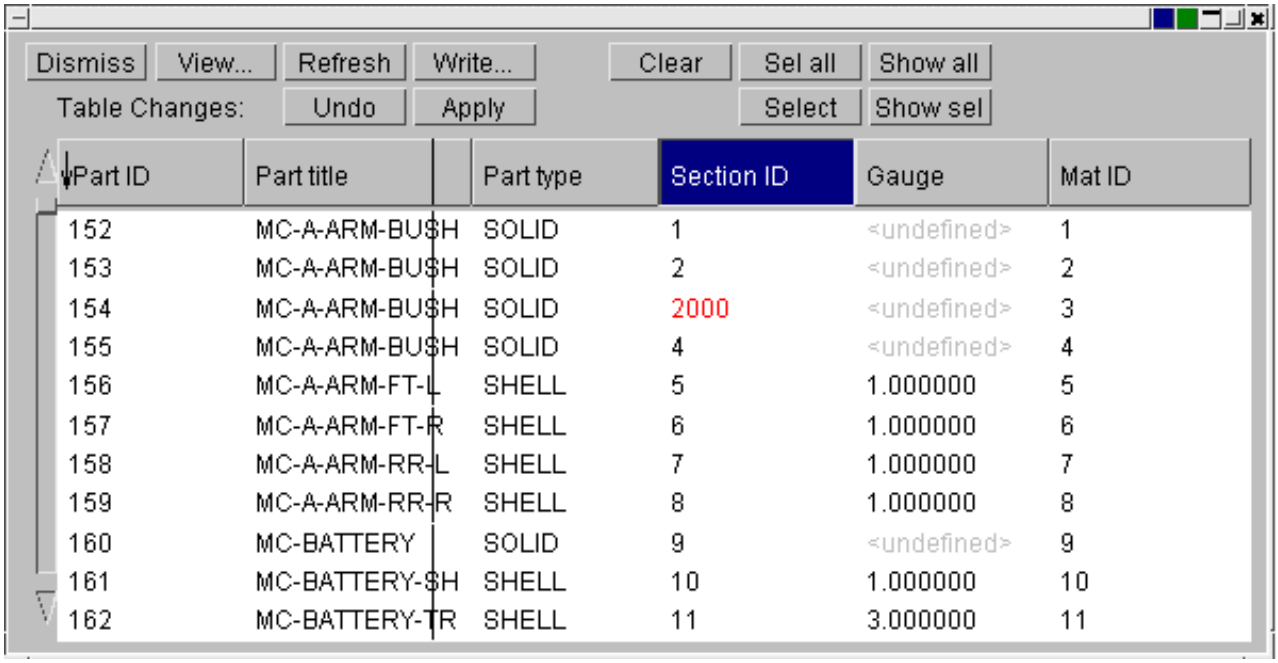
If the same section card applied to multiple parts and the user changes a value (such as gauge) on one part (or some but not all of the parts) Primer will create a copy of the original section card and modify that.



Changing the width of a column

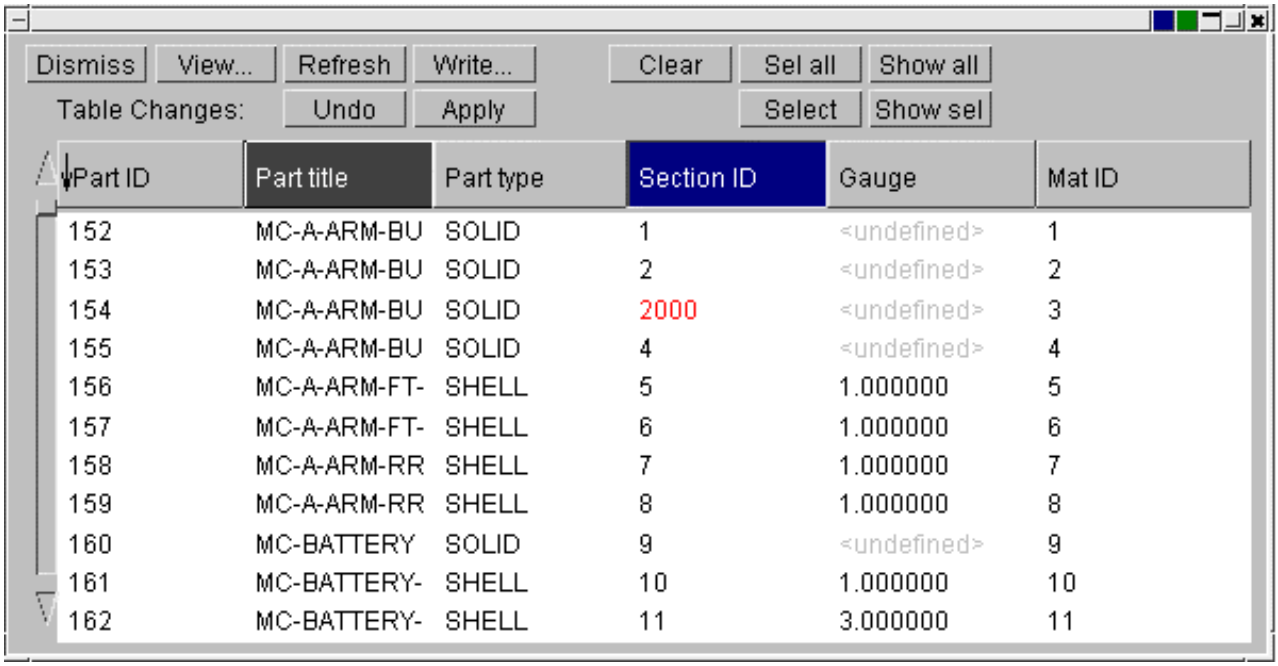
Columns can be made wider or narrower by clicking on the edge of a column header and dragging the mouse. The cursor symbol will change to a double ended arrow while you are dragging. E.g. below the use has clicked on the

header between **Part title** and **Part type** and is dragging to the left.



Changing the order of columns

The order of the columns can be changed by clicking on a column header and dragging the column left or right to a new location. When you are at a valid location the cursor will be a **+** symbol and when you are at an invalid location the cursor will be a **x** symbol. E.g. below the user has clicked on Part title and has started dragging the column. If (s)he wanted to move the column to be between the Gauge and Mat ID columns (s)he would drag the header until the cursor symbol changed to **+** between the 2 columns.



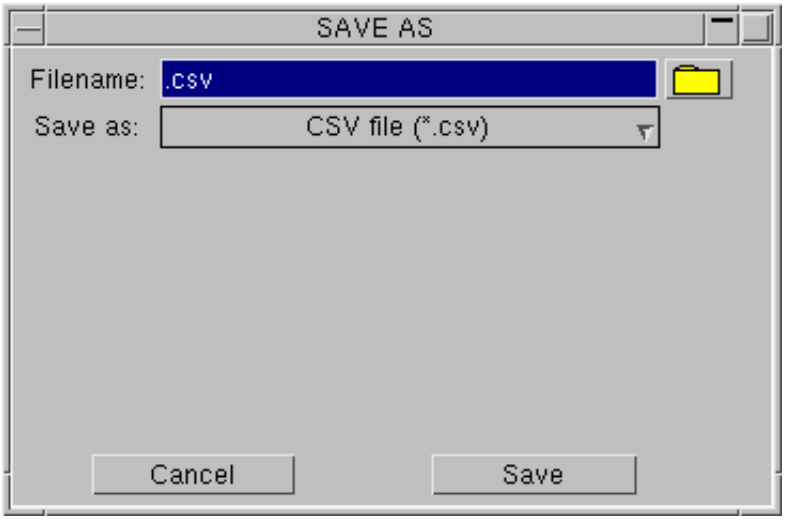
Saving part table information to file

The information in the part table can be saved to a file. Currently there are 3 different formats:

- A csv file suitable for importing into a spreadsheet such as Excel.
- A HTML file suitable for a web page, viewable by any web browser.
- A postscript file suitable for printing to a postscript printer or viewing with a postscript previewer such as ghostscript.

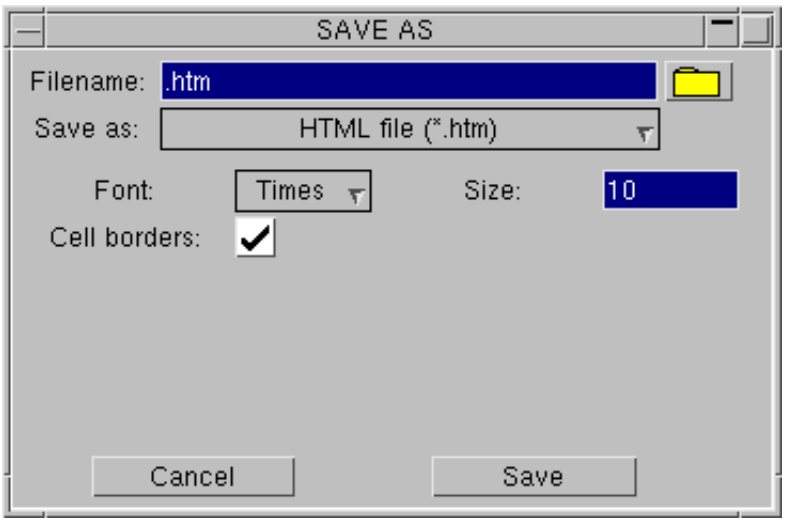
Choose the appropriate format using the **Save as:** popup.

CSV file



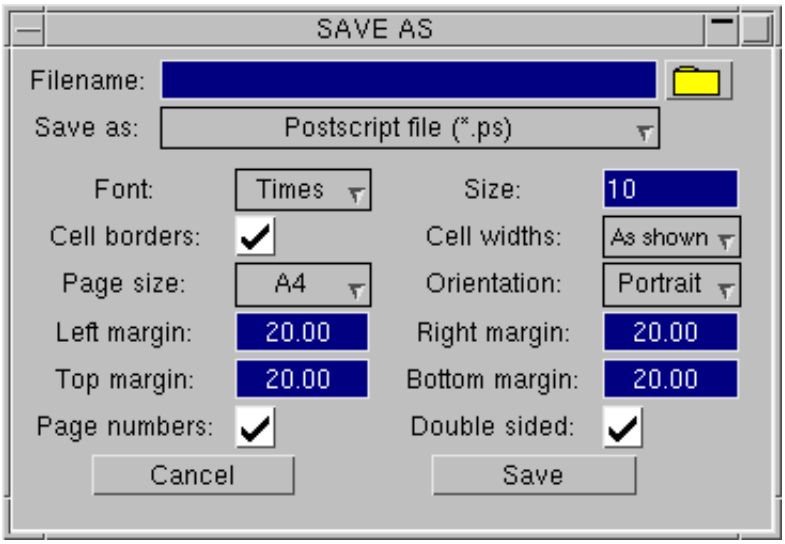
There are no options for CSV files. Give the name of the file to save.

HTML file



For html files you can choose the font, font size and if table cell borders are shown.

Postscript file



For postscript files you can choose the font, font size, table cell visibility, page size and orientation, margins, page numbering and double sided printing. Additionally you can choose to make all columns have equal widths or to use the column widths as displayed in the part table.

Mass in part table

The Part table allows display of different kinds of part mass, namely Structural mass, assign mass, nonstructural mass, dyna part mass, component mass, lumped mass, added mass and percentage added mass.

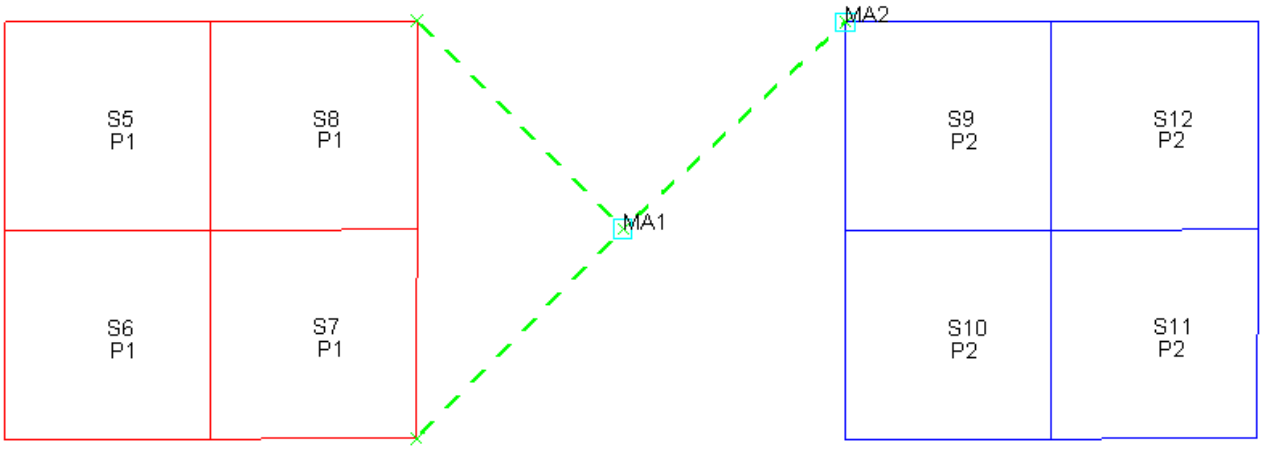
NS mass (non-structural mass) is the mass that applies on shell or beam parts as a result of mass per unit area setting (MAREA) on the section card. It may also be applied using the *ELEMENT_MASS_PART card.

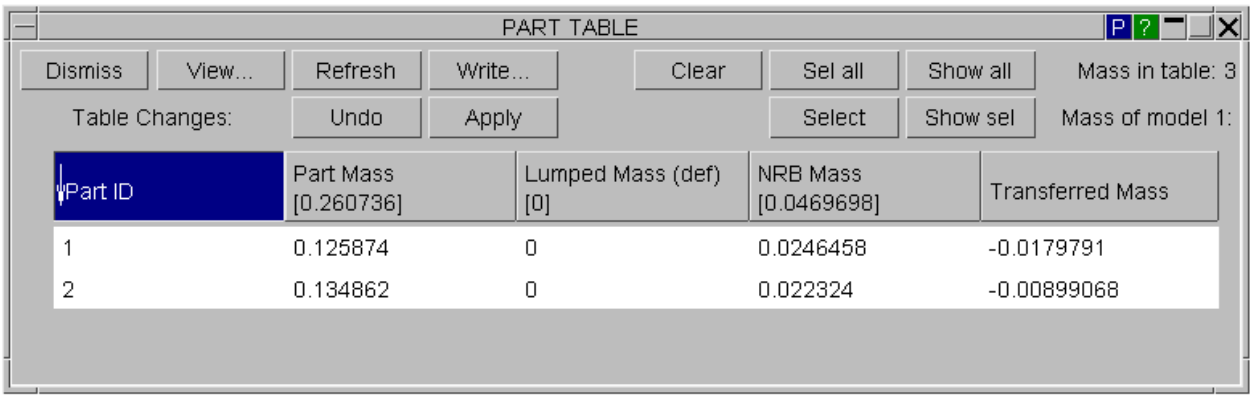
Lumped mass (def) is the sum of lumped masses attached to the nodes of the part, including assigned mass. For rigid parts the lumped mass (including masses on constrained extra nodes) is included in the Dyna Part mass, so it is not included in the column total (though the sum per part is listed for information).

Dyna part mass tries to use the same mathematical formulation as LS-Dyna. It is the sum of structural & non-structural mass belonging to nodes of part including lumped mass for rigid part (unless it is Part_Inertia). A deformable part 'loses' mass where nodes attach to rigid part/nrb. A rigid part 'gains' mass where it attaches to deformable nodes. Slaves in rigid body merges get zero mass, master parts acquire the mass of the slave(s).

Component mass is an attempt to describe the "engineering" mass of a part. This is the sum of structural & non-structural mass belonging to nodes of part including lumped mass for both deformable & rigid parts (unless Part_Inertia). Mass is NOT transferred from deformable to rigid parts/nrbs. Also in this context rigid body merges are ignored. The total of this column should be the model mass (without added mass).

Note on masses on nodes of nodal rigid bodies. Masses on nodes of nodal rigid bodies attached to a part will be included in the NRB mass column for the part. If the mass is on a node which does not directly attach to a part its mass will be shared amongst the nodes which do attach.

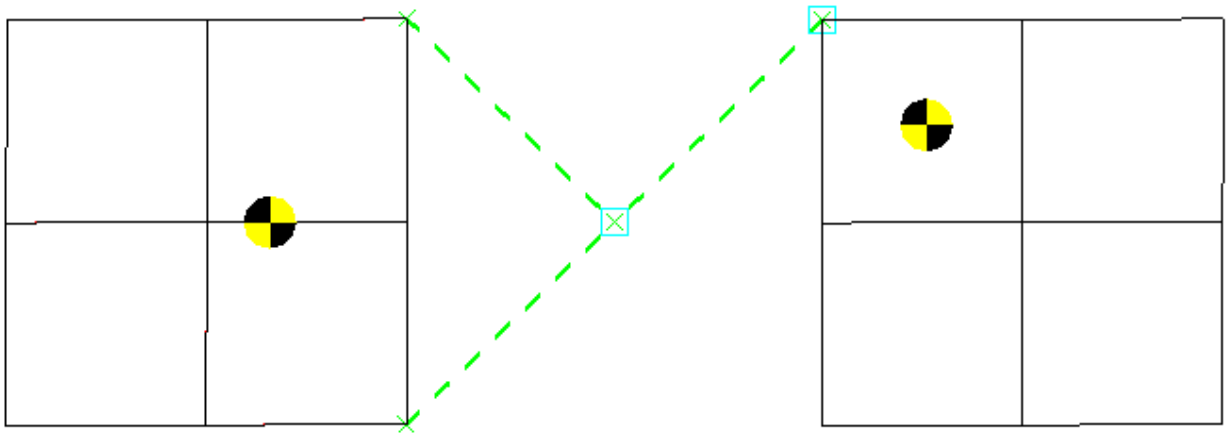




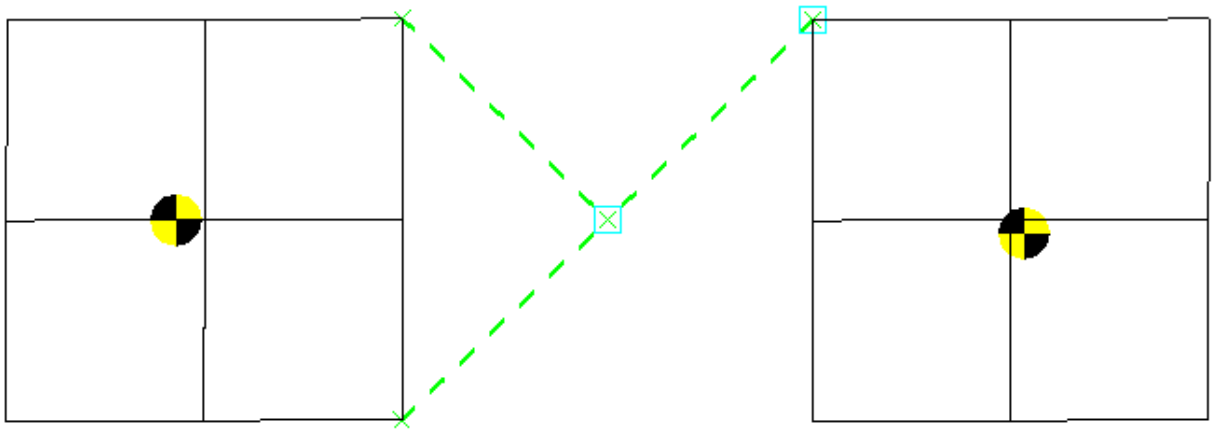
Part ID	Part Mass [0.260736]	Lumped Mass (def) [0]	NRB Mass [0.0469698]	Transferred Mass
1	0.125874	0	0.0246458	-0.0179791
2	0.134862	0	0.022324	-0.00899068

In this example NRBmass for Part 1 will include mass 2/3 of mass of MA1 and 2 quarter element shares. Mass for Part 2 will include 1/3 of mass of MA1 and all of mass MA2 and 1 quarter element share.

If mass properties (CofG and Inertia) are activated in the table as above, they will include Lumped Mass and NRB mass.



If these columns are not displayed, the calculation will ignore the mass associated with the NRB and should give the same result as reported in the LS-Dyna otf (d3hsp) file. This treatment de-couples the NRB from the deformable parts.



Added mass is the timestep added mass on deformable parts that arises due the model mass scaling (DT2MS < 0.0).The percentage added mass is the ratio of added mass to part mass.

Structural mass is the sum of the structural element masses, except for rigid Part_Inertia, where is the raw value <TM>.

Part mass is defined by Primer as follows:

For a deformable part it is the sum of the structural mass, the assigned mass and the nonstructural mass.

For a rigid part that is not _INERTIA it is the sum of structural mass, the assigned mass, the nonstructural mass and the attached lumped mass. This will include mass on constrained extra nodes. If it is a master part, the mass of its slaves will be added in. If it is a slave itself its part mass will be reported as zero.

For a rigid part_inertia it is the true LS-DYNA part inertia value, that includes adjustment for rigid body merges and constrained extra nodes which carry the inertia flag (IFLAG). If the part is slaved itself its part mass will be reported as zero.

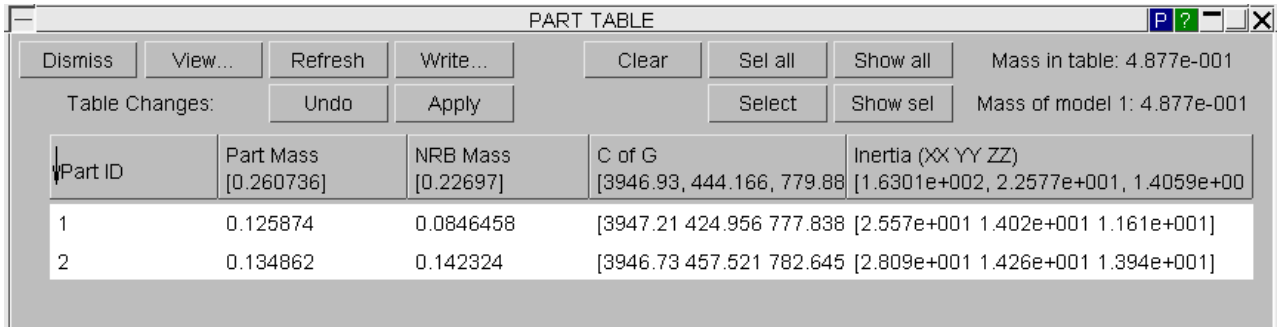
C of G and Inertia in part table

The C of G and Inertia tensor of individual parts may be displayed by using the drop down from the appropriate row.

If multiple parts are selected, the combined C of G will be displayed. These values are echoed in the dialogue box.

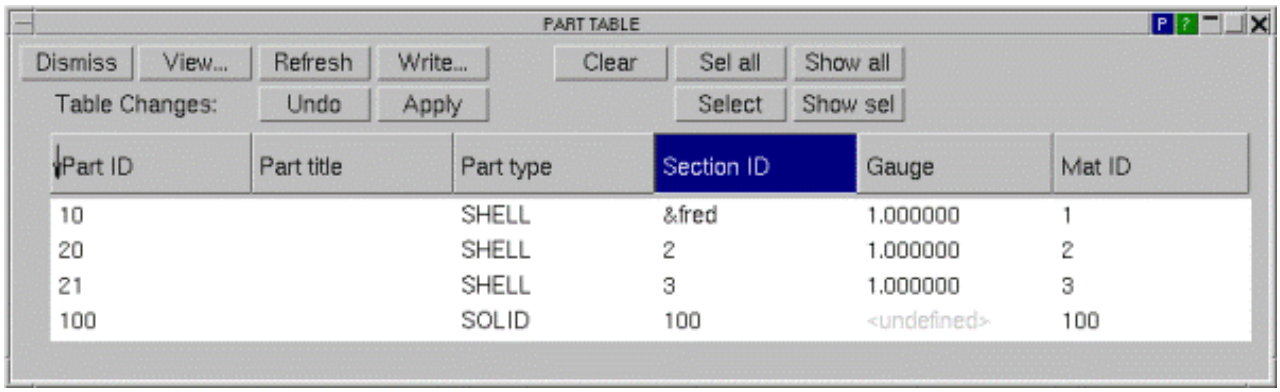
The value given on the top row is the combined C of G and combined Inertia for the parts displayed on the table.

If NRB mass/Lumped mass/Added mass columns are displayed, these masses will be included in the mass property calculations.

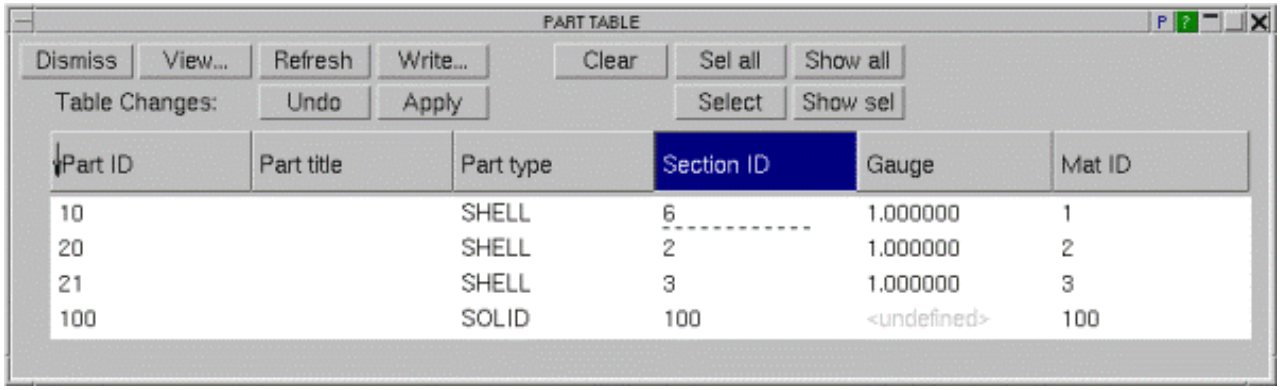


Parameters in the part table

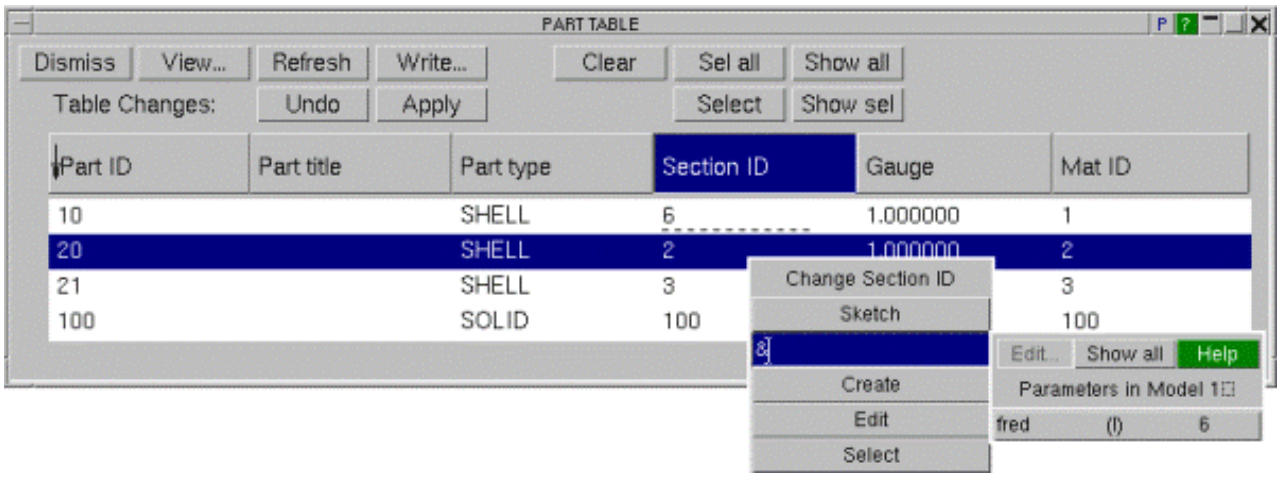
Parameters can be displayed in the part table. This works in a very similar way to how parameters are displayed in keyword edit panels. If a field contains a parameter, the parameter name is displayed with an "&" at the front. The example below shows a part table containing a part where the section ID is defined using a parameter, "fred".



The parameter value can be viewed by clicking on the "P" button in the top right of the part table. This button can be used to toggle between displaying parameters or their value. If the parameter value is displayed, then the value will be underlined to indicate it as a parameter value.

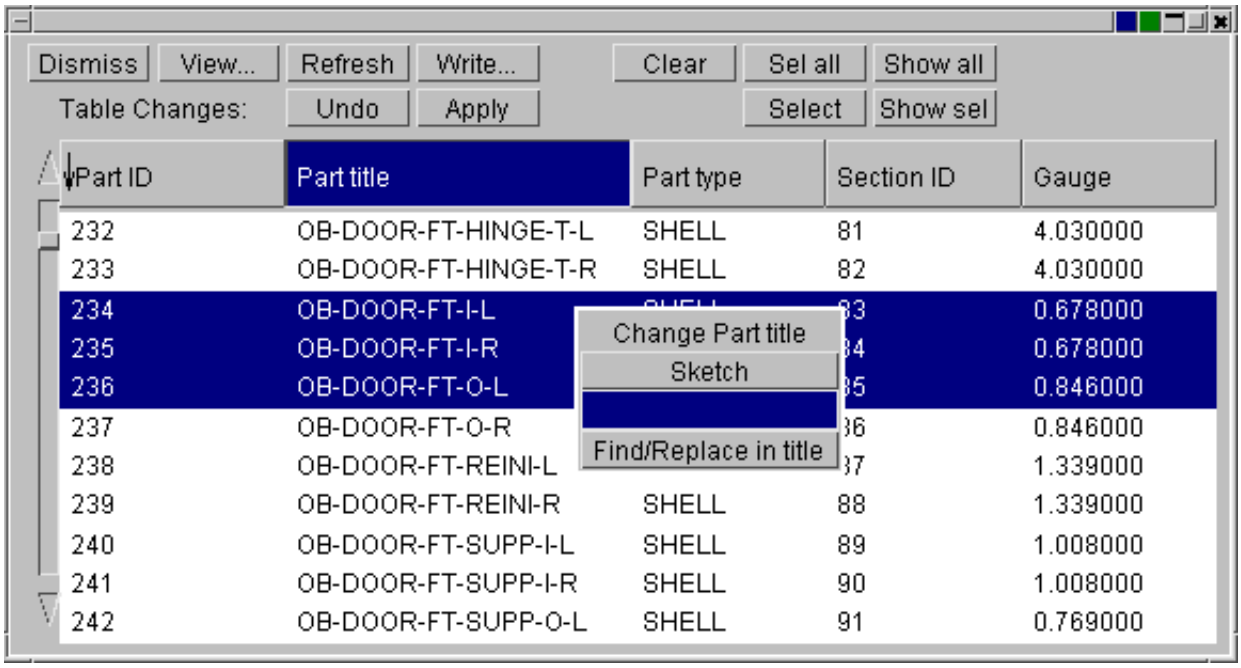


For editable fields, parameters can be added/edited by right clicking on the field and typing an "&" into the input box on the popup. This works in the same way as inputting/editing parameters in edit panels. Typing in & will also bring up a further popup for selection of parameters should they already exist in the model. See section 2.10 for more details.

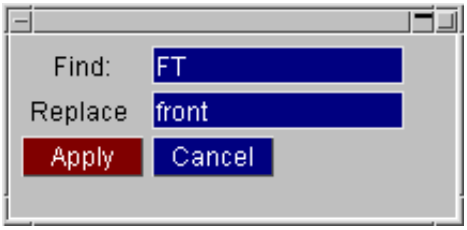


Find/Replace in part title

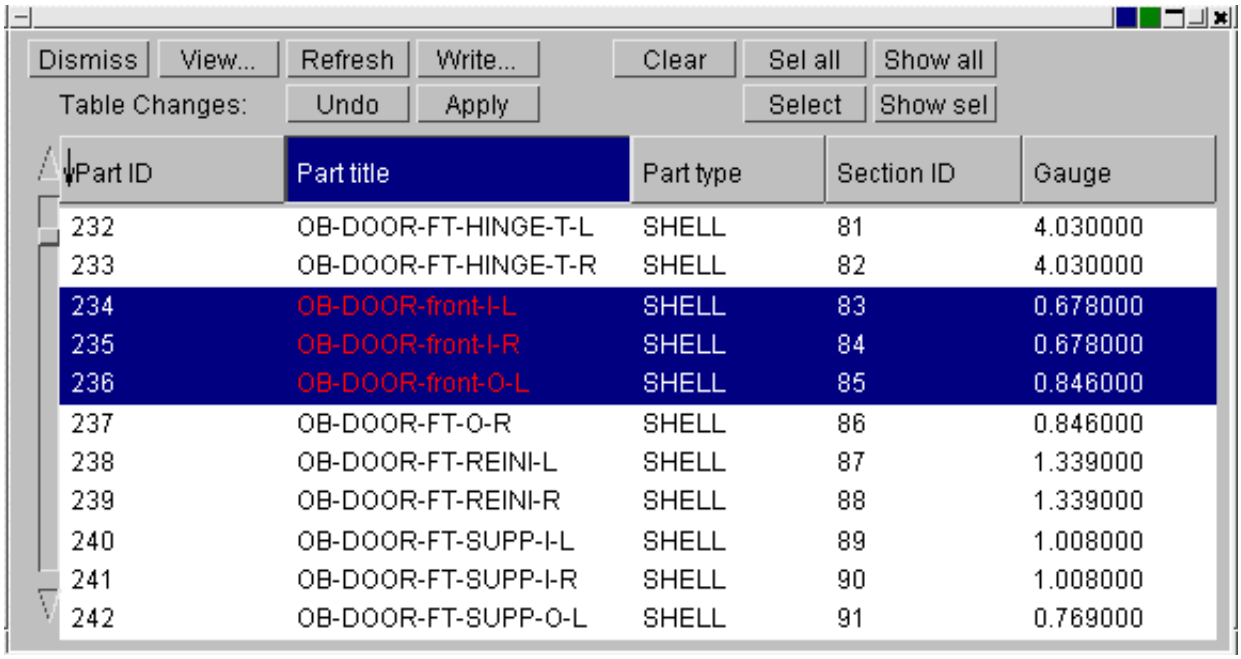
It is possible to search for particular text strings within part titles and replace the text string with another specified text string. This is done through the popup invoked by right clicking over the part title field of selected parts on the part table.



For part titles, the **Find/Replace in title** button is available. Clicking on this will open the find/replace panel.

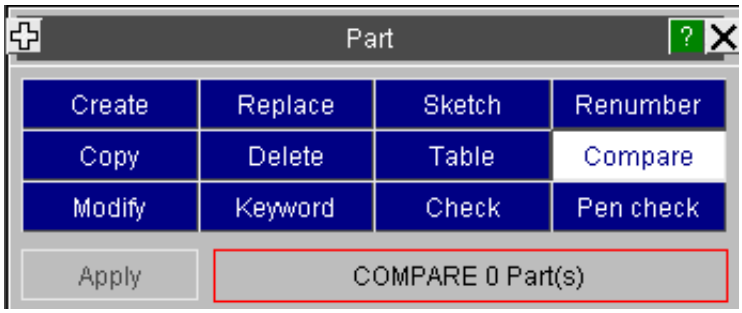


This is used to specify the text sting to search for, and also the text string to replace it with. Clicking **Apply** will apply the title modifications to the part table. Note that the search is case sensitive.

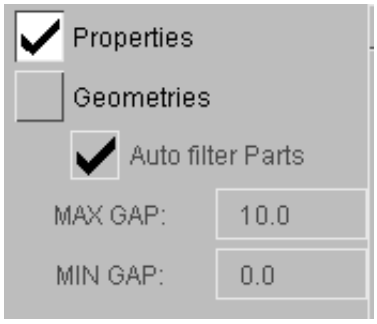


As with other changes to the data in the part table, they are shown in red and not applied to the stored part data until **Apply** is clicked.

7.3 PART COMPARE



Part compare runs in one of two modes



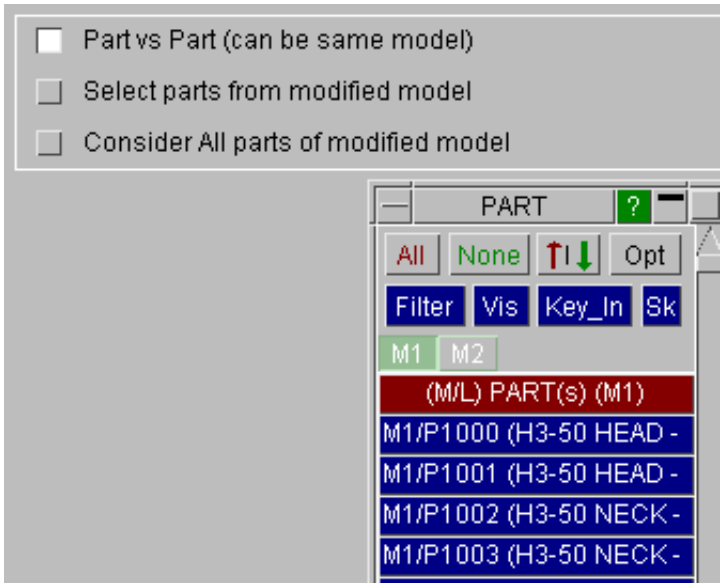
Properties uses the functionality of the part table to make a comparison between the properties of parts. Any parts which do not match for all the criteria tested will be displayed on the table.

Geometries will run a contact type check to detect gaps (using defined min/max values) between a pair of shell parts.

7.3.1 Selection of parts to compare

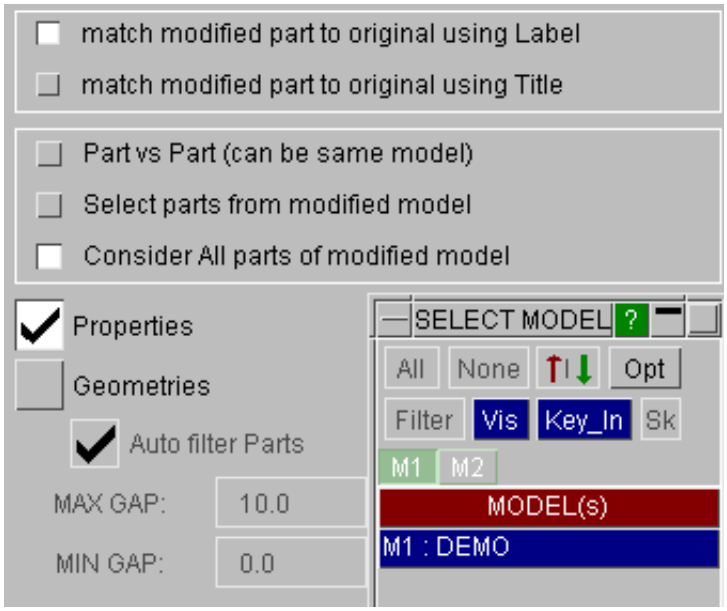
Comparing two parts

In its simplest mode the user chooses two parts top be compared directly. These may be in different models or the same model and will be compared in **Properties** mode.

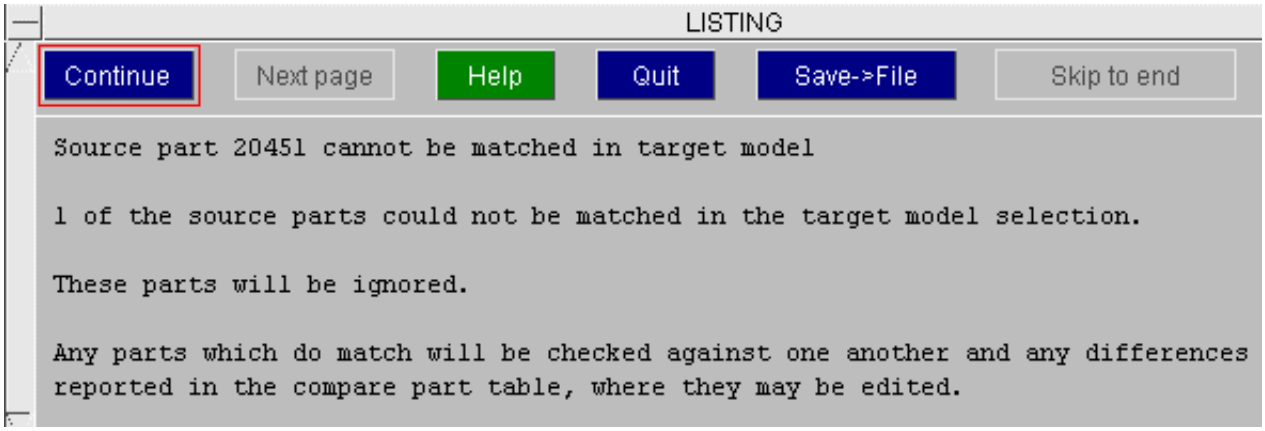


Comparing all parts in one model to another

If multiple models are in memory, the function can be used to compare multiple parts across models. In the mode **Consider all parts of modified model** the user selects the matching method (label or title) and then selects the modified model. If there are two models in memory the other model is automatically taken as the original model, if more than two, the user will need to select the original model.



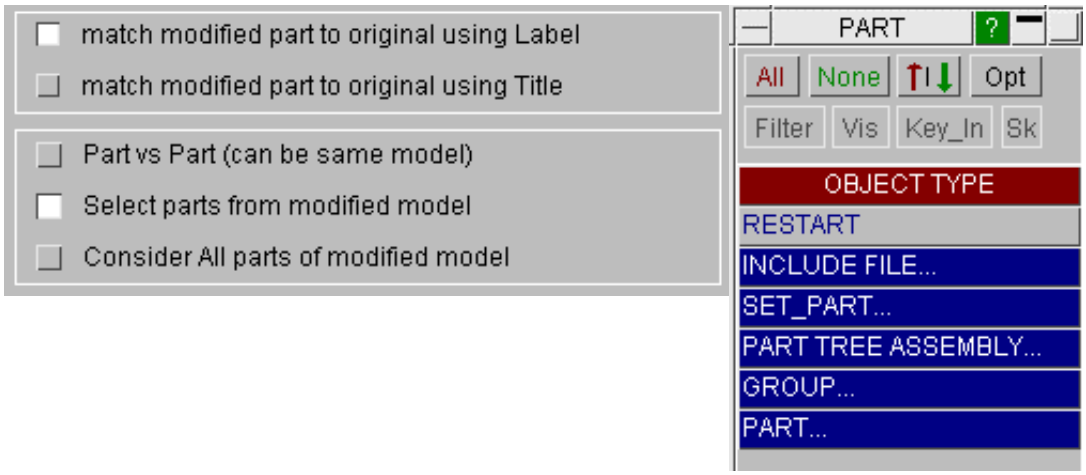
Primer then attempts to match all parts to those of the source model and if applicable will report any that failed to find a match in an information panel.



Comparing some parts in one model to another

This is similar to the above, but after selection of the modified model, another object menu will allow you to select a subset of parts in the modified model, for example by include file(s) or one or more assemblies. Once the section is complete, pressing **Next>** will start the comparison process. For large models this may be considerably quicker than comparing all parts (particulary wrt the mass calculation).

Primer will attempt to match the selected parts of the modified model to parts in the original model, using label match or title match as per the option.



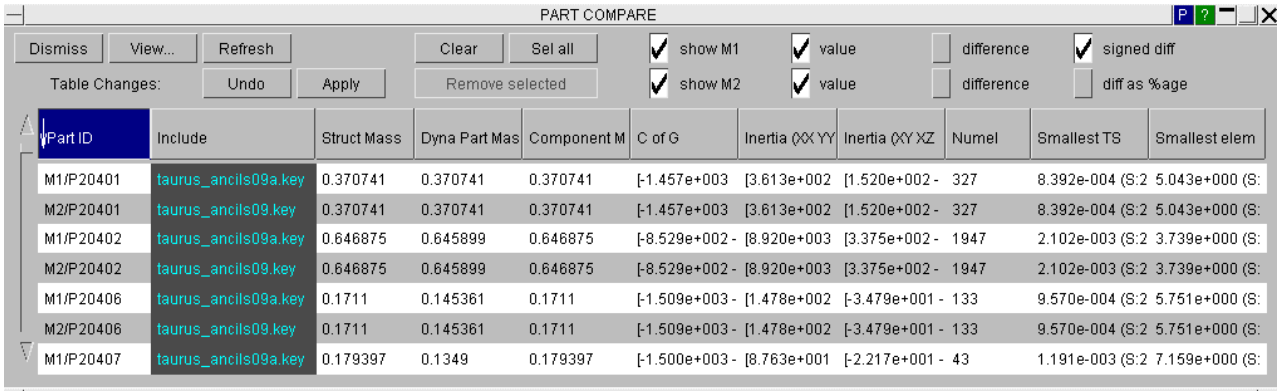
7.3.2 Display of different properties

Properties mode uses the part table.

All the matched pairs are then checked against one another for all the values that the part table treats (see section 7.2) and a part table is constructed for the part pairs which show differences.

The parts are listed in the form M1/Px, M2/Px, M1/Py, M2/Py, etc and the sorting of the Part ID column will always restore this order.

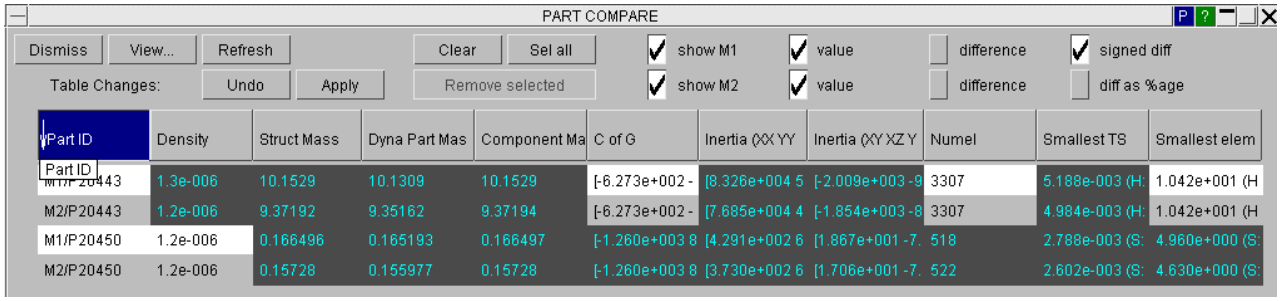
All the appropriate columns are displayed and the difference is highlighted. In this example, many parts have changed include file.



Part ID	Include	Struct Mass	Dyna Part Mas	Component M	C of G	Inertia (XX YY	Inertia (XY XZ	Numel	Smallest TS	Smallest elem
M1/P20401	taurus_ancils09a.key	0.370741	0.370741	0.370741	[-1.457e+003	[3.613e+002	[1.520e+002 -	327	8.392e-004 (S:2	5.043e+000 (S:
M2/P20401	taurus_ancils09.key	0.370741	0.370741	0.370741	[-1.457e+003	[3.613e+002	[1.520e+002 -	327	8.392e-004 (S:2	5.043e+000 (S:
M1/P20402	taurus_ancils09a.key	0.646875	0.645899	0.646875	[-8.529e+002 -	[8.920e+003	[3.375e+002 -	1947	2.102e-003 (S:2	3.739e+000 (S:
M2/P20402	taurus_ancils09.key	0.646875	0.645899	0.646875	[-8.529e+002 -	[8.920e+003	[3.375e+002 -	1947	2.102e-003 (S:2	3.739e+000 (S:
M1/P20406	taurus_ancils09a.key	0.1711	0.145361	0.1711	[-1.509e+003 -	[1.478e+002	[-3.479e+001 -	133	9.570e-004 (S:2	5.751e+000 (S:
M2/P20406	taurus_ancils09.key	0.1711	0.145361	0.1711	[-1.509e+003 -	[1.478e+002	[-3.479e+001 -	133	9.570e-004 (S:2	5.751e+000 (S:
M1/P20407	taurus_ancils09a.key	0.179397	0.1349	0.179397	[-1.500e+003 -	[8.763e+001	[-2.217e+001 -	43	1.191e-003 (S:2	7.159e+000 (S:

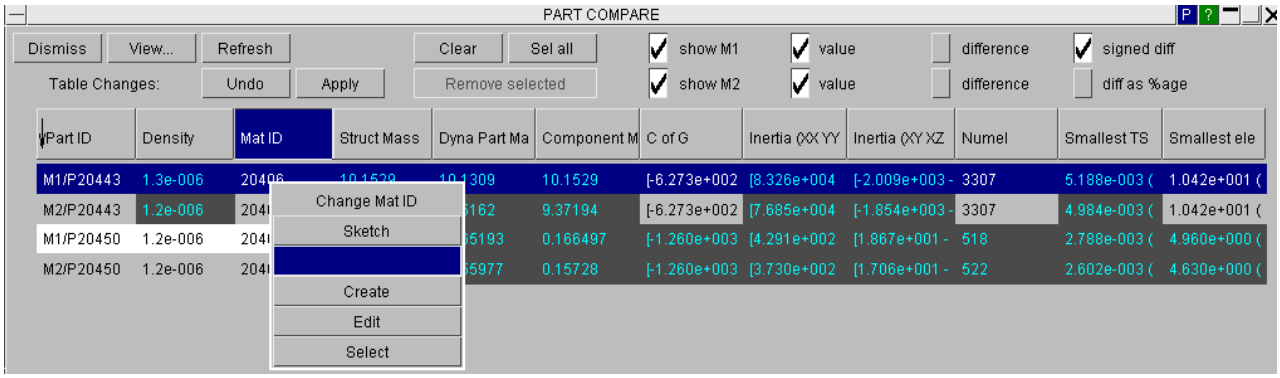
You may use shift-select to select unwanted part pairs and then apply **Remove Selected**.

If we are not interested in the include change, we can use **View..** to de-activate that column and **Refresh** to rebuild the table. Any part pairs from which the **only** diference is their include are now removed from the data stacks.

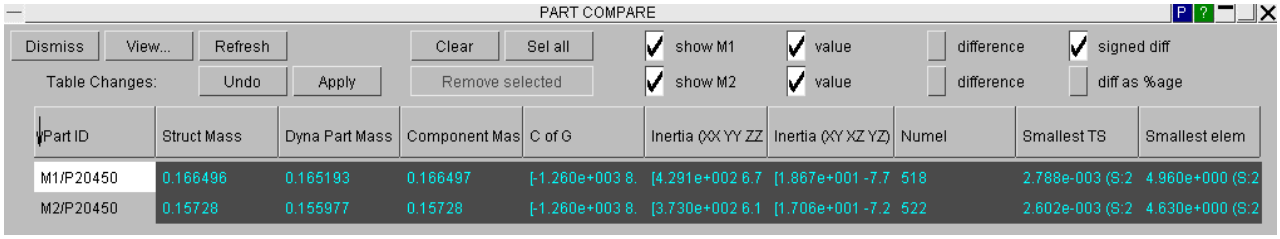


Part ID	Density	Struct Mass	Dyna Part Mas	Component Ma	C of G	Inertia (XX YY	Inertia (XY XZ Y	Numel	Smallest TS	Smallest elem
M1/P20443	1.3e-006	10.1529	10.1309	10.1529	[-6.273e+002 -	[8.326e+004 5	[-2.009e+003 -9	3307	5.188e-003 (H:	1.042e+001 (H
M2/P20443	1.2e-006	9.37192	9.35162	9.37194	[-6.273e+002 -	[7.685e+004 4	[-1.854e+003 -8	3307	4.984e-003 (H:	1.042e+001 (H
M1/P20450	1.2e-006	0.166496	0.165193	0.166497	[-1.260e+003 8	[4.291e+002 6	[1.867e+001 -7.	518	2.788e-003 (S:	4.960e+000 (S:
M2/P20450	1.2e-006	0.15728	0.155977	0.15728	[-1.260e+003 8	[3.730e+002 6	[1.706e+001 -7.	522	2.602e-003 (S:	4.630e+000 (S:

We may then use the part table functionality to investigate and edit data as appropriate. For example, by using **View...** to activate display of **Mat ID** and editing the material for M1/P20443 to correct the density from 1.3e-6 to 1.2e-6.

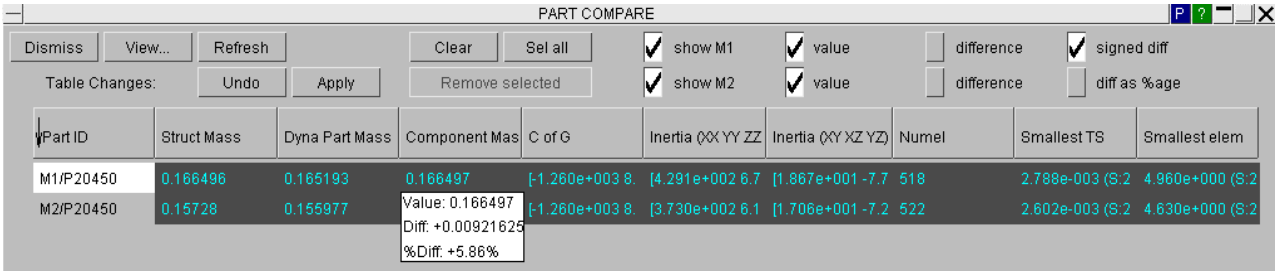


Refresh of the table will then remove part 20443 as the data is consistent across models

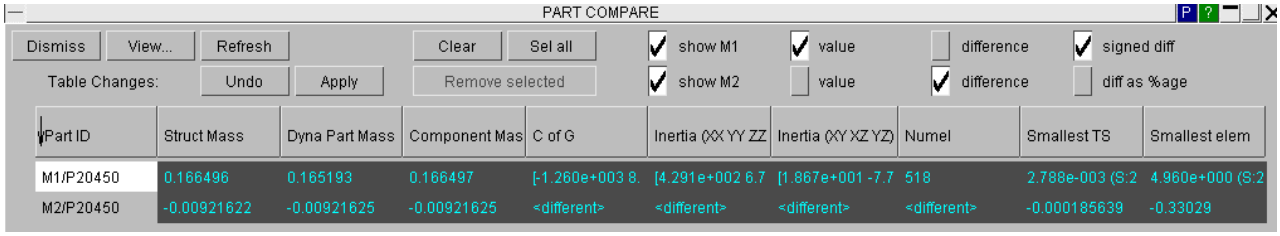


Showing the difference

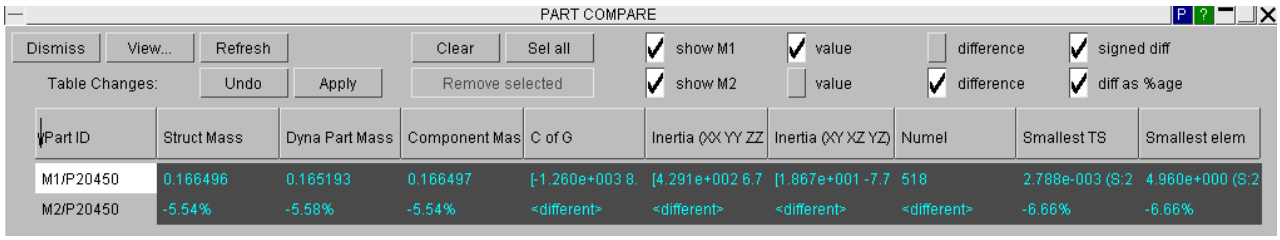
By default the table shows values with hover text to show the absolute and percentage differences wrt the other value.



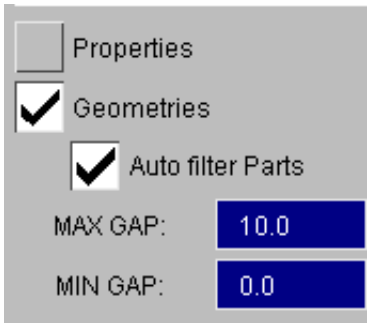
By activating the difference switch you may show the absolute difference for floating point numbers. For other types the string <different> will be written.



The difference for floating point numbers may also be usefully expressed as a percentage.

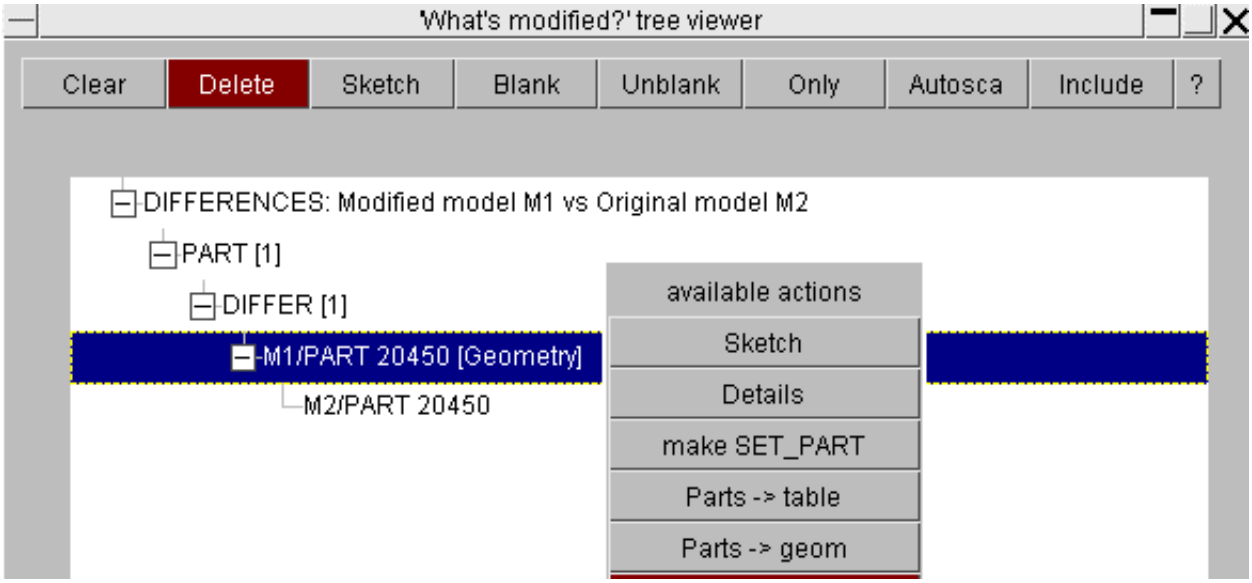


7.3.3 Display of different geometries

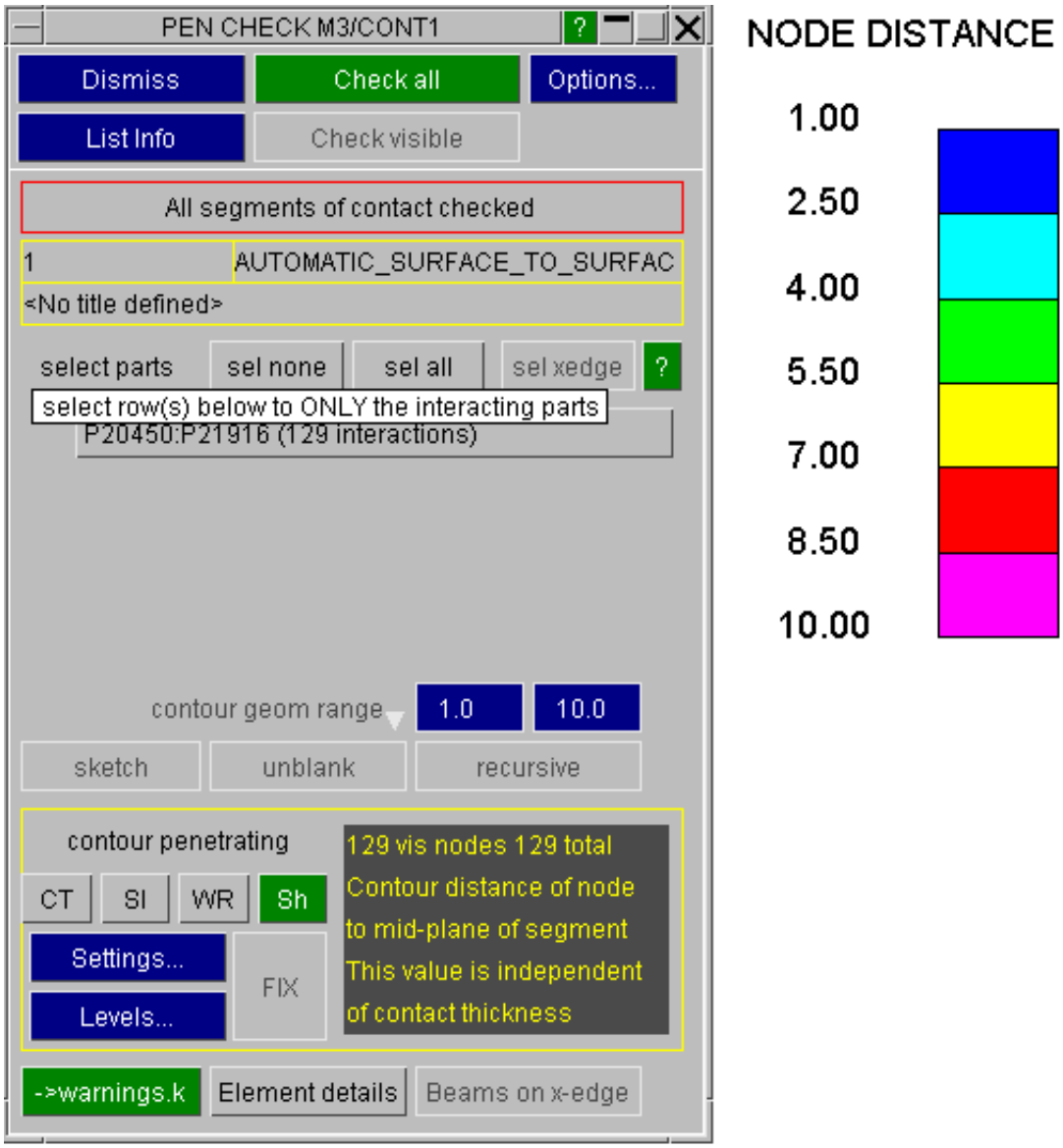


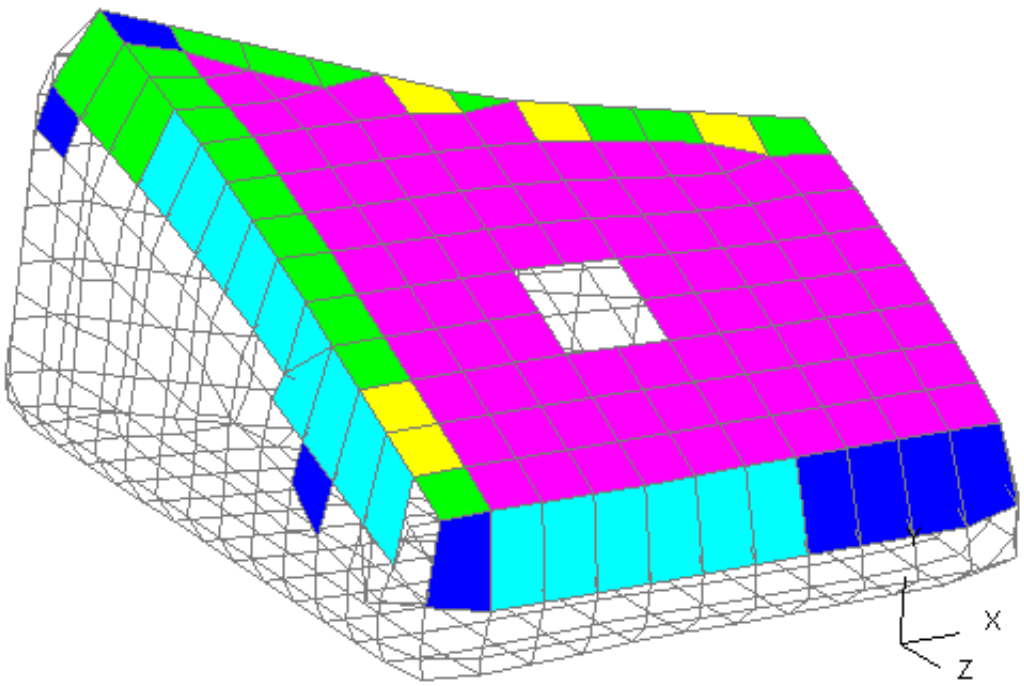
In **Geometry** mode, matched part pairs will be compared to one another using a contact type check which will detect gaps within user defined min/max values (default 0-10mm). If many parts are being checked, the option **Auto filter parts** is recommended to block the test (which can take a few secs per contact check) for part pairs which are unlikely to be geometrically different (i.e. they have the same element count, same geometric CofG and same surface area).

Any part pairs found to be geometrically different are sent to the 'What's modified?' tree viewer where they can be investigated in detail by using **Parts -> geom.**

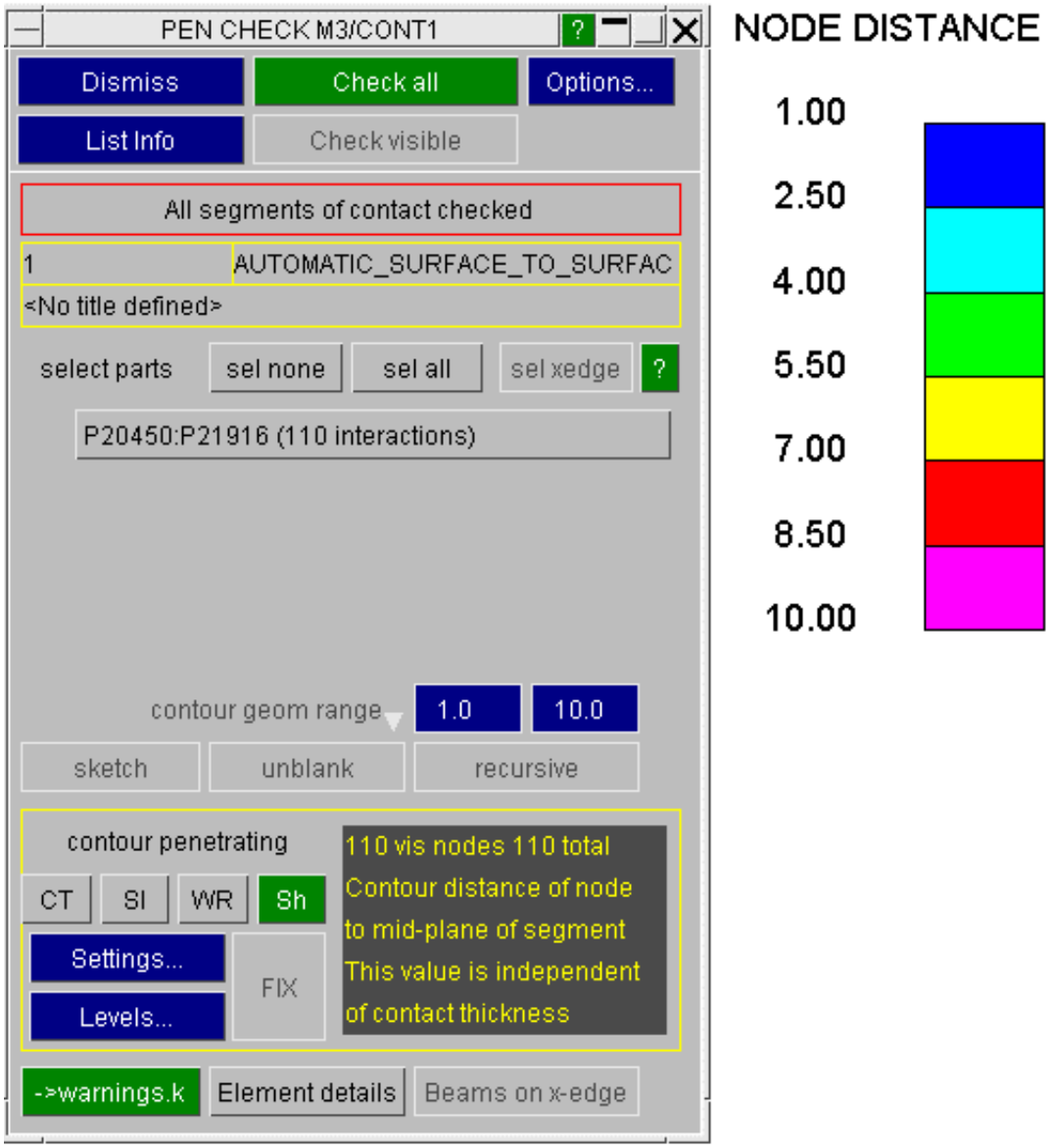


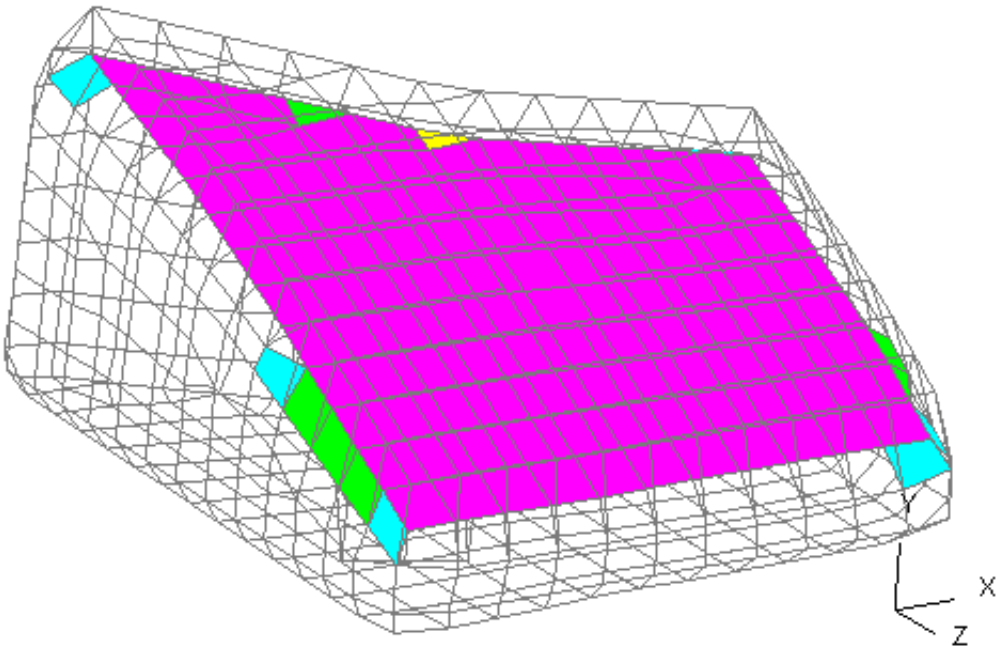
This will copy the parts into a separate model and create a surface to surface contact between them. This can be used to contour the distance away of the nodes of one part to the segments of the other by using **CT** button.


PEN CHECK M3/CONT1
Dismiss
Check all
Options...
List Info
Check visible
All segments of contact checked
1
AUTOMATIC_SURFACE_TO_SURFAC
<No title defined>
select parts
sel none
sel all
sel xedge
?
select row(s) below to ONLY the interacting parts
P20450:P21916 (129 interactions)
contour geom range
1.0
10.0
sketch
unblank
recursive
contour penetrating
CT
SI
WR
Sh
Settings...
Levels...
FIX
129 vis nodes 129 total
Contour distance of node
to mid-plane of segment
This value is independent
of contact thickness
->warnings.k
Element details
Beams on x-edge
NODE DISTANCE
1.00
2.50
4.00
5.50
7.00
8.50
10.00


X
Z

Observe that the contact is deliberately reversed if **Parts -> Geom** is selected for M2.





Which view is more informative will depend on the geometries involved.

Because this is a rather non-standard use of the contact checker, the user is restricted to operations within this panel until Dismiss is pressed and he is returned to the modified tree. Unfortunately this inhibits all functions in the View box, so a shaded image draw button **Sh** has been added to the check panel. Dynamic viewing operations (rotate and zoom) are not affected by the restriction.

8 Images

Primer can read images in as the background or watermark.

It can also capture graphics and copy to file in four ways:

- By Laser plotting, generating a Postscript file, using the **POSTSCRIPT file** command. Sections 8.0 to 8.4 below
- By a screen grab, generating Bitmap or JPEG files, using the **Image file (.jpg etc)** command, see Section 8.5
- By generating a 3D PDF file, using the **3D PDF** command, see Section 8.6
- By generating a WebGL file (viewable in a web browser). See section 8.7.

These commands are found under the **IMAGES** button

8.0 LASER: Introduction to laser plotting

8.1 Using the Laser Control panel

8.2 Changing paper size and margins

8.3 Creating Encapsulated Postscript (EPS) files

8.4 Notes on laser plotting

8.5 Raster images

8.6 3D PDF

8.7 WebGL

8.8 Read background image and watermark

8.0 LASER: Introduction to Laser Plotting

By default all graphics images generated by PRIMER are sent only to the screen, but you can choose to copy them to laser files (postscript and pdf files for a laser printer).

This is done by pressing the "Plot" button when you are in Postscripts/PDF.

8.0.1 Laser language and file format used.

At present PRIMER writes Postscript laser files, using PS ADOBE level 2.0 commands, and PDF files. These are ASCII files that can be viewed and edited using any common editor.

"Encapsulated" Postscript files are not written, but later in Section 8.3 the very simple edits required to convert a file to encapsulated form are given.

Laser output is switchable between A4 (297 x 210mm), A3 (296 x 420mm) and US "letter" (11" x 8.5") paper sizes. The Postscript language makes it easy to edit files to fit other sizes.

The laser driver defaults to "PDF" file format, you can opt for "Postscript" laser file.

8.0.2 Number and orientation of plots on a page.

The laser driver defaults to "landscape" orientation, with one plot per page. You can opt for "portrait" orientation and, in both cases, put multiple plots on a page in a variety of layouts.

8.0.3 Resolution setting.

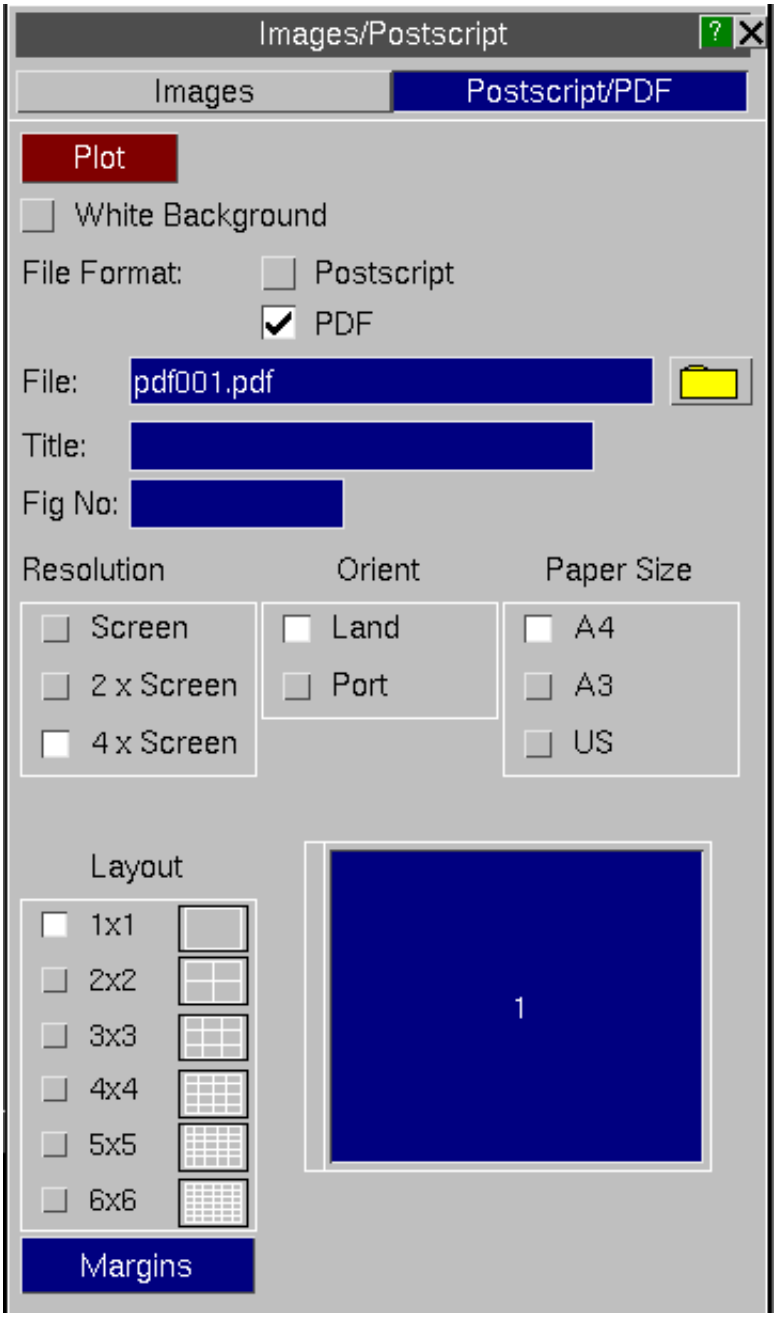
Postscript and pdf files generated can have higher than current screen resolution. Screen resolution, twice and four times screen resolutions are all available.

8.1 Controlling laser plotting using the Laser Plotting panel

This figure shows the basic laser plotting panel.

This is invoked by the Postscript/pdf command under Images->Write in the top menu box.

It both controls and shows the status of the current laser file (if any).



8.1.1 Plot button

Press "Plot" button when you want to plot the current view on the screen. With the White Background option switched on, images will be plotted with a white background. Entity labels and screen text will be switched to black. Once the image has been captured the screen will return to its original colours.

Any plot directed to laser file is sent by default to the next free sub-image (if the file has multiple plots per page), or file (if only a single image per file, or the multiple page is full).

When multiple sub-images in a file are in use the next image to be written is shown by depressing the appropriate icon in the file layout panel. You can override this and choose a different sub-image: see Section 8.1.5 below.

8.1.2 Choosing the laser filename

When no file is currently in use the **File:** entry box will be available. You can give any valid filename for the next laser file to be written, or let PRIMER choose one for you. You can also use the button to select a file via the standard file filter box.

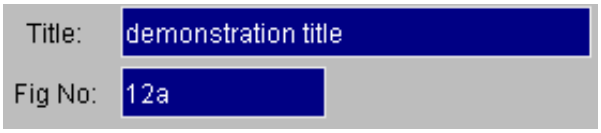
If the file already exists you will be queried to check that you genuinely want to overwrite it: you cannot append to existing laser files.

The default naming convention used by PRIMER for postscript laser files is **`postNNN.ps`**, where:

> **`NNN`** is a 3 digit number (with leading zeros if required) in the range `001 - 999`.

Any existing files are skipped when the next file in the sequence is computed.

8.1.3 Defining a label and figure number for laser plots.



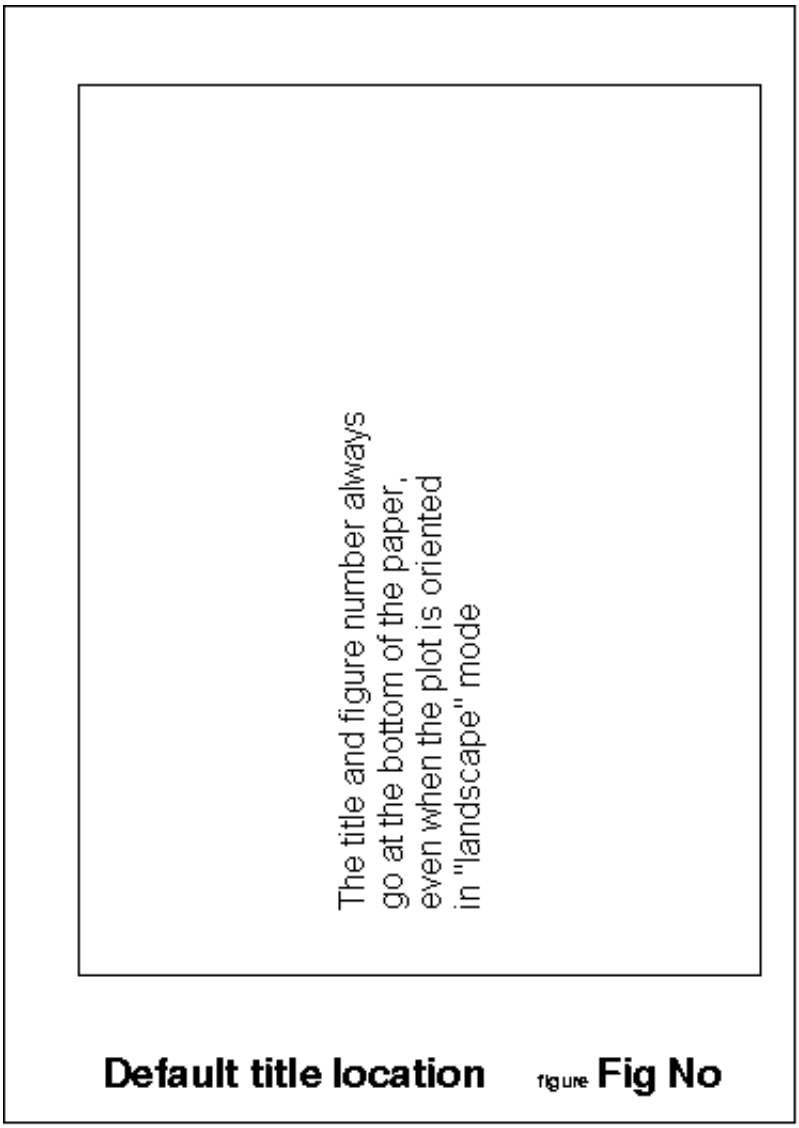
By default laser files are not labelled and have no figure number, but you may add either or both of these. They are always put at the bottom of the page, along the short edge, regardless of the orientation used for plots.

This figure shows the standard locations for title and figure number on laser plots.

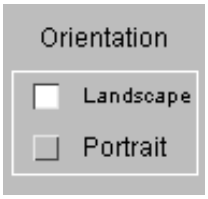
The title may be up to 80 characters long, and is split over two lines if necessary by PRIMER.

The figure number may be any string (not just a number), and is preceded by the word "figure". It is suggested that it is 6 characters or less long: here "12a" was used.

This plot is written in "landscape" format, and reinforces the point that the title and figure number always go at the bottom of the paper, regardless of the orientation of the plot contents.

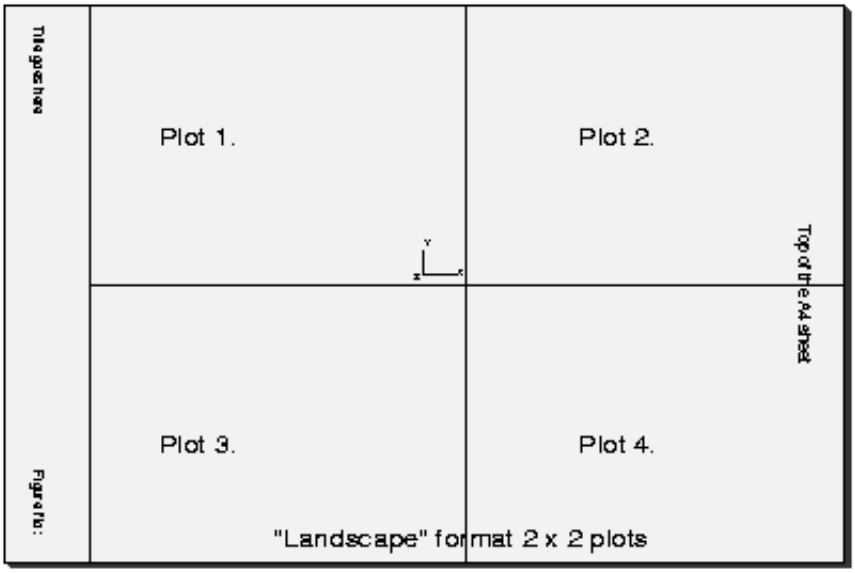


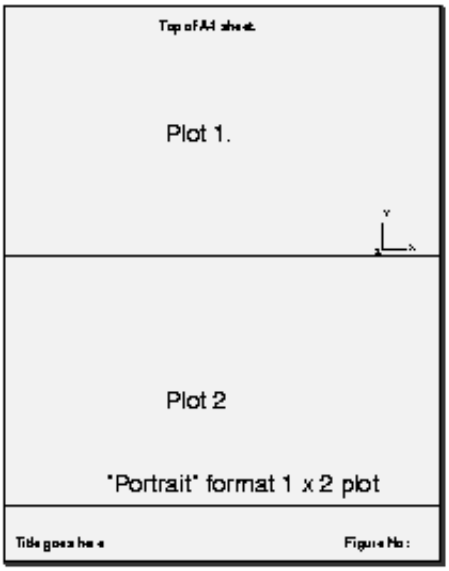
8.1.4 **Orientation** Setting Landscape or Portrait plot orientation.



By default plots are in "Landscape" orientation, with the long side of the plot aligned with the long side of the paper, but you can choose "Portrait" format instead.

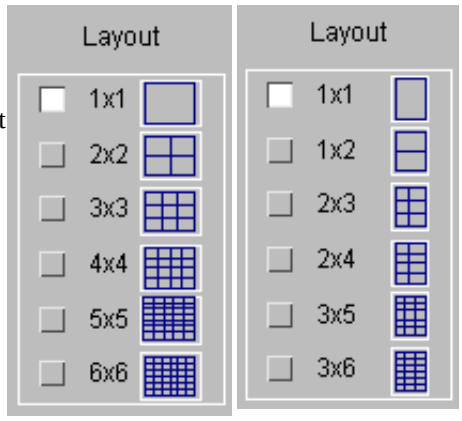
The figure below shows examples of both landscape and portrait format plots, showing how they are aligned on the paper.





This example shows examples of Landscape and Portrait plots, showing how they are oriented on the paper.

8.1.5 Layout



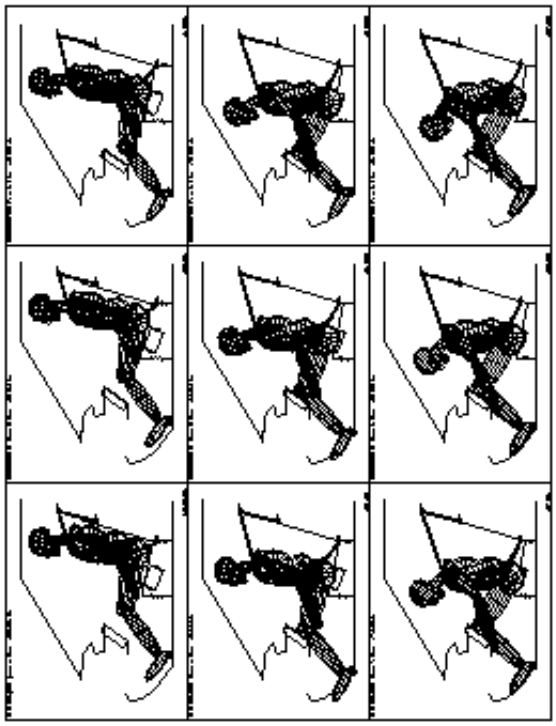
LANDSCAPE PORTRAIT

In both landscape and portrait formats it is possible to have more than one plot on a page.

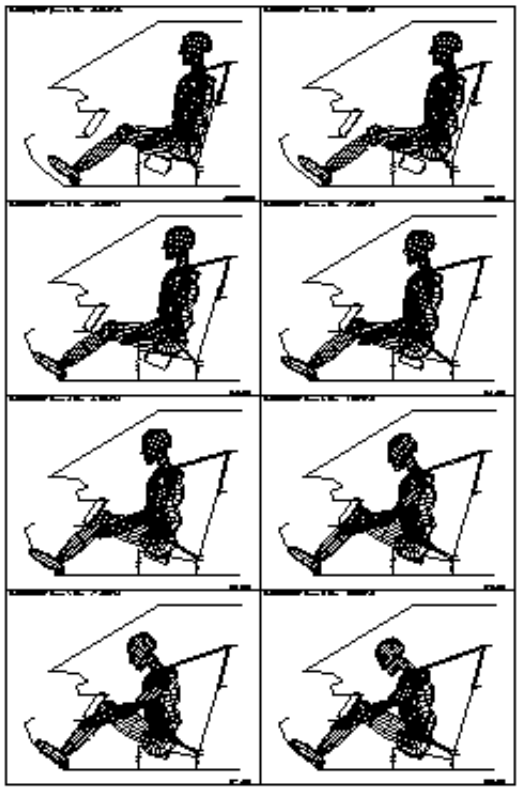
Various pre-programmed permutations of <#x> x <#y> plots are available as shown here.

Each individual plot on a page will be referred to from now as a "sub-image".

The figures below show examples of 3x3 Landscape and 2x4 Portrait multiple plots.



EXAMPLE OF 3 x 3 LANDSCAPE OUTPUT Figure 7.1.5a



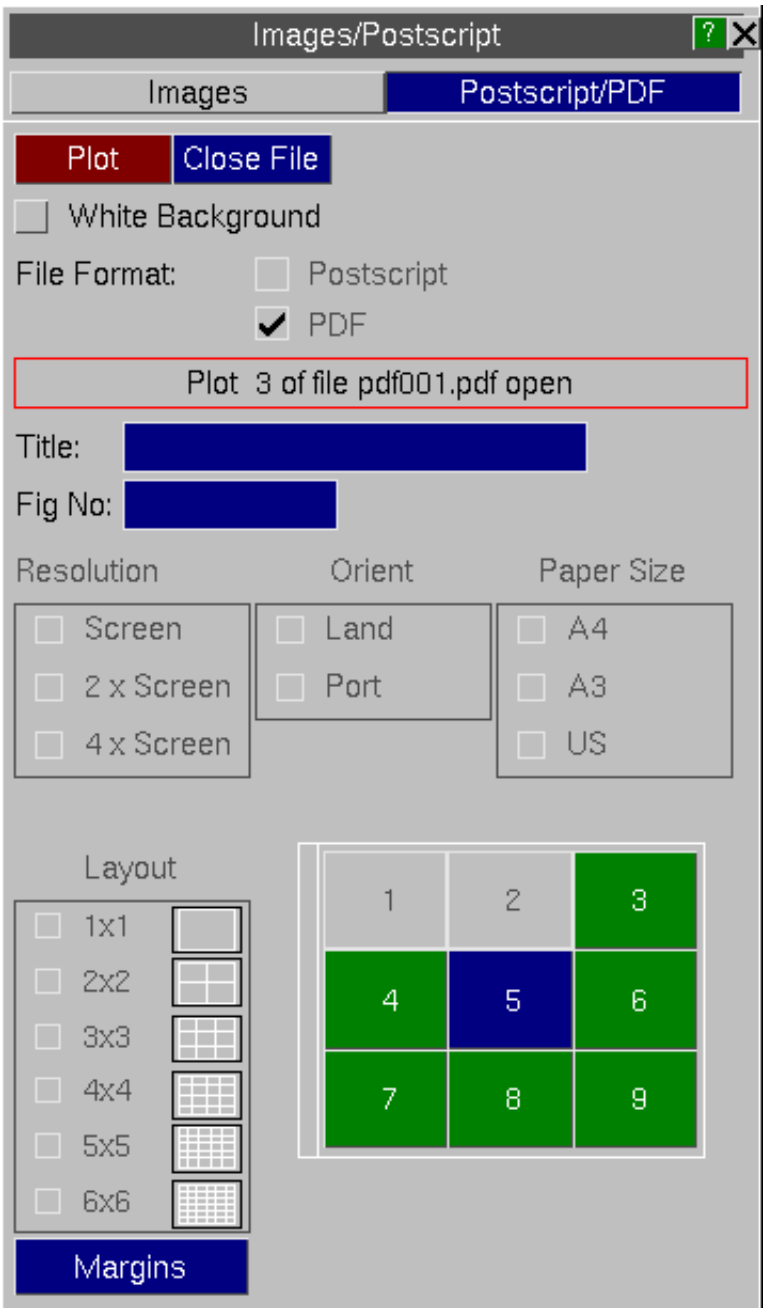
EXAMPLE OF 2 x 4 PORTRAIT OUTPUT Figure 7.1.5b

Controlling the order in which multiple plots are drawn.

The right hand menu shows a typical laser panel for a 3x3 portrait plot in which sub-images 1 and 2 are complete.

Normally sub-images are written in the order #1 to #n, but if the user wanted the next plot to be drawn to sub-image #5 instead of #3, he would click on the [5] icon where the button gets coloured in blue instead of the [3] icon as it normally would.

Next sub-image would be the next free one, i.e. #3 to receive the next plot. The [3] icon will be coloured in blue.



The status of files, and sub-images within files.

PRIMER laser files, and sub-images within files, have one of four possible states.

Inactive	**Green**	No graphics written yet, and not selected for the next plot.
Selected	**Blue**	No graphics written yet, but selected to receive the next plot.
Closed	Greyed out	File/sub-image complete, and cannot receive any more information.

The colours referred to above are used for the button icons on multiple sub-image panels, as shown in the figure above. Only green icons (ie those which are currently inactive) may be selected to receive the next image.

How sub-image status affects the destination of graphics.

(1) If no graphics have been written to a sub-image then the next plotting command will send laser output to the the sub-image currently "selected".

By default this will be the lowest numbered sub-image that has not yet been written to, but you can choose another as described above.

(2) Once graphics have been sent to the sub-image its status changes to "closed". This means that it cannot receive further graphics.

Interaction between sub-images and files

A file with only a single image in it is treated in exactly the same way as an individual sub-image above, except that it is (implicitly) always "selected" for plotting until something is drawn in it.

A file with sub-images remains current (ie open) until all of the sub-images in it have been "closed", or the user closes it prematurely with a **CLOSE FILE** command. Then PRIMER defaults to the next default filename as defined in Section 8.1.2 above.

The importance of closing files.

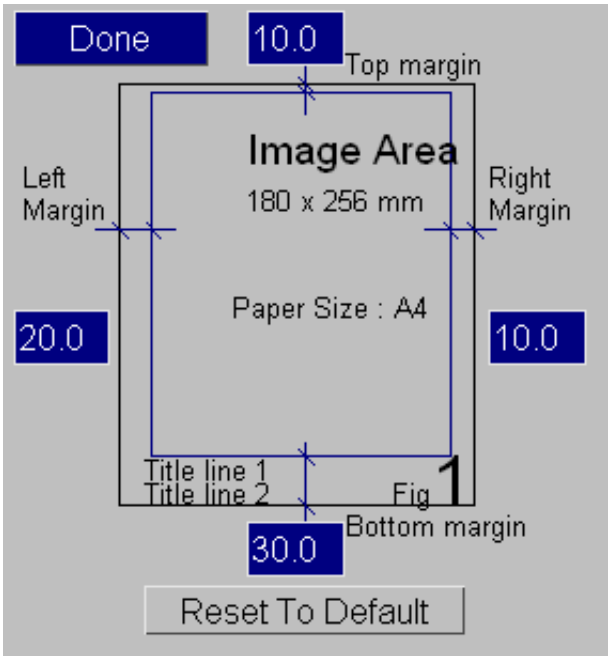
While a file is still current it is still connected to the programme, and at least some of its contents will still be held in system buffers. If you want to send it to a printer you **must** close it first using a **CLOSE FILE** command.

This flushes any remaining data to disk and disconnects the file from the programme.

8.2 MARGINS... Modifying laser paper size on the page.

The **MARGINS** button in the laser control panel gives a special sub-menu that allows you to select the margins on all sides:

The margins will only apply to the axis of the plot that comes closest to the paper borders; the other axis margins will be overridden to maintain the correct aspect ratio of plots (ie no image distortion).



8.3 Creating Encapsulated Postscript (EPS) files.

EPS format is used by many software packages to import postscript images. The laser files written by PRIMER are not in EPS format, but only two very simple edits at the top of the file are required to change this.

The first seven lines of any PRIMER laser file look like this:

To convert it to EPS format you must add a "**%%BoundingBox:**" line, and delete the "**statusdict**" line. Thus this file becomes:

```
%!PS-Adobe-2.0
%%EndComments
%%Pages: 1
%%Page: 1 1

statusdict begin
/altest save def
```

```
%!PS-Adobe-2.0
%%BoundingBox: 0 0 595 842
%%EndComments
%%Pages: 1
%%Page: 1 1

/altest save def
```

The arguments of the "BoundingBox" line are the Postscript coordinates:

<lower left> <lower right> <upper left> <upper right>

These must be expressed in raw Postscript space of 72 points per inch, and they assume that the paper is in portrait format with its origin at its lower left corner.

The values in the example above refer to A4 format: 210 x 297 mm = 595 x 842 points; US "letter" paper would give 8.5" x 11" = 612 x 792 points. Clearly a smaller bounding box would select only a subset of the image.

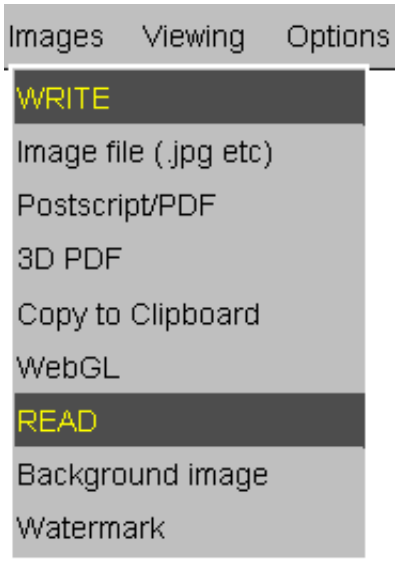
For more information on encapsulated postscript see the "PostScript Language Reference Manual, 2nd edition" by Adobe Systems Incorporated. (Published by Addison Wesley, ISBN 0-201-18127-4)

8.4 Notes on laser plotting

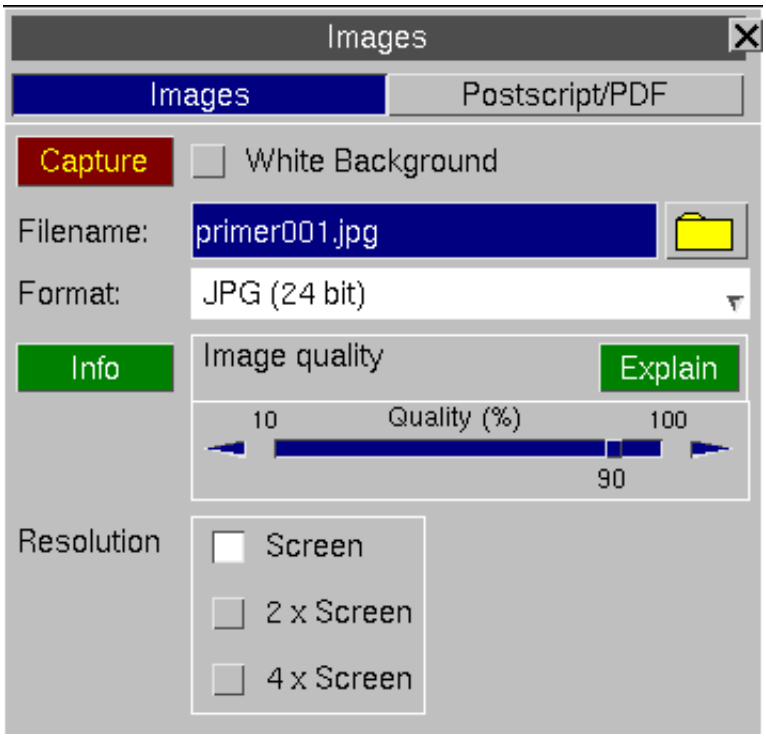
- Users on 3D devices should note that turning the laser on will temporarily force the graphics mode back to 2D. This is because a laser plot is intrinsically a 2D image and is computed in software.
- Transient graphics added "dynamically" to the screen are never copied to laser files. Examples are cursor-pick symbols, and also the information added interactively with the **DYNAMIC_LABEL** function.
- If an attempt to open a laser file fails because the file/directory refuses "write" permission, or the disk is full, you are warned and laser output is switched off.
- You can switch laser output **off** and **on** at will in the course of assembling a file with multiple images. Sub-images will only be written when the laser is on.
- Some of the defaults here may be preset outside PRIMER via preferences in the **`.oa_pref`** file: see Appendix II.

8.5 Raster Images

The function is invoked by selecting **Image file** from the **Images** option in the main menu.

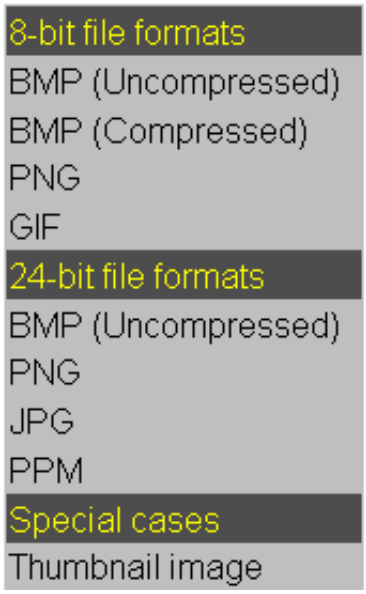


8.5.1 CAPTURE: Copying the current image to file



Captures a single static frame in one of the following formats:

With the White Background option switched on, images will be captured with a white background. Entity labels and screen text will be switched to black. Once the image has been captured the screen will return to its original colours.



8-bit file formats

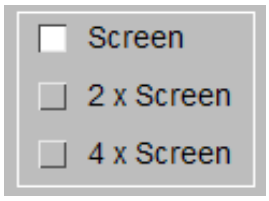
BMP Uncompressed	Uncompressed 8 bit Microsoft windows bitmap. The approximate size of the file is [image width * image height] bytes.
BMP Compressed	8 bit runlength encoded (RLE) Microsoft windows bitmap.
PNG	8 bit lossless compressed **P**ortable **N**etwork **G**raphics image.
GIF	8 bit lossless compressed **G**raphics **I**nterchange **F**ormat.

24-bit file formats

BMP	Uncompressed 24 bit Microsoft windows bitmap. The approximate size of the file (in bytes) is file size = 3 * image width *image height
PNG	24 bit lossless compressed **P**ortable **N**etwork **G**raphics image. PNG offers the similar degree of compression as GIF but has better colour quality.
JPG	Joint Photographic Experts Group compressed format. This gives image quality nearly comparable to 24 bit-plane bitmaps, but with a file of < 5% the equivalent size. JPEG format is supported by all common visualisation packages and is recommended for all applications unless image quality is of paramount importance.
PPM	24 bit uncompressed **P**ortable **P**ix**M**ap. The approximate size of the file is [3 * image width *image height] bytes.

Various `.bmp` formats are available, and there are Controls for the dithering of the 8 bit-plane variants and palette optimisation .

RESOLUTION



All images can be output at either the screen resolution or at a resolution of either 2 or 4 times the screen resolution.

8.5.2 Controls on the quality of 8 bit-plane bitmap files.

24 bit BMP files tend to be huge, and the space saved by using the compressed 8 bit format is attractive. The trouble is that without further processing the image quality obtained when 24 bit images are truncated to 8 bits is definitely not!

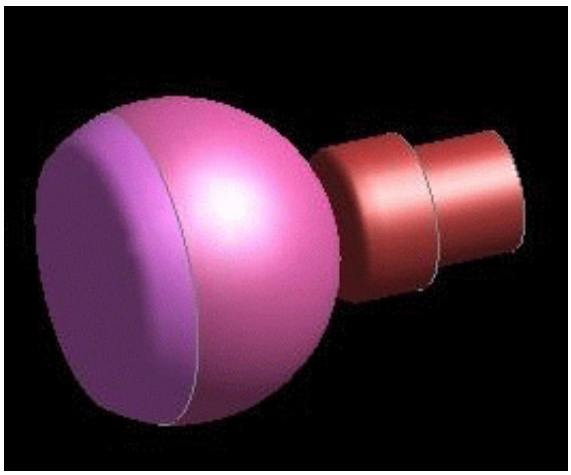
The problem is the number of colours available in the 8 bit format is 2**8 = 256, and this gives rise to "banding" when the least significant bits of the original colour definitions are lost as the original 16 million colours are truncated.

Dither image

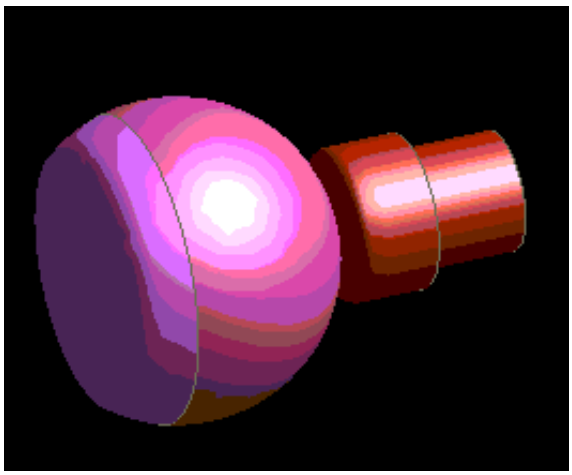
The following sequence of images show how the different levels of dithering affect the quality and file size of a compressed 8 bit image, comparing it with the JPEG equivalent.

For static images there is no advantage in using BMP files over JPEGs: they are larger and of inferior quality. However if you are unable to use MPEG animation, and have to revert to AVI format, the various permutations of image quality and filesize below will be of interest when trying to obtain the best compromise between image quality and overall file size.

The ideal would be an AVI file composed of JPEGs, or MJPEG format. Sadly all such formats are proprietary and, perhaps as a consequence, are not supported by the typical players currently available. Hopefully this will change in the future.



Here is the original 24 bit-plane image, saved as a JPEG file. **Size 5.1kB**



This is the undithered equivalent bitmap image. **Size 7.3kB**.

Note how the discretising affect of mapping onto a limited colour palette has caused "banding" which makes the image almost unusable. However at least the files are small!

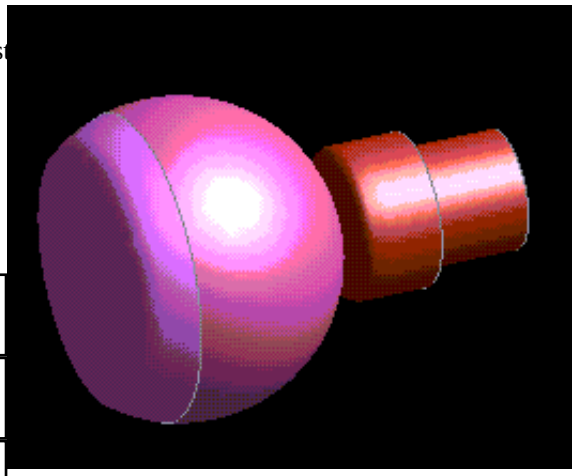
"Dithering" is a technique in which an ordered pattern of noise, **xxxxx** in the table below, is added to the least significant bits of a colour value to make it alternate between two adjacent shades.

Consider the bits for a single colour in this image that have been truncated from 24 bits (8 bits each of Red, Green and Blue). Truncated bits are shown in lower case.

Original Red byte 001?????	truncates to	001ooooo
Adding the dither pattern oooxxxxx to the bottom 5 bits of the original byte gives		
001????? + oooxxxxx	giving either or	001ooooo 010ooooo

The result may be truncated to 001ooooo, or the increased to the next shade up 010ooooo, depending on the trailing bits ????? and the noise value **xxxxx** at that pixel.

The effect is to produce a composite shade that is somewhere between the two originals.



This is the "Shifted 2x2" dithered bitmap image. **Size 19kB**.

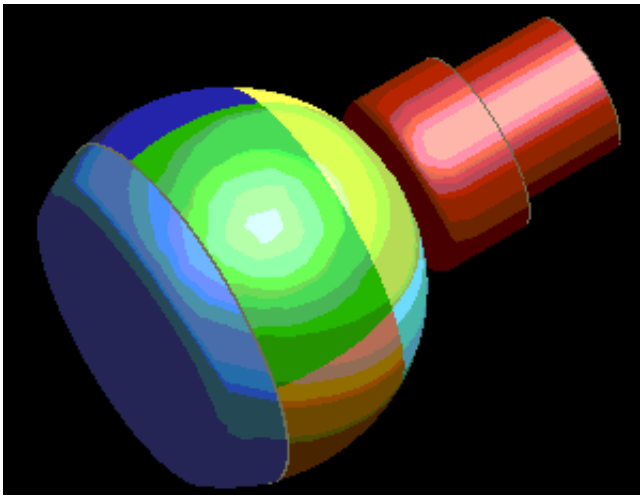
Some reduction in banding is evident, but the quality is still poor and not worth the saving in file size over normal 2x2 unless you are desperate for space.

Palette Optimization

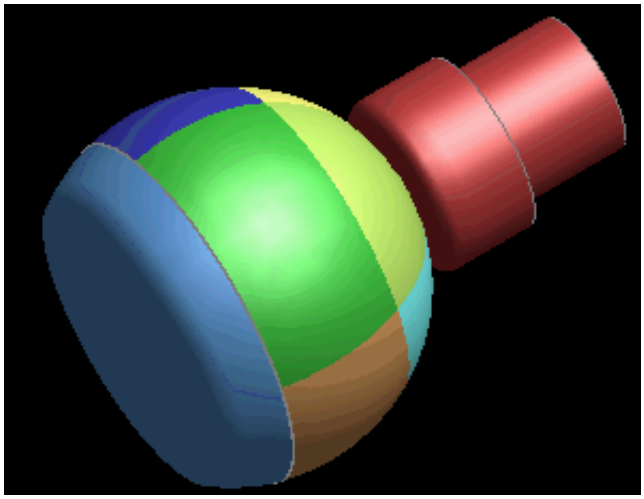
When 8 bit images are produced the 24 bit palette has to be reduced to only 256 colours. To do this the best way is to use Palette Optimization to choose the most representative colours used in the image.

Without Palette Optimization 256 colours can be chosen uniformly along the original 24 bit palette, missing out important colours.

The following figures show the differences in images with Palette Optimization.



This is the original image, saved as a GIF, with no dithering or palette optimization. **Size 6.1kB**

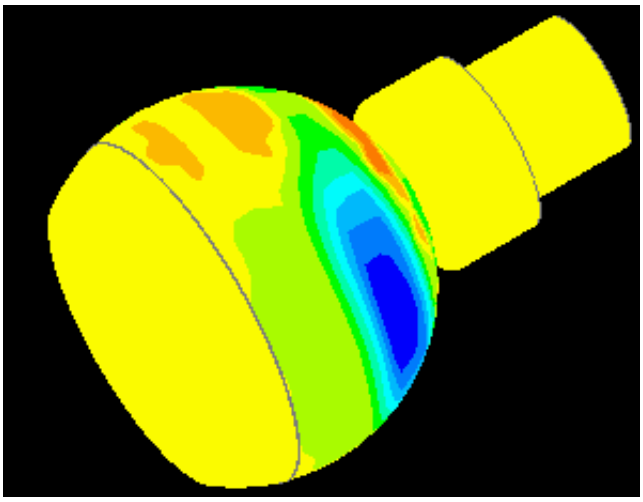


This is the image, saved as a GIF, with palette optimization. **Size 12.6kB**

Note that whilst there are still bands, the coarseness of them has diminished.

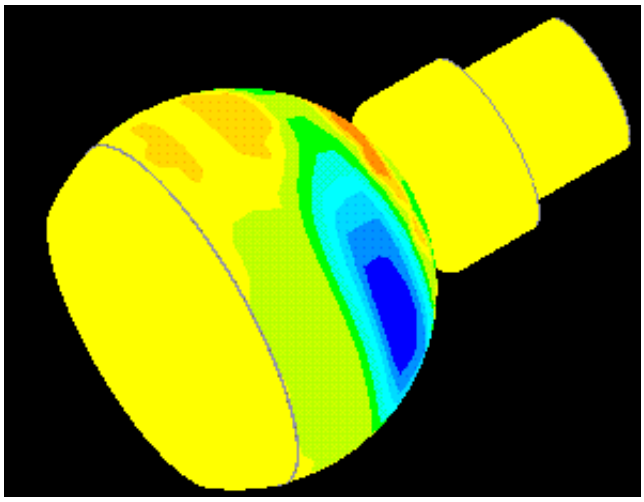
The two images above were created from a shaded image plot in PRIMER and therefore contained a lot of different colours. The banding is present because the palette has been reduced to the 256 most representative colours in the image.

The following figures are images of a contour plot in PRIMER.



This is the image, saved as a GIF, with palette optimization. **Size 3.1kB**

Note that due to the smaller number of colours in the image, there is no banding.



This is the image, saved as a GIF, with dithering. **Size 3.7kB**

Dithering is not well suited to images with distinct colours since by its nature it produces colours that are somewhere in between neighbouring colours. This is effective with shaded images, but not with images where there are sharp changes of colour.

8.5.3 INFO: Further online help about formats

This gives an online "help" message explaining what the various formats are.

8.5.4 Capturing the contents of "menu" windows.

The **IMAGES** command only captures the contents of the graphics window. To copy any other window on the menu interface to a bitmap file use the **SAVE->BITMAP** option in the popup menu belonging to the `[-]` button at its top left corner. (See section 2.4.1).

This distinction is required because the "menu" windows are typically running in X11 window manager overlay planes, whereas the graphics window may be X11 or OpenGL, and is generally located in the screen's image planes. Trying to capture an image which is a composite of different windows, bit-plane depths, physical location in the hardware and graphics type is possible but difficult!

8.5.5 Capturing the contents of all the Primer windows

If you want to grab an image of the whole Primer window contents: graphics, menus, the lot, then:

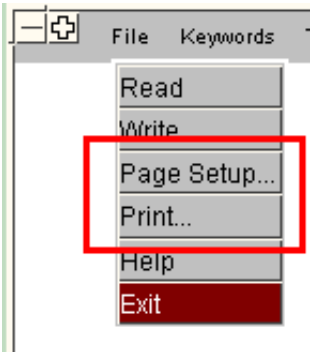
On Windows platforms:
- Use <Alt><Print Screen> with the mouse inside the Primer window
- This will place the image in the Windows "paste" buffer.
- It can then be pasted into other applications.

On Unix platforms:
- Use "xwd -out <filename>" and click inside the Primer window.
- This will create and "xwd" format file which can be used elsewhere.
- Some Unix platforms may have other, non-standard screen grabbing software.

If some windows appear to have the wrong colours this will be because the X "visuals" in the various windows are different, which is invariably the case when OpenGL graphics are used, and this has confused the screen-grabbing programme. To fix this:
- Set the **`SM_USE_VISUAL`** environment variable to "default" (eg **`setenv SM_USE_VISUAL default`**)
- Select device **XMENU...** when starting Primer, (see section 1.2), and choose the visual marked "default"

These two actions will force both menus and graphics to be drawn in the default visual of the window manager, which will maximise your chance of getting a sensible screen grab. However the graphics window will be using X11 (2D) rather than OpenGL (3D) graphics, and the resulting image may be inferior.

8.5.6 PRINT (Windows only)

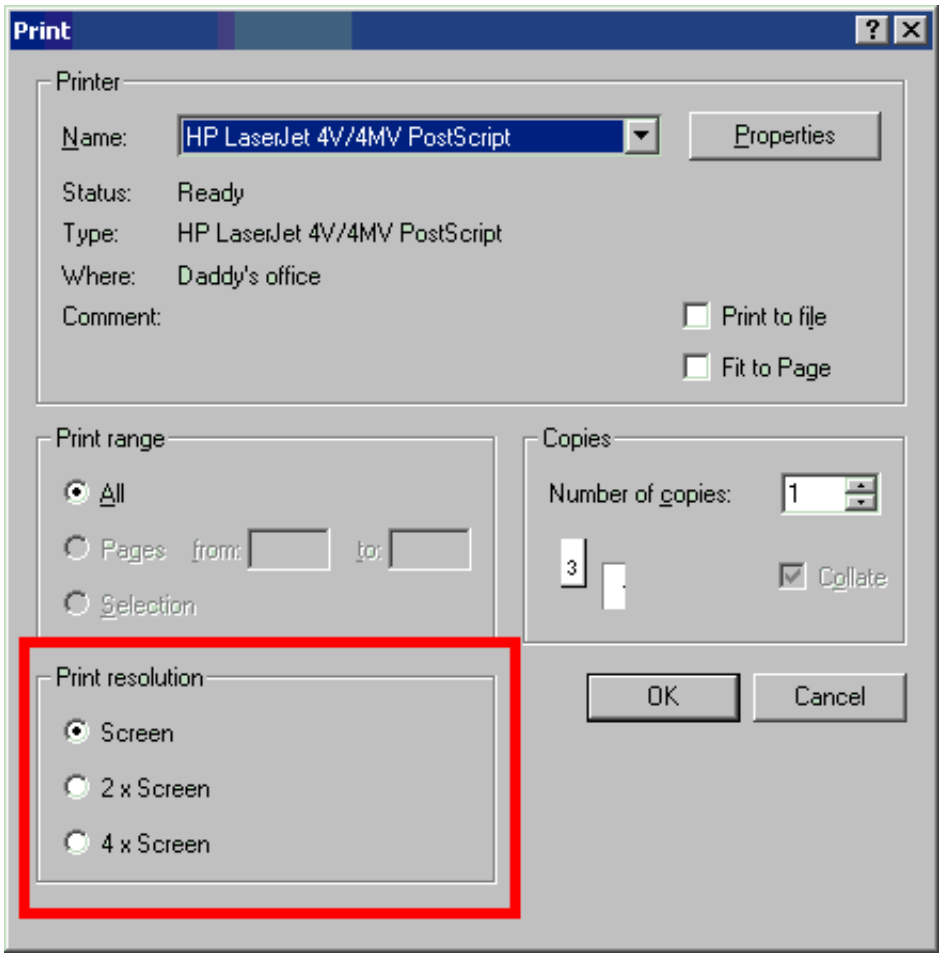


On Windows platforms only the graphics window image may be sent directly to a printer using **FILE > Print...** and **FILE > Page Setup...**

These map the standard Windows printer panels, with the addition of an optional print resolution button (see below).

(***Unix and Linux users*** will need to save a .bmp, .jpg or Postscript file and queue it externally for printing.)

Print Resolution

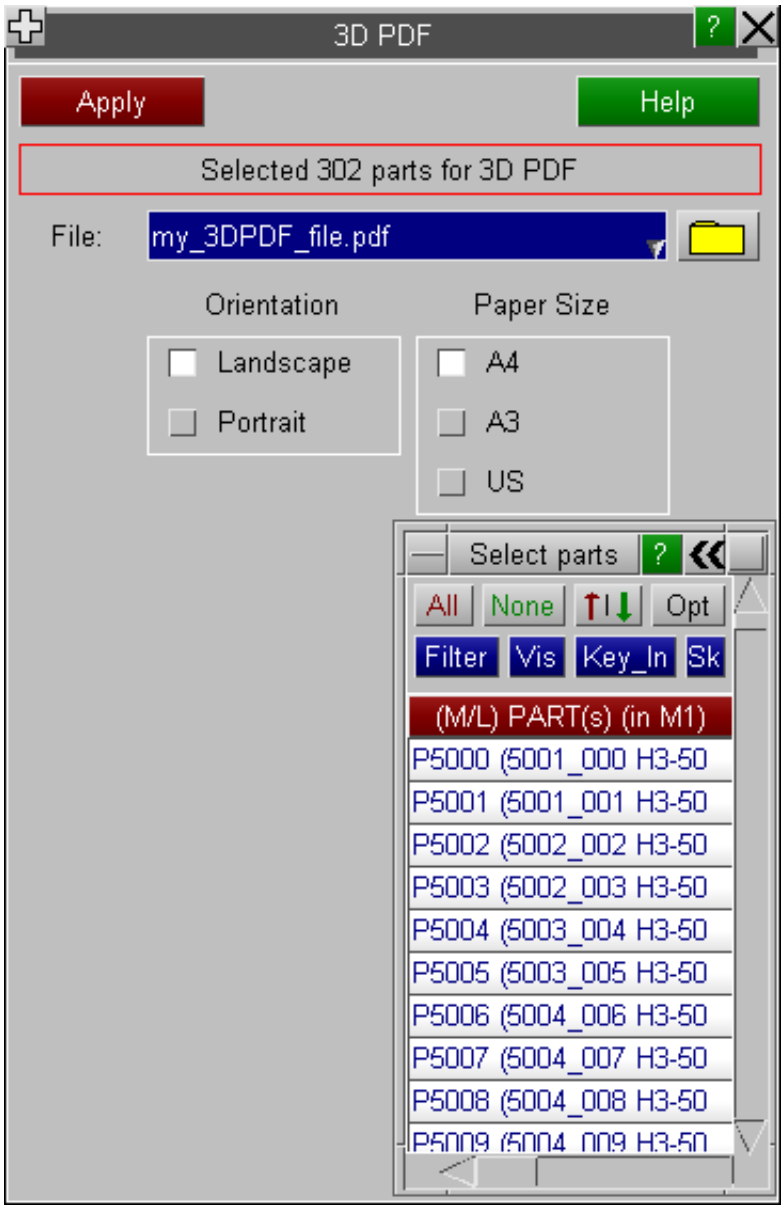


By default "Print" output will be at the screen resolution, but you can choose the options of 2x and 4x screen resolution. Higher resolution output may be preferable if printing to larger paper sizes.

Note that higher resolution files will be much larger (4x and 16x respectively) and may take correspondingly longer to processes and print.

8.6 3D PDF

Three dimensional PDF files can be written by selecting **3D PDF** from the **Images** option in the main menu. The following screen will be displayed.



Input a file name in the **File** box or select a file using the file selector (folder icon). Select the parts to be exported to the 3D PDF file from the object menu and choose an orientation and paper size. Note, at present, only parts consisting of SOLID, SHELL, THICK SHELL or BEAM (true sections) elements will be exported.

Once a file name and one or more parts have been specified, press **Apply** to generate the 3D PDF file. As well as the geometry, their include files, part names, colours and transparencies are also exported.

The generated 3D PDF file is a version 1.7 PDF file which can be viewed in Adobe Acrobat Reader version 8 or later. Within Acrobat Reader the model can manipulated similarly to within PRIMER, e.g. rotate (left mouse button) , pan (Ctrl + left mouse button), zoom (right mouse button). Six default views (+XY, -XY, +YZ, -YZ, +XZ and -XZ) are pre-defined and include file and part visibility can be toggled on and off via the Model Tree. It is also possible to zoom to specific parts (or includes) by right clicking on them in the Model Tree and selecting Zoom to Part. This also centres rotation on the selected part. Alternatively the centre of rotation can be defined under Camera Properties via Select model under Alignment.

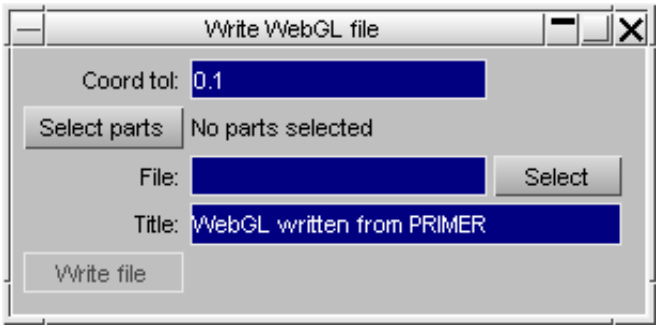
Some preferences can be set within Acrobat Reader that enhance the way 3D PDFs behave. In Acrobat Reader go to Edit > Preferences > 3D & Multimedia. Unchecking the ‘Enable view transitions’ box suppresses the default animation when views are changed. Changing ‘Optimization Scheme for Low Framerate’ (under Auto-Degrade Options) from Bounding Box to None stops parts being replaced with a box when the model is rotated (there doesn’t seem to be much penalty in performance). These settings are saved once the program is closed. For more information on 3D PDFs and associated functionality see, for example, the "Adobe Acrobat XI Help and tutorials" by Adobe Systems Incorporated.

8.7 WebGL

WebGL files enable you to view 3D graphics in a web browser. Currently support for WebGL files in desktop web browsers is good (Chrome, Firefox and Opera have good support) but mobile devices are not yet as well developed.
At the time of writing Internet Explorer 10.0 and older does not support WebGl files but support can be added by use of a plugin called IEWebGL (which is automatically detected by the files produced by PRIMER). See https://github.com/iewebgl/iewebgl for more details. However support for WebGL should be available in version 11.0 of Internet Explorer so the plugin will then not be required..
For more details of which browsers and versions support WebGL see http://caniuse.com/webgl.

8.7.1 Creating a WebGL file

Selecting the **WebGL** command in the **Images** menu starts the WebGL menu.



Give the name of the HTML file to write by either using the **File** textbox or using the **Select** button.
WebGL files can contain a large amount of data for big models so PRIMER tries to reduce the file size, compressing the coordinates by rounding them to a tolerance. This is given by the **Coord tol** textbox. In the above examples nodal coordinates will be rounded to 0.1 units. If your model is in metres rather than millimetres this value may need to be adjusted. Increasing the value will give smaller file sizes but may alter the visual appearance of the model.

Currently PRIMER can only write whole parts to WebGL files. Use the **Select parts** button to choose which parts to write to the file. The parts will be written in shaded mode to the file with the current colour and transparency set in PRIMER. Only solids, shells and beams are supported at present. Beams will be drawn with their true section properties.

Once the parts have been chosen and a filename is given **Write file** will write the WebGL file.

8.7.2 Viewing WebGL files in a browser

Assuming you have a web browser that supports WebGL (see above) the WebGL file should open without any extra steps. If the file is large it may take several seconds to load the data from the file in which case you should see a 'loading' screen.

Once the file is loaded the 3D model will be displayed in a default view. The mouse can be used to modify the view:
The left mouse button will rotate the model
The middle mouse button will translate/pan the model
The right mouse button will scale the model

Parts displays a part tree which allows you to blank/unblank any parts in the model. If your model in PRIMER contained include files then they will also be shown in the tree.

8.8 Read background image and watermark

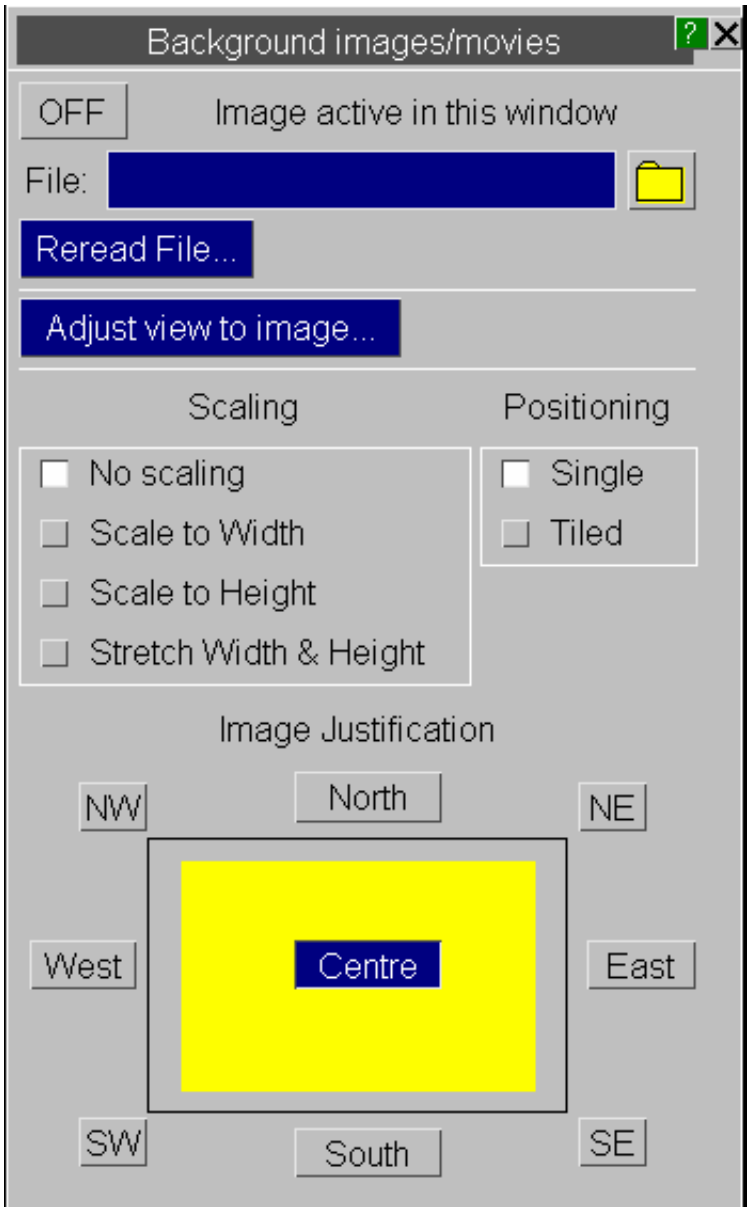
Read an image file to display as a background image behind a model instead of a solid background colour.

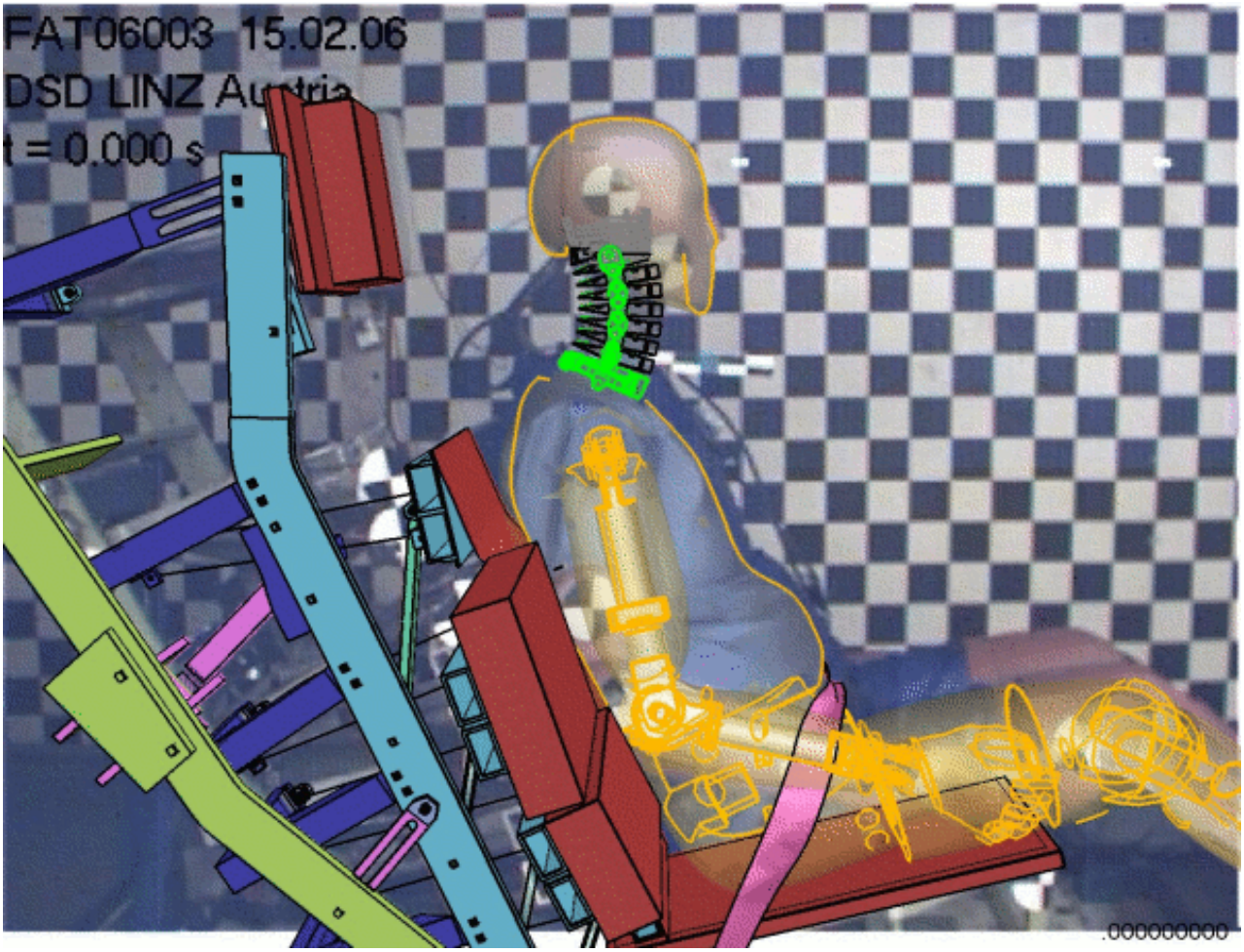
The formats we support are the same as we are able to write, see Capture.

Scaling Options

If the image dimensions do not match the graph window dimensions then the image can be scaled to fit or it can be tiled.

Below is an example background with the model overlayed on top.

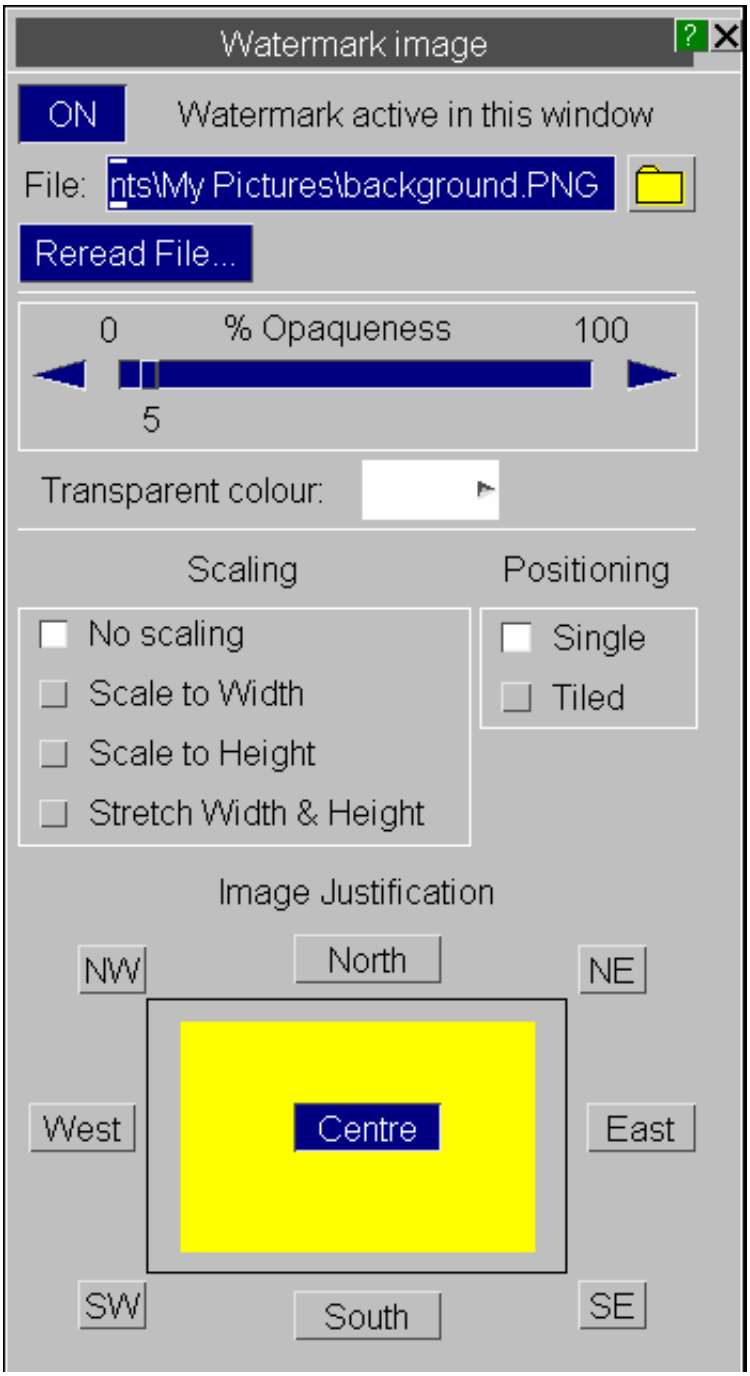


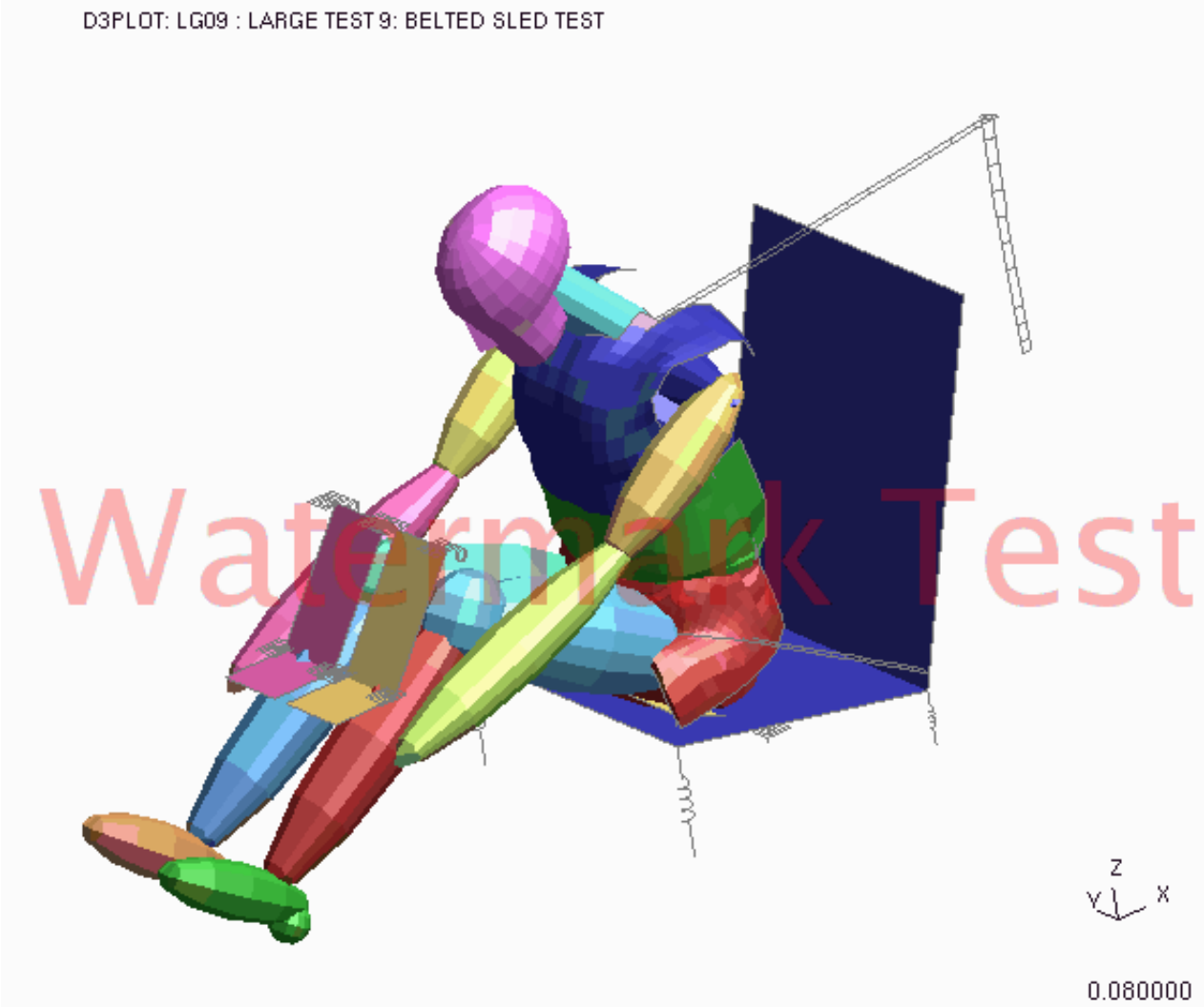

FAT06003 15.02.06
DSD LINZ Austria
t = 0.000 s
.000000000

It is possible to add a "watermark" to a plot. Simply load in an image file in the watermark panel and set its transparent colour and overall transparency.

It will be drawn in front of the normal image, using the transparency settings you have defined. The position and size can also be set.

Below is an example with black as the transparent colour (the image was created on a black background) with 20% opaqueness.





Watermark Test

9 Viewing Controls

9.1 PRIMER Coordinate Space and View Layout

9.2 Basic Commands in the Viewing Control Box

9.3 The "Compass Rose"

9.4 Dynamic Viewing

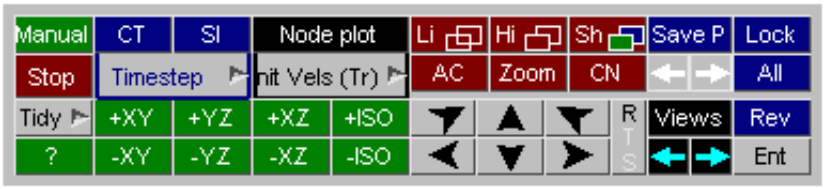
9.5 Further Commands in the Viewing menu.

9.6 Saved properties

9.7 Special 3D Graphics Driver Options

9.0 VIEWING CONTROL

"Viewing" refers to the manipulation and presentation of images, rather than their actual generation. All basic commands live in the "Viewing and Drawing Control" box, located at the bottom right hand corner of the screen. The remaining commands can be found in the **Viewing** option in the main menu.

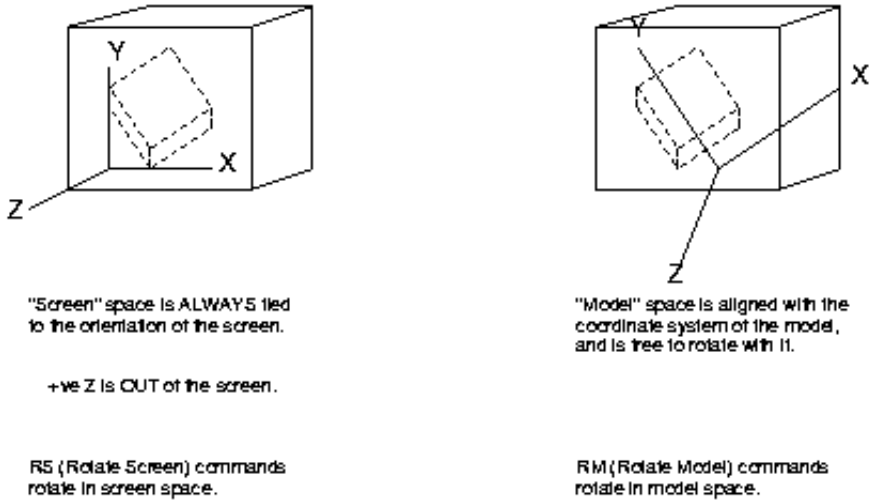


Drawing and Viewing box.

Viewing Menu

9.1 PRIMER coordinate space systems and view layout.

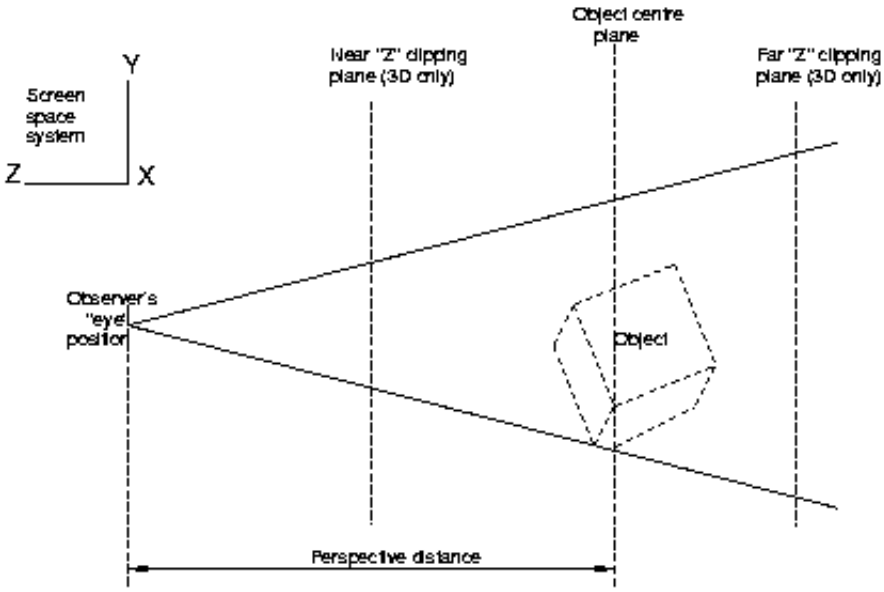
PRIMER uses two right-handed coordinate space systems for viewing: "screen" space, and "model" space. This is illustrated in figure 9.1(a):



Initially the two space systems are coincident, ie the initial view on the model is a plan on XY looking down the Z axis. Transformations to the current view can be applied in either space system, with the result that the model coordinate system will rotate with respect to the (fixed) screen system.

The current model orientation, (ie the axis system in the right hand side of figure 9.1(a) above), is shown by the "triad" in the bottom right of each plot.

The object exists at a point in screen space, and is seen through a viewing "frustrum" as shown in figure 9.1(b) below. The observer's (your) eye point is located at the vertex of a rectangular section frustrum, with the object some distance away in the -ve Z screen space system. The screen image is a 2D projection of what the eye sees: the sides of the frustrum clip the view to the left/right and bottom/top edges of the screen.



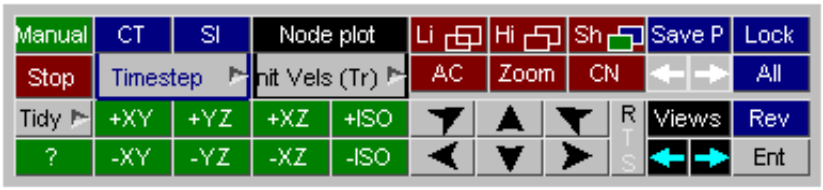
It is important to note the following:

1. Rotation of the image always takes place about the point that is the XY centre of the screen (in screen space coordinates), at the "object centre plane" (which gives the screen Z location). Thus in the example above roughly at the "O" of "Object".
2. The example above implies a perspective projection. In fact PRIMER defaults to a parallel (orthographic) projection, in which the sides of the frustrum are parallel and perspective is irrelevant. Nevertheless the object

centre plane is still significant, since this still gives the screen Z coordinate about which rotations take place. You can turn perspective on/off and alter its distance at will.

3. You can change the scale ("zoom" in/out) of your image. This effectively changes the angle of view of the frustrum above: zooming in makes it narrower, zooming out wider. But note that this does not change your distance from the object: changing the scale is like putting a lens of a different focal length on your camera, to change your distance from the object you must alter the perspective distance.
4. The near and far "Z" clipping planes shown here only apply on 3D hardware that is capable of this. See section 9.6.

9.2 The Viewing Control box

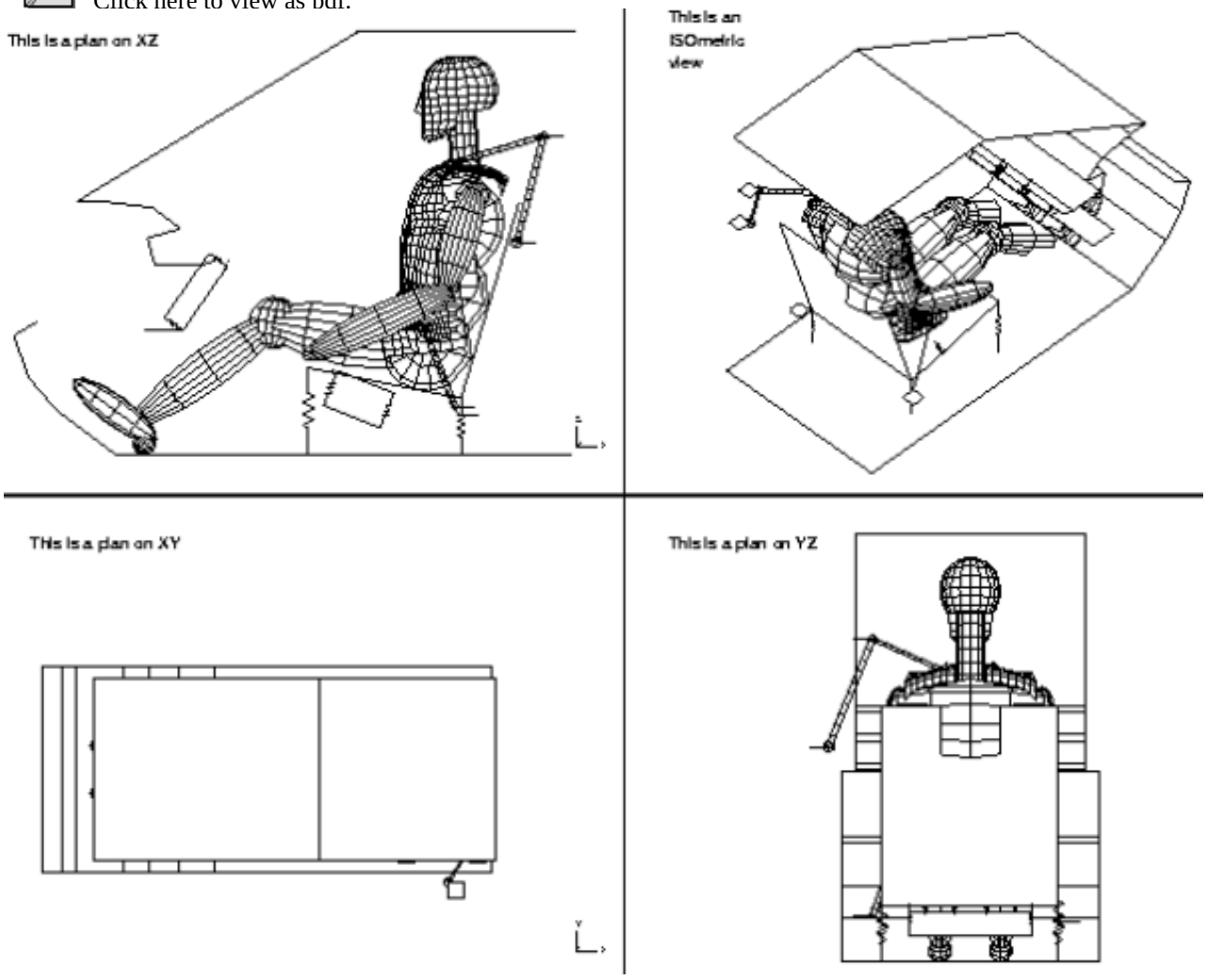


All aspects of viewing are controlled from within the "Viewing and Drawing Control" box. Its layout is shown on the right.

9.2.1 Pre-programmed views.

+/-XY Is a plan on model XY, looking down/up Z.
+/-XZ Is a plan on model XZ, looking down/up Y,
+/-YZ Is a plan on model YZ, looking down/up X,
+/-ISO Is an isometric view (equal angles for X,Y,Z).
These views are also available from shortcut keys 1,2,3 etc.

Click here to view as pdf.

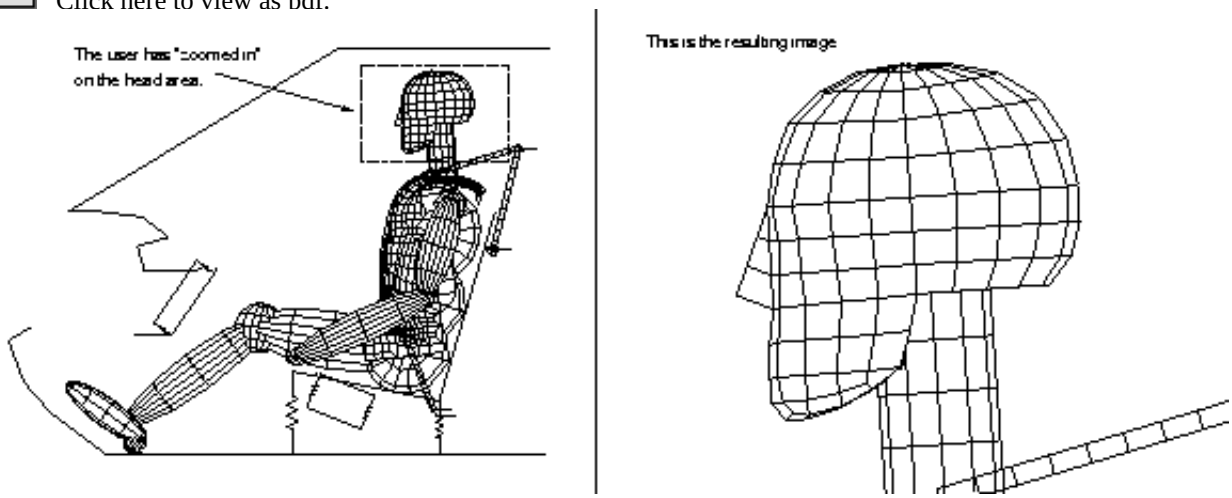


These views only apply rotations, they do not affect scale or position.

9.2.2 "Zooming in" (magnifying an area). Zoom

ZM Zooms in by using the cursor to pick a rectangular screen area that is to be enlarged to fill the screen (also available from shortcut key Z).

Click here to view as pdf.



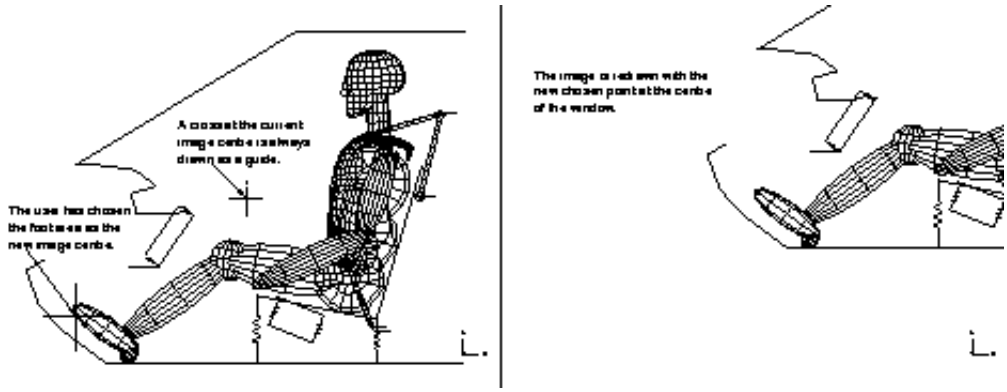
Both the scale and the current image centre are changed.

When defining the rectangle with the cursor you can "rubber-band" the box by picking its first point, then pressing the button and moving to see the effect of the second point.

9.2.3 "CeNtre" (Centre Node). CN

CN Lets you choose a new node that will become the new image centre and about which rotations will be performed. The point chosen with the cursor moves to the window centre, but the scale is not changed.

Click here to view as pdf.



9.2.4 AC (Autoscale Current) Scaling the view to fit the screen AC

AC Calculates the correct scale and centre position required to make the current image fit neatly onto the screen (also available as shortcut key A). This takes account of blanking, clipping, deformations, etc.

9.2.5. Zooming using +/-

Shortcut keys + and - magnify/reduce the current view. Zooming takes place about the current cursor position on the plot unless CN is active, in which case it takes place about the chosen centre node.

9.2.6 Zooming using mouse wheel

The current view can be magnified by moving the mouse wheel down (towards the user). The view can be reduced by moving the mouse wheel up. Zooming takes place about the current cursor position on the plot unless CN is active, in which case it takes place about the chosen centre node.

9.2.7 Scrolling through previous views

The and buttons allow scrolling through previous views. Up to 100 previous views are stored automatically, and you can cycle backwards or forwards through these at will. The view storage "wraps round" at its ends, so going forwards past view #100 takes you to view #1 again.

9.2.8 Storing and retrieving properties.

The **Save P** button and the two arrow buttons below it allow you to save "properties", which are the current viewing attributes, and then cycle backwards and forwards between saved properties. A "property" includes any permutation of colour, transparency, display mode, blanking, entity switches and current view - all the factors which control the appearance of the image on the screen. Properties may be saved to property files (.prp) and reloaded at any time, and these files are portable between different Oasys Ltd. LS-DYNA Environment programmes.

Properties are described in more detail in section 9.6

9.2.9 Blanking control buttons

Rev reverses all blanking (or use shortcut key R). **Lock** stores the current blanking status. **All** restores the last stored blanking status.

9.2.10 Other buttons on the Viewing Control panel

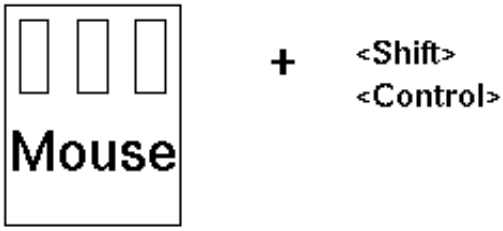
Manual invokes the on-line manual. **Stop** interrupts the current operation. Click here to see a description of **Tidy**.

9.2.11 Alternatives to the commands in this section.

The commands listed here are useful for making explicit changes to the view of your model, for example to set it to a view exactly co-incident with axis orientation or locate its centre at a specific position. However there are two other ways of making these changes which, in some contexts, may be better:

Compass Rose

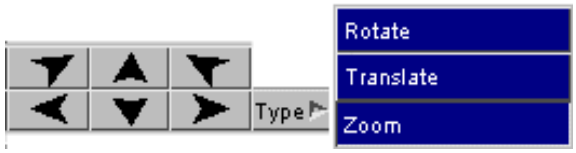
Dynamic viewing



The "Compass Rose" is described in section 9.3, and dynamic viewing in 9.4.

9.3 Using the "Compass Rose"

The "Compass Rose" provides three sets of buttons that allow the model to be rotated, translated and scaled with single mouse clicks.



9.3.0 General information on using the Compass Rose

The Compass Rose operates in one of three modes selected by the **Rotate**, **Translate** and **Zoom** buttons available from the **Type** pop-up menu as shown in the example above (**Rotation** mode is selected). The options provide:

Rotate Arrow buttons provide pre-defined increments of rotation about each of the X,Y,Z axes in screen or model space.

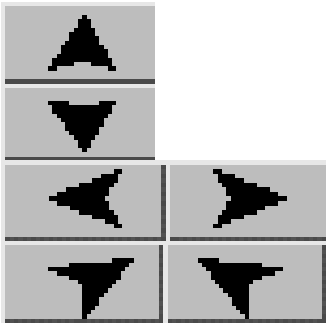
Translate Arrow buttons provide pre-defined increments of translation in each of the X,Y,Z axes in screen or model space.

Zoom Buttons provide pre-defined increments of magnification and reduction of viewing scale.

Also, a single click on a function button will deliver a single instance of that function; but holding the button down will, after an initial delay (see Settings), cause it to repeat its function at a pre-defined rate. If command-file recording is on each such repeat is recorded as a separate button click: this makes it possible to record and replay sequences of view changes.

9.3.1 Rotation functions.

The arrow buttons have the following meanings:



Rotates about *x* axis

Rotates about *y* axis

Rotates about *z* axis

Each click generates an increment of rotation about the relevant axis or, if held down, continuous rotation.

The sign of the rotation is intuitive for system rotations, for example => gives clockwise (+ve) rotation.

9.3.2 Translation functions.

The Translation functions are very similar to the rotation ones: translation defaults to screen space, and uses a similar system of arrow buttons.

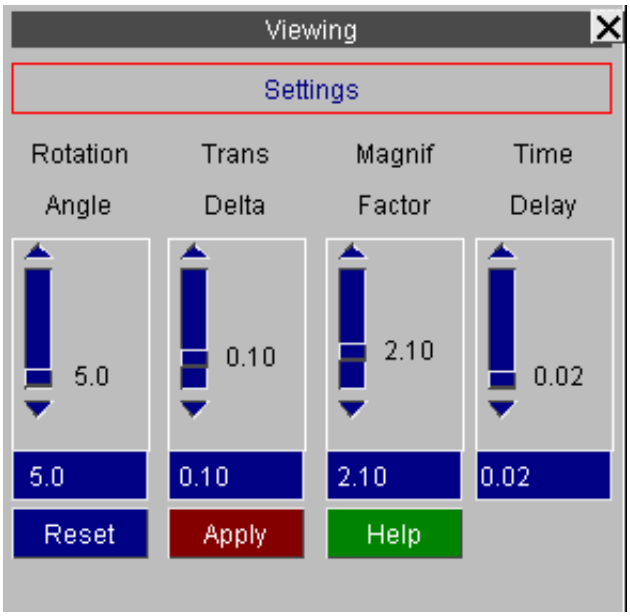
9.3.3 Magnification functions

The Magnification function is very simple:

+ Increases the size of the image by the pre-defined increment;
-- Decreases it by the same increment.
The image is scaled about the current window centre.

9.3.4 Setting attributes

In each case the the option **Settings** in the **Viewing** menu gives access to further options which allow you to modify the pre-defined settings (angular increments, time delays, etc).



The attributes that can be set are:

Rotation Angle Angular increment in degrees. The default is 5 deg, but any value 0.0 < value < 180.0 is valid. The **ANGLE** increment of five degrees is a reasonable value for single clicks, but is really too large to give the impression of smooth rotation under continuous motion: you will probably find that a value of 1 or 2 degrees is better for that.

Trans Delta The translation increment as a fraction of screen span. Default is 0.1 (ie 10%), but any value 0.0 < value < 0.5 is valid.

Magnif Factor The magnification factor. Default is 1.1 (ie 10%), but any value 1.0 < value < 5.0 is valid.

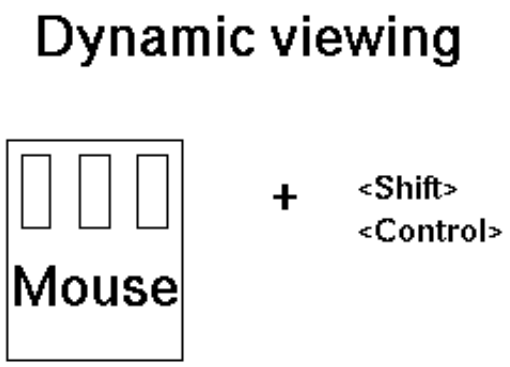
Time Delay Is the time delay (in seconds) between continuous transformations when a button is held down. The default is 0.02s, but values 0.0 <= value < 0.5 are valid. The **Time Delay** is the minimum permitted time delay between frames. If the hardware is taking longer than this to render each frame it does not add to the delay, it simply pads it out if the inter-frame time is shorter than this interval.

You may be tempted to cut the time delay between transformations down to zero, and on very fast hardware (typically 3D machines) this gives good results, but on slower hardware you may find that this gives uneven results as competing system demands lead to variable elapsed intervals between frames. It is for this reason that the delay of 0.02 seconds is the default: it barely slows transformation (50 frames per second is twice as fast as a TV set scans!), but it does even out the delay time between frames giving a smoother result - especially under X-Windows. Experiment on your hardware.

9.4 Dynamic Viewing (Using the mouse to change views).

"Dynamic" viewing is the name given to the process in which you perform viewing transformations by moving the mouse around the screen. This is the final, and probably most useful, way of controlling views described in this section.

The three classes of transformation available in the compass rose (rotation, translation and scaling) are all available in dynamic viewing as well. However rotation and translation are only available in "screen" space: it would be too confusing to try to relate cursor movement to "model" space transformations.



9.4.0 Graphics modes during dynamic viewing

All dynamic viewing operations require a combination of two screen "meta" keys, (**<left control> and <left shift>**), and mouse buttons. The meta key(s) used dictates the graphics mode in which the image is transformed as follows:

<left shift> + <mouse>	Transforms the image in the current graphics mode. For example if it is a hidden-line plot, then dynamic viewing will take place in hidden-line mode.
<left control> + <mouse>	Transforms the image in "wire-frame" mode for the duration of the drawing operation. (i.e. no hidden-surface removal or lighting.)
<Left shift> &}<left control>} + <mouse>	Transforms the image in pre-computed free-edge mode for the duration of the drawing operation. (i.e. wire-frame of free edges only, no hidden-surface removal or lighting.)

In the latter two cases the original drawing mode is always returned to at the end of the dynamic viewing operation. The wire-frame and free edge modes are provided to make transformations quicker for very large models and/or slow computers: free edge is very fast.

For the last case, with **<left shift>** & **<left control>** held down together, the order of pressing and releasing the meta-keys matters: press **<left shift>** before **<left control>,** and release in the opposite order, otherwise you will (correctly) get the image redrawn in wire-frame mode as the **<left control>** key is pressed and released.

9.4.1 Dynamic Rotation.

Dynamic rotation uses **<left mouse>** + **<left shift>** &/or **<left control>**

(The distinction between the keyboard meta-keys is explained in section 9.4.0 above.)

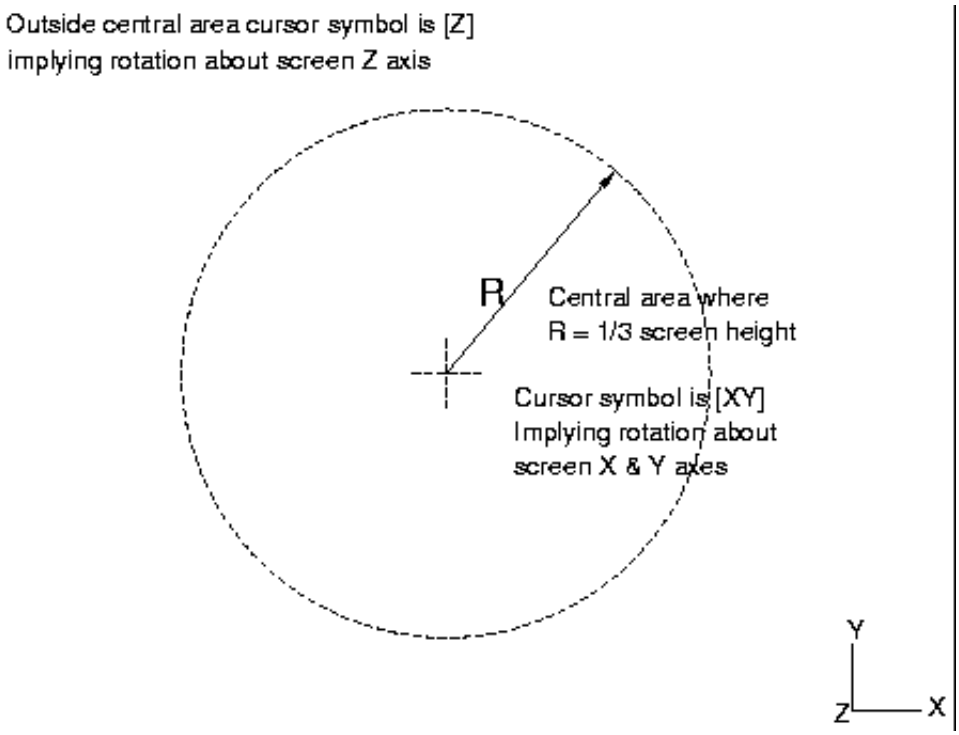
Rotation always take place in the screen coordinate system, and may be about the XY axes or Z: this depends upon the starting position of the mouse. This is shown in figure 9.4.1:

If the mouse initial position is *inside* the central circle (radius (screen height/3)) then rotation is about screen XY axes.

If the initial position is outside this circle then rotation will be about screen Z.

You can tell which mode you are in by the cursor symbol. This red, and:

XY rotation uses **[XY]**
Z rotation uses **[Z]**



The relationship between mouse and image motion is intuitive in both modes. It is as if you had grabbed a point on the object near you, (this side of the object centre plane), and used this to move the image about its centre:

XY mode Moving the mouse left/right rotates about the screen Y axis;

Moving the mouse up/down rotates about the screen X axis.

Z mode Moving the mouse in a circular direction rotates about the screen Z axis.

Rotation remains locked in its initial XY or Z mode for the duration of a dynamic viewing operation, regardless of where you subsequently move the cursor to, until you release a mouse or keyboard button.

9.4.2 Dynamic Translation.

Dynamic translation uses **<mid mouse>** + **<left shift>** &/or **<left control>**

(The distinction between the keyboard meta-keys is explained in section 9.4.0 above.)

The cursor symbol is yellow, and looks like:

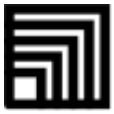
Translation always take place in the screen coordinate system, in the X and Y directions.

The relationship between mouse and image motion is intuitive: the object tracks the mouse motion in the screen XY plane. The initial position of the mouse is irrelevant.

9.4.3 Dynamic Magnification (Scaling).

Dynamic scaling uses **<right mouse>** + **<left shift>** &/or **<left control>**

(The distinction between the keyboard meta-keys is explained in section 9.4.0 above.)

The cursor symbol is green, and looks like: 

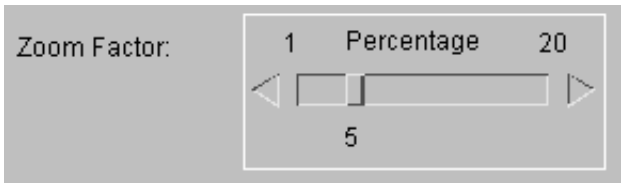
Mouse motion to the right and up makes the image larger, left and down smaller. The initial position of the mouse is irrelevant.

Dynamic magnification using the mouse scroll-wheel

If your mouse is equipped with a scroll-wheel then it will also perform dynamic magnification in the graphics window in which the cursor is present.

- Magnification is centred at the current cursor position unless "**CN** Centre node" has been used to lock centring on a node.
- Scrolling towards you magnifies the image scale.
- Scrolling away from you reduces the image scale.

By default each scroll wheel "click" will change the magnification factor by +/- 5%, but this can be changed using the **Options > Menu attributes** panel, and altering the **Zoom factor.**



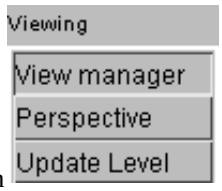
9.4.4 3D Mouse

From v11.0 onwards, dynamic viewing is also possible through the use of a 3D mouse. PRIMER currently supports 3D mice produced by 3DConnexion. The 3D mouse is used in conjunction with a traditional mouse, by using one control to

simultaneously pan, scale and rotate the model, while the traditional mouse is used for entity selection. Tilting or rotating the command cap of the 3D mouse will rotate the model around the geometric central point of the the visible entities. A rotation point can be manually set using "**CN** Centre node".

Different models of 3D mice also contain buttons that can be used within Primer for various operations. You can assign functions, macros and javascripts to the buttons on a 3D mouse by using the shortcut panel. See section 2.2.6 for more information.

9.5 Further commands in the Viewing Menu



The following commands with sub-menus in this box are described:

View manager... Storing and retrieving views (this can also be accessed from the button **Views** in the Drawing and Viewing Box).

Perspective... Controlling perspective, also "locate target and eye"

Update Level... Setting plot update frequency

9.5.1 VIEWS... Storing and retrieving "view" information.

What is a view?

A "view" is all the information required to set up the current view of the object. In practice this means:

- The current rotation matrix (3 direction cosines).

- The current image centre location in space (x,y,z coordinate).

- The current magnification scale.

- The current perspective distance.

Up to 100 such views may be stored and retrieved at will from a file, and any number of such files may exist. A view is given a name and number when it is stored, and these are used when retrieving it

Up to and including release 9.3: Views are stored parametrically.

What this means is that views are not tied to a particular model, they will work for any model of similar dimensions. So if you are working on a set of variants of an analysis you can share the views on file between them: this is why they are stored in a separate, model-independent file. It is only when the shape and/or size of a model differs wildly from the original from which the view was created that this shareability fails.

From release 9.3.1 onwards: Views are stored explicitly

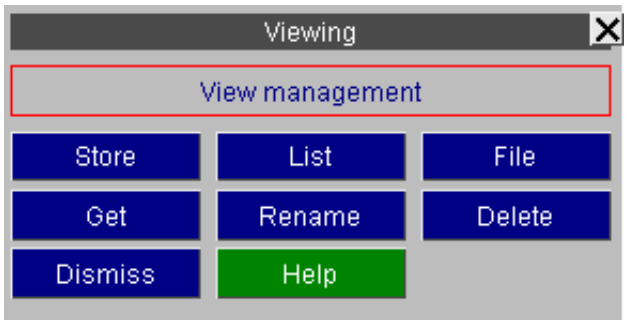
The parametric method described above was not a success, as users wishing to compare models visually found it misleading. Therefore from release 9.3.1 onwards views are now stored explicitly, and no account is taken of model size or position. Put qualitatively: the camera now stays in the same place with the same settings.

Retrieval of views is backwards-compatible. A view stored prior to V9.3.1 will read successfully into V9.3.1 onwards, but will be converted to "explicit" format if subsequently saved.

Using views

PRIMER always has a current "view" definition. This dictates how the image will appear when a drawing command is issued. You can save the current view to file at any time. Likewise you can retrieve a stored view to replace the current one at any time.
The current view only exists in memory, and changing it has no influence on any views stored on file. (Indeed you don’t need to have a stored view file: the default is none.)

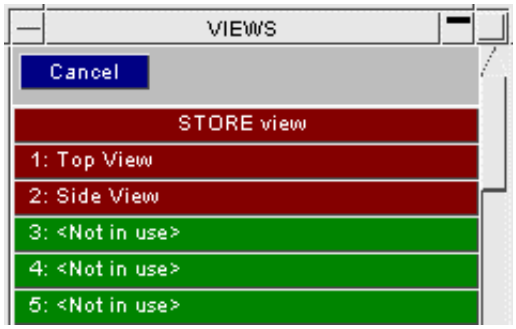
Managing views

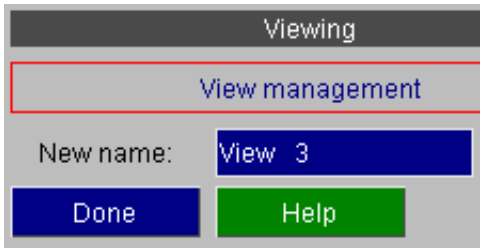


When you press the **Views**... button you get the View Management panel shown in figure 9.5.1(a).

The controls are described below.

9.5.1.1 **STORE** Storing the current view on file.





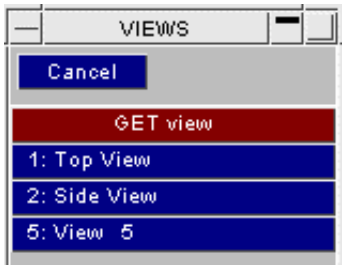
You are presented with the **STORE** view menu showing views 1 to 100, and you must choose which one this view is to be stored as.

Views currently in use are red, with their current names shown; unused views are green, and marked **<Not in use>**.

Once you have defined the view number, then give a name. A default name of **"View <n>"** is provided, but any name is valid.

Up to 100 views can be stored in a file, and views can be overwritten at will. If no explicit file has been opened the default file plot.view is opened automatically and used.

9.5.1.2 **GET** Retrieving a view from file.



You can only retrieve from file views that already exist.

You are presented with the **GET** view menu of stored views, and must pick one. In this example three views are available.The attributes of the stored view are converted from parametric form to your model's coordinate system, and then become the current view.

If the **`UPDATE_LEVEL`** is 2 or above (see section 9.5.3) then the view takes effect immediately, otherwise it becomes effective the next time you issue a drawing command.

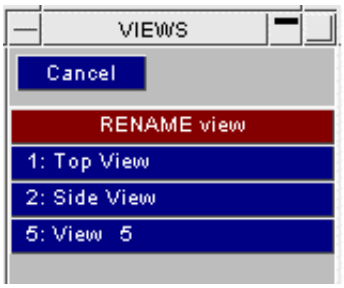
9.5.1.3 **RENAME** Renaming stored views.

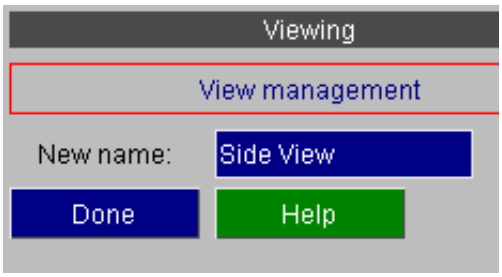
You can rename any stored view.

Select a view from the **RENAME view** menu, then give it a new name.

Any (or no) name is acceptable. This is simply a label by which the view is known.

Here the user has chosen to rename view #2, currently called "Side view".

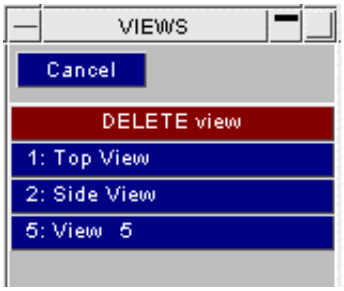




9.5.1.4 **DELETE** Deleting stored views.

You can only delete existing views. Select a view from the **DELETE** view menu, and it will be deleted.

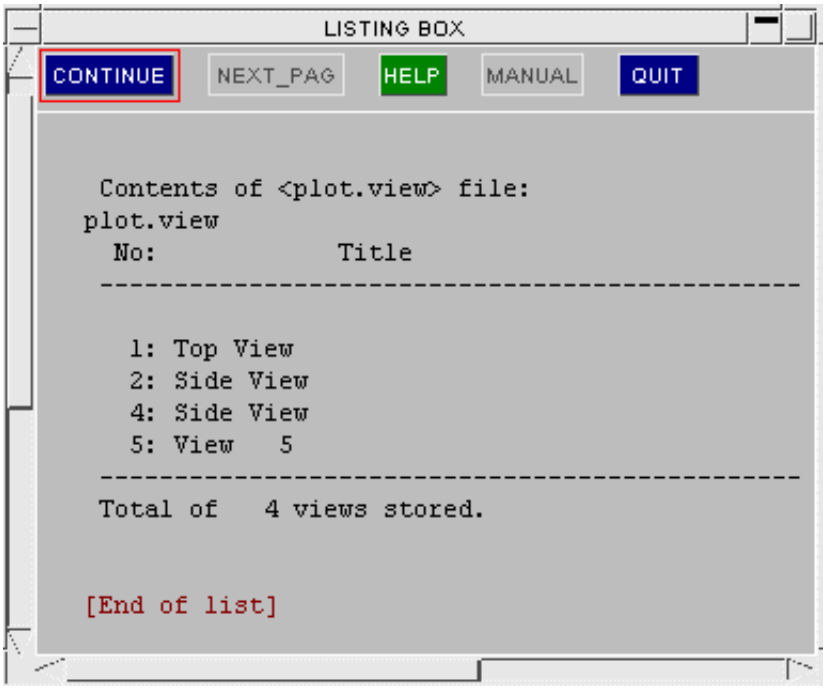
There is no warning or confirmation dialogue, so make sure that you want to do this! **CANCEL** can be used to abort the operation.



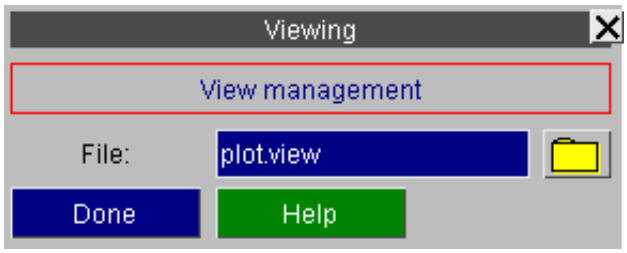
9.5.1.5 **LIST** Listing stored views

You can list information about stored views to screen with the **LIST** option.

If you have a lot of views this is a better way of listing them than trying to use the menus in a confined space.



9.5.1.6 **FILE** Selecting/Defining a View Storage filename

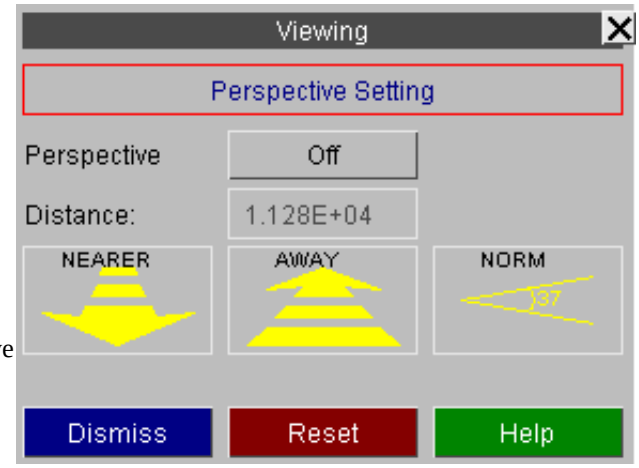


By default views are stored in a file called plot.view, and generically the view storage file is referred to as the **`<plot.view>`** file.

However you may choose any filename, and you may have any number of view storage files. To open a new or existing **`<plot.view>`** file use the **FILE** command, and enter a new filename.

To use the file filter click the button to the right of the field.

9.5.2 **PERSP**... Setting Perspective Attributes.



By default perspective in PRIMER is off.

Figure 9.5.2(a) shows the perspective control panel it this default state. Note that the distance changing options are greyed out as a consequence of perspective being turned off.

Figures 9.5.2(b) and (c) show the effect of turning perspective OFF and ON for a rectangular box. In the left image, where perspective is off, the image is foreshortened and looks strange; in the right image, with it on, the box looks more normal.

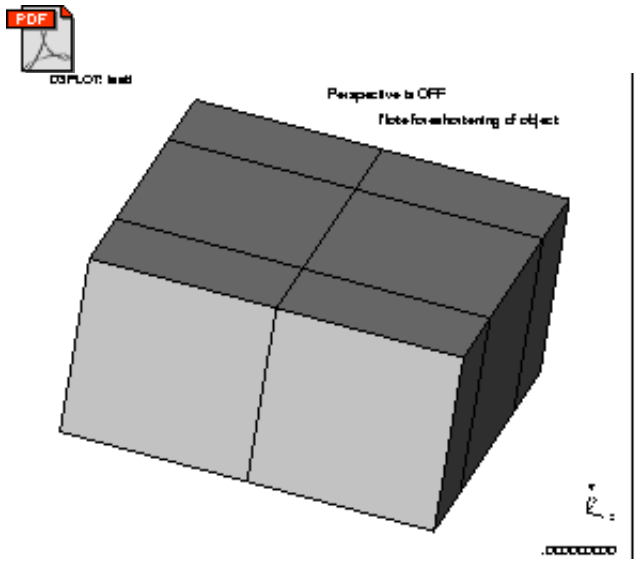


Figure 9.5.2(b): Perspective OFF

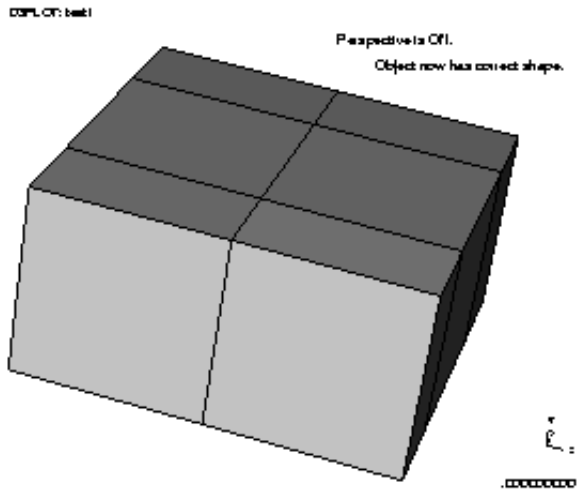
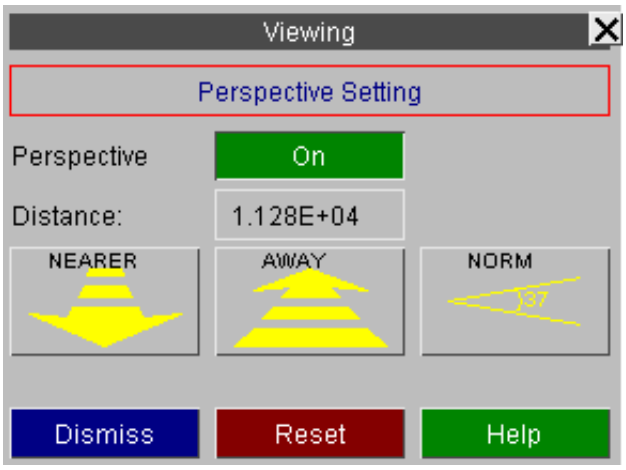
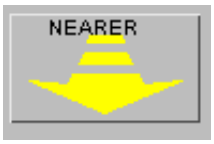


Figure 9.5.2(c): Perspective ON

Figure 9.5.2(d) shows the control panel when perspective is turned **ON.**

Note that the distance changing options are now live. These are used as follows.



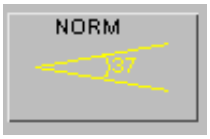


NEARER Reducing perspective distance.

Clicking on this button reduces the perspective distance by 5%. This brings the viewer closer to the object.

AWAY Increasing perspective distance.

Clicking on this button increases the perspective distance by 5%. The takes the viewer further away from the object.



NORM Restore perspective distance to its **NORM**al value

This button restores the perspective distance to its standard setting, which gives a field of view of about 37 degrees.

For all three buttons above the effect is immediate if the **`UPDATE_LEVEL`** (see 9.5.3) is 2 or greater. In addition holding down a button gives a repeated action after an initial delay, so that you can, in effect, see the effect dynamically as you change the distance.

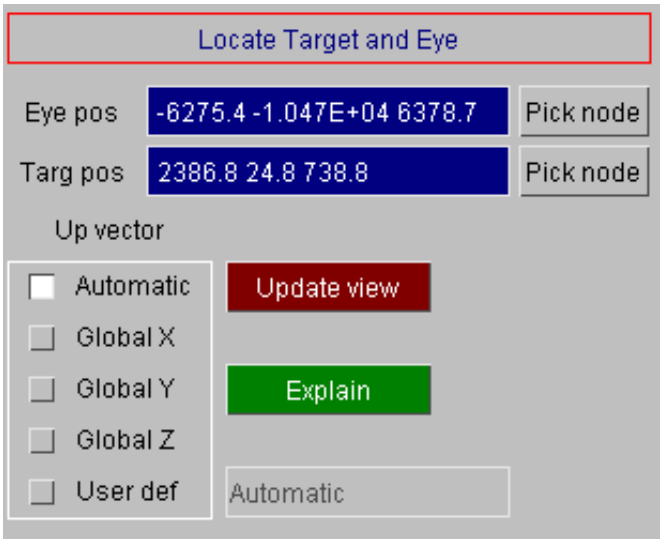
Setting the distance explicitly.

You can type in an explicit Distance from the observer.

9.5.2.1 Locate Target and Eye

Normal PRIMER viewing effectively positions the model in front of a stationary camera, then rotates, pans and enlarges it to place the desired region in the field of view of the lens.

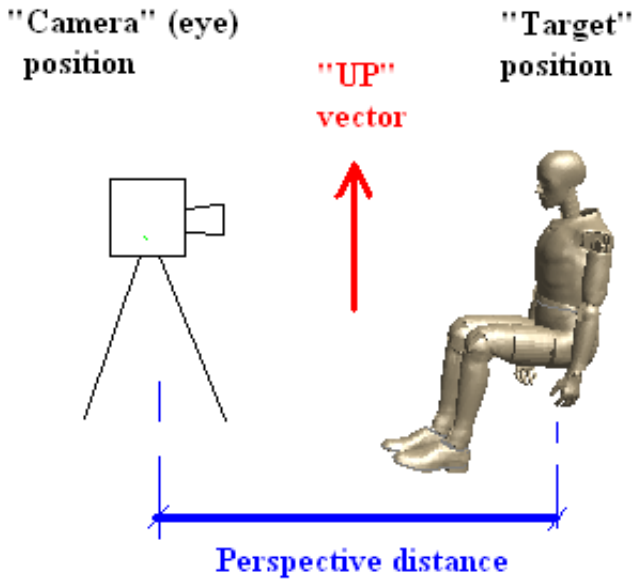
However it is possible to set the "eye" (camera) position and also the "target" point on the structure at which the camera is pointing, and PRIMER will compute the viewing transformation required to give the image from this point.



There are three components in a "Locate target and eye" definition:

Target position	This is the coordinate in space at which the camera is pointing.
Eye position	This is the coordinate in space at which the camera (eye) is located
"Up" vector	This is the vector defining "which way is up". Panning the camera up and down would move it up and down this axis

The distance between the camera (eye) and target points is implicitly the current perspective distance, and this is reset when you **Update** the view. Perspective is switched on automatically if this is not already the case.

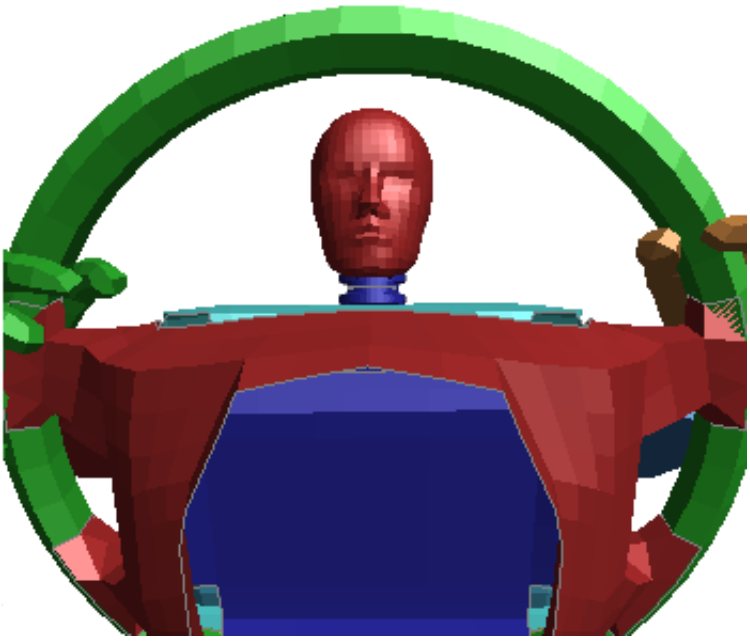


Both target and eye positions may be defined explicitly as coordinates in space, or you may screen-pick a node and its coordinate will be extracted.

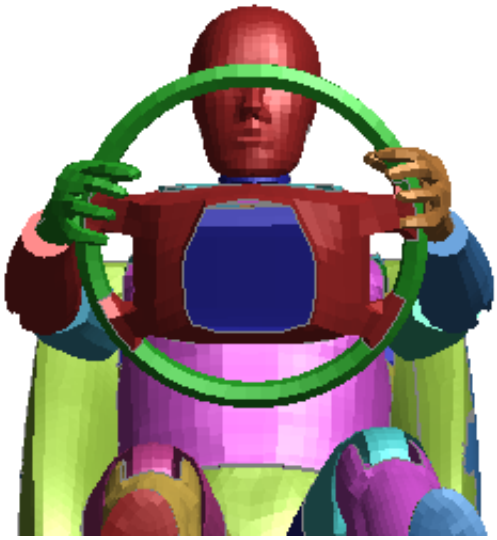
By default PRIMER tries to deduce the "Up" vector automatically, but you can override this by choosing a global vector, or by defining your own arbitrary vector.

The relationship between Perspective Distance and Scale.

If you use the "locate target and eye" feature you will almost certainly position your eye fairly close to the structure, which will bring you much closer than the normal perspective distance set by PRIMER which is 3x the diagonal of the bounding box around the model. When the perspective distance becomes small the fore-shortening effect it causes becomes much more obvious



In this image the target point is the dummy's nose, and eye point has been placed on the steering column just behind the wheel.



In this image the target point is the same, but the perspective distance has been increased by a factor of three, effectively moving the eye point backwards out of the paper.

Photographers will recognise that the perspective distance is, quite literally, the distance between subject and camera,

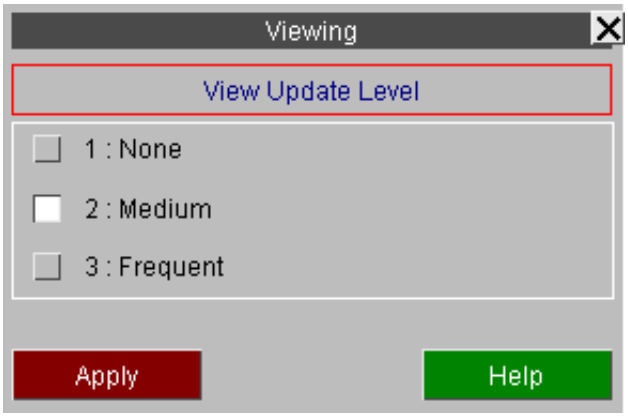
whereas the scale is the "zoom power" (or, more precisely, focal length) of the lens on the camera. Both images above show the dummy head at approximately the same ***scale***, but the difference in ***perspective distance*** gives rise to very different images.

If you are attempting to select viewing attributes to match an existing image you may find this quite difficult to achieve by hand since there are 11 independent variables to match in such an operation:

- Camera position (x,y,z coordinate = 3 variables)
- Subject position (ditto = 3 variables)
- "Up" vector (ditto = 3 variables)
- Scale (1 variable)
- Perspective distance (1 variable)

The **Match Image** function below will calculate this for you when given at least four points on the image and structure to match.

9.5.3 UPDATE... Controlling the View updating frequency.



PRIMER has an **UPDATE_LEVEL** setting which dictates how often the view is updated following commands that change it.

Figure 9.5.3 shows the **UPDATE** panel and its three settings. These have the following meanings:

UPDATE_LEVEL = 1 **No updates**

The plot is never updated automatically. Changes only become apparent when you issue an explicit drawing command, eg **DR, CT**, etc.

UPDATE_LEVEL = 2 **Medium updates**

The plot is updated immediately when any view control command is given.

The current image is amended as necessary following blanking, clipping, etc if any viewing command, including dynamic viewing, is used. In other words a viewing change command is tantamount to an explicit redraw command in the current mode which would, of course, reflect any changes in the model geometry.

UPDATE_LEVEL = 3 **Frequent updates**

The plot is updated immediately as at level 2 above, but also following any blanking, clipping, etc, command that would change the image if explicitly redrawn.

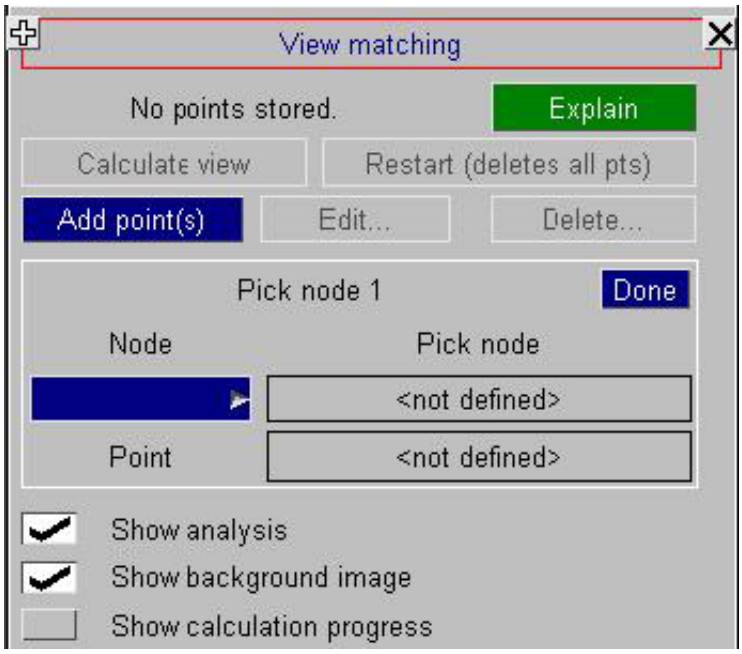
Therefore the effects of blanking, etc are seen immediately.

Note 1 The default setting is 2 on a windows device.

Note 2 Level 3 is only recommended if you have a very fast display and/or a small model since it requires frequent redraws.

Note 3 Users with slow devices and/or with large models may find that level 1 is preferable to decrease redrawing effort.

9.5.4 Match Image



Automatically aligns the current model image with the background by calculating the transformation parameters required.

Lining up an image requires the calculation of 11 unknowns:

- The camera position (3 coordinates)
- The direction in which the camera is pointing (3 vector terms)
- The "Up" axis of the camera (3 vector terms)
- The distance of the object from the camera, ie perspective distance (1 term)
- The focal length of the camera lens, ie image scale (1 term)

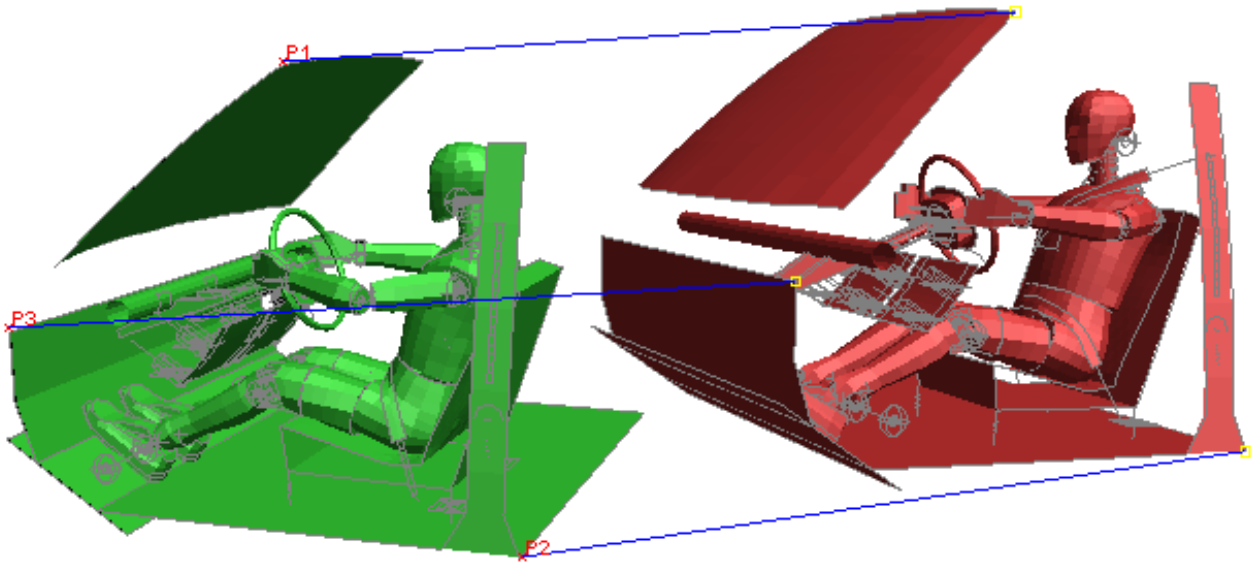
(In the orthographic case, where the object is viewed in a parallel sided frustrum, the perspective distance can be omitted leaving only 10 values to be computed.)

This calculation can be performed by Primer if four or more nodes on the model are matched to their corresponding points on the image. Generally 5 or 6 points are required for a good match.

Add point(s) Defining <node : point> pairs for matching.

In the (artificial) example below the green image on the left has been read in as a background image, and the task is to get the red analysis image on the right to lie on top of it.

The user has defined 3 points so far: the nodes, identified by yellow pick symbols on the right, correspond to their matching points (red symbols and labels) on the left; the blue line shows which points and nodes are associated. These are screen-picked by selecting first the node, and then the corresponding point, and so on for the next pair.

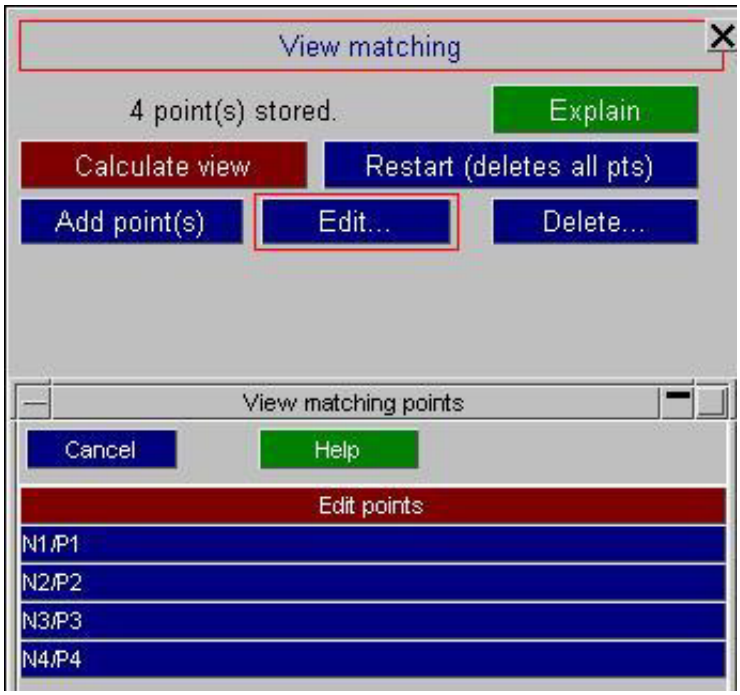


Calculate: aligning analysis with image.

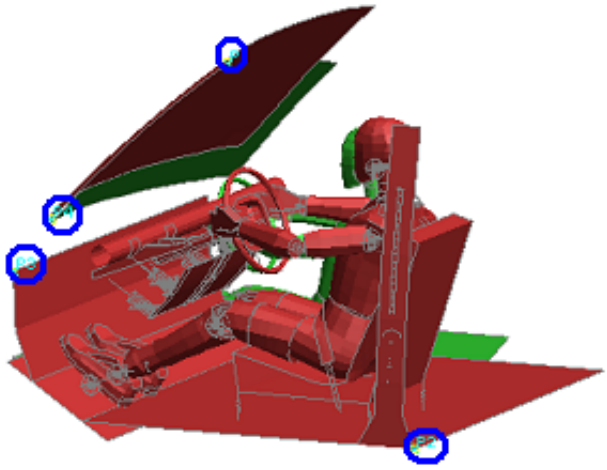
Once four or more <node : point> pairs have been defined it is possible to calculate the revised view. This will calculate the revised viewing parameters and update the image immediately. If the images can be matched and the points have been well chosen then the analysis should lie exactly over the target image.

Edit...: correcting poorly chosen points.

In the example below points have deliberately been chosen badly to obtain a poor match. (The error here is choosing points, ringed in blue, that lie more or less in a plane, making it difficult to calculate perspective distance correctly. In addition choosing only four points is often inadequate, and more can be required for a good solution.)



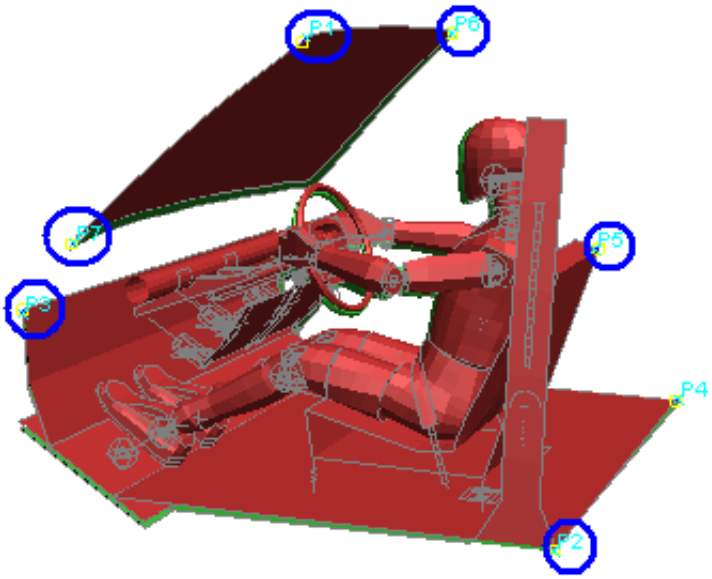
To edit a point screen-pick either its node or point (or select it from the menu), then repick its node or point.



Delete and **Restart**: Deleting points.

Delete allows you to delete individual points by selecting them as above. Each point is deleted immediately.
Restart deletes all points letting you make a fresh start.

You can **Add**, **Edit** and **Delete** points in any order. Here is the example above with 6 points (circled in blue) chosen rather more judiciously, and it can be seen that the correspondence is now very good.

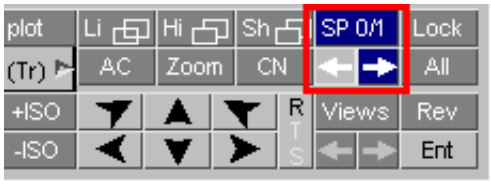


What is stored for matching.

"Node" data is stored as a reference to a node in a model.

"Point" data is stored as a parametric (x,y) screen space coordinate, so points will remain valid so long as the aspect ratio of the window remains the same. However in most cases if a window is resized it is best to delete all the points and start again if further matching is required.

9.6 Saved properties



Saving and restoring the current view, colour, transparency and other attributes controlling the appearance of the image.

Saved properties were added in release 11 and they perform the following functions:

- All the attributes controlling the appearance of the plot are recorded whenever a property is saved using **Save P**. The attributes stored are:
    - Colour, transparency, plotting mode and blanking status of all items in the selected model
    - All settings in the Entity panel, ie visibility and labelling switches
    - The current view parameters: scale, orientation, position, perspective.
- Any number of properties can be saved in memory in PRIMER, and you can scroll backwards and fowards through them using the **<=** and **=>** buttons.
- The attributes reset whenever a saved property is made current are controllable, and notably the current view is ***not*** restored by default.
- Properties can be saved to file (extension .prp). This is an ASCII (human readable) file, written in a format that makes it portable between programmes, notably between PRIMER and D3PLOT, but others too if desired, making it possible to achieve the same image appearance in different programmes.
- Although the colour, transparency, display mode and blanking status are stored with respect to the items in the source model, reuse of the properties file is not limited to this model and it can be used to set properties on any model that shares similar contents and label ranges.

There is some overlap of capabilities between the ability to toggle between and save "Views", and the ability to include the current view in a saved property. This is an historical accident due to the way the software has developed, and the while saved properties always contain view information the default is ***not*** to apply this by default when a property is restored.

9.6.1 **Save P** Saving the current attributes as a "property"

Initially PRIMER has no properties saved, so the saving button will show **Save P**

Once you click on it to save a property it will be updated to be **SP i/j** where

i is the current property number

j is the current total number of saved properties

You can still click on the renamed **SP i/j** to save further properties.

Cycling through saved properties using ⬅ and ➡

Once you have saved one or more properties you can use the **<=** and **=>** buttons to cycle between them. Cycling left (**<=**) reduces the property number, and right (**=>**) increases it. It is possible to cycle backwards (left) to current property 0, which is explained below.

Property number 0, the "current" property

PRIMER always maintains a "current" property which is what you see on the screen, and this is given the special number 0.

When you navigate to a saved property **i** this effectively copies that saved set of attributes to the current one, and likewise whenever you save a property you make a copy of the current property 0.

If you have navigated to a saved property **i** and you subsequently do something which changes the appearance of the image on the screen, for example blanking something, then the current property number gets reset to 0, and the **SP i/j** button will be updated to show **SP 0/j**.

This is because the current property no longer matches the saved property **i**, so it is no longer true to say that you are at property **i**. (The saved property **i** is not affected by this change: remember that making a saved property current copies it to current property 0, and it is only this current property that has been updated.)

9.6.2 **Options**: managing saved properties

Hovering the cursor over the **Save P** (or **SP i/j**) button maps the `Save Props` popup in which you can select **Options** to control saved properties.

This panel lists the current saved properties status, in this example currently at state 2 of 2, and allows you to select a saved property state directly by number.

Clear all saved properties deletes all saved properties in memory.

Saved Attributes lets you control which components of a property are updated when you navigate to a saved property. All attributes are always ***saved***, this controls what is updated when the property is ***restored***.

- **Blanking, Colour, Transparency** and **Plotting mode** are all attributes of the items in a model.

  A saved property always contains ***all*** these attributes for all items in a model, regardless of whether or not they are currently visible. If items are added to the model after the property was saved their attributes will not be stored, since they weren't known about at the time of saving, so they will not be updated when the property is restored. (See below for further notes on the effects of changing model contents.)

- **Entity and Label switches** are the settings in the Entity panel and are model independent. A saved property contains the current status of all such switches for all possible item types, whether or not they are present in a given model.

- **Viewing parameters** are also model independent. The scale, orientation, location and perspective settings are stored. (This setting is ***not*** selected by default.)

The effect on a saved property of changing model contents.

In order to be economical of memory the model-dependent data in a saved property is stored in "runlength encoded" form. This utilises the fact that in most models a sequence of items will all have the same visual attributes, for example all elements in a part will tend to have the same colour, transparency, etc. Therefore for a given item the property data each row is saved (qualitatively) as:

Label A	**Label B**	**Property information**

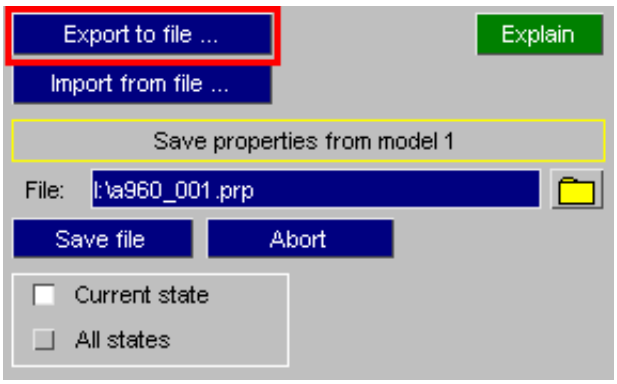
All items lying in the label range A to B implicitly have the same Blanking, Colour, Transparency and Display mode.

Therefore when the model is changed the following happens:

Change made to model	**Consequence when a saved property is restored**
Items in the range A to B are deleted	If the deleted items lie within this range, but do not include either A or B themselves, then there is no effect. If either item A or B are deleted then the whole of this "row" of properties is lost, and on a subsequent restore items within this range will not be updated.
New items are added	Items added within an existing label range A to B will automatically be included within that range, and their properties will be updated on a restore. Items outside any saved range will not be updated since they were not known about when the property was saved. Where new items overlap an existing A .. B range then only the subset lying within that range will be updated.
Existing items are renumbered	Internally PRIMER stores the item "object" rather than just its label, so renumbering objects automatically updates their label ranges in saved property rows of data. Therefore if existing label range A .. B is modifed to A’ .. B’ all items within that new range will be updated as before, but this might or might not include the same range of items. If the new range is invalid, for example B’ is now less than A’, then no update will take place.

It will be clear that the logic above is not perfect, and that changing a model may well invalidate some or all saved properties. This is an inevitable consequence of the compact storage method used, and is considered to be a reasonable compromise. Storing data in a "change resistant" form would require many orders of magnitude more memory, and this could have unacceptable side-effects on the performance of the code.

Export to file... saving properties to file

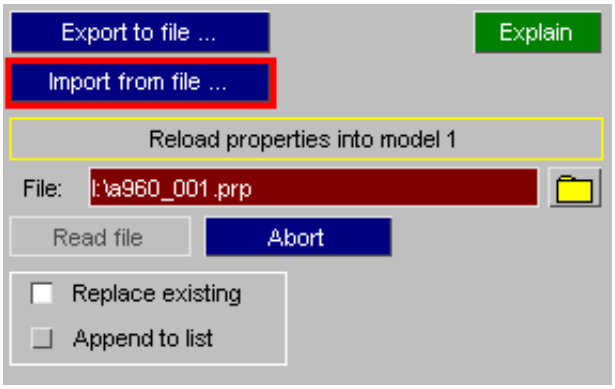


The complete contents of either the current property state 0 only, or all saved property states, can be saved to an external properties file.

This is an ASCII (human-readable) file that is designed to be both programme and model independent, and its format is given below.

The next free filename in the sequence **`<jobname>_nnn.prp`** will be presented as the default name, but you are free to use any name. Extension "**`.prp`**" is recommended though for compatibility with other Oasys Ltd. LS-DYNA Environment software.

Import from file... reloading properties from file



A previously saved file of properties can be reloaded into memory in this PRIMER session, either replacing any existing properties or appending them to the current list.

The most recently created file in the sequence **`<jobname>_nnn.prp`** will be presented as the default filename and if, as here, no such file exists it will be listed on a red background and you will have to specify an alternative.

9.6.3 The format of the saved properties (.prp) file

The saved properties file (.prp) is intended to be both programme-independent and model independent, so that attributes of a model's appearance can be shared between different programmes and variants of the same model. In particular the properties file can be shared between PRIMER and D3PLOT.

The file format is ASCII, so it is human readable and can be manipulated in a text editor, and its format is similar to LS-DYNA keyword format in that:

- Each data blocks begins with a "Keywords" that have an asterisk **`*`** in column 1.
- Any number of comment lines may be inserted, and they start with **`$`**, **`%`** or **`#`** in column 1.

However one significant difference is that all data is in free format, with no restrictions on field width or spacing between columns of data, so data will be formatted as:

string	a string of some number of characters
integer	either a decimal number, or a hexadecimal one if it starts "0x..."
float	a floating point number

A properties file contains the following blocks:

Header name	Status	Description	Notes
`*PROPERTIES`	Required	Defines the parameters of the following property state	This sequence of blocks is repeated for each saved property.
`*PROP_MASKS`	Required	Describes the format of the data to follow	
`*PROP_DATA`	Required	Contains the actual property data for model items	
`*PROP_SWITCHES`	*Optional*	Contains information about "entity" panel settings	
`*PROP_VIEW`	*Optional*	Contains information about the current view settings	
`*PROP_END`	Required	Acts as an "end of property definition" marker	

Each block is described in more detail below.

For each saved property data blocks should appear in the following order:

```
*PROPERTIES
<code>     <file version>
<saved id> <title>
```

<code>	string	is the programme name, here PRIMER
<file version>	integer	is the version number of this file. This commences at 0 for release 11.
<saved id>	integer	is the saved property id, starting at 0 for "current".
<title>	string	is an optional title. At present this will be ignored.

This header block describes the basic parameters of the new saved property entry.

```
*PROP_MASKS
row 1: <keyword>   <word>   <data mask>
row 2: <keyword>   <word>   <data mask>
  :        :          :          :
row n: <keyword>   <word>   <data mask>
```

<keyword>	string	One of a known series of mask names.
<column>	integer	The column number on the line, starting at 1.
<mask>	integer	Integer or hexadecimal value giving bits used.

The purpose of this block is to allow different programmes, which will almost certainly store information in different formats, to stipulate how they are presenting data, and also to specify how many columns (words) of data will be supplied in the *PROP_DATA block below.

You don't need to understand this block unless you plan to generate property files yourself, or to read PRIMER-generated property files into some other software. If this is the case please see "More about *PROP_MASKS" below.

```
*PROP_DATA
row 1: <item type>  <start label> <end label>  <word #1> <word #2> ... <word
#n>
row 2: <item type>  <start label> <end label>  <word #1> <word #2> ... <word
#n>
  :          :              :            :             :          :              :
row n: <item type>  <start label> <end label>  <word #1> <word #2> ... <word
#n>
```

<item type>	string	Item name, eg NODE, PART, etc
<start label>	integer	The first label in the range, or FIRST or ALL
<end label>	integer	The end label in the range, or LAST. Omitted if the start label is ALL.
<word #1>	integer	The first word of data, ie column 1
<word #2>	integer	The second word of data, ie column 2
<word #n>	integer	The last word of data, ie column n

The storage method here echoes the internal runlength-encoded format in which all items in the label range <start> ... <end> have the same property values.

ALL is used instead of <start> .. <end> labels when all items of the type share the same attributes.

FIRST is used in place of label <start> if this is the first item of its type, and **LAST** in place of label <end> if it is the last label. This is so that other models, perhaps with slightly different label ranges, will still apply the properties correctly.

Data words #1 to #n must be supplied for every item even if they do not contain any useful data, in which case they can be zero. The number of words expected on each line, #n, is inferred from the highest <column> entry in the preceding ***PROP_MASKS** block.

```
*PROP_SWITCHES
row 1: <item type>   <drawn>  <labelled>  <named>
row 2: <item type>   <drawn>  <labelled>  <named>
            :           :          :         :
row n: <item type>   <drawn>  <labelled>  <named>
```

<item type>	string	Item name, eg NODE, PART, etc
<drawn>	integer	Whether this item is drawn
<labelled>	integer	Whether this item is labelled
<named>	integer	Whether this item is named

This data block is optional: if omitted the "entity" panel settings will be left unchanged when the file is read.

Each data field <drawn>, <labelled>, <named> is, at its simplest, 1 for true and 0 for false. However within PRIMER some item types have sub-keywords, and further bits can be used to denote the individual status of these.

```
*PROP_VIEW
Matrix row 1: <X cosine>  <Y cosine>  <Z cosine>
Matrix row 2: <X cosine>  <Y cosine>  <Z cosine>
Matrix row 3: <X cosine>  <Y cosine>  <Z cosine>
Offsets:      <X trans>   <Y trans>   <Z trans>
Scale:        <Scale factor>
Perspective:  <On/off>    <Distance>
```

<X/Y/Z cosine>	float	The X/Y/Z components of the unit cosines for that matrix row
<X/Y/Z trans>	float	The X/Y/Z component of the translations required to position the model in front of the eye position.
<Scale>	float	The scale factor from model space to screen (4096 x 4096) space
<On/off>	integer	Whether perspective is on (1) or off (0)
<Distance>	float	The perspective distance (from eye position to model centre)

This data block is optional. If it is omitted the view will not be updated when the file is read.

***PROP_END**
(This block has no data)

This block signifies the end of the current property definition.

Here is an example properties file from a small model.

```
$ File J:\sled_model_binout\new_lg09_004.prp written at Tue Dec 06 15:40:43 2011
$
$ PRIMER Version : 11.0
$ File Version : 0
$
*PROPERTIES
$
$ Code File version
PRIMER 0
$ State id Title
0
$
*PROP_MASKS
$
$ Attribute Word Bits
$ ---------------------------
BLANKED 1 0x1
$
ALPHA_MASK 2 0x78000000
RED_MASK 2 0x78000
GREEN_MASK 2 0x780000
BLUE_MASK 2 0x7800000
INDEX_MASK 2 0xf
```

```
FIXED_BIT 2 0x10
MODE_MASK 2 0x60
BRIGHT_MASK 2 0x7800
SHINE_MASK 2 0x780
SPECIAL_MASK 2 0x8000000f
BY_MODEL 2 0x80000001
BY_IFILE 2 0x80000002
BY_SUBSET 2 0x80000003
BY_PART 2 0x80000004
BY_SECTION 2 0x80000005
BY_MATERIAL 2 0x80000006
$
$
*PROP_DATA
$
$ Type Label #1 Label #2 Word #1 Word #2
$
-------------------------------------------------------------------------------------
$
NODE ALL 0 0x7fff9ff0
BEAM ALL 0 0x7f879ff0
SHELL FIRST 64 0 0x78079ff0
SHELL 65 128 0 0xfad61fe4
SHELL 129 256 0 0x7f801f90
SHELL 257 328 0 0x7ff81ff0
SHELL 329 392 0 0x787f9ff0
SHELL 393 456 0 0x7c879ff0
SHELL 457 520 0 0x785f9ff0
SHELL 521 584 0 0x787d1ff0
SHELL 585 LAST 0 0x37f81ff0
DISCRETE FIRST 1 0 0x787d1ff0
DISCRETE 2 2 0 0x7d781ff0
DISCRETE 10000097 LAST 0 0x7f879ff0
SEATBELT ALL 0 0xf83f9fe4
ACCELEROMETER FIRST 1 0 0x78079fe2
ACCELEROMETER 2 2 0 0x78781fe3
ACCELEROMETER 3 LAST 0 0x7f801fe4
PRETENSIONER ALL 0 0x787f9ff7
RETRACTOR ALL 0 0x7ff81ff5
SENSOR ALL 0 0x7f879ff6
SLIPRING ALL 0 0x78781ff3
SEGMENT FIRST 1 0 0x783f9fe2
SEGMENT 488 491 0 0x7fb81fe8
SEGMENT 492 1079 0 0x78079fe4
SEGMENT 1080 LAST 0 0x7f879fe2
SET_NODE FIRST 1 0 0x78079fe2
SET_NODE 2 2 0 0x78781fe3
SET_NODE 3 LAST 0 0x7f801fe4
SET_SEGMENT FIRST 1 0 0x78079fe2
SET_SEGMENT 2 2 0 0x78781fe3
SET_SEGMENT 3 LAST 0 0x783f9fed
PART FIRST 1 0 0x78079ff0
PART 2 2 0 0x7ad61ff0
PART 3 3 0 0x7f801f90
PART 4 4 0 0x7ff81ff0
PART 5 5 0 0x7f879ff0
PART 1001 1001 0 0x7fd81ff0
PART 1002 1002 0 0x78079ff0
PART 1003 1003 0 0x78781ff0
PART 1004 1004 0 0x7f801ff0
PART 2001 LAST 0 0x783f9ff0
MATERIAL FIRST 1 0 0x78079fe2
MATERIAL 2 2 0 0x78781fe3
MATERIAL 3 3 0 0x7f801fe4
MATERIAL 4 4 0 0x7ff81fe5
MATERIAL 5 5 0 0x7f879fe6
MATERIAL 1001 1001 0 0x7fd81fee
MATERIAL 1002 1002 0 0x78079fe2
MATERIAL 1003 1003 0 0x78781fe3
MATERIAL 1004 1004 0 0x7f801fe4
MATERIAL 1005 1005 0 0x7ff81fe5
```

```
MATERIAL 2001 LAST 0 0x783f9fed
SECTION FIRST 1 0 0x78079fe2
SECTION 2 2 0 0x78781fe3
SECTION 3 3 0 0x7f801fe4
SECTION 4 4 0 0x7ff81fe5
SECTION 5 5 0 0x7f879fe6
SECTION 1001 1001 0 0x7fd81fee
SECTION 1002 1002 0 0x78079fe2
SECTION 1003 1003 0 0x78781fe3
SECTION 1004 1004 0 0x7f801fe4
SECTION 1005 1005 0 0x7ff81fe5
SECTION 2001 LAST 0 0x787d1fea
PRESCRIBED_MOTION 2 LAST 0 0x7ff81fe3
SPC ALL 0 0x7fff9fe2
CONSTRAINED FIRST 1 0 0x78079fe2
CONSTRAINED 2 2 0 0x7ad61fe3
CONSTRAINED 3 3 0 0x7f801fe4
CONSTRAINED 4 4 0 0x7ff81fe5
CONSTRAINED 5 LAST 0 0x785f9fe7
JOINT FIRST 14 0 0x78079ff0
JOINT 15 LAST 0 0x7f801ff0
CONTACT ALL 0 0xffff9fe7
DATABASE_HISTORY ALL 0 0x78079ff2
DEFINE_COORDINATE ALL 0 0x78079fe2
DEFINE_SD_ORIENTATION FIRST 1 0 0x78079fe2
DEFINE_SD_ORIENTATION 2 LAST 0 0x7f801fe4
$
$
*PROP_SWITCHES
$
$ Entity type switches Drawn Labels Names
$ ---------------------------------------------------------------------------
$
NODE 0 0 0
BEAM 0x1 0 0
SHELL 0x1 0 0
DISCRETE 0x1 0 0
SEATBELT 0x1 0 0
ACCELEROMETER 0x1 0 0
PRETENSIONER 0x1 0 0
RETRACTOR 0x1 0 0
SENSOR 0x1 0 0
SLIPRING 0x1 0 0
SEGMENT 0 0 0
SET_NODE 0 0 0
SET_SEGMENT 0 0 0
PART 0 0 0
MATERIAL 0 0 0
SECTION 0 0 0
PRESCRIBED_MOTION 0 0 0
SPC 0 0 0
CONSTRAINED 0 0 0
JOINT 0x1 0 0
CONTACT 0 0 0
DATABASE_HISTORY 0 0 0
DEFINE_COORDINATE 0 0 0
DEFINE_CURVE 0 0 0
DEFINE_SD_ORIENTATION 0 0 0
$
$
*PROP_VIEW
$
$ Current viewing attributes
$ --------------------------
$
Matrix row 1: 8.769390E-001 -4.765534E-001 -6.238726E-002
Matrix row 2: 1.884609E-001 2.215460E-001 9.567727E-001
Matrix row 3: -4.421267E-001 -8.507872E-001 2.840920E-001
Offsets: -2.400639E+002 -1.826899E+001 1.233605E+002
Scale: 1.391819E+000
Perspective: 0 4.503000E+003
```

```
$
$
*PROP_END
$
$
$ End of file
```

More about the *PROP_MASKS block

You only need to understand property "masks" if you plan to create your own property files, or to read the PRIMER-generated ones into 3rd party software.

A "mask" defines the bits in a word that are used to contain data. In this context a "word" is always a single precision 32 bit integer, so you will be defining which of these 32 bits contain the data you want.

As an example let us take the problem of defining colour, which is specified by 4 components, generally known as RGBA in computer graphics:

Component	Property mask	Description
Red	**RED_MASK**	For each of red, green and blue the value must be in the range 0 to 100%
Green	**GREEN_MASK**	
Blue	**BLUE_MASK**	
Alpha (transparency)	**ALPHA_MASK**	A value must lie in the range 0% (fully transparent) to 100% (fully opaque)

Therefore bright red, with no transparency, would comprise 100% Red, 0% Green, 0% Blue, 100% Alpha.

Example 1: External data contains each colour component as a separate floating point value in the range 0.0 to 100.0

In this case the easiest solution would be to express your colours as 4 separate values. These must be integers, and the full bit field must imply 100%, so the easiest solution would be to convert the floating point range 0.0 to 100.0 into values in the range 0 to 255 by multiplying by 2.55 and writing the result as integers. The data masks you define might then be:

RED_MASK 1 255 **GREEN_MASK 2 255** **BLUE_MASK 3 255** **ALPHA_MASK 4 255**	Each colour channel is defined in a separate integer word Red = word #1, Green = word #2, Blue = word #3, Alpha = word #4 and lies in the range 0 - 255

And a typical property line to define some shells with labels 1 to 10 that are cyan (green + blue) and 50% transparent would then be

Item name	Start label	End label	Word 1: Red value	W2: Green value	W3: Blue value	W4: Alpha value	.. further columns
SHELL	**1**	**10**	**0**	**255**	**255**	**128**	**...**

The choice of columns 1 to 4 for the RGBA components is arbitrary, you could choose any columns you like.

Example 2: External data contains each colour component packed in a single 32 bit word

A more compact, and very common, way of storing RGBA data is to express each colour component in the range 0 - 255, which requires 8 bits or 1 byte, and to pack these four bytes into a single 32 bit word. Drawn as a diagram we could express the 32 bits in this word as:

Highest byte: Alpha bits	Blue bits	Green bits	Lowest byte: red bits

AAAAAAAA	**BBBBBBBB**	**GGGGGGGG**	**RRRRRRRR**

We can now define our colour masks, assuming that the colour word is in column #1, as

```
RED_MASK       1      0x000000ff
GREEN_MASK     1      0x0000ff00
BLUE_MASK      1      0x00ff0000
ALPHA_MASK     1      0xff000000
```

Hexadecimal (0x...) format has been used here, but the values could equally well - if less conveniently - be expressed in decimal. For example the Red mask **0x000000ff** is the same as decimal **255**, and it would be legal to use that instead. Using this format our 50% transparent cyan shells would now be defined more compactly as:

Item name	Start label	End label	Word 1: RGBA	.. further columns
SHELL	**1**	**10**	**0x80ffff00**	**...**

Again hexadecimal has been used here, since the decimal equivalent would be an unwieldy negative number.

What property masks are required?

You only have to provide property masks for the values you want to change. When property files are read in they only overwrite the attributes that they define so, for example, if you only included blanking information in a file the colour and lighting attributes of the model would be unchanged when it was read. Another example might be that you only have RGB colour information, and no Alpha (transparency) data. In that case omitting the Alpha mask and data word would leave item transparency unchanged when a file is read.

Which columns may data occupy?

Up to 20 columns of data may be provided, numbered 1 to 20, and any attribute may exist in any column. When the *PROP_MASKS data block is read the highest column number is remembered and the subsequent *PROP_DATA block must contain that many columns of data on each line. It doesn't matter if data in a given column is not read, for example if you already have formatted data and you want to ignore some of it simply define masks that only specify the data you want.

Valid property masks for PRIMER:

Mask name	**Meaning**	
BLANKED	The bit(s) used to designate that an item is blanked, ie blanked (non-zero) or unblanked (zero)	
MODE_MASK	The display mode: 0= wireframe, 1 = hidden, 2 = shaded, 3 = current	
ALPHA_MASK	The Alpha (transparency) bits. 100% = opaque.	It is assumed that a fully occupied bit field is 100% of the given component value for all these types
RED_MASK	The Red bits	
GREEN_MASK	The Green bits	
BLUE_MASK	The Blue bits	
BRIGHT_MASK	The diffuse brightness	At present lighting is only controllable on a programme-wide basis in PRIMER, so these fields are written but not read. However other codes, notably D3PLOT, can read them to obtain lighting data. As with colour a fully occupied bit field is assumed to be a value of 100%
SHINE_MASK	The specular brightness (shininess)	

The following are PRIMER-specific and reflect its internal storage of colour. External programmes would not normally use these, and can ignore them. They are included here for completeness.	
INDEX_MASK	The bits set aside for indexed colours in the sequence Black (0), white (1), ... grey (15)
FIXED_BIT	This bit is set if the item has a fixed colour as opposed to an inherited one using a code below
SPECIAL_MASK	These bits define the mask for "inherited" colour codes below
BY_MODEL	Colour is based on model id
BY_IFILE	Colour is based on include file id
BY_SUBSET	Colour is based on subset (part tree assembly) id
BY_PART	Colour is based on the parent part's colour
BY_SECTION	Colour is based on the parent section colour
BY_MATERIAL	Colour is based on the parent material colour

9.7 Accelerated Graphics

From release 11 onwards PRIMER graphics have been rewritten to make more use of the capabilities of modern graphics cards. Broadly speaking the data to be displayed can be stored in memory on the graphics card itself, and also more of the work of rendering graphics can be "handed over" to the graphics card as well. Since the card has the information stored locally there is no need to transfer data from main memory each time a scene is redrawn, making redisplay during dynamic viewing much faster.

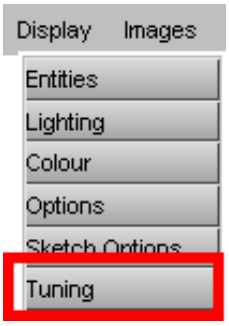
If it all goes horribly wrong ... PRIMER crashes when it reads any model

In rare cases, generally on an old machine or when the graphics driver is out of date, PRIMER 11 may crash when it first reads a model because it attempts to display it and the graphics go wrong. This can happen if the graphics card purports to offer "modern" features, but they don't work properly. If you get this behaviour it will be necessary to switch off accelerated graphics, reverting to pre-version 11 behaviour, which can be done by setting environment variable

PRIMER_NO_VARRAY to **true**

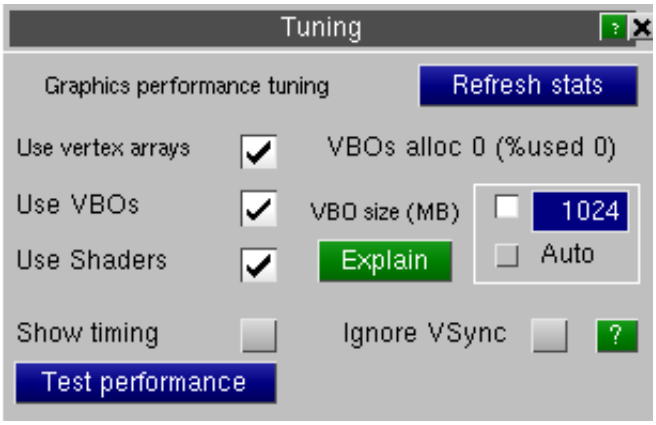
If this happens please seek advice from Oasys Ltd, but we will almost certainly suggest that as a first step you update your graphics driver.

9.7.1 Tuning accelerated graphics.



PRIMER detects the attributes of your graphics card automatically, determines what features are available, and automatically switches on the best accelerations that should be possible as a result. Therefore accelerated graphics should just happen with no intervention from the user being required.

However in the real world not all hardware behaves as it should, graphics drivers can be out of date, and things can go wrong. Therefore a **Tuning** panel is provided to enable you to control acceleration features. This is invoked from the **Options > Tuning** popup menu.



It is beyond the scope of this manual to explain the details of graphics acceleration. However if graphics are giving problems you should use the following procedure:

1. Turn off all three settings in the left hand column: **Use vertex arrays**, **Use VBOs**, **Use Shaders**. This will cause graphics to revert to the traditional "immediate mode" used prior to release 11.

2. Read in a reasonably large model that takes an appreciable amount of time to render on your hardware, autoscale it and use **Test Performance** to see how it performs. This will spin it round 360 degrees and report the time taken. Alternatively to measure the time taken by ordinary graphics operations turn on **Show Timing**, which will report times to the dialogue box for all drawing operations.

3. You will see two timing figures, expressed in milliseconds, displayed in the dialogue box each time the image is updated. The first is the time taken to render this frame, and the second is the average time taken to render the last 30 frames. Because of the "granularity" of system timers the average time is a better indication of performance.

Then use the following steps to try to determine what permutation of options will work on your machine. If any options are greyed out then your graphics card does not support them.

1. Turn on **Use vertex arrays** and use **Test Performance** again. If it all goes horribly wrong then proceed to "upgrading your graphics driver" below, but hopefully you will see a speed improvement. However if vertex arrays will not work there is no point in continuing to steps 5 and 6 below.

2. Turn on **Use VBOs** and repeat the test. Again, things should speed up, but if they don't you may have to accept that Vertex Buffer Objects (VBOs) do not function correctly on your machine.

3. Turn on **Use Shaders** and repeat the test. This should speed up shaded plots, but it will have no impact on line or hidden mode plots. If it doesn't work, or goes wrong, then shaders cannot be used on your machine.

Hopefully some permutation of the above will give a speed-up and acceptable graphics. Once you have determined what works use **Save Tuning Settings** to save the settings to your oa_pref file for use in future PRIMER sessions. You can revisit this tuning panel as often as you like, typically when you have installed a new graphics driver, to repeat the process above and find out if some different settings work better.

Ignore VSync tells the graphics card not to wait for the vertical refresh signal from your monitor before rendering the next frame. When frame rates get close to this refresh rate, typically 60Hz (~16mS/frame), ignoring this signal will give a slight improvement in performance, and for small models it may drop below this rate. However if this setting makes much difference you really need to use a bigger model otherwise you will be measuring factors other than just graphics performance.

It can pay to upgrade your graphics driver periodically as described in section 9.7.2 below, and also to configure it correctly as described in section 9.7.3.

Setting VBO (Vertex Buffer Object) attributes (advanced topic)

This is an advanced topic, and you don't have to concern yourself with these settings unless you are experiencing graphics memory problems.

A typical shell model will use about 55MBytes of graphics memory per million elements, and a typical modern graphics card will have at least 1GByte of video memory, so problems are only likely to arise it you are processing huge models or you have a very low-spec graphics card.

Vertex Buffer Objects (VBOs) are area of memory on the graphics card itself in which graphics data (vertex information) can be stored. The advantage of this is that when it comes to redrawing, for example during dynamic rotation, the graphics card has the necessary data immediately to hand and does not have to wait for it to be delivered

from main memory. This can speed up redrawing speed dramatically, often by a factor of 3x or more, so its use is worthwhile and PRIMER makes as much use of it as it can.

If this resource is exhausted while creating a plot then PRIMER will revert to storing subsequent graphics in main memory, and sending them to the graphics card each time the image is redrawn. This transition is seamless, and a plot may contain any permutation of "on card memory" and "in main memory" graphical data, however reversion to main memory will slow down performance so it is worth trying to exploit VBOs to their utmost.

PRIMER allocates VBO space in "chunks" of 128MBytes at a time, and reports in this panel how many such chunks have been allocated together with the %age of space within them that is currently being used. You can use this as a guide to how your graphics hardware is coping with the loads placed upon it.

The space available for VBO use is actually greater than the amount of memory on the graphics card, since the card can transfer data to and from dedicated main memory. However it is still finite, and has to be shared with other applications on the desktop, so its use has to be managed and this done via the **VBO Size**

The **VBO Size** will have one of the following settings:

Explicit Size (MB)

This is the default. PRIMER will attempt to determine how much video memory there is in on your graphics card and set the limit to that.

This is not possible on all cards, see below, and if the limit cannot be determined then **Not set** will be shown and no checking will take place. In this situation you can type in a sensible value (in MBytes) and this value will then be used for checking.

As stated above video memory is not limited purely to the memory physically present on the graphics card itself. The graphics card driver will also allocate a (potential) region of main memory and use this as an "overflow" if required, swapping data between card and main memory as required.

It can do this much faster than the application can transfer conventional graphics information from main memory, so it is generally faster to swamp the graphics card with data and let it manage it, rather than trying to manage graphics memory in the application itself. However taken to extremes this will crash the application, so default adopted is to use the total amount of physical memory on the card.

You might - rarely - want to ***increase*** this value if, and only if, all the following are true:
- You can see that the number of VBO chunks allocated x 128 = the limiting size in MBytes
- The %age in use is 100%
- Graphics seems slow

You might want to ***decrease*** this value if:
- The application has crashed with a warning from the graphics driver that it is out of memory.
- Graphics seems to be "jerky", and other graphical applications seem to be suffering when PRIMER is running.

However in this situation you may prefer to use the **Automatic** option.

This option will only be available if the graphics card / OpenGL installation permits it - see below.

In this case PRIMER will only allocate a further "chunk" of VBO memory if interrogation of the graphics card reveals that spare space is available.

This is "nice" to the graphics card, and to any other graphical applications competing for memory on the card, but it tends to result in quite a conservative usage of VBO space, resulting in slower performance once available memory is exhausted. It is useful as a "quick fix" if you have memory problems, but it will result in sub-standard graphics speed for very large models.

The process is dynamic, so a solitary PRIMER session might manage to obtain more VBO space than two or more such sessions running concurrently, or a session competing for resources with other graphical software.

If you **Save Tuning Settings** the mode and any value will be saved, and restored next time you run PRIMER.

PRIMER can determine the amount of memory available from the following hardware / software combinations:
- NVidia cards from OpenGL 2.0 onwards
- ATI cards from OpenGL 1.5 onwards

This should cover the vast majority of high performance desktop machines, but on laptops with "Integrated" graphics and older machines it may not be possible to determine graphics memory parameters.

This is a complex topic. If you experience graphics problems that you think are memory related it may be best to

contact Oasys ltd, since there are further diagnostics that can be turned on which may give more insight into what is going wrong.

9.7.2 Upgrading your graphics driver.

If you have problems with accelerated graphics, or not all options are available, or just if your graphics driver has not been updated for a while it is worth upgrading it to see if this fixes the problem. To do this you need to find out what graphics card and driver you have installed, then download the correct new driver for this card / operating system combination.

Finding out what graphics card and driver you have installed.

The following instructions should enable you to determine the type of graphics card you have installed and the revision number of its driver.

Windows XP

- Right click anywhere on the desktop background, and select **`Properties`**
- Select the **`Settings`** tab, then select **`Advanced`**
- Select the **`Adapter`** tab, and the **`Adapter type`** gives you your card name and manufacturer
- Select **`Properties`** within this section, followed by the **`Driver`** tab
- This will list the driver date and version.

Windows Vista and 7

- Right click anywhere on the desktop background, and select **`Screen Resolution`**
- Select **`Advanced settings`**
- This takes you to the **`Adapter`** window, listing card name and manufacturer
- Select **`Properties`** within this section, followed by the **`Driver`** tab
- This will list the driver date and version

Linux

- type **`glxinfo | grep -i string`** which should give the card manufacturer and name

  For example on a machine with an ATI card this produces:

  ```
  OpenGL vendor string: ATI Technologies Inc.
  OpenGL renderer string: ATI FirePro V7750 (FireGL)
  OpenGL version string: 3.3.10225 Compatibility Profile Context FireGL
  ```

  And on a machine with an NVidia card this produces:

  ```
  OpenGL vendor string: NVIDIA Corporation
  OpenGL renderer string: Quadro FX 3800/PCI/SSE2
  OpenGL version string: 3.3.0 NVIDIA 256.35
  ```
- Knowing the make of card you can then look in file /var/log/Xorg.0.log for more details. For example in the 2nd example above
  `grep -i nvidia /var/log/Xorg.0.log | grep -i driver` gives:

  ```
  (II) Loading /usr/lib64/xorg/modules/drivers/nvidia_drv.so
  (II) NVIDIA dlloader X Driver 256.35 Wed Jun 16 18:45:02 PDT 2010
  (II) NVIDIA Unified Driver for all Supported NVIDIA GPUs
  ```

  So this 2nd machine has an NVidia Quadro FX3800 card using driver release 256.35 dated June 16th 2010

Installing a new graphics driver.

Go to the appropriate manufacturer’s website, find a "driver downloads" link, and fill in the fields giving card and operating system type. This will the find the appropriate driver for you, and if this post-dates your existing one it is recommended that you download and install the new one.

Windows:

Simply download and "run" the executable, following the instructions the installer gives. You will probably need

to reboot the system when it has finished.

Linux:

You will need to be running as user "root". Download the install package, then shut down the X server by moving to single user mode (Command "**`init 3`**" to go to runlevel 3, or "**`init 1`**" for runlevel 1.) Unpack the install package if it is a gzip or similar file, then run it using "**`sh <package name>`**" to run the installer and follow the instructions. You will probably need to reboot the system when it is complete.

When you have finished installing a new driver go back to "tuning" above to see if any of the options you had to turn off previously will now work. This may well be the case as newer versions of a driver often fix bugs or provide support for more recent versions of OpenGL that support these features.

If you previously had to set environment variable **`PRIMER_NO_VARRAY`** in order to make things work you need to delete this variable before trying again. (Just setting it to "false" is not enough, it must be deleted.)

9.7.3 Configuring your graphics driver

Graphics "as supplied" from vendors are configured to work best for generic 3D applications, and some configuration is required to get the best settings for Oasys Ltd software.

Configuring NVidia cards on Windows - versions from release 12.1 onwards

From release 12.1 onwards all Oasys ltd software can detect automatically the presence of an NVidia graphics card, and configure the graphics driver settings automatically. You should not need to take any action, and it is no longer necessary to configure the graphics installation on the machine. In fact the best performance is likely to be achieved if you leave the driver set at its base "default 3d app" profile.

In more detail...

(You don’t need to understand this. This section is provided for interest and to explain the above more fully.)

NVidia provide two main ranges of graphics cards:

Quadro	Optimised for OpenGL and typical CAE / Engineering applications
GeForce	Optimised for DirectX, and typical games

Both cards will handle both OpenGL and DirectX, and both can be used for CAE as well as games, but Oasys Ltd recommend that the Quadro range be used for engineering software since it has been developed for that purpose and will give more reliable performance in several respects. We have occasionally encountered problems with the GeForce range in the past.

In order to get the optimum performance from software NVidia test applications and tune their drivers, creating a "profile" for each piece of software that contain application-specific settings. These profiles, and there are over 4000 of them as of April 2015, come pre-configured in all NVidia Windows graphics drivers, and where an executable name (for example `minecraft.exe`) never changes then the driver can detect for itself which executable is being run, and use the correct profile automatically.

However where software names change over time, such as the Oasys Ltd progression of `primer`**`12`**`_x64.exe`, `primer14_x64.exe`, and so on NVidia cannot reasonably be expected to keep updating their drivers to know about the new names, and instead the onus is on the application itself to interrogate and configure the graphics driver.

This can be done via the NVAPI toolkit, and from release 12.1 onwards Oasys Ltd software now able to do this. It looks for the Oasys LS-DYNA profile in the graphics driver and "tells" the driver to use that profile instead of the default one.

This process has been tested on a range of cards and versions of the Windows operating system, and so far no problems have been found. However if problems *do* occur then the following environment variables can be used to modify this process:

Variable name	Possible values	Meaning

OASYS_NV_ACTION	0 or unset (default)	Update driver if required to apply Oasys LS-DYNA profile to this application
	1	List to <stdout> all Oasys Ltd settings in the graphics driver on this machine
	2	Refresh the Oasys Ltd settings on this machine: clear all existing ones and reinstall them
	3	Delete all Oasys Ltd settings on this ,achine
OASYS_NV_VERBOSE	1	By default this is not set, but if it is set then any changes made to the driver are listed to <stdout>
OASYS_NV_NOT_D3PLOT	1	By default this is not set, but if it is set then D3PLOT tuning will not be performed in the driver
OASYS_NV_NOT_PRIMER	1	By default this is not set, but if it is set then PRIMER tuning will not be performed in the driver
OASYS_NV_NOT_THIS	1	By default this is not set, but if it is set then THIS tuning will not be performed in the driver

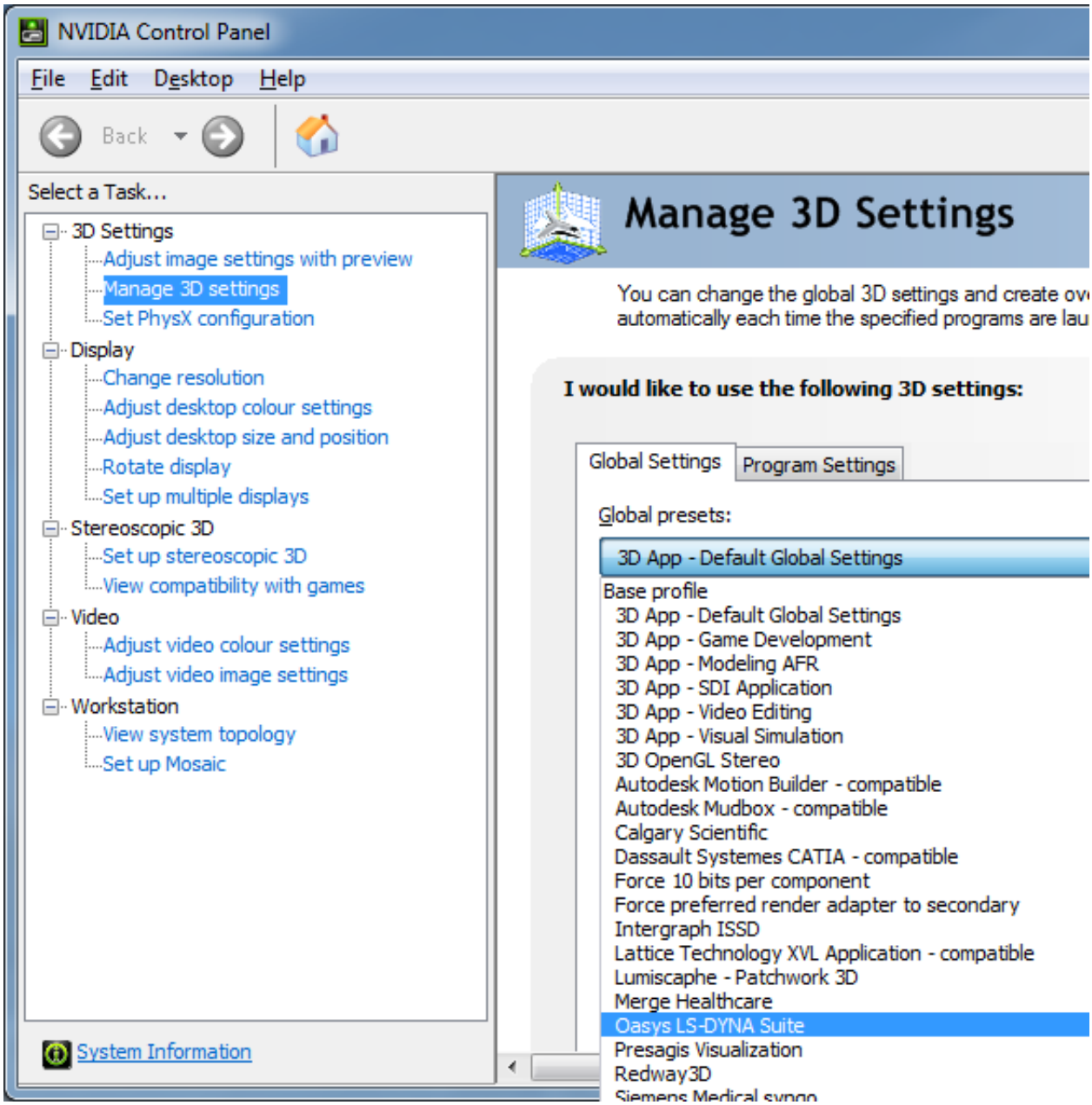
All the above should "just work", but if it doesn’t please contact Oasys Ltd for help.

Configuring NVidia cards on Windows - versions prior to release 12.1

Recent installations:

- Right click anywhere on desktop background, and select **NVIDIA Control Panel:**

  Select **Manage 3D settings** from the tree on the left hand side. The example below is from a Quadro FX card on a Windows 7 machine, but others should be very similar.



- You must then decide whether you want to configure the graphics driver for all applications on your machine or just for a limited range of executables.

  Our recommendation is to configure for all applications, using **`Global settings`** as shown above. The configuration used should work well for any CAE package - and certainly better than NVidia's default "**`3D App -Default Global Settings`**", since these are tuned for benchmark tests and not real life applications.

  If you want to apply settings only to PRIMER you will need to swap to the Program Settings tab, add PRIMER to the list, and then proceed as below.

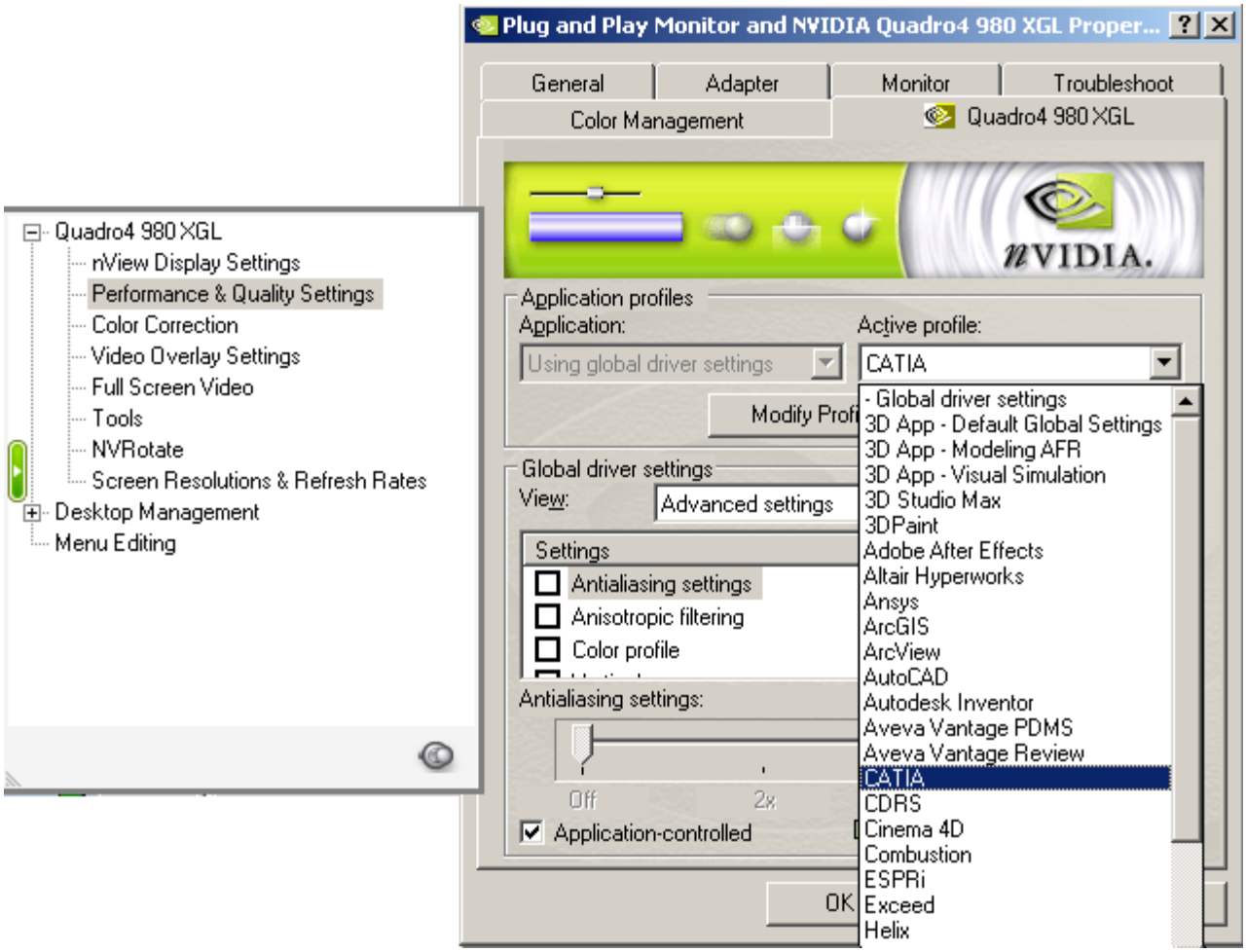
- If your driver is recent (early 2011 onwards) you will find an **`Oasys Ltd. LS-DYNA environment`** setting as shown above, and you should select that. If your driver is older we would recommend using **`Dassault Systemes CATIA - compatible`**.

  Either of these settings turns off attempts in the driver to cache coordinate data, and will result in smooth animation. Using the default settings may lead to jerky animation, or long pauses.

Older installations

- Right click anywhere on the desktop background, and select **`Properties`**
- Select the **`Settings`** tab, then select **`Advanced`**
- Select the tab showing your driver name.

In this example on an old Windows XP machine it is **`Quadro4 980 XGL`**



- Select **`Performance & Quality Settings`** from the left hand menu
- Select **`Catia`** for the **`Active profile`**

Configuring NVidia cards on Linux

No configuration is necessary.

Configuring ATI cards on Windows and Linux

ATI do not provide an application-specific user interface, instead they have an XML configuration file **`atiogl.xml`** which lives in the following locations:

Windows platforms	**`C:\Windows`**
Linux platforms	**`/etc/ati`**

This may need to be edited to add driver configuration for PRIMER as follows if it is not already present in the file. More recent (early 2011 onwards) driver releases should already have this entry, so look for it first and only edit the file if it is missing:

```
----------- Start of file --------------
<PROFILES>
<!-- ========================== -->
<!-- Workstation Applications   -->
<!-- ========================== -->
```

... any number of entries

```
<!-- Oasys Ltd-->
```

```
<profile exename="primer14.exe">
   <OpenGLCaps>0x00008000</OpenGLCaps>
</profile>

<profile exename="primer14_64.exe">
   <OpenGLCaps>0x00008000</OpenGLCaps>
</profile>

<profile exename="primer14_x64.exe">
   <OpenGLCaps>0x00008000</OpenGLCaps>
</profile>
```

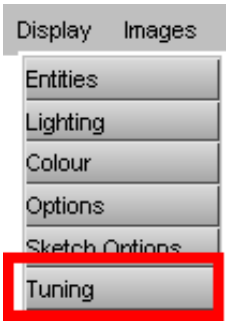
... any number of further entries

```
--------------- End of file -------------
```

9.8 Support for 3D mouse

From V11 onwards PRIMER supports 3D "mice" (spaceballs) supplied by 3DCONNEXION. If such a device is found on your machine then PRIMER will detect and use it automatically, allowing all six degrees of freedom of translation and rotation to be controlled.

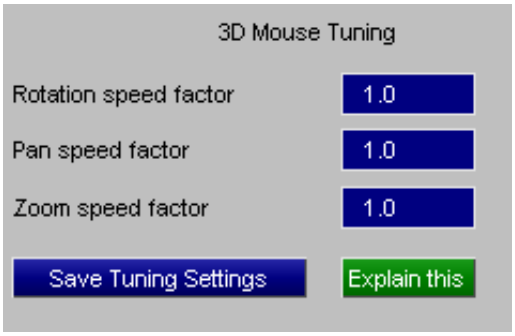
9.8.1 Tuning 3D mouse behaviour



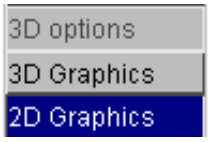
The 3DCONNEXION install package comes with a tuning panel that controls many aspects of the device, but it is also possible to adjust PRIMER's use of it via the **Display > Tuning** panel.

This allows the speed of response to 3D mouse movement to be adjusted: values > 1.0 will speed up response, and < 1.0 will slow it down.

Rotation speed factor	Controls the speed of response in all three rotational degrees of freedom
Pan speed factor	Controls the speed of response to left/right and up/down "panning" in the plane of the screen
Zoom speed factor	Controls the speed of response to in/out panning - tantamount to scale changes in non-perspective projection.



9.9 Special 3D graphics driver options.



On a 3D graphics driver special 3D-only viewing options become available, (greyed out under 2D), as shown here.

9.9.0 Brief description of 3D vs. 2D graphics.

In 2D mode PRIMER treats the display device as a dumb 2D device on which lines, polygons and text can be drawn. All coordinates are expressed in 2D integer space, ie [x,y] only, and all calculation of hidden-surface removal, lighting, etc must be done in software. Dynamic viewing relies on the software recalculating and redisplaying images quickly.

In 3D mode much more intelligence is available in the graphics driver, and much of the effort of computing images can

be shifted from the software to the hardware. In particular:

- Graphics coordinates exist in 3D [x,y,z] space, and the hardware does the transformation and projection onto the 2D screen. The software only has to provide the raw coordinates for an image once, and thereafter to change the view only a new scale, centre and rotation matrix.
- The hardware can compute shading, lighting and hidden-surface removal. So, again, the software only needs to provide raw coordinates, topology, light source data, etc, and then just ask the hardware to render it.
- The hardware can provide functions, such as Z-clipping, that are not available in software.

So 3D devices, especially those with hardware acceleration, give much faster graphics.

However there are also drawbacks to using 3D graphics: more memory is required since the full scene has to be sent to the driver using [x,y,z] floating-point coordinates. In addition laser plots cannot be generated by the 3D driver, so the capability to switch temporarily back to 2D mode has to be preserved.

Therefore there are options to control aspects of 3D graphics, and also the ability to switch back and forth between 3D and 2D modes.

9.9.1 Switching between 3D and 2D modes.

You can switch explicitly between 2D and 3D modes using the **3D Graphics** and **2D Graphics** buttons.
Some other graphics options also cause a switch.

On a 3D graphics driver the default mode is 3D, but certain graphics operations will switch the mode back to 2D. These are:

- Switching on dithered shading mode: continues until you switch it back explicitly.
- Plotting with laser output turned on. 2D mode is only transient during the course of the plotting operation, it is switched back to 3D automatically after each plot.

There are other circumstances when you might also want to switch explicitly to 2D mode:

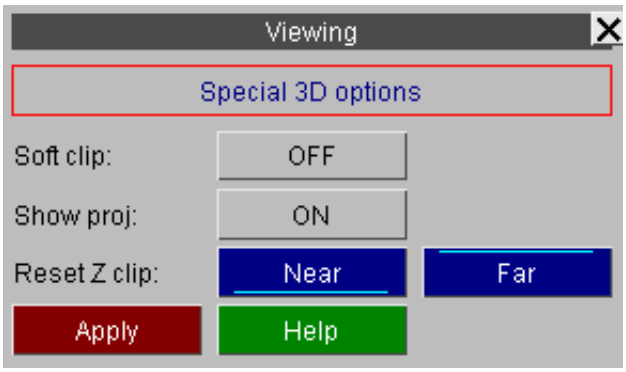
- When producing LC line-contour plots the result in 3D mode can be a bit patchy, with contour lines dropping in and out of view. This is a function of the Z buffering in hardware, and software (2D) images look much better.
- When using the OPACITY switch for contact surface and beam plotting. This works after a fashion when in 3D mode, but the transparent structure overlay does not use proper hidden-surface removal. The results are better in 2D mode where more control is available in the software.
- When animating large models. The amount of data stored for a 2D animation can be far less than for 3D, and can get round memory shortage problems. (However you would do better to use the X-Windows driver in this situation: see 4.4.5.4.)

9.9.2 **3D_OPTS...** Further 3D options.

The **3D_OPTS...** button gives a control panel for further 3D options.

The Special 3D options panel is shown in figure 9.7.2.

These options are described below.



9.9.2.1 **Soft clip** Clipping graphics outside the current screen window.

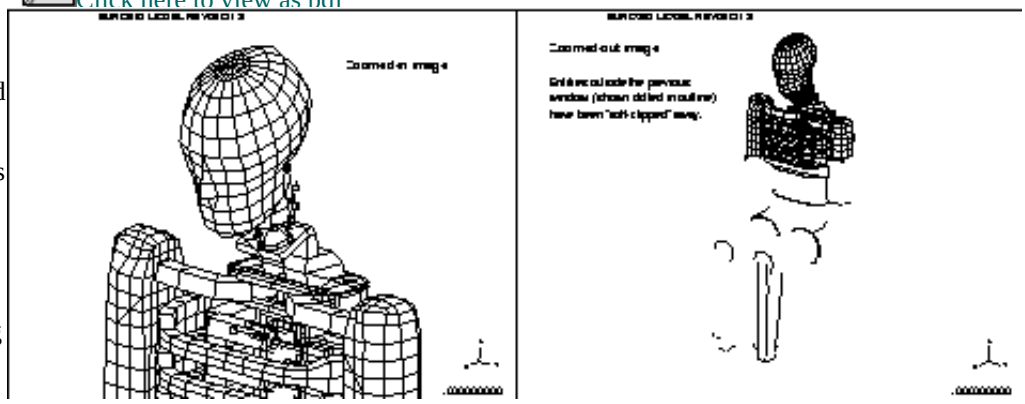
If you are dealing with a very large model, but are only looking at a small part of it, the 3D graphics driver can work unnecessarily slowly in its default mode of operation. This is because the whole model is sent to and manipulated by the graphics driver, despite the fact that you are only looking at a small part of it, in anticipation of your wanting to zoom out to see the whole of it.

If you turn Soft Clip on, and redraw the image, the graphics will run faster. This is because the software has "clipped" (ie removed) those parts of the image not visible in the current window before sending it to the 3D graphics driver, so the 3D driver has to process fewer graphics entities. However this also means that if you zoom out those parts of the image outside the previous window will not be there. This is illustrated in figure 9.7.2.1(a) and (b).

In this example the user has zoomed in on the neck and upper chest region of a side-impact dummy (left hand image), and then zoomed out to what should show the full dummy. This exposes the jagged edges left by the 3D clipping algorithm.

To see the missing elements you need to issue an explicit drawing command at the new scale to recalculate the clipping and send more elements to the 3D graphics driver.

Click here to view as pdf



9.9.2.2 SHOW_PROJ Showing the viewing frustrum

On 3D devices it is possible to show the current viewing "frustrum" at the bottom left corner of the plot by turning **SHOW_PROJ** on.

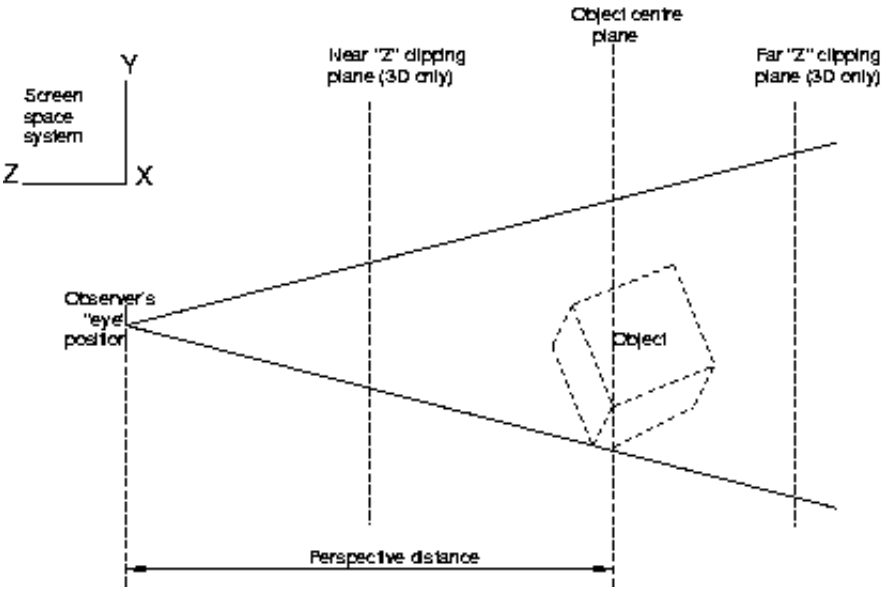
This shows the information in figure 9.7.2.2 (a copy of figure 9.1(b)).

The frustrum shown here assumes perspective projection.

The Z clipping plane locations are shown when **SHOW_PROJ** is on, and this can be very helpful when using Z clipping, as otherwise it is easy to "lose" the clipping planes.

The default near and far plane positions are drawn in green, and the plane locations in blue. So you can visualise movement relative to initial locations.

Click here to view as pdf



9.9.2.3 Using the Z clipping planes

The Z clipping planes are shown in figure 9.7.2.2. There are two planes: a "near" and a "far" one, which the hardware uses to clip the image in the +/- screen Z axis.

By default they are set just outside the +/-Z limits of the structure (shown as green lines in the projection box), so that no clipping takes place, but you can move them (shown as blue lines in the box) using the following mouse and keyboard meta-key combination:

<right shift> + <left mouse> Moves the near clipping plane.

Cursor symbol is

<right shift> + <right mouse> Moves the far clipping plane

Cursor symbol is

<right shift> + <mid mouse> Moves the both clipping planes

Cursor symbol is

In all cases moving the mouse up moves the plane(s) away from you, and down moves towards you. This is a form of dynamic viewing: the planes move and the image gets updated as the cursor moves. It is recommended that you turn the **SHOW_PROJ** switch, described above, on as this will enable you to see the planes moving in the projection box.

To reset the planes to their default positions use the Reset Z clip **NEAR** and **FAR** buttons. This will reset them to their initial positions (shown by the blue lines in the projection box).

10 Scripting

Introduction

JavaScript is a freely available scripting language that is normally found performing the "work" behind interactive web pages, however its syntax and structure also make it an excellent tool for providing an externally programmable interface to programmes in general.

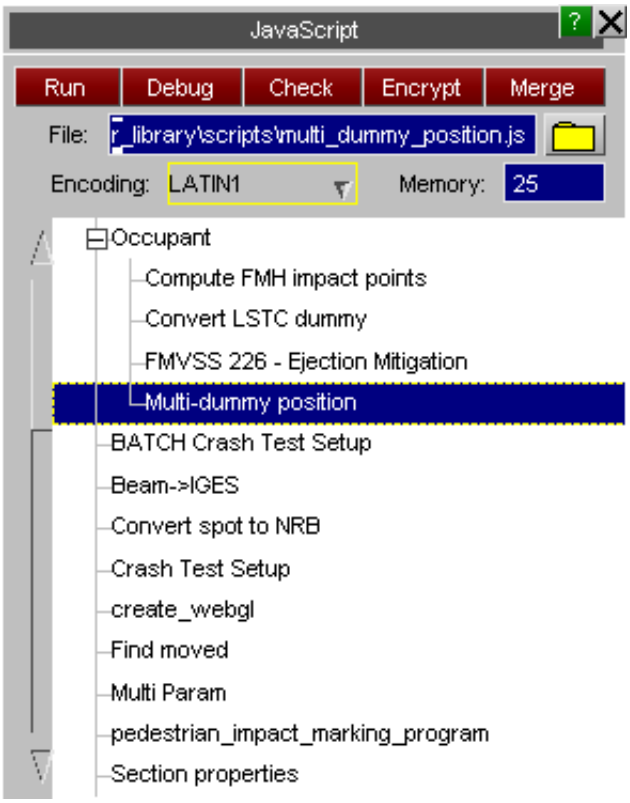
Within PRIMER it is implemented as an Application Programming Interface (API) which provides a range of functions that allow you to extract data from and poke it into the database, open windows, generate plots, and so on. This is documented in a separate JavaScript API Manual.

Anyone familiar with C or shell script programming will find existing JavaScripts are instantly readable, and can be given minor edits without further ado. For those who are more ambitious a good guide to the language is "**JavaScript, A definitive Guide**" by David Flanagan, published by O'Reilly. A brief tutorial on their use is given below.

10.1 Using JavaScript in PRIMER.

Javscripts need to be ***compiled***, meaning turned from something human-readable into a set of instructions that a computer can understand; and then ***run*** in their compiled form. They can be changed and rerun in their modified form at any time without having to exit and re-enter PRIMER, making the "write, test, modify, re-test" development cycle very quick and easy.

10.1.1 Compiling and Running a script



Run Will both compile and run the script unless it contains syntax errors, in which case it stops with an error message when compilation fails.

Debug Starts the JavaScript debugger, JaDe to debug the script.

Check Only compiles the script, reporting any errors found, and does not run it.

Encrypt A script can be encrypted so that the source code is hidden but the script can still be run (when compiling and running the script PRIMER decrypts the file in memory). Once encrypted the source code cannot be retrieved by an ordinary user so make sure that you keep the original file somewhere safe. As a last resort contact OASYS Ltd who can decrypt the script if required.
If a script is split up into separate files by Use the files are all combined together into the main file before encrypting.

Merge If a script is split up into separate files by Use the files are all combined together into a single file. This may be useful if you want to give the script so someone else and you do not want to have to give lots of different files.

Memory size is the memory allocated for garbage collection in the JavaScript engine. Please see the garbage collection section for more details.

Initially the **Run**, **Debug** ... buttons will be greyed out. If you type a script location into the text box or select the location using the 'folder' button they will be enabled. Alternatively left clicking on a script in the tree will automatically run the script. Right clicking on the script will bring up a popup menu with the **Run**, **Debug** ... options.

File encodings for scripts

Version 10.0 of PRIMER introduced the ability for unicode text to be used on widgets created in a script. Previous versions of PRIMER only supported English text so the default ASCII encoding was used for script files (this is still the default encoding for script files).

If you want to use unicode text in widgets then you must use a file encoding that is capable to representing the unicode 'characters' you require. The **File encoding** popup allows you to change the file encoding used when reading the script file. PRIMER supports the following file encodings:

Encoding	Description
LATIN-1	Default 'ASCII' encoding
BIG5	Taiwan/Hong Kong (traditional)
EUC-CN	Extended unix code (Simplified Chinese)
EUC-JP	Extended unix code (Japanese)
EUC-KR	Extended unix code (Korean)
GB	Chinese (simplified)
GBK	Chinese
ISO-2022-CN	Chinese
ISO-2022-CN-EXT	Chinese (extended)
ISO-2022-JP	Japanese
ISO-2022-JP-2	Japanese (extended)
ISO-2022-KR	Korean
JOHAB	Korean
SHIFT-JIS	Japanese
UTF-8	Should NOT have a byte order mark (BOM).
UTF-16	Should have a byte order mark (BOM). If not present assumes big endian
UTF-16LE	Little endian with or without byte order mark (BOM)
UTF-16BE	Big endian with or without byte order mark (BOM)
UTF-32	Should have a byte order mark (BOM). If not present assumes big endian
UTF-32LE	Little endian with or without byte order mark (BOM)
UTF-32BE	Big endian with or without byte order mark (BOM)

Please contact Oasys Ltd if you have problems or require another encoding to be supported.

To show the unicode text the appropriate font must be used. This can be set using the preferences `primer*cjk_unix_font` and `primer*cjk_windows_font`.

10.1.2 Dealing with errors in scripts

Script errors come in two forms:

Syntax errors

Are mistakes of JavasScript grammar or spelling, resulting in error messages during compilation.

These are easy to detect and correct since the line number and offending syntax are both described by the compiler. The script needs to be edited to correct the problem and then recompiled. Sometimes several iterations of the compile/edit cycle are required to eliminate all errors from a script.

Run-time errors

Are errors of context or logic in scripts that are syntactically correct, and thus have compiled, but which fail at some stage when being run.

A typical example of a run-time error is an attempt to divide a value by zero, yielding the illegal result infinity. More subtle errors involve passing an invalid value to a function, accessing an array subscript that is out of range, and so on.

10.1.3 Setting the Garbage Collection Memory Size

(This is an advanced topic, and you don't need to understand it.)

JavaScripts execute inside a memory "arena", allocated dynamically from the operating system, which grows in size as storage is requested within the script. This growth occurs due to requests for "new" variables within the script and also when API functions allocate and return values and objects, and it is limited only by what the operating system can deliver.

The nature of JavaScript means that objects frequently become redundant, and it is wasteful not to reuse the storage that they occupy, therefore there is a "Garbage Collection" process running behind the scenes which periodically checks storage and releases that which is no longer needed. This process is automatic and hidden from the user, it just "happens".

However Garbage Collection is quite a CPU-hungry process, so it is only carried out periodically when a certain threshold is reached. This can sometimes be observed during script execution as a periodic "pause for thought", and if you are monitoring memory usage with a system tool you may see it drop during these pauses.

Clearly this threshold value must be large enough not to trigger excessively frequent (and costly) garbage collections, while at the same time not being so large that scripts build up large amounts of excess memory to the detriment of the rest of the programme.

The **Memory size** value in the JavaScript panel is the amount of memory allocated for garbage collection. Every time a new object, array, string or double precision number is used a garbage collection 'thing' is also allocated. The Memory size is the total memory for these 'garbage collection things', **NOT** the total memory for the script. The total memory for the script could be significantly higher than this value. e.g the memory required for a Model object could be several kbytes but the memory for the 'garbage collection thing' for the Model object will something like 10 bytes for a 64bit operating system.

When the memory used for garbage collection 'things' reaches a significant proportion of **Memory Size** (normally about 2/3) then garbage collection will take place to try to reclaim memory. If no memory can be reclaimed and the total memory used for garbage collection reaches **Memory size** then the script will terminate with an error.

If your script has to retain a large number of objects, arrays, strings etc in memory then you may have to increase the value for **Memory size**. This can also be done using the `primer*javascript_memory_size` preference or adding a special memory comment at the top of the script.

To recap:

- This threshold does ***not*** limit the memory the script can use, that is limited only by the operating system.
- It sets the memory for Garbage Collection 'objects'.
- Scripts which allocate a lot of memory, and which exhibit frequent pauses, ***may*** run faster with a larger value.

- ... and finally:

  If you don't understand this topic don't worry. Most scripts will run quite happily with the default value, and you can ignore this setting unless they appear to be struggling, in which case try raising it. (As good an approach as any is to keep on doubling this value until the script works, but don't use very large sizes unnecessarily.)

10.1.4 Assigning JavaScripts to shortcut keys

Scripts can be assigned to shortcut keys to make the?m quick and easy to run. To do this use the shortcut keys panel which is accessed by pressing **Options** -> **Shortcuts** or by pressing the shortcut key '**?**'

Using the above button will only assign the script to the shortcut key for this session of PRIMER. If you want to always assign the script to a shortcut key use the **`primer*F1_key`** preferences etc in the **`shortcut_keys`** branch for PRIMER in the oa_pref preference file. Note that the preference can be used to run either a shortcut, a script or a macro. For PRIMER to know that the preference should run a script and not a macro, the script **MUST NOT** have the extension **prm**.

10.1.5 Maintaining a library of JavaScripts

When PRIMER starts it automatically looks for scripts in the directories:

- $OA_ADMIN/primer_library/scripts (if $OA_ADMIN is defined)
- $OA_INSTALL/primer_library/scripts
- $OA_HOME/primer_library/scripts

Each script that is found is assigned to the tree in the script panel.

From version 12 PRIMER will also look in subdirestories and the directory names will be used to create branches in the tree view of the scripts.

The directory that PRIMER looks in for script files can be changed in the oa_pref files in $OA_ADMIN, $OA_INSTALL and $OA_HOME by using the `script_directory` preference.

For example if you change the **`script_directory`** preference in the oa_pref file in the $OA_INSTALL directory to /test/primer_scripts then PRIMER will look for script files in the directories:

- $OA_ADMIN/primer_library/scripts (if $OA_ADMIN is defined)
- /test/primer_scripts
- $OA_HOME/primer_library/scripts

Using the "description:" comment at the top of a script to identify its purpose.

To help to identify scripts special comments are searched for in the top 20 lines of each script, and if **`description:`** is found, for example the comment line:

```
// description: Some description of the script's purpose
```

Then the description line is shown as hover text when the mouse is placed over that script in the tree.

Using the "name:" comment at the top of a script to change its name

Normally the name shown for a script will be its filename, stripped of any leading pathname and trailing ".js" extension.

However if the string **`name:`** is found in the first twenty lines of the script, then the following name will be used instead. For example the line:

```
// name: temporary
```

Will result in the script appearing with the name "**`temporary`**" in tha JavaScript panel. This does not affect the actual name of the script, only the name shown in the tree

Using the "memory:" comment at the top of a script to change the required memory

Sometimes the memory required for garbage collection needs to be changed.

If the string **`memory:`** is found in the first twenty lines of the script, then the size given will be used for the memory (unless the size in the memory textbox is larger than this value). For example the line:

```
// memory: 50
```

Will result in the script using 50Mb for garbage collection memory.

Using the "hide:" comment at the top of a script to hide it from the tree

Normally all scripts that have an extension beginning with js will be shown in the tree. You may want to stop some scripts from being shown in the tree. For example if your script is broken up into 'include files' by using 'Use' then you may want to hide the includes any only show the main script. If the string **`hide:`** is found in the first twenty lines of the script, and the value is TRUE then the script will not be shown. i.e. the line:

```
// hide: TRUE
```

Will result in the script not being shown in the tree. Alternatively by convention any scripts which have the extension 'jsi' or 'jsm' will be hidden as they are assumed to be ''include' files or 'module' files.

Using the "folder:" comment at the top of a script to change which branch a script is shown in

When PRIMER searches the script directory it looks recursively through directories and by default the directories will appear as branches in the tree. You may want to change which directory a script is shown in. The special 'folder' comment can be used to do this. For example the line

```
// folder: occupant/test
```

Will result in the script being shown in the tree in branch occupant->test.

10.1.6 Running a JavaScript in "batch" mode.

All the above assumes that JavaScripts will be run interactively from the user interface, however it is also possible to run a script in "batch" mode using the command line interface. The relevant command-line commands are:

/SCRIPT	**READ** `<script>`	Read, compile and execute <script>

To run a JavaScript from batch these commands need to be placed in a command file and run using the command line "**-cf=***command filename*" option. For example the command file might be:

```
... some other commands
/SCRIPT READ my_script.js
...some further commands
```

And the command line required to run PRIMER might be something like:

```
$OASYS/primer14_x64.exe -d=default -cf=command_file -exit analysis_name
```

Obviously multiple script invocations may be placed in a command file. For more information see:

Using command files	Describes command files, and explains how to create and use them
Command line arguments	Describes the various command line arguments, and how to use them

10.2 A Brief Tutorial on JavaScript in PRIMER

While most people associate JavaScript with web pages and html it is a full-featured programming language. Additionally JavaScript is not Java! JavaScript is completely unrelated to Java.

Hopefully, enough people are familiar enough with JavaScript through the internet to be able to use it in PRIMER. JavaScript has all of the functionality you would expect from a programming language, such as:

- variables (strings, numbers, booleans, objects, arrays)
- functions
- control flow statements such as `if`, `while`, `do`, `for`, `switch` etc.
- objects
- arrays
- regular expressions
- maths functions (sin cos, log, sqrt etc)

Additionally, PRIMER extends JavaScript by defining several new object classes specifically for PRIMER. A detailed reference on these classes is given in the separate JavaScript API Manual available from Oasys Ltd. Over time this functionality will be extended. If you need to do something which is not possible with the current functionality then contact Oasys Ltd.

Probably the best way to see what sort of things are easily possible in PRIMER using JavaScript is to look at the example scripts which are given out with PRIMER in the **$OASYS/primer_library/examples** directory. However, additionally a few example scripts are given here and documented. For more details on the classes described in these examples see the separate JavaScript API Manual.

10.2.1 Example scripts

Example 1: Make a 10x10 mesh of nodes and shells

Problem

How can you make a grid of new nodes and create a mesh of shells?

Solution

```
// Declare variables
var x, y, i, j;
// Create a new model using Model constructor
var m = new Model();
// Give message saying what we are doing
Message("Making nodes");
// Loop over y nodes
for (y=0; y<11; y++)
{
// Loop over x nodes
    for (x=0; x<11; x++)
    {
// make node in model m using Node constructor
        var n = new Node(m, 1+x+(y*11), x*10, y*10, 0);
    }
}
Message("Making shells");
for (i=1; i<=10; i++)
{
    for (j=1; j<=10; j++)
    {
// Make shell using Shell constructor
        var s = new Shell(m, i+(j*10), 1, ((i-1)*11)+j+0, ((i-1)*11)+j+1,
                                          ((i-0)*11)+j+1, ((i-0)*11)+j+0);
    }
}
// Update the graphics in model
m.UpdateGraphics();
// View XY plane
View.Show(View.XY);
// Autoscale view
View.Ac();
```

Discussion

To make the nodes and shells we first need to choose the model that they will be made in. The Model class in PRIMER gives you access to any existing models and also allows you to make new models. The line

```
var m = new Model();
```

creates a new model in PRIMER and m is then a Model object which you can use. We then need to make a grid of nodes. To make a 10x10 mesh of shells we need to make 11 nodes in the x direction and 11 in the y direction so we use 2 'for' loops to make the nodes. Just as a model can be made using the Model constructor a node can be made with the Node constructor. The line

```
var n = new Node(m, 1+x+(y*11), x*10, y*10, 0);
```

makes a new node in model m. The other arguments are the x, y and z coordinates of the node. We can make the shells in the same way using the Shell constructor. The arguments required are the model, shell number (EID), the part to create the shell in (PID - in this case we are using part 1) and the 4 nodes (N1-N4).

Once we have made all of the nodes and shells we have to tell PRIMER that things have been made and so the graphics need updating. This is done with the UpdateGraphics method for the model. Once we have refreshed the graphics we select the view we want to see and autoscale the plot.

The source code for this example is available here.

Example 2: Change the number of integration points on a *SECTION_SHELL card depending on thickness

Problem

You want to loop over all of the sections in your model (model 1) giving more integration points to the thicker sections.

Solution

```
// Get the model object for model 1
var m = Model.GetFromID(1);
// Get the first section card in the model
var s = Section.First(m);
// While there is a section card look at it
while (s)
{
// If this section card is a section shell type then look at it
    if (s.type == Section.SHELL)
    {
// If thickness is < 1 then assign 2 ipts, < 2 assign 3 ipts, otherwise 5 ipts
        if      (s.t1 < 1.0) s.nip = 2;
        else if (s.t1 < 2.0) s.nip = 3;
        else                 s.nip = 5;
    }
// Get the next section after this one
    s = s.Next();
}
```

Discussion

```
var m = Model.GetFromID(1);
```

retrieves the model pointer for model 1. Once we have the model pointer we can then look at the contents of the model.

```
var s = Section.First(m);
```

will return the section object for the first section in the model (or null if there are no sections). Once we have a section object then it is simple to look at the properties of the object and change them as required. First we look to see if the section type is correct (i.e the section is a shell section). If it is we look at the thickness and set the number of integration points.

```
s = s.Next();
```

will return the next section after this one or null if one does not exist. The while loop will keep looping until s is not 'true' so when we get to the last section `s.Next` will return null and we will break out of the while loop.

Note that this is a very simple example and we have not done any error checking. For example it is possible that one (or more) of the section cards in the model has been refeered to by a part card but has not actually been read in or created. In this case PRIMER creates a 'latent' definition. The example should check for this problem. This can easily be done by looking at the exists property of the section. The check on the section type could be extended by doing:

```
if (s.type == Section.SHELL && s.exists)
```

The source code for this example is available here.

11 Quick Find

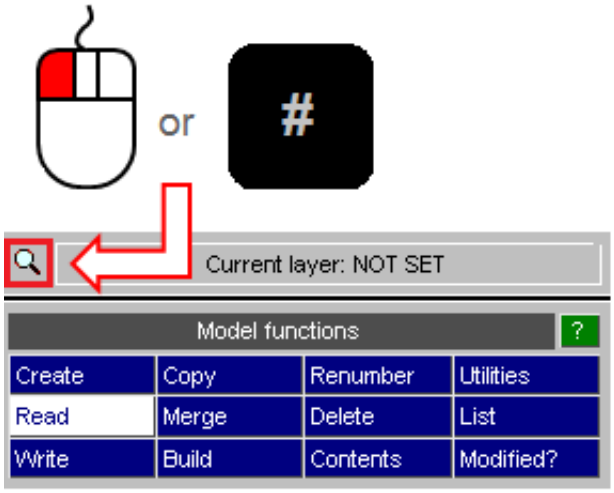
Introduction

Quick Find can be used to search for and quickly:

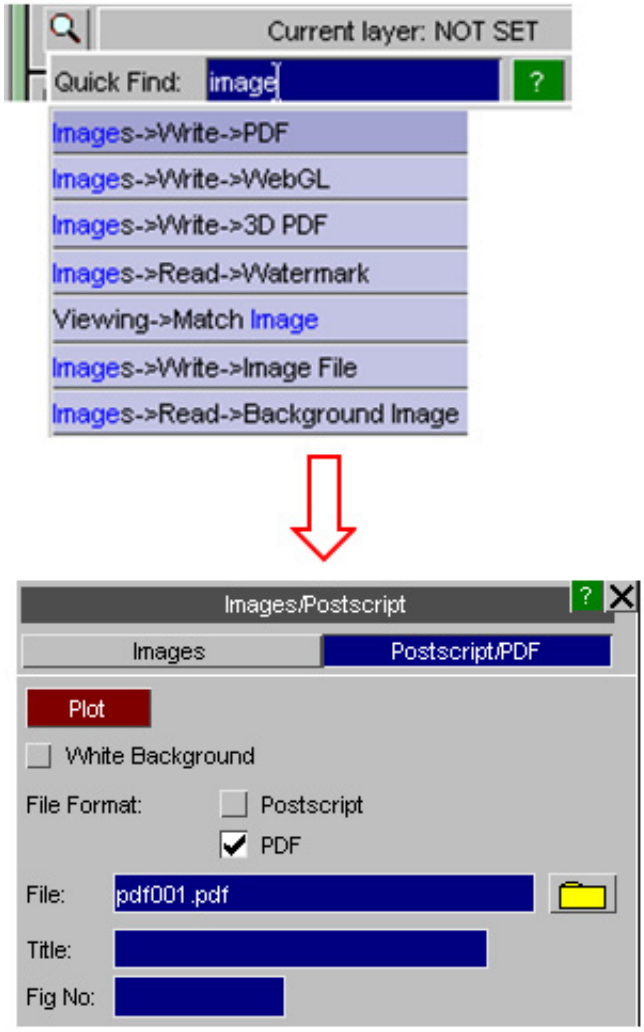
- Go to menus / functionality
- Open model entity edit panels
- Blank / Unblank / Only Include files
- Open tutorials
- Open specific pages in the LS-DYNA keyword PDF manual

all from a single location.

It can be accessed by clicking on the magnifying glass above the model functions menu or by pressing the '#' key.



Typing in the textbox brings up a list of found items that match the entered text. Items in the list can be selected by clicking on them or by using the up and down arrow keys and pressing enter. The selected item will then perform the task, e.g. open a menu.



Fuzzy Matching

A ’fuzzy’ matching method is used to match the entered text with the searchable items. It judges that something has matched when the characters of the entered text appear in the same order as the item that can be searched for.

For example if you type ’mptp’ then ’Tools->**M**easure->**P**art **T**o **P**art’ would be a match, but ’Tools->**M**easure->**P**oin**t** Angle’ wouldn’t because the final ’p’ doesn’t match. (Note that the search is case insensitive).

Additionally, if the entered search pattern contains spaces and the characters do not all match in the same order then PRIMER will look to see if the words can be swapped to find a match.

For example ’create node’ would find ’Tools->NODE->Create’ even though the words do not appear in that order.

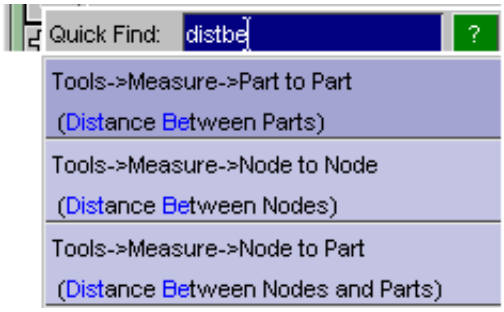
This hopefully makes it easier to find items as you do not need to know the precise search term.

The found items are listed in order of how closely they match the entered text so items that more closely match appear nearer the top of the list. It determines this by assigning a score to each match, with higher scores given to items that contain consecutively matched characters and if the characters appear at the start of words.

Search Terms

The default search term associated with a menu item is the trail of menus/buttons you would need to manually open/press, e.g. to get to measure part to part you would need to go to Tools, then Measure then Part to Part, hence the search term 'Tools->Measure->Part to Part'.

In addition, some menus have alternative search terms associated with them. For example Measure Part to Part can also be found from the alternative text 'Distance Between Parts':



This can be useful for cases where you don't know or can't remember under which menu some functionality lives.

Note that the alternative text appears in brackets under the default search term so you can see how you would get to the menu manually.

If you can't find menus that you know exist in PRIMER it is likely that you are using different terminology to what we expect. If so, please contact Oasys Ltd and we can add alternative text based on what you are entering as your search text.

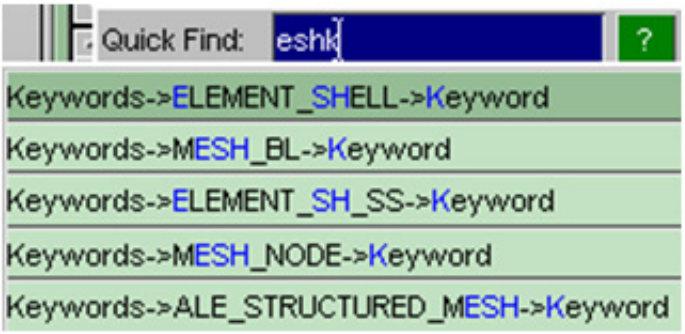
Alternative text associated with a menu may also describe some of the features on a menu. For example the overlay colour of elements is set in the Display Options menus, but if you didn't know this it would be hard to find.

In this case the alternative text 'Set Overlay Colour' is associated with this menu:

As you can see the alternative text 'Beam True Sections' is also associated with this menu as the switch to select this option is also on the Display Options menu.

Keyword Menus

As well as Tools / Mesh Tools menus, menus that live under Keywords can also be searched for. For example if you want to open the *ELEMENT_SHELL keyword panel you could type 'eshk':



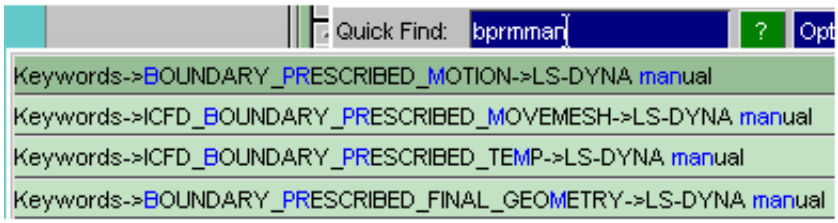
Keyword menus are coloured green in the list to help differentiate them from Tools / Mesh Tools menus which are coloured blue.

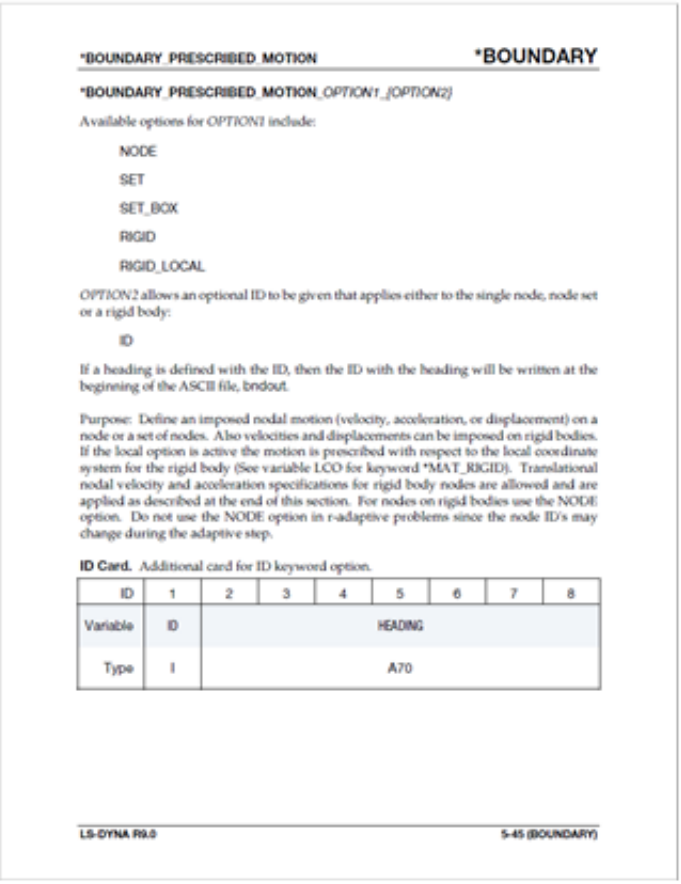
Note that certain menus may not be searchable depending on the contents of any models loaded in PRIMER. For example, if a model doesn't contain any SPH elements you would not be able to search for 'Keywords->ELEMENT_SPH->Modify'.

LS-DYNA Manual

If the installation of the Oasys Ltd software has the $OA_INSTALL/manuals/lsdyna/keywords.txt file it is also possible to search for a keyword and open the LS-DYNA PDF manual at the appropriate page. This should be the case in a full installation.

For example, to open the manual at the *BOUNDARY_PRESCRIBED_MOTION page you could type ’bprmman’:





*BOUNDARY_PRESCRIBED_MOTION *BOUNDARY

BOUNDARY_PRESCRIBED_MOTION_*OPTION1_{OPTION2}

Available options for *OPTION1* include:

NODE

SET

SET_BOX

RIGID

RIGID_LOCAL

OPTION2 allows an optional ID to be given that applies either to the single node, node set or a rigid body:

ID

If a heading is defined with the ID, then the ID with the heading will be written at the beginning of the ASCII file, bndout.

Purpose: Define an imposed nodal motion (velocity, acceleration, or displacement) on a node or a set of nodes. Also velocities and displacements can be imposed on rigid bodies. If the local option is active the motion is prescribed with respect to the local coordinate system for the rigid body (See variable LCO for keyword *MAT_RIGID). Translational nodal velocity and acceleration specifications for rigid body nodes are allowed and are applied as described at the end of this section. For nodes on rigid bodies use the NODE option. Do not use the NODE option in r-adaptive problems since the node ID's may change during the adaptive step.

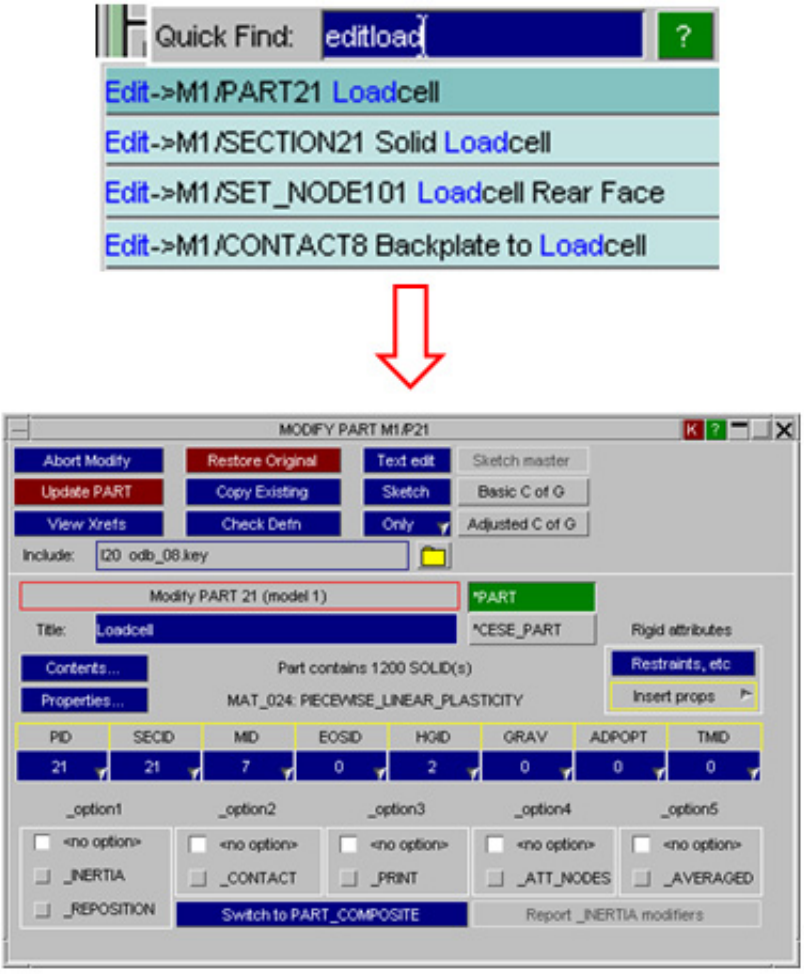
ID Card. Additional card for ID keyword option.

ID	1	2	3	4	5	6	7	8
Variable	ID	HEADING						
Type	I	A70						

LS-DYNA R9.0 5-45 (BOUNDARY)

Model Entities

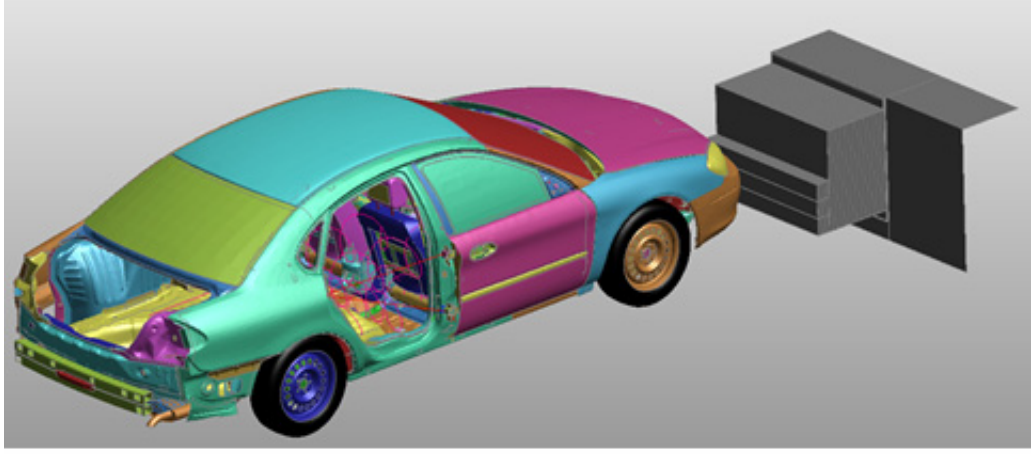
If a model is loaded in PRIMER it is also possible to search for entities and open their edit panels, for example if you wanted to edit a part with the title ’Loadcell’ you could search for ’editload’:

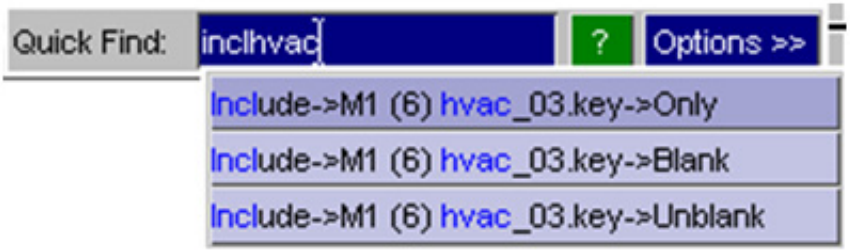


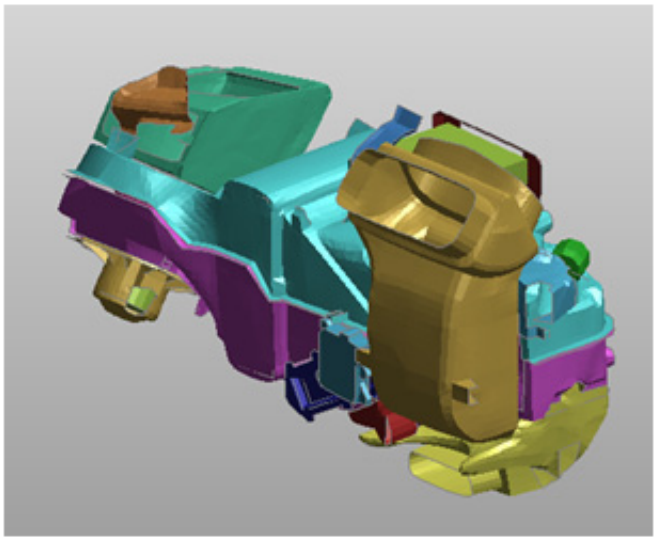
Note that this is limited to entities that can have titles, i.e. *NODEs, *ELEMENT_xxxx entities are not searchable. This is so that the list of searchable items is kept to a manageable size.

If a model has include files they can be searched for and ’Blanked’, ’Unblanked’ or ’Only’d

For example to ’Only’ the hvac_03.key include file:

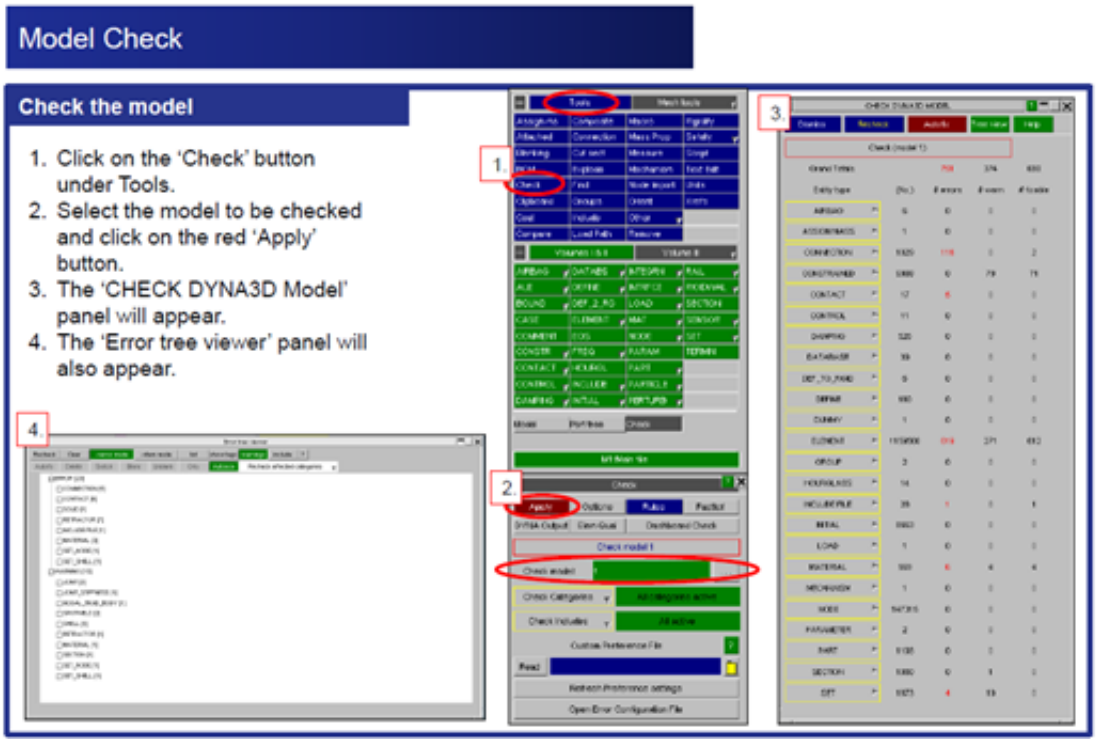



Quick Find:
inclhvac
?
Options >>
Include->M1 (6) hvac_03.key->Only
Include->M1 (6) hvac_03.key->Blank
Include->M1 (6) hvac_03.key->Unblank



Tutorials

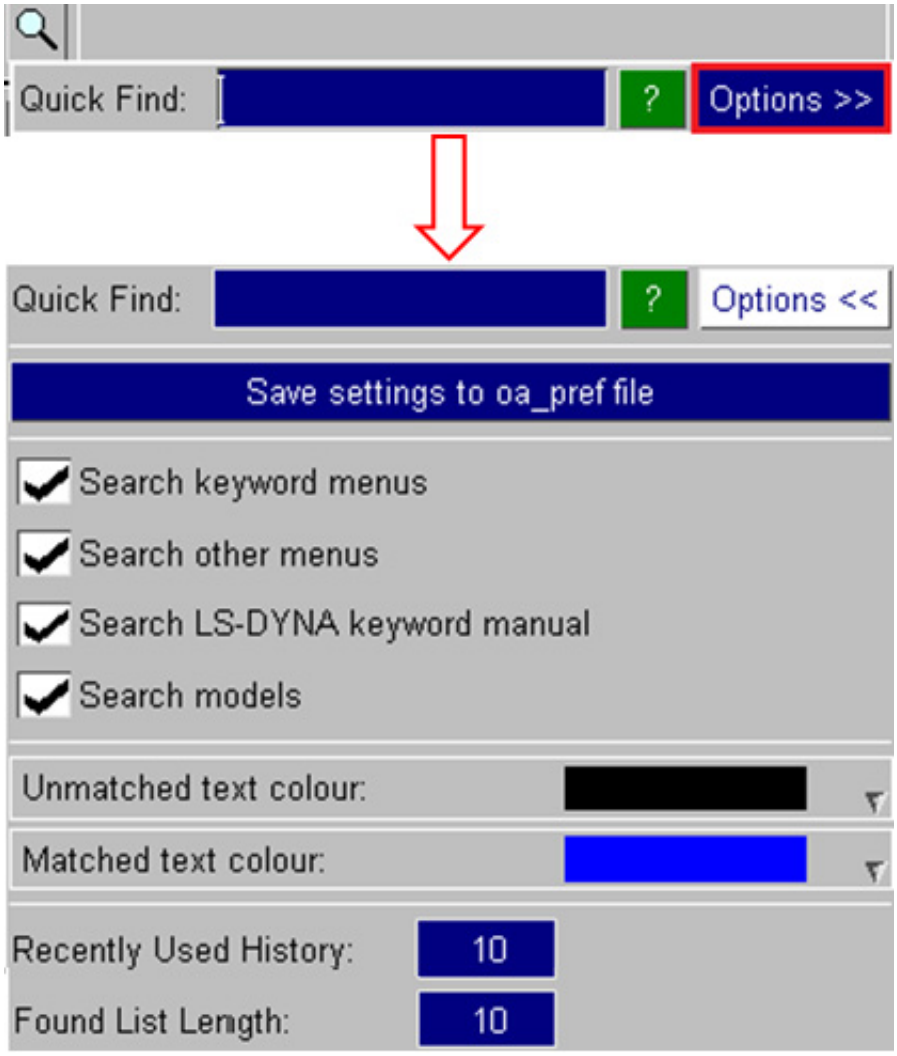
The full installation of the Oasys Ltd software contains some pdf tutorials for various features within the software. They are installed in the $OA_INSTALL/manuals/tutorials/primer directory and can be found and opened using Quick Find.



Options

There are a few options that can be set to alter how Quick Find works. These can be accessed by pressing the ’Options >>’’ button.

- Save settings to the oa_pref file
- Set which items are searchable
- Set the text colours for matched and unmatched characters
- Recently selected items are saved by PRIMER and appear higher in the list of available options. By default the last ten selected items are saved, but this can be changed here. To turn it off set it to zero.
- Set the maximum number of found items to display in the list



APPENDICES

I Standard object Names and Acronyms

II Dummy "tree" file structure

III Origami "tree" file structure

IV Airbag Folding Example

V Seatbelt "tree" file structure

VI Format translation during model read

VII Format translation during model write

VIII "Curve" file structures

IX Primer database format

X Headform ’tree’ file example

XI Target and Position ’tree’ file example

XII: Dialogue (typed in) Command Syntax

XIII: Summary of "oa_pref", command-line and Environment Variable settings

XIV: Automated model build from command line (csv file)

XV: Accessing model and include file mass properties

APPENDIX I: Standard Object Names and Acronyms

Inside PRIMER every class of object (nodes, parts, solids, etc...) has a standard "acronym" that is used when labelling items on the screen, and which can sometimes be needed when the user types in a specific object label. For example the acronym for a node is "N", thus node 27 will always be labelled as "N27".

In addition, when PRIMER has more than one model in memory it is necessary to prefix the object label with its model number. The acronym for a model is "M", thus if node 27 exists in both models 1 and 3 the two labels will be respectively:

M1/N27	(Model #1, Node #27)
M3/N27	(Model #3, Node #27)

You only have to remember these when using the **Key in** method of defining objects (section 6.2), and even then only when the object type is not implicit. For example to select node 10 in model #1 you will need to "key in":

10	If both object type (NODE), and the model id (1) are preset.
N10	If object type is ambiguous, but model id is preset.
M1/N10	If neither object type or model id are preset

In most situations the <model> and <object type> are implicit: either because of the context of the operation, or because of prior selections, or because of "filter" settings, and only numbers are required.

As well as acronyms every object has a "formal" name that is used when referring to it in error messages, diagnostic output, panel buttons, etc. This is the same as its LS-DYNA keyword where relevant, although PRIMER adds a few categories which are not found in LS-DYNA input.

The complete list of object types, their standard acronyms and their formal names is:

Object type		Standard Acronym	Formal name
Model		M	MODEL
Include File		INC	INCLUDE FILE
Airbag definition		ABAG	AIRBAG
	Interaction	AINT	AIRBAG_INTERACTION
	Reference Geometry	ARDT	AIRBAG_REFERENCE
	Shell Reference Geometry	ASRG	AIRBAG_SHELL_REFERENCE
ALE Arbitrary Lagrange/Euler		ALEX	ALE
	Multi-Material Group	ALMM	ALE_MULTI-MATERIAL_GROUP
	Reference System Curve	ALRC	ALE_REFERENCE_SYSTEM_CURVE
	Reference System Node	ALRN	ALE_REFERENCE_SYSTEM_NODE
	Reference Syetem Switch	ALRS	ALE_REFERENCE_SYSTEM_SWITCH
	FSI Switch MMG	ALFS	ALE_FSI_SWITCH_MMG
Boundary conditions (general)		BNDY	BOUNDARY
	Prescribed Motion	BPRM	PRESCRIBED_MOTION
	Spc	BSPC	SPC
Component		COMP	COMPONENT
	Gebod	CGBD	COMPONENT_GEBOD
	Hybrid III	CHB3	COMPONENT_HYBRIDIII
Constrained (generic types)		CNST	CONSTRAINED
	Generalized Weld	GWLD	GENERALIZED_WELD
	Interpolation	ITRP	INTERPOLATION
	Joint	JNTC	JOINT

	Joint Stiffness	JSTF	JOINT_STIFFNESS
	Lagrange in Solid	LAIS	LAGRANGE_IN_SOLID
	Linear	LINC	LINEAR
	Nodal Rigid Body	NRBC	NODAL_RIGID_BODY
	Node Set	NSET	NODE_SET
	Points	PNTS	POINTS
	Rivet	RIVT	RIVET
	Spotweld	SWLD	SPOTWELD
	Spline	SPLN	SPLINE
Generalized stiffnesses		JSTF	GENERALIZED
Contact (generic)		CGEN	CONTACT
	"Sliding" (general 3D)	CONT	CONTACT_SLIDING
	Geometric	CENT	CONTACT_GEOMETRIC
	Gebod	CGEB	CONTACT_GEBOD
	Interior	CINT	CONTACT_INTERIOR
	Rigid Surface	CRIG	CONTACT_RIGID_SURFACE
	1D (rebar)	C_1D	CONTACT_1D
	2D (slide lines)	C_2D	CONTACT_2D
	Auto Move	C_AM	CONTACT_AUTOMOVE
	Coupling	C_CO	CONTACT_COUPLING
	Guided Cable	C_GC	CONTACT_GUIDED_CABLE
Control cards (all)		CTRL	CONTROL
Damping (general)		DAMP	DAMPING
	Global	GDMP	DAMPING_GLOBAL
	Modal	MDMP	DAMPING_MODAL
Database (general)		DBAS	DATABASE
	Ascii	DASC	DATABASE_ASCII
	Binary	DBIN	DATABASE_BINARY
	Extent ascii	DAEX	DATABASE_EXTENT_ASCII
	Extent binary	DBEX	DATABASE_EXTENT_BINARY
	Extent ssstat	DASS	DATABASE_EXTENT_SSSTAT
	History	DH	DATABASE_HISTORY
	<Scalar Item>	DSCA	DATABASE_<scalar item>
	Cross-section	XSEC	DATABASE_CROSS_SECTION
	Nodal force group	NFGR	DATABASE_NODAL_FORCE_GROUP
	PWP Flow	PWPF	DATABASE_PWP_FLOW
	Tracer particles	TRAC	DATABASE_TRACER
Define (generic)		DEFN	DEFINE
	Alebag Bag	ALBG	DEFINE_ALEBAG_BAG
	Alebag Hole	ALHL	DEFINE_ALEBAG_HOLE
	Alebag Inflator	ALIN	DEFINE_ALEBAG_INFLATOR
	Box	BOX	DEFINE_BOX
	Connection Properties	CPRP	DEFINE_CONNECTION+PROPERTIES
	Coordinate System	CSYS	DEFINE_COORDINATE
	Contact Volume	CVOL	DEFINE_CONTACT_VOLUME
	Staged Construction Part	DSCP	DEFINE_STAGED_CONSTRUCTION_PART
	Construction Stages	DSTG	DEFINE_CONSTRUCTION_STAGES
	Death Times	DTIM	DEFINE_DEATH_TIMES
	Friction	FRIC	DEFINE_FRICTION
	Hex Spotweld Assembly	HSWA	DEFINE_HEX_SPOTWELD_ASSEMBLY
	Load Curve	LC	DEFINE_CURVE
	Curve Entity	LENT	DEFINE_CURVE_ENTITY
	Curve Compensation	LCMP	DEFINE_CURVE_COMPENSATION
	Curve Feedback	LFBK	DEFINE_CURVE_FEEDBACK
	Curve Trim	LTRM	DEFINE_CURVE_TRIM
	Spring/Damper Orientation Vector	SDOV	DEFINE_SD_ORIENTATION
	Set Adaptive	STAD	DEFINE_SET_ADAPTIVE
	Transformation	TFRM	DEFINE_TRANSFOR ATION

	Table	TABL	DEFINE_TABLE
	Vector	VECT	DEFINE_VECTOR
	Spotweld Failure Resultants	SWFR	DEFINE_SWFR
	Spotweld Rupture Parameter	SWRP	DEFINE_SWRS
	Spotweld Rupture Stress	SWRS	DEFINE_SWRP
Deformable to Rigid		DTOR	DEFORMABLE_TO_RIGID
Element (generic)		EL	ELEMENT
	Solid	H	SOLID
	Beam	B	BEAM
	Shell	S	SHELL
	Shell Source Sink	SHSS	SH_SS
	Thick shell	T	TSHELL
	Discrete (spring/damper)	D	DISCRETE
	Lumped inertia	IN	INERTIA
	Lumped mass	MA	MASS
	Mass Part	MP	MASS_PART
	Seatbelt	SB	SEATBELT
	Accelerometer	ACC	ACCELEROMETER
	Pretensioner	PRET	PRETENSIONER
	Retractor	RETR	RETRACTOR
	Sensor	SENS	SENSOR
	Slipring	SLIP	SLIPRING
	SPH	SPH	SPH
	Trim	TRIM	TRIM
Encrypted		CRYP	ENCRYPTED
Equation of State		EOS	EOS
Hourglass		HG	HOURGLASS
Initial conditions (generic)		INIT	INITIAL
	Stress Section	INSS	INITIAL_STRESS_SECTION
	Axial Force Beam	IAFB	INITIAL_AXIAL_FORCE_BEAM
Integration Beam		INTB	INTEGRATION_BEAM
Integration Shell		INTS	INTEGRATION_SHELL
Interface		IFCE	INTERFACE
Loads (general)		LOAD	LOAD
	ALE Convection	LALE	LOAD_ALE_CONVECTION
	Moving Pressure	LMOV	LOAD_MOVING_PRESSURE
	Body	LBOD	LOAD_BODY
	Segment	LSEG	LOAD_SEGMENT
	Segment Set	LSSG	LOAD_SEGMENT_SET
	Shell	LSHE	LOAD_SHELL
	Segment Nonuniform	LSGN	LOAD_SEGMENT_NONUNIFORM
	Segment Set Nonuniform	LSSN	LOAD_SEGMENT_SET_NONUNIFORM
	Thermal Variable Shell	LTVS	LOAD_THERMAL_VARIABLE_SHELL
Material (structural)		MAT	MATERIAL
	(Thermal)	TMAT	THERMAL_MATERIAL
Node		N	NODE
	Node Scalar	N_SC	NODE_SCLAR
	Node Rigid Surface	N_RS	NODE_RIGID_SURFACE
	Node Transform	N_TR	NODE_TRANSFORM
Parameter		PARM	PARAMETER
Part		P	PART
Part Adaptive Failure		PADF	PART_ADAPTIVE_FAILURE
Part Modes		PMOD	PART_MODES
Part Senor		PSEN	PART_SENSOR
Part Move		PMOV	PART_MOVE
Perturbation		PERT	PERTURBATION
Rail (general)		RAIL	RAIL
	Train	RTRN	RAIL_TRAIN

	Track	RTRK	RAIL_TRACK
Rigid (stone) walls		WALL	RIGIDWALL
Section		SECT	SECTION
Segment (for contact, etc)		SEG	SEGMENT
Sensor (general)		SNSR	SENSOR
	Control	SCON	SENSOR_CONTROL
	Switch	SSWT	SENSOR_SWITCH
	Define	SDEF	SENSOR_DEFINE
Set (generic)		SET	SET
	Beam	S_BM	SET_BEAM
	Discrete	S_DS	SET_DISCRETE
	Multi-Material Group	S_MM	SET_MULTI-MATERIAL_GROUP
	Node	S_NO	SET_NODE
	Part	S_PT	SET_PART
	Segment	S_SG	SET_SEGMENT
	2D Segment	S_2D	SET_2D_SEGMENT
	Shell	S_SH	SET_SHELL
	Solid	S_SO	SET_SOLID
	Thick shell	S_TS	SET_TSHELL
Termination		TERM	TERMINATION
Translate (Nastran, etc)		TRAN	TRANSLATE
User-defined data/subroutines		USER	USER

The following object names are unique to PRIMER, although they can be written to an LS-DYNA input deck after the *END card:

Object type		Standard Acronym	Formal name
Airbag "Origami"		ORIG	ORIGAMI
	Fold defn	FOLD	FOLD
	Orientation	ORNT	ORIENTATION
Assembly (subset)		SASS	ASSEMBLY
Assign Mass		ASSM	ASSIGN MASS
Connection		CONX	CONNECTION
Dummy definition		DUMM	DUMMY
	Assembly	ASSY	ASSEMBLY
Headform		HEAD	HEADFORM
	Target Point	TARG	TARGET POINT
	Headform Position	POSN	HEADFORM POSITION
IP Pendulum		IPPI	IP Pendulum
Group		GROP	GROUP
Mechanism		MECH	MECHANISM
	Connection	MCON	CONNECTION
	Point	MPNT	POINT
	Child	MMCH	CHILD
Seatbelt fitting		BDEF	SBELT Defn

APPENDIX IIa: Dummy "tree" file format.

A "dummy" is not a special model, rather it is a normal model in which the connectivity of parts is defined in a specific hierarchy using a "tree" file.

The following terminology needs to be understood:

"Dummy" Is the name of a model, or subset of a model, which constitutes a connected series of assemblies which can be manipulated in a hierarchical fashion. In fact it need not be a dummy (occupant) at all: any mechanism could be represented in this way so long as its assemblies form a discrete "tree" (or trees).

"Assembly" Is one or more PARTs and/or SETS of parts. These are manipulated as a rigid body for the purposes of positioning, although they may contain both rigid and/or deformable materials. Assemblies are connected via JOINT_ STIFFNESS definitions in a strictly defined parent/child hierarchy.

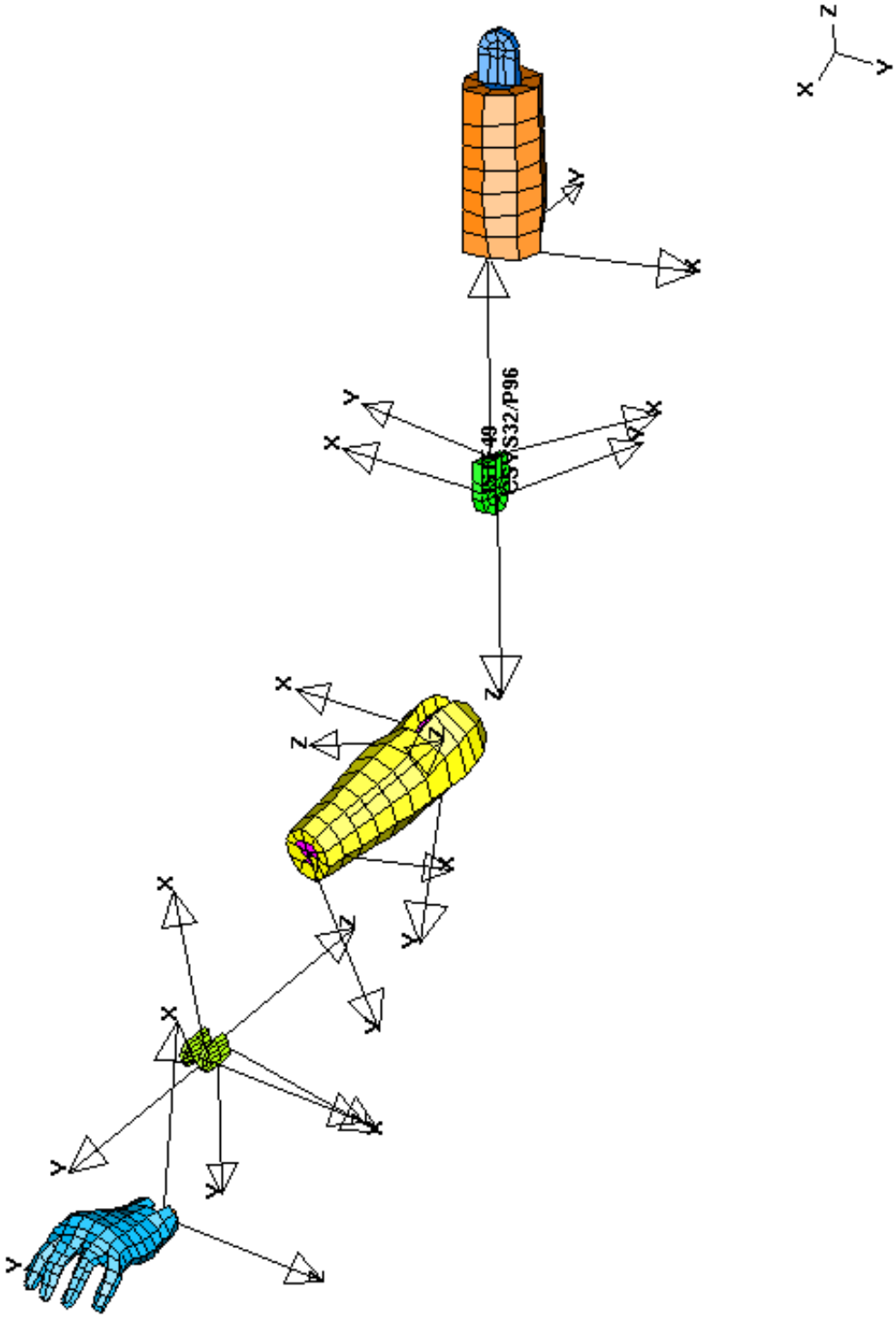
"Point" A location attached to an assembly that can be used to drive mechanism-style positioning, and also to impose restraints at a point. (Added in release 9.3, Feb 2007)

"Tree" Assemblies must be connected in a strict hierarchy. The "root" assembly is grand^{N}-parent to all other assemblies, which are arranged in a parent/child order. For example the pelvis is parent to the upper leg, which in turn is parent to the lower leg, with the foot being the youngest child.

"Parent/child" This terminology is used to represent the relationship between assemblies. A "child" assembly always follows the motion of its "parent", to the <nth> generation. For example waggling the upper leg also moves the lower leg and foot, all in a rigid-body sense. An assembly may have any number of children, and either zero or one parents. A child with no parents (an orphan) is usually the "root" of a dummy, but it is legal to have more than one "orphan" in a model.
However it is strictly illegal for a child to have more than one parent, as this would create a dynamically indeterminate mechanism requiring compatibility equations for its solution. Put another way assemblies may not connect to themselves, either directly or indirectly.

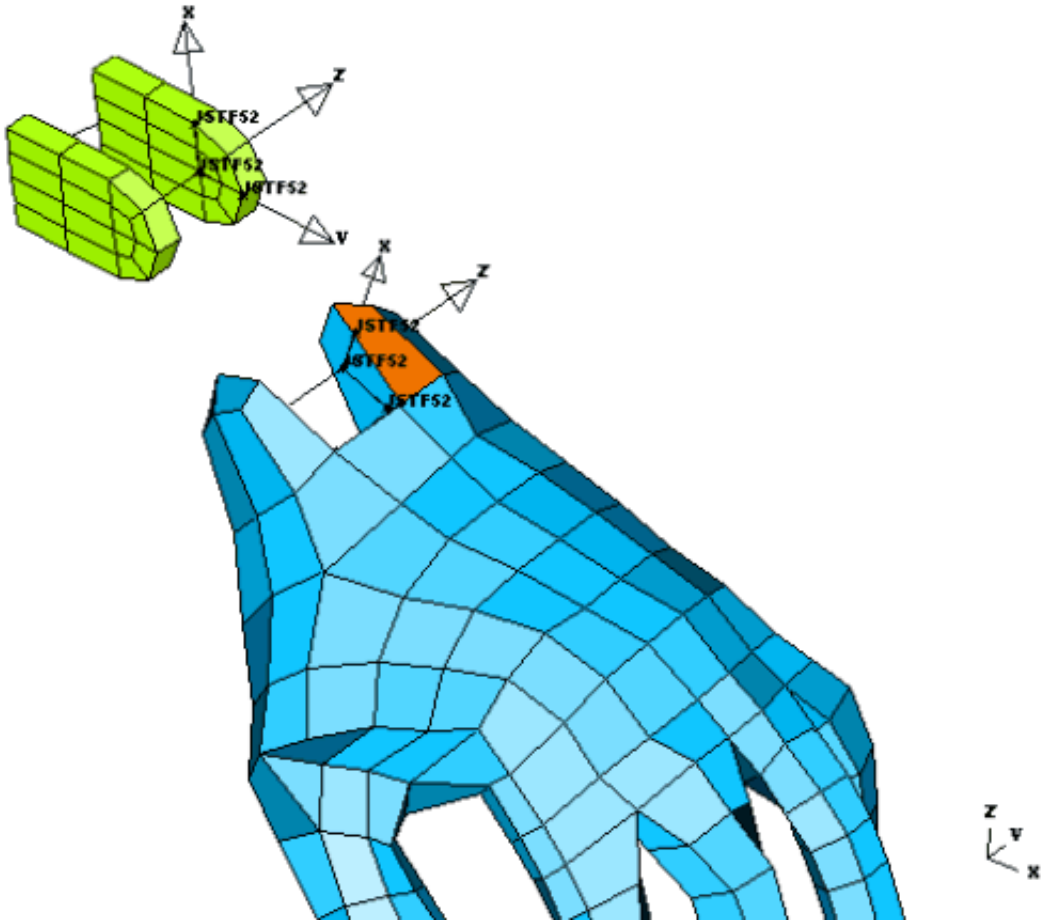
A "tree" file is made up of an extra set of keywords, and one such file for each dummy in a model is appended to the normal LS-DYNA (keyword) file after its *END card, where it is ignored by LS-DYNA, but read by PRIMER. A model may contain any number of dummies, and each must have a unique label within that model.

The purposes of appending the "tree" section to the analysis input deck are to try to make sure that a dummy and its tree data don’t get separated; that any renumbering which takes place is consistent in both dummy and tree file; and to permit adjustment of a dummy in a complete deck without having to go through the rigmarole of reading it in and repositioning it from scratch each time.



This figure illustrates "tree" file usage by showing an exploded diagram of the arm assemblies in a typical dummy. The assemblies are "upper arm", "elbow", "lower arm", "wrist" and "hand"; and the joint stiffnesses between them are shown.

It can be seen from the geometry of the connections that in this particular dummy rotation can only take place about one axis at each joint, and that the coordinate systems of the joint stiffnesses are aligned to these. It is also clear that angular rotations must be limited, which is done by using "stop angles" in the joint stiffnesses.



Here the connection between wrist and hand has been enlarged (still artificially separated) to show its organisation in more detail.

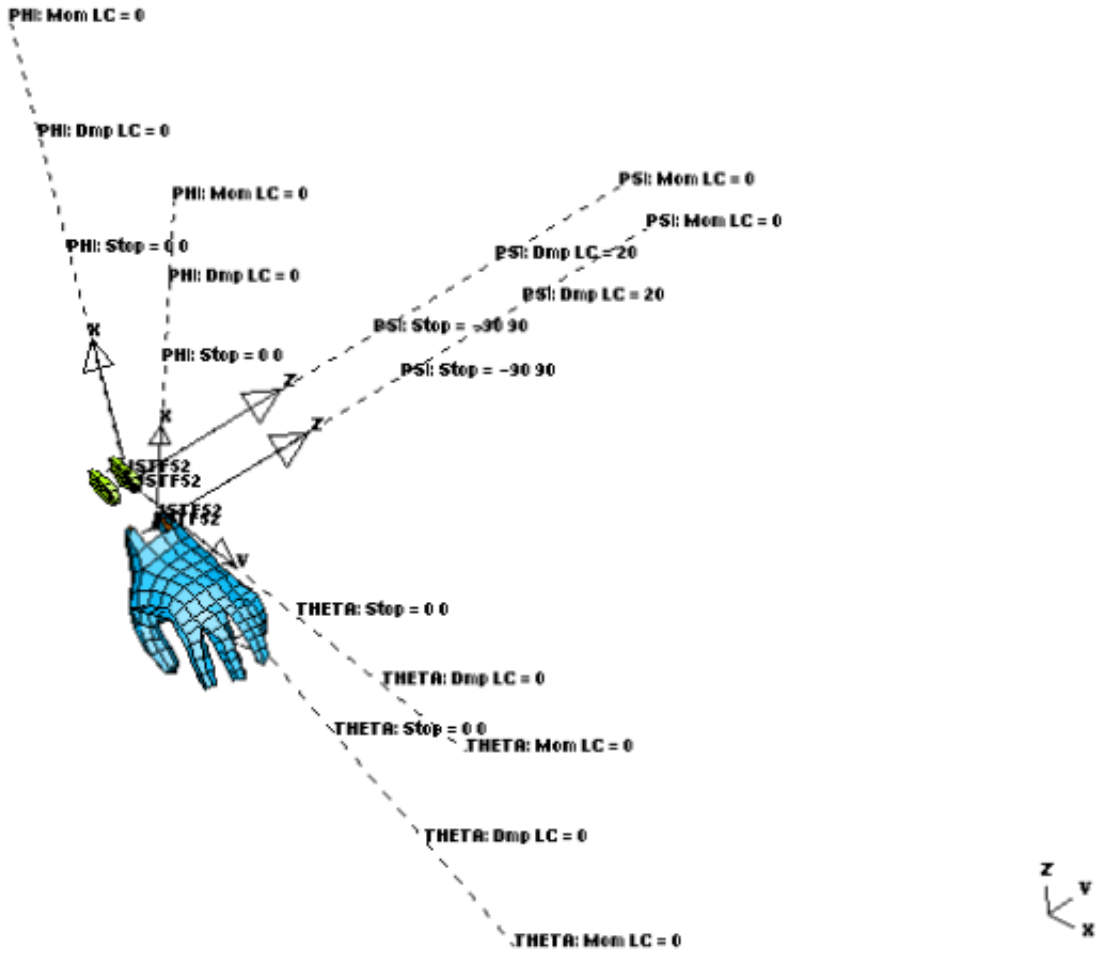
Clearly rotation of the hand can only take place in the "up and down" direction, which is about the local Z (Psi) axis of the joint stiffness.

The two sides of the joint stiffness definition can be seen in terms of their coordinate systems, and it is clear from this image that the following sensible modelling rules have been followed for them:

(1) Both coordinate systems have their origins at the same point in space. LS-DYNA does not require this, but it is difficult to visualise a connection clearly unless it is the case.

(2) Both coordinate systems are defined in terms of nodes attached to parts on their respective assemblies: either structural or extra nodes on rigid bodies. In this way the coordinate systems move with their parent parts.

(3) The node at the origin of the "parent" coordinate system on the wrist is the designated node about which the hand rotates. Again, not required but sensible.

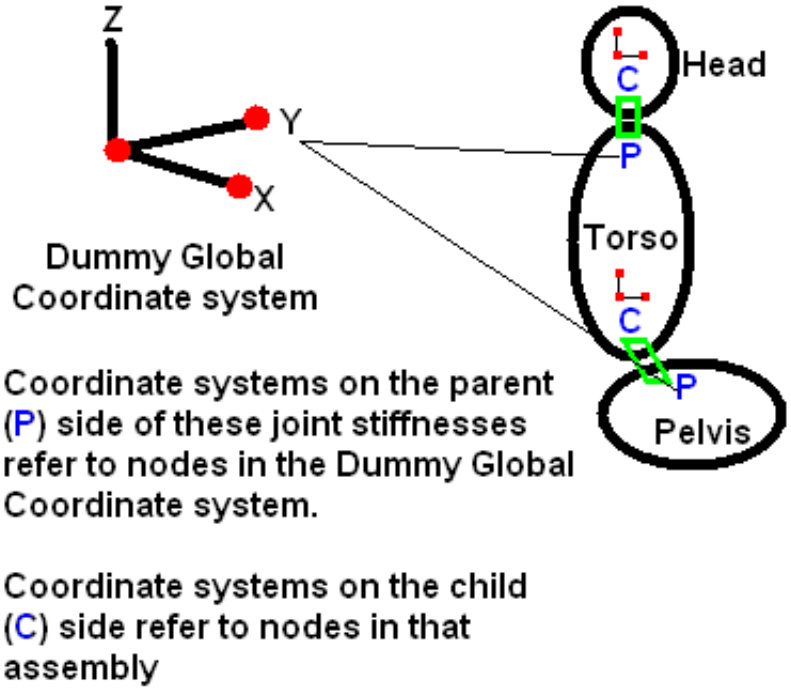


Here the **NOTATE** option in VIS_2 has been turned on to add to the joint stiffness graphics their loadcurves and stop angles. This shows that parameters have only been defined for rotation about local Z (Psi), that the "stop angles" are +/- 90 degrees, and that only damping is applied (so the initial angle between the coordinate systems generates no moment).

There is a separate revolute joint, not shown, through the wrist which constrains rotation to this axis only, so the joint stiffness does not need to restrict others. However to prevent PRIMER allowing positioning about axes other than Psi the "tree" file restricts rotation at this joint to local Z (axis 3). Here is the fragment of tree file defining wrist to hand connection:

```
*ASSEMBLY
        16Wrist left
         1         0         1
         9
$ Child #1 is the left hand (Assy #18, Joint stiff #52, rotates
$ about node #3980, rotation restricted to Z (3) axis)
        18        52      3980         3
```

From release 13 onwards the rule that "nodes used to define the coordinate system (used as the nodes in ***DEFINE_COORDINATE_SYSTEM_NODES**) on the parent side of a joint stiffness definition must be in the parent assembly" has been relaxed, and these nodes may also be in any assembly that is an ancestor of the child assembly. This change has been made since some dummies have been modelled in the following way:



This method of modelling means that angles for the core Pelvis, Torso and Head assemblies are always reported as "relative to global" rather than "relative to parent". This can make setup easier for some models when comparing to test since measuring the angle of an assembly with respect to some global reference, typically vertical, is easier than measuring relative to some other assembly.

This makes no difference to the way in which dummies work in PRIMER, only to the values of the reported angles.

The syntax of a "tree" file is as follows. All cards other than *DUMMY_START and *DUMMY_END are optional, and may appear in any order within these two _START/_END cards.

```
*DUMMY_START
<Label> <Title>
```

The following line is **optional**.

<move_xsec>

<Label>	I10	Must be unique within a model, as this identifies the dummy.	
<Title>	A70	An arbitrary character string describing the dummy.	
The line below is ***optional***, it was added in release 12. If it is omitted data fields will be given their default values.			
<move_xsec>	I10	Default = 1	Whether to move ***DATABASE_CROSS_SECTION_PLANE** definitions with parts in an assembly with the assembly as it is positioned. 1 = Yes, 0 = No.

```
*H_POINT
<Hx> <Hy> <Hz> <Root Assembly label> <Root initial X angle> <Root initial Y
angle> <Root initial Z angle>
```

The following 3 lines are **optional**. Either they must be omitted, or all 3 lines must be supplied:

```
<XX Cosine> <XY Cosine> <XZ Cosine>
<YX Cosine> <YY Cosine> <YZ Cosine>
<ZX Cosine> <ZY Cosine> <ZZ Cosine>
```

<Hx>	E10.0	X coordinate of dummy H-Point
<Hy>	E10.0	Y ditto
<Hz>	E10.0	Z ditto
<Root assembly label>	I10	***Optional***: The label of the assembly below that will be treated as containing the H point, and to which initial dummy orientations will apply. If this is omitted the H-Point will be assigned automatically to the lowest numbered assembly that has no parent.
<Initial X angle>	E10.0	***Optional***: Initial rotations about [X,Y,Z] axes for root assembly (degrees) By default the root assembly is assumed to have initial angles [0,0,0], and this is what will appear in the positioning panel if nothing is defined here, but you can optionally set initial angle values to be used instead. These will not affect the orientation of the root part, rather they can be considered to be initial numeric offsets. These angles will be updated when the dummy is positioned.
<Initial Y angle>	E10.0	
<Initial Z angle>	E10.0	
The following 3 rows of **optional** direction cosines were added in release 9.3 to define the orientation of the Dummy as a whole. They will only be written out if the Dummy's cosines are not a unit matrix, which will only be the case of the Dummy has been rotated either via Orient or by using the Rotate or Reflect options in the Dummy positioning panel itself. (Positioning individual Dummy assemblies will not modify these overall Dummy cosines.) If they are not present a set of unit cosines is assumed, implying that the Dummy as a whole has not been rotated.		
<X Cosines>	3E20.0	The X row of cosines: XX, XY, XZ
<Y Cosines>	3E20.0	The Y row of cosines: YX, YY, YZ
<Z Cosines>	3E20.0	The Z row of cosines: ZX, ZY, ZZ

The H-Point need not be defined. If this definition is absent then [0,0,0] is assumed. Rotations of any orphan assemblies, generally the root assembly, take place about the H-Point.

```
*UNITS
<mass unit> <length unit> <time unit>
```

<mass unit>	A10	A valid mass unit name: kg, Te, lb
<length unit>	A10	A valid length unit name: m, mm, in
<time unit>	A10	A valid time unit name: s

Units may be upper or lower case, anywhere in the A10 field. Units need not be consistent, eg kg, in, s is legal (if daft!). If this definition is absent then no units are assumed and the dummy is dimensionless.

```
*AXES
<Coord system label>
```

Csys label	I10	(Optional) An existing *DEFINE_COORDINATE system label.

This defines a local axis system for the dummy, which is used when calculating the rotation angles of the"root" part. If this definition is absent then the internal reference axes of the dummy will be aligned with the global cartesian axes when initially read in, so this card may be omitted if you want your root part's angles to be expressed as rotations about the global axes.

Some sort of reference system for root assembly rotations is required since this needs to rotate with the dummy if the whole dummy is oriented, otherwise root angles would change in a confusing manner.

```
*DYNA_POSITION
<node_1> <node_2>
```

Node label <node_1>	I10	Node at base of dummy neck
Node label <node_2>	I10	Node on dummy left or right hip

These nodes are used during the "Dyna method" positioning process. This keyword may be omitted if only PRIMER positioning is to be used.

```
*ASSEMBLY
<label>     <Title>
<#SET_PARTs> <#PARTs> <#children>  <#SET_NODES>  <locked>   (<csys>) <#contacts>
<dyna_pos>
<SET_PART_1> <SET_PART_2> ...      <SET_PART_n>
<PART_1>     <PART_2> ...          <PART_n>
<SET_NODE_1> <SET_NODE_2> ...      <SET_NODE_n>
<Part set>   <Box>      <tk factor><active> ... (1 line per contact)
<CHILD_1>    <JSTF_1>   <NODE_1a>  <dof codes>  (<NODE_1b>) (<soft angles>)
(<min_X>     <max_X>    <min_Y>    <max_Y>       <min_Z>     <max_Z>)
<CHILD_2>    <JSTF_2>   <NODE_2a>  <dof codes>  (<NODE_2b>) (<soft angles>)
(<min_X>     <max_X>    <min_Y>    <max_Y>       <min_Z>     <max_Z>)
 : : :
```

<Label>	I10	Label number for this assembly. This must be unique within this dummy, (but assemblies are "local" to a dummy, so the same label may occur in different dummies).	
<Title>	A70	Arbitrary name for this assembly.	
<#set_parts>	I10	The number of *SET_PARTs in this assembly	
<#parts>	I10	The number of *PARTs in this assembly	
<#children>	I10	How many "child" assemblies this assembly is "parent" to.	
<#set nodes>	I10	The number of *SET_NODES in this assembly	
<locked>	I10	Locked degrees of freedom during mechanism-style positioning. Any permutation of 123456, or 0 for none.	
<csys>	I10	Optional local coordinate system for assembly restraints during mechanism-style positioning	
<#contacts>	I10	Number of contact definitions	
<dyna_pos>	I10	Dyna position data flag (1 if data to be read, 2 if assembly is also flagged to be rigidified)	
<set_part_1...	8I10	Define <#set_parts> entries	8 entries per line, using as many lines as required
<part_1 ...	8I10	Define <#parts> entries	8 entries per line, using as many lines as required
<set_node_1...	8I10	Define <#set_nodes> entries, 8 to a line *(The option of Node Sets in assemblies is new in PRIMER release RC2)*	8 entries per line, using as many lines as required

An assembly may be made up of any number of SET_PARTs and/or PARTs and/or SET_NODEs, whichever is more convenient. Parts may be defined more than once, ie occur both explicitly and in sets, only a single instance will be used. For visualisation purposes you should have at least one part since node sets are not easy to interpret visually.			
<Part set>	I10	Part set for contact	<#contacts> lines of data, each definition starts a new line.
<Box>	I10	Optional box to delimit contact	
<tk factor>	E10.0	Factor on true shell thickness for contact purposes	
<active>	I10	Flag to denote contact active (1) or inactive (0)	
<child_n>	I10	The <nth> child assembly label. Required.	
<jstf_n>	I10	The GENERALIZED_STIFFNESS connecting parent assembly to child <n>. If omitted free rotation about dummy axes is assumed unless node NB is defined.	
<node_na>	I10	The NODE on the parent assembly about which child <n> rotates. If omitted the child rotates about the dummy H-Point.	
<dofs_n>	I10	The degrees of freedom about which child <n> may rotate with respect to the parent. The codes are "1", "2", "3" for local Phi, Theta, Psi (x,y,z) axes respectively. Given in any order, for example 13 = rotation permitted about Phi and Psi. If omitted no rotation about any axes will be permitted.	
<node_nb>	I10	Optional second node. If no joint stiffness is defined then if NB is defined the rotation axis will be about the vector NA - NB.	
<soft_angles>	I10	***Optional***. If zero, or omitted, then the second line of "soft stop angles" is not read. If 1 then the soft angles line is read.	
The following line is only read if field **<soft_angles>** above is set to 1. It defines alternative "soft" stop angles for the 3 degrees of freedom of rotation of the child to be used during positioning, instead of the stop angles on the Generalized Stiffness card. The purpose of these angles is to limit articulation during positioning so that penetrations of the structure on either side of the joint do not occur. They are ignored during actual analysis where the "true" stop angles on the Generalized stiffness cards will be used.			
<min_X>	E10.0	Most -ve permitted "soft" rotation angle about local X axis	These values are defined in degrees, must lie in the range +/-180.0, and the -ve value must be less than or equal to the +ve value. If ***both*** "soft" stop angles for a given degree of freedom are 0.0 (or omitted) then the "true" stop angles on the Generalized Stiffness card will be used instead for that degree of freedom.
<max_X>	E10.0	Most +ve permitted "soft" rotation angle about local X axis	
<min_Y>	E10.0	Most -ve permitted "soft" rotation angle about local Y axis	
<max_Y>	E10.0	Most +ve permitted "soft" rotation angle about local Y axis	
<min_Z>	E10.0	Most -ve permitted "soft" rotation angle about local Z axis	
<max_Z>	E10.0	Most +ve permitted "soft" rotation angle about local Z axis	

This ability to define "soft" stop angles is available in PRIMER release 12 and later. The format is backwards compatible, and decks from older releases will read without modification.

The single line if <soft_angles> is zero or omitted

```
<child> <jstf>  <node_a> <dofs>  <node_b>
```

or the pair of lines if **<soft_angles>** = 1

```
<child> <jstf>  <node_a> <dofs>  <node_b> <soft_angles>
<min_X> <max_X> <min_Y>  <max_Y> <min_Z>  <max_Z>
```

is repeated for **<#children>** child assemblies

The following line is only read if field **<dyna_pos>** on line #2 is non-zero. It defines the three nodes to be used for positioning by the LS-DYNA method.

<node_1>	I10	Dyna positioning node #1
<node_2>	I10	Dyna positioning node #2
<node_3>	I10	Dyna positioning node #3

```
*POINT_NODE
<title>
<assembly id> <node id> <restrained DoFs> (<csys>) (<h_pt>)
```

<title>	A80	Title for the point
<assembly id>	I10	Label of assembly to which point is attached
<node id>	I10	Label of node from which point coordinates are taken
<restrained DoFs>	I10	A restraint code made up of any permutation of 123456, or 0 for none
<csys>	I10	Optional: a coordinate system to give restraint in local axes.
<h_pt>	I10	Optional: set to 1 if this is an automatically generated point at the dummy H-Point

```
*POINT_LOCATION
<title>
<assembly id> <px> <py> <pz> <restrained DoFs> (<csys>) (<h_pt>)
```

<title>	A80	Title for the point
<assembly id>	I10	Label of assembly to which point is attached
<px>	E10.0	X coordinate of point
<py>	E10.0	Y coordinate of point
<pz>	E10.0	Z coordinate of point
<restrained DoFs>	I10	A restraint code made up of any permutation of 123456, or 0 for none
<csys>	I10	Optional: a coordinate system to give restraint in local axes.
<h_pt>	I10	Optional: set to 1 if this is an automatically generated point at the dummy H-Point

Points are optional. They are relevant only during mechanism-style positioning, and provide two related functions:

- By restraining degrees of freedom at a point an assembly can be given a "point" restraint. This can be in a local system if <csys> is defined for the point.

- Points can also be used to "drive" mechanism-style movement by giving them new target coordinates and letting PRIMER iterate to achieve these.

Any number of points may be defined for a dummy, and an assembly may have any number attached to it.

`*POSITION`

Any number of positions may be stored for a dummy, and position information is identical for dummies and mechanisms. A description of these and their card format is given below.

***DUMMY_END**

Terminates the dummy definition. This is assumed if a physical <end of file> is found, but is mandatory if a second dummy definition (or other keyword) is to follow.

Rules for "tree" files.

Syntax:

*- Syntax is based on LS-DYNA keyword format. Thus numbers should be right-justified in their respective fields, and comment lines ($....) may be inserted anywhere. Character strings should be left justified.

*Any **`*DUMMY_...`** cards must appear after the **`*END`** card in an input file, as LS-DYNA will not recognise them.

*- Only the keywords described above may appear between the **`*DUMMY_START`** and **`*DUMMY_END`** cards.

Numbering:

*There are no restrictions on the labels for dummies or assemblies, other than that they must be valid, positive integers.

*Dummy numbers must be unique: you cannot have two dummies numbered "2" within a model.

*Assembly numbers within a dummy must be unique: you cannot have two assemblies numbered "10" within a dummy. Assembly numbers are "private" to their parent dummy definition. Thus if you have two dummies in a model both may have an assembly numbered "8" without there being any conflict.

*Thus when dummies are merged into a model, or models already containing dummies are merged, the dummy numbers may need to be incremented but the assembly numbering within a model will not be changed. Note that the merge operation may result in the numbering of part/set/node/etc entities being changed, this will be reflected in the dummy definition too.

*Assemblies can only refer to PARTs, SET_PARTs, NODEs, JSTFs, etc that are within their current model. Thus a dummy tree file definition cannot be used independently of a model and, more importantly, when transferring dummy definitions between input decks by hand take great care to ensure that the node/part/set/etc numbering in the new deck matches that in the old one. (If PRIMER is used to merge decks this is handled automatically.)

Modelling rules:

Problems are likely to arise if "free-standing" items such as **`*DEFINE_BOX`**, **`*DEFINE_COORDINATE_SYSTEM`**s defined other than by nodes, and such like, are used.

The reason is that these items do not belong unambiguously to PARTs, so they may not be flagged for rotation/translation/reflection when a part is so operated upon. As a consequence orienting a dummy or its assemblies may move items in or out of them. Therefore it is recommended that:

`*DEFINE_BOX`	Should not be used, since this may not be oriented correctly. Contacts, walls, etc should avoid selecting slave entities by volume, and use instead PART, SET, SEGMENT or NODE ids.
`*DEFINE_VECTOR`	Should also not be used, for the same reasons.
`*DEFINE_COORDINATE`	Should only be used in its _NODES variant, and the nodes employed should be attached to the parent PART that is supposed to control its orientation. The attachment method doesn't matter: explicit nodes on elements, extra nodes on rigid bodies, rigid body merges, nodal rigid bodies, etc; so long as the nodes are subordinate to the PART in question.

***DEFINE_SD_ORIENT..** Can be used so long as some thought is given to its use. These orientation vectors are unambiguously subordinate to their parent DISCRETE elements, so they meet the requirement that they are attached to PARTs. But, obviously, problems will arise if a single orientation vector is used for springs in different PARTs which may be rotated independently. You should use separate orientation vectors for all springs unless it is clear that a group of springs will always be rotated together (effectively rigidly). It is not mandatory to use the two node method for defining these vectors, although this may help to make the display of them clearer since it will both locate them in space and give them an explicit length**.**

The restrictions on **BOX**es and **VECTOR**s may be eased in the future if a method of defining them by nodes becomes available. This will move them to the same status as the ***DEFINE_COORDINATE_NODES** option.

***MAT_RIGID** May be used, but if the options to constrain the material or request output in local coordinate systems are used, then separate ***MAT_RIGID** definitions should be repeated for each assembly of parts. For restraining motion of rigid bodies it might be preferable to use the ***BOUNDARY_PRESCRIBED_MOTION_RIGID** method, but not its **_LOCAL** variant!

***MAT_xx orthotropic** Where orthotropic materials are used problems will only arise if global orthotropicity is defined. This presents the same problem as above in a different guise, since different PARTs sharing a common material can only have a single set of material axes defined. Again, the solution is to have separate material definitions for each part (or rigidly connected set of parts).

These two restrictions on the use of material models may be removed in the future, as PRIMER may detect this conflict and generate separate material definitions automatically; but for the time being they stand, and represent good modelling practice anyway.

A **PART** may not appear in more than one assembly in a dummy, otherwise a conflict will arise when it is positioned because more than one assembly will be trying to update the part's nodal positions. This will be detected as an error and the dummy definition will be rejected.

Similarly a **NODE** should not have its motion "driven" during positioning by more than one assembly. This could happen if a part defined in assembly A extended to include elements and/or nodes common with parts in assembly B. This too will be detected and flagged as an error. However there are some specific exceptions to this rule:

1. **Extra nodes on a rigid part in assembly A may legally be part of structure in assembly B.**

   This is common modelling practice where two assemblies need to be able to articulate during positioning, but to be clamped together during analysis. The positioner detects this situation and ensures that motion of assembly A does not update the coordinates of its "extra" nodes in assembly B.

2. **Rigid parts in assembly B that are slaves to a master part in Assembly A are permitted.**

   This is also common modelling practice to achieve articulation during positioning, but a rigid connection during analysis. Normally PRIMER carries motion of a master rigid part through to its slaves, but in the positioner this is detected and the slave part's nodes will not be updated if they are found to be present on a different assembly to that of the master part.

3. **If a part consisting of null shells (material type 9) is used to "skin" the Dummy in order to give a continuous surface for contact it is legal for this part to cover more than one assembly.**

   Such a part will not be included when nodal coordinates are updated following positioning, so the assemblies in question should also include "structural" parts (as would be required during analysis anyway). However such a part may still only be defined once in the dummy, even if its connectivity spans multiple assemblies. However there is an exception to this exception:

   Exception: "Isolated" null shell parts *are* included when an assembly is positioned.

   Experience with working dummies has shown that some modellers build target markers into their dummies by creating targets made out of null shell parts, attached to the dummy by tied contacts. The "exclude null shell parts" rule implicit in exception (3) above meant that these target markers got left behind when the dummy moved since their nodes were not on parts in the assembly.

   To fix this problem null shell parts ***are*** moved with an assembly if their nodes are not common to any "structural" (meaning on non-null parts or on node sets) nodes in the assembly. The restriction that such parts should not span more than one assembly remains, meaning that target markers defined in this way must have separate null shell parts for each assembly's markers.

4. **Orientation ("3rd") nodes on beams in assembly A that are located in assembly B may *optionally* be ignored.**

   Normally it would be both illegal and also extremely poor modelling practice to locate the 1st two nodes of a

beam in one assembly and its "3rd node" in another one. However there are occasions when this might occur, for example when beams are circular in section so their orientation is not important, and any old node will suffice for orientation.

The pre-positioning check will detect this and warn you, and strictly speaking you should correct the model to eliminate the problem. However there is an "**Ignore B3**" option following the check which, if chosen, will suppress propagation to 3rd nodes of beams. This setting is "remembered" for the Dummy or Mechanism for the duration of this PRIMER session so that the warning will not pop up again each time you perform a positioning operation.

This should be regarded as a quick and dirty way to deal with an annoying problem, and not as a solution. In particular if you are preparing models for others to use you should not rely on this flag since your future users will have to select it every time they position your model.

5. **PRIMER connections between assemblies may *optionally* be ignored**

Connections in PRIMER may be "made" if they have been explicitly realised, or "potential" if they would be realised via a pass through the Connection menu. The Dummy and Mechanism positioner considers a "made" connection between two assemblies to be something that links them together, and it will give a warning about this in the pre-positioning check. However it will not consider "potential" connections in the same location.

You can choose to ignore connections in the pre-positioning check via the "**Ignore Conn**" option, but you need to think through what this will imply. In particular a "made" connection which straddles two assemblies may be pulled apart when they are positioned causing it to become invalid. For example a spotweld beam may not longer be tied to its surface, or a weld may become too long.

On the whole it is probably best to avoid joining Dummy and Mechanism assemblies via PRIMER connections, but if it is to be done this way then it is recommend that they are rechecked when the positioning process is completed.

The list above is almost certainly not exhaustive: builders of dummy models should think through carefully what happens when assemblies of PARTs are oriented independently of each other.

Example of a "tree" file

This example refers to a typical dummy. Each assembly below has been defined (in the main body of the input deck) using ***SET_PART**s.

```
$
$
*DUMMY_START
1This is a test DUMMY definiton
$
*H_POINT
       0.0        0.0        0.0
$
*UNITS
        te         mm          s
$
*ASSEMBLY
         1Neck, Thorax, Torso
         3          0           5
        21         22          23
$
$ Child #1 is the head
         2          9       20589        123
$ Child #2 is the left yoke
         9          4       10597        123
$ Child #3 is the right yoke
        10          5       14963        123
$ Child #4 is the Left upper leg
         3         16        2252        123
$ Child #5 is the Right upper    leg
         4         17        2044        123
$
$
*ASSEMBLY
         2Head
         1          0           0
        20
```

```
$
$
*ASSEMBLY
         3Upper leg left
         1          0           1
        24
$
$ Child #1 is the left knee/lower    leg
         5          2        4329        123
$
$
*ASSEMBLY
         4Upper leg right
         1          0           1
        25
$
$ Child #1 is the left knee/lower    leg
         6          1        4320        123
$
$
*ASSEMBLY
         5Lower leg left
         1          0           1
        26
$
$ Child #1 is the left ankle &    foot
         7         22         4562        123
$
$
*ASSEMBLY
         6Lower leg right
         1          0            1
        27
$
$ Child #1 is the right ankle    & foot
         8         23         5437        123
$
$
*ASSEMBLY
         7Foot left
         1          0            0
        28
$
*ASSEMBLY
         8Foot right
         1           0            0
        29
$
*ASSEMBLY
         9Yoke left
         1           0            1
        30
$
$ Child #1 is the upper left arm
        11          18        18559        123
$
$
*ASSEMBLY
         10Yoke right
         1           0             1
        31
$
$ Child #1 is the upper right    arm
        12          19         19281        123
$
$
*ASSEMBLY
         11Upper arm left
         1           0             1
        32
$
```

```
$ Child #1 is the left elbow
        13          11         19138        123
$
$
*ASSEMBLY
         12Upper arm right
         1           0             1
        33
$
$ Child #1 is the right elbow
        14          14         19860        123
$
$
*ASSEMBLY
         13Elbow left
         1           0             1
        34
$
$ Child #1 is the lower arm left
        15          10         19136         123
$
$
*ASSEMBLY
         14Elbow right
         1           0             1
        35
$
$ Child #1 is the lower arm right
        16          13         19858         123
$
$
*ASSEMBLY
         15Lower arm left
         1           0             1
        36
$
$ Child #1 is the left wrist
        17          12         19140         123
$
$
*ASSEMBLY
         16Lower arm right
         1           0             1
        37
$
$ Child #1 is the right wrist
        18          15         19862         123
$
$
*ASSEMBLY
         17Wrist left
         1           0             1
        38
$
$ Child #1 is the left hand
        19          20         18888         123
$
$
*ASSEMBLY
         18Wrist right
         1           0             1
        39
$
$ Child #1 is the right hand
        20          21         19610         123
$
$
*ASSEMBLY
         19Hand left
         1
        40
```

```
$
*ASSEMBLY
        20Hand right
         1
        41
$
*DUMMY_END
```

APPENDIX IIb: Mechanism file format.

Mechanisms are similar to Dummies in many ways: they share a common Assembly and Point definition syntax, and when a Dummy is "free dragged" in mechanism-style positioning PRIMER automatically builds an internal mechanism to perform the drag.

However there are also some significant differences:

- Mechanisms do not have the hierarchical parent/child structure of dummies, as their connectivity can be completely arbitrary. Therefore although their keyword card formats are similar it would be misleading to refer to them as a "tree file".
- Because of the absence of a hierarchy Mechanism assemblies are a little different different from Dummy ones, in particular they do not define "child" assemblies.
- Mechanism assemblies are joined together by "Connections" which have to be defined explicitly, whereas Dummy connectivity is defined by the "child" lines on Assembly cards.
- Finally Mechanisms can have child mechanisms or dummies, whereas dummies cannot.

It is not expected that mechanisms will be created outside PRIMER as defining them interactively within the programme is very easy, so the card formats are included mainly for completeness.

```
*MECHANISM_START
<label> <title>
```

The following line is **optional**.

<move_xsec> <mcon_labels>

<Label>	I10	Must be unique within a model, as this identifies the Mechanism.	
<Title>	A70	An arbitrary character string describing the Mechanism.	
The line below is ***optional***, it was added in release 12. If it is omitted data fields will be given their default values.			
<move_xsec>	I10	Default = 1	Whether to move ***DATABASE_CROSS_SECTION_PLANE** definitions with parts in an assembly with the assembly as it is positioned. 1 = Yes, 0 = No.
<mcon_labels>	I10	Default = 1	Whether mechanism connections are written with labels. This option has been added in Primer 15.0 to make it possible to reference line or hinge connections by their label from a coupler connection. When writing a keyword file from Primer, this option will be turned on (set to 1) if and only if there will be coupler connections written for this mechanism definition. 1 = Yes, 0 = No.

This card starts a new Mechanism definition, giving its label and title, and possibly further optional values. All cards between this and the corresponding ***MECHANISM_END** are "private" to this mechanism definition.

```
*ASSEMBLY
<label>     <Title>
<#SET_PARTs> <#PARTs>    <unused>    <#SET_NODEs>  <locked>   <csys>
<#contacts>
<SET_PART_1> <SET_PART_2> ...        <SET_PART_n>
<PART_1>     <PART_2> ...            <PART_n>
<SET_NODE_1> <SET_NODE_2> ...        <SET_NODE_n>
```

```
<Part set>   <Box>       <Tk factor> <active>
 : : :
```

<Label>	I10	Label number for this assembly. This must be unique within this mechanism, (but assemblies are "local" to a mechanism, so the same label may occur in different mechanisms).	
<Title>	A70	Arbitrary name for this assembly.	
<#set_parts>	I10	The number of *SET_PARTs in this assembly	
<#parts>	I10	The number of *PARTs in this assembly	
<unused>	I10	This field is unused for mechanism definitions (see note 1 below)	
<#set_nodes>	I10	The number of *SET_NODE definitions in this assembly	
<locked>	I10	Locked degrees of freedom during positioning. Any permutation of 123456, or 0 for none.	
<csys>	I10	Optional local coordinate system for assembly restraints during positioning	
<#contacts>	I10	Number of contacts between assemblyand fixed "structure"	
<set_part_1...	8I10	Define <#set_parts> entries.	8 entries per line, using as many lines as required
<part_1 ...	8I10	Define <#parts> entries.	8 entries per line, using as many lines as required
<set_node_1...	8I10	Define <#set_nodes> entries	8 entries per line, using as many lines as required
Note 1:	PRIMER version 9.3RC1 contained Nodal Rigid Bodies in this "slot". These have been withdrawn and replaced with Set Nodes since the latter are more flexible and permit nodes to be defined explicitly in assemblies. Models which contain Nodal Rigid Bodies in assemblies may be converted to the new format by replacing the nodal rigid body labels with the labels of their node sets, which in many cases will be identical. The behaviour of the assembly during mechanism analysis will be identical.		
An assembly may be made up of any number of SET_PARTs and/or PARTs and/or SET_NODEs, whichever is more convenient. Parts may be defined more than once, ie occur both explicitly and in sets, only a single instance will be used. Assemblies should contain at least one part otherwise visualising and dragging them may prove difficult.			
<Part set>	I10	Part set for contact	<#contacts> lines of data, each definition starts a new line.
<Box>	I10	Optional box to delimit contact	
<tk factor>	E10.0	Factor on true shell thickness for contact purposes	
<active>	I10	Flag to denote contact active (1) or inactive (0)	

```
*CONNECTION_PIN
<Title>
<label>
<assy_1> <assy_2> <node> <locked>
```

<title>	A80	Optional title for connection	
<label>	I10	Label for connection. This data row should be included if and only if the mcon_labels flag is 1 on the *MECHANISM_START card. In particular it did not appear before Primer 15.0.	
<assy_1>	I10	Assembly #1	
<assy_2>	I10	Assembly #2	
<node>	I10	Node at connection position.	
<locked>	I10	0 for unlocked joint, 1 for locked.	
<cx>	E10.0	Connection position X coord	**Note**: These fields are only present in V10 onwards. From V10.0 onwards a pin location may be defined either by a node or by an explicit position. If a node is defined it is used, regardless of any position <cx,cy,cz>, otherwise the stipulated position is used.
<cy>	E10.0	Connection position Y coord	
<cz>	E10.0	Connection position Z coord	

```
*CONNECTION_LINE
<Title>
<label>
<assy_1>  <assy_2>  <node_1>     <node_2> <pos_slide> <neg_slide> <cur_dist>
<locked>/<-1>/<-2>
<pos_rot> <neg_rot> <curr_angle> <locked> <a3_active> <assy_3>    <factor_1>
<factor_2>
<c1_x>    <c1_y>    <c1_z>       <c2_x>   <c2_y>      <c2_z>
```

<title>	A80	Optional title for connection	
<label>	I10	Label for connection. This data row should be included if and only if the mcon_labels flag is 1 on the *MECHANISM_START card. In particular it did not appear before Primer 15.0.	
<assy_1>	I10	Assembly #1	
<assy_2>	I10	Assembly #2	
<node_1>	I10	First node on line	
<node_2>	I10	Second node on line	
<pos_slide>	E10.0	Permitted slide distance in +ve direction	
<neg_slide>	E10.0	Permitted slide distance in -ve direction	
<cur_dist>	E10.0	Current slide distance	
<locked> or <-1> or <-2>	I10	0 for unlocked joint, 1 for locked. (9.3RC1 format) -1 to signify continuation in 9.3RC2 format -2 to signify continuation in 10.0 format	**Note**: Format of this card changed between 9.3RC1 and RC2. And changed again with 10.0

<pos_rot>	E10.0	Permitted +ve rotation (degrees: 0 to +180.0)	**Note**: This continuation line is 9.3RC2 format & later only
<neg_rot>	E10.0	Permitted -ve rotation (degrees 0 to -180.0	
<curr_angle>	E10.0	Current rotation angle (degrees)	
<locked>	I10	0 for unlocked joint, 1 for locked.	
<a3_active>	I10	1 if Assembly #3 active	**Note**: these fields are only present from V10.0 onwards
<assy_3>	I10	Assembly #3	
<factor_1>	E10.0	Factor on <assy_1> motion	
<factor_2>	E10.0	Factor on <assy_2> motion	
<c1_x>	E10.0	Point 1 X coordinate	**Note**: This card is only present from V10.0 onwards, signified by <-2> in column 8 of the 1st card. From V10.0 onwards either or both locations may be defined either by a node or by an explicit position. If a node is defined it is used, regardless of any position <cx,cy,cz>, otherwise the stipulated position is used.
<c1_y>	E10.0	Point 1 Y coordinate	
<c1_z>	E10.0	Point 1 Z coordinate	
<c2_x>	E10.0	Point 2 X coordinate	
<c2_y>	E10.0	Point 2 Y coordinate	
<c2_z>	E10.0	Point 2 Z coordinate	

```
*CONNECTION_HINGE
<Title>
<label>
<assy_1>  <assy_2>  <node_1>     <node_2> <locked>/<-1>/<-2>
<pos_rot> <neg_rot> <curr_angle> <locked>
<c1_x>    <c1_y>    <c1_z>       <c2_x>   <c2_y>      <c2_z>
```

<title>	A80	Optional title for connection
<label>	I10	Label for connection. This data row should be included if and only if the mcon_labels flag is 1 on the *MECHANISM_START card. In particular it did not appear before Primer 15.0.
<assy_1>	I10	Assembly #1
<assy_2>	I10	Assembly #2
<node_1>	I10	First node on line
<node_2>	I10	Second node on line

<locked> or <-1> or <-2>	I10	0 for unlocked joint, 1 for locked. (9.3RC1 format) -1 to signify continuation in 9.3RC2 format -2 to signify continuation in 10.0 format	**Note**: Format of this card changed between 9.3RC1 and RC2. And changed again with 10.0
<pos_rot>	E10.0	Permitted +ve rotation (degrees: 0 to +180.0)	**Note**: This continuation line is 9.3RC2 format only
<neg_rot>	E10.0	Permitted -ve rotation (degrees 0 to -180.0	
<curr_angle>	E10.0	Current rotation angle (degrees)	
<c1_x>	E10.0	Point 1 X coordinate	**Note**: This card is only present from V10.0 onwards, signified by <-2> in column 5 of the 1st card. From V10.0 onwards either or both locations may be defined either by a node or by an explicit position. If a node is defined it is used, regardless of any position <cx,cy,cz>, otherwise the stipulated position is used.
<c1_y>	E10.0	Point 1 Y coordinate	
<c1_z>	E10.0	Point 1 Z coordinate	
<c2_x>	E10.0	Point 2 X coordinate	
<c2_y>	E10.0	Point 2 Y coordinate	
<c2_z>	E10.0	Point 2 Z coordinate	

```
*CONNECTION_COUPLER
<Title>
<label>
<mcon_1>  <mode_1>  <coeff_1>
<mcon_2>  <mode_2>  <coeff_2>
<mcon_3>  <mode_3>  <coeff_3>
```

This card has been added from Primer 15.0 onwards.

<title>	A80	Optional title for connection
<label>	I10	Label for connection
<mcon_1>	I10	Connection #1. This needs to be either a line or a hinge.
<mode_1>	I10	Coupling mode #1. This should be 0 for translational coupling or 1 for rotational coupling. If connection #1 is a hinge, this needs to be 1.
<coeff_1>	E10.0	Coefficient c1 in the defining equation for the coupler
<mcon_2>	I10	Connection #2. This needs to be either a line or a hinge.
<mode_2>	I10	Coupling mode #2. This should be 0 for translational coupling or 1 for rotational coupling. If connection #2 is a hinge, this needs to be 1.
<coeff_2>	E10.0	Coefficient c2 in the defining equation for the coupler

<mcon_3>	I10	Optional connection #3. If defined, this needs to be either a line or a hinge.
<mode_3>	I10	Coupling mode #3. This should be 0 for translational coupling or 1 for rotational coupling. If connection #3 is a hinge, this needs to be 1. This will be ignored when connection #3 is 0.
<coeff_3>	E10.0	Coefficient c3 in the defining equation for the coupler. This will be ignored when connection #3 is 0.

```
*POINT_NODE
<title>
<assembly id> <node id> <restrained DoFs> (<csys>)
```

<title>	A80	Title for the point
<assembly id>	I10	Label of assembly to which point is attached
<node id>	I10	Label of node from which point coordinates are taken
<restrained DoFs>	I10	A restraint code made up of any permutation of 123456, or 0 for none
<csys>	I10	Optional: a coordinate system to give restraint in local axes.

```
*POINT_LOCATION
<title>
<assembly id> <px> <py> <pz> <restrained DoFs> (<csys>)
```

<title>	A80	Title for the point
<assembly id>	I10	Label of assembly to which point is attached
<px>	E10.0	X coordinate of point
<py>	E10.0	Y coordinate of point
<pz>	E10.0	Z coordinate of point
<restrained DoFs>	I10	A restraint code made up of any permutation of 123456, or 0 for none
<csys>	I10	Optional: a coordinate system to give restraint in local axes.

***CHILD_DUMMY** *or* ***CHILD_MECHANISM** *(Card format is the same for both)*

```
<title>
<parent assy> <child label> <nslaved> <linked DoFs> <locked>
<child assy #1>
<child assy #2>
: : :
```

<title>	A80	Optional title
<parent assy>	I10	Label of "driving" assembly in parent mechanism
<child label>	I10	Label of "child" mechanism or dummy definition.
<nslaved>	I10	Number of assemblies in child that are "slaved" to this master
<linked DoFs>	I10	Any permutation of 123 giving degrees of freedom linking master to child.
<locked>	I10	1 if child is fully locked to master

<child assy *n*>	I10	Linked child assemblies 1 to <nslaved>, 1 per line.

***POSITION**

Any number of positions may be stored for a mechanism, and position information is identical for dummies and mechanisms. A description of these and their card format is given below.

***MECHANISM_END**

Terminates the mechanism definition.

APPENDIX IIc: "Positions" in Dummy and Mechanism data.

Both Dummies and Mechanisms may have any number of saved "positions" stored within their data cards. The format used is identical for both types, and all the information below applies equally to both.

Position card format

```
*POSITION
<title>
```

then for each assembly

```
<Assembly id #1>
<Cx> <Cy> <Cz>
<Xx> <Xy> <Xz>
<Yx> <Yy> <Yz>
<Zx> <Zy> <Zz>
```

... and so on for assemblies #2 to #n in this Dummy or Mechanism

<title>	A80	Title for position (required, must be unique)
<Assembly id>	I10	Assembly label
<Cx> <Cy> <Cz>	3E20.0	Notional centroid of assembly
<Xx> <Xy> <Xz>	3D20.0	X components of direction cosines
<Yx> <Yy> <Yz>	3D20.0	Y components of direction cosines
<Zx> <Zy> <Zz>	3D20.0	Z components of direction cosines

Data stored for Positions

As the card format above shows a "Position" stores a title, then a centroid and 3x3 set of direction cosines for each assembly in the Dummy / Mechanism. The direction cosines are written in double precision.

The "reference" position

The first time a position is **Saved** an extra "reference position", the current configuration, is created and all user-defined positions follow this. This always represents the current position and orientation of the dummy assemblies, and it is updated whenever positioning takes place to keep it current. It is marked "do not delete" because it is required for saving and retrieving of other positions to work correctly. A moment's thought about the process to be gone through when retrieving a saved position will show why this is necessary.

Firstly consider how the current orientation of each Dummy and Mechanism assembly is defined and maintained:

- Dummy and Mechanism assemblies do not have an intrinsic coordinate system when first created.
- Therefore the positioner assigns a coordinate system to each assembly the first time it is used. This consists of

  A notional centroid. This is actually the average nodal coordinate rather than the true centre of gravity, but it serves its purpose.
  An orientation defined by a rotation matrix of direction cosines. This is the "inital orientation" of the assembly, and it always starts off as a unit matrix.
- Thereafter any positioning operation which moves the assembly will update both centroid and rotation matrix.
- The "current", as in what you see on the screen and what would be written out to a keyword file, position of each assembly is stored as its "reference position".

Now consider what happens when the current position is saved:

- An exact copy of the current centroid and rotation matrix is saved as an explicit *POSITION

Finally consider what happens when a stored *POSITION is retrieved, which you can think of as working like this:

- Firstly the current position of the dummy, as defined by the current reference position, is "undone" to get back to the original position and orientation when first seen by the positioner.
- Then the translation and orientation implied by the saved *POSITION is applied to restore the saved status and achieve the new shape.

In practice matrix arithmetic allows this do be done by a single transformation of "current" to "new", but the key point is that you must know the "current" position in order to achieve the correct"new" position. This is why the reference position is always maintained, and it will be obvious that deleting it means that PRIMER will no longer know what the current position is, and therefore will be unable to restore a saved position correctly.

Positions are relative within the Dummy / Mechanism

Positions are always stored relative to the "reference position", making it possible for **Retrieve** to restore any previous configuration.

This also means that "global" orientations of the whole Dummy / Mechanism are effectively cancelled out, making saved positions "local". For example if a saved position raises a left hand relative to the wrist this will still be the case even if the Dummy as a whole has been translated and rotated to some totally different location.

Re-using Position data when geometry changes

Position data generates a notional centroid, the average coordinate of the bounding box round an assembly, the first time a position is saved and thereafter this value is "remembered". Since this value is non-structural it does not matter if the geometry or content of an assembly are modified: this centroid remains unchanged and saved positions will still work. (Technically ill-conditioning could occur if the assembly's dimensions are made wildly different, but this is unlikely to happen in practice.)

However if nodes on the assembly used for **Connections** with its neighbours are moved it is possible that retrieving previously saved position will result in these **Connections** becoming separated. A moment's thought reveals why: the location of such a node in a previously saved position is unlikely to match up with the relevant connection position on the assembly on the other side of the connection.

Therefore the general rule is that saved positions will continue to function despite changes or additions/subtractions to assemblies so long as the locations of nodes used in connections are not modified.

Re-using Position data in different contexts

Dummies and Mechanisms can be slaved to masters as "children", whereupon position **Save** / **Retrieve** operations will operate on all assemblies in master and child(ren).

This is not a problem since positions are referenced purely by name, therefore:

- During a **Save** operation each assembly will acquire another entry of the specified name.

  Care should be taken not to use the same name in different contexts, leading to two or more positions for an assembly having the same name. While **Save** will work, a subsequent **Retrieve** may pick up the wrong position.

- During a **Retrieve** operation each assembly is scanned for a previously saved position of the given name. If no such position is found then that assembly's geometry is not altered.

  As explained above if an assembly contains two or more positions of the same name the first encountered will be used, which may cause unexpected outcomes.

Positions are an attribute of assemblies, so those saved while positioning a Dummy or Mechanism in isolation will be "visible" and available for use when it is used as a "child", and vice-versa.

Editing position data by hand

Essentially don't try it!

When a position is first saved an extra "reference" position, not visible to the user, which contains the assembly's current configuration is saved first, then user-defined positions follow. This reference position is updated whenever the assembly is repositioned, and all transformations are stored relative to this.

In addition any **Orient** type transformations of the whole Dummy or Mechanism will result in all saved positions being updated accordingly.

It would be virtually impossible to track all these transformations by hand, and it is strongly recommended that the only hand-editing carried out on these data should be limited to total deletion of a position should this be required.

Deleting position data manually.

It is possible to delete saved positions manually using an editor, and so long as you obey the following rules it can be done safely.

- Any explicitly saved *POSITION other than the first "reference" one can be deleted with no ill-effects.

  Simply delete the *POSITION card and all subsequent data up to but not including the next **keyword*.
  Saved positions have names rather than numbers so there is no need to worry about changing labels.

- The first "reference" *POSITION, marked "do not delete" can also be deleted ***if, and only if***, all other *POSITION cards are also deleted.

  As the discussion in The "reference" position above explains the reference position is required for other saved positions to be restored correctly.

It would be reasonable, and perfectly legitimate, to "clean up" an input deck by deleting all *POSITION data so that the positioner starts again from scratch next time it is used. Remember that Dummy and Mechanism assemblies have no intrinsic coordinate system, and it doesn't really matter what system is actually used so long as the usage is consistent thereafter.

APPENDIX IId: The Dummy Angles File (.daf)

From release 9.3RC1 onwards PRIMER can read and write "Dummy Angle" files, extension .daf.

The following formatting rules apply:

- These are simple ASCII files, intended to be editable by hand if required.
- All lines starting with **$**, **%** or **#** are treated as comment lines.
- All numerical data is in free format, however each section of data should be on a single line.
- All angles are in degrees, and should be in the range +/-180.0

The file format is: (note that this format evolved during release 9.3 development, see Format Changes below for details)

$ Dummy Angles File. Generated on *<current date>* **$ Model title: *<current model title>*** **<... any number of comment lines starting with $, % or # ...>** These comment lines are remembered, and will be copied to any newly written file for this model. All comment lines up to, but not including, the "**$ H-Point ...**" line are saved.

The H-Point location. Note that its "label" field is always zero.			
$ H-Point	**X position**	**Y position**	**Z position**
0	***<X coord>***	***<Y coord>***	***<Z coord>***
The "whole dummy" angles. Note that their label field is always -1.			
$ Whole Dummy	**X Angle**	**Y Angle**	**Z Angle**
-1	***<Theta X>***	***<Theta Y>***	***<Theta Z>***
Then each assembly's title, label and angles are listed. • These do not have to appear in numerical order, although this is recommended for clarity. • The title is a comment line for ease of reading: it is ignored when the file is read, and need not match the existing title of assembly <n> • If an assembly does not have an angle specified in this file its current angle will not be changed when the file is read.			
$ Assembly #1 title			
1	***<Theta X>***	***<Theta Y>***	***<Theta Z>***
$ Assembly #2 title			
2	***<Theta X>***	***<Theta Y>***	***<Theta Z>***
And so on for all assemblies in the Dummy			

Here is an example of an actual file:

```
$ Dummy Angles File. Generated on Wed Aug 27 13:46:49 2008
$ Model title: FT-ARUP HYBRID III 50TH - VERSION 5.1S2 (MM, TON, S)
$
$ The comments on this line, and all following up to the "H-Point"
$ line below will be saved and written out again.
$
$ H-Point X position Y position Z position
0 6.8459E+002 0.0000E+000 0.0000E+000
$
$ Whole Dummy X Angle Y Angle Z Angle
$
-1 0.0000E+000 1.0000E+001 1.8000E+002
$
```

```
$ Assembly X Angle Y Angle Z Angle
$
$ Lower Torso
1 0.0000E+000 0.0000E+000 0.0000E+000
$ Thorax
2 0.0000E+000 0.0000E+000 0.0000E+000
$ Head & Neck
3 0.0000E+000 4.9999E+000 0.0000E+000
$ Upper leg left
4 0.0000E+000 1.8107E+001 0.0000E+000
$ Upper leg right
5 0.0000E+000 0.0000E+000 0.0000E+000
$ Lower leg left
6 7.1490E-004 5.0000E+001 0.0000E+000
$ Lower leg right
7 5.2281E-004 2.7816E+001 0.0000E+000
$ Foot left
8 -1.6916E-003 1.7390E+001 0.0000E+000
$ Foot right
9 -1.6445E-003 0.0000E+000 -1.4572E-004
$ Yoke left
10 3.8573E-004 2.1671E-004 -5.0000E+001
$ Yoke right
11 -3.8573E-004 -1.5796E-004 -5.0000E+001
$ Upper arm left
12 1.2956E-004 -1.5649E-004 1.0004E-001
$ Upper arm right
13 -4.7659E-004 2.2656E-004 -1.5000E+001
$ Elbow left
14 3.8414E-004 -1.4245E-004 5.5000E+001
$ Elbow right
15 0.0000E+000 0.0000E+000 5.5000E+001
$ Lower arm left
16 6.8736E-004 0.0000E+000 9.2774E+001
$ Lower arm right
17 -1.0686E-004 -4.4135E-004 6.1678E+001
$ Wrist left
18 -1.4033E-004 -1.2171E-003 -9.0001E+001
$ Wrist right
19 2.6753E-004 0.0000E+000 -9.0000E+001
$ Hand left
20 -4.7274E-004 0.0000E+000 -1.9801E+001
$ Hand right
21 0.0000E+000 1.6804E-004 2.3132E+001
$ <End of file>
```

Format changes during V9.3 development

The format of this file evolved during the development of PRIMER 9.3 as described below.
Reading of earlier formats is automatic and no user intervention is required to read a 9.3RC1 or RC2 format .daf file into release 9.3, however if you propose to hand-edit older files you may need to consider the information below.

Direction cosines instead of Euler angles in 9.3RC1

The original format of this file, in 9.3RC1, used direction cosines instead of Euler angles to record assembly orientations; however the Euler angles were written as comment lines above these cosines. PRIMER 9.3 will read these files, but will write the new format described above using Euler angles.

If you have such an "old" file it is recommended that you read it into release 9.3 and write it out again immediately to convert it to the current format.

"Whole dummy" angles not present in 9.3RC2

PRIMER 9.3RC2 used Euler angles as described in the format above, but did not include the "whole dummy" orientation angles. These have been added in release 9.3 using the "label" -1.

If this line is omitted PRIMER assumes that no "whole dummy" orientation is required, making the change backwards-compatible.

APPENDIX III: Origami "tree" file example

The following is an Origami "tree" file example

```
*END
$
$
$ ============
$ ORIGAMI data
$ ============
$
$
*ORIGAMI_START
         1Origami 1
         1         2
$
*AXES
         1
$
*OPTIONS
        80         2         2         2         1
         2        -1         4         5         6         3         7
 1000000.0       1.1       0.9       0.0
$
$ =============
$ List of FOLDs
$ =============
$
$ LINE 1 (Basic data)
$     FIELD 1: LABEL
$     FIELD 2: TYPE
$              =0: NULL
$              =1: THIN
$              =2: THICK
$              =3: TUCK
$              =4: SPIRAL
$              =5: SCRUNCH
$              =6: ALIGN
$     FIELD 3: UPDOWN
$              =0: UP
$              =1: DOWN
$     FIELD 4: RIGHTLEFT
$              =0: RIGHT
$              =1: LEFT
$     FIELD 5: REFERENCE COORDINATE    SET FLAG
$     FIELD 6: LAYERS FOR TUCK FOLD
$     FIELD 7: SUBSET FOLDING
$     FIELD 8: CREATE ALIGN FOLD TRAM    LINES
$
$ LINE 2 ( Reference nodes)
$     FIELD 1: FOLD_NODE
$     FIELD 2: TUCK_ZSPLIT_N1
$     FIELD 3: TUCK_ZSPLIT_N2
$     FIELD 4: LAYER_ZMIN_N1
$     FIELD 5: LAYER_ZMIN_N2
$     FIELD 6: LAYER_ZMAX_N1
$     FIELD 7: LAYER_ZMAX_N2
$
$ LINE 3 (Sets, etc.)
$     FIELD 1: NODES_LEFT
$     FIELD 2: NODES_CENTER
$     FIELD 3: NODES_RIGHT
$     FIELD 4: SHELL_LEFT
$     FIELD 5: SHELL_CENTER
$     FIELD 6: SHELL_RIGHT
$     FIELD 7: TUCK FOLD TYPE
$
$ LINE 4 (Positions, etc.)
$     FIELD 1: THICKNESS
```

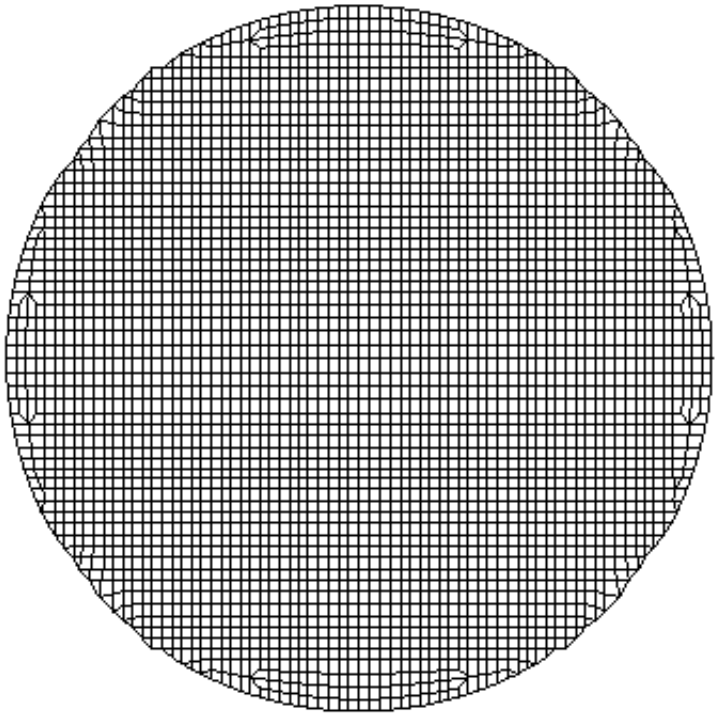
```
$     FIELD 2: FOLD_XPOS
$     FIELD 3: FOLD_XTOL
$     FIELD 4: INPLANE_ANGLE
$
$ LINE 5 (Fold specific data)
$     FIELD 1: Factor for unused portion    of spiral.
$     FIELD 2: Out-of-plane fold angle.
$     FIELD 3: Scale factor for fold    point separation.
$     FIELD 4: Location of ZSPLIT    for tuck.
$
$ LINE 6 (Layering and align data)
$     FIELD 1: Minimum value for layer.
$     FIELD 2: Maximum value for layer.
$     FIELD 3: Tramline offset distance    for align.
$
$ LINE 7 (Reference point.)
$     FIELD 1: X.
$     FIELD 2: Y.
$     FIELD 3: Z.
$
$ LINE 8 (Local X vector.)
$     FIELD 1: X.
$     FIELD 2: Y.
$     FIELD 3: Z.
$
$ LINE 9 (Vector in X-Y plane.)
$     FIELD 1: X.
$     FIELD 2: Y.
$     FIELD 3: Z.
$
$*FOLD
         1         3         0         0         1         0         0         0
       314         0         0         0         0         0         0
         2         2         2         1         1         1         0
               5.0           114.50000                 0.0                 0.0
        0.10000000               180.0                 1.0                 0.0
     -1.0000000E+20       1.0000000E+20                 0.0
               0.0                 0.0                 0.0
               1.0                 0.0                 0.0
               0.0                 1.0                 0.0
$
*FOLD
         2         3         0         1         1         0         0         0
      1729         0         0         0         0         0         0
         2         2         2         1         1         1         0
               5.0          -114.50000                 0.0                 0.0
        0.10000000               180.0                 1.0                 0.0
     -1.0000000E+20       1.0000000E+20                 0.0
               0.0                 0.0                 0.0
               1.0                 0.0                 0.0
               0.0                 1.0                 0.0
$
*FOLD
         3         1         0         1         1         0         0         0
      2343         0         0         0         0         0         0
         2         2         2         1         1         1         0
               5.0          -36.750000                 0.0                90.0
        0.10000000               180.0                 1.0                 0.0
     -1.0000000E+20       1.0000000E+20                 0.0
               0.0                 0.0                 0.0
               1.0                 0.0                 0.0
               0.0                 1.0                 0.0
$
$ ===============
$ List of ORIENTs
$ ===============
$
$ LINE 1
$     FIELD 1: LABEL
$     FIELD 2: TYPE
$              =0: TRANSLATION
```

```
$              =1: ROTATION
$              =2: SCALE
$     FIELD 3: TRANSLATE/ROTATE TYPE
$              =0: X
$              =1: Y
$              =2: Z
$              =3: vector
$              =4: N1->N2
$     FIELD 4: SCALE TYPE
$              =0: X,Y,Z
$              =1: N1,N2,N3
$     FIELD 5: N1
$     FIELD 6: N2
$     FIELD 7: N3
$     FIELD 8: CENTRE NODE
$
$ LINE 2
$     FIELD 1: TRANSLATE DISTANCE    TYPE
$              =0: MAGNITUDE OF VECTOR
$              =1: USER DEFINED
$     FIELD 2: USER DEFINED DISTANCE
$     FIELD 3: ROTATE/SCALE CENTRE    TYPE
$              =0: GLOBAL AXIS
$              =1: COORDINATE
$              =2: NODE
$              =3: N1
$     FIELD 4: CENTRE[X]
$     FIELD 5: CENTRE[Y]
$     FIELD 6: CENTRE[Z]
$     FIELD 7: ANGLE
$
$ LINE 3
$     FIELD 1: VECTOR[X]
$     FIELD 2: VECTOR[Y]
$     FIELD 3: VECTOR[Z]
$     FIELD 4: SCALE[X]
$     FIELD 5: SCALE[Y]
$     FIELD 6: SCALE[Z]
$
$
*ORIENT
         1         0         0         0         0         0         0         0
         0     100.0         0       0.0       0.0       0.0       0.0
       1.0       0.0       0.0       1.0       1.0       1.0
$
*ORIENT
         2         1         0         0         0         0         0      5726
         0       0.0         2       0.0       0.0       0.0      45.0
       1.0       0.0       0.0       1.0       1.0       1.0
$
*ORIGAMI_END
```

It is strongly recommended that you don’t attempt to edit Origami files by hand, as it can be very hard to identify exactly what the individual numbers mean. To change folds or orients read them back into PRIMER and edit them there.

Also, try not to separate Origami definitions from their parent input decks: they reference SET and other entities within these decks, and confusion will arise if these labels are not treated consistently.

APPENDIX IV: Airbag Folding example



The following example is of a somewhat simplified geometry of an airbag. This is designed to provide a fairly complete demonstration of the capabilities to make a folded airbag. It is not designed to represent a realistic fold pattern or to represent accurate deployment of an airbag. In fact, there is no *AIRBAG card in this model so it will not deploy.

Figure A4.1 shows the starting geometry for this model. The units are in millimetres and the fabric is 0.25mm thick. The airbag is a simple drivers side (or pancake) airbag.

For reference, the model is provided with the PRIMER manual in this appendix and is entitled "airbag_folding_example.key".

The first issue in folding this model is to create an ORIGAMI. This is done by selecting **DEFINE_ORIGAMI** and then **CREATE.** When choosing the materials, it is important to select only the materials which are to be folded. In this case the whole model is wanted for folding so **WHOLE MODEL** can be used to select the entire model. **SELECT** the ORIGAMI and press **SET_FOLD** to start folding the airbag.

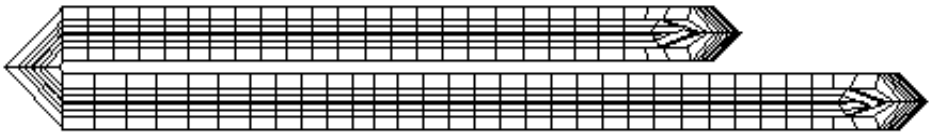
The folding pattern consists of 9 folds. The folds are:

1. Tuck fold along x-axis
2. Tuck fold along x-axis (interferes with first fold)
3. Thin fold along y-axis
4. Thin fold along y-axis using subset folding
5. Thin fold along y-axis using subset folding
6. 90 thick fold along y-axis
7. Align fold using tramlines
8. 90 thin fold along y-axis
9. Spiral fold using a local coordinate system and layers

The first fold is a tuck fold. The fold is in the x direction and in the xy plane (the default folding plane). The fold point is at 114.5 and the direction is from right to left. The fold point is defined by selecting a node using the FOLD_POINT button. By default the folder chooses all of the airbag to the right of the fold line. As this is what we want this is OK. Figure A4.2 shows a side view of the airbag after the tuck fold. In this example the fold separation has been set very high (5.0mm) so you can see the tuck fold. In reality the separation would be much smaller (probably the same order as the fabric thickness).

The second fold is defined in exactly the same way except that the direction is from left to right and the fold point is at -114.5. By default the folder chooses all of the airbag to the left of the fold line which is what is required.

A side view after FOLD 2 is shown in Figure A4.3. As the two tuck folds interfere with each other the first tuck fold has been moved so that no penetrations occur. As the fold separation is very large this effect has been exagerated. In reality the amount would be much smaller.



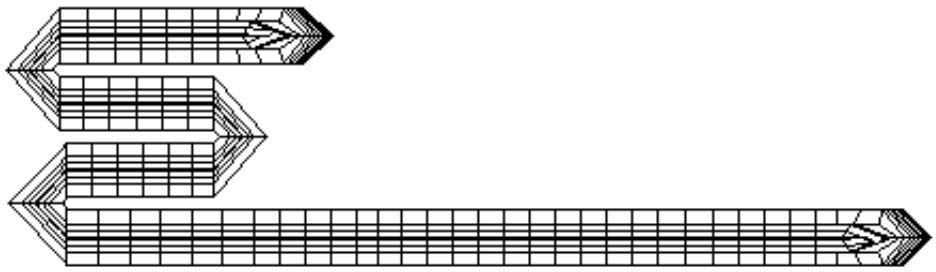
Folds 3 to 5 will show how you can use subset folding to quickly fold an airbag. Fold 3 is a thin fold. We want to fold it in the y direction soa 90 fold angle needs to be selected. Figure A4.4 shows the airbag after this fold has been done.

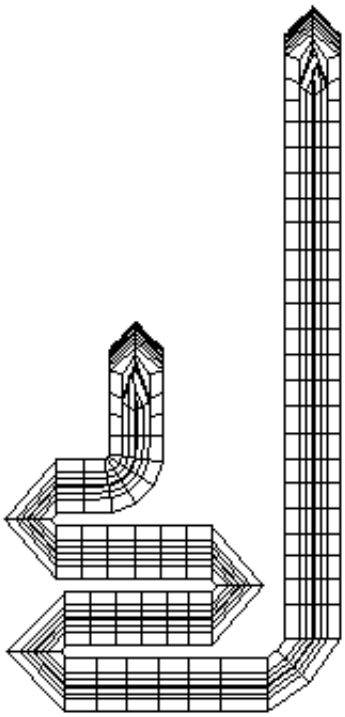
The fourth fold is a thin fold in the opposite direction. By default the folder will fold all of the airbag which is on one side of the fold line to the other side. In this case we only want to fold the top layer of the airbag in figure A4.4. We have three ways of performing this fold.

1. Defining a set to fold rather than the whole origami
2. Using layers to select a specific vertical range of the airbag to fold.
3. Using subset folding.

Subset folding is the easiest option to use. We can use this because all the nodes which we want to fold in this fold (4th fold) have been folded in the previous fold (fold 3). i.e these folds are a subset of the previous fold nodes. Press the **SUBSET FOLDING** button. The node selection should autoamtically update to the folds we need. The previous fold was in the left to right direction. This fold needs to be right to left.

The 5th fold is done in exactly the same way. As we are already using subset folding everything (including the fold direction) will be set correctly. Figure A4.5 shows the airbag after the first 5 folds.





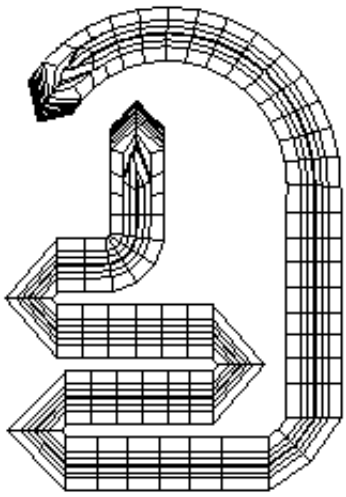
Fold 6 can also be done with subset folding. This is to show that subset folding is not just for thin folds. It also works for thick folds. We only want to fold this by 90 instead of 180. This is easily changed by using **THICK FOLD OPTIONS** and changing the angle. Apart from this complication the process is identical to folds 3 to 5.

Fold 7 is an **ALIGN** fold. This is used if your nodes are not exactly where you want them to be. In this case we want to make adjacent nodes to the fold line a constant distance of 5mm from the fold line. This is done by setting the fold point as normal. As we were using subset folding in the previous folds we need to make sure that it is turned off as these folds are no longer a subset of the previous fold. To set the align fold options use **ALIGN FOLD OPTIONS** and make sure that the **MAKE TRAMLINES** option is set with a constant distance of 5mm.

Fold 8 is a thin fold of 90. This presents no real problems and is similar to previous fold definitions. Figure A4.6 shows the airbag after the first 8 folds have been done.

The last fold is a spiral fold. This presents problems because the part of the airbag we want to fold does not lie on the xy plane which is the default folding plane for the folder. This can be overcome by defining a local coordinate system for this fold. To do this press **DONE** to exit the **SET_FOLD** menu and return to the main folding window. **LOCAL_SYSTEM** allow you to define a new folding plane. We define a plane which is in the global XZ axis. This is done by selecting N1 somewhere on the vertical part of the airbag where we want to do the spiral fold. N2 is selected so that local x points in the global Z direction. N3 is now chosen so that the 3 points give the local xy plane.

This allows us to define the plane for the spiral fold but if we try to fold this we will fold other parts of the airbag which are on the same side of the fold line. We need to either define a subset of the bag to fold or use layers to specify a range of the airbag to fold. In this case it is easier to define the **UPPER LAYER** so that anything above that local z coordinate is discarded when folding.



Once the airbag has been folded it can be positioned to the desired location using the **POSITION FOLDED BAG** option. This example model has 4 orientations stored with the origami so that the airbag can be repositioned as needed. They can be modified or deleted easily or new orientations can be added just like folds. The 4 orientations that are stored are:-

1. Translation, 100mm along the X axis
2. Rotation of 45 about X, centred on node 5726
3. Translation along vector using N1->N2 method with distance 50mm
4. A scale in a local coordinate system of 0.8 in the Y direction only.

As all of the parameters are already stored in the example, airbag_folding_example.key, it will not necessarily amount to much experience gained from looking at this file. The best way is to make a copy; edit the file; and remove all of the ***ORIGAMI, *FOLD** and ***ORIENT** definitions at the end of the file. Working from this, it should be possible to learn many of the important folding parameters.

APPENDIX V: Seatbelt "Tree" File Structure

A seatbelt definition is not a special file, rather it is an ordinary LS-DYNA input deck to which an extra section has been added after the ***END** card which allows PRIMER to recognise and manipulate seatbelt data.

Writing *BELT cards in the format of an earlier version of PRIMER

The format of the *BELT cards described below has evolved as PRIMER has developed, and while an older deck will always read into a newer version of PRIMER the opposite is not true: each major revision of PRIMER writes cards which will not read into earlier major versions. For example PRIMER V12 will not read a deck generated by PRIMER V13.

To get round this problem there is a preference which will restrict what is written to the format of an earlier version:

```
primer*mdumm_keyout_format:   V11   |   V12   |   V13   |   CURRENT
```

The default is **CURRENT**, meaning compatible with the current version of PRIMER, but setting an earlier version, eg V12, will restrict output to that version.

Note: restricting output to an earlier version will almost certainly result in some data being omitted with a consequent loss of information. The detailed listings below state where extra informaton has been added from V12 onwards, so the V11 format should be considered to be "baseline".

Details of *BELT cards

The description "tree" file is misleading, although it is adopted for historical reasons, since there is no hierarchy as such. Rather the file contains extra information about belt geometry, fitting, meshing and contact, and it is organised as follows.

```
*BELT_START
<label>       <Title>
<Solid set>   <Shell set>   <Tshell Set>   <Segm set>    <Part set>
<#rows>       <width>       <length>       <thickness>   <xoff_b>     <xoff_p>
<xoff_r>    <xoff_f>
<iter>        <conv_tol>    <tk_flag>      <tk_scale>    <pen_dist>   <overlap>
<proj_dist> <curve_ang>
<friction>    <acute_angle> <min_shell_tk>
<path order>  <drawn mode>  <offset mode>  <auto depen>  <depen iter> <radial
method>
<sep belt>    <sep factor>  <ramp #iter>   <slip radius> <slip angle> <slip
distance>
```

Label	I10	The label of this belt definition	
Title	A70	The title of this definition (optional)	
Solid set	I10	A set of solid elements defining "structure"	The "structure" can be any combination of these sets, and defines the dummy and vehicle elements contacted by the seatbelt. Not all sets need to exist, but at least one is required to give a structure definition.
Shell set	I10	A set of shell elements ditto	
Tshell set	I10	A set of thick shell elements ditto	

Segm set	I10	The set of segments that PRIMER creates from the three element sets above, used for contact between belt and structure during fitting. *From V12 onwards this reference to a segment set will not be written out here unless it has been referred to on a *BELT_CONTACT card. Likewise the set definition itself will not be written to the main deck unless it has been referred to by something other than *BELT_START. The reason for these changes is that the segment set is really volatile and gets recreated each time the belt is refitted, so writing it out serves no useful purpose.* *In addition when a new dummy replaces an old one the segment set may reference nodes that no longer exist, and thus become more trouble than it is worth.*	
Part set	I10	A set of parts	This may also be used to define structure, with or instead of the element sets above.
#rows	I10	The number of rows of 2D elements across the belt. This must lie in the range 0 (for all 1D belt elements) to 20. Default = 1	
width	E10.0	The overall belt width. Each 2D element wuill be <width> / <#rows> wide. Default = 50.0	
length	E10.0	The characteristic length of each belt element. Default = 25.0	
thickness	E10.0	The thickness of 2D belt elements. Default = 1.0	
xoff_b	E10.0	The offset of a database cross-section from a B-Post slipring	
xoff_p	E10.0	The offset of a database cross-section from a free (eg pelvis) slipring	
xoff_r	E10.0	The offset of a database cross-section from a retractor	
xoff_f	E10.0	The offset of a database cross-section from a fixed or end point.	
iter	I10	The number of fitting iterations between contact bucket resorts. Default = 25	
conv_tol	E10.0	The convergence tolerance at which fitting halts. Default = 1.0e-5	
tk_flag	I10	Thickness used during fitting: 0 (default) = use true thickness; 1 = use true thickness * factor; 2 = use neutral axis (no thickness)	
tk_scale	E10.0	Factor used when <tk_flag> = 1. Default = 1.0	
pen_dist	E10.0	Maximum penetration distance considered for contact into solid & thick shell elements. Default = 25.0, but also limited to shortest side length * 0.4.	
overlap	E10.0	Fraction by which facets are extended during contact checking to stop nodes "falling into gaps". Default = 0.05	
proj_dist	E10.0	Initial projection distance by which belt path is "thrown outwards" at start of fitting. Default = 35.0	
curve_ang	E10.0	Maximum permitted transverse belt curvature in degrees. Default = 0.0, meaning no limit, permitted values are 0 to 180 deg.	
The following line was introduced in V12			
friction	E10.0	Transverse friction coefficient in the range 0.0 - 1.0, default 0.0.	
acute_angle	E10.0	Angle (in degrees) considered to be "acute" in belt. Default 90.0, but can be in range 0.0 - 180.0 If this field is zero then the default of 90.0 degrees is assumed.	
min_shell_tk	E10.0	Minimum thickness used for contact with shells. If defined actual contact thickness is **`max(true thickness, min thickness)`** (*Field added in V14*)	

The following line was introduced in V13			
path order	I10	The curvature of the meshing path. 0 = linear, 1 = mixed linear/spline, 2 = cubic spline	
drawn mode	I10	How the fitting path is drawn. 0 = wireframe, 1 = wireframe with thickness, 2 = shaded with thickness	
offset mode	I10	How fitting path is offset from basic path points. 0 = full offset distance, 1 = offset distance * 0.1, 2 = no offset	
auto depen	I10	Whether or not to perform automatic depenetration of the belt path prior to fitting. 0 = no, 1 = yes.	
depen #iter	I10	Maximum number of iterations of automatic depenetration, default = 4.	
radial method	I10	How the fitting path twist is computed. 0 = from "Local" normals of nearby structure, 1 = from base path curvature.	
The following line was introduced in V13			
sep belt	I10	Whether or not to perform self-contact between adjacent segments of belt. 0 = no, 1 = yes	
sep factor	E10.0	Initial factor on true separation distance. Default = 1.0 (ie true thickness)	
ramp #iter	I10	Number of fitting iterations over which a separation factor > 1.0 ramps linearly down to 1.0. Default = 500	
slip radius	E10.0	These three attributes are all properties of "radiused", ie explicitly meshed, sliprings if these are used in preference to explicit slipring elements.	The radius of the slipring.
slip angle	E10.0		The angle (in degrees) subtended by elements around the slipring.
slip distance	E10.0		The distance each side of the slipring over which finely meshed elements extend.

```
*BELT_MESH
<node_set>    <nsbo_set>    <elem_set>   <cline_len> <np>        <ns>
<sbelt_f>     <sbelt_l>     <retr_f>     <retr_l>    <slip_f>    <slip_l>
<xsec_f>   <xsec_l>
<nrb_set_f>   <nrb_set_l>
<pid_1d>      <slen_1d>
<pid_sh>      <t1_sh>       <t2_sh>      <t3_sh>     <t4_sh>     <psi_sh>
<pid_2d>      <t1_2d>       <t2_2d>      <t3_2d>     <t4_2d>     <psi_2d>
(If <np> on line 1 is defined the line above is repeated <np> times, otherwise
it appears once if the belt
 definition contains 2d belt element data, otherwise not at all.)
<base p1>     <base p2>     <fit p1>     <fit p2>    <mode>      <#belt 1>
<#belt 2>
(The line above appears exactly <ns> times.  In decks prior to V13 neither <ns>
nor these lines are written.)
```

node_set	I10	Set of all nodes in seatbelt	These sets are only created if the option to generate a contact for the belt is used. They define the contact between the belt and the structure, the latter being defined via segment set <segm set> on the *BELT_START card.
nsbo_set	I10	Set of nodes on 1D seatbelt elements only	
elem_set	I10	Set of shell or 2D seatbelt elements	

cline_len	E10.0	Centreline length	***Optional***: This is not an input property, but rather the centreline length of the belt when last meshed in PRIMER. This data field is new in V12, and may be omitted or set to zero if no length is known, or if this card comes from an earlier keyword file.	
np	I10	Number of 2d belt element property cards	***New in V13***: the number of lines of 2d seatbelt property data that will be written below. This field is not written in earlier decks, and if it is omitted exactly one line of 2d seatbelt property data is output at the end of this keyword if the belt contains 2d seatbelt element property data, otherwise not at all.	
ns	I10	Number of "segments" of belt path and their meshing details.	***New in V13***: the number of "segments", ie stretches of belt path between fixed ends or acute points or sliprings. This variable defines how many lines of detailed segment meshing data will be written at the end of this keyword.	
sbelt_f	I10	First 1D seatbelt element id	Since there is no *SET_SEATBELT card in LS-DYNA the seatbelt fitter stores the first and last 1D belt element ids, assumes that elements are contiguous in this range and that it "owns" all these elements.	
sbelt_l	I10	Last 1D seatbelt element id		
retr_f	I10	First retractor id	Likewise it assumes that the list of retractors is contiguous between first and last, and that it owns all of these.	This is true for both 1D and 2D slipring and retractor elements.
retr_l	I10	Last retractor id		
slip_f	I10	First slipring id	Likewise it assumes that the list of sliprings is contiguous between first and last, and that it owns all of these.	
slip_l	I10	Last slipring id		
xsec_f	I10	First cross-section id	**`*DATABASE_CROSS_SECTION`** definitions are new in PRIMER release 11 and can be added to any path point. These are the first and last definitions which must have explicit labels using the **`_ID`** suffix.	Cross-sections can be added to all belt mesh types: 1d belt, 2d belt and shell.
xsec_l	I10	Last cross-section id		
nrb_set_f	I10	First nodal rigid body set id	There is no *SET_NODAL_RIGID_BODY card, so it is assumed that this list is contiguous, as above, and that the belt "owns" all these definitions.	Belt fitting in PRIMER pre-dates the optional Nodal RB labels in LS-DYNA, so set ids are used instead for backwards compatibility.
nrb_set_l	I10	Last nodal rigid body set id		
pid_1d	I10	The part ID for any 1D seatbelt elements		Only need to be defined if the belt contains 1D belt elements
slen_1d	E10.0	The initial slack length for any 1D seatbelt elements		
pid_sh	I10	The part id of any shell elements		Only need to be defined if the belt contains any conventional shell elements
t1_sh *to* t4_sh	4E10.0	***Optional*** thicknesses at the 4 nodes of these elements		
psi_sh	E10.0	***Optional*** orthotropic element angle for these elements		
*From V13 onwards the following line is repeated <**np**> times. In earlier decks where <**np**> is not output the following line appears exactly once if the belt contains 2d belt elements, otherwise not at all.*				

pid_2d	I10	The part id of any 2D seatbelt elements	Only need to be defined if the belt contains any 2D seatbelt shell elements
t1_2d *to* t4_2d	4E10.0	***Optional*** thicknesses at the 4 nodes of these elements	
psi_2d	E10.0	***Optional*** orthotropic element angle for these elements	
*The following line is only written from V13 onwards. It is repeated <**ns**> times, once for each belt meshing segment.*			
base pt 1	I10	The point number in the basic belt path at end #1 of this segment	This information is repeated for each of <***ns***> meshing segments and is used when the belt is remeshed either manually or automatically. Prior to V13, when this information was not written, the belt fitter "crawled up" any existing belt to extract this information - not always with 100% reliability!
base pt 2	I10	The point number in the basic belt path at end #2 of this segment	
fit pt 1	I10	The point in the belt fitting path at end #1 of this segment	
fit pt 2	I10	The point in the belt fitting path at end #2 of this segment	
mode	I10	**In "old style"** the meshing mode for this segment. 0 = 1d belt elements only, 1 = 2d belt elements only, 2 = shell elements only, 3 = mixed shells and 1d belt elements, 4 = mixed shells and 2d belt elements. I old style only the bottom byte, ie mask 0xf, will ever be populated. **In "new style"**, V14 onwards, meshing also the mode, but now a bitwise OR of the following: 0x0010 always present to designate "new style", the bottom byte will always be zero. **Span** / **If 1d belt** / **If 2d belt** / **If shell** **E1** / 0x00100 / 0x00200 / 0x00400 **CE** / 0x01000 / 0x02000 / 0x04000 **E2** / 0x10000 / 0x20000 / 0x40000 For a given row only a single column's entry may be used. For example the code 0x12410 means end 1 is shells, centre is 2d belts, end 2 is 1s belts.	
#belt 1	I10	In an "old style" mixed mode segment (ie mode = 3 or 4) the number of 1d or 2d belt elements at end #1. In "new style"meshing the number of elements of any type in span E1, or zero if the span is of a single element type.	
#belt 2	I10	In an "old style" mixed mode segment (ie mode = 3 or 4) the number of 1d or 2d belt elements at end #2. In "new style"meshing the number of elements of any type in span E2, or zero if the span is of a single element type.	

```
*BELT_CONTACT
<no_su_cont>   <su_su_cont>
```

no_su_contact	I10	Label of Nodes to Surface contact used when contact between nodes on 1D belt elements and dummy structure is defined.	This card is only used if the option to create a contact between belt and dummy "structure" has been used.
su_su_contact	I10	Label of Surface to Surface contact used when contact between belt shells / 2D belt elements and dummy structure is defined.	

```
*BELT_PATH
<npts>
```

```
<bits>       <x1>           <y1>           <z1>     (<nid>)
(Optional row 1 of further data depending on <bits>)
(Optional row 2 of further data depending on <bits>)
 : : : :
<bits>       <xn>           <yn>           <zn>     (<nid>)
(Optional row 1 of further data depending on <bits>)
(Optional row 2 of further data depending on <bits>)
(Optional row 3 of further data depending on <bits>)
```

npts	I10	Number of points in the basic "line" belt path	
There then follow <npts> cards of the following data:			
bits	I10	Bitwise encoded integer containing information about the attributes of this point	This encoding is internal to PRIMER
X, Y, Z	3E10.0	X,Y,Z coordinates of point <n>	Either or both of [X,Y,Z] and <nid> may be defined. However if both are defined and the coordinates differ then the rules used to locate the path point depend on the version of PRIMER: • **V13 and earlier:** if the node and path point coordinate are within 5mm the path point is moved to the node. If outside 5mm the node is ignored and removed from the path, and the coordinate is used verbatim. • **V14 onwards**: the rules are more flexible and depend on the "belt_path_match_method" and "belt_path_match_tol" preferences. For backwards compatibility the default behaviour is the same as V13, but see "Action on node/point coordinate mismatch" in section 6 (tools) on seatbelt fitting for an explanation of the new rules.
nid	I10	***Optional***: Node at point <n>	
Optional row #1	I10	Currently unused, and will currently be zero	These two rows of data are only written if the twist at this belt point has been controlled during belt fitting. Row #1 defines the twist on side A, and row #2 on side B. Twist vectors are written in bit-packed form and are not user-definable. From V11 onwards if both "twist nodes" 1 and 2 have been defined the vector between them will supersede any encoded orientation vectors. At fixed end and intermediate points this N1N2 vector will define the belt orientation precisely, at other points it will define the plane in which the belt's transverse vector lies. This provides an alternative way for users to define belt twist.
	3E20.0	Encoded orientation vectors	
	I10	***Optional***: orientation "twist" node N1 (V11 onwards)	
Optional row #2	I10	Currently unused, and will currently be zero	
	3E20.0	Encoded orientation vectors	
	I10	***Optional***: orientation "twist" node N2 (V11 onwards)	
Optional row #3	E10.0	***Optional***: Point-specific projection distance (V14 onwards)	This row of data is written from V14 onwards. It is only required if a non-default projection distance has been specified for this path point.

The format of the ***BELT_PATH** row entries is complex, involving internal bit fields designating retractor, slipring, etc. It may also contains normal information path twist that is similarly bit-packed into data words. These formats are internal to PRIMER and may change in subsequent releases.

It is ***strongly*** recommended that you do not attempt to edit these values.

***BELT_PARAMETER**

<length parameter> <per segm params>

This card is new in V12 and is entirely optional. If omitted no parameter value will be updated.

length_parameter	A10	The name of a ***PARAMETER** (with no leading "&") that will be updated with the total belt length whenever the belt is remeshed.
per segm params	I10	***Added in V13:*** if this value is 1 then additional parameters will be created for each meshing segment of the belt giving the length of that segment only. The syntax used is the name of the **length_parameter** defined above with "**_sn**" added where ***n*** is the segment number. For example if the length parameter is **blen**, and the belt has three meshing segments, parameters **blen_s1**, **blen_s2** and **blen_s3** will be added.

***BELT_END**

No data fields

Users should not attempt to edit this file by hand unless they are very certain about what they are they doing.

Deleting this file from an input deck will not affect its behaviour during analysis, but will cause PRIMER to lose the ability to remesh the belt.

APPENDIX VI: Format translation during **MODEL> READ**

Compressed LS-DYNA formats (.gz and .zip)

These formats do not require "translation", just decompression. The following are supported.

File format	Nomenclature		Organisation
.gz	name.key => name.key.gz		One-to-one compression of ascii .key files to .key.gz using "gzip".
.zip (single files)	name.key => name.zip		One-to-one compression of ascii .key files .zip using "Winzip" or similar
.zip (package)	master.key include1.key include2.key etc	} } name.zip } }	Master and any number of include files are packed into a single .zip archive. Master file is embedded as name.key Include files are embedded as **INCL**/include1.key, INCL/include2.key, etc

PRIMER can both read and write any of these compressed formats directly, ie you can open "file.key.gz" or "file.zip" directly.

Compressed input is automatic, detected from the file name and contents. Compressed output is under user control, see **Model Write, Pre-out, Compress tab**

Encrypted .zip files.

The ".zip" format optionally allows files to be encrypted. When this is done the master archive index is "in clear", but files within it can be encrypted in a variety of ways.

Reading encrypted .zip files

From V16 onwards PRIMER can read encrypted .zip files. It will detect the encrypted status and prompt you for a password.

Writing encrypted .zip files

PRIMER does not write encrypted .zip files.

Binary LS-DYNA format.

PRIMER 15 introduces a new binary format which has the following attributes:

- It is a "mirror image" of the normal ascii text LS-DYNA input file, retaining both content and layout, but is encoded in binary.
- This gives a file size that is typically 30% of the equivalent ascii file.
- Because no ascii <==> binary conversion is required during i/o both reading and writing are faster.

The format is proprietary to PRIMER and cannot be read by LS-DYNA or any other commercial software, so it will be necessary to convert it back to conventional ascii format before use in external software. This can be done in two ways:

1. Read into PRIMER and write out in ascii format

2. Use the "decode_binary" command-line utility provided as part of the Oasys LS-DYNA version 15 suite. This utility is also available on the Oasys Ltd website for free download, no licence is required.

Reading of binary files is automatic since it is detected from the *START_BINARY keyword in the file itself. Writing of binary files is controlled under the "compess" tab in Model Write, see **Model Write, Pre-out, Compress tab**

Description of binary format files

Each line of a normal keyword file is made up of numbers and text, and these are stored in a way that is readable by humans using ascii text. For example consider the format of a *NODE card in LS-DYNA which is:

Field name	**NID**	**X**	**Y**	**Z**	**TC**	**RC**
Format width	I8	F16	F16	F16	I8	I8
Internal #bytes	4	4	4	4	1	1

Therefore this card requires (8 + 16 + 16 + 16 + 8 + 8) = 72 ascii characters for output, yet only (4 + 4 + 4 + 4 + 1 + 1) = 18 bytes for internal storage, which is 25% of the space.

Moreover when this file is written out in ascii format considerable cpu time has to be spent converting the internal binary format to the external ascii format, in fact for many models this conversion process is the most "expensive" part of the output process.

So writing this card in its internal binary format is not only faster, but also results in substantially smaller files, and this is what PRIMER's binary format does. The file also encodes the format of each card, so a binary file can be converted back to a normal ascii file without any loss of information.

Format of the binary file.

A binary file starts off in ascii just like a normal keyword output file, then swaps to binary when the keyword

`*START_BINARY`

is encountered. Thereafter the file is binary and not human readable. The details of the binary format are proprietary.

This is not a standard LS-DYNA keyword, and LS-DYNA will be unable to read the file unless it is converted back to ascii.

File size reduction

Because information about the format as well as the contents of the line have to be stored, the actual file size tends to be about 30% of the original (small format) ascii file.

Speed of output of binary files

Speed-up of file writing tends to be in the range 6.5x on a fast local disk, to better than 18x on a remote disk on a slow network.

Reading of binary files by PRIMER

PRIMER 15 will read binary files directly. No special action is required on the part of the user as the file is opened in the normal way and PRIMER then detects the **`*START_BINARY`** keyword and interprets the binary format directly.

Speed of reading binary files

From a fast local disk input tends to be about 1.5x faster, rising to 3.0x times faster on a slow remote disk. (Binary input means that the speed bottleneck moves from file read to internal database construction.)

Compression of binary format files

Compression and binary format may be used together, ie a binary file can be further compressed to .gz or .zip format.

Experience shows that at the standard level of compression a typical model which compresses to 20 - 25% of its original size in ascii format will compress to 15 - 20% in binary format, so there is some gain to be had.

Compression at the default level of 1 increases the time taken to write and read binary format files by a small amount, usually less than 5%. On a very slow network there may be an improvement in speed if the benefit of transmitting the smaller file size outweighs the cost of compression.

NASTRAN Bulk data file format

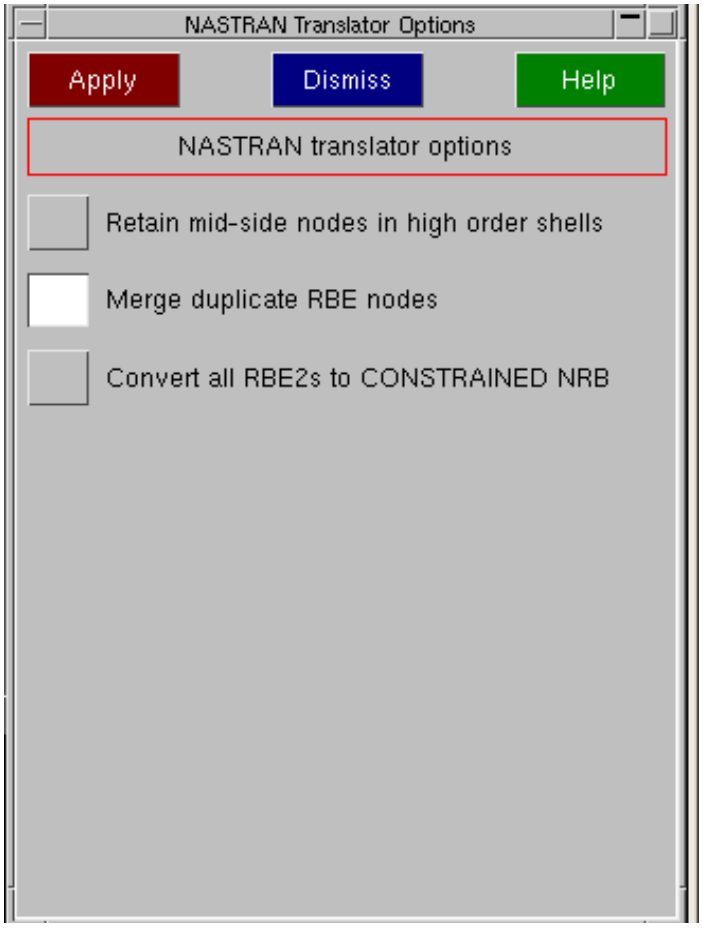
The following table shows the Nastran keywords supported, and how they are translated to internal LS-DYNA data when read in.

Nastran keyword	Internal LS-DYNA keyword
CBAR (+BAROR) **CBEAM (+BEAMOR)** **CBUSH** **CROD** **CTUBE**	***ELEMENT_BEAM** (+3rd nodes as required)
CDAMP1 **CDAMP2** **CELAS1** **CELAS2** **CVISC**	***ELEMENT_DISCRETE** (with appropriate attributes as deduced from PELAS, PDAMP and PVISC cards).
CHEXA **CPENTA** **CTETRA**	***ELEMENT_SOLID**
CONM2	***ELEMENT_MASS** *or* ***ELEMENT_INERTIA**
CORD1R	***DEFINE_COORDINATE_NODES**
CORD2R	***DEFINE_COORDINATE_SYSTEM**
CQUAD4 **CQUAD8** **CQUADR** **CTRIA3** **CTRIA6** **CTRIAR**	***ELEMENT_SHELL**
FORCE	***LOAD_NODE**
GRID	***NODE** (includes restraints if specified)
INCLUDE	***INCLUDE**
MAT1	***MAT_ELASTIC**
MAT8	***MAT_ENHANCED_COMPOSITE_DAMAGE**
PBAR **PBEAM** **PBUSH** **PROD** **PTUBE**	***PART** ***SECTION_BEAM**
PBARL (Currnetly "BAR", "BOX", "BOX1", "I", "I1", "ROD" and "TUBE" TYPEs are supported.)	***SECTION_BEAM** ***PART**
PCOMP	***PART_COMPOSITE**
PDAMP **PVISC**	***PART** ***SECTION_DISCRETE** ***MAT_DAMPER_VISCOUS**

PELAS	***PART** ***SECTION_DISCRETE** ***MAT_SPRING_ELASTIC**
PLOAD	***LOAD_SHELL**
PLOTEL	***ELEMENT_PLOTEL**
PSHELL	***PART** ***SECTION_SHELL**
PSOLID	***PART** ***SECTION_SOLID**
RBAR **RBE1** **RBE2**	***CONSTRAINED_NODAL_RIGID_BODY** *or* ***CONSTRAINED_SPOTWELD** (RBE2 only.) (+sets of nodes [***SET_NODE**] as required)
RBE3	***CONSTRAINED_INTERPOLATION**
SPC **SPC1**	***BOUNDARY_SPC**
TEMP	***LOAD_THERMAL_CONSTANT_NODE**

Notes:

1. NASTRAN **RBE2** cards with only two fully constrained nodes are translated to LS-Dyna ***CONSTRAINED_SPOTWELD** by default. If these two nodes are not fully constrained, then it is convereted to LS-Dyna ***CONSTRAINED_NODAL_RIGID_BODY**. Now there is a new option **"Convert all RBE2s to CONSTRAINED NRB"** available while reading the NASTRAN input file, which allows all two noded NASTRAN **RBE2** cards to be converted to LS_Dyna ***CONSTRAINED_NODAL_RIGID_BODY** cards. For this option there is a preference **primer*convert_rbe2_cnrb:** also, which is set FALSE by default.



2. Continuation characters are fully supported by the NASTRAN input translator. Even those cards that span multiple lines but do not contain explicit continuation characters are now translated properly.
3. Include files are fully supported by the input translator, and the include file structure is preserved in memory after the translation process is complete. Hence, if the model is written out in a format that supports include files, the resulting model will be written out across include its corresponding include files.
4. Both SMALL and WIDE format cards are supported by the input translator.
5. The "extra" data on certain cards that are defined in the LS-DYNA keyword manual under **`*TRANSLATE_NASTRAN`** (cards **`CELAS1, PSHELL, PSOLID`**) is not supported by the PRIMER translator. These values will be ignored if found.

I-DEAS Universal File Reader

The I-DEAS Universal file reader reads and processes the following I-DEAS Universal file modules.

Module Number	Contents	Data Processed
151	Title	
755	Restraints	Translational and rotation restraints in the global axis system.
773, 1710 & 1714	Materials	ID and Name
776	Beam Cross Sections	ID and Name (see below for more details)
780 & 2412	Elements	(see below for more details)
781 & 2411	Nodes	ID, Global Coordinates
789, 2437, 2448	Physical Sections	ID and Name (see below for more details)

Element Types

I-DEAS TYPE	Colour	LS-DYNA Type	Additional Data / Comments
21	Any	2 Noded Beam	Beam Orientation Node
91	Any	3 Noded Shell	
94	Any	4 Noded Shell	
101	Any	6 Noded Thick Shell	
104	Any	8 Noded Thick Shell	
111	Any	4 Noded Solid	
112	Any	6 Noded Solid	
115	Any	8 Noded Solid	
122	Cyan	*CONSTRAINED_RIVET	Only first 2 nodes processed
122	Red	*CONSTRAINED_SPOTWELD	Only first 2 nodes processed
122	Yellow	*CONSTRAINED_NODAL_ RIGID_BODY	I-DEAS element ID is used as the LS-DYNA node set ID.
122	Green	*CONSTRAINED_NODE_SET	I-DEAS element ID is used as the LS-DYNA node set ID
122	Blue	*CONSTRAINED_GENERALIZED_ WELD_SPOT	I-DEAS element ID is used as the LS-DYNA node set ID

136	Any	Translational Spring	
137	Any	Rotational Spring	
138	Any	Grounded Translational Spring	
139	Any	Grounded Rotational Spring	
141	Any	Translational Damper	
161	Any	Lumped Mass	

Physical Sections

IDEAS TYPE	**LS-DYNA Type**	**LS-DYNA Data Processed**
90 (Thin Shell)	*SECTION_SHELL	ID, Name, Shell thickness
100 (Thick Shell)	*SECTION_TSHELL	ID, Name
110 (Solid)	*SECTION_SOLID	ID, Name
133 (Translational Spring)	*SECTION_DISCRETE	ID, Name
134 (Rotational Spring)	*SECTION_DISCRETE	ID, Name
161 (Lumped Mass)	*ELEMENT_MASS	Mass

Beam Cross Sections

I-DEAS TYPE	LS-DYNA Type	LS-DYNA Data Processed
0 (Keyed In)	Belytschko-Schwer	Area, Iss, Itt, Irr , Shear Area = Area
1 (Rectangular)	Hughes-Lui (Rectangular)	TS1, TS2, TT1, TT2
3 (Circular)	Hughes-Lui (Circular)	TS1, TS2, TT1, TT2
4 (Pipe)	Hughes-Lui (Circular)	TS1, TS2, TT1, TT2
999 (General)	Belytschko-Schwer	Area, Iss, Itt, Irr, Shear Area = Area

In I-DEAS an elements properties are defined by both a set of Material data and a set of Physical. In LS-DYNA an elements properties are defined by a single PART number which then references a Material and a Section. When importing a Universal file PRIMER will automatically generate PARTS for all the combinations of Materials and Physicals used in the universal file. The ID’s assigned to each PART can be made equal to either the elements Material ID or the elements Physical (Section) ID.

If PART numbers are based on Material ID and a Material is used with more than one Physical (Section) then the numbering scheme described above can not be followed. If this is the case then the PART using the Material which has the lowest Physical (Section) ID will be assigned the and ID equal to the Material ID and any other PARTs using the same material will be given ID’s greater then the highest PART ID. Similarly if PART numbers are based on Physical (Section) ID and a Section is used with more than one Material then unique PART numbers will be generated for each PART.

The following table shows the difference in basing the PART number on the material or section property ID. Note that as there is only a clash with the material numbers then the option of basing the PART number on the section ID still results in the PARTS being given ID’s the same as the section properties.

Element ID	**Material ID**	**Section ID**	**Part Number based on**	
			Material ID	**Section ID**

1	1	101	1	101
2	1	102	4	102
3	1	103	5	103
4	1	104	6	104
5	2	201	2	201
6	2	202	7	202
7	3	301	3	301

SPRING ELEMENTS

In I-DEAS spring elements are only assigned a Physical (Section) ID. PRIMER assumes that the Material ID of a spring element is the same as the Physical (Section) ID. Spring elements should NOT therefore be defined using Physical (Section) ID’s that are the same as Material ID’s used by other element types.

BEAM ELEMENTS

In I-DEAS the section properties for beam elements are defined using a Beam Cross Section. This means it is possible for beams to have the same Material and Physical ID’s but to have different Beam Sections. When translating beam elements for I-DEAS PRIMER ignores the Physical ID and uses the Beam Cross Section ID as the LS-DYNA section ID. Beam Cross Sections should NOT therefore be defined using the same ID’s as Physical Sections used by element types that are not beams.

ABAQUS Input file format

The following table shows the Abaqus keywords supported, and how they are translated to internal LS-DYNA data when read in.

Abaqus keyword	**Internal LS-DYNA keyword**
`*BEAM GENERAL SECTION` (Currently only " SECTION=GENERAL" supported.)	**`*SECTION_BEAM`** (ELFORM = 12)
`*BEAM SECTION` (Currently only "SECTION=CIRC" supported.)	**`*SECTION_BEAM`** (ELFORM = 1)
`*CONTACT PAIR` (Currenlty only supproted for optional parameter SMALL SLIDING) **`*SURFACE`** **`*SURFACE INTERACTION`**	**`*CONTACT_NODES_TO_SURFACE`**
`*DISTRIBUTING COUPLING`	**`*CONSTRAINED_INTERPOLATION`**
`*ELEMENT`	**`*ELEMENT`** (Currently only beam elements (TYPE=B31), shell elements and 8 nodes solid elements supported)
`*ELSET`	**`*SET_SHELL`** *or* **`*SET_SOLID`**
`*HEADING`	**`*TITLE`**
`*KINEMATIC COUPLING`	**`*CONSTRAINED_INTERPOLATION`**

`*MATERIAL` **`*DENSITY`** **`*ELASTIC`**	**`*MAT_ELASTIC`**
`*MPC` (Currently only BEAM type supported.)	**`*CONSTRAINED_NODAL_RIGID_BODY`**
`*NODE`	**`*NODE`**
`*NSET`	**`*SET_NODE`**
`*PLASTIC`	**`*MAT_PIECEWISE_LINEAR_PLASTICITY`**
`*SHELL SECTION`	**`*SECTION_SHELL`**
`*SOLID SECTION`	**`*SECTION_SOLID`**

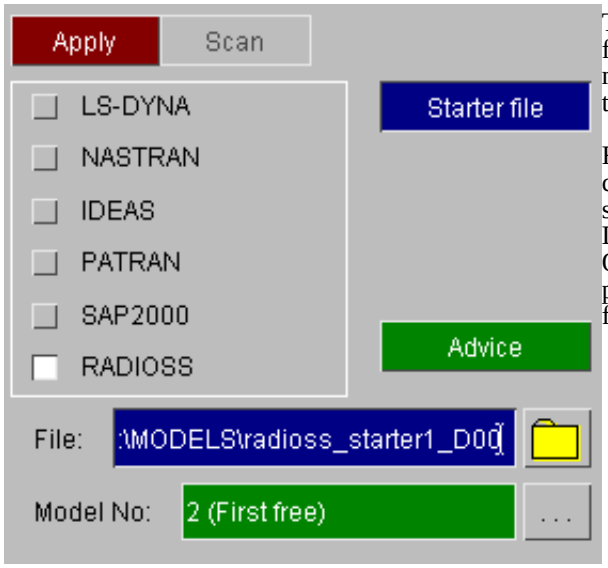
Patran Neutral File (.ntl) Reader

The following modules from Patran "neutral" files up to Patran level 2.5 format are read. All others are ignored.

Patran Module id	**Data type**	**Stored as**
25	Analysis title	Analysis title
1	Nodes	Nodes
2	Elements	Shape <iv> = 2 (bar), #nodes = 2 • <config> 0, 1, 2 become a beam element • <config> 5 becomes extra nodes on rigid parts • <config> 7 becomes a lumped mass • <config> 8 becomes a spherical joint • <config> 30 becomes a seatbelt element • <config> 31 becomes a retractor • <config> 32 becomes a slipring The following configurations becomes springs/dampers, grounded if <n1> = <n2>: • <config> 6, 10 becomes a translational spring • <config> 11 becomes a rotational spring • <config> 20 becomes a translational damper • <config> 21 becomes a rotational damper Shape <iv> = 2 (bar), #nodes > 2 • If #nodes = 4 become revolute joints • If #nodes = 6 become translational joints. Shape <iv> = 3 (tria): • <config> 0, 3 become 3 noded shells (n3 = n4) Shape <iv> = 4 (quad): • <config> 0, 3 become 4 noded shells Shape <iv> = 5 (tetra): • <config> = 0, 4 become 4 noded solid tetrahedra (n4=n5=n6=n7=n8) Shape <iv> = 7 (wedge): • <config> = 0, 4 become 6 noded solid wedges (n5=n6, n7=n8) • <config> = 1 become 6 noded thick shells (n3=n4, n7=n8) Shape <iv> = 8 (hex): • <config> = 0, 4 become 8 noded solid hexahedra • <config> = 1 become 8 noded thick shells All other element types are ignored.

RADIOSS fixed file format

The RADIOSS Translator function is aimed specifically at translating RADIOSS starter and engine files into LS-DYNA keyword format data.



The RADIOSS translator is invoked identically to all other formats read into PRIMER. Please refer to section 3.2 of the main manual for details on the **READ** function, remembering to select the RADIOSS sub-type.

For a translation the starter file must be present. An engine file can optionally be selected. The generic file extension for a starter file is 00 (Radioss starter files usually end with d00 or D00), but this can easily be modified in the file selector panel. Once the correct file has been selected and the **APPLY** button pushed, another window will be created which allows an engine file to be selected and some options to be set.

The buttons in the RADIOSS translation defaults box are as follows:-

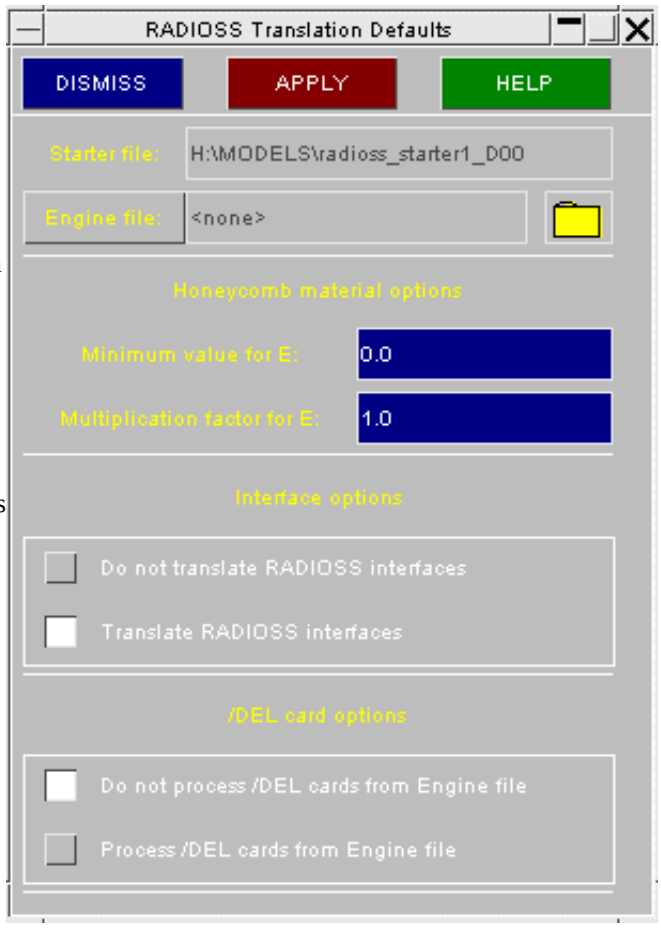
DISMISS terminate the translation process, returning to the generic READ panel.

APPLY accept the defaults in the panel and proceed with the translation.

HELP will create a message box full of useful information about the function of this panel.

ENGINE FILE is by default greyed out and the text box next to the button reads <none>. If you have a RADIOSS engine file to go with the starter file you are translating pressing this button will then allow an engine file to be selected by either typing in the name in the text box or pressing the button which will bring up the file selection box. Engine files by default have the extension 01 (Radioss engine files usually end with d01 or D01)

MINIMUM VALUE FOR E and **MULTIPLICATION FACTOR FOR E** are used to set options for translating honeycomb materials. When honeycomb materials are defined in RADIOSS the material curve can be steeper than the Young's modulus. This is not allowed in LS-DYNA. These options allow a minimum permissible Young's modulus to be set.



INTERFACE OPTIONS

The radio buttons under Interface Options can be used to either translate or skip any interfaces (contacts) that are in the starter file.

/DEL CARD OPTIONS

The radio buttons under /DEL card options can be used to either process or skip any /DEL cards that are present in the engine file.

TRANSLATOR FUNCTIONALITY

This section of the appendix is meant to give a brief insight into the methods that the translator uses to process and store data, and the types of RADIOSS data which are understood.

Engine File

The following keywords are translated:

/DTIX Initial timestep translated as **DTINIT** on ***CONTROL_TIMESTEP;** maximum timestep ignored

/DT Scale factor on timestep translated as **TSSFAC** on ***CONTROL_TIMESTEP**. Minimum timestep converted to a mass-scaling timestep (-ve value of **DT2MS** on ***CONTROL_TIMESTEP**). Any other keywords (e.g. specifying a particular element type) are ignored.

/GFILE T_{freq} translated as interval between plot file outputs (**DT** on ***DATABASE_BINARY_D3PLOT**). T_{start} and type of file ignored.

/RFILE N_{cycle} translated as number of cycles between restart dumps (**CYCL** on ***DATABASE_BINARY_D3DUMP**). File types ignored.

/RUN T_{stop} translated as termination time (**ENDTIM** on ***CONTROL_TERMINATION**). Run name and number ignored.

/TFILE T_{his} translated as interval between time history file outputs (**DT** on ***DATABASE_BINARY_D3THDT**). Type of file ignored

/VERS Tells Primer which version of RADIOSS the starter file is

The following keywords are translated, but for their effect upon the model see the relevant sections:

/DEL See below

/VEL See below

The following keywords are ignored:

/ANIM

/DYREL

/INTER (but birth & death times will be read from Starter file)

/KEREL

/PATRAN

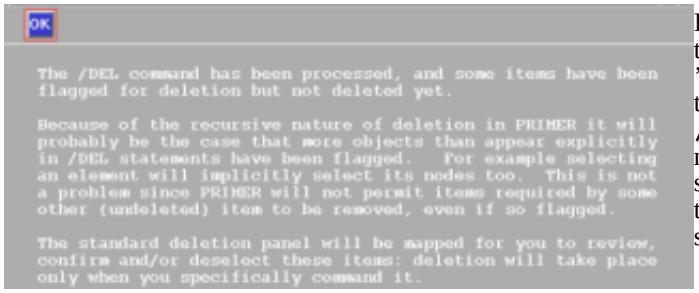
/PRINT

/TITLE (but title will be read from starter file)

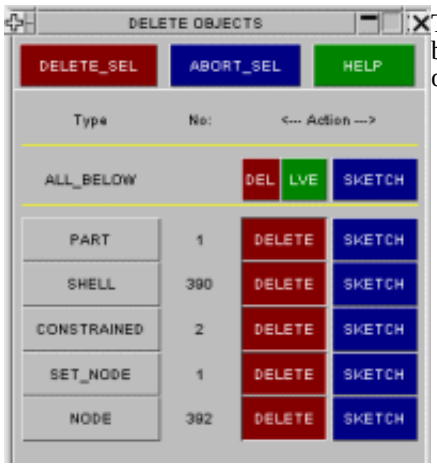
/BCS & BCSR cannot read restraints from the engine file, only from the Starter file.

/RBODY

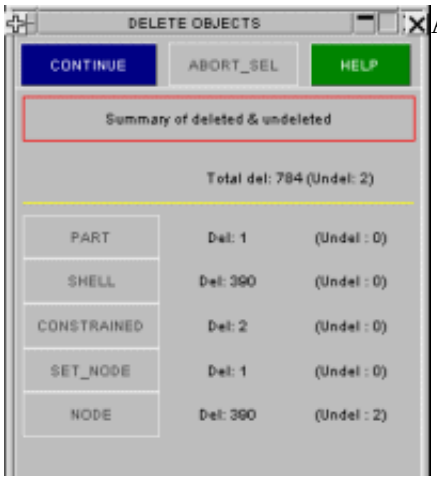
/DEL CARDS



If there are any **/DEL cards** in the engine file and they have been selected to process by using the 'Process **/DEL cards**' radio button Primer will allow the user to delete or leave the elements selected in the **/DEL cards**. If there are any elements that are marked for deletion, two windows will appear on the screen after the translation has finished. The first is a text window explaining what is happening and the second shows what objects will be deleted.



The selected elements can now either be deleted by pressing the **DELETE SEL** button or left by pressing the **ABORT SEL** button. Alternatively if only some of the elements need to be deleted the appropriate elements can be selected.



After deleting the window shows if the deletion has been successful.

/VEL CARDS

In RADIOSS, a set of nodes can be given the same translational velocity by using a **/VEL/TRA** card. Similarly they can be given the same rotational velocity by using a **/VEL/ROT** card. Each card can be used in any combination of X, Y and Z. Many of these combinations have no equivalent in LS-DYNA so some translations are not exact.

If a set of nodes occurs on a **/VEL/TRA** card only it is translated as a ***CONSTRAINED_NODE_SET**. This is an exact translation.

If a set of nodes on a **/VEL/TRA** card have a translational degree of freedom XYZ and they also appear in a **/VEL/ROT** card with degree of freedom XYZ, the set is translated to a ***CONSTRAINED_NODAL_RIGID_BODY**. This is an exact translation.

If a set of nodes appears on a **/VEL/TRA/** and a **/VEL/ROT** card with any other degree of freedom there is no equivalent in LS-DYNA. The rotational degrees of freedom are ignored and a ***CONSTRAINED_NODE_SET** is created with the translational degree of freedom.

Starter File

The table below summarises which features can be translated. For more details on each feature see the relevant section.

Feature	Translated	Ignored
Header		
Title & Control Data	title, gravity load	all other data
Materials	Types 1,2,3,4,14,19,21,22,23,27 & 28 (see below)	All other materials are translated as *MAT_NULL (see below)
Mats for time history		
Nodes	fully supported	
Nodes for time history	fully supported	
Skew frames for node time history		no equivalent

Boundary Conditions	all except GRILAG	GRILAG
Skew Frames		
Elements		
	all except 2-D solid and 3 noded springs	
Properties		
	all except type 12 (see below)	
Functions		
Concentrated loads		Sensor not translated
Pressure loads		Sensor not translated
Initial velocity	all except I_{vel}	I_{vel}
Accelerometers		
Interfaces	all types & most data - see below	
Rigid Walls	Plane, parallelogram - all data except as in "ignored" box	Node (moving wall); dist for search; direction of init. vel; prlgrm converted to rectangle
Rigid Body	see below	
Rigid Body for time history		
Added Mass		
Fixed Velocity		
Rivets spotwelds		
Sections		
Cylindrical joints		
Monitored volumes		

Header and control cards

The first line in the file must start **`RADIOSS STARTER`** and the INVERS number (columns 17 to 24) must be 31. If this is not the case, Primer will assume that the file is not a Radioss version 3.1 starter file and will not be able to translate the file.

CARD1 - TITLE is used by Primer for the analysis title.

IGRAX, IGRAY and IGRAZ are translated to LOAD_BODY_X, _Y and _Z respectively.

Materials

Only some materials in the RADIOSS starter file can be translated. This is because there are some RADIOSS materials that have no equivalent in LS-DYNA.

RADIOSS material law 1: Elastic

RADIOSS material type 1 translates directly to an elastic material **`(*MAT_ELASTIC`**) in LS-DYNA

RADIOSS material law 2: Elastic/plastic

RADIOSS uses a power-law equation to describe the stress-strain behaviour. There is no exact equivalent in LS-DYNA so a piece-wise linear approach is used (material type **`*MAT PIECEWISE_LINEAR_PLASTICITY`**). The stress-strain curve is created using the power law and input parameters. Similarly, the rate effect equation is implemented by creating a further curve for LS-DYNA.

Primer tests to see if LS-DYNA material type **`*MAT_PLASTIC_KINEMATIC`** can be used instead (bi-linear elastic plastic). This will happen if the RADIOSS power is 0.0 (no strain hardening), or if the power is 1.0 and no maximum stress has been defined (bi-linear power law).

Because of the look-up table operations, the LS-DYNA model will not run very quickly; if exact correlation with RADIOSS is not required then we recommend that the materials be converted by hand back to bi-linear elastic-plastic (type 3).

RADIOSS material law 21: Foam/Honeycomb

The most direct equivalent of this RADIOSS material is LS-DYNA material type **`*MAT SOIL_CONCRETE (78)`**. This material allows a curve of pressure vs compaction while still allowing a pressure-sensitive yield criterion. However, there is a special case where the behaviour can be replicated more simply using the crushable foam material **`*MAT CRUSHABLE_FOAM(63)`** - see final paragraph in this section.
The pressure-vs-volume strain relationship is converted to be suitable for LS-DYNA (LS-DYNA true strain = -log(1/(1.+RADIOSS strain))) and the maximum volume strain is included as a steep slope at the end of the curve.

In RADIOSS, the curve can be steeper than the elastic modulus: often the RADIOSS models have very low Young's moduli. This is not permitted in LS-DYNA: it would cause the timestep to be reduced and there would be energy gain caused by hysteresis loops going the wrong way. Therefore we must raise the Young's modulus or change the curve. Primer checks each section of the curve, and if steeper than the bulk modulus, revises the curve and warns the user. This avoids the problems listed above but because the LS-DYNA curve is then softer than the RADIOSS curve, unwanted bottoming-out behaviour may occur.

As an option, the user may request that all RADIOSS materials which use law 21 have their Moduli multiplied by a factor (e.g.10) - the curve segments will then be permitted to be stiffer. The new modulus will then be checked against a user-supplied minimum modulus, and raised to match it if necessary. To do this, use the options in the RADIOSS translation window. Set values for the multiplication factor and minimum value for Young's Modulus.

The yield vs pressure relationship is determined by a quadratic power law with plateau value (A0, A1, A2 and AMAX). These must be converted by Primer to a load curve for LS-DYNA, and this in turn requires a sensible range of pressures to form the x-axis of the load curve. Primer uses a range of -5 to +50: this is sensible for foams in the usual millimetre units but would be very inappropriate if the units are not millimetres and Newtons.

The RADIOSS tensile and unloading bulk moduli are ignored (in LS-DYNA, assumed to be defined by Young's modulus and Poisson's ratio)

If A0=0.0,A1=0.0,A2=3.0 then the behaviour is that of crushable foam. Primer translates the material as type 63 (**`*MAT_CRUSHABLE_FOAM`**). The curve is converted to stress-strain using the following assumptions:

LS-DYNA volumetric strain = 1.0 - (1.0/(RADIOSS strain + 1.0))

LS-DYNA direct stress = 3 x pressure

Again, the slopes of the curve are checked against the Young's modulus and revised if necessary to maintain the timestep.

RADIOSS material law 19: Elastic orthotropic

This material is translated to LS-DYNA material **`*MAT_ORTHOTROPIC_ELASTIC (2)`**. The stiffness reduction in compression parameter (Re) is ignored.

RADIOSS material laws 3,4,22,23,27

These RADIOSS materials are similar to material law 2 with some extra parameters and are translated in the same way, so some information may be lost (for example damage parameters). The user should check these material models after translation to satisfy themselves that a suitable translation has been achieved. If not some editing of the keyword deck may be required. Also in some circumstances the LS-DYNA material type **`*MAT_ENHANCED_COMPOSITE_DAMAGE`** may be more appropriate.

RADIOSS material law 28

This material is translated to LS-DYNA material 26 (**`*MAT_HONEYCOMB`**). The load curves are translated to volumetric strain for LS-DYNA. RADIOSS allows the user to specify the local coordinate system of each element individually, with respect to a reference plane defined on property set type 6. In LS-DYNA there is no similar function, but by swapping the values for each direction on the material card in LS-DYNA it is possible to achieve the same effect. Currently the swapping has to be done by hand since Primer cannot do this automatically. It should be also noted that the orthotropic angle referred to in the RADIOSS property set 6 cannot be incorporated into the LS-DYNA keyword deck.

Material law 36

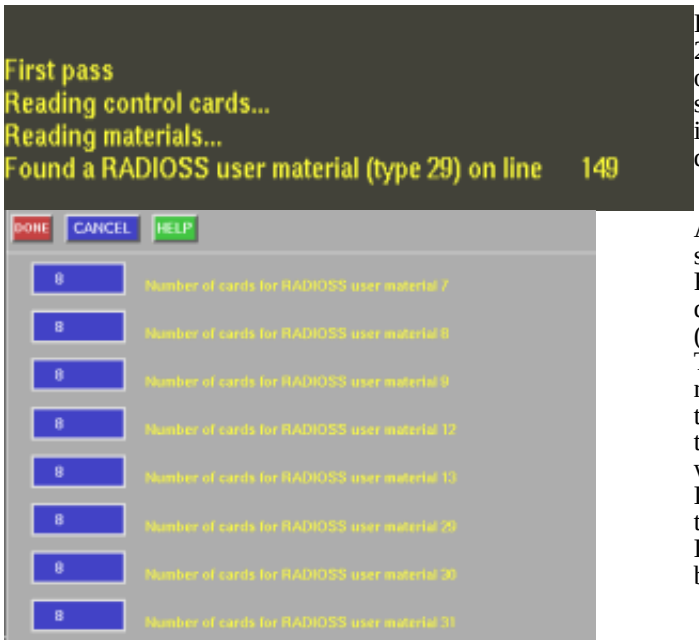
This is translated into material 24 in LS-DYNA (**`*MAT_PIECEWISE_LINEAR_ PLASTICITY`**). Only the first

function is translated so any multi-functional parts are lost.

Unsupported Materials

Where an unsupported material model is encountered Primer will translate them as **`*MAT_NULL`**. The user then has to change the material model and input parameters by hand to a suitable material model before running the LS-DYNA model. All translated **`*MAT_NULL`** materials have zero density to act as a error trap in LS-DYNA.

User defined materials



In a RADIOSS starter file, material laws 7, 8, 9, 12, 13, 29, 30 and 31 are user defined materials. As the length of a user defined material in RADIOSS is unknown some information is needed by the translator when one is encountered. A warning will be printed in the dialogue box.

Another window will also be displayed on the screen showing all the user defined material numbers in RADIOSS and how many cards each one requires. The default number for each material law is set to 8 cards (each material in RADIOSS has 7 compulsory cards). The user needs to edit the number of cards for the required user defined material in this window. Once this has been done and the **DONE** button pressed the translation will proceed. The user defined materials will be translated to user defined materials in LS-DYNA as shown in the following table. However, there will be no material parameters in the materials in LS-DYNA. The user will need to edit the material back to something sensible.

RADIOSS material law	LS-DYNA user defined material
7	41
8	42
9	43
12	44
13	45
29	46
30	47
31	48

Equivalent RADIOSS and LS-DYNA user defined materials.

Skew Frames

Moving skew frames are translated to **`*DEFINE_COORDINATE_NODES`** (fully supported)

Fixed skew frames are translated to **`*DEFINE_COORDINATE_VECTOR`** (fully supported)

Property sets

Type 0 (void) Skipped

Type 1 (shells) Supported. Only N, Thick and A_{shear} are translated.

Type 2 (truss) Supported. Initial gap is not translated.

Type 3 (beam) Supported. Rotations at nodes are ignored.

Type 4 (spring) RADIOSS spring properties are defined by a number of parameters describing force as a function of deflection and deflection rate, with various options for treating unloading. In the general case, there is no equivalent in LS-DYNA, but if certain combinations of parameters have been left zero in the RADIOSS deck, a close equivalent in LS-DYNA does exist.

Primer identifies the following cases:

a) If curve N1 is zero (linear spring) but K is non-zero, the spring is assumed to be linear elastic. This translation is exact when C is also zero; if C is non-zero then the rate effect will be missing in LS-DYNA.
b) If curve N1 is zero (linear spring) and K is also zero, a linear damper is assumed with coefficient C. This is an exact translation.
c) Otherwise the spring is assumed to be nonlinear elastic, with rate effect: curve N1 is force-deflection and curve N2 is rate effect. This translation is exact only when H, A and B are zero. If H is non-zero, the unloading response is missing in LS-DYNA; this could lead to serious errors since no permanent deflection could occur.

Type 5 (Rivet and Spotweld) Supported. Maximum normal and tangential forces are translated.

Type 6 (Orthotropic solid element) Translated to a normal solid element property. Any orthotropic angles will be lost.

Type 7 (Airbag) Translated into a Wang Nefske airbag in LS-DYNA. In LS-DYNA the ambient density is required when creating a control volume. Primer attempts to create a suitable value for the density by looking at the values given for the Gas constant and the specific heat. This value may not be correct and will need checking. Functions used for airbags in Radioss use total mass vs. time. In LS-DYNA the function is mass flow rate vs. time. The Radioss function is differentiated to get the mass flow rate curve for LS-DYNA.

Type 8 (General spring). The best equivalent of a general spring in LS-DYNA is a discrete beam element. As with property type 4 springs only some combinations of values can be translated correctly. A linear discrete beam will be created if the properties in all 6 degrees of freedom are linear, and a non-linear discrete beam will be created if all are non-linear. If a mixture of properties are found 2 discrete beams need to be created - one linear and one non-linear - with the appropriate properties for each degree of freedom. As the spring is changed into a beam element in LS-DYNA the element number will change to avoid any potential clashes.

Type 9 (Orthotropic Shell) and **type 10 (Composite shell**).These are translated as a normal shell. Any orthotropic properties will be lost.

Type 11 (Composite shell). Material angles will be lost but a user defined integration law is created with the correct layer thicknesses.

Type 12 (3 noded springs). Not supported. These will be skipped.

Type 13 (beam type spring). Translated in the same way as type 9 general springs.

Type 14 (Solid) Supported

Accelerometers

Accelerometers in RADIOSS are defined by one node and a skew system (coordinate system). In LS-DYNA they are defined using 3 nodes which define a local triad. These nodes must also be on one rigid body. To translate the accelerometers two extra nodes are created which together with the first node make up this triad. A rigid part is then created which contains a beam representation of this triad. Finally the 3 nodes are used to create the accelerometer. N.B as any skew system can be used in RADIOSS to define an accelerometer, if a moving skew frame (defined by nodes) is used which is defined with 3 nodes somewhere else in the model the accelerometer in LS-DYNA will give different results. This is because the accelerometer in RADIOSS will give results depending on how the moving skew frame rotates. The accelerometer for LS-DYNA will give the results depending on how the rigid triad rotates.

Interfaces

Interfaces in RADIOSS are defined in a very similar way to LS-DYNA and so can be translated. The following table shows which contact types in LS-DYNA the RADIOSS interfaces are translated to.

RADIOSS Interface type	LS-DYNA contact type
2 TIED	***CONTACT_TIED_NODES_TO_SURFACE or** *CONTACT_TIED_SURFACE_TO_SURFACE
3 SLIDE/VOID	***CONTACT_AUTOMATIC_NODES_TO_SURFACE or** *CONTACT_AUTOMATIC_SURFACE_TO_SURFACE
4 SLIDE/VOID	***CONTACT_AUTOMATIC_SINGLE_SURFACE**
5 SLIDE/VOID	***CONTACT_AUTOMATIC_NODES_TO_SURFACE or** *CONTACT_AUTOMATIC_SURFACE_TO_SURFACE

6 SLIDE/VOID	***CONTACT_RIGID_BODY_TWO_WAY_TO_RIGID_** BODY
7 SLIDE/VOID	***CONTACT_AUTOMATIC_NODES_TO_SURFACE or** *CONTACT_AUTOMATIC_SURFACE_TO_SURFACE or *CONTACT_AUTOMATIC_SINGLE_SURFACE
8 SLIDE	***CONTACT_DRAWBEAD**
10 TIED/VOID	***CONTACT_AUTOMATIC_NODES_TO_SURFACE or** *CONTACT_AUTOMATIC_SURFACE_TO_SURFACE

Surface input types 1 (segments), 2 (nodes), 4 (shell property sets) and 5 (shell elements) are translated directly. Input type 3 (shell materials) has no equivalent in LS-DYNA so it is translated to a list of parts which use these materials. Interfaces in RADIOSS can have the same entities in the slave and the master side of the contact. This could cause problems in LS-DYNA so for each contact any entities which occur in the slave and master surfaces are removed from the master side.

For RADIOSS interface type 4, any entities which are left in the master side of the contact after this process are added to the slave side and a single surface contact created.

Interface type 7 can be used to replace more than one type of contact in RADIOSS. If a type 7 contact is found which does not use nodes for the slave it is translated to a surface to surface contact and a single surface contact.

Rigid Walls

Rigid walls in RADIOSS are translated to LS-DYNA rigid walls as follows

RADIOSS Type	**LS-DYNA type**
1 infinite plane	***RIGIDWALL_PLANAR_OPTION**
2 infinite cylinder	***RIGIDWALL_GEOMETRIC_CYLINDER**
3 sphere	***RIGIDWALL_GEOMETRIC_SPHERE**
4 parallelogram	***RIGIDWALL_PLANAR_FINITE_OPTION**

The OPTION on the LS-DYNA rigidwalls can be MOVING if the rigid wall in RADIOSS has a velocity. However in RADIOSS a rigidwall can have a velocity in any direction. In LS-DYNA velocity is restricted to the direction of the normal defining the rigidwall. The vector magnitude of the velocity from the RADIOSS rigidwall is used and so the direction and magnitude of the velocity may be incorrect.

Rigid Bodies

Rigid bodies in RADIOSS can be defined by either nodes or property sets. The equivalent of a rigid body defined by nodes in LS-DYNA is a ***CONSTRAINED_NODAL_RIGID_BODY**. However if this is used then no external motions can be applied to the rigid body (e.g. ***BOUNDARY_PRESCRIBED_MOTION_RIGID**). To overcome this, if a rigid body in RADIOSS is defined by nodes a new part, beam section and rigid material are created. Beams with negligible mass are created from the primary node on the rigid body to all other nodes. This also helps in visualisation of the rigid body.

If the primary node on a rigid body is at (0,0,0) then RADIOSS will calculate the centre of mass and inertia for the rigid body on initialisation and then move the node to the centre of mass. If this node was included in the rigid body definition in LS-DYNA the inertia and centre of mass would be incorrect. If the master node is found to be at the origin it is not translated and the next node in the rigid body is used as the master node.

Rigid bodies defined by property sets are translated as rigid parts by Primer with ***CONSTRAINED_RIGID_BODIES** to merge the parts together. If the master node is not at the origin it is added to the rigid body by using a ***CONSTRAINED_EXTRA_NODE** card.

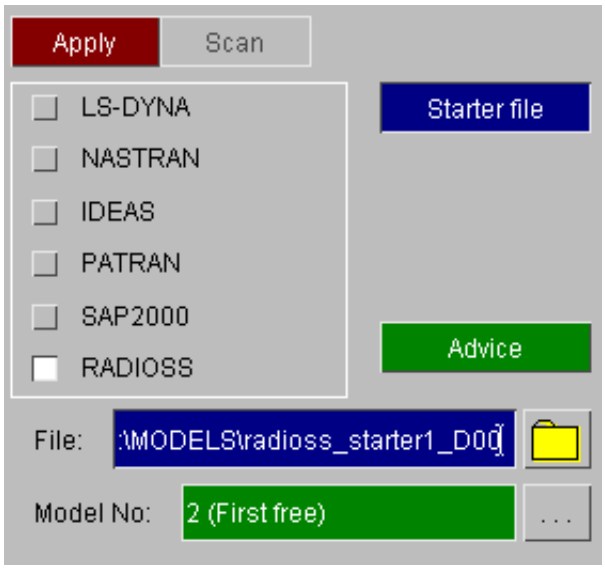
If there is a mass and inertia tensor for the rigid body it is translated and a ***PART_INERTIA** card defined. If there is no mass and inertia a ***PART** card is defined and LS-DYNA will calculate the mass and inertia tensor. If there is only one of the mass and inertia present a warning will be printed.

Fixed velocities

Fixed velocities in RADIOSS are translated to ***BOUNDARY_PRESCRIBED_MOTION_ NODE** unless the node is a primary node on a rigid body. If this is the case it is translated to ***BOUNDARY_PRESCRIBED_MOTION_RIGID**.

RADIOSS block format

The RADIOSS Translator function is aimed specifically at translating RADIOSS starter and engine files into LS-DYNA keyword format data. The block format is the default format for the RADIOSS translator.



The RADIOSS translator is invoked identically to all other formats read into PRIMER. Please refer to section 3.2 of the main manual for details on the **READ** function, remembering to select the RADIOSS sub-type.

For a translation the starter file must be present. An engine file can optionally be selected. The generic file extension for a starter file is 00 (Radioss starter files usually end with d00 or D00), but this can easily be modified in the file selector panel. Once the correct file has been selected and the **APPLY** button pushed, another window will be created which allows an engine file to be selected and some options to be set.

The buttons in the Radioss Translation Defaults box are as follows:

APPLY accept the defaults in the panel and proceed with the translation.

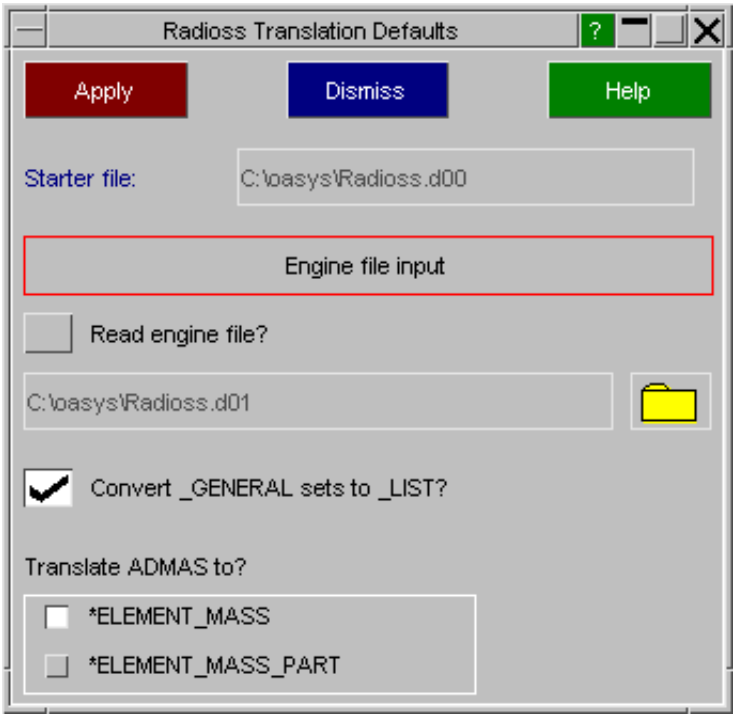
DISMISS terminate the translation process, returning to the generic READ panel.

HELP will create a message box full of useful information about the function of this panel.

READ ENGINE FILE button can be checked to select an engine file by either typing in the name in the text box or by pressing the button which will bring up the file selection box. Engine files by default have the extension 01 (Radioss engine files usually end with d01 or D01). The engine file is translated by default as long as a *01 file exists that corresponds to the *00 starter file.

CONVERT _GENERAL sets to _LIST can be used to convert sets that use the _GENERAL option to _LIST sets.

Translate ADMAS to? can be used to specify the method of translation for /ADMAS cards.



Current known issues

- Not all options are supported for defining groups and surfaces.
- Only a small number of materials are translated at present.

Notes on specific keywords

Please see the following table which uses the following colours to indicate how well supported the keyword is.

Not supported
Limited support. A small subset of the options is supported (details given)
Reasonable support. Most options supported (details given)
Radioss 4.4 only - Reasonable support (details given)
Fully supported
Radioss 4.4 only - Fully supported

Starter file

Keyword	Notes
/ACCEL	Accelerometers are translated to *DATABASE_HISTORY_NODE_LOCAL_ID. **Note that Fcut is not translated and the output will not be filtered.**
/ADMAS	Masses are translated to *ELEMENT_MASS. Additionally so you can tell which masses are created for each original /ADMAS card, a *GROUP is created for each /ADMAS containing the *ELEMENT_MASS masses created.
/ANALY	Not supported.
/BCS	/BCS cards using a node group will be translated to *BOUNDARY_SPC_SET_ID. Any secondary nodes will be translated to *BOUNDARY_SPC_NODE.
/BEAM	Translated to *ELEMENT_BEAM.
/BRICK	Translated to *ELEMENT_SOLID.
/BRIC20	Translated to *ELEMENT_SOLID. Midside nodes are ignored.
/CLOAD	/CLOAD cards using a node group will be translated to *LOAD_NODE_SET. Any secondary nodes will be translated to *LOAD_NODE_POINT. **Note that the sensor input is not translated**.
/CYL_JOINT	Not supported.
/DEF_SHELL	Element formulation value is set to 2 (Belytschko-Tsay). If Ishell in input deck <> 1 (Belytschko), a warning is issued.
/DEF_SOLID	A warning is issued and the element formulation value set to 1.
/END	Anything after /END is ignored.
/FUNCT	Translated to *DEFINE_CURVE.
/GRAV	Not supported.
/GRBEAM	All types are supported. BEAM is translated to a *SET_BEAM. PART is translated to a *SET_BEAM_GENERAL using option PART. BOX and BOX2 are translated to a *SET_BEAM_GENERAL using option BOX (box is created). SUBSET, MAT and PROP are translated to a *SET_BEAM_GENERAL using option PART (an equivalent list of parts is generated).

/GRBRIC	All types are supported. BRIC is translated to a *SET_SOLID. PART is translated to a *SET_SOLID_GENERAL using option PART. BOX and BOX2 are translated to a *SET_SOLID_GENERAL using option BOX (box is created). SUBSET, MAT and PROP are translated to a *SET_SOLID_GENERAL using option PART (an equivalent list of parts is generated).
/GRNOD	All types **except SURF and NODENS** are supported. NODE is translated to a *SET_NODE. GENE is translated to *SET_NODE_GENERATE. PART is translated to a *SET_NODE_GENERAL using option PART. BOX and BOX2 are translated to a *SET_NODE_GENERAL using option BOX (box is created). SUBSET, MAT and PROP are translated to a *SET_NODE_GENERAL using option PART (an equivalent list of parts is generated).
/GRQUAD	All types are supported. QUAD is translated to a *SET_SHELL. PART is translated to a *SET_SHELL_GENERAL using option PART. BOX and BOX2 are translated to a *SET_SHELL_GENERAL using option BOX (box is created). SUBSET, MAT and PROP are translated to a *SET_SHELL_GENERAL using option PART (an equivalent list of parts is generated).
/GRSH3N	All types are supported. SH3N is translated to a *SET_SHELL. PART is translated to a *SET_SHELL_GENERAL using option PART. BOX and BOX2 are translated to a *SET_SHELL_GENERAL using option BOX (box is created). SUBSET, MAT and PROP are translated to a *SET_SHELL_GENERAL using option PART (an equivalent list of parts is generated).
/GRSHEL	All types are supported. SHEL is translated to a *SET_SHELL. PART is translated to a *SET_SHELL_GENERAL using option PART. BOX and BOX2 are translated to a *SET_SHELL_GENERAL using option BOX (box is created). SUBSET, MAT and PROP are translated to a *SET_SHELL_GENERAL using option PART (an equivalent list of parts is generated).
/GRSPRI	All types are supported. SPRI is translated to a *SET_BEAM. PART is translated to a *SET_BEAM_GENERAL using option PART. BOX and BOX2 are translated to a *SET_BEAM_GENERAL using option BOX (box is created). SUBSET and PROP are translated to a *SET_BEAM_GENERAL using option PART (an equivalent list of parts is generated).
/GRTRUS	All types are supported. TRUS is translated to a *SET_BEAM. PART is translated to a *SET_BEAM_GENERAL using option PART. BOX and BOX2 are translated to a *SET_BEAM_GENERAL using option BOX (box is created). SUBSET and PROP are translated to a *SET_BEAM_GENERAL using option PART (an equivalent list of parts is generated).
/IMPDISP	If a node group is used it is translated to *BOUNDARY_PRESCRIBED_MOTION_SET with VAD=2. Secondary nodes are translated to *BOUNDARY_PRESCRIBED_MOTION_NODE with VAD=2. **Note that the sensor input is not translated**. If a skew system is used a *DEFINE_VECTOR card is created using the skew system and direction and the DOF is set to 4.
/IMPVEL	If a node group is used it is translated to *BOUNDARY_PRESCRIBED_MOTION_SET with VAD=0. Secondary nodes are translated to *BOUNDARY_PRESCRIBED_MOTION_NODE with VAD=0. **Note that the sensor input is not translated**. If a skew system is used a *DEFINE_VECTOR card is created using the skew system and direction and the DOF is set to 4 for translation or 8 for rotation.
/INISTA	Not supported.
/INIVEL	If a node group is used it is translated to *INITIAL_VELOCITY. Secondary nodes are translated to *INITIAL_VELOCITY_NODE. **Note that the title will be lost as the LD-DYNA keywords do not support it**.

/INTER	TYPE2: Converted to *CONTACT_TIED_SHELL_EDGE_TO_SURFACE_OFFSET. This is used instead of *CONTACT_TIED_NODES_TO_SURFACE as it is better for tying spotweld beams onto shells (the rotational dof is handled correctly with tied shell edge, with tied nodes to surface it is not). The _OFFSET option is used so that the contact will work correctly if any nodes on the contact segments are in constraints (e.g. nodal rigid bodies TYPE 3: Converted to *CONTACT_AUTOMATIC_SURFACE_TO_SURFACE TYPE5: Converted to *CONTACT_AUTOMATIC_NODES_TO_SURFACE TYPE6: Converted to *CONTACT_RIGID_BODY_TWO_WAY_TO_RIGID_BODY TYPE7: Converted to *CONTACT_AUTOMATIC_NODES_TO_SURFACE Types 8, 10, 11 and 14 not supported. **Note that some contacts will be better translated as *CONTACT_AUTOMATIC_SINGLE_SURFACE. The user will need to review the contacts and decide this**.
/IOFLAG	Not supported.
/LINE	Only type SEG is supported. It is translated to a *SET_SEGMENT.
/MADYMO	Not supported.
/MAT	ELAST (law 1): Translated to *MAT_ELASTIC. PLAS_JOHNS (law 2): If m is zero then there are no temperature effects and *MAT_SIMPLIFIED_JOHNSON_COOK is used. Otherwise *MAT_JOHNSON_COOK is used. PLAS_BRIT (law 27): Translated to *MAT_098 (Simplified Johnson Cook). HONEYCOMB (law 28): Translated to either *MAT_HONEYCOMB or to *MAT_MODIFIED_HONEYCOMB. If iflag1 != iflag2 - Translated to *MAT_126 along with a warning message. If iflag1= iflag2=0 - x = x/(1+x) ; New entry is inserted into beginning of load curve if current 1st entry is positive; Translated to *MAT_026. If iflag1 = iflag2 = 1 - Translated to *MAT_126. If iflag1 = iflag2 = 1 - Translated to *MAT_126; SFA = 1; AOPT, MACF set to 0. Checks are thrown in to see whether any MAT_MODIFIED_HONEYCOMB types use a solid section. If yes, and if no other material type uses that section, element formulation value is set to 9. A warning is issued and the element formulation value is retained at 1 if other types use the section as well. PLAS_TAB (law 36): Translated to *MAT_024 (Piecewise Linear Plasticity). Material card refers to a table that contains relevant load curves and strain rates if multiple load curves are specified. *MAT_ADD_EROSION is invoked if failure is specified. VISC_TAB (law 38): Translated to *MAT_057 (Low Density Foam).
/MEMORY	Not supported.
/MONVOL	**Types AREA and COMMU are not supported**. Type PRES is translated to *AIRBAG_SIMPLE_PRESSURE_VOLUME. Type GAS is translated to *AIRBAG_ADIABATIC_GAS_MODEL. Type AIRBAG is translated to *AIRBAG_SIMPLE_AIRBAG_MODEL. **No attempt is made to translate the airbag properties**. Only the surface is translated (to a *SET_SEGMENT).
/NODE	Translated to *NODE
/PART	Translated to *PART
/PENTA6	Translated to *ELEMENT_SOLID
/PLOAD	Not supported.

/PROP	/PROP/SHELL (pid 1) translated to *SECTION_SHELL /PROP/TRUSS (pid 2) translated to *SECTION_BEAM with ELFORM = 3 (truss). /PROP/BEAM (pid 3) translated to *SECTION_BEAM with ELFORM = 2 (resultant beam). /PROP/SPRING (pid 4) translated to *SECTION_BEAM with ELFORM = 6 (discrete beam). A *MAT_ELASTIC_SPRING_DISCRETE_BEAM material is created for linear and non-linear elastic springs. For elasto-plastic springs a *MAT_INELASTIC_SPRING_DISCRETE_BEAM material is created. Note that if B is non-zero the log term used in the Radioss and Dyna formulations will be used and this is not the same. Note that hardening types > 1 have no equivalent. /PROP/RIVET (pid 5) properties are copied to any *CONSTRAINED_SPOTWELD elements created from /RIVET elements. Fn is translated to SN. Ft is translated to SS. N and M on the *CONSTRAINED_SPOTWELD are set to 1.0. **The maximum length and rotation flag are not supported**. /PROP/SOL_ORTHO (pid 6) translated to *SECTION_SOLID. Relevant *ELEMENT_SOLID cards are converted to *ELEMENT_SOLID_ORTHO cards. Local material directions are also inserted. /PROP/SPR_GENE (pid 8) and /PROP/SPR_BEAM (pid 13) translated to *SECTION_BEAM with ELFORM = 6 (discrete beam). A *MAT_GENERAL_SPRING_DISCRETE_BEAM material is created. Note that if B is non-zero the log term used in the Radioss and Dyna formulations will be used and this is not the same. Note that hardening types > 1 have no equivalent. SCOOR is set to 2 if all beams that refer to parts that, in turn, refer to this section are of finite length SCOOR is set to 0 if all beams that refer to parts that, in turn, refer to this section are of zero length SCOOR is set to 2 if we have a mixture; zerolength beams are then added to a set and a warning issued. /PROP_SH_SANDW (pid 11) translated to *SECTION_SHELL. *PART cards are created for each layer. Relevant *INTEGRATION_SHELL card written. /PROP/SOLID (pid 14) translated to *SECTION_SOLID. Calls DEF_SOLID if elform is set to 0.
/QUAD	Translated to *ELEMENT_SHELL
/RANDOM	Not supported.
/RBODY	Translated to *CONSTRAINED_NODAL_RIGID_BODY with a PNODE node. Note values of ICOG other than 1 are ignored (there is no equivalent in LS-DYNA). Mass and inertia properties are ignored (there is no equivalent in LS-DYNA). INERTIA option is now invoked if positive components of inertia tensor are input along with positive value of mass. If the tensor is spherical a small perturbation is added to Ixx. If a local coordinate system is given and the tensor has off diagonal terms this is illegal for IRCS=1 in LSDYNA. In this case the tensor is rotated back to the global system and IRCS, CID and CID2 are all set to zero.
/REFSTA	Not supported.
/RIVET	Translated to *CONSTRAINED_SPOTWELD
/RLINK	Not supported.
/RWALL	/RWALL/CYL is translated to *RIGIDWALL_GEOMETRIC_CYLINDER. /RWALL/SPHER is translated to *RIGIDWALL_GEOMETRIC_SPHERE /RWALL/PLANE is translated to *RIGIDWALL_PLANAR /RWALL/PARAL is translated to *RIGIDWALL_PLANAR_FINITE Moving rigid walls are only supported for the PLANE and PARAL types.

/SECT	Translated to *DATABASE_CROSS_SECTION_SET cards. **Note that the 3 nodes defining the plane are ignored and the output will be in the global coordinate system. To get output in a local coordinate system in LS-DYNA an *ELEMENT_SEATBELT_ACCELEROMETER would need to be made with the 3 nodes (which would be referenced on the *DATABASE_CROSS_SECTION_SET card). This is not done because the *ELEMENT_SEATBELT_ACCELEROMETER needs to be on a rigid body. Making the 3 nodes part of a rigid body could significantly alter the results**. If a triangle group is used as well as a shell group the shells in the triangle group will be added to the shell group. Check that this will not cause problems elsewhere. If secondary nodes are used as well as a node group the secondary nodes will be added to the node group. Check that this will not cause problems elsewhere.
/SENSOR	Not supported.
/SH3N	If the shell has a thickness it is translated to *ELEMENT_SHELL_THICKNESS, otherwise *ELEMENT_SHELL. It is assumed that there are no label clashes with /SHELL elements.
/SHELL	If the shell has a thickness it is translated to *ELEMENT_SHELL_THICKNESS, otherwise *ELEMENT_SHELL
/SKEW	/SKEW/FIX translated to *DEFINE_COORDINATE_VECTOR /SKEW/MOV translated to *DEFINE_COORDINATE_NODES
/SPMD	Not supported.
/SPRING	Translated into *ELEMENT_BEAM (discrete beams). Numbering may change to prevent clashes with beam and truss elements
/SUBSET	Subset hierarchy and part contents are reproduced in Primer (see the part tree). Each subset is also translated to a *SET_PART. Note that as *SET_PARTs cannot be nested like subsets they will just be expanded to the list of parts. e.g. if subset 1 has child subsets 2 and 3, *SET_PART 2 will contain the parts in subset 2, *SET_PART 3 will contain the parts in subset 3, but *SET_PART 1 will contain the parts in subsets 1, 2 and 3.
/SURF	SEG is translated to a *SET_SEGMENT in DYNA. PART is translated to a *SET_SEGMENT_GENERAL using option PART. BOX and BOX2 are translated to a *SET_SEGMENT_GENERAL using option BOX (box is created). SUBSET, MAT and PROP are translated to a *SET_SEGMENT_GENERAL using option PART (an equivalent list of parts is generated) **SURF, GRSHEL, GRSH3N, ELLIPS and MDELLIPS are not supported.**
/TETRA4	Translated to *ELEMENT_SOLID
/TETRA10	Translated to *ELEMENT_SOLID
/TITLE	Translated to *TITLE
/TH	Only types NODE, SHEL, SH3N, BRIC, QUAD, BEAM, TRUSS and SPRING (i.e. nodes and elements) are translated. For type NODE, if there is a skew frame it is translated to *DATABASE_HISTORY_NODE_LOCAL_ID. Everything else is translated to *DATABASE_HISTORY_xxx_ID where xxx is the appropriate type. *DATABASE_RWFORC and *DATABASE_SECFORC files are used by both the /TH cards and the RADIOSS Engine keyword /ANIM/VECT/FOPT. The /TH keywords grab their DT value from the /TFILE keyword (if present). The /ANIM/VECT/FOPT keyword gets its DT value from the /ANIM/DT keyword (if present). In case of a clash, the /TH and /TFILE keywords take preference over the /ANIM keywords.
/TRUSS	Translated into *ELEMENT_BEAM (truss). Numbering may change to prevent clashes with beam and spring elements

Engine file

Engine file Keyword	Notes
/ANIM	The following options are translated: /DT - Start time is ignored; Frequency of output is specified /BRICK/TENS/STRAIN /SHELL/TENS/STRAIN /NODA/DT /NODA/DMAS /VECT/CONT /VECT/FOPT The following options are handled by default: /MASS /BEAM/FORC /TRUS/FORC /SPRING/FORC
/BCS	Modifies or inserts new *BOUNDARY cards.
/BCSR	Modifies or inserts new *BOUNDARY cards.
/DEL	Initiates a delete panel that permits the user to leave or delete elements specified in the /DEL card.
/DELINT	Not supported .
/DT	Cycle frequency (scale factor) is read in and translated. deltatmin is ignored. Options are ignored.
/DT1	Not supported.
/DTIX	Not supported.
/DYREL	Not supported.
/FUNCT	Redefines existing *DEFINE_CURVE cards. Inserts new card if the given curve Ifunc does not exist.
/GFILE	Not supported.
/INIV	Redefines existing *DEFINE_CURVE cards. Inserts new card if the given curve Ifunc does not exist.
/INTER	Not supported.
/KEREL	Not supported.
/KILL	User gets a warning to the effect that the restart file is written anyway (as in STOP)
/MADYMO	Not supported.
/MON	/OFF option warns the user that CPU time information is written by default.
/OUTP	Not supported.
/PARITH	Not supported.
/PATRAN	Not supported.
/PRINT	Not supported.
/PROC	Translated to *CONTROL_PARALLEL.
/RBODY	The RADIOSS Starter keyword is supported. The Engine keyword, however, is not.

/RFILE	Translated to *DATABASE_BINARY cards.
/RUN	Mandatory keyword translated to *CONTROL_TERMINATION.
/SHFRA	Not supported.
/STOP	Translated to *CONTROL_TERMINATION.
/TFILE	Translated to *DATABASE_BINARY cards. (refer /TH in the Starter file section).
/@TFILE	Not supported.
/TH	Version 31 is not currently supported. The VERS option is hence ignored.
/TITLE	Supported.
/VEL	Not supported.
/VERS	Mandatory keyword - 31 option warns the user that the translator was written to support version 4.1

Other notes

*DATABASE_ABSTAT
*DATABASE_GLSTAT
*DATABASE_MATSUM
*DATABASE_RCFORC
*DATABASE_RWFORC
*DATABASE_SECFORC
cards are automatically generated (as there is no LS-DYNA equivalent for most of the RADIOSS time history output (e.g. /TH/RWALL) unless one or more card has already been generated during translation..

Additionally a *CONTROL_CONTACT card is created with RWPNAL=1 so nodes on rigid bodies can interact with rigid walls.

During translation any keywords that are not supported will be written to a file <*filename*>.skipped. Any warnings that are generated will be written to a file <*filename*>.warnings

SAP2000 file format

The SAP2000 Translator function is aimed specifically at translating the SAP2000 ASCII input file (.s2k) into LS-DYNA keyword format data.

Briefly, SAP2000 (viz "Structural Analysis Program") marketed by CSI (Computers & Structures Inc.) is used for linear and simple non-linear analyses of building structures.

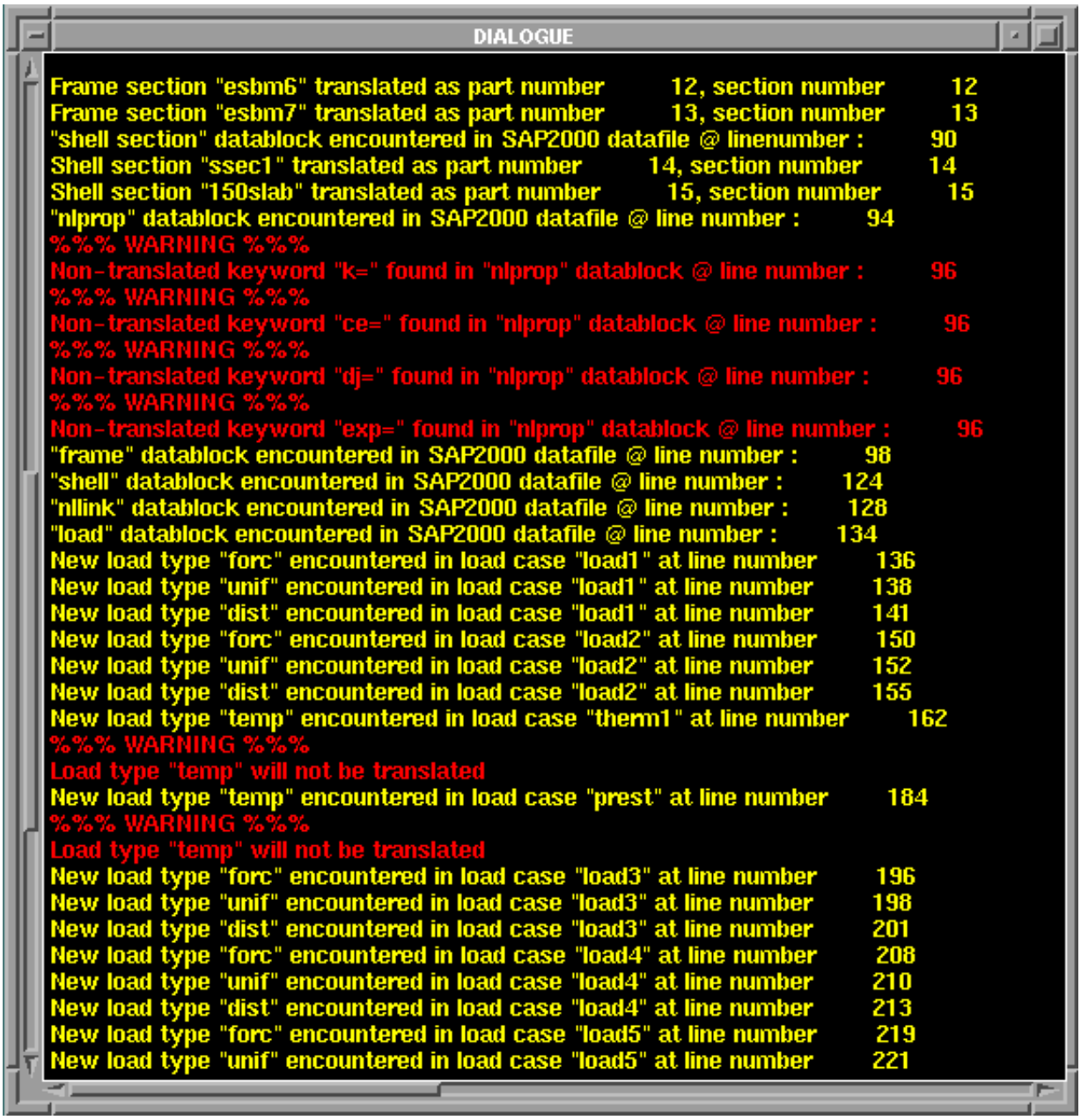
LS-DYNA is well suited to the cutting edge performance based earthquake engineering and enables extremely complex structures to be analysed against time history data recorded from past seismic events. In this way damage to the structure can be quantified and the most effective structure designed as a result of LS-DYNA analyses.

The SAP2000 translator is invoked identically to all other formats read into PRIMER. Please refer to the main manual for details on the **READ** function, remembering to select the SAP2000 sub-type,

The generic file extension expected is .s2k, but this can easily be modified in the file selector panel). Once the correct file has been selected and the **APPLY** button pushed, the SAP2000 translator will begin.

There are several items of note that the user should be aware of prior to use of the translator function:

1. The translator program creates several binary scratch files during the translation process. You should ensure that sufficient space is available on your disk. As a general guide you may expect to require 10 Mb of space for a complex model containing 10000 frame elements. These files will disappear as soon as the translation is completed.
2. The speed of the translation is not impressive. Please be patient whilst frame elements are read into PRIMER, as these will most likely be the largest portion of your model. There is a running commentary in the DIALOGUE panel which will present the current status of the translation. If an excessive number of frame rigid offsets or end-releases are specified in the model this will considerably increase the amount of time required to complete the translation.



3. The DIALOGUE panel will contain useful information about the status of the translation and what the translator is doing. It is advisable to enlarge the panel prior to beginning the translation process so that the data can be viewed more easily. Any warnings printed in this panel may be important, so please check anything printed.

Main translation panel

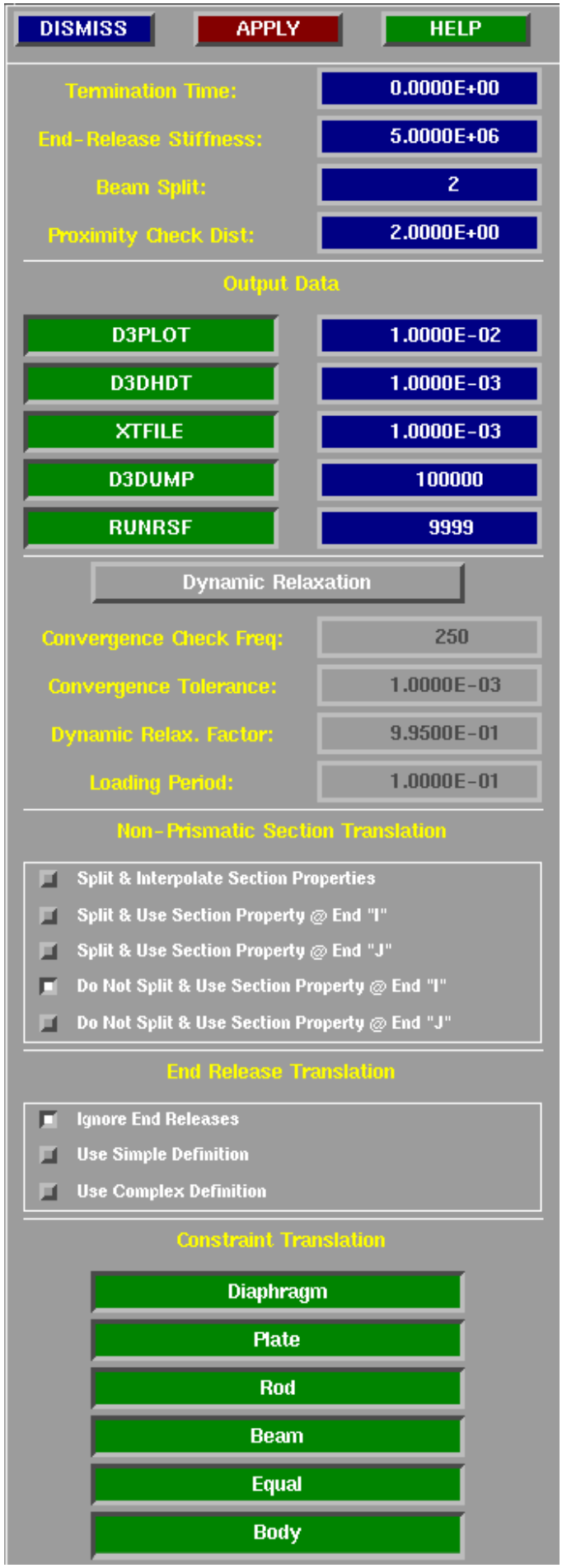
Before any data is read, a SAP2000 defaults panel will appear on the screen. This panel allows some options to be set prior to the translation process:

DISMISS terminate the translation process, returning to the generic READ panel.
APPLY accept the defaults in the panel and proceed with the translation.
HELP will create a message box full of useful information about the function of this panel.

Termination Time will set the final termination time of your transient LS-DYNA analysis. This value is also used to define loadcurve timing values.
End-Release Stiffness defines the stiffness of the discrete springs used represent pin releases at the end of frame elements. It is important the spring elements do not control the time step of the LS-DYNA analysis. If this occurs, the model may terminate due to numerical instabilities. To avoid allowing springs to dictate the time step of the model the stiffness should be small. However, this must be balanced against the need for the spring to keep the pinned nodes together. Refer to LS-DYNA User Manual for more data.
Beam Split defines the default number of beams into which a SAP2000 frame will be split if required - refer to the OPTIONS panel.
Proximity Check Dist defines a linear distance in the model units. If two nodes in the same rigid offset group are further apart than this distance a warning will be produced. This option is useful for checking the model.

Output Data allows the LS-DYNA DATABASE keyword cards which control the output frequency of results to be created in PRIMER. Each individual card can be switched on and off (green highlight implies on). The output frequency can only be modified when the card is 'on', else it is greyed-out.

Dynamic Relaxation allows dynamic relaxation (DR) options to be turned on and off (green highlight implies 'on'). When the DR option is selected, the following items can be edited:
Convergence Check Freq Number of cycles between DR convergence checks.
Convergence Tolerance value at which convergence is achieved.
Dynamic Relax Factor DR damping factor. This is a scale factor on velocity.
Loading Period Period over which the load which is being relaxed onto the model (i.e. self weight) is ramped to the full value. This 'ramping' avoids dynamic shock and oscillation in the model.



Non-Prismatic Section Translation
the concept of non-prismatic frame sections does not easily translate into LS-DYNA. For this reason a choice of several options is required. SAP2000 allows the section type to be defined at either enc of the non-prismatic frame element. These sections are then used to define the intermediate section properties. The options are as follows:

- Split the frame element into 'nseg' beam elements (nseg is defined on the SAP2000 frame element card), and interpolate the section data at either end of the frame creating new frame sections. Note that this can become overly messy if nseg is set to greater than 2. Also note that nseg can be overridden by the beam splitting options in the OPTIONS panel.
- Split the frame element as before but use the section data from end 'I' all the way through the frame.
- Split the frame element as before but use the section data from end 'j' all the way through the frame.
- Do not split the non-prismatic frames (unless specified otherwise) and use section 'I'.
- Do not split the non-prismatic frames (unless specified otherwise) and use section 'j'.

Refer to the Frame Element section of this appendix for more details on frame splitting and non-prismatic frames.

End Release Translation end releases (i.e. pin ends) also do not have an equivalent in LS-DYNA. Hence there are three options:

- Ignore end releases entirely.
- Use the simple definition.
- Use the complex definition.

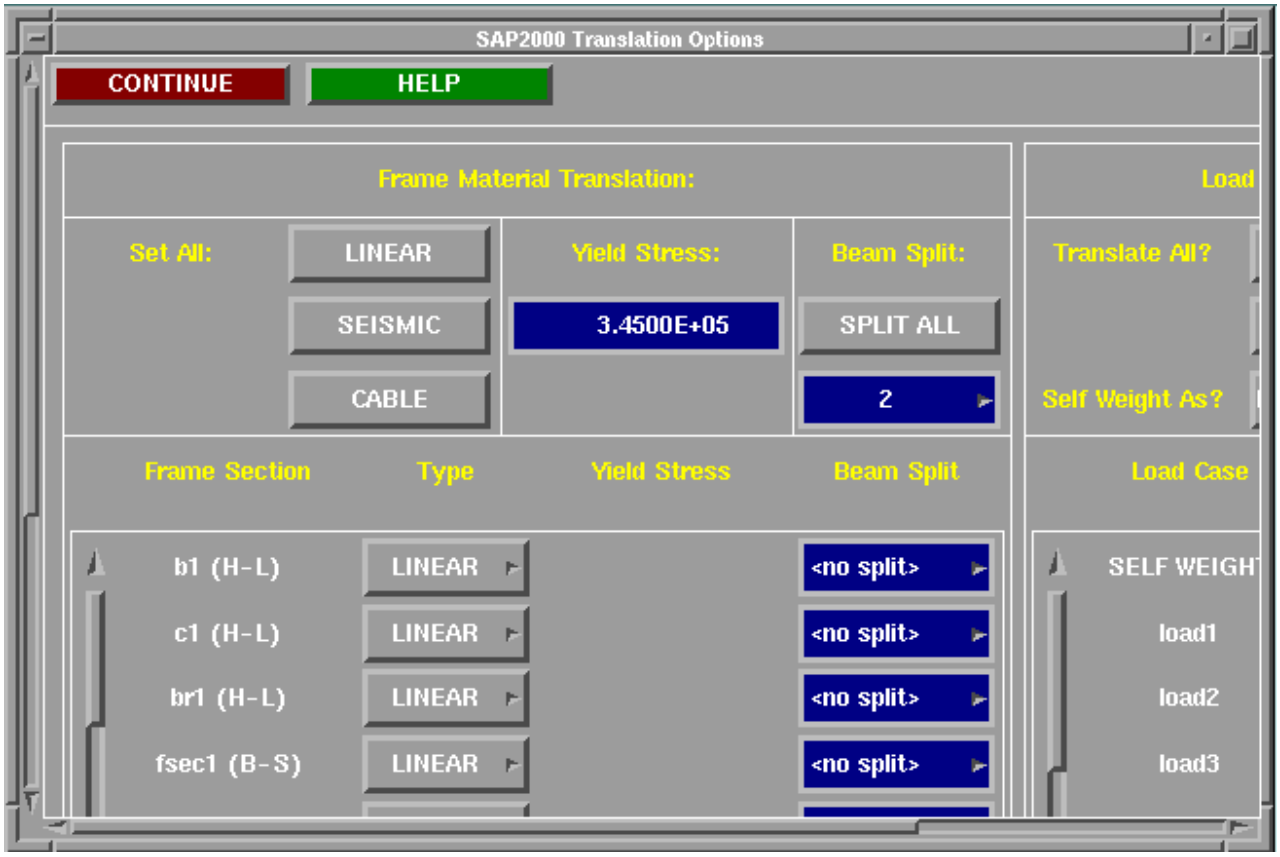
Please refer to the Frame Element section of this appendix for more details.

Constraint Translation Many of the constraint types in SAP2000 cannot be converted into LS-DYNA. This portion of the panel allows each type of constraint to be selected individually:

On (green highlight); all constraints of this type will be converted into nodal rigid bodies with all degrees of freedom connected.
Off (greyed out); all constraints of this type will be ignored.

Once the **APPLY** button from the main panel has been pushed the translation process will begin.

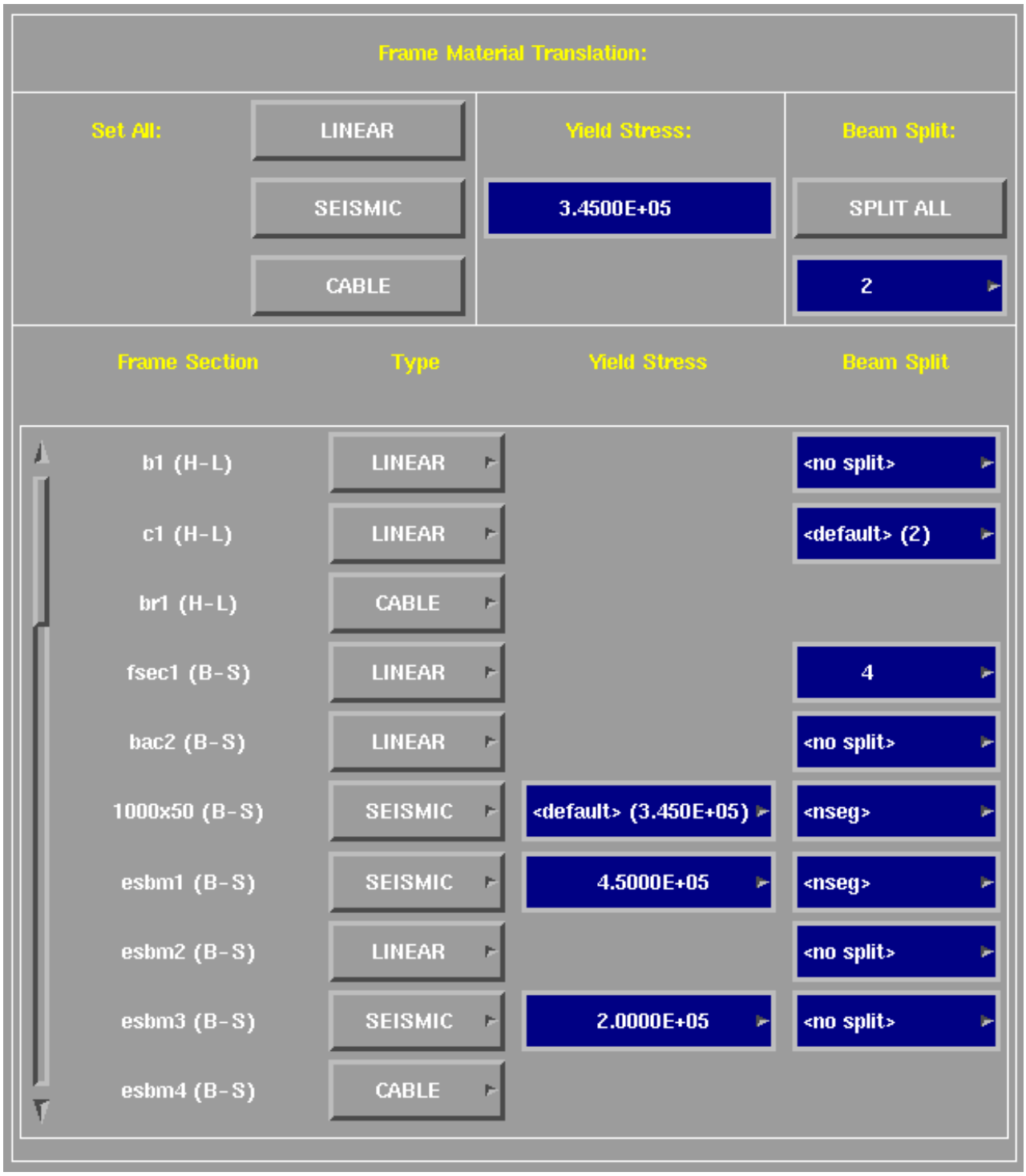
Stage one of the process reads all of the data from the SAP2000 ascii file and stores the majority of the data in the binary scratch files. Once stage one is completed a second window panel is created: SAP2000 OPTIONS - see figure below.



This panel is split into three areas; the static top area (**CONTINUE** - proceed with translation, and **HELP**), a frame options region and a loading options region.

Frame Translation Options

this panel allows the user to modify the material type of a frame component (part), or decide whether to split frame elements on an individual part basis.



The material options are as follows:

LINEAR: standard linear elastic material.

SEISMIC: will convert the standard linear material into a non-linear 'seismic beam' material capable of modelling plastic hinges and failure. The plastic moments of these frame components are automatically calculated provided that the geometry of the frame section was included in the SAP2000 ascii file. This shape is combined with the yield stress of the material to calculate the plastic properties. Where frame elements are expected to fail due to applied moments it is advisable to split seismic beams into at least 2 beams in order to capture the correct moment at each end. Note that;

seismic beams can only be created from Belytschko-Schwer beam sections (B-S).

CABLE: converts the standard frame sections to linear elastic discrete springs. These elements cannot be split and will ignore any end release definitions on frames in a 'cable' component - end releases have no meaning!

SET ALL enables all frame components to be set to a specific material type. The global **Yield Stress** and **Beam Split** options are adjacent. These global values become default values which apply to the entire model.

In the global **Beam Split** area all beams can be split with the current default value (i.e. the value from the main panel, 'nseg' (defined for each frame element in the SAP2000 ascii file) or some other even number typed in). A further option **SPLIT NONE** is available if no beams are to be split. A pop-up box can be displayed (by clicking the right mouse button) from the text box in order to define the default value.

Below the 'global' area is a list of frame component names, followed by data relevant to each individual component. If more than 10 components exist in the model the slider can be used to scroll through the components.

Each frame component name is followed by '(H-L)' implying Hughes-Lieu beam section or '(B-S)' for Belytschko-Schwer beam sections. As mentioned earlier, this enables the user to distinguish between components which can and cannot be modified to seismic beams. Refer to the Frame Section part of this appendix for details on how the frame section type is chosen in the translation.

The material **Type** is displayed in the next column. This can be chosen by cycling through the material options by clicking the left mouse button with the curser positioned over the screen button, or using the right mouse button to invoke a pop-up box.

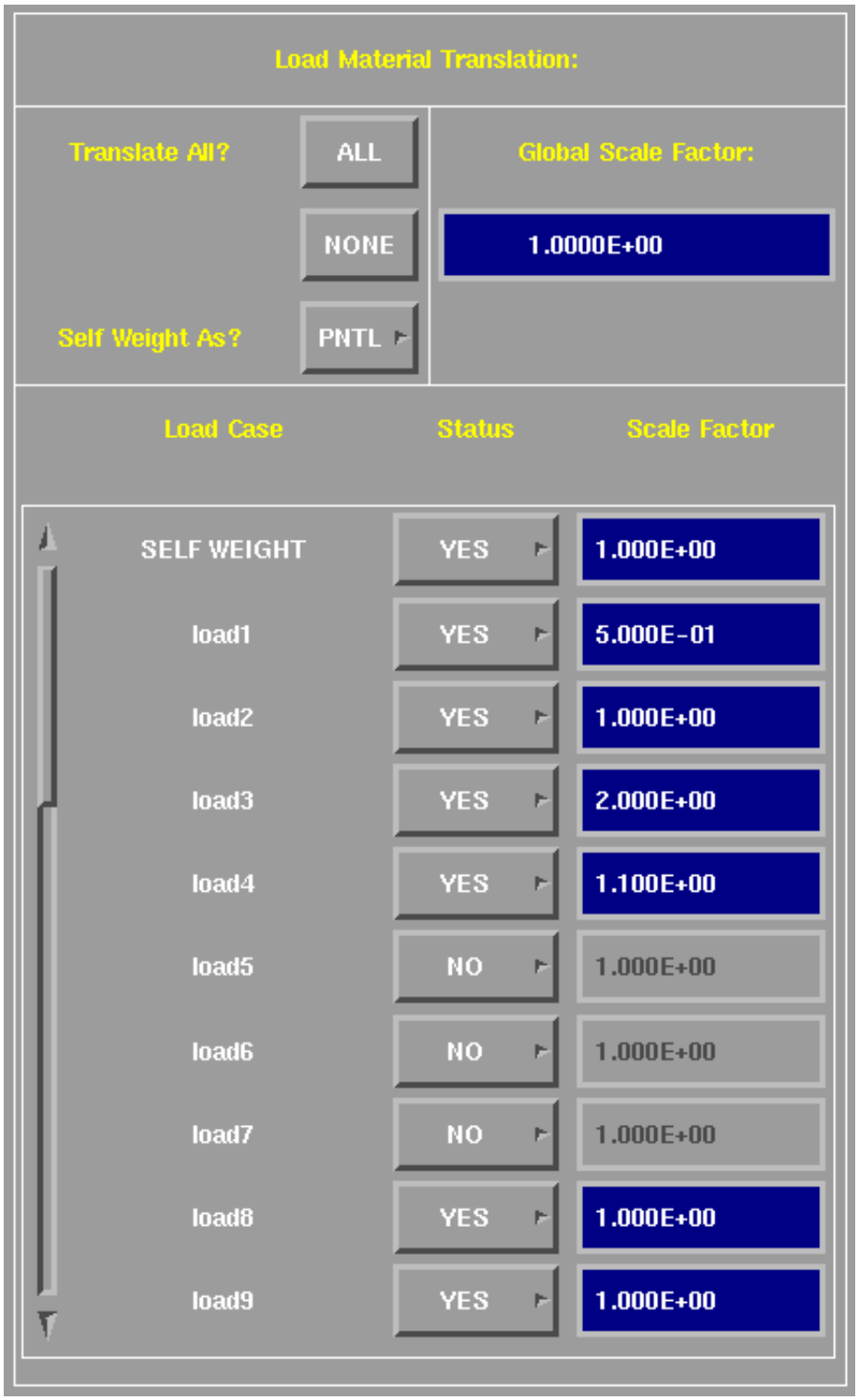
Yield Stress is only appropriate for seismic materials, and will only become available if a seismic material is chosen. The yield stress value can be typed in directly or the right mouse button can be used to invoke a pop-up which will reset the current default value.

The last column contains the **Beam Split** option for individual components. The options are: type a value in directly (even numbers only) or use the right mouse button to invoke a pop-up to choose <no split>, <nseg> or <default> for 'do not split this component', 'split using the nseg value on the frame element card' or 'split using the default value above'. The beam split option will be unavailable for cable materials as it is inappropriate.

Load Translation Options

Translate All allows the user to request all or none of the load cases defined in the SAP2000 ascii file to be translated.

Self Weight As? allows the self weight of the model to be represented in either of two ways: **PNTL** (POINT LOADS) where each node in the model that has weight associated with it will have a point load applied to it loading in the negative (downward) Z direction, or **GRAV** (GRAVITY) where the loading is applied as a gravity acceleration to the mass of the model. Use the self weight scale factor to multiply by the gravitational constant (i.e. 9.81 m/s2) for the **GRAV** option. The **PNTL** / **GRAV** option is chosen by invoking a pop-up box with the right mouse button.



A **Global Scale Factor** for all load cases can also be set in this window. Note that each load case that is selected for translation will be multiplied by the product of its own individual scale factor and the global scale factor.

The names of all load cases that are not empty are printed in a list in the panel. If there are more than 10 load cases then a slider will be created to enable the user to scroll through the load cases. Each separate load case can be selected and deselected by changing the load **Status** to **YES** (select) or **NO** (deselect). Where a load is selected for translation, an individual **Scale Factor** can be typed directly in. If the load is not selected, then the individual scale factor will be greyed out.

Once happy with the chosen options, the **CONTINUE** button should be pushed to complete the analysis.

Translator Functionality

This section of the appendix is meant to give a brief insight into the methods that the translator uses to process and store data, and the types of SAP2000 data which are understood.

Alternative Coordinate Systems

Alternative coordinate systems are supported by the translator in cartesian, cylindrical and spherical forms. Internally these are all mapped onto a cartesian system. The coordinate system will only be installed in the PRIMER database if it is specifically referenced by an entity in the model; i.e. a nodal restraint.

Joints

Joints (or nodes) can be read into PRIMER in cartesian, cylindrical or spherical coordinates systems (global or alternative). The joint coordinates are always transformed and stored in a global cartesian system. Note that the joint number will be retained in the final LS-DYNA model.

SAP2000 JOINT = LS-DYNA `*NODE`

Local Joint coordinate Systems

These are supported, but only when defined in a cartesian coordinate system. These joint local systems are stored identically to normal coordinate systems.

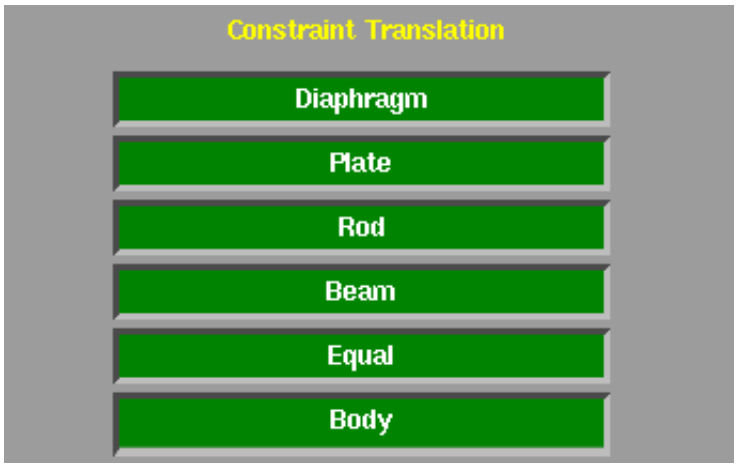
Restraints

Restraints are supported in either local, alternative or global coordinate systems (cartesian only). If a joint references a coordinate system, the coordinate system will be installed in the PRIMER database.

SAP2000 RESTRAINT = LS-DYNA `*BOUNDARY_SPC`
SAP2000 COORDINATE SYSTEM = LS-DYNA `*DEFINE_COORDINATE_SYSTEM`

Constraints

The following constraint types are recognised by the translator - the rest are ignored:



Each constraint type will be translated into PRIMER as a nodal rigid body where all degrees of freedom are constrained together for all nodes in the constraint group irrespective of the constraint type. Alternatively, it is possible to turn off and ignore constraints of a specific type.

SAP2000 CONSTRAINT = LS-DYNA `*CONSTRAINED_NODAL_RIGID_BODY`

Please refer to the section on Rigid Links etc for more details.

Welds

Welds are supported. If the nodes in the group are within the required distance, the nodes are grouped together in a nodal rigid body.

SAP2000 WELD = LS-DYNA `*CONSTRAINED_NODAL_RIGID_BODY`

Please refer to the section on Rigid Links etc for more details.

Patterns

Both gradient and standard pattern types are supported, but are stored in a binary scratch file. These are never installed in the PRIMER database as there is no LS-DYNA equivalent.

Spring Elements

Spring elements in SAP2000 are roughly equivalent to grounded springs in LS-DYNA, having only one node.

Spring elements in LS-DYNA cannot be coupled across freedoms, hence only the diagonal terms of the stiffness tensor of a coupled spring are translated. Each different component is translated as an individual element.

A new node is created adjacent to the original node. The new coexistent node is then fully restrained using a `*BOUNDARY_SPC`.

The orientation of each spring component is preserved by creating spring-damper orientation vectors (`*DEFINE_SD_ORIENTATION`). The translator will first check to see if an existing vector is parallel to the required direction. If a parallel vector is found it is referenced by the discrete element. Else, a new sd-orientation vector is created.

SAP2000 SPRING ELEMENT = LS-DYNA collection of `*ELEMENT_DISCRETE`s

Mass Elements

Mass elements are translated, and can be defined in global, local or alternative coordinate system space.

The translational mass defined in SAP2000 has directional components. This is not supported in LS-DYNA, hence the average value is used. Rotational masses are transformed into a global inertia tensor.

SAP2000 MASS ELEMENT = LS_DYNA `*ELEMENT_MASS` & `*ELEMENT_INERTIA`

Materials

Elastic isotropic materials are the only material type translated into LS-DYNA (Young's Modulus, Poisson's Ratio, mass density and weight density). All thermal materials are ignored.

SAP2000 MATERIAL = LS-DYNA `*MAT_ELASTIC`

Frame Elements & Sections

Frame Sections

PRISMATIC FRAMES:

The `*SECTION_BEAM` formulation used for a particular frame section depends on the data available: if the area (A) and the second moments of area (Iss, Itt & J) are present in the SAP2000 ascii file, a Belytschko-Schwer (resultant) beam formulation will be used (note that the shear area defaults to half the geomertic area, and that if different values of shear area are defined for the 's' and 't' axes an average value will be used), else the section geometry will be used to define a Hughes-Lieu (integrated) beam formulation. If there is inadequate data for either resultant or integrated formulations are present, then the translator will error terminate. Also note that for the more complex section shapes, if a Hughes-Lieu beam section is created, the associated `*INTEGRATION_BEAM` cards will also be generated. The component (`*PART`) will tie together the frame section name, the newly created section card, and the associated material.

NON-PRISMATIC FRAMES:

These frame sections are defined using existing frame sections. In these cases two new parts are created which reference the existing beam section definitions. Both parts will use the same frame section name.

SAP2000 FRAME SECTION = LS-DYNA `*SECTION_BEAM` (& `*INTEGRATION_BEAM` where required) & `*PART`

Frame Orientation

The beta angle system of orienting frame elements in SAP2000 is converted into a third node system in LS-DYNA. In the process of conversion the element axes become reversed (i.e. if the strong axis direction in SAP2000 was axis 1, then it would become axis 2 in LS-DYNA). This does not cause concern though as all frame elements will retain the correct orientation with respect to strong and weak axes.

Rigid Offsets

Where horizontal beams interface with vertical columns, the centre line geometry attributed to a one dimensional beam element is incorrect. The horizontal beam elements should terminate at the outer face of the vertical column. To facilitate this rigid links are created.

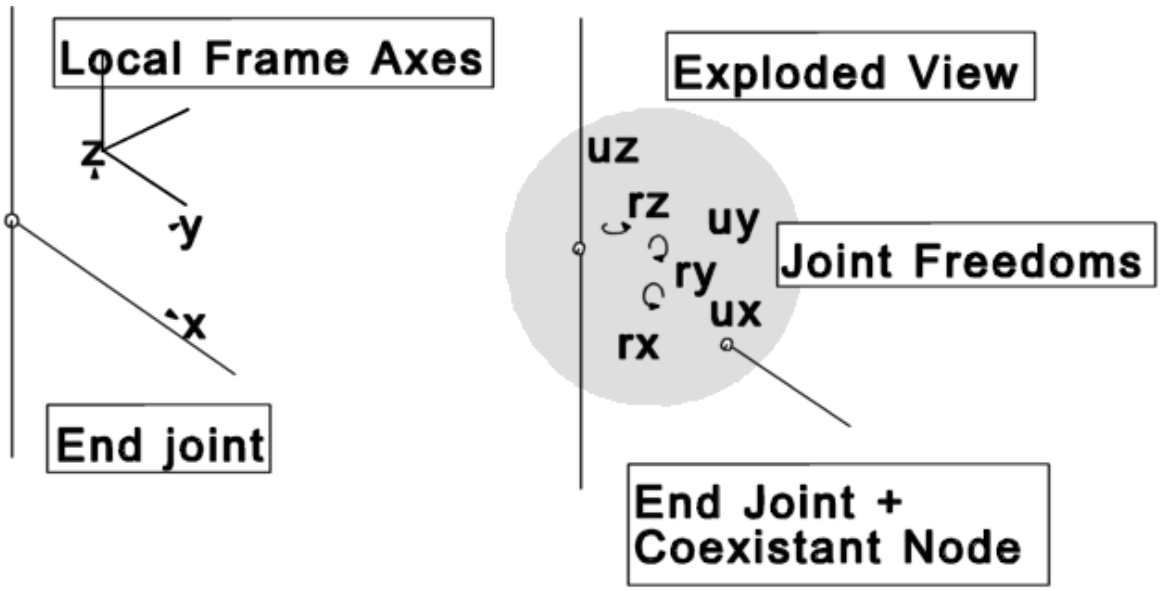
These rigid offsets are modelled in LS-DYNA as groups of rigid beam elements with small mass. Each rigid offset group will have its own component part. In this way the separate rigid offset groups can move independently. If the master node (i.e. the node at the centre of the rigid offset) has any kind of restraint associated with it, the rigid material definition associated with that particular rigid offset will also be restrained in a similar manner. However, these master node restraints must be in the global axis system.

The proximity check distance can be used as a model checking feature.

If any two nodes in the same rigid offset are further apart than the specified distance a warning will be printed in the dialogue box.

Please refer to the section on Rigid Links etc. for more details on how the rigid offsets are dealt with.

End Releases



End releases are used to ’release’ a selection of degrees of freedom at the end of a frame element. Note that if the frame

element has a rigid offset, the end release will occur at the point that the flexible region of the frame attaches to the rigid region (i.e. at the free end).

The translation of end releases into LS-DYNA will vary according to which translation option is selected in the DEFAULTS panel.

1. Simply ignore the end release definitions.
2. Use the 'simple' definition of end releases. This is a fairly crude translation, but utilises the fact that the majority of end releases are simple pins; i.e. all rotational freedoms (rx, ry & rz) are released leaving the translational freedoms fixed (ux, uy & uz). The simple end release definition separates the released joint into two coexistent nodes. These two nodes are then connected by a single translational spring with a stiffness defined on the DEFAULTS panel.

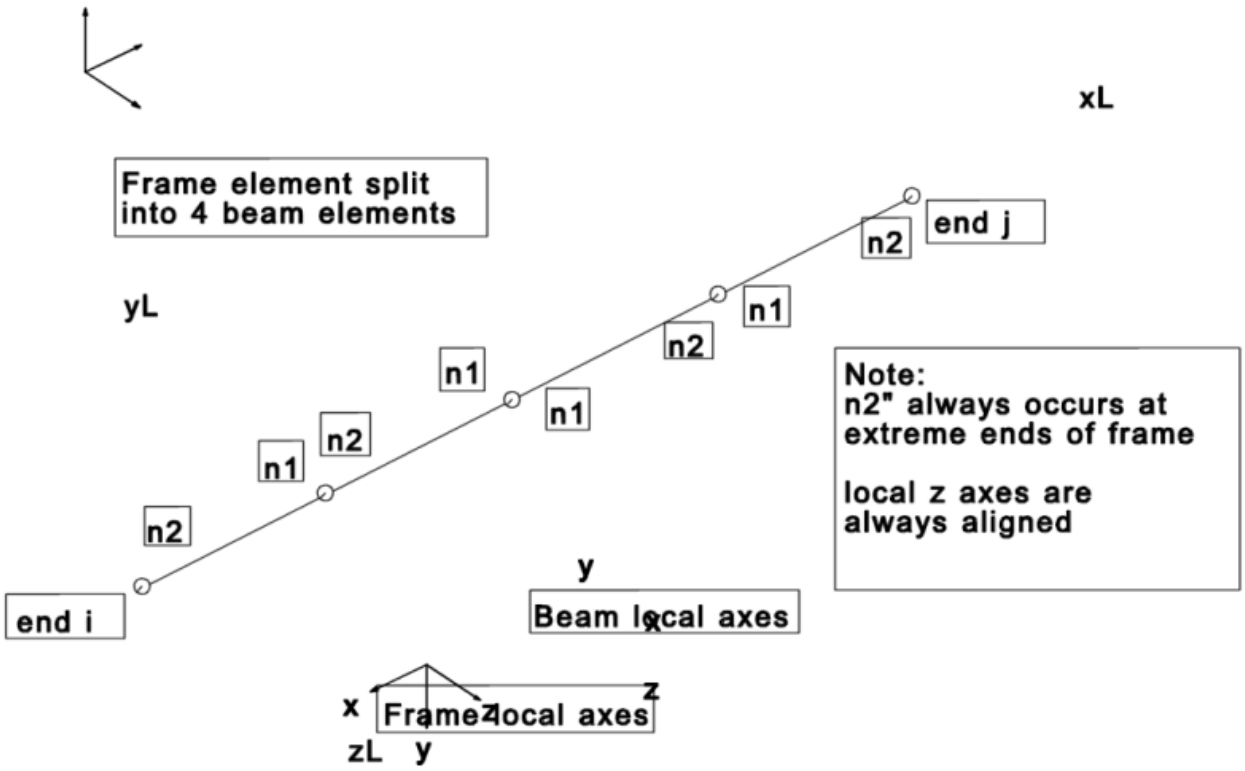
   The spring has no orientation vector and merely holds the two nodes together while allowing the two nodes to rotate freely relative to each other. This form of end release, while fast to translate and simple to understand, it does not read which freedoms have been released: it always assumes a pure pin.
3. The 'complex' end release definition is by far the most correct method, but it can become very slow and difficult to implement. With this in mind, it should not be used where there are a huge number of end releases to be translated in the model. The end release itself is a combination of linear constraint equations, nodal restraints and oriented discrete springs. The following table outlines what is used where and when:
   - ux, uy & uz are not released;
     The LS-DYNA `*CONSTRAINED_LINEAR` keyword is used to create a set of linear constraint equations between the two coexistent nodes linking all translational freedoms. Note that if either of the two nodes is included in any rigid part or an existing constraint a series of oriented springs will be used - see below.
   - rx, ry & rz are not released;
     As type (I) but for all rotational freedoms.
   - uy & uz are not released, ux is released;
     One translational spring defined to work in the plane normal to the axis of the beam.
   - ry & rz are not released, rx is released;
     One rotational spring defined to work in the plane normal to the axis of the beam.
   - rx is not released;
     A nodal restraint is used to prevent the node on the beam side of the pair of nodes from twisting. A coordinate system is created to ensure that the restrained freedom is aligned with the axis of the beam.
   - all other combinations;
     Translational or rotational springs using spring-damper orientation vectors are used to hold the non-released freedoms together.

Note that where oriented springs (using spring-damper orientation vectors) or the coordinate system in the fifth case are used, the program searches through all existing vectors or coordinate systems to locate a parallel vector or coordinate system. Only if a parallel vector / system cannot be found will a new vector / system be created. This searching for parallel vectors can take a considerable amount of time on large models.

Frame Splitting

It is often desirable to split a frame element into a number of beam elements. Where splitting occurs one of the beam elements within the frame will use the element number of the original frame element. Also the two extreme node numbers will be retained.

Frames can only be split into an even number of beams, and the mid node will always be equally spaced between the two end nodes even if different length end releases are specified for the frame.



The geometry of the split frame is shown above. As shown in the figure beams are oriented in such a way that the 'end 2' node of a beam always occurs at the extreme end of the frame. This is because seismic beams only develop plasticity at end 2. Hence if a seismic beam is split, it can develop a plastic hinge at both ends. The necessity of having end 2 nodes at the extreme ends of the frame requires the local beam coordinate system to be swapped halfway along the frame. Note that the local 'z' axis (generally the strong axis) is maintained in the same direction whilst the local 'y' axis is modified.

Defining which beams are to be split is done on a component by component basis. Each frame component can be split into a different number of beam elements. The basic choices are as follows:

1. Split into 'nseg' (or nearest even number), as defined for the frame element in the SAP2000 ascii deck.
2. Split into the default number of beams which is set on the SAP2000 main panel.

3. Split into an arbitrary number of beam elements as defined on the frame translations options panel.

A fourth option 'no split' can be specified. If this option is selected the frame component will not be split.

Non-prismatic sections which are split require an additional piece of information to be defined: what section properties are to be used. The choices are:

1. Use the section properties from end one for all beams.
2. Use the section properties from end two for all beams.
3. Interpolate the section properties along the length of the frame.

This option is a global parameter set for the whole model on the SAP2000 main panel.

Where the section data is interpolated from end one to end two, entirely new parts and sections are created. Clearly this has the potential for becoming very messy if the frame is split into too many beams each with their own section.

Frame Material Conversion

The basic translation converts all frame elements into linear elastic beams. However, two further options are available:

Seismic beams (only available for Belytschko-Schwer beam formulations)

A new `*MAT_SEISMIC_BEAM` material is generated and the plastic data calculated from the yield stress and the

section geometry data defined in the SAP2000 ascii deck. If no geometry data is present then default unit values will be used for the plastic data which must be modified prior to running the analysis.

Cables

Frame components flagged as cable elements will be turned into discrete spring elements. Currently the spring materials are linear elastic with stiffness as the product of Young's modulus and cross sectional area. Each element, however, has a scale factor on the stiffness of the inverse of the element length. Hence the element stiffness is EA/L. Note that cables cannot be split and that end releases are ignored as they have no meaning in this context.

Mass and Self Weight

Additional non-structural mass is distributed applied to the model as lumped mass elements at the free ends of the frame, and where the frame is split also at all mid-nodes.

A similar method is used to distribute the self-weight and additional non-structural self-weight. Point loads are created at the free-ends of the frame and, if split, at the mid-node. If the frame has been split then the point loads will be arranged such that the correct moment will be achieved at the free ends of the frame - refer to the loading section for more details on this arrangement.

Shell Elements & Sections

Shell Sections

The SAP2000 'Shell' and 'Membrane' formulation are identically translated into LS-DYNA. However the 'Plate' formulation does not exist. In this case the translator employs a standard shell formulation.

SAP2000 SHELL SECTION = LS-DYNA `*SECTION_SHELL`

Shell Elements

Shell elements are directly translated into LS-DYNA, although the topology for the element has to be modified. The same element number is used in LS-DYNA as for SAP2000.

SAP2000 SHELL ELEMENT = LS-DYNA `*ELEMENT_SHELL`

Self Weight

The weight associated with each node on the shell element is estimated, and applied to the element as a series of point loads in the negative z-direction.

Solid Elements & Sections

Solid Sections

SAP2000 does not define any section properties for solid elements. Instead all the 'section' data is lifted directly from the material definition. LS-DYNA requires that these sections are defined.

SAP2000 [SOLID SECTION] = LS-DYNA `*SECTION_SOLID`

Solid Elements

Solid elements are translated directly into LS-DYNA, although the topology of the element has to be modified. The shorthand definition of the solid element is also supported by the translator.

SAP2000 SOLID ELEMENT = LS-DYNA `*ELEMENT_SOLID`

Self Weight

The weight associated with each node on the solid element is estimated, and applied to the element as a series of point

loads in the negative z-direction.

Nllink Elements & Nlprop Data

Nonlinear properties (Nlprop)

The nonlinear properties are defined in terms of shear properties are non-shear properties. The various property types and their LS-DYNA counterparts are list below:

SAP2000 NLPROP (no type) = LS-DYNA `*MAT_SPRING_ELASTIC` (stiffness = ke, effective elastic stiffness from SAP2000 definition) (shear & non-shear)

SAP2000 NLPROP (damper) = LS-DYNA `*MAT_SPRING_ELASTIC` (stiffness = ke, effective elastic stiffness from SAP2000 definition) & `*MAT_DAMPER_VISCOUS` (damping coefficient = c, from SAP2000 definition) (shear & non-shear)

SAP2000 NLPROP (gap & hook) = LS-DYNA `*MAT_SPRING_ELASTIC` with clearance defined on the discrete section card (stiffness = ke, effective elastic stiffness from SAP2000 definition) (shear & non-shear)

SAP2000 NLPROP (plastic1) = LS-DYNA `*MAT_SPRING_ELASTIC` & `*MAT_SPRING_ELASTOPLASTIC` (These two springs combine to give a kinematic hardening hysteresis response. The elastic spring controls the post-yield stiffness and the elastoplastic spring defines the yield strength. The pre-yield strength of the elastoplastic spring when added to the elastic spring gives the correct pre-yield stiffness, ke. The post-yield stiffness of the elastoplastic spring is zero.) (shear and non-shear)

SAP2000 NLPROP (isolator1) = LS-DYNA `*MAT_SPRING_ELASTIC` (stiffness = ke, effective elastic stiffness from SAP2000 definition) (non-shear) LS-DYNA `*MAT_SPRING_ELASTIC` & `*MAT_SPRING_ELASTOPLASTIC` (as for 'plastic1' type) (shear)

SAP2000 NLPROP (isolator2) cannot be translated

Nonlinear Link Elements

A single nllink element will be translated into a collection of discrete elements using spring-damper orientation vectors to imitate the local coordinate system to the nllink element.

The first element discrete element defined will use the same element number as the nllink element.

Where zero length single node elements have been defined in SAP2000, the translator will generate a second node and then fully restrain it.

SAP2000 NLLINK = LS-DYNA `*ELEMENT_DISCRETE` (collection of ...)

Mass, Inertia & Self Weight

The mass associated with nllink elements can be defined for each translational direction. This is not possible in LS-DYNA, so the average mass is used and distributed evenly between the two nodes defining the nllink element.

The inertia of the nllink element is converted into a global inertia tensor and then evenly distributed between the two nodes defining the nllink element.

The self weight of the nllink element is distributed evenly between the two nodes and represented as point loads.

Loading

Each load case defined in the SAP2000 model can be individually selected or deselected. Each selected load case has a scale factor applied to it (default = 1.0) for the translation. The translator will then take all load cases that have been selected and apply them to the model. A global scale factor is also defined which is applied to every load in the model on top of the individual scale factor discussed previously. Note that each load case that is selected will be associated with its own load curve. This enables the scale factor on each load case to be changed at a later date.

The self weight load case is a special case. The self weight loading can either be applied to the model as a series of point loads which have been defined according to the self weight of each element in the model (which may or may not be equivalent to the product of mass and gravitational acceleration), or as a acceleration body load to the whole structure (`*LOAD_BODY_Z`). If the later is the case, the scale factor on the self weight load case should be the

gravitational acceleration parameter (i.e. 9.81 m/s2). The point loads should need no additional scale factor as these will already be defined in terms of structure weight.

Joint Forces

These are simple point loads. Each point load defined will be split into a global load vector comprising of up to three point loads, each parallel with a global axis.

SAP2000 JOINT FORCE = LS-DYNA `*LOAD_NODE_POINT`

Joint Displacement

These are nodal displacement boundary conditions. Each displacement boundary condition is split into a global load vector, each non-zero component of the vector having a separate displacement boundary condition.

SAP2000 JOINT DISPLACEMENT = LS-DYNA `*BOUNDARY_PRESCRIBED_MOTION_NODE`

Spring Displacement

As for Joint Displacement, but for the grounded node of a spring element.

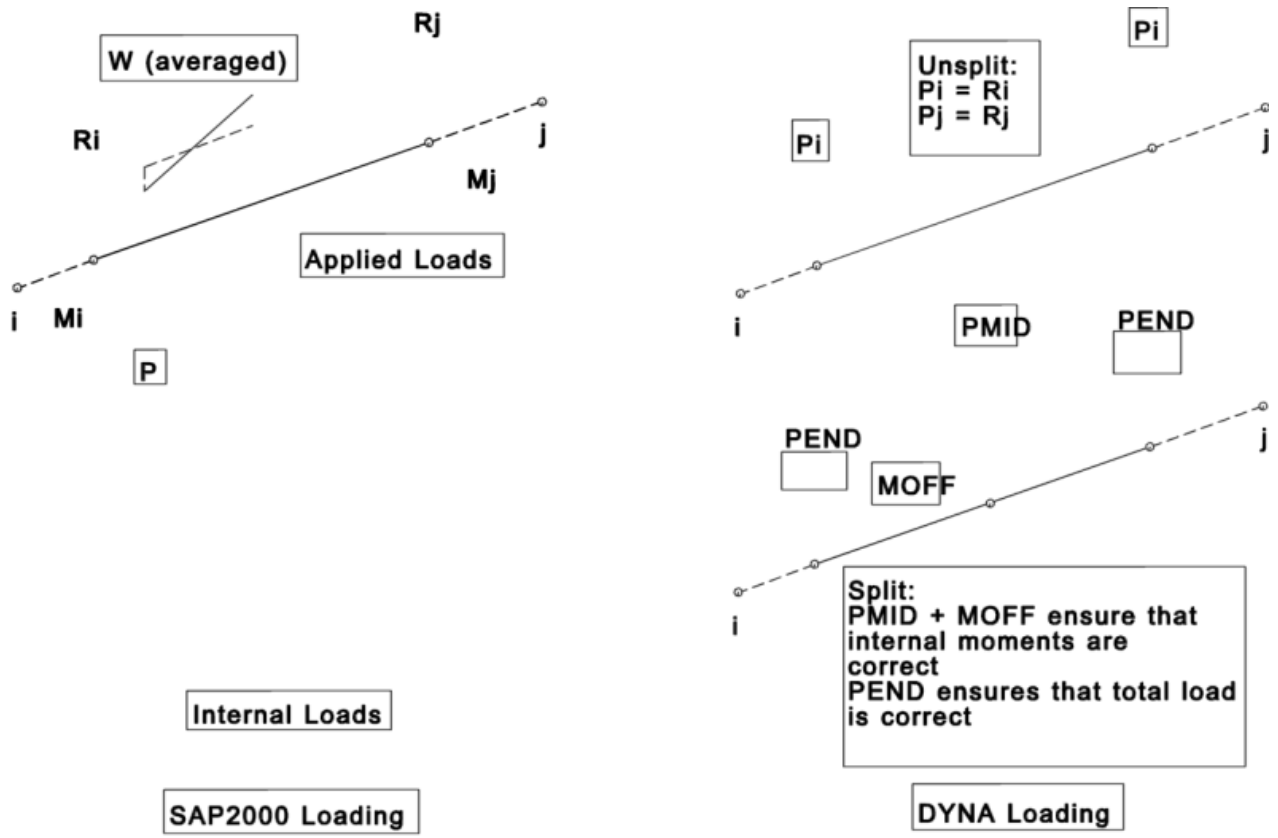
SAP2000 SPRING DISPLACEMENT = LS-DYNA `*BOUNDARY_PRESCRIBED_MOTION_NODE`

Concentrated & Distributed Span Loads

All frame loading types are supported by the translator, converting the applied load into a global load vector. Each non-zero component of the load vector is then split into a collection of point loads.

If the frame element to which the load is applied has not been split, two point loads are applied to the free ends of the frame. The point loads are equal and opposite to the reaction force the beam would create if it were fully fixed.

If the frame element has been split then the loading arrangement attempts to ensure that the correct moment is achieved at the free ends of the frame. A point load and moment are applied to the mid-node of the frame which create the same end moment conditions as the original loading condition. Two additional point loads are used to balance up the total load on the frame.



SAP2000 SPAN LOADING = LS-DYNA `*LOAD_NODE_POINT`

Uniform Shell Loading

All forms of uniform shell loading are translated. The applied load is split into a global load vector. The area associated with each node on the shell element is estimated and for each non-zero component, a point load is applied to each node on the element.

SAP2000 UNIFORM SHELL LOADING = LS-DYNA `*LOAD_NODE_POINT`

Surface Loading

The surface loading for shells (faces and edges) and solids (faces) is applied normal to the face of the element. This load is then converted into a global load vector and a point load applied to each node on the face / edge for each non-zero component.

SAP2000 SURFACE LOADING = LS-DYNA `*LOAD_NODE_POINT`

Dynamic Relaxation Loading

If dynamic relaxation is required then an additional set of loads will be created for dynamic relaxation only, except for any displacement loads. The load curve used to apply dynamic relaxation loads will ramp the loading from zero to the full working load. This ramping period is used to prevent dynamic shock to the model caused by instantaneous application of loading.

Rigid Links and Rigid Bodies

A node in LS-DYNA cannot be a member of more than one rigid body or constraint system. If such a case did occur, LS-DYNA would fail to initialise. To avoid such a problem the translator performs a check to ensure that no nodes occur in more than one nodal rigid body or offset. If a clash is detected, then the two rigid groups of which the offending node is a member will merged into one new rigid group.

IGES file format

The International Graphics Exchange Specification (IGES) is a standard format for model geometry. Primer will read fixed ASCII file format 5.3 IGES files.

The following IGES entity types are read by PRIMER.

Entity type	Entity description	Notes
100	Arc	
102	Composite curve	
104	Conic arc	Forms 2 (hyperbola) and 3 (parabola) not supported.
106	Copius data/Linear path/Simple closed planar curve	Forms 1, 2, 11, 12 and 63 supported
108	Plane	
110	Line	Forms 1 (semi bounded) and 2 (unbounded) not supported
112	Parametric spline curve	
116	Point	
118	Ruled surface	
120	Surface of revolution	
122	Tabulated cylinder (extruded surface)	
124	Transformation matrix	
126	Rational B-Spline curve	
128	Rational B-Spline surface	
141	Boundary	
142	Curve on parametric surface	
143	Bounded surface	
144	Trimmed (parametric) surface	
308	Subfigure definition	
314	Colour	
406	Property	Only forms 3 (level function) and 15 (Name) supported
408	Singular subfigure instance	

All other entity types are ignored.

APPENDIX VII: Format translation during **MODEL**> **WRITE**

ABAQUS "Input" file output.

The following table shows the limited set of DYNA keywords that are supported for conversion to Abaqus .inp file format. The translator leaves the internal LSDYNA data unchanged.

Internal LS-DYNA keyword	**Abaqus written output**	**Note**
*CONSTRAINED_INTERPOLATION	*ELEMENT,TYPE=DCOUP3D, *ELSET= **KINEMATIC COPULING** or **DISTRIBUTING COPULING**	
*CONSTRAINED_NODAL_ RIGID_BODY	*MPC	[1]
*CONTACT_NODES_TO_SURFACE	*CONTACT_PAIR *SURFACE *SURFACE_INTERACTION	
*ELEMENT_BEAM	*ELEMENT	[2]
*ELEMENT_SHELL	*ELEMENT	[3]
*ELEMENT_SOLID	*ELEMENT	[4]
*MAT_ELASTIC	*MATERIAL *DENSITY *ELASTIC	
*MAT_PIECEWISE_LINEAR_PLASTICITY	*PLASTIC	
*NODE	*NODE	
*SECTION_BEAM (ELFORM = 1)	*BEAM SECTION	[5]
*SECTION_BEAM (ELFORM = 12)	*BEAM GENERAL SECTION	[6]
*SECTION_SHELL	*SHELL SECTION	
*SECTION_SOLID	*SOLID SECTION	
*SET_NODE	*NSET	
*SET_SHELL	*ELSET	
*SET_SOLID	*ELSET	
*TITLE	*HEADING	

Notes:

1. *CONSTRAINED_NODAL_RIGID_BODY cards are converted to *MPC of **BEAM** type only.
2. *ELEMENT_BEAM crads are converted to *ELEMENT with **Type B31** only.
3. **Three noded shells** are written as *ELEMENT, **TYPE=S3R** while **four noded shells** are written as *ELEMENT, **TYPE=S4.**
4. Four noded, six noded and eight noded solids are converted to types C3D4, C3D6 and C3D8 respectively.
5. All *SECTION_BEAM cards with elform =1 are converted to ***BEAM SECTION with SECTION=CIRC**.
6. All *SECTION_BEAM cards with elform =12 are converted to ***BEAM GENERAL SECTION with SECTION=GENERAL**.

IDEAS (Master Series) Universal file format.

This section shows which LS-DYNA keywords are supported, and how they are translated to entities in an I-DEAS universal file.

Primer uses the same conventions as N/CODE for the universal file so N/CODE can be used to re-translate the universal file back into a keyword deck.

Primer carries out several checks when writing a universal file to ensure that there are no problems when importing the universal file into I-DEAS.

These are:

- I-DEAS needs unique labels for every element. During output of the universal file Primer will check and fix any element clashes which occur, printing a warning about what has been renumbered. If this happens the model in Primer is renumbered, not just the output in the universal file. If you do not want the elements renumbered in your original LS-DYNA deck then ensure the keyword deck is saved before writing the universal file.

- I-DEAS does not allow Physical names longer than 40 characters in the universal file and does not allow duplicate names. As LS-DYNA allows part names up to 72 characters long there is a potential problem with names being duplicated. Primer truncates any part names in the LS-DYNA deck to 36 characters and then checks to see if any duplicate part names exist. If there are duplicates then '_1' is added to the first duplicate, '_2' to the second etc. to ensure names are unique.

- Node sets in I-DEAS (used for rivets, spotwelds, constrained node sets and generalised welds) need an independent node and at least one dependent node. Primer checks to see if any have been defined that use less than 2 nodes. If any have a warning is displayed and the node set is not translated.

- Springs in I-DEAS master series do not allow a material to be associated with them. To deal with this when translating a universal file to a LS-DYNA keyword deck N/CODE attempts to create a material with the same id as the physical id. This can cause problems so when Primer creates a universal file it checks any parts which contain discrete elements. If a material already exists with the same id as the discrete part then the part is renumbered so that N/CODE will not encounter any problems. This renumbering will not occur if the discrete part refers to a material with the same id.

The universal file import in Primer is very basic. At present only nodes and elements are read. To read data from a universal file N/CODE should be used to create an LS-DYNA keyword deck. This can then be read into Primer.

Supported keywords

*Materials. Structural and thermal material tables are written to the universal file (module 773). In addition to the structural materials dummy materials are written for lumped masses, sliprings, retractors, joints, stonewall plates, contact surfaces and airbag segments. Materials for conventional elements get written as isotropic materials with null properties. Materials for springs get written out as null materials with no properties.

*
Physical property tables (module 772 for I-DEAS V, module 778 for I-DEAS VI). Solid, beam, shell, thick shell, seat belt properties are written. The property number in I-DEAS is created from the part number in the LS-DYNA keyword deck. Dummy physical properties are written for lumped mass, slip rings, retractors, joints, extra nodes on rigid bodies, stonewalls, contacts and airbags

*
Beam cross section properties (module 776) are written for rectangular or circular section Hughes-Liu beams. A, I_{yy}, I_{zz} and J are written for Belytschko-Schwer beams

*
Nodes

*
Solid, beam, shell, thick shell, discrete and lumped mass elements are all translated (module 780)

*
Seat belts, retractors and sliprings are translated as green, read and blue rod elements (module 780) consistent with N/CODE

* Contact surfaces defined by either parts or segments are translated (module 2417). Groups are created in I-DEAS called CONTACT_SLAVE_n and CONTACT_MASTER_n. If the contact is a nodes to surface contact the slave group is called CONTACT_NODES_n or if the contact is a single surface algorithm a single group is created called CONTACT_SINGLE_n. Segments are translated as red and blue plate elements (slave and master). For contacts defined by parts one element from each part is placed in the group. This is consistent with N/CODE
* Extra nodes on rigid bodies are translated as pipe elements connecting the extra node to a node on the first element found in the rigid part. (module 780)
* Node and element time history blocks are translated (module 2417) by creating a group in I-DEAS called NODE_BLOCK or ELEMENT_BLOCK containing the history entities
* Restraints (**`*BOUNDARY_SPC`**) (module 755)
* Constraints (**`*CONSTRAINED_SPOTWELD, *CONSTRAINED_RIVET, *CONSTRAINED_NODE_SET, *CONSTRAINED_GENERALIZED_WELD_SPOT`**) (module 754)
* Point loads (module 782)
* Displacement, velocity and acceleration boundary conditions on nodes (module 2417). A group in I-DEAS is created called BC_DISP_n:(dof i), BC_VELO... or BC_ACCE... where n is a unique group number and i is the degree of freedom (1 to 6). Boundary conditions on rigid bodies cannot be written
* Nodal force groups (**`*DATABASE_NODAL_FORCE_GROUP`**) are written (module 2417) by creating a group in I-DEAS called REACTION
* Nodal rigid bodies are written (module 2417) by creating a group in I_DEAS called NODAL_RIGID n
* Joints (module 780)
* Initial velocities (module 2417) are written by creating I-DEAS groups called VELOCITY x y z, where x, y and z are the components of the initial velocity
* Rigidwalls are written (module 2417) by creating a group in I-DEAS called RIGIDWALL n. All the slave nodes for the rigid wall are placed in this group. If the rigidwall is finite in size a yellow plate element is created which represents the extent of the stonewall and this is also placed in the group. Only planar rigidwalls are written.
* Airbags are written (module 2417) by creating a group in I-DEAS called AIRBAG n. If the airbag is defined by segments they are translated as cyan plate elements. If it is created by parts one element from each part is placed in the group (compatible with N/CODE).

For further details on how LS-DYNA entities can be created in I-DEAS and how they are represented in a universal file see the N/CODE user manual.

Patran "Neutral" file output

Patran "neutral" (.ntl) file data can be written for both level 2.5 and level 3 versions.

As with universal file output elements of different types cannot share the same label, so prior to output a check is made for clashes and element labels are renumbered if required.The following data are output:

* **Analysis title** is translated directly as module 25

* **Nodes** are translated directly as module 1 (extra nodes are generated for grounded springs).

* **Elements** are translated as follows:

Dyna element type	**Level 2.5 output**	**Level 3 output**
8 noded solids (hexa) 8 noded thick shells (quad) 6 noded solids (wedge) 6 noded thick shells (tria) 4 noded solids (tetra)	<iv> = 8, config = 0 <iv> = 7, config = 0 <iv> = 5, config = 0	<iv> = 8, config = 4 <iv> = 8, config = 1 <iv> = 7, config = 4 <iv> = 7, config = 1 <iv> = 5, config = 4
4 noded shells (quad) 3 noded shells (tria)	<iv> = 4, config = 0 <iv> = 3, config = 0	<iv> = 4, config = 3 <iv> = 3, config = 3
Beams	<iv> = 2, config = 0, 1, 2 (= elform)	
Translational spring Rotational spring Translational damper Rotational damper	<iv> = 2, config = 10 <iv> = 2, config = 11 <iv> = 2, config = 20 <iv> = 2, config = 21	
Seatbelt element	<iv> = 2, config = 30	
Retractor	<iv> = 2, config = 31	
Slipring	<iv> = 2, config = 32	
Lumped mass	<iv> = 2, config = 7	

* **Structural materials** are translated as module 3:

  type = 1 for deformable materials.
  type = 2 for rigid ones in level 2.5 format, type = 1 in level 3 format.

  Thermal materials are always translated as type = 1.

  All material data fields are set to zero. "Fake" (empty) materials are also generated for those element types (eg lumped masses) that don’t have materials in Dyna, but do in Patran.

* **Part and section data** are written out as Patran "properties", packet type 4. The Dyna part id becomes the property id, and other data are generated as follows:

Element type	Common data	Level 2.5	Level 3.0
Solids:	<shape> = 8 <nvals> = 1 (material label)	config = 0	config = 4
Shells	<shape> = 4 <nvals> = 5 (matl label, t1... t4)	config = 0	config = 3
Thick Shell	<shape> = 8 <nvals> = 1 (material label)	config = 1	
Beams	<shape = 2> <nvals> = 5 (matl label, ts1 .. tt2)	config = 0	
Discrete	<shape> = 2 <nvals> = 1 (material label)	config = 20 (springs) config = 21 (dampers)	
Seatbelt	<shape> = 2 <nvals> = 1 (material label)	config = 30	

Where materials are not defined, for example in "latent" parts, "fake" (empty) material ids are generated and specified.

In addition a single property for each of the following element types is created, if they exist in the dyna model. Again "fake" (empty) materials are also generated for these:

Retractors	<shape> = 2, config = 31
Sliprings	<shape> = 2, config = 32
Lumped masses	<shape> = 2, config = 7

NASTRAN output

The following table shows the limited set of DYNA keywords that are supported for conversion to a Nastran bulk data file. The translator leaves the internal LSDYNA data unchanged.
For property cards $HMNAME comments are written, as are the $HMMOVE comments which maintain the PBAR collectors.
See special note on rigid parts.

Internal LS-DYNA keyword	Nastran written output	Note
*BOUNDARY_SPC_NODE	SPC	[2]
*BOUNDARY_SPC_SET	SPC1	[3]
*CONSTRAINED_INTERPOLATION	RBE3	
*CONSTRAINED_NODAL_ RIGID_BODY	RBE2	
*CONSTRAINED_SPOTWELD	RBE2 or Solid welds	[10]
*DEFINE_CORDINATE_NODES *DEFINE_COORDINATE_xxx (except _NODES)	CORD1R CORD2R	
*DEFINE_SD_ORIENT	CORD2R	[1]
*ELEMENT_BEAM	CBAR	[8]
*ELEMENT_DISCRETE	CELAS1, CDAMP1	[9]

*ELEMENT_MASS *ELEMENT_INERTIA	CONM2	
*ELEMENT_PLOTEL	PLOTEL	
*ELEMENT_SHELL	CQUAD4, CQUAD8, CTRIA3, CTRIA6	[7]
*ELEMENT_SOLID	CHEXA, CTETRA, CPENTA	
*INCLUDE	INCLUDE	
*LOAD_NODE	FORCE	
*LOAD_SHELL	PLOAD	
*LOAD_THERMAL_CONSTANT_NODE	TEMP	
*MAT (all structural exc. discrete, composite)	MAT1	
*MAT_ENHANCED_COMPOSITE_DAMAGE	MAT8	
*NODE	GRID	
*PART_COMPOSITE *MAT_54 (with *INTEGRATION)	PCOMP	
*SECTION_BEAM (exc. type 6)	PBAR, PBARL	[5]
*SECTION_DISCRETE	PELAS, PDAMP	[6]
*SECTION_SHELL	PSHELL	[4]
*SECTION_SOLID	PSOLID	
*TITLE	TITLE =	

Notes:

1. To avoid label clashes these will be labelled at the *DEFINE_SD_ORIENT label offset with the highest *DEFINE_COORDINATE label.
2. Local coordinate systems on SPCs are maintained.
3. *BOUNDARY_SPC_SET cards defined in a local coordinate system are automatically written out as SPC cards instead of SPC1 cards.
4. Thickness on the PSHELL card will be taken as the T1 value on the *SECTION_SHELL card. Shells of variant thickness must be set up on the *ELEMENT_SHELL card.
5. For DYNA integrated beams the values of A,I1,I2,J and shear factor (if SHRF=0) are calculated from the section defined.
6. Values of k and dc are read from *MAT_SPRING_ELASTIC and *MAT_DAMPER_ VISCOUS. For other discrete materials the illegal field ’xxxxxxx’ will be written to the Nastran file.
7. For shells _THICKNESS and _BETA attributes are translated directly.
8. DYNA beam release codes, orientation vectors and global offset vectors are translated.
9. For a discrete element or set of coincident discrete elements, new RBE2s are created at each end in order to convert the discretes to zero length (if necessary) and to generate nodes to which the CORD2Rs may be referenced. All nastran elements will reference component number 1 (translational) or 4 (rotational). If no *DEFINE_SD_ORIENT vector exists for a non-zero length discrete, the CORD2R will be created from the two nodes.
10. *CONSTRAINED_SPOTWELD cards are translated to NASTRAN RBE2 cards if option "Convert spotweld beams to hexa" is not set prior to the write operation. Also see section Conversion of spotweld beams below.
11. In PRIMER release 9.3 and later the option of "SMALL" and "WIDE" format output is provided. Note that all GRID (node) cards are written out in the "WIDE" format even if the user opts to write the deck out in the "SMALL" format. This is done in order to preserve the precision of nodal coordinates. When the "WIDE" option is chosen by the user, only those cards which contain floating point fields are written out in the "WIDE" format. Cards containing only integer data are written out in the "SMALL" format always.

Special note on rigid parts:
Rigid parts will be represented by the generation of an RBE2 over the nodes of the part itself, any parts slaved to it and any *CONSTRAINED_EXTRA_NODES.

An additional node, created at the centre of mass, serves as the independent node and will carry an SPC if there is any

DYNA material constraint (CMO.ne.0).
If the part is of type *PART_INERTIA a CONM2 element will be added at the defined centre of mass and element printing will be suppressed.
Local coordinate systems for the PART_INERTIA (IRCS=1) and for DYNA material constraint(CMO=-1) are treated appropriately.

Conversion of spotweld beams:
If the option **Convert spotweld beams to hexa** is activel, existing LS-Dyna beam type spotwelds will be written out as Nastran solid weld elements. Note that the beams need to be created and correctly projected in the Dyna model before exporting the Nastran file.

The area of the NASTRAN spotweld is the same as the area of the DYNA beam.

Each LS-Dyna beam will be converted to a Nastran CHEXA solid element of equivalent cross-sectional area, the nodes of which are tied using RBE3 elements to the appropriate shells. If a solid node lies directly over a shell node an RBE2 is used instead as Nastran does not permit RBE3s with zero weighting. Solid elements which are too close to free edges or facets which exceed 30 degrees will be moved inboard to enable the nodes to be tied on.

Typical shell edge length value is set to 10.0 which is appropriate for typical crash models with units in mm. If Primer reports that it "*Could not generate solid elements for n spotweld beams*" this value may be increased to capture more of the welds.

If any spotwelds fail to convert, a set of failed beams is created.

Conversion of tied contacts:
If the option **Convert tied contact to RBE3** is active, CONTACT_TIED_NODES_TO_SURFACE in LS-Dyna will be translated for Nastran output.

The conversion process assumes that a set of slave nodes (defined by node set/part/part-set) is to be tied to a set of master shells (defined by part/part set/shell set).

The Primer contact checker will be used to determine which nodes actually tie in the LS-Dyna model. **Only these nodes** will be tied in the Nastran output.

In default mode, each slave node (this is typically a node on a solid) will be tied as a dependent using an RBE3 element to the nodes of the corresponding master shell. However, Nastran does not appear to permit zero weighting in RBE3, so if the nodes coincide (or project directly onto one another) an RBE2 will be made.

Use RBE2 to project. This option may be used if the slave nodes are excessively distant from their master shells. It will generate an RBE2 on the slave node which projects to the master shell.

On completion of the contact a node set will be created called "*Slave nodes tied. Contact n*".

Conversion of MIG Welds to RBE3:
If the option **Convert MIG Weld to RBE3** is active, MIG Weld beams in LS-Dyna will be translated to CBAR/CBUSH elements with RBE3 elements at both ends.

The node at the meshed-in end of the MIG weld beam will be offset along the beam axis as specified in **"Node offset %"** and then RB3'd back to shells attached with original node.

Convert MIG beam to CBUSH. This option may be used to convert MIG weld beams (spotwelds of type MIG) to CBUSH elements, else converted to CBAR elements by default.

D3PLOT/PTF file output

This file format is available for users who wish to visualise their model in a post-processor without having to initialise it in LS-DYNA. No data conversion is required since this file is "native" LS-DYNA format, however the following explains the output options:

Write elements and Nodes

This writes out the geometry and topology of nodes, solids, beams, shells, thick shells and SPH elements. It would be possible to write more recent and exotic d3plot file contents, such as Airbag particles, DES elements, etc, but these are complex to write and are not likely to be visualised in this way.

Write Geometry Surfaces as Shells

If your model contains "geometric" items, typically imported from an IGES file, the PRIMER will render these as surfaces, lines and points, however they have no existence as "nodes or elements" and will not normally be written to the d3plot file. If you select this option then the tesselated triangles used to display surfaces will be added to the d3plot file as triangular shells, and their vertex points will be added as nodes.

These elements and nodes are given sythesised labels, starting from the highest "real" item + 1.

Write state 1 (undeformed data)

If this item is chosen then an initial "state" of "results" will also be generated, similar to the result of initialising the deck in LS-DYNA and obtaining the results at the first cycle. This will contain:

Nodal coordinates	Based on the existing undeformed geometry
Nodal velocities (if available)	If any INITIAL cards specify nodal velocities then the initial velocities of all nodes will be written, otherwise this block is omitted.
Nodal temperatures (if available)	If any INITIAL_TEMPERATURE cards specify nodal temperatures, otehrwise omitted
Shell thicknesses	Taken from SECTION_SHELL, PART_COMPOSITE or ELEMENT_SHELL cards
Solid, shell and thick shell initial stresses and plastic strains (if available)	Taken from *INITIAL_STRESS_xxx cards, otherwise omitted
Solid, shell and thick shell initial strains (if available)	Taken from *INITIAL_STRAIN_xxx cards, otherwise omitted

Cognoscenti of the d3plot format will know that its shell output is switchable, and also that the number of integration points of data written for shells can be varied via MAXINT and other fields on the *DATABASE_EXTENT_BINARY card. In order to avoid writing large files entirely full of zeros these settings are ignored and instead:

- Shell stresses and plastic strains are only output if there are one or more *INITIAL_STRESS_SHELL or _TSHELL cards
- The number of shell integration points written out is the greater of 1 and the largest NTHICK value on any *INITIAL_STRESS_(T)SHELL card
- Shell strain tensors are only written out if there are one or more *INITIAL_STRAIN_(T)SHELL cards, and then only for inner and outer integration points.
- Solid strain tensors are only written out if there are one or more *INITIAL_STRAIN_SOLID cards

The order of output of per-integration point data. *Note: not the same as LS-DYNA*

The aforementioned cognoscenti will also be aware that the order in which shell through-thickness data are written depends on the MAXINT value, with special rules applying where MAXINT = 3, and also in some cases where MAXINT = 5.

This file totally ignores those rules and writes initial stress and plastic strain data in the order in which it appears on the *INITIAL_STRESS_(T)SHELL card, also ignoring any T value specifying the parametric coordinate through the shell. So if NTHICK = n (and NPLANES = 1) then layer 1 will be the first row of data, layer 2 the second, and so on for any value of NTHICK.up to layer n. This includes the special cases of n = 3 and n = 5.

In addition where multiple in-plane integration points have been specified, NPLANE > 1, the file assumes the first NPLANE rows on the *INITIAL_STRESS_(T)SHELL cards are the on plan values for the first through thickness integration point, the next NPLANE rows are values for the next through thickness point and so on...

In the case of no initial stress or strain data then MAXINT will be set to 1 and the shell element stress tensor, plastic strain and strain tensor output flags will be turned off in order to minimise the size of the file. There is little virtue in writing a file full of millions of redundant zeros!!

The "results" in state 1 in this file are not identical to the results produced by initialising the input deck in LS-DYNA, and are not intended to be.

Please make sure that you understand the differences, in particular with respect to the shell stress and strain output. If you need a "proper" set of results then you need to write out the keyword file and initialise it in LS-DYNA in the normal way.

Add space for on plan int point results

If the file doesn’t contain any *INITIAL_STRESS_(T)SHELL cards the file will only contain one data slot for stress data per (T)SHELL. Users may want to create a JavaScript user defined UBIN component in D3PLOT to add values to the model and this limitation means they can only write it at one layer and one on plan point.

Selecting the option to add space for on plan integration point results will make PRIMER write a file with data slots available for multiple on plan points (the number of layers will still be limited to 1).

NOTE: The ELFORM field on the *SECTION card needs to be set to a formulation that supports multiple on plan integration points otherwise D3PLOT will still think it only has space for one on plan point.

APPENDIX VIII: "Curve" file formats

(1) T/HIS curve file format.

This section describes the format of a T/HIS "curve" file.

Line 1 Title
Line 2 X axis label
Line 3 Y axis label
Line 4 Curve label
Line 5 X, Y point 1
Line 6 X, Y point 2
...
Line n+4 X, Y point n

The X and Y values can be in any format as long as the two values are separated by a space or a comma.

A comment line may be included anywhere in the file by starting the line with a ’$’.

Several curves can be put in one file sequentially, separated by the word CONTINUE.

The title and three label lines must be present for each curve.

(2) "raw" x,y file format/CSV format.

This section describes the format of a "raw" x,y data file.

A raw x,y data file contains lines which have a pair of x,y data points on them.

Line 1X, Y point 1
Line 2X, Y point 2
...
Line nX, Y point n
The X and Y values can be in any format as long as the two values are separated by a space, a comma or a tab.

A comment line may be included anywhere in the file by starting the line with a ’$’.

APPENDIX IX: Primer database format

Database files in Primer are a powerful way of organising and retrieving data. For databases to function correctly in Primer two types of files are required.

‘oa_database’ files

‘oa_index’ files

oa_database files

Primer looks for databases by looking for the existence of a file called ‘oa_database’ in the directories $OASYS (the directory where the Oasys Ltd. LS-DYNA environment is installed) and $HOME (the home directory of the user. In the instance where an ‘oa_database’ file is found in both these directories then the entries from both files are included.

The databases in the file from the installation directory $OASYS are available to every user. If the file was in $HOME then the databases would only be available to that user.

Any line in the file which begins with a $ (dollar) is treated as a comment.

Each line in a ‘oa_database’ file refers to a database (file or directory) and contains 3 fields separated by a space, a comma or a tab character.:-

1. **Database type**. This indicates what type of entities are in this database. **e.g. MODEL* LCUR* MATL***. The database type must end with an asterisk (*). Only databases which are relevant are displayed in each context e.g. material import will only offer MATL* databases, model build will only offer MODEL*
2. **A directory or filename for MODEL***. For LCUR* and MATL* the directory contains the database information. This directory must also contain an oa_index file which explains how the info is to be used (see below)
3. **A name**. The database names will be used in windows when selecting a database.

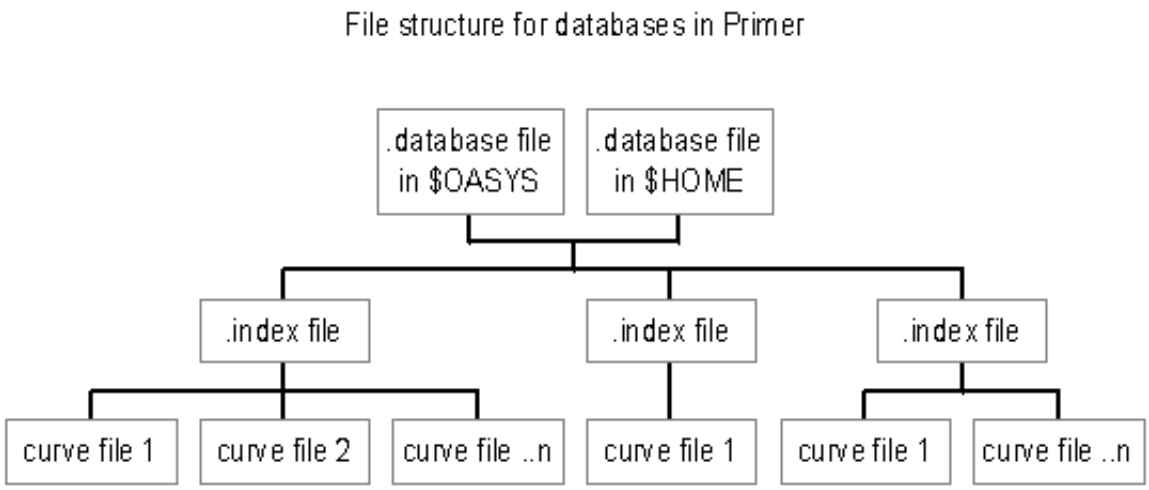
An example of a ‘database’ file taken from an $OASYS directory is given below containing 2 LCUR and 2 MATL databases.

```
$ database file for Primer in $OASYS
$
$ Any databases which are defined in here will be available to all users
$=====================================================================
$
$ type directory to find oa_index file in   database name
$ ==== ==================================   =============
LCUR*  /disk/database/loadcurve/seismic     Seismic loadcurve database
LCUR*  /disk/database/loadcurve/material    Material loadcurve database
MATL*  /disk/database/material/steel        Steel Database
MATL*  /disk/database/material/aluminium    Aluminimum Database
MODEL* /disk/database/proj1.dba             Model Build Proj1
```

oa_index files

An oa_index file contains information on how a database is formatted. One index file called ‘oa_index’ is needed for each database entry in the ‘database’ file and it must be present in the directory specified for each database.

The oa_index file then describes this database and gives the filenames of all the entries in this database. For example the LCUR* database ‘Seismic loadcurve database’ in the above database file points to the directory /disk/database/loadcurve/seismic. An oa_index file must be present in this directory to describe how the database ‘Seismic loadcurve database’ is formatted and how many entries there are. This is summarised in figure A9.1.



Any line in the file which begins with a $ (dollar) is treated as a comment. The first uncommented line in an oa_index file **MUST** contain how many fields there are for each database. The next lines in the file then give the headings for each field.

There are some limitations on the fields in an oa_index file.
1. There is a maximum limit of 20 fields.
2. **The first field for each entry must be the filename which contains the data**. (*The filename given is automatically appended to the full path of the directory the index file is located in.)*
3. Each entry in the oa_index file is limited to 40 characters in length

An example of a oa_index file is given below.

```
$ Example oa_index file
$
$ NUMBER OF COLUMNS IN DATABASE
$
   3
$
$=====================================================================
$ THE COLUMN NAMES *** THE FIRST COLUMN MUST BE THE FILENAME ***
Filename
Units
Curve Description
$=====================================================================
$
$ THE DATABASE ENTRIES
$
kobe_x-accel.cur
m/s2
X Acceleration - Kobe
$
kobe_y-accel.cur
m/s2
Y Acceleration - Kobe
$
kobe_z-accel.cur
m/s2
Z Acceleration - Kobe
```

In this example above there are 3 fields called
- filename
- units
- curve description

MATL* - Material Databases

There is a maximum limit of 256 Material Databases. The entries in a oa_index file for a MATeriaL database **MUST** be formatted using 3 fields as follows.
- filename
- material name (this will be used to automatically match the material in a model with a material in the database

after it has been converted to upper case)
- LS-DYNA material model type (without the ***MAT_** prefix)

i.e
```
3
$
Filename
Material Name
LS_DYNA Material Type
$
steel_yield-200.key
   STEEL : YIELD 200 MPa
   PIECEWISE_LINEAR_PLASTICITY
$
steel_yield-250.key
   STEEL : YIELD 250 MPa
   PIECEWISE_LINEAR_PLASTICITY
```

Each data file should contain a single LS-DYNA material input card (in the KEYWORD format) along with any ***DEFINE_CURVE** and ***DEFINE_TABLE** definitions that are referenced by the material card. The material should be defined in the database file as material number 1 and any ***DEFINE_CURVE** and ***DEFINE_TABLE** cards should also be numbered sequentially from 1.

When a material is imported from a database file the following rules are used :
- If the file contains more than 1 material then only the material with the lowest material ID will be imported from the file.
- All ***DEFINE_CURVE** and ***DEFINE_TABLE** definitions will be imported .
- Any other LS-DYNA keyword data cards will be ignored.

APPENDIX X: Headform "tree" file example

```
*HEADFORM_START
$
$ Headform is at: 0.0 horizontal and 0.0 vertical
$ Reference point is at: 0.0 x 0.0 y 0.0 z
$
$  <label><title>
         1FTA FREE MOTION HEADFORM
$
*REF_POINT
$   <node>
         1
$
*UNITS
$<mass>   <length>  <time>
TE        MM        S
$
*AXES
$<coordID>
         1
$
*COMPONENTS
$ part_set cont_part   cont_no
         1         1         1
$
*TARGET
$<current>
$ No target point defined
$
*HEADFORM_END
```

APPENDIX XI: Target and Position "tree" file example

```
*TARGET_POINT_START
$  <label><posn>    <title>
         1AP2       Example target point
$
$ Target point coordinates
$      <x>       <y>       <z>
     500.0     500.0       0.0
$
$ Horizontal angles
$    <min>     <max>
      25.0      90.0
$
$ Velocity, contact part set and current position
$
$    <Vel><Cont set><head_pos>
   5364.48         2         1
$
*HEAD_POSITION
$  <label><name>
         1max h= 90.0
$
$ H-point position, headform angles and relation to target point
$      <x>       <y>       <z>   <horiz>    <vert>  <maxmin>
     500.0     500.0       0.0      90.0       0.0       max
$
*TARGET_POINT_END
```

APPENDIX XII: Dialogue (typed in) Command Syntax

Primer has a limited "dialogue" command set that can be used in any of three ways:
1. In graphical (screen menu) mode commands can be typed into the "Dialogue Box" at any time.
2. In non-graphical (text only) mode commands are typed in at the terminal prompt.
3. In command files, run either interactively or in batch, commands are executed as if typed in.

In all cases the command ***input*** syntax is identical, although there are minor differences in output between "screen menu" and "text-only" modes: in the latter case all output has to go to the controlling terminal ("stdout"), whereas in the former separate windows are used for "help", "listing" and other output.

The Dialogue command structure

The command structure forms a hierarchical "tree", with the top-level **PRIMER_MANAGER** at its "root".

The following rules apply:
- Command words may be abbreviated to any degree so long as:
  - they are unique in the context of their current menu
  - they must have at least their first two characters given

  For example **`READ DK_DYNA_KEYWORD`** may be abbreviated to **`RE DK`**.
- Navigation up and down menu levels is performed as follows:
  - <command> takes you to that command's (sub-)menu level
  - Forward slash "/" takes you back to the top **PRIMER_MANAGER** level before executing the following command(s)

  For example **`READ`** above takes you down into the **READ_MODEL** sub-menu
  The command **`/WRITE DK`** would work at the **READ_MODEL** prompt because it would return to the top level before parsing the **`WRITE`** command
- There is also a "global menu" of commands which is available at any (sub-)menu prompt.
  - These are primarily graphics commands that do not require a context.
  - The commands can be listed with the **`GM`** (for Global Menu) command

- Any command can be aborted by typing **`Q`**(uit). This will return control to the next highest command prompt in the "tree".
- At any prompt you can type **`H`**(elp) to receive advice about what to do next.

Commands that perform model-based operations:

READ

Reads in models in the following range of formats:

DK_DYNA3D_KEYWORD		Reads LS-DYNA keyword input decks
BDF_NASTRAN_BULK_DATA	TRANS	Nastran "Bulk data" (.bdf) file
UNV_UNIVERSAL_FILE	TRANS	Ideas/Master Series "universal" (.unv) files
RADF_RADIOSS_FIXED_FORMAT	TRANS	Radioss fixed format "starter" (00) files
RADB_RADIOSS_BLOCK_FORMAT	TRANS	Radioss block format "starter" (00) files
PNF_PATRAN_NEUTRAL_FILE	TRANS	Patran 2.5 "neutral" (.ntl) file
SAP_2000	TRANS	SAP 2000 input (.s2k) files
ABAQUS	TRANS	Abaqus keyword input decks

In all cases the syntax is **READ <format> <filename> <target model id>** for example: **READ DK test.key 2**

All the "non-DYNA3D" options imply a degree of translation, click on the TRANS buttons in the table above to view the relevant sections in Appendix VI which describe translation during input.

WRITE

Writes out models in the following range of formats:

DK_DYNA3D_KEYWORD DB_DYNA3D_KEYWORD DM_DYNA_MERGE_INCLUDES DO_DYNA_OMIT_INCLUDES		All four options write out LS-DYNA keyword files: • DK writes separate include files • DB writes out the master file only • DM merges include data into a single master file • DO omits data in include file
BDF_NASTRAN_BULK_DATA	TRANS	Writes out the model in Nastran Bulk Data File (BDF) format.
I5_IDEAS_LEVEL_5 **I6_IDEAS_LEVEL_6** **MS_IDEAS_MASTER_SERIES**	TRANS	All three options write Ideas or Master Series universal files, but the modules used reflect the version chosen.
P25_PATRAN_2.5_NEUTRAL **P3_PATRAN_3_NEUTRAL**	TRANS	Some of the <config> arguments change between Patran levels 2.5 and 3 neutral files.
ZTF_FILE		Writes the <name>.ztf file for post-processing in D3PLOT
GROUP_FILE		Writes <namennn>.bin file(s) to make *GROUP data available for post-processing in D3PLOT
PARTS_GROUP		Expanded ascii parts group file
SF_SUMMARY_FILE		Writes a summary file giving the main parameters of the model

In all cases the syntax is **WRITE<format> <filename> <target model id>** for example: **WRITE DK test.key 2**

The Ideas/Master Series, Patran and Nastran output formats require some translation, and sometimes some adjustments to element labels. Click on the TRANS buttons in the table above to view the relevant sections in Appendix VII which describe translation during output.

COPY

The syntax is **EXECUTE <list> <target>**

This copies the model(s) in **<list>** into the **<target>** model, which must not already exist.

DELETE

The syntax is **EXECUTE <list>**

This deletes the model(s) in **<list>**.

ORIENT

Applies orientation to models, or subsets of them.

The syntax is: **<orientation type> <object type> <list of objects> <orientation parameters>**

Where:

<orientation type>	Is one of **TRANSLATE, ROTATE, REFLECT, SCALE, CONTACT, AXIS_ROTATE, TRIADTOTRIAD**
<object type>	Is one of: • **MODEL** • **PART** • **<element type> (e.g. SOLID, SHELL,...)** • **NODE**
<list of objects>	Is a valid **<list>** of the relevant object type
<orientation parameters>	Depend on the <orientation type>: **TRANSLATE** **[dX, dY, dZ]** translation distances **ROTATE** **[rX, rY, rZ]** rotation angles in degrees **[cX, cY, cZ]** centre of rotation **REFLECT** **X** or **Y** or **Z** reflection axis **<Distance along axis>** **SCALE** **[sX, sY, sZ]** scales in each of X,Y,Z **[cx, cY, cZ]** centre of scaling **CONTACT** **[cX, cY, cZ]** centre of rotation **<N>** number of iterations **AXIS_ROTATE** **[nX, nY, nZ]** direction of translation **[rX, rY, rZ]** vector defining axis of rotation **<Angle>** rotation angle in degree **TRIADTOTRIAD** No parameters for this option.

Options for orient are given in the tabel below. These are typed in before **`<orientation type>`**

`USE_INC_TRANS`	For Include transform	Use include transform where possible
`DONT_USE_INC_TRANS`	For Include transform	Do not use include transform (def)
`MOVE_EXNO`	For controlling selection	Move extra nodes when rigid part oriented (def)
`NOMOVE_EXNO`	For controlling selection	Do not move extra nodes
`MOVE_BM3`	For controlling selection	Move n3 when beam oriented (def)
`NOMOVE_BM3`	For controlling selection	Do not move n3
`MOVE_SLAVE`	For controlling selection	Move slave when rigid master part oriented
`NOMOVE_SLAVE`	For controlling selection	Do not move slave (def)
`PROPAGATE`	For controlling selection	Propagate orient of parts and nodes (see below)
`NOPROPAGATE`	For controlling selection	Do not propagate (def)
`ADD_GAP <distance>`	For CONTACT operation	Distance shifted from the point of contact)

These commands should not all be given on one line! Give each section separately to avoid confusion, for example:

PRIMER_MANAGER >>> **`ORIENT TRANSLATE`**
ORIENT WHAT? **`MODEL`**
Give MODEL <list>: **`1 2 3`**
Give X,Y,Z translations: **`1.0 2.0 0.0`**

`DT_DATA_TRANSFER`

Transfers selected data from a "source" model to a "target" one, according to a range of parameters.

The syntax is:

`SOURCE <model id>`	Required. Defines the source model <model id> from which data are extracted.
`TARGET <model id>`	Required. Defines the target model <model_id> into which data will be transferred.

DATA_TYPE <type(s)>

Required, no default.

Defines the type(s) of data which will be transferred. Valid <type>s are:

MAT	Structural materials
EOS	Equations of state
SECTION	Element section data
HOURGLASS	Hourglass definitions
TMAT	Thermal materials

Selections are additive. Thus you could give the commands:

DATA_TYPE MAT SECTION TMAT

To transfer those three types of data.

You may also give the commands:

NOT <type>	(e.g.**NOT EOS**) to remove a selection from the current list
CLEAR	to clear the current selection
STATUS	to show what is currently selected.
DONE	to return to **DT_DATA_TRANSFER** prompt.

MATCH_BY <method>

Optional, defaults to "match by **NAME**".

Defines how objects in the source file are matched for data transfer. Valid <methods> are:

ID	If labels match
NAME	If **<source>** name (i.e._**TITLE**) is equal to or a subset of **<target>** name.
BOTH	First by **ID**, then by **NAME**.
ALL	All the data in the <source> is transferred to the <target>
STATUS	to show what is currently selected.
DONE	to return to **DT_DATA_TRANSFER** prompt.

ACTION <action>

Optional, defaults to action **CS_COPY_TO_SEPARATE**.

Defines how data transferred from source to target models is stored in the target model. Valid actions are:

CS_COPY_TO_SEPARATE	This causes transferred data to be stored in a separate include file in the target model. This include file will be given the name "dt_transfer_from_<*target*>.key" This is the default behaviour, and is recommended since it will make it easy to identify data that has been transferred.
CO_COPY_TO_ORIGINAL	The include file of the destination in the target model is unchanged by the transfer operation. For example if a material in include aaa.key is overwritten it remains in that file.
CM_COPY_TO_MASTER	Any data transferred into the target file is made to exist in the master file.
RO_READ_ONLY	This is rather different to the options above: • It is assumed that the <source> file will be included verbatim via an *INCLUDE keyword at run-time; • It is also assumed that the contents of this file will not be changed. Therefore the following actions are taken: • Any data transferred into the <target> file is placed in a special "Include" file which is marked "read only". • When the <target> model is finally written out this file is not included, rather the original <source> file is referred to via an *INCLUDE statement, using its full original pathname.
STATUS	Shows the current status of **ACTION**.
DONE	Returns to **DT_DATA_TRANSFER** prompt.

NAME_MATCH <method>

Optional, defaults to **EITHER**.

Defines how names in source and target models are matched when the **MATCH_BY** method is **NAME**.

T_IN_S	**T**arget ***I***n ***S***ource. A match is made if the target's title string is equal to or a subset of (<=) the source title.
S_IN_T	**S**ource **I**n ***T***arget. A match is made if the source's title string is equal to or a subset of (<=) the target title.
EITHER	A match is made if either **S_IN_T** or **T_IN_S** is true.
EXACT	A match is made only if the target and source titles are identical.
STATUS	Shows the current status of **NAME_MATCH**
DONE	Returns to **DT_DATA_TRANSFER** prompt.

Note that name matching is also subject to the following rules:

- Matching is case ***IN***sensitive. So **ABC = abc = ABC = ABC**
- Leading and trailing spaces are ignored. (But intermediate

  spaces between words are significant, thus
  " **ABCDEF** " matches "**ABCDEF**"
  "**ABC DEF**" does not match "**ABCDEF**"

SUPERSEDED

Optional: defaults to **SAVE**.

This controls how objects that are overwritten ("superseded") in the target model are handled.

SAVE	Relabels the "old" objects and transfers them to include file "dt_renumbered.key". They will not be referenced by anything and can be deleted by a **CLEANUP_UNUSED** operation.
DELETE	The incoming definition supersedes the object in the target model, and the original definition is lost.
STATUS	Shows the current status of **SUPERSEDED**
DONE	Returns to **DT_DATA_TRANSFER** prompt.

FEEDBACK Optional: defaults to PART_STATUS in non-graphical mode.

"Feedback" controls how much information about what happened during a data transfer operation is given to the user. Any permutation of the options below can be selected.

SKETCH	Sketches all updated objects on the current plot. In non-graphical mode this option is ignored.
PART_STATUS	Creates two tables: the first lists all parts that were affected by this data transfer operation, the second lists those that were not.
NOT <option>	Unsets <option>. For example **NOT SKETCH** means that sketching will not take place.
CLEAR	Clears all options leaving none selected.
STATUS	Shows the current status of **FEEDBACK**
DONE	Returns to **DT_DATA_TRANSFER** prompt.

STATUS Lists the status of all subcommands and settings in the **DT_DATA_TRANSFER** menu.

APPLY Checks the input parameters, and carries out the actual transfer operation.

Several of these settings can also be preset in the "oa_pref" file. See the data transfer section.

BOM (Bill of Materials)

Allows reading or writing of a Bill of Materials file. There are 5 options available:

READ_BOM	Read a bill of materials file. The columns in the file will be auto-detected from the headers and the file read.
NODE/MA/EL	Node/Mass/element renumbering activated
CREATE	Create mode activated
DEFAULT	Default (no create) mode without renumbering
WRITE_BOM	Write a bill of materials file

For all options the syntax is **<filename> <model id>.** Where:

<filename>	Is the filename of the Bill of materials to read/write
<model id>	Is the number of the model to read the BOM into/write the BOM from.

PART_INFO

Allows writing of a part information .csv file for all parts in a model. There are 2 options available:

WRITE	Write information in full with mass properties
NOMASS_PROPS	Write information excluding mass properties

For all options the syntax is **<filename> <model id>.** Where:

<filename>	Is the filename of the part information file to write
<model id>	Is the number of the model to write the part information file from.

ASSIGN_MASS

The syntax is **APPLY_ALL <model id>**

This applies all the assign mass definitions in model number **<model id>**.

MD_MATERIAL_ DATABASE

Performs a material database "import" operation.

The syntax is **<model id> <database id>.** Where:

<model id>	Is the number of the model in which material properties will be updated.
<database id>	Is the <i_th> *MATL database in the predefined database file.

Once these two parameters have been defined a "MATCH_ALL" operation is carried out.

SPOTWELD Performs spotweld functions. A number of options are available for this menu.

READ Read spotweld menu options, four file types are supported:

SPOTWELD	Read Primer spotweld file
CATIA	Read Catia spotweld file
UG	Read a UG weld file
CONNECTION	Read Primer XML connection file

The following syntax is **<filename> <model number> <part for welds>**. Where:

FILENAME	Connection file name
MODEL NUMBER	Model number to read connections into
PART FOR WELDS	Part for the spotweld beams/solids to go into. This must be a valid spotweld part: *SECTION_BEAM with ELFORM=9 (for beams) and material type *MAT_SPOTWELD

After the spotweld file is read, PRIMER tries to make the connections. Any bad welds are reported to the screen and written to a PRIMER spotweld file called **<inout weld file name>_bad**.
For example: 5 bad welds from **example.weld** would be written to the file **example.weld_bad**.

An example input line is: **/SPOTWELD READ CATIA example.weld 1 40000**
This would read the Catia spotweld file **example.weld** into model 1, part 40000.

WRITE Write spotweld menu options, four file types are supported:

SPOTWELD	Write Primer spotweld file
UG	Write a UG weld file
CONNECTION	Write Primer XML connection file
USER	Write a user defined connection file

The following syntax is **<filename> <model number>** . Where:

FILENAME	Connection file name
MODEL NUMBER	Model number to write connections out of

TYPE Here you can set the type of spotweld that will be created when reading in a connections file. Note this does not affect the connections read in from a Primer XML connections file, as this will contain the spotweld type already. Available types are:

BEAM	Create spotweld beams
HEXA	Create a single spotweld solid
4_HEXA	Create a solid spotweld nugget with 4 solids
8_HEXA	Create a solid spotweld nugget with 8 solids
12_HEXA	Create a solid spotweld nugget with 12 solids
16_HEXA	Create a solid spotweld nugget with 16 solids

DIAMETER Here you can set the diameter of the spotweld that will be created when reading in a connections file. Note this does not affect the connections read in from a Primer XML connections file, as this will contain the spotweld diameter already.

DUMMY and **MECHANISM**

ASSEMBLY	Select an assembly by name or number, then perform one of the following operations upon it:	**FIX** *dof code* — Restrain the assembly in degrees of freedom *dof code* **TRANSLATE** *dx, dy, dz* — Translate assembly ***by*** amount *dx,dy,dz* **RX** or **RY** or **RZ** *theta* — Rotate assembly ***to*** angle *theta* degrees about x/y/z **RESET** — Undo all dummy transformations and return to initial state **DONE** — Finish with assembly and return to **DUMMY >** prompt
POINT	Select a point by name or number, then perform one of the following operations upon it: (Note: moving the point implicitly moves its "owner" assembly.)	**FIX** *dof code* — Restrain the point in degrees of freedom *dof code* **TRANSLATE** *dx, dy, dz* — Translate point assembly ***by*** amount *dx,dy,dz* **POSITION** *x, y, z* — Translate point assembly ***to*** coord *x, y, z* **RESET** — Undo all dummy transformations and return to initial state **DONE** — Finish with point and return to **DUMMY >** prompt
CONNECTION	Select a connection by name or number	**SLIDE** *distance* — Applies to **LINE** connections only, and will slide the joint by *distance* down its AB axis. **ANGLE** *theta* — Applies to **LINE** and **HINGE** connections only, and rotations the assemblies to achieve angle *theta* (in degrees) about the AB axis.
POSITION	Specify a position *name* or *id*	Retrieves and applies the stored position *name* or *id*
SAVE	Specify a position id and (optional) *name*	Saves the current configuration as a saved position id, with optional *name*.
H_POINT	Specify coordinate *x, y, z*	Will move the Dummy H-Point ***to*** coord *x, y, z*
READ_CONFIG	Specify a *filename*	Retrieves a free-standing dummy configuration file (the keywords and data between ***DUMMY_START** and ***DUMMY_END**). *Filename* will usually have the extension .dcf
READ_DUMMY_ANGLE	Specify a *filename*	Retrieves and applies the overall orientation, H-Point and joint angles stored in a Dummy Angles File (usually extension .daf).

ACCEPT	Accept the current dummy position, save its updated geometry and return to the main [Primer >] prompt.
RESET	Undo all transformations and restore the initial geometry of the dummy, remaining at this prompt level.
QUIT	Undo all transformations and restore the initial geometry of the dummy, then return to the main [Primer >] prompt.

Meanings of terms in the table above	
dof code	Is a numeric Degree of Freedom code made up of any permutation of 123456, where 1 = Tx, 2 = Ty, 3 = Tz, 4 = Rx, 5 = Ry, 6 = Rz For example code **136** means restraint in Tx, Tz, Rz Code **0** may also be used, meaning "free all restraints"
dx, dy, dz	Is a translation vector, ie a relative movement from the current position, made up of three numbers. For example **10.0 20.0 30.0** means translate 10.0 in X, 20.0 in Y, 30.0 in Z. "Wildcard" syntax is permitted: any number entered as an asterisk ("*"), and omitted trailing digits, are treated as "free" values. For example: **10.0** means translate 10.0 in X, but permit Y and Z to adopt any value. *** * 20.0** means translate 20.0 in Z, but permit X and Y to adopt any value
x, y, z	Is an absolute coordinate. For example **10.0 20.0 30.0** means coordinate X=10, Y=20, Z=30. Wildcards as for translations above are permitted
theta	Is an angle in degrees for the given degree of freedom. In a dummy model angles are absolute values expressed in the coordinate system of the connection between this assembly and its parent. In most cases this will mean the system implied by the local axes of the joint stiffness definition at the joint.

BUILD Allows the user to access Primer's model build functionality. Available options are:

DATABASE To select a database file for build use:

The following syntax is **DATABASE <filename.dba>**

TEMPLATE To select a template file for build use:

The syntax is **TEMPLATE <filename.tpl>**

SIMPLE
RIGOROUS
MASTER

You apply the build process by selecting the build mode:
SIMPLE will build without checks or renumbering
RIGOROUS will build with checks and renumbering
MASTER <a.key> will just write out a masterfile to a.key (no orient)

An example of the syntax is:

```
BU DATAB all_models.dba TEMPL odb_40.tpl SIMP
```

Alternatively you can set up multiple models using a build file:

READ To set up multiple models using a build file:

The syntax is **READ <build.csv>**

Before READ you may set the following:

KEEP Keep each Primer model after keyout

DEL Delete Primer model after keyout (default)

ECHO Echo each line from input file

NOECHO Do not echo (default)

To sketch target points either before or after building:

SKETCH Sketch target points in file

The syntax is **SKETCH <build.csv>**

An example of the syntax is:

```
BU KEEP RE headpos.csv
```

The build mode is **SIMPLE** in this case

PREFERENCE Read the preference file. Available options are the location of the preference file:

SYSTEM look for the preference file in the $OASYS directory

HOME look for the preference file in the home directory

PATH use file path as supplied

After the option is chosen, you are prompted for the filename. An example of the syntax is:

```
HOME oa_pref
```

CHECK

This function will check the model specified. Syntax is:

```
MODEL n CHECKFILE filename APPLY
```

The name of an output file is specified with the CHECKFILE option. For example, to run an error check on model 2:

```
MODEL 2
CHECKFILE nyname.txt
APPLY
```

Omitting the checkfile option will give the default filename derived from the root of the dyna key file - <file>.check:

```
MODEL 2 APPLY
```

The **MODEL** option may be omitted if only one model in memory.

Before applying you may set the format option to **FULL_LIST** or **SHORT_LIST** (default). Just type **FULL_LIST** or **SHORT_LIST** to activate the option.

FULL_LIST will ensure that label lists are written for all item types. By default these are not written for the potentially highly populous types (NODES, ELEMENTS etc.).

AUTOFIX

This function will auto fix all errors/warning in the model. Syntax is:

```
MODEL n APPLY
```

The **MODEL** option may be omitted if only one model in memory.

CONNECTION The option allows the user to remake connection in the model. Available options are:

ALL_WELDS	remake all welds
ALL_GLUE	remake all adhesive
ALL_BOLTS	remake all rigid bolts
UNMADE_WELDS	remake unmade welds
UNMADE_GLUE	remake unmade adhesive
UNMADE_BOLTS	remake unmade rigid bolts

You can also modifiy the pitch of spotwelds by selecting the following option:

MODIFY_PITCH Change the pitch of spotwelds.

Required inputs for this operation are:

SELECT_SPOTWELDS	Select the spotwelds you wish to modify
NEW_PITCH	The new pitch value you wish to apply

After choosing **SELECT_SPOTWELDS** above, the user is prompted to select the type of entity through one of a number of different methods:

ALL	Select all spotwelds in the model
BY_LAYER_PART	Select the parts that have a layer that contains the selected parts
ATTACHED_TO_PANEL	Select spotwelds that are attached to the selected parts
BY_SINGLE_SEAM	Select spotwelds that are attached to ALL the selected parts
BY_MULTIPLE_SEAM	Select spotwelds that are attached to ANY combination of the selected parts
QUIT	Quit the selection process

When selecting parts for some of the above, you first define the type then the labels of items of the type. For example, to select a part range type **PART 10 to 19** or to select a part set type **PART_SET 5**.

Optional things to modify are:

BREAK_ANGLE	Used to determine how the lines of spotwelds are split (default 45.0 degrees)
MAX_PITCH	Maximum distance allowed between two spotwelds so that they can be considered on the same line (default 80.0mm)
SIMILAR_PITCH_ON	Group connections by similar pitch
SIMILAR_PITCH_OFF	Do not group connections by similar pitch (default)

See section 6.10.12 for more information on these inputs.

Finally:

APPLY	Apply the new pitch to the selected spotwelds.
DONE	Quit without applying the new pitch

RENUMBER Allows you to renumber entities within the model. Available options are

MODEL	Select the model you want to operate on, for example MODEL 2
SELECT/DESELECT	Select or deselect items to renumber. After choosing select or deselect, you then must select the type of entity, and finally the labels of the items you wish to renumber. The type is one of: • **MODEL** • **PART** • **<element type> (eg SOLID, SHELL,...)** • **NODE etc.**
APPLY	Choose a start point and apply the renumbering

Syntax is **MODEL n SELECT type**, followed by **APPLY.**

For example:

```
model 2
sel part
10 to 19
apply
```

MODEL option may be omitted if only one model in memory.

SCRIPT Reads a Primer javascript. Syntax is **READ <filename>**.

BELT Refit an existing seatbelt to a dummy. Primer will select a seatbelt definition in the model automatically. Available options are:

SELECT	Select a different belt definition for refitting
REFIT	Refits the current belt definition to its dummy
PR_BASIC	Retrieve and update basic belt dimensions and form-finding parameters
PR_REFIT	Retrieve and update belt refit parameters
DONE	To return to the main menu prompt

SEATSQUASH Impliment auto seatsquash function on a model. Available options are:

PRIMER The simple Primer method of seatsquash.

Required inputs for this method are:

SEAT_FOAM	Select the seat foam
SEAT_TOP	Select the top of the seat
SEAT_BOTTOM	Select the bottom of the seat
DUMMY	Select the dummy
CONTACT	Select the contact between the dummy and the seat
X	X displacement per iteration
Y	Y displacement per iteration
Z	Z displacement per iteration

After choosing **SEAT_FOAM**, **SEAT_TOP**, **SEAT_BOTTOM** or **DUMMY** above, the user is prompted to select the type of entity, then the labels of items of the type. For example, to select a part range type **PART 10 to 19**, o select a part set type **PART_SET 5** or to select a dummy type **DUMMY 17**.

Optional inputs for this method are:

MAX_ITER	Maximum number of iterations (default 100)
SOLID_REL	Stop if the solid element relative volume reaches this value (default 0.2)

Finally:

STATUS	Report what still has to be defined
APPLY	Initiate the seatsquash

DYNA The Dyna method of seatsquash. Sets up an LS-DYNA analysis that carries out the seatsquash operation.

Required inputs for this method are:

SEAT_ALL	Select the seat parts
SEAT_DEFORM	Select the seat parts to remain deformable
DUMMY	Select the dummy
CONTACT	Select the contact between the dummy and the seat
X	X displacement per iteration
Y	Y displacement per iteration
Z	Z displacement per iteration

After choosing **SEAT_ALL**, **SEAT_DEFORM** or **DUMMY** above, the user is prompted to select the type of entity, then the labels of items of the type. For example, to select a part range type **PART 10 to 19**, o select a part set type **PART_SET 5** or to select a dummy type **DUMMY 17**.

Optional inputs for this method are:

MAX_ITER	Maximum number of iterations (default 100)
DUMMY_POS_TIME	Time to position the dummy (default 0.075)
ANALYSIS_TIME	The termination time of the created analysis (default 0.1)
DAMPING	Global damping (default 50.0)
CONT_KEEP	Select contacts to keep in the analysis (by default all apart from the one selected above will be deleted)

Finally:

STATUS	Report what still has to be defined
APPLY	Initiate the seatsquash

IMPORT Import a Dynain file

EXPORT Export a Dynain file

DYNA_DUMMY Impliment Dyna dummy positioning method. Available options are:

POSITION Sets up a dyna analysis to position a dummy. When entering the POSITION setcion you will be asked if you wish to automatically determine assembly nodes. This is recommended if you have not already set these nodes up within PRIMER and saved the information to the dummy tree keywords in the keyword file associated with the dummy. Se section 6.12 for more information of how to do this.

Required inputs for this method are:

NECK_NODE	Node at the base of the neck (this may be set and saved to the keyword file pria to command line positioning)
HIP_NODE	Node on the left or right hip (this may be set and saved to the keyword file pria to command line positioning)
POSITION	Choose the desired position. This should be setup pria to command line positioning. See section 6.12.2 for information on saving positions.

Optional inputs/things to modify for this method are:

RIGID_ON	Select assemblies to rigidify
RIGID_OFF	Select assemblies to turn off rigidification
DEFORM_PARTS	Select parts to remain deformable in rigidified assemblies
ANALYSIS_TIME	The end time of the analysis (default 250.0)
DAMPING	Global damping (default 50.0)
DAMPING_OFF	Turn global damping off
CABLE_DAMPING	In-line damping applied to cables (default 0.5)
FORCE_RAMP	Force ramp up time in cable (default 10.0)
CABLE_FORCE	Force in cables (default 1.0)

Finally:

APPLY	Initiate setup

IMPORT This imports a dynain file produced in an LS-DYNA dummy positioning analysis.

Required inputs for this method are:

FILENAME	Select the dynain file to import
POSITION	Choose the desired position. This should be setup pria to command line positioning. See section 6.12.2 for information on saving positions.

Optional inputs for this method are:

COORDINATES	Import the deformed coordinates
SOLID_INIT	Import *INITIAL_STRESS_SOLID
SHELL_INIT	Import *INITIAL_STRESS_SHELL
BEAM_INIT	Import *INITIAL_STRESS_BEAM
DELETE_INIT	Delete existing *INITIAL_STRESS cards
ALL_OFF	Turn all options off

Finally:

APPLY	Import the data

Dialogue commands that control viewing - the "Global Menu"

The following commands can be used to control graphics and viewing. When running in "text-only" mode they are ignored and will have no effect.

Drawing commands: • **LINE** • **HIDDEN_LINE** • **SHADED**	These commands generate plots of the following types: • Line gives a wireframe plot (no hidden surface removal) • Hidden-line shows element edges, but with hidden surfaces removed • Shaded produces a shaded and lit plot, implicitly with hidden surfaces removed.
Commands which control the current view: • **SXY, SYZ, SZX** • **ISOMETRIC** • **RS <x y z>** • **RM <x y z>** • **ZM** • **CENTRE** • **MG <scale>** • **AU_AUTOSCALE** • **AC_AUTOSCALE_CURRENT** • **ZERO_VIEW**	These commands control how the image appears on the screen. • SXY is a view on the XY plane, down Z; SYZ on YZ down X; SZX on ZX down Y • Isometric is a view down the vector X=Y=Z (at 45 deg to each axis) • RS <x y z> rotates the image by <x> degrees about ***screen*** X, <y> about Y etc • RM <x y z> rotates the image by <x> degrees about ***model*** X, etc • ZM "zooms" in to the rectangle dragged out by the cursor • CENTRE centres the image at the cursor location • MG <scale> magnifies the image by <scale>. Values less than 1.0 reduce its size. • AU autoscales on the complete model, ignoring the effects of blanking etc (ie as if all unblanked) • AC autoscales on what is currently visible, taking into account blanking etc • ZERO_VIEW resets the viewing routines to their default state: Autoscaled view on SXY.
Other commands: • **GM_GLOBAL_MENU** • **GH_GLOBAL_HELP**	Commands to do with the global menu • GM lists the commands above, with a brief description • GH gives more general help on the "global" menu.

APPENDIX XIII: Summary of "oa_pref", command-line and Environment Variable settings

The data below has been described elsewhere in this manual, but is summarised below to concentrate it all in one place.

"oa_pref" files	Allow customisation of Primer via site-wide, user and directory specific settings.
Command-line arguments	May be appended to the line which executes Primer to specify operation modes, graphics drivers, etc
Environment variables	Can be used to modify behaviour and to preset many attributes.

The "oa_pref" preferences file.

This file contains code-specific preferences that can be used to modify the behaviour of Oasys Ltd LS-DYNA Environment products. It is optional and, where entries (or the whole file) are omitted Primer will revert to its default settings.

"oa_pref" naming convention and locations

The file is called "oa_pref", optionally with a leading "." (.oa_pref) on Unix systems.

It is looked for, and read if found, in all of the following places in the order given:
- The top level configuration directory **$OA_ADMIN_xx** (where xx =14for release14, etc)
- The installation directory **`$OA_INSTALL`**
- The user's home directory: **`$OA_HOME`** if set, otherwise **`$HOME`** (Unix/Linux) or **`%USERPROFILE%`** (Windows)
- The current working directory
- from any directory of the user's choosing - see custom oa_pref file

Files do not have to exist in any of these locations, and if none exists the programme defaults will be used.

If the same preference is set in more than one place the last occurence will dominate, thus a setting read from a pref file in the user's home directory will over-rule a setting in a pref file in the installation directory.

That said, preferences (excepting those that control the graphics) in the higher level directories may by locked by using # instead of * in the syntax. In this case the locked setting will win.

On Unix and Linux:

`$HOME` on Unix and Linux is usually the home directory specified for each user in the system password file.

The shell command "**`printenv`**" (or on some systems "**`setenv`**") will show the value of this variable if set. If not set then it is defined as the "**`~`**" directory for the user. The command "**`cd; pwd`**" will show this.

On Windows:

`%USERPROFILE%` on Windows is usually:
Windows XP: **`C:\Documents and Settings\`***`<user id>`***`\`**
Vista/W7: **`C:\users\`***`<user id>`*

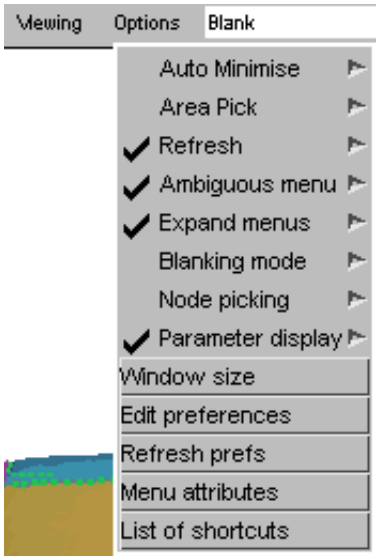
Issuing the "**`set`**" command from an MS-DOS prompt will show the value of this and other variables.

Generally speaking you should put
- Organisation-wide options in the version in OA_INSTALL,
- User-specific options in OA_HOME if you wish to define it, otherwise $HOME / $USERPROFILE
- Project-specific options in the current working directory.

Re-reading the "oa_pref" file

You no longer need to quit and restart Primer to return your settings to the state they are in on initial start up.

Primer can now restore settings to their default value and re-read the system and home oa_pref files using the feature **Refresh Preference Settings**. It is available under CHECK > OPTIONS and from the main Options drop-down.



"oa_pref" file syntax

The syntax used for Primer is: **primer*<keyword>: <argument>** or for a locked pref **primer#<keyword>: <argument>**

for example:

```
primer*initial_plot_mode: SHAD
```

The rules for formatting are:

- The `<programme>*<option>:` string must start at column 1;
- This string must be in lower case, and must not have any spaces in it.
- The `<argument>` must be separated from the string by at least one space.
- Lines starting with a "`#`" are treated as comments and are ignored.

"Locking" a preference

Normally a preference read from oa_pref file in location A can be modified if the same preference is read later on from oa_pref file in location B. However it it possible to "lock" a preference against being changed by using a "#" rather than a "*" in its syntax. This means that once read it will not be changed if read again from subsequent files.

For example:

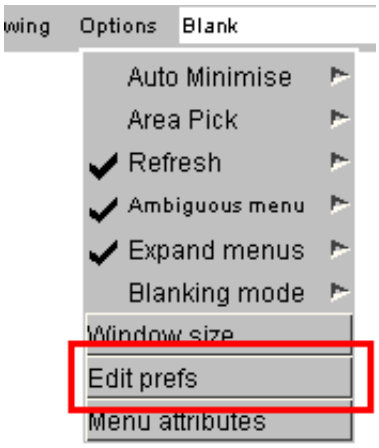
`primer*initial_plot_mode: SHAD`	Using a "*" means that this preference can be modified.
`primer#initial_plot_mode: SHAD`	Using a "#" means that it is locked against further changes

This facility enables "company wide" preferences to be set and locked in the top level (eg OA_ADMIN) preference file, thus enforcing their use.

The interactive Preferences Editor

You are free to edit oa_pref files by hand, but there is an interactive "Preferences Editor" that may be called from within PRIMER that makes the job much easier.

It is started by **Options, Edit Prefs:**



The preferences editor reads an XML file that contains all possible preferences and their valid options, and allows you to change them at will. In this example the user is changing the background colour in PRIMER.

Note that changes made in the Preferences editor will not affect the current session of PRIMER, they will only take effect the next time it is run.

If you have write permission on the oa_pref file in the $OASYS directory you will be asked if you want to update that file, otherwise you will only be given the option of updating your own file in your $HOME / $USERPROFILE directory.

In this example the user is changing the background colour.

The option is "active" (ie present in the oa_pref file) and currently is set to BLACK.

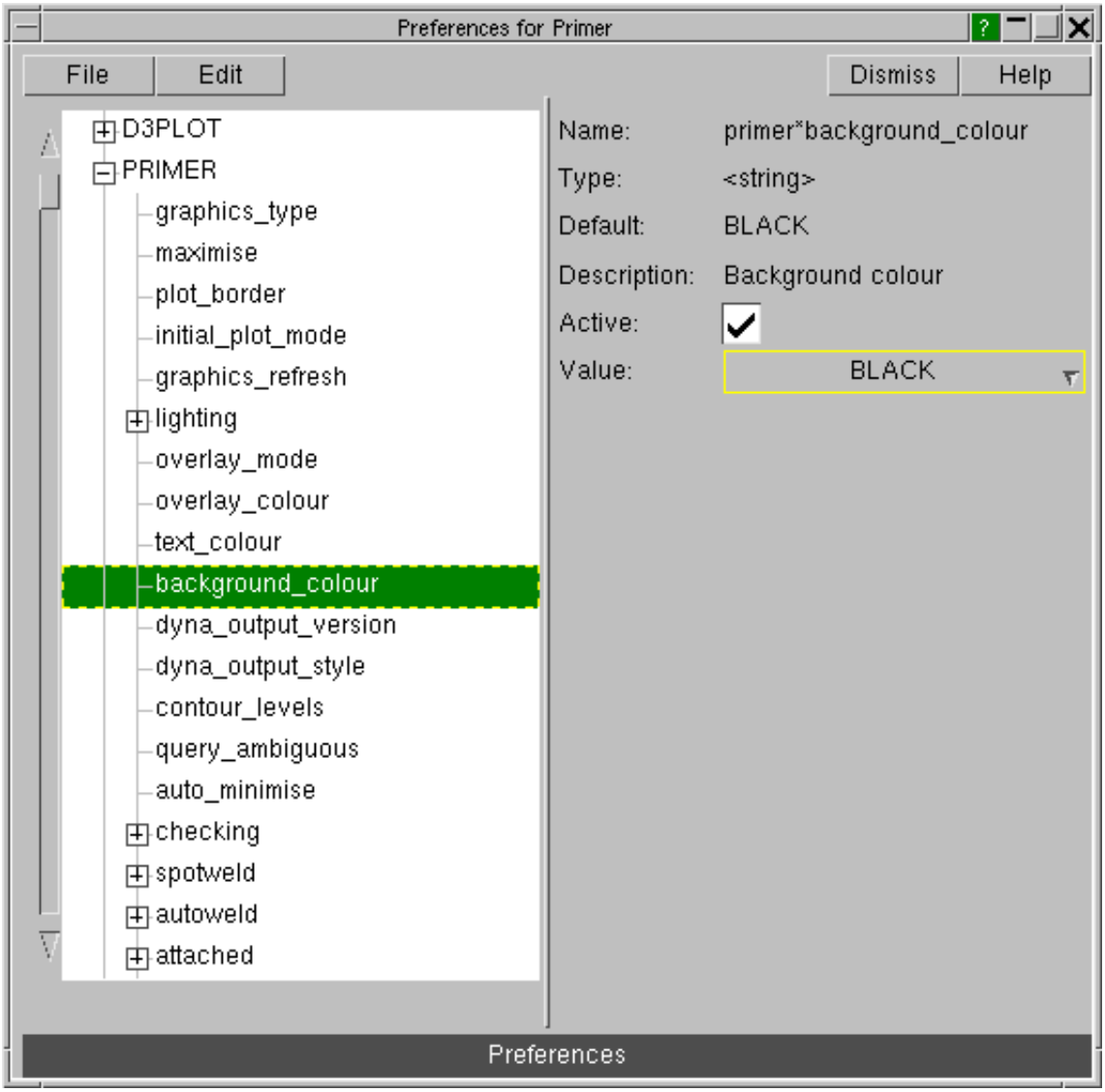
Usage is:
- Select an option in the Tree on the left hand side
- Make it active / inactive
- If active select a value from the popup, or type in a value if necessary

The colour of the highlighting in the left hand side tree is significant:

Green	Means that the option has been read from your $HOME file.
Red	Means that the option has been read from the $OASYS file.

In either event, regardless of the data source, the updated option will be written to the file chosen when you started the preferences editor.

Because of the order of file reading (see above), and option read from the master $OASYS file, amended, and written to your local $HOME file will take precedence when you next run PRIMER.



"oa_pref" arguments valid for Primer

Preference	Type	Description	Valid arguments	Default
checkpoint_dir	<string>	Directory for checkpoint files, or "none" to suppress them altogether		<none>
conx_table_columns	<string>	Columns initially shown in connection table		<none>
error_handler	<string>	how to handle errors and exceptions	no_action, mini_dump, trap_continue, trace_exit	mini_dump
part_table_columns	<string>	Columns initially shown in part table		<none>
pdf_keyword_args	<string>	Commands to use when opening keyword manual using user specified pdf application.		<none>
save_dialogue_dir	<string>	Directory in which to save dialogue info on exit (otherwise deleted)		<none>
segment_count_for_bucket_sort_warning	<integer>	Threshhold for warning of excessive number of segments in bucket		1000000
start_in	<string>	Directory to start Primer in		<none>
show_license_warning	<logical>	Display Window containing License System messages	TRUE, FALSE	TRUE
save_window_positions	<logical>	Save position of undocked windows between sessions	TRUE, FALSE	TRUE

The following controls the default ascii file output

Preference	Type	Description	Valid arguments	Default
ABSTAT_asc	<string>	ABSTAT ascii file selected	ON, OFF	OFF
BNDOUT_asc	<string>	BNDOUT ascii file selected	ON, OFF	OFF
DEFGEO_asc	<string>	DEFGEO ascii file selected	ON, OFF	OFF
DEFORC_asc	<string>	DEFORC ascii file selected	ON, OFF	OFF
ELOUT_asc	<string>	ELOUT ascii file selected	ON, OFF	OFF
GCEOUT_asc	<string>	GCEOUT ascii file selected	ON, OFF	OFF
GLSTAT_asc	<string>	GLSTAT ascii file selected	ON, OFF	OFF
H3OUT_asc	<string>	H3OUT ascii file selected	ON, OFF	OFF
JNTFORC_asc	<string>	JNTFORC ascii file selected	ON, OFF	OFF
MATSUM_asc	<string>	MATSUM ascii file selected	ON, OFF	OFF
NCFORC_asc	<string>	NCFORC ascii file selected	ON, OFF	OFF
NODFOR_asc	<string>	NODFOR ascii file selected	ON, OFF	OFF
NODOUT_asc	<string>	NODOUT ascii file selected	ON, OFF	OFF
RBDOUT_asc	<string>	RBDOUT ascii file selected	ON, OFF	OFF
RWFORC_asc	<string>	RWFORC ascii file selected	ON, OFF	OFF
SECFORC_asc	<string>	SECFORC ascii file selected	ON, OFF	OFF
RCFORC_asc	<string>	RCFORC ascii file selected	ON, OFF	OFF
SBTOUT_asc	<string>	SBTOUT ascii file selected	ON, OFF	OFF
SLEOUT_asc	<string>	SLEOUT ascii file selected	ON, OFF	OFF
SPCFORC_asc	<string>	SPCFORC ascii file selected	ON, OFF	OFF
SPHOUT_asc	<string>	SPHOUT ascii file selected	ON, OFF	OFF
SWFORC_asc	<string>	SWFORC ascii file selected	ON, OFF	OFF
TPRINT_asc	<string>	TPRINT ascii file selected	ON, OFF	OFF
TRHIST_asc	<string>	TRHIST ascii file selected	ON, OFF	OFF

The following relate to assemblies

Preference	Type	Description	Valid arguments	Default
assembly_read_mode	<string>	Action for existing assemblies when reading assembly data in part tree	RESET, OVERWRITE, ASK	ASK

The following relate to assign mass function

Preference	Type	Description	Valid arguments	Default
assign_mass_percent_error_tolerance	<real>	Error tolerance (percent) for Assign Mass Calculation		5.0
assign_mass_includes_timestep_mass	<string>	massing up function includes timestep added mass	OFF, ON	OFF
assign_mass_includes_attached_mass	<string>	Default is to include attached mass which is free or belongs to ASSM of lower label if overmassing	FREE_AND_LOWER, NONE, ALL	FREE_AND_LOWER

The following control attributes of the attached function

Preference	Type	Description	Valid arguments	Default
attached_tied_contact	<string>	Attach through tied contacts (program setting intially ON)	ON, OFF, PROG	PROG
attached_beam_pid	<string>	Attach through beam PIDs (program setting intially OFF)	ON, OFF, PROG	PROG
attached_beam_3rd_node	<string>	Attach through beam third node (program setting intially OFF)	ON, OFF, PROG	PROG
attached_recursive	<string>	Find attached recursively (program setting intially OFF)	ON, OFF, PROG	PROG
find_attached_through	<string>	Sets up the initial entity switches for attached		<none>

The following controls the default binary file output

Preference	Type	Description	Valid arguments	Default
ABSTAT_bin	<string>	ABSTAT binary file selected	ON, OFF	OFF
BNDOUT_bin	<string>	BNDOUT binary file selected	ON, OFF	OFF
DEFGEO_bin	<string>	DEFGEO binary file selected	ON, OFF	OFF
DEFORC_bin	<string>	DEFORC binary file selected	ON, OFF	OFF
ELOUT_bin	<string>	ELOUT binary file selected	ON, OFF	OFF

GCEOUT_bin	<string>	GCEOUT binary file selected	ON, OFF	OFF
GLSTAT_bin	<string>	GLSTAT binary file selected	ON, OFF	OFF
H3OUT_bin	<string>	H3OUT binary file selected	ON, OFF	OFF
JNTFORC_bin	<string>	JNTFORC binary file selected	ON, OFF	OFF
MATSUM_bin	<string>	MATSUM binary file selected	ON, OFF	OFF
NCFORC_bin	<string>	NCFORC binary file selected	ON, OFF	OFF
NODFOR_bin	<string>	NODFOR binary file selected	ON, OFF	OFF
NODOUT_bin	<string>	NODOUT binary file selected	ON, OFF	OFF
RBDOUT_bin	<string>	RBDOUT binary file selected	ON, OFF	OFF
RWFORC_bin	<string>	RWFORC binary file selected	ON, OFF	OFF
SECFORC_bin	<string>	SECFORC binary file selected	ON, OFF	OFF
RCFORC_bin	<string>	RCFORC binary file selected	ON, OFF	OFF
SBTOUT_bin	<string>	SBTOUT binary file selected	ON, OFF	OFF
SLEOUT_bin	<string>	SLEOUT binary file selected	ON, OFF	OFF
SPCFORC_bin	<string>	SPCFORC binary file selected	ON, OFF	OFF
SPHOUT_bin	<string>	SPHOUT binary file selected	ON, OFF	OFF
SWFORC_bin	<string>	SWFORC binary file selected	ON, OFF	OFF
TPRINT_bin	<string>	TPRINT binary file selected	ON, OFF	OFF
TRHIST_bin	<string>	TRHIST binary file selected	ON, OFF	OFF

The following control BOM read

Preference	Type	Description	Valid arguments	Default
alternate_bom_read_method	<logical>	when reading BOM set material from title match; create section, hourglass	TRUE, FALSE	FALSE
bom_read_apply_target_massing	<string>	option to process target massing when reading BOM	ASK, TRUE, FALSE	ASK

The following control attributes of the model checking function

Preference	Type	Description	Valid arguments	Default
rechecking_level	<string>	Post fix/edit recheck level in error tree viewer	0, 1, 2, 3	2
contact				
sliding_contact				
tied_contact				
element_quality_checks				
element_quality_checks_active	<string>	Element quality check settings at program start	ON, OFF, PREF	PREF
structural_element				
elem_free_end_both	<string>	Report 1D elements only if both nodes are free	ON, OFF	OFF
elem_free_end_check	<string>	Both nodes of 1D elements should be structural	ON, OFF	OFF
elem_free_end_rigid	<string>	Report rigid 1D elements with free ends	ON, OFF	OFF
element_aspect_ratio_check	<string>	Element quality aspect ratio checking	ON, OFF	OFF
element_max_aspect_ratio	<real>	Element quality maximum aspect ratio		5
element_jacobian_check	<string>	Element quality jacobian checking	ON, OFF	OFF
element_min_jacobian	<real>	Element quality minimum jacobian	0.0 - 1.0	0.7
element_jacobian_calc_method	<string>	Element jacobian calculation method	GAUSS, NODAL	GAUSS
element_length_check	<string>	Element quality length checking	ON, OFF	OFF
element_min_length	<real>	Element quality minimum length		5
element_skew_check	<string>	Element quality skew checking	ON, OFF	OFF
element_max_skew	<real>	Element quality maximum skew	0.0 - 180.0	60
element_taper_check	<string>	Element quality taper check	ON, OFF	OFF
element_max_taper	<real>	Element quality maximum taper		0.5

element_tet_collapse_check	<string>	Element quality tetrahedral collapse check	ON, OFF	OFF
element_min_tet_collapse	<real>	Element quality minimum tetrahedral collapse		0.1
element_tria_angle_check	<string>	Element quality tria internal angle checking	ON, OFF	OFF
element_max_tria_angle	<real>	Element quality tria maximum internal angle	0.0 - 180.0	120
element_min_tria_angle	<real>	Element quality tria minimum internal angle	0.0 - 180.0	30
element_quad_angle_check	<string>	Element quality quad internal angle checking	ON, OFF	OFF
element_max_quad_angle	<real>	Element quality quad maximum internal angle	0.0 - 180.0	140
element_min_quad_angle	<real>	Element quality quad minimum internal angle	0.0 - 180.0	40
element_warpage_check	<string>	Element quality warpage checking	ON, OFF	OFF
element_max_warpage	<real>	Element quality maximum warpage	0.0 - 180.0	20
maximum_warpage_criterion	<real>	Used to condition the calculation for highly warped elements	0.0 - 180.0	180.0
ignore_null_part_element_checks	<logical>	Element quality checks will not include elements of null parts (*MAT_NULL)	TRUE, FALSE	TRUE
tet_skew_check	<string>	Skew check on Tet elements	ON, OFF	OFF
part_quality_check	<string>	Part quality check	ON, OFF	OFF
part_quality_min_elem	<integer>	Min required number of elements for part quality check		1
part_quality_percent_threshold	<real>	Threshold for part quality check		10.0
weighting_factors				
quality_wt_admass	<real>	Weighting factor for max added mass		0.0
quality_wt_aspr	<real>	Weighting factor for aspect ratio		1.0
quality_wt_jac	<real>	Weighting factor for jacobian		1.0
quality_wt_lang	<real>	Weighting factor for min angle		1.0
quality_wt_leng	<real>	Weighting factor for length		1.0
quality_wt_skew	<real>	Weighting factor for skew		1.0
quality_wt_tap	<real>	Weighting factor for taper		0.0
quality_wt_tet	<real>	Weighting factor for tetrahedral collapse		0.0
quality_wt_tstp	<real>	Weighting factor for min timestep		0.0
quality_wt_uang	<real>	Weighting factor for max angle		1.0
quality_wt_warp	<real>	Weighting factor for warpage		1.0
error_configuration_file	<string>	user file to configure error/warning/ignore status		<none>
error_tags	<string>	show Primer error code for every error and warning	ON, OFF	OFF
history				
database_node_check	<string>	Check for absence of database history node	ON, OFF	OFF
database_shell_check	<string>	Check for absence of database history shell	ON, OFF	OFF
database_solid_check	<string>	Check for absence of database history solid	ON, OFF	OFF
database_beam_check	<string>	Check for absence of database history beam	ON, OFF	OFF
database_sbelt_check	<string>	Check for absence of database history seatbelt	ON, OFF	OFF
database_discrete_check	<string>	Check for absence of database history discrete	ON, OFF	OFF
include				
part_element_include_check	<string>	Check elements in same include as part	ON, OFF	OFF
element_node_include_check	<string>	Check nodes in same include as element	ON, OFF	OFF
mass_node_include_check	<string>	Check ELEMENT_MASS(_NODE) in same include as node	ON, OFF	OFF
section_include_check	<string>	Check sections in same include as part	ON, OFF	OFF
material_include_check	<string>	Check materials in same include as part	ON, OFF	OFF
joint				
minimum_joint_mass_value	<real>	minimum value of mass on joint node		0.0
minimum_cylindrical_joint_size	<real>	minimum value for size of cylindrical joint		0.0
label_clash				
element_label_clash	<string>	Warn if there is a clash in elements ID	ON, OFF	OFF
set_label_clash	<string>	Warn if there is a clash in sets ID	ON, OFF	OFF
material_label_clash	<string>	Warn if there is a clash in materials ID	ON, OFF	OFF
material				
mat24_vp_check	<string>	Check VP set when strain rate is active	ON, OFF	OFF

mat24_strain_check_limit	<real>	Limiting strain value for Matl curve and table check		10.0
mat24_required_table_curve _separation_factor	<real>	Minimum required table curve separation factor		0.01
mat24_check_curve _discretization	<string>	Check for accuracy loss from discretization of MAT24 curves	ON, OFF	OFF
allowable_relative_error_for _discretized_mat24_curve	<real>	allowable relative error in interpolated Y value for discretized curve		0.01
model_check				
model_check_airbag	<string>	Check category AIRBAG during Model Check	ON, OFF	OFF
model_check_ale	<string>	Check category ALE during Model Check	ON, OFF	OFF
model_check_boundary	<string>	Check category BOUNDARY during Model Check	ON, OFF	OFF
model_check_case	<string>	Check category CASE during Model Check	ON, OFF	OFF
model_check_component	<string>	Check category COMPONENT during Model Check	ON, OFF	OFF
model_check_connection	<string>	Check category CONNECTION during Model Check	ON, OFF	OFF
model_check_constrained	<string>	Check category CONSTRAINED during Model Check	ON, OFF	OFF
model_check_contact	<string>	Check category CONTACT during Model Check	ON, OFF	OFF
model_check_control	<string>	Check category CONTROL during Model Check	ON, OFF	OFF
model_check_damping	<string>	Check category DAMPING during Model Check	ON, OFF	OFF
model_check_database	<string>	Check category DATABASE during Model Check	ON, OFF	OFF
model_check_define	<string>	Check category DEFINE during Model Check	ON, OFF	OFF
model_check_def_to_rigid	<string>	Check category DEF_TO_RIGID during Model Check	ON, OFF	OFF
model_check_dummy	<string>	Check category DUMMY during Model Check	ON, OFF	OFF
model_check_element	<string>	Check category ELEMENT during Model Check	ON, OFF	OFF
model_check_eos	<string>	Check category EOS during Model Check	ON, OFF	OFF
model_check_frequency	<string>	Check category FREQUENCY during Model Check	ON, OFF	OFF
model_check_group	<string>	Check category GROUP during Model Check	ON, OFF	OFF
model_check_hourglass	<string>	Check category HOURGLASS during Model Check	ON, OFF	OFF
model_check_include_file	<string>	Check category INCLUDE FILE during Model Check	ON, OFF	OFF
model_check_initial	<string>	Check category INITIAL during Model Check	ON, OFF	OFF
model_check_integration	<string>	Check category INTEGRATION during Model Check	ON, OFF	OFF
model_check_interface	<string>	Check category INTERFACE during Model Check	ON, OFF	OFF
model_check_load	<string>	Check category LOAD during Model Check	ON, OFF	OFF
model_check_material	<string>	Check category MATERIAL during Model Check	ON, OFF	OFF
model_check_mechanism	<string>	Check category MECHANISM during Model Check	ON, OFF	OFF
model_check_mesh	<string>	Check category MESH during Model Check	ON, OFF	OFF
model_check_node	<string>	Check category NODE during Model Check	ON, OFF	OFF
model_check_parameter	<string>	Check category PARAMETER during Model Check	ON, OFF	OFF
model_check_part	<string>	Check category PART during Model Check	ON, OFF	OFF
model_check_particle	<string>	Check category PARTICLE during Model Check	ON, OFF	OFF
model_check_perturbation	<string>	Check category PERTURBATION during Model Check	ON, OFF	OFF
model_check_rail	<string>	Check category RAIL during Model Check	ON, OFF	OFF
model_check_rigidwall	<string>	Check category RIGIDWALL during Model Check	ON, OFF	OFF
model_check_section	<string>	Check category SECTION during Model Check	ON, OFF	OFF
model_check_sensor	<string>	Check category SENSOR during Model Check	ON, OFF	OFF
model_check_set	<string>	Check category SET during Model Check	ON, OFF	OFF
model_check_termination	<string>	Check category TERMINATION during Model Check	ON, OFF	OFF
model_check_translate	<string>	Check category TRANSLATE during Model Check	ON, OFF	OFF
model_check_user	<string>	Check category USER during Model Check	ON, OFF	OFF
model_check_em_2daxi	<string>	Check category EM_2DAXI during Model Check	ON, OFF	OFF
model_check_em_boundary	<string>	Check category EM_BOUNDARY during Model Check	ON, OFF	OFF
model_check_em_circuit	<string>	Check category EM_CIRCUIT during Model Check	ON, OFF	OFF
model_check_em_circuit_rogo	<string>	Check category EM_CIRCUIT_ROGO during Model Check	ON, OFF	OFF
model_check_em_contact	<string>	Check category EM_CONTACT during Model Check	ON, OFF	OFF
model_check_em_contact _resistance	<string>	Check category EM_CONTACT_RESISTANCE during Model Check	ON, OFF	OFF
model_check_em_eos	<string>	Check category EM_EOS during Model Check	ON, OFF	OFF
model_check_em_external_field	<string>	Check category EM_EXTERNAL_FIELD during Model Check	ON, OFF	OFF
model_check_em_mat	<string>	Check category EM_MAT during Model Check	ON, OFF	OFF

model_check_em_rotation_axis	<string>	Check category EM_ROTATION_AXIS during Model Check	ON, OFF	OFF
model_check_em_solver	<string>	Check category EM_SOLVER during Model Check	ON, OFF	OFF
model_check_icfd_control	<string>	Check category ICFD_CONTROL during Model Check	ON, OFF	OFF
model_check_icfd_database	<string>	Check category ICFD_DATABASE during Model Check	ON, OFF	OFF
model_check_icfd_define	<string>	Check category ICFD_DEFINE during Model Check	ON, OFF	OFF
model_check_icfd_initial	<string>	Check category ICFD_INITIAL during Model Check	ON, OFF	OFF
model_check_chemistry_control	<string>	Check category CHEMISTRY_CONTROL during Model Check	ON, OFF	OFF
model_check_chemistry_det_initiation	<string>	Check category CHEMISTRY_DET_INITIATION during Model Check	ON, OFF	OFF
model_check_chemistry_composition	<string>	Check category CHEMISTRY_COMPOSITION during Model Check	ON, OFF	OFF
model_check_chemistry_path	<string>	Check category CHEMISTRY_PATH during Model Check	ON, OFF	OFF
model_check_geometry	<string>	Check category GEOMETRY during Model Check	ON, OFF	OFF
model_keyout_check	<string>	Generate check info on model keyout	ON, OFF	OFF
separate_check_file	<string>	write check info into fname.check	ON, OFF	OFF
model_quality_checks				
model_quality_checks_active	<string>	Model quality check settings at program start	ON, OFF, PREF	PREF
model_timestep_check	<string>	Model timestep check	ON, OFF	OFF
model_min_timestep	<real>	Model minimum timestep		1.e-6
model_max_timestep	<real>	Model maximum timestep		<none>
model_added_mass_check	<string>	Model added mass check	ON, OFF	ON
model_max_added_mass	<real>	Model maximum added mass		<none>
model_max_added_mass_percent	<real>	Model maximum added mass percent		5.0
part_added_mass_check	<string>	Part added mass check	ON, OFF	OFF
part_max_added_mass	<real>	Part maximum added mass		<none>
part_max_added_mass_percent	<real>	Part maximum added mass percent		5.0
spotweld_part_max_added_mass	<real>	Spotweld Part maximum added mass		<none>
spotweld_part_max_added_mass_percent	<real>	Spotweld Part maximum added mass percent		5.0
element				
element_timestep_check	<string>	Element timestep checking	ON, OFF	OFF
element_min_timestep	<real>	Element minimum timestep		1.e-6
spotweld_min_timestep	<real>	Spotweld minimum timestep		1.e-6
element_added_mass_check	<string>	Element added mass check	ON, OFF	OFF
element_max_added_mass	<real>	Element maximum added mass		<none>
element_max_added_mass_percent	<real>	Element maximum added mass percent		5.0
spotweld_max_added_mass	<real>	Spotweld maximum added mass		<none>
spotweld_max_added_mass_percent	<real>	Spotweld maximum added mass percent		5.0
element_overlap_check	<string>	Element overlap checking	ON, OFF	OFF
element_overlap_different_part	<string>	treat elements in different parts as overlapping	ON, OFF	OFF
part_checks				
empty_part	<string>	check for empty parts	ON, OFF	ON
def_part_continuity_check	<string>	Deformable part mesh continuity check	ON, OFF	OFF
deformable_size	<real>	optional maximum part size for continuity check		0.0
part_normal_check	<string>	Part normal consistency check	ON, OFF	OFF
part_crack_check	<string>	Part crack check	ON, OFF	OFF
check_percent_of_triangle	<string>	Check the percentage of triangle in shell parts	ON, OFF	OFF
max_percentage_of_triangle	<real>	Value of the max percentage of triangle admitted in a part		10.0
rigid				
nodal_rigid_body_minimum_mass	<real>	minimum mass required for nodal rigid body (0.0 for off)		0.0

nrb_release_flag_check	<string>	Warning issued if release flags set on nodal rigid bodies	ON, OFF	OFF
rigid_part_minimum_mass	<real>	minimum mass required for rigid part (0.0 for auto)		0.0
accelerometer_minimum_mass	<real>	minimum mass required for accelerometer		0.0
rigid_body_merge_check	<string>	Rigid body merge checking	ON, OFF	OFF
rigid_body_merge_max_separation	<real>	Maximum size for rigid body merge		100.0
rigid_body_continuity_check	<string>	Rigid body mesh continuity check	ON, OFF	OFF
rigid_body_min_elem	<integer>	Min required num of elems for rigid body continuity check		3
rigid_part_size	<real>	optional maximum part size for continuity check		0.0
nodal_rigid_body_size_check	<string>	Check size of nodal rigid bodies	ON, OFF	OFF
maximum_nodal_rigid_body_size	<real>	Maximum size for nodal rigid body		100.0
xsec				
check_database_cross_section_parts	<string>	check that parts cut by finite Database Xsec are in PSET (if defined)	TRUE, FALSE	FALSE
other				
allow_unused_nrb_master_node	<string>	allow unused nodal rigid body master node	ON, OFF	OFF
control_solution_undef_isnan_check	<string>	Flag undefined ISNAN parameter as an error	ON, OFF	OFF
check_missing_output_fields	<string>	Model check includes tests for non-zero data fields omitted on output due to LS-DYNA version	WARNING, ERROR, NO_CHECK	WARNING
warn_param_8_chars	<string>	Warn if 8 character parameter names found	ON, OFF	ON
section				
shell_thickness_check	<string>	On section update warn if *ELEMENT_SHELL_THICKNESS is set	ON, OFF	ON
composite				
create_composite_long	<logical>	If TRUE _LONG is added to *ELEMENT_SHELL_COMPOSITE when creating a composite layup	TRUE, FALSE	TRUE
composites_sketch_mode	<string>	determines whether the composite ply directions are sketched using lines or arrows	LINES, ARROWS	LINES
composites_mapping_parameter	<real>	Determines the strength of the map lines on the interpolated composite direction field		1.5
composites_shell_qual_angle	<real>	Angle which determines which shells are sketched as having bad quality during composites orient.		45.0
ztf_output_composite_action	<string>	Action for dialogue box when *PART_COMPOSITE/*ELEMENT_SHELL_COMPOSITE cards are being written out in ztf without any piles.	Yes_Del, Yes, No	<none>

The following apply to the connections function

Preference	**Type**	**Description**	**Valid arguments**	**Default**
connection_file_type	<string>	Default spotweld file type	PRIMER_SPOTWELD, PRIMER_CONNECTION, CATIA, UG, USER	PRIMER_CONNECTION
connection_read_script	<string>	User defined JavaScript to read connections		<none>
connection_write_script	<string>	User defined JavaScript to write connections		<none>
connection_same_part	<logical>	TRUE if a part can be joined to itself with a connection entity (weld or glue)	TRUE, FALSE	FALSE
connection_allow_clinch	<logical>	TRUE if shell elements of different parts meshed together forming a clinch can be connected	TRUE, FALSE	FALSE

connection_node_element _numbering_rule	<string>	default rule for numbering of connection nodes and elements	HIGHEST_PLUS_ONE, LAYER_FIRST_FREE, LAYER_HIGHEST_FREE, LAYER_HIGHEST_PLUS ONE	LAYER_HIGHEST _PLUS_ONE
connection_general_item _numbering_rule	<string>	default rule for numbering of connection general items	HIGHEST_PLUS_ONE, LAYER_FIRST_FREE, LAYER_HIGHEST_FREE, LAYER_HIGHEST_PLUS ONE	LAYER_HIGHEST _PLUS_ONE
connection_create_length _rigorous	<logical>	TRUE:do not make a connection if it fails max/min length settings. FALSE:connect fewer layers to produce a shorter connection	TRUE, FALSE	FALSE
rigidify_makes_contact _node_sets	<logical>	Interactive user may set this false and use Connection > Contacts to repair the model	TRUE, FALSE	TRUE
connection_xml_file_paths	<string>	Write xml filename to post-end connections with absolute or relative path or not at all	ABSOLUTE, RELATIVE, DONT_WRITE	DONT_WRITE
connection_save_settings	<logical>	TRUE to save current settings with connection during creation	TRUE, FALSE	TRUE
maximum_washer _diameter_for_bolts	<real>	default max washer diameter for bolts (expressed in mm)		20.0
adhesive				
adhesive_solid_percentage	<real>	Max percentage of solids created for adhesive. Anything less is invalid	0 - 100	50.0
connection_patch_cohesive	<logical>	Flag to use the cohesive nodal order for 6 noded solid elements	TRUE, FALSE	FALSE
autoweld				
autoweld_diff_seam _proximity	<real>	Proximity check for different autoweld seams	0 - 1	0.5
autoweld_same_seam _proximity	<real>	Proximity check for same autoweld seams	0 - 1	0.5
bolts				
use_parent_layer_for _bolts	<logical>	bolt connection items put into parent layer where possible	TRUE, FALSE	TRUE
strict_layer_method_for _bolts	<logical>	abort bolt creation if include range undefined	TRUE, FALSE	FALSE
bolt_entity_numbering_rule	<string>	layer rule for numbering of bolt FE (over-ruling setting)	CONNECTION _LABEL_RULE, LAYER_FIRST_FREE, LAYER_HIGHEST_FREE, LAYER_HIGHEST_PLUS ONE	CONNECTION _LABEL_RULE
adjust_bolt_mass_on_create	<logical>	on bolt creation add mass for stability if needed (by creating PART_INERTIA)	TRUE, FALSE	FALSE
bolt_feature_line_hole	<logical>	consider feature lines as potential hole edges for bolt creation	TRUE, FALSE	FALSE
add_database_history_beam _for_bolts	<logical>	for beam type bolts create DTHB on bolt create/remake	TRUE, FALSE	FALSE
use_zero_length_discrete _beam_for_bolts	<logical>	use zero length discrete beam for bolts	TRUE, FALSE	TRUE

use_element_beam _thickness_for_zero_length _discrete_beam_bolts	<logical>	use element beam thickness for bolts with zero length beams	TRUE, FALSE	TRUE
nrb_bolts_may_use_set_collect	<logical>	use set collect for NRB bolts that span includes	TRUE, FALSE	TRUE
modified_bolt_layer_method	<logical>	make bolt if at least 2 layers can connect	TRUE, FALSE	FALSE
bolt_angle_tolerance	<real>	default shell angle tolerance when creating bolts		30.0
simplify_merge_bolt	<logical>	This setting removes empty rigid master parts from merge type bolts	TRUE, FALSE	TRUE
nrb_bolts_attach_to_existing _nodal_rigid_bodies	<logical>	nrb bolts merge to existing nodal rigid bodies	TRUE, FALSE	TRUE
spotweld				
spot_max_scanlines	<integer>	Max number of lines displayed for spotweld read panel		50
automatically_create _connections_from_welds	<string>	automatically make connections from existing (MAT100) welds without them	ON, OFF	ON
create_connection_popup	<string>	option to process connection popups when reading spotweld connections	ASK, TRUE, FALSE	ASK
spotweldbeam_length_check	<string>	Spotweld beam length checking	ON, OFF	ON
spotweldbeam_min_length	<real>	Spotweld beam minimum length		0.5
spotweldbeam_max_length	<real>	Spotweld beam maximum length		10.0
spotweldbeam_max_total _length	<real>	Spotweld beam maximum total length		20.0
spotweldbeam_panel_check	<string>	Spotweld beam panel checking	ON, OFF	ON
spotweldbeam_max_panels	<integer>	Spotweld beam maximum number of panels		5
spotweldbeam_distance _check	<string>	Spotweld beam pitch checking	ON, OFF	ON
spotweldbeam_min_distance	<real>	Spotweld beam minimum pitch		10.0
spotweld_warpage_check	<string>	solid spotweld warpage checking	ON, OFF	OFF
spotweld_max_warpage	<real>	solid spotweld maximum warpage	0.0 - 180.0	20
spotweldbeam_pid	<logical>	TRUE if _PID option is to be used when creating spotweld beams	TRUE, FALSE	TRUE
solid_spotweld_consider _free_edges	<string>	consider free edges of panels to align solid welds on create/remake	ON, OFF	ON
solid_spotweld_edge _search_dist	<real>	Search distance for finding nearby free edges to align to		50.0
solid_spotweld_rotate _edge_align	<string>	align a flat edge of solid spotwelds with closest free edge of attaching panels	ON, OFF	OFF
solid_spotweld_align_feature	<string>	align to nearby feature lines as well as free edges	ON, OFF	OFF
solid_spotweld_align _feature_angle	<real>	Break angle to define feature line to align to		20.0

solid_spotweld_ignore _inner_align	<string>	ignore inner layers of panels being joined together when aligning to free edges	ON, OFF	OFF
connection_check_thickness _change	<string>	warn if a section change may mandate change of attached spotweld material properties	ON, OFF	OFF
warn_of_connection_attached _to_shell_on_edge	<string>	warn if spotweld connection node attaches to shell with free edge	ON, OFF	OFF
allow_mig_weld_to _feature_line	<logical>	TRUE if a MIG welds can be created on feature lines as well and free edges	TRUE, FALSE	FALSE
additional_valid_spotweld _material_types	<string>	Addtional material types considered valid for PRIMER spotwelds (e.g. '*MAT_003, *MAT_240')		<none>
spotweld_remesh_max_factor	<real>	Factor for maximum mesh size when remeshing spotwelds	1.0 - 2.0	1.2
spotweld_remesh_min_factor	<real>	Factor for minimum mesh size when remeshing spotwelds	0.0 - 1.0	0.8
spotweld_remesh_corner_angle	<real>	Minimum angle for corners on free edges to be preserved during spotweld remeshing	0.0 - 180.0	10.0
spotweld_lines				
spot_line_search_tolerance	<real>	Search tolerance for finding new starting free edge nodes for edge locked spotweld lines		15.0
contact				
contact_spr_initial_setting	<string>	Initial setting for SPR field of contacts when creating	ON, OFF	ON
contact_mpr_initial_setting	<string>	Initial setting for MPR field of contacts when creating	ON, OFF	ON
contact_check_shells_for _thinning	<string>	report shells where LS-Dyna contact thickness less than 90% of actual	ON, OFF	OFF
contact_check_master _slave_contents	<string>	check master and slave side of contact for contents	ON, OFF	ON
contact_stiffness_check	<string>	Check for contact stiffness mismatch	ON, OFF	OFF
max_allowable_contact _stiffness_ratio	<real>	Max allowable ratio of contact stiffnesses		100.0
contact_check_mode	<string>	use MPP or SMP treatment for contact analysis	MPP, SMP	MPP
contact_penchk_dup_shells	<string>	Allocation of contact segments to duplicate shells	AUTOMATIC, THINNEST, THICKEST	AUTOMATIC
sliding_contact_checking				
contact_mpp_penetration _threshold	<real>	if set to zero use min(0.001, SMP penetration threshold) for report		0.0
report_crossed_3d_elems	<string>	report crossed edges for 3d elements	ON, OFF	ON
contact_penetration_checks	<string>	Contact penetration checking	ON, OFF	OFF
contact_penetration_max _allowable_value	<real>	allowable penetration expressed as value		0.0

contact_penetration_min_remaining_depth	<real>	allowable penetration expressed as remaining segment depth		0.0
contact_penetration_min_remaining_depth_factor	<real>	allowable penetration expressed as remaining factor on segment depth		0.0
contact_ignore_pen_on_same_rigid_body	<logical>	if node and segment on same rigid body ignore the penetration	TRUE, FALSE	true
tied_contact_checking				
contact_check_constrained_clash	<string>	check for segment clashes between constrained contacts	ON, OFF	ON
contact_check_all_connection_nodes_tie	<string>	all nodes of connections must tie	ON, OFF	ON
contact_check_tied	<string>	check that all tied contacts tie at least one node	ON, OFF	OFF
contact_check_all_slave_nodes_by_part_tie	<string>	all slave nodes defined by part must tie	ON, OFF	OFF
contact_check_all_slave_nodes_by_node_set_tie	<string>	all slave nodes defined by node set must tie	ON, OFF	OFF
contact_check_all_slave_nodes_on_shell_edge_tie	<string>	all slave nodes on shell edge must tie	ON, OFF	OFF
contact_check_all_slave_nodes_on_solid_face_tie	<string>	all slave nodes on exterior faces of solids must tie	ON, OFF	OFF
contact_allow_spotweld_offset_option	<logical>	allow *CONTACT_SPOTWELD_OFFSET (though Dyna manual prohibits it)	TRUE, FALSE	FALSE
constructed_properties				
construct_material_card_for_latent	<string>	Construct material card for latent materials in model	ON, OFF	OFF
constructed_material_card_density	<real>	Default density for material cards constructed by Primer		7.85e-9
constructed_material_card_stiffness	<real>	Default stiffness for material cards constructed by Primer		200000
constructed_material_card_spring_stiffness	<real>	Default stiffness for spring material cards constructed by Primer		1.0
constructed_section_card_thickness	<real>	Default thickness for section shell cards constructed by Primer		1.0

The drive mappings allow PRIMER to convert equivalent folder names from Windows to Unix and visa versa

Preference	Type	Description	Valid arguments	Default
drive_a	<string>	Mapping from Windows drive A: to unix path		<none>
drive_b	<string>	Mapping from Windows drive B: to unix path		<none>
drive_c	<string>	Mapping from Windows drive C: to unix path		<none>
drive_d	<string>	Mapping from Windows drive D: to unix path		<none>
drive_e	<string>	Mapping from Windows drive E: to unix path		<none>
drive_f	<string>	Mapping from Windows drive F: to unix path		<none>
drive_g	<string>	Mapping from Windows drive G: to unix path		<none>
drive_h	<string>	Mapping from Windows drive H: to unix path		<none>
drive_i	<string>	Mapping from Windows drive I: to unix path		<none>
drive_j	<string>	Mapping from Windows drive J: to unix path		<none>
drive_k	<string>	Mapping from Windows drive K: to unix path		<none>
drive_l	<string>	Mapping from Windows drive L: to unix path		<none>
drive_m	<string>	Mapping from Windows drive M: to unix path		<none>
drive_n	<string>	Mapping from Windows drive N: to unix path		<none>
drive_o	<string>	Mapping from Windows drive O: to unix path		<none>

drive_p	<string>	Mapping from Windows drive P: to unix path		<none>
drive_q	<string>	Mapping from Windows drive Q: to unix path		<none>
drive_r	<string>	Mapping from Windows drive R: to unix path		<none>
drive_s	<string>	Mapping from Windows drive S: to unix path		<none>
drive_t	<string>	Mapping from Windows drive T: to unix path		<none>
drive_u	<string>	Mapping from Windows drive U: to unix path		<none>
drive_v	<string>	Mapping from Windows drive V: to unix path		<none>
drive_w	<string>	Mapping from Windows drive W: to unix path		<none>
drive_x	<string>	Mapping from Windows drive X: to unix path		<none>
drive_y	<string>	Mapping from Windows drive Y: to unix path		<none>
drive_z	<string>	Mapping from Windows drive Z: to unix path		<none>

Dynamic label settings.

Preference	Type	Description	Valid arguments	Default
dynamic_label_format	<string>	Number format type for dynamic labels	AUTO, SCIENTIFIC, GENERAL	AUTO
dynamic_label_dec_places	<integer>	Number of decimal places to display on dynamic labels	0 - 9	3
edit_panels				
edit_modify_mode	<logical>	Start edit panel in modify mode if 1 or more items exist	TRUE, FALSE	TRUE
modify_single_item	<logical>	In modify mode automatically edit the item if only one item exists in the active models	TRUE, FALSE	FALSE
remember_values_on_element_edit_panel	<string>	option to 'remember' integer, float values on edit panel	ASK, TRUE, FALSE	ASK
edit_find_option	<string>	default on the Only/Find button chosen when opening an edit panel	ONLY, FIND, BLANK, UNBLANK	ONLY
edit_panel_numbering_rule	<string>	default rule for numbering of created entities in edit panels	HIGHEST_PLUS_ONE, LABEL_FIRST_FREE, LABEL_FIRST_LATENT, LAYER_FIRST_FREE, LAYER_HIGHEST_FREE, LAYER_HIGHEST_PLUS_ONE	LAYER_HIGHEST_PLUS_ONE
element_seatbelt_accelerometer				
element_seatbelt_accelerometer_intopt	<string>	Used for setting INTOPT = 1 on *Element_seatbelt_accelerometer card.	ON, OFF	OFF

The following options control many aspects of image appearance. The **`overlay_<xxx>`** options affect only the hidden-line overlay on shaded and all data-bearing plots. They have no effect on wireframe (**LINE**) or hidden-line plots.

Preference	Type	Description	Valid arguments	Default
background_mode	<string>	Draw background using SOLID or FADED colours	SOLID, FADED	SOLID
background_colour	<string>	Background colour	WHITE, GREY, BLACK, RED, GREEN, BLUE, CYAN, MAGENTA, YELLOW, AUTO	BLACK
background_bottom_colour	<string>	Background bottom colour for Faded backgrounds	WHITE, GREY, BLACK, RED, GREEN, BLUE, CYAN, MAGENTA, YELLOW, AUTO	AUTO
backing_store	<string>	Backing store refresh switch and method	OFF, ON, PIXMAP, PBUFFER	ON
beam_min_size	<integer>	Screen length at which to swap to blob beam symbols	1 - 100	25
beam_blob_diameter	<integer>	Beam blob diameter	1 - 100	15
contact_shaded_display	<string>	How contact segments are rendered in shaded display mode	SOLID_HATCHED, SOLID, STIPPLE_1, STIPPLE_2, STIPPLE_4, STIPPLE_8, STIPPLE_16	STIPPLE_1

contact_colour	<string>	Colour used to draw contact surfaces	DEFAULT, BY_SIDE, WHITE, GREY, BLACK, RED, GREEN, BLUE, CYAN, MAGENTA, YELLOW	DEFAULT
contour_text_size	<integer>	Contour bar text size	0 - 500	54
contour_levels	<integer>	Number of contour (CT/SI/VEC) levels	2 - 13	6
curve_white_background	<logical>	When TRUE, set the background of DEFINE_CURVE/TABLE graphs to white	TRUE, FALSE	FALSE
date_size	<integer>	Date (clock) size	0 - 500	30
feature_angle	<real>	Feature line angle		60.0
force_back_buffer	<string>	Whether to draw always to back buffer (for remote rendering)	OFF, ON	OFF
graphics_refresh	<string>	Refresh graphics window when exposed	OFF, ON	ON
geometry_surface_refinement	<integer>	Number of iterations to refine the tessellation of a geometry surface	0 - 10	2
geometry_refinement _distance_tolerance	<real>	Distance tolerance of tessellation to actual geometry surface	1E-6 - 1.0	1E-5
geometry_refinement _angle_tolerance	<real>	Dot product tolerance between two trias on a geometry surface tessellation	0.0 - 1.0	0.999
graphics_type	<string>	Graphics format to start Primer with	X8, X24, X, Opengl, Default	<none>
graticule_active	<string>	Turns graticule ON or OFF	OFF, ON	OFF
graticule_decimal_places	<integer>	Number of decimal places for graticule numbers	0 - 9	3
graticule_display_numbers	<string>	When ON, graticule numbers sre displayed.	OFF, ON	ON
graticule_exponent	<integer>	Exponent for graticule number format	-99 - 99	3
graticule_line_colour	<string>	Graticule line colour	WHITE, GREY, BLACK, RED, GREEN, BLUE, CYAN, MAGENTA, YELLOW, RED/MAGENTA, YELLOW/ORANGE, YELLOW/GREEN, GREEN/CYAN, CYAN/BLUE, RED/ORANGE, LIGHT_BLUE, NOT_BACKGROUND, TEXT	TEXT
graticule_number_format	<string>	Format for graticule numbers	AUTOMATIC, MANUAL, SCIENTIFIC, GENERAL	AUTOMATIC
graticule_plane_colour	<string>	Graticule plane colour	WHITE, GREY, BLACK, RED, GREEN, BLUE, CYAN, MAGENTA, YELLOW, RED/MAGENTA, YELLOW/ORANGE, YELLOW/GREEN, GREEN/CYAN, CYAN/BLUE, RED/ORANGE, LIGHT_BLUE	GREY
graticule_show_grid	<string>	Shows the graticule grid	OFF, ON	OFF
graticule_text_colour	<string>	Graticule text colour	WHITE, GREY, BLACK, RED, GREEN, BLUE, CYAN, MAGENTA, YELLOW, RED/MAGENTA, YELLOW/ORANGE, YELLOW/GREEN, GREEN/CYAN, CYAN/BLUE, RED/ORANGE, LIGHT_BLUE, NOT_BACKGROUND, TEXT	TEXT
graticule_text_size	<integer>	Graticule text size	0 - 500	25

graticule_transparency	<integer>	Transparency for graticule	0 - 100	100
image_format	<string>	Default image format	BMP_8_C, BMP_8_UN, PNG_8, GIF_8, BMP_24_UN, PNG_24, JPG_24, PPM_24	JPG_24
initial_plot_mode	<string>	Initial drawing mode	LINE, HIDDEN, SHADED	SHADED
intel_hd_use_shaders	<string>	Control usage of hardware shaders on Intel HD graphics cards	AUTO_DETECT, FORCE_OFF, FORCE_ON	AUTO_DETECT
label_size	<integer>	Label size	0 - 500	30
lumped_mass_size	<integer>	Lumped mass symbol size	1 - 100	25
lumped_mass_symbol	<string>	Lumped mass symbol type	square, cube, automatic	automatic
mesh_fixed_point_size	<integer>	Size to draw fixed points at when (re)meshing	1 - 100	10
overlay_colour	<string>	Overlay colour	WHITE, GREY, BLACK, RED, GREEN, BLUE, CYAN, MAGENTA, YELLOW, RED/MAGENTA, YELLOW/ORANGE, YELLOW/GREEN, GREEN/CYAN, CYAN/BLUE, RED/ORANGE, LIGHT_BLUE, ELEMENT	GREY
overlay_mode	<string>	Overlay drawn	OFF, FREE, FREE_IGN_PART, FEATURE, ALL	FREE
overlay_line_width	<integer>	Width of overlay lines	1 - 10	1
placement	<string>	Location for initial window on multi-screen display	LEFT, RIGHT, BOTTOM, TOP, LEFT_BOTTOM, LEFT_TOP, RIGHT_BOTTOM, RIGHT_TOP	<none>
plot_border	<string>	Border drawn on plot	OFF, ON	ON
plot_date	<string>	Date drawn on plot	OFF, ON	OFF
plot_model_names	<string>	Model names drawn on plot	OFF, ON	ON
plot_triad	<string>	Triad for coordinate axes drawn on plot	OFF, ON	OFF
recursive_blanking	<string>	How blanking propagates recursively	NO_RECURSION, DRAWABLE_ONLY, UNCONDITIONAL	NO_RECURSION
seatbelt_size	<integer>	Notional seatbelt size	1 - 1000	50
segment_hatching	<string>	Contact and other segment hatching	OFF, ON	ON
shell_graphics_mode	<string>	Drawing method for shell elements, flat and smooth use true thickness	THIN, FLAT, SMOOTH	THIN
shell_normal_length	<real>	Multiplier for determining the length of shell normals.		0.01
sketch_colour	<string>	Sketch colour	WHITE, GREY, BLACK, RED, GREEN, BLUE, CYAN, MAGENTA, YELLOW	WHITE
spotweld_size	<integer>	Spotweld beam graphics size	1 - 100	40
spring_size	<integer>	Spring beam graphics size	0 - 500	100
swap_nodal_coords	<string>	Whether to swap ordinary and reference nodal corodinates	ALWAYS, ASK	ASK
swap_shell_topology	<string>	Whether to swap ordinary and airbag reference shell topology	ALWAYS, ASK	ASK
target_size	<integer>	Target graphics size	0 - 100	20
text_colour	<string>	Text colour	WHITE, GREY, BLACK, RED, GREEN, BLUE, CYAN, MAGENTA, YELLOW	WHITE
true_beam_sections	<logical>	Whether or not to draw beam elements using true sections	TRUE, FALSE	FALSE

white_background_image	<logical>	Write images with white background	TRUE, FALSE	FALSE

The following control settings and warnings when reading include files

Preference	Type	Description	Valid arguments	Default
auto_offset_user_clash	<string>	Handling of clashes involving *INCLUDE_AUTO_OFFSET_USER	IGNORE, REMOVE_USER	IGNORE

The following control settings and warnings when reading files

Preference	Type	Description	Valid arguments	Default
compressed				
keyin_compress_threaded	<logical>	TRUE if compressed input runs in a separate thread (faster)	TRUE, FALSE	TRUE
keyin_compress_diag_level	<integer>	Diagnostic level for compressed input. (0 = off, 1 to 3 progressively more)	0 - 3	0
keyin_compress_bsize	<integer>	Compression buffer size in kbytes (4 - 16384 permitted, advanced setting)	4 - 16384	1024
keyin_compress_nspin_m	<integer>	Threaded master spin limit (0 - 100000000 permitted, advanced setting)	0 - 100000000	4000
keyin_compress_nspin_s	<integer>	Threaded slave spin limit (0 - 100000000 permitted, advanced setting)	0 - 100000000	4000
keyin_compress_yield_m	<integer>	Threaded master yield threshold in ms (0 - 1000 permitted, advanced setting)	0 - 1000	1
keyin_compress_yield_s	<integer>	Threaded slave yield threshold in ms (0 = 1000 permitted, advanced setting)	0 - 1000	1
keyin_compress_sleep_m	<integer>	Threaded master sleep threshold in ms (0 = 1000 permitted, advanced setting)	0 - 1000	60
keyin_compress_sleep_s	<integer>	Threaded slave sleep threshold in ms (0 = 1000 permitted, advanced setting)	0 - 1000	40
keyin_compress_timeout	<integer>	Threaded timeout threshold in ms (0 = 100000 permitted, advanced setting)	0 - 100000	10000
normal				
keyin_normal_threaded	<logical>	TRUE if normal input runs in a separate thread (faster)	TRUE, FALSE	TRUE
keyin_normal_diag_level	<integer>	Diagnostic level for normal input. (0 = off, 1 to 3 progressively more)	0 - 3	0
keyin_normal_bsize	<integer>	Threaded buffer size in kbytes (4 - 16384 permitted, advanced setting)	4 - 16384	1024
keyin_normal_nspin_m	<integer>	Threaded master spin limit (0 - 100000000 permitted, advanced setting)	0 - 100000000	4000
keyin_normal_nspin_s	<integer>	Threaded slave spin limit (0 - 100000000 permitted, advanced setting)	0 - 100000000	4000
keyin_normal_yield_m	<integer>	Threaded master yield threshold in ms (0 - 1000 permitted, advanced setting)	0 - 1000	1
keyin_normal_yield_s	<integer>	Threaded slave yield threshold in ms (0 = 1000 permitted, advanced setting)	0 - 1000	1
keyin_normal_sleep_m	<integer>	Threaded master sleep threshold in ms (0 = 1000 permitted, advanced setting)	0 - 1000	60
keyin_normal_sleep_s	<integer>	Threaded slave sleep threshold in ms (0 = 1000 permitted, advanced setting)	0 - 1000	40
keyin_normal_timeout	<integer>	Threaded timeout threshold in ms (0 = 100000 permitted, advanced setting)	0 - 100000	10000
additional_dyna_output	<string>	Additional search string for dyna output files		<none>
additional_dyna_output _search	<logical>	Action to search for dyna output files using additional search string (provided search string is not empty)	TRUE, FALSE	FALSE
compressed_dyna_output _search	<logical>	Action to search for compressed (.gz and .zip) dyna output files	TRUE, FALSE	FALSE

convert_implicit_parameter	<logical>	Action to convert implicit parameters in [...] to values	TRUE, FALSE	FALSE
convert_rbe2_cnrb	<logical>	Action to convert RBE2 nastran card to CONSTRAINED NRB	TRUE, FALSE	FALSE
copy_hm_comment_title	<string>	Set material and section titles to HM comment titles if no current title is set	ON, OFF	ON
correct_seatbelt_topology	<string>	Action to correct errors in 1D seatbelt topology on input	AUTOMATIC, MANUAL	MANUAL
duplicated_keyword_warning	<string>	Give 'Stop and acknowledge' style warning whenever duplicated keyword is read	ON, OFF	ON
extensions_for_file_read	<string>	considered extensions for file read (e.g. *.k* *.gz).		*.k* *.dyn *.gz *.zip
find_data_for_scan	<logical>	Find missing parameter and/or include transform data during model scan	TRUE, FALSE	FALSE
force_keyword_format	<string>	Force the size of data fields to be SMALL (8 or 10 columns wide) or LARGE (20 columns wide)	SMALL, LARGE	SMALL
ignore_lspp_comments	<logical>	Ignore comment lines from LSPP starting '$#' when storing comments	TRUE, FALSE	TRUE
inherit_file_format	<logical>	Include file inherits large/small keyword format of parent if no explicit size defined	TRUE, FALSE	FALSE
input_buffer_size	<integer>	File buffer size for reading ascii files	2 - 2147483646	4096
input_echo_frequency	<integer>	Progress echo interval when reading ascii files	1 - 2147483646	1000
input_exception_BOUNDARY	<string>	Allow 'excess' input *BOUNDARY cards to be skipped or ignored. READ: all cards are read and stored as normal; SKIP: cards are not read into memory but are written to a 'skip' file; IGNORE: cards are neither read nor written to a 'skip' file	READ, SKIP, IGNORE	READ
input_exception_INITIAL	<string>	Allow 'excess' input *INITIAL cards to be skipped or ignored. READ: all cards are read and stored as normal; SKIP: cards are not read into memory but are written to a 'skip' file; IGNORE: cards are neither read nor written to a 'skip' file	READ, SKIP, IGNORE	READ
input_exception_LOAD	<string>	Allow 'excess' input *LOAD cards to be skipped or ignored. READ: all cards are read and stored as normal; SKIP: cards are not read into memory but are written to a 'skip' file; IGNORE: cards are neither read nor written to a 'skip' file	READ, SKIP, IGNORE	READ
keyin_filename_syntax	<string>	For include files (and master if not found) permute filename with .gz, .zip, .kby and find newest version on disk	NEWEST, VERBATIM	NEWEST
large_format_80cols	<logical>	Wrap large format at 80 columns	TRUE, FALSE	FALSE
preread_parameters	<logical>	Enable pre-read scan of input deck for *PARAMETER definitions	TRUE, FALSE	FALSE
read_ansa_comments	<string>	read ANSA comments upon input (comments used to construct assemblies)	ON, OFF	ON
read_despite_error_warning	<string>	Give 'Stop and acknowledge' style warning when file read succeeds despite errors	ON, OFF	ON
read_duplicates	<string>	How duplicate labelled items are handled during keyword input	ALL, NONE, LS971, R9, NODE	R9
read_embedded_comments	<logical>	Read and store embedded comments in the keyword file	TRUE, FALSE	TRUE
read_hm_comments	<string>	read HM comments upon input (comments used to construct assemblies, colours and in some cases, titles)	ON, OFF	ON
read_missing_include_file	<logical>	set to TRUE/FALSE to enable/disable the 'Missing include file browse panel'	TRUE, FALSE	TRUE
save_read_log_dir	<string>	Directory in which a copy of the PRIMER keyword read log, primer_readlog.txt, is written		JOBDIR

single_card_ABAG	<logical>	Ignore 'extra' data lines when reading *ABAG headers	TRUE, FALSE	FALSE
single_card_EQOS	<logical>	Ignore 'extra' data lines when reading *EQOS headers	TRUE, FALSE	FALSE
single_card_HGLS	<logical>	Ignore 'extra' data lines when reading *HGLS headers	TRUE, FALSE	FALSE
single_card_MATL	<logical>	Ignore 'extra' data lines when reading *MATL headers	TRUE, FALSE	FALSE
skip_severe_errors	<logical>	Whether PRIMER should skip severe errors or not.	TRUE, FALSE	FALSE
suppress_text_box_messages_on_keyin	<string>	suppress all text box messages on keyin (just write to log)	ON, OFF	OFF
warn_num_implicit_parameters	<integer>	Number of [...] implicit parameters to trigger warning, zero to turn off	0 - 2000000000	100000
warn_parameter_order	<logical>	Warn if parameters in INCLUDE_TRANSFORM used before being defined	TRUE, FALSE	TRUE
zero_field_spillover	<logical>	Allow zero field spillover when reading keyword files	TRUE, FALSE	FALSE
zero_volume_warning	<logical>	Warn about shells of zero area or solids of zero volume	TRUE, FALSE	TRUE

These settings control the attributes and behaviour of Primer's IPP model build tool.

Preference	Type	Description	Valid arguments	Default
ipp_replacement_impactor	<string>	set filename substitute user defined impactor on completion of IPP build		None
ipp_positioning_method	<string>	control positioning bias of impactor at 'difficult' points	bias aim point to target, bias contact point to target, combined	combined

These settings control the attributes and behaviour of Primer's generic Keyword editor.

Preference	Type	Description	Valid arguments	Default
kw_edit_init_defs	<integer>	Number of definitions to show initially	1 - 100	5
kw_edit_init_rows	<integer>	Maximum number of rows to show initially	1 - 100	10

The following control element labelling

Preference	Type	Description	Valid arguments	Default
clabel_case_sensitive	<string>	Whether or not character labels are case sensitive	KEYIN_ONLY, NEVER, ALWAYS	KEYIN_ONLY
clabel_syntax	<string>	Syntax permitted for character labels	VERY_LOOSE, LOOSE, RATIONAL	VERY_LOOSE
label_warning	<logical>	Display a warning if the maximum number of labels is reached	TRUE, FALSE	TRUE
max_labels	<integer>	Maximum number of labels to display	1 - 2147483646	1000
NODE_labelled	<string>	Nodes labelled	ON, OFF	OFF
SOLID_labelled	<string>	Solids labelled	ON, OFF	OFF
BEAM_labelled	<string>	Beams labelled	ON, OFF	OFF
SHELL_labelled	<string>	Shells labelled	ON, OFF	OFF
TSHELL_labelled	<string>	Thick shells labelled	ON, OFF	OFF
DISCRETE_labelled	<string>	Springs/dampers labelled	ON, OFF	OFF
MASS_labelled	<string>	Lumped masses labelled	ON, OFF	OFF
keyout_large_warning	<logical>	Display a warning if the keyout format is changed from small to large	TRUE, FALSE	TRUE

The following strings and values control laser plotting setup

Preference	Type	Description	Valid arguments	Default
laser_paper_size	<string>	Default paper size	US, A4	A4
laser_orientation	<string>	Default page orientation	Portrait, Landscape	Landscape
laser_mode	<string>	Default laser mode	Colour, Greyscale	Greyscale
laser_insert_file	<string>	Valid filename		<none>
laser_top_margin	<real>	Top margin size in mm		10
laser_bottom_margin	<real>	Bottom margin size in mm		30

laser_left_margin	<real>	Left margin size in mm		20
laser_right_margin	<real>	Right margin size in mm		10

The following control overall window layout, organisation and behaviour

Preference	Type	Description	Valid arguments	Default
auto_minimise	<string>	If an edit panel obscures the graphics window, it will automatically minimise when the mouse moves off the panel	OFF, ON, PICK, ALWAYS	OFF
existing_panel_action	<string>	Action for existing panels when new one mapped	NONE, IN_SITU, TIDY	NONE
keep_rhs_lowered	<logical>	(Superseded) Keep all of docked area on Right Hand Side lowered in stacking order	TRUE, FALSE	FALSE
keep_rhs_top_lowered	<logical>	Keep top half of docked area on Right Hand Side lowered in stacking order	TRUE, FALSE	TRUE
keep_rhs_bottom_lowered	<logical>	Keep bottom half of docked area on Right Hand Side lowered in stacking order	TRUE, FALSE	FALSE
keep_gbox_lowered	<logical>	Keep graphics box lowered in stacking order	TRUE, FALSE	TRUE
maximise	<logical>	Maximise window when PRIMER started	TRUE, FALSE	FALSE
panel_placement	<string>	Where new floating panels are located	LEFT, R_BORD, RIGHT, TOP, B_BORD, BOTTOM, FREE	FREE
reset_layout	<logical>	Reset standard layout after change in master window size or shape	TRUE, FALSE	TRUE
rhs_initial_keywords_state	<string>	Initial appearance of Keywords button area	EXPANDED, CONTRACTED	EXPANDED
rhs_initial_tools_state	<string>	Initial appearance of Tools button area	EXPANDED, CONTRACTED	EXPANDED
rhs_number_columns	<integer>	Number of columns of Tools, Keywords and Tabs buttons	3 - 100	4
scale_edit_panels	<logical>	Whether or not to scale editing panels horizontally to show large labels	TRUE, FALSE	TRUE
scale_kw_editor	<logical>	Whether or not to scale the KW editor horizontally to show large labels	TRUE, FALSE	TRUE

The following options control image lighting

Preference	Type	Description	Valid arguments	Default
shaded_ambient	<real>	Percentage ambient light (0-100)	0.0 - 100.0	30
shaded_diffuse	<real>	Percentage diffuse brightness (0-100)	0.0 - 100.0	70
shaded_shininess	<real>	Percentage specular brightness (0-100)	0.0 - 100.0	70
shaded_saturation	<real>	Percentage colour saturation (0-100)	0.0 - 100.0	50

The following options control display of local material directions

Preference	Type	Description	Valid arguments	Default
locaxes_type	<string>	Set size of triad as a function of raw or screen coordinates, or as a function of average element length	SCREEN_SPACE, MODEL_SPACE, AVG_EL_LENGTH	SCREEN_SPACE
locaxes_size	<real>	Triad size		75
locaxes_scale	<real>	Triad scale factor		0.2
locaxes_colour	<string>	Display triads for composite parts	DEFAULT, MTL_COLOUR	DEFAULT

The following options relate to macros

Preference	Type	Description	Valid arguments	Default
macro_auto_record	<string>	Filename to automatically record macro to		<none>
macro_directory	<string>	Directory in which PRIMER looks for macros		$OA_INSTALL/primer_library/macros

macro_echo_to_dialogue	<string>	Echo of recorded macro commands to dialogue box	ON, OFF	ON
macro_echo_to_terminal	<string>	Echo of recorded macro commands to terminal window	ON, OFF	OFF

The following options relate to mass display

Preference	Type	Description	Valid arguments	Default
include_mass_for_2d_seatbelt_elements	<string>	include mass of 2d seatbelt elements in the part mass	ON, OFF	ON
information_mass_display	<string>	switch to turn off mass information in the quick-pick information panel	OFF, ON	ON

The following options affect the properties and behaviour of materials (see section 5 Materials)

Preference	Type	Description	Valid arguments	Default
mat_user_ref_curves	<string>	Whether and how MAT_USER definitions reference load-curves	NONE, IFILE_ONLY, ALL	IFILE_ONLY
enable_integers_for_mat_user_defined	<string>	For MAT_USER (MAT_041) write integers for U_1.., P_1.. where rounding permits	ON, OFF	OFF
mat_etan_max	<real>	MAX Strain for curves with ETAN option		1.0
mat_database_sort_alphabetical	<string>	Switch to sort materials in alphabetical order when importing from a database	FALSE, TRUE	FALSE
mat_database_import_title	<string>	Switch to import material titles when importing from a database	FALSE, TRUE	FALSE

These settings apply to mechanisms and dummies

Preference	Type	Description	Valid arguments	Default
mechanism_joint_check	<string>	Coincidence check for nodal pairs in joints	NO_CHECK, SILENT, PROMPT	PROMPT
mechanism_accuracy	<real>	Default calculation accuracy (0.1 - 100.0)	0.1 - 100.0	1.0

The following options affect the appearance and behaviour of the graphical user interface (see section 2.4.4.1), left handed support, and the mouse

Preference	Type	Description	Valid arguments	Default
display_factor	<real>	Factor on display size (0.5 - 2.0, automatic if undefined)	0.5 - 2.0	1.2
display_brightness	<real>	Menu brightness (0.0-1.0)	0.0 - 1.0	1.0
display_saturation	<real>	Menu colour saturation (0.0-1.0)	0.0 - 1.0	1.0
button_gradation	<real>	Button shade gradation (0.0-1.0)	0.0 - 1.0	0.0
dv_sync_windows	<string>	Dyn view method(s) for synchronising windows	ICON, ICON+CAPS, ICON+NUM, ICON+CAPS+NUM	ICON+CAPS
dv_left_shift	<string>	Dyn view action for shift + Left mouse	ROTATION_XYZ, ROTATION_XY, ROTATION_Z, ROTATION_SPHERE, TRANSLATION, ZOOM_UP_+VE, ZOOM_DOWN_+VE, UNUSED	ROTATION_XYZ
dv_middle_shift	<string>	Dyn view action for shift + Middle mouse	ROTATION_XYZ, ROTATION_XY, ROTATION_Z, ROTATION_SPHERE, TRANSLATION, ZOOM_UP_+VE, ZOOM_DOWN_+VE, UNUSED	TRANSLATION
dv_right_shift	<string>	Dyn view action for shift + Right mouse	ROTATION_XYZ, ROTATION_XY, ROTATION_Z, ROTATION_SPHERE, TRANSLATION, ZOOM_UP_+VE, ZOOM_DOWN_+VE, UNUSED	ZOOM_UP_+VE

dv_left_ctrl	<string>	Dyn view action for ctrl + Left mouse	ROTATION_XYZ, ROTATION_XY, ROTATION_Z, ROTATION_SPHERE, TRANSLATION, ZOOM_UP_+VE, ZOOM_DOWN_+VE, UNUSED	ROTATION_XYZ
dv_middle_ctrl	<string>	Dyn view action for ctrl + Middle mouse	ROTATION_XYZ, ROTATION_XY, ROTATION_Z, ROTATION_SPHERE, TRANSLATION, ZOOM_UP_+VE, ZOOM_DOWN_+VE, UNUSED	TRANSLATION
dv_right_ctrl	<string>	Dyn view action for ctrl + Right mouse	ROTATION_XYZ, ROTATION_XY, ROTATION_Z, ROTATION_SPHERE, TRANSLATION, ZOOM_UP_+VE, ZOOM_DOWN_+VE, UNUSED	ZOOM_UP_+VE
dv_left_both	<string>	Dyn view action for shift+ctrl + Left mouse	ROTATION_XYZ, ROTATION_XY, ROTATION_Z, ROTATION_SPHERE, TRANSLATION, ZOOM_UP_+VE, ZOOM_DOWN_+VE, UNUSED	ROTATION_XYZ
dv_middle_both	<string>	Dyn view action for shift+ctrl + Middle mouse	ROTATION_XYZ, ROTATION_XY, ROTATION_Z, ROTATION_SPHERE, TRANSLATION, ZOOM_UP_+VE, ZOOM_DOWN_+VE, UNUSED	TRANSLATION
dv_right_both	<string>	Dyn view action for shift+ctrl + Right mouse	ROTATION_XYZ, ROTATION_XY, ROTATION_Z, ROTATION_SPHERE, TRANSLATION, ZOOM_UP_+VE, ZOOM_DOWN_+VE, UNUSED	ZOOM_UP_+VE
dv_shift_action	<string>	Dynamic viewing mode for shift + mouse button	CURRENT, WIREFRAME, FREE_EDGE, UNUSED	CURRENT
dv_ctrl_action	<string>	Dynamic viewing mode for ctrl + mouse button	CURRENT, WIREFRAME, FREE_EDGE, UNUSED	WIREFRAME
dv_both_action	<string>	Dynamic viewing mode for shift+ctrl + mouse button	CURRENT, WIREFRAME, FREE_EDGE, UNUSED	FREE_EDGE
font_scaling	<logical>	whether text in GUI buttons can be scaled down to fit	TRUE, FALSE	TRUE
font_silent	<logical>	whether to write explanatory text if wanted fonts are not found	TRUE, FALSE	FALSE
font_size	<string>	Menu font size	TINY, SMALL, DEFAULT, LARGE, HUGE	DEFAULT
font_type	<string>	Menu font typeface and strength	HELVETICA, HELVETICA-BOLD, TIMES, TIMES-BOLD, COURIER, COURIER-BOLD	HELVETICA
left_handed	<string>	Left handed switching of mouse and/or keyboard	NONE, MOUSE, KEYBOARD, ALL	NONE
zoom_factor	<real>	Zoom Factor for mouse wheel (0.01-1.0)	0.01 - 1.0	0.05
czoom_factor	<real>	Factor for right mouse dynamic zoom (0.01-0.2)	0.01 - 0.2	0.05
kzoom_factor	<real>	Factor for +/- keyboard short-cut keys	0.01 - 100.0	2.0
menu_dragging_mode	<string>	Mode used when moving menu panels with the mouse	WIREFRAME, OPAQUE	WIREFRAME
mouse_3d_rotation_factor	<real>	Factor applied to the speed of rotation when using a 3D mouse		1.0
mouse_3d_pan_factor	<real>	Factor applied to the speed of panning when using a 3D mouse		1.0

mouse_3d_zoom_factor	<real>	Factor applied to the speed of zooming when using a 3D mouse		1.0
mouse_action_middle_button	<string>	Set the action for the middle mouse key during picking	APPLY, REJECT, DESELECT	REJECT
mouse_action_right_button	<string>	Set the action for the right mouse key during picking	APPLY, REJECT, DESELECT	REJECT

The following settings control meshing and hole properties

Preference	**Type**	**Description**	**Valid arguments**	**Default**
align_solid_axes_tol_angle	<real>	Angle tolerance for aligning solid elements	0.0 - 90.0	15.0
hole_element_method	<string>	Elements around hole specified by number/size	NUMBER, SIZE	NUMBER
hole_element_number	<integer>	Number of elements around hole	3 - 1000	4
hole_element_size	<real>	Size of elements around hole		10.0
hole_diameter	<real>	Hole diameter		10.0
hole_rotate_angle	<real>	Rotation angle		10.0
remesh_area	<logical>	Remesh area when removing hole	TRUE, FALSE	TRUE
washer	<logical>	Create washer elements around hole	TRUE, FALSE	TRUE
washer_diameter	<real>	Washer diameter		20.0
washer_elements	<integer>	Number of washer elements	1 - 5	1
mesh_element_size	<real>	Element size for shells		5.0
mesh_feature_line	<logical>	Limit mesh area by feature lines	TRUE, FALSE	FALSE
mesh_feature_line_angle	<real>	Mesh feature line angle		20.0
swage_auto_pid	<logical>	PID for Swage Mesh shells be picked automatically	TRUE, FALSE	TRUE
swage_base_width	<real>	BASE width for Swage Mesh		40.0
swage_top_width	<real>	TOP surface width for Swage Mesh		20.0
swage_height	<real>	Height for Swage Mesh		10.0
swage_radius1	<real>	Radius for fillets at the TOP surface for Swage Mesh		0.0
swage_break_angle	<real>	Angle to consider sharp bends for Swage Mesh path		30.0
swage_element_pitch	<real>	Element pitch size for Swage Mesh shells		5.0
swage_top_element_method	<string>	TOP surface shells of the Swage Mesh specified by number/size	NUMBER, SIZE	SIZE
swage_top_element_number	<integer>	Number of shells at the TOP surface of the Swage Mesh	3 - 1000	2
swage_top_element_size	<real>	Size of shells at the TOP surfaces of the Swage Mesh		5.0
swage_side_element_method	<string>	SIDE surface shells of the Swage Mesh specified by number/size	NUMBER, SIZE	SIZE
swage_side_element_number	<integer>	Number of shells at the SIDE surfaces of the Swage Mesh	3 - 1000	2
swage_side_element_size	<real>	Size of shells at the SIDE surfaces of the Swage Mesh		5.0
swage_side_ends_type	<string>	Types of the Ends of the SIDE surface of the Swage Mesh	CHAMFERED, SLANTED, VERTICAL	CHAMFERED
swage_side_mesh_type	<string>	Method to create the SIDE Surface shells of the Swage Mesh	STRUCTURED, FREE	STRUCTURED
swage_remesh_distance	<real>	Distance to remesh the original surface for Swage Mesh		20.0
swage_remesh_break_angle	<real>	Shell normals break angle to identify the Re-mesh surface for the Swage Mesh		20.0
swage_remesh_length	<real>	Element size for the remeshed surface for the Swage Mesh		5.0

The following affect model build

Preference	Type	Description	Valid arguments	Default
set_database_dir	<string>	Default directory for databases when using model database		<none>
set_template_dir	<string>	Default directory for templates when using model database		<none>
model_build_delete_latent_items	<string>	on build remove missing items if possible	ON, OFF	ON
move_include_to_master	<string>	on build move extra data file to master if any delete	OFF, ON, ASK	ASK
extensions_for_database_from_dir	<string>	considered extensions for database from dir		.key;.k;.dyn
build_csv_impact_abs_path	<logical>	build from csv - use full path for impactor, relative for others	TRUE, FALSE	false
write_all_includes	<logical>	Write all includes to model build directories (default is master only)	TRUE, FALSE	FALSE
model_build_load_range_info	<logical>	load model build label range info from .csv file	TRUE, FALSE	TRUE
database_template_block_error	<logical>	halt database/template build process if errors detected	TRUE, FALSE	FALSE
model_build_diffcheck	<string>	Run model modified diff check (to detect edit of models in memory) before commencing build	OFF, ON	OFF

The following control settings when merging models

Preference	Type	Description	Valid arguments	Default
merge_range_location	<string>	Choose the model to take any master file numbering range information from upon model merge	FIRST, SECOND, NEITHER	FIRST
merge_thumb_location	<string>	Choose the model to take any master file thumbnail information from upon model merge	FIRST, SECOND, NEITHER	NEITHER
merge_incl_path_location	<string>	Choose the model to take any master file INCLUDE_PATH information from upon model merge	FIRST, SECOND, NEITHER	FIRST
merge_set_collect	<logical>	Merge clashing *SET_COLLECT cards rather than renumbering them	TRUE, FALSE	FALSE

The following options relate to model modified

Preference	Type	Description	Valid arguments	Default
threshold_for_modified_nodal_coordinates	<real>	Threshold difference for comparing nodal coordinates		0.0
sig_fig_for_modified_floats	<integer>	Number of significant figures for comparing real numbers	1 - 7	6
check_for_modified_header_comments	<string>	Check for differences in (include) header comments	OFF, ON	ON
check_for_modified_kw_comments	<string>	Check for differences in keyword comments	OFF, ON	ON

Options to control setup and processing morphing.

Preference	Type	Description	Valid arguments	Default
edge_morph_point_interpolation	<string>	Interpolation method for morph points on edges of a high order box.	LINEAR_FROM_CORNERS, FROM_SELECTED_EDGE_POINTS	FROM_SELECTED_EDGE_POINTS
interpolated_morph_points	<string>	Morph points on high order boxes whose movement is interpolated while selected points are moved.	NONE, HIDDEN, ALL	HIDDEN

morph_box_min_size	<real>	Minimum side length of morph boxes created. When the bounding box around selected entities is smaller, the morph box corners are moved outside to attain the minimum size.		10.0
morph_interpolation_combination	<string>	Method of combining different linear interpolations for face and internal morph points of a high order box.	SUPERPOSE, AVERAGE	SUPERPOSE

The following control options used during creation of nodal rigid bodies

Preference	Type	Description	Valid arguments	Default
nrb_create_master_node_at_centre	<string>	Create a master node at the centre of an NRB (create at edge of hole / bolt connections)	TRUE, FALSE	TRUE
nrb_pick_create_centre_node	<string>	Create a master node at the centre of an NRB when in the picking nodes mode	TRUE, FALSE	FALSE
nrb_creation_method	<string>	Sets the default creation method for NRBs	PICK_NODES, PICK_WITH_TOL, PICK_CLOSEST_NODES, STANDARD_MENU, EDGE_OF_HOLE	TRUE

The following options relate to node merge/replace

Preference	Type	Description	Valid arguments	Default
replace_shell_reference_geometry	<logical>	If nodes in shell reference geometry will be replaced by node merge/replace	TRUE, FALSE	true
replace_collapse_shell	<logical>	When using the node replace feature, if this setting is on quad shells will be collapsed to tria shells if possible	TRUE, FALSE	true

If a selection menu is not wide enough to display all the contents, it can be expanded automatically by the following (see section 2.4.4.3)

Preference	Type	Description	Valid arguments	Default
menu_expand	<string>	Automatic menu expansion or undocking on/off switch	OFF, ON, EXPAND, UNDOCK	UNDOCK
menu_expand_delay	<real>	Factor on delay time before expansion	0.1 - 5.0	1.0
menu_expand_speed	<real>	Factor on menu expansion speed	0.1 - 5.0	1.0
menu_sketch	<string>	Whether or not to show sketch menu items when cursor hovered over menu row	OFF, ON	ON
menu_label	<string>	Whether or not menu sketching also shows item labels	OFF, ON	ON

These settings apply to object picking

Preference	Type	Description	Valid arguments	Default

area_select	<string>	Whether being inside an area pick is based on element/face centre or any node	CENTRE, NODE	CENTRE
area_through	<string>	Whether all 3d elements inside a mesh, or only external ones, are selected in a screen area pick	ALL, EXTERNAL	ALL
a_through_factor	<real>	Factor on depth testing when using EXTERNAL area through selection mode	0.001 - 1000.0	1.0
predictive_pick	<string>	Whether or not to show what will be picked based on the current cursor position	OFF, ON	ON
predictive_label	<string>	Whether or not predictive picking also shows item labels	OFF, ON	ON
query_ambiguous	<string>	If screen picking is ambiguous, ON will offer the selection menu, OFF will select nearest	OFF, ON	ON

These settings apply to orientation

Preference	Type	Description	Valid arguments	Default
box_rotation_mode	<string>	Controls how boxes are oriented during rotation	AUTO, LOCAL, NO_LOCAL	AUTO
distorted_element_warning	<logical>	Warn when an element is not moved, but its nodes are	TRUE, FALSE	FALSE
elements_in_new_part	<logical>	Move elements to a new part	TRUE, FALSE	TRUE
move_attached_extra_nodes	<logical>	Automatically orient slave nodes of a rigid part which is oriented	TRUE, FALSE	TRUE
move_beam_third_nodes	<logical>	Orient third node of beams (affecting their transverse axes)	TRUE, FALSE	TRUE
move_connection_with_fe	<logical>	Automatically orient connection entity whenever all its FE entities are moved	TRUE, FALSE	TRUE
move_slave_rigid_bodies	<logical>	Orient slave rigid bodies whenever the master in a rigid body merge is moved	TRUE, FALSE	FALSE
move_weld_with_panels	<logical>	Automatically orient a weld whenever all shells it attaches to are oriented	TRUE, FALSE	FALSE
orient_copy_include	<string>	Include file which copied items are put into	SAME, CURRENT, DUPLICATE	SAME
orient_mat_fabric_axes	<logical>	Orient the axes of *MAT_FABRIC cards	TRUE, FALSE	TRUE
orient_reference_geometry	<logical>	Orient airbag reference geometry	TRUE, FALSE	TRUE
propagate_orient	<logical>	Option to orient items cross referenced by the oriented part/node	TRUE, FALSE	FALSE
copy_related_items	<logical>	if FALSE, copy orient will only copy part (with section,etc) elements and nodes	TRUE, FALSE	TRUE
propagate_copy_orient_part	<logical>	Option to copy orient items referenced by a copy-oriented part	TRUE, FALSE	FALSE
propagate_copy_orient_node	<logical>	Option to copy orient items referenced by a copy-oriented node	TRUE, FALSE	FALSE
use_old_sections	<logical>	Reuse existing sections and materials for copied parts	TRUE, FALSE	FALSE
welds_in_new_part	<logical>	Move welds to a new part	TRUE, FALSE	FALSE
clear_selection_on_orient_apply	<logical>	Clear object menu selection on apply of orient	TRUE, FALSE	FALSE

The following control settings when writing files

Preference	Type	Description	Valid arguments	Default
add_dyna_key_file_extension	<logical>	Add file extension to dyna keyword files if unspecified	TRUE, FALSE	TRUE
dyna_key_file_extension_master	<string>	file extension for Dyna master keyword files		.key
dyna_key_file_extension_include	<string>	file extension for Dyna include files		.key
binary				
keyout_binary_kby_extension	<logical>	Output files with binary content given extension .kby	TRUE, FALSE	TRUE
keyout_binary_master_ascii	<logical>	When writing in binary format master file is all ASCII	TRUE, FALSE	FALSE
keyout_binary_top_c_ascii	<logical>	When writing in binary format header and initial cards are ASCII	TRUE, FALSE	TRUE

compressed				
keyout_compress_switch	<string>	If ON, keyword output (ascii or binary) is compressed to .gz or .zip format. KEEP copies input.	OFF, ON, KEEP	KEEP
keyout_compress_format	<string>	Format used to compress files in ON case: individual .gz, individual .zip, package in .zip	GZ, ZIP, ZPACK	GZ
keyout_compress_level	<integer>	Compression level (1 = least and fastest, to 9 = most and slowest)	1 - 9	1
keyout_compress_threaded	<logical>	TRUE if compressed output runs in a separate thread (faster)	TRUE, FALSE	TRUE
keyout_compress_raw_io	<logical>	TRUE if compressed output does not use system level i/o buffering	TRUE, FALSE	TRUE
keyout_compress_diag_level	<integer>	Diagnostic level for compressed output. (0 = off, 1 to 3 progressively more)	0 - 3	0
keyout_compress_bsize	<integer>	Compression buffer size in kbytes (4 - 16384 permitted, advanced setting)	4 - 16384	1024
keyout_compress_nspin_m	<integer>	Threaded master spin limit (0 - 100000000 permitted, advanced setting)	0 - 100000000	4000
keyout_compress_nspin_s	<integer>	Threaded slave spin limit (0 - 100000000 permitted, advanced setting)	0 - 100000000	4000
keyout_compress_yield_m	<integer>	Threaded master yield threshold in ms (0 - 1000 permitted, advanced setting)	0 - 1000	1
keyout_compress_yield_s	<integer>	Threaded slave yield threshold in ms (0 = 1000 permitted, advanced setting)	0 - 1000	1
keyout_compress_sleep_m	<integer>	Threaded master sleep threshold in ms (0 = 1000 permitted, advanced setting)	0 - 1000	60
keyout_compress_sleep_s	<integer>	Threaded slave sleep threshold in ms (0 = 1000 permitted, advanced setting)	0 - 1000	40
keyout_compress_timeout	<integer>	Threaded timeout threshold in ms (0 = 100000 permitted, advanced setting)	0 - 100000	10000
normal				
keyout_normal_threaded	<logical>	TRUE if normal output runs in a separate thread (faster)	TRUE, FALSE	TRUE
keyout_normal_raw_io	<logical>	TRUE if normal output does not use system level i/o buffering	TRUE, FALSE	TRUE
keyout_normal_diag_level	<integer>	Diagnostic level for normal output. (0 = off, 1 to 3 progressively more)	0 - 3	0
keyout_normal_bsize	<integer>	Threaded buffer size in kbytes (4 - 16384 permitted, advanced setting)	4 - 16384	1024
keyout_normal_nspin_m	<integer>	Threaded master spin limit (0 - 100000000 permitted, advanced setting)	0 - 100000000	4000
keyout_normal_nspin_s	<integer>	Threaded slave spin limit (0 - 100000000 permitted, advanced setting)	0 - 100000000	4000

keyout_normal_yield_m	<integer>	Threaded master yield threshold in ms (0 - 1000 permitted, advanced setting)	0 - 1000	1
keyout_normal_yield_s	<integer>	Threaded slave yield threshold in ms (0 = 1000 permitted, advanced setting)	0 - 1000	1
keyout_normal_sleep_m	<integer>	Threaded master sleep threshold in ms (0 = 1000 permitted, advanced setting)	0 - 1000	60
keyout_normal_sleep_s	<integer>	Threaded slave sleep threshold in ms (0 = 1000 permitted, advanced setting)	0 - 1000	40
keyout_normal_timeout	<integer>	Threaded timeout threshold in ms (0 = 100000 permitted, advanced setting)	0 - 100000	10000
ascii_file_format	<string>	Output format for ascii files (use output_os for keyword file)	NATIVE, UNIX	NATIVE
assembly_output_format	<string>	Format for writing post *END assembly data	PRIMER, HYPERMESH, ANSA, CUSTOMER	PRIMER
autocreate_ztf	<logical>	Write out jobname.ztf file for D3PLOT	TRUE, FALSE	FALSE
check_for_clashing_element _and_set_labels_on_keyout	<logical>	Report and offer to fix clashing element and set labels	TRUE, FALSE	TRUE
customer_comment _output_name	<string>	Name that appears on customer comment output button. Limited to 10 characters		customer
directory_name_for_include _keyout	<string>	Directory name for include keyout IN_SUBDIR mode		INCL
dyna_output_version	<string>	The version of dyna used when writing a keyword file	940, 950, 960, 960+, 970v3858, 970v5434, 970v6763, 971R2, 971R3, 971R4, 971R5, 971R6, 971R6.1, R7.0, R7.1, R8.0, R9.0, R10.0, R11.0	R9.0
dyna_output_style	<string>	The floating point format used when writing a keyword file	native, common, rounded	common
dyna_v3_title_output_version	<integer>	LS_DYNA version code that handles _TITLE in Vol 3 keywords	940 - 2147483646	2147483646
emergency_keyout_dir	<string>	Directory for emergency keyout dump following a crash (otherwise default used)		<none>
element_mass_part_970	<logical>	TRUE if *ELEMENT_MASS_PART can be written to a 970 deck	TRUE, FALSE	FALSE
freestanding_pgp_output _location	<string>	Where to write free-standing PGP blocks in a keyword file)	BEFORE_END, TOP_OF_FILE, AFTER_PARAMETERS, AFTER_INCLUDES	BEFORE_END
freestanding_pgp_output _method	<string>	How to write free-standing PGP blocks in a keyword file)	EMBEDDED, AS_INCLUDE	EMBEDDED
include_file_paths	<string>	Write absolute/relative pathnames in INCLUDE statements	ABSOLUTE, RELATIVE	ABSOLUTE
include_file_method	<string>	Default method of writing include files	IN_SUBDIR, IN_SAME_DIR, SELECT_FILES	IN_SUBDIR
include_mass_comment	<string>	Write the mass of an include file as a comment	ON, OFF	OFF
keyword_order_style	<string>	Order of keywords in output	ALPHABETICAL, CLASSIC	CLASSIC
mdumm_keyout_format	<string>	Output format of *BELT, *DUMMY and *MECHANISM keywords	V11, V12, V13, V14, V15, V16, CURRENT	CURRENT

model_mass_comment	<string>	Write the model mass and C of G as a comment	ON, OFF	OFF
omitted_message_limit	<integer>	Limit to per-keyword type messages about omitted data fields	0 - 100	20
output_971R5_control_shell _in_old_format	<logical>	True will keyout *Control_shell in old (971R5.0) format	TRUE, FALSE	FALSE
output_buffer_size	<integer>	File buffer size for writing ascii files	2 - 2147483646	4096
output_echo_frequency	<integer>	Progress echo interval when writing ascii files	1 - 2147483646	1000
output_card_format	<string>	Card format for keyout	SMALL, LARGE, KEEP, AUTO, BINARY, L_80C	SMALL
output_joint_coinc_check _distance	<real>	Critical separation distance (seek help from Oasys ltd)	0.0 - 1.0e37	1.5e-3
output_joint_coinc_check _threshold	<real>	Critical coordinate magnitude (seek help from Oasys ltd)	0.0 - 1.0e37	8190.0
output_os	<string>	Operating system type for include file naming	NATIVE, WINDOWS, UNIX	NATIVE
parameter_in_string	<string>	Processing of parameters in text and titles during keyout	AUTOMATIC, REPLACE_AMPERSAND, INSERT_VALUE, VERBATIM	AUTOMATIC
preselect_master_file	<logical>	pre-select master file for select files keyout	TRUE, FALSE	TRUE
rigidwall_force_transducer _version	<string>	The version of dyna that *RIGIDWALL_FORCE_ TRANSDUCER is supported from	971R6.1, R7.0, R7.1	R7.1
short_matl_name	<string>	Option to write materials as *MAT_NNN	ON, OFF	OFF
skip_undefined_message	<string>	Preference for skipping the writing of undefined item message	ON, OFF	OFF
solid_two_line_output	<logical>	Write out all solids in LS Dyna version 970 and above 2 line format	TRUE, FALSE	FALSE
suppress_keyout_all _connections	<string>	suppress write of all connections	ON, OFF	OFF
suppress_keyout_autocreated _connections	<string>	Don’t write connections created from welds by Primer	ON, OFF	OFF
suppress_keyout_geometry	<string>	Don’t write geometry to keyword file	ON, OFF	OFF
write_data_field_headers	<logical>	Write comment line of data field acronyms to keyword file	TRUE, FALSE	FALSE
write_embedded_comments	<logical>	Write embedded comments to keyword file	TRUE, FALSE	TRUE
write_xref_comments	<logical>	Write cross reference comments to keyword file	TRUE, FALSE	TRUE
write_existing_include_path	<logical>	Write existing *INCLUDE_PATH	TRUE, FALSE	TRUE
write_group_attributes	<logical>	Write group attributes (colours, transparency etc) to keyword file	TRUE, FALSE	FALSE
write_hm_comments	<string>	Write HM comments to keyword file (auto writes if they have been read)	TRUE, FALSE, AUTO	AUTO
write_include_filename _comment	<string>	Option for what is written as a comment at the top of include files	FULL, FILENAME, NONE	FULL
write_include_filename_to _its_own_line	<logical>	unconditionally write filename to its own line	TRUE, FALSE	false
write_latent_includes	<logical>	Suppress reference to unscanned includes	TRUE, FALSE	FALSE
write_parameters_as_values	<logical>	Write parameters as numerical values	TRUE, FALSE	FALSE

write_primer_part_colours	<logical>	Write primer part colour comments to keyword file	TRUE, FALSE	TRUE
write_thumbnails	<logical>	Write any include file thumbnail images to keyword file	TRUE, FALSE	TRUE
write_ansa_comments	<logical>	Write ANSA comments to keyword file	TRUE, FALSE	FALSE

Options to control the behaviour and appearance of the Part Tree.

Preference	**Type**	**Description**	**Valid arguments**	**Default**
ptree_parts_top_level	<logical>	If TRUE parts are always expanded at the top level	TRUE, FALSE	TRUE
part_tree_blank_mode	<string>	Sets the mode for blanking/unblanking/only for includes/assemblies in the part tree	PART, ELEMENT, LEGACY	ELEMENT
part_tree_drag_material	<string>	Take or leave materials referenced by the part card when moving parts in the part tree	TAKE, LEAVE	TAKE
part_tree_drag_section	<string>	Take or leave sections referenced by the part card when moving parts in the part tree	TAKE, LEAVE	TAKE
part_tree_drag_hourglass	<string>	Take or leave hourglass cards referenced by the part card when moving parts in the part tree	TAKE, LEAVE	LEAVE
part_tree_drag_equation_of_state	<string>	Take or leave equation of state cards referenced by the part card when moving parts in the part tree	TAKE, LEAVE	LEAVE
part_tree_drag_other	<string>	Take or leave junior items that reference parts when moving parts in the part tree	TAKE, LEAVE	LEAVE
part_tree_include_hovertext	<string>	If set to ON, hover text will be displayed for the full path of the include in the part tree	ON, OFF	ON
part_tree_show_blanking_status	<string>	If set to ON, the blanking status for includes/parts will be displayed in the part tree	ON, OFF	OFF
part_tree_show_blanking_mode	<string>	The blanking status display on the part tree can either be based on blanked/unblanked parts or elements	PART, ELEMENT	PART
ptree_show_beam	<logical>	If TRUE a Beam category will be included in the tree	TRUE, FALSE	FALSE
ptree_show_box	<logical>	If TRUE a Define box category will be included in the tree	TRUE, FALSE	FALSE
ptree_show_coordinate	<logical>	If TRUE a Define coordinate category will be included in the tree	TRUE, FALSE	FALSE
ptree_show_contact	<logical>	If TRUE a Contact category will be included in the tree	TRUE, FALSE	FALSE
ptree_show_curve	<logical>	If TRUE a Define curve category will be included in the tree	TRUE, FALSE	FALSE
ptree_show_database_history	<logical>	If TRUE a Database history category will be included in the tree	TRUE, FALSE	FALSE
ptree_show_eos	<logical>	If TRUE an Equation of state category will be included in the tree	TRUE, FALSE	FALSE
ptree_show_function	<logical>	If TRUE a Define function category will be included in the tree	TRUE, FALSE	FALSE
ptree_show_group	<logical>	If TRUE a Groups category will be included in the tree	TRUE, FALSE	FALSE
ptree_show_hourglass	<logical>	If TRUE an Hourglass category will be included in the tree	TRUE, FALSE	FALSE
ptree_show_joint	<logical>	If TRUE a Joints category will be included in the tree	TRUE, FALSE	FALSE
ptree_show_mass	<logical>	If TRUE a Mass category will be included in the tree	TRUE, FALSE	FALSE
ptree_show_nodal_rigid_body	<logical>	If TRUE a Nodal RB category will be included in the tree	TRUE, FALSE	FALSE
ptree_show_prescribed_motion	<logical>	If TRUE a Boundary prescribed motion category will be included in the tree	TRUE, FALSE	FALSE
ptree_show_pretensioner	<logical>	If TRUE a Pretensioner category will be included in the tree	TRUE, FALSE	FALSE
ptree_show_retractor	<logical>	If TRUE a Retractor category will be included in the tree	TRUE, FALSE	FALSE

ptree_show_rigidwall	<logical>	If TRUE a Rigidwall category will be included in the tree	TRUE, FALSE	FALSE
ptree_show_seatbelt	<logical>	If TRUE a Seatbelt category will be included in the tree	TRUE, FALSE	FALSE
ptree_show_sd_orientation	<logical>	If TRUE a Define spring/damper orientation category will be included in the tree	TRUE, FALSE	FALSE
ptree_show_section	<logical>	If TRUE a Section category will be included in the tree	TRUE, FALSE	FALSE
ptree_show_set_beam	<logical>	If TRUE a Set beam category will be included in the tree	TRUE, FALSE	FALSE
ptree_show_set_discrete	<logical>	If TRUE a Set discrete category will be included in the tree	TRUE, FALSE	FALSE
ptree_show_set_node	<logical>	If TRUE a Set node category will be included in the tree	TRUE, FALSE	FALSE
ptree_show_set_part	<logical>	If TRUE a Set part category will be included in the tree	TRUE, FALSE	FALSE
ptree_show_set_segment	<logical>	If TRUE a Set segment category will be included in the tree	TRUE, FALSE	FALSE
ptree_show_set_shell	<logical>	If TRUE a Set shell category will be included in the tree	TRUE, FALSE	FALSE
ptree_show_set_solid	<logical>	If TRUE a Set solid category will be included in the tree	TRUE, FALSE	FALSE
ptree_show_set_tshell	<logical>	If TRUE a Set tshell category will be included in the tree	TRUE, FALSE	FALSE
ptree_show_shell	<logical>	If TRUE a Shell category will be included in the tree	TRUE, FALSE	FALSE
ptree_show_slipring	<logical>	If TRUE a Slipring category will be included in the tree	TRUE, FALSE	FALSE
ptree_show_solid	<logical>	If TRUE a Solid category will be included in the tree	TRUE, FALSE	FALSE
ptree_show_spc	<logical>	If TRUE a Boundary SPC category will be included in the tree	TRUE, FALSE	FALSE
ptree_show_spring	<logical>	If TRUE a Spring category will be included in the tree	TRUE, FALSE	FALSE
ptree_show_surface	<logical>	If TRUE a (contact) Surface category will be included in the tree	TRUE, FALSE	FALSE
ptree_show_table	<logical>	If TRUE a Define table category will be included in the tree	TRUE, FALSE	FALSE
ptree_show_tshell	<logical>	If TRUE a Thick Shell category will be included in the tree	TRUE, FALSE	FALSE
ptree_show_vector	<logical>	If TRUE a Define vector category will be included in the tree	TRUE, FALSE	FALSE
part_replace				
part_replace_process_haz_connection	<string>	undo HAZ welds and remake if ’remake connection’ option is active in part replace	ON, OFF	ON
part_replace_remake_connection	<string>	remake affected Primer connections after part replace	ON, OFF	ON

The following are used for some of the pedestrian tools in PRIMER

Preference	Type	Description	Valid arguments	Default
hic_yellow	<real>	HIC value below which = ’green’, above = ’yellow’ (ENCAP reg)		650.0
hic_orange	<real>	HIC value below which = ’yellow’, above = ’orange’ (ENCAP reg)		1000.0
hic_brown	<real>	HIC value below which = ’orange’, above = ’brown’ (ENCAP reg)		1350.0
hic_red	<real>	HIC value below which = ’brown’, above = ’red’ (ENCAP reg)		1700.0
hic_low	<real>	Low HIC value used in pedestrian area calculator (GTR reg)		800.0
hic_high	<real>	High HIC value used in pedestrian area calculator (GTR reg)		1700.0
hic_grid	<integer>	Grid spacing used in pedestrian area calculator (GTR reg)		10

The following control permissions when writing files

Preference	Type	Description	Valid arguments	Default
directory_permission	<integer>	Octal permission code for new directories	0 - 777	<none>

The following control settings when renumbering entities

Preference	Type	Description	Valid arguments	Default
database_norenumber	<string>	If set to ON, DATABASE_HISTORY cards will not be renumbered	OFF, ON	OFF
nset_norenumber	<string>	SET_NODE cards used by 2d seatbelts will not be renumbered beyond the 7 digit range for versions older than 971R7.1	OFF, ON	ON
material_norenumber	<string>	If set to ON, MAT cards will not be renumbered	OFF, ON	OFF
section_norenumber	<string>	If set to ON, SECTION cards will not be renumbered	OFF, ON	OFF
label_declash	<string>	If set to ON, elements are declashed to avoid problems when reading models into other software	OFF, ON	OFF
nrb_declash	<string>	If set to ON, PRIMER will avoid labelling NRB's with the same label as parts, and vice-versa	OFF, ON	ON
disable_safe_ranges	<string>	If set to ON, safe ranges will be ignored during renumbering	OFF, ON	OFF
range_norenumber	<string>	If set to ON, labels in the specified range will not be renumbered	OFF, ON	OFF
rigid_patch_ref_size	<real>	Rigid parts with diagonals smaller than this value are designated as rigid patches		50
norenumber_min	<integer>	Min label that will not be renumbered		1
norenumber_max	<integer>	Max label that will not be renumbered		1
renumber_latent_in_main_renumber_panel	<logical>	If TRUE Primer will renumber latent entities in the main model renumber panel and in individual entity renumbering panels	TRUE, FALSE	FALSE
label_lock_csv	<string>	csv file that contains locked and safe label ranges		<none>

The following control settings related to quickfind

Preference	Type	Description	Valid arguments	Default
quickfind_search_keyword_menus	<logical>	Whether to search for items in the keyword menus	TRUE, FALSE	TRUE
quickfind_search_other_menus	<logical>	Whether to search for items in other menus	TRUE, FALSE	TRUE
quickfind_search_keyword_manual	<logical>	Whether to search for items in the LS-DYNA keyword manual	TRUE, FALSE	TRUE
quickfind_search_models	<logical>	Whether to search for items in models	TRUE, FALSE	TRUE
quickfind_unmatched_text_colour	<string>	Text colour for unmatched characters	WHITE, GREY, BLACK, RED, GREEN, BLUE, CYAN, MAGENTA, YELLOW	BLACK
quickfind_matched_text_colour	<string>	Text colour for matched characters	WHITE, GREY, BLACK, RED, GREEN, BLUE, CYAN, MAGENTA, YELLOW	BLUE
quickfind_found_list_length	<integer>	Number of items to display in the found list	1 - 20	10
quickfind_recent_history	<integer>	Number of recently selected items to store	0 - 2147483646	10

The following affect scripting

Preference	Type	Description	Valid arguments	Default
javascript_memory_size	<integer>	Maximum memory allocated for garbage collection		25
jit_compiling	<logical>	Whether to use just in time (JIT) compiling in scripts for performance.	TRUE, FALSE	TRUE
modules_directory	<string>	Directory for PRIMER to look for modules in		<none>
script_directory	<string>	Directory for PRIMER to look for scripts in		$OA_INSTALL/primer_library/scripts
ejection_mitigation				
em_ihi_template	<string>	CSV template for interior head impact model build		<none>
free_motion_headform				

fmh_ihi_template	<string>	CSV template for interior head impact model build		<none>
pedestrian_markup				
pm_adult_head_template	<string>	CSV template for adult head model build		<none>
pm_child_head_template	<string>	CSV template for child head model build		<none>
pm_lower_leg_template	<string>	CSV template for lower leg model build		<none>
pm_upper_leg_template	<string>	CSV template for upper leg model build		<none>

Options to control setup and processing seatsquash.

Preference	Type	Description	Valid arguments	Default
squash_x_increment	<real>	X displacement that the dummy will move per iteration to move the dummy out of the seat		0.0
squash_y_increment	<real>	Y displacement that the dummy will move per iteration to move the dummy out of the seat		0.0
squash_z_increment	<real>	Z displacement that the dummy will move per iteration to move the dummy out of the seat		0.0
squash_max_iter	<integer>	Maximum number of iterations that PRIMER will try to do when moving the dummy out of the seat.		100
squash_vol	<real>	If any solid element becomes excessively deformed and reaches this threshold, the seat squash process will stop		0.2
squash_imp_exp_coords	<logical>	Import and export coordinates of a specific part	TRUE, FALSE	TRUE
squash_init_stress_solid	<logical>	Initial stress field for solid elements	TRUE, FALSE	TRUE
squash_init_stress_shell	<logical>	Initial stress field for shell elements	TRUE, FALSE	TRUE
squash_init_stress_beam	<logical>	Initial stress field for beam elements	TRUE, FALSE	TRUE
squash_init_foam_ref_geom_create	<logical>	NONE	TRUE, FALSE	FALSE
squash_init_foam_ref_geom_remove	<logical>	NONE	TRUE, FALSE	TRUE
squash_init_stress_delete	<logical>	NONE	TRUE, FALSE	TRUE
squash_remove_ammg	<logical>	NONE	TRUE, FALSE	TRUE
squash_redraw_after_each_iter	<logical>	If you want to see the progress of the seatsquash then select the Redraw after each iteration checkbox.	TRUE, FALSE	TRUE
squash_try_improve_tet_mesh	<logical>	PRIMER can try to 'smooth' tet meshes to make them better.	TRUE, FALSE	TRUE
squash_create_foam_ref_geom	<logical>	You can opt to create *INITIAL_FOAM_REFERENCE_GEOMETRY cards for the nodes in the seat foam before the deformation. This is only available for hyperelastic materials and certain solid element formulations.	TRUE, FALSE	TRUE
squash_time_posit_dummy	<real>	Time to position dummy		7.5E-2
squash_total_analysis_time	<real>	Total analysis time		0.1
squash_dumping_glob	<real>	Value assigned to global dumping (*DAMPING_GLOBAL).		50.0
squash_use_define_construction	<logical>	Use DEFINE_CONSTRUCTION cards to output the dynain file (the default is to use INTERFACE_SPRINGBACK)	TRUE, FALSE	FALSE

Options to control setup and processing of seatbelt fitting.

Preference	Type	Description	Valid arguments	Default
belt_rows	<integer>	Number of rows of elements across belt		1

belt_width	<real>	Width of belt (total across all rows)	1e-10 - 1e10	40.0
belt_thickness	<real>	Thickness of belt elements	1e-10 - 1e10	1.0
belt_length	<real>	Characteristic length of belt elements	1e-10 - 1e10	25.0
belt_xsec_bpost_slipring_offset	<real>	Database Cross-section offset at B-Post slipring	0.0 - 1e10	200.0
belt_xsec_pelvis_slipring_offset	<real>	Database Cross-section offset at Pelvis slipring	0.0 - 1e10	150.0
belt_xsec_retractor_offset	<real>	Database Cross-section offset at Retractor	0.0 - 1e10	0.0
belt_xsec_fixed_point_offset	<real>	Database Cross-section offset at fixed and end points	0.0 - 1e10	0.0
belt_iterations	<integer>	Number of iterations between resorts during fitting		25
belt_convergence	<real>	Convergence tolerance during fitting	1e-10 - 1.0	1.0e-5
belt_contact_thickness	<string>	Contact thickness used for shell elements during fitting	TRUE, FACTORED, NEUTRAL	TRUE
belt_contact_tfact	<real>	Factor used on FACTORED contact thickness during fitting	0.0 - 1.0	1.0
belt_penetration_dist	<real>	Maximum permitted contact penetration distance into solids	0.0 - 1e10	5.0
belt_overlap	<real>	Overlap between adjacent segments during contact	0.0 - 1e10	0.05
belt_projection_dist	<real>	Initial projection distance during fitting	0.0 - 1e10	35.0
belt_max_curvature	<real>	Limiting transverse angular difference (deg). 0 = no limit.	0.0 - 1e10	0.0
belt_friction	<real>	Transverse friction coefficient for belt to structure contact.	0.0 - 1.0	0.0
belt_acute_angle	<real>	Angle considered to be acute in the belt path during fitting.	0.0 - 180.0	90.0
belt_parallel_fit	<logical>	Whether to use parallelised belt fitting.	TRUE, FALSE	TRUE
belt_fitting_options_menu	<string>	Whether to map the fitting options panel automatically during fitting	AUTO_MAP, MANUAL	AUTO_MAP
belt_fitting_path_order	<string>	Shape of belt fitting path interpolated from basic path points	SPLINE, MIXED, LINEAR	SPLINE
belt_fitting_path_display_mode	<string>	Method used to display belt fitting path	SKELETON_WIRE, THICK_WIRE, THICK_SHADED	THICK_SHADED
belt_fitting_path_offset_mode	<string>	Distance by which fitting path is projected outwards from basic path	FIXED_OFFSET, TENTH_OF_FIXED, NO_OFFSET	FIXED_OFFSET
belt_fitting_auto_depenetration	<string>	Whether belt is depenetrated prior to start of fitting process	OFF, ON	OFF
belt_auto_depen_max_iter	<integer>	Max number of iterations used for automatic depenetration	1 - 10	4
belt_fitting_radial_method	<string>	Method used to determine the outward (radial) direction at belt path points	LOCAL_NORMAL, PATH_TWIST	LOCAL_NORMAL
belt_fitting_self_depenetration	<string>	Whether the belt considers contact against itself during fiting	OFF, ON	ON

belt_self_depen_factor	<real>	Initial factor on belt thickness for contact against itself	1e-5 - 100.0	1.0
belt_self_depen_ramp_iter	<integer>	Number of iterations over which self depen factor ramps down to 1.0	1 - 10000	500
belt_mesh_definition_method	<string>	Method of defining element types used to mesh each belt segment	AUTOMATIC, OLD, NEW	AUTOMATIC
belt_meshed_slipring_radius	<real>	Meshed slipring radius	1e-5 - 1e5	5.0
belt_meshed_slipring_angle	<real>	Angle (deg) that elements subtend when passing around the slipring radius	0.1 - 90	15.0
belt_meshed_slipring_distance	<real>	Distance that short elements span either side of the slipring	1e-5 - 1e5	20.0
belt_path_match_method	<string>	How to deal with mismatch between belt path point and nodal coordinates	PREFER_COORD, PREFER_NODE	PREFER_COORD
belt_path_match_tol	<real>	Tolerance to match nodes to path points when fitting existing path to new dummy.	0.0 - 1e10	1.0
belt_path_nfind_tol	<real>	Tolerance when finding new nodes at path points when fitting existing path to new dummy.	0.0 - 1e10	10.0
belt_label_gap_allowed	<integer>	Permitted gap in start/end label ranges		0

The following affect how *SET_GENERATE are treated in PRIMER

Preference	**Type**	**Description**	**Valid arguments**	**Default**
check_set_generate_on_renumber	<string>	check SET_GENERATE content on renumber selected	ON, OFF	ON
highest_label_considers_set_generate_entries	<string>	Consider SET_GENERATE upper bounds when determining highest label?	TRUE, FALSE, PROMPT	FALSE

Keys can have functions assigned to them:

Preference	**Type**	**Description**	**Valid arguments**	**Default**
F1_key	<string>	Shortcut for F1		<none>
F2_key	<string>	Shortcut for F2		<none>
F3_key	<string>	Shortcut for F3		<none>
F4_key	<string>	Shortcut for F4		<none>
F5_key	<string>	Shortcut for F5		<none>
F6_key	<string>	Shortcut for F6		<none>
F7_key	<string>	Shortcut for F7		<none>
F8_key	<string>	Shortcut for F8		<none>
F9_key	<string>	Shortcut for F9		<none>
F10_key	<string>	Shortcut for F10		<none>
F11_key	<string>	Shortcut for F11		<none>
F12_key	<string>	Shortcut for F12		<none>
A_key	<string>	Shortcut for A		AUTOSCALE
B_key	<string>	Shortcut for B		BLANK
C_key	<string>	Shortcut for C		CLOSE_ALL
D_key	<string>	Shortcut for D		DRAG_CUT
E_key	<string>	Shortcut for E		ENTITIES
F_key	<string>	Shortcut for F		FRINGE
G_key	<string>	Shortcut for G		<none>
H_key	<string>	Shortcut for H		HIDDEN
I_key	<string>	Shortcut for I		ICONISE
J_key	<string>	Shortcut for J		ATTACHED

K_key	<string>	Shortcut for K		RESET_ATTR
L_key	<string>	Shortcut for L		LINE
M_key	<string>	Shortcut for M		MEASURE
N_key	<string>	Shortcut for N		CUT_PLANE
O_key	<string>	Shortcut for O		DISPLAY
P_key	<string>	Shortcut for P		TOGGLE_ALL_PP
Q_key	<string>	Shortcut for Q		QUICK_PICK
R_key	<string>	Shortcut for R		REVERSE
S_key	<string>	Shortcut for S		SHADED
T_key	<string>	Shortcut for T		TIDY_MENUS
U_key	<string>	Shortcut for U		UNBLANK
V_key	<string>	Shortcut for V		VIEW_MENU
W_key	<string>	Shortcut for W		IMAGE_MENU
X_key	<string>	Shortcut for X		CUT_SECTION
Y_key	<string>	Shortcut for Y		CYCLE_OVERLAY
Z_key	<string>	Shortcut for Z		ZOOM
a_key	<string>	Shortcut for a		AUTOSCALE
b_key	<string>	Shortcut for b		BLANK
c_key	<string>	Shortcut for c		CLOSE_ALL
d_key	<string>	Shortcut for d		DRAG_CUT
e_key	<string>	Shortcut for e		ENTITIES
f_key	<string>	Shortcut for f		FRINGE
g_key	<string>	Shortcut for g		<none>
h_key	<string>	Shortcut for h		HIDDEN
i_key	<string>	Shortcut for i		ICONISE
j_key	<string>	Shortcut for j		ATTACHED
k_key	<string>	Shortcut for k		RESET_VIS
l_key	<string>	Shortcut for l		LINE
m_key	<string>	Shortcut for m		MEASURE
n_key	<string>	Shortcut for n		CUT_PLANE
o_key	<string>	Shortcut for o		DISPLAY
p_key	<string>	Shortcut for p		TOGGLE_CURR_PP
q_key	<string>	Shortcut for q		QUICK_PICK
r_key	<string>	Shortcut for r		REVERSE
s_key	<string>	Shortcut for s		SHADED
t_key	<string>	Shortcut for t		TIDY_MENUS
u_key	<string>	Shortcut for u		UNBLANK
v_key	<string>	Shortcut for v		VIEW_MENU
w_key	<string>	Shortcut for w		IMAGE_MENU
x_key	<string>	Shortcut for x		CUT_SECTION
y_key	<string>	Shortcut for y		CYCLE_OVERLAY
z_key	<string>	Shortcut for z		ZOOM
SPACE_key	<string>	Shortcut for space		<none>
ONE_key	<string>	Shortcut for 1		VIEW_P_XY
TWO_key	<string>	Shortcut for 2		VIEW_P_YZ
THREE_key	<string>	Shortcut for 3		VIEW_P_XZ
FOUR_key	<string>	Shortcut for 4		VIEW_P_ISO
FIVE_key	<string>	Shortcut for 5		VIEW_N_XY
SIX_key	<string>	Shortcut for 6		VIEW_N_YZ
SEVEN_key	<string>	Shortcut for 7		VIEW_N_XZ
EIGHT_key	<string>	Shortcut for 8		VIEW_N_ISO
NINE_key	<string>	Shortcut for 9		<none>
ZERO_key	<string>	Shortcut for 0		<none>
EXCLAMATION_key	<string>	Shortcut for !		<none>
DOUBLEQUOTE_key	<string>	Shortcut for "		<none>
HASH_key	<string>	Shortcut for #		<none>
DOLLAR_key	<string>	Shortcut for $		<none>
PERCENT_key	<string>	Shortcut for %		<none>

AMPERSAND_key	<string>	Shortcut for &		<none>
SINGLEQUOTE_key	<string>	Shortcut for ’		<none>
LEFTBRACKET_key	<string>	Shortcut for (		<none>
RIGHTBRACKET_key	<string>	Shortcut for)		<none>
ASTERISK_key	<string>	Shortcut for *		<none>
PLUS_key	<string>	Shortcut for +		ZOOM_IN
COMMA_key	<string>	Shortcut for ,		<none>
MINUS_key	<string>	Shortcut for -		ZOOM_OUT
DOT_key	<string>	Shortcut for .		<none>
SLASH_key	<string>	Shortcut for /		SHORTCUT
COLON_key	<string>	Shortcut for :		<none>
SEMICOLON_key	<string>	Shortcut for ;		<none>
LESSTHAN_key	<string>	Shortcut for <		<none>
EQUALS_key	<string>	Shortcut for =		ZOOM_IN
GREATERTHAN_key	<string>	Shortcut for >		<none>
QUESTIONMARK_key	<string>	Shortcut for ?		SHORTCUT
AT_key	<string>	Shortcut for @		<none>
LEFTSQUAREBRACKET_key	<string>	Shortcut for [		<none>
BACKSLASH_key	<string>	Shortcut for \		<none>
RIGHTSQUAREBRACKET_key	<string>	Shortcut for]		<none>
CIRCUMFLEX_key	<string>	Shortcut for ^		<none>
UNDERSCORE_key	<string>	Shortcut for _		ZOOM_OUT
BACKTICK_key	<string>	Shortcut for ‘		<none>
LEFTCURLYBRACKET_key	<string>	Shortcut for {		<none>
PIPE_key	<string>	Shortcut for |		<none>
RIGHTCURLYBRACKET_key	<string>	Shortcut for }		<none>
TILDE_key	<string>	Shortcut for ~		<none>
SM_BUTTON1_key	<string>	Shortcut for 3D SpaceMouse Button 1		VIEW_P_XY
SM_BUTTON2_key	<string>	Shortcut for 3D SpaceMouse Button 2		VIEW_N_XZ
SM_BUTTON3_key	<string>	Shortcut for 3D SpaceMouse Button 3		VIEW_P_XZ
SM_BUTTON4_key	<string>	Shortcut for 3D SpaceMouse Button 4		VIEW_P_YZ
SM_BUTTON5_key	<string>	Shortcut for 3D SpaceMouse Button 5		<none>
SM_BUTTON6_key	<string>	Shortcut for 3D SpaceMouse Button 6		<none>
SM_BUTTON7_key	<string>	Shortcut for 3D SpaceMouse Button 7		<none>
SM_BUTTON8_key	<string>	Shortcut for 3D SpaceMouse Button 8		<none>
SM_BUTTON9_key	<string>	Shortcut for 3D SpaceMouse Button 9		<none>
SM_BUTTON10_key	<string>	Shortcut for 3D SpaceMouse Button 10		<none>
SM_BUTTON11_key	<string>	Shortcut for 3D SpaceMouse Button 11		<none>
SM_BUTTON12_key	<string>	Shortcut for 3D SpaceMouse Button 12		<none>
SM_BUTTON13_key	<string>	Shortcut for 3D SpaceMouse Button 13		<none>
SM_BUTTON14_key	<string>	Shortcut for 3D SpaceMouse Button 14		<none>
SM_BUTTON15_key	<string>	Shortcut for 3D SpaceMouse Button 15		<none>
SM_BUTTON16_key	<string>	Shortcut for 3D SpaceMouse Button 16		<none>
SM_BUTTON17_key	<string>	Shortcut for 3D SpaceMouse Button 17		<none>
SM_BUTTON18_key	<string>	Shortcut for 3D SpaceMouse Button 18		<none>
SM_BUTTON19_key	<string>	Shortcut for 3D SpaceMouse Button 19		<none>
SM_BUTTON20_key	<string>	Shortcut for 3D SpaceMouse Button 20		<none>
SM_BUTTON21_key	<string>	Shortcut for 3D SpaceMouse Button 21		<none>

SM_BUTTON22_key	<string>	Shortcut for 3D SpaceMouse Button 22		<none>
SM_BUTTON23_key	<string>	Shortcut for 3D SpaceMouse Button 23		<none>
SM_BUTTON24_key	<string>	Shortcut for 3D SpaceMouse Button 24		<none>
SM_BUTTON25_key	<string>	Shortcut for 3D SpaceMouse Button 25		<none>
SM_BUTTON26_key	<string>	Shortcut for 3D SpaceMouse Button 26		<none>
SM_BUTTON27_key	<string>	Shortcut for 3D SpaceMouse Button 27		<none>
SM_BUTTON28_key	<string>	Shortcut for 3D SpaceMouse Button 28		<none>
SM_BUTTON29_key	<string>	Shortcut for 3D SpaceMouse Button 29		<none>
SM_APPLICATION_key	<string>	Shortcut for 3D SpaceMouse Application Button		SHORTCUT_3D
SM_FIT_key	<string>	Shortcut for 3D SpaceMouse Fit Button		AUTOSCALE

The following settings control the interactive text editor, and the contents of files.

Preference	Type	Description	Valid arguments	Default
text_editor	<string>	Text editor to use for editing comments		<none>
text_edit_show_names	<logical>	Whether to show field header names in file to be edited	TRUE, FALSE	TRUE
text_edit_show_rules	<logical>	Whether to show rules about editing files as comments	TRUE, FALSE	TRUE

The following settings allow high performance graphics settings to be tuned. It is recommended that you do not modify these in the preferences editor, but rather use the Display, Tuning option and then SAVE_SETTINGS.

Preference	Type	Description	Valid arguments	Default
gtune_varray	<integer>	Whether or not to use vertex arrays	0 - 2	0
gtune_vbo_verts	<integer>	Whether or not to use VBOs for vertices	0 - 2	0
gtune_vbo_coords	<integer>	Whether or not to use VBOs for coordinates	0 - 2	0
gtune_vbo_limit	<integer>	How VBO usage is limited (explicit size in MBytes, or -1 for auto)	-1 - 1048576	-1
gtune_shader	<integer>	Whether or not to use shaders	0 - 2	0
gtune_mbr	<integer>	Whether or not to use the MBR extension for VBOs	0 - 3	0
threading_status	<string>	Bitwise encoded threading status - do not hand edit!		<none>

The following parameters can be used to preset **TRANSFER_DATA** operations

Preference	Type	Description	Valid arguments	Default
transfer_source_file	<string>	Source model filename		<none>
transfer_data_type	<list>	One or more datatypes to be matched from the list	MAT, SECTION, EOS, HOURGLASS, TMAT	<none>
transfer_match_by	<string>	The method for matching items between source and target models	ID, NAME, BOTH, ALL	NAME
transfer_action	<string>	Where in the target model to copy transferred data	CS, CO, CM, RO	CS
transfer_name_match	<string>	The name matching method used	T_IN_S, S_IN_T, EITHER, EXACT	EITHER
transfer_superseded	<string>	What happens to superseded data	SAVE, DELETE	SAVE
transfer_missing	<string>	Transfer missing items into target model	TRUE, FALSE	FALSE
transfer_populate	<string>	items for populate during transfer	DISCRETE, JOINT, NONE	NONE

The following control undo functionality

Preference	Type	Description	Valid arguments	Default

undo_enabled	<logical>	Undo enabled	TRUE, FALSE	TRUE
undo_deletion	<logical>	Turn on/off undo of deletion	TRUE, FALSE	TRUE
undo_create_entity	<logical>	Turn on/off undo of entity creation	TRUE, FALSE	TRUE
undo_modify_entity	<logical>	Turn on/off undo of modifying an entity	TRUE, FALSE	TRUE
undo_merge_nodes	<logical>	Turn on/off undo of merging nodes	TRUE, FALSE	TRUE
undo_split_elements	<logical>	Turn on/off undo of splitting elements	TRUE, FALSE	TRUE
undo_replace_nodes	<logical>	Turn on/off undo of node replace	TRUE, FALSE	TRUE
undo_coat_shell	<logical>	Turn on/off undo of coat shell	TRUE, FALSE	TRUE
undo_simple_mesh	<logical>	Turn on/off undo of simple mesh	TRUE, FALSE	TRUE
undo_tet_mesh	<logical>	Turn on/off undo of tetrahedral mesh	TRUE, FALSE	TRUE
undo_drag_node	<logical>	Turn on/off undo of node dragging	TRUE, FALSE	TRUE
undo_beam_on_nodes	<logical>	Turn on/off undo of beams creation on nodes	TRUE, FALSE	TRUE
undo_binary_format	<logical>	Use binary keywords format for undo files (faster i/o)	TRUE, FALSE	TRUE
undo_renumber_entity	<logical>	Turn on/off undo of renumbering entities	TRUE, FALSE	TRUE
undo_max_percent_memory	<integer>	Maximum percentage of memory to use for storing undo info	0 - 100	5

The following control treatment of unicode

Preference	Type	Description	Valid arguments	Default
cjk_unix_font	<string>	Font to use for CJK text on unix machines		-misc-fixed-medium-r-normal-*-12-*-*-*-*-*-*-*
cjk_windows_font	<string>	Font to use for CJK text on windows machines		MS Gothic 12
file_encoding	<string>	Character encoding for script files	Latin-1, BIG5, EUC-CN, EUC-JP, EUC-KR, GB, GBK, ISO-2022-CN, ISO-2022-CN-EXT, ISO-2022-JP, ISO-2022-JP-2, ISO-2022-KR, JOHAB, Shift-JIS, UTF-8, UTF-16BE, UTF-16LE, UTF-16, UTF-32BE, UTF-32LE, UTF-32	Latin-1

The following control automatic unit determination

Preference	Type	Description	Valid arguments	Default
configure_model_units	<string>	To configure model length unit. AUTO will determine whether length units are m or mm per model	AUTO, METRES, MM	AUTO
dimension_limit_for_diagonal_upper	<real>	If set, if model diagonal exceeds this value, model units presumed mm	0.0 - 1e10	100.0
dimension_limit_for_diagonal_lower	<real>	If model diagonal below this value, model units presumed m	0.0 - 1e10	10.0
dimension_limit_for_max_shell_thickness	<real>	if set, if maximum struct shell thickness exceeds this value model units presumed mm	0.0 - 1e10	0.5
dimension_limit_for_min_shell_thickness	<real>	if minimum struct shell thickness below this value model units presumed m	0.0 - 1e10	0.1
dimension_limit_for_characteristic_element_length_upper	<real>	if set, if characteristic element length above this value model units presumed mm	0.0 - 1e10	2.0

dimension_limit_for_characteristic_element_length_lower	<real>	if characteristic element length below this value model units presumed m	0.0 - 1e10	0.5
dimension_limit_for_min_density	<real>	if set, if minimum struct material density below this value model units presumed mm else m	0.0 - 1e10	0.01

The following control element visibility

Preference	Type	Description	Valid arguments	Default
NODE_drawn	<string>	Nodes drawn	ON, OFF	OFF
SOLID_drawn	<string>	Solids drawn	ON, OFF	ON
BEAM_drawn	<string>	Beams drawn	ON, OFF	ON
SHELL_drawn	<string>	Shells drawn	ON, OFF	ON
TSHELL_drawn	<string>	Thick shells drawn	ON, OFF	ON
DISCRETE_drawn	<string>	Springs/dampers drawn	ON, OFF	ON
MASS_drawn	<string>	Lumped masses drawn	ON, OFF	OFF

Global preferences.

Global preferences that apply to all programs can be specified using **`"oasys"`** as the program name.

`oasys*<keyword>: <argument>`

At present the following global preferences can be defined.

If a preference is defined twice using both **`"oasys*"`** and **`"primer*"`** then the **`"primer*"`** setting will override the global setting.

Preference	Type	Description	Valid arguments	Default
file_names	<string>	Controls input filename syntax. LSTC = d3*, OASYS = job.ptf*	OASYS, LSTC	OASYS
html_application	<string>	Location of HTML browser		<none>
html_application_linux	<string>	Location of HTML browser for linux (use if the same oa_pref file is used for windows and linux)		<none>
html_application_windows	<string>	Location of HTML browser for windows (use if the same oa_pref file is used for windows and linux)		<none>
image_format	<string>	Default image format	BMP_8_C, BMP_8_UN, PNG_8, GIF_8, BMP_24_UN, PNG_24, JPG_24, PPM_24	JPG_24
intel_hd_use_shaders	<string>	Control usage of hardware shaders on Intel HD graphics cards	AUTO_DETECT, FORCE_OFF, FORCE_ON	AUTO_DETECT
locale	<string>	Language and country locale to use (overrides system one)		<none>
maximise	<logical>	Maximise window when Program is started	TRUE, FALSE	FALSE
pdf_application	<string>	Location of PDF browser		<none>
pdf_application_linux	<string>	Location of PDF browser for linux (use if the same oa_pref file is used for windows and linux)		<none>
pdf_application_windows	<string>	Location of PDF browser for windows (use if the same oa_pref file is used for windows and linux)		<none>
placement	<string>	Location for initial window on multi-screen display	LEFT, RIGHT, BOTTOM, TOP, LEFT_BOTTOM, LEFT_TOP, RIGHT_BOTTOM, RIGHT_TOP	<none>

start_in	<string>	Directory to start Program in		<none>
temp_file_expiry	<integer>	Age in days after which a temporary filename can be reused, 0 = never	0 - 10000	31
show_license_warning	<logical>	Display Window containing License System messages	TRUE, FALSE	TRUE
post_uses_primer	<logical>	ADMIN/INSTALL pref which allows D3Plot, T/his to take an available Primer license	TRUE, FALSE	TRUE
save_window_positions	<logical>	Save position of undocked windows between sessions	TRUE, FALSE	TRUE

The following control directories

Preference	Type	Description	Valid arguments	Default
home_dir	<string>	"home" directory for user		<none>
install_dir	<string>	Directory Oasys Ltd software is installed in		<none>
manuals_dir	<string>	Directory user manuals are installed in		<none>
temp_dir	<string>	temporary directory for user		<none>
checkpoint_dir	<string>	Directory for checkpoint files, or "none" to suppress them altogether		<none>

The following control laser options

Preference	Type	Description	Valid arguments	Default
laser_paper_size	<string>	Default paper size	US, A4	A4
laser_orientation	<string>	Default page orientation	Portrait, Landscape	Landscape
laser_top_margin	<real>	Top margin size in mm		10
laser_bottom_margin	<real>	Bottom margin size in mm		30
laser_left_margin	<real>	Left margin size in mm		20
laser_right_margin	<real>	Right margin size in mm		10

The following control menu and mouse attributes

Preference	Type	Description	Valid arguments	Default
display_factor	<real>	Factor on display size (0.5 - 2.0, automatic if undefined)	0.5 - 2.0	1.2
display_brightness	<real>	Menu brightness (0.0-1.0)	0.0 - 1.0	1.0
display_saturation	<real>	Menu colour saturation (0.0-1.0)	0.0 - 1.0	1.0
button_gradation	<real>	Button shade gradation (0.0-1.0)	0.0 - 1.0	0.0
dv_sync_windows	<string>	Dyn view method(s) for synchronising windows	ICON, ICON+CAPS, ICON+NUM, ICON+CAPS+NUM	ICON+CAPS
dv_left_shift	<string>	Dyn view action for shift + Left mouse	ROTATION_XYZ, ROTATION_XY, ROTATION_Z, ROTATION_SPHERE, TRANSLATION, ZOOM_UP_+VE, ZOOM_DOWN_+VE, UNUSED	ROTATION_XYZ
dv_middle_shift	<string>	Dyn view action for shift + Middle mouse	ROTATION_XYZ, ROTATION_XY, ROTATION_Z, ROTATION_SPHERE, TRANSLATION, ZOOM_UP_+VE, ZOOM_DOWN_+VE, UNUSED	TRANSLATION
dv_right_shift	<string>	Dyn view action for shift + Right mouse	ROTATION_XYZ, ROTATION_XY, ROTATION_Z, ROTATION_SPHERE, TRANSLATION, ZOOM_UP_+VE, ZOOM_DOWN_+VE, UNUSED	ZOOM_UP_+VE

dv_left_ctrl	<string>	Dyn view action for ctrl + Left mouse	ROTATION_XYZ, ROTATION_XY, ROTATION_Z, ROTATION_SPHERE, TRANSLATION, ZOOM_UP_+VE, ZOOM_DOWN_+VE, UNUSED	ROTATION_XYZ
dv_middle_ctrl	<string>	Dyn view action for ctrl + Middle mouse	ROTATION_XYZ, ROTATION_XY, ROTATION_Z, ROTATION_SPHERE, TRANSLATION, ZOOM_UP_+VE, ZOOM_DOWN_+VE, UNUSED	TRANSLATION
dv_right_ctrl	<string>	Dyn view action for ctrl + Right mouse	ROTATION_XYZ, ROTATION_XY, ROTATION_Z, ROTATION_SPHERE, TRANSLATION, ZOOM_UP_+VE, ZOOM_DOWN_+VE, UNUSED	ZOOM_UP_+VE
dv_left_both	<string>	Dyn view action for shift+ctrl + Left mouse	ROTATION_XYZ, ROTATION_XY, ROTATION_Z, ROTATION_SPHERE, TRANSLATION, ZOOM_UP_+VE, ZOOM_DOWN_+VE, UNUSED	ROTATION_XYZ
dv_middle_both	<string>	Dyn view action for shift+ctrl + Middle mouse	ROTATION_XYZ, ROTATION_XY, ROTATION_Z, ROTATION_SPHERE, TRANSLATION, ZOOM_UP_+VE, ZOOM_DOWN_+VE, UNUSED	TRANSLATION
dv_right_both	<string>	Dyn view action for shift+ctrl + Right mouse	ROTATION_XYZ, ROTATION_XY, ROTATION_Z, ROTATION_SPHERE, TRANSLATION, ZOOM_UP_+VE, ZOOM_DOWN_+VE, UNUSED	ZOOM_UP_+VE
dv_shift_action	<string>	Dynamic viewing mode for shift + mouse button	CURRENT, WIREFRAME, FREE_EDGE, UNUSED	CURRENT
dv_ctrl_action	<string>	Dynamic viewing mode for ctrl + mouse button	CURRENT, WIREFRAME, FREE_EDGE, UNUSED	WIREFRAME
dv_both_action	<string>	Dynamic viewing mode for shift+ctrl + mouse button	CURRENT, WIREFRAME, FREE_EDGE, UNUSED	FREE_EDGE
font_scaling	<logical>	whether text in GUI buttons can be scaled down to fit	TRUE, FALSE	TRUE
font_silent	<logical>	whether to write explanatory text if wanted fonts are not found	TRUE, FALSE	FALSE
font_size	<string>	Menu font size	SMALL, DEFAULT, LARGE	DEFAULT
font_type	<string>	Menu font typeface and strength	HELVETICA, HELVETICA-BOLD, TIMES, TIMES-BOLD, COURIER, COURIER-BOLD	HELVETICA
left_handed	<string>	Left handed switching of mouse and/or keyboard	NONE, MOUSE, KEYBOARD, ALL	NONE
zoom_factor	<real>	Zoom Factor for mouse wheel (0.01-1.0)	0.01 - 1.0	0.05
czoom_factor	<real>	Factor for right mouse dynamic zoom (0.01-0.2)	0.01 - 0.2	0.05
kzoom_factor	<real>	Factor for +/- keyboard short-cut keys	0.01 - 100.0	2.0
menu_dragging_mode	<string>	Mode used when moving menu panels with the mouse	WIREFRAME, OPAQUE	WIREFRAME
mouse_3d_rotation_factor	<real>	Factor applied to the speed of rotation when using a 3D mouse		1.0
mouse_3d_pan_factor	<real>	Factor applied to the speed of panning when using a 3D mouse		1.0

mouse_3d_zoom_factor	<real>	Factor applied to the speed of zooming when using a 3D mouse		1.0
mouse_action_middle_button	<string>	Set the action for the middle mouse key during picking	APPLY, REJECT, DESELECT	REJECT
mouse_action_right_button	<string>	Set the action for the right mouse key during picking	APPLY, REJECT, DESELECT	REJECT

The following control treatment of recent files popups

Preference	**Type**	**Description**	**Valid arguments**	**Default**
recent_files_dropdown	<string>	Turn the recent files popup on or off	OFF, ON	ON
recent_files_max_but	<integer>	Maximum number of buttons displayed in a recent files popup	1 - 50	10
recent_files_max_char	<integer>	Maximum number of characters displayed on each recent files button	1 - 512	50

The following control treatment of unicode

Preference	**Type**	**Description**	**Valid arguments**	**Default**
cjk_unix_font	<string>	Font to use for CJK text on unix machines		-misc-fixed-medium-r-normal-*-12-*-*-*-*-*-*-*
cjk_windows_font	<string>	Font to use for CJK text on windows machines		MS Gothic 12
file_encoding	<string>	Character encoding for script files	Latin-1, BIG5, EUC-CN, EUC-JP, EUC-KR, GB, GBK, ISO-2022-CN, ISO-2022-CN-EXT, ISO-2022-JP, ISO-2022-JP-2, ISO-2022-KR, JOHAB, Shift-JIS, UTF-8, UTF-16BE, UTF-16LE, UTF-16, UTF-32BE, UTF-32LE, UTF-32	Latin-1

Command-Line arguments to Primer.

"Command-line" arguments are added to the execution line itself as extra arguments to the code. Where they conflict with settings in the "oa_pref" file the command-line arguments take precedence.

Command-line arguments are a sequence of one or more character strings following the basic PRIMER execution line, for example:

C:\executables primer14_x64.exe ***arg1 arg2 ... argn***

Most, but not all, command-line arguments take the form:

<**-keyword**> **=** <**argument**> for example **-d=opengl**

Command-line argument syntax

- Each argument should be a discrete string with no white space, which means that there must be no spaces between keyword, "=" and argument.

  For example "**-d=opengl**" is valid, but "**-d = opengl**" is not.

- Fixed arguments (such as **opengl**) are not case sensitive.

  Filenames are case-sensitive on Linux and Unix operating systems, but not on Windows. However if you are working on Windows and accessing files on a remote disk mounted on a Unix / Linux system it is recommended that you honour the case of filenames in order to avoid confusion.

- All command-line arguments start with "**-**", ***except*** the name of a keyword file to be processed.

  For example: **primer14_64.exe -d=opengl -start_in=c:\temp example.key**

- Command-line arguments may appear in any order.

  However it is conventional practice to make any input (keyword) file to be processed the last argument on the line. This is what most readers will expect and it makes the meaning clearer.

Directory and Filename arguments containing white space characters

Problems can arise if arguments are file or directory names that contain white space, for example "Documents and Settings". This requires quotes ".." to placed around such names in order to make them discrete character strings. For example

C:\executables\primer14_x64.exe C:\home\example files\test.key

Will not not work because of the white space character between **example** and **files**. In order to turn this into a single string you must place quotes around that argument, for example:

C:\executables\primer14_x64.exe "C:\home\example files\test.key"

A further problem can arise on Windows when using cmd.exe to run things indirectly as the rules for quoting arguments there are arcane to say the least, and research on MSDN may be required to get the syntax right for a given example.

It is ***much*** easier to avoid using directory and file names that contain white space! (If you want to separate names then use the underscore character "_", for example **C:\home\example_files**.)

Command-line arguments valid in Primer

<table>
<tr><th>Function</th><th>Format</th><th>Options</th></tr>
<tr><td>Setting the graphics device

By default no graphics device is defined, and the device selection panel is mapped.

These options can be especially useful if you want to bypass the device selection panel and always start Primer with a particular graphics driver.</td><td><code>-d=<device></code></td><td>
<table>
<tr><td><code>-d=opengl</code></td><td>Use OpenGL 3D graphics</td></tr>
<tr><td><code>-d=x24</code></td><td>24 bit-plane X-Windows graphics</td></tr>
<tr><td><code>-d=x8</code></td><td>8 bit-plane X-Windows graphics</td></tr>
<tr><td><code>-d=x</code></td><td><code>X24</code> if available, otherwise <code>X8</code></td></tr>
<tr><td><code>-d=default</code></td><td>Whichever is available in the order <code>OpenGL</code>, <code>X24</code>, <code>X8</code></td></tr>
<tr><td><code>-d=batch</code>
<code>-d=tty</code></td><td>No graphics - text-only mode</td></tr>
</table>
</td></tr>
<tr><td>Specifying "full screen" mode on startup

Normally Primer occupies about 70% of the display when it starts, the "maximise" argument changes this to become the full screen.</td><td><code>-maximise</code></td><td></td></tr>
<tr><td>Specifying window placement on a multi-display desktop

By default the top right corner of the desktop is used.

The most common arrangement is two screens side by side, for which "left" and "right" may be used. However "top" and "bottom" are also available for the case of two screens one above the other, and the options may be concatenated for a 2x2 display.

These options can be combined with <code>-maximise</code> to fill the relevant screen.

Users on Windows platforms where tools such as NVidia’s "NView" are available may find that it is better to leave window placement to that tool, so that Primer’s windows behave in a fashion consistent with other application windows.</td><td><code>-placement=<where></code></td><td>This option is intended for use where the desktop is spread as a "Single Logical Screen" over multiple monitors.
<table>
<tr><th><code><where></code> values</th><th>Meaning</th></tr>
<tr><td><code>left</code></td><td>Left hand monitor</td></tr>
<tr><td><code>right</code></td><td>Right hand monitor</td></tr>
<tr><td><code>top</code></td><td>Upper monitor</td></tr>
<tr><td><code>bottom</code></td><td>Bottom monitor</td></tr>
<tr><td colspan="2">The above may be concatenated for a 2x2 display, for example</td></tr>
<tr><td><code>top_left</code></td><td>Top left monitor</td></tr>
<tr><td><code>bottom_right</code></td><td>Bottom right monitor</td></tr>
</table>
</td></tr>
<tr><td>Defining a command file name

By default no command file is assumed.</td><td><code>-cf=<filename></code></td><td><filename> can be any text file containing valid commands.</td></tr>
<tr><td>Defining a macro file name

By default no macro file is assumed.</td><td><code>-macro=<filename></code></td><td><filename> can be any text file containing valid macro commands.

More information about macros can be found in 6.27MACROS</td></tr>
</table>

Defining a file of variables to be used in macro files By default no variables are defined	**-macro_var=<*filename*>**	<*filename*> can be any text file containing valid macro variable defintions. The syntax of this file is any number of lines formatted: **variable_name, value** (The comma is required) Both **variable_name** and **value** are treated as text strings, and the effect when the macro is run is to replace any occurrence of **$variable_name** with the string **value**. Comment lines may be added to the file by placing a **$** in their first column.
Defining a JavaScript file name By default no JavaScript file is assumed.	**-js=<*filename*>**	<*filename*> can be any text file containing a valid JavaScript. More information about scripts can be found in 10 Scripting.
Defining an argument to be used in JavaScript files By default no arguments are defined	**-js_arg=<*argument*>**	<*argument*> can be any text string. The arguments can be accessed in the script by using the global *arguments* array. Multiple arguments can be given to a script by using more than one `-js_arg` command line argument.
Defining a JavaScript file to encrypt By default no arguments are defined	**-js_encrypt=<*filename*>**	<*filename*> can be any text file containing a valid JavaScript. Encrypts the script, writing a new file with the suffix '_encrypted.js'
Requesting termination at the end of a command or macro file This is ignored if no command or macro file is defined	**-exit**	
Run PRIMER in "batch" mode where the main application window is not displayed on the screen.	**-batch**	
For the **-batch** option to work you must also specify a command file "**-cf=*filename***", and a macro "**-macro=*filename***" or a JavaScript "**-js=*filename***" This option will automatically set "**-exit**" so that PRIMER terminates after playing the command file, macro or script. You may also wish to use "**-auto_confirm**" as described below.		
Requesting that "auto confirm" should apply when **-batch** is used.	**-auto_confirm**	Note!! Only meaningful when used in conjunction with **-batch**
For historical reasons used of "**-batch**" on its own will not "auto confirm" (ie give the default response) any "stop and ask what to do next" messages. This can result in batch scripts appearing to lock up because they are waiting for a user resonse, but this is not visible on the screen. Using **-auto_confirm** in conjunction with **-batch** will restore this behaviour, meaning that scripts will not lock up for this reason.		

Requesting batch creation of ZTF and group files This generates both <*filename*>.ztf and groups <*filename*>.bin files for subsequent post-processing in D3PLOT. When combined with "**`-d=batch`**" then: • ZTF and group (.bin) files are created, then Primer exits • No licence to run Primer is required	**`-ztf=<filename>`**	<*filename*> must be a valid LS-Dyna keyword (.key) file, with or without the ".key" extension.
Specifying the directory in which to start. PRIMER will make this your "current working directory", so that all files which do not have explicit pathname prefixes are assumd to be in this directory.	**`-start_in=<directory>`**	<*directory*> must be a valid directory name on your system.
Specifying the directory to receive the keyword read log file. This copies all messages normally written to the dialogue box during keyword input to a file **`primer_readlog.txt`** in the directory of your choice. This output file can also be specified via a preference, and interactively from the keyword read **Options** panel. See **Options**: Save Keyin log to file for more details.	**`-rlog_dir=<directory>`**	<*directory*> must be a valid directory name on your system.
Inhibiting user "oa_pref" files	**`-ignore_user_pref`**	This argument will inhibit reading of oa_pref file in home area and in the current working directory. Thus only the system/admin oa_pref files will apply and any file specified with -pref argument.
Specifying a custom "oa_pref" file This causes an extra, optional "oa_pref" file to be read.	**`-pref=<filename>`**	<filename> must be a valid "oa_pref" file. If it has no path prefixed, the file is assumed to be in the OA_INSTALL directory. Any legal filename may be used.

Redirecting console output to a file This option is only available on Windows. On Unix / Linux use standard shell redirection instead, for example: **`primer14_64.exe -d=opengl > filename`**	**`-eo`** **`-eo=default`** **`-eo=<filename>`**	If <filename> is given then it is used as the filename to write the output to. In order to permit multiple sessions to coexist on the same machine the process id will be appended to the main part of the filename. For example if <filename> is "primer_output.log" then the actual filename will be "primer_output_<pid>.log. If no filename is given or the filename is "default" then filename generation is automatic, and the first valid of: %TEMP%\primer_log_<pid>.txt %TMP%\primer_log_<pid>.txt %HOMESHARE%\primer_log_<pid>.txt %USERPROFILE%\primer_log_<pid>.txt will be used.
Defining a list of filenames to be opened By default no list of files is assumed	**`-ml=<filename>`**	If <filename> is defined it should be a list of filenames, each on a new line. These are assumed to be LS-DYNA keyword files (regardless of any extension) and will be opened as models 1 to N. There is a limit of 255 models in PRIMER, so the number of models should not exceed 255.
An LS-DYNA keyword filename By default no filename is assumed	**`<filename>`**	If <filename> is defined it is assumed to be an LS-DYNA keyword file and will be opened as model 1.
A design variable file By default no filename is assumed	**`-ls_opt=<filename>`**	If this is defined, the design variables are applied. This applies to morphing from PRIMER 16.0 onwards as described in the mesh morphing section.

Environment variables that affect Primer.

Environment variables are set at the both at the operating system and user levels, and can be used to influence the behaviour of Oasys Ltd LS-DYNA Environment products. Generally they are better suited to site-wide customisation in the Shell when the software is installed, but users are free to make their own local settings.

Unix/Linux systems running "C" shell (/bin/csh) or its derivatives such as /bin/tcsh:

The format of the command is:

`setenv <parameter> <argument list>`

For example:

```
setenv DISPLAY my_machine:0
setenv SM_USE_VISUAL default
setenv DISPLAY_FACTOR 1.2
```

(Note that the "**`oasys_xx`**" shell is written using C shell syntax, so if it is amended the format above should be used.)

Unix/Linux systems running "Bourne" (/bin/sh) or "Korn" (/bin/ksh) shells

The format of the command is:

```
<parameter>=<argument list>; export <parameter>
```

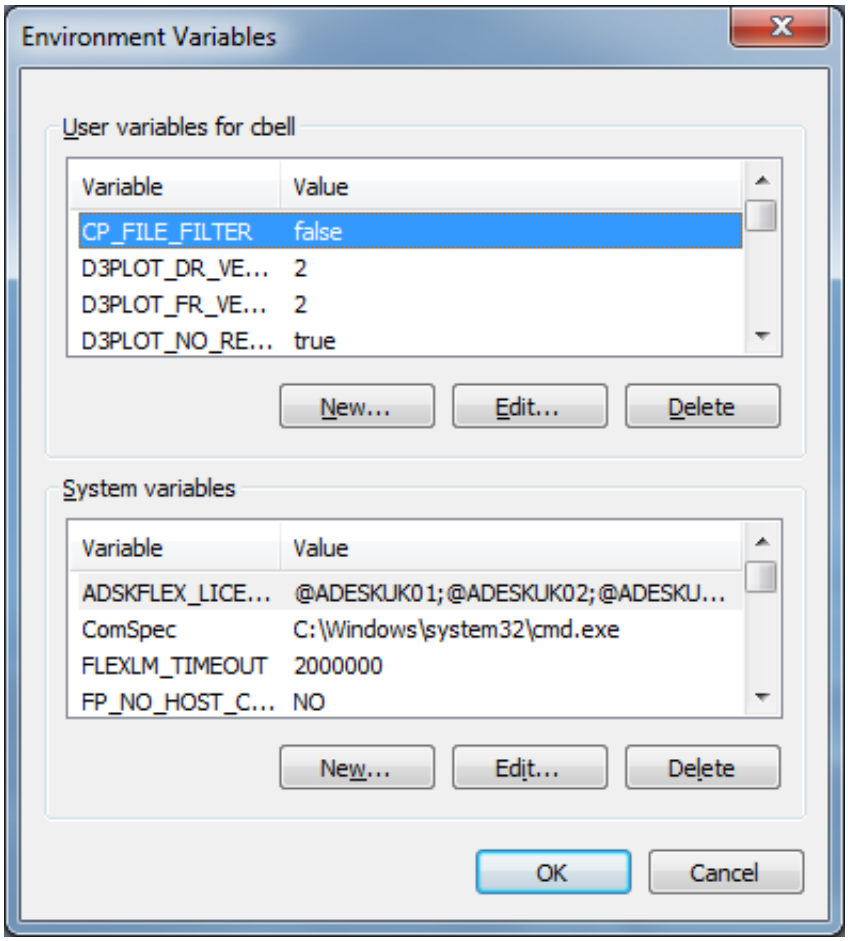
For example:

```
DISPLAY=my_machine:0; export DISPLAY
SM_USE_VISUAL=default; export SM_USE_VISUAL
DISPLAY_FACTOR=1.2; export DISPLAY_FACTOR
```

Windows systems

On Windows Vista / Windows 7:

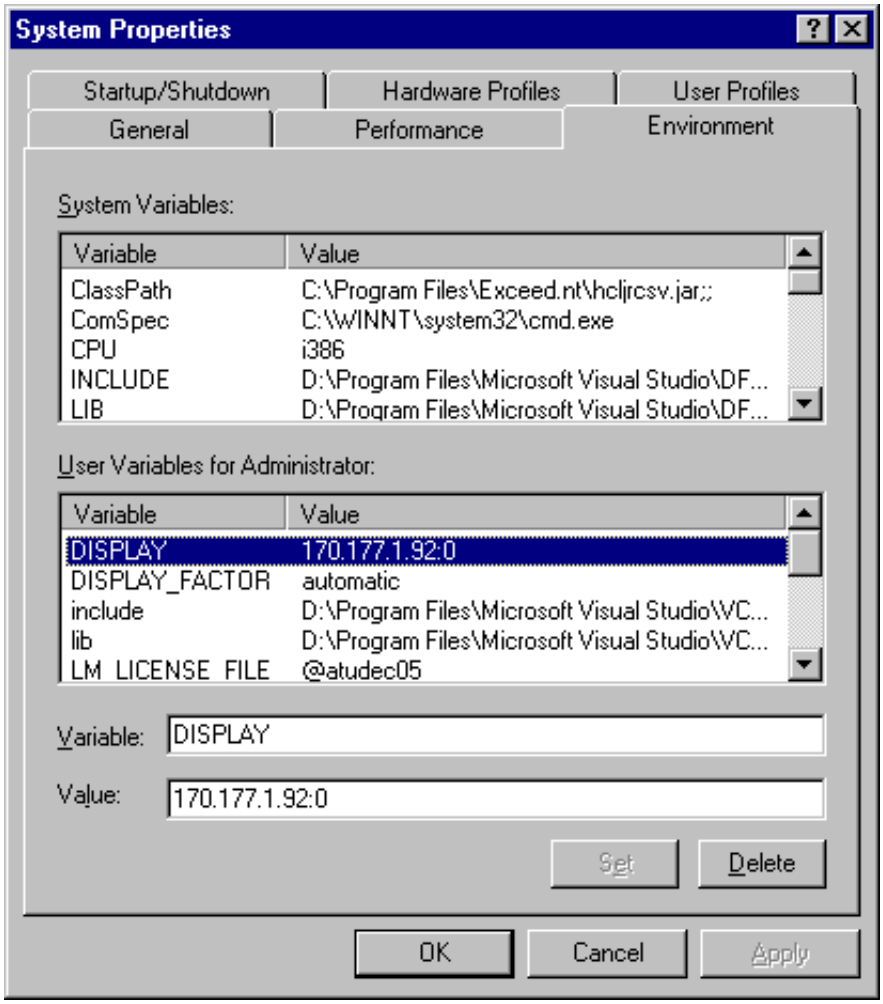
==> Properties
|
V
Advanced System setting
|
V
Environment variables



On Windows XP / 2000

==>Properties
|
V

Advanced
|
V
Environment



Then insert the relevant **`Variable`** and **`Value`** strings into the User or System settings as desired.

In this example it can be seen that user Administrator has set the **`DISPLAY`** environment variable to **`170.177.1.92:0`**.

Environment variables that control the behaviour of PRIMER.

Variable name	Description	Possible Values	Default
The following variables control the graphics and attributes of the display window and menu system.			
DISPLAY	The X11 display id on which graphics will be drawn. If this is not defined (most systems initialise this to "**:0**") then no connection can be made to an X server, and no graphics will be drawn.	(<*machine name*>)**:<*server id*>**(<*.screen id*>)	**:0**
DISPLAY_SATURATION DISPLAY_BRIGHTNESS DISPLAY_FACTOR	Saturation controls the colour saturation (intensity) of menus Brightness controls the colour brightness of menus "Factor" sets the relative display scale, and can range from 0.5 (making menus larger) to 2.0 (making them smaller). It may also be set to "**automatic**" which derives a factor from the physical screen dimensions.	0.0 to 1.0 0.0 to 1.0 0.5 to 2.0, or **automatic**	**1.0** **1.0** **1.0**
SM_USE_VISUAL	Sets the X11 "visual" id to be used for screen menus. Where a graphics display provides "overlay" planes these should normally be used, otherwise this should be left undefined or set to "default". Using an explicit visual id is possible, and this should be defined in hexadecimal (eg 0xf16). Experience has shown the on some Silicon Graphics systems using the "overlay" planes can result in very strange colours in other windows, in which case "default" should be used. Also on some W2000 and graphics board combinations problems may also arise with overlay planes and, again, "default" should be used.	**overlay** **default** <*visual id*> in hex	**overlay**
SM_AUTO_CONFIRM	This variable is often used when replaying command files which, when recorded, paused and asked the user to confirm things. (For example HELP and Warning messages.) If the variable is set (true) then these will not pause and will behave as if the user had pressed "OK" - meaning that command files can play back without user intervention.	**true** or **false**	**false**
USE_PIXMAPS	Controls whether or not the menus use "pixmaps" (off-screen memory) to produce smooth scrolling. Turning this off (false) will save memory, and may help memory problems on a display that has only limited memory available for the X server, but will give slightly jerky window scrolling.	**true** or **false**	**true**

PRIMER_NO_PIXMAP **PRIMER_NO_PBUFFER**	Section 2.4.4.2 Controlling screen refresh, shows a new method that has superseded these variables. May be used to suppress backing store redraws for the OpenGL graphics window. Should be used on OpenGL / X graphics combinations only if you receive errors starting "GLX ...", and then only after consultation with Oasys Ltd.	**true** or **false**	**false**
PRIMER_NO_VARRAY **PRIMER_NO_SHADER** **PRIMER_NO_VBO** **PRIMER_NO_MBR**	These options turn off various aspects of graphics tuning, respectively vertex arrays, shaders, vertex buffer objects, and buffer range extensions. It should not normally be necessary to use these variables, and they are really for debugging purposes. More information about graphics tuning may be found in Section 9.7 Accelerated Graphics	<Any value>	*<none>*
SAVE_UNDER	This flag was introduced to fix a specific bug on Compaq Alpha OSF4.x operating systems. Normally the window manager requests a redraw of windows that have been updated, even when they are currently obscured by something else. However the OSF4 window manager series failed to do this, leading to "bare" patches underneath popup menus when these were unmapped. Setting this flag to **false** results in more redraws on these systems since it suppresses the default "save under" property of X11 windows, but it does at least prevent windows getting bare areas. Compaq have fixed the bug in OSF5, and possibly in later releases of OSF4.	**true** or **false**	**true**
CP_FILE_FILTER	Used during checkpoint file replay to override any file and pathname stored in the checkpoint file, bringing up the file filter instead. This allows checkpoint files to be replayed on different systems.	**true** or **false**	**false**
The following two variables apply on **Windows platforms only**, and should only be used if the menu system is clearly obtaining the wrong display size from the system, as evidenced by fonts and menus being very much the wrong size.			
DISPLAY_HEIGHT	Set an explicit display height in millimetres	<height in mm>	<none>
DISPLAY_WIDTH	Set an explicit display width in millimetres	<width in mm>	<none>
The following variables affect the functioning of the code:			

PRIMER_FILE_FORMAT	An alternative way of controlling the format of ASCII files written on Windows systems. Windows has the peculiarity that, by default, it writes both <carriage return> and <line feed> characters at the end of each line, whereas Unix and Linux platforms only write a <line feed>. The presence of the <carriage return> can confuse some software on Unix/Linux, and its absence can confuse some software on Windows, so in a mixed machine environment there is - unfortunately - no single answer that is "best". PRIMER offers the following options: **native** — Uses the default for the machine's architecture, which adds <carriage return> on Windows. **unix** — Suppresses the <carriage return> (makes no difference on Unix/Linux machines.) This setting has the same effect as the **ascii_file_format** preference above, and is provided for users who wish to set file output format on a "per machine" basis rather than globally with the oa_pref file. If the oa_pref option is used it will supersede this setting.	**native** or **unix**	**native**
The following affect threading. The options they provide are available under **Model, Utilities**, and these variables just provide a way of setting different default values.			
PRIMER_NUM_THREADS	Sets the number of threads on which PRIMER is to run parallelised operations. By default this is the number of cores in the processor, subject to a maximum of 4. Setting this to 1 will suppress threading. Setting it to more than the number of cores on your processor will work, but it will not give any speed improvements - and in fact may slow things down.	**1** to **n**	*<none>*
PRIMER_THREAD_DIAGNOSTICS	Turns on threading diagnostics. This may be a value in the range 0 to 4. 0 is off, and 1 to 4 give progressively more diagnostic information	**0** to **4**	0
The following controls the display of on-line manual pages on Unix systems only. (Windows systems use the default web browser.)			

NETSTART	Command string to start Netscape on Unix/Linux hosts. This is used to fire up the Netscape browser in order to read manual pages from within PRIMER.	Any valid Unix command string.	*<none>*
The following variables are provided for debugging purposes only, and should not normally be used.			
DB_POINTER_CHECK	Runs a check during every internal database allocation and return operation to scan for duplicated or erroneous pointers. This will result in very much (potentially 100x) slower operation of internal memory management, and is normally only used to track down internal errors.	**false**, **1** or **2** (Turned off, level #1 or level #2 checking)	**false**
XSYNC	Runs the X server in "synchronised" (unbuffered) mode. This will give woefully slow graphics, and is used for debugging purposes only.	**true** or **false**	**false**
WARN_REDEFINE	Makes the menu system issue a warning if a button is redefined. Again this is normally only used for debugging purposes.	**true** or **false**	**false**
PRIMER_NO_ERROR_HANDLER	Suppresses the trapping of crashes and the associated logic that offers to save emergency keyout files. Error trapping replaces the current call stack so when running under a debugger it destroys information about the original stack, making it impossible to trace where errors occurred. Setting this variable turns off the special error handler and gives normal (system) handling of crashes.	Any value	*<none>*

APPENDIX XIV: Automated model build from command line

Primer offers the ability to build multiple models from a csv input file.

The command line syntax is simply: **BUILD READ filename.csv**. This may be typed onto the command line of the Primer dialogue box or into a file which Primer reads in batch mode.

Each csv file contains a header line which determines the type of build and consequent format.

Currently the following types are supported:

DATABASE uses the database/template method as used by MODEL->BUILD function
IHI interior head impact
PEDHEAD pedestrian head impact
PEDHEAD_ANGLE pedestrian head impact with approach angle
PEDLEG_LOWER pedestrian lower leg impact
PEDLEG_UPPER pedestrian upper leg impact
PEDLEG_UPPER_2 pedestrian upper leg impact with explicit Z specification

And the following general formats:
GENERAL_TRANSLATE
GENERAL_TRANSLATE_PARAMETER
GENERAL_TRANSLATE_ROTATE
GENERAL_TRANSLATE_VECTOR
GENERAL_TRANSLATE_TRIAD

Here are examples of each type, with $comments included.

DATABASE

DATABASE
Database, vehicle.dba
$
$ following 2 lines are optional
$ tag to specify root directory for output files
Rootdir, /data/DEMO
$ tag for reporter summary template
Reporter_summary, /data/reporter/summary.opt
$
$ the following loadcases consist of
$ directory to be created for output, template mask, output filename, optional reporter individual template
ODB, odb.tpl, odb_model.key, reporter_odb.opt
SIDE, side.tpl, side_impact.key, reporter_side.opt
Etc…

Primer will build the models for each load case and write dyna keyword master files to /<rootdir>/ODB/odb_model.key, /<rootdir>/SIDE/side_impact.key, etc. Component files will be referenced by *INCLUDE or *INCLUDE_TRANSFORM (see below)

Also in /data/DEMO a job submission file (.lst) will be written, which can be read by the submission shell. The shell will submit each job to a node on the processing cluster, where on the completion of each the individual reporter template will be run. When the last job is finished the summary template will be run.

Note on automatic positioning using database/template method.

See also orienting include files for model database.

As some component files (e.g. barriers) are unlikely to be in exactly the correct position, Primer provides an automated method for positioning them on a per build basis.

This positioning is achieved by matching named slave and master orient points which are contained within the database and templates. Typically a master point is kept in a template (associated with a loadcase) but may also reside in the database. A slave point must reside in the database as it is associated with a component file to be oriented.

Suppose, for example, the template odb.tpl contains a master orient point named "od_barrier" with an associated node id (master origin node) and a part name/id (e.g. "bumper skin") to be used for contact purposes. Similarly in the database, on the entry where the odb barrier is kept, a slave orient point of the same name exists, which refers to two nodes (origin and O-X nodes) and a part name/id (e.g. "barrier null shells").

On completion of the build Primer will associate the matching master and slave points. The odb barrier will be translated such that the slave origin node is moved to the position of the master node. A temporary contact will then be created between the master part(s) and the slave part(s) and the odb barrier will be translated into position using the slave vector (defined by the slave origin and O-X nodes). If the initial position gives contact penetration the barrier will be translated against the direction of the vector until just out of contact, otherwise it will be moved along the vector until just about to contact.

Interior Head Impact

```
IHI
Model, /data/DEMO/interior.key
Impactor, fmh.key
$
$ following lines up to loadcase are optional
$ tag to activate depenetration function
$ method = contact, contact name/id, dof <x, xz or xyz>
Depenetrate, contact, head to trim, XZ
$
$ tag to activate auto-vertical angle positioning
$ chin shell set name/id, dof <x, xz or xyz>
Vertical, 5000, XZ
$
$ tag to set root directory for output files
Rootdir, /data/DEMO
$ tag to set root name for output files
Rootname, ihi_test
$ tags for reporter templates
Reporter, individual.opt
Reporter_summary, summary.opt
$
$ loadcase lines consist of
$ directory, x, y, z target coords, horizontal angle, vertical angle, velocity for each
AP1, 1897.38, 602.244, 1205.9, 140, 40, 5364.0
AP3, 1679.84, 661.876, 1087.14, 120, 30, 5364.0
BP1, 2495.27, 578.925, 1237.79, 90, 23, 6705.0
$ for automatic vertical angle positioning, the loadcase line contains two more entries,
$ AUTO to tell PRIMER this is the automatic case, and the angle to rotate back by
$ after the chin has touched the trim. In this case, the vertical angle is the
$ maximum vertical angle the headform can rotate to.
BP2, 1997.65, 567.356, 13455.8, 160, 50, 5364.0, AUTO, 10.0
```

It is also possible to define a positional node for each loadcase line. This headform node is used to position the headform. In the absence of a positional node, Primer will just use the headform default node. The positional node is always defined after the velocity on the loadcase line, so the following lines are all valid:

BP2, 1997.65, 567.356, 13455.8, 160, 50, 5364.0, 1000000 - (positional node defined, no automatic vertical angle positioning)
BP2, 1997.65, 567.356, 13455.8, 160, 50, 5364.0, AUTO, 10.0 - (positional node not defined, but the automatic vertical angle positioning has been activated)
BP2, 1997.65, 567.356, 13455.8, 160, 50, 5364.0, 1000000, AUTO, 10.0 - (Both positional node and automatic vertical angle positioning have been activated)

In the case of the Oasys Ltd free motion headform the base coordinates of the impactor are taken from the head coordinate system and so do not need to be defined here.

Note - the unpositioned head must face in +ve global X (local X aligned with global X axis) and the vehicle must face in -ve global X. This is applicable for all model build applications.

For each loadcase the horizontal and vertical angles of the impactor are set by rotating about the base node, then the head is moved such that the base node is on the target point.

If the depenetrate option is active and the contact is valid, i.e. the latent ids of the trim parts have been added to the slave side, Primer will automatically depenetrate the head. The motion during depenetration is controlled by the dof setting:
Dof = X, head is locked onto the local X axis and depenetrated without changing local y or z coordinates
Dof = XZ, (recommended) head is locked in the local XZ plane and depenetrated without changing the local y coordinate
Dof = XYZ, head is moves freely during depenetration
The advantage of using the xz or xyz methods is that the additional freedom will allow the initial impact point to be achieved more closely to the target point.

For more information on the auto vertical angle positioning, see section 6.13.3. The CSV model build method works in the same way as the method within Primer's FMH positioning panel.

On completion of the build, a master dyna keyword will be output to directories, /<rootdir>/AP1/ihi_test.key, etc. in which the main model is referenced by *INCLUDE and the impactor by *INCLUDE_TRANSFORM.

Additionally, in <rootdir> a submission (.lst) file will be written which may be read by the Shell.

By default reporter variables XCOORD, YCOORD,ZCOORD,H_ANG,V_ANG,VELOCITY are passed into the file. These may be accessed by the individual reporter templates.

Automated positioning methods use Include_Transform. For any method which involves rotational transforms of arbitrary angles, the use of *DEFINE_BOX inside the include file is to be discouraged. LS-Dyna rotates boxes by rotating the two vertices, consequently the process may change the shape and volume of the box adversely.

Pedestrian Head Impact

```
PEDHEAD
model, /data/DEMO/PEDESTRIAN_HEAD/biw.key
impactor, child_head.key
$define 3 coordinates on impactor either by method=define or method=nodes
$orient, define, <name/id of csys>
$orient, nodes, <name/id>, <name/id>, <name/id>
orient, nodes, base node, x node, y node
$
$tag to activate depenetration
$ method = contact, contact name/id, dof <x, xz or xyz>
$ method = partset, partset name/id, dof
depenetrate, contact, head to bonnet contact, XZ
$
$tag for projection method used to obtain target point
$ method = 0 (default): project along global Z
$ method = 1: project back along line of flight
projection_method, <value>
$
$optional offset for deployable bonnet, applied along line of flight
offset, <distance>
$ tag for root directory for output
rootdir, /data/DEMO/NCAP_RUN_2
$ tag for root name for output files
rootname, childhead
$
reporter, individual.opt
reporter_summary, summary.opt
$
$tag for master model style
$ style = 0 (default): Standard
$ style = 1: GM style - All *DEFINE_TRANSFORMATION definitions are written to a common user-defined file.
Each *DEFINE_TRANSFORMATION definition is given a unique label which is equivalent to the directory name if
valid. Each master model then refers to the appropriate *DEFINE_TRANSFORMATION by its label.
$ style = 2: CASE style - A single master model is written. Each *DEFINE_TRANSFORMATION and the
corresponding *INCLUDE_TRANSFORM definition is specified using a *CASE definition
master_style, <value>
$
$tag for file to which all transforms will be written to (master_style 1 and 2)
transform_file, <filename>
$
$tag for define transform title (master_style 1)
deftrans_string, <title>
$
$tag for building only the first loadcase
$ value = 0 (default): all loadcases built
$ value = 1: build all loadcases but write only the first master model; a common file will hold
*DEFINE_TRANSFORMATION definitions for all loadcases
first_point_only, <value>
$
$ Create a *BOUNDARY_SPC definition using the specified node set
node_set_bspc, <node set label >
$
$ loadcase lines consist of
```

$ directory name, zone name(as dir if blank), X coord, Y coord, (optional Z coord)
C1A, C1A, 899.98401, 1393.1749, 800.0000
C1A_2, C1A, 889.98401, 1393.1749
C1B, , 841.03717, 1276.2445
C2A, , 804.94501, 1171.8967
Etc.

The pedestrian head model, unlike the free motion headform, does not carry a specified base coordinate system. Thus the "orient" line is necessary to define

- the base coordinate (at the centre of the head)
- the X coordinate (to give the line of flight pointing in the correct direction)
- the Y coordinate (which defines the normal to the XZ plane for the impactor).

The three points may be defined by the name or id of a DEFINE_COORDINATE_SYSTEM or by defining three nodes.

If these are DATABASE_HISTORY_NODE_ID, they may be defined by name as alternative to label, which has the advantage that they will not be affected by renumbering.

Depenetration is activated either by referencing a surface-surface contact between the head skin and the bonnet or a single part set comprising both. In the latter case, Primer will generate a contact between impactor and target parts.

Finally, the loadcase lines consist of a unique directory name for output key file, the zone name (which if blank, will default to same as directory name), and an X coordinate and Y coordinate. If the Z coordinate is not defined, as is usually the case, it will be calculated by projection determining the target geometry from the depenetrate information. For this reason, if depenetrate is not active, the Z coordinate must be defined explicitly.

In the root directory a submission file (.lst) will be written which may be used by the submission shell.

By default reporter variables ZONE, XCOORD, YCOORD and ZCOORD are passed into the file. These may be accessed by the individual reporter templates.

Note: Optional fields such as zone name may be specified by including a **space** between commas (,<space>,). Omitting the space might result in erroneous processing.

Master model options: Three options are available for the generation of master models:

- By default (or if **master_style** is set to 0), Primer will generate a master keyword file in each output directory, which will reference the main model with *INCLUDE and the impactor with *INCLUDE_TRANSFORM.
- If **master_style** is set to 1, all *DEFINE_TRANSFORMATION definitions are written to a common user-defined file (**transform_file**), each definition carrying a unique label equivalent to the directory name, and a title composed of a loadcase name, a user-defined string (**deftrans_string**) and depenetration information. This **master_style** mode will also add a pair of cancelling transforms to each loadcase that correspond to negative and positive aim point coordinates.
- If **master_style** is set to 2, all output is written to a single user-define master model (transform_file). The main model and the impactor are both referenced once using *INCLUDE and *INCLUDE_TRANSFORM respectively. Data pertaining to individual loadcases is written using *CASE specifiers, each carrying a unique label/id.

Pedestrian Head Impact with Approach Angle

```
PEDHEAD_ANGLE
model, /data/DEMO/PEDESTRIAN_HEAD/biw.key
impactor, child_head.key
$define 3 coordinates on impactor either by method=define or method=nodes
$orient, define, <name/id of csys>
$orient, nodes, <name/id>, <name/id>, <name/id>
orient, nodes, base node, x node, y node
$
$tag to activate depenetration
$ method = contact, contact name/id, dof <x, xz or xyz>
$ method = partset, partset name/id, dof
depenetrate, contact, head to bonnet contact, XZ
$
$ tag for root directory for output
rootdir, /data/DEMO/NCAP_RUN_2
$ tag for root name for output files
rootname, childhead
$
reporter, individual.opt
reporter_summary, summary.opt
$
$ loadcase lines consist of
$ directory name, zone name(as dir if blank), X coord, Y coord, angle of flight, (optional Z coord)
C1A, C1A, 899.98401, 1393.1749, 30, 1000
C1A_2, C1A, 889.98401, 1393.1749, 40
C1B, , 841.03717, 1276.2445, 50.5, 1000
C2A, , 804.94501, 1171.8967, 60
Etc.
```

This is identical to the Pedestrian Head Impact model with the addition that an angle of rotation can be specified for each loadcase line. This angle will be applied to rotate the line of flight in the XZ plane of the impactor. This angle is specified as the fifth field. The Z coordinate, in this case, is not defined.

Automated positioning methods use Include_Transform. For any method which involves rotational transforms of arbitrary angles, the use of *DEFINE_BOX inside the include file is to be discouraged. LS-Dyna rotates boxes by rotating the two vertices, consequently the process may change the shape and volume of the box adversely.

Pedestrian Lower Leg Impact

```
PEDLEG_LOWER
model, /data/DEMO/PEDESTRIAN_LEG_LOWER/biw.key
impactor, child_head.key
$define 3 coordinates on impactor either by method=define or method=nodes
$orient, define, <name/id of csys>
$orient, nodes, <name/id>, <name/id>, <name/id>
orient, nodes, base node, x node, y node
$
$ following lines up to loadcase are optional
$tag to activate depenetration
$ method = contact, contact name/id, dof <x, xz or xyz>
$ method = partset, partset name/id, dof
depenetrate, contact, head to bonnet contact, XZ
$
$ z coordinate
z, 200.0
$
$ tag for root directory for output
rootdir, /data/DEMO/NCAP_RUN_2
$ tag for root name for output files
rootname, childhead
$
reporter, individual.opt
reporter_summary, summary.opt
$
$ loadcase lines consist of
```

$ directory name, zone name(as dir if blank), X coord, Y coord
C1A, C1A, 899.98401, 1393.1749
C1A_2, , 889.98401, 1393.1749
C1B, , 841.03717, 1276.2445
C2A, C2A, 804.94501, 1171.8967
Etc.

The pedestrian lower leg impact model involves an impactor translation without rotation. The "Orient" line is used to specify base, X, and Y coordinates.

"Depenetration" is specified as in the Pedestrian Head Impact case.

A constant Z can be specified here. The impactor will not be moved in the Z direction.

Reporter templates, root directory, and name may also be specified.

Finally, the loadcase lines consist of a unique directory name, zone name, an X coordinate, and a Y coordinate.

Pedestrian Upper Leg Impact

PEDLEG_UPPER
model, /data/DEMO/PEDESTRIAN_LEG_UPPER/biw.key
impactor, child_head.key
$define 3 coordinates on impactor either by method=define or method=nodes
$orient, define, <name/id of csys>
$orient, nodes, <name/id>, <name/id>, <name/id>
orient, nodes, base node, x node, y node
$
$tag to activate depenetration
$ method = contact, contact name/id, dof <x, xz or xyz>
$ method = partset, partset name/id, dof
depenetrate, contact, head to bonnet contact, XZ
$
$ tag for root directory for output
rootdir, /data/DEMO/NCAP_RUN_2
$ tag for root name for output files
rootname, childhead
$
reporter, individual.opt
reporter_summary, summary.opt
$
$ loadcase lines consist of
$ directory name, zone name(as dir if blank), X coord, Y coord, Angle, Velocity, optional parameter value, optional parameter name
C1A, , 899.98401, 1393.1749, -40, 5400.0, 5.1, andy
C1A_2, ,889.98401, 1393.1749, -50.5, 5800.5
C1B, C1B, 841.03717, 1276.2445, -45.0, 4200.3, 5,2, bob
C2A, C2A, 804.94501, 1171.8967, -30.5, 4800.9, 5,3, fred
Etc.

Impact angle and velocity can be specified for each loadcase.

The Z-coordinate will be calculated by projection determining the target geometry from the depenetrate information as in the Pedestrian Head impact case.

Note on parameter specification: A parameter name and value may be specified for each loadcase. These are usually intended to furnish different mass values for different loadcases. This may be done by, for example, specifying a parameter for material density and the corresponding desired value

Pedestrian Upper Leg Impact With Explicit Z

PEDLEG_UPPER_2
model, /data/DEMO/PEDESTRIAN_LEG_UPPER_2/biw.key
impactor, child_head.key
$define 3 coordinates on impactor either by method=define or method=nodes
$orient, define, <name/id of csys>
$orient, nodes, <name/id>, <name/id>, <name/id>
orient, nodes, base node, x node, y node

$
$tag to activate depenetration
$ method = contact, contact name/id, dof <x, xz or xyz>
$ method = partset, partset name/id, dof
depenetrate, contact, head to bonnet contact, XZ
$
$ tag for root directory for output
rootdir, /data/DEMO/NCAP_RUN_2
$ tag for root name for output files
rootname, childhead
$
reporter, individual.opt
reporter_summary, summary.opt
$
$ loadcase lines consist of
$ directory name, zone name(as dir if blank), X coord, Y coord, Z coord, Angle, Velocity, optional parameter value, optional parameter name
LH_1, LH_1,162.957 ,-441.114 , 536, 35.0 ,6.1 ,5.82E-06,RHO
LH_2, LH_2,147.729 ,-387.247 , 536, 35.0 ,6.1 ,5.82E-06,RHO
LH_3, LH_3,165.903 ,-334.394 , 536, 38.5 ,6.8 ,8.71E-06,RHO
Etc.

This is identical to the PEDLEG_UPPER type except for the fact that the Z coordinate is explicitly specified here rather than be computed by projection using depenetration information.

Impact angle and velocity can be specified for each loadcase.

Note on parameter specification: A parameter name and value may be specified for each loadcase. These are usually intended to furnish different mass values for different loadcases. This may be done by, for example, specifying a parameter for material density and the corresponding desired value

General Translate

GENERAL_TR
model, interior.key
impactor, impactor.key
$orient tag uses method=nodes or method=define
orient, nodes, 4675, 4685, 4679
$
$ following lines up to loadcase are optional
$ depenetrate tag, method = contact or method = partset
depenetrate, partset, 10, XZ
$
$ tag to set root directory for output files
Rootdir, /data/DEMO
$ tag to set root name for output files
Rootname, ihi_test
$ tags for reporter templates
reporter, individual.opt
reporter_summary, summary.opt
$
$ loadcase lines consist of
$ directory, X, Y, Z target cords, optional velocity
RUN1, 1897.38, 602.244, 1205.9, -2000.0
Etc.

If the impactor is to be translated without rotation, this format is applicable. The impactor is moved such that the base coordinate (node 4675) lies on the the target point and then is depenetrated appropriately.

General Translate Parameter

GENERAL_TR_PARAMETER
Model, interior.key
Impactor, impactor.key
.
. as above
.
$ loadcase lines consist of output directory, zone, X, Y, Z target coords, Horizontal angle (rotation about Z), Vertical

angle (rotation about Y), Velocity, Optional Parameter Value, Optional Parameter Name
LH_1, LH_1, 162.957, -441.114, 536, 60, 0, 6.1, 5.82E-06,RHO
LH_2, LH_2, 147.729, -387.247, 536, 60, 0, 6.1, 5.82E-06,RHO

If the impactor is to be translated and rotated about Y or Z and, optionally, furnished a parameter value and corresponding parameter name, this format is applicable. Rotations will be applied to the impactor before it is translated. The centre of rotation is the impactor base coordinate.

Note on parameter specification: A parameter name and value may be specified for each loadcase. These are usually intended to furnish different mass values for different loadcases. This may be done by, for example, specifying a parameter for material density and the corresponding desired value

Note on 'Zone' field: Unlike other GENERAL types, GENERAL_TR_PARAMETER includes a zone field (2nd column) which defaults to directory name if left blank

Note on interactive editing : Interactive editing is currently not available for this build type

General Translate Rotate

GENERAL_TR_ROTATE
Model, interior.key
Impactor, impactor.key
.
. as above
.
$ loadcase lines consist of output directory, X, Y, Z target coords, angle about X, Y, Z, optional velocity
RUN1, 1897.38, 602.244, 1205.9, 0, -40, 140, 5340.
RUN2, 1679.84, 661.876, 1087.1, 0, -30, 120, 5340.

If the impactor is to be rotated into position as well as translated this format is applicable. For each loadcase, three global rotations are defined. These will be applied to the impactor before it is translated. The centre of rotation is the impactor base coordinate.

General Translate vector to vector

GENERAL_TR_VECT
Model, interior.key
Impactor, impactor.key
.
. as above
.
$ loadcase lines consist of output directory, X, Y, Z, X', Y', Z' coords, optional velocity
RUN1, 1897.38, 602.244, 1205.9, 1900.0, 620.0, 1210.0
RUN2, 1679.84, 661.876, 1087.1, 1700.0, 670.0, 1090.0
Etc.

For the vector to vector transformation format, a vector PP' is defined for each loadcase. The impactor is rotated so that the line of flight vector coincides with the PP' vector. The axis of rotation is the normal to the two vectors.

General Translate triad to triad

GENERAL_TR_TRIAD
Model, interior.key
Impactor, impactor.key
.
. as above
.
$ loadcase lines consist of output directory, X, Y, Z, X', Y', Z', X", Y", Z" coords, optional velocity
RUN1, 1897.38, 602.244, 1205.9, 1900.0, 620.0, 1210.0, 1910.0, 620.0, 1210.0
RUN2, 1679.84, 661.876, 1087.1, 1700.0, 670.0, 1090.0, 1710.0, 680.0, 1090.0
Etc.

For the triad to triad transformation, a triad PP'P" is defined for each loadcase. The impactor is rotated such that its base triad (as defined by the orient tag) aligns with the target triad.

List of options

Primer's automated model build function supports a number of csv file options (tags), many of which are common across different build types barring DATABASE. The following tables list all such options along with their applicability:

Options specifiable in the top part of the csv file		
Option	Meaning	Applicability
deftrans_string	User-defined string for *DEFINE_TRANSFORMATION title	PEDHEAD, PEDHEAD_ANGLE. Only applicable to GM and CASE master styles
depenetrate	Contact or part set used to depenetrate the impactor, as well as the depenetration method - X, XZ, or XZ	All
first_point_only	All *DEFINE_TRANSFORM definitions are written. However, only the first master model is written	PEDHEAD, PEDHEAD_ANGLE. Only applicable to GM master style
impactor	Impactor model	All
master_style	Can be default (0), GM style (1) or CASE style (2): 1. Default style will generate one master model per loadcase in individual folders 2. GM style will output all *DEFINE_TRANSFORMATION definitions to one file. Individual aster models minus *DEFINE_TRANSFORMATION will be written for each loadcase as in default style 3. CASE style will output all data to one file. Information pertaining to each loadcase will be written under unique *CASE specifiers	PEDHEAD, PEDHEAD_ANGLE
model	Vehicle model	All
node_set_bspc	Generate a *BOUNDARY_SPC for the specified node set	All
offset	Provide an additional offset for the impactor to, for example, account for deployable hoods	PEDHEAD
orient	Specify the line of flight for the impactor	All except IHI
projection_method	Control the movement of the impactor - along global Z or back along the line of flight - in order to obtain target points	PEDHEAD
reporter	Reporter individual file	All
reporter_summary	Reporter summary file	All
rootdir	Output folder for master models	All
rootname	Master model name	All except CASE master style
transform_file	File that will store all *DEFINE_TRANSFORMATION definitions in case of GM master style, and all data in case of CASE master style	PEDHEAD, PEDHEAD_ANGLE. Only applicable to GM and CASE master styles
vertical	Activate auto-vertical angle positioning by specifying a chin shell set and degree of freedom - X, XZ, or XYZ	IHI
z	Constant Z for all loadcases	PEDLEG_LOWER

Options specifiable in loadcase rows	
Option	Applicability

Directory	All
Zone	All except IHI and GENERAL types (other than GENERAL_TRANSLATE_PARAMETER)
X, Y	All
Z	All except PEDLEG_LOWER
Angle(s)	IHI, PEDHEAD_ANGLE, PEDLEG_UPPER, GENERAL_TRANSLATE_PARAMETER, GENERAL_TRANSLATE_ROTATE
Velocity	IHI, PEDLEG_UPPER, PEDLEG_UPPER2, all GENERAL types
Parameter name	PEDLEG_UPPER, PEDLEG_UPPER2
Parameter value	PEDLEG_UPPER, PEDLEG_UPPER2

APPENDIX XV: Finding model mass properties

include file mass, C of G, Inertia

You can obtain include file mass properties in the following ways:

(1) Using **INCLUDE MASS** off the include tree popup

(2) Writing summary file (**MODEL**->**UTILITIES**->**Write Summary file**) with **Write mass for includes** active

(3) Writing out model with **write mass to each include file** set in the Pre-Output check panel (or as an oa_pref setting).

(4) Using **report include mass** from part tree icon

(5) putting all parts of include on Part table and looking at part mass, C of G and Inertia on top row

Methods 1, 2, 3 are all consistent in their handling of Rigid Body merges, giving:

(A) a sum of part masses which only includes effect of merges where both parts are in the include and

(B) a sum of part masses with consideration of all merges.

Method 4 just gives the answer using the “correct” method (B). This also gives C of G and Inertia tensor for parts in include.

Model mass, C of G, Inertia

You can get model mass, C of G, and Inertia:

(1) Using **report model mass** from part tree icon, this will also give timestep added mass adjusted values

(2) writing summary file (**MODEL**->**UTILITIES**->**Write Summary file**), the data also appears in dialogue box

(3) writing out model with **write model mass** option active, the data also appears in dialogue box

(4) putting all parts on part table and looking at part mass, C of G and Inertia totals on the top row

APPENDIX XVI: XML format for model build

Database file

<PRIMER_DATABASE	category = ’database name’	opening tag
	protected = ’no’	
Version data		
<DEFAULT_VERSION	id = ’<id>’	if not specified highest available version if default
<VERSION	descr = ’version name’	
	id = ’<id>’	<id> is integer version id
Component data		
<H1	category = ’title’	
	subcategory = ’subtitle’	
	thumbnail = ’filename’	optional
	owner = ’name of owner’	optional
<COMPONENT	file = ’filename’	component file with absolute or relative path
	version = ’<id>’	version of this component
<EXTRA_DATA	file = ’filename’	file for contact, etc which apply across components
<RENUMBERING	nelidlow = ’<n>’	lower id for nodes/elem/nrb/nsets
	nelidup =	upper id
	idlow =	lower id for other types
	idup =	upper id
	frozenlow =	lower id for range where labels are frozen
	frozenup =	upper
		entity renumber is optional
<CONNECTION_FILE	file = ’filename’	name of xml connection file
	version = ’<id>’	version of this file
<CONNECTION_SETTINGS	target_title = ’<title>’	destination component for connections
	target_subtitle = ’<subtitle>’	
Orientation of component by master and slave point		
<MASTERPOINT	name = ’point name’	lower id for nodes/elem/nrb/nsets
<SLAVEPOINT	Origin = ’<x><y><z>’	coordinate
	Origin_node = ’node’	may be id or name of node
	Ox = ’<x><y><z>’	2nd coordinate to define depenetration vector
	Ox_node = ’node’	
	Vector = ’<x><y><z>’	depenetration vector
	Rotate = ’<rx><ry><rz>’	rotation angles (defined for masterpoint only)
	Part = ’<id>’	part for contact
	Partname = ’name’	
	Partset = ’<id>’	part set for contact
	Partsetname = ’name’	

Here is an axample of a simple database file to show the correct nesting of the elements.

```
<PRIMER_DATABASE category = ’New database’ protected=’no’>

<VERSION descr = ’Version-1’ id = ’1’ />
<VERSION descr = ’Version-2’ id = ’2’ />
<VERSION descr = ’Version-3’ id = ’3’ />

<H1 category = ’New database’ subcategory = ’aaa’ owner = ’fred bloggs’ >
<COMPONENT file = ’Component_files/a1.key’ version = ’1’ />
<RENUMBERING nelidlow = ’100000’
nelidup = ’199999’
idlow = ’1000’
idup = ’99999’
frozenlow = ’1’
frozenup = ’1000’ />
<SLAVEPOINT name = ’point A’
Origin_node = ’3000’
Vector = ’1 0 0’
```

```
Part = ’1000’ />
</H1>

<H1 category = ’New database’ subcategory = ’bbb’
<COMPONENT file = ’Component_files/b1.key’ version = ’1’ />
<COMPONENT file = ’Component_files/b2.key’ version = ’2’ />
<COMPONENT file = ’Component_files/b3.dat’ version = ’3’ />
<EXTRA_DATA file = ’Component_files/extra1.k’/>
<EXTRA_DATA file = ’Component_files/extra2.k’/>
</H1>

<H1 category = ’New database’ subcategory = ’ccc’ >
<CONNECTION_FILE file = ’connection.xml’ version = ’3’ />
<connection_settings target_title = ’New database’ target_subtitle = ’aaa’/>
</H1>

</PRIMER_DATABASE>
```

Template File

<PRIMER_TEMPLATE		opening tag
Selecting a component		
<SELECTED	category = ’title’	identifies matching component
	subcategory = ’subtitle’	
Orientation		
<MASTERPOINT	(as defined above)	master point may be defined in template

Here is an example of template file format.

```
<PRIMER_TEMPLATE>
<SELECTED category = ’New database’ subcategory = ’aaa’/>
<SELECTED category = ’New database’ subcategory = ’bbb’/>
<SELECTED category = ’New database’ subcategory = ’ccc’/>
<masterpoint name = ’point A’
Origin = ’1000 200 110’
Rotate = ’0 30 0’
Part = ’1200’
/>
</PRIMER_TEMPLATE>
```

APPENDIX XVII: MAT100 <DT> added mass for solid spotwelds

LS-Dyna has a special method of adding mass to MAT100 spotweld solids, which may result in a rather higher true %age added mass than the user is lead to expect.

The method is only applied when the parameter <DT> on the MAT100 card is greater than zero.

In this case, in addition to the normal calculation of added mass (let's call it X), LS-Dyna scales the density of the MAT100 solids on a per element basis and reports this extra added mass (let's call this Y) in the otf (d3hsp) file as

"added mass for type 100 hexahedron spot welds=Y"

The true added mass ratio is **(X + Y) / physical mass.**

LS-Dyna (LS971R4), however, takes the physical mass as original physical mass (before density adjusted) + Y, and reports the added mass ratio as **X / physical mass**.

Primer (version 12.1 onward) contour functions and added mass reports on the part table will use the the LS-Dyna method to give consistency with d3hsp file.

The **CALC DT2MS** function accessed under KEYWORD > CONTROL will report the MAT100 DT added mass. This applies for spotweld beams and solids, but it is only for solids that the mass is counted as physical mass.

It is worth noting that changing DT2MS on the *CONTROL_TIMESTEP card will not affect the MAT100 added mass.

APPENDIX XVIII: Morphing keyword format

When morph boxes are created on the interactive Morph panel, keyword files written from PRIMER will contain tghe morph point and morph box information after *END. This is to ensure that the boxes are available again in later PRIMER sessions. Likewise, morph flow definitions created on the Variables tab of the Morph panel will be written after *END.

If the morph entities are only created interactively inside PRIMER, they can be used without detailed knowledge of the keyword format described here. This appendix has only been written for completeness and to make it easier to understand some of the more advanced functions in the JavaScript API classes MorphPoint, MorphBox and MorphFlow.

```
*MORPH_POINT
<label> <x> <y> <z>
```

<label>	I10	Must be unique within a model to identify the morph point for references.
<x>	E10.0	X-coordinate
<y>	E10.0	Y-coordinate
<z>	E10.0	Z-coordinate

This card defines a morph point in the model with its coordinates. Morph points need to be defined for the corner points of morph boxes, and in high order boxes, whose format is described below, there will be more morph points on the edges and faces as well as internal points.

```
*MORPH_BOX(_HIGH_ORDER)
<label> <title>
```

The following line should be included **if and only if** the _HIGH_ORDER suffix is set:

```
<nx> <ny> <nz>
```

The following cards are required again regardless of the _HIGH_ORDER suffix:

```
<point1> <point2> ... <pointn>
<init_x1> <init_y1> <init_z1>
: : :
<init_xn> <init_yn> <init_zn>
<set_node>
```

<label>	I10	Label for the morph box	
<title>	A70	Title for the morph box	
The line below is included ***if and only if*** the _HIGH_ORDER_SUFFIX is set.			
<nx>	I10	Number of morph points along local (parametric) X-direction	
<ny>	I10	Number of morph points along local (parametric) Y-direction	
<nz>	I10	Number of morph points along local (parametric) Z-direction	
The following lines are needed for all morph boxes (independently of their suffix). If the _HIGH_ORDER suffix is set, the value **n = nx * ny * nz** should be used for the number of points below. Otherwise **nx**, **ny**, **nz** will default to 2 for linear boxes, so **n** will be 8 for the number of corners of the cuboid.			
<point1>, <point2>, ...	8I10	Morph point IDs for the box	8 entries per line, using as many lines as required to specify **n** points. In particular, when the _HIGH_ORDER suffix is not used, there will be one such card with 8 entries.

<init_x1>, ...	E10.0	Initial X-coordinates for the morph points, used for the **Reset** functions (interactive and JS API)	The initial coordinates for one morph point are next to each other on one data row. A new row is used for each morph point.
<init_y1>, ...	E10.0	Initial Y-coordinates for the morph points, used for the **Reset** functions (interactive and JS API)	
<init_z1>, ...	E10.0	Initial Z-coordinates for the morph points, used for the **Reset** functions (interactive and JS API)	
<set_node>	I10	*SET_NODE_COLUMN ID for a set containing all nodes which will be morphed with the box. The values A1, A2, A3 on the open-ended cards on that node set card are parametric X, Y and Z coordinates, and A4 is not used. This node set is automatically created and the parametric coordinates are automatically calculated when a morph box is created on the interactive panel or the constructor on the MorphBox JS API class.	

This card is used to define a morph box, which can be either linear or of high order.

```
*MORPH_FLOW
<name>
```

The following card is required when <type> is **CONTINUOUS**:

```
<type> <min> <max>
```

The following card is required when <type> is **STEP**:

```
<type> <min> <max> <step>
```

The following cards are required when <type> is **DISCRETE**:

```
<type> <nvals>
<value1> <value2> ...
```

The following cards are an open-ended list needed for all options of <type>:

```
<point1> <dx1> <dy1> <dz1>
<point2> <dx2> <dy2> <dz2>
: : :
```

<name>	A80	Name for the morph flow. When the morph flow is used to apply morphing with a JSON file passed to the command line, then this name should match the variable name in the JSON file.
<type>	A10	Range type for a variable associated with the morph flow. This is not used in PRIMER except that it is written to JSON design variable files, where it can be used by optimisation programs like LS-OPT to pre-populate variable ranges. This character string should be **CONTINUOUS** (for a continuous variable range between two bounds), **STEP** (for a discrete variable list of uniform step size between two bounds) or **DISCRETE** (for an explicitly specified discrete list not necessarily with uniform step size).
<min>	E10.0	Minimal variable value when <type> is **CONTINUOUS** or **STEP**
<max>	E10.0	Maximal variable value when <type> is **CONTINUOUS** or **STEP**
<step>	E10.0	Variable step size when <type> is **STEP**
<nvals>	I10	Number of variable values when <type> is **DISCRETE**. This option allows to specify a discrete list of variable values, and this field specifies the length of this list.
The following fields are present if <type> is set to **DISCRETE**:		

<value1>, <value2>, ...	8E10.0	Variable values for the case of a discrete list. The number of values is as specified in <nvals>.	8 entries per line, using as many lines as required to specify <nvals> many values
This open-ended list of cards is defined independently of <type>:			
<point1>, ...	I10	Morph point ID	There are as many such data rows as morph points are controlled by the flow. The keyword ends at the next keyword header (or at the end of the file).
<dx1>, ...	E10.0	X-component of vector along which the respective point moves when a variable from a JSON file is applied	
<dy1>, ...	E10.0	Y-component of vector along which the respective point moves when a variable from a JSON file is applied	
<dz1>, ...	E10.0	Z-component of vector along which the respective point moves when a variable from a JSON file is applied	

Technical Topics to do with Graphics

This section attempts to explain some of the technical aspects of using software and graphics on networked computers. It covers both Unix and Windows operating systems, and explains how the Oasys Ltd. LS-DYNA environment suite utilises these.

If all you want to know is how to set up the DISPLAY environment variable click here.
If you want to troubleshoot X11 graphics problems click here

Otherwise read on...

1 Window Managers
- **1.1 What is a window manager?**
- **1.2 What are the main differences between X11 and "Windows" window managers?**
- **1.3 What inter-operability is there between X11 and Microsoft Windows?**
- **1.4 What window managers will the Oasys Ltd. LS-DYNA environment run under?**

2 Graphics
- **2.1 A brief description of how modern graphics hardware works.**
- **2.2 The graphics card**
  - Bit-planes
  - Hidden-surface removal
  - Shading and Lighting
  - Double-buffering
  - Texture Mapping
  - Overlay Planes
- **2.3 Graphics over a network**
  - The network link
  - 2D X11 graphics
  - 3D OpenGL graphics
  - Special aspects of networked graphics
- **2.4 More about graphics hardware**
  - What are the "client" and the "server"?
  - What the ?"!!##? are X11 "visuals"?
  - What visuals does the Oasys Ltd. LS-DYNA environment use?
  - How does OpenGL work with X11?
  - What are OpenGL "objects"?
- **2.5 The "X Virtual Frame Buffer" (Xvfb) server**

3 Controlling Graphics in Oasys Ltd. LS-DYNA environment
- **3.1 Opening an X11 connection to a display**
  - The syntax of the **DISPLAY** variable
  - The simple case: displaying on this machine <= **Probably all you need to know**
  - Directing networked graphics with the **DISPLAY** variable
  - Setting **DISPLAY** on Unix and Linux systems
  - Setting **DISPLAY** on Windows systems
  - Troubleshooting X11 graphics
- **3.2 The Oasys Ltd "Menu Interface" Configuring Menu Interface environment variables.**
  - Setting the correct physical resolution for your display.

1. Window Managers

1.1 What is a window manager?

All modern computers provide an interface with the user that is based on "windows":

- On Unix (and Linux) operating systems this will be based on the X11 graphics system;
- On Microsoft Windows operating systems this will use the native Windows.

(This terminology leads to confusion: windows with a lower case "w" refers to any window on any operating system; Windows with an upper case "W" refers to the generic Microsoft Windows operating systems [NT, 2000, 95, 98, Millenium, CE, ...]).

The functionality in both cases is quite similar: both user and software can request windows on the screen that are dedicated to a particular application. These windows can be resized at will, and their appearance modified in a range of ways.

Clearly something has to mediate the demands of the user and applications, reconcile them with the hardware that is available, manage the mouse and keyboard inputs, decide which windows lie on top of others (the "stacking order") and so on. We will refer to this as the Window Manager.

The way Window Managers work under Microsoft Windows and X11 are radically different, although the functionality presented to the user is very similar in both cases. This document will not attempt to explain their innards, only their characteristics in so far as they affect the Oasys Ltd. LS-DYNA environment.

1.2 What are the main differences between X11 and "Windows" window managers?

X11

- Is ubiquitous on "engineering workstations" running all variants of Unix, and also on Linux-based systems.
- Is intrinsically operating system, network and hardware transparent. Any machine running the X11 protocol can, via a network, display windows on any other machine also running X11. This capability is independent of hardware type, manufacturer and screen attributes.
- Is "open source" provided by the "X Consortium" (http://www.x.org). Anyone can download, examine and modify the source code and, more importantly, its openness almost guarantees that X-based applications will continue to operate across a range of hardware platforms in the future.
- Supports accelerated 3D graphics by "overloading" the X protocol with the extra information required. The OpenGL graphics library has become the industry standard for 3D graphics, and all modern X11 based window managers support this. Consequently 3D graphics may also be run over networks under X11.

Microsoft Windows

- Is provided only on machines running some version of Windows.
- Is not really designed to operate over networks. Inter-operability between machines and operating systems is nearly impossible without 3rd party software.
- Is proprietary to Microsoft.
- Also supports 3D graphics (OpenGL and others) via direct access to the graphics hardware.

1.3 What inter-operability is there between X11 and Microsoft Windows?

Sadly there is no intrinsic support for the one system in the other, nor any signs of there being any in the forseeable future. Given the gradual convergence of "workstation" and "PC" hardware this seems strange, but no-one ever said that the world had to make sense!

However it is possible via 3rd party software packages to emulate Windows under X11, and also X11 under Windows, and users with mixed hardware are currently forced to take this approach.

1.4 What window managers will the Oasys Ltd. LS-DYNA environment run under?

The Oasys Ltd. LS-DYNA environment suite is developed primarily on Unix-based machines using X11. It is also developed on PCs running Microsoft Windows, but at present it requires an X11 emulator to be running on Windows-based platforms.

"Native" Windows support is being developed (slowly) and will be available at some time, as yet unknown, in the future.

2 Graphics

2.1 A brief description of how modern graphics hardware works.

All modern workstations and PCs separate the main processor, the graphics card and the display. At its simplest:

The main processor (CPU, etc)	*Sends graphics requests to*	**The graphics card**. Which has its own memory and, in the case of better cards, considerable "intelligence" for 3D graphics (lighting, shading, hidden surface removal, etc)	*Sends video output to*	**The Display.** Cathode ray tube, LCD, plasma, projector, etc

2.2 The graphics card

There is a huge range of graphics cards available which are grouped here by approximate category. However graphics cards tend to be optimised for particular applications, and in practice these categories will be blurred:

A base-level card	Will typically have 8 bit-planes, giving an acceptable colour range (2^8 = 256 colours) - but no more. Hidden-surface removal, shading and lighting will typically have to be done in software, probably quite slowly. Double-buffering may be available using 4:4 bit-planes, but shading will be poor (2^4 = 16 colours in each buffer).
A higher specification card	Will typically have 24 bit-planes, giving a "true" colour range (2^{24} = 16777216 colours). Such a card may have a hardware Z-buffer, giving faster hidden-surface removal. It may also have some hardware support for shading and lighting. Double-buffering will typically be available using 12:12 bit-planes, giving good shading (2^{12} = 4096 colours in each buffer, which in practice is virtually indistinguishable from 24 bit-plane "true" colour).
A top specification card	Will probably have at least 48 bit-planes, and maybe some overlay planes as well. Hardware Z-buffering, lighting and shading will be available - and extremely fast (many such cards have several processors dedicated to this). Texture mapping in hardware will tend also to be available. Double buffering using 24:24 bit-planes will be "True" colour, and "overlay" planes will allow user-menu and graphics to be separate, reducing the number of image redraw evnts required.

An explanation of some of the terms above:

Bit-planes

After the two width and height dimensions this is the third: "depth". Put simply the more bit-planes a screen has the more colours it can display at the same time.

More specifically the number of colours that can be displayed simultaneously is 2 to the power of the number of planes, thus typical arrangments are:

An 8 bit-plane screen

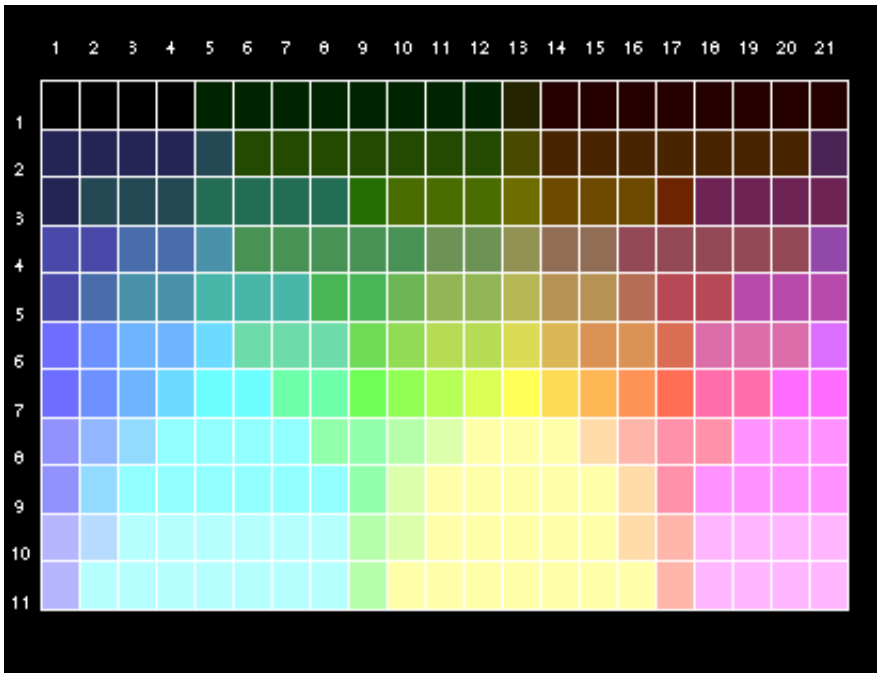
Has 2^8 = 256 colours, usually arranged as:

3 bits Red	= 8 shades
3 bits Green	= 8 shades
2 bits Blue	= 4 shades

The colourmap opposite shows the typical colour ramp available on an 8 bit-plane display.

Note how some of the adjacent shades show poor gradation.

This works surprisingly well, since the human eye is less sensitive to blue than it is to red or green.
Double-buffering such a display to 4:4 bit-planes tends to give poor shading since, even if dithering is used to give more shades, the colour range available is still poor.



A 24 bit-plane screen

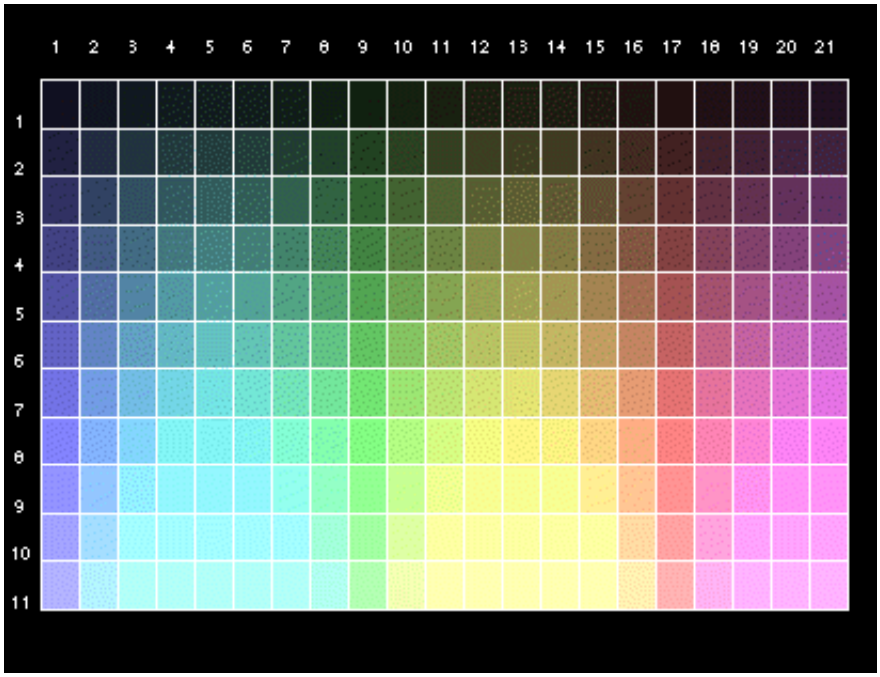
Has 2^{24} = 16777216 colours, arranged as:

8 bits Red	= 256 shades
8 bits Green	= 256 shades
8 bits Blue	= 256 shades

The colourmap opposite shows a typical 24 bit-plane colour ramp.

Note how all shades vary smoothly: even in the very dark and light regimes.

This is often referred to as "true" colour since the human eye can only resolve about 100 shades of each primary colour, and this system produces over twice as many shades. Double-buffering to 12:12 bit planes gives 4096 shades per buffer (4 bits each of red, green and blue) which in practice, with hardware dithering, is difficult to distinguish from full 24 bit-plane colour.



Hidden-surface removal, and "Z" or "depth" buffering.

Most 3D computer graphics is made up of many facets (polygons) which must be displayed correctly such that those nearest the observer obscure those which are further away. This is referred to as hidden surface removal.

The most common method of performing this is to have an auxiliary buffer which is never drawn, but which contains the depth relative to the viewer. This is the Z dimension, where X is width and Y is height, hence "Z-buffer" or "Depth buffer". A pixel in an incoming polygon is only drawn to the display if its depth dimension is closer than the depth currently stored for that pixel in this buffer.

Better graphics perform this in hardware, with little or no speed penalty, but on primitive displays this must be performed in software resulting in a dramatic reduction in "hidden surface" display speed.

Shading and Lighting

Most images that attempt any degree of realism (which is not always the case in engineering plots) require some degree of lighting, which results in a colour variation across facets.

Better graphics cards can be pre-programmed with light source and intensity information, and they will compute the lighting parameters of each incoming facet given its position in space. They will then shade it as required, computing all the colour variations themselves.

More primitive systems may require this to be carried out in software, again giving a dramatic reduction in speed of shaded and lit plots.

Double-buffering

When smooth animation or image rotation/scaling etc is required it is necessary to perform "double-buffering".

The display memory is split into two buffers: one, the front buffer, currently visible; the other, the back buffer, not displayed. Generally each buffer will use half the total graphics bit-planes available, trading off colour resolution against speed. The notation **N:N** user here (eg **12:12**) implies double-buffering using **N** planes in each buffer.

The new image is drawn (invisibly) into the back buffer, and when it is complete the front and back buffers are swapped over exposing the new image, and the process is repeated. The buffer swap operation is performed in hardware, and can usually be done in one vertical retrace of the display - which is faster than the eye can detect - so the transition appears to be smooth.

Texture Mapping

Is a bit like wall-papering. A pre-defined pattern (or "texture"), such as a brick pattern for a wall, is "mapped" onto each facet with the appropriate lighting and shading superimposed.

This tends currently not to be used for engineering applications as, historically, it has been a slow process. However as graphics cards get faster expect this to appear as a rendering option.

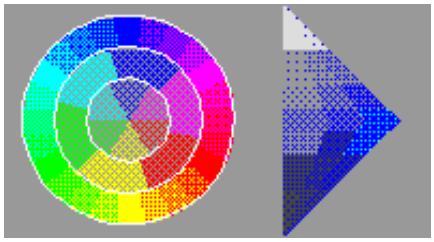
Overlay Planes

Many modern graphics cards permit their memory to be partitioned so that the bit-planes used for the "image" (perhaps 24 planes) are separate from those used for other windows and menus (typically 8 planes). The latter bit-planes are given the additional, special attribute that one (reserved) colour is "transparent", which makes them usable as an "overlay".

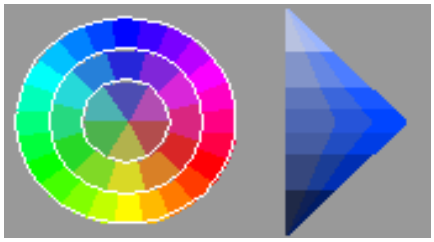
Imagine a transparent sheet that covers your image, which can be wiped clean and redrawn without affecting the image underneath: this is how overlay planes function. The advantage is that other windows, popup menus, etc can be drawn, mapped and unmapped in the overlay planes without corrupting the image planes beneath them, removing the requirement for the (potentially slow) redraw of that image.

From release 8.1 the "screen menu" component of the Oasys Ltd. LS-DYNA environment may be run in the overlay planes where the hardware supports these.

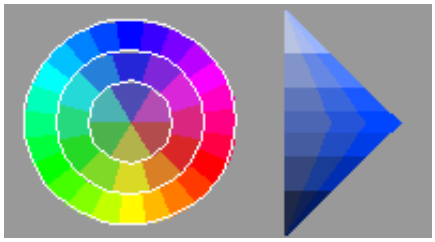
The following images (from the colour "palette" in D3PLOT) demonstrate the effect on the colours available of running the screen menu system in visuals of different depths. (The difference between 8 and 24 bit-plane depths will be undetectable in most browsers.)



4 bit-planes. Dithering is required to achieve most shades.



8 bit-planes. Nearly all shades are directly available



24 bit-planes. All shades are directly available.

2.3 Graphics over a network

In most cases you will be drawing graphics on the display of the machine you are using, and this will almost always be the fastest method since the processor and graphics card can communicate directly at high speed and, if necessary, share memory.

This need not be the case under X11, where networked graphics may also be used:

The main processor (CPU, etc)	*Sends graphics requests ....*	Over a network	*... to a second computer*	**The graphics card**. Which has its own memory and, in the case of better cards, considerable "intelligence" for 3D graphics (lighting, shading, hidden surface removal, etc)

The network

- Is almost always some form of ethernet running TCP/IP (***T***erminal ***C***ontrol ***P***rotocol / ***I***nternet ***P***rotocol)
- But other mechanisms, for example "DecNet", are possible.
- May be of any speed, but obviously the faster the better. Recent machines will have 100 MBit/second ("100 Base T") network ports.

Under X11 networked graphics are used in"client/server" mode by:

- Opening a connection to a remote machine, by setting the DISPLAY environment variable to a "server:screen" on that machine.
- The "client" (application) software sends its graphics requests down the network to that machine.
- The "server" software on that machine, generally part of the window manager, translates these into graphics on the remote machine.

The 3D OpenGL protocol operates over a network under X11 in exactly the same manner:

- A remote connection is opened as before.
- The client application sends graphics requests, this time 3D ones, to the remote machine.
- The server software on that machine (which must include the GLX extension) turns these into 3D graphics requests on that machine.

Special aspects of networked graphics:

If every image redraw, rotation operation or animation frame requires all the graphics information to be sent down the network between client and server machines, networked graphics can become quite slow - especially if the network is low speed. Both "raw" X11 and OpenGL protocols recognise this, and provide a means of storing graphics information in the remote server to speed up redraws.

2D X11 images:

- Can be stored in off-screen "pixmaps", which are effectively copies of the screen in memory. They can be mapped to the screen sufficiently quickly to provide good animations.
- These are suitable for image redraws; and for pre-computed animation frames, where an unchanging sequence of images is played repeatedly.
- They do not help for rotation, scaling or translation since a 2D image must be rebuilt from scratch to show the new appearance.

3D OpenGL graphics imformation:

- Can be stored in "objects" in the server. An "object" contains all the 3D information required to describe a graphical scene.
- Objects can be used for image redraws and for animations, by sending them to the graphics card for re-rendering.
- They can also be used for rotation, scaling and translation since enough 3D information is present to rebuild the image in its new form: only the new rotation and scale information (perhaps a dozen numbers) need be sent down the network to transform a scene of any complexity.

Both mechanisms provide a means, in 2D and 3D respectively, of maximising graphics display speed in some circumstances. If the client machine is doing the "thinking", and the server machine the "drawing", both can operate at full speed at their respective tasks. This can be an efficient means of displaying data from very large databases, since it splits the data retrieval task from the rendering one, although in most cases display on the local machine will be the fastest.

2.4 More about graphics hardware

This section is optional reading for those who want to understand more about the innards of graphics, and find out how to use their machinery more effectively.

What are the "client" and "server"?
What the ?"!!##? are X11 "visuals"?
What visuals does the Oasys Ltd. LS-DYNA environment use?
How does OpenGL work with X11?
What are OpenGL "objects"?

What are the "client" and "server"?

The "client" is the process making the graphics requests, for example D3PLOT, T/HIS, etc.

- You own this process - on Unix systems a "ps" command will show it as belonging to your user id.
- In order to display graphics it must open a display connected to a "server": either locally or over a network.

The "server" is the process running either on this machine (the local host) or remotely elsewhere on a network that receives graphics requests from client processes and turns them into graphics on a screen.

- The server is owned by the operating system of the machine - on Unix systems a "ps" command will show it belonging to user id "root".
- It serves all graphics requests that come to it, not just your client's: it is a shared resource.
- You request facilities (eg windows, colourmaps, backing store) but since it is a shared resource these may not always be available.

In most cases a machine will be running a single X server, by convention #0. However there is a special case of the "X Virtual Frame Buffer" (Xvfb) that is used for batch mode graphics which, by convention, will run as server #1 - see section 2.5 below.

What the ?"!!##? are X11 "visuals"?

The X11 protocol has to operate on a wide range of hardware ranging from old "black and white" displays to modern colour ones, and it is fact of life that the hardware on these machines varies. Some machines support a range of graphics options simultaneously: for example they can have some windows operating in "greyscale" mode alongside others in full colour; they may also provide "overlay" planes. Most modern machine provide a range of visual types and depths (#bit-planes).

To make some sort of sense of all this the X11 protocol groups graphics capability into six categories of "visual": two greyscale and four colour.

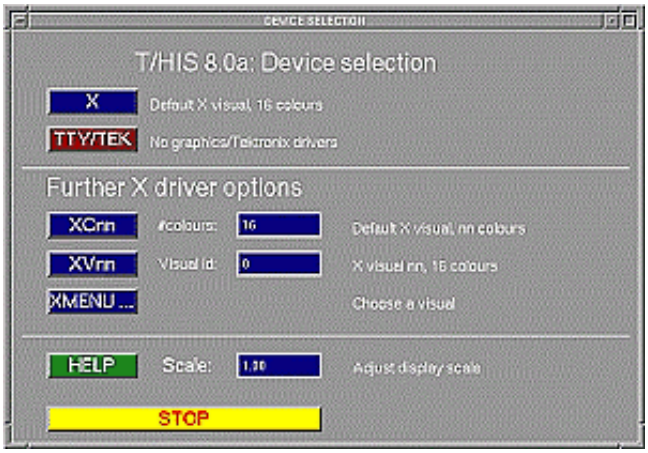
Visual type	What it is	Description and usage
StaticGray	Preset ("static") range of greyscale shades	• Primitive and seldom seen today. Offers only a fixed range of grey shades. • Colour graphics are rendered as shades of grey.
GrayScale	Configurable range of greyscale shades	• Also seldom seen today. Grey shades can be defined by the application. • Colour graphics are rendered as shades of grey.
StaticColor	Preset ("static") range of colour shades	• Seen on older colour displays: colours can only be selected from the pre-defined ones set up. • Usually 4 or 8 bit-planes deep. • Oasys Ltd. LS-DYNA environment uses this for graphics if nothing better is available.
PseudoColor	Configurable range of colour shades	• Found on almost all colour displays and used typically for menus and user interfaces. • Usually 4 or 8 bit-planes deep, but rarely also 12 bit-planes (4096 cols). • Each shade can be defined individually by the application, for example 100 shades of pink • Is used for graphics by Oasys Ltd. LS-DYNA environment, by setting up a colour "cube", if no better visual is available.
TrueColor	Preset linear ramp of red, green and blue shades	• Found on most colour displays. Each colour (r,g,b) has a linear ramp forming a colour "cube" from which shades can be selected. The resolution of each axis of the cube is $2^{\#bits}$ of that colour. • Usually 8, 12 or 24 bit-planes deep, #bits being organised R|G|B as 3|3|2, 4|4|4 and 8|8|8 respectively. • The 24 bit-plane case is "true" colour in that it provides more shades (16777216) than the eye can resolve. This is the preferred visual/depth combination for graphics in the Oasys Ltd. LS-DYNA environment. • The 8 bit-plane case gives only a limited range of shades (256) limiting its usefulness for graphics

DirectColor	Configurable ramps of red, green and blue shades	• Each colour (R,G,B) has its own configurable "ramp", permitting a non-linear and dynamically changeable colour cube. • Invariably 24 or more planes deep. • Needed only for specialised applications. • The Oasys Ltd. LS-DYNA environment only uses this if nothing better is available, and configures linear RGB ramps - effectively TrueColor.

What visuals does the Oasys Ltd. LS-DYNA environment use?

This depends upon the application, whether you are working in 2D or 3D, and what is available on the selected display.

T/HIS 2D "XY" plotting using X11 graphics only.



For graphics

The "**X**" option selects the default visual of the display, which is that in which the main window manager is running. Typically only 16 colours will be requested, since T/HIS graphics are very simple, and these can be shared with other applications. The advantage of this is that screen dumps (eg "xwd") will work correctly giving the proper colours in all windows and sub-windows.

The default visual is usually either 8 bit-plane PseudoColor, or 24 bit-plane TrueColor. Since the demands of the application are so simple either will generally provide all that is required.

The other options **XCnn, XVnn, XMENU...** will only very rarely be required on modern displays, contact Oasys Ltd for advice if the default **X** doesn't work.

For menus

The menu interface will also run in the default visual, unless explicitly forced elsewhere with the **SM_USE_VISUAL** environment variable.

D3PLOT and PRIMER 3D graphics using OpenGL and X11.

For graphics

When "3D graphics" is used (the recommended choice) OpenGL chooses its own visual from those available on the display. There is no user control over this.

The "2D X11 graphics" options should select:

X8

If animation and dynamic rotation speed is critical, and some loss of image quality during shading can be tolerated.

This will choose an 8 bit-plane visual, usually TrueColor or PseudoColor, with only 256 colours - hence the poor shading. However since animation and dynamic viewing are performed using off-screen "pixmaps" (memory containing copies of the screen image), the smaller depth means faster transfer of data from memory to screen.

X24

If shaded image quality is more important than animation and dynamic rotation speed.

This will choose a 24 bit-plane visual (usually TrueColor) with 16777216 colours, so shading should be excellent. However the amount of data to be transferred from Pixmap to Screen is three times as much as in the 8 bit-plane case, and is correspondingly slower. Memory consumption in the X11 server is also greater.

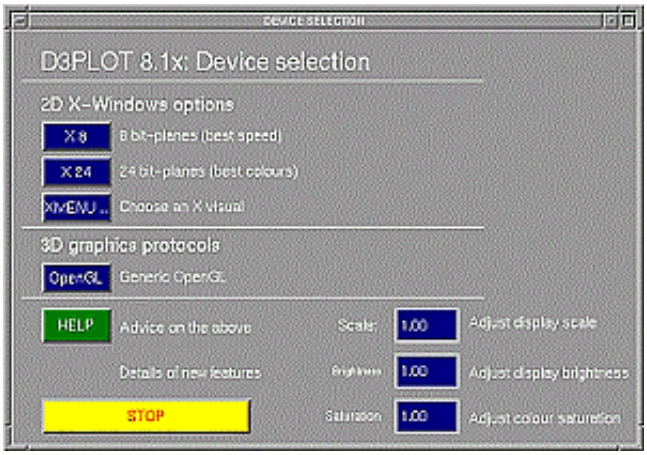
XMENU...

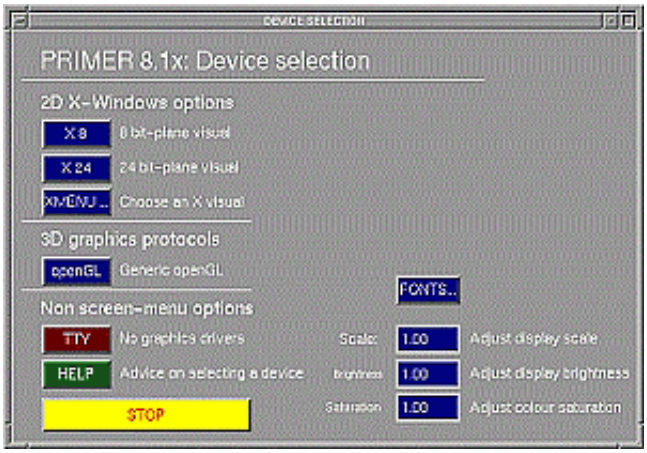
Lets the user select a specific visual.

This is seldom needed - contact Oasys Ltd for advice if the default choices do not give satisfactory results.

For menus

The menu interface will also run in the deepest overlay visual that exists on the server, or in the default visual if none exists, unless explicitly forced elsewhere with the **SM_USE_VISUAL** environment variable.





How does OpenGL work with X11?

Initially the client process opens a conventional X11 connection with the server, then it requests a further OpenGL connection on top of this. Broadly speaking:

- The X11 server manages the windows themselves (sizing, placement, redraw requets), and all mouse and keyboard events.
- The OpenGL connection provides a parallel path for rendering high speed graphics in the window(s).

Where the graphics are being displayed on the local machine a "direct" OpenGL rendering connection is used to give the fastest possible graphics data transfer between client process and display. This bypasses the X11 server.

Where OpenGL graphics are being rendered over a network then an "indirect" OpenGL rendering connection is used:

- The remote X server must have the OpenGL/X (GLX) extension installed. If it doesn’t have this the connection will be refused.
- X11 protocol requests are "overloaded" with OpenGL rendering data and sent to the remote server.
- The GLX extension in that server turns them into local rendering requests on the remote machine.

Using OpenGL over a network generates more traffic than "raw" X11 graphics. This isn't surprising: a coordinate in X11 is an (X,Y) data pair expressed as two "short" (16 bit) integers. The same coordinate in OpenGL is an (X,Y,Z) triplet expressed as 3 "floating" (32 bit) numbers - 3 times as much data. For this reason networked OpenGL graphics is best used with "objects" stored in the server, as this requires geometry information to be sent only once. "Objects" are described below.

What are OpenGL "objects"?

Normally OpenGL rendering requests are sent to the display, drawn and then forgotten. If the image needs to be redrawn, for example due to rotation, then the requests must be regenerated, resent, and so on. When rendering takes place on a local display, via a "direct" rendering connection, then this method gives the best trade-off between speed and memory consumption.

However if you are displaying remotely over a network then the continual re-transfer of the same data in order to draw successive frames is slow, and a method of storing the graphics data in the server would give an obvious efficiency gain. OpenGL "objects" provide this capability:

- Graphics data requests ("move to a point", "draw a line", "change colour", etc) are stored locally in an "object".
- To display that object again the only command required is "redraw the object".
- Redrawing from a stored object tends to be faster than from explicit requests.

If it is faster why don't all OpenGL applications use objects all the time, even when rendering locally?

- Objects use a ***lot*** of memory.

  The application programmer has no control over how graphics data are stored in objects, and cannot therefore practise any storage economies that are made possible by his knowledge of how his data are organised.

  It is not difficult to write a data storage scheme that is much (eg 2x or 3x) more compact than that afforded by objects. This translates into huge savings of memory which, if they stop the machine using virtual memory (swapping), more than offset any speed gains from using objects.
- Objects are slow to create initially.

  The first pass, in which the data is sent to populate the object, is quite slow. This is a considerable price to pay if the image is only to be displayed once, as any savings only come from the 2nd and subsequent renderings of it.

Of the Oasys Ltd. LS-DYNA environment only D3PLOT permits the use of "objects" for data display, and it is recommended that this mode is used only when OpenGL graphics are being used on a remote server over a network. The default "vector" mode in D3PLOT, in which it manages the storage of its own graphics data "vectors", should be used on local displays.

2.5 The "X Virtual Frame Buffer" (Xvfb) server

So far it has always been assumed that graphics requests, however generated, will end up as images on a screen somewhere. However this is not always what is required: in the special case of batch mode running we wish to generate images for capture as bitmaps, but we don't want to have to display them in order to do this. Most conventional X11 servers will only permit access to clients when someone is logged in at their screen and has permitted remote access, which is not much use for batch jobs.

This is where the X Virtual Frame Buffer (Xvfb) server is used.

- It is an X11 server process just like any other.
- It can co-exist with an ordinary X11 server on a machine (it is distinguished by its #server id).
- It accepts and processes X requests from clients just like an ordinary server.
- But it generates images internally in memory, not on a physical display.
- It can act autonomously, not requiring a user to be logged on at a display.
- In fact it can run on a machine that has no display or graphics hardware.

Where facilities for batch graphical processing have been provided by Oasys Ltd the Xvfb server will also be provided, and will have to be running somewhere on an accessible machine (or locally). By default it will be started with the properties:

- Server id #1 (hence the **DISPLAY** variable will be **<*machine name*>:1**)
- 1280x1024x8 (width x height x bit-planes depth) resolution

However these properties may be modified if required. For more information please contact Oasys Ltd.

3 Controlling Graphics in Oasys Ltd. LS-DYNA environment

All Oasys Ltd graphical programmes (PRIMER, D3PLOT and T/HIS) use the same menu interface; however the "graphics" display (of data) varies according to the specialised nature of each programme:

- D3PLOT has a sophisticated 2D (X11) and 3D (OpenGL) rendering capability, which is optimised for speed.
  - Networked graphics are supported efficiently via Pixmaps (2D X11) and Objects (3D OpenGL).
  - Graphics and other memory usage can be controlled.

- PRIMER also has 2D (X11) and 3D (OpenGL) rendering capability, but optimised to display a wide range of entity types - speed being less of an issue.
  - Performance over a network is adequate, but no special provision is made for this.
  - No user control over graphics memory is provided.

- T/HIS has only 2D (X11) rendering capability, as this is all that is required for XY graph plots.

All the Oasys Ltd. LS-DYNA Environment software requires an X11 based window manager or emulator to be running (even when OpenGL is used for rendering). The following topics related to this are described below:

- Defining the DISPLAY environment variable, which determines where graphics are drawn.
- Configuring the parameters of the Oasys Ltd "menu interface".

3.1 Opening an X11 connection to a display: the DISPLAY environment variable.

The syntax of the DISPLAY variable
The simple case: displaying on this machine <= **Probably all you need to know**
Examples of networked graphics setup
Configuring under Unix/Linux
Configuring under Microsoft Windows.
Troubleshooting X11 graphics

The syntax of the DISPLAY environment variable

The X11 protocol requires that an "address" and "screen" number are nominated for its graphical output. This done with the **DISPLAY** environment variable, which has the form

Entries in ***(..)*** can be omitted.

(<*address*>):<*server*>(.<*screen*>)

- The **<*address*>** is a computer name or, more precisely, a network address. If omitted it means the local machine.
- The **<*server*>** is the X11 "server" process. This is typically server #0, but it is possible to have more than one server running.
- The **<*screen*>** is the screen number, starting at 0, on that computer (some machines have > 1 screen). It can be omitted if the display has only one screen.

The simple case: displaying on the screen attached to this computer.

In the vast majority of cases all you will want to do is to display graphics on the screen attached to this computer. Therefore you need default "address", "server" #0, default "screen", which is achieved by setting:

DISPLAY = "**:0**"

Sone examples of achieving this under different operating systems:

Unix/Linux	C shell (/bin/csh, /bin/tcsh)	**setenv DISPLAY :0**
	Bourne/Korn shell (/bin/sh, /bin/ksh)	**DISPLAY=:0; export DISPLAY**
Windows	In the System Properties panel	Variable = **DISPLAY** Value = **:0**

If you are not interested in networked graphics then this is all you need to know about establishing a connection, and you can ignore the rest of this section.

The following examples show how **DISPLAY** could be set to display graphics in a range of local and networked locations:

DISPLAY	Where the graphics will appear	Comments
tigger:0	Server #0 (screen 0) on machine "tigger"	The IP (***I***nternet ***P***rotocol network) address of "tigger" must be known to your system
170.177.15.2:0	Server #0 (screen 0) on the machine with the IP address 170.177.15.2	Since the IP address is given explicitly the remote machine name need not be known.
:0.1	Screen #1 on server #0 on this machine	For a system with 2 screens, this will display on the second.
rainbow:0.1	Screen #1 on server #0 on machine "rainbow"	The IP address of machine "rainbow" must be known, and it is assumed to have at least two screens.

The way the **DISPLAY** variable is set, and remote machine names are mapped to IP addresses, depends upon the operating system in use.

Defining DISPLAY on Unix and Linux systems:

The **DISPLAY** enviroment variable (**$DISPLAY**) is set by:

C shell (/bin/csh /bin/tcsh)	**setenv DISPLAY rainbow:0** **setenv DISPLAY 170.177.15.2:1**	**Server #0 (screen 0 on machine "rainbow"** **Server #1 (screen 0) on machine 170.177.15.2**
Bourne/Korn Shell (/bin/sh /bin/ksh)	**DISPLAY=rainbow:0; export DISPLAY** **DISPLAY=170.177.15.2:1; export DISPLAY**	Ditto

To save the **DISPLAY** variable a given user could place it in their startup files: "~/.cshrc" (for C shell) or "~/.login".

Machine name (hostname) to IP address resolution is defined in the file **/etc/hosts**. (This is owned by root, and requires superuser privileges if it is to be updated.)

This has any number of rows of the form:

<ip address>	**<name>**	**(<name>)**	**(<name>) ...**	IP addresses must have at least one "name", but may have any number of alternative aliases

For example:

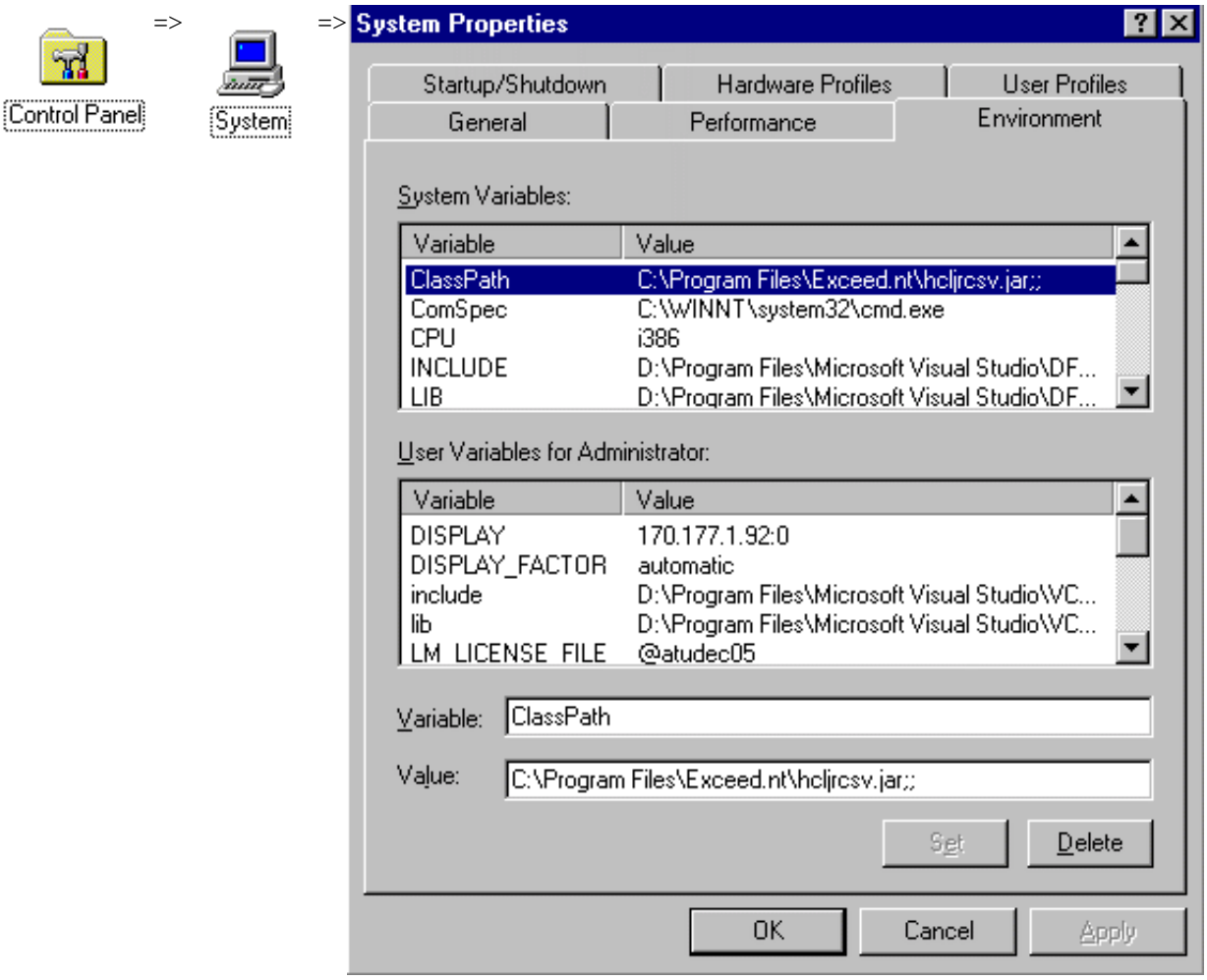
170.177.15.2	**atuhp002**	**atghp002**	**fred**	(This machine is known by any of three names)
193.20.116.16	**abcsgi16**			
193.20.116.20	**abcsgi20**			

Defining DISPLAY on Windows systems:

The **DISPLAY** enviroment variable (**$DISPLAY**) is set in the **System Properties** panel.

The example below is from a Windows NT 4 system, but other variants of Windows will be similar.

This is accessed by:



In the **System Properties** panel select the **Environment** tab, as shown here.

Click on the **User Variables for <userid>** (here **Administrator**) and insert:

Variable:	**DISPLAY**	
Value:	***170.177.1.92:0***	Insert your own IP address or name

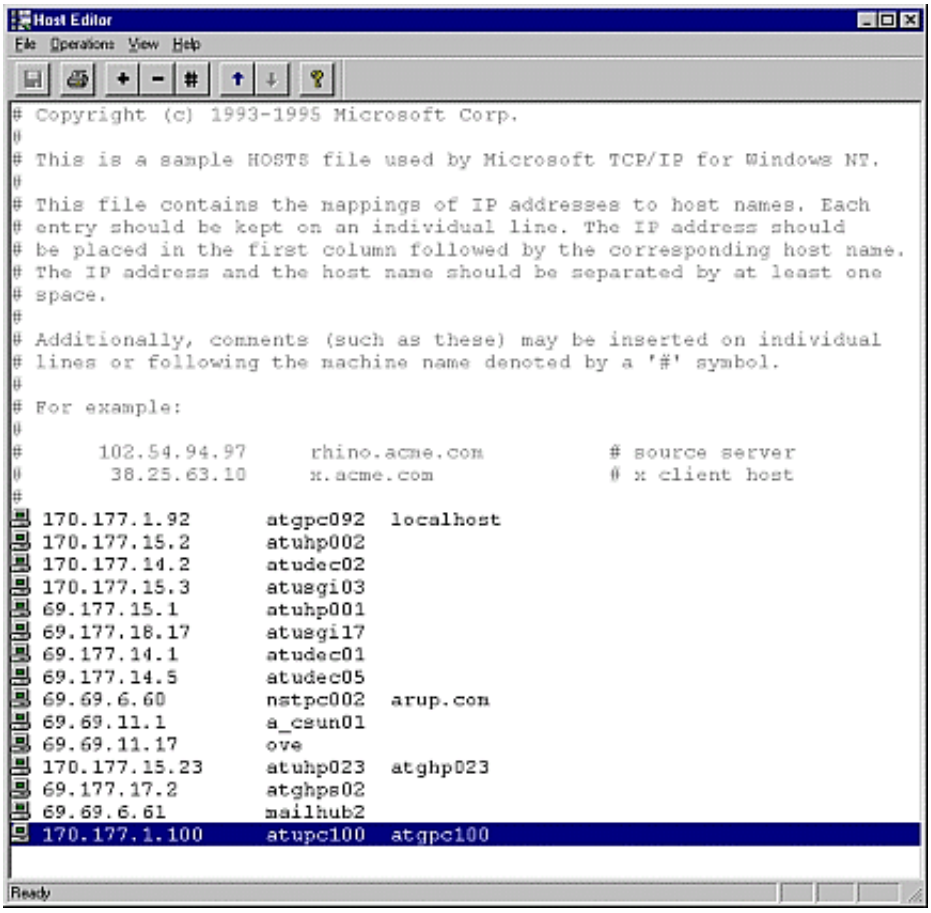
The click on **Set** to add it to your environment variable list.

Machine name (hostname) to IP address resolution is provided by a "Hosts" file (on Windows NT in C:\WINNT\SYSTEM32\drivers\etc) that has the same format as the Unix version above, ie:

<ip address> <name> (<name>) (<name>) ...

This may be updated using a text editor (eg WordPad).

Or on the Hummingbird Exceed™ emulator, which Oasys Ltd recommends, it may be maintained via a **Host Editor:**



The + and - buttons are used to add/remove entries in this panel

Troubleshooting X11 graphics.

Problem	Possible resolution
When you try to start an application you get the message: **`Could not open display`**	This means that the **DISPLAY** variable has not been defined. See above for how to define it on Unix/Linux systems and Windows systems.
When you try to start an application you get the message: **`Could not open display`** ***`<name>:<server>.<screen>`***	This means that your machine (the client) cannot "see" the server machine, or the X11 server on that machine. 1. Is machine <*name*> correct? Check its spelling, and remember that it is case-sensitive. 2. Is <*name*> in your /etc/hosts file (unix) or Hosts file (windows)? 3. Is the IP address in there for <*name*> correct? 4. Is the network path to <*name*> working correctly? (Try "ping"ing it.) 5. Is machine <*name*> turned on and plugged into the network? 6. Is the X11 window manager running on machine <*name*>? 7. Does <*server*>.<*screen*> exist on machine <*name*>?

When you try to start an application you get the message: **`Xlib: connection to <name>:<server>.<screen> refused by server`** **`Xlib: Client is not authorised to connect to Server`**	This means that your client process has made contact with the server's X11 window manager, but has been refused permission to open a window on it. This is a security feature of the X11 system: server window managers must grant permission for clients to open windows, which may be done as follows: On a transient basis, not "remembed" once you log out: Type "**xhost +**" in any window on the server. This will grant permission for any remote client to open windows on this display. To be more selective about which remote clients you will allow to open windows on a display type "**xhost +*name***" where *<name>* is a remote computer name. On a permanent basis, "remembered" across logout/login.: In the file **/etc/X0.hosts** add a list of computer names (each on a new line) that are permitted to open windows on this display. (This is for server #0, for server #1 put it into file **/etc/X1.hosts,** etc.) To allow access to any host put a **"+"** into this file. For more information (on a Unix/Linux host) type "**man xhost**" which describes X11 access control.
The application appears to start, but then fails with a message along the lines of: **`X connection to <name>:<server>.<screen> broken (explicit kill or server shutdown)`**	This usually means that you have forgotten to reset your **DISPLAY** variable, and have popped up a window on someone else's screen. They, understandably, have got annoyed and killed the window (an "explicit kill"). Check that you are displaying graphics where you intended! If this isn't the problem it may indicate that the server to which you are trying to connect is in distress and can't cope with the extra workload - see below.
The X11 server gets very slow, or locks up completely. Normally there are no error messages, but a heavily overloaded server may produce "synchronisation" errors or other symptoms of its impending demise.	This can happen occasionally when the window manager on a server fails to cope with the load placed on it, typically during animation, and dies (a "server shutdown"). Server shutdowns may also occur if they run out of memory: usually caused by performing large "pixmap" or "object mode" animations in D3PLOT which cause the server to grab lots of memory. (Under Unix/Linux memory cannot easily be returned to the system's free pool once it has been allocated so, like middle-aged spread, memory consumption of a process will tend to grow but never diminish. This is not such a problem under Windows.) An X server in distress may be shut down and restarted by the following methods: • Log out from the console, then select "command line", or "no windows", or some similar option (this will depend on the vendor and operating system) for a new login. Log in, then straight out again, and resume the normal "windows" login. This will shutdown then restart the X11 server, which usually sorts out problems. • If the display has locked up (no response to mouse or keyboard) then log in from a remote machine as "root", and kill the window manager process explicitly. ("**ps -ealf | grep X**" will usually find the process, and "**kill -9 <process id>**" will zap it.). • If you can't log in remotely, or don't have root access, reboot or turn the machine off and on again! Cruel but effective.

OpenGL Extension missing on remote server. **`Xlib: extension "GLX" missing on "<name>:<server>.<screen>`**	You may see this if you try to open a remote OpenGL window on a local server that does not have the OpenGL/X extension "GLX" installed. You will not be able to open remote OpenGL clients until it has been installed. Note that you may still see this message on a machine that is able to display OpenGL graphics locally. This means that it can handle a "direct" OpenGL connection from a local client (which largely bypasses the X server), but that it does not have the ability to render "indirect" OpenGL requests. See "How does OpenGL work with X11?" for more information. But in the meantime you will have to use a **X..** option to display remote graphics.
Other errors, typically: **`X Error of failed request:`** **`Major opcode ...`** **`Minor opcode ...`** **`Resource id ...`**	These usually suggest an error in the Oasys Ltd. LS-DYNA environment. Please make a copy of the error message and contact Oasys Ltd for help and advice.

3.2 The Oasys Ltd "Menu Interface"

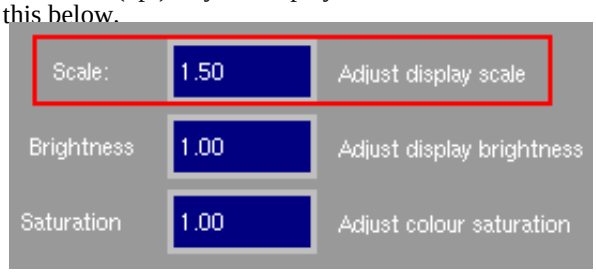
All graphical Oasys Ltd. LS-DYNA environment software uses a menu interface, which is effectively a window manager that runs in the application's own window.

Its default configuration should work satisfactorily in most cases, but there are some environment variables that can be set to change its behaviour:

Configuring Menu Interface environment variables.

Setting the correct physical resolution for your display.

Environment Variable	Possible values	What it does
USE_PIXMAPS	**true**[default] or **false**	Uses "pixmap" backing store to make the scrolling of windows smoother. If memory consumption in the X server is a problem, which sometimes occurs in older X terminals and emulators, this may be set to **false** to conserve memory. You will notice that scrolling menu windows gives a little bit of "flicker", but functionality is unaffected.
SM_USE_VISUAL	**overlay**[default(1)] **default** ***<visual id>***	Defines which bit-planes the menu system runs in. • "**overlay**" puts it into the deepest overlay planes that exist on the display. • "**default**" uses the default planes. • ***<visual id>*** is the visual id (in hexadecimal) in which it is to run - ask Oasys Ltd for advice first! On some older displays, especially SGis, using the overlay planes may result in strange colours elsewhere on the screen. And on seriously old machines the overlay planes may only be 2 bits deep resulting in the menu system appearing only in shades of grey. In these cases set this variable to **default** to force the menu system into the default (image) planes. (1) "**overlay**" is the default for PRIMER and D3PLOT to avoid conflict with the image planes, but "**default**" is used for T/HIS to make screen grabs simpler. See the notes on "Overlay Planes" above for more information about this.

SAVE_UNDER	**true**[default] or **false**	Determines whether or not the "save under" property of the window manager is requested for the area under popup menus. This property "saves" the portion of the image under these sub-windows, meaning that they do not have to be redrawn explicitly when made visible again. On Compaq OSF 4 operating systems the default behaviour can result in an empty grey area appearing under these menus. This is due to a bug in the Compaq implementation of the X server which appears to be fixed in OSF 5. Setting this variable to **false** will cure the problem on these platforms, although this can lead to a lot of image redraws if the visual used for the menu system is not **overlay** (or the machine has no overlay planes).
DISPLAY_FACTOR	<*undefined*>[default] **automatic** 0.5<**value**<2.0	The Oasys Ltd menu system tries to present a consistent appearance over a range of display resolutions and sizes. To do this it attempts to determine the physical resolution in dots per inch (dpi) of the display and, in conjunction with the pixel resolution (eg 1280x1024), to provide the best appearance it can. Unfortunately not all X servers are correctly configured to provide this information, and there is no standard way of performing this configuration. If the menus on your display seem too big or too small, or just plain "wrong", the following process is recommended: Set **DISPLAY_FACTOR** to **automatic**, then run D3PLOT to see how it looks. If that is still unsatisfactory then try a range of **Scale** values in the front panel to try to arrive at a satisfactory value. If that still doesn't work properly it is probably because the physical resolution (dpi) of your display has not been set correctly see how to do this below. 

For examples of how to set environment variables see Defining **DISPLAY** on Unix and Linux systems: or Defining **DISPLAY** on Windows systems:. The method is identical.

Setting the correct physical resolution for your display.

This is only necessary if you think that your display is assuming the wrong physical properties for the screen, as evidenced by fonts and/or windows coming out the wrong size. Unfortunately there is no standard way of doing this, and the following hardware-dependent solutions may not meet your case. If you still have problems please contact Oasys Ltd for help.

Windows platforms
HP-UX systems
Compaq OSF systems
PCs running Windows with the Exceed emulator.

Windows platforms (2000, XP, Vista)

In most cases the operating system should "know" the physical attributes of the display, but we have seen a few cases (especially on Vista) where this is not the case and it is necessary to give the Oasys Ltd. LS-DYNA environment this infiormation explicitly. This is done via the two environment variables:

DISPLAY_HEIGHT	The height of the display in millimetres
DISPLAY_WIDTH	The width of the display in millimetres

It is best to set these as "system" variables, as then they will apply for all users on that machine; but if you only have permission to set environment variables for your user name that will suffice. Once set close down and restart the Oasys Ltd. LS-DYNA environment, and it will use these values rather than those supplied by the operating system.

HP platforms (HP-UX operating) system.

Method 1:

Edit the /usr/lib/X11/X0screens file directly.

As user root edit this file and add this line to the end of the file

MonitorSize *nn* Inches

Where <*nn*> is your monitor size, eg 20.

Then logout completely from the console, revert to "command line" login, and allow the window manager to restore the "desktop login" prompt. This will restart the X server with the new properties.

Method 2:

Use the SAM utility to change this file.

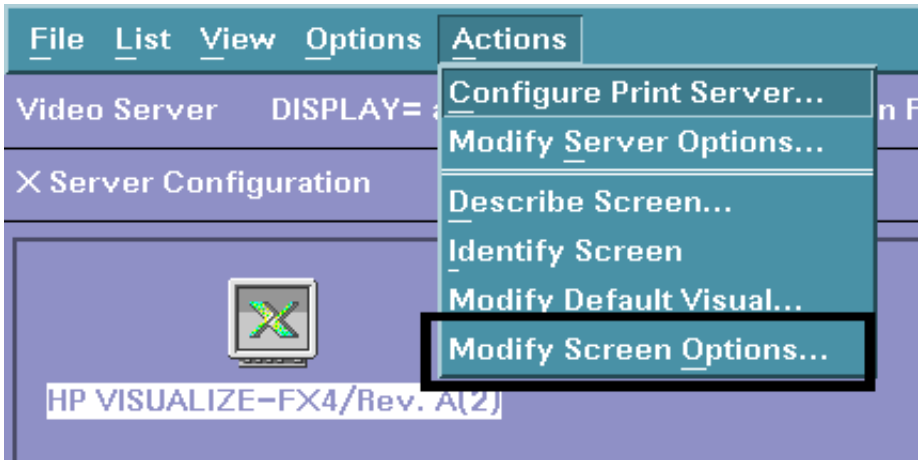
As user root start "sam".

Select **Display**

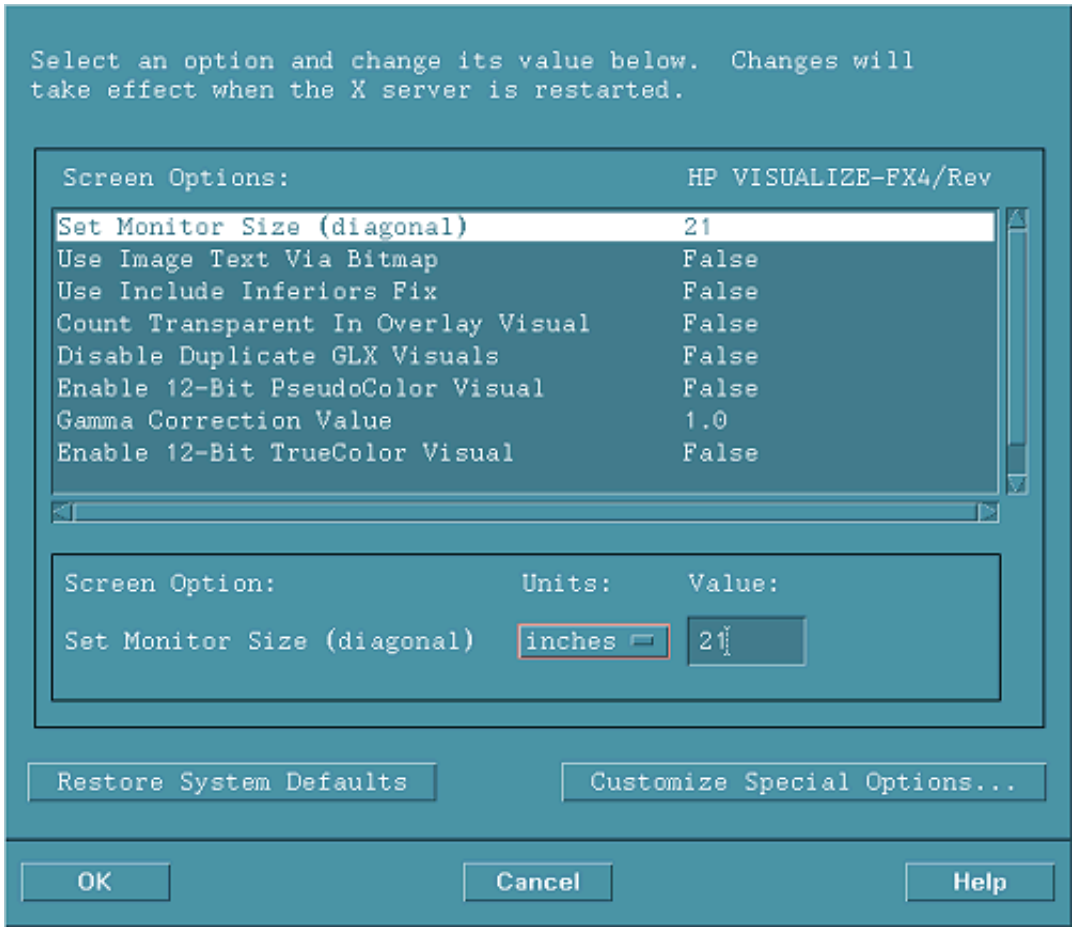
=> **X Server Configuration**

Select your graphics card, and from the **Actions** pull-down menu select:

Modify Screen Options



Set the monitor size to the required dimensions, and follow the instructions about restarting.



On Compaq OSF 4/5 systems

Edit the /usr/lib/X11/Xserver.conf file
Note: On OSF 5 systems I have found that the Xserver.conf file existed in several different places (eg /etc/X11, /var/...). It was clear from the documentation which one it should have been using, and it was equally clear from trial and error that it was using a different one.

I would suggest that you do a "find" on your system to find all occurrences, ie:

```
find / -name Xserver.conf -print
```

and perform the edits given here in each in turn until you find one that works.

Thanks Compaq!

As root edit this file.
Look for the line starting

```
args <
```

Add to the end of the following argument list

```
-dpi nn
```

Where <*nn*> is the screen resolution in dots per inch.

For example on my machine these lines now read:

```
args <

-pn -nice 2 -wm -dpi 90

>
```

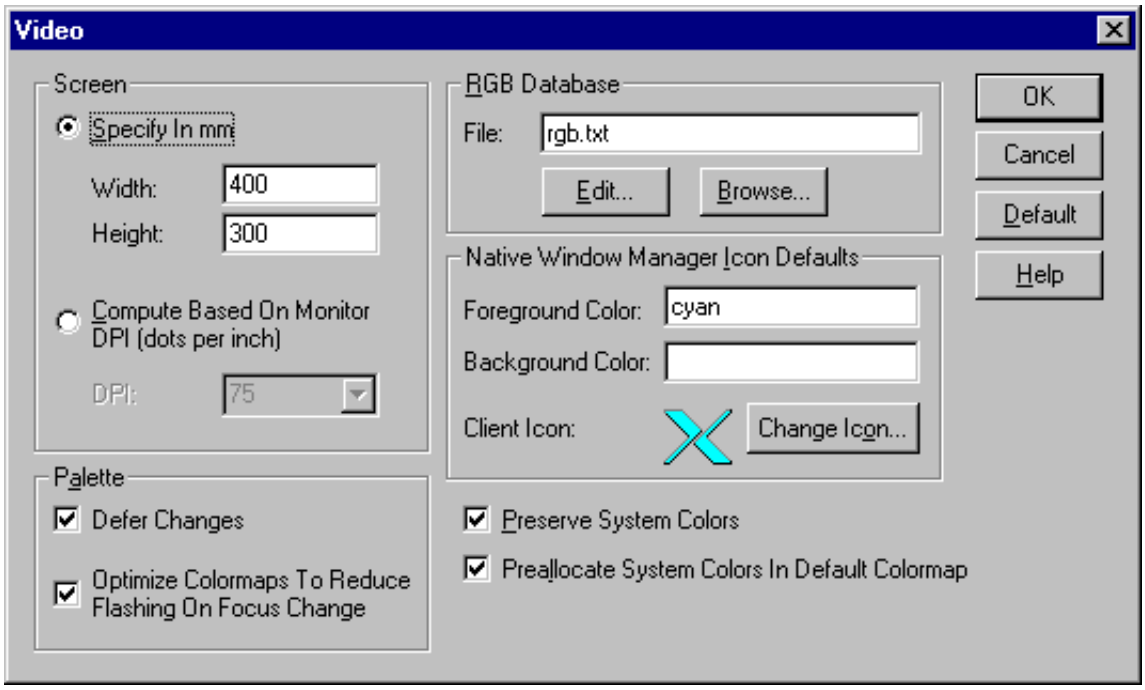
To establish the <nn> value in dots per inch:

- Find the screen resolution from xdpyinfo if unknown (eg 1280 x 1024)
- Measure the vertical and horizontal visible dimensions in inches with a ruler (eg 15" x 12")
- Calculate the average (here of 1280/15 and 1024/12) to get a "dots per inch" value (here 85)

Finally logout of the console, taking it right back to command line mode, then allow the X server to restart the graphical login with the new properties.

On PCs running Windows using the Exceed emulator.

From the main Exceed directory use



In the **Video** panel set the measured (this time in mm) screen dimensions.

Here my (21") display is 400x300mm.

You will need to restart Exceed to make these settings take effect.

Installation organisation

The version 16 installation can be customised to try and avoid a number of issues that often occur in large organisations with many users.

- Large organisations generally imply large networks, and it is often the case that the performance of these networks can be intermittent or poor, therefore it is common practice to perform an installation of the software on the local disk of each machine, rather then having a single installation on a remote disk.

  This avoids the pauses and glitches that can occur when running executable files over a network, but it also means that all the configuration files in, or depending upon, the top level "Admin" directory have to be copied to all machines and, more to the point, any changes or additions to such files also have to be copied to all machines.

- In larger organisations the "one person per computer" philosophy may not apply, with the consequence that users will tend to have a floating home area on a network drive and may not use the same machine every day.

  This is not usually a problem on Linux where the "home" directory is tied to the login name not the machine. However on Windows platforms it means that %USERPROFILE%, which is typically on the local C drive of a machine, is not a good place to consider as "home" since it will be tied to a given computer, therefore a user who saves a file in his home directory on machine A may not be able to access it from machine B.

- In a similar vein placing large temporary files on the /tmp partition (Linux) or the C: drive (Windows) may result in local disks becoming too full, or quotas exceeded.

This section gives only a brief summary of the installation organisation, and you should refer to the separate Installation Guide if you want to find out more about the details of installation, licensing, and other related issues.

Version 16.0 Installation structure

In version 16.0 the option is provided to separate a top-level ’administration’ directory from the ’installation’ one where the executables are located.

For large installations on many machines this allows central configuration and administration files to exist in one place only, but executables to be installed locally on users’ machines to give better performance. Version 16.0 also allows the following items to be configured

- The location for user manuals and other documentation.
- The definition of a user’s home directory.
- The definition of the temporary directory for scratch files.

In addition parsing of the ’oa_pref’ (preferences) file will now handle environment variables, so that a generic preference can be configured to give a user-specific result, and preferences may be ’locked’ so that those set at the administration level cannot be changed by users.

These changes are entirely optional, and users performing a simple installation on a single machine do not need to make any changes to their existing installation practice.

Directory	**Status**	**Directory Content and purpose**	**oa_pref file option**
`OA_ADMIN_xx`	*Optional*	Top level configuration files. (**`xx`** =16 for release 16.0, thus **`OA_ADMIN_16`**) Admin level oa_pref file Other configuration files Timeout configuration file	

OA_ADMIN	*Optional*	Same as **OA_ADMIN_16**, provided for backwards compatibility with earlier releases. It is recommended that plain **OA_ADMIN**, without the **_xx** version suffix, is not used since otherwise there is no easy way of distinguishing between parallel installations of different releases of the Oasys Ltd software in an installation. *If* **OA_ADMIN_16** *is not defined then this non-release specific version is checked.*	
OA_INSTALL_xx	*Optional*	(**xx** =16 for release 16.0, thus **OA_ADMIN_16** All executables Installation level oa_pref file	**oasys*install_dir: <*pathname*>**
OA_INSTALL	*Optional*	Same as **OA_INSTALL_16**. If no "**OA_ADMIN_xx**" directory is used and all software is simply placed in this "install" directory, which would be typical of a single-user installation, then it is recommended that the **_xx** version suffix is used in order to keep parallel installations of different releases of the Oasts Ltd software separate on the machine. *If* **OA_INSTALL_16** *is not defined then this non-release specific version is checked*	**oasys*install_dir: <*pathname*>**
OA_MANUALS	*Optional*	Specific directory for user manuals. If not defined then will search in: **OA_ADMIN_xx/manuals** (xx = major version number) **OA_INSTALL/manuals**	**oasys*manuals_dir: <*pathname*>**
OA_HOME	*Optional*	Specific "home" directory for user when using Oasys Ltd software. If not defined will use: **$HOME** (Linux) **%USERPROFILE%** (Windows)	**oasys*home_dir: <*pathname*>**
OA_TEMP	*Optional*	Specific "temporary" directory for user when using Oasys Ltd software. If not defined will use: **P_tmpdir** (Linux, typically /tmp) **%TEMP%** (Windows, typically C:\temp)	**oasys*temp_dir: <*pathname*>**

It will be clear from the table above that no Environment variables have to be set, and that all defaults will revert to pre-9.4 behaviour. In other words users wishing to keep the status quo will find behaviour and layout unchanged if they do nothing.

OA_INSTALL_XX
Previously the software used the **OA_INSTALL** (renamed from **OASYS**) environment variable to locate the directory the software was installed in.

- On Windows this is no longer required as the software can work out its own installation directory. As this environment variable is no longer required it is recommended that it is removed from machines it is currently set on as in some cases where more than one version has been installed in different directories it can cause problems.

- On LINUX systems the "oasys_16" script that starts the SHELL automatically sets this Environment Variable and passes it to any application started from the SHELL. If you run applications directly from the command line and bypass the SHELL then you should set **OA_INSTALL_XX** so that the software can locate manuals and other required files.

OA_ADMIN_XX
Users wishing to separate configuration and installation directories will be able to do so by making use of the new top level **OA_ADMIN_xx** directory.

Installation Examples

The following diagrams illustrate how the installation might be organised in various different scenarios..

a) Single user installation on one machine

There is no need to worry about separating administration and installation directories, and the default installation of all files in and below the single installation directory will suffice.

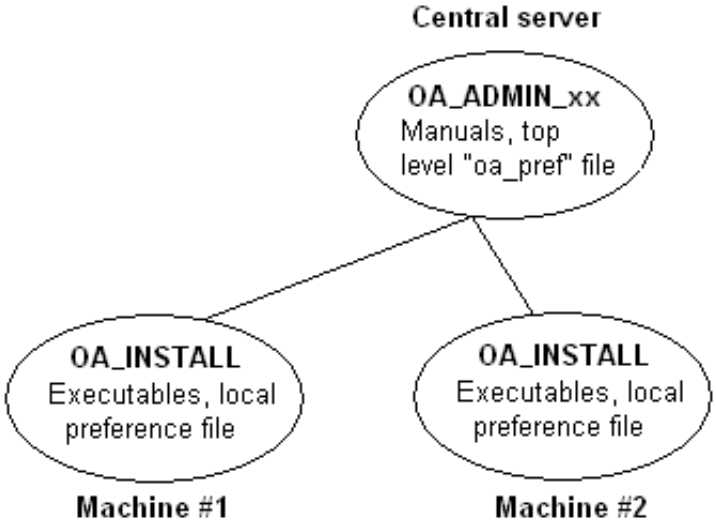
It is suggested that the **_xx** version suffix of **OA_INSTALL_xx** is used in order to keep parallel installations of different releases of the Oassys Ltd software separate on the machine.

b) A few machines on a small network, each user has his own machine

The top level administration directory can be installed on a network server, possibly also locating the manuals centrally.

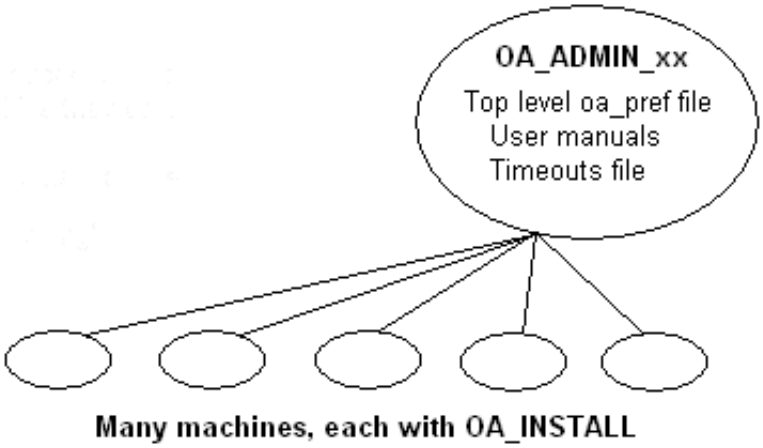
Each user's machine has its own 'installation' directory to give good performance, but there is no need to manage home or temporary directories centrally since each user 'owns' his machine.

If network performance is good an alternative would be to install executables on the central server, meaning that local OA_INSTALL directories are not required.



c) Large corporate network

There is no need to worry about separating administration and installation directories, and the default installation of all files in and below the single installation directory will suffice.



Dynamic configuration using the top level oa_pref file.

A further improvement is that all environment variables below **OA_ADMIN_xx** may either be set explicitly, or dynamically using the options in the oa_pref file at the top **OA_ADMIN_xx** level. This permits parallel installations of different versions of the software to co-exist, with only the top level administration directory names being distinct. For example:

Release 16.0	**Release 16.1**
Top level directory **OA_ADMIN_16**	Top level directory **OA_ADMIN_161**
oa_pref file in **OA_ADMIN_16** contains: **oasys*install_dir:** *<pathname for **16.0** installation>* **oasys*manuals_dir:** *<pathname for **16.0** manuals>* **oasys*home_dir:** *<pathname for home directory>* **oasys*temp_dir:** *<pathname for temporary files>*	oa_pref file in **OA_ADMIN_161** contains: **oasys*install_dir:** *<pathname for **16.1** installation>* **oasys*manuals_dir:** *<pathname for**16.1** manuals>* } would almost certainly be unchanged between major } versions, although they could be different if desired
Pathnames in the oa_pref file may contain environment variables which will be resolved before being applied.	

The hierarchy of oa_pref file reading

It will be clear from the above that in a large installation the "oa_pref" files have a significant role. Each piece of software reads them in the following order:

OA_ADMIN_xx	Top level configuration
OA_INSTALL_xx	Installation level
OA_HOME	User’s personal "home" file
Current working directory	File specific to the current directory (rarely used)

The rules for reading these files are:

- If a given directory does not exist, or no file is found in that directory, then no action is taken. This is not an error.
- A more recently read definition supersedes one read earlier, therefore "local" definitions can supersede "global" ones (unless it was locked).
- If two of more of the directories in the table above are the same then that file is only read once from the first instance.

Locking Preference Options

From version 9.4 onwards preference options can be locked. If a preference option is locked in a file then that preference option will be ignored in any of the subsequent preference files that are read.

Therefore by locking a preference in a top-level file in the hierarchy above, eg in **OA_ADMIN_xx**, and then protecting that file to be read-only, an administrator can set preferences that cannot be altered by users since any definitions of that preference in their private oa_pref files will be ignored.

Preferences are locked by using a hash (#) rather than an asterisk (*) between the code name and the preference string. For example:

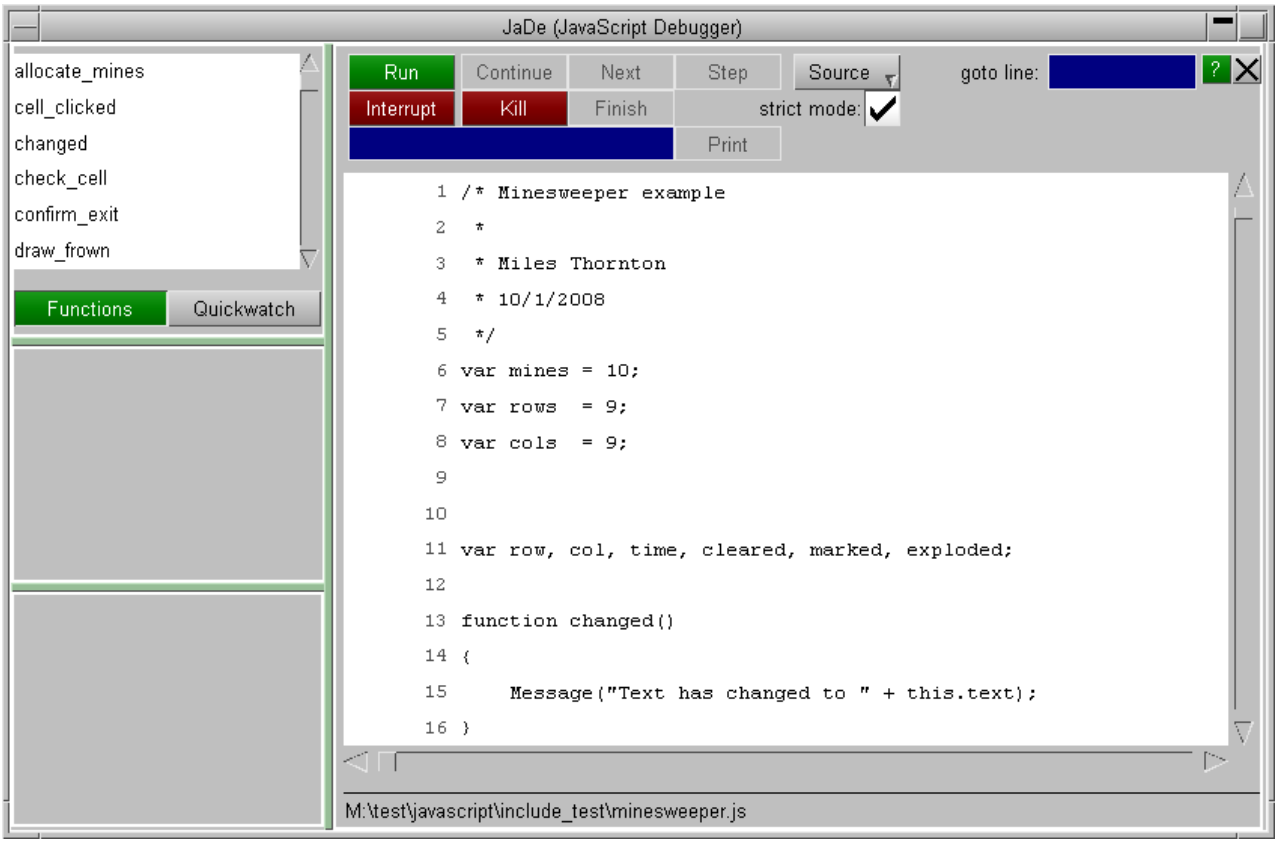
primer*maximise: true	Normal case using "*", means an unlocked preference

`primer#maximise: true`	Locked case using "**#**"

These changes may be made either by editing the file manually, or by using the preferences editor.

JaDe: The JavaScript debugger

JaDe is included in D3PLOT, PRIMER and T/HIS to help debug and develop JavaScripts. It is started by selecting a script and pressing the **Debug** button in the JavaScript menu in any of the programs. The initial screen is shown below.



It is fairly basic but hopefully has enough functionality for people to be able to find and fix problems in scripts.

Viewing the script files and functions

The main part of the window shows the script file. If your script is broken up into separate file (by using Use) then you can get a list of the different files and view them by using the **Source** popup. To go to a particular line in the file use the **goto line** textbox.

A list of the functions in the script is shown in the **Functions** menu on the top left. If you want to look at a particular function then click on the function name and the main text window will jump to the correct file and line.

Adding/removing breakpoints

A breakpoint is a line in the script where execution will pause in JaDe. To add a breakpoint either left click on the line you want the breakpoint on or right click on the line and select **Create breakpoint** from the popup. A red circle is then drawn on the line to show that there is an active breakpoint.

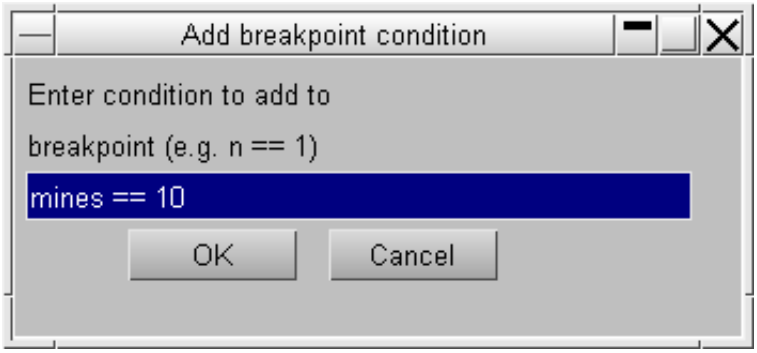
```
112 function allocate_mines()
113 {
114     var n = mines;
115
116     while (n)
117     {
```

Additionally the breakpoint will also be added to the list in the breakpoint window (bottom left of JaDe). You can click on this at any time and the main text window will jump to the correct file and line.
Active breakpoints are shown with a red circle. Breakpoints can be activated/deactivated by clicking on the line again. Unactive breakpoints are shown as a grey circle instead of a red one. They are also shown in grey text in the breakpoint window .

To delete a breakpoint right click on the line and select **Delete breakpoint**. The breakpoint will be deleted.

Conditional breakpoints

Sometimes it is useful to only stop at a breakpoint if a certain condition is met. For example in the above example we may only want to stop at line 114 if `mines` is 10. You can do this by right clicking on the the breakpoint and selecting **Add condition**.



A window is mapped allowing you type in the condition you want to try to meet. The condition should be a JavaScript expression which evaluates to true if you want the breakpoint to stop execution, or false if you want the breakpoint to be skipped. In this example the condition is `n == 10`.

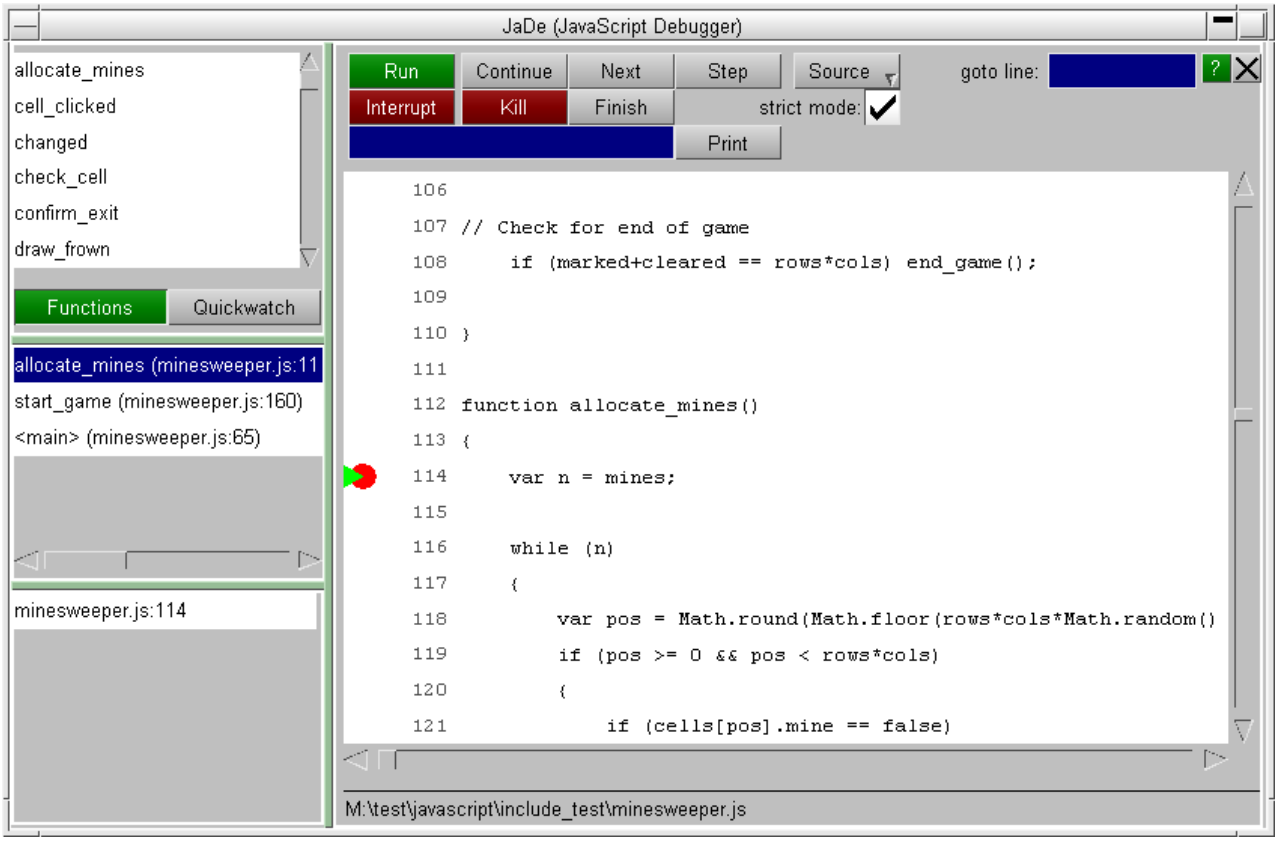
If a breakpoint has a condition associated with it a C is drawn on the circle and in the breakpoint window. The condition can be edited again or removed by right clicking on the breakpoint and selecting either **Edit condition** or **Remove condition** from the popup.

Running the script

Running the script is controlled by the buttons at the top of the debugger window. By default the script will be run in the debugger in 'strict mode'. This tries to pick up things which you might not have intended by running the script in a stricter environment doing more checking. You can toggle this on/off by using the **strict mode** checkbox.

Starting and stopping

To start the script press the **Run** button. Execution of the script will start. If you have not defined any breakpoints then the script will run until it finishes (unless there are some script errors or exceptions). If there is a breakpoint then the debugger will stop execution of the script when it reaches it. If the script is running and you want to pause execution of the script at any time you can press **Interrupt**.



The line that the debugger has paused the script on is shown by a green triangle. In the above example it is paused at line 114. The middle panel on the left shows the call stack. See the call stack section below for more details.

Stepping and continuing

Once the script is paused in the debugger you can step through the source code by using the **Continue**, **Next**, **Step** and **Finish** buttons.

Continue will resume execution of the script again.
Next continues to the next line in the current function. i.e. it will step *over* a function call.
Step continues execution to the next source line (which may be in a different function. i.e. it will step *into* a function call).
Finish will finish executing the current function and stop at the next line in the calling function (the function above this in the call stack).

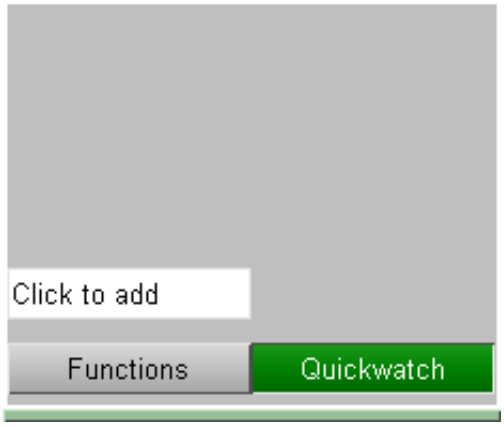
Alternatively, if you want to continue until a particular line you can right click on the line you want to continue until and select **Continue to here** from the popup.

Printing the value of a variable

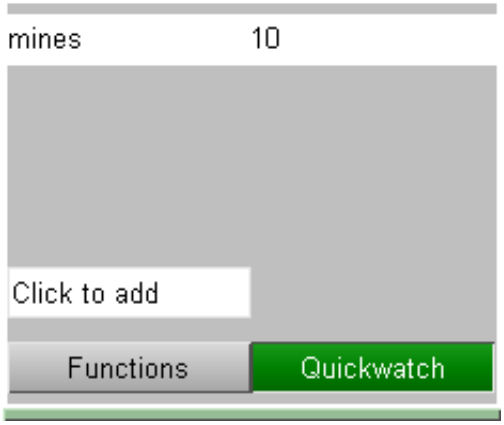
If you want to see the value of a variable you can type the name of the variable you want to see in the textbox at the top of the debugger and press **Print**. JaDe will evaluate the variable and output the result in the statusbar at the bottom of the debugger.

Using Quickwatch

If you want to look at the values for lots of variables it is annoying to have to type the variable name in and press **Print** for each one. A better way is to use **Quickwatch** at the top left of JaDe

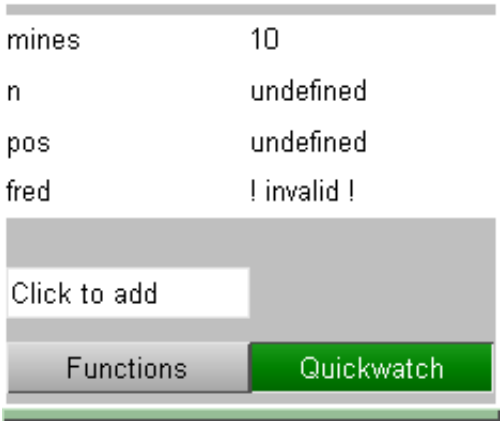


Type the name of the variable that you want to watch in the **Click to add** textbox. A line will be added for the variable showing its name and value. e.g. in the following image the variable `mines` is being displayed and its current value is `10`. If the value is very long hover over the value to get the whole string.



You can add any number of variables to watch. To remove one right click on the variable and select **Remove quickwatch** from the popup.

If a variable exists and has been assigned to then the value is displayed. e.g. `mines` in the following example.
If the variable exists but it has not yet had a value assigned its value is the `undefined` value. e.g. `pos` in the following example.
If the variable does not exist the value is shown as `! invalid !` . e.g. `fred` in the following example.



The call stack

The call stack shows which functions have been called in the script to get to the current point. It is the middle left window in JaDe.

The top line shows the function that the script is currently paused at. The other lines show the calling functions in order. The above example can be read as:

1. The script starts
2. On line 65 in script file minesweeper.js in the 'main' program the function `start_game` is called.
3. On line 160 in script file minesweeper.js in function `start_game` the function `allocate_mines` is called
4. On line 114 in script file minesweeper.js in function `allocate_mines` the script is paused.

This information is sometimes very useful in more complicated scripts to find out the order things are done in.

The function that the user is currently looking at is highlighted in blue. You can move up or down the call stack by clicking on a line. The main text window will jump to the correct file and line. The line will be shown with a blue triangle instead of a green triangle.

Exceptions

Sometimes when developing a script you get errors that you need to try to investigate and fix. e.g. an object is null when it should be defined or you try to call a method that does not exist for an object. In these cases an exception is thrown by JavaScript and the script would terminate is run normally. JaDe will trap the exception and stop at the line where the exception occured. e.g. If for example you has the following code:

```
var w = new Window('Example', 0.5, 1.0, 0.5, 1.0);
w.BadMethod();
w.Show()
```

There is no method called `BadMethod` for a `Window`. JaDe will stop at this point and allow you to look at the script.

Licences used in software

The Oasys LS-DYNA environment Ltd software uses several third party libraries and executables. The licences for them are given below

Expat

Copyright (c) 1998, 1999, 2000 Thai Open Source Software Center Ltd
and Clark Cooper
Copyright (c) 2001, 2002, 2003, 2004, 2005, 2006 Expat maintainers.
Permission is hereby granted, free of charge, to any person obtaining
a copy of this software and associated documentation files (the
"Software"), to deal in the Software without restriction, including
without limitation the rights to use, copy, modify, merge, publish,
distribute, sublicense, and/or sell copies of the Software, and to
permit persons to whom the Software is furnished to do so, subject to
the following conditions:
The above copyright notice and this permission notice shall be included
in all copies or substantial portions of the Software.
THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY
CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT,
TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE

FreeType

Portions of this software are copyright The FreeType Project (www.freetype.org). All rights reserved.

```
                    The FreeType Project LICENSE
                    ----------------------------
                            2006-Jan-27
                    Copyright 1996-2002, 2006 by
          David Turner, Robert Wilhelm, and Werner Lemberg
Introduction
============
  The FreeType  Project is distributed in  several archive packages;
  some of them may contain, in addition to the FreeType font engine,
  various tools and  contributions which rely on, or  relate to, the
  FreeType Project.
  This  license applies  to all  files found  in such  packages, and
  which do not  fall under their own explicit  license.  The license
  affects  thus  the  FreeType  font  engine,  the  test  programs,
  documentation and makefiles, at the very least.
  This  license  was  inspired  by  the  BSD,  Artistic,  and  IJG
  (Independent JPEG  Group) licenses, which  all encourage inclusion
  and  use of  free  software in  commercial  and freeware  products
  alike.  As a consequence, its main points are that:
    o We don't promise that this software works. However, we will be
      interested in any kind of bug reports. ('as is' distribution)
    o You can  use this software for whatever you  want, in parts or
      full form, without having to pay us. ('royalty-free' usage)
    o You may not pretend that  you wrote this software.  If you use
      it, or  only parts of it,  in a program,  you must acknowledge
      somewhere  in  your  documentation  that  you  have  used  the
      FreeType code. ('credits')
  We  specifically  permit  and  encourage  the  inclusion  of  this
  software, with  or without modifications,  in commercial products.
  We  disclaim  all warranties  covering  The  FreeType Project  and
  assume no liability related to The FreeType Project.
  Finally,  many  people  asked  us  for  a  preferred  form  for  a
  credit/disclaimer to use in compliance with this license.  We thus
  encourage you to use the following text:
   """
```

```
    Portions of this software are copyright    The FreeType
    Project (www.freetype.org).  All rights reserved.
   """
  Please replace  with the value from the FreeType version you
  actually use.
Legal Terms
===========
0. Definitions
--------------
  Throughout this license,  the terms 'package', 'FreeType Project',
  and 'FreeType  archive' refer  to  the  set  of files  originally
  distributed  by the  authors  (David Turner,  Robert Wilhelm,  and
  Werner Lemberg) as the 'FreeType Project', be they named as alpha,
  beta or final release.
  'You' refers to  the licensee, or person using  the project, where
  'using' is a generic term including compiling the project's source
  code as  well as linking it  to form a  'program' or 'executable'.
  This  program is  referred to  as  'a program  using the  FreeType
  engine'.
  This  license applies  to all  files distributed  in  the original
  FreeType  Project,  including  all  source  code,  binaries  and
  documentation,  unless  otherwise  stated  in  the  file  in  its
  original, unmodified form as  distributed in the original archive.
  If you are  unsure whether or not a particular  file is covered by
  this license, you must contact us to verify this.
  The FreeType  Project is copyright (C) 1996-2000  by David Turner,
  Robert Wilhelm, and Werner Lemberg.  All rights reserved except as
  specified below.
1. No Warranty
--------------
  THE FREETYPE PROJECT  IS PROVIDED 'AS IS' WITHOUT  WARRANTY OF ANY
  KIND, EITHER  EXPRESS OR IMPLIED,  INCLUDING, BUT NOT  LIMITED TO,
  WARRANTIES  OF  MERCHANTABILITY  AND  FITNESS  FOR  A  PARTICULAR
  PURPOSE.  IN NO EVENT WILL ANY OF THE AUTHORS OR COPYRIGHT HOLDERS
  BE LIABLE  FOR ANY DAMAGES CAUSED  BY THE USE OR  THE INABILITY TO
  USE, OF THE FREETYPE PROJECT.
2. Redistribution
-----------------
  This  license  grants  a  worldwide, royalty-free,  perpetual  and
  irrevocable right  and license to use,  execute, perform, compile,
  display,  copy,  create  derivative  works  of,  distribute  and
  sublicense the  FreeType Project (in  both source and  object code
  forms)  and  derivative works  thereof  for  any  purpose; and  to
  authorize others  to exercise  some or all  of the  rights granted
  herein, subject to the following conditions:
    o Redistribution of  source code  must retain this  license file
      ('FTL.TXT') unaltered; any  additions, deletions or changes to
      the original  files must be clearly  indicated in accompanying
      documentation.  The  copyright  notices  of  the  unaltered,
      original  files must  be  preserved in  all  copies of  source
      files.
    o Redistribution in binary form must provide a  disclaimer  that
      states  that  the software is based in part of the work of the
      FreeType Team,  in  the  distribution  documentation.  We also
      encourage you to put an URL to the FreeType web page  in  your
      documentation, though this isn't mandatory.
  These conditions  apply to any  software derived from or  based on
  the FreeType Project,  not just the unmodified files.  If you use
  our work, you  must acknowledge us.  However, no  fee need be paid
  to us.
3. Advertising
--------------
  Neither the  FreeType authors and  contributors nor you  shall use
  the name of the  other for commercial, advertising, or promotional
  purposes without specific prior written permission.
  We suggest,  but do not require, that  you use one or  more of the
  following phrases to refer  to this software in your documentation
  or advertising  materials: 'FreeType Project',  'FreeType Engine',
  'FreeType library', or 'FreeType Distribution'.
  As  you have  not signed  this license,  you are  not  required to
  accept  it.  However,  as  the FreeType  Project  is  copyrighted
```

```
  material, only  this license, or  another one contracted  with the
  authors, grants you  the right to use, distribute,  and modify it.
  Therefore,  by  using,  distributing,  or modifying  the  FreeType
  Project, you indicate that you understand and accept all the terms
  of this license.
4. Contacts
-----------
  There are two mailing lists related to FreeType:
    o freetype@nongnu.org
      Discusses general use and applications of FreeType, as well as
      future and  wanted additions to the  library and distribution.
      If  you are looking  for support,  start in  this list  if you
      haven't found anything to help you in the documentation.
    o freetype-devel@nongnu.org
      Discusses bugs,  as well  as engine internals,  design issues,
      specific licenses, porting, etc.
  Our home page can be found at
    http://www.freetype.org
--- end of FTL.TXT ---
```

FFmpeg

```
  FFmpeg is free software; you can redistribute it and/or
  modify it under the terms of the GNU Lesser General Public
  License as published by the Free Software Foundation; either
  version 2.1 of the License, or (at your option) any later version.

  FFmpeg is distributed in the hope that it will be useful,
  but WITHOUT ANY WARRANTY; without even the implied warranty of
  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
  Lesser General Public License for more details.

  You should have received a copy of the GNU Lesser General Public
  License along with FFmpeg; if not, write to the Free Software
  Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA
```

Jpeg

```
The authors make NO WARRANTY or representation, either express or implied,
with respect to this software, its quality, accuracy, merchantability, or
fitness for a particular purpose.  This software is provided "AS IS", and you,
its user, assume the entire risk as to its quality and accuracy.
This software is copyright (C) 1991-2012, Thomas G. Lane, Guido Vollbeding.
All Rights Reserved except as specified below.
Permission is hereby granted to use, copy, modify, and distribute this
software (or portions thereof) for any purpose, without fee, subject to these
conditions:
(1) If any part of the source code for this software is distributed, then this
README file must be included, with this copyright and no-warranty notice
unaltered; and any additions, deletions, or changes to the original files
must be clearly indicated in accompanying documentation.
(2) If only executable code is distributed, then the accompanying
documentation must state that "this software is based in part on the work of
the Independent JPEG Group".
(3) Permission for use of this software is granted only if the user accepts
full responsibility for any undesirable consequences; the authors accept
NO LIABILITY for damages of any kind.
These conditions apply to any software derived from or based on the IJG code,
not just to the unmodified library.  If you use our work, you ought to
acknowledge us.
Permission is NOT granted for the use of any IJG author's name or company name
in advertising or publicity relating to this software or products derived from
it.  This software may be referred to only as "the Independent JPEG Group's
software".
We specifically permit and encourage the use of this software as the basis of
commercial products, provided that all warranty or liability claims are
assumed by the product vendor.
```

Libcurl

COPYRIGHT AND PERMISSION NOTICE
Copyright (c) 1996 - 2012, Daniel Stenberg, <daniel@haxx.se>.
All rights reserved.
Permission to use, copy, modify, and distribute this software for any purpose with or without fee is hereby granted, provided that the above copyright notice and this permission notice appear in all copies.
THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT OF THIRD PARTY RIGHTS. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
Except as contained in this notice, the name of a copyright holder shall not be used in advertising or otherwise to promote the sale, use or other dealings in this Software without prior written authorization of the copyright holder.

Libfame

libfame - Fast Assembly MPEG Encoder Library
Copyright (C) 2000-2001 Vivien Chappelier
This library is free software; you can redistribute it and/or modify it under the terms of the GNU Library General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.
This library is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Library General Public License for more details.
You should have received a copy of the GNU Library General Public License along with this library; if not, write to the Free Software Foundation, Inc., 675 Mass Ave, Cambridge, MA 02139, USA.

Libgif

The GIFLIB distribution is Copyright (c) 1997 Eric S. Raymond
Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:
The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.
THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Libpng

This copy of the libpng notices is provided for your convenience. In case of any discrepancy between this copy and the notices in the file png.h that is included in the libpng distribution, the latter shall prevail.
COPYRIGHT NOTICE, DISCLAIMER, and LICENSE:
If you modify libpng you may insert additional notices immediately following this sentence.
This code is released under the libpng license.
libpng versions 1.2.6, August 15, 2004, through 1.5.11, June 14, 2012, are Copyright (c) 2004, 2006-2012 Glenn Randers-Pehrson, and are distributed according to the same disclaimer and license as libpng-1.2.5 with the following individual added to the list of Contributing Authors
Cosmin Truta
libpng versions 1.0.7, July 1, 2000, through 1.2.5 - October 3, 2002, are Copyright (c) 2000-2002 Glenn Randers-Pehrson, and are distributed according to the same disclaimer and license as libpng-1.0.6

```
with the following individuals added to the list of Contributing Authors
   Simon-Pierre Cadieux
   Eric S. Raymond
   Gilles Vollant
and with the following additions to the disclaimer:
   There is no warranty against interference with your enjoyment of the
   library or against infringement.  There is no warranty that our
   efforts or the library will fulfill any of your particular purposes
   or needs.  This library is provided with all faults, and the entire
   risk of satisfactory quality, performance, accuracy, and effort is with
   the user.
libpng versions 0.97, January 1998, through 1.0.6, March 20, 2000, are
Copyright (c) 1998, 1999 Glenn Randers-Pehrson, and are
distributed according to the same disclaimer and license as libpng-0.96,
with the following individuals added to the list of Contributing Authors:
   Tom Lane
   Glenn Randers-Pehrson
   Willem van Schaik
libpng versions 0.89, June 1996, through 0.96, May 1997, are
Copyright (c) 1996, 1997 Andreas Dilger
Distributed according to the same disclaimer and license as libpng-0.88,
with the following individuals added to the list of Contributing Authors:
   John Bowler
   Kevin Bracey
   Sam Bushell
   Magnus Holmgren
   Greg Roelofs
   Tom Tanner
libpng versions 0.5, May 1995, through 0.88, January 1996, are
Copyright (c) 1995, 1996 Guy Eric Schalnat, Group 42, Inc.
For the purposes of this copyright and license, "Contributing Authors"
is defined as the following set of individuals:
   Andreas Dilger
   Dave Martindale
   Guy Eric Schalnat
   Paul Schmidt
   Tim Wegner
The PNG Reference Library is supplied "AS IS".  The Contributing Authors
and Group 42, Inc. disclaim all warranties, expressed or implied,
including, without limitation, the warranties of merchantability and of
fitness for any purpose.  The Contributing Authors and Group 42, Inc.
assume no liability for direct, indirect, incidental, special, exemplary,
or consequential damages, which may result from the use of the PNG
Reference Library, even if advised of the possibility of such damage.
Permission is hereby granted to use, copy, modify, and distribute this
source code, or portions hereof, for any purpose, without fee, subject
to the following restrictions:
1. The origin of this source code must not be misrepresented.
2. Altered versions must be plainly marked as such and must not
   be misrepresented as being the original source.
3. This Copyright notice may not be removed or altered from any
   source or altered source distribution.
The Contributing Authors and Group 42, Inc. specifically permit, without
fee, and encourage the use of this source code as a component to
supporting the PNG file format in commercial products.  If you use this
source code in a product, acknowledgment is not required but would be
appreciated.
A "png_get_copyright" function is available, for convenient use in "about"
boxes and the like:
   printf("%s",png_get_copyright(NULL));
Also, the PNG logo (in PNG format, of course) is supplied in the
files "pngbar.png" and "pngbar.jpg (88x31) and "pngnow.png" (98x31).
Libpng is OSI Certified Open Source Software.  OSI Certified Open Source is a
certification mark of the Open Source Initiative.
Glenn Randers-Pehrson
glennrp at users.sourceforge.net
June 14, 2012
```

Libxlsxwriter

Libxlsxwriter is released under a FreeBSD license:
Copyright 2014-2016, John McNamara
All rights reserved.
Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:
1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
The views and conclusions contained in the software and documentation are those of the authors and should not be interpreted as representing official policies, either expressed or implied, of the FreeBSD Project.
Libxlsxwriter includes ‘queue.h‘ from FreeBSD and the ‘minizip‘ component of ‘zlib‘ which have the following licenses:
Queue.h from FreeBSD:
Copyright (c) 1991, 1993
The Regents of the University of California. All rights reserved.
Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:
1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
4. Neither the name of the University nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.
THIS SOFTWARE IS PROVIDED BY THE REGENTS AND CONTRIBUTORS ‘‘AS IS’’ AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
Zlib has the following License/Copyright:
(C) 1995-2013 Jean-loup Gailly and Mark Adler
This software is provided ’as-is’, without any express or implied warranty. In no event will the authors be held liable for any damages arising from the use of this software.
Permission is granted to anyone to use this software for any purpose, including commercial applications, and to alter it and redistribute it freely, subject to the following restrictions:
1. The origin of this software must not be misrepresented; you must not claim that you wrote the original software. If you use this software in a product, an acknowledgment in the product documentation would be appreciated but is not required.
2. Altered source versions must be plainly marked as such, and must not be misrepresented as being the original software.
3. This notice may not be removed or altered from any source distribution.
Jean-loup Gailly Mark Adler
jloup@gzip.org madler@alumni.caltech.edu

Openssl

```
LICENSE ISSUES
  ==============
  The OpenSSL toolkit stays under a double license, i.e. both the conditions of
  the OpenSSL License and the original SSLeay license apply to the toolkit.
  See below for the actual license texts.
  OpenSSL License
  ---------------
/* ====================================================================
 * Copyright (c) 1998-2017 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.openssl.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    openssl-core@openssl.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.openssl.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */
 Original SSLeay License
 -----------------------
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
```

```
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ``AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */
```

PCRE

```
PCRE LICENCE
------------
PCRE is a library of functions to support regular expressions whose syntax
and semantics are as close as possible to those of the Perl 5 language.
Release 7 of PCRE is distributed under the terms of the "BSD" licence, as
specified below. The documentation for PCRE, supplied in the "doc"
directory, is distributed under the same terms as the software itself.
The basic library functions are written in C and are freestanding. Also
included in the distribution is a set of C++ wrapper functions.
THE BASIC LIBRARY FUNCTIONS
---------------------------
Written by:       Philip Hazel
Email local part: ph10
Email domain:     cam.ac.uk
University of Cambridge Computing Service,
Cambridge, England.
Copyright (c) 1997-2008 University of Cambridge
```

```
All rights reserved.
THE C++ WRAPPER FUNCTIONS
-------------------------
Contributed by:   Google Inc.
Copyright (c) 2007-2008, Google Inc.
All rights reserved.
THE "BSD" LICENCE
-----------------
Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions are met:
    * Redistributions of source code must retain the above copyright notice,
      this list of conditions and the following disclaimer.
    * Redistributions in binary form must reproduce the above copyright
      notice, this list of conditions and the following disclaimer in the
      documentation and/or other materials provided with the distribution.
    * Neither the name of the University of Cambridge nor the name of Google
      Inc. nor the names of their contributors may be used to endorse or
      promote products derived from this software without specific prior
      written permission.
THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE
LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
POSSIBILITY OF SUCH DAMAGE.
End
```

PDFHummus

Is licensed under the Apache License:

```
Copyright 2011 Gal Kahana PDFWriter
Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at
    http://www.apache.org/licenses/LICENSE-2.0
Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.
```

POV-Ray

Is licensed under the GNU AFFERO GENERAL PUBLIC LICENSE Version 3, 19 November 2007 which may be found here http://www.povray.org/povlegal.html

Oasys Ltd use the POV-Ray executable in unmodified form as a separate, stand-alone entity. We have not modified the source code or the executable in any way.

We convey the executable as part of our installation package, and in accordance with the licence:

- Users who install POV-Ray must accept the licence terms cited above.
- We provide a download of the POV-Ray executable and source code on our website http://www.oasys-software.com/dyna/en/

SmoothSort

```
Is licensed under the Creative Commons Attribution-ShareAlike 3.0 license which
may be found here:
```

https://creativecommons.org/licenses/by-sa/3.0/legalcode

Oasys Ltd acknowledge Wikibooks as the source of this algorithm, which is used in unmodified form.

Spidermonkey

```
Mozilla Public License Version 2.0
==================================
1. Definitions
--------------
1.1. "Contributor"
    means each individual or legal entity that creates, contributes to
    the creation of, or owns Covered Software.
1.2. "Contributor Version"
    means the combination of the Contributions of others (if any) used
    by a Contributor and that particular Contributor’s Contribution.
1.3. "Contribution"
    means Covered Software of a particular Contributor.
1.4. "Covered Software"
    means Source Code Form to which the initial Contributor has attached
    the notice in Exhibit A, the Executable Form of such Source Code
    Form, and Modifications of such Source Code Form, in each case
    including portions thereof.
1.5. "Incompatible With Secondary Licenses"
    means
    (a) that the initial Contributor has attached the notice described
        in Exhibit B to the Covered Software; or
    (b) that the Covered Software was made available under the terms of
        version 1.1 or earlier of the License, but not also under the
        terms of a Secondary License.
1.6. "Executable Form"
    means any form of the work other than Source Code Form.
1.7. "Larger Work"
    means a work that combines Covered Software with other material, in
    a separate file or files, that is not Covered Software.
1.8. "License"
    means this document.
1.9. "Licensable"
    means having the right to grant, to the maximum extent possible,
    whether at the time of the initial grant or subsequently, any and
    all of the rights conveyed by this License.
1.10. "Modifications"
    means any of the following:
    (a) any file in Source Code Form that results from an addition to,
        deletion from, or modification of the contents of Covered
        Software; or
    (b) any new file in Source Code Form that contains any Covered
        Software.
1.11. "Patent Claims" of a Contributor
    means any patent claim(s), including without limitation, method,
    process, and apparatus claims, in any patent Licensable by such
    Contributor that would be infringed, but for the grant of the
    License, by the making, using, selling, offering for sale, having
    made, import, or transfer of either its Contributions or its
    Contributor Version.
1.12. "Secondary License"
    means either the GNU General Public License, Version 2.0, the GNU
    Lesser General Public License, Version 2.1, the GNU Affero General
    Public License, Version 3.0, or any later versions of those
    licenses.
1.13. "Source Code Form"
    means the form of the work preferred for making modifications.
1.14. "You" (or "Your")
    means an individual or a legal entity exercising rights under this
    License. For legal entities, "You" includes any entity that
    controls, is controlled by, or is under common control with You. For
    purposes of this definition, "control" means (a) the power, direct
    or indirect, to cause the direction or management of such entity,
    whether by contract or otherwise, or (b) ownership of more than
```

fifty percent (50%) of the outstanding shares or beneficial ownership of such entity.

2. License Grants and Conditions

2.1. Grants
Each Contributor hereby grants You a world-wide, royalty-free, non-exclusive license:
(a) under intellectual property rights (other than patent or trademark) Licensable by such Contributor to use, reproduce, make available, modify, display, perform, distribute, and otherwise exploit its Contributions, either on an unmodified basis, with Modifications, or as part of a Larger Work; and
(b) under Patent Claims of such Contributor to make, use, sell, offer for sale, have made, import, and otherwise transfer either its Contributions or its Contributor Version.

2.2. Effective Date
The licenses granted in Section 2.1 with respect to any Contribution become effective for each Contribution on the date the Contributor first distributes such Contribution.

2.3. Limitations on Grant Scope
The licenses granted in this Section 2 are the only rights granted under this License. No additional rights or licenses will be implied from the distribution or licensing of Covered Software under this License. Notwithstanding Section 2.1(b) above, no patent license is granted by a Contributor:
(a) for any code that a Contributor has removed from Covered Software; or
(b) for infringements caused by: (i) Your and any other third party's modifications of Covered Software, or (ii) the combination of its Contributions with other software (except as part of its Contributor Version); or
(c) under Patent Claims infringed by Covered Software in the absence of its Contributions.

This License does not grant any rights in the trademarks, service marks, or logos of any Contributor (except as may be necessary to comply with the notice requirements in Section 3.4).

2.4. Subsequent Licenses
No Contributor makes additional grants as a result of Your choice to distribute the Covered Software under a subsequent version of this License (see Section 10.2) or under the terms of a Secondary License (if permitted under the terms of Section 3.3).

2.5. Representation
Each Contributor represents that the Contributor believes its Contributions are its original creation(s) or it has sufficient rights to grant the rights to its Contributions conveyed by this License.

2.6. Fair Use
This License is not intended to limit any rights You have under applicable copyright doctrines of fair use, fair dealing, or other equivalents.

2.7. Conditions
Sections 3.1, 3.2, 3.3, and 3.4 are conditions of the licenses granted in Section 2.1.

3. Responsibilities

3.1. Distribution of Source Form
All distribution of Covered Software in Source Code Form, including any Modifications that You create or to which You contribute, must be under the terms of this License. You must inform recipients that the Source Code Form of the Covered Software is governed by the terms of this License, and how they can obtain a copy of this License. You may not attempt to alter or restrict the recipients' rights in the Source Code Form.

3.2. Distribution of Executable Form
If You distribute Covered Software in Executable Form then:
(a) such Covered Software must also be made available in Source Code Form, as described in Section 3.1, and You must inform recipients of the Executable Form how they can obtain a copy of such Source Code Form by reasonable means in a timely manner, at a charge no more than the cost of distribution to the recipient; and
(b) You may distribute such Executable Form under the terms of this License, or sublicense it under different terms, provided that the

```
    license for the Executable Form does not attempt to limit or alter
    the recipients' rights in the Source Code Form under this License.
3.3. Distribution of a Larger Work
You may create and distribute a Larger Work under terms of Your choice,
provided that You also comply with the requirements of this License for
the Covered Software. If the Larger Work is a combination of Covered
Software with a work governed by one or more Secondary Licenses, and the
Covered Software is not Incompatible With Secondary Licenses, this
License permits You to additionally distribute such Covered Software
under the terms of such Secondary License(s), so that the recipient of
the Larger Work may, at their option, further distribute the Covered
Software under the terms of either this License or such Secondary
License(s).
3.4. Notices
You may not remove or alter the substance of any license notices
(including copyright notices, patent notices, disclaimers of warranty,
or limitations of liability) contained within the Source Code Form of
the Covered Software, except that You may alter any license notices to
the extent required to remedy known factual inaccuracies.
3.5. Application of Additional Terms
You may choose to offer, and to charge a fee for, warranty, support,
indemnity or liability obligations to one or more recipients of Covered
Software. However, You may do so only on Your own behalf, and not on
behalf of any Contributor. You must make it absolutely clear that any
such warranty, support, indemnity, or liability obligation is offered by
You alone, and You hereby agree to indemnify every Contributor for any
liability incurred by such Contributor as a result of warranty, support,
indemnity or liability terms You offer. You may include additional
disclaimers of warranty and limitations of liability specific to any
jurisdiction.
4. Inability to Comply Due to Statute or Regulation
---------------------------------------------------
If it is impossible for You to comply with any of the terms of this
License with respect to some or all of the Covered Software due to
statute, judicial order, or regulation then You must: (a) comply with
the terms of this License to the maximum extent possible; and (b)
describe the limitations and the code they affect. Such description must
be placed in a text file included with all distributions of the Covered
Software under this License. Except to the extent prohibited by statute
or regulation, such description must be sufficiently detailed for a
recipient of ordinary skill to be able to understand it.
5. Termination
--------------
5.1. The rights granted under this License will terminate automatically
if You fail to comply with any of its terms. However, if You become
compliant, then the rights granted under this License from a particular
Contributor are reinstated (a) provisionally, unless and until such
Contributor explicitly and finally terminates Your grants, and (b) on an
ongoing basis, if such Contributor fails to notify You of the
non-compliance by some reasonable means prior to 60 days after You have
come back into compliance. Moreover, Your grants from a particular
Contributor are reinstated on an ongoing basis if such Contributor
notifies You of the non-compliance by some reasonable means, this is the
first time You have received notice of non-compliance with this License
from such Contributor, and You become compliant prior to 30 days after
Your receipt of the notice.
5.2. If You initiate litigation against any entity by asserting a patent
infringement claim (excluding declaratory judgment actions,
counter-claims, and cross-claims) alleging that a Contributor Version
directly or indirectly infringes any patent, then the rights granted to
You by any and all Contributors for the Covered Software under Section
2.1 of this License shall terminate.
5.3. In the event of termination under Sections 5.1 or 5.2 above, all
end user license agreements (excluding distributors and resellers) which
have been validly granted by You or Your distributors under this License
prior to termination shall survive termination.
************************************************************************
*                                                                      *
*  6. Disclaimer of Warranty                                           *
*  -------------------------                                           *
*                                                                      *
```

```
*  Covered Software is provided under this License on an "as is"       *
*  basis, without warranty of any kind, either expressed, implied, or  *
*  statutory, including, without limitation, warranties that the       *
*  Covered Software is free of defects, merchantable, fit for a        *
*  particular purpose or non-infringing. The entire risk as to the     *
*  quality and performance of the Covered Software is with You.        *
*  Should any Covered Software prove defective in any respect, You     *
*  (not any Contributor) assume the cost of any necessary servicing,   *
*  repair, or correction. This disclaimer of warranty constitutes an   *
*  essential part of this License. No use of any Covered Software is   *
*  authorized under this License except under this disclaimer.         *
*                                                                      *
************************************************************************
************************************************************************
*                                                                      *
*  7. Limitation of Liability                                          *
*  --------------------------                                          *
*                                                                      *
*  Under no circumstances and under no legal theory, whether tort      *
*  (including negligence), contract, or otherwise, shall any           *
*  Contributor, or anyone who distributes Covered Software as          *
*  permitted above, be liable to You for any direct, indirect,         *
*  special, incidental, or consequential damages of any character      *
*  including, without limitation, damages for lost profits, loss of    *
*  goodwill, work stoppage, computer failure or malfunction, or any    *
*  and all other commercial damages or losses, even if such party      *
*  shall have been informed of the possibility of such damages. This   *
*  limitation of liability shall not apply to liability for death or   *
*  personal injury resulting from such party’s negligence to the       *
*  extent applicable law prohibits such limitation. Some               *
*  jurisdictions do not allow the exclusion or limitation of           *
*  incidental or consequential damages, so this exclusion and          *
*  limitation may not apply to You.                                    *
*                                                                      *
************************************************************************
8. Litigation
-------------
Any litigation relating to this License may be brought only in the
courts of a jurisdiction where the defendant maintains its principal
place of business and such litigation shall be governed by laws of that
jurisdiction, without reference to its conflict-of-law provisions.
Nothing in this Section shall prevent a party’s ability to bring
cross-claims or counter-claims.
9. Miscellaneous
----------------
This License represents the complete agreement concerning the subject
matter hereof. If any provision of this License is held to be
unenforceable, such provision shall be reformed only to the extent
necessary to make it enforceable. Any law or regulation which provides
that the language of a contract shall be construed against the drafter
shall not be used to construe this License against a Contributor.
10. Versions of the License
---------------------------
10.1. New Versions
Mozilla Foundation is the license steward. Except as provided in Section
10.3, no one other than the license steward has the right to modify or
publish new versions of this License. Each version will be given a
distinguishing version number.
10.2. Effect of New Versions
You may distribute the Covered Software under the terms of the version
of the License under which You originally received the Covered Software,
or under the terms of any subsequent version published by the license
steward.
10.3. Modified Versions
If you create software not governed by this License, and you want to
create a new license for such software, you may create and use a
modified version of this License if you rename the license and remove
any references to the name of the license steward (except to note that
such modified license differs from this License).
10.4. Distributing Source Code Form that is Incompatible With Secondary
Licenses
```

```
If You choose to distribute Source Code Form that is Incompatible With
Secondary Licenses under the terms of this version of the License, the
notice described in Exhibit B of this License must be attached.
Exhibit A - Source Code Form License Notice
-------------------------------------------
  This Source Code Form is subject to the terms of the Mozilla Public
  License, v. 2.0. If a copy of the MPL was not distributed with this
  file, You can obtain one at http://mozilla.org/MPL/2.0/.
If it is not possible or desirable to put the notice in a particular
file, then You may include the notice in a location (such as a LICENSE
file in a relevant directory) where a recipient would be likely to look
for such a notice.
You may add additional accurate notices of copyright ownership.
Exhibit B - "Incompatible With Secondary Licenses" Notice
---------------------------------------------------------
  This Source Code Form is "Incompatible With Secondary Licenses", as
  defined by the Mozilla Public License, v. 2.0.
```

Win-iconv

```
win_iconv is a iconv implementation using Win32 API to convert.
win_iconv is placed in the public domain.
Yukihiro Nakadaira <yukihiro.nakadaira@gmail.com>
```

Zlib

```
    (C) 1995-2013 Jean-loup Gailly and Mark Adler
    This software is provided 'as-is', without any express or implied
    warranty.  In no event will the authors be held liable for any damages
    arising from the use of this software.
    Permission is granted to anyone to use this software for any purpose,
    including commercial applications, and to alter it and redistribute it
    freely, subject to the following restrictions:
    1. The origin of this software must not be misrepresented; you must not
       claim that you wrote the original software. If you use this software
       in a product, an acknowledgment in the product documentation would be
       appreciated but is not required.
    2. Altered source versions must be plainly marked as such, and must not be
       misrepresented as being the original software.
    3. This notice may not be removed or altered from any source distribution.
    Jean-loup Gailly        Mark Adler
    jloup@gzip.org          madler@alumni.caltech.edu
```

Fonts on Linux

Like earlier versions the Oasys suite version 16 uses "legacy" X11 fixed fonts on Linux. (This will change in version 17.)

This can cause problems on more recent Linux distributions which do not include the required fonts packages by default, symptoms being any or all of:

- Messages about missing fonts when software starts.
- Some fonts are missing in the user interface.
- Some or all fonts look wrong.

This problem can be solved by installing the missing "xorg" legacy font packages on the system. The following are a typical minimum installation, more may be available if you search your repository for "xorg" fonts.

You will need root privileges to install these, so unless you are familiar with working as root and using commands such as "rpm", "yum" or "yast" please seek help from your IT department, or alternatively contact Oasys Ltd.

RedHat / CentOS 7

```
xorg-x11-fonts-100dpi
xorg-x11-fonts-75dpi
xorg-x11-fonts-ISO8859-1-100dpi
xorg-x11-fonts-ISO8859-1-75dpi
xorg-x11-fonts-Type1
xorg-x11-fonts-misc
```

You don’t have to install all of these.

The 75dpi and 100dpi font packages are the same typefaces at different resolutions. You should choose the one which gives the best looking results on your display, but on modern displays the 100dpi one is likely to look better.

SUSE 12

```
xorg-x11-fonts-core
xorg-x11-fonts
```