

D3PLOT 17.0

D3PLOT 17.0 – Contents

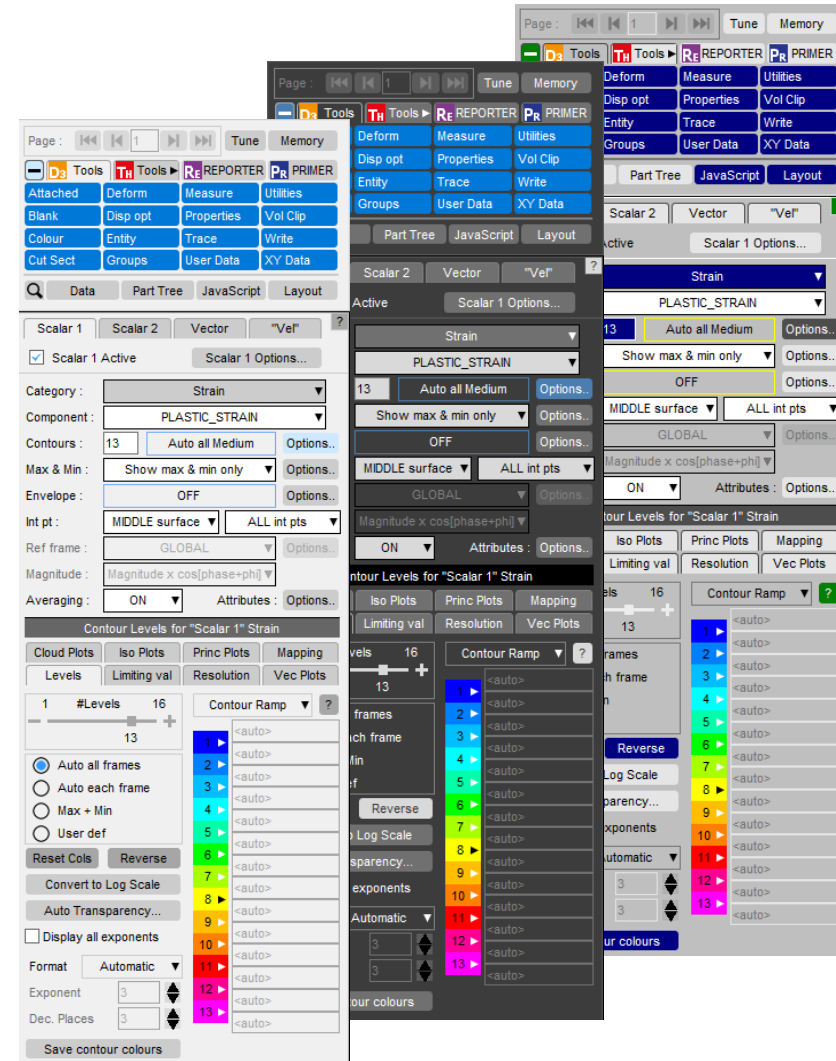
- [User Interface Improvements](#)
- [Shell Thickness](#)
- [Add REPORTER to your Workflow](#) (a new, integrated D3PLOT ↔ T/HIS ↔ REPORTER link)
- [User-defined Colours](#)
- [Speed Improvements](#)
- [Tune Panel and Threading](#)
- [D3PLOT → PRIMER link](#)
- [Material Attributes – Lighting](#)
- [New Preferences](#)
- [Reloading Program Settings](#)
- [Command Line Options](#)
- [JavaScript](#)
- [Replay of Drag-Drop Files](#)

User Interface Improvements

Themes

For a more comfortable and accessible experience:

- We have introduced three new theme options:
 - Light
 - Dark
 - Classic
- Buttons have been flattened with their corners rounded to give a more modern look and feel.

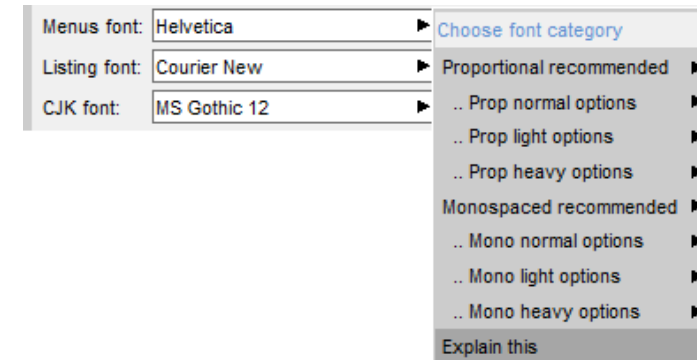
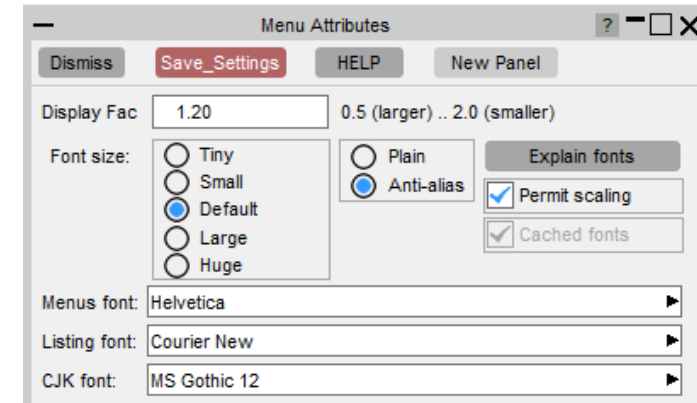


Improved Fonts

To make it easier to create publication-quality images and simplify the communication of your work:

- The range of font sizes is extended to cope with a wider range of displays.
- Font quality has been improved to include an 'anti-alias' option by default for better appearance.
- A wider range of fonts are now supported for menus.

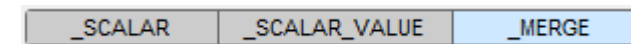
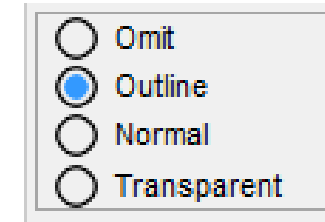
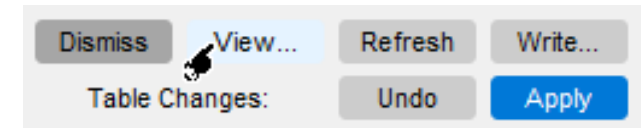
These options can be accessed via the Menu Attributes panel.



Buttons

To make buttons easier and more intuitive to use:

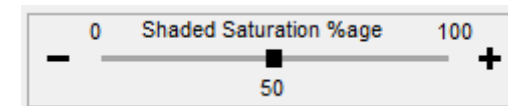
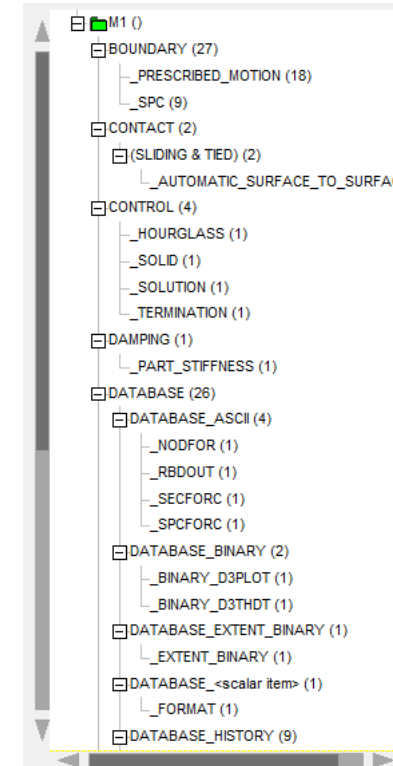
- Clickable buttons are now distinguishable on hover by a slight colour change.
- Radio buttons are now a standard size and shape to be more distinct from tickboxes and more recognisable as radio buttons.
- Button spacing has been adjusted for toggle button groups, so that users understand that they are linked together.



Scroll Bars and Sliders

To make navigating lists quicker and easier:

- Scroll bar length is now related to the number of list items visible.
- Left-clicking on an empty area of the scroll bar allows users to jump through sections of the list.
- Right-clicking on an empty area on the scroll bar allows the user to jump to that specific point in the list.
- Slider appearance has been improved to make them more distinct from scroll bars.

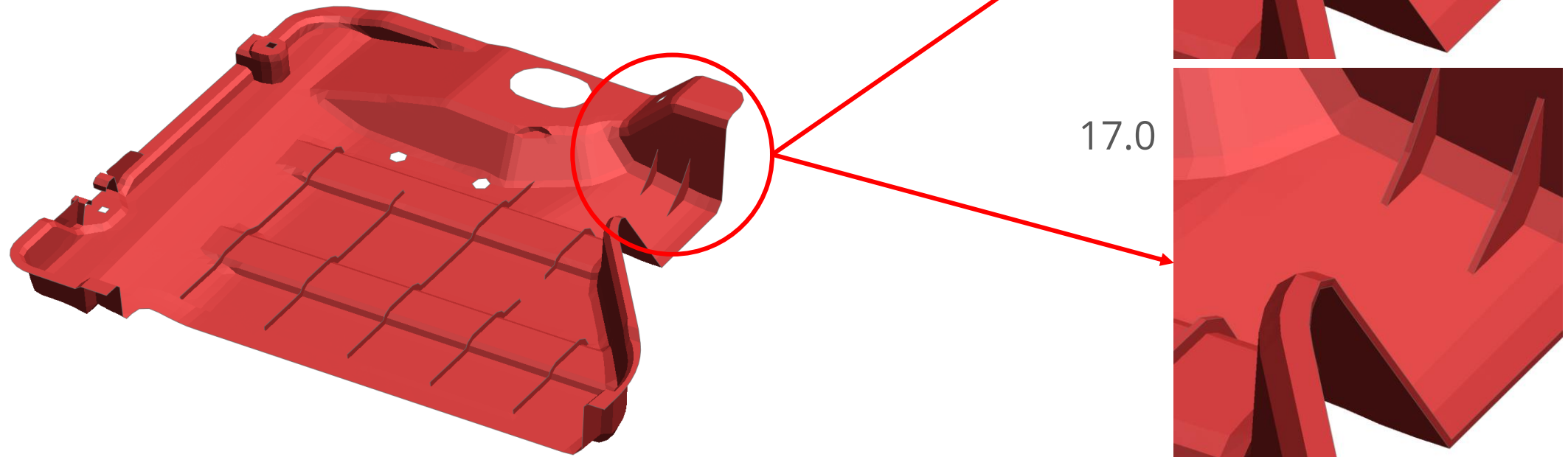


Shell Thickness

Shell Thickness

D3PLOT can now display shell elements showing their true thickness.

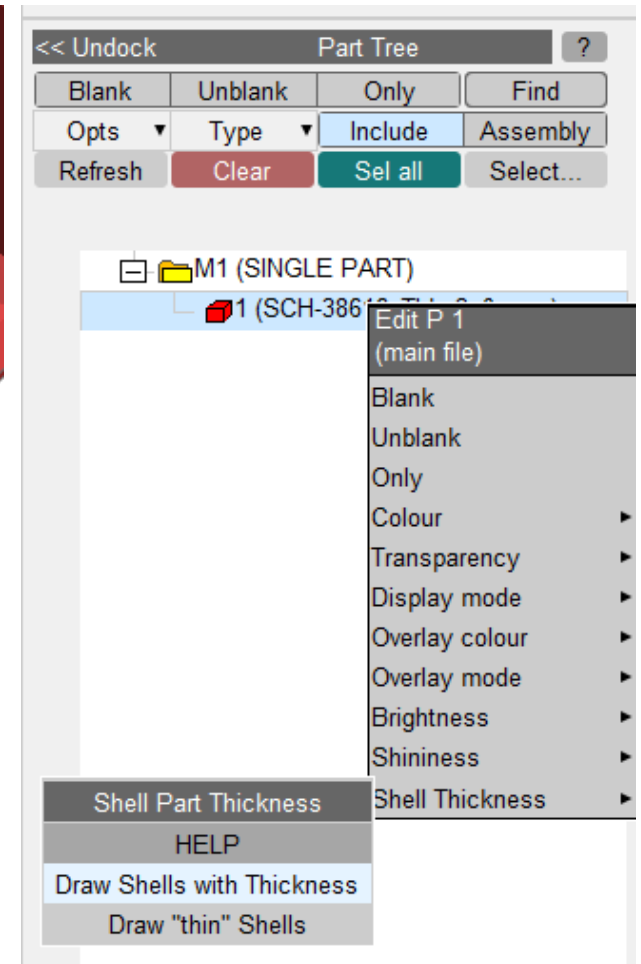
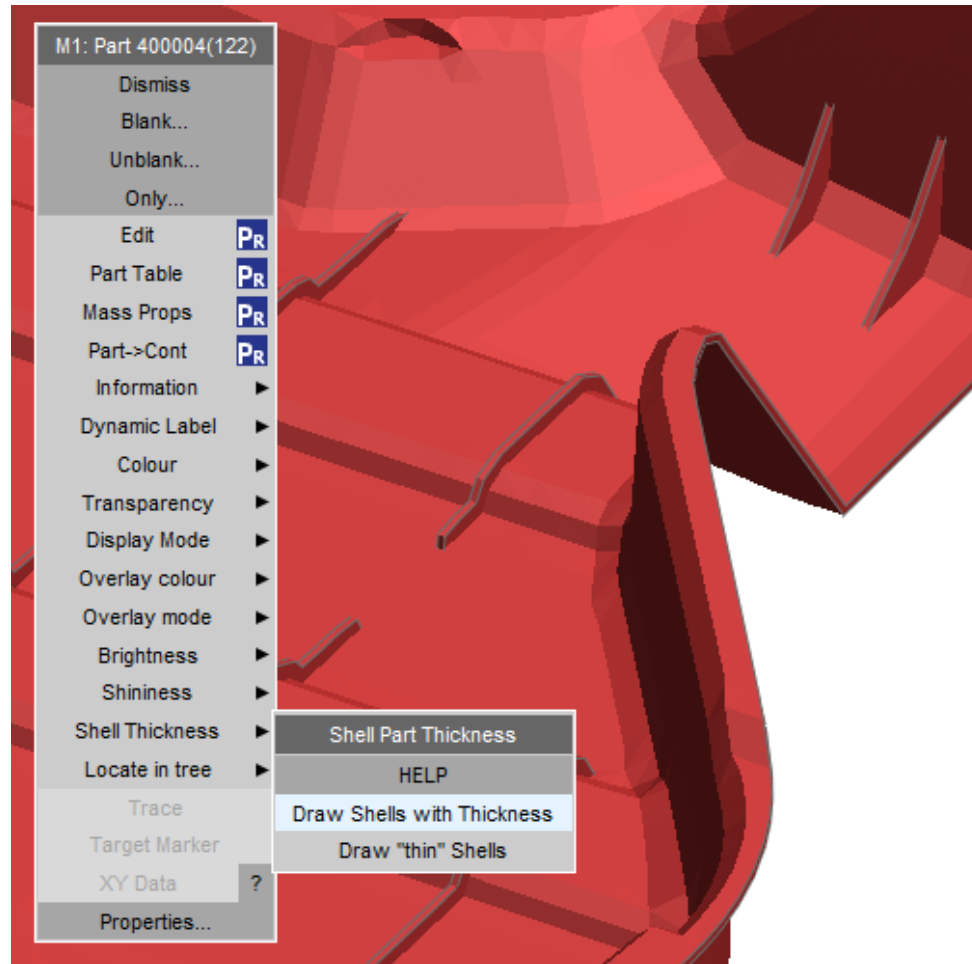
If thicknesses have been written to the *.ptf/d3plot* file then those values are used and can vary from state to state. If not then thickness are read from the *.ztf* file for each part.



Shell Thickness

Unlike in PRIMER, shell thickness is set on a per-part basis because drawing shells with thickness slows things down.

The thickness option can be set for individual parts using either Quick Pick or the Part Tree.



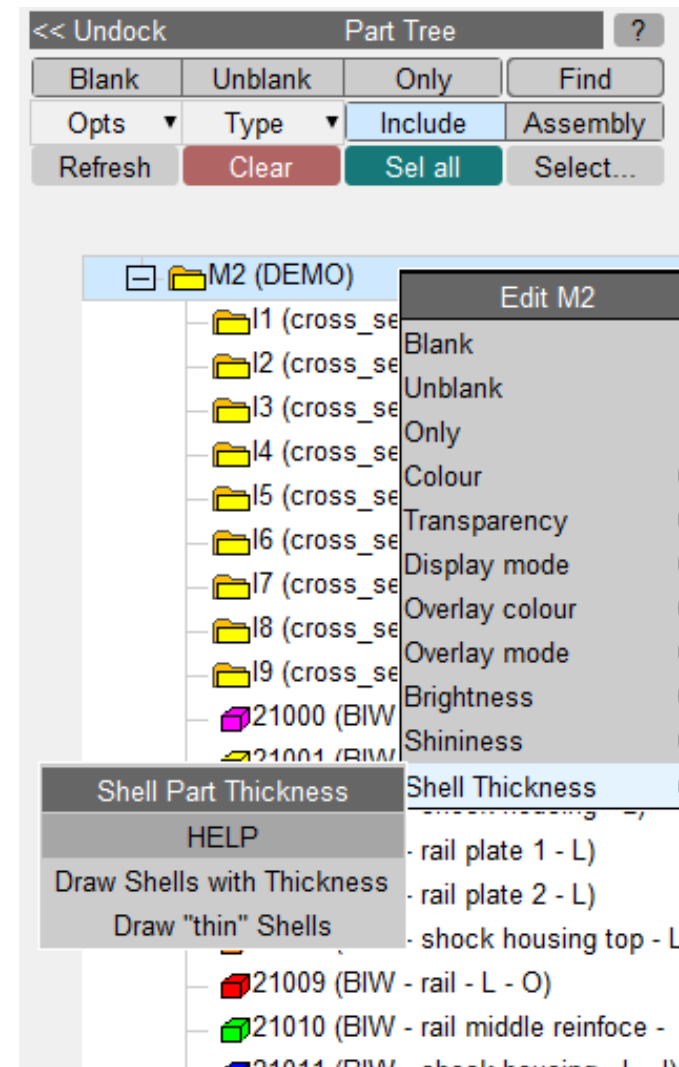
Shell Thickness

To set the thickness option for all parts, either:

- Right click on the model in the Part Tree.

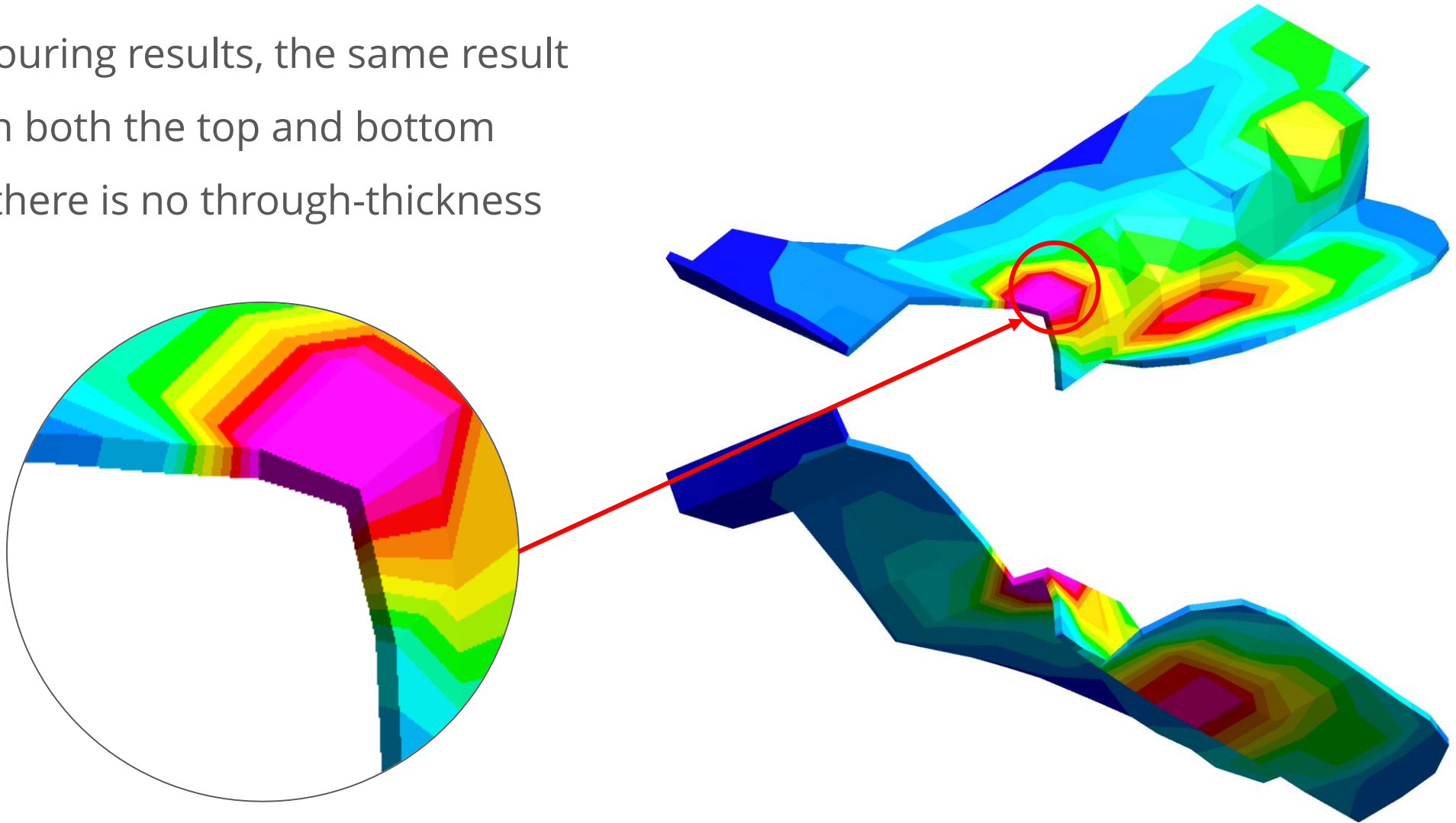
or

- Use Quick Pick and select all the parts in the graphics window by dragging out a selection box.



Shell Thickness

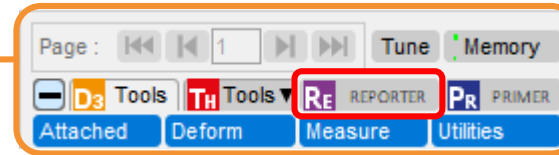
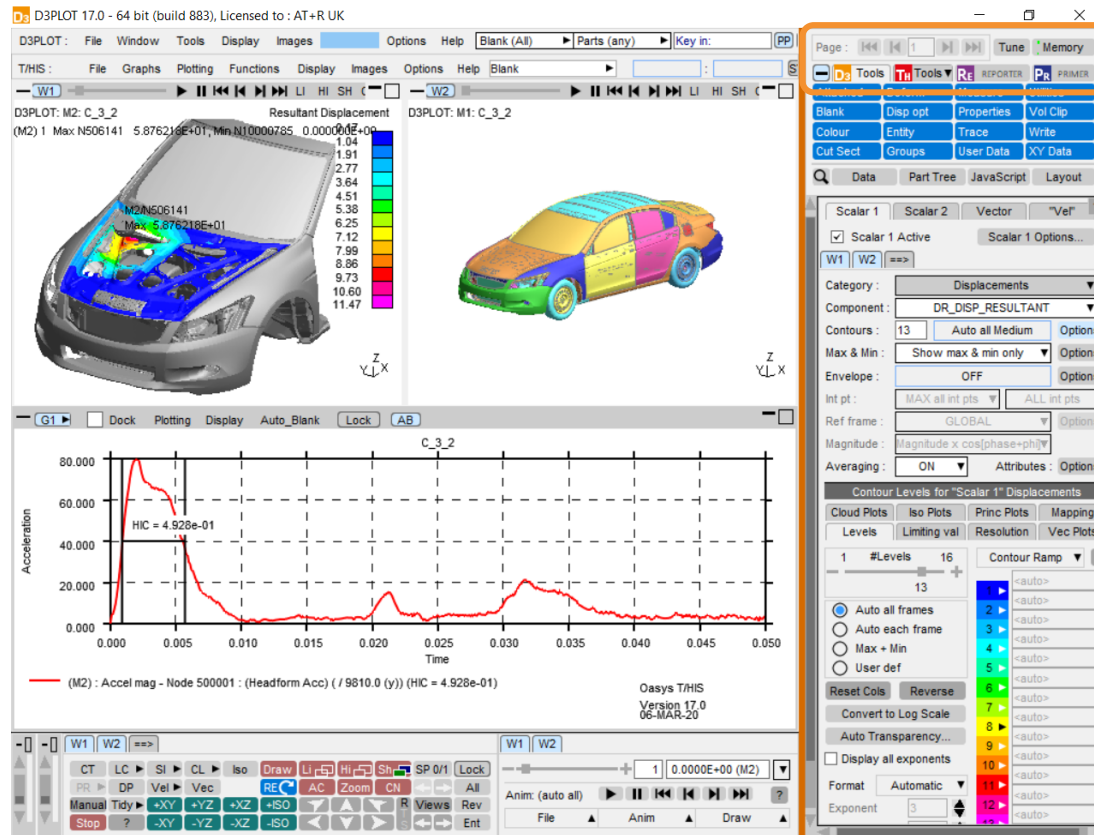
When contouring results, the same result is shown on both the top and bottom surfaces – there is no through-thickness variation.



Add REPORTER to your Workflow

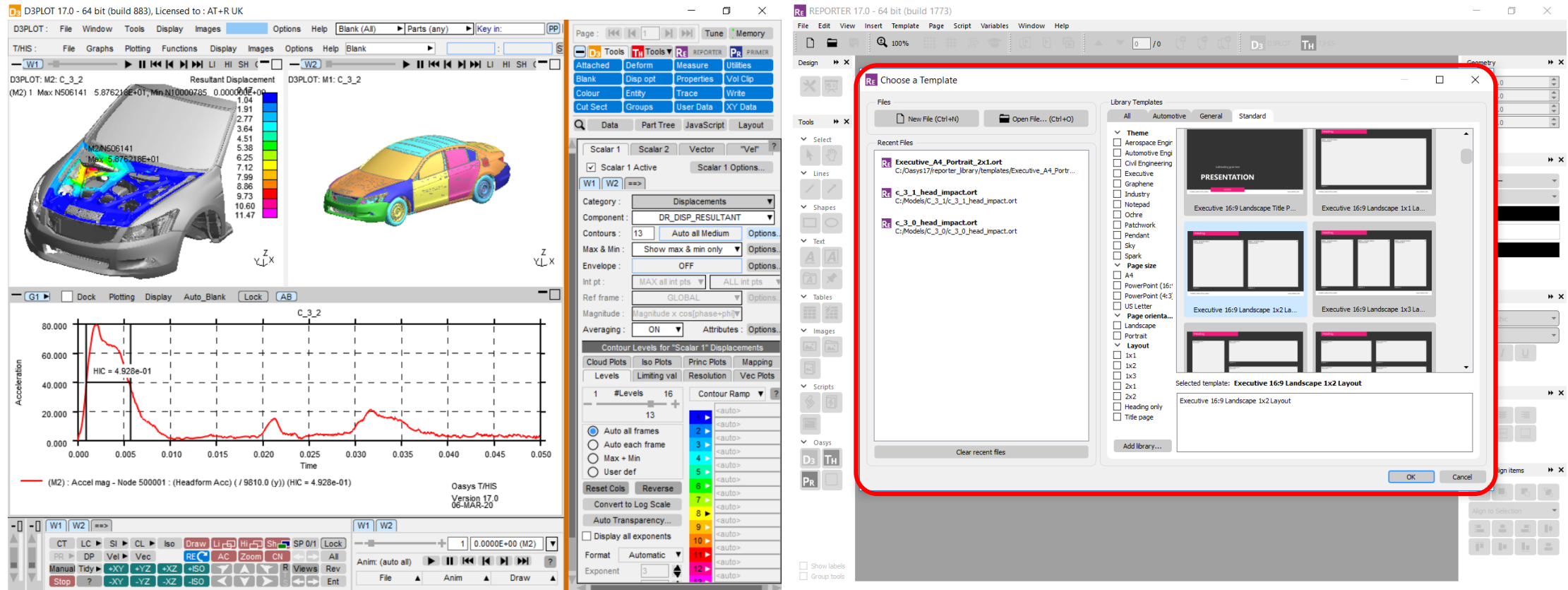
In version 17.0, REPORTER has been fully integrated with D3PLOT and T/HIS. Now, you can seamlessly add plots and graphs to reports as part of your normal post-processing workflow. Automated LS-DYNA reporting is just a few clicks away.

REPORTER can now be launched from D3PLOT and T/HIS



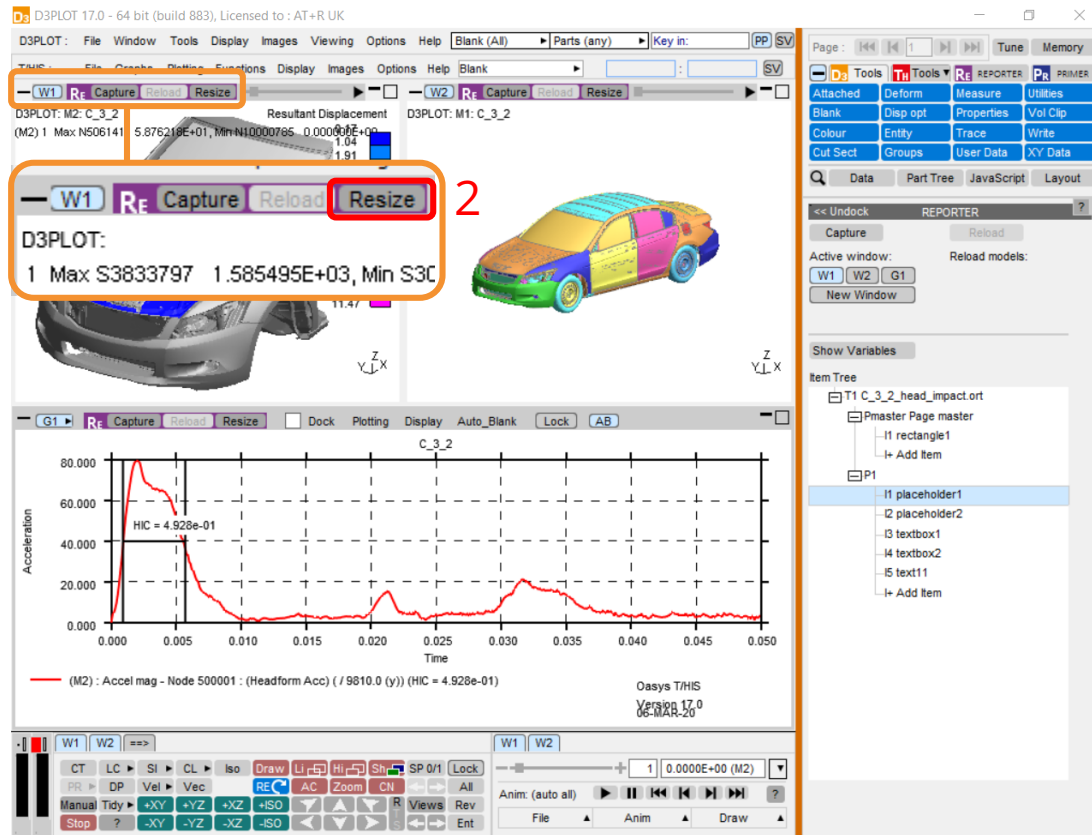
- You have been post-processing an analysis and want to put an image into a report.
- Press the “REPORTER” button above the TOOLS menu to launch REPORTER in a linked, shared memory mode.

Choose a Template to start creating your Report

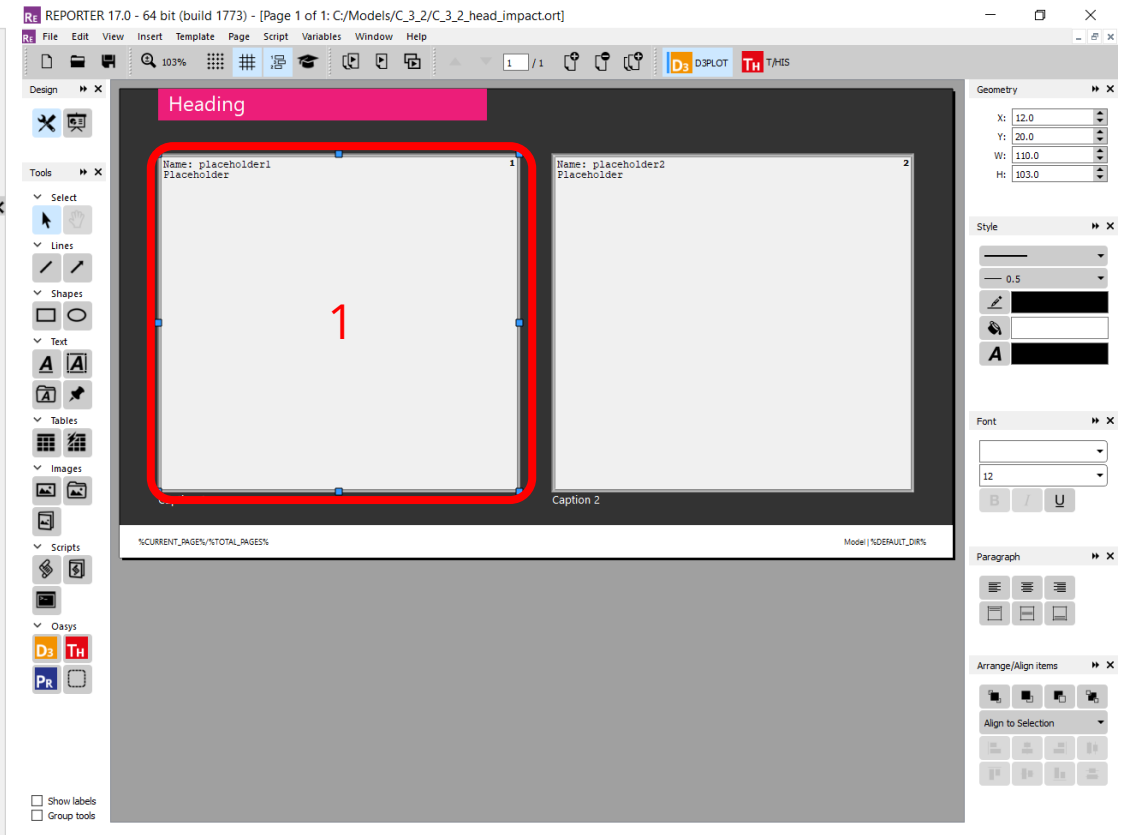


- Open a new blank template, an existing template or choose from a range of standard layouts to start creating a report.

Ensure Image looks right at Target Aspect Ratio

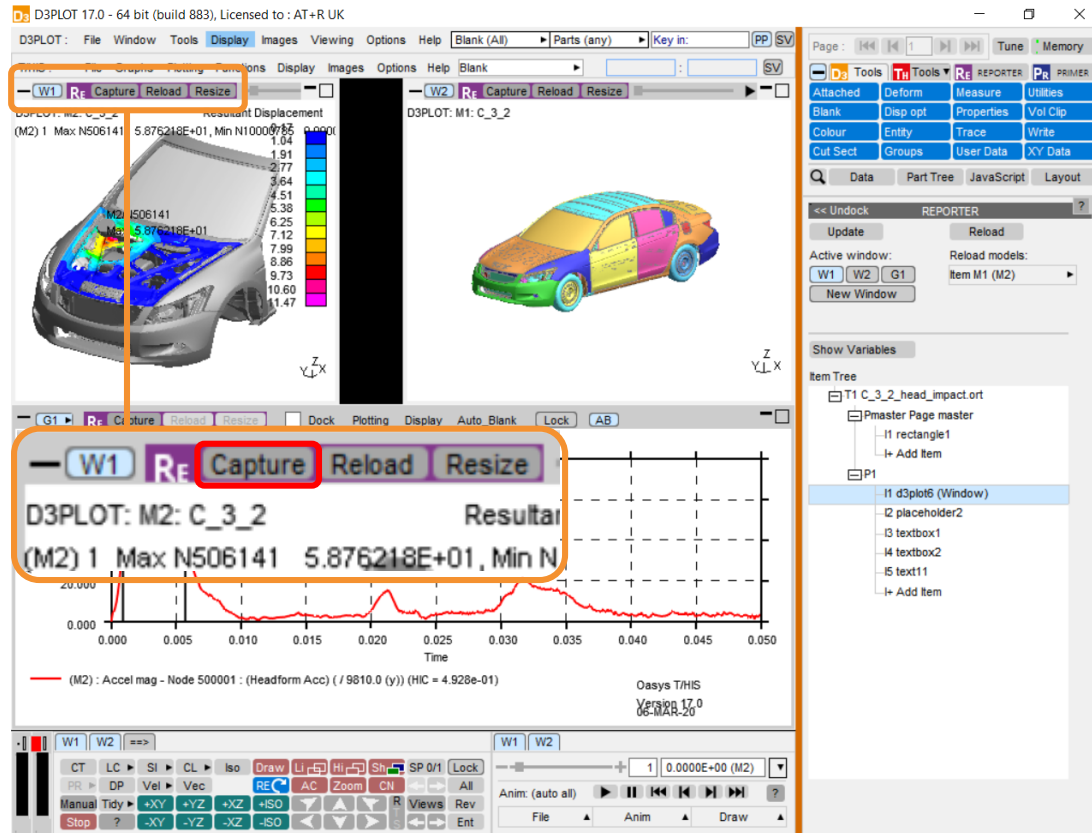


2. Press "Resize" button at the top of the D3PLOT window to resize to the REPORTER item aspect ratio.

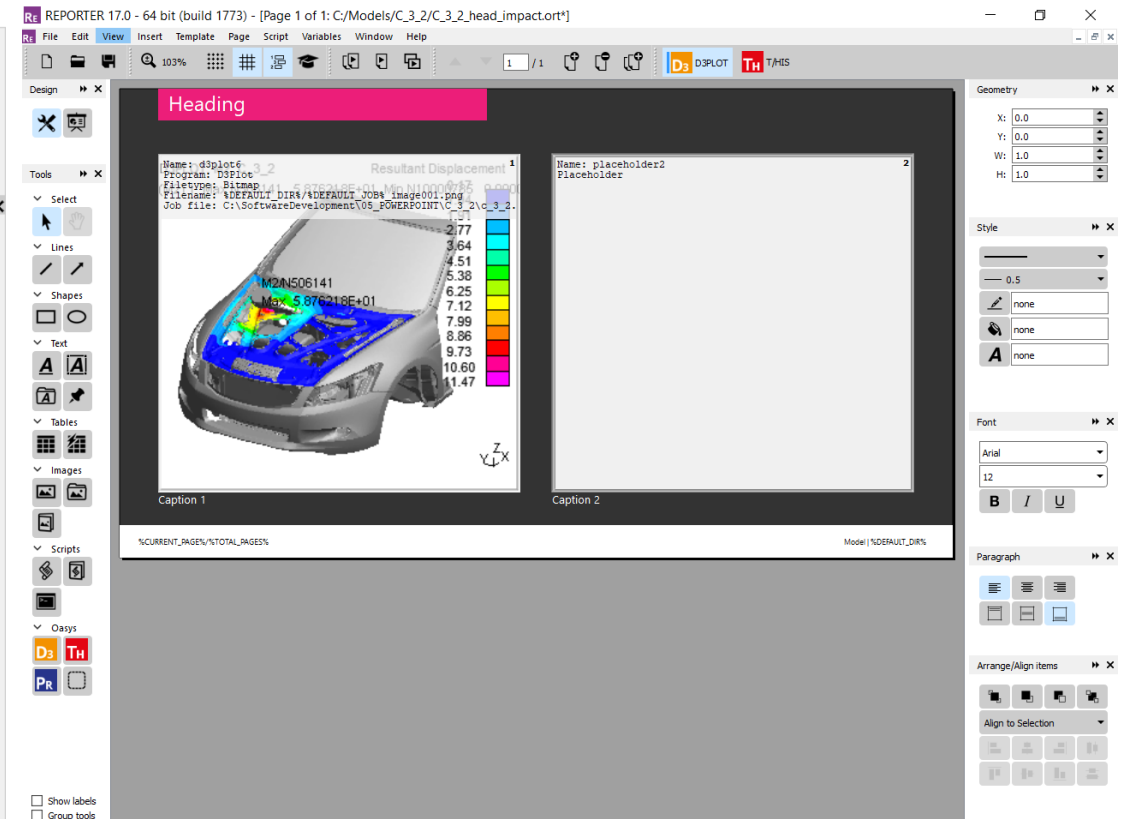


1. Select item in REPORTER that you want to capture into.

Capture an Image

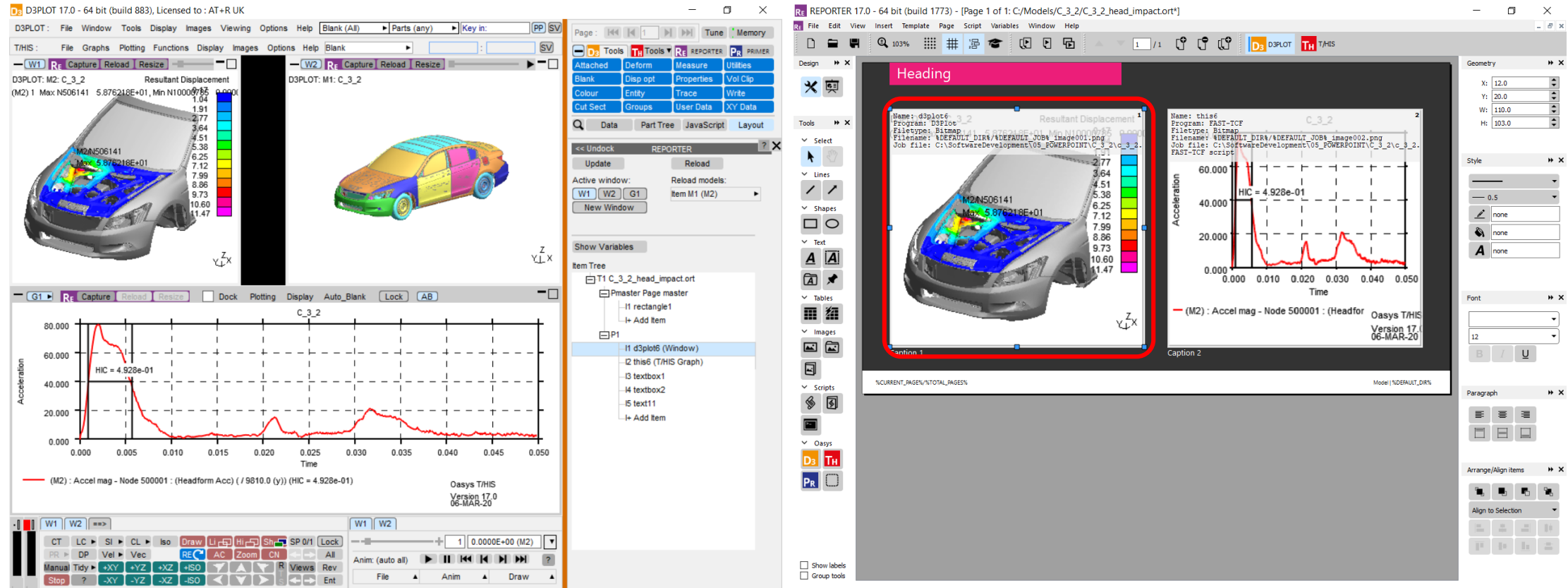


- Press the **Capture** button at the top of the D3PLOT window to send the capture to REPORTER.



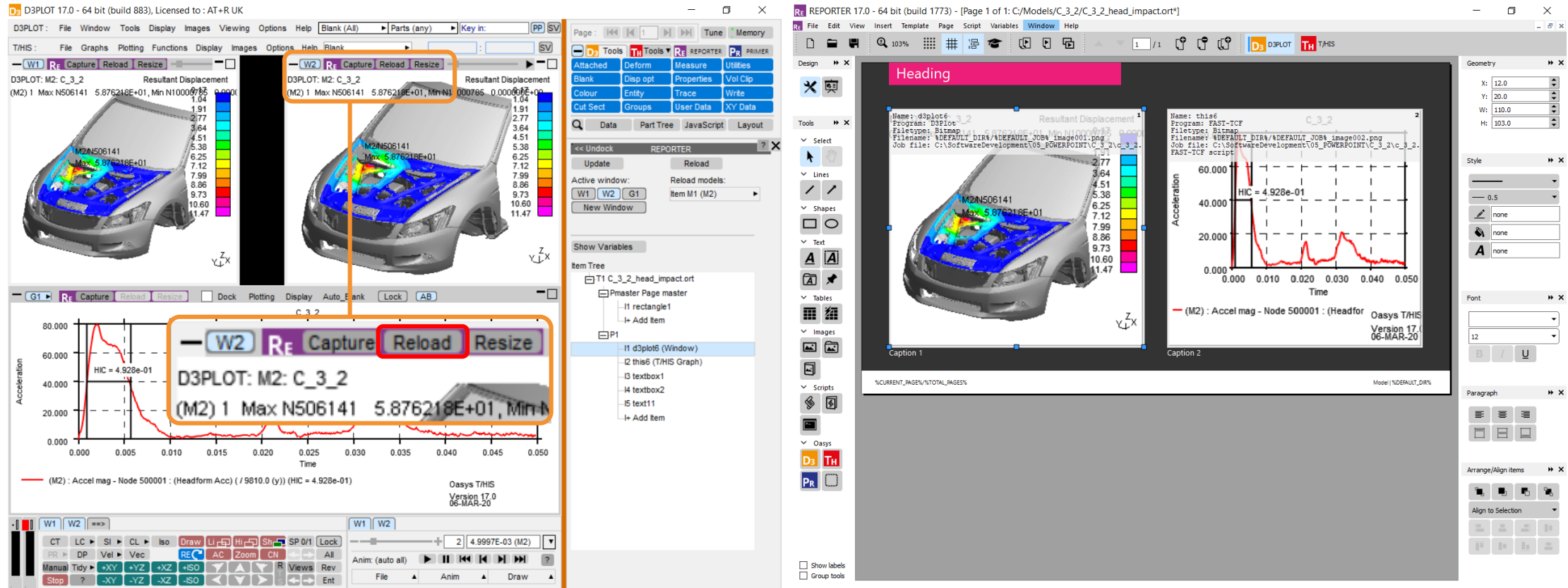
- Capture information populates item in REPORTER.

Reload items from existing REPORTER Templates



- Select item in REPORTER that you want to reload.

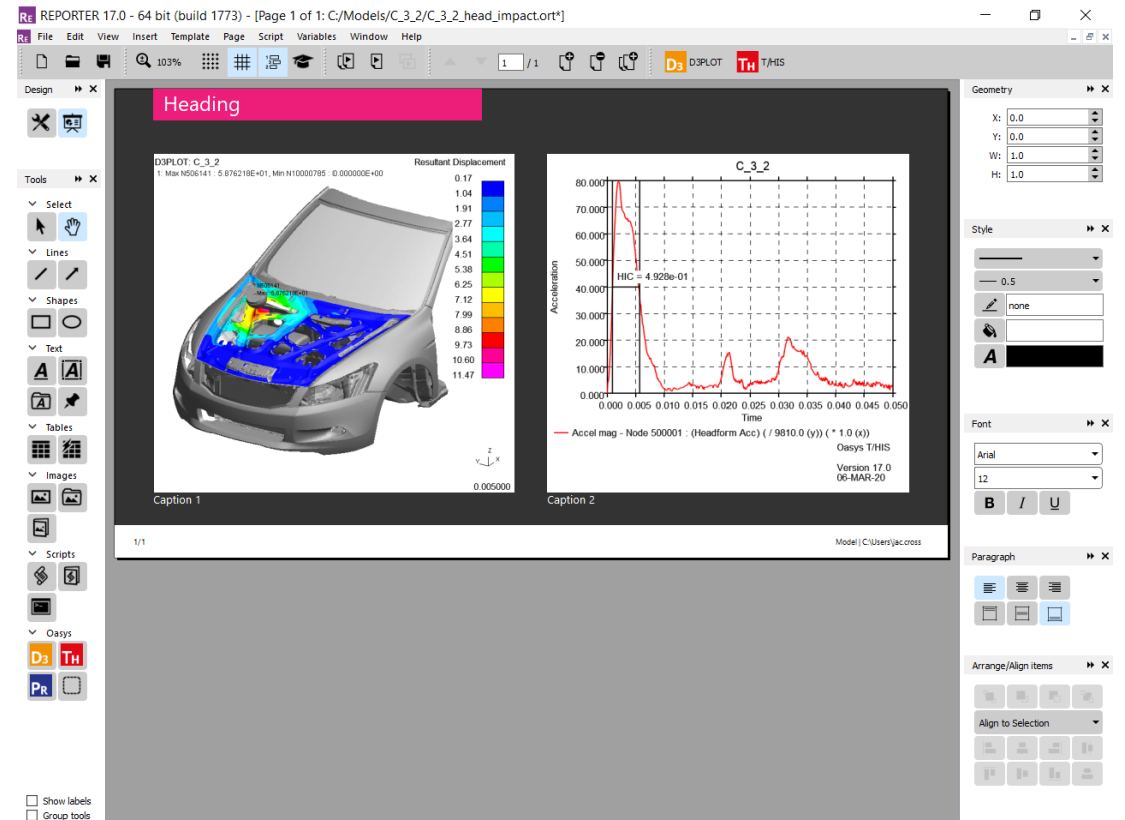
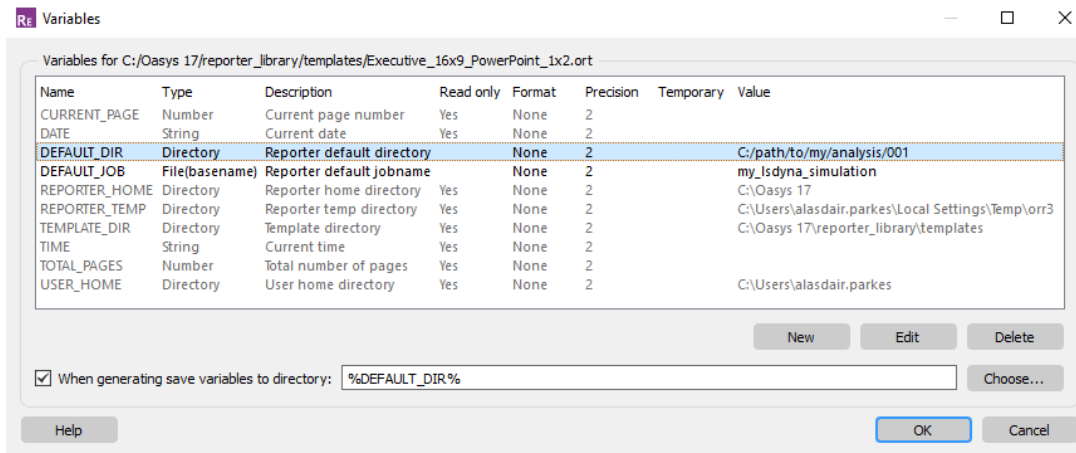
Items can be reloaded into a live POST session



- Selected REPORTER item is reloaded into the selected window in POST.

Generate Templates from REPORTER

- Generate your template by selecting:
File → Generate
- Generation takes place in existing linked POST sessions, reducing the time taken to generate reports.
- Use REPORTER variables to generate your template after each LS-DYNA simulation.



Useful to know...

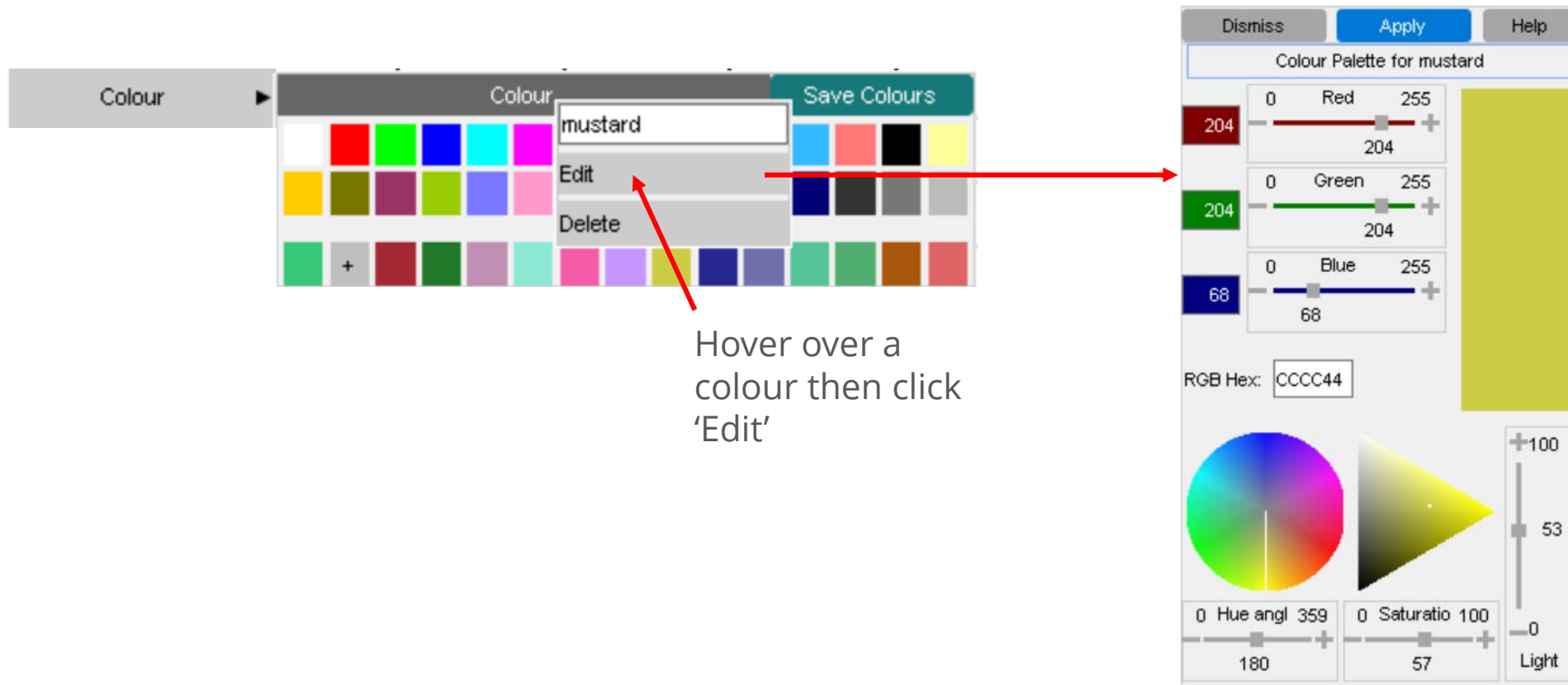
- PRIMER item capture and generation are unchanged from earlier versions.
- REPORTER has a new Placeholder item type that can be filled with any of D3PLOT, T/HIS or PRIMER items.
- REPORTER can link to both D3PLOT and T/HIS in the same session.
- Captures are now single window / graph – to capture a multi-window layout, capture each window into individual REPORTER items and then design your page layout with greater flexibility in REPORTER.
- Variables are now easier to work with and accessible through the REPORTER panel in D3PLOT and T/HIS.
- Existing templates are fully supported and can be edited and generated using the 16.0 method with a simple option in REPORTER.

User-defined Colours

User-defined Colours

User-defined colours are available in D3PLOT via a new colour popup.

The user-defined colours are created from the colour popup using a colour palette:



User-defined Colours

- The new colour popup has the existing 16 core colours from D3PLOT and an additional 14 core colours that were previously only in T/HIS.
- The additional 14 core colours and any user-defined colours are also available in the dialogue input.
- Up to 150 user-defined colours can be created.
- Once saved, the same user-defined colours are available in T/HIS.
- The colours are saved to an XML file, *user_colours.xml*. A *user_colours.xml* file can exist both in the install directory and a user's home area (if any colours are defined in both places, the user's colours take priority), or a location specified by a preference.
- The *user_colours.xml* files will be read by both D3PLOT and T/HIS.

Speed Improvements

Speed Improvements

The speed of D3PLOT has been improved for a number of commonly used features.

Contour plotting and animations

- In large models containing mostly shells a speed increase of up to 200% for SI/CT contour plotting and for the playback of animations, especially when elements are blanked or cut sections are on.
- In large models containing mostly solids speed increases of up to 400% for the same operations.
- 300%+ speed increase in the rescanning of contour bar min/max values when elements are blanked or unblanked.

(The overall time to plot results and to build animations may still be dominated by reading data from disk, especially over a network drive.)

Speed Improvements

Predictive picking

- Predictive picking in Quick Pick is now 300 > 400% faster.
- After sketching an item unnecessary recalculations of visible items have also been reduced to give a smoother experience.

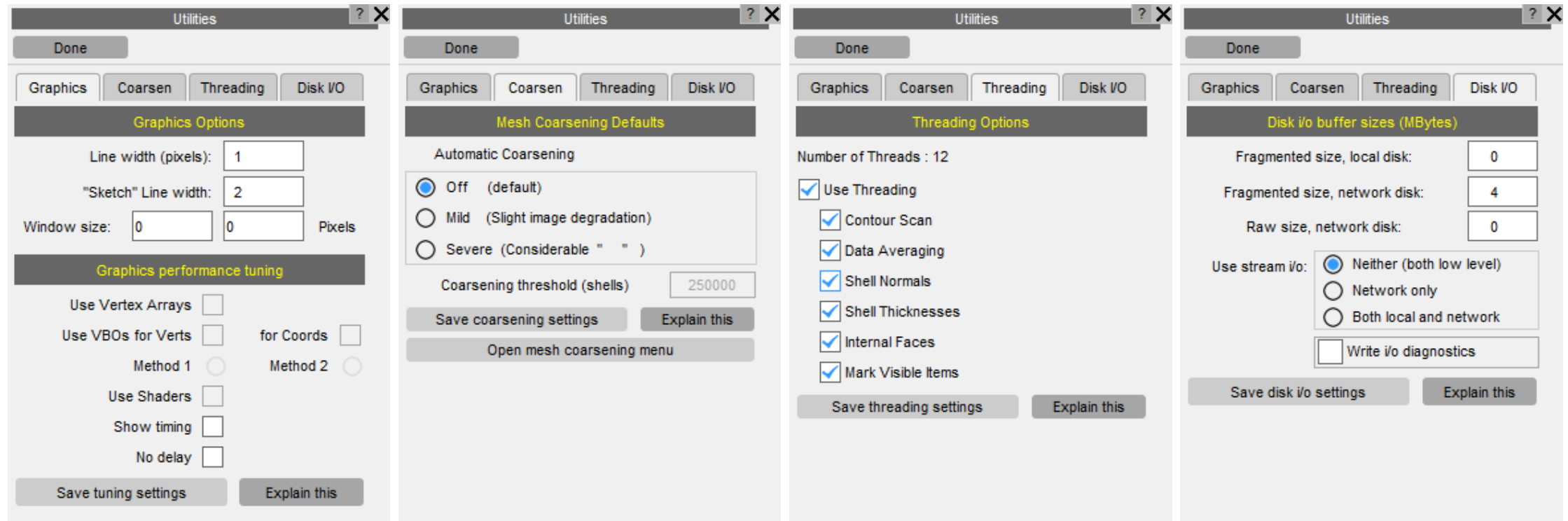
Multiple models

- No slow down with multiple models in the same window compared to the same models in different windows. Previously, D3PLOT was slower if you added multiple models to the same window.

Tune Panel and Threading

Tune Panel

The Tune panel has been redesigned to have tabs:

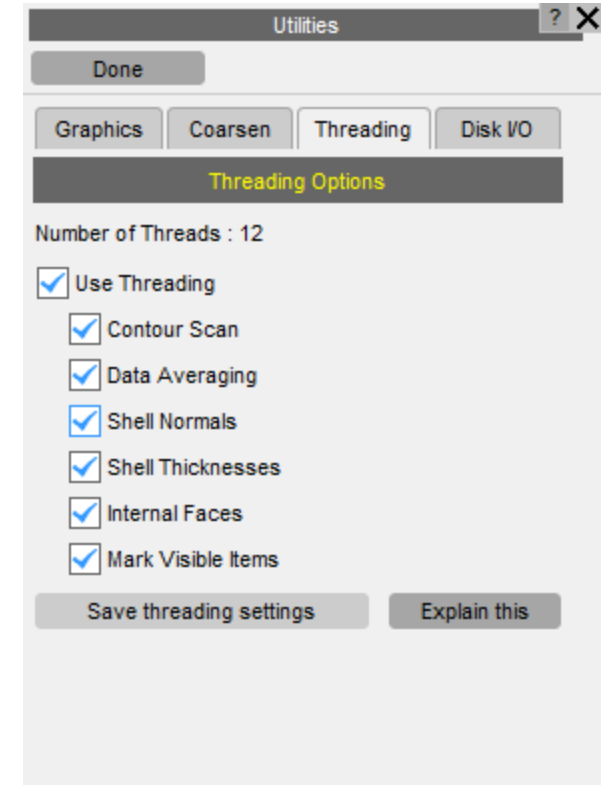


The Threading options
are new in D3PLOT 17.0.

Threading

The new threading options speed up the following operations:

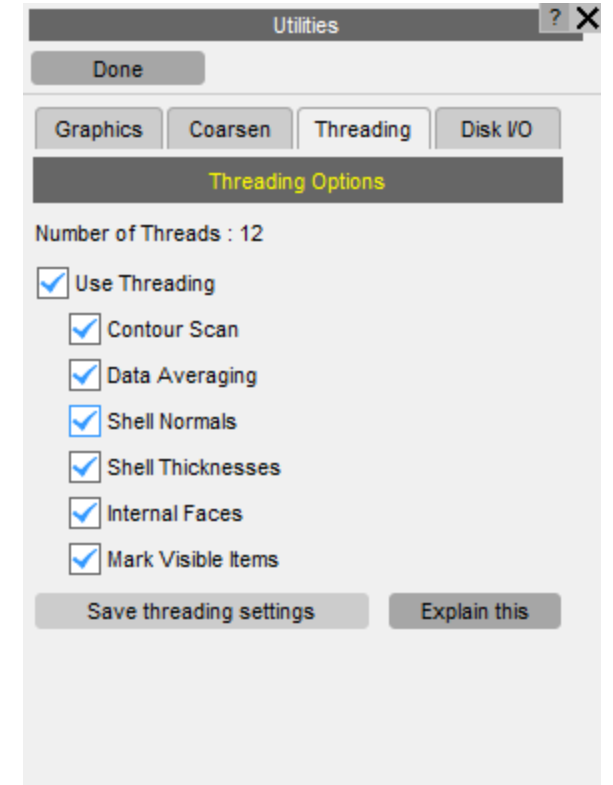
- Contour Scan (finding minimal/maximal values of data components for the contour bar).
- Data Averaging at nodes (used as part of contour calculation when averaging is turned on).
- Shell normal calculations (for graphics update when hardware shaders are turned off).



Threading

The new threading options speed up the following operations:

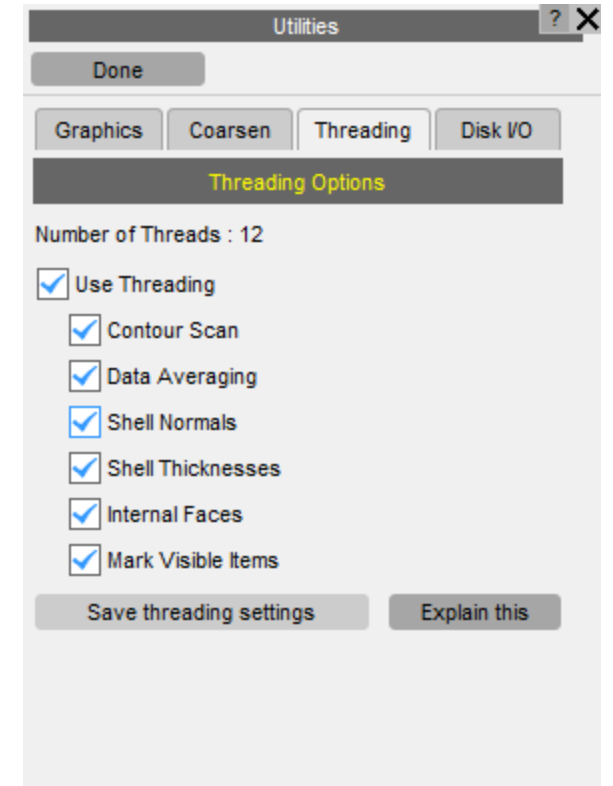
- Shell thickness calculations (used to calculate surfaces when the new “thickness” option for shells is turned on).
- Internal face calculations (used when blanking items or turning on/off display of entity types).
- Marking visible items (used when blanking items or turning on/off display of entity types).



Threading

The results should always be the same regardless of threading, but if you do get unexpected behaviour, threading can be turned off either for individual operations or altogether. These options can also be saved as preferences.

The maximum number of threads can be controlled with the environment variable **D3PLOT_NUM_THREADS_17**.



D3PLOT → PRIMER Link

D3PLOT → PRIMER Link

D3PLOT can automatically locate an associated LS-DYNA keyword file to load in a linked PRIMER session. The logic used to locate the keyword deck has been improved in Oasys Suite 17.0:

1. The model name is now written to the .ztf file and is read by D3PLOT. If this model exists, it is auto-loaded in PRIMER. If the .ztf file is missing, the approach depends on filename convention:

If you use the LSTC results filename convention (d3plot)

2. D3PLOT looks to see if there is a single LS-DYNA keyword file (.key/.k/.kby or a .gz/.zip variant thereof) in the working directory and auto-loads it in PRIMER.

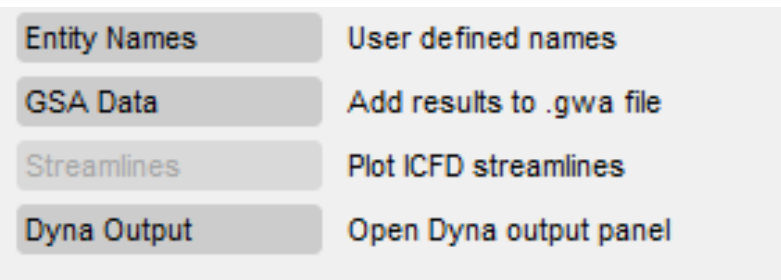
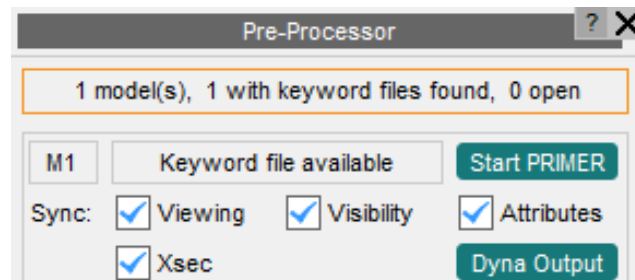
If you use the Oasys results filename convention (.ptf)

2. D3PLOT looks to see if a <job>.ptf has been loaded. If yes, D3PLOT looks for a matching <job>.key/.k/.kby or a .gz/.zip variant thereof.
3. Failing that, D3PLOT looks to see if there is a single LS-DYNA keyword file in the working directory.

The final fall-back, as always, is for you to manually select an input deck to load in PRIMER.

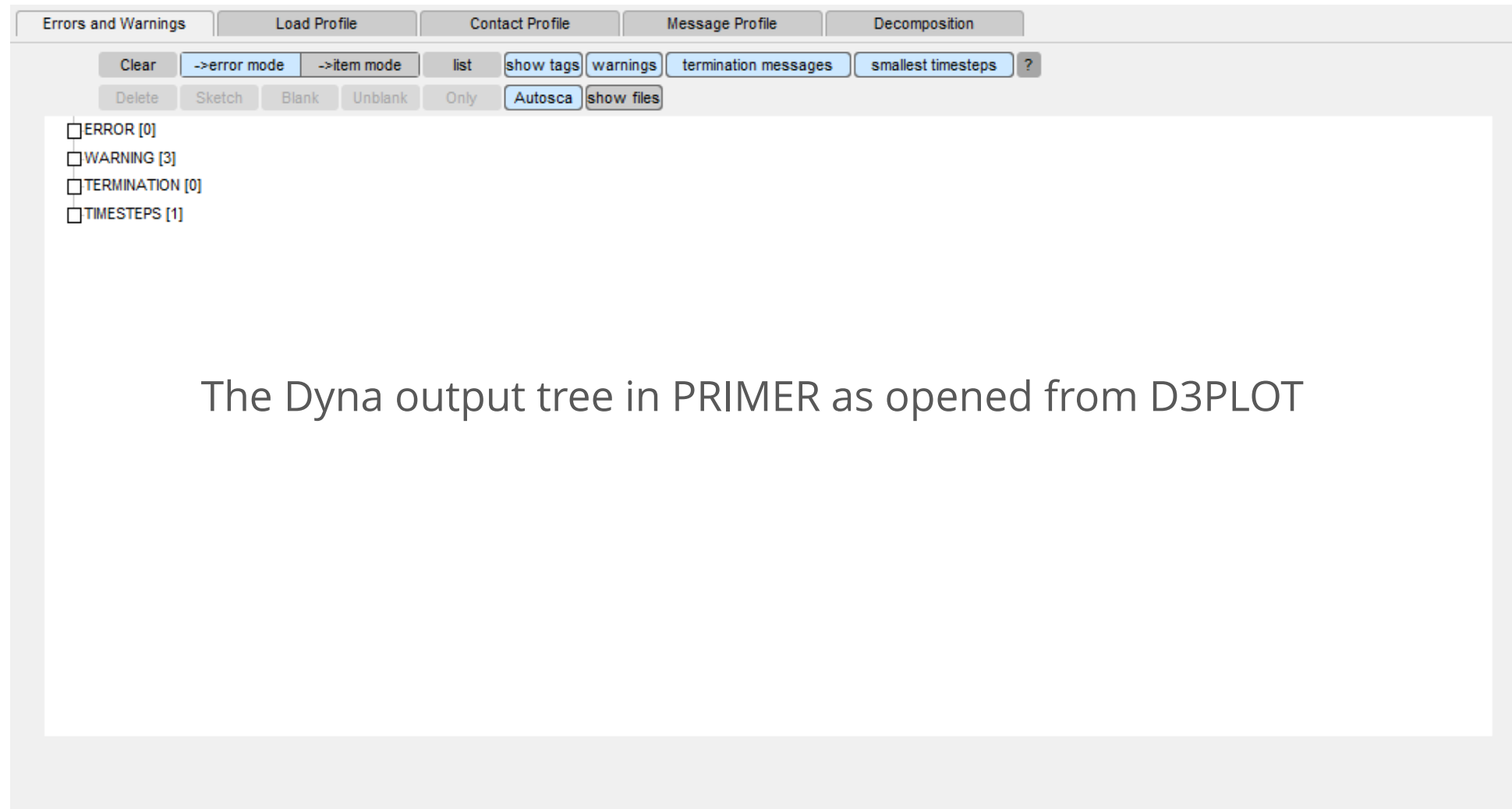
D3PLOT → PRIMER Link

- In addition to opening the associated key file in PRIMER, the DYNA output tree panel in PRIMER can also be opened from D3PLOT.
- This feature is added in the PRIMER integration panel and the Utilities panel:



- Clicking these buttons will open integrated PRIMER if necessary and map the DYNA output tree panel, which will list all of the errors and warnings from *.otf/d3hsp* and message files.

D3PLOT → PRIMER Link



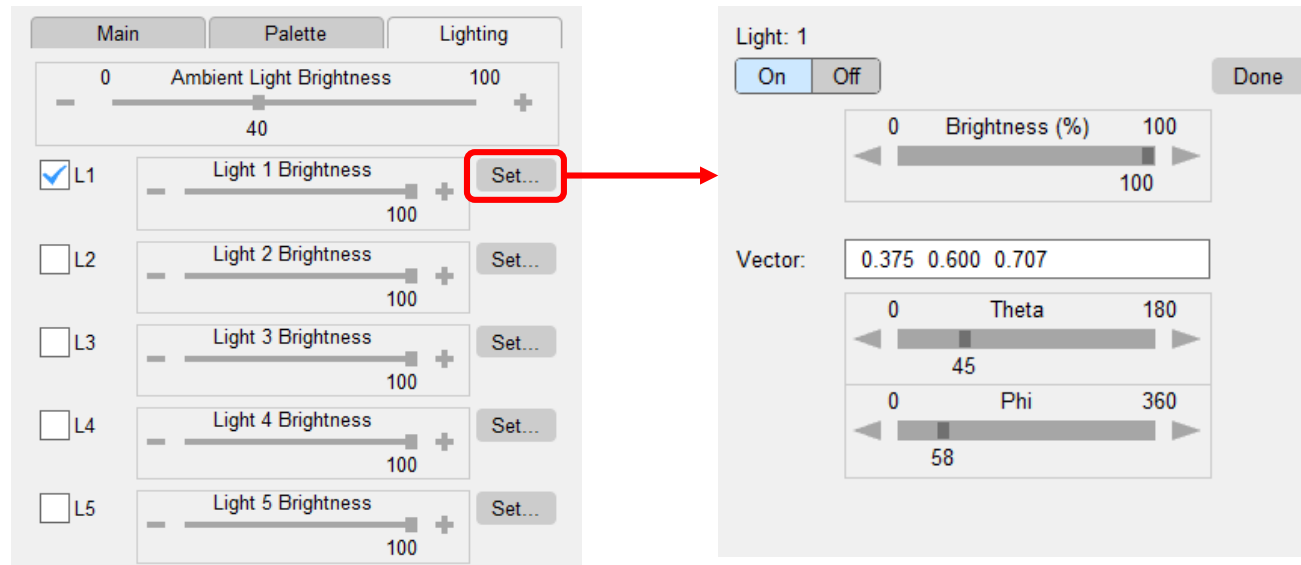
The Dyna output tree in PRIMER as opened from D3PLOT

Material Attributes

Lighting

Lighting Panel

- We have now added a Lighting panel to the Material Attributes panel for quicker customisation of the appearance of your models.
- Two new sliders allow you to experiment with how the lights look, rather than having to calculate the lighting vectors yourself.



Reloading Program Settings

Reloading Program Settings

It is often useful to save certain program settings and reload them at a later time to restore the program to a previous state, such as:

- The models loaded in D3PLOT.
- The number of graphics windows.
- The camera position.
- The blanking status of entities.
- The current display options.

D3PLOT has a few different methods for doing this, each one saving and reloading a different group of settings:

- Session Files.
- Settings Files.
- Properties Files.
- Template Files.
- View Files.

Each method is summarised in the following slides so that they can be compared to make it easier for you to decide which one is most appropriate for your situation.

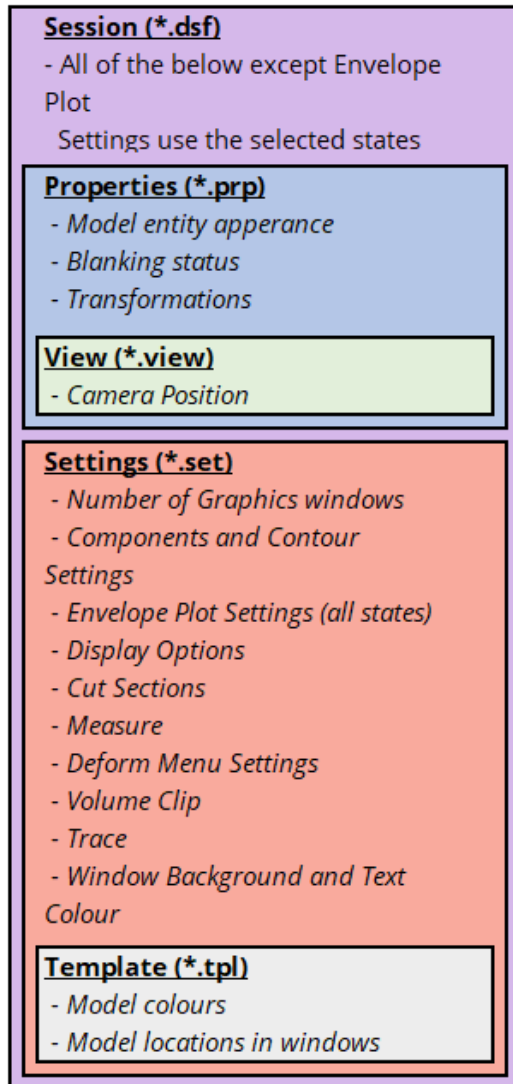
Reloading Program Settings

	Session	Settings	Properties	Template	View
Number of Graphic Windows and Location of Models	x	x		x	
Model Colours	x	x		x	
Camera Position	x	x	x		x
Model Appearance (Colour, transparency, display mode, etc.)	x		x		
Blanking Status	x		x		
Components and Contour Settings	x	x			
Envelope Plot Settings	x (selected states)	x (all states)			
Display Options	x	x			
Cut Sections	x	x			
Measure	x	x			
Deform Menu Settings (except Transform)	x	x			
Transform	x		x		
Volume Clip	x	x			
Traces	x	x			
Window Background and Text Colour	x	x			

The table shows which settings are saved by each method/file type.

- The **Session file** is the **most comprehensive** method as it saves everything contained in both the Settings and Properties files. In addition it saves the selected states of an envelope plot, whereas a Settings file will select all the states when it is reloaded irrespective of what was originally selected.
- **Settings files** save and reload **program and window status** information
- **Properties files** save and reload **model-specific** information.
- **Template files** only save a small subset of settings, but are useful for controlling which windows models are located in and their colours.
- **View files** only save the camera position and are useful for replicating views when capturing images.

Reloading Program Settings



This diagram shows how each file type relates to each other:

- View files are a subset of Properties files.
- Template files are a subset of Settings files.
- Properties and Settings files are subsets of Session files.

New Preferences

New Preferences

The following preferences have been added to D3PLOT:

Preference	Description
d3plot*threading_active	Whether or not to use threading
d3plot*thread_contour_scan	Whether or not to use threading for contour scanning
d3plot*thread_data_averaging	Whether or not to use threading for data averaging at nodes
d3plot*thread_shell_normals	Whether or not to use threading for shell normal calculations
d3plot*thread_shell_thickness	Whether or not to use threading for shell thickness calculations
d3plot*thread_internal_faces	Whether or not to use threading for internal face calculations
d3plot*thread_mark_visible	Whether or not to use threading for marking visible items
d3plot*show_checkpoint_files	<ul style="list-style-type: none">• To show/hide checkpoint playback panel upon D3PLOT start up• The value of the preference can be TRUE/FALSE (default – FALSE)

Command Line Options

New Command Line Options

The following command line options have been added to D3PLOT:

- **-no_ztf** to avoid reading additional data from a .ztf file.
- **-no_prop** to avoid reading a property file.
- **-no_set** to avoid reading a settings file.
- **-no_group** to avoid reading an ascii groups file.
- **-no_ifs** to avoid reading interface force segments and data.
- **-no_xtf** to avoid reading springs, masses, joints etc.
- **-no_lsda** to avoid reading spotweld, SPC etc. data.
- **-show_checkpoint_files=<TRUE/FALSE>**
 - To show/hide checkpoint playback panel upon D3PLOT start up.
 - The value of the preference can be TRUE/FALSE (default – FALSE).
- **-checkpoint_dir=<directory>**
 - Directory path to write checkpoint files.
 - <directory> must be a valid directory name on your system.
 - If the value is <none> then the checkpoint files are not recorded for the D3PLOT session.

JavaScript

JavaScript

- JavaScript API enhancements:
 - A new slider widget is available for graphical user interfaces
 - The *onClick* event can now be used for label widgets

Extending Classes

There have been a few instances where users have extended an existing class in D3PLOT by adding their own property (for example adding a 'type' property to the Widget class). When we add new functionality to the API this potentially causes a conflict.

For Arup to be able to extend and enhance the API in the future any constant, property or method names beginning with a lowercase or uppercase letter are reserved.

If you need to add your own properties or methods to one of the existing classes then to avoid any potential future conflict you should ensure that the name begins with either an underscore (_) or a dollar sign (\$) or the name is prefixed with your own unique identifier.

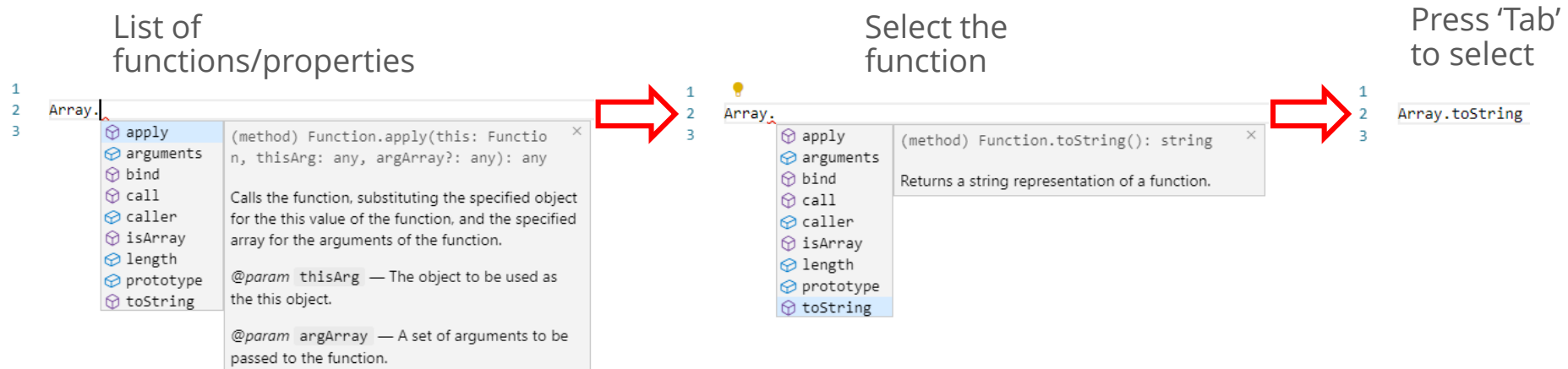
For example if company 'ABC' need to add a property called 'example' then to avoid any potential future conflict use one of:

- `_example`
- `$example`
- `ABC_example`

JavaScript – Intellisense

Writing scripts can be time consuming when trying to remember which functions are available and the number and order of arguments in the functions. To find out, you would normally have to go searching for it in the manual and doing this over and over again is frustrating and time consuming.

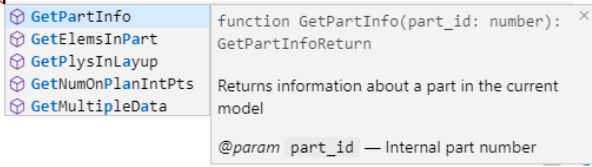
One way to speed up writing scripts is to use software that provides information as you write your script. One such program is called Visual Studio Code (produced by Microsoft and free to use) which uses something called 'Intellisense' to list objects, functions and parameters as you type, e.g.,



JavaScript – Intellisense

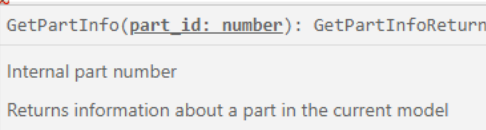
In Oasys Suite 17.0, a d3plot.d.ts file located in the 'Oasys 17/Intellisense' directory has been added to provide Intellisense for the D3PLOT JS API, e.g.,

```
2 var info = getpa
```



List D3PLOT JS-API functions.

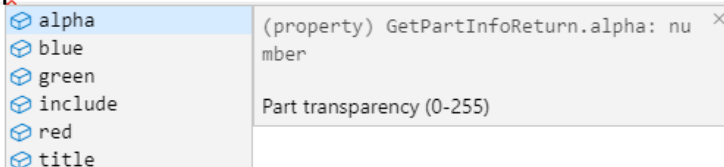
```
2 var info = GetPartInfo()
```



Describe the function, parameters and the return type.

For functions that return objects list the properties of the object.

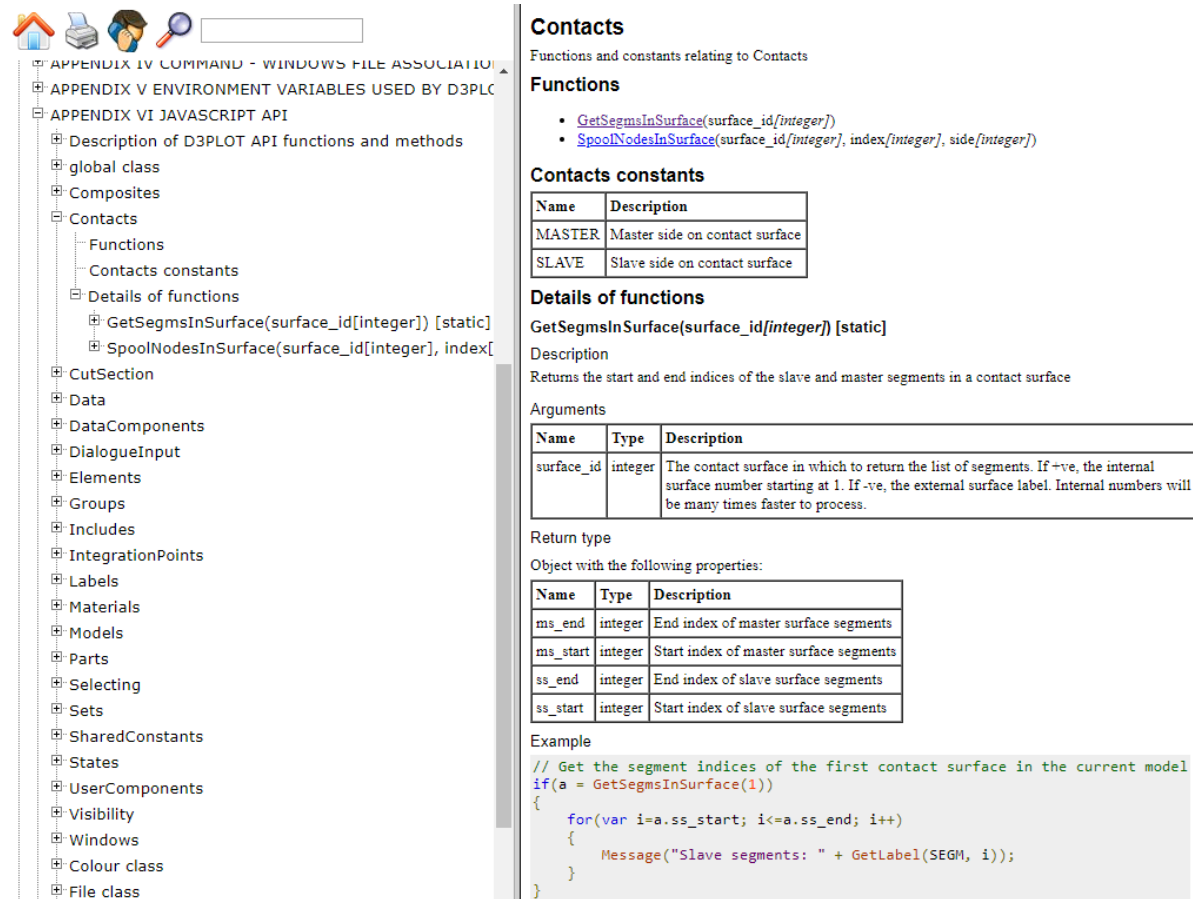
```
2 var info = GetPartInfo(1);
3
4 info.
```



A word document in the '**Oasys 17/Intellisense**' directory gives instructions on how to download Visual Studio Code and set things up to use the Intellisense file.

JavaScript – Manual

The layout of the JavaScript API section of the manual has been updated so it is now more like PRIMER, T/HIS and REPORTER.



Contacts
Functions and constants relating to Contacts

Functions

- [GetSegmsInSurface\(surface_id\[integer\]\)](#)
- [SpoolNodesInSurface\(surface_id\[integer\], index\[integer\], side\[integer\]\)](#)

Contacts constants

Name	Description
MASTER	Master side on contact surface
SLAVE	Slave side on contact surface

Details of functions

GetSegmsInSurface(surface_id[integer]) [static]

Description
Returns the start and end indices of the slave and master segments in a contact surface

Arguments

Name	Type	Description
surface_id	integer	The contact surface in which to return the list of segments. If +ve, the internal surface number starting at 1. If -ve, the external surface label. Internal numbers will be many times faster to process.

Return type
Object with the following properties:

Name	Type	Description
ms_end	integer	End index of master surface segments
ms_start	integer	Start index of master surface segments
ss_end	integer	End index of slave surface segments
ss_start	integer	Start index of slave surface segments

Example

```
// Get the segment indices of the first contact surface in the current model
if(a = GetSegmsInSurface(1))
{
    for(var i=a.ss_start; i<=a.ss_end; i++)
    {
        Message("Slave segments: " + GetLabel(SEGM, i));
    }
}
```

JavaScript – Manual

APPENDIX VI JAVASCRIPT API

+ Description of D3PLOT API functions and methods

+ global class

+ Composites

+ Contacts

+ CutSection

+ Data

+ DataComponents

+ DialogueInput

+ Elements

+ Groups

+ Includes

+ IntegrationPoints

+ Labels

+ Materials

+ Models

+ Parts

+ Selecting

+ Sets

+ SharedConstants

+ States

+ UserComponents

+ Visibility

+ Windows

+ Colour class

+ File class

+ Include class

+ Options class

+ PopupWindow class

+ Widget class

+ WidgetItem class

+ Window class

+ Examples

The majority of the D3PLOT functions are in the global scope, but to make them easier to locate, they are presented in different categories in the navigation tree.

These are followed by sections for classes shared with PRIMER.

Sections for D3PLOT global functions.

Classes shared with PRIMER.

JavaScript – Manual

All the D3PLOT global functions are also listed in alphabetical order in the global class section:

global class

The following pages list global class functions from PRIMER which can be used in D3PLOT.

Most of the D3PLOT functions are also in the global scope, but to make them easier to locate, they are presented in different categories (e.g. Contacts, CutSection, Data) in subsequent sections. They are also listed below in alphabetical order.

Class functions

- [ErrorMessage](#)(string[*Any valid javascript type*])
- [Execute](#)(data[*object*])
- [Exit](#)()
- [GetCurrentDirectory](#)()
- [GetStartInDirectory](#)()
- [Getenv](#)(name[*string*])
- [Message](#)(string[*Any valid javascript type*])
- [MilliSleep](#)(time[*integer*])
- [NumberToString](#)(number[*integer/real*], width[*integer*], pref_int (optional)[*boolean*])
- [Print](#)(string[*Any valid javascript type*])
- [Println](#)(string[*Any valid javascript type*])
- [SetCurrentDirectory](#)((directory path[*string*])
- [Sleep](#)(time[*integer*])
- [System](#)(string[*Any valid javascript type*])
- [WarningMessage](#)(string[*Any valid javascript type*])

D3PLOT functions

- [Blank](#)(type_code[*integer*], item[*integer*], window_id (optional)[*integer*])
- [CreateUbinComponent](#)(component_name[*string*], component_type[*integer*], data_type[*integer*], if_existing[*integer*], dispose (optional)[*integer*], location (optional)[*integer or string*])
- [CreateWindow](#)(model_list[*Array of integers integer*])
- [DeleteUbinComponent](#)(handle[*integer*])
- [DeleteWindow](#)(window_list[*Array of numbers number*], dispose_flag (optional)[*integer*])
- [DialogueInput](#)(string_1, (string_2 ... string_n)[*string*])
- [DialogueInputNoEcho](#)(string_1, (string_2 ... string_n)[*string*])
- [GetCutCoords](#)(type_code[*integer*], item[*integer*], state_id (optional)[*integer*])
- [GetCutForces](#)(window_id[*integer*], include_blanked (optional)[*integer*], part_id (optional)[*integer*], state_id (optional)[*integer*], model_id (optional)[*integer*])
- [GetCutSection](#)(window_id[*integer*], state_id (optional)[*integer*], model_id (optional)[*integer*])
- [GetData](#)(component[*integer*], type_code[*integer*], item[*integer*], int_pt (optional)[*object integer*], extra (optional)[*integer*], fr_of_ref (optional)[*integer*], state_id (optional)[*integer*], dda (optional)[*integer*], consider_blanking (optional)[*integer*], mag_or_cur (optional)[*integer*])
- [GetElemAxes](#)(type_code[*integer*], item[*integer*], state_id (optional)[*integer*])
- [GetElemBetaAngle](#)(type_code[*integer*], item[*integer*], ply_id[*integer*], int_pnt (optional)[*integer*], state_id (optional)[*integer*])
- [GetElemsAtNode](#)(node[*integer*], type_code[*integer*], state_id (optional)[*integer*])
- [GetElemsInPart](#)(part_id[*integer*], state_id (optional)[*integer*])
- [GetElemsInPly](#)(ply_id[*integer*], state_id (optional)[*integer*])
- [GetGroupInfo](#)(group_id[*integer*])
- [GetIncludeInfo](#)(include_id[*integer*])
- [GetItemsInSet](#)(set_type[*integer*], set_id[*integer*])
- [GetLabel](#)(type_code[*integer*], item[*integer*], state_id (optional)[*integer*])
- [GetMid](#)(type_code[*integer*], item[*integer*], layer_id (optional)[*integer*], state_id (optional)[*integer*])

Replay of Drag-Drop Files

Replay of Drag-Drop Files

- The checkpoint files will now support replay of the drag and drop of the files into D3PLOT graphics area.

Contact Information



www.arup.com/dyna

For more information please contact us:

UK

The Arup Campus
Blythe Valley Park
Solihull
B90 8AE
United Kingdom

T: +44 121 213 3399

dyna.support@arup.com

China

Arup China
39/F-41/F Huaihai Plaza
1045 Huaihai Road (M)
Xuhui District, Shanghai
200031
China

T: +86 21 3118 8875

china.support@arup.com

India

Arup India Pvt Ltd
Ananth Info Park, HiTec City
Madhapur Phase-II
Hyderabad
500081, Telangana
India

T: +91 40 44369797 / 98

india.support@arup.com

USA West

Oasys Ltd
c/o 560 Mission Street Suite 700
San Francisco
CA 94105
United States

T: +1 415 940 0959

us.support@arup.com

or your local Oasys distributor