REPORTER

REPORTER Manual

from Oasys Ltd

Version 19.0

For help and support from Oasys Ltd please contact:

UK

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

China

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

India

Arup India Pvt Ltd
10th floor, Western Dallas Center
Plot no. 83/1, Knowledge City
Rai Durg
Hyderabad 500032
Telangana, India
Tel: +91 40 69019797 / 98
Email: india.support@arup.com

USA West

Oasys Ltd
c/o 560 Mission Street Suite 700
San Francisco
United States
CA 94105
Tel: +1 415 940 0959
Email: us.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 Ansys, Inc.

Preamble **0.1**
Introduction 0.1
Development Status 0.1
Systems supported 0.1
Revision History 0.1
Text conventions used in this manual 0.20
Themes for the Graphical User Interface **0.21**
Setting the theme 0.21
1. Setting up and running REPORTER **1.1**
1.1 Setting up REPORTER 1.1
1.2 Running REPORTER 1.2
1.3. A one-minute introduction to REPORTER 1.2
2. Menu Layout **2.1**
2.1 Basic menu layout 2.1
2.2 Mouse and keyboard usage for the screen-menu interface 2.5
2.3 Using the "file filter" boxes. 2.5
2.4 Log file 2.7
2.5 View Controls 2.10
2.6 Running a script file 2.12
2.7 Preferences 2.12
3. Opening and closing templates and reports **3.1**
3.1 Creating a new template 3.1
3.2 Reading an existing template or report 3.1
3.3 Reading a library template 3.1
3.4 Editing template properties 3.3
3.5 Saving a template 3.4
3.6 Saving a report 3.4
4. Inserting and editing pages **4.1**
4.1 Adding a new page 4.1
4.2 Adding a new page from the library 4.1
4.3 Deleting pages 4.2
4.4 Duplicating pages 4.2
4.5 Reordering pages 4.3
4.6 Changing the current page 4.3
4.7 Changing the page properties 4.3
4.8 Inserting pages from file 4.3
4.9 Importing and exporting pages 4.3
4.10 Page masters 4.4
4.11 Page Setup 4.4
4.12 Generating a single page 4.4
5. Inserting and editing simple objects **5.1**
5.1 Using the Grid and Snap options 5.1
5.2 Setting line style, thickness, colour, and fill colour 5.1
5.3 Inserting and editing shapes, images, and text 5.2
5.4 Editing shapes, image, and text objects 5.6
5.5 Copying objects and using the clipboard 5.8
5.6 Reordering items on the page 5.9
5.7 Search and replace 5.11
5.8 Locking items 5.12
6. Advanced objects **6.1**
6.1 D3PLOT objects 6.1
6.2 T/HIS objects 6.8
6.3 PRIMER objects 6.11
6.4 Program objects 6.15
6.5 File objects 6.17
6.6 Library objects 6.19
6.7 Table objects 6.22
6.8 Autotable objects 6.26
6.9 Script objects 6.30
6.10 Script File objects 6.31
6.11 Note objects 6.33
6.12 Placeholder objects 6.33
7. REPORTER Integration **7.1**
7.1 Linking the Programs 7.1
7.2 Item Tree 7.1
7.3 Capture 7.1
7.4 Reload 7.3
7.5 Generate 7.7
7.6 Variables 7.7
7.7 Exceptions to the Version 17 Method and Existing Templates from Version 16 and Earlier 7.8
8. Generating and outputting reports **8.1**

8.1 Effect of object order on generating a report. 8.1
8.2 Generating reports 8.2
8.3 Outputting a generated report 8.4
8.4 Combining output from multiple reports 8.8
8.5 Animation support for output file formats 8.8
9. Working with Variables 9.1
9.1 User defined variables 9.1
9.2 Predefined variables 9.2
9.3 Formatting TIME and DATE variables 9.3
9.4 Creating and editing variables 9.4
9.5 Creating a variable using D3PLOT 9.6
9.6 Creating a variable using T/HIS 9.7
9.7 Creating a variable using an external program/script 9.8
9.8 Creating a variable using a FAST-TCF script 9.9
9.9 Creating a variable from the command line 9.10
9.10 Creating a variable from javascript 9.10
9.11 Deleting variables 9.10
9.12 Inserting a variable 9.11
9.13 Using variables in D3PLOT and T/HIS command files and FAST-TCF scripts. 9.13
9.14 Saving all the variables to a file after generating a report 9.15
9.15 Variable expressions 9.15
10. Hyperlinks 10.1
10.1 Adding basic hyperlinks 10.1
10.2 Adding hyperlinks in D3PLOT external data (blob) plots 10.1
11. Conditional formatting 11.1
11.1. Adding a condition 11.1
11.2. Condition types 11.2
12. Fonts 12.1
12.1 Supported Fonts 12.1
12.2 Legacy Fonts 12.1
12.3 Font Mapping 12.2
12.4 Fonts in report output 12.3
13. Scripting 13.1
13.1 Example scripts 13.1
A. Command line arguments and oa_pref options A.1
A.1 Command line arguments A.1
A.2 oa_pref options A.3
B. Library objects B.1
B.1. Standard library programs B.1
B.2. Standard library pages B.4
B.3. Standard library images B.5
B.4 Adding pages to the library B.6
B.5 Adding scripts to the library B.7
B.6 Adding images to the library B.8
B.7 User defined library directories B.9
B.8 Standard library templates B.10
C. FAQ C.1
C.1 Running REPORTER C.1
C.2 Generating output C.1
C.3 Extending REPORTER C.1
C.4 Other questions C.1
Answers C.1
D. JavaScript class reference D.1
global class D.2
Colour class D.8
File class D.16
Image class D.28
Include class D.36
Item class D.37
Options class D.58
Page class D.59
Reporter class D.64
Template class D.69
Utils class D.80
Variable class D.82
Window class D.86
E. Writing external programs/scripts E.1
Returning variables from programs E.1
Accessing existing variables in REPORTER E.1
Example program: Extracting the smallest timesteps (Text output) E.3
F. Unicode support F.1

F.1 Output formats that support unicode F.1
Installation organisation **G.1**
Version19.0.0 Installation structure G.1
Licences used in software **H.1**
Apple Public Source H.1
Draco H.1
Expat H.1
FreeType H.1
FFmpeg H.3
HDF5 H.5
Jpeg H.6
Libcurl H.7
Libgif H.7
Libpng H.7
Libxlsxwriter H.9
MPEG-LA H.10
Openssl H.10
PCRE2 H.12
PDFHummus H.13
POV-Ray H.13
SmoothSort H.13
Spidermonkey H.13
Treeview H.17
Turf H.18
Win-iconv H.18
x264 H.18
Zlib H.18

Preamble

Introduction

REPORTER is a tool to automate the post processing of LS-DYNA models. It allows you to create a standard template for a report. With command files and scripts it links with D3PLOT, PRIMER, and T/HIS, and other programs to create the necessary images and graphs when you come to generate an actual report from this template. It can also be run in batch mode so that when a model has finished being analyzed a report can be automatically generated according to a pre-built template.

Development Status

This manual documents REPORTER19.0 (part of Oasys Ltd LS-DYNA Environment19.0). The code is still being developed.

Systems supported

REPORTER is available for Windows (64 bit) and Linux (64 bit).

Revision History

Version 19.0

- Support for high-DPI displays has been improved. Text, icons, and widgets should now scale up in size appropriately when REPORTER is used with a high-DPI display. Windows Display scaling settings are respected for integer values.
  Fixes bugs 43343 and 45645.

- Now, when you open a template in REPORTER, if the saved values of DEFAULT_DIR and DEFAULT_JOB are empty or default, and if REPORTER was launched from D3PLOT or T/HIS in a linked session, then the values of DEFAULT_DIR and DEFAULT_JOB are updated to match the path and filename of the current model open in D3PLOT or T/HIS. This means that when capturing items from D3PLOT and T/HIS into templates, the variables will be used in place of hardwired paths (previously you would have had to edit those manually). This makes it easier to run the template on multiple different models.
  Fixes enhancement 44209.
- Drive mapping support has been added to REPORTER and can be controlled using the oasys*drive_? oa_pref settings.
  Fixes bug 16728.
- We have improved the handling of errors and warnings when generating content in REPORTER. Previously, you had to decide whether to stop generating the report before being able to see the error message. Now, you get to examine any errors and warnings before deciding whether to proceed.
  When generating a report, if an error is encountered, the Logfile window will now appear immediately. This gives you the opportunity to examine the error before choosing whether to ’Stop’, ’Continue’ (Logfile will reappear if another error occurs), or ’Continue to end’.
  You can now navigate up and down through the errors and warnings in the log file using the controls provided.
  You can prevent the Logfile window from appearing when an error occurs by unchecking ’Check for errors during report generation’. In this case, you will still be asked at the end of generation if you would like to inspect the log file.
  Also, the following options are now automatically saved to your REPORTER preferences upon selection:
  - Include debug information from PRIMER, D3PLOT, and T/HIS
  - Don’t automatically exit from PRIMER, D3PLOT, and T/HIS
  - Check for errors during report generation
  Fixes enhancement 39802.
- When an error occurs during report generation, or when a Script item calls Window.GetOptions, the Logfile or options window is now brought to the front of your display (or the icon on the taskbar will flash, depending on what is permitted by your operating system). This helps notify you that action is required, especially when a PRIMER, D3PLOT or T/HIS session may have been launched in front of REPORTER, temporarily obscuring it.
  Fixes bug 43178.
- Unlike PRIMER, D3PLOT and T/HIS, REPORTER does not support the Use function, or JavaScript modules. To help address this, a new "Script File" item has been added to REPORTER. Script File items have all the features of Script items, except that rather than containing an embedded JavaScript, they point to an external script file. This allows you to edit the script in your preferred editor. Furthermore, since the REPORTER JavaScript environment persists for the duration of the REPORTER session (and is shared by all scripts), you can now create one or more Script File items containing any common functions, followed by one or more Script File items that use those functions. This should provide much of the benefits of the Use function and JavaScript modules available in PRIMER, D3PLOT and T/HIS.
  Fixes enhancement 47275.
- When converting an item from one type to another, it is now possible to preserve the the item’s name.
  Fixes bug 46932.
- Previously, setting the font property of an Image object to a font family name that was not available in REPORTER would result in an exception being thrown. Now, the font name is mapped to the best available match in REPORTER.
  Fixes bug 47338.
- When hovering over an animated Item with the Hand tool, forcing a page switch using Generate or switching to Design View using the ’p’ shortcut would cause the hover controls to remain on screen unintentionally. Now fixed so that hover controls no longer persist.
  Fixes bug 45908.
- REPORTER now respects the OA_INSTALL_xx environment variable (taking precedence over OA_INSTALL environment variable, but not oasys*install_dir oa_pref option).
  Fixes bugs 39730, 47855.
- Fixed a crash that occurred while launching the HTML manual through the Help menu on Unix systems. Updated code for launching HTML manual to work with a wider range of default browsers on Windows.
  Fixes bug 45812.
- The default image name for a new T/HIS Item is now incremented correctly to provide a unique name.
  Fixes bug 47780.
- Fixed an issue in which pre-existing D3PLOT and T/HIS items of blank type were not being given a new default image file name when changing type.
  Fixes bug 47791.

Version 18.1

- Corrected neck and femur inputs for Humanetics Harmonized HIII 5F v2.02 model dummies so that they use cross sections rather than beams (in a small number of templates). Further investigation into how the Harmonized model varies from the regular version is currently underway (Case 46414).
  Fixes bug 46756.
- T-HIS Items with JavaScript type and Image output were not producing an image on Generate. Now fixed to behave in the same manner as old method FAST-TCF capture. Old T-HIS Items with JavaScript type and Image output that had been configured to work correctly by writing the image from the JavaScript may need to be edited.

Fixes bug 46521.
- Right-aligned text was sometimes clipped in PDF output. Now fixed.
  Fixes bug 43230.
- Improved REPORTER's handling of filepaths for the Autotable 'File' option. Updated guidance has been added to the manual.
  Fixes bug 46292.
- When running REPORTER in batch from SHELL with REPORTER iconised, an odd windowing bug was causing the Combined Output Options to sometimes produce corrupted files. This should no longer be the case.
  Fixes bug 45660.
- Updating an image capture for a D3PLOT Item with an image file extension of .jpg, .bmp, or .ppm will no longer overwrite these extensions with .png.
  Fixes bug 46113.
- %TEMPLATE_DIR% was not set early enough when a Template was being opened, causing Image Items that were attempting to use this variable to appear as 'missing images'. Now fixed.
  Fixes bug 46258.
- The filetype, job, resolution, and cropping properties have been added to the Javascript Item class (for certain Item types).
  Fixes enhancement 45772.

Version 18.0

- Support for the playback of MP4 movies and animated GIFs has been added to REPORTER. These can be captured directly from D3PLOT for use with a D3PLOT Item, or be added to a template through an Image or Image File Item. Playback can be controlled by hovering over the Item or using the buttons in the new animation toolbar.
  Fixes enhancements 3485, 9337, 13491, 29174, and 36334.
- The "LS-DYNA Version and Revision" version.js Library Program script has been updated so that it works with newer d3hsp/OTF files that can include (1) An Ansys legal notices header above the LSTC header (2) LS-DYNA revision names incorporating the newer Git revision hashes instead of the older SVN revision numbers.
  Fixes bug 45562.
- The Legacy GUI theme has been deprecated and is no longer accessible from the Preferences dialog. If you wish, you can still select Legacy theme via reporter*guit_theme in your oa_pref file.
  Fixes enhancement 45506.
- If you open a template created in REPORTER 16.1 or earlier in REPORTER 18.0 or later, any D3PLOT and T/HIS items will continue to be captured and generated using the old pre-version-17 method (in versions 17.0 and 17.1, they were automatically converted to the new method, but this caused problems for some users' templates).
  Fixes bug 45224.
- Along with the rest of the Oasys Suite, REPORTER has transitioned to LM-X licensing for version 18.0, so the FLEXlm licence is no longer supported.
  Fixes enhancement 44733.
- The first sixteen colours saved in PRIMER, D3PLOT or T/HIS are now read into REPORTER via the user_colours.xml file and are accessible via the 'Custom colors' panel (select 'More colours...'). Any user colours added in REPORTER via 'Add to Custom Colors' can be saved to user_colours.xml via the new 'Fonts and Colours' tab in the Preferences dialog and will thereafter be accessible in PRIMER, D3PLOT and T/HIS.
  Fixes enhancement 44540.
- Font and user colour settings can now be saved directly from REPORTER via the new 'Fonts and Colours' tab in the Preferences dialog. There is also a new 'Startup' tab with preferences for maximising the main window, and also to specify the starting directory for REPORTER (previously these preferences could only be modified via the preferences editor in the other programs).
  Fixes enhancement 39537.
- The 'LS-DYNA' tab in the Preferences dialog has been renamed 'Oasys Items'. All of the settings in the Program Locations dialog have been moved to this tab, and they can now be saved from REPORTER along with the other preferences.
  Fixes enhancement 42966.
- You can now use the -pptx command line argument to trigger PowerPoint output. Both -pptx and -ppt arguments have the same behaviour (they output .pptx files).
  Similarly, the JavaScript API method Template.Ppt has been deprecated in favour of Template.Pptx (both continue to output .pptx files).
  Fixes enhancement 44261.
- REPORTER has been able to write PowerPoint (.pptx) files since version 11. As of version 18, support for the old .ppt VBA output has been removed.
  Fixes enhancement 44259.
- All Dialogs accessed via the Menu bar (or created via JavaScript) should now appear on the same display screen as the Main Window. If manually repositioned, these Dialogs should remember their previous position when being reopened. When moving the Main Window to a new display screen, all of these Dialogs should follow to the new screen (excluding an open Logfile or a maximised Dialog).
  Fixes bug 14096.
- All MainWindow keyboard shortcuts can now be used while the Logfile has focus.
  Fixes bug 43423.

- You can now specify a font mapping table CSV file in a location other than the installation location, to make customised font mapping more convenient.
  Fixes enhancement 44127.
- The JavaScript function Template.EditVariables now accepts an optional bool argument to determine whether selected Variables should be displayed alphabetically (true) or in the list order in which they were passed to the function (false).
  Fixes bug 44362.
- The page number listed in the page navigation box will now correctly correspond to the current page (usually the last in the Template) after writing a PDF, Pptx, or HTML file.
  Fixes bug 45394.
- The page number in the MainWindow title bar will now correctly display the current page (usually the last in the Template) after report generation.
  Fixes bug 45632.
- Filename in the MainWindow title bar will now correctly have an asterisk appended to it to indicate the file has been modified when Variables are edited or updated.
  Fixes bug 43422.
- D3PLOT Items will no longer sometimes fail to load the image when conducting a fresh ’old method’ Capture.
  Fixes bug 45056.

Version 17.1

- Fixed an issue where D3PLOT and T/HIS sessions launched by REPORTER as part of a batch process would sometimes fail to terminate when REPORTER closed, if the batch process included the command line argument -exit.
  Fixes bug 44451.
- When using the drag feature to resize multiple Items simultaneously using a handle type that is not present on both Items (e.g. the non-corner handles specific to ’rectangular’ Items), using this handle will no longer resize the Item that would not normally have this handle.
  Fixes bug 44279.
- If an image file for an Image Item cannot be located when opening a Template, a ’missing image’ icon is now printed on the page in its place.
  Fixes bug 44278.
- The erroneous ’Cannot crop image’ warning will no longer appear in the Logfile when first creating an Image Item (prior to choosing an image file).
  Fixes bug 44276.
- If the reporter_font_cache file became corrupted, this could cause REPORTER to crash upon opening. Now fixed.
  Fixes bug 44257.
- Report Page should no longer appear to change size (without any change in zoom percentage) after generating a thumbnail (e.g. by saving the Template or opening the Template Properties window).
  Fixes bug 44049.
- Warning messages for missing fonts when loading a Template are now displayed only once for each font. The same is true for an incompatible Font + Style combination.
  Fixes bug 43639.
- Characters like ’<’ no longer interfere with the formatting of certain messages in the Logfile.
  Fixes bug 43903.
- REPORTER now correctly reads the CURRENT directory oa_pref file from the directory pointed to by the start_in preference when this is used in the OA_ADMIN, OA_INSTALL, or OA_HOME oa_prefs.
  Fixes bug 43902.
- A warning message is now printed to the Logfile if any of PRIMER, T/HIS, or D3PLOT can’t be located on starting REPORTER.
  Fixes bug 43189.
- Edit windows are now mapped to appear above the selected Item in all cases.
  Fixes bug 43201.
- Edit D3PLOT Dialog no longer maps to the top-left of screen 1.
  Fixes bug 43579.
- All MainWindow keyboard shortcuts can now be used while the Logfile has focus.
  Fixes bug 43423.
- It is no longer possible for the first click using the Select tool after saving a Template to select the wrong Item or coordinates on the page.
  Fixes bug 43462.
- Item outlines set to none using lineStyle=Reporter.LINE_NONE in the javascript API are no longer printed when writing to pdf or pptx.
  Fixes bug 43975.
- Selection box now updates correctly when moving invisible Items with no outline, fill, or text colour.
  Fixes bug 43231.
- Recent Files list is now limited to 50 files and no longer deleted when opening Reporter while skipping the Choose Template window (e.g. by opening a Template from File Explorer by double-clicking on the .ort file).
  Fixes bug 43609.
- When a D3PLOT item created with the old (v16) capture method was updated using the new (v17.0) method, it remained flagged as using the old method. Now fixed. Also, it was possible to update the parent of an old

multi-capture D3PLOT item using the new method, leading to orphaned children. Now, a warning is shown to prompt you to convert the item to the new method first.
Fixes bug 43613.

Version 17.0

- REPORTER can now be linked to both D3PLOT and T/HIS, or the D3PLOT->T/HIS link, by opening one program from another. This allows reports to be created and edited interactively. Windows and graphs can be captured into REPORTER easily and reloaded back from REPORTER, all in the same session. When generating reports, D3PLOT and T/HIS items will be generated in the same session of their respective programs without loading the same model more than once.
  Fixes enhancement 40886.
- Custom cell borders in Tables are no longer displayed if the table line colour is set to ’none’. The Cell Borders window now starts with the custom border width set to the current line width setting for the table rather than ’none’. Non-custom border widths no longer revert to the original line width of the table when adding new custom borders.
  Fixes bug 43091.
- In the JavaScript API, a new constructor has been added for the Page class, with an options Object argument allowing page name, colour and index to be specified. The original constructor is now deprecated.
  Fixes enhancement 41733.
- REPORTER’s user interface has been upgraded, with new Light and Dark themes, new icon and cursor designs, and improved toolbars.
  Fixes enhancement 37477.
- The cursors for the different Tools have been updated to make it clearer when you need to drag a box, or just click somewhere on the page.
  Fixes enhancement 35877.
- A range of standard template layouts is now provided with REPORTER to provide creative inspiration, and to help you quickly create reports for a variety of applications.
  Fixes enhancement 13490.
- A recent files list has been added to the library template selection dialog and can be accessed from the File menu.
  Fixes enhancement 8820.
- The close button (’X’) has been enabled for all dialog windows.
  Fixes enhancement 42660.
- It is now possible to deselect individual items using Ctrl + Click or Shift + Click when the user has multiple items selected.
  Fixes enhancement 42457.
- A splash screen showing new features in now displayed when REPORTER opens.
  Fixes enhancement 42683.
- Formatting control for the DATE variable has been added using %DATE(format)%.
  Fixes enhancement 42117.
- It is now possible to select all items on the current page with the shortcut Ctrl+A and to deselect all items in the current template with the shortcut Ctrl+Shift+A or the Esc key.
  Fixes enhancement 42659.
- Reporter could occasionally write error messages ’QFont::setPixelSize: Pixel size <= 0’ to the terminal window on Linux. This has now been fixed.
  Fixes bug 41639.
- If REPORTER was launched on a server running a virtual display using Xvfb versions 1.18 or later, REPORTER would abort with the message ’Floating point exception(core dumped)’. Now fixed.
  Fixes bug 42402.
- The Logfile now displays more licensing information when REPORTER starts.
  Fixes enhancement 42196.
- It is now possible to save preferences directly from REPORTER via the ’Save preferences’ button on the preferences dialog (accessed from File -> Preferences). Various new preferences have been added to improve customisation of REPORTER. The Template Generation preferences have been moved to the new Template -> Properties menu.
  Fixes enhancement 42116.
- You can now File -> Save As... Template, Report, PDF, PowerPoint and HTML. You can still write PDF, PowerPoint and HTML from File -> Write PDF etc.
  Fixes enhancement 42030.
- The Automotive library templates now support a range of dummy models from different suppliers. The default entity IDs/labels are now provided to make setup easier.
  Fixes enhancement 27000.
- New GUI themes (Light and Dark) have been added to give REPORTER a modernised look and feel. The Legacy theme will continue to be supported for now, but support may be removed in future versions.
  Fixes enhancement 37479.
- A ’Generate’ toolbar has been added to the top of the main window as another way of generating templates, pages and selected items.
  Fixes enhancement 17290.
- A ’Page’ toolbar has been added to the top of the main window to improve access to page creation, deletion, duplication, and page navigation controls.
  Fixes enhancement 42466.

- With ’Snap to grid’ enabled, items on the page no longer snap when selected (only when moved). Changed behaviour of snapping such that items snap to grid based on chosen reference corner defined in preferences. When snap size exceeds nudge size, nudge now uses snap size instead to ensure nudging always possible.
  Fixes bug 42328.
- Sizes of rows/columns in tables no longer automatically reset unless the ’Reset heights’ or ’Reset widths’ buttons are explicitly pressed. A new checkbox has been added to the Edit Table Dialog: ’Fix overall table size while adding/deleting/resizing rows and columns’. When unticked, row/column operations are able to adjust overall table size. When ticked, rows/columns scale proportionally after each operation to fit original table size. A similar checkbox has been added to Edit Autotable Dialog, applied only to columns.
  Fixes bug 41773.
- Trying to close a session with modified files prompts the user to save changes. REPORTER would close anyway even if the user cancelled the save dialog. Now fixed.
  Fixes bug 41935.
- When reading a T/HIS item containing a FAST-TCF script, REPORTER would remove any semicolons (e.g. found in entity names). Semicolons are now preserved.
  Fixes bug 41735.
- In the JavaScript API Image class, it is now possible to set an alternative background colour (including ’none’) via a new third argument in the Image constructor. Also, previously, setting lineColour to ’none’ had no effect, and setting lineWidth equal to zero resulted in a line width of 1. Now fixed so that either setting will result in no line being drawn.
  Fixes enhancement 41602.
- Problems encountered when modifying individual Table cell border widths via the JavaScript API, especially when reducing the existing border width, have been fixed.
  Fixes bug 42600.
- The page orientation now automatically changes to landscape when you select a PowerPoint page size in File -> Page setup.
  Fixes enhancement 41651.
- Image size can now be controlled for ’Blank’ PRIMER, D3PLOT and T/HIS items (so that any images created via scripts or command files can have image size control).
  Fixes enhancement 41674.
- Templates containing the text "%CURRENT_PAGE%/%TOTAL_PAGES%" would result in an error message when written to PowerPoint. Now fixed.
  Fixes bug 41676.
- The Tools have been grouped into categories, and it is now possible to switch on labels for the tool buttons.
  Fixes enhancement 41603.
- Improved access to the page master view: a page master view toggle button (shortcut key ’m’) has been added to the View toolbar, and corresponding toggles have been added to the View and Page menus. The old Master dock widget has been removed.
  Fixes enhancement 41604.
- Newly-created items now remain selected, so that they can more conveniently be moved, resized or deleted. Keyboard shortcuts have been added for the Select tool (’s’) and the Hand tool (’h’).
  Fixes enhancement 40700.
- Images drawn using the JavaScript API Image class are now drawn with antialiasing to improve image quality.
  Fixes enhancement 41022.
- The attributes (geometry, style, font, paragraph, alignment) of all selected items can now be controlled via new toolbars in the main window.
  Fixes enhancement 7764.
- It is now possible to set the line width and line colour of Autotable items. It is now also possible to set the fill and text colour for all cells/columns in Table/Autotable items via the Style toolbar.
  Fixes enhancement 40106.

Version 16.1

- When opening a template was aborted, it could sometimes cause REPORTER to crash. Now fixed.
  Fixes bug 41717.
- When a Table or Autotable item was created or resized, it would actually be given 0.99 times the requested width and height. Now fixed.
  Fixes bug 40697.
- Page hyperlinks did not work for Table items in HTML output, and they did not work for Table or Autotable items in PDF output. Now fixed.
  Fixes bug 39208.
- Page hyperlinks were not given the underlined magenta style for Table items in Presentation view. Now fixed.
  Fixes bug 39207.
- It was not possible to get/set the lineColour property for Table items via the JavaScript API. Now fixed.
  Fixes bug 40105.
- If items are moved or resized using the mouse, or if items are aligned, distributed or rearranged from the context menu, the changes are now recorded as changes to the template, and a save prompt appears when the template is closed.
  Fixes bug 39981.
- If captures were deleted from a D3PLOT item that had already been edited, REPORTER would crash. Now fixed.

Fixes bugs 38195 and 40531.

- In Presentation view, if the location of a Script item was clicked with the Hand tool, the script would run, even if the Script item was not a button script. Now fixed so that only button scripts can be run by clicking on them.
  Fixes bug 40016.
- For non-legacy fonts in Tables and Autotables, cell text was aligned incorrectly in PDF output if ’middle’ text alignment was selected. Now fixed.
  Fixes bug 40065.
- Setting the output of an Autotable Library Program to a variable and clicking OK would cause REPORTER to crash. Fixed by hiding this Library Program feature for Autotables, since it was inapplicable in the first place.
  Fixes bug 40602.
- If an item containing a Program or Library Program was generated via the Item.Generate method in a Script item, the Program or Library Program output would not be returned. Now fixed.
  Fixes bug 40384.
- In the JavaScript API, some of the properties in the objects returned by Item.GetCellProperties and Item.GetColumnProperties were incorrect for Program and Library Program cells. These have been fixed. A new <output> property has been added for Item.GetCellProperties, and a new <programArgs> property has been added for Item.GetColumnProperties. Also, the <program> and <programArgs> properties were not set correctly by Item.SetCellProperties and Item.SetColumnProperties. Now fixed.
  Fixes bugs 40223 and 40528.
- The intrusion plot in the General LS-DYNA Vehicle template now supports parts of any element type, rather than only shell parts.
  Fixes bug 40351.
- If an *INCLUDE or *INCLUDE_PATH card contained valid white space at the beginning of a continuation line, it was incorrectly eliminated when read by certain Library Program scripts. Now fixed.
  Fixes bug 40293.
- The *INCLUDE_PATH card was not supported by Library Programs that recursively searched a keyword file and its include files. Support for *INCLUDE_PATH now added.
  Fixes bug 40291.
- Standard templates now support results files with the newer LSTC naming conventions "<name>.d3plot" and "<name>.d3hsp" as well as older "d3plot" and "d3hsp" filenames. In addition, the templates will now search the results directory for any "*.ptf" or "*.d3plot" file if one matching the keyword filename cannot be found.
  Fixes bug 40229.
- When generating a PRIMER object the MENU_AUTO_CONFIRM environment variable is no longer set by default. This is because when creating/updating the capture it was not set and the mismatch meant that some macros did not play correctly. If a specific template needs this for some reason it can be set by using the option in the templates preference.
  Fixes bug 40591.

Version 16.0

- Previously, REPORTER only supported four fonts (Courier, Helvetica, Times, Symbol). REPORTER now supports many more fonts (TrueType, OpenType, and certain Type1 fonts), giving you greater control over the look of your reports, and allowing you to create templates that match your organisation’s branding. Support for Chinese, Japanese, Korean and other non-Latin fonts is much improved.
  Fixes enhancements 9008 and 15826, and bug 16376.
- Table and Autotable items can now be exported in Microsoft Excel format, complete with formatting (cell size, text alignment, font style, borders, colours, merged cells).
  Fixes enhancement 38249.
- Various new functions have been added to the Item class of the JavaScript API to enable control over Table and Autotable items. It is now possible to: insert/delete/resize rows/columns, merge/unmerge cells, get/set cell properties (e.g. text, alignment, font, colour, border width) and get/set cell conditions.
  Fixes enhancements 38250 and 38251.
- PNG images with transparency would appear white in PDF output. Now fixed.
  Fixes bug 38792.
- After a PRIMER, D3PLOT or T/HIS capture, the special readonly variables (e.g. REPORTER_TEMP) were being set as writeable, meaning that they would be mistakenly written to the template/report file if it was subsequently saved. Furthermore, after a capture, all variables were being changed to temporary variables. Both now fixed.
  Fixes bug 38791.
- When editing a variable of type ’Directory’, if "Browse..." was clicked to browse for a directory with a UNC path (e.g. "\\example.com\data\analysis\001"), the value returned would contain forward slashes ("//example.com/data/analysis/001"). Backslashes are now preserved.
  Fixes bug 39303.
- When running on Linux a warning "libpng warning: iCCP: known incorrect sRGB profile" would be written to stdout. Now fixed.
  Fixes bug 38711.
- The red, green, blue and name properties in the Colour class did not work. When getting the property null would be returned instead of the correct value. Now fixed.
  Fixes bug 38772.

Version 15.1

- Various library scripts would terminate with an error if include files in keyword files could not be found. They now print a warning instead.
  Fixes bug 38400.
- Table cells with fill colour ’none’ were being saved correctly, but would be interpreted as ’black’ during copy/paste. Now fixed.
  Fixes bug 38334.
- If the border width of Table items was set to 0.75, it was not saved correctly. Also, cell borders were not preserved when a Table item was copied. Both fixed.
  Fixes bug 37666.
- Lock symbols for some items remained visible in Presentation view. Now fixed.
  Fixes bug 37524.
- The getter for the JavaScript Image class font property did not work correctly. Now fixed.
  Fixes bug 38019.
- If a report (.orr file) was opened by double-clicking on it, scripts set to automatically run on opening would run. This is the intended behaviour for templates (.ort files), but not for reports. Now fixed.
  Fixes bug 36531.
- After aligning selected items, their positions were not redrawn immediately. Now fixed.
  Fixes bug 37803.
- It was possibly to nudge and delete locked items. It was also possible to move them via the alignment options in the context menu. These are now disabled for locked items.
  Fixes bug 37393.
- With the exception of the Logfile dialog, all dialogs are now modal, to prevent instability and unexpected behaviour.
  Fixes bugs 36484, 36660 and 36661.
- A warning message would appear when an empty page was duplicated. Now fixed.
  Fixes bug 36304.
- Script buttons were highlighted incorrectly on hover in Presentation view. Now fixed.
  Fixes bug 37188.

Version 15.0

- The shortcut (Ctrl+W) for File -> Close didn’t work. Now fixed.
  Fixes bug 37062.
- When inserting variables into a text field (Ctrl+I), you can now double-click to select the variable (increases speed of use).
  Fixes enhancement 36640.
- Reporter could crash if the File->Open library template menu was entered before the list of library templates had been read. Now fixed.
  Fixes bug 36642.
- On Linux, if REPORTER was started from the command line with a file argument (e.g. reporter14_x64.exe example.ort) then the REPORTER window would be shown too high so the window title bar was not visible. Now fixed.
  Fixes bug 36503.
- Depending on the relative aspect ratio between template and display, Zoom -> Fit width and Zoom -> Fit height would not fit correctly (scroll bars would persist). Now fixed.
  Fixes bug 36438.
- On Windows, if REPORTER was started from the command line with both maximise and file arguments (e.g. reporter14_x64.exe -maximise example.ort) then the REPORTER window would not maximise. Now fixed. Also, the active template now resizes in synchrony with the main window.
  Fixes bug 36017.
- A special readonly TEMPLATE_DIR variable has been added to REPORTER. The variable value contains the directory path of the current template (for a new template created using File -> New, TEMPLATE_DIR will contain an empty string until it has been saved). The TEMPLATE_DIR variable should be useful when you want to refer to files (e.g. images or scripts) stored relative to the template.
  Fixes enhancement 35693.
- Oasys REPORTER executables weren’t always being digitally signed. Now fixed.
  Fixes bug 35004.
- It is now possible to embed Image items into a template (.ort file). By checking the box in the Image item dialog, the image data is embedded directly into the template rather than relying on the link to an external image file.
  Fixes enhancement 33088.
- A checkbox has been added in the Script item dialog to skip the generation of button scripts when a template or page is generated. This means that button scripts can now be configured to run only when clicked.
  Fixes enhancement 34966.
- The Variables dialog now expands to fit variables with long names, values or descriptions, rather than pushing them to the right of the visible pane in the scrollable area.
  Fixes enhancement 34965.
- In PowerPoint files written by REPORTER, Text, Textbox, Table and Autotable items now use ’Exactly’ rather than ’Single’ line spacing in order to improve visual compatibility with REPORTER.

Fixes bug 35338.
- Read-only properties ’filename’ and ’path’ have been added to the Template Class in the JavaScript API. These replace the ’name’ property, which has been deprecated (and made read-only) from this version onwards.
  Fixes enhancement 35186.
- Variable value replacement could previously fail for nested variables at the beginning of a string. Now fixed.
  Fixes bug 35152.
- Improvements have been made to the keyboard focus in all REPORTER dialog windows. Tab focus is now consistent and the enter key now has the expected effect.
  Fixes bug 35225.
- Editing margins and vertical text justification were not possible in Autotables. They are now possible.
  Fixes bug 30631.
- If the Window.Information Class function was used in a Script object, an ’Error’ icon was displayed instead of an ’Information’ icon. Now fixed.
  Fixes bug 34971.
- When in Presentation view, if an object was created by dragging out a rectangle to define its size, the rectangle would continue to change size after the mouse button was released. Now fixed.
  Fixes bug 35036.
- Toggling of the ’View page item generation order’ button was not synchronised with the checkbox on the View menu. Now fixed.
  Fixes bug 35130.
- If the Design view button was clicked when already in Design view, it would toggle to the Presentation view, and vice versa. Toggling now removed.
  Fixes bug 35098.

Version 14.1

- If a variable was saved with a floating format precision of 0 the precision was not read correctly when reading a template file. Now fixed.
  Fixes bug 34590.
- If a table cell background colour was set to ’none’ it was not read correctly when reading a template file. Now fixed.
  Fixes bug 34588.
- If a variable contained an expression then the expression would be overwritten by the evaluated value after D3PLOT or T-HIS objects were run. Now fixed.
  Fixes bug 34566.
- If a variable contained a format precision then the precision would be overwritten with the default after D3PLOT or T-HIS objects were run. Now fixed.
  Fixes bug 34556.
- If REPORTER was started by right clicking on a template file and using ’Open with...’ the current directory was set to C:/Windows/System32 on Windows. The current directory is now set to the directory that the template file is read from.
  Fixes bug 34555.
- The title library script would not extract the title from an include file if the include file was specified using " +" continuations. Now fixed.
  Fixes bug 34511.
- If a program argument contained backslashes then REPORTER would convert them to forward slashes for internal storage but then not convert back to backslashes on Windows when running the program. This could make some external programs fail (e.g. Perl scripts). Now fixed.
  Fixes bug 34254.
- T-HIS objects using the JavaScript type did not pass the job file correctly to T-HIS. Now fixed.
  Fixes bug 33922.
- If a D3PLOT object captured a plot for one model and was then replayed on a slightly different model then sometimes the wrong elements could be (un)blanked in the plot. Two new options have been added for D3PLOT in File->Program Locations to help in these situations. These options can also be set by preferences.
  Fixes bug 33886.
- If the border colour for a table was set to ’none’ this was not correct in PowerPoint files produced by REPORTER. Black was incorrectly used. Now fixed.
  Fixes bug 33197.
- The logic in REPORTER for reading variables assumed that the name would only contain A-Z, a-z, 0-9 and underscore characters. However the variable dialog and some scripts actually allowed many other characters and this meant that the variable was not processed correctly if other characters were used. The logic has now been changed and variable names can now contain spaces (converted to a single underscore character), A-Z, a-z, 0-9 and the following special characters ’{’, ’}’, ’[’, ’]’, ’/’, ’\’, ’@’, ’.’ and ’^’. This allows units to be included in variable names. For example the following is now a valid variable name:
  ACCELERATION_[mm/s^2]
  Fixes bug 31871.

Version 14.0

- When editing an auto table and updating the backgound colour in conditional formatting the table editing

window would be lowered so it was hidden. Now fixed.
Fixes bug 33482.
- When editing conditional formats the background colour could be shown incorrectly. Now fixed.
Fixes bug 33481.
- If you specified that the output from a library program should be saved in a variable it did not work if the variable didn't already exist. Now fixed.
Fixes bug 33329.
- REPORTER could give the wrong initial added mass, initial percent added mass and smallest timestep values for some otf files. Now fixed.
Fixes bug 33272.
- Updating a capture for a D3PLOT object did not work if there was a pre-JavaScript defined. Now fixed.
Fixes bug 33232.
- REPORTER could crash if a temporary variable was deleted by using Delete Temporary Variables in the Variables menu and recreated. Now fixed.
Fixes bug 32988.
- Rows and columns can now be added or deleted from any location in a table.
Enhancements 12788, 13840, 31875.
- Cells in a table can now be merged.
Enhancements 15549, 22387, 31875.
- The borders can now be set for individual cells in a table.
Enhancements 12855, 15549, 22387.
- Conditional formatting can now be copied and pasted from one cell to another.
Enhancements 13842, 15549, 22387.
- The output from a library program and text can now be used together in a table cell.
Enhancement 13866.
- Library programs can now not produce output. This may be useful in some situations. For example the output can set to a variable and the variable used later in a table cell.
- A File.Move method has been added.
Enhancement 31605.
- A File.Copy method has been added.
Enhancement 31603.
- When replacing variables in D3PLOT JavaScripts \ was not escaped to \\. Now fixed.
Fixes bug 32631.

Version 13.1

- The icon shown in the top left of the titlebar when a template was opened was the old (version 12) icon instead of the new (version 13) icon. Now fixed.
Fixes bug 31046.
- The -iconise command line option did not work correctly on linux. Now fixed.
Fixes bug 30982.
- The elapsed time library scipt did not work for the R8 and R9 versions of LS-DYNA. Now fixed.
Fixes bug 30565.
- Endash characters (–) were incorrectly written to pdf files as spaces. Now fixed.
Fixes bug 30654.

Version 13.0

- The aspect ratio or size of images in D3PLOT, PRIMER and T/HIS objects can now be controlled.
Enhancement 29004.
- Images in ImageFile, D3PLOT, PRIMER and T/HIS objects can now be justified.
Enhancement 28875.
- An optional index argument has been added to the Page.Duplicate() method.
Enhancement 28334.
- The name argument in the Item constructor was ignored. It is now used correctly.
Fixes bug 28332.
- Variables can now be marked as temporary variables. Temporary variables can be removed from a template with a new 'Delete temporary variables' command in the Variables menu.
Enhancement 27253.
- The format and precision properties were missing from the Variable class. They have now been added.
Fixes bug 22881.
- An ImportItem method has been added to the Page class.
Enhancement 28221.
- The numbering of 'generated' items on a page missed out tables. Tables have been added as they can contain scripts that need to be generated.
Enhancement 27384.

Version 12.1

- When combining reports if one of the reports to combine did not exist hyperlinks could link to the wrong pages. This has been fixed.
  Fixes bug 27640.
- Values for variables with expressions were not written to the reporter_variables file correctly when generating the report. This has been fixed.
  Fixes bug 27256.
- REPORTER would delete files in the temporary directory if the directory was set with the oasys*temp_dir preference.
  Fixes bug 26416.
- Ampersand characters (&) in table objects created corrupt pptx files. This has been fixed.
  Fixes bug 26906.

Version 12.0

- Circles and ellipses were not rendered correctly in pdf files. Now fixed.
  Fixes bug 25848.
- REPORTER could crash if a library image was inserted very near the top or right hand edge of a page. This has been fixed.
  Fixes bug 24988.
- FAST-TCF objects did not use the job filename when generating. FAST-TCF would look in the directory for the job. If there were multiple jobs in the same directory it could choose the wrong file. This has been fixed.
  Fixes bug 24979.
- Extra commands added to a D3PLOT capture which were longer than 80 characters did not work. This has been fixed.
  Fixes bug 24933.
- Objects can now be locked on the page so they cannot accidentally be moved.
  Enhancement 23503.
- REPORTER will now prompt the user to replace variables in the macro after doing a capture in PRIMER. For each matched text string you can choose whether to replace it with a variable or you can do "Yes to All" or "No to All".
  Enhancement 21309.
- A new Batch() method has been added to JavaScript so scripts can test if REPORTER is running in batch mode.
  Enhancement 23990.
- A Duplicate method has been added to the Page class.
  Enhancement 17602.
- Static methods GetAll and GetFromName have been added to the Variable class.
  Enhancement 22772.
- Report hyperlinks can now have the form ’#page title’ to link to another page in the report.
  Enhancement 22974.
- A DeletePage() method has been added to the Template class in JavaScript.
  Enhancement 22974.
- A new EditVariables method has been added to the Template class so that a variables editing panel can be called from JavaScript.
  Enhancement 22387.
- A System() method has been added to JavaScript.
  Enhancement 15772.
- Script objects can now be shown as a button in presentation mode. Clicking the button runs the script.
  Enhancement 22387.
- A single script object in a template can now be set to run automatically when the template is opened.
  Enhancement 22387.
- A new Window class has been added to JavaScript to enable standard dialogs to be used.
  Enhancement 22387.
- T/HIS objects can now be defined with a JavaScript instead of a FAST-TCF script.
  Enhancement 22387.
- New Page and Item classes have been added to JavaScript.
  Enhancement 22387.
- Library templates can now be opened directly from the File menu.
  Enhancement 22387.
- A File.Delete() method has been added.
  Enhancement 22387.

Version 11.2

- Less than signs (<) in text and textbox objects created corrupt pptx files. This has been fixed.
  Fixes bug 24613.
- If the Variable constructor was used to redefine an existing variable in a script REPORTER would give an error if you tried to get any of the variable properties later in the script. This has been fixed.
  Fixes bug 23586.

Version 11.1

- If a D3PLOT object with multiple captures was updated again in D3PLOT to delete or replace captures REPORTER could produce errors when generating or change the order of the captured images. This has been fixed.
  Fixes bug 22227.
- If a D3PLOT object was made that had more than 20 captures and the template saved to file , REPORTER would not be able to read the file again. This has been fixed.
  Fixes bug 22226.
- D3Plot objects using a command file (instead of capturing) did not work in version 11. This has been fixed.
  Fixes bug 22147.
- When saving old style T/HIS objects (i.e. using a command file instead of a FAST-TCF script) the command file was not saved. This has been fixed.
  Fixes bug 22117.
- Opening the HTML manual did not work on Linux. This has been fixed.
  Fixes bug 21989.

Version 11.0

- When writing PowerPoint files REPORTER now correctly writes animated gifs.
  Fixes enhancement 17601.
- REPORTER could crash if you created a table that used a library program for a cell and you saved the output of the program to a variable. This has been fixed.
  Fixes bug 21346.
- The library program which reported the LS-DYNA version and revision from the otf file did not work correctly for new (R7) LS-DYNA output because there is now a new ’SVN Version’ line in the otf file. Additionally the version and revision were expected to be to be a single ’word’. This has been fixed.
  Fixes bug 21243.
- Outlines were not written for Oasys Ltd. or File Image type objects to PowerPoints. Now added.
  Fixes bug 21242
- REPORTER would hang when reading a template file if one of the page titles in the file contained an ampersand (&). This was because the ampersand was not escaped properly when writing the template. This has been fixed.
  Fixes bug 21235
- You can now specify an outline border for file image objects.
  Fixes enhancement 18206
- REPORTER can now use D3PLOT to generate multiple images in one session. The second and subsequent images are automatically created as image file objects linked to the d3plot object.
  Fixes enhancements 7777 and 13034
- A JavaScript can now be run for D3PLOT and PRIMER objects.
  Fixes enhancement 15550
- When capturing an image from D3PLOT, REPORTER now automatically shows the images.
  Fixes enhancements 7779 and 10668.
- A new PRIMER object has been added.
  Fixes enhancements 8095 and 16530
- REPORTER can now write PowerPoint pptx files directly.
  Fixes enhancement 11858
- REPORTER can now combine multiple reports into a single pptx/pdf/html.
  Fixes enhancements 7712, 8956, 9020 and 10742
- REPORTER could think that a script had changed when cancelling from the editor if the script was created on windows but edited on unix.
  Fixes bug 7769
- When writing a pdf file jpeg images are now written as jpegs rather than pngs as they can be much smaller.
  Fixes enhancement 17920
- Added the ability to see the item generation order.
  Fixes enhancement 18489

Version 10.2

- REPORTER did not automatically change LS-DYNA filenames from h3hsp to %DEFAULT_JOB%.otf (and visa-versa) when importing a library page. This has been fixed.
  Fixes bug 19200
- REPORTER could crash when writing a pdf file that had overflow pages in an auto-table if there was an error when the report was generated. This has been fixed.
  Fixes bug 19197
- The "cropping" button was the default focus in the D3Plot object edit menu (i.e. was applied when hitting enter) rather than the "OK" button. This has been fixed.
  Fixes bug 19113
- REPORTER was not able to create and import image files which were not JPEG when generating a D3Plot captured object. This has been fixed.

Fixes bug 18403
- REPORTER could crash if the user added a page to the reporter_library/pages area which contained certain REPORTER items. This has been fixed.
  Fixes bug 18432

Version 10.1

- If the page layout is changed from landscape to portrait or visa versa any items that are off the page are automatically moved to stay on the page
  Fixes bug 14307
- If multiple conditional formatting conditions were set for a table, autotable, textbox or file object background, then REPORTER would display the last condition matched rather than the first one. This has been corrected.
  Fixes bug 17794

Version 10.0

- Added the -loghtml command line options to allow the log file to be saved as html instead of plain text.
- Added a Templates tab to preferences to allow the user to change whether existing files should be overwritten when generating images for multiple pages in T/HIS. This is saved as a property of each template
- Added the -iconise and -oasys_batch command line options
- Checkbox for turning on/off error checking during generation when an error was found was not working correctly.
  Fixes bug 15143
- Added the ability to set the format of a variable on the variable edit panel.
  Closes enhancement 8819.
- Fixed problem with rounding errors on spinbox input values on edit panels.
  Fixes bug 15548.
- When resizing/moving a table object, the relative width/height of the columns/rows is now maintained.
  Closes enhancement 15546.
- Added a new library script for reading variables from a CSV file.
  Closes enhancement 15476.
- The "P" key can now be used to swap between design view and presentation view.
  Closes enhancement 9333.
- "Fit page" is now the default zoom level when opening a file.
  Closes enhancement 13863.
- Added the ability to use the control key plus the mouse scroll wheel to zoom in and out of the page.
  Closes enhancement 15516.
- Added the ability to distribute selected items evenly horizontally or vertically either to the page or within the currently selected items.
  Closes enhancement 15509.
- Added the ability to align items to the top/bottom/left/right of the the page.
  Closes enhancement 9300.
- You can now specify an outline border for Oasys Ltd. image objects.
  Closes enhancement 15503.
- The escape key can now be used to deselect any selected objects. It is still used to quit out of fullscreen mode.
  Closes enhancement 15530.
- The total number of pages in the document is now displayed at the top of the window.
  Closes enhancement 15513.
- Added preferences to allow the user to specify the format of the default DATE and TIME variables.
  Closes enhancement 15529.
- Modified the default variable DATE so that it just shows the date rather than the date and time. A new default variable TIME has been added
  Closes enhancement 15453.
- The maximum number of pixels you can crop off an image edge has been increased from 1000 to 10000.
  Closes enhancement 15451.
- Textboxes were not copied when duplicating a page. This has been fixed.
  Fixes bug 15441.
- Added the ability to write the output of a library program to a variable.
  Closes enhancement 9031.
- Added the ability to align multiple objects together. Option are left, centre, right, top, middle or bottom.
  Closes enhancement 9300.
- Added the ability to select multiple objects on a page. Multiple objects can be dragged, cut/copied/pasted, saved/imported, generated, resized etc.
  Closes enhancements 8980, 9106, 9300.
- Added the ability to format a variable. For example if a number, how many decimal places.
  Closes enhancement 13867.
- The text on the status bar could get overwritten during generation of items. Now fixed.
  Fixes bug 14230.
- Setting the background colour of various object types via conditional formatting has been added.
  Closes enhancement 9026.

- It is now possible to set the background colour of cells in tables.
  Closes enhancement 15319.
- A note object has been added for adding notes to the design view of a report.
  Closes enhancement 13825 .

Version 9.4.2

- " Hyperlinks for HTML files are now converted to relative links.
  Fixes bug 16138.
- If you inserted a normal program into a template by selecting the program tool and dragging an area Reporter would think that the object was a library program, not a ’normal’ program.
  Fixes bug 15133.

Version 9.4

- Reporter could crash when accessing variables after using the JavaScript method Template.GetVariableValue() with a variable name that did not exist in the template.
  Fixes bug 14347.
- If a job file was selected before doing a capture for a T/HIS object REPORTER would not try to substitute DEFAULT_DIR (and other variables) in the filename. Now fixed.
  Fixes bug 14329.
- If you modified an items outline, fill or text colour or modified its line thickness or style this did not flag the template as requiring a save. This has now been fixed.
  Additionally templates which require saving are now marked with a * in the window title.
  Fixes bug 13960.
- Exiting from REPORTER using File->close and using the top right window close button now gives the same error message and options to save any modified templates. Previously the messages were different and this caused confusion to some users.
  Fixes bug 13430.
- D3PLOT objects with multiple filenames would not work if one (or more) of the filenames contained spaces. This was due to a bug in D3PLOT. Now fixed.
  Fixes bug 12409.
- When writing PowerPoint output blank table cells were given the default font size by PowerPoint. As this is very large it caused the table row to be larger.
  Fixes bug 13874.
- User defined script directories can now be defined by using the library_directory preference. This allows users to add their own library scripts if REPORTER is installed in a read only location.
  Closes enhancement 13503.
- If a library program is added it is now possible to set the font, size, style and justification in the menu.
  Additionally if you edit an existing library program this menu is now used instead of the ’normal’ program menu.
- When generating a report more feedback is now given in the status bar so you know what REPORTER is doing (e.g. running a D3PLOT object in background).
  Closes enhancement 13888.
- Report generation can now be stopped at any point by a new ’Stop’ button in the status bar.
  Closes enhancements 10708 and 11271.
- D3PLOT and T/HIS can now be run from REPORTER without any windows being mapped by either giving the -batch command line option to REPORTER or by setting the batch mode checkbox in File->Program locations.
  Additionally REPORTER can be minimised during report generation so you can use other programs.
  Closes enhancement 10709.
- HTML output has been improved for tables. Previously cell heights could be too high on Internet Explorer and additionally text that was too big for a cell was not cropped.
  Fixes bug 13846.
- Once a ’Capture’ has been done for D3PLOT or FAST-TCF objects the ’Capture’ button is changed to say ’Update capture’ as it was not clear that pressing the button again would allow you to change the existing capture rather than starting again from scratch.
  Closes enhancement 13757.
- PowerPoint output could sometimes only be done once for each Reporter session.
  Now fixed.
  Fixes bug 13873.
- Page ranges set by the user in the printer dialog were ignored and the whole report was printed. Now fixed.
  Fixes bug 13887.
- The Hyperlink dock box was not mapped correctly when a hyperlink was clicked. A similar problem occurred with the ’master page’ dock box.
  Fixes bug 13827.
- Clicking on a hyperlink that referred to a non-existant report could crash Reporter.
  Fixes bug 13836.
- PDF output for table cells was not cropped if it was too large for the cell.
  Fixes bug 13883.
- If you edited an existing FAST-TCF object that used variables somewhere in the script and you pressed capture to change the script REPORTER prompted you to try to replace text with variables in the new script but no

replacements were done. Now been fixed.
Fixes bug 13833.
- Image cropping has been added for Image, ImageFile, D3PLOT and Fast-tcf objects.
  Closes enhancement 12854.
- Text wrapping, border style, border colour and background colour have been added to the textfile object.
  Closes enhancement 8631.
- A new text colour button has been added to the Style toolbox to change the colour of text (previously the outline colour button changed the colour of text). This was necessary as the new textbox objects have fill colour, border colour and text colour.
- A new textbox object has been added to Reporter.
  Closes enhancements 9107, 7800 and 3881.

Version 9.3.1

- Visual basic output did not work on windows for text file items that had more than one line of text. Now fixed.
  Fixes bug 13165.
- Images for advanced objects in HTML output were scaled incorrectly. Now fixed.
  Fixes bug 13159.
- Reporter now shows files with extension .pptx as well as extension .ppt when writing PowerPoint files.
- Writing text objects to a PowerPoint file did not work correctly with PowerPoint 2007 (the text was written with a single letter on each line). Additionally:
  - File objects had a black background if a visual basic macro from Reporter was read into PowerPoint 2007.
  - Justification of text objects was not correct if a visual basic macro from Reporter was read into PowerPoint 2007.
  - Tables had the wrong border and background colours in PowerPoint 2007.
  - The colour of some lines could be incorrect in PowerPoint 2007.

  Now fixed.
  Fixes bugs 13022 and 13138.
- Output from writing text objects to a Powerpoint file and to a visual basic macro could be inconsistent. The textboxes produced when writing a PowerPoint file directly were not resized to fit the text, and textboxes produced from a visual basic macro would have different margins to those produced when writing a PowerPoint file directly. This is now fixed.
- Reporter would not play a d3plot command file with ’button click’ data correctly. The button click data would be stripped from the command file and the commands treated as dialogue commands. Now fixed.
  Fixes bug 13027.
- In an automatically generated table column text entries containing variables would not generate correctly (the variable would be replaced by a blank string) if the variable name was in lower case. Now fixed.
  Fixes bug 12995.
- On some platforms when generating a report, a warning message from T/HIS and D3PLOT could be passed to REPORTER in two or more chunks (it should be passed to reporter as a single string). REPORTER would mistakenly think that the second and subsequent chunks were error messages and try to alert the user that an error occured. This has now been fixed.
  Fixes bug 12738.
- If a library object failed to generate properly (e.g. if the otf filename was incorrect) then the next time that Reporter generated the report you could get ’Cannot get File data in File destructor’ errors. This has been fixed.
  Fixes bug 12629.
- When writing tables to powerpoint directly or writing a visual basic macro, the colour and width of table borders was ignored. Now fixed.
  Fixes bug 12733.
- The -maximise command line option and maximise oa_pref option did not work correctly on some screens. This has now been fixed.
  Fixes bug 12941.
- The hostname library script would fail if the hostname of the machine contained a hyphen (-).
  Fixes bug 12413.
- When drawing a polygon with the image.Polygon() function you could not define the line colour as ’none’ (it always gave a black outline). This has now been fixed.
  Fixes bug 9585.
- If you edited a normal table after generating program data in any of the cells the program output was lost during the edit. This has now been fixed.
  Fixes bug 12348.
- If you saved output to html (or vba, pdf) and the file existed you were asked twice if you wanted to overwrite it.
  Fixes bug 12428.
- Variable expressions were not correctly evaluated when used in text. Instead of the variable value being evaluated the entire string was evaluated which could sometimes mean that the expression could not be evaluated correctly. This has now been fixed.
  Fixes bug 12347.
- Powerpoint output was incorrect for several object types:
  - Bold, italic and underlined text was shown as normal text.
  - Arrowheads were not drawn on arrows.
  - Rectangles and ellipses without fill were still drawn with fill.

- Dashed and dotted lines were drawn as solid lines.
- Autotable cells could have the wrong font style and justification.

This has now been fixed.
Fixes bug 12433.

Version 9.3 (October 2008)

- When doing conditional formatting the default font for each condition is now the same as the existing font before you asked for conditions (so for example you have to change only the colour). Previously the defult font was always 10pt Courier. Closes enhancement 11906.
- If you double click on a variable in the Edit variable menu it now edits the variable. Closes enhancement 11904.
- In design mode, programs that use library scripts now have %REPORTER_HOME%/reporter_library/scripts removed from the beginning of the text that is shown on the object so it is easier to see what the program is. Fixes bug 7701.
- A library script has been added to read a reporter variables file. Closes enhancement 11902.
- Printing did not work for autotable objects. This has now been fixed. Fixes bug 11848
- The library directory for Reporter has been renamed to ’reporter_library’. Existing scripts which use ’library’ will be modified when Reporter reads the file.
- In the menu that is mapped when the user right clicks on an object, Edit and Delete were next to each other. Occasionally people pressed Delete by mistake. A space has been added to the menu either side of the Delete button to make it harder to delete the object by accident. Fixes bug 11332.
- When the dyna filetype preference was changed in Reporter it did not change the filetypes for any existing objects in the template.
  Additionally, when opening a template, if the preference was set to the Oasys Ltd. filetypes, Reporter would silently change any ’d3hsp’, ’d3thdt’ and ’d3plot’ definitions to ’%DEFAULT_JOB.otf’, ’%DEFAULT_JOB.thf’ and ’%DEFAULT_JOB.ptf’ and there was no way to undo this change.
  Now if you change the preference interactively Reporter looks to see if any filenames need updating. If they do then it asks you if you want to change them.
  Similarly, if you read a template Reporter checks and asks you if you want to change them. However, this is not done if the batch option has been set.
  Fixes bugs 9782, 10613 and 11438.
- Library scripts which retrieve data from the end otf file have been made significantly quicker. Fixes bug 9479
- It is now possible to have D3PLOT and FAST-TCF objects that do not return images to REPORTER. Fixes bugs 9028 and 9108.
- A new ’Expression’ variable type has been added that allows user to do simple maths with variables. e.g. (%THREE%+%ONE)*%THREE%/%TWO%. In fact it will evaluate the expression as a JavaScript expression so Math.sqrt(), Math.sin() etc are also available. Fixes bugs 9010, 9017 and 9111.
- After reading in a template, Reporter now shows the first page, not the last page. Fixes bug 9006.
- All dialog boxes in Reporter now have a maximise button to make them easier to resize if they need to be made bigger (e.g. if editing a FAST-TCF object). Fixes bug 8793
- Normal table objects have now been added to Reporter. Closes enhancements 7233, 7703 and 7704.
- Postscript output has been removed from Reporter for version 9.3. Use pdf output instead.
- Added File.Mkdir() method to create a directory.
- Added File.APPEND constant to enable appending to files.
- Library scripts in tables did not work if there was a space in the installation directory of Reporter. Additionally any variables that were used as arguments would not have been expanded correctly (they would get the value from the current template instead of the value from the reporter_variables file). Fixes bug 9451.
- Added pdf_image_downsample, pdf_image_downsample_resolution and pdf_image_downsample_threshold preferences to allow image downsampling when writing pdf files.
- Added use_file_vars preference to enable filenames returned from D3PLOT and T/HIS to be replaced with directory/file variables automatically if they match

Version 9.2.3 [Build 36] (21/11/2006)

- Reporter would create a corrupt pdf file if a page contained a zero size image. This has now been fixed. Fixes bug 9315
- If special characters like > and < were used in a condition name Reporter could not read the template file. Now fixed. Fixes bug 9220.
- Fixed problem with text in pdf files not printing properly on some printers. Fixes bugs 9134 and 9212.
- The output from a table can now be written to a CSV file during generation. Closes enhancement 9133.
- Reporter now gives the user the ability to stop report generation if an error occurs. Closes enhancement 9126.
- Some objects with a line colour and/or fill colour of none were not being rendered properly (black was used instead). This has now been fixed. Fixes bug 9081.
- Reporter would get the start in directory wrong for T/HIS and D3PLOT if there was a single jobfile that contained spaces. This could cause T/HIS to crash. This has now been fixed. Fixes bug 9038.
- Library scripts could not be used as table items (an error occured when they were run). This has now been fixed. Fixes bug 9024.
- It is now possible to generate a single page of a report. Closes enhancement 9011.
- Powerpoint could be left open after writing a powerpoint file. This would happen if the -exit command line argument was given after the -ppt argument. This has now been fixed. Additionally Powerpoint will now not be

closed if there is an existing presentation open in Powerpoint. Fixes bug 8998.
- The extension orp was not automatically appended when exporting a page (if the filename has no extension). It is now added if required. Additionally ps is added for postscript, pdf for Acrobat, htm for HTML (html on unix), bas for Visual basic macros, and ppt for Powerpoint. Fixes bug 8988.
- If a library page (e.g. checking page) was inserted into a template and the Oasys Ltd. filenaming scheme was used (file.thf instead of d3thdt etc.) the objects would not generate properly as they referred to d3thdt, d3hsp etc. This has now been fixed. Fixes bug 8954.
- Reporter is now more intelligent when pasting multiple copies of an item. Additionally the pasted item is now selected. Fixes bug 8861.
- On Solaris 10 it was possible what errors when generating T/HIS objects did not get logged properly. This meant that sometimes the user was not notified that an error occured. This has now been fixed. Fixes bug 8487.

Version 9.2.1 [Build 35] (26/7/2006)

- Switching between templates on HP unix machines caused Reporter to get stuck in a loop refreshing the screen until the mouse was moved out of the template. This has now been fixed
- Multiple spaces in arguments to external programs were simplified to a single space. This was incorrect and has now been fixed. Fixes bug 8857.
- Recapturing from T/HIS could fail if there were multiple models. This has now been fixed. Fixes bug 8842.
- When capturing from D3PLOT and T/HIS on Windows sometimes DEFAULT_DIR was not replaced in the filename. This occured if slashes (/ or \) did not match between the variable and filename. Now fixed. Additionally, now if DEFAULT_DIR does not match REPORTER will try to use other Directory variables to match. Fixes bugs 8314 and 8758.
- Compounded variables (i.e. variables that contained variables) did not expand correctly. Now fixed. Fixes bug 8669.
- Arguments to an external program which used variables that contained spaces would not be passed to the program correctly. Now fixed. Fixes bug 8666.
- Brackets (,),[,],{,} and slashes \,/ in arguments to an external program could cause Reporter to hang. Now fixed. Fixes bug 8665.
- Fixed bug that caused spurious pages to be created when a page was duplicated. Fixes bug 8716.

Version 9.2 [Build 34] (24/5/2006)

- Fixed bug that caused the current page number on a master page to be incorrect when printing. Fixes bug 8628.
- Fixed bug that caused corrupt pdf output if there were images on the master page. Fixes bug 8629.
- Fixed problems with missing output from running external programs
- Adding a new page while an object was selected would erroneously leave the selection handles drawn on the new page. Now fixed. Fixes bug 8530.
- Fix problem in javascript File class that caused errors in File destructor.
- Output from T/HIS and D3PLOT was not written to the logfile for Solaris 10. Now fixed.
- Errors and warnings from D3PLOT and T/HIS are now fed to REPORTER via stderr so they now correctly come through as errors and REPORTER is aware of them.
- The log window is now raised when it is mapped as previously it could get lost behind the main window.
- Hyperlink rectangle produced in pdf files for text objects with hyperlinks is now correct if the text object used variables. Fixes bug 8405.
- Objects that are not visible are now not selectable. Fixes bug 8404.

Version 9.2 Beta 4 [Build 33] (4/4/2006)

- Fix problem with centre justified text in HTML (it was not positioned correctly as the style was incorrect).
- Hyperlinks from objects other than tables containing variables now work correctly.
- Hyperlinks now open a report in presentation mode (this was broken in an earlier release).
- Output from program items with hyperlinks is now correctly written when writing a report.
- Cursor used when hovering over hyperlinks is now correct on Windows
- Replacing subsequent variables in table cell contents and hyperlinks would fail if the first variable in the text did not exist. This is now fixed.
- Fixed JavaScript compiling problems on SGI that caused crashes.

Version 9.2 Beta 3 [Build 30] (20/2/2006)

- Add unicode support for writing pdf files. Partially fixes enhancement 7799 (no ps support yet). Unicode characters can be used in text objects and table headers.
- Add ability for capturing from T/HIS to read a cvs file. As no jobfile is returned N/A is shown. Fixes bug 8151.
- D3Plot objects can now use multiple models and/or windows. When using capture new models can be opened. When you return to Reporter all of the models and windows are remembered. Fixes enhancement 7237.
- Object coordinates can now be specified by using 2 corners or by using a corner and width/height. This can be set by a preference. Fixes enhancement 7811.
- You can now search and replace strings in objects. Fixes enhancement 7820.
- Text items can now be vertically justified as well as horizontally. This should help line up output from text items

and program items. Fixes enhancement 7812.
- D3PLOT and T/HIS are now passed the '-maximise' command line argument to ensure that they are full screen.
- The FAST-TCF and T/HIS tools are now combined into one tool as people found having two tools confusing. Fixes enhancement 7818.
- Reporter now has different cursors depending on which tool is used. Fixes enhancement 7817.
- Variables can now be given a type to help manage/distinguish them.
- File and directory variables can now be browsed for. Fixes dynatrack cases 7688 and 6857.
- You can now find and loop over all the warnings and errors written to the logfile.
- If an error occurs when generating Reporter now shows a dialog box to tell the use and gives the ability to show the error. Fixes bug 7771
- Added this changelog to the help menu in Reporter.
- Added ability to create, drag etc in presentation mode. Fixes dynatrack bug 7766.
- Added 'hand' tool to presentation view which allows you to follow hyperlinks etc.
- Added a 'write Report' option in the file menu to make saving as a report easier (previously you had to do SaveAs and change filetype). Fixes enhancement 7778.
- Reporter now remembers the directory from the last file you selected and uses that as the start directory for the next file selection. Fixes enhancement 7714.
- Added powerpoint size as a page size. Fixes enhancement 7709.
- Existing bitmaps are now deleted before generating advanced objects. This is to guard against picking up old data by mistake. Fixes enhancement 7772.
- Variables now have their own menu. Fixes enhancement 7819.
- Variables are now saved by default when generating. Fixes enhancement 7687.
- Now gives an error if a save did not work because a file or directory is write protected.
- Automatically replace job names with DEFAULT_DIR and DEFAULT_JOB when capturing. Can be turned off with a preference. Fixes enhancement 7657.
- A default size is now given to an object if the user doesn't drag when creating an object. This size can be set with an oa_pref option. Fixes enhancement 7696.
- CURRENT_PAGE variable now works correctly on a master page when writing pdf, vba and ppt. Fixes bug 7892
- Colour buttons now set correctly for WindowsXP style in Colour Dialog. Fixes bug 7647
- Added conditional formatting for textfile objects. Fixes bug 7606
- Shift and Ctrl keys now constrain lines, arrows, rectangles and ellipses when dragging. Fixes bug 7733
- version.js script bug fixed. Fixes bug 7695.
- The initial text properties are now set correctly for text file items. Fixes bugs 7647 and 7605.
- LSTC/OASYS Ltd. filenaming can now be set as a preference. Fixes bug 7692 and enhancement 7630
- Images are now embedded when saving as a report. Fixes bug 7660.
- Online manual now linked to Reporter from Help menu
- Reporter now prompts you to save a template before closing if any changes have been made
- Variables can now be used in condition values
- When the mouse enters the report you now get the keyboard focus
- -log= argument now works.
- bug fix 7774. Reporter now traps template files that don't exist on the command line and skips remaining arguments but does not skip -exit or -log= so it doesn't hang
- Change name to Reporter.
- Unicode support added for text object strings (no postscript or pdf support)
- The -generate command line option now always generates the report. Previously it only generated in design mode. This meant that if you opened a report you could not generate it (as it is opened in presentation mode)
- '\' characters in filenames etc are now converted to '/' characters on unix machines.
- Change logic for multiple models in T-HIS to that Presenter passes the directory of the first model as the -start_in argument.
- Added us-ncap.js library script to plot US-NCAP graph
- Added fontAngle and fontJustify properties to javascript Image class to give more control of text rendering

Version 9.2 [Build 21] (14/11/2005)

- Added maximise preference for Presenter
- Presenter now reads the start_in and vba_directory preferences
- Presenter now picks up variables from T-HIS correctly when there are multiple analyses
- In the variables dialog the whole row is now highlighted when you select a variable instead of just the first column.
- When adding a library program Presenter now checks to see if any compulsory arguments are missing.
- When a new file is created a new page is now automatically started.
- Added more error checking to data_file_from_variables.js script (bug fix 7635)
- Added LogPrint, LogWarning and LogError methods to global javascript object
- Added File->close option (was previously under Window->close but obviously people expect it to be under the file menu! (bug fix 7637)
- If you change drawing mode when in presentation mode you are now automatically taken back to design mode (bug fix 7636)
- If you right click on an object when in any drawing mode you will change to select mode, select the item and map the popup menu (bug fix 7634).
- Added ability to reorder pages (enhancement 7571)
- Variable values and descriptions are now escaped properly when saving so special characters can be used (&, <,>

etc)
- When capturing a FAST-TCF script, if the job file is not empty it is read into T/HIS (previously it was only done if there was a script as well)
- When you edit a text item a crosshair is now shown at the point the text is justified to
- If you paste an item on the same page it is now offset from the original by the nudge distance so it is obvious to the user that a new item has been pasted. If you paste into a different page or template it will be placed in the original position
- Right clicking on the page when you do not have a selected item now gives you the option to paste an item at that location (if you have copied or cut an item previously)
- Table items can now be written directly to PowerPoint
- Table items can now be written to vba
- Add -ppt command line option to write powerpoint files
- Subroutines in visual basic macros written by Presenter are now automatically split if necessary to keep them below the 64k limit for VBA (previously there was one Subroutine per page)
- If a table with overflow pages is read from a report, the overflow pages are now correctly displayed. Previously you would have to regenerate or edit the table.
- Added support for multiple models for T-HIS and Fasttcf scripts
- PRESENTER_DEFAULT_DIR is now set to the user home directory instead of the temp directory when starting. Setting it to the temp directory caused lots of problems (e.g. the next time you start Presenter that directory probably won't exist!)
- New library script added to create D3Plot data files from csv file
- Bug fix. When dragging new items they were sometimes not drawn properly (Presenter thought that they were off the screen when they were not)
- Dragging a new item is now double buffered so you don't get flicker
- New library script added to create D3Plot data files from variables file
- Presenter now tries to preserve variables in FAST-TCF scripts when the user uses the capture feature to update the script.
- If user does not type extension when saving file '.opt' is now automatically added to the filename.
- Added Ctrl+V shortcut for Paste item
- Bug fix. When you save a template using SaveAs the template name is now updated after the save to the new name
- Bug fix. When a report was generated the template could lose the keyboard focus so PgUp, PgDown etc did not work properly.
- Bug fix. Presenter crashed when double clicking on page if in line, arrow mode etc
- Add ability to load and save fasttcf scripts from editing panel
- Added next page and previous page to Page menu
- Added window menu with window list, tile, cascade etc
- When a file is opened or a new file is created it now appears maximised instead of a window
- Fixed bugs in page setup dialog (not initialised properly for some page sizes and orientations)
- Fixed bugs when writing advanced item images to vba and ppt. They were not sized correctly
- Fixed bug that caused Presenter to crash on windows when paging up/down and selecting items
- Changed comments.js script so that newlines are added correctly.
- Revise and fix javascript destrructor and garbage collection problems
- Add javascript method Close to template object
- Add ability to include debug information in logfile from D3Plot and T/HIS
- Bug fix 7218. Printing advanced items positioned them incorrectly
- Add Star method to Image class
- Add ability to change linecap and linejoin styles in Image class
- Added Polygon, Polyline and Fill methods to Image class
- T/HIS is now called with display=X instead of display=batch so that FAST-TCF works correctly
- Bug fix 6841. When changing the visibility of items by using the checkboxes in the view menu the template did not update immediately. It now does.
- Bug fix 6948. Presenter could crash when inserting an image if it was close to the edge of the page. Now fixed.
- Bug fix 6950. If a keyword file/otf file did not have a title the scripts to return the title returned an empty string. Some people thought that the script was not working. If there is no title the scripts now return 'no title'
- Bug fix 6953. Scripts containing errors caused Presenter to crash on linux.
- Bug fix 6954. Insert Variable dialog box was being mapped with the 'Save variables' buttons from the File->variables dialog box. Now removed. Additionally, I have changed the dialog caption to something more sensible.
- Bug fix 6957. When duplicating a page image items did not get duplicated.
- Builds now automatically add the date compiled (which is shown in the help about dialog box)
- Bug fix. total_mass.js did not work. Now fixed.
- Add overflow pages for automatically generated tables which have too many rows to fit on one page (in the area allocated to the table) Currently works for drawing, printing, postscript, html and pdf
- Add direct PowerPoint output for windows version
- Write JavaScript API documentation
- Bug fix 6655. Scripts could run very slowly on Windows machines but very quickly on HP workstations. This was because the script i/o was written using the C++ standard library. It has been rewritten in C and is now significantly faster.
- The variable PRESENTER_DEFAULT_DIR is now initially set to the same value as PRESENTER_TEMP when creating a new template. This is so that if you capture from D3Plot or T/HIS the images you create are put

in a sensible location until you change PRESENTER_DEFAULT_DIR to whatever value you want.
- FlexLM licensing has now been added to Presenter. The dll lmgr9a.dll must be given out and put in the same directory as the executable for windows.
- You can now change the script used in T/HIS when capturing. If you press 'Capture...' for a second time. T/HIS will replay the FAST-TCF script and you can then update as required an resave.
- Enhancement 6508. You can now edit the command file used in D3Plot when capturing. Additionally you can now change the settings that D3Plot creates. If you press 'Capture...' again D3Plot will now replay the settings and properties file and you can then update as required and resave.
- Bug fix 6688. Right clicking on an object when in presentation mode and anything other than select mode caused Presenter to crash. This has been fixed.
- Bug fix 6654. When capturing from D3Plot, if the image file was longer than 80 characters, Presenter would not correctly write the command file. This has now been fixed.
- Bug fix 6653. If a library javascript file was missing Presenter could crash. Presenter will now write an error to the logfile window
- Comment lines in oa_pref files are now correctly skipped
- Added this ChangeLog
- Initial internal releases of Reporter.

Version 9.0

Build	Date	Description
0 - 0.9		Initial internal releases of REPORTER
1.0	November 2003	First release

Text conventions used in this manual

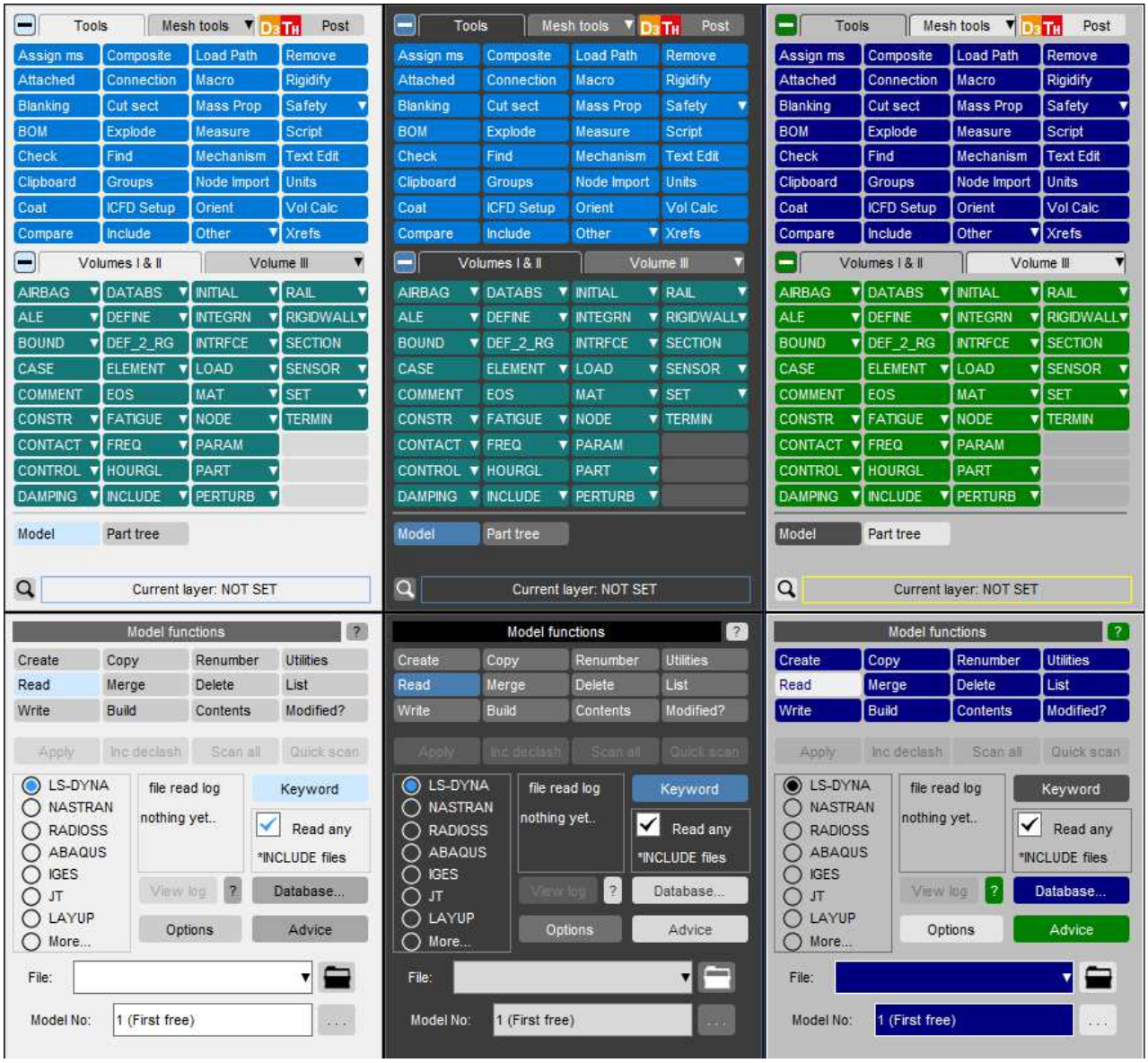
Typefaces

Four different typfaces 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.
`Operator type`	This is used to show what you must type
Button text	This is used for screen menu buttons and headings

Themes for the Graphical User Interface

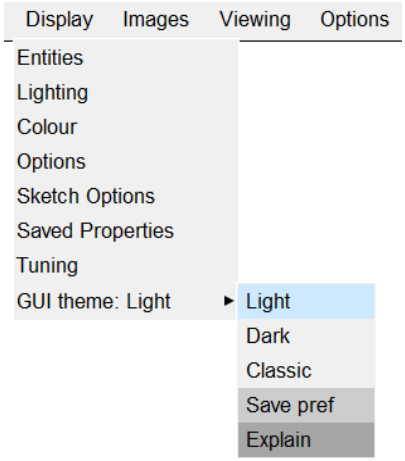
In addition to our Classic GUI theme, beginning in Oasys Suite 17.0, users can select either a Light or Dark theme. Both of these provide a more modern look and feel for the software, as well as offering different colour and contrast options for comfort and accessibility.

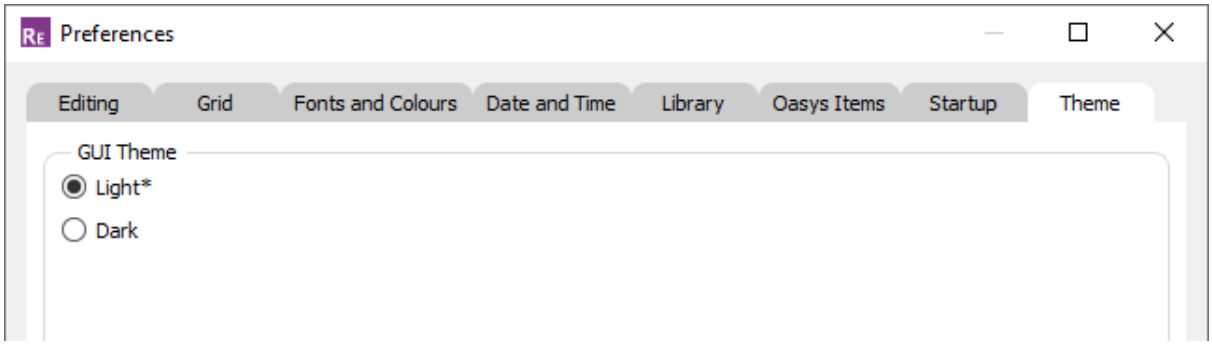


Setting the theme

The default software theme in Oasys Suite19.0is Light. This can be changed from the Oasys SHELL by choosing from the **Themes** pop-up. This automatically saves the selected theme as your preference for all programs.

The theme can also be set for individual programs from the **Display** menu in PRIMER, D3PLOT and T/HIS or the **Preferences** menu (**File**->**Preferences...**) in REPORTER. This choice is not automatically retained after exiting the program, so you must select a theme, then select **Save pref** to ensure a theme is used for all future sessions.





1. Setting up and running REPORTER

1.1 Setting up REPORTER

1.1.1 Prerequisites

Oasys Ltd LS-DYNA Environment software

You should already have the standard Oasys Ltd LS-DYNA Environment software T/HIS (including FAST-TCF) and D3PLOT installed, and have licenses for the software.
The folders that the Oasys Ltd LS-DYNA Environment software is installed in must not have any special characters in folder names (e.g. &, !, ~, ’, "). Just use letters, numbers spaces and underscores for folder names.
e.g. the following example is invalid: C:\Program Files\Ove Arup & Partners\arup19
this is valid: C:\Program Files\Ove Arup\arup19

1.1.2 REPORTER installation

For more details, refer to the installation guide (copies available at https://www.oasys-software.com/dyna/downloads/oasys-suite).

Licensing

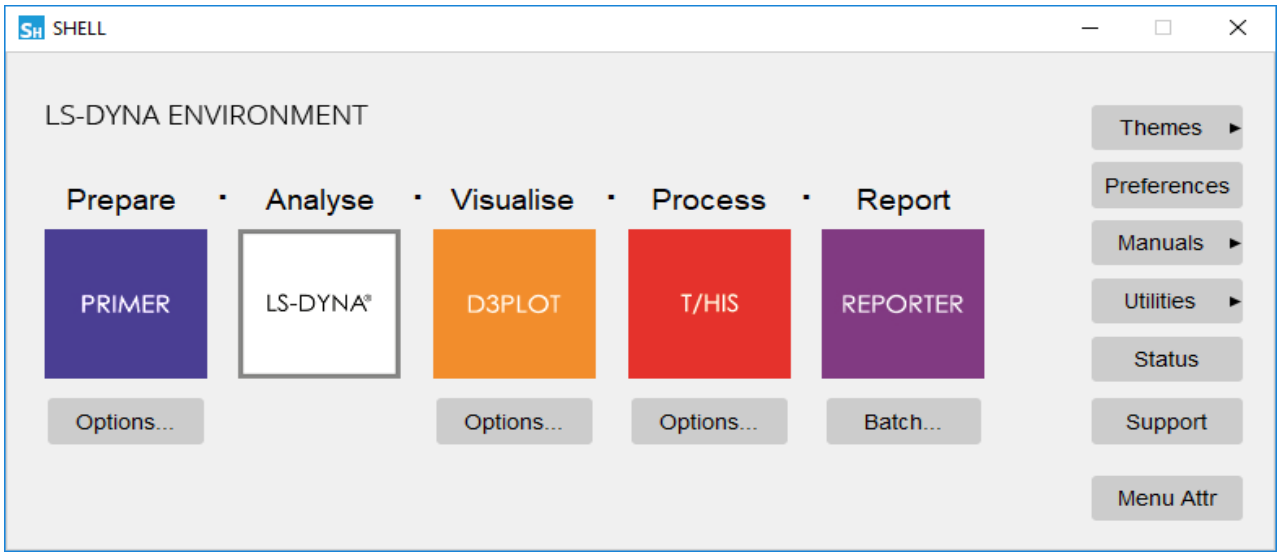
REPORTER uses LMX licensing. For REPORTER to run you must have a valid license for REPORTER or alternatively a license for D3PLOT, PRIMER or T/HIS.

Troubleshooting

If REPORTER does not run then check the following.
1. Do you have a license to run REPORTER? If not contact Oasys Ltd.
2. Do you have D3PLOT and T/HIS installed?
3. Do you have licenses for D3PLOT and T/HIS?

1.2 Running REPORTER

REPORTER is run by selecting the **REPORTER** button menu of the Oasys Ltd shell.



Alternatively, you can right click on the button to give starting options for REPORTER.

1.3. A one-minute introduction to REPORTER

REPORTER is designed to help you automate your LS-DYNA analysis post-processing. The idea is that you create a template which contains the intructions or 'recipe' for how to process an analysis. When you run REPORTER on some analysis results, it takes this template, applies it to the analysis and creates a report which you can save as PowerPoint, PDF or HTML.

For example, you may wish to run a set of standard checks on an analysis after it has run – to check that the analysis teminated normally, that there was not too much added mass, that the energy balance is acceptable, etc. You could create a checking template in REPORTER and then this would be applied to each analysis you want to check.

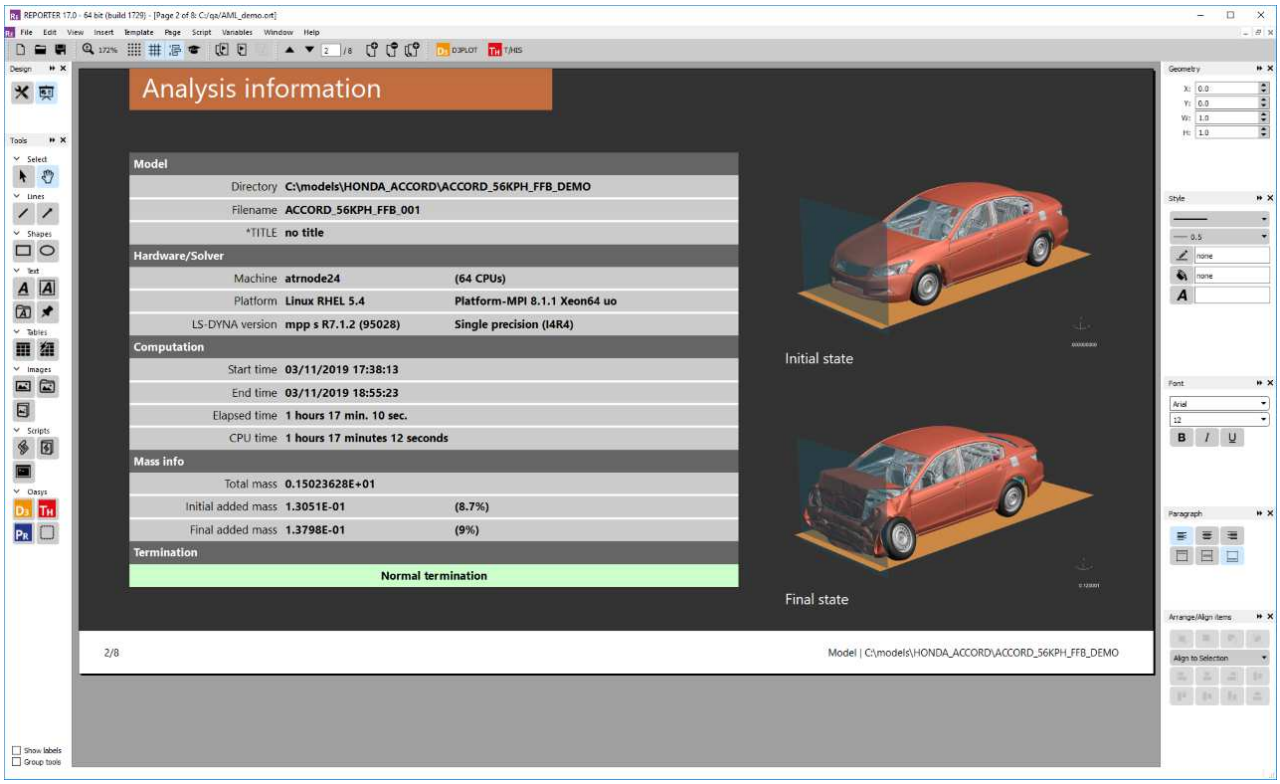
A summary of the steps required to make a template is:
1. Start REPORTER. See Running REPORTER for more details.
2. Create a template. See Creating a new template for more details.
3. Create pages (and/or a master page) if required. See Inserting and editing pages for more details.
4. Add objects on to pages. These can be simple things such as lines, text etc or advanced things like D3PLOT or T/HIS objects. See Inserting and editing simple objects and Advanced objects for more details.
5. Use variables to make the template generic. See Working with Variables for more details.
6. Save the template. See Saving a template for more details.

Once you have created a template you can apply it to analyses as many times as you want.
1. Start REPORTER . See Running REPORTER for more details.
2. Open the template. See Opening a template for more details.
3. Set the current analysis variable(s). See User defined variables for more details.
4. Generate the report. See Generating reports for more details.
5. Create output such as report, PowerPoint, HTML, PDF etc. See Outputting a generated report for more details.

2. Menu Layout

2.1 Basic menu layout

A typical REPORTER session will look like this:



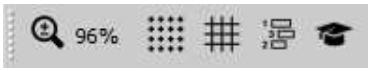
Within this main window there are a number of sections

- "Menu Bar" Access to the main pull down menus.
- "File toolbar" toolbar for opening, saving, and creating report template.
- "View toolbar" toolbar for changing the view.
- "Design" toolbar to switch between the presentation and design view.
- "Style" toolbar to modify the line type, colour, etc of objects in the report.
- "Tools" toolbar for creating and editing shapes and advanced objects.
- "Main Report Area" Main working area.

File toolbar

The file toolbar gives a quick way to create a new template, open a template or save a template. See chapter 3 for more details.

View toolbar



The view toolbar gives a quick way of zooming in and out of the template using the magnifing glass button. This is the same as using the **Zoom** submenu from the **View** menu. There are also 4 further buttons which control (from left to right respectively):

- the grid visibility,
- the snap option,
- the visibility of the item generation order, and,
- the page master view.

Generate toolbar

The generate toolbar includes 3 buttons which can be used to generate the entire report, the current page, or the currently selected items. These options are also available in various other places (e.g. the File and Template menus, the Page menu, and the right-click context menu for certain items).

Page toolbar

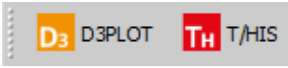
The page toolbar provides arrow buttons to navigate up or down a page as well as an input box to jump to a specific page. The 3 remaining buttons enable addition, deletion, and duplication of pages. See Chapter 4 for more details.

Animation toolbar

Starting with 18.0, the D3PLOT, Image, and Image File Item types now support animation. The animation toolbar provides buttons for controlling the playback of all animated Items on the current page. From left to right, these buttons have the following functions:

- Restart - pause all animations and go back to the first frame.
- Step back - pause all animations and go back one frame.
- Play/Pause - play or pause all animations.
- Step forward - pause all animations and go forward one frame.
- End - pause all animations and skip to the final frame.
- Speed - adjust the speed at which animations are played (as a factor of their base frames per second).

Oasys link toolbar



The Oasys link toolbar provides buttons for opening linked instances of D3PLOT and T/HIS.

Design toolbar

The two buttons on the design toolbar buttons allow you to swap between the "design" view (wrench and screwdriver icon) and the "presentation" view (easel icon) . See chapter8 for more details. By default the Design toolbar is docked on the left hand side. However you can drag it and make it a floating menu if you wish. The "p" keyboard shortcut can be used to toggle between "design" view and "presentation" view.

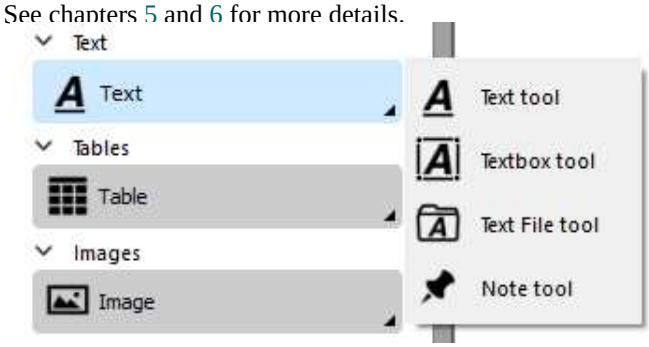
Tools toolbar

The Tools toolbar on the left contains the various objects which you can place on the page. These may be simple objects such as lines, rectangles, text etc or more complicated objects such as a D3PLOT object or a library program. All of these items are also accessible from the **Insert** menu.

The level of detail shown alongside the Tool icons can be controlled using the "Show labels" and "Group tools" checkboxes. Here we have "Show labels" ticked so that each icon is given accompanying text. This is the same text that is shown when letting the mouse hover over the button for a couple of seconds. Ticking the "Group tools" checkboxes collapses each group (e.g. Lines, Shapes etc.) into a single button. Clicking and holding this button (or clicking on the small triangle in the corner) then presents a popup menu with all the available Items in that group (see image below for an example using the Text group).

By default the Tools toolbar is docked on the left hand side. However you can drag it and make it a floating menu if you wish.

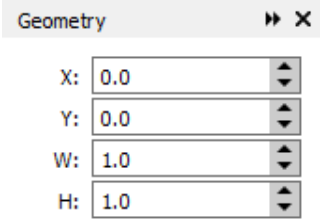
See chapters 5 and 6 for more details.



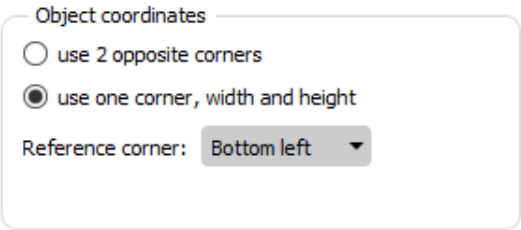
Editing toolbars

On the right of the page, various toolbars are available for quick editing of items. Changes made here affect all currently selected items. All of these toolbars are undockable and can be rearranged at will (or removed entirely by pressing the X button). Removed toolbars can be replaced using **View**...**Toolbars** .

Geometry



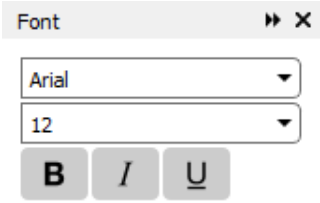
The Geometry toolbar provides a means of controlling the precise size and location of Items on a page. The options displayed here correspond to the chosen‘’Object coordinates’ in the Editing tab of the Preferences window (see below). When‘’Use one corner, width, and height’ is selected, the X and Y values in the Geometry toolbar give coordinates for the chosen Reference Corner of the selected item while W and H give its width and height. When ‘Use 2 opposite corners’ is selected, X, Y, W, H are replaced by X1, Y1, X2, Y2 respectively.



Style

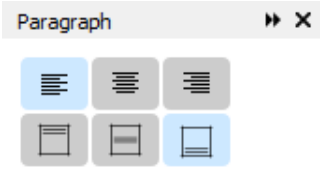
The Style toolbar can be used to change the line style, line width, line colour, fill colour and text colour for shapes. See Section 5 for more details.

Font



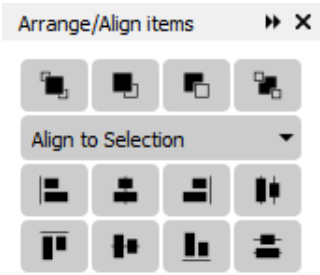
The Font toolbar can be used to change the font, font size, and style (bold, italic, underline). This can be used, for example, to quickly change the font settings for all entries in a table.

Paragraph



The Paragraph toolbar can be used to change the horizontal and vertical alignment of text within items. This affects Items in the Text and Tables item groups.

Arrange/Align items



The Arrange/Align toolbar can be used to change the positioning of Items relative to one another. The top row of buttons alter how Items are stacked on top of one another (e.g. for positioning Text or a Note on top of another Item). The drop down button on the second row has options ’Align to Selection’ or ’Align to Page’ which determines how the buttons on the bottom two rows are implemented. The bottom two rows feature the alignment buttons. With ’Align to Selection’ in use, pressing the ’Align left’ button will align all selected items to the leftmost item. With ’Align to Page’ selected, pressing the ’Align left’ butotn will align all selected items to the left of the page. Functionality of the other buttons is provided in tooltips, accesed by hovering over the button for a few seconds with the cursor.

2.2 Mouse and keyboard usage for the screen-menu interface

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

- 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.

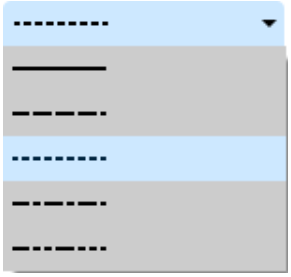
The primitive "widgets" in the menu interface are used as follows:

Buttons

Screen buttons are depressed and highlighted by clicking on them. Some buttons remain set when they have been selected and will continue to appear depressed and highlighted.

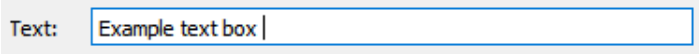
Buttons may be set by REPORTER itself, for example the cursor arrow button on the right, to indicate that this option is in force. They may also be greyed out, to indicate that the option is not currently available (e.g. the hand button on the right).

"Popup" window invocation: Some buttons when selected will invoke a "popup" window, from which a selection can be made. The popup is invoked by clicking on the triangle.



Text boxes

To enter text in a text box: first make it "live" by clicking on it then type in text into the screen that appears. You can use the left and right arrow keys for line editing within a box, text entry takes place after the current cursor position. The cursor is shown as a flashing vertical bar.

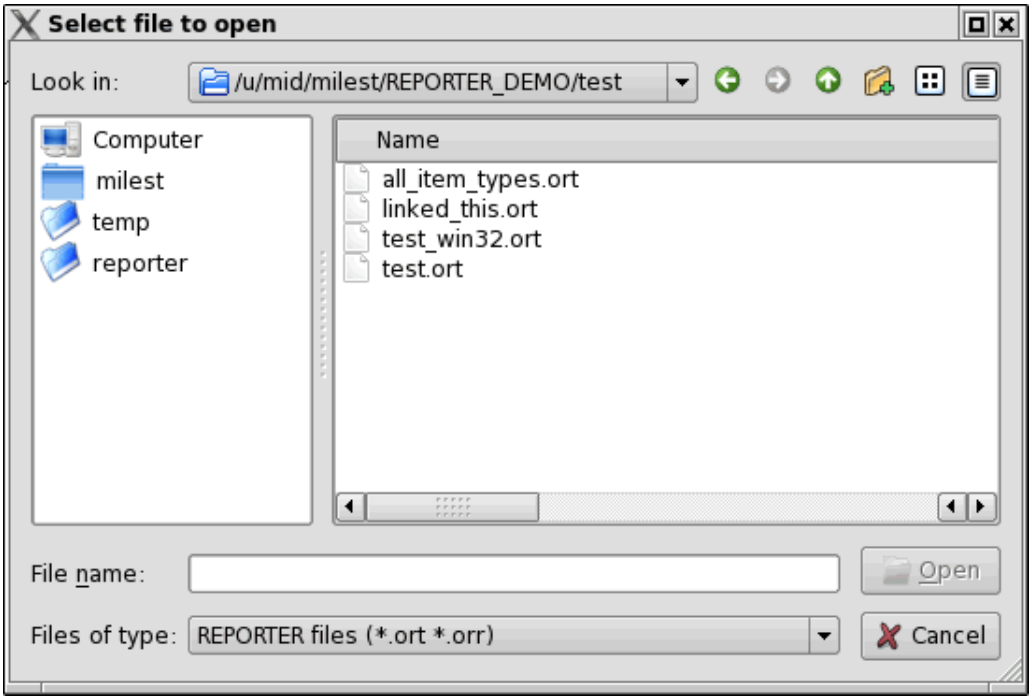


Right clicking the mouse button in a text box maps the menu on the right which allows you to copy and paste text from the clipboard and (where applicable) insert a variable (see chapter9).

2.3 Using the "file filter" boxes.

Wherever REPORTER 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 **Choose** button, which may be used to invoke a basic file filter box. The appearance of this is operating system dependent.

Basic UNIX file filter box



The files can be filtered according to file types by using the **File type** popup, in this case the pathname is **`/u/mid/milest/REPORTER_DEMO/test/`** and the pattern is `*.ort` (REPORTER template) and **`*.orr`** (REPORTER report).

The main window show a list of the directories within the present one and a list of files that match the filter selection. Files or directories can be selected by double-clicking on them.

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.

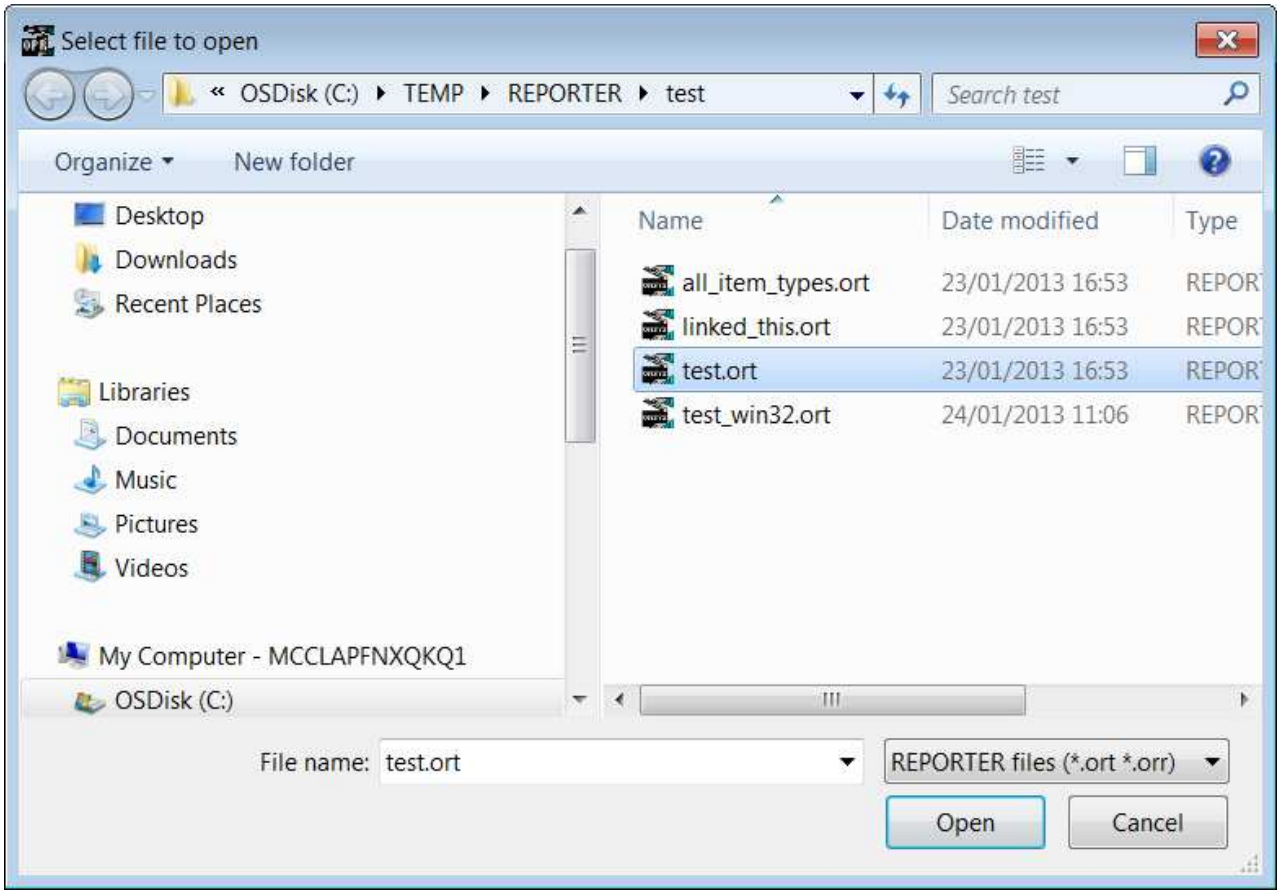
The **Open** button closes the file filter box and opens the selected file

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

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`**. 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.

Basic"Windows" file filter box



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

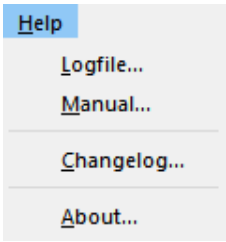
To open files that do not have the **`(*.orr)`** extension you will need to select **All files (*.*)** from the **Files of type** pull-down menu.

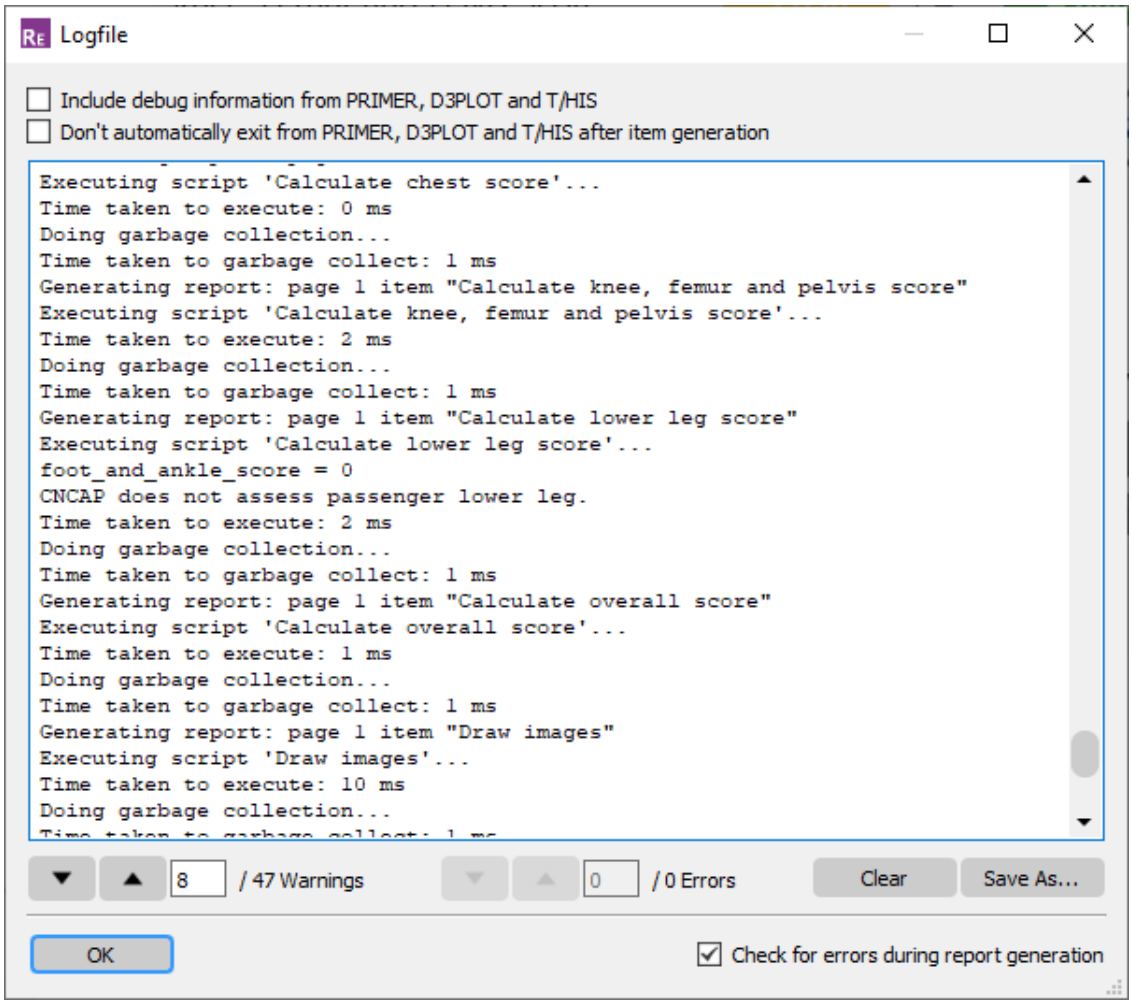
2.4 Log file

REPORTER creates a log file as it runs. This log file shows how REPORTER is trying to run programs, how it is creating images, and so on.

If any errors or warnings are encountered, they will be written to the log file. This helps you solve any problems.

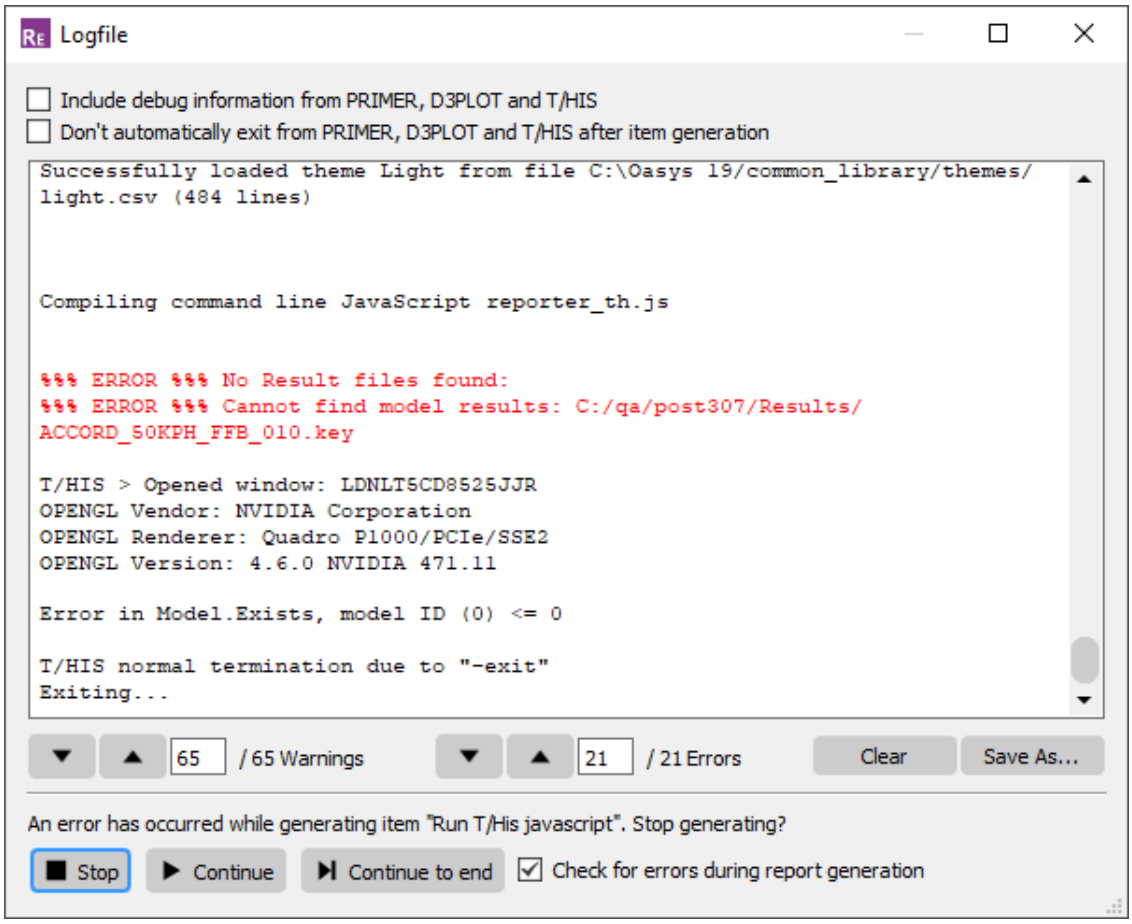
The log file is accessible from **Logfile** in the **Help** menu. A typical log file window is shown below.





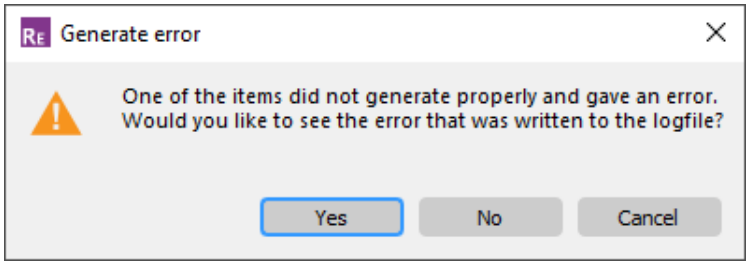
Clicking **Clear** will empty the log file contents. You can save the contents of the log to a text file using the **Save As...** option.

When generating a report (or page, or selected items), if **Check for errors during report generation** is checked (the default), then if an error is encountered, the Logfile window will appear. This gives you the opportunity to examine the error before choosing whether to **Stop**, **Continue**, or **Continue to end**:



You can navigate up and down through the errors and warnings in the log file using the controls provided. If you choose **Stop**, the report generation will stop, allowing you to fix the error immediately. If you choose **Continue**, generation will continue, and the Logfile window will reappear if another error occurs. If you choose **Continue to end**, generation will continue to the end without showing you the Logfile window again if further errors occur.

You can prevent the Logfile window from appearing when an error occurs by unchecking **Check for errors during report generation**. In this case, you will still be asked at the end of generation if you would like to inspect the log file:



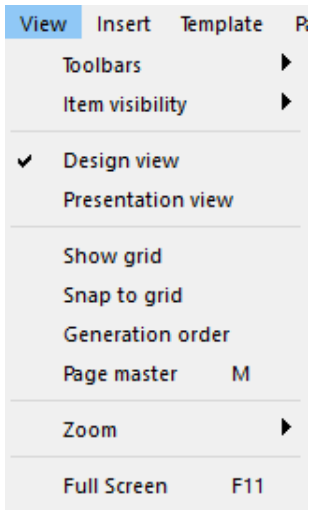
The Logfile window also has some additional options for PRIMER, D3PLOT, and T/HIS items. Selecting **Include debug information from PRIMER, D3PLOT, and T/HIS** will request extra debug messages to be sent back to the REPORTER log file during generation. This can help debug problematic items. Selecting **Don't automatically exit from PRIMER, D3PLOT, and T/HIS after item generation** will mean that those programs will remain open, allowing you to check what has happened before manually exiting them to proceed with the rest of the report generation.

The following options are automatically saved to your REPORTER preferences upon selection:

- **Include debug information from PRIMER, D3PLOT, and T/HIS**
- **Don't automatically exit from PRIMER, D3PLOT, and T/HIS after item generation**
- **Check for errors during report generation**

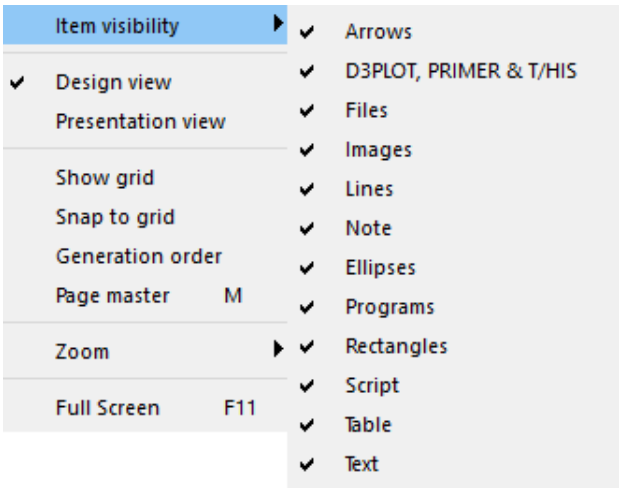
2.5 View Controls

What is and isn't displayed on the screen and how far zoomed in or out the page is can be controlled from the **View** menu



2.5.1 Item visibility options

What of type of Items are visible on screen can be controlled by selecting or deselecting the various options in the **View -> Item visibility** menu.



2.5.2 Design/Presentation view

The Design view and Presentation view checkboxes allow you to swap between design and presentation view. See chapter8 for more details.

2.5.3 Assorted options

The 'show grid', 'Snap to grid', 'Generation order', and 'Page master' options are the same as those offered in the View toolbar. That is, they toggle the:

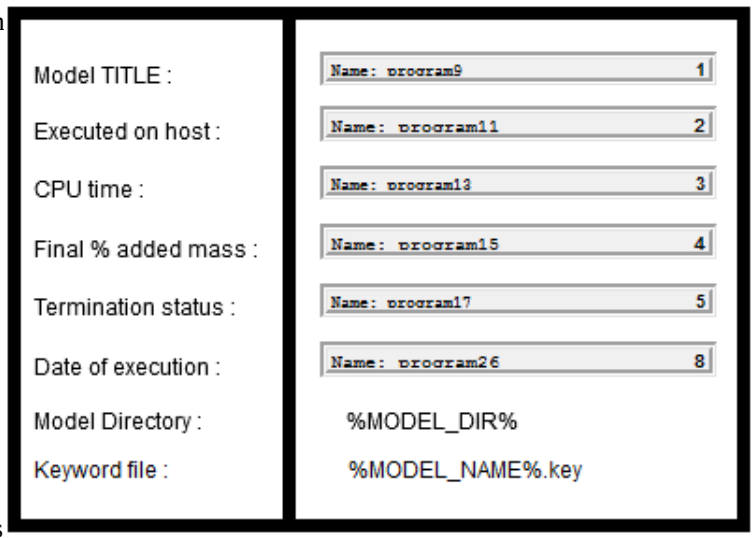
- grid visibility,
- snap option,
- visibility of the item generation order, and,
- page master view.

The effect of toggling the visibility of the item generation order is discussed inmore detail in the following section.

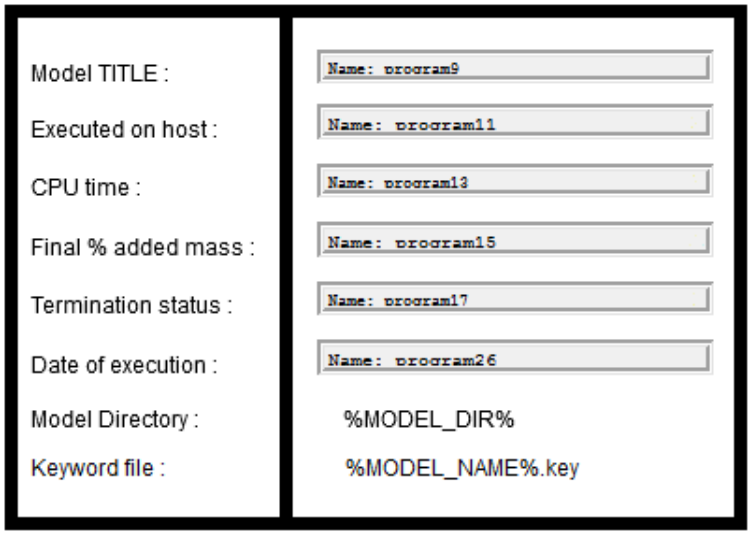
2.5.4 Generation order

The Generation order checkbox allows you to turn off whether the order that objects will be generated in is shown. The order is important if you are using variables to make sure that variables are not used before they are defined. To help with this REPORTER can show the order that the objects are generated in.

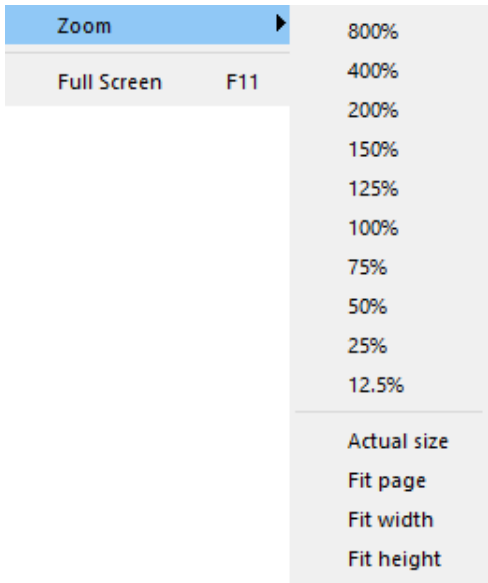
When the generation order button is turned on REPORTER shows a number next to each item that will be generated. The number is the order that the items will be generated on this page. In the image on the right you can see that the first 5 library programs in the table are generated one after another but the last one is generated later on (8th on the page). Showing the numbers helps to identify problems with objects being generated in the wrong order (e.g. perhaps the last library program should have been generated 6th on the page instead of 8th). See chapter8 for more details on generation order.



When the generation order button is turned off the numbers are not shown. The numbers are only shown in the design view. They are not shown in any output generated from REPORTER.



2.5.5 Zoom



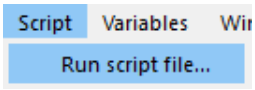
Clicking on the **Zoom** option in the **View** menu will bring up the **Zoom** menu.

- **25% 150% etc** - will zoom in or out relative to the standardised size at 100%
- **Actual size** - will resize the page to the actual size that the work is (100%)
- **Fit page** - will scale the page so that it fits into the window
- **Fit width** - will scale the page so that the width of the page will fit the screen
- **Fit height** - will scale the page so that the height of the page will fit the screen

2.5.6 Full screen view

The **Full screen** option in the **View** menu will enlarge the "Main report area" of the REPORTER window to fill the whole of the screen. You can return to the normal REPORTER window by pressing the `ESC` key.

2.6 Running a script file



To run a javascript script in REPORTER use the **Script**->**Run script file...** function. This is equivalent to running a script from the command line or inserting a script object onto a page. For more details on scripting see the scripting chapter.

2.7 Preferences

Preferences for REPORTER can set from **File**->**Preferences...** . Any options set in this menu will affect only this session of REPORTER unless the 'Save Preferences' button is pressed, in which case the choices made will be saved to the oa_pref file in the user's HOME directory and loaded for all future REPORTER sessions.

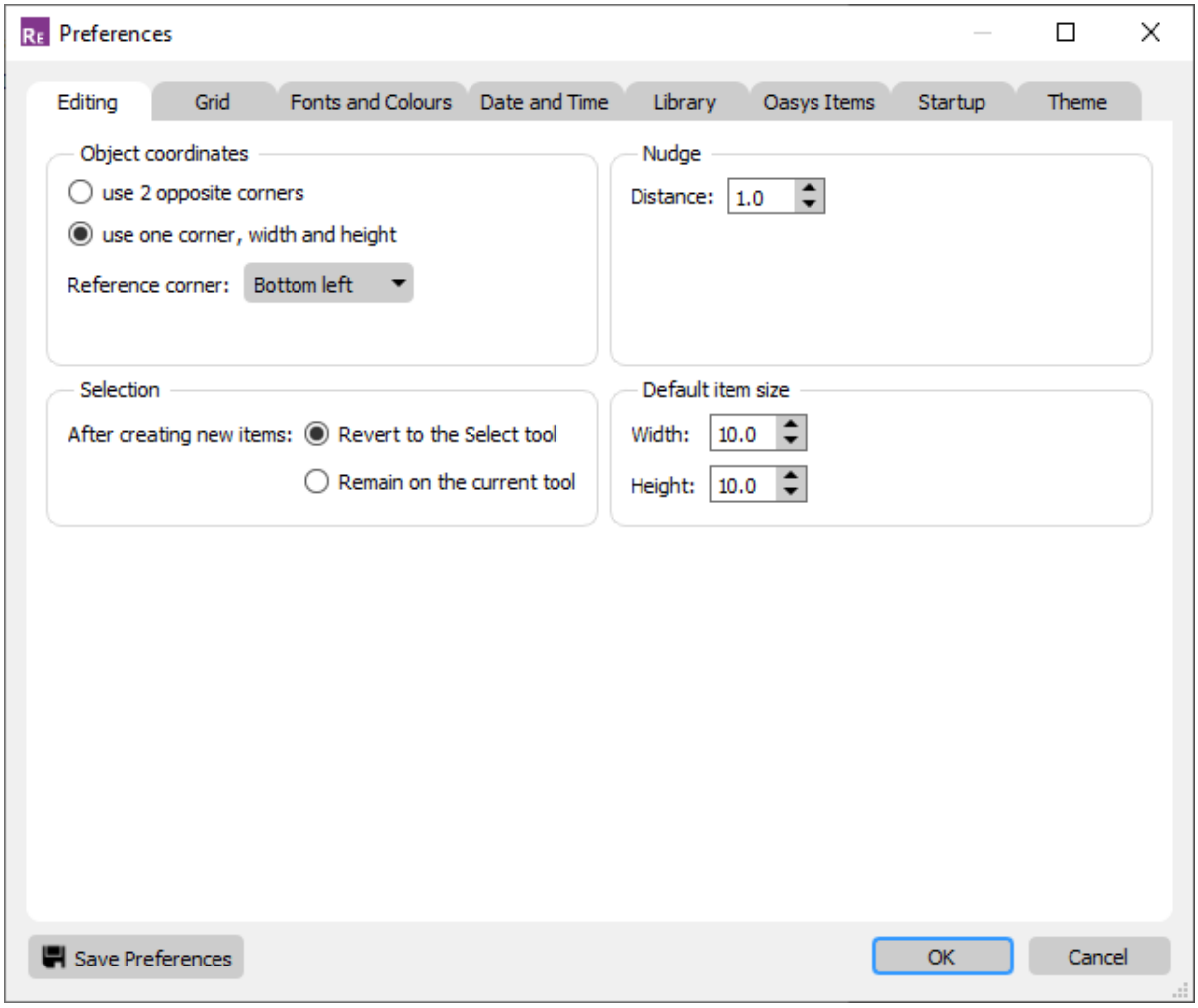
2.7.1 Editing

When creating or editing objects in REPORTER that occupy a rectangular area on the page the position and size of the object can be given by 2 different methods.

1. By giving the coordinates of 2 opposite corners of the rectangle.
2. By giving the coordinates of one corner and the width and height of the object. The default is to use the bottom left corner.

The nudge distance is the amount that a selected item will be moved when the arrow keys are used. Note that if you have snap active (see Preferences - Grid) this may give unexpected results (e.g. a larger nudge than expected). If snap is active and nudge distance is smaller than snap size, nudge will automatically use the snap size as the nudge distance to ensure that nudging is always possible.

The Selection options are fairly self-explanatory. After creating a new Item, the default behaviour is now to revert to the Select tool so that the user can quickly jump to editing this Item with a double-click (or move the Item around the page etc.). If you would rather remain on the current tool (remembering that the Select tool can always quickly be accessed with the 's' key), then this option can be changed here.

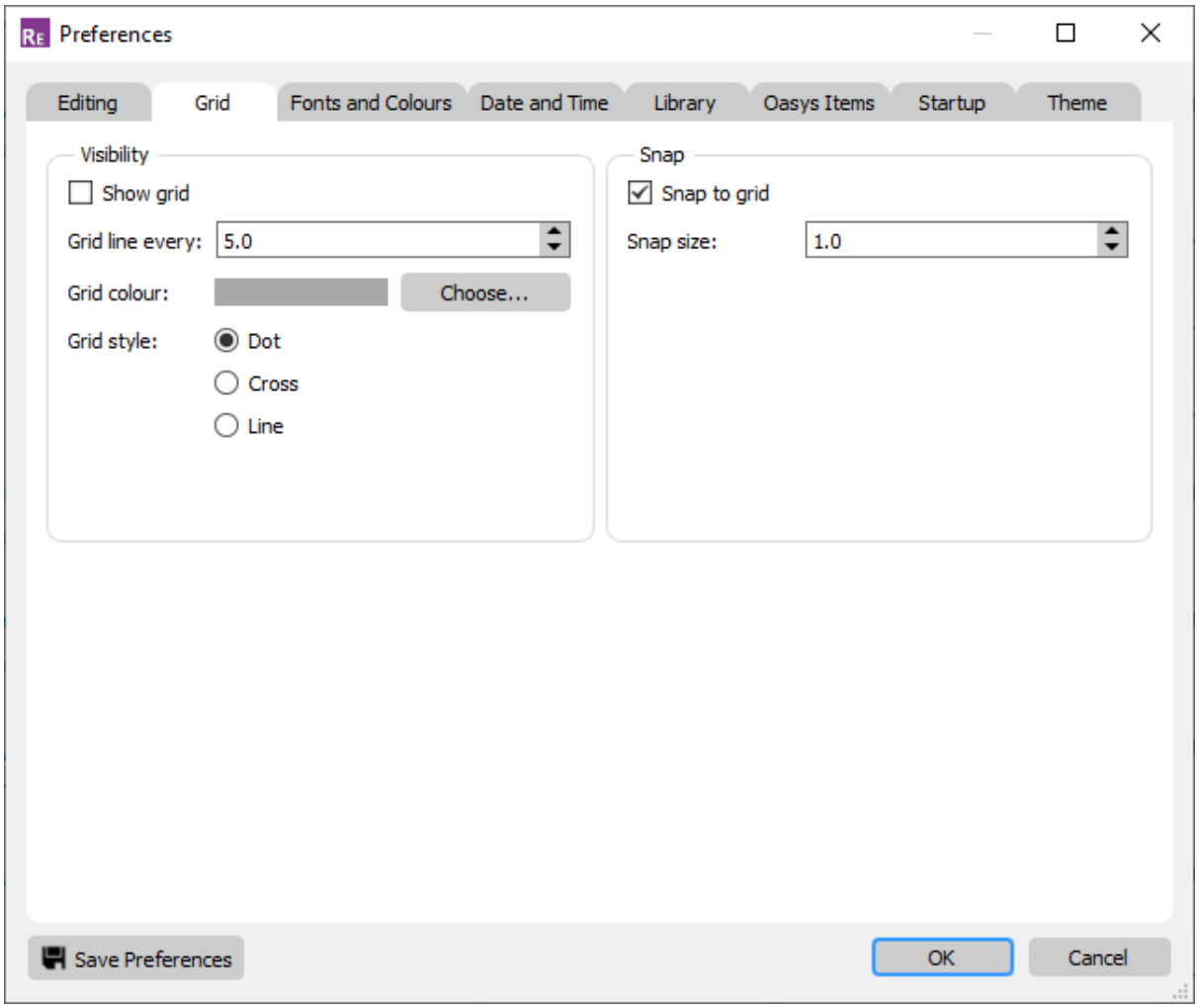


2.7.2 Grid

These options are for helping layout Items on the page.

The colour, style and size of the grid drawn on the page can be altered with these preferences. Note that the grid size does not have to be the same as the snap size.

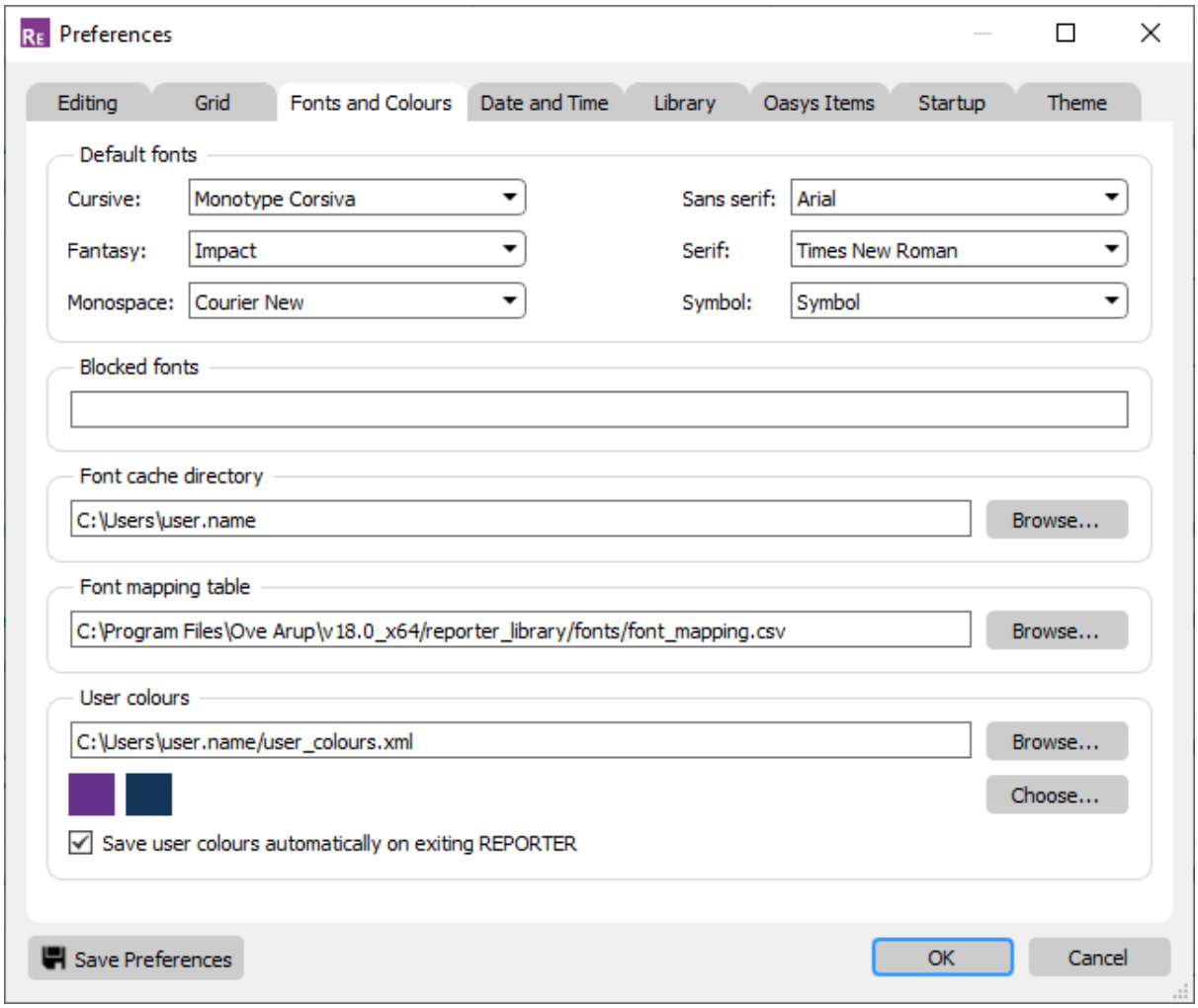
Snap will make the reference corner coordinates round to the snap size. e.g. in the image below snap is set to 1.0mm, so item coordinates will be rounded to the nearest mm.



2.7.3 Fonts and Colours

These preferences control the font cache and font mapping, including the default fonts used when no suitable alternative can be found. For more information on how REPORTER supports fonts, see Section9 .

You can also control which XML file is used to store your user colours, and whether or not user colours are saved automatically on exiting REPORTER.

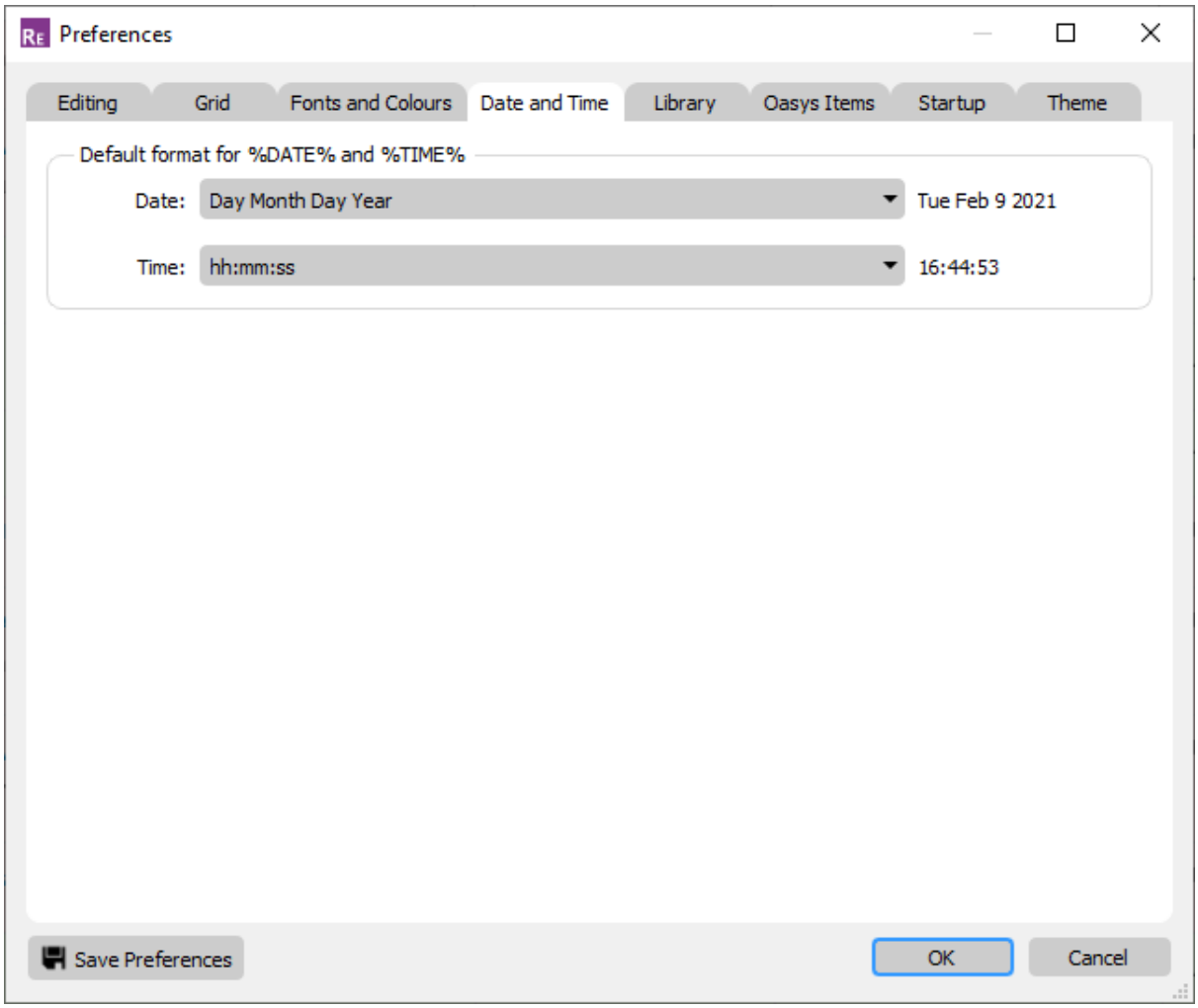


2.7.4 Date and Time

These preferences set the default formatting for the %DATE% and %TIME% variables. The available options are:

- %DATE% - Day Month Day Year, dd/MM/yyyy, MM/dd/yyyy, yyyy/MM/dd.
- %TIME% - hh:mm:ss, hh:mm:ss A, hh:mm, hh:mm A.

For more information about what these formatting options mean (and for further formatting choices), see Section9.



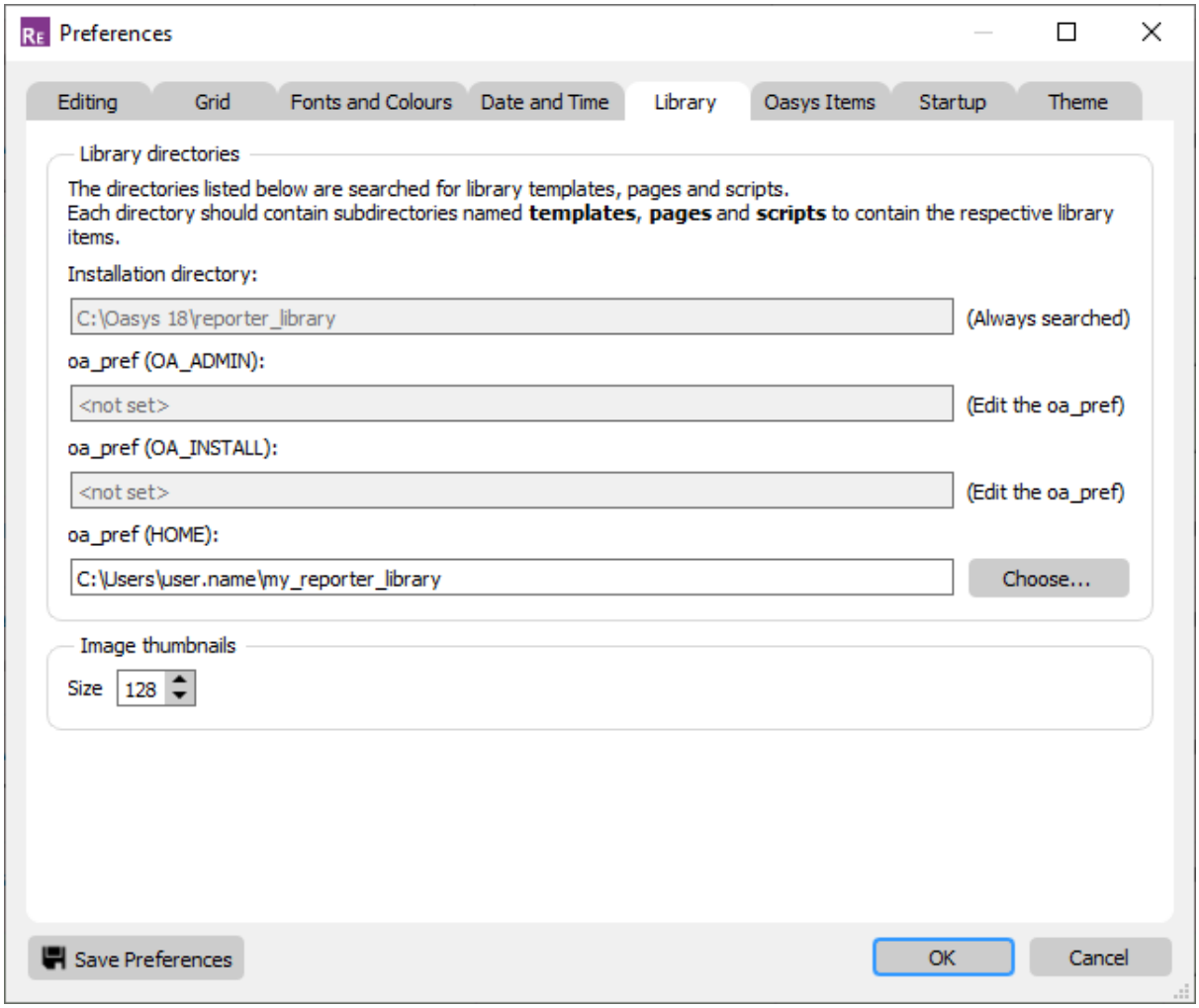
2.7.5 Library

By default REPORTER looks for library items in a subdirectory `reporter_library` in the directory where REPORTER is installed. For detail on how to add extra library items (e.g. templates, pages, images, and scripts) see Appendix B.

Due to file permissions, it may not always be possible to add library items to the `reporter_library` directory. For this reason it is possible to specify other directories for REPORTER to use in addition to the default directory. This can be done using the `library_directory` oa_pref option. This option can be set once for each oa_pref file: one in OA_ADMIN, one in OA_INSTALL, and one in HOME. REPORTER will treat any directory given by `library_directory` as a user defined library directory. The oa_pref file in HOME is likely the easiest to access, and is the same file to which preferences are usually saved when the 'Save Preferences' button is pressed.

In order to be read correctly, any user-defined library directory should contain subdirectories named **templates**, **pages**, and **scripts** containing their respective library items. For example, if `library_directory` is set to `/home/user/reporter_library` then you should put your scripts in `/home/user/reporter_library/scripts`.

Finally, the 'Size' spinbox in the 'Image thumbnails' section controls the size of thumbnails that are drawn for library images.



2.7.6 Oasys Items

The filename convention preference determines how LS-DYNA filenames are referred to by REPORTER in library scripts etc. If you are using the Oasys Ltd SHELL then you should use Arup naming.

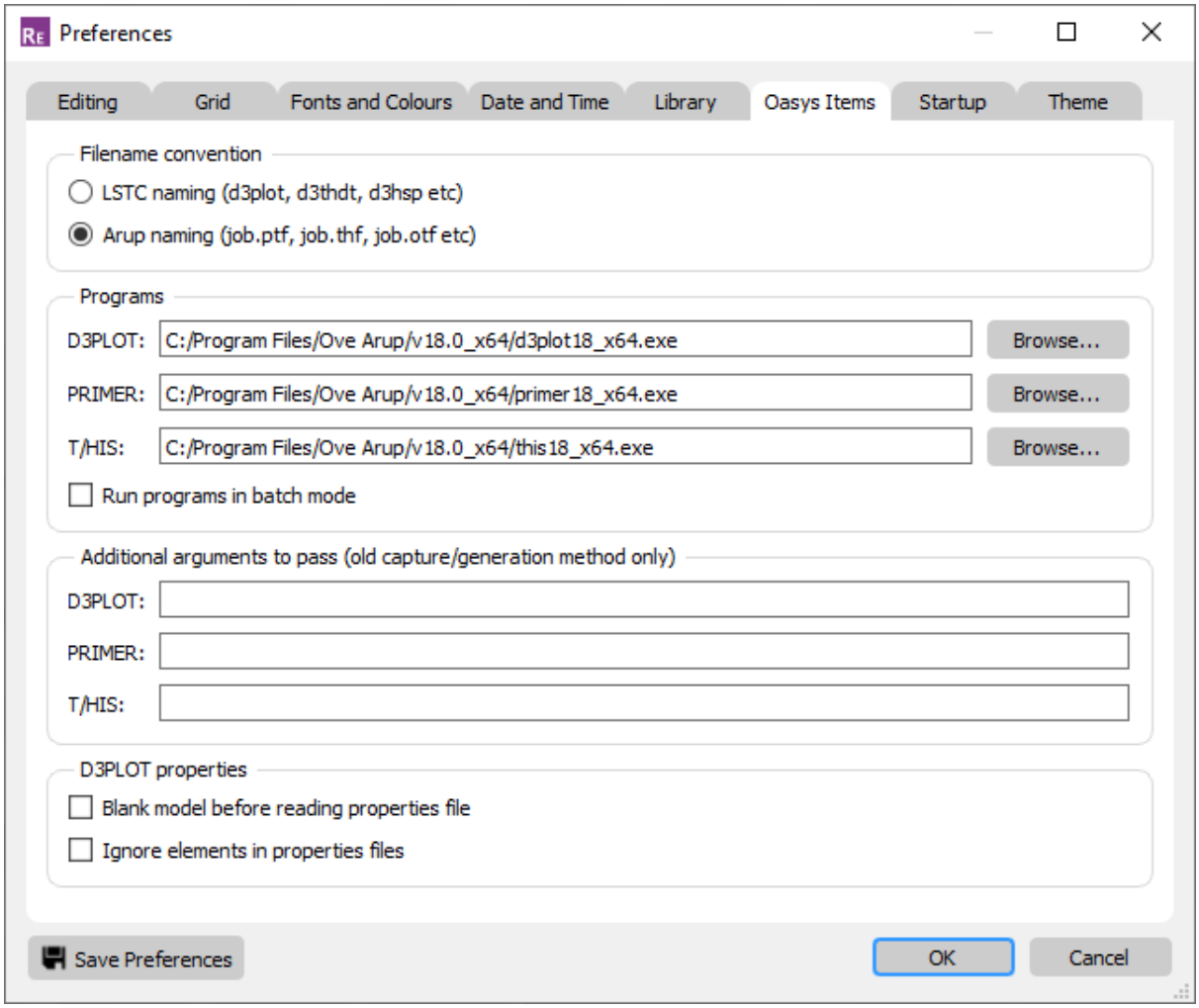
The program executable options would only be useful if you want to use an older version of D3PLOT, PRIMER and/or T/HIS for some reason.

If REPORTER is started in batch mode with the `-batch` command line argument then on Windows D3PLOT, PRIMER and T/HIS will be run without any windows being shown. Setting the **Run programs in batch mode** checkbox will set this option when running REPORTER interactively.

It may occasionally be necessary to pass extra arguments to D3PLOT, PRIMER or T/HIS when generating a report. Extra arguments to pass can be given in the D3PLOT, PRIMER and T/HIS **Additional arguments to pass** textboxes.

The **D3PLOT properties** options allow you to change how properties files are reloaded in D3PLOT.
By default when D3PLOT reads a properties file it only alters the blanking status of parts and elements that are in the file. Any parts that are not in the file will not be affected. The default for parts that are not in the file is to leave them unblanked. This means that if you record a properties file for one model and replay it on another model which has extra parts, the extra parts will not be blanked. With the **Blank model before reading properties file** option set D3PLOT will blank the model before reading the properties file so any extra parts not in the file will be blanked.

Older versions of D3PLOT always wrote the blanking status of elements as well as parts to the properties file, even if all of the elements in the part were blanked or unblanked and just writing the part status would be sufficient. If a properties fie recorded for one model was used for another model which had some remeshed parts then the elements in the properties file would not match the actual elements in the model for the remeshed parts. This could result in some of the elements not being blanked or unblanked correctly. With the **Ignore elements in properties files** option set D3PLOT will ignore any elements when reading the properties file and only consider parts.



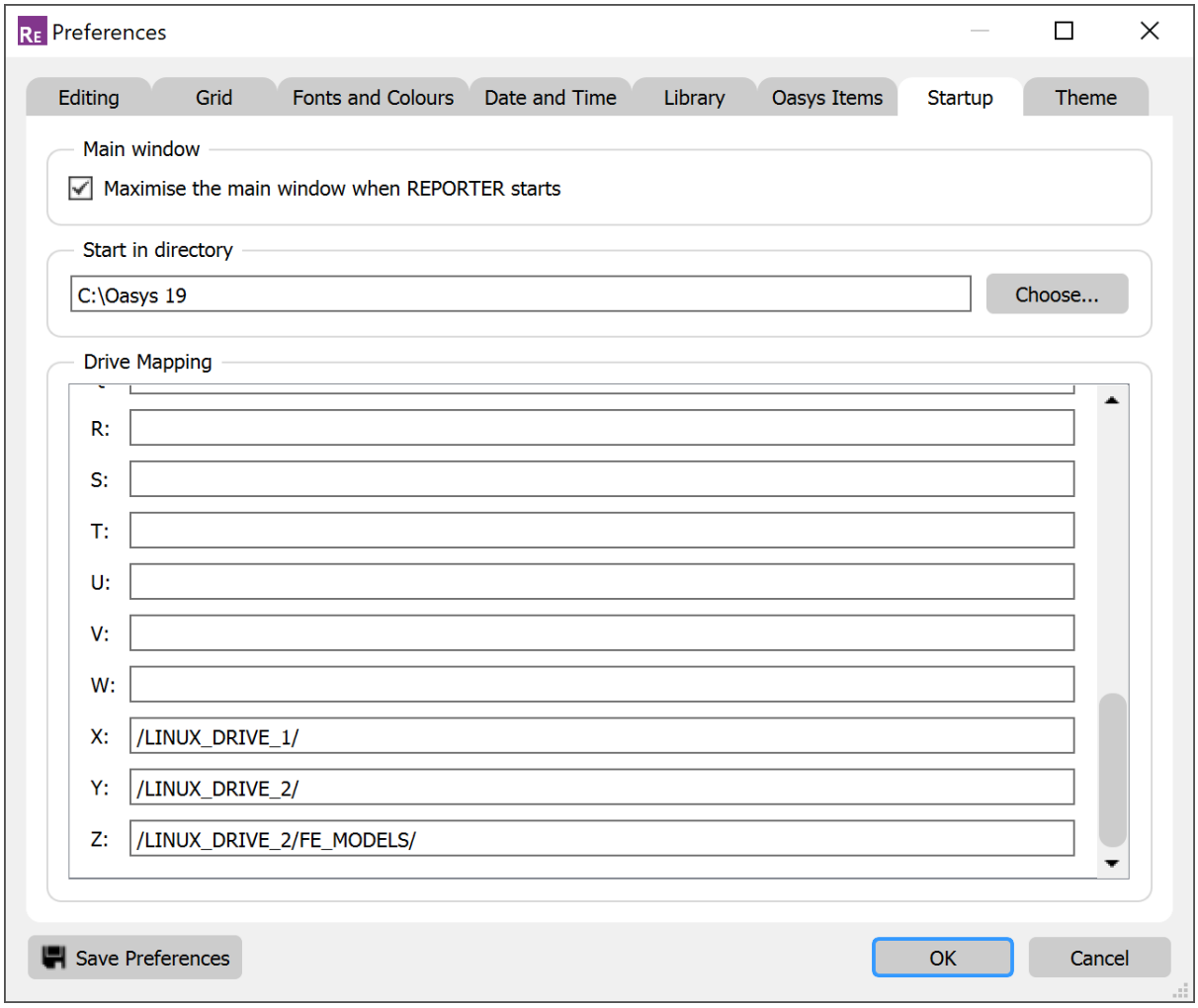
2.7.7 Startup

Here you can control whether the main window is maximised when REPORTER starts (recommended).

You can also control the directory that REPORTER starts in. By default, this is the installation directory..

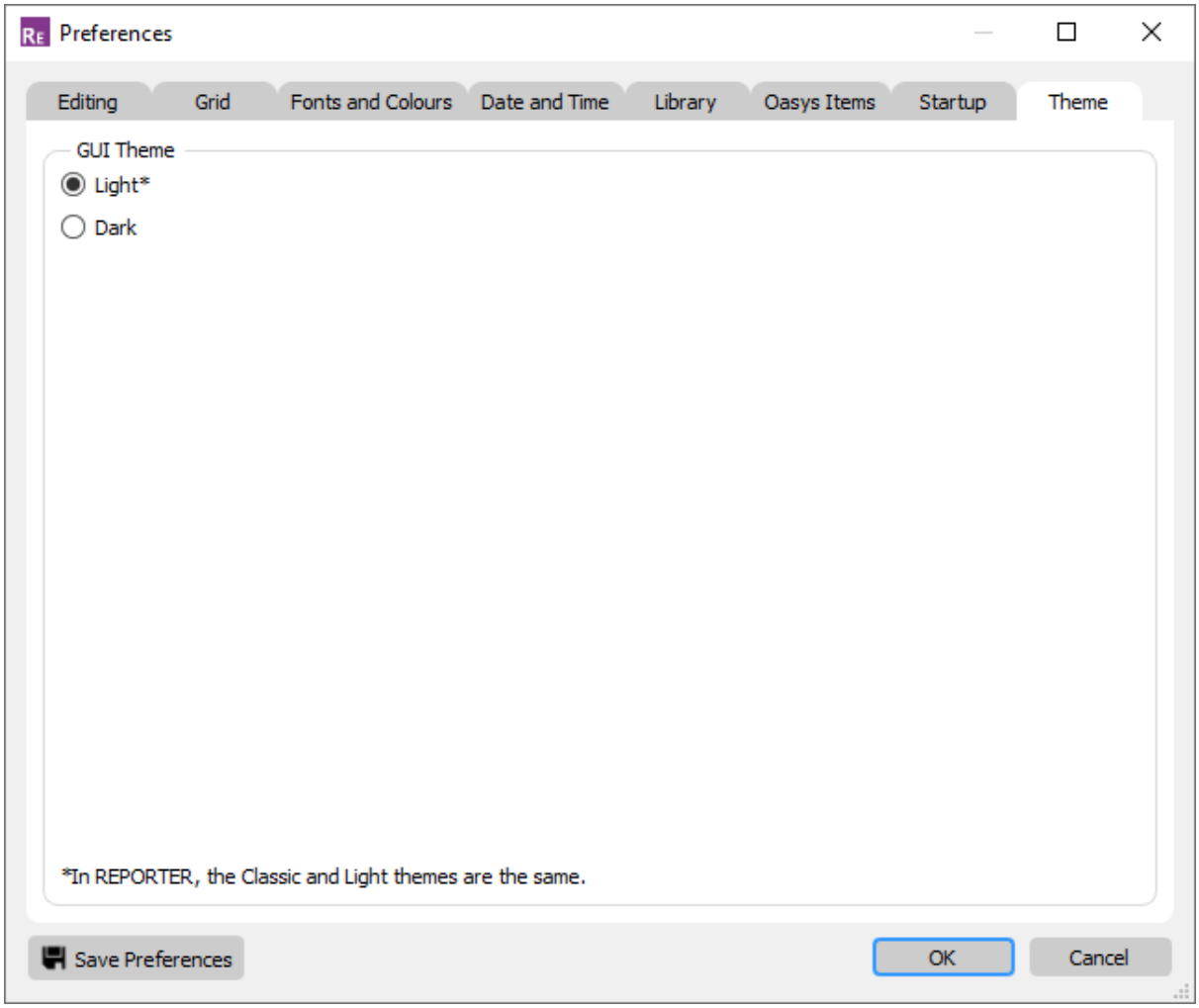
Drive mappings can be changed by editing the text in the input boxes under Drive Mapping. The values used here are written to and read from the oasys*drive_x preference in your HOME oa_pref file (where 'x' can be any letter from 'a' to 'z'). These mappings are useful for being able to work with a single Template across both Windows and Unix systems.

Using the example in the image below where the Z:\ drive is mapped to /LINUX_DRIVE_2/FE_MODELS/, a REPORTER session running on Windows would silently replace the string '/LINUX_DRIVE_2/FE_MODELS/' with 'Z:\' while running commands to manipulate files (without changing the XML content of the .ort Template file). So long as this mapping matches up with the actual drive mapping on your Windows system, REPORTER will find the correct files without complaint. This also works in the other direction (replacing 'Z:\' with '/LINUX_DRIVE_2/FE_MODELS/' when running REPORTER on Linux).



2.7.8 Theme

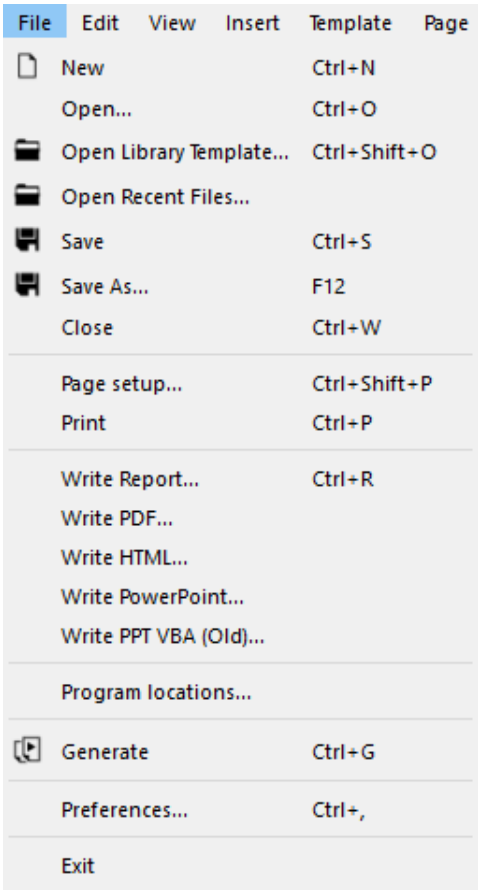
Here the GUI theme for REPORTER can be set. Legacy is deprecated and we do not recommend its usage. Light is the default theme, with Dark providing an alternative experience for users.



For more information on Themes, see the general Themes section.

3. Opening and closing templates and reports

Templates can be created, opened, or saved by either using the **File** menu or the **File Buttons**



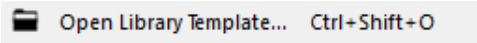
3.1 Creating a new template

A new template can be created from either the **New file** option in the **File** menu or by using the **New file** button.

3.2 Reading an existing template or report

An existing report template can be opened from either the **Open file** option in the **File** menu or by using the **open file** button.

3.3 Reading a library template



When REPORTER starts, the **Choose a Template** window is shown, allowing you to open a file from the recent files list or to select a template from the library. Library templates are presented on different tabs. The **Standard** tab has some standard layout templates to help you start creating reports, and to provide some inspiration. The **Automotive** tab has a number of built in templates to create reports for standard automotive crash test protocols (EuroNCAP, IIHS etc.). You can also open the **Choose a Template** window at any time with **File → Open Library Template**.

In the **Choose a Template** window, select a template by clicking on its thumbnail with the mouse and then clicking **OK** to open the template. You can also double-click on templates to open them immediately. The **Cancel** button will exit you from this window without opening a template.

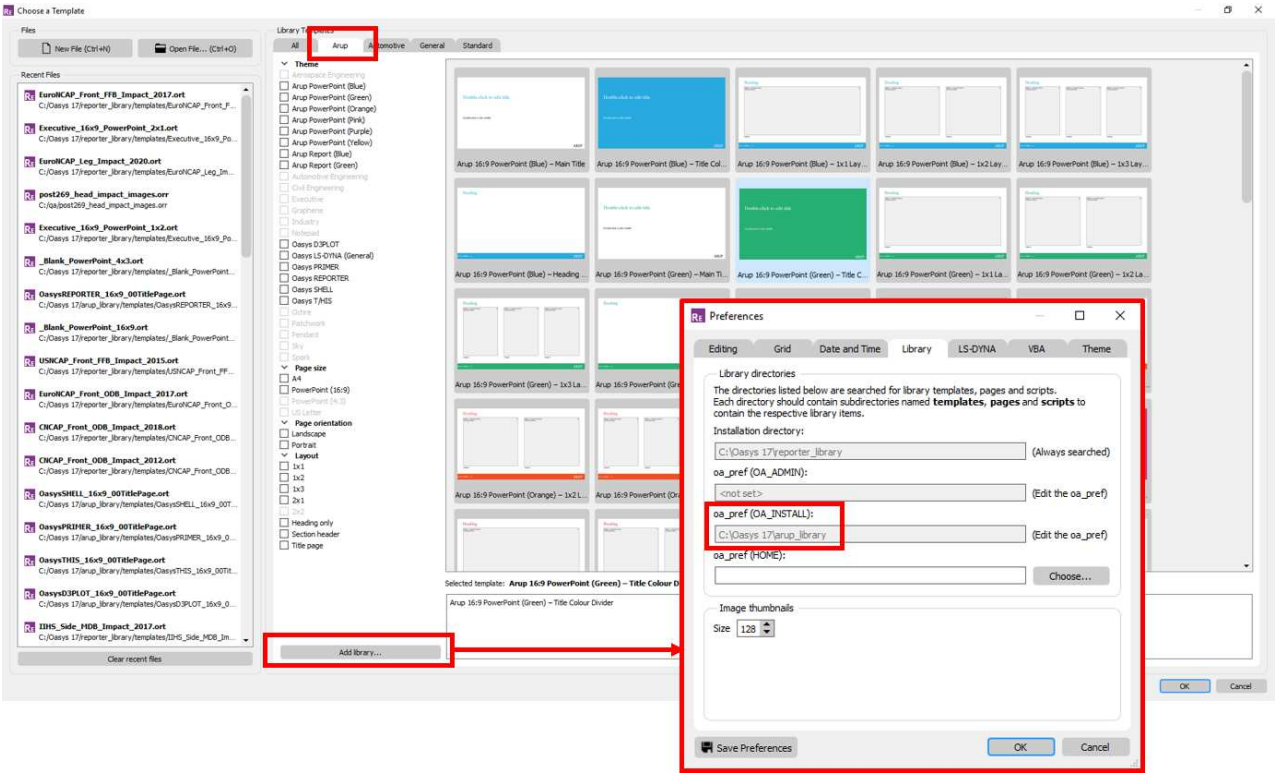
The options presented to you when using the **Choose a Template** window will match the filters ticked in the checkboxes.

Pressing **New File** will open a new template with a blank portrait A4 page.

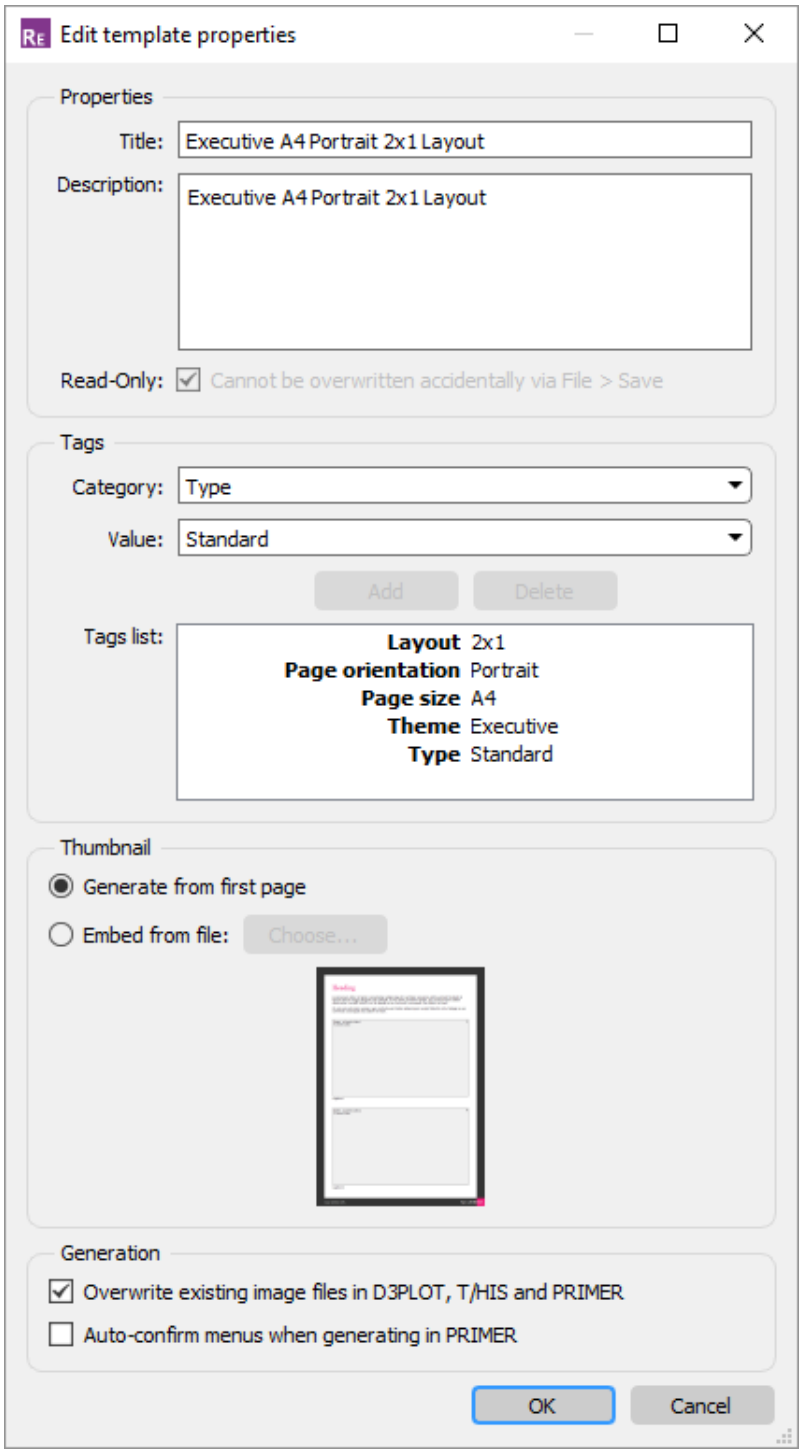
Pressing **Open File...** will allow you to open a template or report from elsewhere on your system.

The Recent Files section contains a list of your most recently accessed templates and reports. Pressing **Clear recent files** will delete this list (the files themselves will not be deleted).

As well as using the library templates suplied with REPORTER, you can create your own library of templates. If you add tags to your templates and save them to a common directory, you can add them as a library by pressing **Add library...**. This will open the Library tab of the Preferences menu, allowing you to include your directory of templates. For example, at Arup, we have a library of templates all tagged with **Type: Arup** so that they all appear together in an "Arup" tab for easy access.



3.4 Editing template properties



The Edit Template Properties window can be accessed by selecting **Template → Template properties...**.

Properties

The Properties section contains the Title and Description for the current template which can be freely changed. The Read-Only property is immutable and signals whether the template can be overwritten. For library templates it is always checked; for user-made templates it is unchecked.

Tags

The tags list provides information about the current template that is used for filtering in the Choose a Page / Template windows. Tags can be added to the list by selecting a category and value from the drop-down menus and pressing **Add**. User-defined category-value pairs may also be added by typing directly into the Category and Value boxes instead of using the drop-down buttons. Tags can be deleted from the list by selecting them from the Tags list panel and pressing **Delete**. The **Type** category determines the tab in which the template appears in the **Choose a Template** window.

Thumbnail

The thumbnail determines the image that is shown to represent the template in the Choose a Template window. This can either be generated from the first page of the template, or embedded from a separate file.

Generation

When generating image files for D3PLOT and T/HIS the **Overwrite existing image files** preference controls what do do if an image with the same name exists. By default (selected) REPORTER will overwrite the image. However, you many want to run the same template multiple times for different models in the same directory. With this unselected a new image will be created for each model.

The **Auto confirm menus** preference controls how macros are replayed when generating a PRIMER object. When creating or updating a PRIMER object, REPORTER ensures that the **`MENU_AUTO_CONFIRM`** environment variable is unset so that any listing boxes created are not automatically dismissed. However in version 16.0 and earlier when generating a PRIMER object REPORTER would set the **`MENU_AUTO_CONFIRM`** environment variable. In some instances this could mean that the macro recorded in the object would not play properly (as when it was recorded the **`MENU_AUTO_CONFIRM`** environment variable was not set). From version 16.1 onwards by default REPORTER will now not set the **`MENU_AUTO_CONFIRM`** environment variable when generating PRIMER objects to be consistent. There may be very rare cases where this needs to be set when generating, in which case the **Auto confirm menus** preference can be used.

These preferences are not programme wide preferences. They are actually stored with the template and read/written as properties so they must be set for each active template.

3.5 Saving a template

A template can be saved by choosing the **Save as** option in the **File** menu and then changing the file type to **template**.

3.6 Saving a report

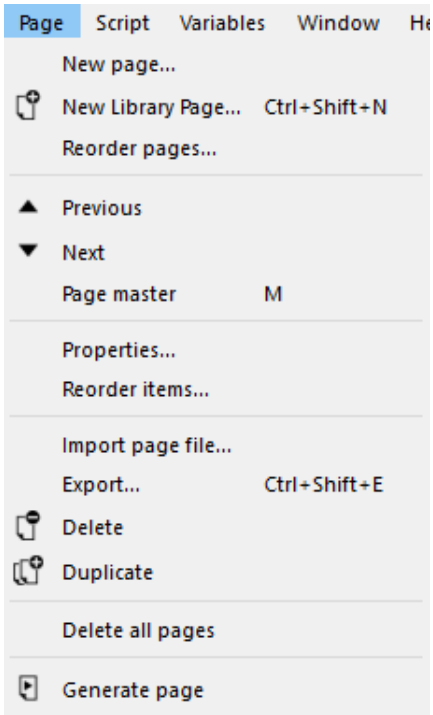
A report can be saved by using the **Save as** option in the **File** menu and setting the file type to **report**. The difference between a report and a template is that a template is the instructions or recipe of how to construct the report. To actually create the report you have to generate it and then create some sort of output. This could mean running D3PLOT command files, programs, FAST-TCF scripts etc.

Alternatively, once the report has been created you can save the whole thing as a report. This saves the output of programs, command files etc. with the template so when you next read the file the results are already available (the report does not need to be regenerated).

4. Inserting and editing pages

A report is generally made up of a number of different pages. Only one page is shown on the screen at any one time. Moving through the pages of the report, adding, deleting, and reordering pages are all controlled from the **Page** menu.

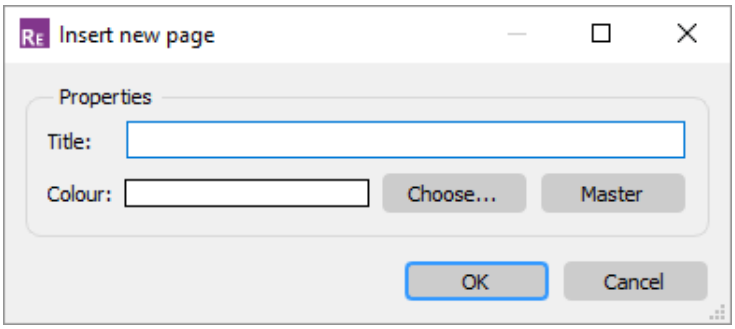
Some of the options within the **Page** menu are also available from the Page toolbar.



4.1 Adding a new page

A new blank page can be added by using the **New page** option in the **Page** menu. This will bring up a **Page layout** window from which you can give the new page a title,and set the background colour.

Choose... can be used to pick the background colour for the page. Alternatively pressing **Master** will make the page inherit the colour from the master page.



4.2 Adding a new page from the library

A new page can also be added by selecting an existing page layout from the library by using the **New Library Page...** option in the **Page** menu (or toolbar).

This will bring up the **Choose a Page** window from which you can select a page layout. Highlight the page layout you want by clicking on its thumbnail with the mouse and then clicking on then **OK** button to create the new page. The **Cancel** button will exit you from this window with out creating a new page.

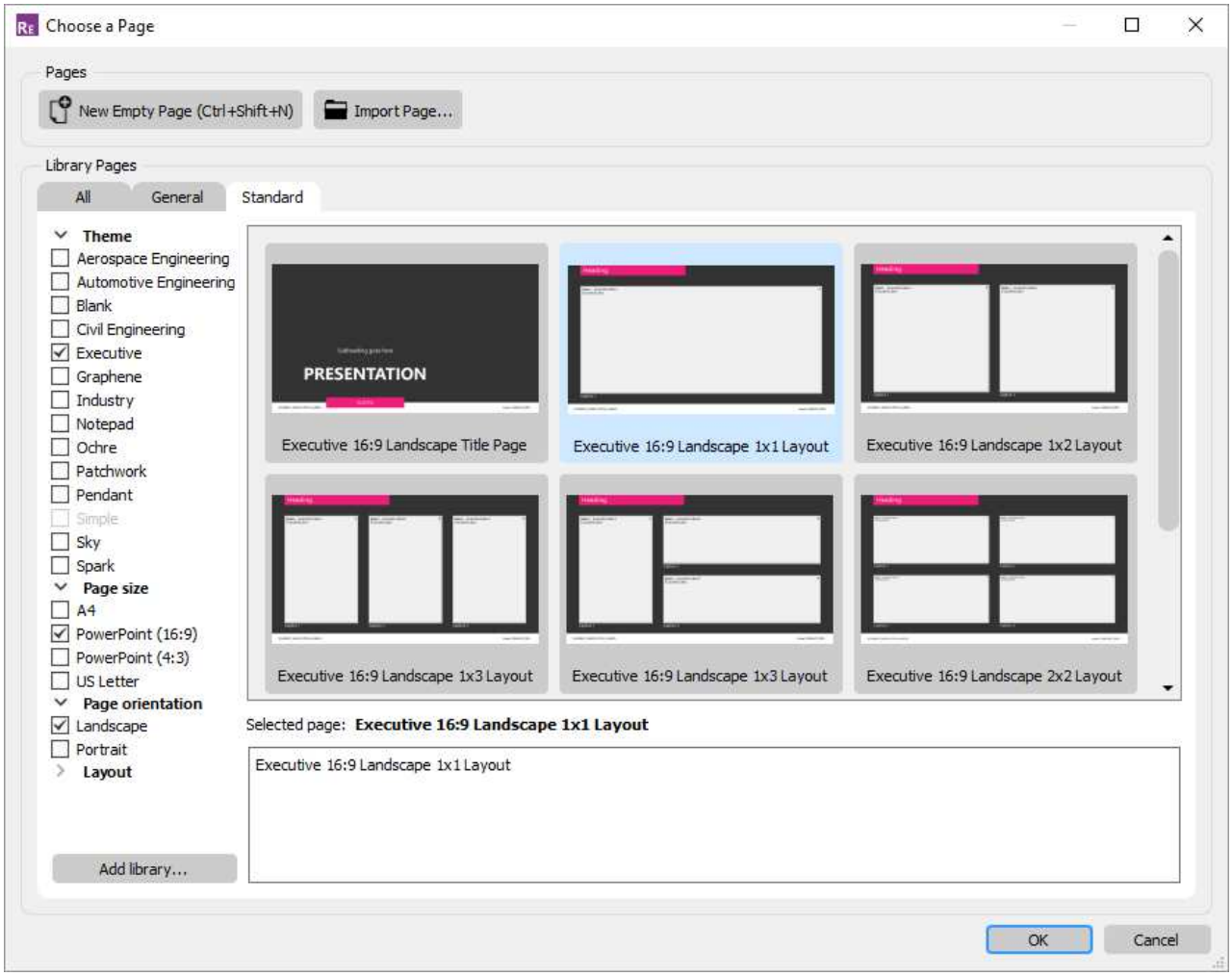
The options initially presented to you when using the **Choose a Page** window will match the Template Properties of the current template. For example, in the image below we are currently using the Exectuive 16:9 Landscape Title Page template in REPORTER. Thus when the **Choose a Page** window is opened the Executive, PowerPoint (16:9), and Landscape filters are all ticked in the tree on the left, and the library page thumbnails on the right all match these filters.

Pressing **New Empty Page** will add a new blank page to your template with the same size and orientation as the current page.

Pressing **Import Page...** will allow you to add an existing page to your REPORTER page library.

Pressing **Add library...** will open the Library tab of the Preferences menu.

See the library object appendix for more details on using the library.



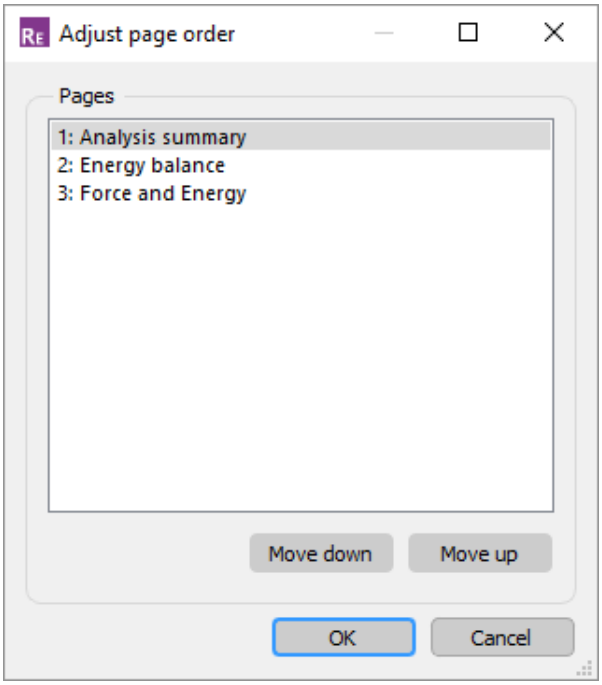
4.3 Deleting pages

You can delete the present page you are working on by using the **Delete page** option in the **Page** menu (or toolbar), or you can delete all the pages in the report template by using the **Delete all** option in the **Page** menu. Both of these option will bring up a confirmation window in which you need to confirm the delete operation.

4.4 Duplicating pages

You can copy the current page by using the **Duplicate page** option in the **Page** menu (or toolbar). This will make a copy of the current page, and insert it after that page. The current page will also be changed to this newly created page.

4.5 Reordering pages

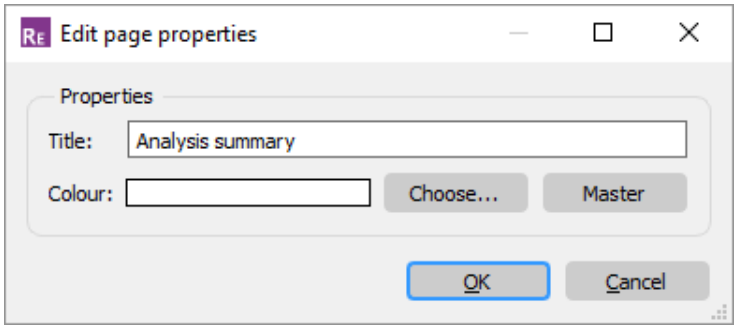


You can change the order of the pages in the report by using the **Reorder pages** option in the **Page** menu. This will bring up the reordering window.
The pages are listed by the page number and title. The page order can be modified by clicking on the page you want to move in the **Pages** box. This will highlight that page, which can then be moved by using the **Move up** and **Move down** buttons. Once finished the **OK** button will save the new order and exit the window. The **Cancel** button allow you to exit this window without making any changes to the page order.

4.6 Changing the current page

You can change the current page you are working on by using the **Prev page** and **Next page** option in the **Page** menu (or toolbar) to change the current working page to the previous or next page in the report. You can also move through the pages by using the `Page Up` and `Page Down` keys. If you have a mouse which has a wheel then the wheel can also be used to move through the pages.

4.7 Changing the page properties



You can change the title of the current page by using the **Properties** option in the **Page** menu. This will bring up an edit page properties window. The new page title is entered into the **Title** text box and the colour can be chaged by clicking on the **Choose** button. Pressing **Master** will make the page inherit the colour from the master page.

Clicking on the **OK** button will save the changes and exit this window. The **Cancel** button will exit this window with out making any changes to the page title

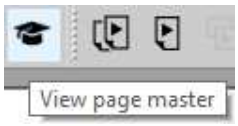
4.8 Inserting pages from file

You can insert all the pages from another template file into the current template by using the **Insert** option on the **Page** menu, and then selecting the required template file from the **File** window.

4.9 Importing and exporting pages

Individual pages can be exported from a template using the **Export page** option. These pages can then be used in the page library or can be imported into another template by using **Import page**. Individual pages should be saved with the extension `.orp` so REPORTER can find them.

4.10 Page masters

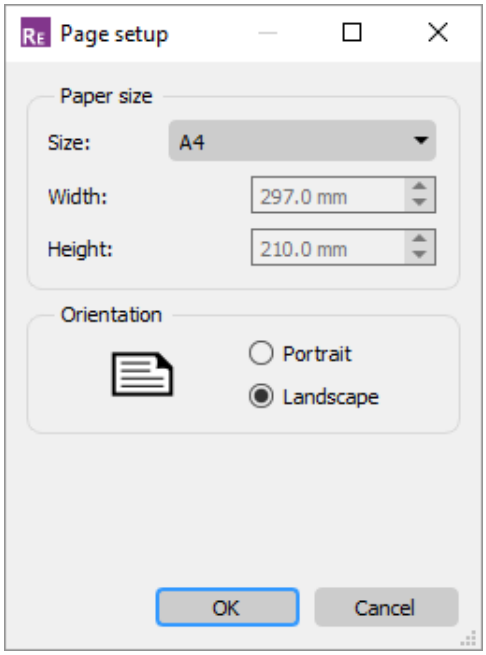


Page masters can be used to automatically add objects to every page in the report. For example you may want to have your project name written on the bottom right corner of every page in the report. You could do this by having a standard page and either use the page library or import it each time you want to create a new page. This will work, however if in the future you want to edit the project name, you would need to edit each page individually.

An alternative is to use page masters. A page master is a type of template used to keep each page looking the same (eg such as using a company logo). Each REPORTER template has one associated master page and any objects that you put on that page will automatically appear on every page in the template.

The master page is accessed via a toggle, either by clicking on the page master icon (see image on the right), pressing the 'm' key, or using **View** ... **Page master**. From within this view, the master page can be edited in exactly the same manner as you would interact with a normal page. To close the master page and return to the normal page view, simply click the toggle as before.

4.11 Page Setup



To set up the page settings choose the **Page setup** option from the **File** menu. This allows you to change the page size and orientation. If the page size and/or orienation is changed objects on existing pages are automatically moved to ensure that they are not outside the page boundaries.

4.12 Generating a single page

Instead of generating the entire report you can generate a single page by using the **Generate page** option in the Page menu (or Generate toolbar). However, note that if some of the objects on the page require data that would be generated on previous pages and those pages have not yet been generated the page generation will not work.

5. Inserting and editing simple objects

REPORTER allows you to create and edit a number of different shapes through the use of the various **Tools** and **Style** button options.

5.1 Using the Grid and Snap options

5.1.1 Grid

The grid option can be turned on by clicking on the **Grid** button. This will create a grid of dots on the screen with a pitch equal to the grid size, this is to help you in aligning objects in the report. These dots will not appear in the generated report. The size and attributes of the grid can be modified by using the **Grid** tab in the preferences.

5.1.2 Snap

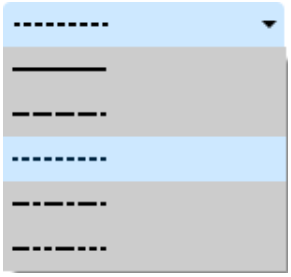
The snap option can be turned on by clicking on the **Snap** button. This will create an invisible grid with a pitch equal to the snap grid size. When positioning and sizing object the point you select will not be the exact position of the mouse pointer but the nearest point on the snap grid.

The size and attributes of the grid can be modified by using the **Editing** tab in the preferences.

5.2 Setting line style, thickness, colour, and fill colour

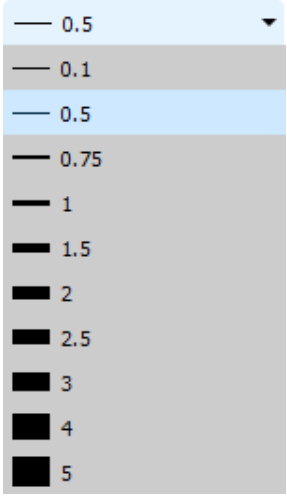
When you select an object the Style widget buttons are updated with the properties of the selected object. Changing any of these while and object is selected will change the property of the object.

5.2.1 Line style



The line style can be set using the **Line style** button (top button in the Style widget). Clicking on this will bring up a **Line style** window from which the line style can be selected.

5.2.2 Line thickness

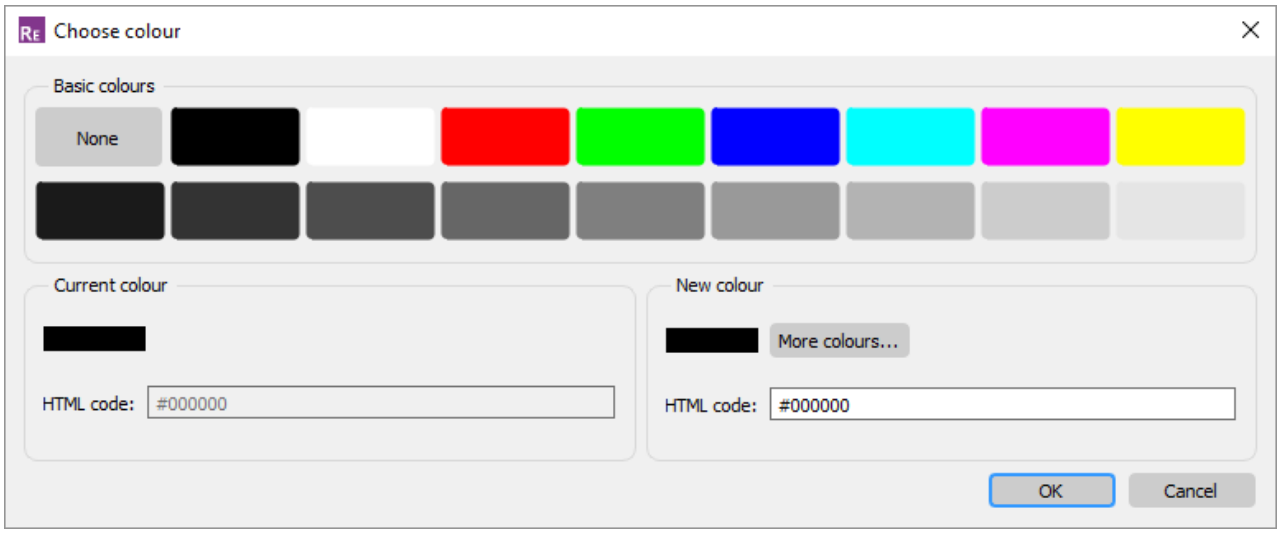


The line thickness can be set by clicking on the **Line thickness** button (second from top in the Style widget). Clicking on the button will bring up a **Line thickness** window from which the line thickness can be selected.

5.2.3 Fill, Line and Text Colour

The fill, line or text colour can be set using the **Fill colour** button the **Line colour** button or the **Text colour** button, the current colour is displayed to the right of the button:

Clicking on this will bring up the **Colour** window.
The new colour can be selected from those on display or by clicking on the **More colours** button a set of red, green, and blue sliders can be brought up which you can use to create you own new colour. For the Fill colour you can also select **no colour** which will give you a transparent Fill colour allowing object below to show through. The **Done** button will exit this window setting the fill or line colour to the new colour. The **Cancel** button will exit you without changing the colour.



5.3 Inserting and editing shapes, images, and text

5.3.1 Lines and arrows

You can create a line or arrow by using the **Line** and **Arrow** tools.

To create a line click and drag the mouse from the point you want the line to start from to the point you want the line to end at. It is the same procedure for creating an arrow with the arrow head appearing at the end point of the line. The line type, thickness, and colour will be set to the current settings.

5.3.2 Rectangles

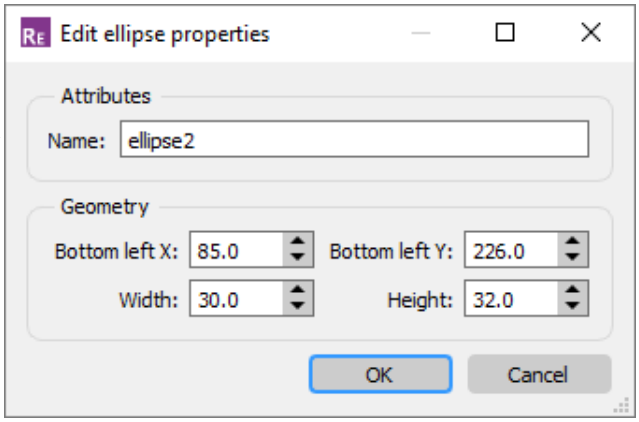
You can create a box by using the **Rectangle** tool.
To create a box click and drag the mouse from one corner of the box to the other. If the `shift` key is held down while doing this a perfect square can be created.
If the `Ctrl` key is held down then the initial click position will be the centre of the rectangle instead of one corner. The line type, line thickness, line colour, and shape fill colour will be set to the current settings.

5.3.3 Ellipses and circles

You can create an ellipse or circle by using the **Ellipse** tool.

To create an ellipse click and drag the mouse from one corner to the other of a rectangle into which the ellipse will be drawn. If the `shift` key is held down while doing this a perfect circle can be created. If the `Ctrl` key is held down then the initial click position will be the centre of the ellipse instead of one corner. The line type, line thickness, line colour, and shape fill colour will be set to the current settings.

The edit window for the **Line**, **Arrow**, **Item** and **Ellipse** Items simply allow you to edit their geometry (e.g. as can be seen in the image on the right for an **Ellipse**).



5.3.4 Text

A single line of text can be added by using the **Text** tool.

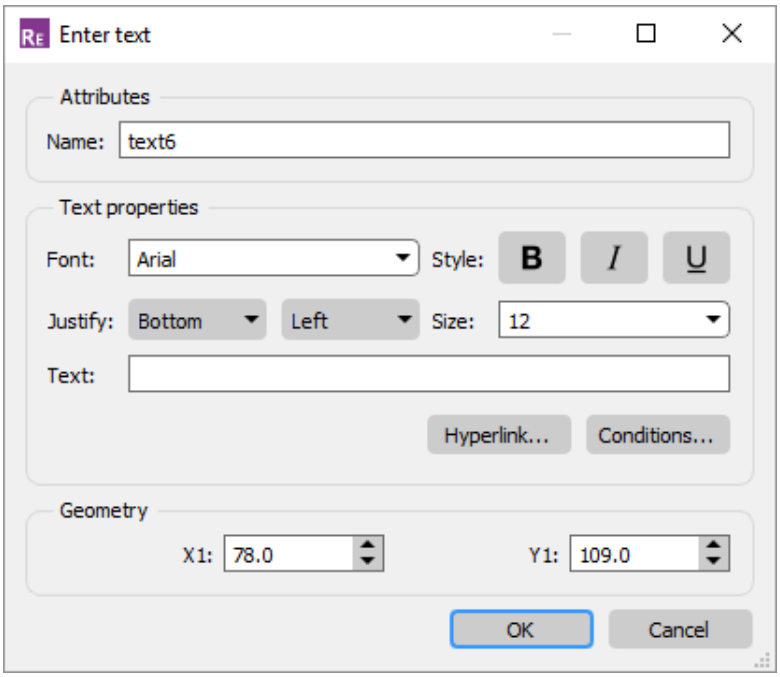
To add text click on the point you want the text to be, this will bring up a **Text** window.

Text can be added using the **Enter text** box. You can also enter variables in the text by right clicking in the text box or pressing **Ctrl+I** which will allow you to bring up an **Insert variables** window from which to select a variable.

The font, style, and size are set in the relevant boxes.

The horizontal and vertical justification of the text can be set independently to enable you to position the text how you want. Changing the vertical alignment can help when trying to align text with program items.

The text colour will be set to the current text colour setting. The **OK** button will exit this window and create the text. The **Cancel** button will exit this window without creating any new text. Also a **Hyperlink...** button (see section10) allows the user to set the text up as hyperlink and the **Conditions...** button (see section11) enables the user to apply conditional formatting to the text.



5.3.5 Textbox

Text inside a box (with multiple lines if necessary) can be added by using the **Textbox** tool.

To add a textbox click and drag the mouse from one corner of the textbox you want to create to the other. This will

bring up a **Textbox** window.

Text can be added using the **Text** box. You can also enter variables in the text by right clicking in the text box or pressing **Ctrl+I** which will allow you to bring up an **Insert variables** window from which to select a variable.

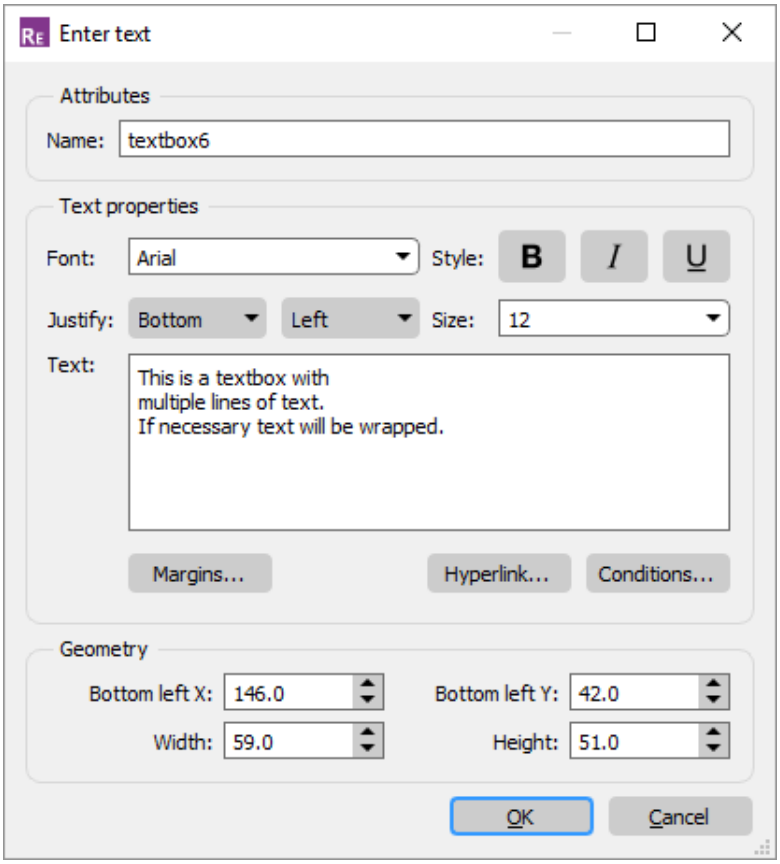
The font, style, and size are set in the relevant boxes.

The horizontal and vertical justification of the text can be set independently to enable you to position the text how you want.

The text colour will be set to the current line colour setting. The **OK** button will exit this window and create the text. The **Cancel** button will exit this window without creating any new text. Also a **Hyperlink...** button (see section10) allows the user to set the text up as hyperlink and the **Conditions...** button (see section11) enables the user to apply conditional formatting to the text.

The background and border colour for the textbox can be set using the fill and line colour buttons in the style toolbar. The border style can also be set with the line style and line thickness buttons in the sytle toolbar.

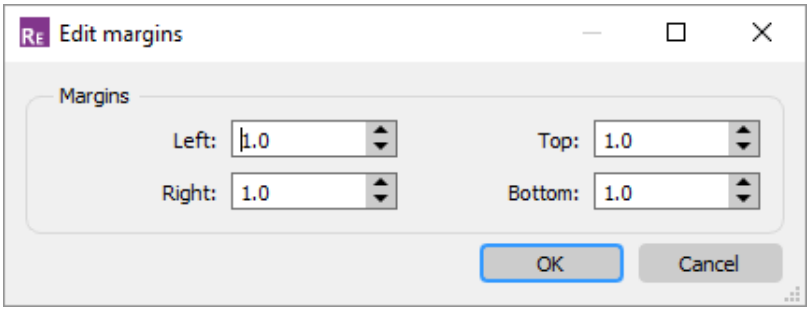
The margins for the textbox can be changed by using the **Margins...** button.



Margins

The Edit margins dialogue box allows you to change the margins around the text in a textbox.

The margins can be set independantly for the top, bottom, left and right sides of the textbox.



5.3.6 Images

PNG, Bitmap, GIF, and JPEG images can be added using the **Images** button.

To add an image, click on the point where you want the bottom left corner of the image to be. This will bring up an **Image** window.

Enter the image filename into the **Image** text box or click on the **Choose...** button to call up a **File** window from which to select the image file. You can also enter variables by right clicking in the text box which will allow you to bring up an **Insert variables** window from which to select a variable.

The **OK** button will close this window and add the image to the page.

The **Cancel** button will exit this window without adding an image.

The **Cropping...** button (see section 5.4.2 below) can be used to crop the image before showing it. Also the **Hyperlink...** button (see section10) allows the user to set the image up as hyperlink.

The **embed image in template** checkbox allows you to embed the image file directly in the template. If this option is selected, REPORTER will display the image from the embedded data rather than searching for the external image file. This feature can be useful when sharing templates with users who do not have access to the original image files. For example, if you create standard templates for users in your organisation, you could embed the company logo image so that you do not need to supply the image file along with the template file.

We recommend that where users have access to the image files, you continue to use the image file link (rather than embed) because this reduces the template file size and means that templates will automatically update if changes are made to the source files.

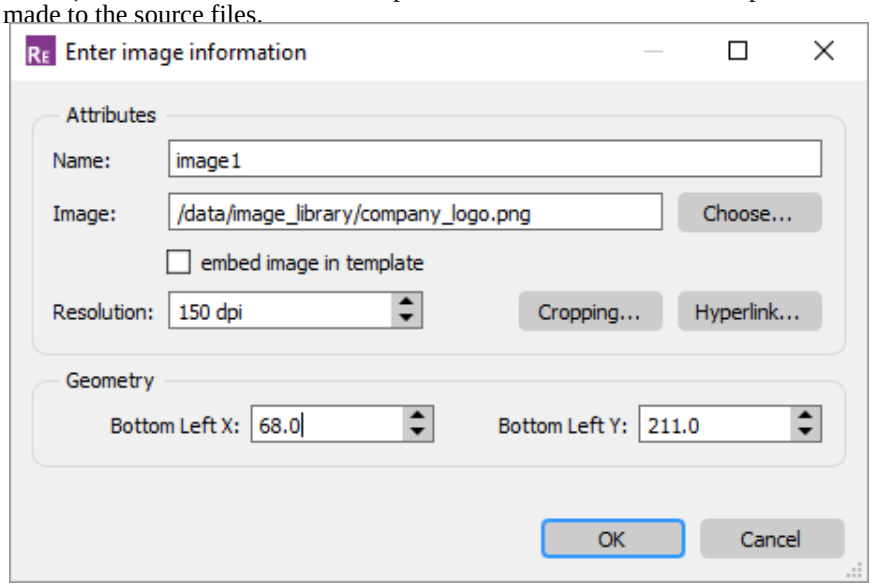
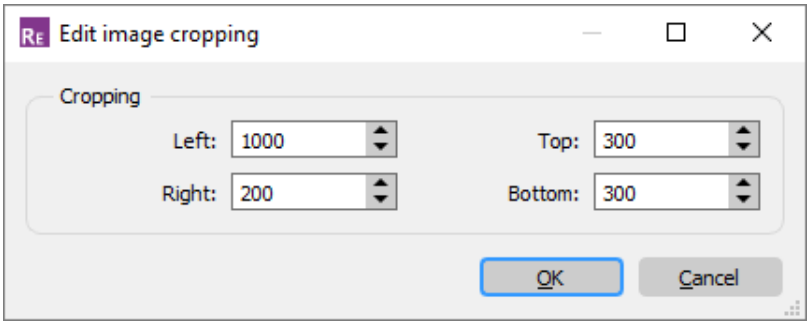


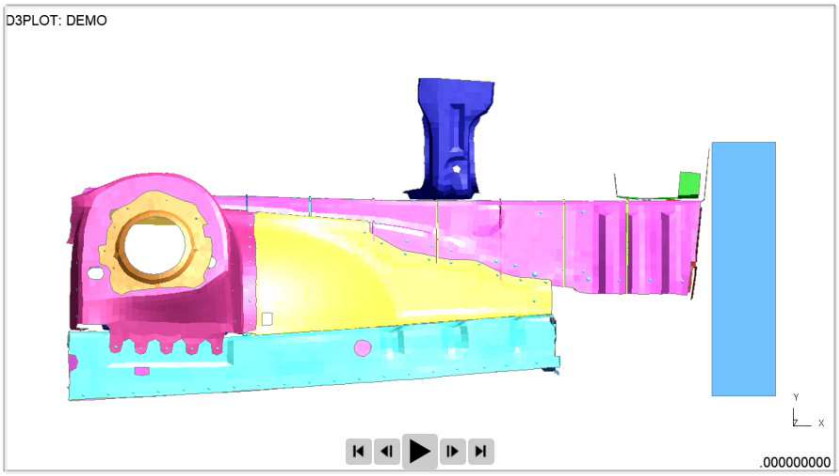
Image cropping

The **Cropping...** button allows you to crop parts of the image before it is shown. Pressing the button maps the panel shown on the right. This allows you to input how many **pixels** will be cropped from the left, right, top and bottom of the image before showing it. Type the values or use the up and down arrows to set the values you require.



Pressing **OK** will update the cropping information for the image. Pressing **Cancel** will abort without changing the values.

Animated Images



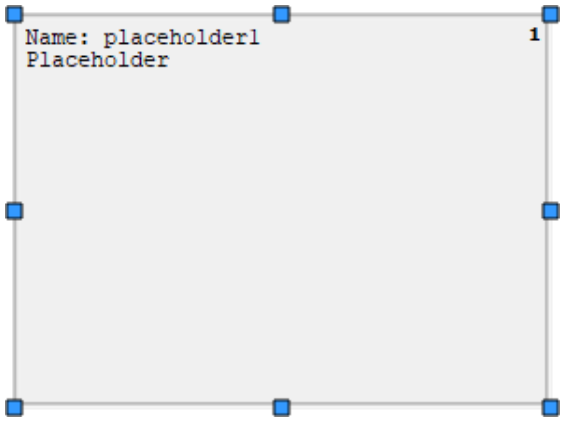
From version 18.0, **Image** Items using the .gif or .mp4 extension can now be animated on the page. These Items can be added to the page through the **Image** window in exactly the same process as described above, although the **Cropping** and **Hyperlink** features are disabled for animated Items. It is still possible to **embed** a .gif Image in the Template.

Playback of the animated **Image** can be controlled using either the buttons in the **Animation toolbar**, or by hovering over the Image with the Hand tool while in Presentation view. Hovering over the animated Image will display a border around its perimeter and centred **Playback buttons**, as can be seen in the image on the right. For animated GIFs, only the Play/Pause button is present while hovering. Clicking anywhere on the Item is sufficient to toggle Play/Pause.

Please see the section on animation support for output file formats to see how animations will be displayed when producing different types of output content.

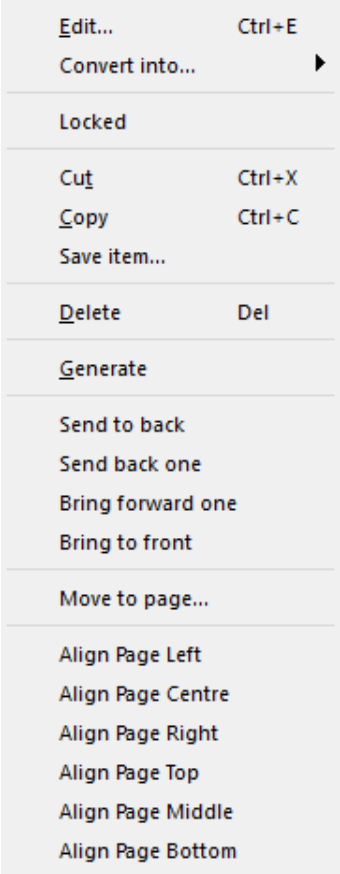
5.4 Editing shapes, image, and text objects

You can edit a existing shape, image, or piece of text by first clicking on the **Select** tool to select the editing tool and left clicking on the object. Multiple objects can be selected by holding down the SHIFT or CTRL keys when clicking on the objects, or by left click mouse dragging a selection box around multiple objects. The object(s) are then drawn with blue boxes or "handles" which allow you to resize the object(s). Additionally the cursor changes to indicate that you can now move the object(s). If you click and drag when over the object(s) you can move them around the page. The cursor keys can also be used to "Nudge" the items around. If you move the mouse over one of the blue "handles" you can resize the object(s). The cursor changes appropriately to indicate how the object(s) will be resized. The escape key can be used to deselect all currently selected objects.



You can also right click on an object regardless of the mode the cursor is in. If the object is not already selected, it is selected and then a popup menu is displayed.

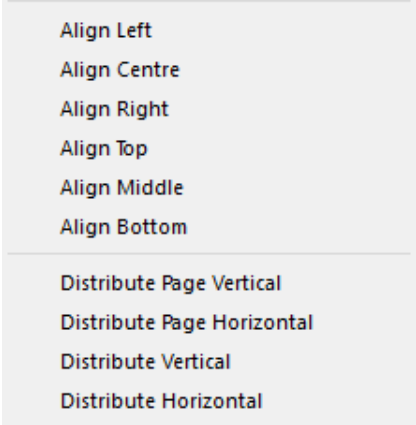
Right clicking when editing an object will bring up a small popup menu.



- **Edit** will bring up an **Edit** window for the object. The **Edit** window will vary depending on what type of object is being edited.
- **Convert Into** allows you to transform one object into another, retaining the same dimensions.
- **Cut** will cut the object(s) from its current page/place and make them available to be pasted in another place.
- **Copy** will make a copy of the selected object(s) and keep them stored in the computers memory until they are pasted or another item is copied.
- **Save** will save a copy of the object(s). This can then be imported elsewhere using **Edit Import...**.
- **Delete** will delete the object(s). This can also be done by pressing the `Delete` key while editing the object(s).
- **Locked** allows you to lock an item on the page so it cannot be moved by dragging with the mouse. See section 5.8 for more details.
- **Generate** will perform any actions required to make the output for the object(s). See section8 for more details.
- **Send to back** will send the selected item(s) behind all the rest of the items on the page.
- **Send back one** will send the selected item(s) behind the next item behind it.
- **Bring forward one** will bring the selected item(s) in front of the next item in front of it.
- **Bring to front** will bring the selected item(s) in front of all other items on the page.
- **Move to page** will move the selected item(s) to the same location(s) on a chosen page.
- **Align Page Left** will align the selected item(s) horizontally with the left hand side of the page.
- **Align Page Centre** will centre the selected item(s) horizontally on the page.
- **Align Page Right** will align the selected item(s) horizontally with the right hand side of the page.
- **Align Page Top** will align the selected item(s) vertically with the top of the page.
- **Align Page Middle** will centre the selected item(s) vertically on the page.
- **Align Page Bottom** will align the selected item(s) vertically with the bottom of the page.

Also note that some of these options are also available through the **Edit** menu.

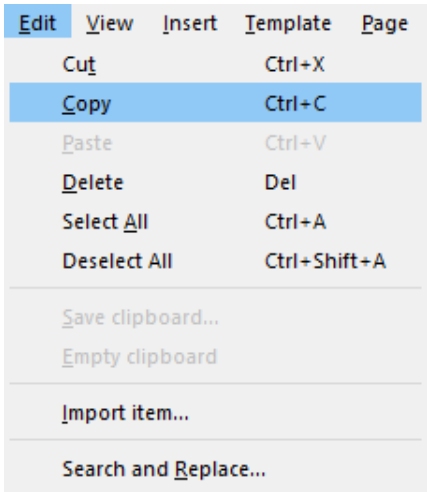
When multiple items are selected, you also get the following options on the popup menu



- **Align Left** will align the selected items horizontally with the left most selected item.
- **Align Centre** will centre the selected items horizontally with respect to the left most and right most selected items.
- **Align Right** will align the selected items horizontally with the right most selected item.
- **Align Top** will align the selected items vertically with the top most selected item.
- **Align Middle** will centre the selected items vertically with respect to the top most and bottom most selected items.
- **Align Bottom** will align the selected items vertically with the bottom most selected item.
- **Distribute Page Vertical** will evenly distribute the selected items vertically on the page.
- **Distribute Page Horizontal** will evenly distribute the selected items horizontally on the page.
- **Distribute Vertical** will evenly distribute the selected items vertically between the top most and bottom most selected items.
- **Distribute Horizontal** will evenly distribute the selected items horizontally between the left most and right most selected items.

5.5 Copying objects and using the clipboard

If you want to copy the object to another page, select the item and use the **Copy** option from the **Edit** menu. This copies the object onto the 'clipboard'.

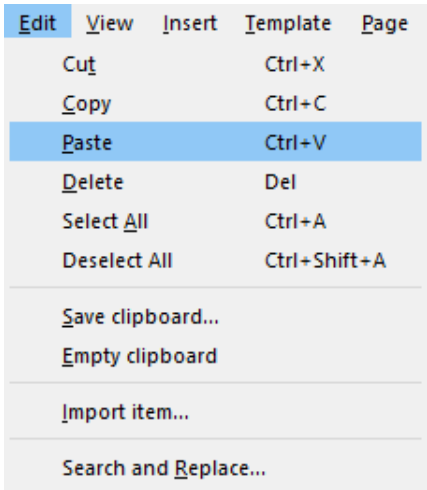


Once you have an object on the clipboard you can **Paste** it onto any page in the template (including the page that you copied the object from). If you paste the object back onto the same page the object will be offset slightly. If you paste the object onto a different page it will be placed in the same position on the page.

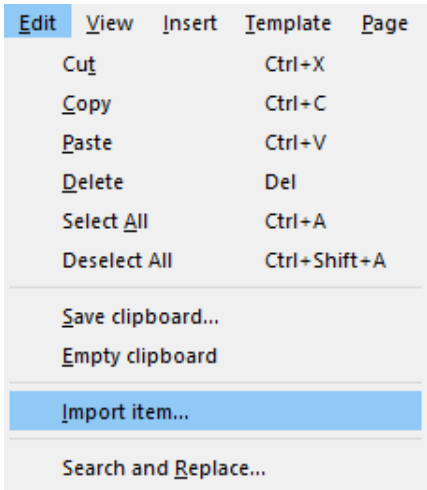
You can also right click on the page at any point and then select **Paste item here.** This will paste the item at the current cursor location.

Alternatively you can save the object to a file by using the **Save clipboard** function. Currently only a single object can be saved. Objects saved from REPORTER should be given the extension `.oro` (REPORTER Object).

Empty clipboard will remove any objects from the clipboard.



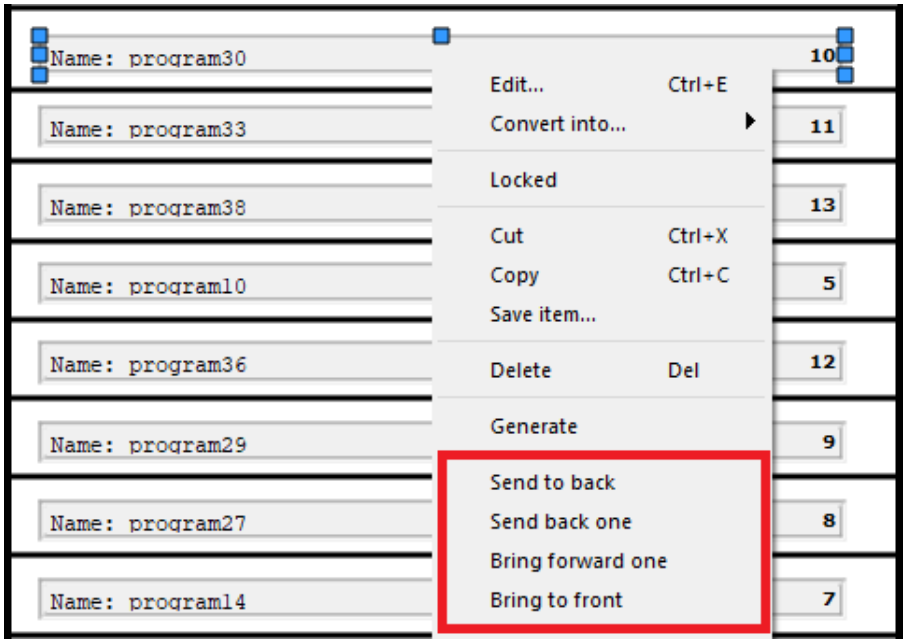
To import an object (that has previously been exported from the clipboard) use the **Import item...** option in the **Edit** menu.



5.6 Reordering items on the page

The order in which items are drawn on the page can be changed by 2 different ways in REPORTER. The order is important as it determines the order in which scripts, programs etc will be run. For more details see section8.1.

The first method can be used when editing objects. Once an item is selected you can right click with the mouse and use the ordering options in the popup to change the object.



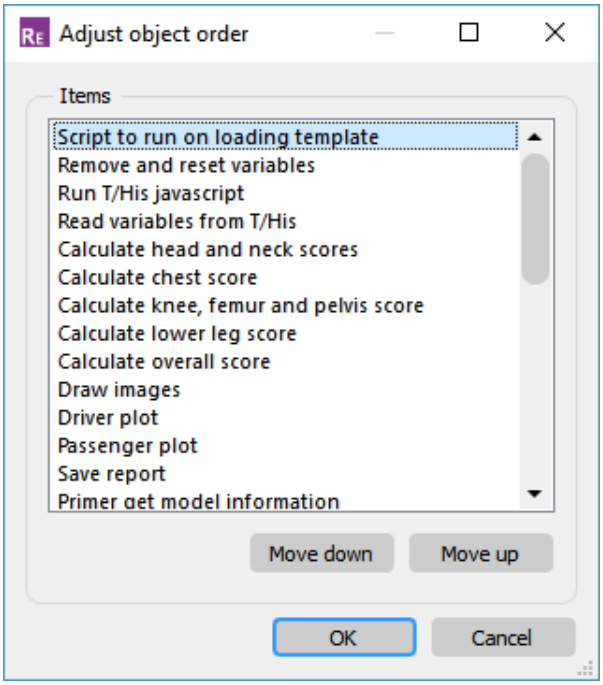
One way of thinking about the object order is to think of a series of 'layers' or transparencies in a stack. Each 'layer' or transparency contains one object. The order in which the transparencies are stacked changes the order in which things are seen. This is exactly the same as layers in various photo editing software.

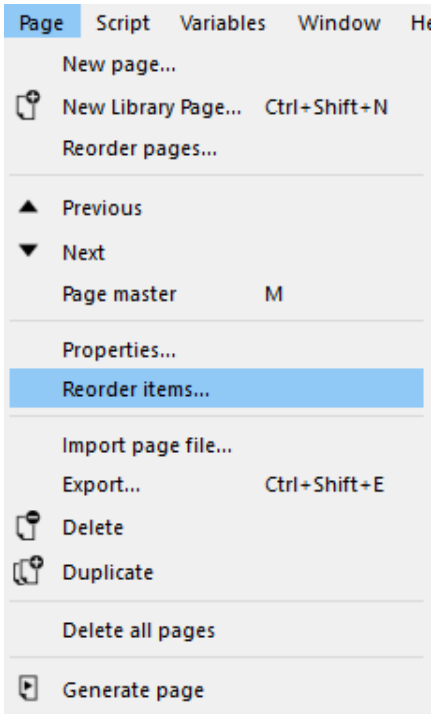
Send to back will make the object the first object drawn on the page (back layer)
Send back one will move the object back a layer
Bring forward one will move the object forward a layer
Bring to front will make the object the last object drawn on the page (top layer)

The second method is to use the **reorder items...** option in the **Page** menu. This brings up a window as shown below. The object stacking order is shown. Clicking on an entry highlights that entry with a green selection box.

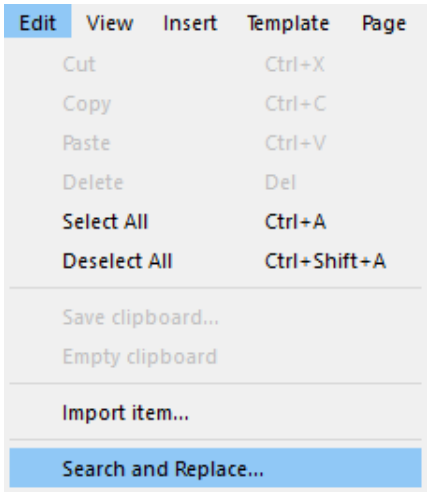
You can use the **Move up** and **Move down** options to change the stacking order of the selected item. Once the objects are in the order you want **OK** will update the page.

Cancel will abort the operation without making any changes.





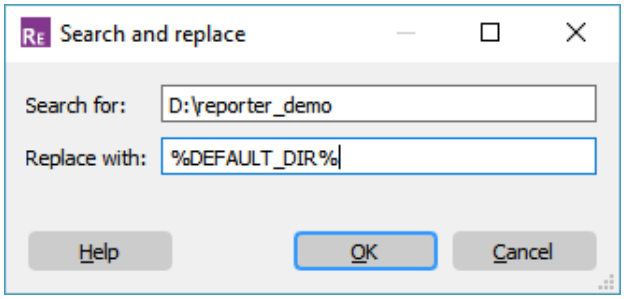
5.7 Search and replace



The search and replace function allows to to search for a text string (or variable) in all objects in the template and replace it with another string (or variable).

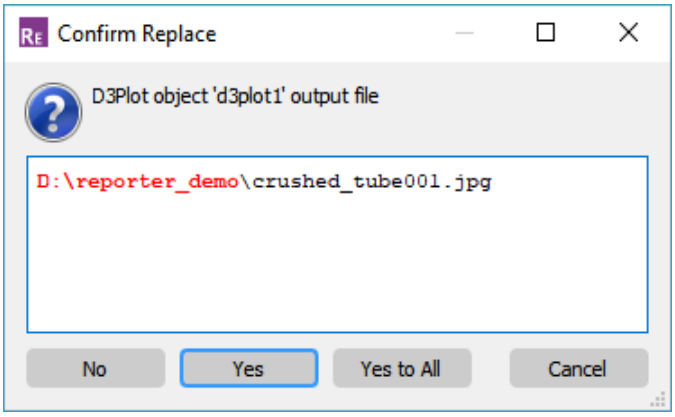
For example, you may make a template which contains D3PLOT objects that have the directories hard wired instead of using a variable for the directory. If you want to generalise the template you can use **Search and Replace...** to replace every instance of the directory name in the template with a variable.

Enter the search and replace strings in the dialog box. You can insert a variable if required by right clicking in the text box and selecting **Insert Variable**.

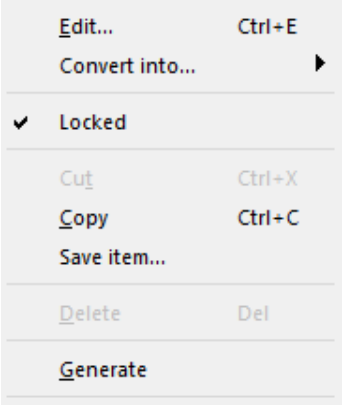


Each time REPORTER finds an instance of the string in an object in the template a confirmation dialog will be mapped giving you the option to replace or skip the string. The object will be selected on the screen so you can see which object you are replacing in and a brief text description wil be given for the object and field that you are looking at.

Cancel will abort the search and replace operation. Pressing **Yes** will do the replace, pressing **No** will skip this instance. If you want to just replace all instances without confirming each one in turn then press **Yes to All**.



5.8 Locking items



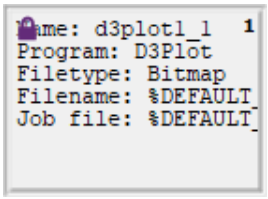
It may be useful to 'lock' objects on the page so that they cannot be moved by dragging. The **Locked** option in the context menu (available by right clicking on a selected item) allows you to do this.

To lock at object first select it and then click on the **Locked** option. If an object is locked it will be shown with a tick symbol. It can be unlocked by clicking on **Locked** again.

Multiple objects can be locked or unlocked at the same time. Toggling the **Locked** option will change the locked property of all of the selected items.

Once an item has been locked it cannot be dragged on the page. It can still be moved by using the arrow keys and the size and/or position can be altered in the edit panel.

You can see which items are locked in design view. A locked item will be drawn with a small padlock symbol.



6. Advanced objects

This section covers the more advanced items you can use in your REPORTER templates, including:

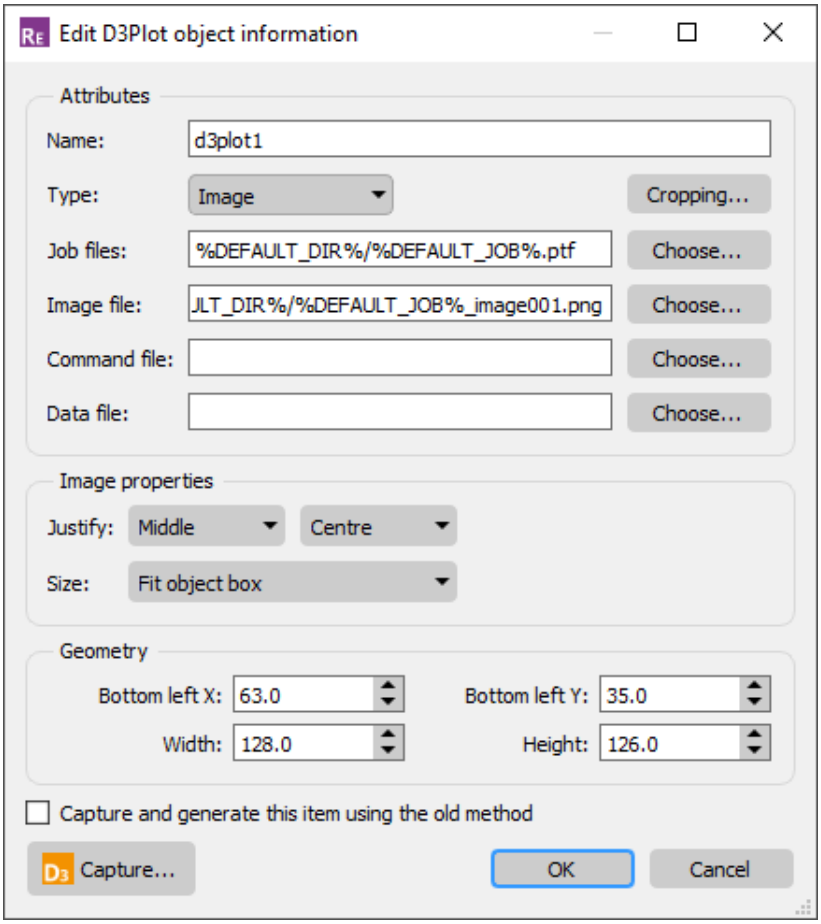
- D3PLOT, T/HIS and PRIMER items – note that from version 17.0 onwards, REPORTER has been fully integrated with D3PLOT and T/HIS to give you a more seamless reporting workflow. See 7. REPORTER Integration for details.
- Placeholder items
- File and Library items
- Tables and Autotables
- Programs and Scripts
- Note items

6.1 D3PLOT objects

D3PLOT items allow you to include output from D3PLOT in your template.

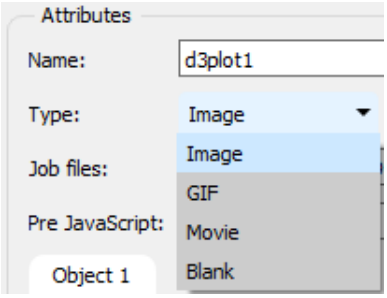
There are two different ways of using D3PLOT items. The first (and by far the easiest) is to use the **Capture...** button to create the object. The second is to use an existing D3PLOT command file to create the output from D3PLOT.

If the Image output type is chosen, the **Cropping...** button can be used to crop away parts of the image from the top, bottom, left and right before showing it. See section 5.4.2 for more details.
The **Justify** buttons in the **Image properties** section allow you to change the justification of the image in the box on the page. REPORTER will not change the aspect ratio of the image. By default the image will be placed centrally in the box and enlarged as much as possible to still fit in the box. The **Justify** buttons can be used to alter the justification in the box if necessary.



There are four different options for the type of output generated from D3PLOT.

- **Image** indicates that the output is a static file.
- **GIF** indicates that the output is an animated GIF.
- **Movie** indicates that the output is an MP4 movie.
- **Blank** indicates that the D3Plot object will not create any output on the page



Playback of animated D3PLOT Items of the **GIF** and **Movie** types can be controlled in the same manner as animated Image Items.

From version 13.0 of REPORTER it is possible to specify the size or aspect ratio of the image created from D3PLOT. This can be changed in the **Image properties** section. If you want to specify a particular aspect ratio or graphics size you can change this. The available options for **Size** are:

- Free size

- Fit object box
- 4:3 aspect ratio
- 16:9 aspect ratio
- 16:10 aspect ratio
- Custom aspect ratio
- Fixed size

The default option from version 17.0 onwards is **Fit object box**. With this option, D3PLOT will capture an image using a graphics window with the same aspect ratio as the object's dimensions in REPORTER.

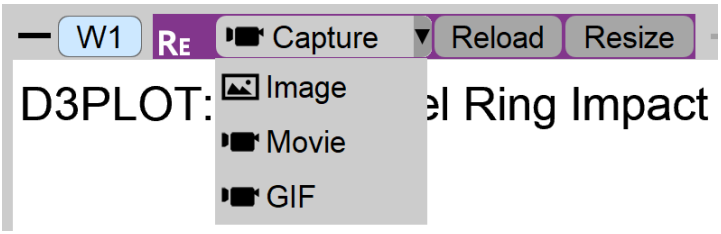
In previous versions of REPORTER, the default option was **Free size**. With this option, the size of the graphics window is calculated by D3PLOT. The actual size will depend on what resolution the monitor is and what scale factors you have chosen for the user interface. This can cause problems if the template is created on one type of display (e.g. a 16:9 monitor) but played back on a different ratio monito (e.g. a 4:3 ratio monitor) as the image size can change and so the output from REPORTER can look different. To make output consistent you can use the other options:

Fit object box will make D3PLOT create a graphics window that has the same aspect ratio as the object box dragged in REPORTER.
4:3 aspect ratio, **16:9 aspect rati**o and **16:10 aspect ratio** will make D3PLOT create a window with the specified aspect ratio.
Custom aspect ratio or **Fixed size** allow you specify a custom width and height. **Fixed size** will make D3PLOT create an image with the specified width and height. **Custom aspect ratio** will make D3PLOT create the largest image it can with the aspect ratio width:height.

6.1.1 Using Capture to create a D3PLOT object

The easiest way to create a D3PLOT object is to use the **Capture...** command. If you press the button REPORTER starts D3PLOT for you if it is not already linked. You can now open the model(s) and do whatever operations you want inside D3PLOT such as rotating, zooming, blanking, selecting the state, setting colours, and so on. Variables can be defined interactively by pressing the **Show Variables** button in the REPORTER panel in D3PLOT.

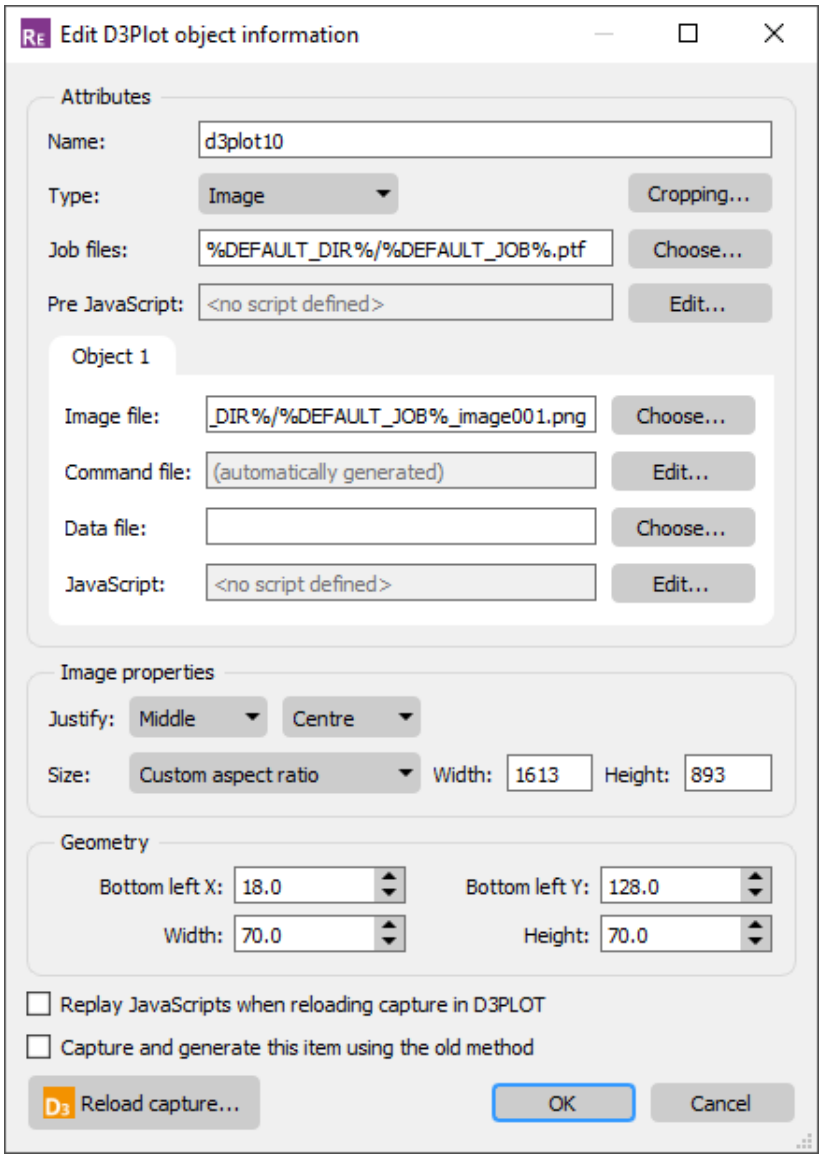
Once you are happy with the image and variables you have in D3PLOT, press the **Capture** button on the top bar of the target window. D3PLOT will automatically create a settings file, a properties file, a command file and a variables file for the current image and add them to the D3PLOT item in REPORTER. These are embedded in the template so you do not have to worry about packaging them with your template file.



To produce an animated GIF or MP4 movie instead of a static image, right-click on the Capture button and select one of the **GIF** or **Movie** options.
The settings used when capturing a GIF or MP4 (e.g. frame rate, image quality) will be taken from the Movies tab of the Images/Media Export panel in D3PLOT.

From version 17.0 onwards, D3PLOT is linked to REPORTER so you can continue working with both programs open. In earlier versions (and if capturing items using the old method), you would need to return to REPORTER using the D3PLOT **File** menu and select **=> Reporter** (which replaces the normal Exit command).

See 7. REPORTER Integration for more tips about how to make the most of D3PLOT linked to REPORTER.



Once a D3PLOT item has been captured, the **Job files** textbox reflects the models used in the capture. See Working with Variables for more information about how to make your D3PLOT item work for different models.

REPORTER automatically assigns a **Image file** or **Movie file** name for you. If required you can change this to whatever name you require. Note that the format of the file is taken from the extension you provide. By default D3PLOT will return a PNG (for **Image** type), a GIF (for **GIF** type), or an MP4 (for **Movie** type) to REPORTER. If you wanted to create a JPEG image, change the extension in the textbox to '*.jpg*'.

The Command file is greyed out as it has been automatically created by D3PLOT and does not need any editing. However, if you wanted to add some extra dialogue commands to be done in D3PLOT when generating the object you can use the **Edit...** button next to the **Command File** textbox to add/edit them.

You can also specify two JavaScripts to run when generating the D3PLOT object; a **Pre JavaScript** and a 'normal' **JavaScript**. To explain why there are two possible JavaScripts we need to consider the order that D3PLOT uses when creating the image. It is:

1. Read the ptf files
2. Run Pre JavaScript (if defined)
3. Read properties and settings files stored in REPORTER template
4. Run any extra Command file dialogue commands from REPORTER (if defined).
5. Run the 'normal' JavaScript (if defined)
6. Read external data file (if defined)

For virtually all cases the 'normal' JavaScript file in step 5 will do what you want. However if you use a JavaScript to create a user defined data component then this must be run before the properties and settings files are read (as that is where the data component for the plot is stored). In this case a **Pre JavaScript** has to be used.

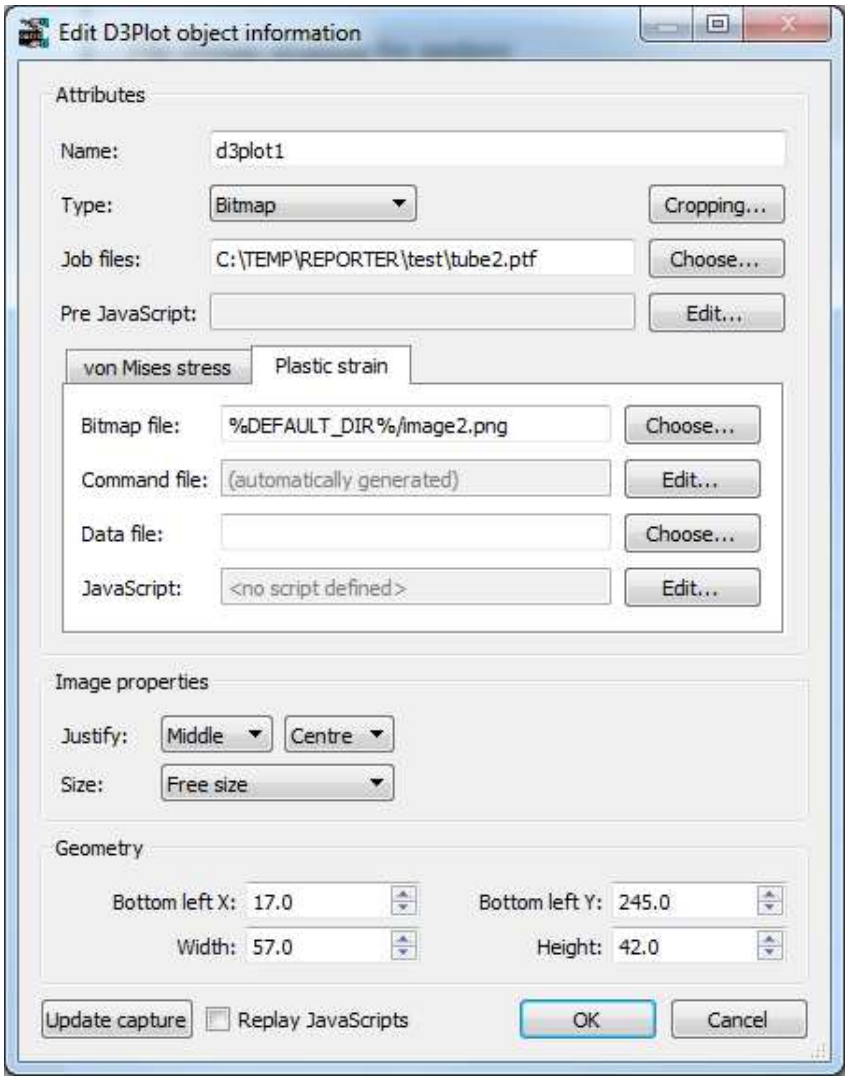
If you want to change the image or the variables you can press **Reload capture...** again at any time. D3PLOT will

start again and restore the current attributes you have set. You can make any changes that you want before pressing the **Capture** button as before. The old settings file, properties files and variables file will be overwritten.

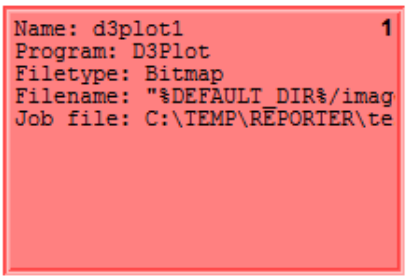
6.1.2 Creating multiple images from a single D3PLOT session

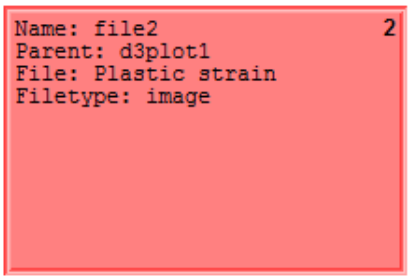
From version 17.0 onwards, you can quickly capture and generate multiple images from a single D3PLOT session as separate D3PLOT items. The old method of capturing D3PLOT items with multiple child images is no longer needed because its main purpose was to avoid having to launch D3PLOT multiple times to generate a single template. Now that D3PLOT and REPORTER are linked, all of the items in a template can be generated from a single session. The old method described below is only preserved to help keep old templates working.

You are not limited to making a single image in D3PLOT. Using the **Reporter Objects** floating menu you can capture as many images as you want in a single D3PLOT session. A tab will be created in the **Edit D3Plot object** window for each image you capture. For example as well as making a von Mises stress image we may also want to make an image showing plastic strain.



Each image has its own properties and settings file and optionally extra command files and/or a JavaScript. In REPORTER an **Image file** is created for the second and subsequent images and these are linked to the 'parent' D3PLOT object.





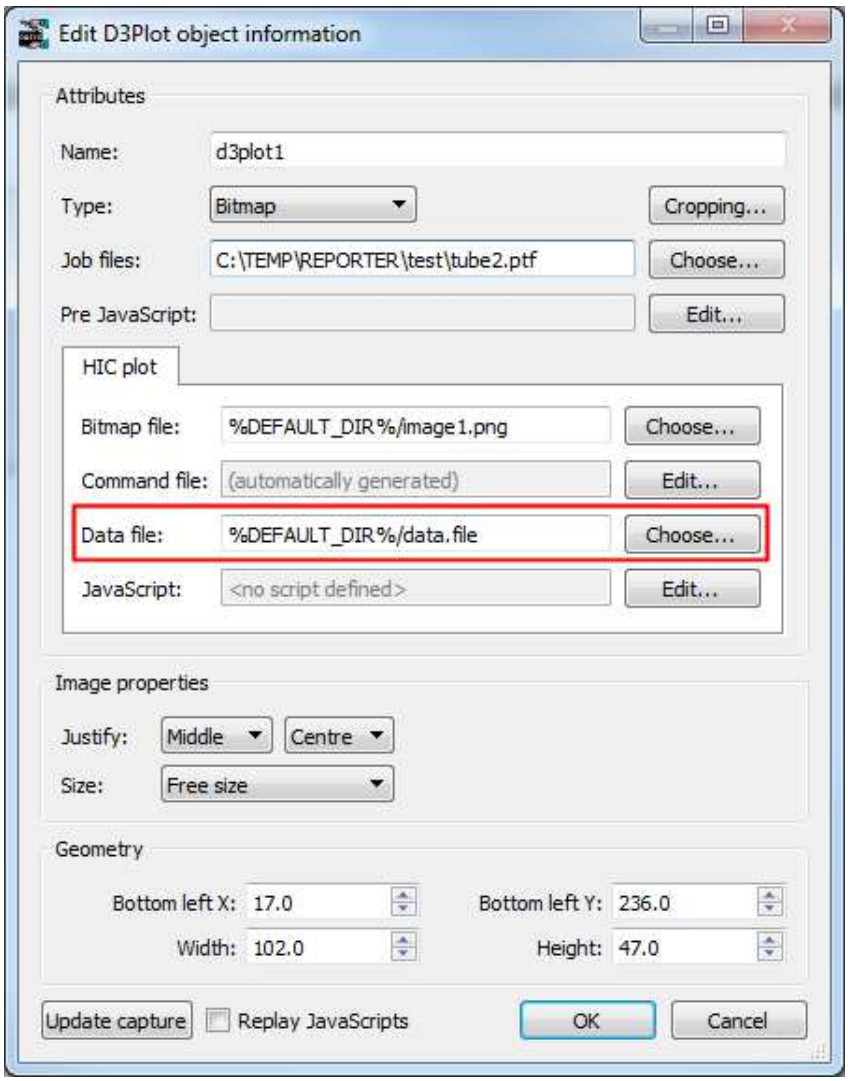
To help show which objects are linked together they are coloured differently to normal objects. The first group will be red, the second green, the third blue...

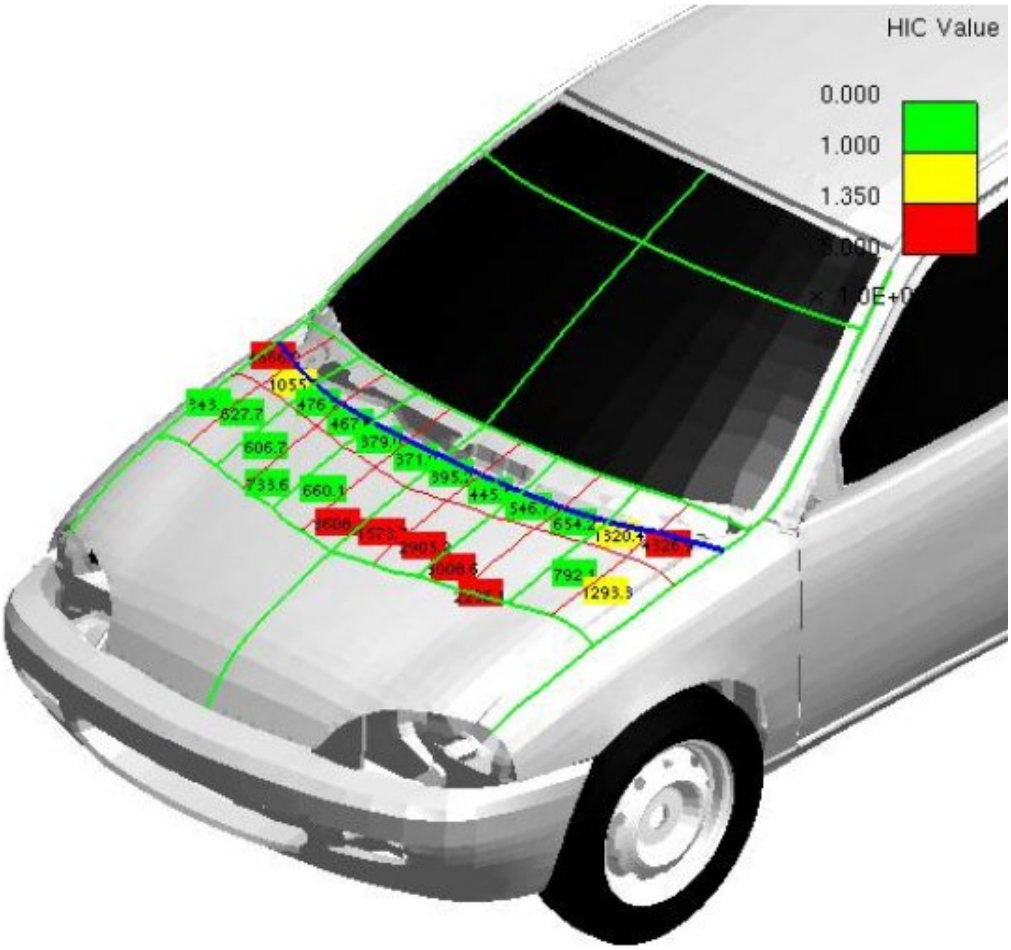
If you modify the 'parent' D3PLOT object the 'child' Image file objects will be added/updated/deleted as required.

6.1.3 Using datafiles to create 'blob' plots

If a **Data file** is given then that is passed to D3PLOT to create an external data plot or 'blob' plot. An example plot is shown below. In a data plot D3PLOT superimposes data values on the 3 dimensional shape. For example, below this is used to show HIC values for Euro-NCAP analyses at various positions on the bonnet.

The easiest way of creating a data file is to use the standard library program in REPORTER. See appendix B and the D3PLOT manual for more details.

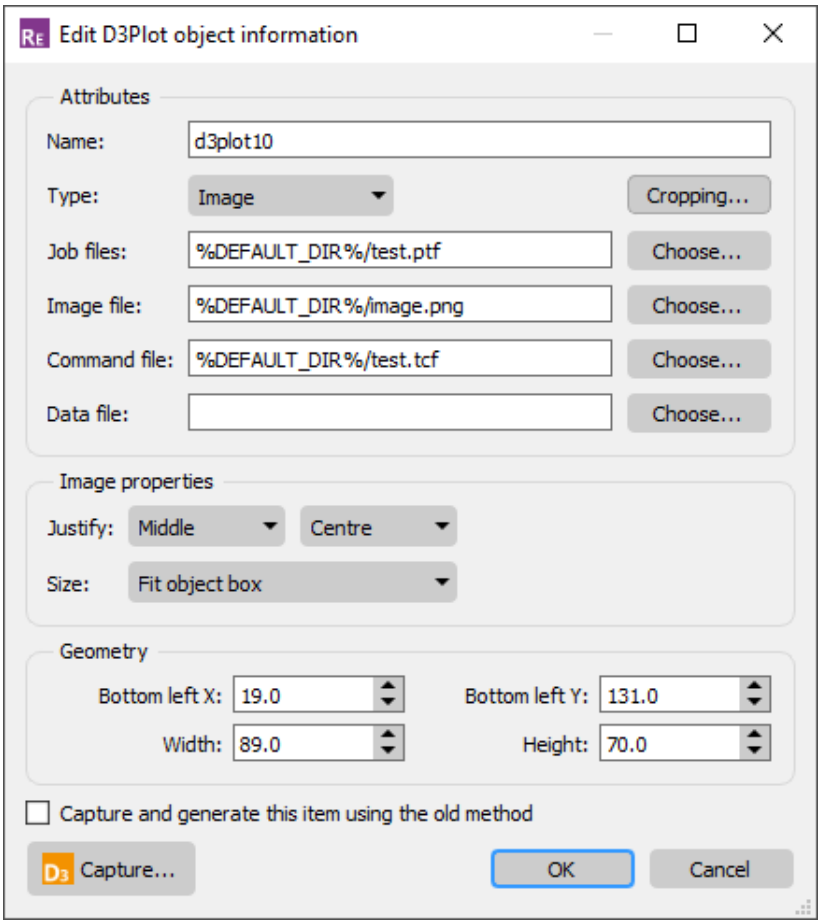



HIC Value
0.000
1.000
1.350
827.7
606.7
660.1
792.4
1293.3

6.1.4 Using a command file to create a D3PLOT object

A really old (and not-at-all-recommended) method to create a D3PLOT object is to create a command file yourself in D3PLOT (which creates the image). In this case the **Image file** must correspond to the name of the image you create in the command file. Give the name of the **Command file** and the **Job file**.

This method is not recommended and is present only to keep old templates working. Use the Capture method instead.



6.1.5 Editing D3PLOT objects

The position and size of D3PLOT objects can be edited in exactly the same way as the simple shape objects. See section 5 for more details.

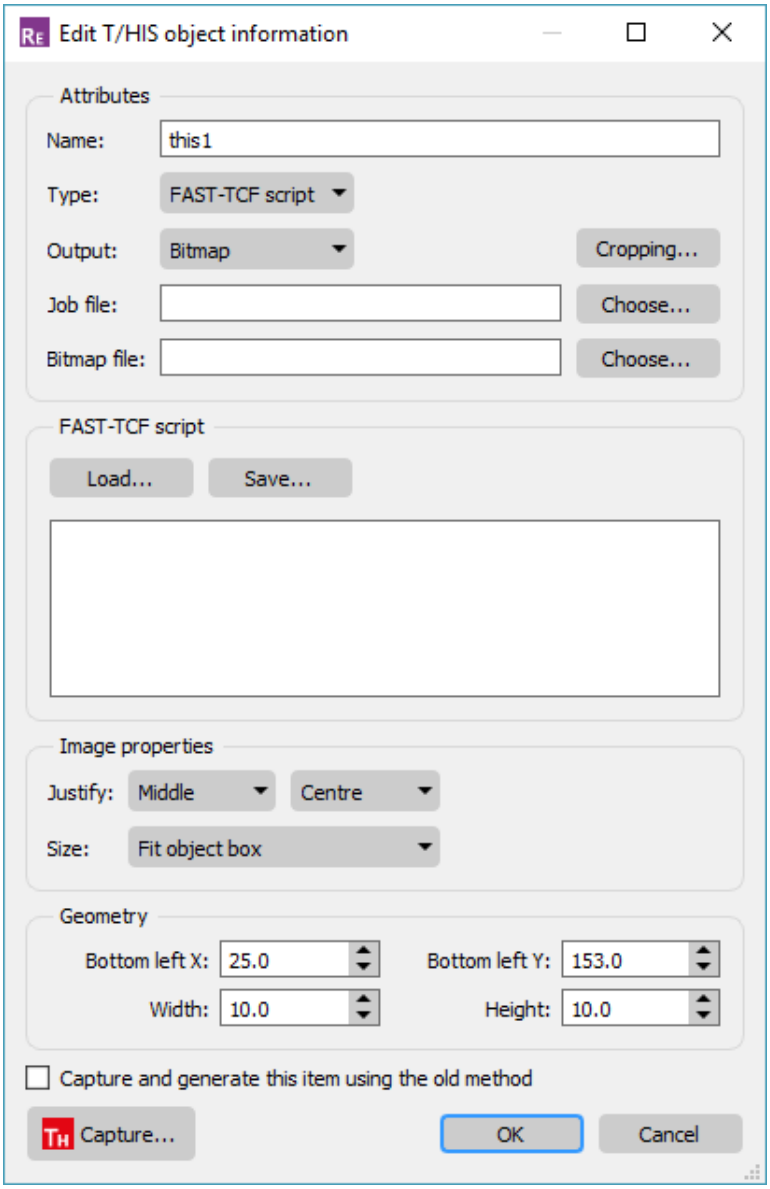
If you have created the D3PLOT object using **Capture...** then the text on the button will change to **Reload capture...** You can modify/update the existing captures if required. See section 6.1.1 for more details.

6.2 T/HIS objects

T/HIS objects allow you to include output from T/HIS in your template.

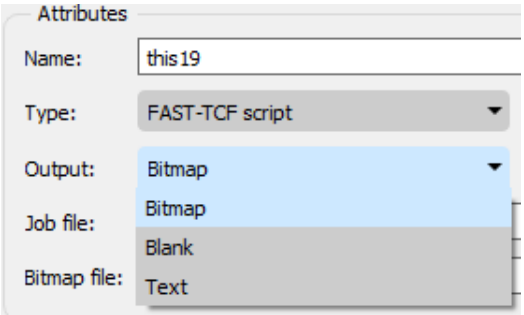
There are three different ways of using T/HIS objects. The first (and by far the easiest) is to use the **Capture...** button to create a FAST-TCF script for the object that T/HIS will run.
The second is to write your own FAST-TCF script.
The third is to use an existing T/HIS command file to create the output from T/HIS.

If the Bitmap output type is chosen, the **Cropping...** button can be used to crop away parts of the image from the top, bottom, left and right before showing it. See section 5.4.2 for more details.
The **Justify** buttons in the **Image properties** section allow you to change the justification of the image in the box on the page. REPORTER will not change the aspect ratio of the image. By default the image will be placed centrally in the box and enlarged as much as possible to still fit in the box. The **Justify** buttons can be used to alter the justification in the box if necessary.



There are three different options for the type of output generated from T/HIS.

- **Bitmap** indicates that the output is an image file.
- **Blank** indicates that the T/HIS object will not create any output on the page
- **Text** indicates that the output is text. This is only valid for the **FAST-TCF script** type. This option would be used if you wanted the output from a FAST-TCF table or HIC command etc.



From version 13.0 of REPORTER it is possible to specify the size or aspect ratio of the image created from T/HIS. This can be changed in the **Image properties** section. If you want to specify a particular aspect ratio or graphics size you can change this. The available options for **Size** are:

- Free size
- Fit object box
- 4:3 aspect ratio
- 16:9 aspect ratio
- 16:10 aspect ratio
- Custom aspect ratio

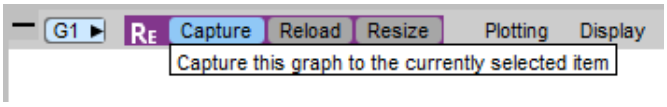
- Fixed size

The default option from version 17.0 onwards is **Fit object box**. With this option, T/HIS will capture an image using a graphics window with the same aspect ratio as the object's dimensions in REPORTER.

In previous versions of REPORTER, the default option was **Free size**. With this option, the size of the graphics window is calculated by T/HIS. The actual size will depend on what resolution the monitor is and what scale factors you have chosen for the user interface. This can cause problems if the template is created on one type of display (e.g. a 16:9 monitor) but played back on a different ratio monito (e.g. a 4:3 ratio monitor) as the image size can change and so the output from REPORTER can look different. To make output consistent you can use the other options:

Fit object box will make T/HIS create a graphics window that has the same aspect ratio as the object box dragged in REPORTER.
4:3 aspect ratio, **16:9 aspect rati**o and **16:10 aspect ratio** will make T/HIS create a window with the specified aspect ratio.
Custom aspect ratio or **Fixed size** allow you specify a custom width and height. **Fixed size** will make T/HIS create an image with the specified width and height. **Custom aspect ratio** will make T/HIS create the largest image it can with the aspect ratio width:height.

6.2.1 Using Capture to create a T/HIS object



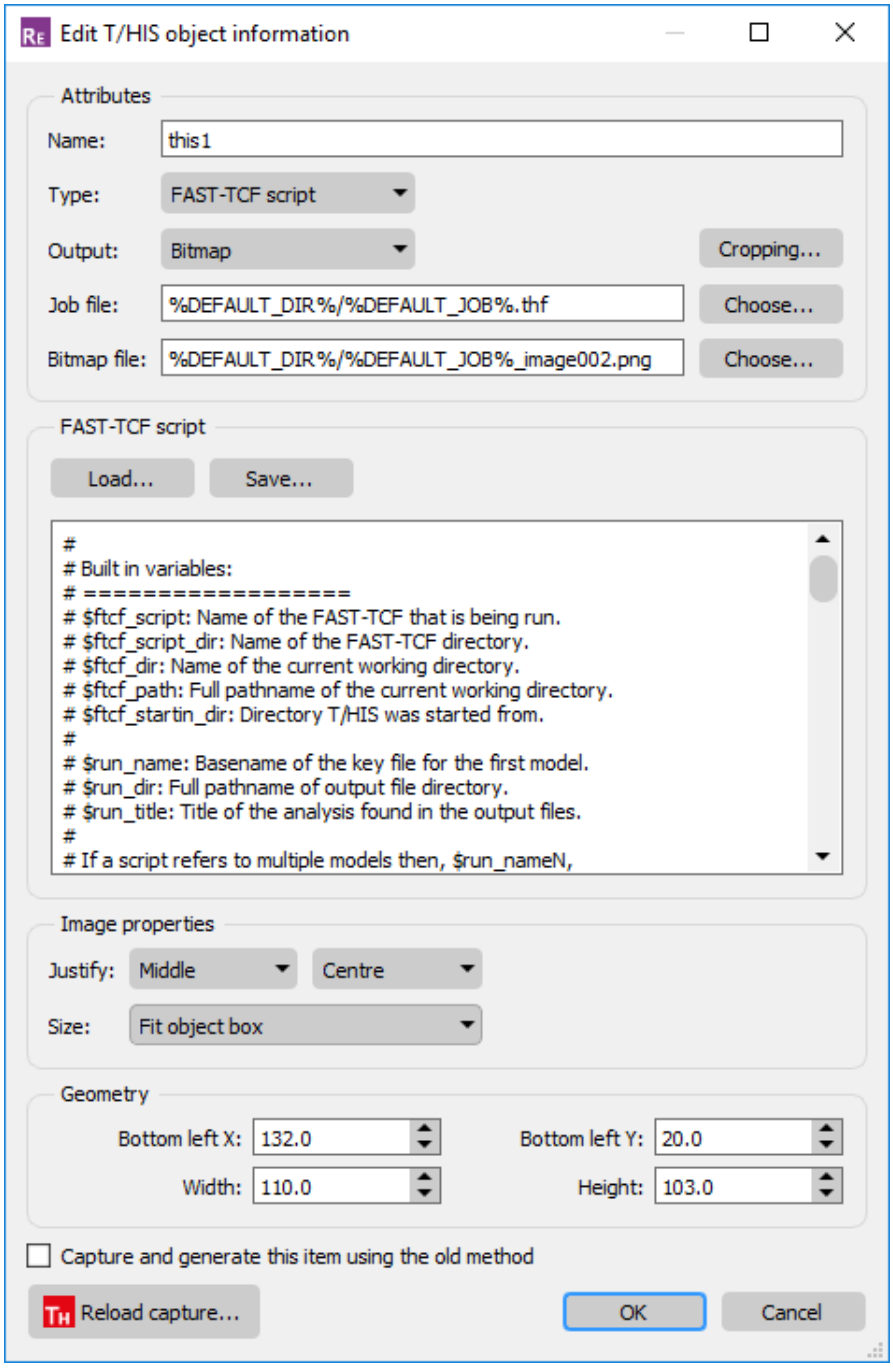
The easiest way to create a T/HIS object is to use the **Capture...** command.
First make sure that the **Type** is set to **FAST-TCF script**.
If you press the **Capture...** button REPORTER starts T/HIS for you if it is not already linked. You can now open the model(s) and do whatever operations you want inside T/HIS to get the cruves that you want on the screen. Once you are happy with the graph you have in T/HIS, press the **Capture** button on the top bar of the target window. T/HIS will automatically create a FAST-TCF script for the current graph and return it to REPORTER. This is embedded in the template so you do not have to worry about packaging it with your template file.

From version 17.0 onwards, T/HIS is linked to REPORTER so you can continue working with both programs open. In earlier versions (and if capturing items using the old method), you would need to return to REPORTER using the T/HIS **File** menu and select **Return to Reporter** (which replaces the normal Exit command).

See 7. REPORTER Integration for more tips about how to make the most of T/HIS linked to REPORTER.

Once a T/HIS item has been captured, the **Job file** textbox reflects the models used in the capture. See Working with Variables for more information about how to make your T/HIS item work for different models.

If you want to change the script you can press **Reload capture...** again at any time. T/HIS will start again and replay the FAST-TCF script. You can make any changes that you want before pressing the **Capture** button as before. The old FAST-TCF script will be overwritten.



6.2.2 Using your own FAST-TCF script to create a T/HIS object

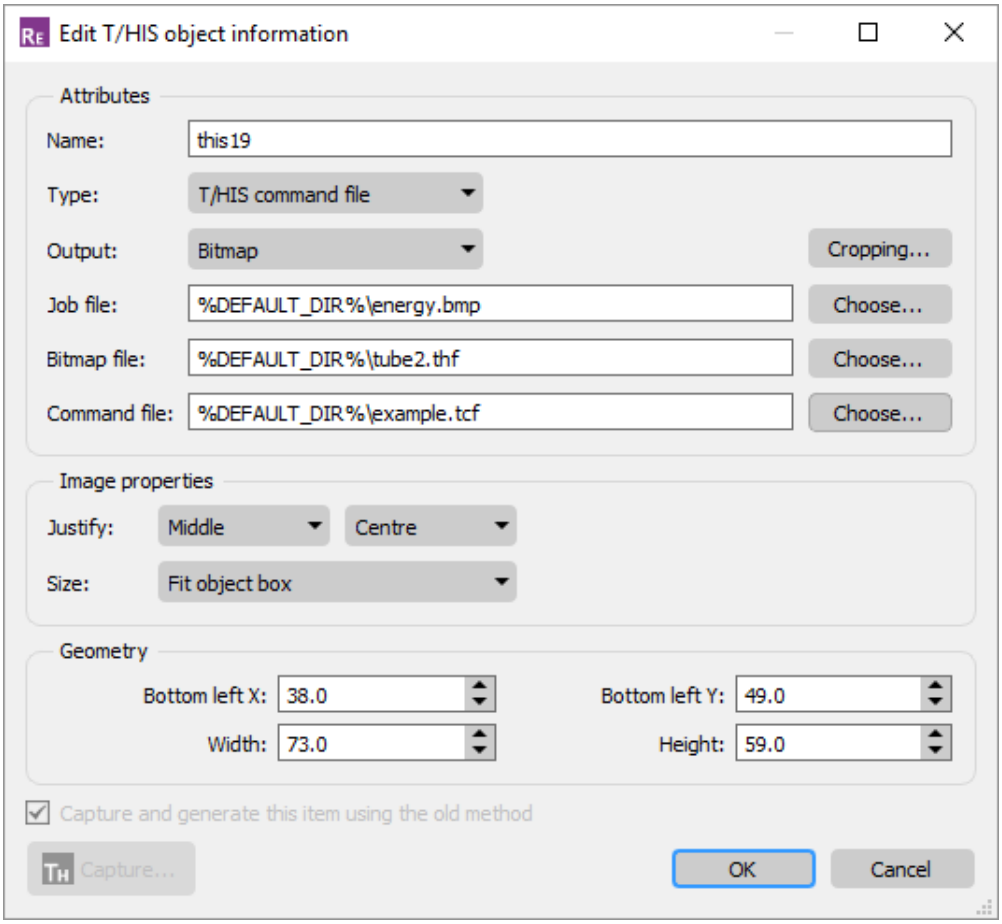
If you want to make your own FAST-TCF script in a T/HIS object then fill in the **Image file** and **Job file** yourself. You can load an existing FAST-TCF script by using the **Load...** button or type in the script. In this case the **Image file** must correspond to the name of the image you create in the script.

6.2.3 Using a command file to create a T/HIS object

The alternative method to create a T/HIS object is to create a command file yourself in T/HIS (which creates the image).

Make sure that **Type** is set to **T/HIS command file.**

In this case the **Image file** must correspond to the name of the image you create in the command file. Give the name of the **Command file** and the **Job file**.



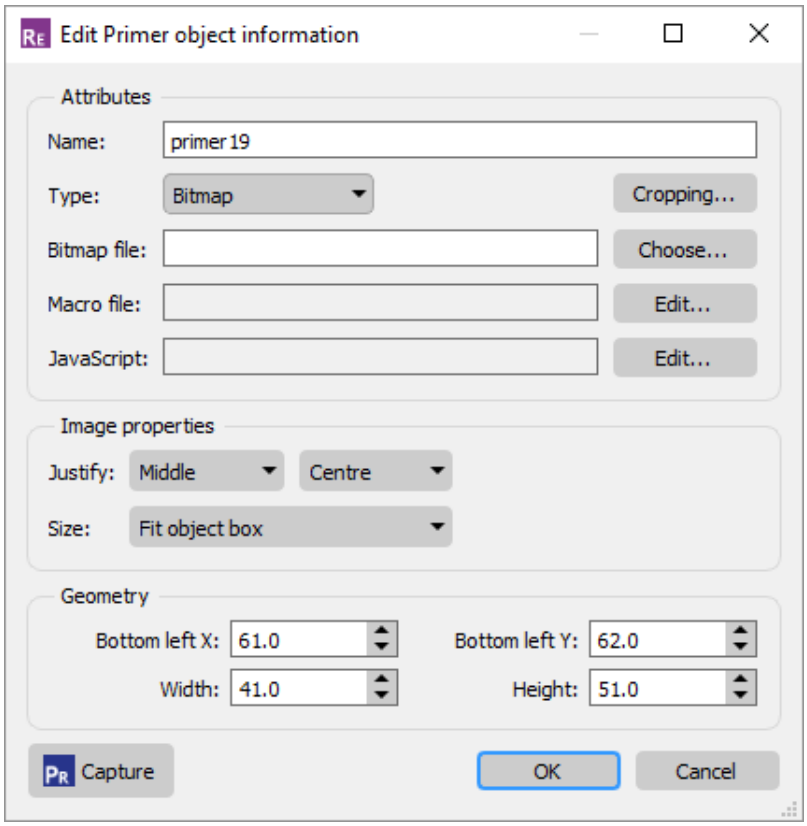
6.2.4 Editing T/HIS objects

T/HIS objects can be edited in exactly the same way as the simple shape objects. See section 5 for more details.

If you have created the object using the **Capture...** then the text on the button will change to **Reload capture...** You can modify/update the existing capture if required. See section 6.2.1 for more details.

6.3 PRIMER objects

PRIMER objects allow you to include output from PRIMER in your template. To create one select the Primer tool from the **Tools toolbar** and click and drag a rectangle on the page. The **Edit Primer object information window** will then be shown.



If you want to create an image using PRIMER to put in the report (e.g. an image showing yield stress or element timestep) select **Bitmap** for the **Type**. In this case the **Cropping...** button can be used to crop away parts of the image from the top, bottom, left and right before showing it (see section 5.4.2 for more details).
The **Image properties** section allows you to change the justification of the image in the box on the page. REPORTER will not change the aspect ratio of the image. By default the image will be placed centrally in the box and enlarged as much as possible to still fit in the box. The **Justify** buttons can be used to alter the justification in the box if necessary.

Alternatively if you do not want any output but just want to run PRIMER to create some other sort of output or run a JavaScript set the **Type** to **Blank**.

6.3.1 Using Capture to create a PRIMER object

From version 13.0 of REPORTER it is possible to specify the size or aspect ratio of the image created from PRIMER. This can be changed in the **Image properties** section. If you want to specify a particular aspect ratio or graphics size you can change this. The available options for **Size** are:

- Free size
- Fit object box
- 4:3 aspect ratio
- 16:9 aspect ratio
- 16:10 aspect ratio
- Custom aspect ratio
- Fixed size

The default option from version 17.0 onwards is **Fit object box**. With this option, PRIMER will capture an image using a graphics window with the same aspect ratio as the object's dimensions in REPORTER.

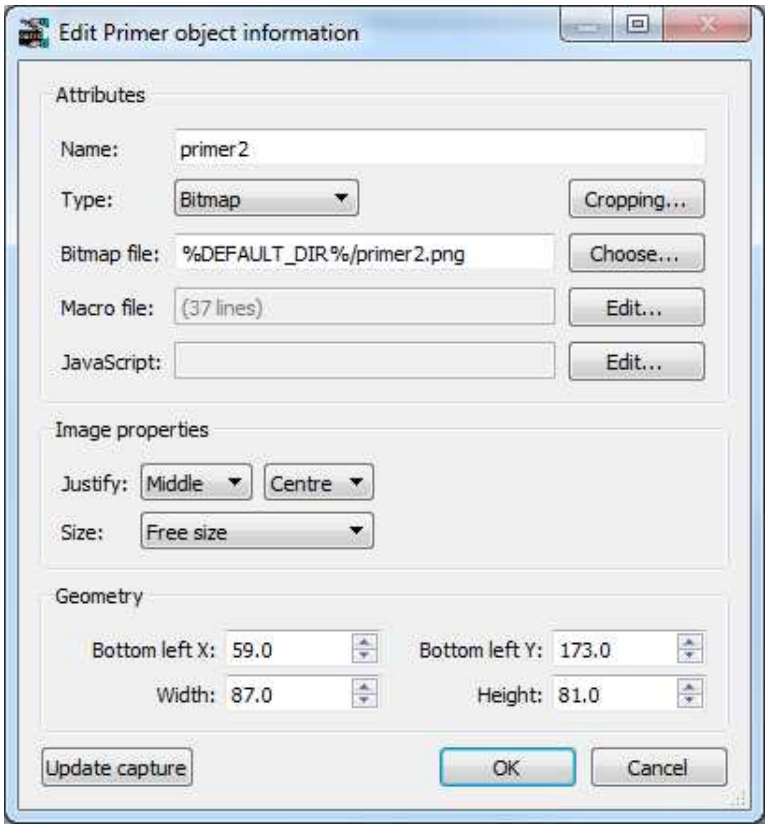
In previous versions of REPORTER, the default option was **Free size**. With this option, the size of the graphics window is calculated by PRIMER. The actual size will depend on what resolution the monitor is and what scale factors you have chosen for the user interface. This can cause problems if the template is created on one type of display (e.g. a 16:9 monitor) but played back on a different ratio monito (e.g. a 4:3 ratio monitor) as the image size can change and so the output from REPORTER can look different. To make output consistent you can use the other options:

Fit object box will make PRIMER create a graphics window that has the same aspect ratio as the object box dragged in REPORTER.
4:3 aspect ratio, **16:9 aspect rati**o and **16:10 aspect ratio** will make PRIMER create a window with the specified aspect ratio.
Custom aspect ratio or **Fixed size** allow you specify a custom width and height. **Fixed size** will make PRIMER create an image with the specified width and height. **Custom aspect ratio** will make PRIMER create the largest

image it can with the aspect ratio width:height.

If you want to specify an image size it should be specified before the capture is done in PRIMER.

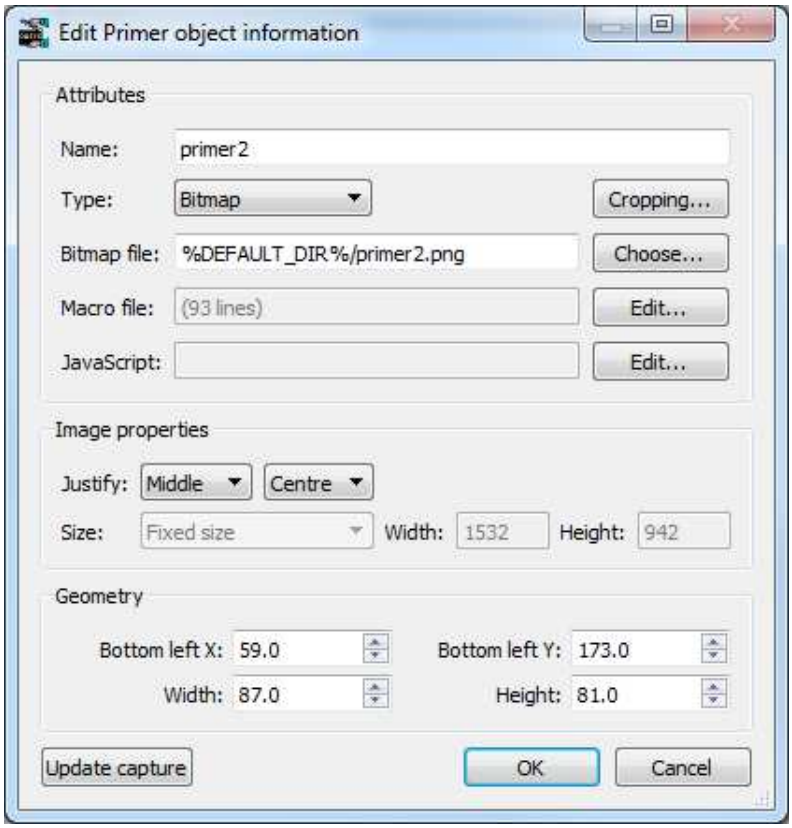
To start PRIMER press the **Capture** button. PRIMER will then automatically record a macro containing all of the commands that you do. When you have finished do **File** and **=> Reporter** to return to REPORTER.
REPORTER will then prompt you to replace any filenames in the macro with variables. You can choose which variables you would like to replace. Alternatively you can replace any variables yourself manually later on (see below).
The **Edit Primer object information window** will then be updated as shown below.



REPORTER will automatically give a name for the bitmap file but you can change it to whatever you want. If required you can edit the macro by using the **Edit...** button next to the **Macro file** textbox (which in the above image shows that it contains 25 lines). This is useful to replace any filenames with variables if required (right click with the mouse or press Ctrl+I in the macro to insert variables). The macro will be saved in the REPORTER template.

As well as using a macro a JavaScript can also be specified to run in PRIMER. The **Edit...** button next to the **JavaScript** textbox can be used to load and edit a JavaScript. The JavaScript will be saved in the REPORTER template.

If a macro contains picking, dragging or dynamic viewing commands then PRIMER needs to maintain the aspect ratio of the graphics window so that they can be replayed correctly. If it dd not do this then the pick command would pick at a different location and it may not work correctly.
When a macro is recorded REPORTER scans the macro for any picking, dragging or dynamic viewing commands. If the macro does not contain any then the image size can be changed after capturing if required.
For example in the image above a macro has been recorded but as it does not contain any picking, dragging or dynamic viewing commands the **Size** can be changed from **Free size** if required.
If the macro does contain picking, dragging or dynamic viewing commands then REPORTER will change the image size to a fixed size and not allow it to be chnaged, to ensure that the macro will replay correctly.
For example in the image below a macro has been recorded but and it as does contain picking, dragging or dynamic viewing commands the **Size** has been changed to **Fixed size** and cannot be changed.



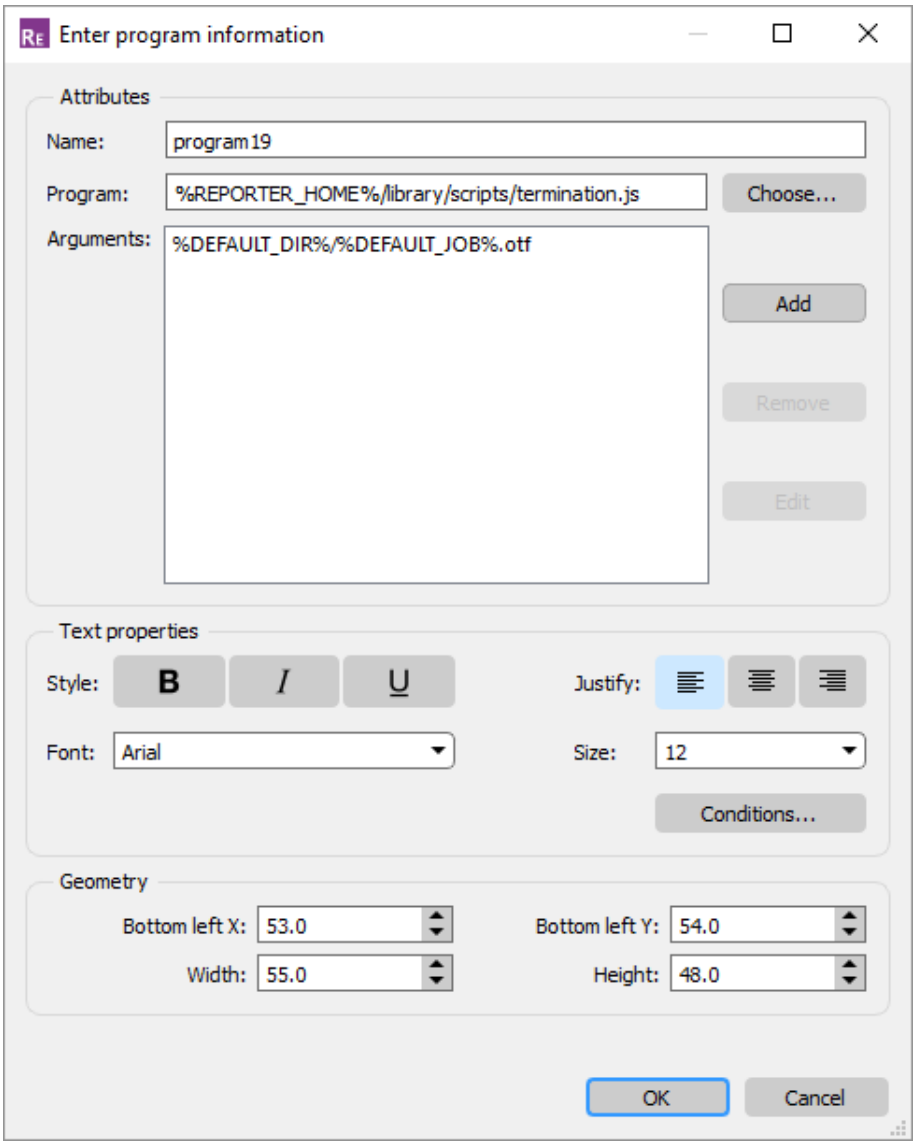
6.3.2 Editing PRIMER objects

PRIMER objects can be edited in exactly the same way as the simple shape objects. See section 5 for more details.

If you want to modify an existing capture you can use **Update capture**. PRIMER will restart and replay the macro you have recorded. Any new commands that you do will then get appended to the macro.

6.4 Program objects

6.4.1 Text output from a program



This option allow you to specify a program from which the text that would normally outputted to the standard output will be inserted into the report by REPORTER when the report is finally generated. The program can be written in anything you want: C, Fortran etc,a scripting language such as Perl or Python, a shell script on unix, a batch file on windows etc. All that matters is that output which would normally be directed to stdout is captured by REPORTER. For more details on writing programs for REPORTER please see Appendix E.

The filename of the program/script is entered in the **Program:** text box or clicking on the **Choose...** button will bring up a **File** window from which to select the program/script. You can also enter variables by right clicking in the text box which will allow you to bring up a **Insert variables**window from which to select a variable. The various text parameters such as font and size can also be set.

The text parameters such as font, justification, size etc can be set for the text that will be captured from the program.

If the program needs arguments then any number can be added by using the **Add** button.

The **Conditions...** button (see section11) enables the user to apply conditional formatting to the text from the program.

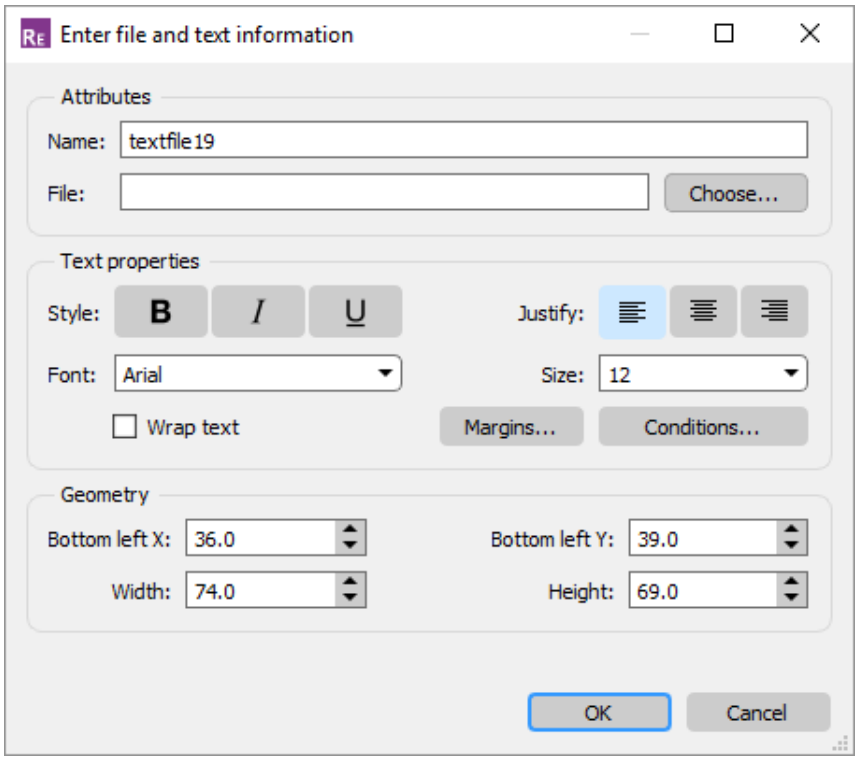
The **OK** button will exit this window and add the new program to the template. The **Cancel** button will exit this window without adding anything to the report

6.4.2 Editing program objects

Program objects can be edited in exactly the same way as the simple shape objects. See section 5 for more details.

6.5 File objects

6.5.1 Text files



To insert text from a file, select the **File Text** from the **Insert** menu.
The **Choose...** button allows the user to select the file by browsing the computer. The positioning and style of the text can be changed.
The **OK** button will exit this screen and create the object/save the changes made.
The **Cancel** button will exit this screen without creating the object/saving the changes.

The text parameters such as font, justification, size etc can be set for the text that will be read from the file.

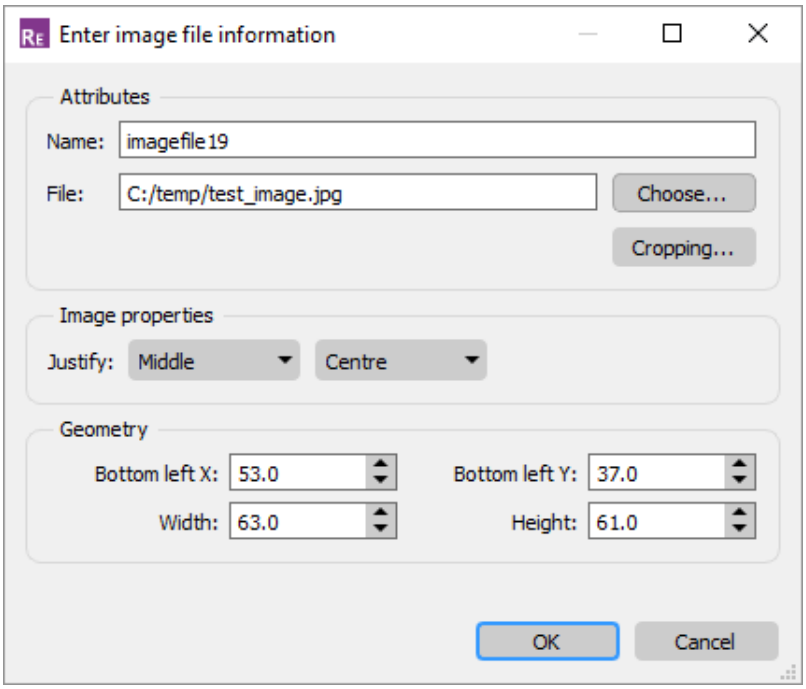
The text, background and fill colour and the border line style can be set using the style toolbar. See section 5.2 for more details.

The margins for the textbox can be changed by using the **Margins...** button.

The **Conditions...** button (see section11) enables the user to apply conditional formatting to the text.

By default text is not wrapped so long lines will be clipped to the width of the object. If you want text to be wrapped onto multiple lines use the **Wrap text** checkbox.

6.5.2 Image files



To insert an image from a file, select the **File Image** from the **Insert** menu or use the Image file tool from the Tools toolbar.
The **Choose...** button allows the user to select the file by browsing the computer.
The **Cropping...** button can be used to crop away parts of the image from the top, bottom, left and right before showing it. See section 5.4.2 for more details.

The **Image properties** section allows you to change the justification of the image in the box on the page. REPORTER will not change the aspect ratio of the image. By default the image will be placed centrally in the box and enlarged as much as possible to still fit in the box. The **Justify** buttons can be used to alter the justification in the box if necessary.

The positioning of the image on the page can be changed by using the **Geometry** section.

The **OK** button will exit this screen and create the object/save the changes made.
The **Cancel** button will exit this screen without creating the object/saving the changes.

Animated Image files

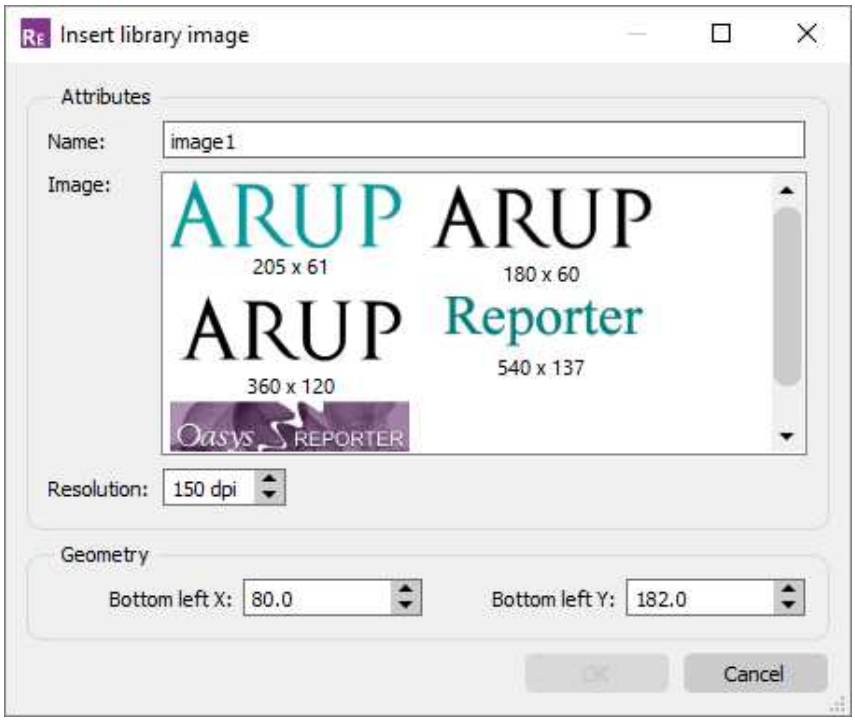
Similarly to Image Items, starting with version 18.0, Image File Items also support GIF animations and MP4 movies.

6.6 Library objects

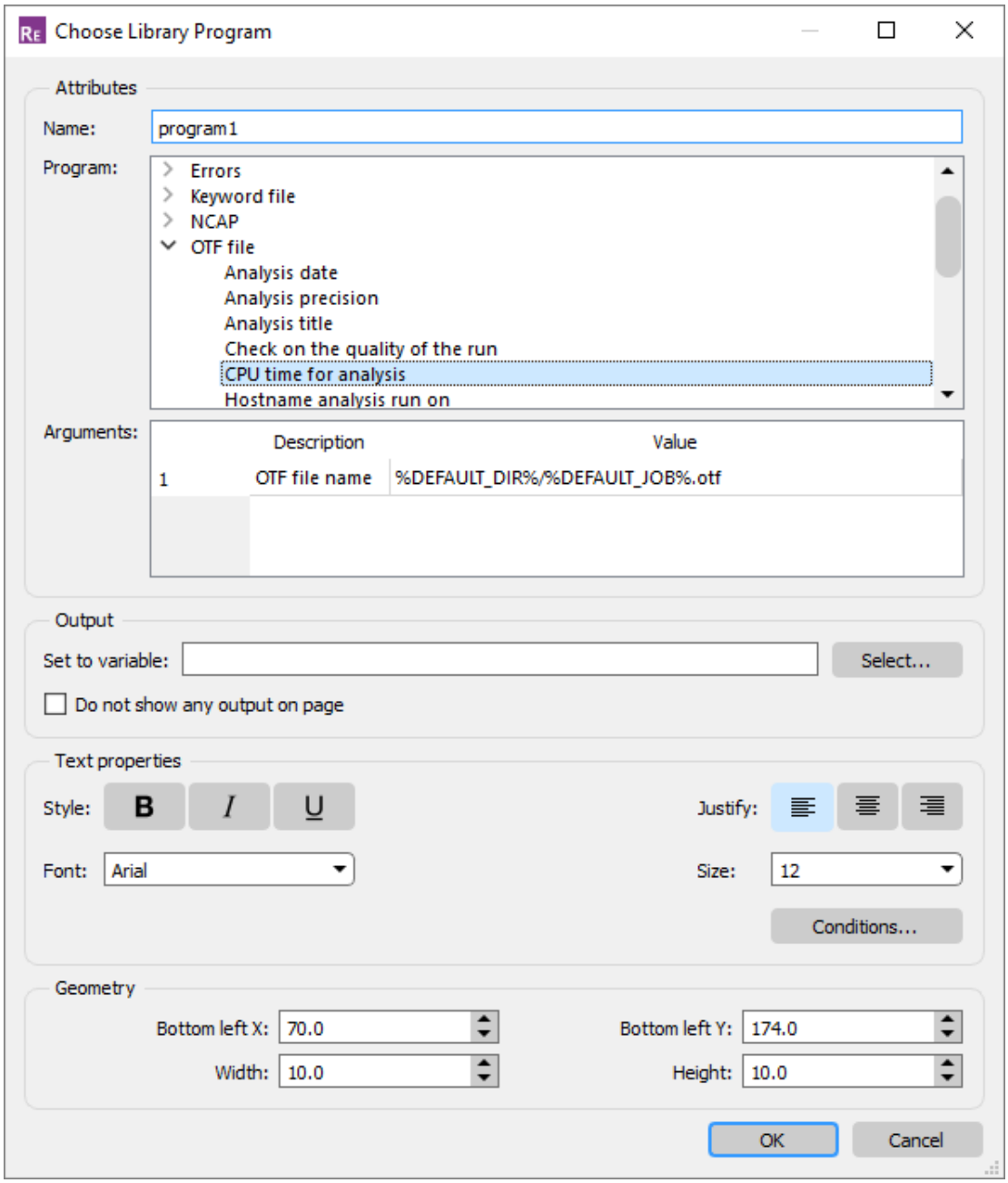
6.6.1 Library images

This option allows the user to view and select an image from the selection held in the image library. The resolution and positioning of the image can also be set. The **OK** button will exit this windows and add the new object to the template. The **Cancel** button will exit this window without adding anything to the report.

See appendix B.3 'Standard library images' - to insert library images.



6.6.2 Library program/script



This option allows you to specify a program/script from the library, the output of which will be inserted into the report by REPORTER when the report is finally generated. (See the library object appendix for more details about using the library)

Once you have selected this option you need to click and drag to create an area in the report where the output is to appear. Then the relevant **Insert** window will be brought up.

From this window you can select the program/script you want from the program list by clicking on it with the mouse. Depending on the program/script a number of argument boxes may appear into which you need to specify any arguments required by the script. By right clicking or pressing **Ctrl+I** in these you can bring up a **Insert variables** window from which to select a variable to use for the argument.

The output from the program/script can be set to a variable using the **Set to variable** input box or **Select** button. Additionally you can specify that the output from the program is not shown on the page by using the **Do not show any output on page** option. This could be useful if you want to run the program to get the output as a variable but use it later in the template (for example in a table) rather than having any output here.

The font properties can be set using the **Text properties** section. The text colour will be set to the current text colour

setting.
The **Conditions...** button (see section11) enables the user to apply conditional formatting to the text.

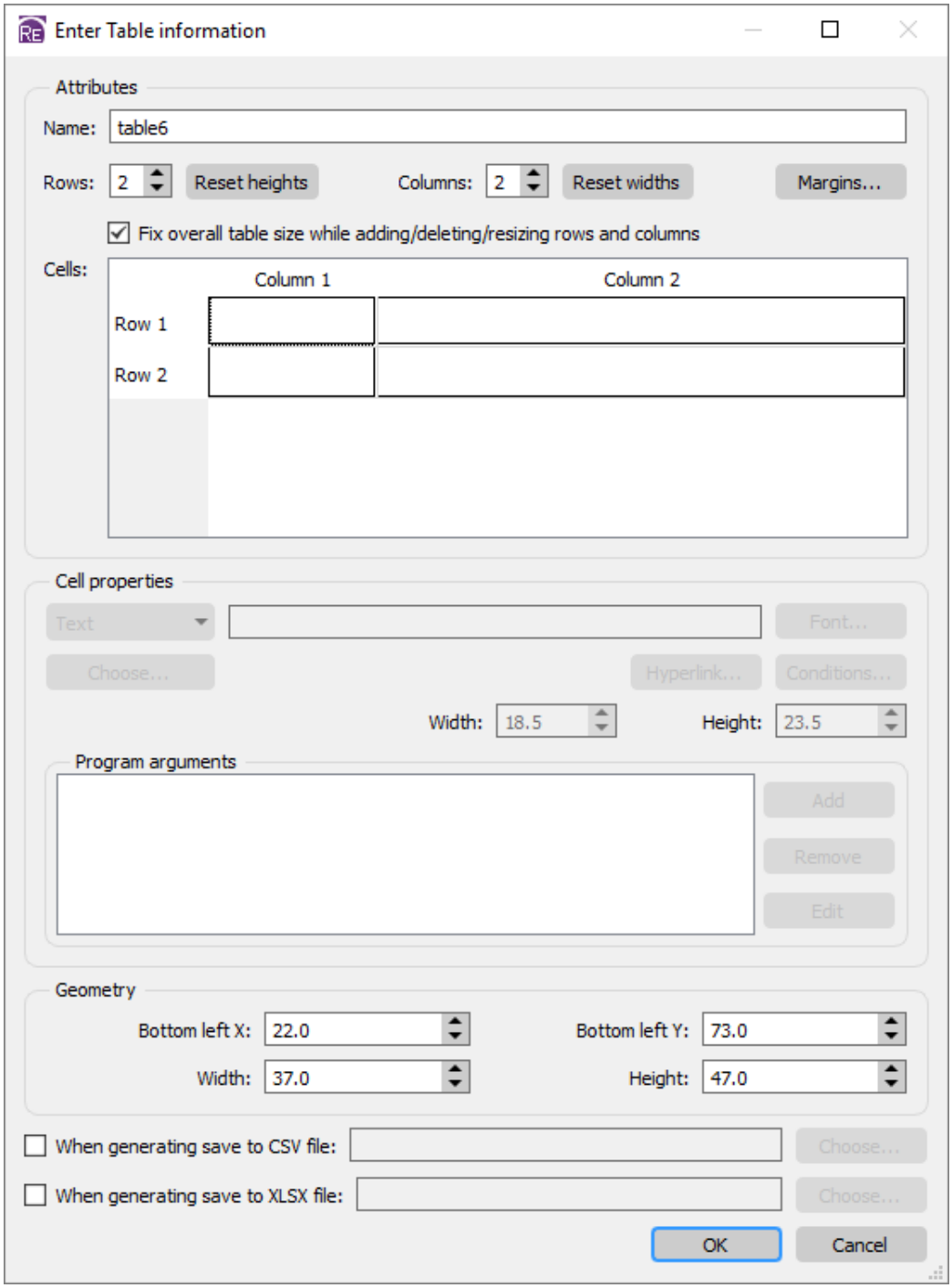
The **OK** button will exit this windows and add the new object to the template. The **Cancel** button will exit this window without adding anything to the report.

6.6.3 Editing library objects

Library objects can be edited in exactly the same way as the simple shape objects. See section 5 for more details.

6.7 Table objects

A table allows you to easily line things up on a page in REPORTER. To create a table drag the area on the page that you want to be a table. The following menu is then mapped.



6.7.1 Changing the number of rows or columns in the table

By default a table will have 2 rows and 2 columns and initially each cell in the table will be blank. The number of rows and/or columns can changed using the **Rows** and **Columns** spin boxes in the **Attributes** section. As the values are changed the **Cells** section in the menu will be updated accordingly.

Alternatively a row or column can be added or deleted at any position in the table by right clicking on a cell or header in the **Cells** section and using the Insert or Delete options in the context menu.

6.7.2 Using the 'Fix overall table size...' checkbox

By ticking the 'Fix overall table size while adding/deleting/resizing rows and columns' checkbox in the **Attributes** section, the overall table size remains fixed no matter what cell/row/column operations are performed. E.g. if the checkbox is ticked and a row is added to the table using the spinbox, the height of all others rows are reduced (scaling proportionally) to maintain the overall table height. The height of the newly added row is equal to that of the adjacent row.

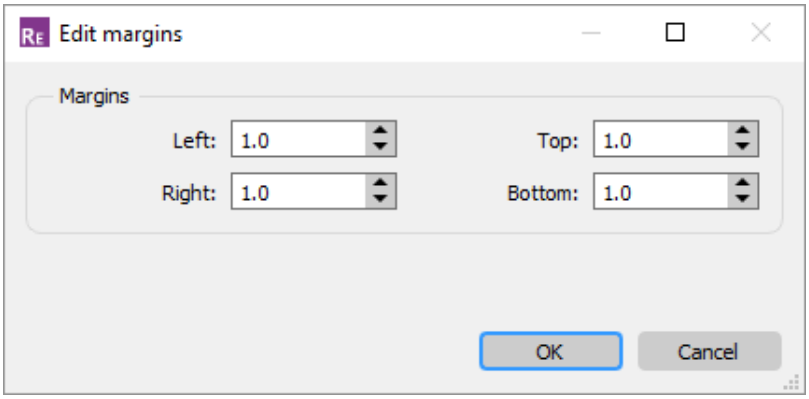
When the checkbox is unticked, cell/row/column operations are able to change the overall table size. E.g. if the checkbox is unticked and a row is added to the table using the spinbox, the height of all other rows are unchanged. The height of the newly added row is equal to that of the adjacent row and the overall height of the table increases by this amount.

The checkbox is ticked by default when first creating a table (such that the table remains within the bounding box drawn on the page), and unticked by default when editing an existing table.

It is important to note that this checkbox affects operations as they are performed within the edit window, not at the point at which 'OK' is pressed. Ticking or unticking the checkbox should therefore occur prior to any cell/row/column operations to obtain the desired functionality.

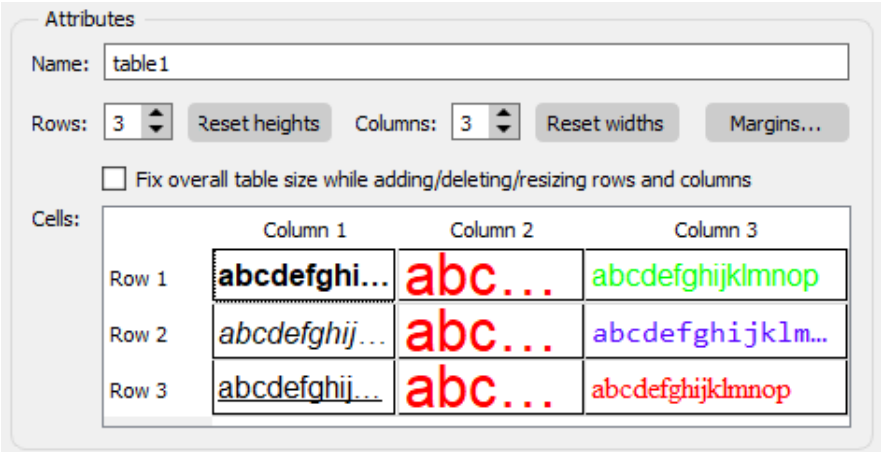
6.7.3 Changing the margins for cells in the table

The margins for the cells in the table can be changed using the margins button in the **Attributes** section.

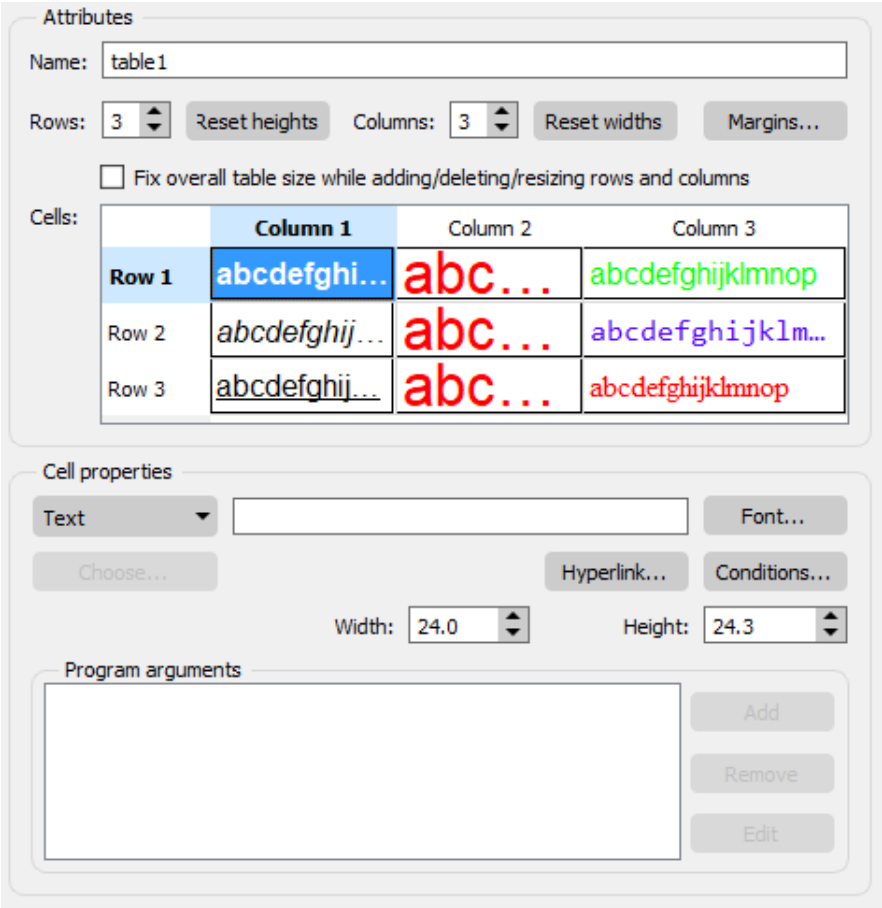


6.7.4 Seeing what is in each cell

The attributes section of the menu shows a simplified view of the table in a spreadsheet form in the **Cells** section. Cells which have text present in them are shown using the correct font, styling, font colour and size so you can quickly see you have the correct settings.



6.7.5 Changing cells

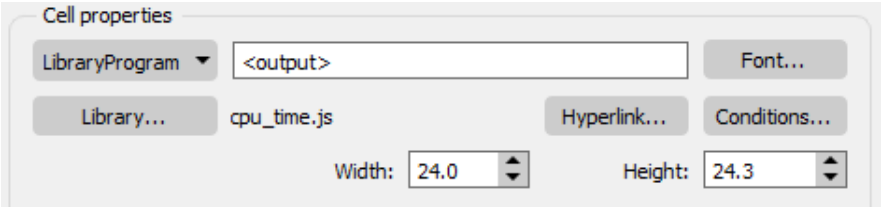


To change a cell (or cells) click on the cell in the simplfified view (or multiple select using Shift and/or Ctrl). The selected cells are highlighted in the simplified view and the **Cell properties** section of the menu becomes active.

The font can be changed with the **Font...** button and hyperlinks and conditional formatting applied to the cell text using the **Hyperlink...** and **Conditions...** buttons.

By default all cells will have the same with and height but you can use the **Width** and **Height** spinboxes to alter the width of this cell (and hence the width of all cells in the same column) and the height(and hence the height of all cells in the same row). To reset widths and/or heights back to be the same use the **Reset heights** and **Reset widths** buttons in the **Attributes** section.

Instead of just using text in the generated data you can run a program instead which could be a standard library program or an external program. In this case the output from the program will be put in the table cell instead. To use a program change the Cell type from **Text** to **Program** using the popup. Once this is done the **Choose...** and **Library...** buttons and the **Program arguments** section become active. For library programs the output from the script can be mixed with other text.



By default the cell text is shown as `<output>`. This will be replaced by the output from the script. Additionally you can prepend or append text to this to add to the cell.

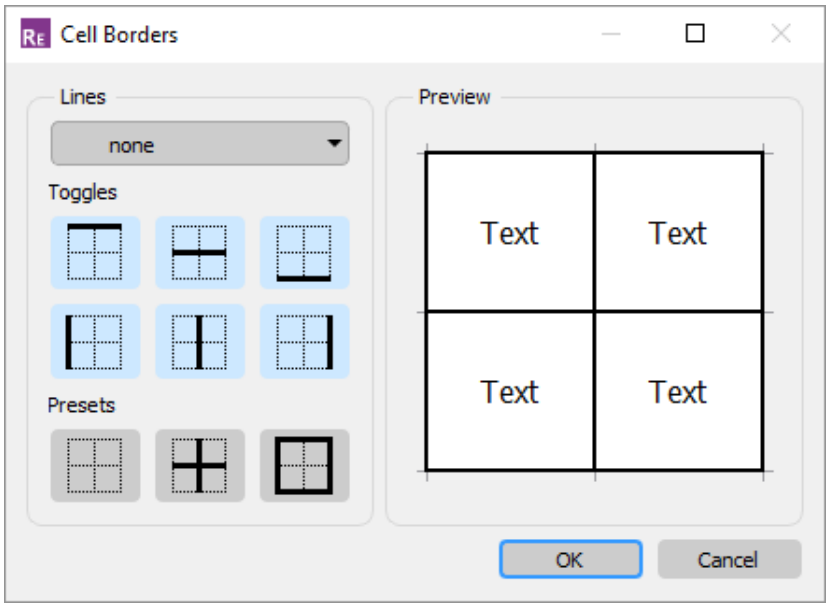
6.7.6 Merging cells

Cells in a table can be merged together into a single cell. Select the cells that you want to merge by either clicking and dragging the cells to merge or by Shift or Ctrl clicking on multiple cells in the **Cells** section. Note that the cells you want to merge must be a rectangular selection. Right click on the selected cells in the **Cells** section and choose **Merge cells** from the context menu.

Cells that have been merged together can be unmerged by selecting the cell, right clicking and choosing **Unmerge cells** from the context menu.

6.7.7 Cell borders

To change the border for all the cells in a table you can use the normal line thickness control in the Style toolbar. The borders for individual cells can also be changed if required. Select the cells that you want to change the border for in the **Cells** section, right click on them and select **Edit borders** from the context menu. This displays the **Cell Borders** menu.

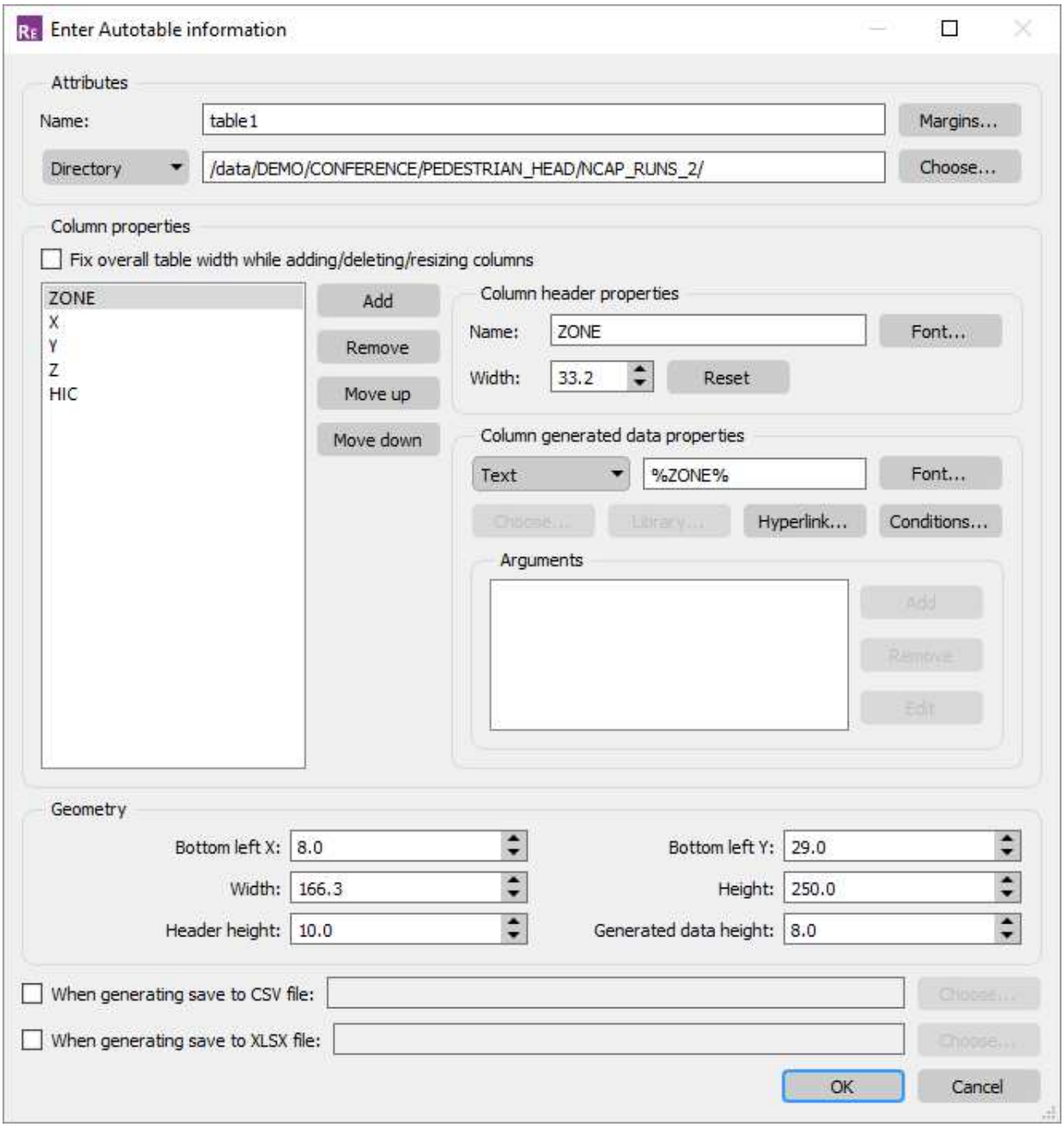


This menu allows you to change the cell borders for each side of the cells individually. Firsdt set the line thickness using the **Lines** combobox. The **Toggles** and **Presets** can then be used to change the borders. Press **OK** to update the cell borders.

6.7.8 Saving to CSV or XLSX

To save the contents of a generated table to a CSV or XLSX file (e.g. for use in Excel), use the 'When generating save to CSV/XLSX file:' checkboxes at the bottom of the table menu. Ticking these checkboxes activates the adjacent text windows, into which a save location can be manually entered (or selected using the 'Choose' buttons).

6.8 Autotable objects



An autotable object in REPORTER is a table which REPORTER will create when the report is generated. An an example, you may want to run multiple analyses and produce a summary table with one line in a table for each analysis. The autotable object allows you to do this.

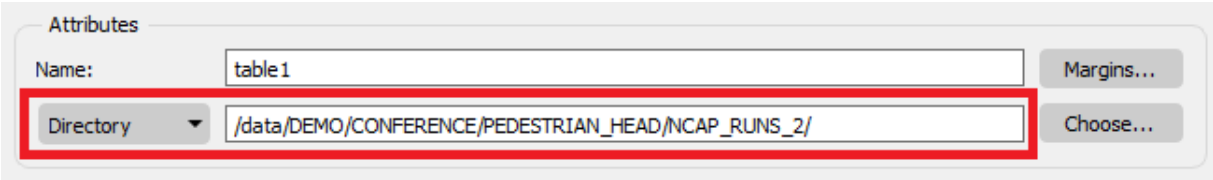
The above image shows the menu to create a table for a set of pedestrian headform analyses. We want to create a table with 5 columns (as shown below) ; the impact zone, the x, y, and z impact points and the calculated HIC.

ZONE	X	Y	Z	HIC
C1A	899.984	1393.17	895.182	4666.223
C1B	841.037	1276.24	896.854	1055.947
C1C	694.404	1399.28	851.726	343.4052
C1D	703.138	1308.79	861.869	627.7126
C2A	804.945	1171.9	898.937	476.1642
C2B	788.008	1057.62	903.647	467.8154

To do this we would run each of the analyses and post-process them with a REPORTER template. Each analysis would calculate the ZONE, X, Y, Z and HIC and store them as variables. These variables would then be saved to a file called `reporter_variables`. The autotable object in the summary template can then pick up these `reporter_variables` files and use them to create the table rows. One row will be created for each file that is read.

6.8.1 Selecting variables files for the table

To create the autotable you need to select where REPORTER will read the `reporter_variables` files from. This is done in the **Attributes** section.



In this example REPORTER will look for any `reporter_variables` files recursively from the directory `/data/DEMO/CONFERENCE/PEDESTRIAN_HEAD/NCAP_RUNS_2`.

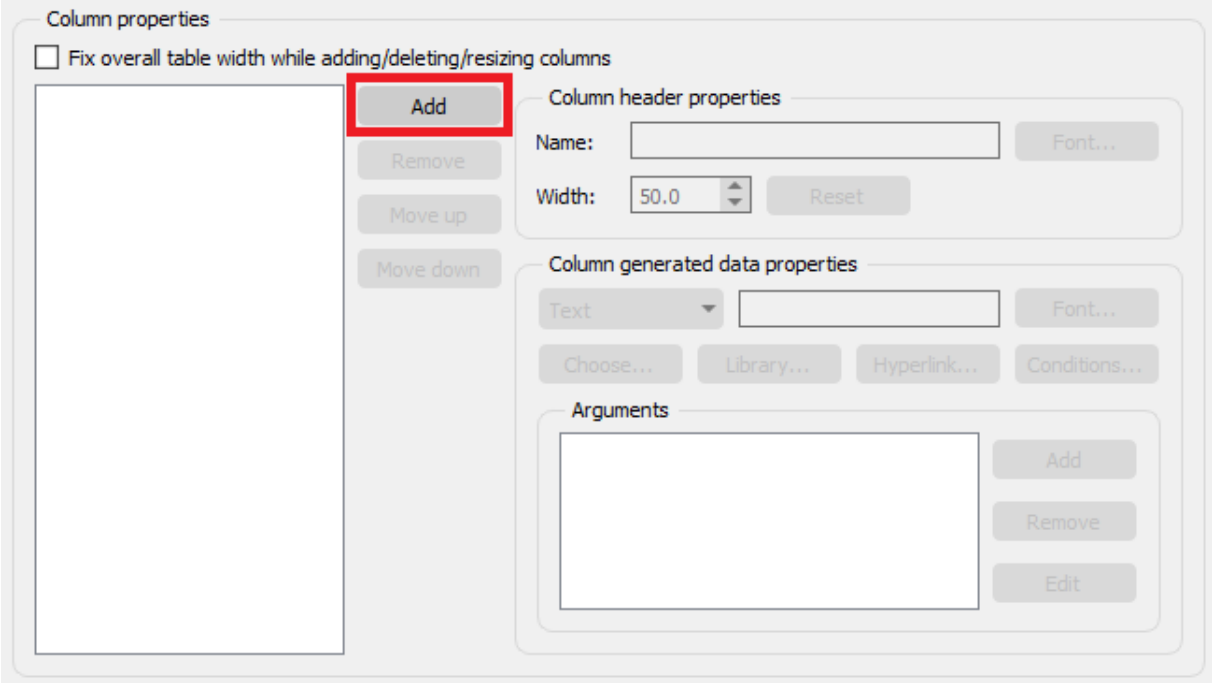
Alternatively you can select a file which will contain a list of filepaths for REPORTER to extract variables from. Each line of the file must point to a single directory containing a `reporter_variables` file, or a single file (of any name) in the `reporter_variables` format. Note that for the file case REPORTER does not look recursively from the specified directory, it looks in that directory only.

6.8.2 Setting the header and generated row heights

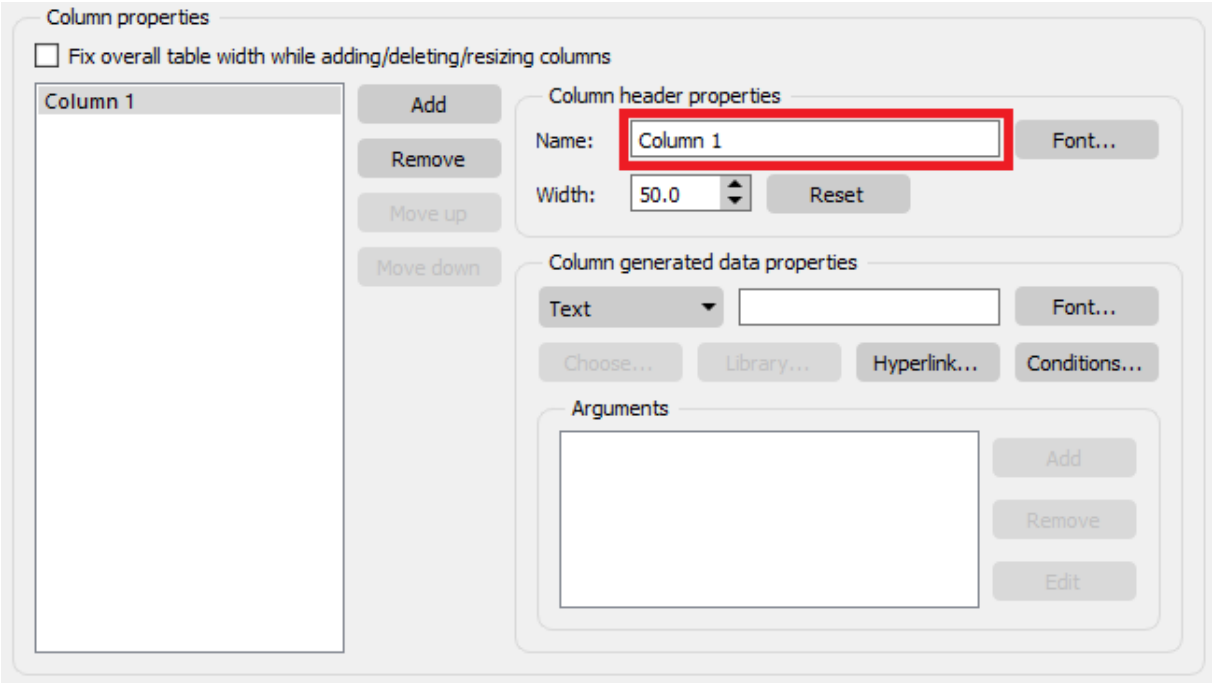
To set the height of the header row and any rows which are generated by REPORTER use the **Header height** and **Generated data height** options in the **Geometry** section.

6.8.3 Adding columns to the table

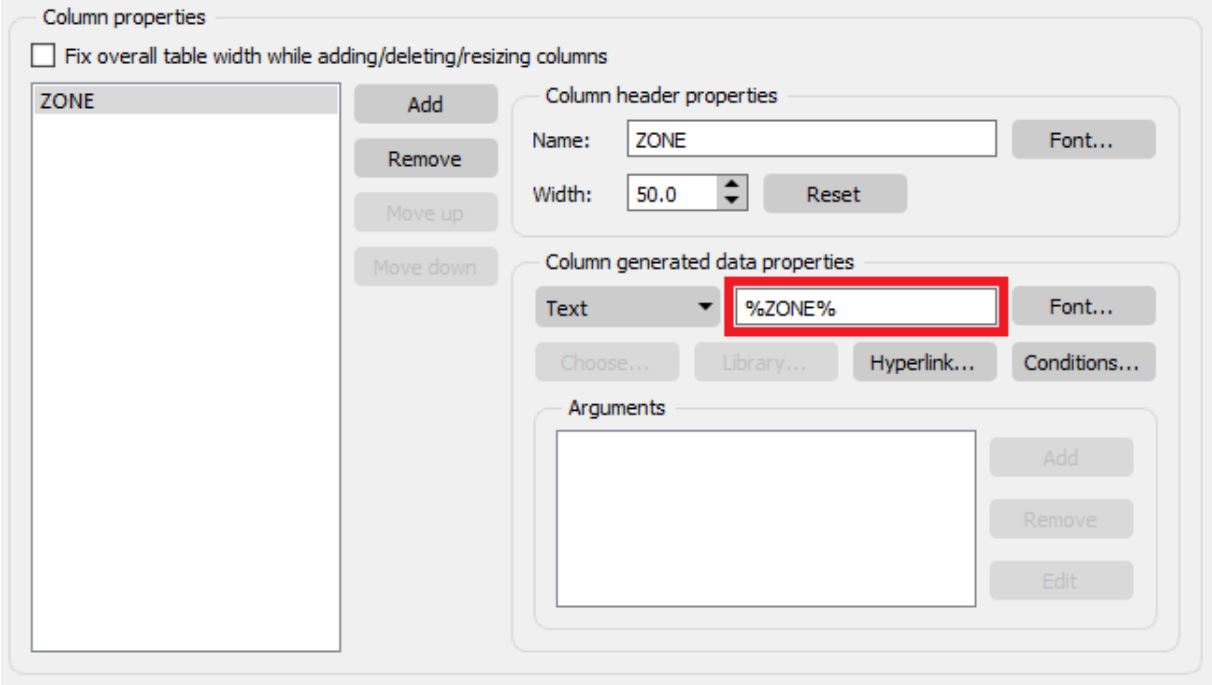
To add a column to the table use the **Add** button in the **Column properties** section.



This will create a new column with the default name **`Column 1`**. This is what will be shown as the column header. You can change the name in the **Name:** textbox and change the font used with the **Font...** button.



Once the column has been created you can decide how the data should be generated. Continuing the example above the first column is the zone so we change the column name to ZONE. The individual analyses that were post-processed by REPORTER saved the zone for the analysis in the variable ZONE, so for the generated data we want to input the text %ZONE% which means the value of variable ZONE. REPORTER will first look for any variables in the `reporter_variables` file. If it finds the variable then the value will be used. If REPORTER cannot find a variable in the `reporter_variables` file it will then look for a variable with the same name in the current template and use that value.



The font can be changed with the **Font...** button and hyperlinks (e.g. see the ZONE column in the above example output) and conditional formatting (e.g. see the HIC column in the above example output) applied using the **Hyperlink...** and **Conditions...** buttons.

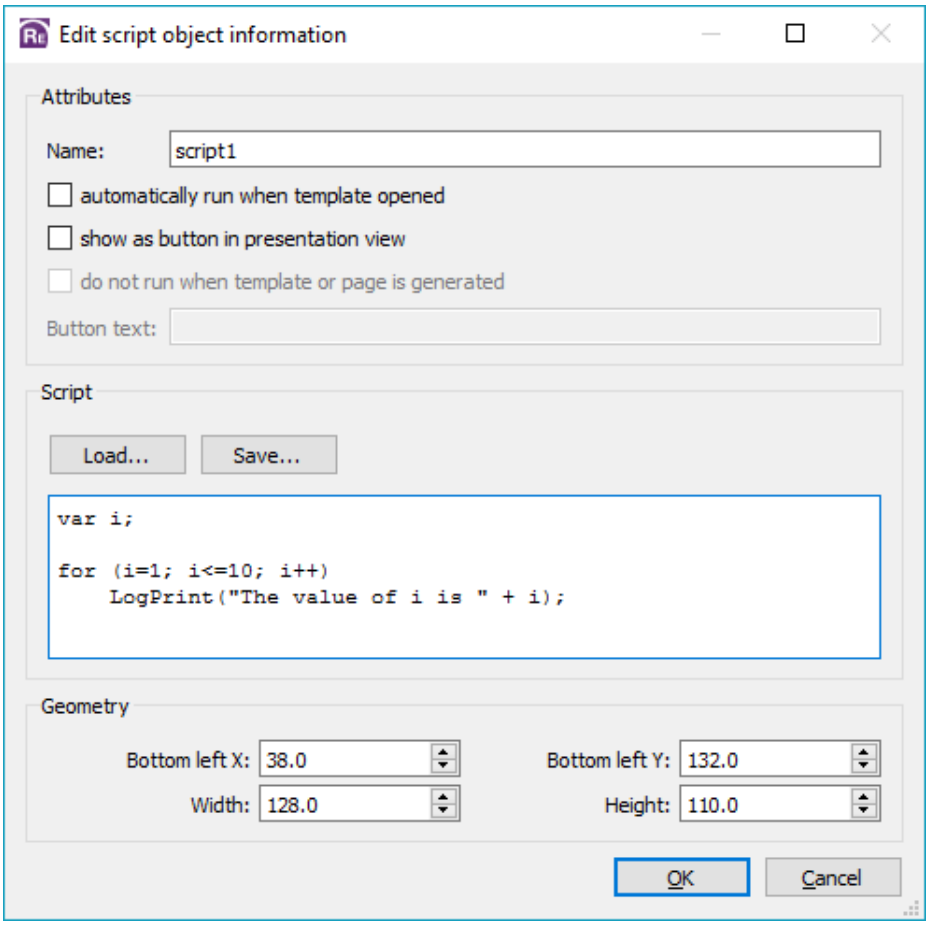
Instead of just using text in the generated data you can run a program instead which could be a standard library program or an external program. In this case the output from the program will be put in the table instead.

You can add as many columns to the table as necessary in exactly the same way.

6.8.4 Using the 'Fix overall table width...' checkbox

The functionality of the 'Fix overall table width while adding/deleting/resizing columns' checkbox is similar to that of the 'Fix overall table size...' checkbox for tables, except that the autotable checkbox only affects columns. The checkbox has no affect on row height (or overall autotable height); these are instead controlled through the various height options in the Geometry section.

6.9 Script objects



Script objects are JavaScripts that REPORTER can run using an embedded JavaScript interpreter. REPORTER also extends JavaScript by defining a number of classes for things specific to REPORTER. See appendix D for a reference to these classes.

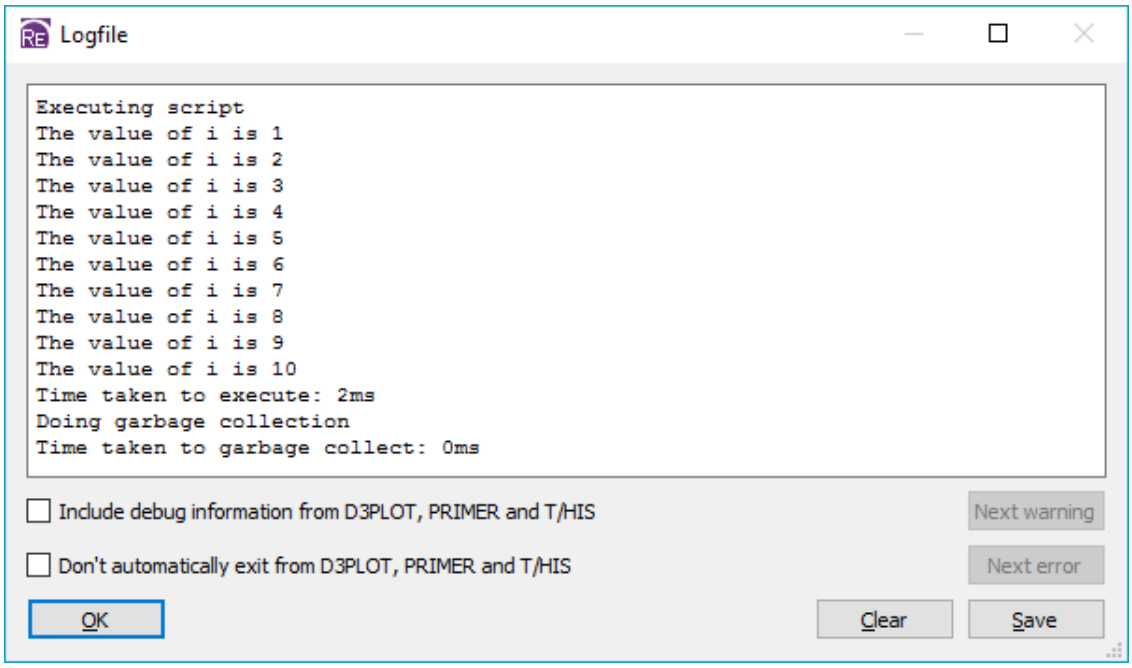
To insert a script, select the Script tool and then click and drag an area on the page. This will draw the area that the Script item will occupy and then map the script window.

You can load a script into the window with the **Load...** button and save the script to file with the **Save...** button. Scripts do not make any output on the page themselves (i.e. the area on the page that the script occupies will not have anything drawn on it from the script) but they can create output indirectly. For example, a script could create an image using the Image class in REPORTER and then this image could be imported with an Image File object.

Scripts do become visible on the page if you select 'show as button in presentation view'. If this checkbox is selected, then the script will run when the user clicks on the button. If you also select 'do not run when template or page is generated', the script will only be run when the button is clicked.

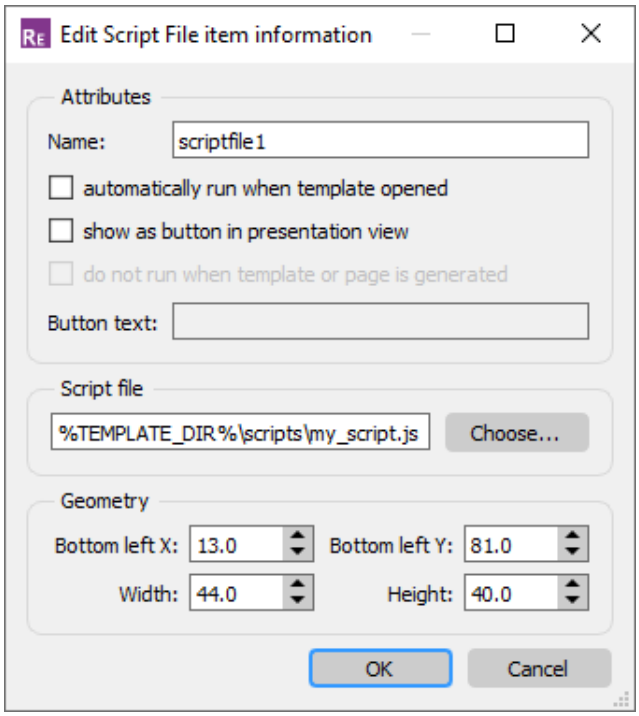
You can select one Script object in your template to be run automatically when the template is opened. REPORTER will only automatically run the first Script object it finds with this checkbox selected.

As a simple example, the script above prints text to the logfile window using the LogPrint function. This doesn't do anything useful in itself, but shows how you can produce useful diagnostic messages. This generates the following output in the logfile window:

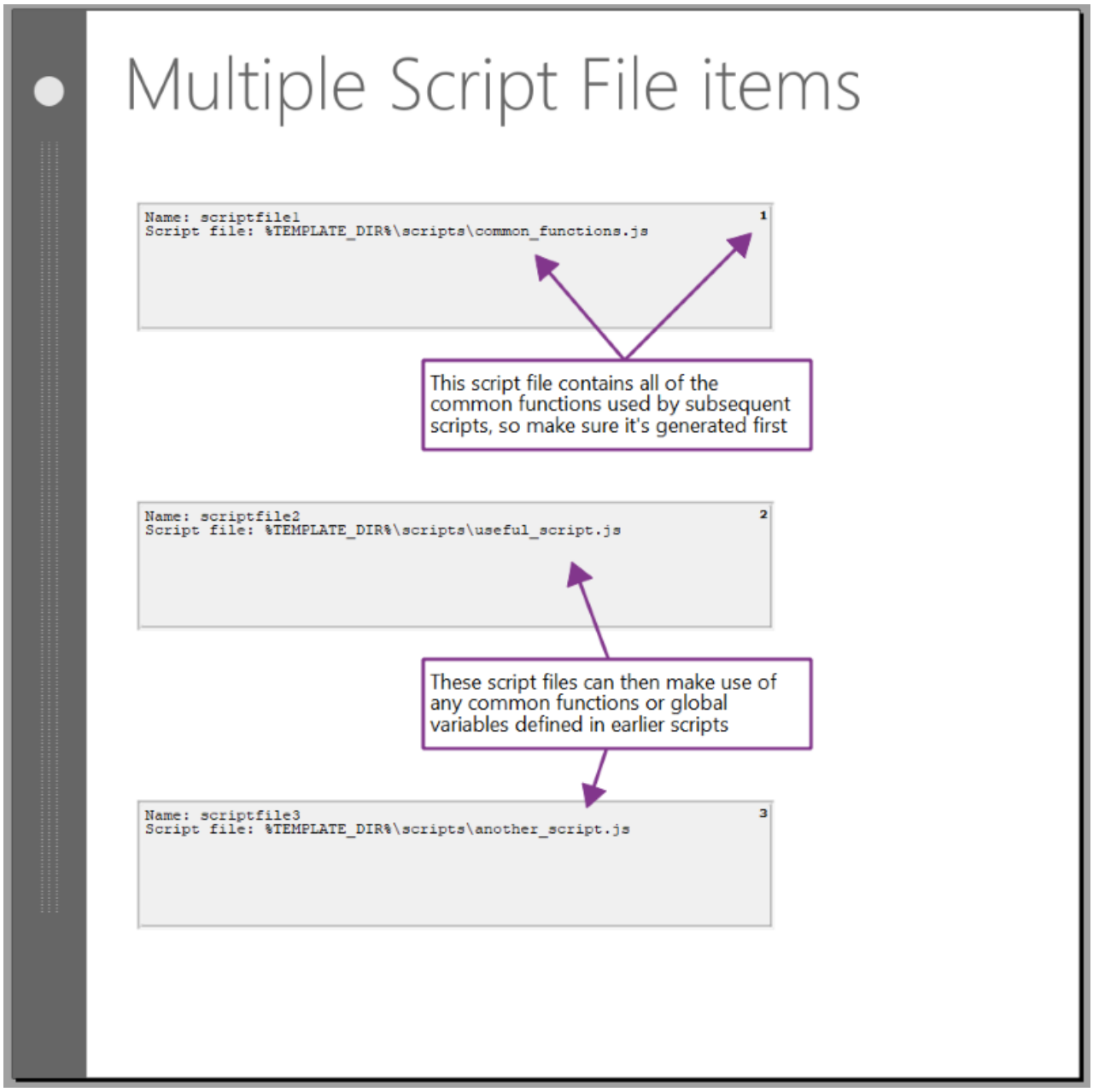


For more information on scripting, see Chapter 13. See also Script File objects, which have all the same features as Script objects, except that they allow you to refer to an external script file rather than embedding your script directly in the template.

6.10 Script File objects

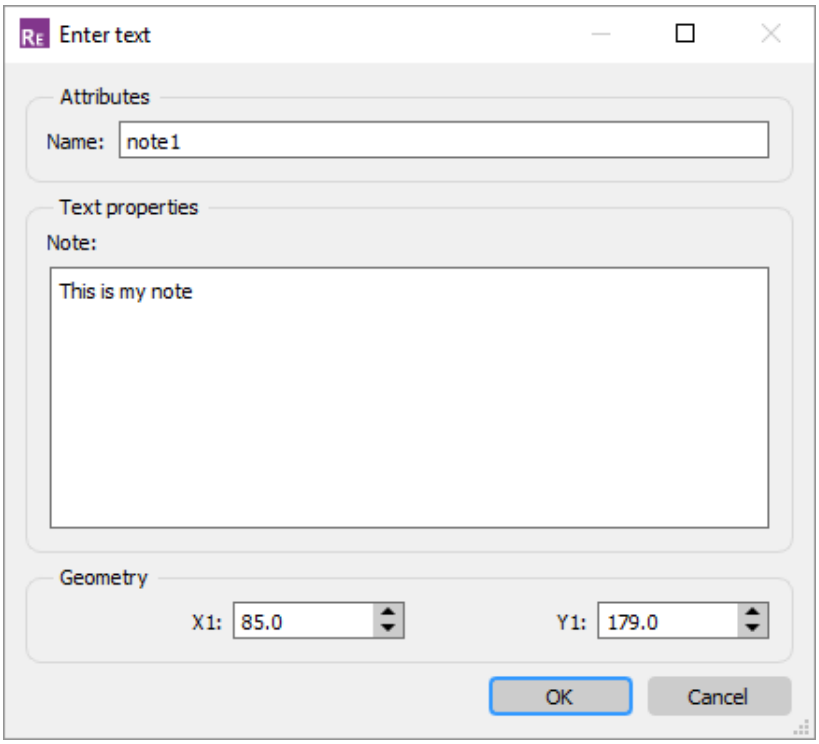


Unlike PRIMER, D3PLOT and T/HIS, REPORTER does not support the Use() function, or JavaScript modules. To help address this, a Script File item has been added, available from REPORTER 19.0 onwards. Script File items have all the features of Script items, except that rather than containing an embedded JavaScript, they point to an external script file. This allows you to edit the script in your preferred editor. Furthermore, since the REPORTER JavaScript environment persists for the duration of the REPORTER session (and is shared by all scripts), you can create one or more Script File items containing any common functions, followed by one or more Script File items that use those functions. This should provide much of the benefits of the Use function and JavaScript modules available in PRIMER, D3PLOT and T/HIS:



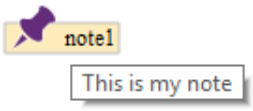
For more information on scripting, see Chapter 13.

6.11 Note objects



Note objects are used to add simple notes to your REPORTER template. They are only displayed in design view. To add a note when in design view, click on the note icon and click on the position on the page you wish to add a note. The following window will be mapped:

The name is what is displayed on the screen. The note is what is displayed when you hover the mouse over the note on the screen:

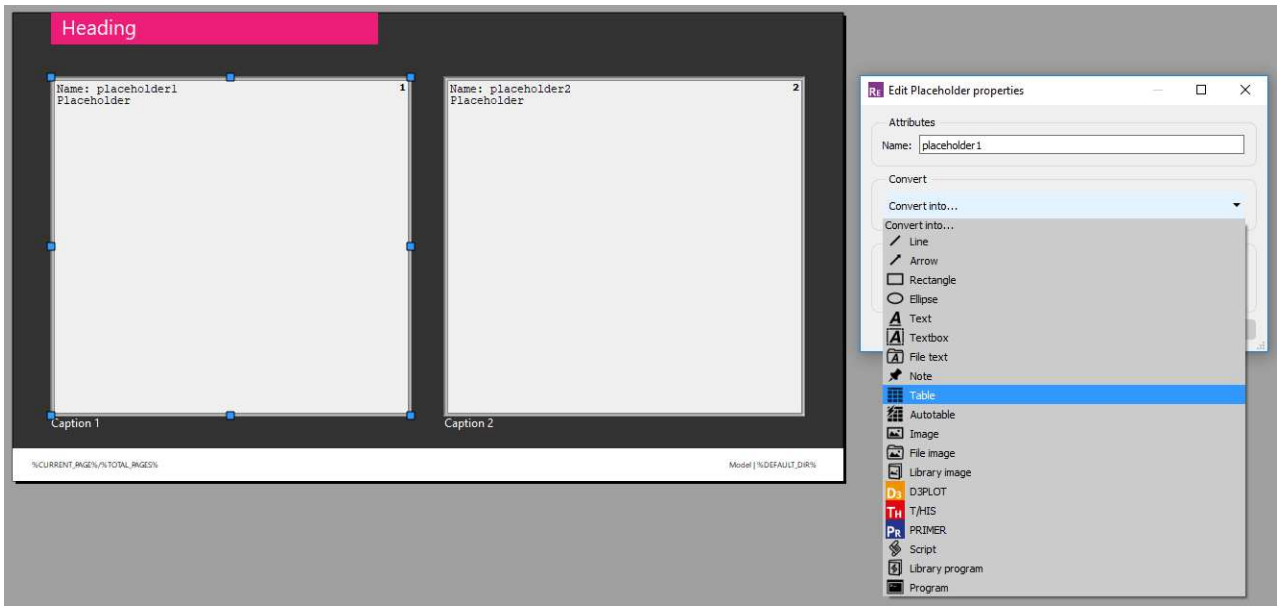


6.12 Placeholder objects

Placeholder items are used in many of the library templates provided with REPORTER. If you choose one of the templates from the **Standard** tab, you will see Placeholder items are used to predefine the page layout. Then, depending on what you want to add to the page, the Placeholder item can be converted into any other item type.

When using D3PLOT and T/HIS to capture plots and graphs, select a Placeholder item and then click **Capture** in D3PLOT or T/HIS. The Placeholder item will automatically be replaced by a D3PLOT or T/HIS item respectively.

You can also convert Placeholder items into PRIMER items, Tables, or indeed any other item type. Double-click on a Placeholder item to edit it, and then use the **Convert into...** drop-down menu to choose another type of item. When you click **OK**, the item will be converted.

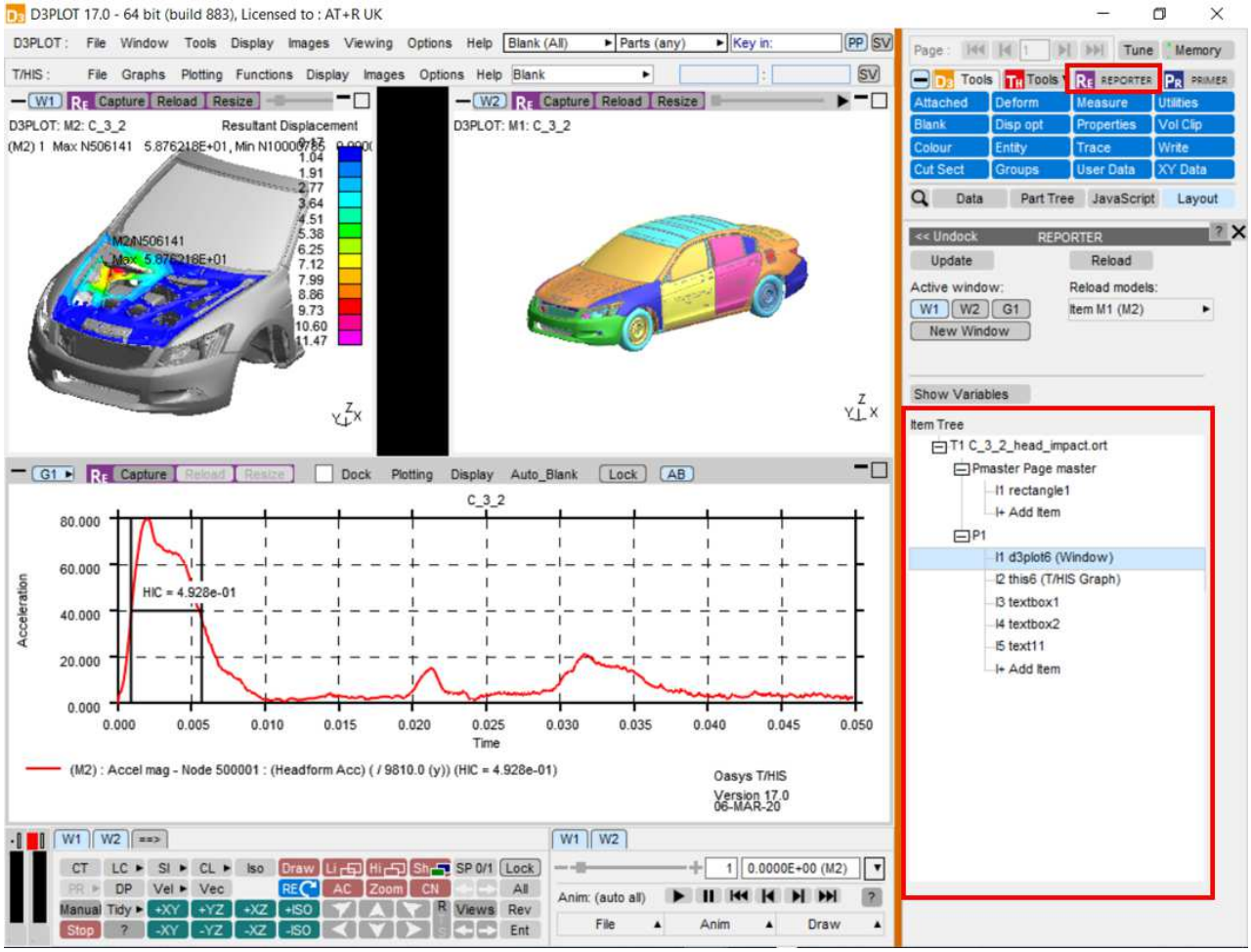

Heading
Name: placeholder1
Placeholder
1
Caption 1
Name: placeholder2
Placeholder
2
Caption 2
%CURRENT_PAGE%/%TOTAL_PAGES%
Model | %DEFAULT_DIR%
Edit Placeholder properties
Attributes
Name: placeholder1
Convert
Convert into...
Convert into...
Line
Arrow
Rectangle
Ellipse
Text
Textbox
File text
Note
Table
Autotable
Image
File image
Library image
D3PLOT
T/HIS
PRIMER
Script
Library program
Program

7. REPORTER Integration

This section describes how to work with D3PLOT, T/HIS and REPORTER to quickly and easily create reports from results.

7.1 Linking the Programs

REPORTER can be opened from D3PLOT and T/HIS using the REPORTER button in the top-right toolbar panel. This opens a linked session of REPORTER, allowing reports to be interactively created and edited. Both D3PLOT and T/HIS can be opened from inside REPORTER too, using the program buttons in the top bar of REPORTER. REPORTER can be connected to both D3PLOT and T/HIS at the same time and the D3PLOT → T/HIS link is also supported. Graphs in T/HIS are treated the same as graphs in a D3PLOT → T/HIS linked session.



7.2 Item Tree

Once a template is opened in REPORTER, all items in the template will appear in the Item Tree in the REPORTER panel in D3PLOT or T/HIS. Selecting an item in the Item Tree will select the corresponding item in REPORTER and vice-versa.

The Item Tree can include items of all types in REPORTER, such as textboxes and images, as well as D3PLOT, T/HIS and PRIMER items. Only Placeholder items, D3PLOT items and T/HIS items can be overwritten with new D3PLOT or T/HIS items. Placeholder items exist to allow a layout to be created for the report before populating it and can be converted into any other item type.

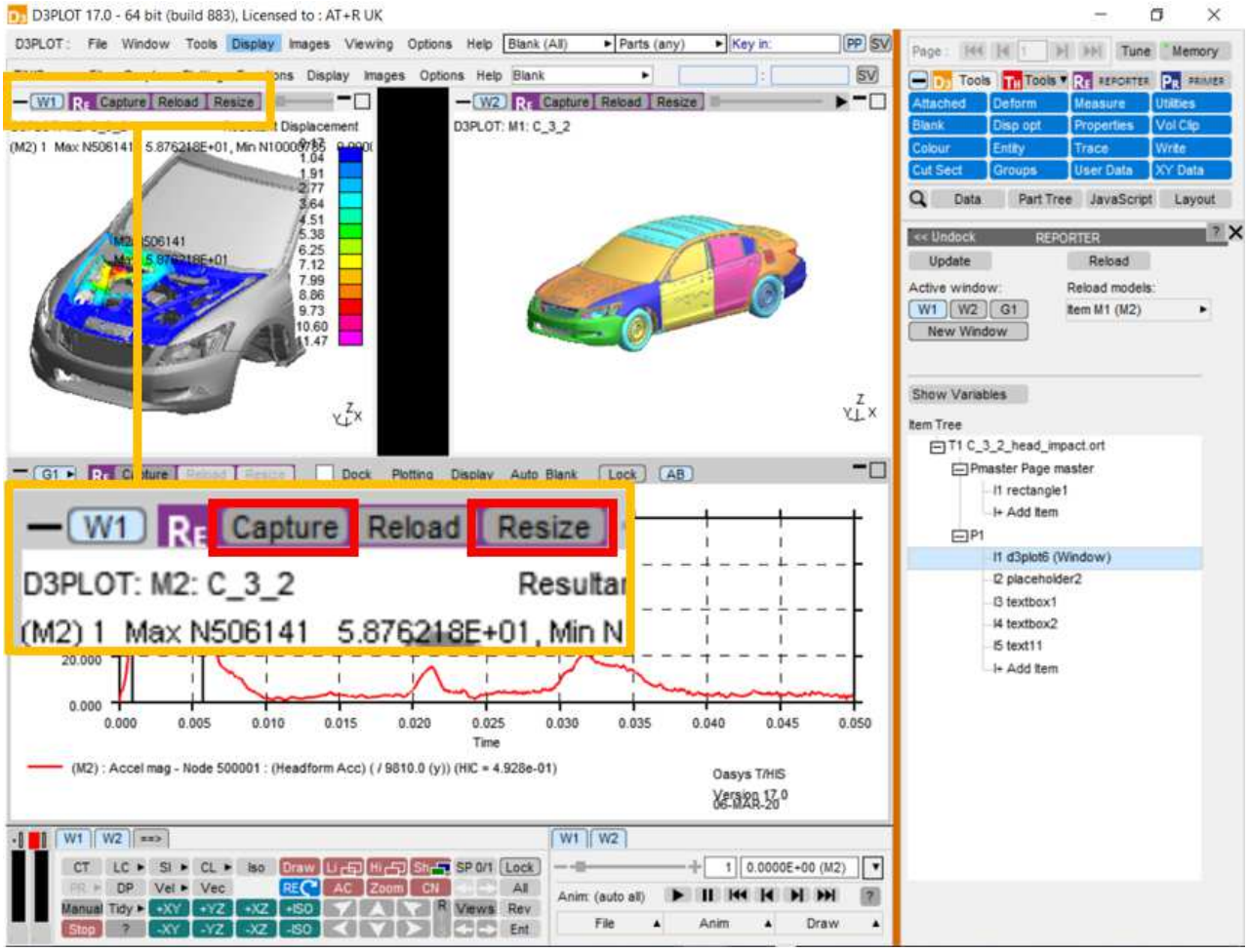
7.3 Capture

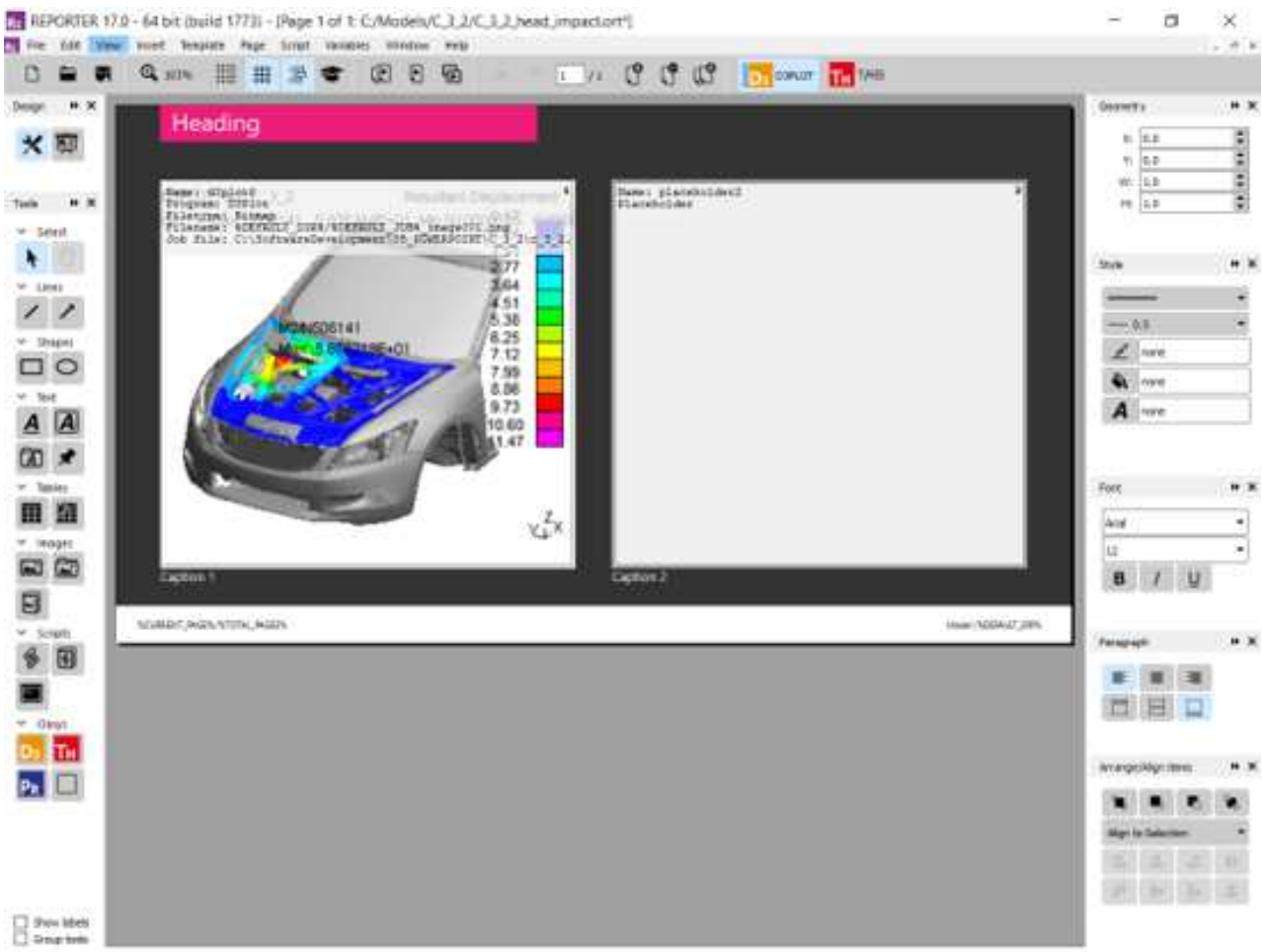
Windows and graphs can be captured into REPORTER, saving an image together with additional information to allow

the capture to be reloaded later. For D3PLOT windows, this is a properties and settings file. For T/HIS graphs, this is a FAST-TCF script. Graphs captured in the D3PLOT → T/HIS link are treated exactly the same as graphs in T/HIS, so the resulting items will be identical.

Note that in the version 17 method, only single windows and graphs can be captured. The intention being that the windows and graphs are easily captured individually and laid out in REPORTER with greater flexibility.

In order to capture a window, first select the target item in REPORTER, either selecting it directly in REPORTER or using the item tree. You can capture into a new item by selecting **I+ Add Item** in the item tree. Once the item is selected, the **Resize** button on the top bar of the window can be used to resize the window to match whatever image size is specified on the selected REPORTER item, such as **Fit object box**. Finally, either press **Capture** on the top bar of the target window or select the window in the **Active window** list in the REPORTER panel and press **Capture** at the top of the panel. This will send the information to REPORTER and the image will appear on the item.



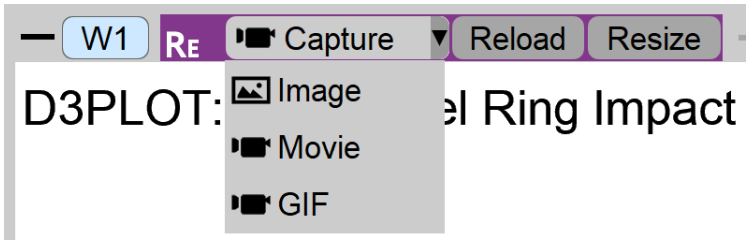


7.3.1 Capturing Movies

From version 18.0, MP4 movies and animated GIFs can be captured with a D3PLOT Item in REPORTER in place of a static image. The process for Capture is unchanged: just right-click on the Capture button in D3PLOT (either in the REPORTER panel or at the top of the target window) to reveal the new Movie (MP4) and GIF options.

When selecting an existing D3PLOT Item in the REPORTER Item Tree, the Update Capture button will always update to switch to that Item type (Image, Movie, or GIF). Left-clicking the Update Capture button will then replace the current capture with one of the same type without the need to use the drop-down menu again. The drop-down menu can be used if switching type (e.g. PNG Image to MP4 Movie) is desired.

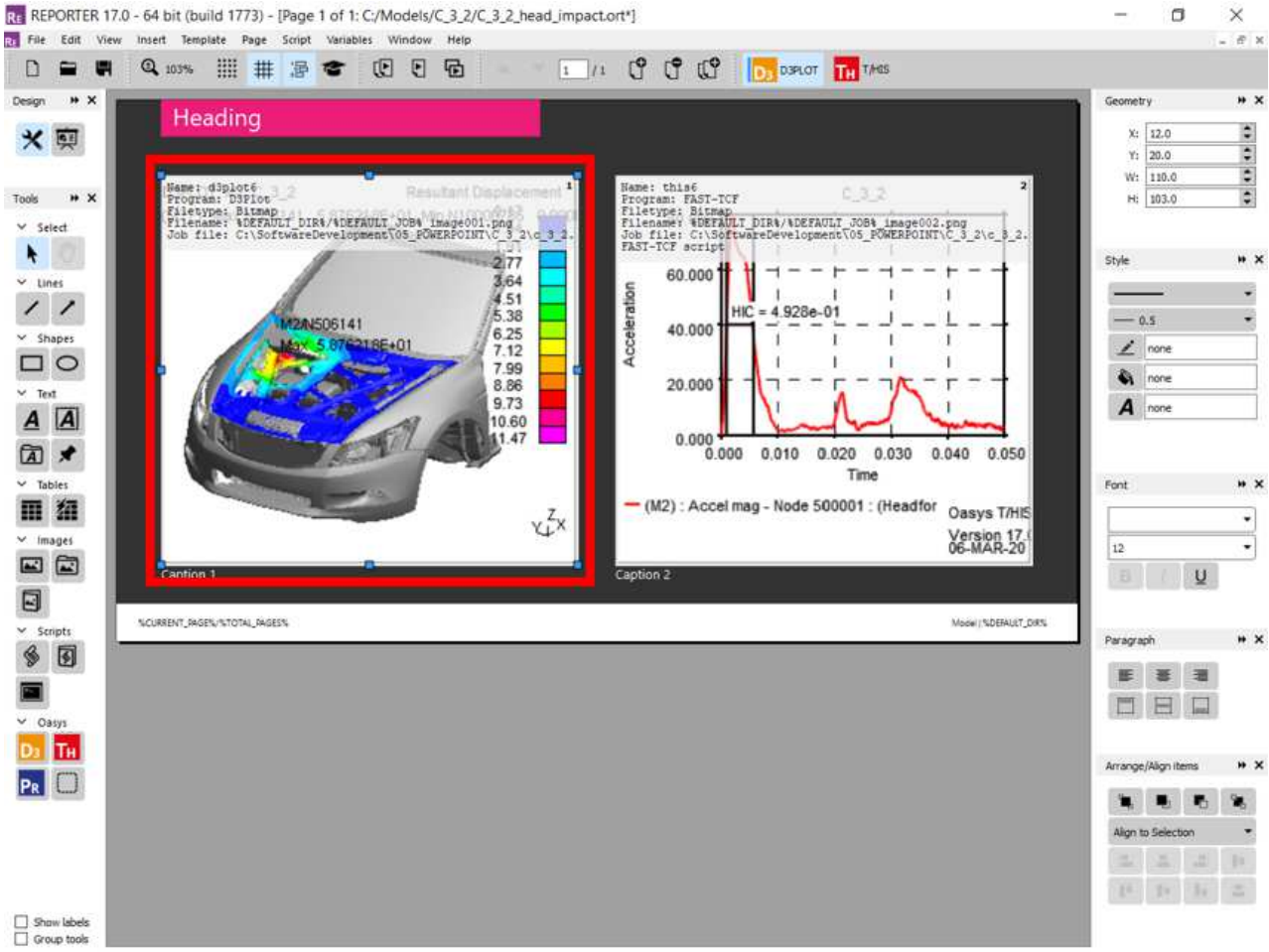
Settings such as frame rate and quality are determined by their current status in the D3PLOT Images -> Write -> Movies panel so be sure to check these before conducting Capture.



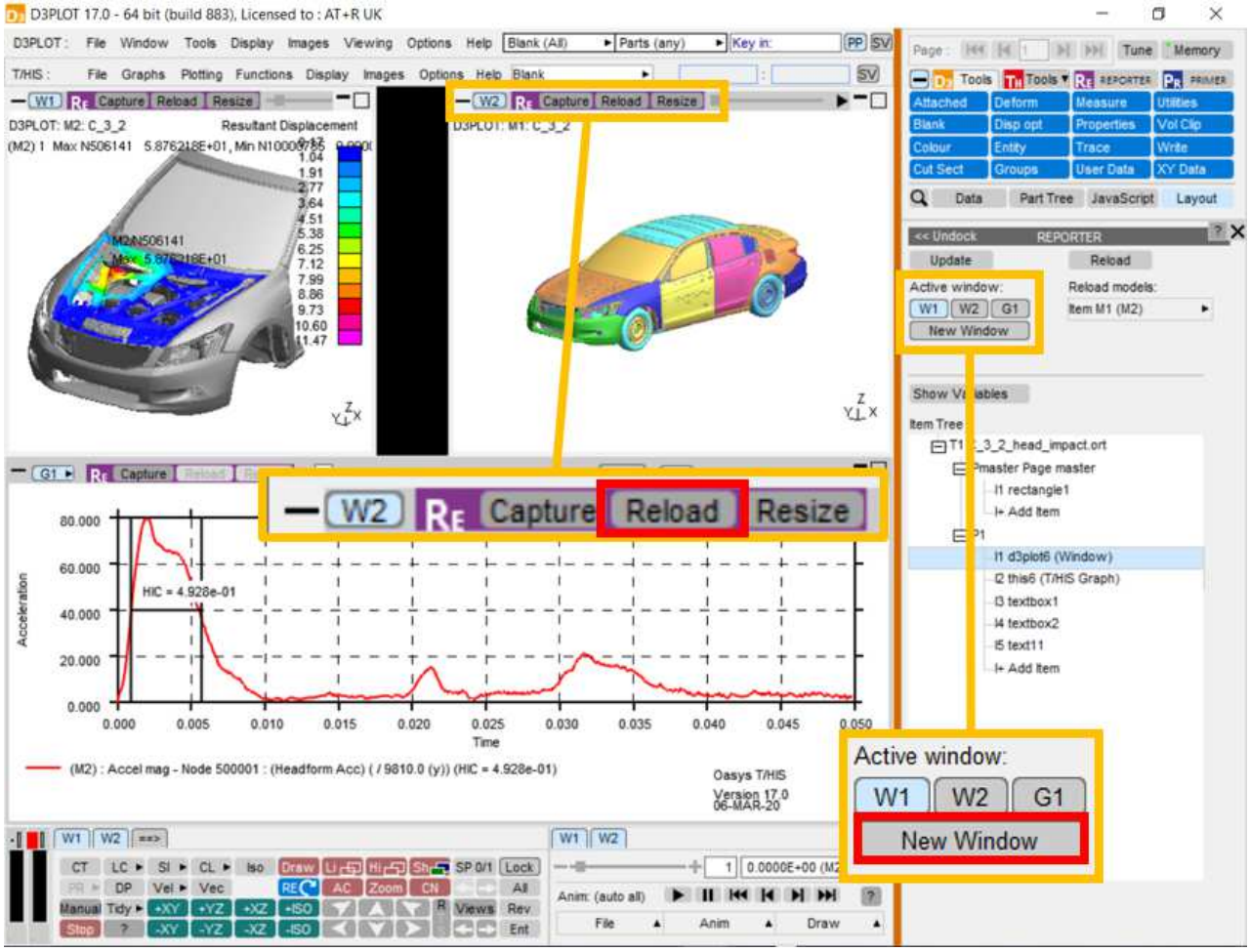
7.4 Reload

Existing REPORTER items can be reloaded back into D3PLOT or T/HIS. Items captured from graphs in the D3PLOT → T/HIS link are treated the same as items captured from standalone T/HIS. As such, they can each be reloaded either into D3PLOT or T/HIS.

First select the item in REPORTER that you want to reload:



Then either press reload at the top of the target window, or select **New Window** in the Active window list:

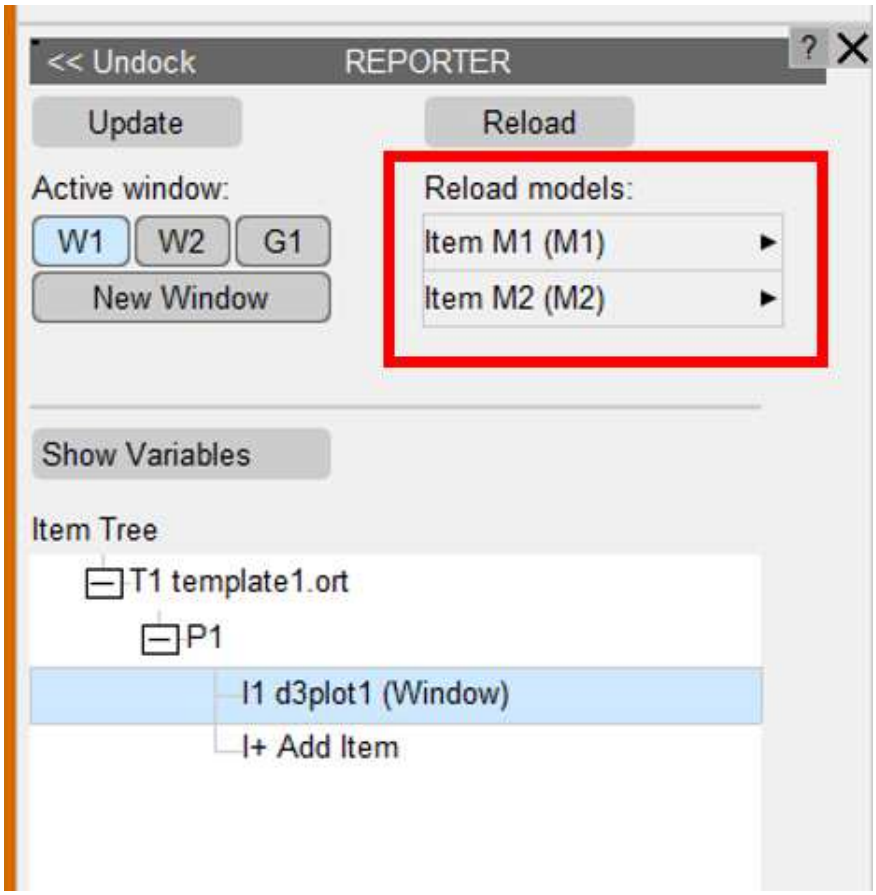


This will clear the target window, open the relevant models, not opening them again if they are already open in the session, then load the stored item information, reproducing the capture.

7.4.1 Reload Models

The models used in an existing item are listed in the **Reload models** list. The models will be listed as ‘Item M*n*’, where *n* is the index of the model in the item, not of the model in the session. If the model is also open in the current session, then the model ID in the current session will be displayed in brackets.

Each entry in the list has a popup attached, allowing the model to replaced either by a model in the current session or by browsing for a model. This will not change the models stored in the item, but instead when the item is reloaded into the current session the replacement models will be used. The resulting window will then need to be captured, either into a new item or to overwrite the original.



7.5 Generate

Once a complete template has been created, it can be generated using **File** → **Generate** in REPORTER. This will generate in an existing session if there is one, otherwise a new session will be started. T/HIS items will be generated in standalone T/HIS, unless the T/HIS link is already open in D3PLOT, in which case they will generate in the link. It is faster to generate in standalone T/HIS.

7.6 Variables

Variables can be added to both D3PLOT and T/HIS items, allowing data related to the capture to be made available in REPORTER. The REPORTER panel can be undocked and expanded to display the variables list by selecting **Show Variables**.

For T/HIS items, variables can be added containing properties of any of the curves in the selected graph or all the curves combined using the **All Curves** option. By default, T/HIS items will have variables for the MAX and MIN values taken over all curves in the selected graph. When selecting the curve for a newly created variable using the curve popup, curves are referred to as 'IC*n*', meaning 'Item Curve *n*', where *n* is the index of the curve in the selected graph. The curve label and number in the current session are also displayed in the popup.

For D3PLOT items, variables can be added for the MAX and MIN values of any of the plotted data components on any of the models. By default, D3PLOT items will have variables for the MAX and MIN values of all plotted data components for each model in the selected window.

Variables can be added using the **+** button and deleted using the **X** button next to the row.

Initially, variables will appear under 'New Variables' until the item is captured, when they will move to 'Existing Variables'. Variables will be given default names based on their item number, variable type and model/curve that they relate to. However, these names and descriptions can be manually edited.

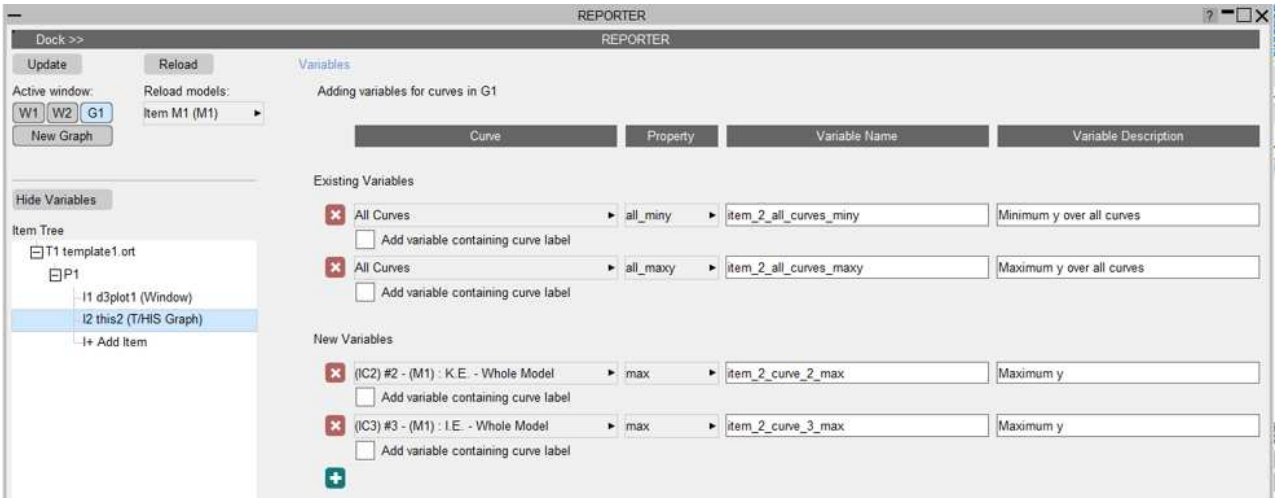
For D3PLOT items, the **Entity ID** and **Entity Type** tickboxes can be used to create additional variables to contain this information. These will have the same name as the original variable with either "_ENT_ID" or "_ENT_TYPE" appended.

For T/HIS items, the **Add variables containing curve label** tickbox will create an additional variable containing

the curve label of the relevant curve, with "_LABEL" appended to the name.

Example of a D3PLOT item with two exsiting variables, referring to models in Window 1:

Example of a T/HIS item with two new variables and two existing variables, referring to curves in Graph 1:



7.7 Exceptions to the Version 17 Method and Existing Templates from Version 16 and Earlier

There are some item types that are not yet supported in the new version 17 method. In this case, the version 16 method will be used and nothing will have changed. These are:

- T/HIS JavaScript items
- Items containing multiple graphs/windows

Any item can be captured and generated using the version 16 method by selecting the **Capture and generate this item using the old method** option in the object information in REPORTER.

Existing version 16 and earlier templates should work exactly as they used to. All items will use the version 17 method unless they meet one of the specified exceptions above. This gives some additional benefits:

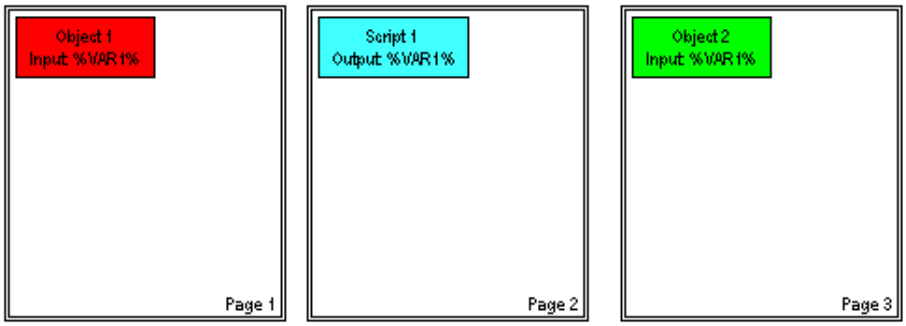
- When generating the report, all supported items will be generated in the same session, without opening the same models multiple times. This will make the process faster.
- The report can be edited interactively using all the perks of the version 17 method.

8. Generating and outputting reports

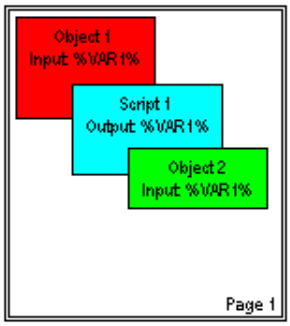
8.1 Effect of object order on generating a report.

The order the various objects are layed out on a page relates to the order in which they will be processed by REPORTER when it generates a report. So if you have a program/script that creates a variable in it's output, that program/script will need to be on the same page or an earlier page than the object that first uses the generated variable. If it is on the same page it also needs to be earlier in the order of objects on the page than any objects that uses that variable.

The following series of example shows what will and won't work. In all the examples Object 1 (red) and Object 2 (cyan) both use a variable (**VAR1**) generated by Script 1 (green) as an input.

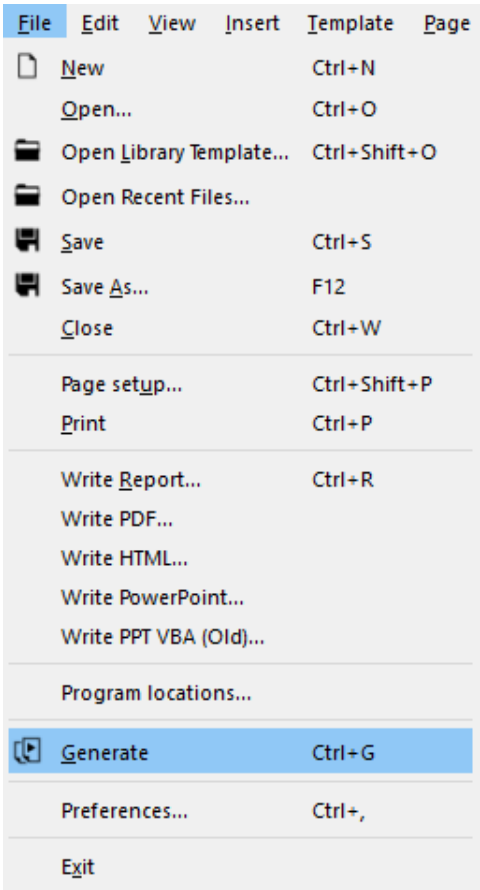


In this case Object 1 is on an earlier page than Script 1 so the variable **VAR1** hasn't been created yet. In this situation REPORTER will give a warning and uses a blank for the variable **VAR1** in Object 1. Object 2 however comes after Script 1 so the variable **VAR1** has been created and Object 2 can be generated normally



In this case Object 1 is on the same page as Script 1, but comes before it in the order of items on the page so there variable **VAR1** hasn't been created yet. In this situation REPORTER will give a warning and uses a blank for the variable **VAR1** in Object 1. Object 2 however comes after Script 1in the order of items on the page, so the variable **VAR1** has been created, and Object 2 can be generated normally.

8.2 Generating reports



Once a report template has been created a report can be generated by selecting the **Generate** option in the **File** menu.

Generating a file causes all of the objects on the page to perform any necessary actions to create the output for that object. For example:

- Text objects could expand variables into the actual values
- File objects would read the text/image file and show it
- Program objects would be "run" to generate the output.
- Tables will be created
- etc.

If any objects are to be created from D3PLOT or T/HIS then REPORTER will start the relevant program to produce the object and then insert the object into the report. REPORTER will also run any specified programs/scripts and insert the output into the report as required.

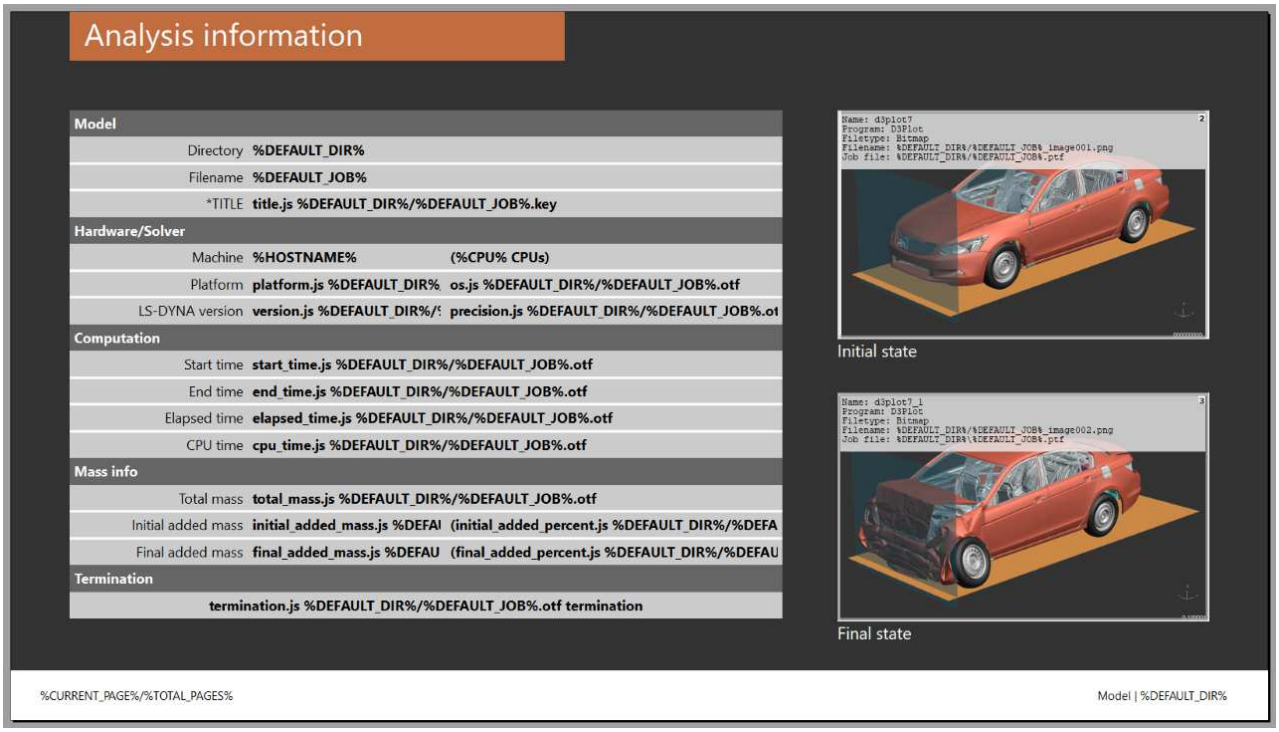
During report generation feedback is given in the status bar showing what REPORTER is doing. For example in the image below REPORTER is currently generating output for object ’oasys21’ on page 1 and the report generation is 29% complete.

You can stop report generation at any time by pressing the **Stop** button in the status bar.

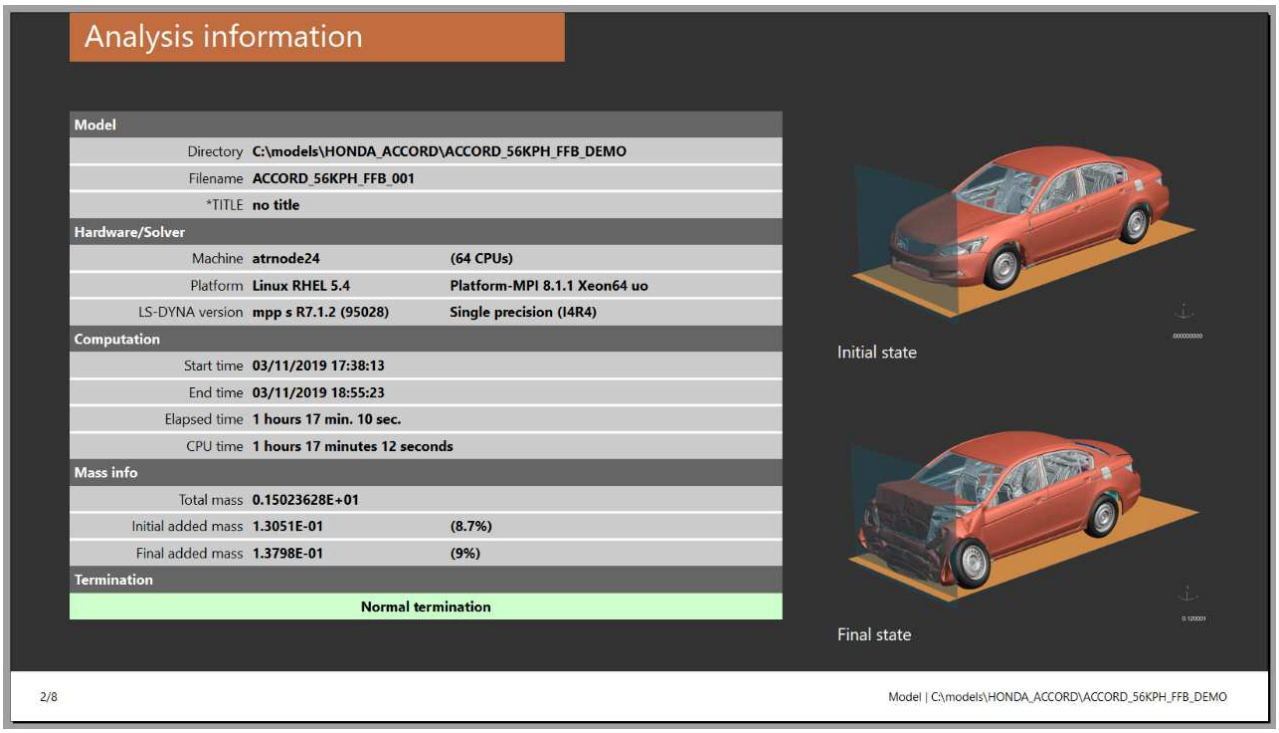
To switch between the the design view (showing the report template) and the presentation view (showing the final report) you use the Design buttons

The images below show an example of a report template before and after generating the page.

Design view before generating report:



Presentation view after generating report:

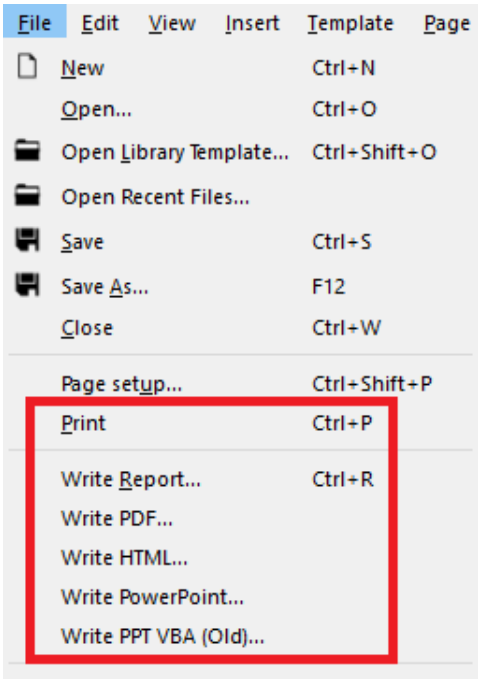


8.2.1 Using the cursor in presentation mode

When you first go into presentation mode after generating a template the cursor mode changes to the "hand" cursor. In this mode you cannot select or edit any objects. The cursor is used for following hyperlinks. This is likely to be extended to other functions in future releases of REPORTER.

You can change the mode back to the select mode in which case all of the normal operations which you can do in design mode can be done including editing. Additionally if you choose any of the other modes you can create new objects even though you are in presentation mode.

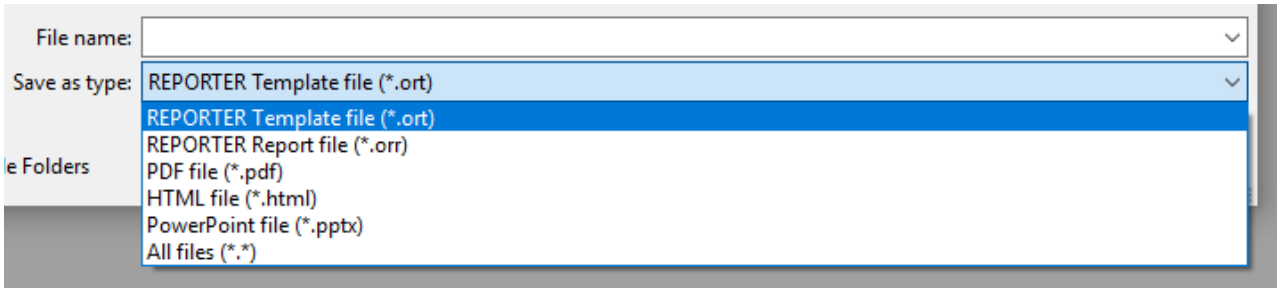
8.3 Outputting a generated report



REPORTER can create various types of output by using the various write option in the **File** menu. Currently the types are:

- **Write Report** – write the file as a report (images etc included with the template)
- **Write PDF** – write an Adobe PDF file
- **Write HTML** – write an HTML web page
- **Write PowerPoint** – write a Microsoft PowerPoint file directly
- **Print** – print the report with a printer

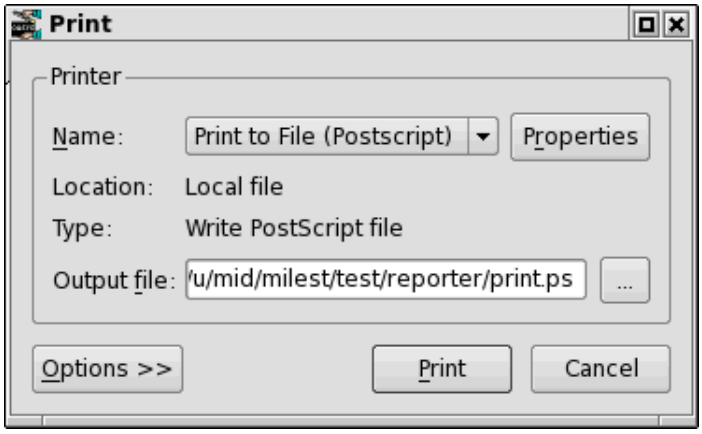
The Write options can also be accessed by selecting the appropriate file type from the Save As menu (see below).



8.3.1 Printing

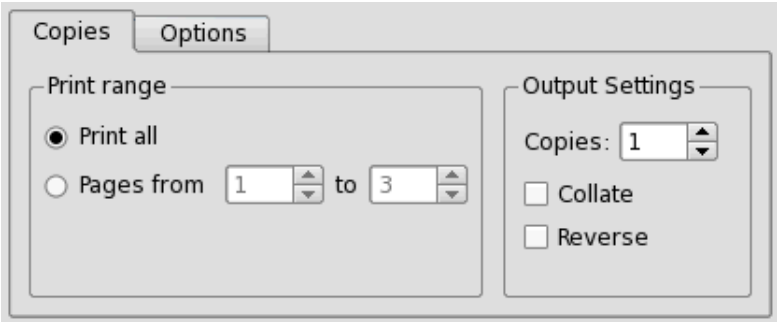
On Windows, the **Print** command will bring up the standard windows printer dialog.

On unix, it will bring up the dialog.

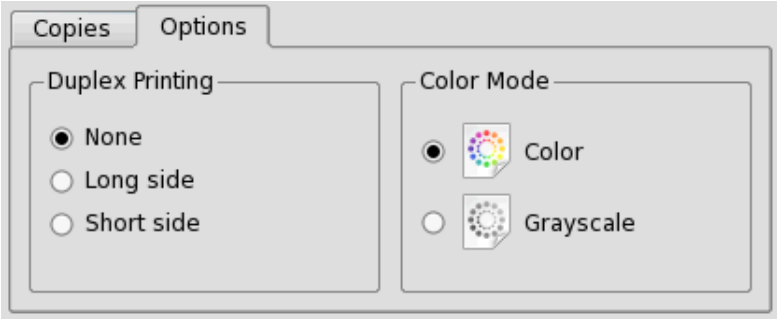


Extra options can be given by pressing the **Options >>** button.

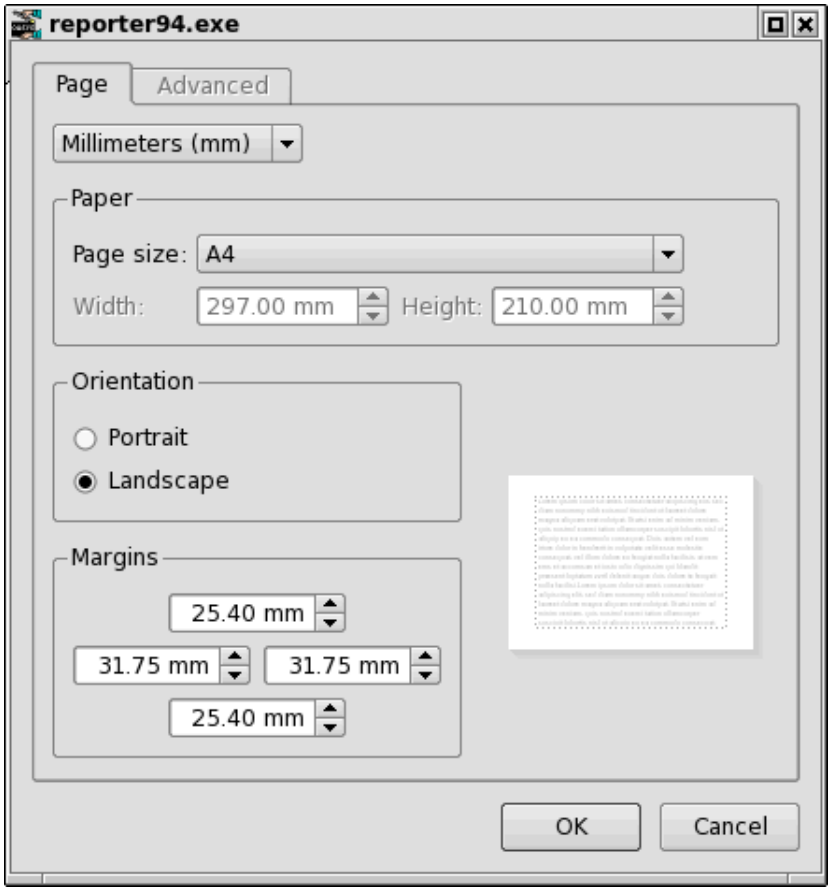
The **Copies** tab allows you to choose what pages should be printed and how many copies.



The **Options** tab allows you to choose double sided printing and black and white or colour output.



The **Properties** button allows you to set the page size and margins.



8.3.2 PDF files

Write PDF will save the report as an Adobe PDF file. Select the name of the PDF file you want to write.

8.3.3 HTML output

Write HTML will save the report as a HTML file for the web. Select the name of the HTML file you want to write. REPORTER will then create a HTML page using frames containing the report. There will be a html HTML for each page in the report and a contents page. All the necessary images and files will be placed in a subdirectory of the main HTML file which is called **<name>**`.html_files`. So for example if you create a file `example.html`, REPORTER will create a directory called `example.html_files` as well and put any extra files in there. So if you want to move the html file to somewhere else remember to move `example.html` and the directory `example.html_files`.

8.3.4 PowerPoint files

Writing PowerPoint files directly

REPORTER can write PowerPoint files directly for Windows and Linux. Select **Write PowerPoint** and give the name of the PowerPoint file you want to create. REPORTER will write the file.

Notes on PowerPoint output

When you use textboxes, text files and tables in REPORTER the output is clipped to the size of the object defined on the page. PDF and HTML output also support this but it is not possible to control the size of a 'textbox' in PowerPoint (in PowerPoint a table is made up of a collection of 'textboxes'). When writing PowerPoint output be aware of the following limitations.

1. If the text is too wide to fit in the 'textbox' it will automatically be wrapped onto multiple lines by PowerPoint.
2. If the combined height of the text, the top margin and the bottom margin is greater than the height of the textbox

PowerPoint will increase the height of the textbox to make it high enough.

If the Powerpoint output is not aligned correctly or is not what you see in REPORTER it is likely to be caused by these problems. Adjusting the size of the object, the text size or the margins will help to fix any problems.

8.4 Combining output from multiple reports

If REPORTER generates several templates and saves them as reports (see section 3.4 for more details) then it is sometimes useful to combine the output into a single pdf, html or pptx file. The easiest way to do this is to use the REPORTER options in the SHELL. See the SHELL manual for more details.

It can also be done on the command line in REPORTER by using the `-combine` command line argument. For example, if you wanted to combine the output from 3 reports to a pdf file and a PowerPoint file this could be done with the command:

```
reporter19 .exe -combine report1.orr report2.orr report3.orr -pdf=combined.pdf
-pptx=combined.pptx -exit
```

8.5 Animation support for output file formats

From version 18.0, animation playback has been added to the D3PLOT, Image, and Image File Items in REPORTER. Currently both animated GIFs and MP4 movies are supported. Due to development constraints, not all types of REPORTER output are capable of playing back this content in this first release. This is summarised in the following table:

	GIF	MP4
Export to PowerPoint	Y	Y
Embed in Templates (.ort) and Reports (.orr)	Y	N
Export to PDF or HTML	N	N

For some extra information on each of these:

- Our top development priority was to be able to export both GIF and MP4 to PowerPoint, which is supported.
- Animated GIFs and MP4 movies will appear as static images in PDF and HTML.
- Although MP4 cannot be embedded, an MP4 Image Item in a Template or Report will still load the .mp4 file if saved with a valid filepath.

We hope to address some of these limitations (particularly embedding MP4 in Templates and Reports) in future versions of REPORTER.

9. Working with Variables

A main feature of REPORTER is that you create a template from which a report can be generated. This allows you to create a standard template for a project and then use that template to automatically create a report for a number of model runs. This is mainly achieved through the use of variables.

Variables are defined with a name and a value which can be a number or a text string, for example.

Variable Name	Value
`CURRENT_PAGE`	2
`DEFAULT_DIR`	/data/test/ tube1
`DEFAULT_JOB`	tube_test1

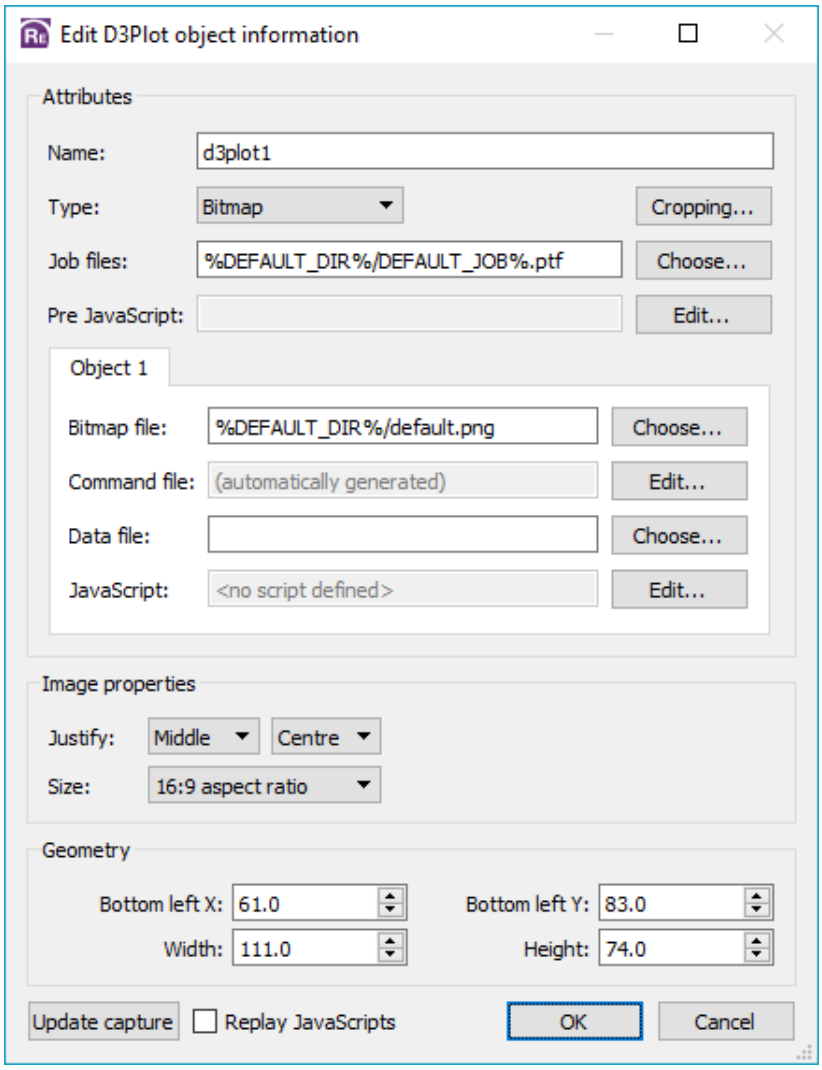
The main advantage of using variables when defining the various objects in the report template is that rather than having to go through the report and change all the various filenames and directory paths when you want to generate a report from a new model, all you need to do is change the variables. This can be done manually by editing the template in REPORTER, or you could insert a program/script into the template that would calculate and define all the necessary variables when REPORTER generates a report.

9.1 User defined variables

For example, if you want to create a report template that has a number of images that are created by a D3PLOT object. If you want to use the template to generate reports for a number of models, the problem is that the various filenames and directory paths will be different for each model. e.g:

Model	Directory Path	Job Name
Crush Tube 1	/data/test/tube1	tube_test1
Crush Tube 2	/data/test/tube2	tube_test2
Crush Tube 3	/data/test/tube3	tube_test3

To get round this problem you can use a variable for the directory path called `DEFAULT_DIR` and a variable for the job name called `DEFAULT_JOB`. When inserting the D3PLOT objects (see Section 6 for more detail about inserting D3PLOT objects) use the variables for the directory path and job name. The variables need to be enclosed by % signs to distinguish them from the rest of the text string.



When generating a report for Crush Tube 2 model, the variables would be defined as follows:

Variable Name	Value
DEFAULT_DIR	/data/test/tube2
DEFAULT_JOB	tube_test2

When REPORTER generates the report it will substitute in the values of the relevant variables, so the two text strings would become:

Bitmap File	/data/test/tube2/def.bmp
Job File	/data/test/tube2/tube_test2.ptf

To generate a report for one of the other templates, all you need to do is change the value of DEFAULT_DIR and DEFAULT_JOB.

9.2 Predefined variables

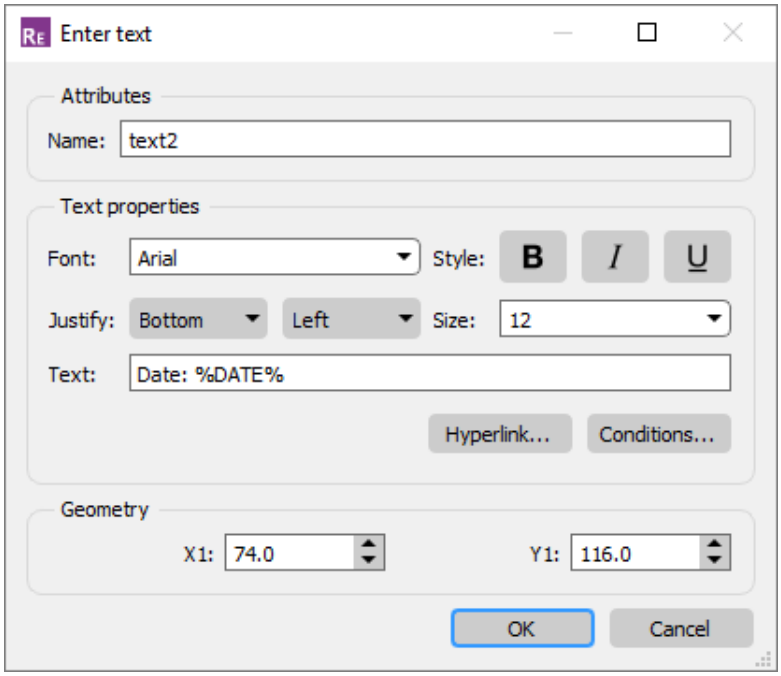
REPORTER already has a number of variables defined. They are:

Variable	Description
CURRENT_PAGE	The current page in the report (can be used when a report is generated)
TIME	The current time (can be used when a report is generated)
DATE	The current date (can be used when a report is generated)
DEFAULT_DIR	A default directory for a job

DEFAULT_JOB	A default jobname
REPORTER_HOME	The directory REPORTER is installed in
REPORTER_TEMP	A temporary working directory
TOTAL_PAGES	The total number of pages (can be used when a report is generated)
TEMPLATE_DIR	The template directory (useful for locating files relative to the current template)

9.3 Formatting TIME and DATE variables

To add the date to each page you can insert a text object (see Section 5 for more detail on text objects) with the relevant variables substituted in (e.g. see the image below).



The default formatting for the date variable is such that if the day were Saturday 1st February 2020, %DATE% would be generated as Sat Feb 1 2020.

For the time variable, if it were 56 seconds past the 34th minute of the 12th hour of the day then %TIME% would be generated as 12:34:56.

Formatting can be changed for individual instances of the %DATE% and %TIME% variables by using bracketed arguments. For example, %DATE(ddd MMM d yyyy)% provides the default formatting described above. Other options are given in the tables below. Any input characters not included in these tables will be treated as regular text. This allows for further formatting customisation (e.g. %DATE(ddd-MMM/d.yyyy)% for Sat-Feb/1.2020).

For the %DATE% variable, formatting expressions and output are as follows:

Expression	**Output**
d	The day as a number without a leading zero (1 to 31)
dd	The day as a number with a leading zero (01 to 31)
ddd	The abbreviated localised day name (e.g. ’Mon’ to ’Sun’)
dddd	The full localised day name (e.g. ’Monday’ to ’Sunday’)
M	The month as a number without a leading zero (1 to 12)
MM	The month as a number with a leading zero (01 to 12)

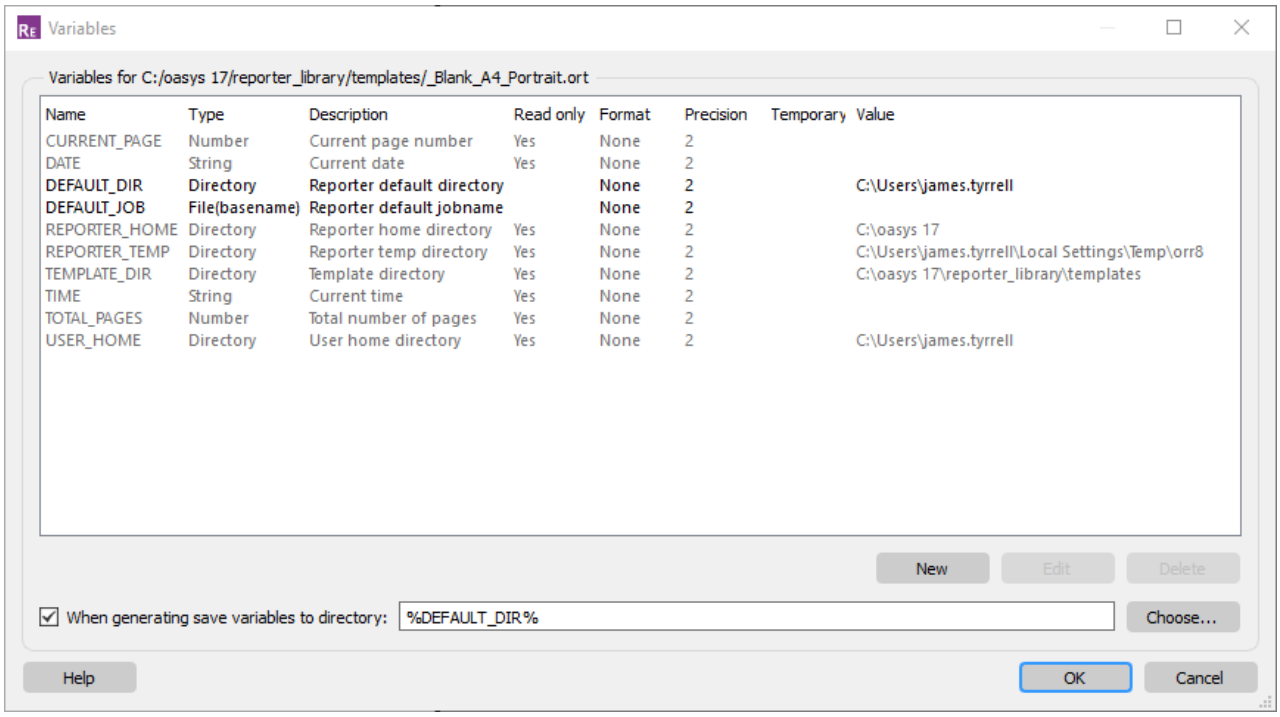
MMM	The abbreviated localised month name (e.g. ’Jan’ to ’Dec’)
MMMM	The full localised month name (e.g. ’January’ to ’December’)
yy	The year as a two digit number (00 to 99)
yyyy	The year as a four digit number

Similarly, for the %TIME% variable:

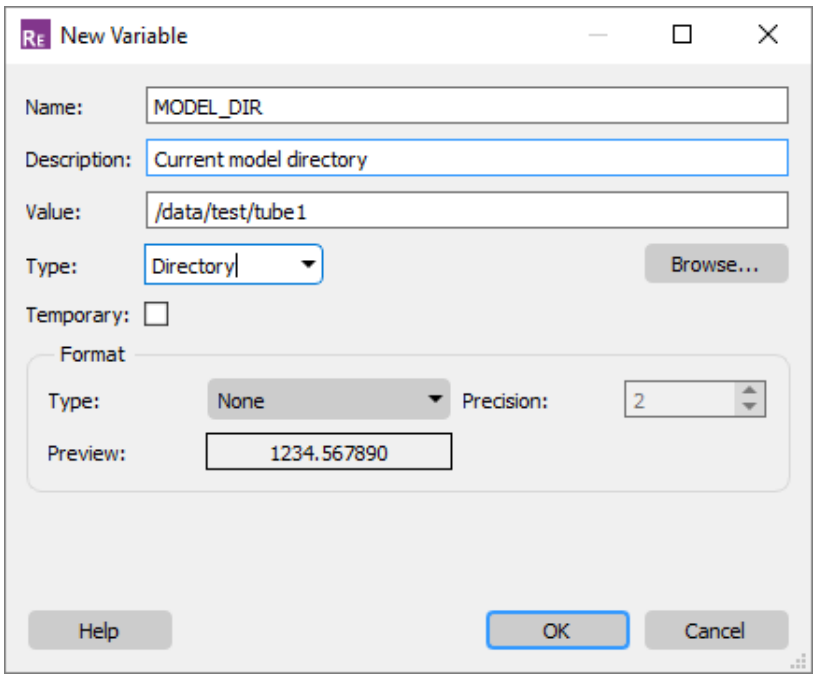
Expression	**Output**
h	The hour without a leading zero (0 to 23 or 1 to 12 if AM/PM)
hh	The hour with a leading zero (00 to 23 or 01 to 12 if AM/PM)
H	The hour without a leading zero (0 to 23, even with AM/PM)
HH	The hour with a leading zero (00 to 23, even with AM/PM)
m	The minute without a leading zero (0 to 59)
mm	The minute with a leading zero (00 to 59)
s	The second without a leading zero (0 to 59)
ss	The second with a leading zero (00 to 59)
z	The millisecond without a leading zero (0 to 999)
zzz	The millisecond with a leading zero (000 to 999)
AP or A	Interpret as an AM/PM time. ’AP’ must be either ’AM’ or ’PM’
ap or a	Interpret as an AM/PM time. ’ap’ must be either ’am’ or ’pm’

9.4 Creating and editing variables

Variables can be viewed, edited, and created by using the **Edit...** option in the **Variables** menu. Selecting this option will bring up the **Variables** window.



Some of the variable such as **CURRENT_PAGE** and **REPORTER_HOME** are standard variables that are predefined by REPORTER. and these cannot be edited or deleted, other user defined variables can be edited or deleted as you chose.



You can create a new variable by selecting **New**. Then in the **New variable** box at the bottom of the window enter the necessary details into the text boxes.

- **Name** - enter the variable name you want to use to refer to this variable. Variable names should only use letter (A-Z) or numbers (0-9) and underscores. REPORTER will automatically convert the name into uppercase and replace any spaces with underscores when the new variable is created.
- **Description** - enter the description for the variable. This is only for reference and is not actually used by REPORTER. However, it is strongly recommended that you give meaningful descriptions for variables.
- **Value** - enter the value for the variable. This can be any text string or number you want.
- **Type** - the variable type allows you to give an indication what the variable will be used for. The following types are predefined in REPORTER.
  - `Directory`
  - `Expression`
  - `File(absolute)`
  - `File(basename)`
  - `File(extension)`
  - `File(tail)`

- `General`
- `Number`
- `String`

Additionally you can give your own variable types if it helps you to manage variables. The `Directory` and `File` types also allow you to choose a directory/file interactively using the **Browse...** button. The different `File` types allow you to extract certain parts of the filename from the file you choose. For example selecting a file '/data/demo/test.key' by using **Browse...** would result in the following:

Variable type	Part of file that is extracted
File(absolute)	/data/demo/test.key
File(basename)	test
File(extension)	key
File(tail)	test.key

- **Temporary** - tick the box if the variable should be temporary or not. This makes no difference to how the variable is used in REPORTER, however for convenience temporary variables can be removed from the template at any point by using the **Delete temporary variables** option in the **Variables** menu.
- **Format** - the format settings allow you to specify how the variable value is displayed within the REPORTER presentation view. Available options are:
  - Floating point number - displays a number variable as a floating point number. The number of decimal places can be specified using the precision setting.
  - Scientific number - displays a number variable as a scientific number. The number of decimal places can be specified using the precision setting.
  - General number - this uses the shorter of the floating point or scientific methods above..
  - Integer - displays a number variable as an integer.
  - Uppercase - displays a string type variable in uppercase.
  - Lowercase - displays a string type variable in lowercase.

  The setting used here is applied to everywhere the variable is displayed in the report, unless a local format setting is used. The format setting does not change the underlying value of the variable.

You then click on the **OK** button to store this new variable. The **Cancel** button will just exit you from this window.

The only variables which can be edited are the user defined ones you create yourself. To edit a variable select the **Variable** option in the **File** menu to bring up the **Variables** window. You can edit the description or value of a variable by clicking on the relevant description or value in the variable list and pressing **Edit**. You cannot edit the variable name. If you want to rename the variable you will have to delete the existing variable and re-create it using the new name.

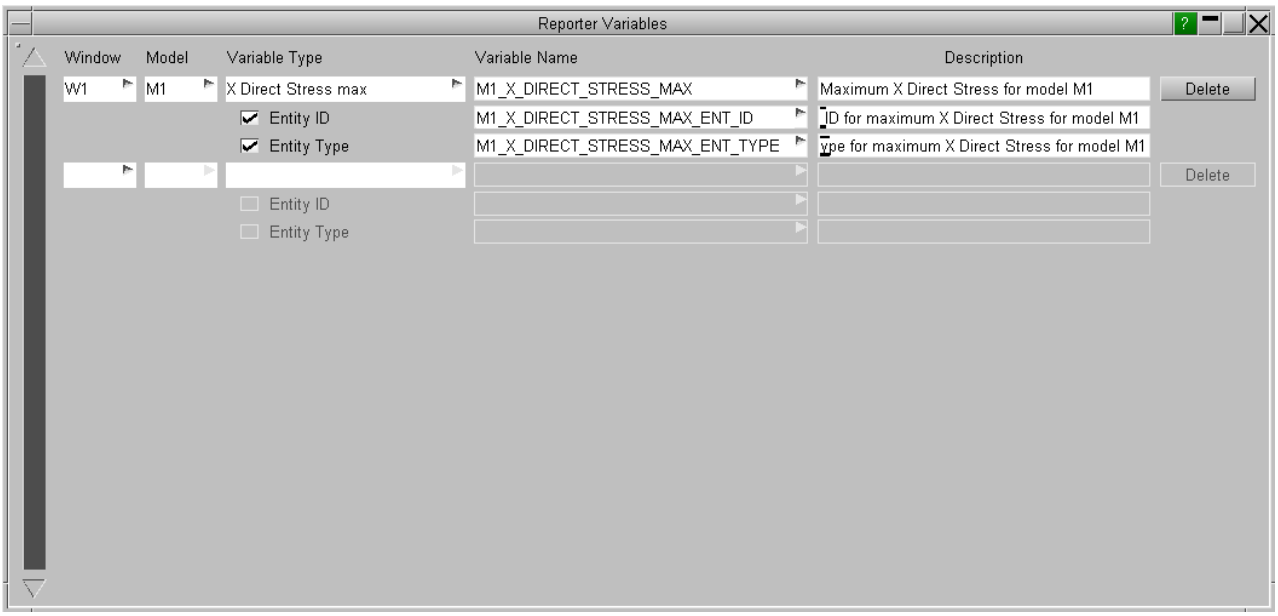
For more information on doing simple maths with variables (by using the expression type) see section9.12.

9.5 Creating a variable using D3PLOT

The **Reporter Objects** floating menu allows you to define values to be returned to REPORTER as variables by pressing the **Variables** button (see section 6.1.1), which launches the **Reporter Variables** floating menu. Variables are limited to the maximum and minimum values displayed on the lefthand side of the D3PLOT window with the corresponding entity type and entity id as additional variables that can be selected if required. Note that variables will only be available for selection in this panel if max/min values are set to be shown in the Data Panel and the plot is a data plot.

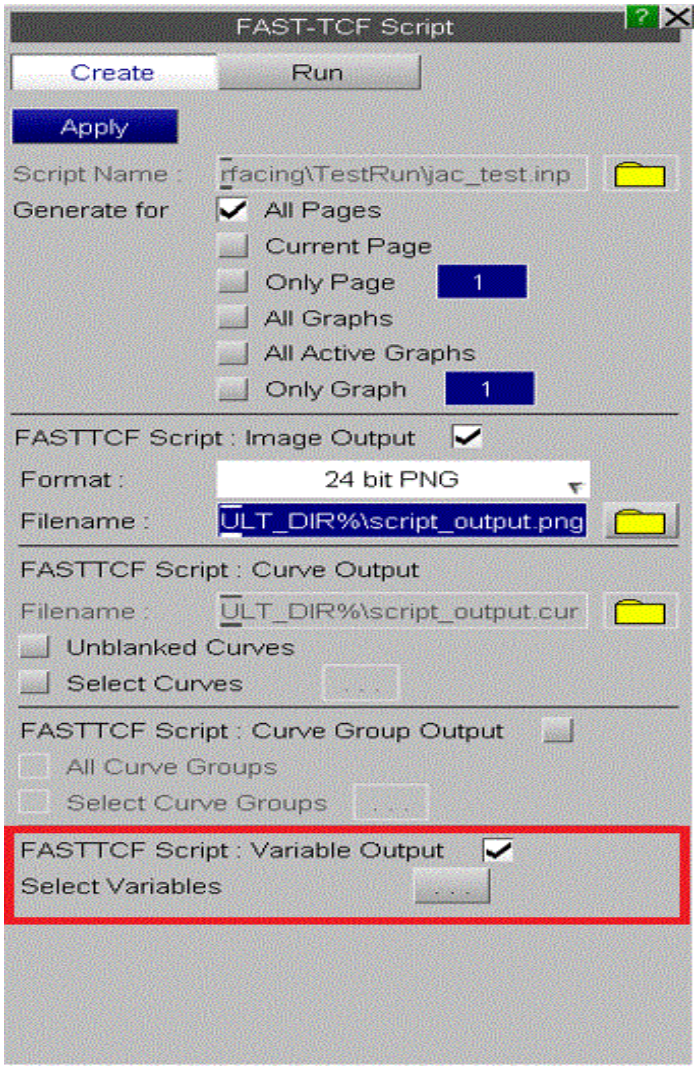
To create a variable first select the window containing the required value using the drop-down menu, then select the correct model using the drop-down menu. The variable type can be selected from the **Variable Type** drop-down menu. Default names and descriptions are then created for the value variable and the corresponding entity id and entity type variables. By default the entity id and entity type variables are not exported to REPORTER. To export them tick the appropriate box. The variable names and descriptions can be edited and the variable name can also be selected using the drop-down menu, which contains a list of the user-defined variables in REPORTER.

You should note that although the variables menu will prevent variable names being duplicated within a single D3PLOT session, care should be taken to avoid duplicating variable names across more than one D3PLOT session as REPORTER will only hold a single value for each variable name.



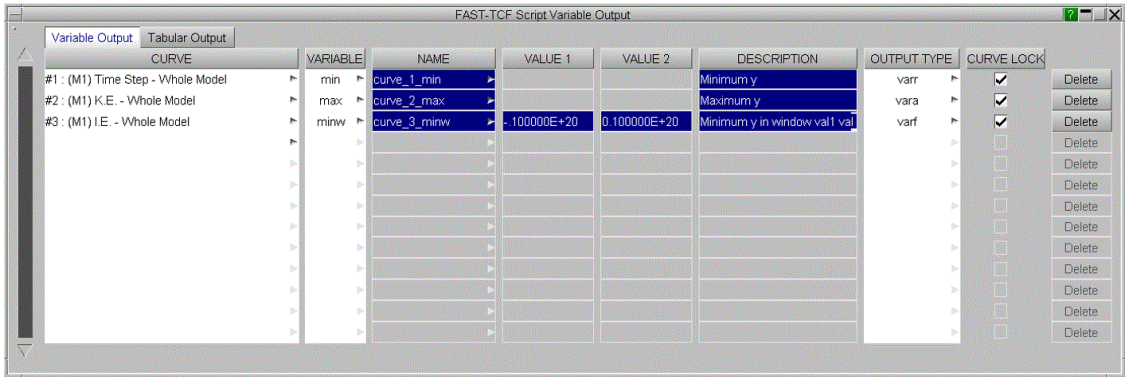
9.6 Creating a variable using T/HIS

The Variables menu can be launched by pressing the button within the FASTTCF Script menu in T/HIS.



In the Variables menu variables can be defined interactively. These variables are exported to REPORTER on exit from a T/HIS REPORTER session.

Variable Output or **Tabular Output** are selected at the top of the menu. Each output request is defined on a row of the table. The curve and variable type are selected using the drop down menu. A default variable name is generated and can be manually edited or a name can be selected from variable names that are present in Reporter. Additional value fields are populated with default values if required and these can be edited. The output description is also populated with default text that can be edited. The output type is selected using the drop down menu. Curve Lock prevents the curve that the variable refers to from being deleted. If a curve is not locked and is deleted, then any variables associated with that curve will also be deleted.



9.7 Creating a variable using an external program/script

Rather than using the **Variables** window to create and define a variable it is also possible to use a program/script to create a variable. (See Appendix E for some examples of programs/scripts)

When REPORTER generates a report and it runs an external program/script, any output lines that take the form

```
VAR <NAME> VALUE="<value>" DESCRIPTION="<description>"
```
or
```
VAR <NAME> VALUE="<value>"
```

will not inserted into the report as text but will be used to create a variable where

- **`<NAME>`** - will become the variable name
- **`<value>`** - will become the value of the variable
- **`<description>`** - will become the variable description

here are a couple of examples

Program/Script Output	Variable Name	Description	Value
`VAR DEFAULT_DIR VALUE="/data/test"`	DEFAULT_DIR	(none)	/data/test
`VAR SPEED VALUE="1000" DESCRIPTION="Impact Speed"`	SPEED	Impact speed	1000

So if you inserted a program/script object "Text output from a program/script" (see Section 9 for more detail on inserting program/script objects) that's output was

```
VAR SPEED VALUE="1000"
```

then REPORTER would create a variable called `SPEED` with the value 1000, and because there is no other output then the inserted text object would come up blank when the report was generated. If the output however was

```
VAR SPEED VALUE="1000"
Impact Speed: %SPEED%
```

then REPORTER would create a variable called `SPEED` with the value 1000, and also create the following text object with the new variable `SPEED` substituted in.

```
Impact Speed: 1000
```

By default any variables that you read from an external program/script will be marked as "temporary". If you do not want the variable to be temporary then the variable name can appended with '!' and this will tell REPORTER not to mark it as temporary. Alternatively '#' can be used to mark the variable as temporary (although this is not needed as it is the default). For example the following two lines would both mark the variable `SPEED` as temporary (the default)

```
VAR SPEED VALUE="1000"
VAR SPEED# VALUE="1000"
```

The following line would mark the variable SPEED as **not** being temporary.

```
VAR SPEED! VALUE="1000"
```

9.8 Creating a variable using a FAST-TCF script

Rather than using the **Variables** window to create and define a variable it is also possible for a FAST-TCF script to create and define variables. You can create a variable in FAST-TCF from one of the following curve results. (See the FAST-TCF section of the T/HIS manual for more details)

Property output	keyword
Minimum x	`minx`
Maximum x	`maxx`
Minimum y	`min`
X at minimum y	`xatmin`
Y at minimum x	`yatmin`
Minimum y in window t1 t2	`minw`
X at minimum y in window t1 t2	`xminw`
Maximum y	`max`
X at maximum y	`xatmax`
Y at maximum x	`yatmax`
Maximum y in window t1 t2	`maxw`
X at maximum y in window t1 t2	`xmaxw`
Average in window t1 t2	`ave`
Hic	`hic`
Hicd	`hicd`
3ms	`3ms`
Y at X	`yatx`
X when Y is passed after gate time	`xygate`
X at first non-zero Y	`xnonz`
X at last non-zero Y	`xfail`
Y value at last non-zero Y	`yfail`
TTI	`tti`

The values for these results need to have already been calculated in the script before you use them to create a variable. The syntax to create a variable takes one of these two forms:

```
var <NAME> <curve> <result> <description>
```
or
```
var <NAME> <curve> <result>
```

- **`<NAME>`** - will become the variable name
- **`<curve>`** - is the curve tag or number
- **`<result>`** - is the result type (min,max,ave,hic,hicd,3ms)
- **`<description>`** - will become the variable description

REPORTER will set the value of the variable to be the value of the result type for the specified curve. Here are a couple of examples

FAST-TCF data			REPORTER data		
FAST-TCF script	**Curve No.**	**Value of the result (Result Type)**	**Variable Name**	**Description**	**Value**
`var DEFORM 1 ave`	1	20 (ave)	DEFORM	(none)	20
`var SPEED 2 max Impact Speed`	2	1000 (max)	SPEED	Impact speed	1000

The variable defined in REPORTER will be marked as temporary.

9.9 Creating a variable from the command line

Variables can be defined in REPORTER when starting from the command line with the `-var` option. For example to define variable `DEFAULT_DIR` you could do:

```
reporter19 .exe -varDEFAULT_DIR=/data/test/tube1
```

If the variable contains spaces then it must be quoted.

```
reporter19 .exe -varDEFAULT_DIR="C:\directory with spaces\tube1"
```

By default variables defined on the command line will not be temporary. You can change this and also specify the variable type on the command line if required. For more details see appendix A.

9.10 Creating a variable from javascript

You can create variables from javascript scripts in REPORTER with the Variable constructor. For example

```
var fred = new Variable(reporter.currentTemplate, "DEFAULT_DIR", "current model
directory", "/data/test1");
```

By default any variable that is made will be marked as temporary but this can be changed. For more details see the Variable javascript reference.

9.11 Deleting variables

9.11.1 Deleting a variable

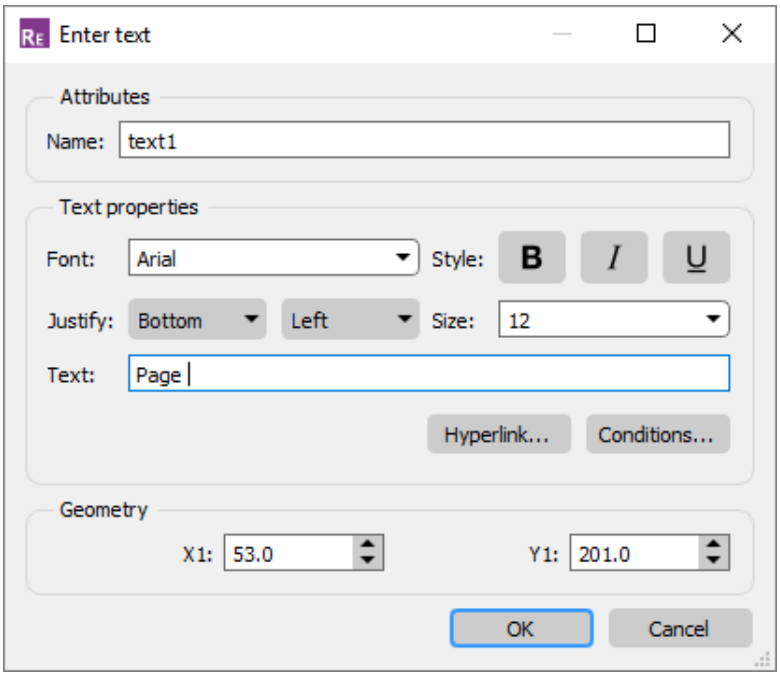
You can delete an user defined variable in the **Variables** window by clicking on the **Delete** button when the relevant variable is selected. Please note that this will delete the variable from the list without bringing up any conformation box. However the variable will not be deleted until **OK** is pressed in the main variables menu.

Predefined variables cannot be deleted.

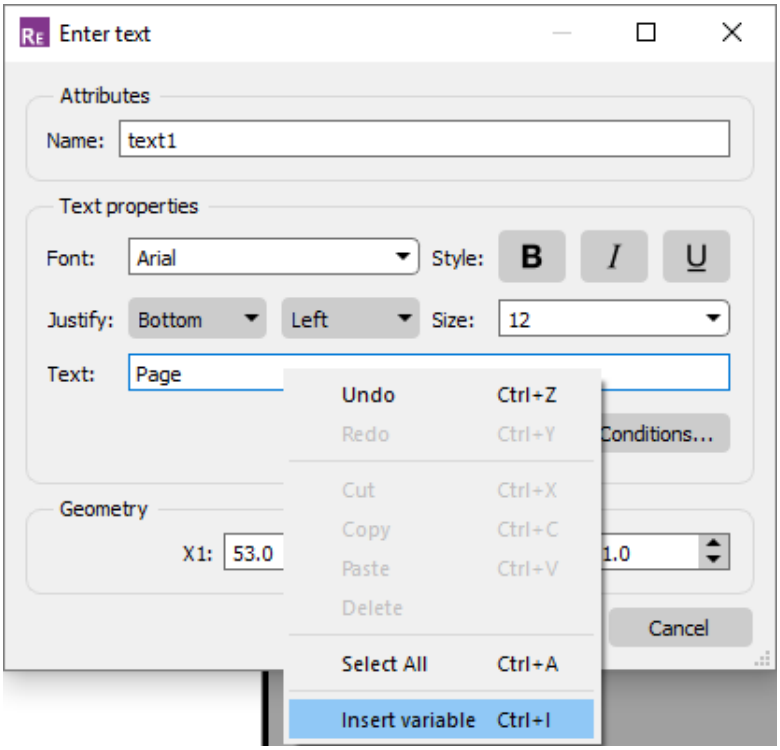
9.11.2 Deleting all temporary variables

Any temporary variables can be deleted by using the **Delete temporary variables** command in the **Variables** menu. Please note that this will delete the tempoarary variables without any conformation.

9.12 Inserting a variable

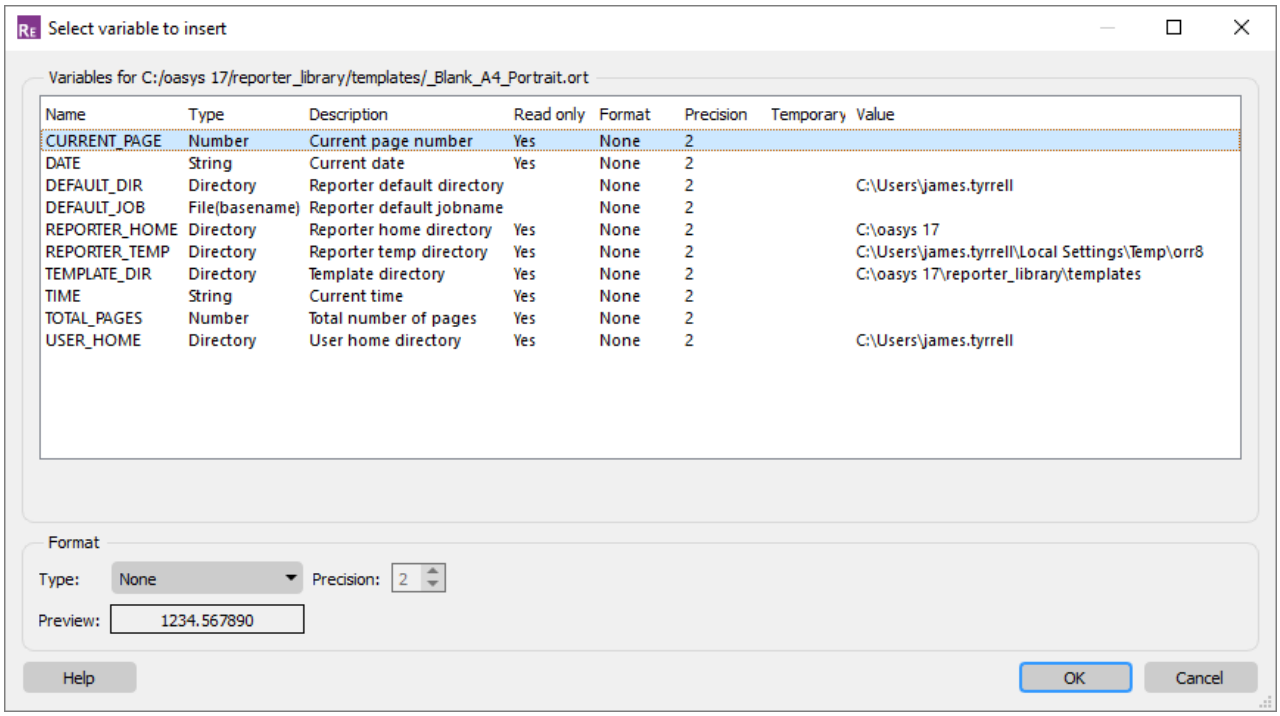


Certain inputs for such things as filenames, text, and program/script arguments can use variables rather than a straight text string. You can insert a variable at the current cursor position by right clicking on the text box



From the popup menu select **Insert variable**.

An **Insert variable** window from which you can select the variable will then be brought up.

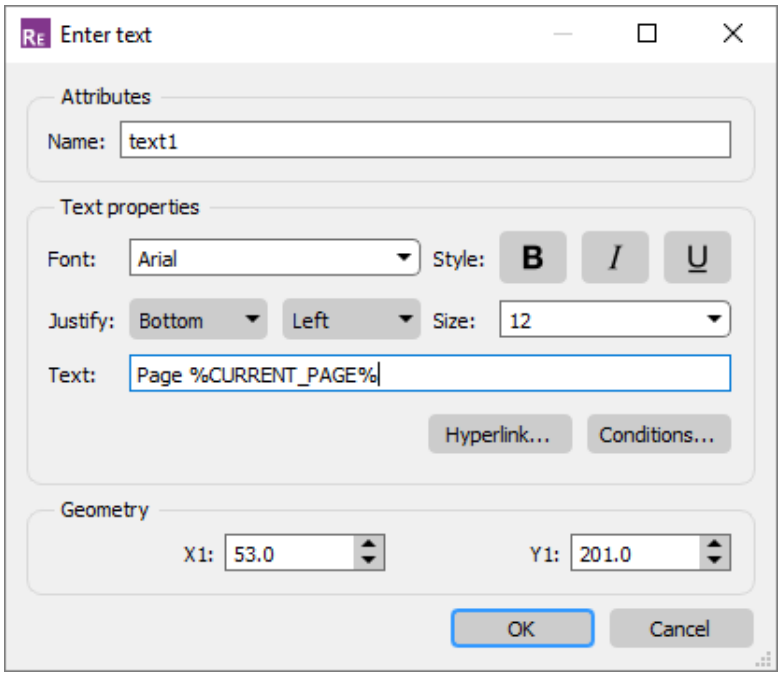


From this window you select the variable you want from the list and click on the **OK** button to insert the variable and exit this window. The **Cancel** button will exit this window with out inserting a variable.

Note in this panel you can set a local format setting for the variable. This is a format that is applied to this instance of variable when viewed in presentation model. The available options are:

- Floating point number - displays a number variable as a floating point number. The number of decimal places can be specified using the precision setting.
- Scientific number - displays a number variable as a scientific number. The number of decimal places can be specified using the precision setting.
- General number - this uses the shorter of the floating point or scientific methods above..
- Integer - displays a number variable as an integer.
- Uppercase - displays a string type variable in uppercase.
- Lowercase - displays a string type variable in lowercase.

This local format setting overrides any global format setting for this variable specified on the main variables panel. However, the format set here is only applied to this instance of the variable.



When entered into a text string the variable needs to be enclosed by **%** signs put at either end of the variable name to

distinguish it from the rest of the text string. In this example the variable **CURRENT_PAGE** has appeared in the text box as**%CURRENT_PAGE%** .

9.12.1 Manually inserting a variable

It is also possible for you to manually enter a variable in by simply typing in the variable name enclosed by **%** signs. When the report is generated the **%CURRENT PAGE%** part of the text string will be replaced with the value of the variable. If a local format is set, this will be displayed within the % signs.

9.12.2 Controlling the precision/decimal places of a variable

The precision of a variable can be set in the **Insert variable** window when inserting it. See the section above on variable format. Alternatively the precision can be set when typing in the variable.

For example, for a variable called **ACCELERATION**, if a local format of a two decimal place floating point number is specified, the variable **ACCELERATION** will appear as %**ACCELERATION(2f)**%. When generated, this will appear as the formatted value. A complete list of the formats is available in the table below.

Format	**Example**	**Input string**	**Output string**
Fixed	%NAME(2f)%	1234.5678 12.345678	1234.56 12.35
Exponential / scientific	%NAME(2e)%	1234.5678 12.345678	1.23e+03 1.23e+01
General. uses exponential format or fixed format (whichever is the most concise)	%NAME(2g)%	1234.5678 12.345678	1.23e+03 12
Integer	%NAME(i)%	1234.5678 12.345678	1235 12
Lower case	%NAME(s)%	Reporter	reporter
Upper case	%NAME(S)%	Reporter	REPORTER

9.13 Using variables in D3PLOT and T/HIS command files and FAST-TCF scripts.

It is also possible to use variables in a D3PLOT or T/HIS command file or FAST-TCF script that is referred to by a D3PLOT or T/HIS object inserted in the template (see Section 6 for more details on inserting D3PLOT and T/HIS objects).

9.13.1 Command files

For a command file you will need to first create the command file using an actual value for the variable and then manually edit the command file to replace this value with the variable name enclosed in % signs.

Example

For example, if you have a simple T/HIS command file that reads in a THF file, creates a curve of x displacement for node 30, and then creates a bitmap image of the curve.

```
READ          31   3   2   3   0   0   0   0
THF           32   3   2  11   0   0   0   0
cube5.thf      4   3   6   5   0   0   0   0
Nodes          4   3   2  12   0   0   0   0
Node 30        3   4   3  14   0   0   0   0
APPLY          5   3   2   2   0   0   0   0
PLOT           1   3   2   1   0   0   0   0
IMAGES        31   3   2  15   0   0   0   0
cube5.bmp     38   3   6  12   0   0   0   0
CAPTURE       38   3   2  25   0   0   0   0
```

I you want to use the variable **DEFAULT_JOB** for the filenames instead of cube5, and the variable **NODE** instead of the node number 30. manually edit the command file to give the following. (Note that the position of the numbers on the right hand side should not modified)

```
READ          31   3   2   3   0   0   0   0
```

```
THF                        32    3    2   11     0    0    0    0
%DEFAULT_JOB%.thf           4    3    6    5     0    0    0    0
Nodes                       4    3    2   12     0    0    0    0
Node %NODE%                 3    4    3   14     0    0    0    0
APPLY                       5    3    2    2     0    0    0    0
PLOT                        1    3    2    1     0    0    0    0
IMAGES                     31    3    2   15     0    0    0    0
%DEFAULT_JOB%.bmp          38    3    6   12     0    0    0    0
CAPTURE                    38    3    2   25     0    0    0    0
```

9.13.2 FAST-TCF scripts

For a FAST-TCF script when you enter the script you need to replace the relevant parts with the variable name enclosed in % signs

Example

For example, a simple FAST-TCF script that will do the same thing as the T/HIS command file above.

```
node 30 disp x tag XDISP
bitmap cube5.bmp XDISP
```

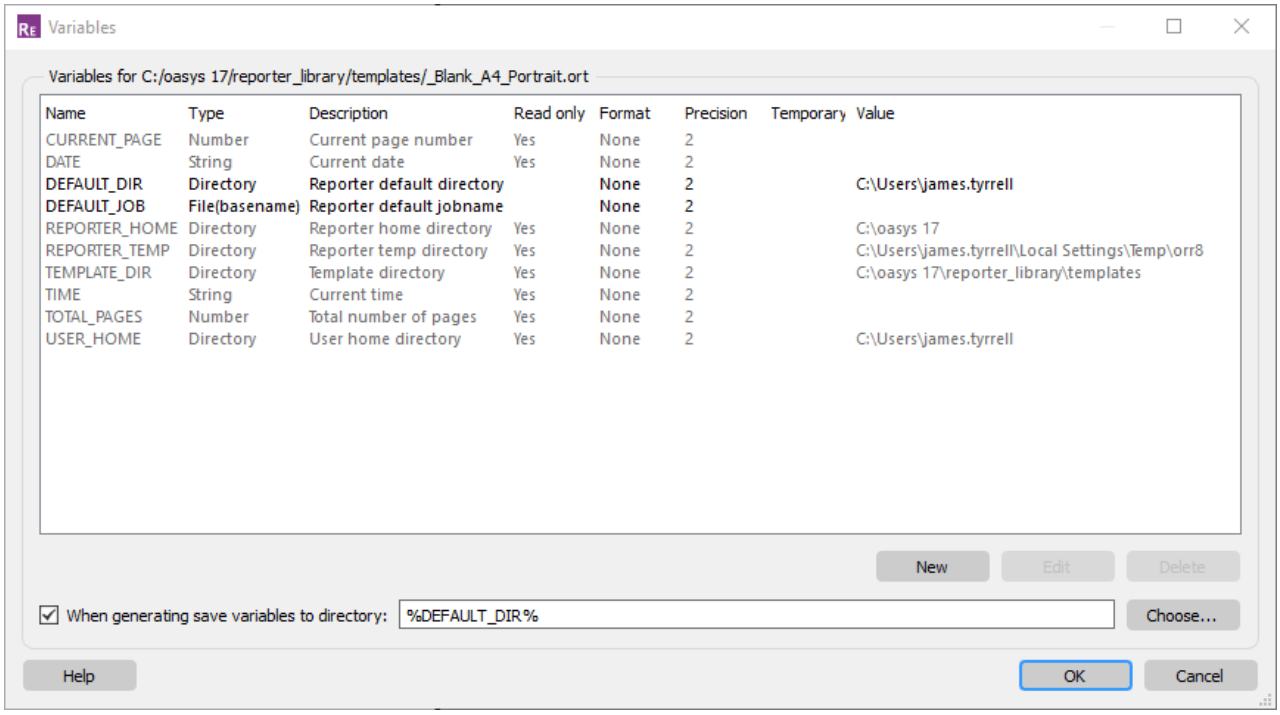
So to make the same changes as the T/HIS command file above (substituting in the variables **DEFAULT_JOB** and **NODE**) gives the following.

```
Node%NODE% disp x tag XDISP
bitmap %DEFAULT_JOB%.bmp XDISP
```

9.14 Saving all the variables to a file after generating a report

After REPORTER generates a report, it can automatically save any variables to a file. The file will be called **`reporter_variables`**. This can be very useful for processing multiple analyses. For example, you could perform several analyses which all dump their variables to a file, and then a summary template could create a table using these files (see section 6 .5 for more details).

At the bottom of the variables window there is a checkbox to turn on this option. You can then give a directory to save the variables into.



You can select the directory or use a variable if required. The directory defaults to `%DEFAULT_DIR%` and is on by default.

9.15 Variable expressions

Sometimes it is useful to do some simple maths on variables in REPORTER. Creating a script to do something this simple is tedious. If you use the **Expression** variable type then REPORTER will evaluate this when required to produce the result. For example assume that you have 2 variables, `FORCE` and `AREA` and you want to calculate a stress. You can do this by:

1. Make a new variable `STRESS`.
2. Set the type to **Expression**.
3. Give the value `%FORCE%/%AREA%` (see secion9.3 for more details) by either typing directly or using the right mouse button and Inserting variables with the menu.

Then if you have some text in the report such as "The stress is %STRESS%" REPORTER will evaluate the stress as required.

The expression can contain +, -, / and * to do addition, subtraction, division and multiplication respectively and can use brackets to enforce which order the expression is evaluated in. The expression is actually evaluated as a JavaScript program so more complex expressions can be formed by using the standard JavaScipt functions (e.g. the Math class). e.g. the following are all valid expressions

- `%FORCE%/%AREA%`
- `Math.sqrt(%X%*%X% + %Y%*%Y%)`
- `Math.min(%X%, %Y%) * Math.sin(Math.PI)`

9.15.1 Rounding values in variable expressions

As the expression is evaluated as a JavaScript program (see the previous section) we can use some of the core functions in JavaScript to alter the variable value. For example, in our example of calculating a variable STRESS from an expression %FORCE%/%AREA% this could have a large number of significant figures in the result.

E.g. if FORCE=10 and AREA=3 then stress is 3.33333333333333 which is far more significant figures than we require.

We can use the core JavaScript function toFixed() to change the number of digits to appear after the decimal point. If we wanted 2 decimal places then we could change the expression to

```
(%FORCE% / %AREA%).toFixed(2)
```

which would change the value of STRESS to 3.33.

Other useful functions are:

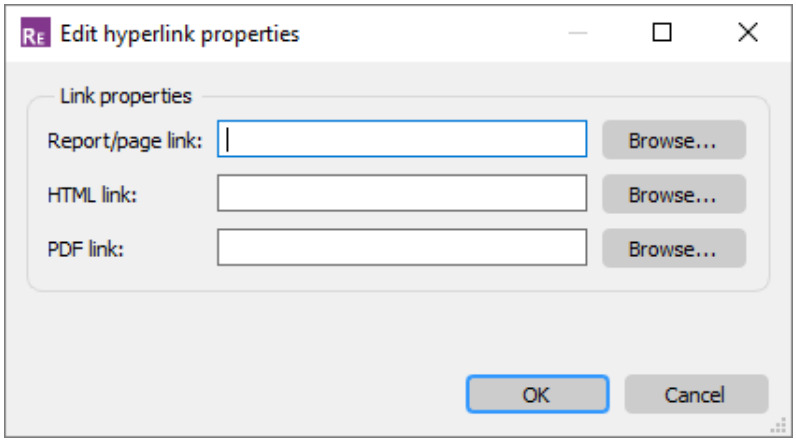
- toExponential(n) which formats the number in exponential (scientific) notation with n digits after the decimal point.
- toPrecision(n) which formats the number with n significant figures.

10. Hyperlinks

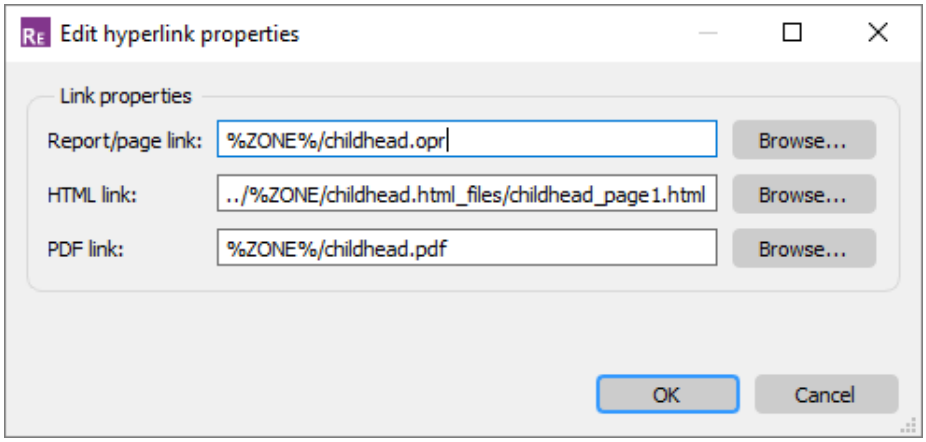
REPORTER currently allows you to create hyperlinks from the following object types
- Text objects
- Image objects
- Table cells
- D3Plot images with external data plots ('blob' plots).

10.1 Adding basic hyperlinks

Objects that support hyperlinks will have a **Hyperlink...** button. Pressing it maps the hyperlink window.



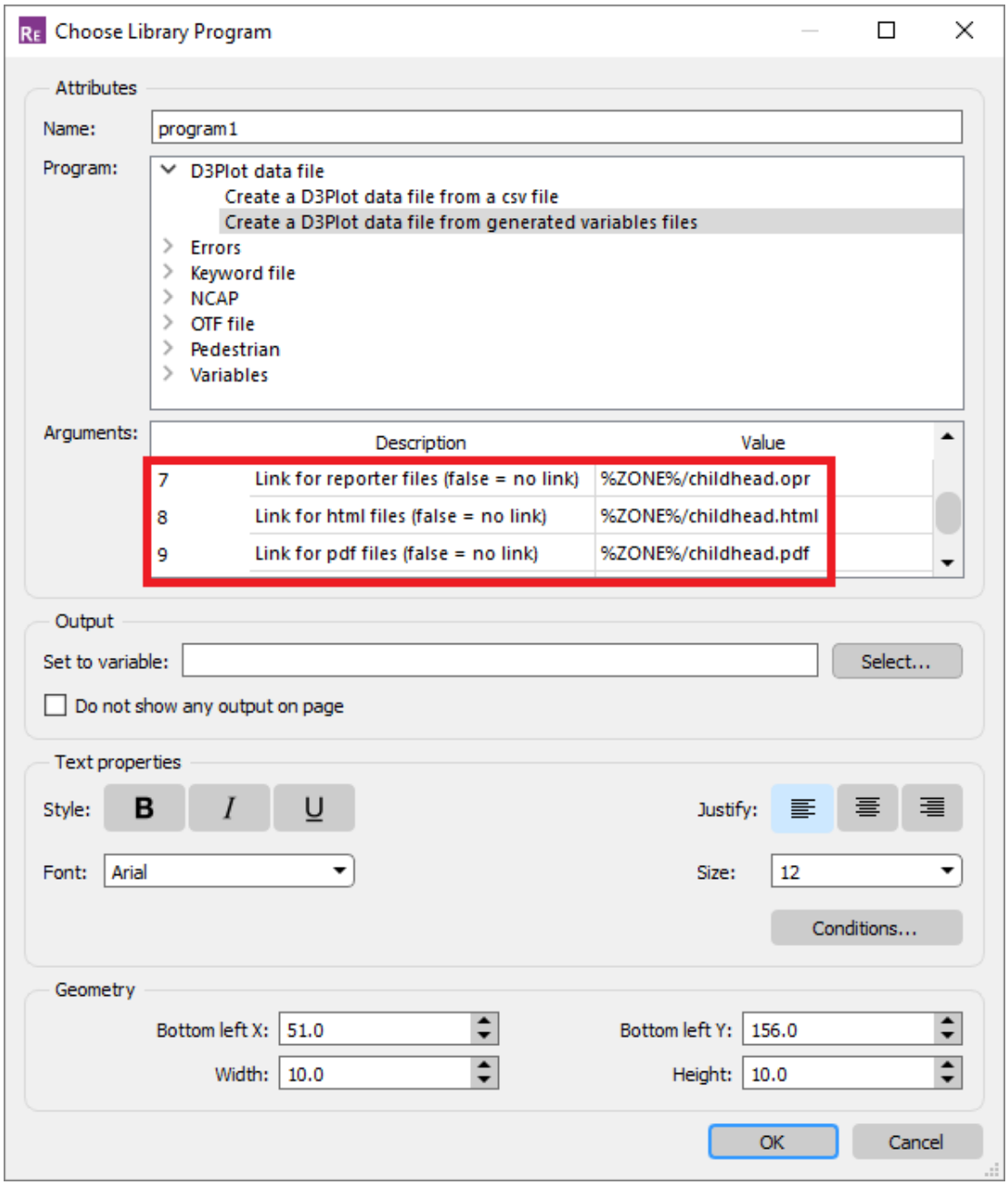
REPORTER can write HTML and pdf and can also save a generated report. As all of these formats support hyperlinks you cannot give a single hyperlink that will work for all of the formats. For this reason REPORTER allows you to give different links for each type. For example in the image below the link is different for each type. If you do not want links for a particular type then leave it blank.



Hyperlinks can be relative or absolute (if you use a relative hyperlink then it is relative to the current document).

10.2 Adding hyperlinks in D3PLOT external data (blob) plots

The data file which D3PLOT uses to create blob plots supports hyperlinks. This enables the user to be able to click on one of the data values on the image and open the report for that data point. The easiest way to create a data file for D3PLOT is with one of the D3PLOT data file library scripts. e.g. below shows the script for generating a data file from `reporter_variables` files.



Arguments 7, 8 and 9 allow you to give your hyperlinks in exactly the same way as a basic hyperlink.

11. Conditional formatting

Conditional formatting can be used in REPORTER to change how text is displayed, depending on if a specific condition has been met. This is very similar to the conditional formatting in Microsoft Excel, but REPORTER can use as many conditions as you wish per object instead of the limit of 3 imposed by Excel.

Conditional formatting is currently supported for the following object types:

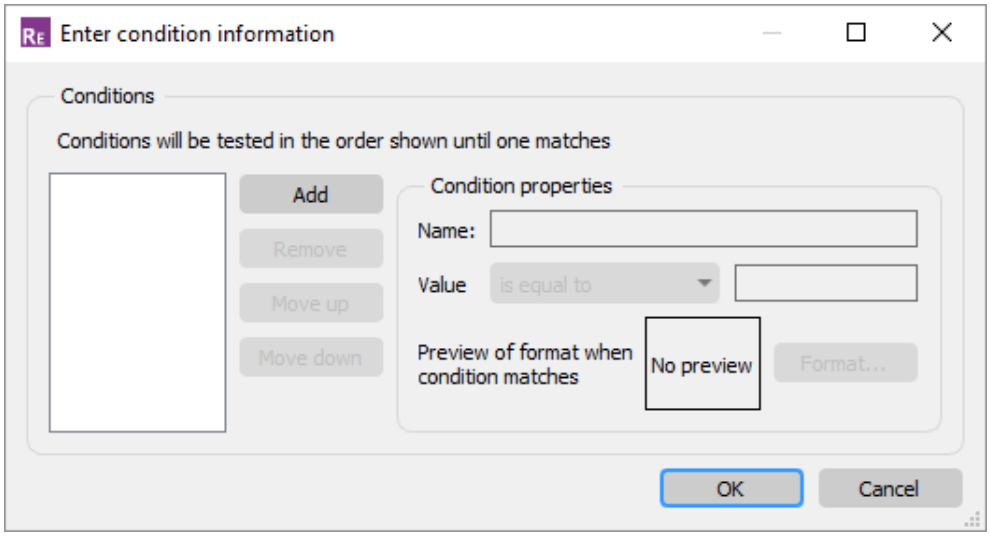
- Text
- Programs/scripts returning text
- Text files
- Table cells
- Text boxes

For example you may want to change the colour of a number in a report depending on the value.
Red if the value is greater than 100
Blue if the number is between 50 and 100
Green if the number is less than 50

This is very easy to do in REPORTER.

11.1. Adding a condition

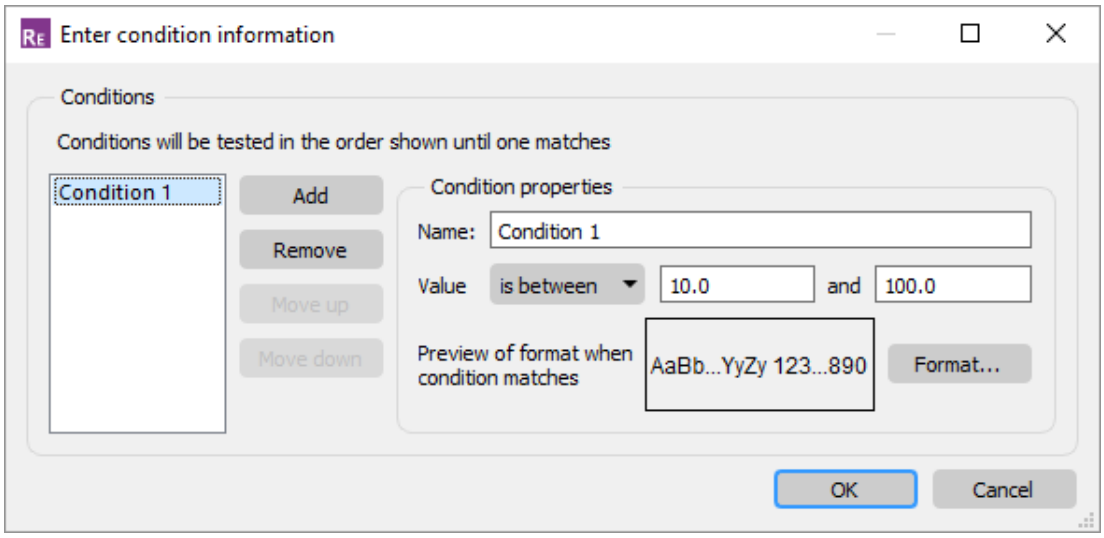
To add a condition for an object, press the **Condition** button. This will start the conditional formatting window.



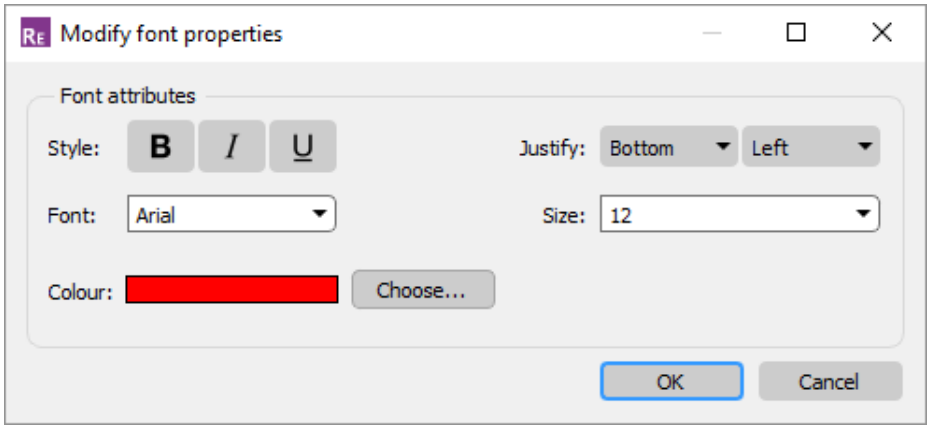
Conditions can be added and removed by using the **Add** and **Remove** buttons. If you have more than one condition, they are tested in the order shown. If the first condition passes the test then that is used, otherwise the second is tested etc. If none of the conditions pass the default font properties for the object are chosen. As the order that they are evaluated is important you can use the **Move up** and **Move down** buttons to change the order.

Once a condition has been added it is given a default name and the condition type is initially set to 'is equal to'

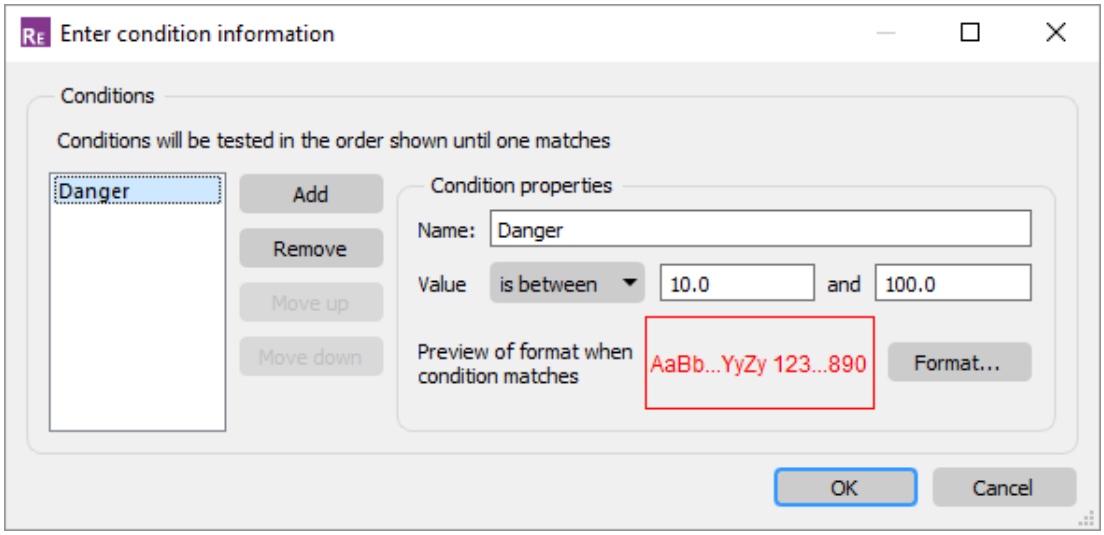
Choose the condition type that you want (see the next section for details) and give the necessary values. For example in the image below the condition will be true if the value is a number between 10.0 and 100.0.



Once you have the correct condition type, the **Format...** button can be used to select the font properties that you want to assign for this condition. In the window (shown below) you can set the font, the style, justification, font size and colour properties.



When you change the font properties, the preview updates to show what the text will look like for this condition. Additionally you can rename the condition to a more meaningful name if required. e.g. in the image below we have made a condition called Danger which will format the text in bold red if the value is a number between 10 and 100.



This process can be repeated as necessary to add as many conditions as you wish.

11.2. Condition types

Condition type	Description
is equal to	Treats the value as a string. Strips leading and trailing white space from the string and compares it to the conditon value. TRUE if the strings are identical. This can also be used to compare integers but should not be used to compare floating point numbers.
is not equal to	As above, but TRUE if the strings are different
is greater than	Treats the value as a real number. It first tries to convert the value and the condition value to real numbers. If this fails the condition is FALSE. If it succeeds then the condition is TRUE if the value is greater than the condition value.
is less than	As above, but TRUE if the value is less than the condition value.
is between	As above, but TRUE if the value is between the two condition values.
is not between	As above, but TRUE if the value is not between the two condition values.
contains string	Treats the value as a string. TRUE if the value contains the condition string.
does not contain string	Treats the value as a string. TRUE if the value does not contain the condition string.
matches regex	Treats the value as a regular expression. TRUE if the regular expression matches.
does not match regex	Treats the value as a regular expression. TRUE if the regular expression does not match.

11.2.1 Regular expressions

REPORTER understands most of the basic operators of perl regular expressions. This section gives a brief introduction into regular expressions (or regexps). For more details please see a suitable book on regular expressions such as Programming Perl.

Regexps are built up from expressions, quantifiers, and assertions. The simplest form of expression is simply a character, e.g. x or 5. An expression can also be a set of characters. For example, [ABCD], will match an A or a B or a C or a D. As a shorthand we could write this as [A-D]. If we want to match any of the captital letters in the English alphabet we can write [A-Z]. A quantifier tells the regexp engine how many occurrences of the expression we want, e.g. x{1,1} means match an x which occurs at least once and at most once. We'll look at assertions and more complex expressions later.

We'll start by writing a regexp to match integers in the range 0 to 99. We will require at least one digit so we will start with [0-9]{1,1} which means match a digit exactly once. This regexp alone will match integers in the range 0 to 9. To match one or two digits we can increase the maximum number of occurrences so the regexp becomes [0-9]{1,2} meaning match a digit at least once and at most twice. However, this regexp as it stands will not match correctly. This regexp will match one or two digits within a string. To ensure that we match against the whole string we must use the anchor assertions. We need ^ (caret) which when it is the first character in the regexp means that the regexp must match from the beginning of the string. And we also need $ (dollar) which when it is the last character in the regexp means that the regexp must match until the end of the string. So now our regexp is ^[0-9]{1,2}$. Note that assertions, such as ^ and $, do not match any characters.

If you've seen regexps elsewhere they may have looked different from the ones above. This is because some sets of characters and some quantifiers are so common that they have special symbols to represent them. [0-9] can be replaced with the symbol \d. The quantifier to match exactly one occurrence, {1,1}, can be replaced with the expression itself. This means that x{1,1} is exactly the same as x alone. So our 0 to 99 matcher could be written ^\d{1,2}$. Another way of writing it would be ^\d\d{0,1}$, i.e. from the start of the string match a digit followed by zero or one digits. In practice most people would write it ^\d\d?$. The ? is a shorthand for the quantifier {0,1}, i.e. a minimum of no occurrences a maximum of one occurrence. This is used to make an expression optional. The regexp ^\d\d?$ means "from the beginning of the string match one digit followed by zero or one digits and then the end of the string".

Our second example is matching the words 'mail', 'letter' or 'correspondence' but without matching 'email', 'mailman', 'mailer', 'letterbox' etc. We'll start by just matching 'mail'. In full the regexp is, m{1,1}a{1,1}i{1,1}l{1,1}, but since each expression itself is automatically quantified by {1,1} we can simply write this as mail; an 'm' followed by an 'a' followed by an 'i' followed by an 'l'. The symbol '|' (bar) is used for alternation, so our regexp now becomes mail|letter|correspondence which means match 'mail' or 'letter' or 'correspondence'. Whilst this regexp will find the words we want it will also find words we don't want such as 'email'. We will start by putting our regexp in parentheses,

(mail|letter|correspondence). Parentheses have two effects, firstly they group expressions together and secondly they identify parts of the regexp that we wish to capture. Our regexp still matches any of the three words but now they are grouped together as a unit. This is useful for building up more complex regexps. It is also useful because it allows us to examine which of the words actually matched. We need to use another assertion, this time \b "word boundary": \b(mail|letter|correspondence)\b. This regexp means "match a word boundary followed by the expression in parentheses followed by another word boundary". The \b assertion matches at a position in the regexp not a character in the regexp. A word boundary is any non-word character such as a space a newline or the beginning or end of the string.

For our third example we want to replace ampersands with the HTML entity '&'. The regexp to match is simple: &, i.e. match one ampersand. Unfortunately this will mess up our text if some of the ampersands have already been turned into HTML entities. So what we really want to say is replace an ampersand providing it is not followed by 'amp;'. For this we need the negative lookahead assertion and our regexp becomes: &(?!amp;). The negative lookahead assertion is introduced with '(?!' and finishes at the ')'. It means that the text it contains, 'amp;' in our example, must not follow the expression that preceeds it.

Characters and Abbreviations in regular expressions

Element	Meaning
c	Any character represents itself unless it has a special regexp meaning. Thus c matches the character c.
\c	A character that follows a backslash matches the character itself except where mentioned below. For example if you wished to match a literal caret at the beginning of a string you would write \^.
\a	This matches the ASCII bell character (BEL, 0x07).
\f	This matches the ASCII form feed character (FF, 0x0C).
\n	This matches the ASCII line feed character (LF, 0x0A, Unix newline).
\r	This matches the ASCII carriage return character (CR, 0x0D).
\t	This matches the ASCII horizontal tab character (HT, 0x09).
\v	This matches the ASCII vertical tab character (VT, 0x0B).
\xhhhh	This matches the Unicode character corresponding to the hexadecimal number hhhh (between 0x0000 and 0xFFFF). \0ooo (i.e., \zero ooo) matches the ASCII/Latin-1 character corresponding to the octal number ooo (between 0 and 0377).
. (dot)	This matches any character (including newline).
\d	This matches a digit.
\D	This matches a non-digit.
\s	This matches a whitespace.
\S	This matches a non-whitespace.
\w	This matches a word character
\W	This matches a non-word character

Sets of Characters

Square brackets are used to match any character in the set of characters contained within the square brackets. All the character set abbreviations described above can be used within square brackets. Apart from the character set abbreviations and the following two exceptions no characters have special meanings in square brackets.

^	The caret negates the character set if it occurs as the first character, i.e. immediately after the opening square bracket. For example, [abc] matches 'a' or 'b' or 'c', but [^abc] matches anything except 'a' or 'b' or 'c'.
-	The dash is used to indicate a range of characters, for example [W-Z] matches 'W' or 'X' or 'Y' or 'Z'.

Using the predefined character set abbreviations is more portable than using character ranges across platforms and languages. For example, [0-9] matches a digit in Western alphabets but \d matches a digit in any alphabet.

Quantifiers

By default an expression is automatically quantified by {1,1}, i.e. it should occur exactly once. In the following list E stands for any expression. An expression is a character or an abbreviation for a set of characters or a set of characters in square brackets or any parenthesised expression.

E?	Matches zero or one occurrence of E. This quantifier means "the previous expression is optional" since it will match whether or not the expression occurs in the string. It is the same as E{0,1}. For example dents? will match ’dent’ and ’dents’.
E+	Matches one or more occurrences of E. This is the same as E{1,MAXINT}. For example, 0+ will match ’0’, ’00’, ’000’, etc.
E*	Matches zero or more occurrences of E. This is the same as E{0,MAXINT}. The * quantifier is often used by a mistake. Since it matches zero or more occurrences it will match no occurrences at all. For example if we want to match strings that end in whitespace and use the regexp \s*$ we would get a match on every string. This is because we have said find zero or more whitespace followed by the end of string, so even strings that don’t end in whitespace will match. The regexp we want in this case is \s+$ to match strings that have at least one whitespace at the end.
E{n}	Matches exactly n occurrences of the expression. This is the same as repeating the expression n times. For example, x{5} is the same as xxxxx. It is also the same as E{n,n}, e.g. x{5,5}.
E{n,}	Matches at least n occurrences of the expression. This is the same as E{n,MAXINT}.
E{,m}	Matches at most m occurrences of the expression. This is the same as E{0,m}.
E{n,m}	Matches at least n occurrences of the expression and at most m occurrences of the expression.

(MAXINT is implementation dependent but will not be smaller than 1024.)

If we wish to apply a quantifier to more than just the preceding character we can use parentheses to group characters together in an expression. For example, tag+ matches a ’t’ followed by an ’a’ followed by at least one ’g’, whereas (tag)+ matches at least one occurrence of ’tag’.

Note that quantifiers are "greedy". They will match as much text as they can. For example, 0+ will match as many zeros as it can from the first zero it finds, e.g. ’2.**000**5’.

Assertions

Assertions make some statement about the text at the point where they occur in the regexp but they do not match any characters. In the following list E stands for any expression.

^	The caret signifies the beginning of the string. If you wish to match a literal ^ you must escape it by writing \^. For example, ^#include will only match strings which begin with the characters ’#include’. (When the caret is the first character of a character set it has a special meaning, see Sets of Characters.)
$	The dollar signifies the end of the string. For example \d\s*$ will match strings which end with a digit optionally followed by whitespace. If you wish to match a literal $ you must escape it by writing \$.
\b	A word boundary. For example the regexp \bOK\b means match immediately after a word boundary (e.g. start of string or whitespace) the letter ’O’ then the letter ’K’ immediately before another word boundary (e.g. end of string or whitespace). But note that the assertion does not actually match any whitespace so if we write (\bOK\b) and we have a match it will only contain ’OK’ even if the string is "Its OK now".
\B	A non-word boundary. This assertion is true wherever \b is false. For example if we searched for \Bon\B in "Left on" the match would fail (space and end of string aren’t non-word boundaries), but it would match in "tonne".
(?=E)	Positive lookahead. This assertion is true if the expression matches at this point in the regexp. For example, const(?=\s+char) matches ’const’ whenever it is followed by ’char’, as in ’static const char *’. (Compare with const\s+char, which matches ’static const char *’.)

(?!E)	Negative lookahead. This assertion is true if the expression does not match at this point in the regexp. For example, const(?!\s+char) matches ’const’ except when it is followed by ’char’.

12. Fonts

12.1 Supported Fonts

REPORTER supports most fonts, giving you control over the look of your reports, and allowing you to create templates that match your organisation's branding. The following font types are supported:

- TrueType fonts and collections (*.ttf* and *.ttc* files)
- OpenType fonts and collections (*.otf* and *.otc* files)
- Certain Type1 fonts (Printer Font Binary *.pfb* files and their *.pfm* metrics files)

The fonts that appear in REPORTER's font dialogs depend on the fonts available on your operating system. REPORTER searches the following locations for fonts:

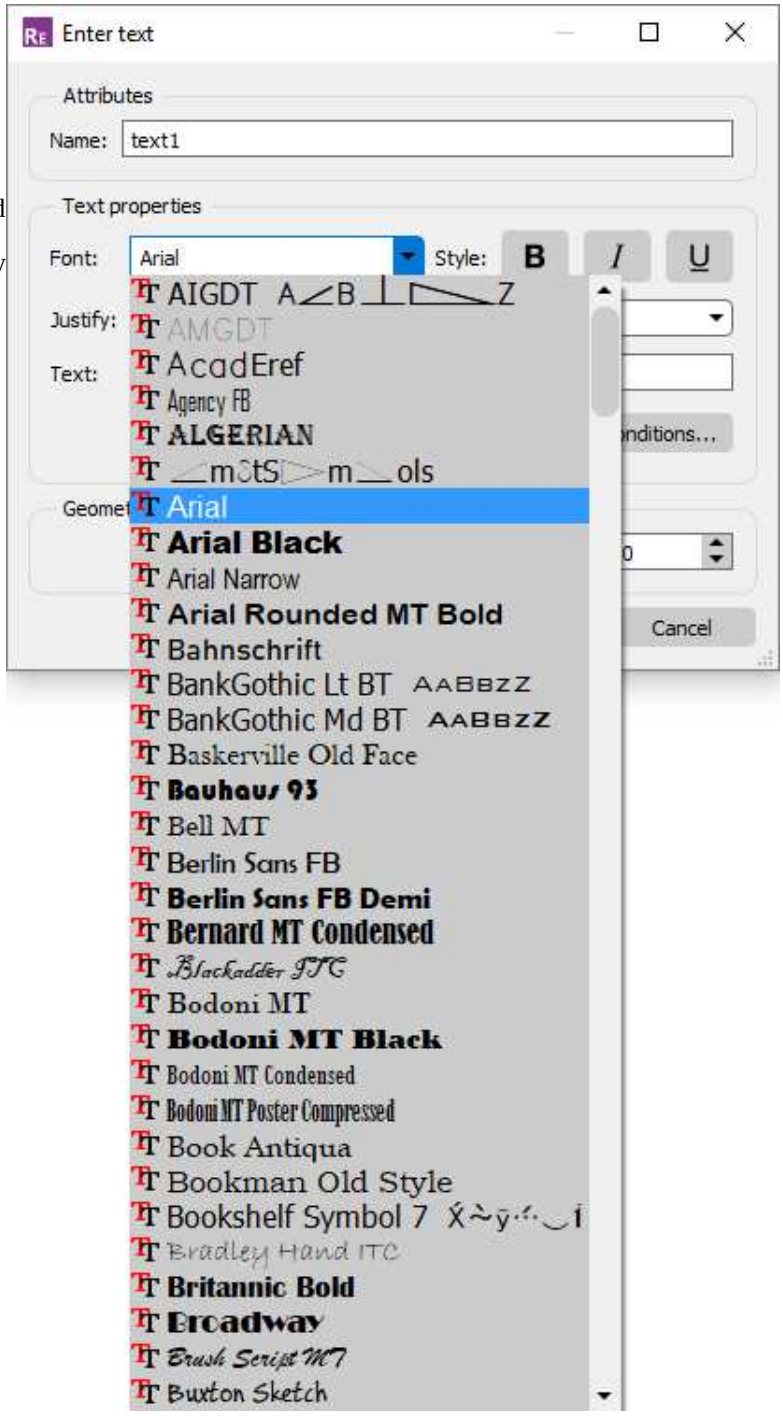
On Windows:

- *%WINDIR%\Fonts* (typically *C:\Windows\Fonts*)

On Linux:

- */usr/share/fonts*
- */usr/share/X11*
- */usr/local/share/fonts*

When REPORTER launches, it scans these locations for font files, processes the files it finds, and then writes the data to a cache file located at *$OA_HOME/reporter_font_cache*. On subsequent launches, REPORTER should load more quickly, because thereafter it only needs to process newly installed fonts.



12.2 Legacy Fonts

In addition, the four **legacy** fonts are always supported regardless of operating system:

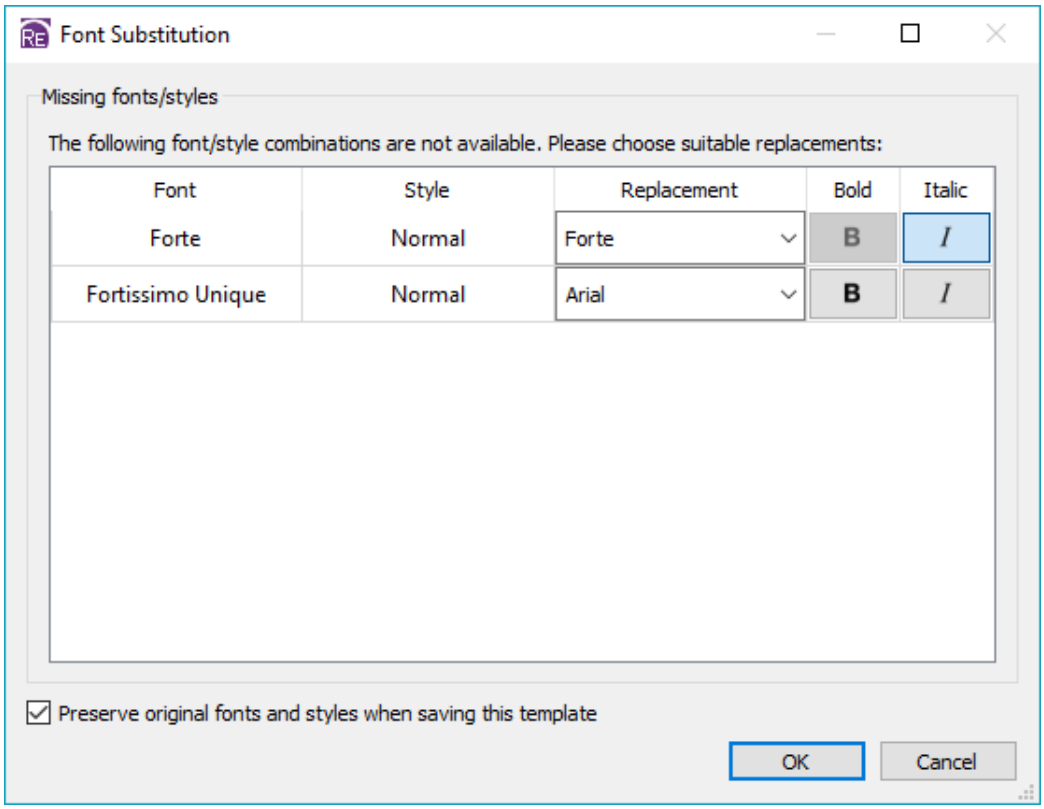
- Courier
- Helvetica
- Symbol
- Times

This means that REPORTER templates created using earlier versions (before REPORTER 16.0, when only these four fonts were available) should continue to work as normal.

12.3 Font Mapping

Font mapping is a method of providing suitable alternatives if a requested font is unavailable on your system. For example, another user may share with you a REPORTER template containing selected fonts that were available on their system, but not on yours. Alternatively, you might create a template on your Windows computer, and then generate it in batch on a Linux server that has a different set of fonts installed.

12.3.1 Font Substitution Dialog



If you open a REPORTER template containing fonts that are missing from your system, a Font Substitution dialog will appear:

In the example above, the font Forte is available on the system, but the style Normal is not available, so Forte Italic is being offered as a replacement. The font Fortissimo Unique cannot be found on the system, so Arial is being offered as a replacement. If you want to change the default replacement, you can select a different font from the drop-down list, and check or uncheck the Bold and Italic buttons. Note that for some fonts, certain style combinations might be unavailable. In the example above, Forte Bold is unavailable so the Bold button is disabled.

Below the list of missing font/style combinations, there is an option to ‘Preserve original fonts and styles when saving this template’ (selected by default). When selected, the original font names and styles will be written when the template is saved. This is useful when you want to work on or edit another user’s template without irreversibly changing the original fonts. It is also useful if you regularly create a template on Windows but then generate it on a Linux server. When run in batch, ‘Preserve original fonts and styles when saving this template’ is switched on by default, meaning that your original Windows font names will be used in any report and PowerPoint files generated (useful if you normally then view the output back on Windows).

If you want to save the Font Substition dialog replacements as permanent changes to your template, simply uncheck the ‘Preserve original fonts and styles when saving this template’ option.

12.3.2 Font Mapping Table

REPORTER uses a font mapping table to determine which font to offer as a suitable replacement for any given font. An extract from the font mapping table is shown below:

font-family	generic	replacement	replacement 2
Arial Special	sans-serif	Arial	Helvetica
Airal Unicode MS	sans-serif	Arial	Helvetica
Bahnschrift	sans-serif		
Baskerville Old Face	serif		
Batang	serif		
BatangChe	serif		
Bauhaus 93	fantasy		
Beesknees ITC	fantasy		
Bell MT	serif	Bitstream Charter	
Berlin Sans FB	sans-serif		
Bernard MT Condensed	serif		
Bon Apetit MT	monospace		
Book Antiqua	serif	Palatino Linotype	DejaVu Serif
Bookman	serif	Bookman Old Style	
Bookman Old Style	serif	Bookman	
Bookshelf Symbol	symbol		
Bradley Hand ITC	cursive		

The font mapping table supplied in the REPORTER installation was compiled from lists of common font mapping alternatives for Windows and Linux. For some fonts, one or two specifically named fonts are given as **replacement** and **replacement 2**. Often, these are alternatives within a font family (e.g. Arial Unicode MS → Arial) or typical Linux replacements for typical Windows fonts (e.g. Bell MT → Bitstream Charter). All of the fonts in the mapping table have a **generic** replacement type. This can be one of:

- cursive
- fantasy
- monospace
- sans-serif
- serif
- symbol

These generic categories are the same as the widely recognised Cascading Style Sheets (CSS) *font-family* property generic categories, with the addition of **symbol** (an extra generic cateogry added in REPORTER to aid support of the legacy font Symbol).

If REPORTER encounters a missing font, it will search first for the **replacement** font, and then the **replacement 2** font (if either is specified). If neither of those fonts are available, it will search for the generic default. These are:

generic	Windows	Linux
cursive	Monotype Corsiva	URW Chancery L
fantasy	Impact	Impact
monospace	Courier New	Courier 10 Pitch
sans-serif	Arial	Liberation Sans
serif	Times New Roman	Nimbus Roman No9 L
symbol	Symbol	Standard Symbols L

Failing that, it will use the generic sans-serif default (Arial on Windows, Liberation Sans on Linux). This will happen if the requested font is not featured in the font mapping table, or if one of the generic defaults is not installed.

The generic default fonts listed above can be customised via Oasys preferences (the *oa_pref* file).

The font mapping table is stored in CSV format at *$OA_INSTALL/reporter_library/fonts/font_mapping.csv*. Please contact Oasys Ltd if you encounter repeated font substitution issues. Alternatively, you can edit it to customise your own font mappings. You can also specify a font mapping CSV file at an alternative location to the installation location by using the preference *reporter*font_mapping_table*.

12.4 Fonts in report output

Fonts are handled slightly differently in each of the supported output formats (PowerPoint, HTML and PDF).

12.4.1 PowerPoint

Font names and styles in the REPORTER template are written to the PowerPoint file. When the file is opened,

Microsoft PowerPoint will use the requested font name if available. If it is unavailable (e.g. if the PowerPoint file was created by a user with access to different fonts from you) then PowerPoint will try to use a suitable alternative, or will revert to the default font. This behaviour is no different from normal PowerPoint use.

If you create a template on Windows but then generate it in batch on a Linux server, ‘Preserve original fonts and styles when saving this template’ is switched on by default. This means that your original Windows font names will be used in the PowerPoint output (useful when the PowerPoint is viewed back on a Windows computer).

12.4.2 HTML

Font names in the REPORTER template are written to the *font-family* property in the HTML file using the information in the font mapping table. For example, if you have selected Times New Roman, the following may appear in HTML:

```
font-family:"Times New Roman","Times","Nimbus Roman No9 L",serif;
```

Your web/HTML browser will try to display the text in Times New Roman, followed by the alternative fonts listed (using similar logic to REPORTER). In this way, compatibility between systems is preserved as far as possible.

12.4.3 PDF

PowerPoint and HTML files just contain the *names* of the fonts used. However, fonts are only supported properly by PDF readers if the relevant subset of the font file itself is embedded in the PDF file. When REPORTER writes PDF output, it will only embed fonts with the appropriate permission bits. If you encounter an error when writing PDF files, check that the fonts you have selected do not have licence restrictions on embeddability.

13. Scripting

REPORTER has a JavaScript interpreter embedded in it to enable you to perform complex operations through scripts. There are currently 3 ways to run a script in REPORTER.

- Running a library script installed in the /library/scripts directory.
- Inserting a script object onto a page. This does not create any direct output itself, but can create output which other objects in the template use.
- Running a script from the command line with the `-script` option.

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 REPORTER. 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, REPORTER extends JavaScript by defining several new object classes specifically for REPORTER. A detailed reference on these classes is given the JavaScript class reference appendix. Over time this functionality may be extended. If you need to do something which is not possible with the current functionality then contact Oasys Ltd.

This chapter is not intended to be an introduction or a tutorial for JavaScript. There are many resources on the web for that. However a few examples are given to show the sort of things that are possible with scripts. Additionally, there are several good books on JavaScript. Highly recommended is JavaScript, The Definitive Guide by David Flanagan, published by O'Reilly, ISBN: 0-596-00048-0.

In REPORTER 17.0 and earlier the implementation supported ECMAScript 5 features of JavaScript. In REPORTER 18.0 the implementation has been upgraded to support ECMAScript 6 (and newer) features of JavaScript.

Probably the best way to see what sort of things are easily possible in REPORTER using JavaScript is to look at the library scripts which are given out with REPORTER in the /library/scripts directory. For more details of the scripts see the library scripts appendix.

13.1 Example scripts

Example 1: Percent change in two values

Problem

Take two input variables VALUE and VALUE_BASE
Calculate new variable PERCENT = 100*(VALUE - VALUE_BASE) / VALUE_BASE)
Check if VALUE_BASE=0 and if so don't do the division but set PERCENT to 100

Solution

```
var percent;

// Get variable values from template
var value = reporter.currentTemplate.GetVariableValue("VALUE");
var base_value = reporter.currentTemplate.GetVariableValue("VALUE_BASE");

// Check that the variables exist
if (value == null) throw Error("no VALUE variable\n");
if (base_value == null) throw Error("no VALUE_BASE variable\n");

// Extract numbers from variables
var v = parseFloat(value);
```

```
var bv = parseFloat(base_value);

// Check that the variables are valid numbers
if (isNaN(v)) throw Error("VALUE " + value + " is not a valid number\n");
if (isNaN(bv)) throw Error("VALUE_BASE " + base_value + " is not a valid
number\n");

// Check for zero (very small) base value
if (Math.abs(bv) < 1.0e-20)
    percent = 100;
else
    percent = 100*((v-bv)/bv);

// Create new variable PERCENT
var pvar = new Variable(reporter.currentTemplate, "PERCENT",
                        "Percent change", percent.toFixed(2));
```

Discussion

Variables in REPORTER are stored in each template so to get the values of the variables `VALUE` and `VALUE_BASE` we need to get the template that we are using. The easiest way to do this is to use the `currentTemplate` property of the `reporter` object that is created when REPORTER starts. Once we have the `template` there is a method `GetVariableValue` that allows us to get a variable value.

`GetVariableValue` returns the value of the variable as a string or null is the variable does not exist. We can easily check for this and terminate with an error if the variable is missing.

We want to get the numerical values of the variables and check if they are valid numbers. The standard javascript functions `parseFloat()` and `isNaN()`allow us to do this.

To check if the value is zero (or very small) we use the standard `Math.abs()` function and calculate a value accordingly.

To create a new variable we use the `Variable` constructor. This takes the template, the variable name, description and value as arguments. Finally, maths in javascript is performed in double precision so the value we calculated will be given to many significant figures. We are not interested in this so we use the standard `Number.toFixed()` function to limit the number of decimal places to 2.

The source code for this example is available here.

Example 2: Magnitude from the three vector components

Problem

Given three variables X, Y and Z calculate the vector magnitude and store it in a variable LENGTH.

Solution

```
// Get variable values from template
var x = reporter.currentTemplate.GetVariableValue("X");
var y = reporter.currentTemplate.GetVariableValue("Y");
var z = reporter.currentTemplate.GetVariableValue("Z");

// Check that the variables exist
if (x == null) throw Error("no X variable\n");
if (y == null) throw Error("no Y variable\n");
if (z == null) throw Error("no Z variable\n");

// Extract numbers from variables
var X = parseFloat(x);
var Y = parseFloat(y);
var Z = parseFloat(z);

// Check that the variables are valid numbers
if (isNaN(X)) throw Error("X " + x + " is not a valid number\n");
if (isNaN(Y)) throw Error("Y " + y + " is not a valid number\n");
```

```
if (isNaN(Z)) throw Error("Z " + z + " is not a valid number\n");

// Calculate magnitude
var length = Math.sqrt(X*X + Y*Y + Z*Z);

// Check for valid magnitude
if (isNaN(length)) throw Error("Bad vector magnitude\n");

// Create new variable LENGTH
var lvar = new Variable(reporter.currentTemplate, "LENGTH",
                        "vector magnitude", length);
```

Discussion

This is done using very similar methods to example 1. The only differences here are using the function `Math.sqrt()` and we do not use the standard `Number.toFixed()` function as the length could be smaller than 2 decimal places. Instead we could use `Number.toPrecision()` or `Number.toExponential()` if we wanted to format the result instead of leaving it with several decimal places.

The source code for this example is available here.

Example 3: Setting a character variable according to the result of a calculation

Problem

Input variable = PERCENT
If (abs(PERCENT) < 5.0) then new variable RESULT = ’OK’
otherwise ’not OK’

Solution

```
var result;

// Get variable value from template
var percent = reporter.currentTemplate.GetVariableValue("PERCENT");

// Check that the variable exist
if (percent == null) throw Error("no PERCENT variable\n");

// Extract number from variable
var p = parseFloat(percent);

// Check that the variable is a valid number
if (isNaN(p)) throw Error("PERCENT " + percent + " is not a valid number\n");

// Check for less than 5
if (Math.abs(p) < 5.0)
    result = "OK";
else
    result = "not OK";

// Create new variable RESULT
var rvar = new Variable(reporter.currentTemplate, "RESULT",
                        "is it OK?", result);
```

Discussion

This uses exactly the same methods as examples 1 and 2. The only difference is that the value used in the Variable constructor is a character string, not a number.

The source code for this example is available here.

Example 4: Reading a T/HIS curve file and operating on it

Problem

input variables = CURVE_FILE and GATE_TIME.
read the T/HIS curve file, calculate average y-value of all points that occur after x-value=GATE_TIME. Return the average in a new variable Y_AVERAGE

Solution

```
var count, line, x, y, X, Y, ytot, ny;

// Get variable values from template
var curveFile = reporter.currentTemplate.GetVariableValue("CURVE_FILE");
var gateTime = reporter.currentTemplate.GetVariableValue("GATE_TIME");

// Check that the variables exist
if (curveFile == null) throw Error("no CURVE_FILE variable\n");
if (gateTime == null) throw Error("no GATE_TIME variable\n");

// Check curve file exists
if (!File.Exists(curveFile)) throw Error("Curve file " + curveFile + " does not
exist\n");

// Check gateTime is a valid number
var t = parseFloat(gateTime);
if (isNaN(t)) throw Error("Gate time " + gateTime + " is not a valid number\n");

// create a new File object
var file = new File(curveFile, File.READ);

// Zero variables
count = 0;
ytot = 0;
ny = 0;

// Keep reading lines from the file until we get to the end of the file
while ( (line = file.ReadLine() ) != File.EOF)
{
    if (line.charAt(0) == '$')
        continue;
    else if (line.match(/CONTINUE/))
        break;
    else
    {
        count++;

// Skip the four title lines at the top of the curve file
        if (count > 4)
        {
// strip leading and trailing apaces
            line = line.replace(/^\s+/, "");
            line = line.replace(/\s+$/, "");
            result = line.match(/([0-9eE+\-\.]+)\s*,?\s*([0-9eE+\-\.]+)/);
            if (result != null)
            {
                x = result[1];
                y = result[2];

// Extract numbers
                X = parseFloat(x);
                Y = parseFloat(y);

// Check that they are valid numbers
                if (isNaN(X)) throw Error("X " + x + " is not a valid
number\n");
                if (isNaN(Y)) throw Error("Y " + y + " is not a valid
number\n");
```

```
// If greater than gate time then include value
                if (X > t)
                {
                    ny++;
                    ytot += Y;
                }
            }
        }
    }
}

// Close the file
file.Close();

// If we have read any values calculate average and set variable
if (ny)
{
    ytot /= ny;
// Create new variable LENGTH
    var ave = new Variable(reporter.currentTemplate, "Y_AVERAGE",
                           "average Y value", ytot);
}
```

Discussion

This example uses the `File` class which REPORTER defines to read the T/HIS curve file. The function `File.Exists()` can be used to test if a filename is valid. Then the `File constructor`, `ReadLine()` and `Close()` functions are used to read the data from the file.

To extract the xy data pairs from the file we use a regular expression. This is perhaps the most complicated part of the program. We want to be able to read x and y values that can be separated by a comma, one or more spaces, or both. If we break the expression `([0-9eE+\-\.]+)\s*,?\s*([0-9eE+\-\.]+)` into it's constituent parts we get:

`([0-9eE+\-\.]+)`. The `[]` groups characters that we allow to match. `-` and `.` have special meanings so they have to be escaped with a `\` character. So this means we are allowing any of the characters `0123456789eE+-.` to match. The `[]` specifies a single character so we use + to mean one or more. Finally, using `()` captures the expression so we can extract the value that matched. So this will match values such as '10', '1.2345', '1.0e+05', '-23.4'

`\s*,?\s*`. The `\s` matches a single space. A `*` means that it will try to match 0 or more spaces (as many as are present). The `,` matches a comma and the `?` means match either 0 or 1 of them. So this expression means "Match 0 or more spaces followed by 0 or 1 commas followed by 0 or more spaces".

More details on regular expressions can be found in the Conditional formatting chapter as these can use regular expressions.

Once we have extracted the data values with the regular expression we can easily calculate the average and make a new variable using the techniques in the first 3 examples.

The source code for this example is available here.

A. Command line arguments and oa_pref options

A.1 Command line arguments

The following command line arguments are available in REPORTER. Unless stated otherwise, all command line options are evaluated in the order that they are given.

Argument	Description
`file.orr` or `-file=file.orr`	Opens REPORTER file "file.orr"
`-pdf=file.pdf`	Creates a pdf file "file.pdf"
`-html=file.html`	Creates a HTML file "file.html"
`-print=printer`	Prints report to `printer`
`-varNAME[!#][::type]=value[::description]`	Creates a variable "NAME" in REPORTER with value "value" and description "description". `::description` and `::type` can be omitted. If the type is omitted it defaults to "General". By default variables defined on the command line will not be marked as temporary. If the variable name is suffixed by '#' then the variable will be temporary. '!' can also be used to mark the variable as not being temporary (although this is not needed as it is the default).
`-pptx=file.pptx (or -ppt=file.pptx)`	Create PowerPoint file "file.pptx".
`-log=logfile`	Save the logfile REPORTER produces in the file "logfile" as plain text after processing all the command line options.
`-loghtml=logfile`	Save the logfile REPORTER produces in the file "logfile" as HTML after processing all the command line options.
`-generate`	Generate a report (previously read with `-file` argument). Note: this is not required if you use any of the `-ps`, `-pdf`, `-html`, or `-pptx` options (they do this automatically)
`-report=file.orr`	Saves generated report (previously read with `-file` argument) to `file.orr`
`-script=script.js`	Runs javascript script `script.js`.
`-argfile=argfile`	Reads command line arguments from file `argfile`, one argument per line. This could be useful if you want to read lots of variables on the command line and you reach the command line length limit.
`-exit`	Automatically exit after processing all other command line options
`-iconise`	Start REPORTER iconised. This is useful for running reporter from scripts when you want to continue working on something else and you do not want the REPORTER window to interfere.
`-new`	Create a new template.

`-batch`	Batch mode. This stops REPORTER prompting the user. For example, normally if an error occurs when generating REPORTER brings up a warning box allowing the user to look at the error. Giving the `-batch` argument stops this. Note that this does NOT make REPORTER run without the user interface (see `-iconise`)
`-oasys_batch`	On Windows run D3PLOT and T/HIS without any windows being shown.
`-combine`	Combine multiple report output into pdf, html or pptx.

So for example:

```
reporter -file=/job/templates/example.orr                    /
         -pdf=/local/output.pdf                              /
         -print=printer                                      /
         -varKEYWORD=/job/keyword/example.key::example deck /
         -html=/local/example.html                           /
         -exit
```

Will:
1. Load the file "/job/templates/example.orr" into REPORTER
2. Install a variable called KEYWORD with value "/job/keyword/example.key" and description "example deck"
3. Create a pdf file "/local/output.pdf"
4. Print the file on printer "printer"
5. Create a HTML file "/local/example.html"
6. automatically exit

A.2 oa_pref options

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 REPORTER will revert to its default settings.

"oa_pref" naming convention and locations

The file is called "oa_pref"

It is looked for in the following places in the order given:
- The site-wide admin directory (**$OA_ADMIN**)
- The site-wide "Oasys Ltd LS-DYNA Environment" directory (**$OA_INSTALL**)
- The user's home directory: **$HOME** (Unix/Linux) or **$USERPROFILE** (Windows)

The first encountered file will be used, so this file can be customised for a particular job or user at will.
Files do not have to exist in any of these locations, and if none exists the programme defaults will be used.

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 **C:\Documents and Settings\<*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 $HOME / $USERPROFILE

"oa_pref" file syntax

The syntax used for Primer is:

`reporter*<keyword>: <argument>`

for example:

`reporter*default_item_width: 10.0`

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.

"oa_pref" options valid for REPORTER

Preference	Type	Description	Valid arguments	Default
date_format	<string>	Format for printing default date variable	Day Month Day Year, dd/mm/yyyy, mm/dd/yyyy, yyyy/mm/dd	Day Month Day Year
default_item_height	<real>	Default width given to item (mm) if it is not dragged when creating	0.0 - 999.9	10.0
default_item_width	<real>	Default height given to item (mm) if it is not dragged when creating	0.0 - 999.9	10.0
file_names	<string>	Controls output file names. LSTC = d3plot, d3thdt, d3hsp etc, OASYS/ARUP = job.ptf, job.thf job.otf etc	OASYS, ARUP, LSTC	OASYS

The following options control font settings in REPORTER

Preference	Type	Description	Valid arguments	Default
default_fonts				
cursive_font	<string>	Default cursive font in REPORTER		’Monotype Corsiva’ [Windows]; ’URW Chancery L’ [Linux]
fantasy_font	<string>	Default fantasy font in REPORTER		’Impact’ [Windows]; ’Impact’ [Linux]
monospace_font	<string>	Default monospace font in REPORTER		Courier New [Windows]; Courier 10 Pitch [Linux]
sans_serif_font	<string>	Default sans-serif font in REPORTER		’Arial’ [Windows]; ’Liberation Sans’ [Linux]
serif_font	<string>	Default serif font in REPORTER		’Times New Roman’ [Windows]; ’Nimbus Roman No9 L’ [Linux]
symbol_font	<string>	Default symbol font in REPORTER		’Symbol’ [Windows]; ’Standard Symbols L’ [Linux]
blacklisted_fonts	<string>	Comma-separated list of font filenames that will not be processed by REPORTER because they typically take too long to load		<none>
cache_directory	<string>	Font cache file directory		$OA_HOME
font_mapping_table	<string>	Full path and filename of font mapping table CSV file		$OA_INSTALL/reporter_library/fonts/font_mapping.csv

graphical_user_interface				
group_tools	<logical>	Group the Tool buttons into a single button for each category	TRUE, FALSE	FALSE
gui_theme	<string>	Graphical User Interface (GUI) theme	LIGHT, DARK, CLASSIC, LEGACY	LIGHT
show_labels	<logical>	Show labels as well as icons on the Tool buttons	TRUE, FALSE	FALSE
grid				
default_grid	<real>	Default grid spacing (mm)	0.0 - 999.9	5.0
default_snap	<real>	Default snap size (mm)	0.0 - 999.9	1.0
grid_colour	<string>	Default grid colour (HEX RRGGBB value)		A9A9A9
grid_style	<string>	Default grid line style	DOT, CROSS, LINE	DOT
show_grid	<logical>	Show grid lines on page	TRUE, FALSE	FALSE
snap_to_grid	<logical>	Snap items to grid	TRUE, FALSE	TRUE

The following options control the library location in REPORTER

Preference	Type	Description	Valid arguments	Default
library_ directory	<string>	User defined library directory for REPORTER		$OA_INSTALL/reporter_ library

The following options control Logfile preferences

Preference	Type	Description	Valid arguments	Default
check_for_ errors	<logical>	Check for errors during report generation	TRUE, FALSE	TRUE
debug_oasys	<logical>	Include debug information from PRIMER, D3PLOT and T/HIS	TRUE, FALSE	FALSE
stay_in_oasys	<logical>	Don’t automatically exit from PRIMER, D3PLOT and T/HIS	TRUE, FALSE	FALSE
maximise	<logical>	Maximise window when REPORTER started	TRUE, FALSE	TRUE
oasys_batch	<logical>	Run D3PLOT, PRIMER and T/HIS in batch mode from REPORTER	TRUE, FALSE	FALSE

The following options control how objects are edited

Preference	Type	Description	Valid arguments	Default
coordinate_ method	<string>	Method used for editing object coordinates	Opposite corners, Width and height	Width and height
default_nudge	<real>	Default nudge distance (mm)	0.0 - 999.9	5.0
object_ reference_ corner	<string>	Corner used as reference when editing objects	TopLeft, TopRight, BottomLeft, BottomRight	BottomLeft
revert_to_ select_tool	<logical>	After creating a new item, the cursor reverts to the Select tool	TRUE, FALSE	TRUE
placement	<string>	Location for initial window on multi-screen display	LEFT, RIGHT, BOTTOM, TOP, LEFT_ BOTTOM, LEFT_TOP, RIGHT_BOTTOM, RIGHT_TOP	<none>

The following options control pdf output

Preference	Type	Description	Valid arguments	Default
image_downsampling				
pdf_image_downsample	<logical>	Downsample images in pdf files	TRUE, FALSE	FALSE
pdf_image_downsample_ resolution	<integer>	Resolution to downsample images to	10 - 3000	150

pdf_image_downsample_threshold	<real>	Factor above pdf_image_downsample_resolution before downsampling is done	1.0 - 10.0	1.5

The following options control which other Oasys Ltd LS-DYNA Environment programmes are used by REPORTER

Preference	Type	Description	Valid arguments	Default
d3plot	<string>	D3PLOT executable to use		<none>
d3plot_args	<string>	Extra command line arguments to pass to D3PLOT		<none>
d3plot_properties_parts_only	<logical>	Only read parts (ignore elements) when reading properties file	TRUE, FALSE	FALSE
d3plot_properties_pre_blank	<logical>	Pre blank all parts before reading properties file	TRUE, FALSE	FALSE
primer	<string>	PRIMER executable to use		<none>
primer_args	<string>	Extra command line arguments to pass to PRIMER		<none>
this	<string>	T/HIS executable to use		<none>
this_args	<string>	Extra command line arguments to pass to T/HIS		<none>
save_colours_on_exit	<logical>	Automatically save the user colours XML file when the program exits.	TRUE, FALSE	TRUE
start_in	<string>	Directory to start REPORTER in		<none>
time_format	<string>	Format for printing default time variable	hh:mm:ss, hh:mm:ss A, hh:mm, hh:mm A	hh:mm:ss
use_default_vars	<logical>	Use default vars in filenames when capturing if possible	TRUE, FALSE	TRUE
use_file_vars	<logical>	Use file/directory vars in filenames when capturing if possible	TRUE, FALSE	TRUE

The following options control unicode

Preference	Type	Description	Valid arguments	Default
cjk_default	<string>	Default language for ambiguous CJK Kanji	Chinese, Japanese, Korean	Japanese
pdf				
chinese_characters	<string>	Style for chinese characters in pdf files	Simplified, Traditional	Traditional
japanese_font	<string>	Font for japanese characters in pdf files	Kozuka Mincho Pro, Kozuka Gothic Pro	Kozuka Mincho Pro

Editing/changing preferences

There is currently no interactive preferences editor for REPORTER. To change preferences for REPORTER please use the interactive preferences editor in Oasys Ltd SHELL, D3PLOT, T/HIS or PRIMER or edit the preferences file by hand.

B. Library objects

B.1. Standard library programs

REPORTER has a number of built-in scripts to retrieve data from the keyword or otf files. New scripts can be added as required. See Adding scripts to the library. By default REPORTER looks for library programs in the subdirectory `reporter_library/scripts` in the directory where REPORTER is installed. Other directories can be added if required. See User defined library directories for more details.

D3PLOT data file programs

Create a D3Plot data file from a cvs file	Create a data file which is suitable for use by D3PLOT. The data will be extracted from a csv (comma separated value) file. See section 6.1
Create a D3Plot data file from generated data files	Create a data file which is suitable for use by D3PLOT. The data will be extracted from `reporter_variables` files. See section 6.1

Error programs

Read PRIMER error file	Read an error file produced by doing a model check in PRIMER and extract the errors

Keyword file programs

The following programs retrieve information from a keyword file.

Analysis title	Prints the title of the analysis from the `*TITLE` card.
Comments between `*KEYWORD` and `*TITLE`	Prints any comment lines in the keyword file between the `*KEYWORD` and `*TITLE` keywords. The $ will be removed from each line. An optional second argument can be used to impose a maximum limit on the number of lines printed.
Create variables for parameters used in analysis	
Extract title and LCSS curve from *MAT_PIECEWISE_LINEAR_ PLASTICITY_TITLE cards	
Include files used in analysis	Prints a list of all the include files used in the analysis. By default the full pathname of include files is written. An optional second argument can be used to give the names relative to the master file
Initial velocity card used in analysis	Prints the first line of any `*INITIAL_VELOCITY` cards in the keyword file. The script will also recursively look in include files for `*INITIAL_VELOCITY` cards.
Timestep from `*CONTROL_TIMESTEP` card	Reads the `DT2MS` value from the `*CONTROL_TIMESTEP` card

NCAP

Create a US-NCAP graph	Create a graph for US-NCAP star rating using HIC and chest acceleration (3ms clip)

OTF file programs

The following programs retrieve information from an OTF file.

Mass info

Added mass at end of analysis	Prints the mass added to the analysis by mass-scaling at the end of the analysis. This will also look at otf files generated from restarts (otf01, otf02 etc)
Added mass at start of analysis	Prints the mass added to the analysis by mass-scaling at the start of the analysis.
Percentage final added mass	Prints the percentage mass added to the analysis by mass-scaling at the end of the analysis. This will also look at otf files generated from restarts (otf01, otf02 etc)
Percentage initial added mass	Prints the percentage mass added to the analysis by mass-scaling at the start of the analysis.
Total mass in analysis	Prints the mass of the model at the start of the analysis

Timestep info

Mass-scaled timestep (DT2MS) echo in OTF file	Prints the `DT2MS` value from the `*CONTROL_TIMESTEP` card echoed to the OTF file.
Smallest initial timestep	Prints the element with the smallest timestep from the 100 smallest timesteps. The line has the form: `<element_type> <element_number> timestep = <timestep>`

Timing info

Elapsed time for analysis	Prints the total elapsed time for the analysis.
Start time for analysis	Prints the date and time that the analysis finished.
Problem cycle for analysis	Prints the cycle in the analysis that the problem terminated.
Problem time for analysis	Prints the time in the analysis that the problem terminated.
Start time for analysis	Prints the date and time that the analysis started (same as Analysis date).
Terminition time(ENDTIM) echo in OTF file	Prints the termination time from the `*CONTROL_TERMINATION` card echoed to the OTF file. This will also look at otf files generated from restarts (otf01, otf02 etc).

Other OTF programs

Analysis date	Prints the date and time that the analysis started
Analysis precision	Prints the precision (single/double) LS-DYNA used for the analysis
Analysis title	Prints the title of the analysis echoed to the OTF file.

CPU time for analysis	Prints the total CPU time used for the analysis. This will also look at otf files generated from restarts (otf01, otf02 etc)
Check on the quality of the run	Looks to see if the analysis terminated normally, if the initial and final added masses, the total energy fluctuation and hourglass energy are below (user definable) limits. Either prints OK or NOT OK.
Hostname analysis run on	Prints the hostname of the machine the analysis was run on.
LS-Dyna version and revision	Prints the version and revision of LS-DYNA used to run the analysis
Normal or Error termination message	Prints `Normal` or `Error` termination message from LS-DYNA.
Number of CPUs used for analysis	Prints the number of CPUs used for the analysis
OS analysis run on	Prints the operating system level of the machine the analysis was run on.
Platform analysis run on	Prints the platform of the machine the analysis was run on.

Pedestrian

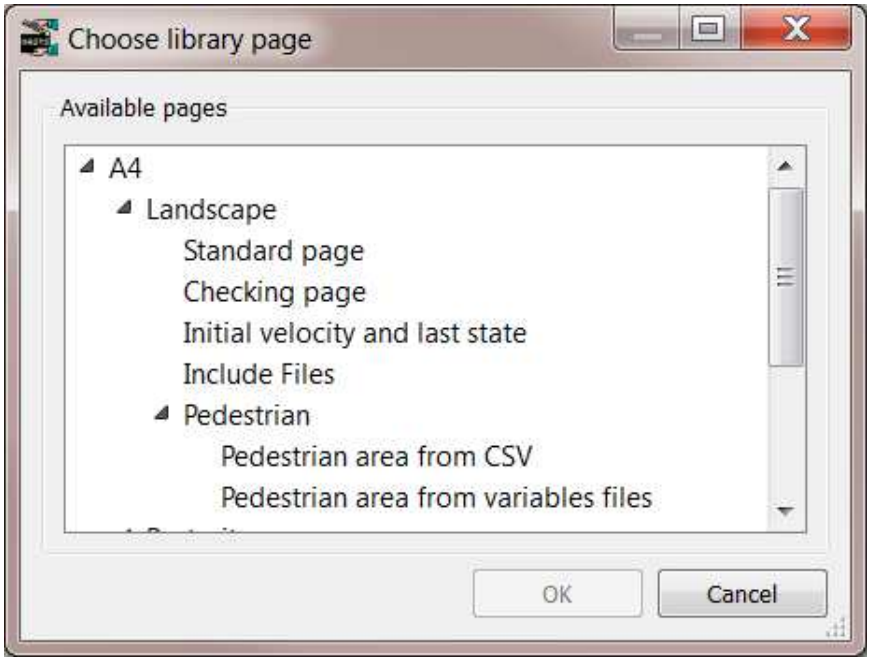
Create a contour image of HIC for pedestrian HIC results in a CSV file	Creates an image showing contours of HIC from values in a csv file. See http://www.oasys-software.com/dyna/en/downloads/extras.shtml#pedestrianarea for more details.
Create a contour image of HIC for pedestrian HIC results in reporter variables files	Creates an image showing contours of HIC from values in reporter variables files. See http://www.oasys-software.com/dyna/en/downloads/extras.shtml#pedestrianarea for more details.

Variables

Read a REPORTER variable file	Read a variables file written by another REPORTER template and install the variables from it into the current template
Read variables from a CSV file	Read variables from a CSV file (one variable per row).
Read variables from a CSV file (data in rows)	Read variables from a CSV file (one variable per column)

B.2. Standard library pages

REPORTER comes with some standard pages which can be installed from a library. They are shown in the image below. The pages are available in landscape and portrait versions. The information on the page is the same in either case.

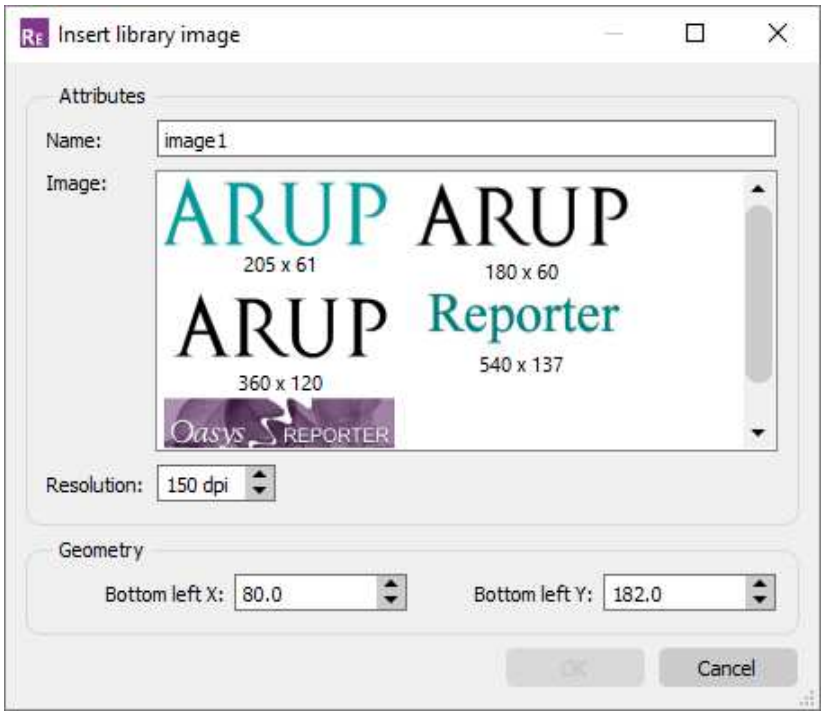


Type	Description
Checking page	Information for the analysis extracted from the OTF file and an energy balance plot from T/HIS.
Include Files	A list of any include files that were used in the analysis
Initial velocity and last state	Images captured from D3PLOT of the initial velocity in the analysis and of the last state.
Standard page	A blank page with a standard footer
Pedestrian area from CSV	Information and a contour plot of HIC values for pedestrian HIC analyses. See http://www.oasys-software.com/dyna/en/downloads/extras.shtml#pedestrianarea for more details.
Pedestrian area from variables files	Information and a contour plot of HIC values for pedestrian HIC analyses. See http://www.oasys-software.com/dyna/en/downloads/extras.shtml#pedestrianarea for more details.

New pages can be added as required. See Adding pages to the library.

B.3. Standard library images

REPORTER comes with some standard images which can be installed from a library. They are shown in the image below.



New images can be added as required. See Adding images to the library.

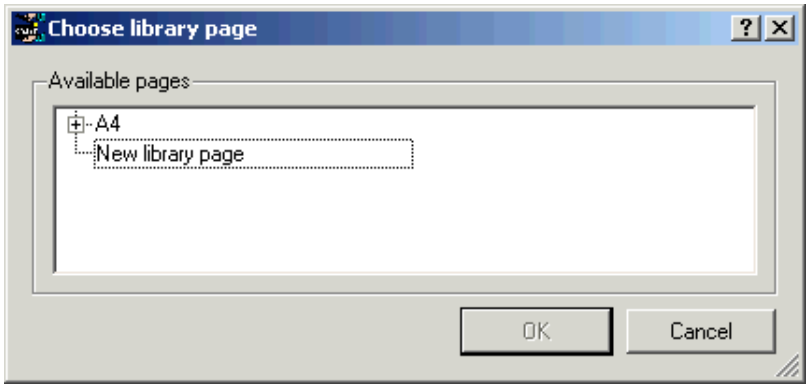
B.4 Adding pages to the library

To add a new page layout to the library you need to:

- Create a the page in REPORTER.
- Export the page, saving it with extension **`.orp`** using Page->Export... (see exporting pages for more details).
- Copy the exported page into the `/library/pages/` directory of your Oasys Ltd LS-DYNA Environment installation.

It will then be shown the next time you start REPORTER. Note that the title of the page is what will be shown in the library page tree so make sure that the page has a sensible title. This can be changed using Page->properties... (see Changing the page properties for more details).

So, for example, if you have a page called 'New library page' and you put it in the `/library/pages/` directory you will get:



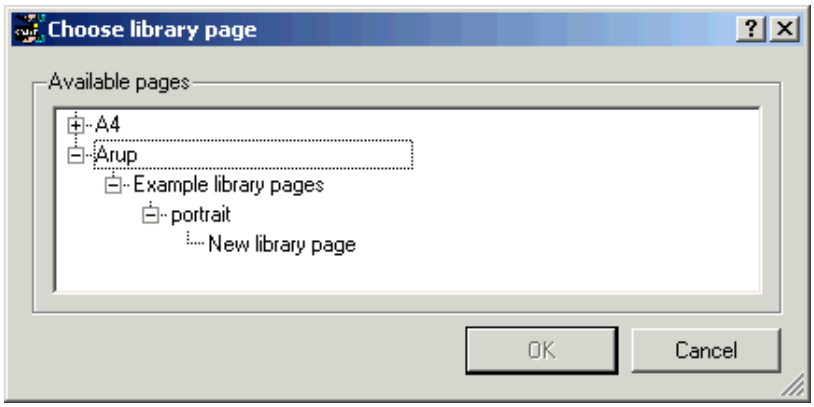
If you want the page to be shown in a different branch of the tree then edit the file using a text editor and change the file as follows. The first line should look like:

```
<REPORTER FILETYPE='page' VERSION='92'>
```

If I wanted a branch in the tree to be 'Arup/Example library pages/portrait' I would change this to

```
<REPORTER FILETYPE='page' VERSION='92' FOLDER='Arup/Example library
pages/portrait'>
```

The page would then be shown in the tree as:



B.5 Adding scripts to the library

REPORTER has a javascript interpreter built into it. The scripts which are available in the library are run inside REPORTER

To add a new script to the library save it into the `/library/scripts/` directory of your Oasys Ltd LS-DYNA Environment installation. Then you need to add the following special comment at the top of the file.

```
/* A description of your script
PROGRAM::<script_name>
DESC::<description>
FOLDER::<folder>                                (optional)
RETURN::<output_type>
[+-]ARG::<description>[::<default text>]    (repeat for as many arguments as
required)
EXPAND_ARGS::false                              (optional)
END_INFO
*/
```

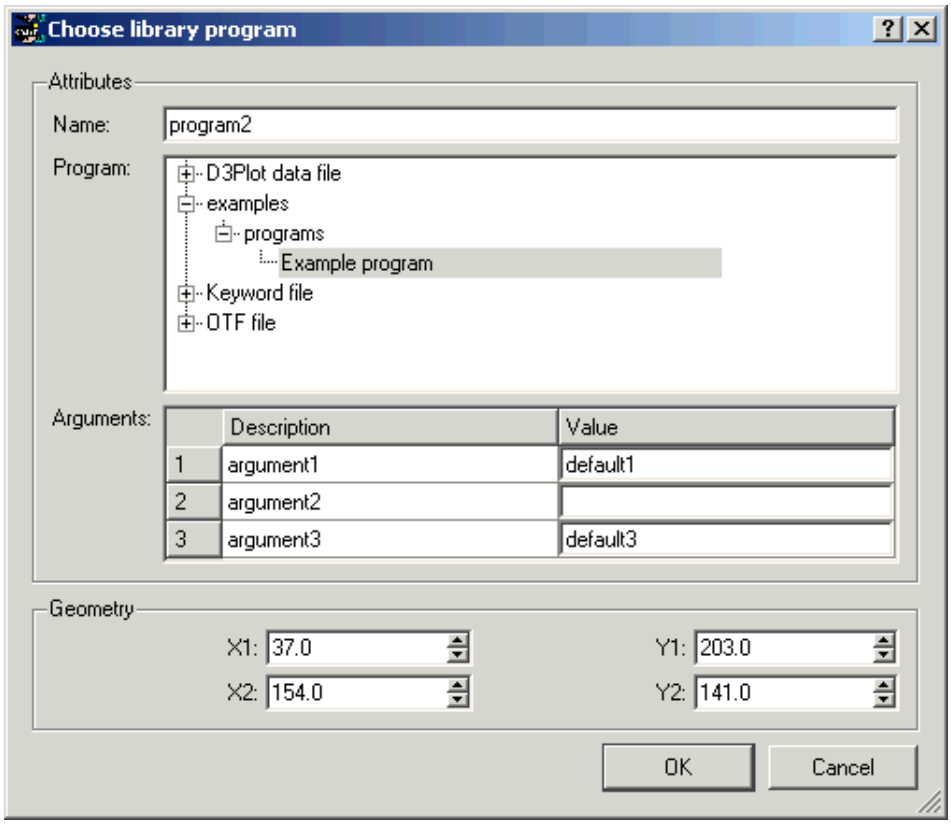
Note the **`/*`** at the beginning and **`*/`** at the end.
The lines have the following meaning:

PROGRAM	`<script_name>` is the name of the javascript program. It should have the extension js
DESC	`<description>` is a description of the program/script that will appear in the **Insert program from library** window
FOLDER	The programs in the **Insert program from library** window are shown in a 'tree' view. `<folder>` indicates which folder or 'branch' of the tree the program is shown in. This is the same as for library pages above.
RETURN	`<output_type>` is the type of output the program returns. Currently the only value supported is `text`.
ARG	`<description>` is the argument description that will appear in the **Insert program from library** window. Optionally the line can be prefixed with a `+` or `-` sign. If a `-` sign is used the argument is optional. If a `+` sign is used (default) the argument is mandatory. Optionally an argument can be followed by `<default_text>` which will be used as a default for the argument in the window.
EXPAND_ ARGS	Normally any variables in program arguments get expanded to their actual values and so you would omit this line. There may be instances where you do not want to expand them. In this case use the line `EXPAND_ARGS::false` (e.g. see data_file_from_variables.js).
END_ INFO	This line indicates the end of the informat and must be included

For example, the following lines

```
/*
PROGRAM::example.js
DESC::Example program
FOLDER::examples/programs
RETURN::text
ARG::argument1::default1
ARG::argument2
-ARG::argument3::default3
END_INFO
*/
```

would give the output:



Rules for writing scripts

As REPORTER runs the scripts internally, they have to be written in a specific way. The following guidelines should be used for writing custom scripts for REPORTER. If these guidelines are too restrictive or you do not want to work this way, remember that you can write external programs for REPORTER in any language you choose. See Appendix E for more details.

- Scripts must be written in javascript! REPORTER contains a javascript interpreter. Other languages are NOT supported.
- To output text back to REPORTER use the output function.
- See the scripting chapter for javascript scripting.
- See the Javascript class reference appendix for extra javascript classes that REPORTER defines.

The scripts in the `/library/scripts` directory give an indication of what is possible with internal scripts. For more details refer to the individual scripts.

The functionality will be extended over time. If you have requests for new features contact Oasys Ltd.

B.6 Adding images to the library

To add an image to the library copy it into the /library/images directory of your Oasys Ltd LS-DYNA Environment installation. It will then be shown next time you start REPORTER. The image should be a bmp, jpg, png or gif image.

Note that if you add images to the library and then use the image in a template, the image will not work for installations that do not have this library image. This is fine if you are using this internally in your company, but be careful when giving a template to another person/company. The way round tis problem is to save your template as a report once it has been generated. When you save as a report any images are embedded to this is then portable. See Outputting a generated report for more details.

B.7 User defined library directories

By default REPORTER looks for library programs in a subdirectory `reporter_library/scripts` in the directory where REPORTER is installed. Extra library programs can be added to this directory using the above logic. However, this may not be possible due to file permissions. For this reason it is possible to specify another directory for REPORTER to use an addition to the default directory. This can be done using the `library_directory` oa_pref option. If this option is set then REPORTER will also treat this directory as a user defined `reporter_library` directory.

Currently only scripts are supported as user library items (i.e. images and pages are currently not supported). User scripts should be put in a subdirectory `scripts` of your `library_directory`.
For example, if `library_directory` is set to `/home/user/reporter_library` then you should put your scripts in `/home/user/reporter_library/scripts`.

In future versions of REPORTER it may be possible to have user defined pages and images.

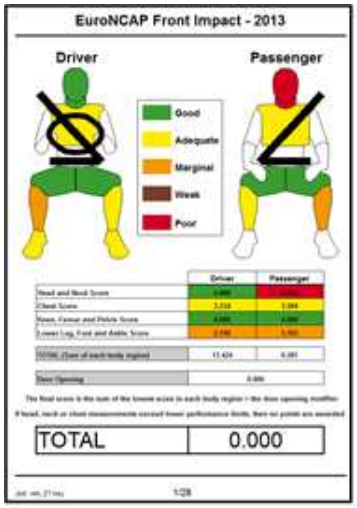
B.8 Standard library templates

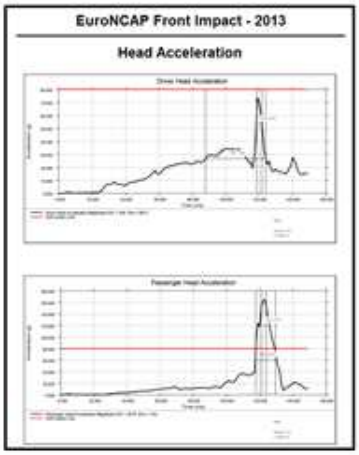
A number of standard templates have been created for automotive crash test protocols, included as part of the installation. They are located in the subdirectory `reporter_library/templates` in the directory where REPORTER is installed.

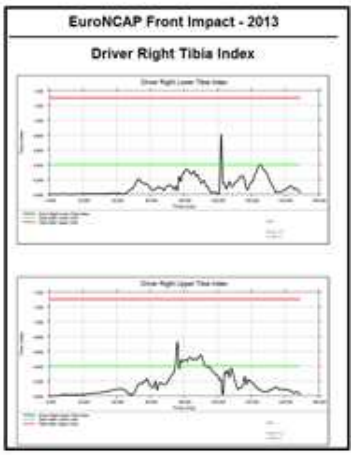
The templates can be selected from **File → Open Library Template...**. After asking a few questions to get information needed to generate the report, the standard templates calculate results according to the protocol, e.g. EuroNCAP Front Impact. They can also be run in a batch mode. In addition to the standard templates for automotive crash test protocols, there is a general LS-DYNA template that can be used for any LS-DYNA model, and a general LS-DYNA vehicle template that can be used for any vehicle analysis.

In general, the templates contain a front summary page showing the overall score for the test, with further pages containing tables showing individual measurements and graphs.

The reports can be written out as PDF, HTML or PPTX files as normal.







There are two generic types of templates:

- Single analysis templates (Front Impacts, Side Impacts...)
- Multiple analysis templates (Pedestrian Head Impacts, Pedestrian Leg Impacts)

The way they work is slightly different, but generally they follow the same process. The following sections will describe how to use the templates. There is also additional documentation for the new Euro NCAP MPDB templates:

- REPORTER Instructions
- Barrier face deformation calculation
- Occupant Load Criterion calculation

The latest templates

Template	Changes from previous version
C-NCAP Front ODB Impact 2018	Now includes rear passenger.
C-NCAP Head Impact 2021	New template.
C-NCAP Leg Impact 2021	New template.
C-NCAP MPDB 2022 Compatibility Assessment	New template. Designed to work with the Arup Cellbond MPDB Shell Model.
C-NCAP MPDB 2023 Compatibility Assessment	New template. Designed to work with the Arup Cellbond MPDB Shell Model.
C-NCAP MPDB 2022 Occupant Assessment	New template. Processes the THOR 50M driver and HIII 5F passenger.
Euro NCAP Far Side Impact 2022	New template.
Euro NCAP Front FFB Impact 2017	Now includes front passenger.
Euro NCAP Front ODB Impact 2017	Final score calculation depends on capping limits being exceeded as opposed to the lower performance limits from 2015.
Euro NCAP Head Impact 2023	Updated to the latest protocols, which include assessment of Cyclist impact points.
Euro NCAP Leg Impact 2023	Updated to the lates protocols, which include assessment of the aPLI impactor.
Euro NCAP MPDB 2020 Compatibility Assessment	New template. Designed to work with the Arup Cellbond MPDB Shell Model.
Euro NCAP MPDB 2023 Compatibility Assessment	New template. Designed to work with the Arup Cellbond MPDB Shell Model.
Euro NCAP MPDB 2020 Occupant Assessment	New template. Processes the THOR 50M driver and HIII 50M passenger.
Euro NCAP Side MDB Impact 2022	Updated to the latest protocol (AOP v9.1.3).
Euro NCAP Side Pole Impact 2022	Updated to the latest protocol (AOP v9.1.3).
General LS-DYNA Model	
General LS-DYNA Vehicle Model	
GTR Head Impact 2020	Major overhaul: a new landscape layout, HIC area calculation is now done with the PRIMER HIC Area Calculator, and band/area sensitivity results are presented.
GTR Leg Impact 2019	New template.
ICFD Assessment	New template. To run in batch, use JSON_FILE rather than CSV_FILE.
IIHS Front ODB Impact 2021	Updated to 2021 protocols.
IIHS Front ODB Impact 2021 – Structure Only	
IIHS Front SOB Impact 2021	Now includes passenger occupant. Updated to 2021 protocols.

IIHS Front SOB Impact 2021 – Structure Only	
IIHS Side MDB Impact 2021	Updated to new IIHS Side Impact 2.0 protocols.
IIHS Side MDB Impact 2021 – Structure Only	
JNCAP Leg Impact 2018	
KNCAP Leg Impact 2019	
USNCAP Front FFB Impact 2015	
USNCAP Side MDB Impact 2015	
USNCAP Side Pole Impact 2015	

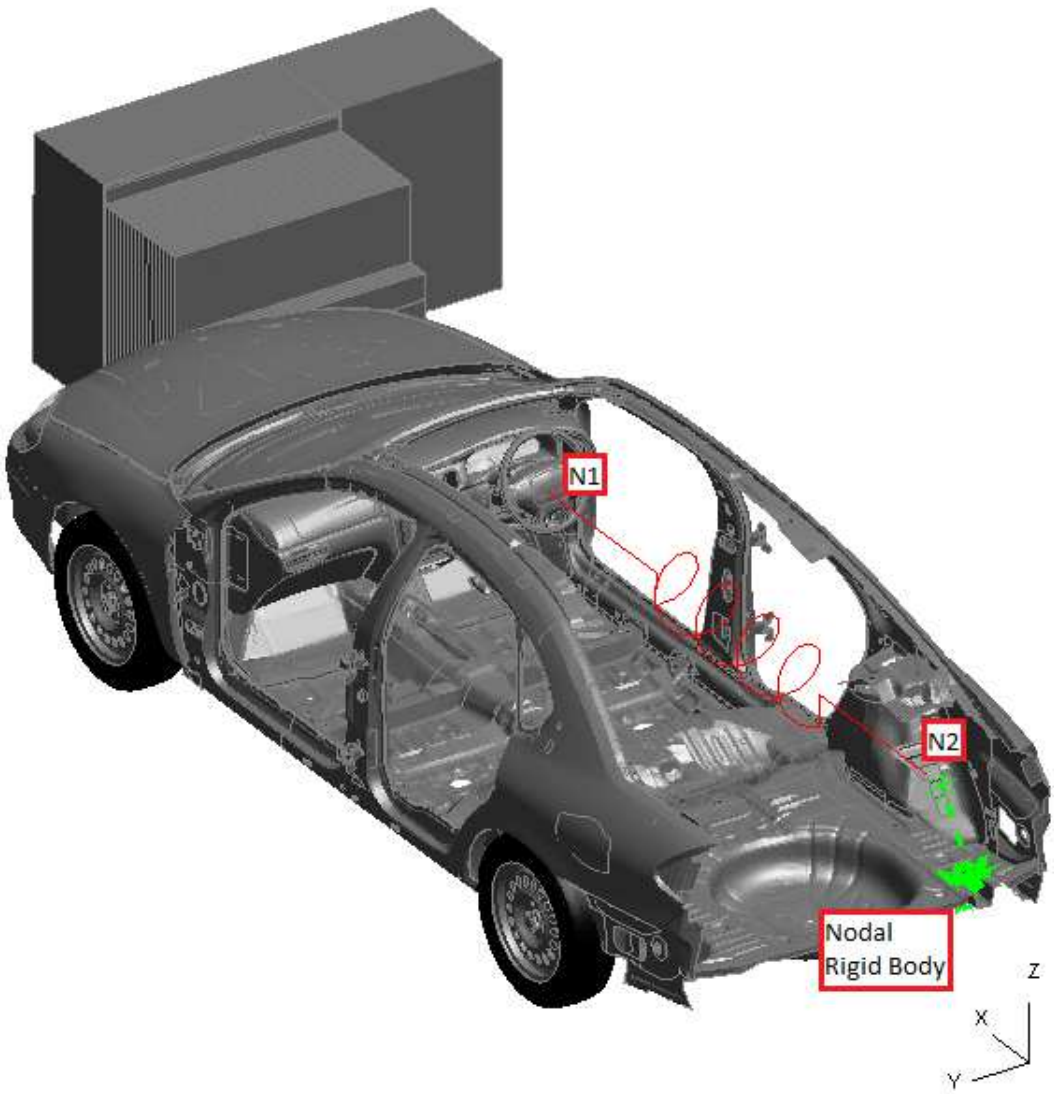
Assumptions

The following assumptions are made about the models to be post-processed. If they are not true of your model then the templates may not work correctly:

- Humanetics dummies should be used for any occupants.
- Intrusion measurements are made with springs.

  Use *ELEMENT_DISCRETE with a low stiffness value on the *MAT_ SPRING_ELASTIC card.

  Node 1 should be attached to the structure that is being measured and Node 2 should be attached to a nodal rigid body where there will be no deformation (normally at the rear of the vehicle). For example, to measure the steering column intrusion in an ODB impact:



One spring will be needed for each intrusion measurement. For the case above there are three springs overlaying each other: one for intrusions in X, one in Y and one in Z. A vector should be used to define the orientation and should be aligned with the X, Y or Z global axis depending on the measurement being made.

B.8.1 Single analysis templates

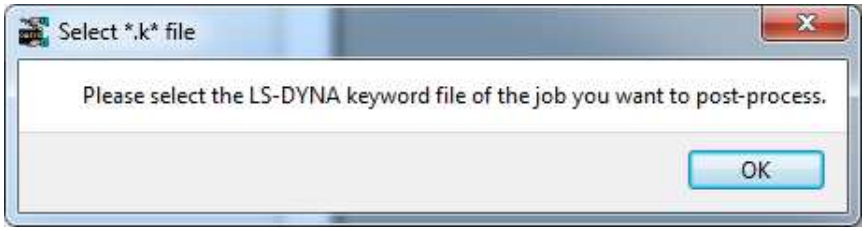
All the single analysis templates follow the same process, so we'll use the EuroNCAP Front ODB Impact template as an example for how to use them. The section will describe how to run them interactively using the menus in REPORTER, but it is also possible to run them in batch mode.

Select the template

Use the **File → Open Library Template** menu and select a template from the **Automotive** tab (see 3.2 Reading an existing template or report for more details).

Generate the template

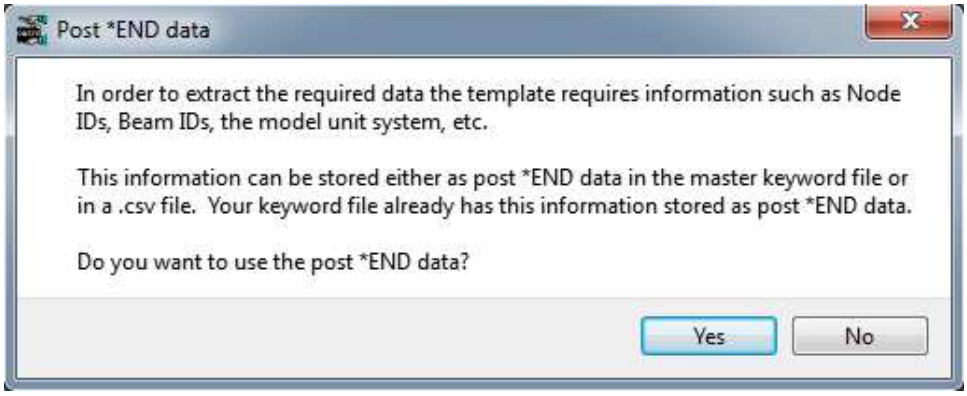
After selecting the template REPORTER should prompt you to select the keyword file of the job you want to post-process:



After pressing 'OK' a file selector is mapped for you to select the keyword file.

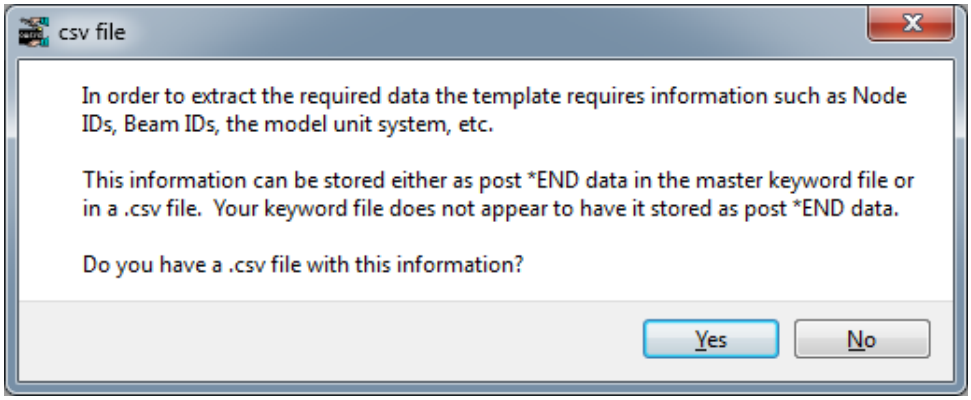
In order to correctly extract the results needed for the protocol the template needs model information such as Node IDs, Beam IDs, the unit system, etc. This needs to be supplied to the template either from a .CSV file or from comments written in the keyword file after the *END keyword. The template will help you to create this information (***you should only need to do this ONCE for a particular vehicle programme so long as IDs remain the same***).

The template will scan the keyword file to see if it contains the required information after the *END keyword. If it does you will be asked if you want to use it:



If you press 'Yes' then the next few questions will not be asked.

If you press 'No' or the keyword file doesn't contain the required information after the *END keyword you will be asked if you have a .CSV file with the information instead:

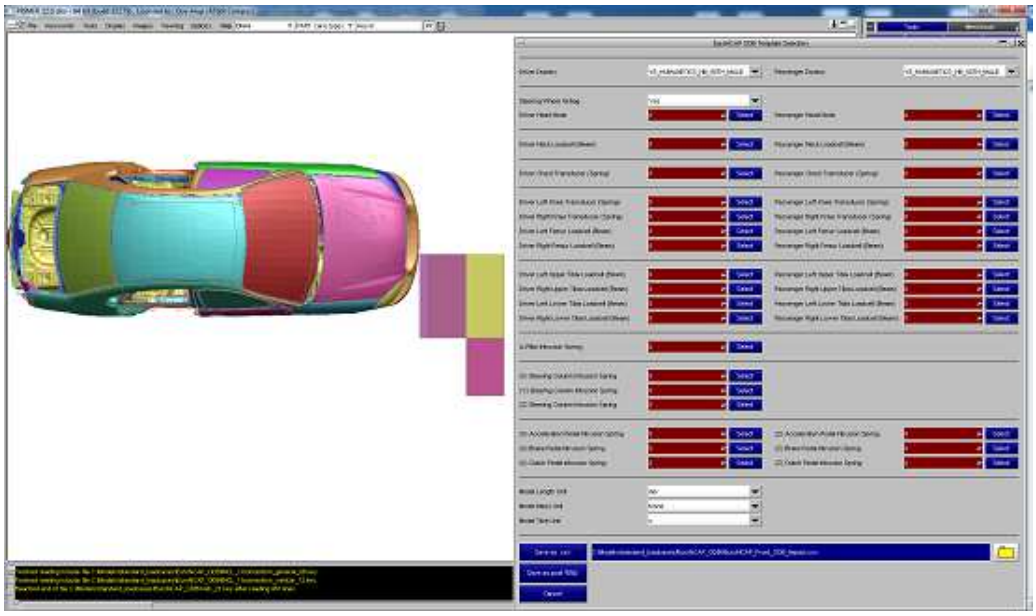


If you have a .CSV file, press 'Yes' and select it in the file selector.

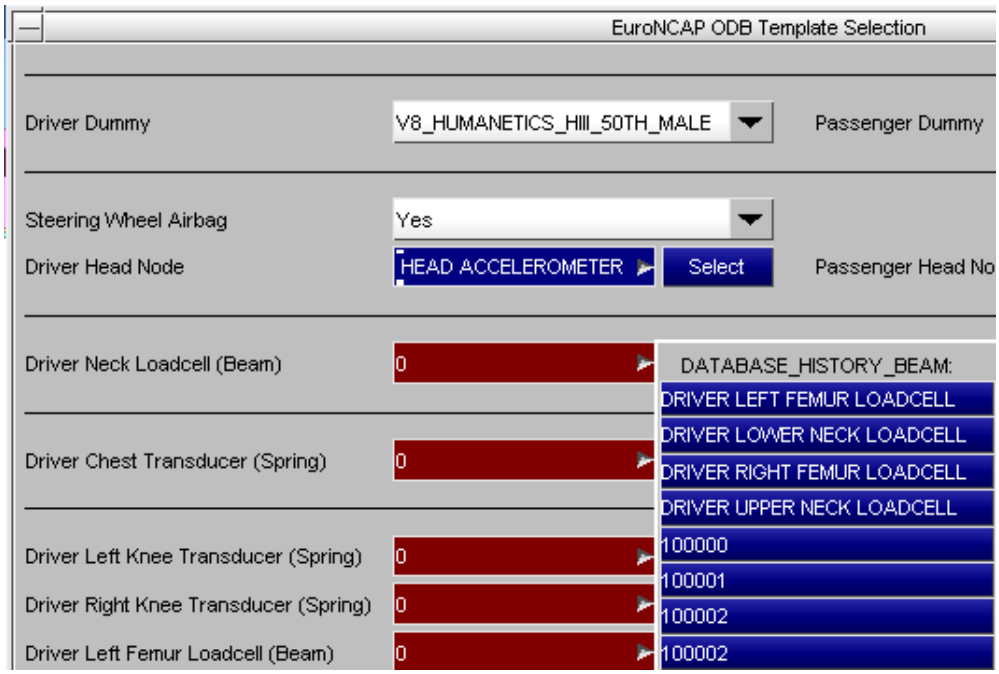
If you press 'No' REPORTER will inform you that it will start PRIMER so you can select the required information

interactively:

After pressing 'OK' PRIMER will start, the model will be read in and a window will be open for you to interactively select the required information:

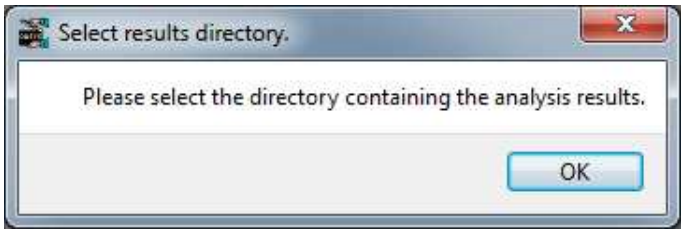


You can select the information either by picking the entities in the graphics window, typing it in to the textbox or selecting it from a list of *DATABASE_HISTORY_XXXX entities in the popup menu. If they are defined with the _ID option then the names will be used instead of the numbers, e.g.



After you have selected the information you can need to save it either as a .CSV file or post *END data. Next time you use the template, perhaps on a slightly modified model, you should be able to reuse this data without having to go back into PRIMER each time (so long as the IDs of entities you want to extract data from stay the same).

PRIMER will close and REPORTER will ask for the directory containing the analysis results (which may be different to the location of your kewyord file).

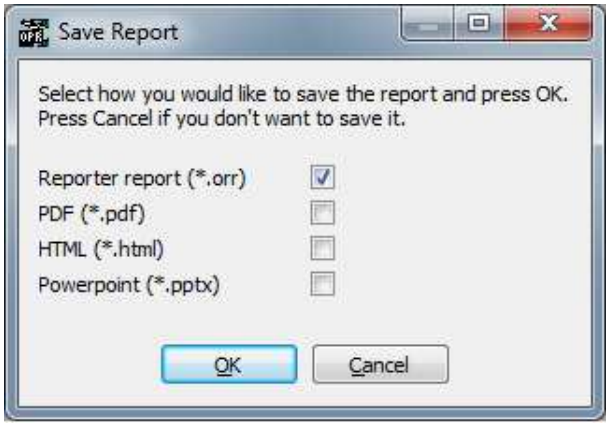


Press OK and select the directory.

Finally you will be asked for a directory where REPORTER should write any images.

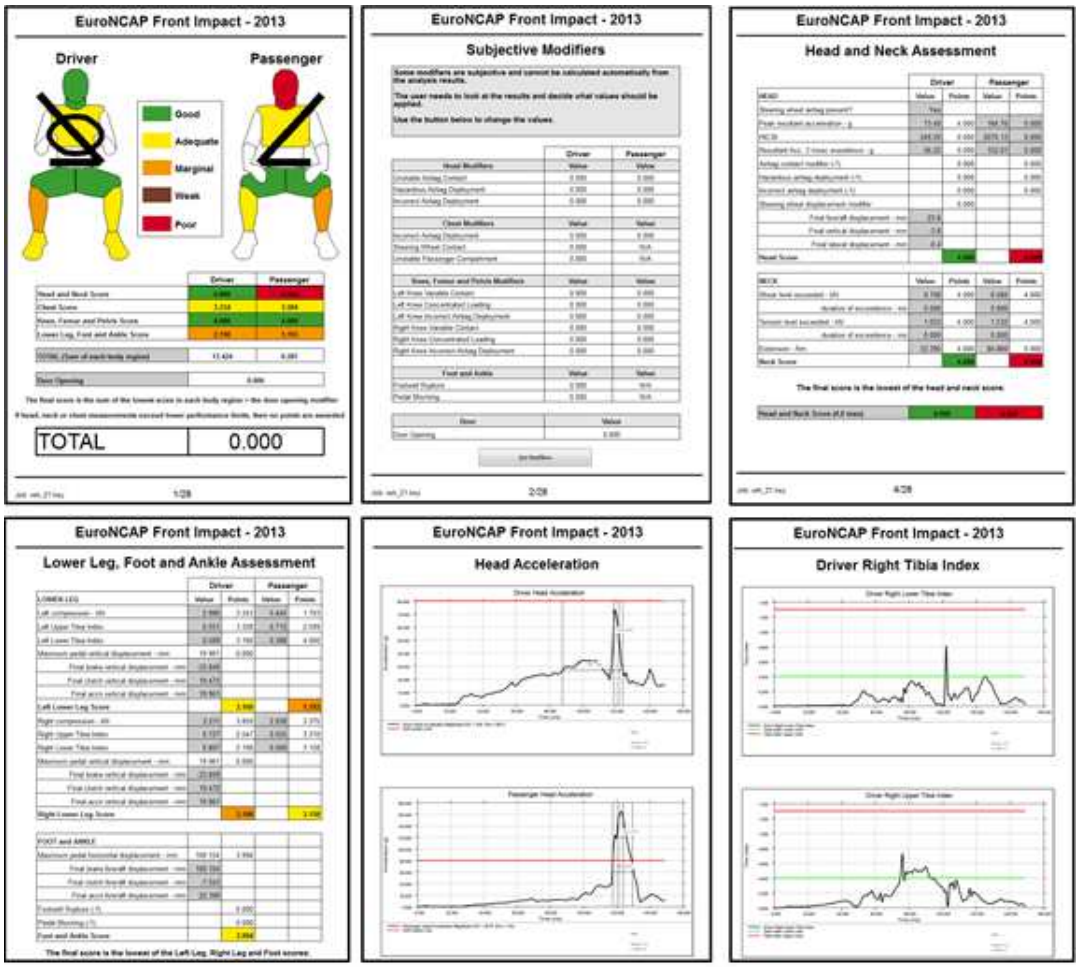
Press OK and select the directory.

REPORTER should now have all the information it needs. T/HIS will load and carry out the post-processing according to the selected protocol, generating the required graphs. Once this is finished, REPORTER will ask a final question asking you how/if you want to save the report:



If you don’t want to save the report, just press ’Cancel’. If you do, select the format(s) you want to save it in, press ’OK’ and select where you want to save it in the file selector that will pop up.

The final report should look something like this, with a front summary page showing the protocol scores in tables and as an image; a page to change subjective modifiers that can’t be calculated automatically from the analysis; tables and graphs showing the analysis results and protocol scores in more detail:



Subjective modifiers

In general most data can be extracted automatically from the analysis results and then processed according to the protocol. However, some data is subjective and requires the user to look at the analysis results and manually set the values.

For example, the EuroNCAP front ODB impact test has some modifiers which are applied as penalty points to the calculated scores, e.g. if an airbag doesn't deploy correctly a 1 point penalty is applied.

These subjective values can be set on the second page of the report after it has been generated. This lists all the subjective modifiers and their current value and a button to edit them:

EuroNCAP Front ODB Impact - 2014

Subjective Modifiers

Some modifiers are subjective and cannot be calculated automatically from the analysis results.

The user needs to look at the results and decide what values should be applied.

Use the button below to change the values.

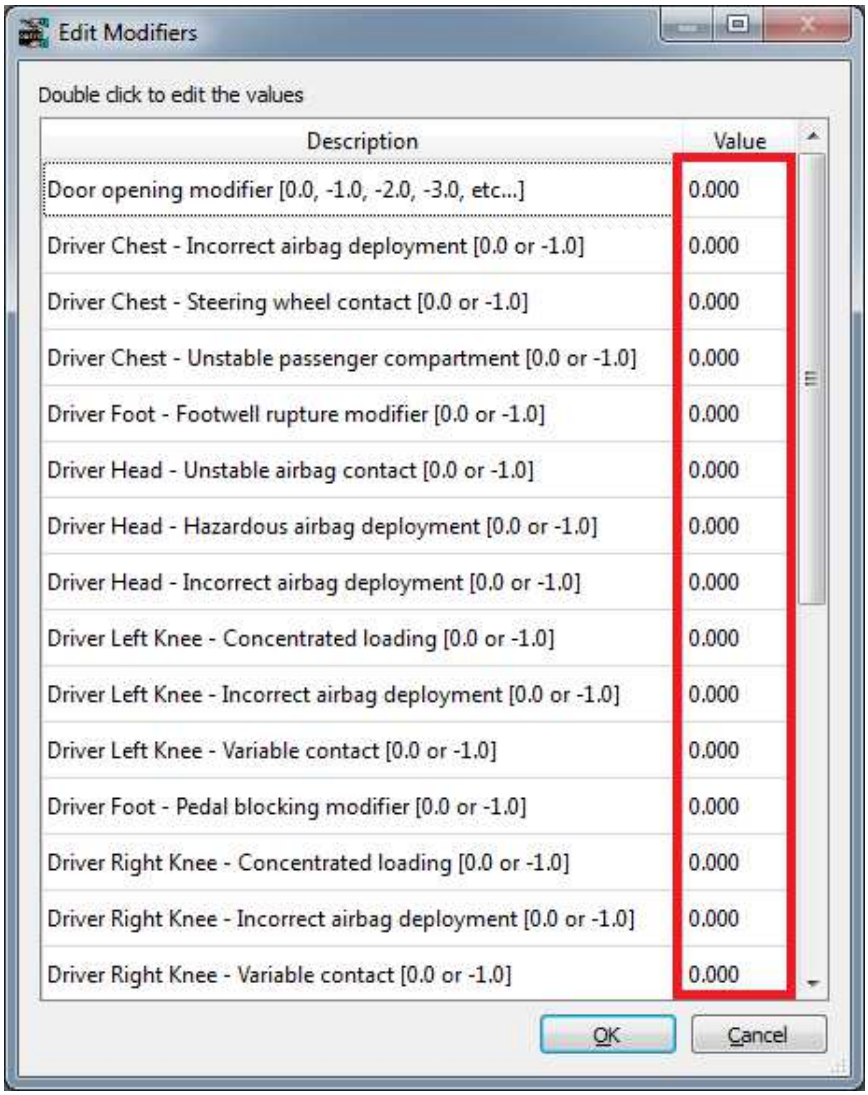
	Driver	Passenger
Head Modifiers	**Value**	**Value**
Unstable Airbag Contact	0.000	0.000
Hazardous Airbag Deployment	0.000	0.000
Incorrect Airbag Deployment	0.000	0.000
Chest Modifiers	**Value**	**Value**
Incorrect Airbag Deployment	0.000	0.000
Steering Wheel Contact	0.000	N/A
Unstable Passenger Compartment	0.000	N/A
Knee, Femur and Pelvis Modifiers	**Value**	**Value**
Left Knee Variable Contact	0.000	0.000
Left Knee Concentrated Loading	0.000	0.000
Left Knee Incorrect Airbag Deployment	0.000	0.000
Right Knee Variable Contact	0.000	0.000
Right Knee Concentrated Loading	0.000	0.000
Right Knee Incorrect Airbag Deployment	0.000	0.000
Foot and Ankle	**Value**	**Value**
Footwell Rupture	0.000	N/A
Pedal Blocking	0.000	N/A

Door	Value
Door Opening	0.000

Set Modifiers

Job: veh_21.key 2/27

Press the ’Set Modifiers’ button and then set the values in the window that pops up:



After setting the values and pressing ’OK’ the template will recalculate the scores. This allows you to carry out ’what-if’ type analyses.

General LS-DYNA Model template

The General LS-DYNA Model template is a basic single analysis template that can be run for any LS-DYNA model.

As with any of the single analysis templates, REPORTER will prompt you to select the keyword file of the job you want to post-process, the directory containing the results, and the directory to which you wish to write images. It will then scan the *.otf (or d3hsp) file in order to provide diagnostic information in a summary table. The information includes the LS-DYNA version used, the computation time and the termination status.

REPORTER will also use the results files to produce an energy balance plot and to produce images of the model at the first and last plot states. These basic report data serve as a quick method for checking the succesful outcome of an LS-DYNA simulation.

General LS-DYNA Vehicle Model template

The General LS-DYNA Vehicle Model template is the same as the General LS-DYNA Model template, with the addition of an intrusion plot output. The intrusion plot shows the deformation of selected parts (e.g. dashboard or driver door components) relative to fixed reference nodes (e.g. three nodes on the undamaged body structure on the far side of the vehicle).

If you have not previously saved the information required to set up the intrusion plot, REPORTER will inform you that it will start PRIMER so you can select the required information interactively. You will need to define:

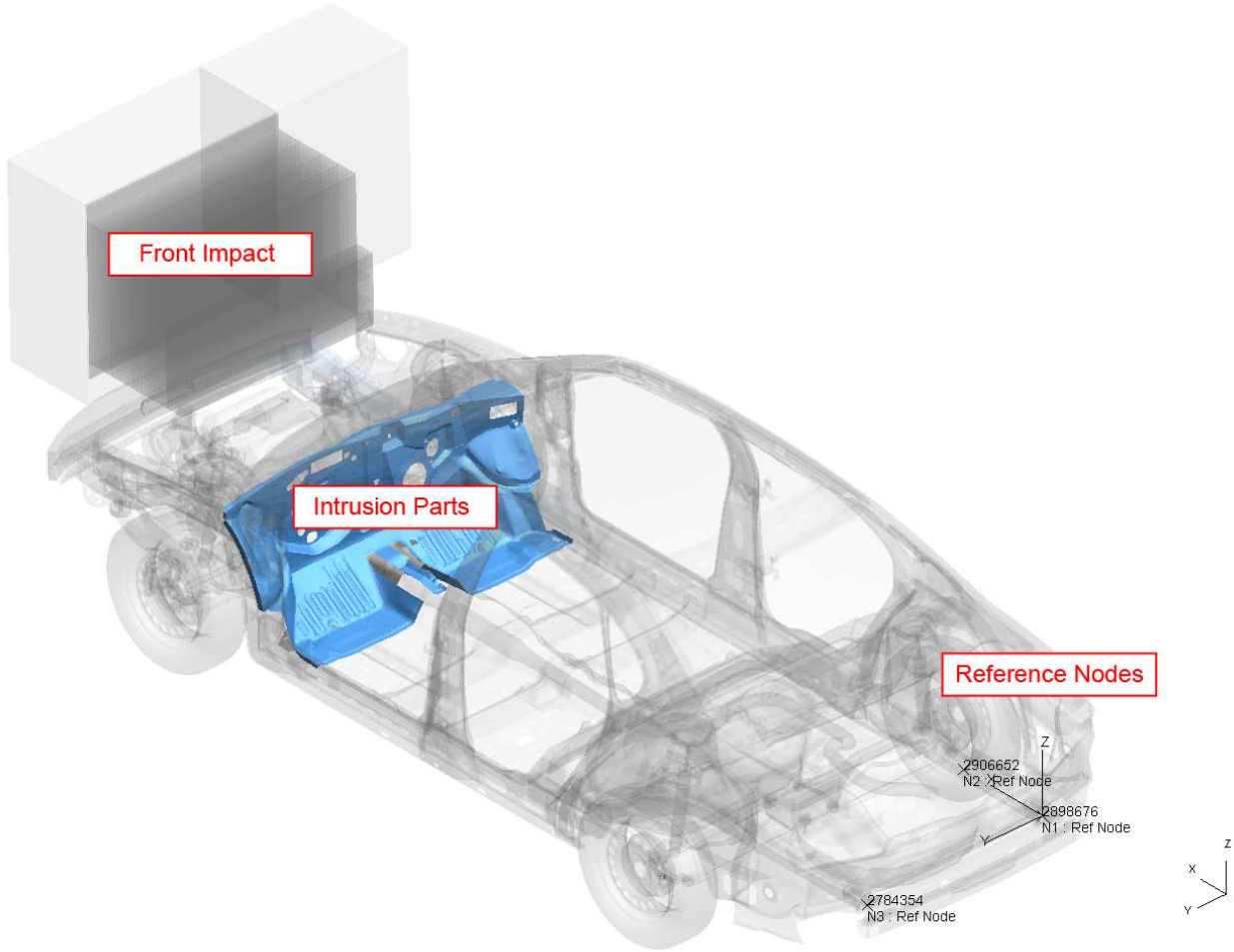
- **Vehicle Impact**

Choose either Front Impact or Side Impact. This controls the camera angle for the intrusion plot. Front Impact assumes that the impact is in the global +X or -X direction whereas Side Impact assumes global +Y or -Y direction.

- **Intrusion Parts**

  Select the parts that will be shown in the intrusion plot. Remaining parts will be blanked.
- **Ref Nodes**

  Displacement magnitude will be plotted relative to a triad of three nodes using D3PLOT's REFERENCE_NODE tool (refer to 6.3.5 in the D3PLOT manual). Select three nodes on a relatively undeformed part of the structure on the far side of the vehicle from the impact, with nodes N1 and N2 aligned with the impact direction.



B.8.2 Multiple analysis templates

For the pedestrian impact protocols multiple analyses are run with impacts on different parts of the vehicle. The scores for each impact are combined to calculate an overall score for the test.

There are two multiple analysis type templates:

- Pedestrian headform impacts
- Pedestrian legform impacts

Generally they follow the same process, but they are different enough that we'll go through an example of how to use them both.

The section will describe how to run them interactively using the menus in REPORTER, but it is also possible to run them in batch mode.

Pedestrian headform

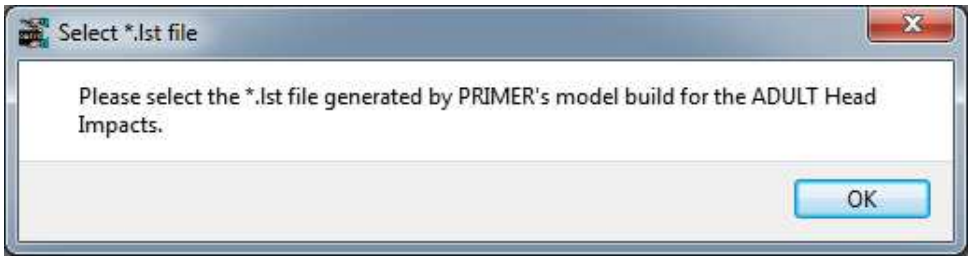
Select the template

Use the **File → Open Library Template** menu and select a template from the **Automotive** tab (see 3.2 Reading an existing template or report for more details).

Generate the template

After selecting the template REPORTER should prompt you to select the *.lst* file for the **adult** head impacts.

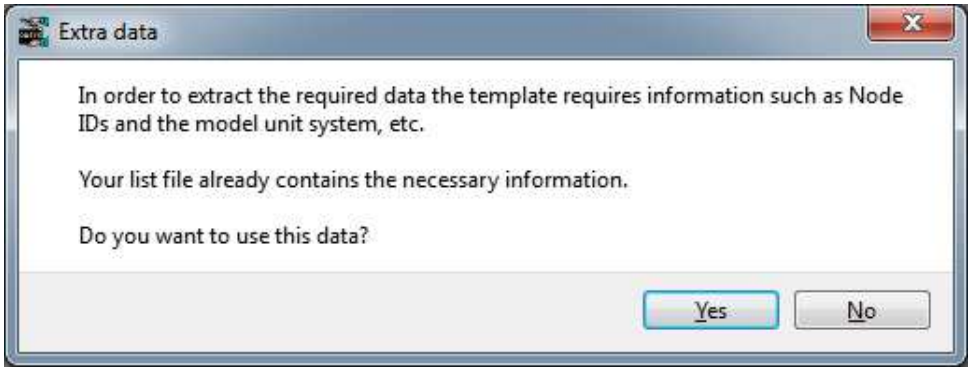
The *.lst* file is a simple text file that lists the file locations of each model keyword file. If you use PRIMERs model build process to create the models it is created automatically. If not, you can create it manually or as part of your own process for building the models. The names of the models are important as they tell the template the location/zone of the impactor. If you have used the pedestrian markup script in PRIMER, they will be named correctly.



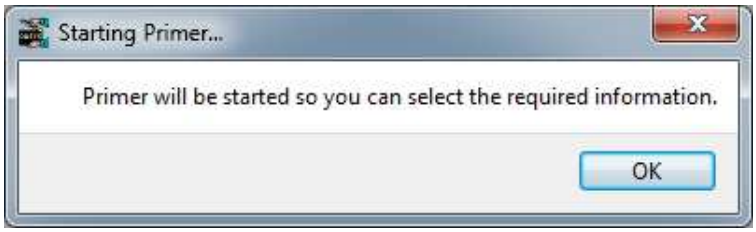
After pressing 'OK' a file selector is mapped for you to select the *.lst* file. If you do not want to process the adult head impacts or you don't have a *.lst* file, press cancel in the file selector.

In order to correctly extract the results needed for the protocol the template needs model information such as Node IDs and the unit system etc. This needs to be supplied to the template from comments written in the *.lst* file and the template will help you to create this information.

The template will scan the *.lst* file to see if it contains the required information. If it does you will be asked if you want to use it:



If you press 'No' or the *.lst* file does not contain the required informtion REPORTER will inform you that it will start PRIMER so you can select the required information interactively:



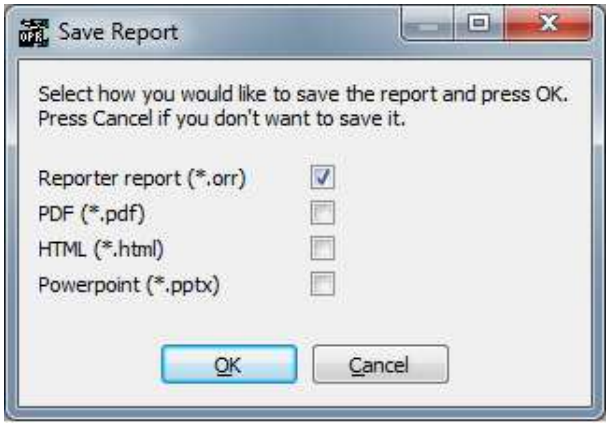
After pressing 'OK' PRIMER will start, the first model in the *.lst* file will be read in and a window will be open for you to interactively select the required information. Select the information in the same way as described in the single analysis template section.

On returning to REPORTER you should be prompted to select the *.lst* file for the **child** head impacts. This follows the same process as for the adult *.lst* file. If you do not want to process the child head impacts or you don't have a *.lst* file, press cancel in the file selector. So long as you have selected at least one *.lst* file, the template should generate.

Finally you will be asked for a directory where REPORTER should write any images.

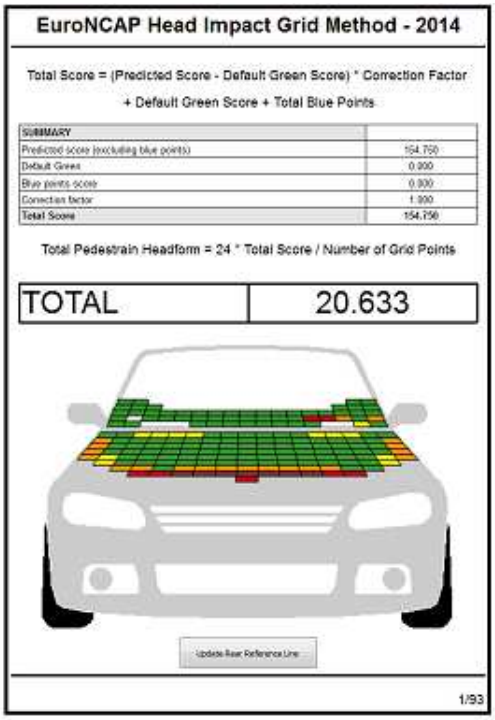
Press OK and select the directory.

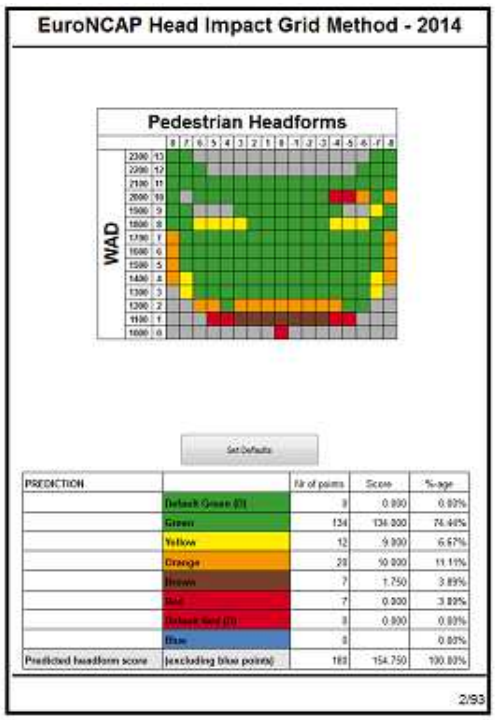
REPORTER should now have all the information it needs. T/HIS will load and carry out the post-processingg according to the selected protocol, generating the required graphs. Once this is finished, REPORTER will ask a final question asking you how/if you want to save the report:

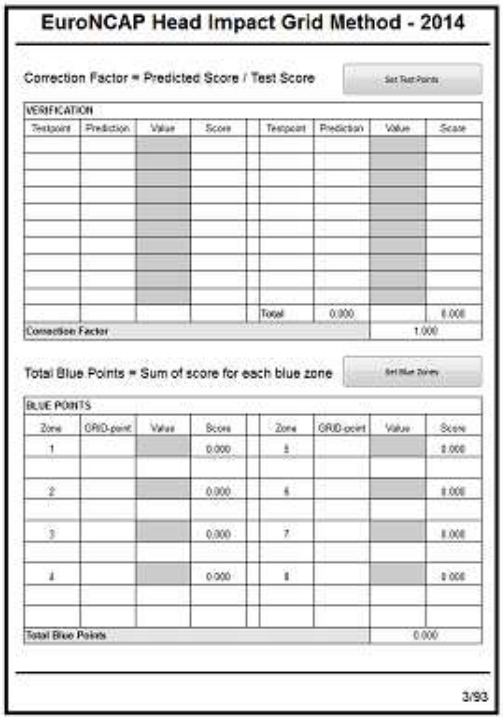


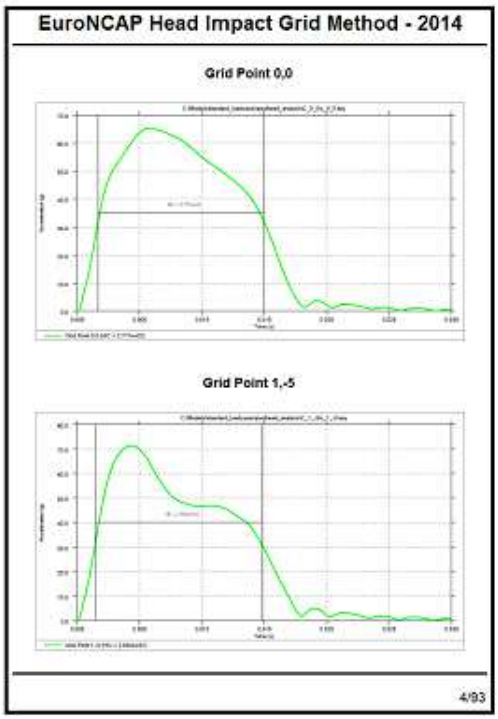
If you don't want to save the report, just press 'Cancel'. If you do, select the format(s) you want to save it in, press 'OK' and select where you want to save it in the file selector that will pop up.

The final report should look something like this, with a front summary page showing the protocol scores in tables and as an image; a pages to set default scores; a page to set test point and blue zone scores; and pages of head acceleration graphs for each impact point:





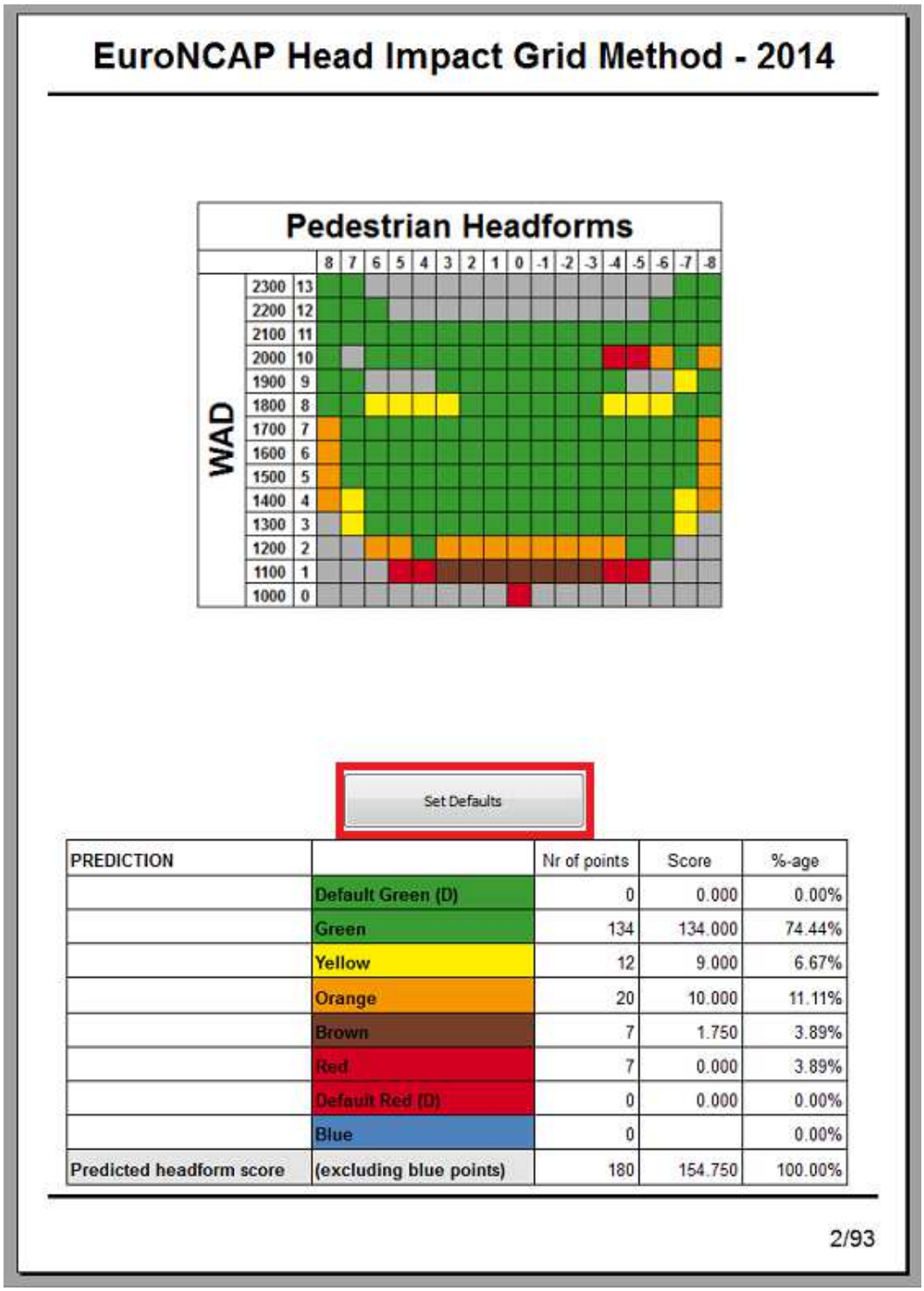




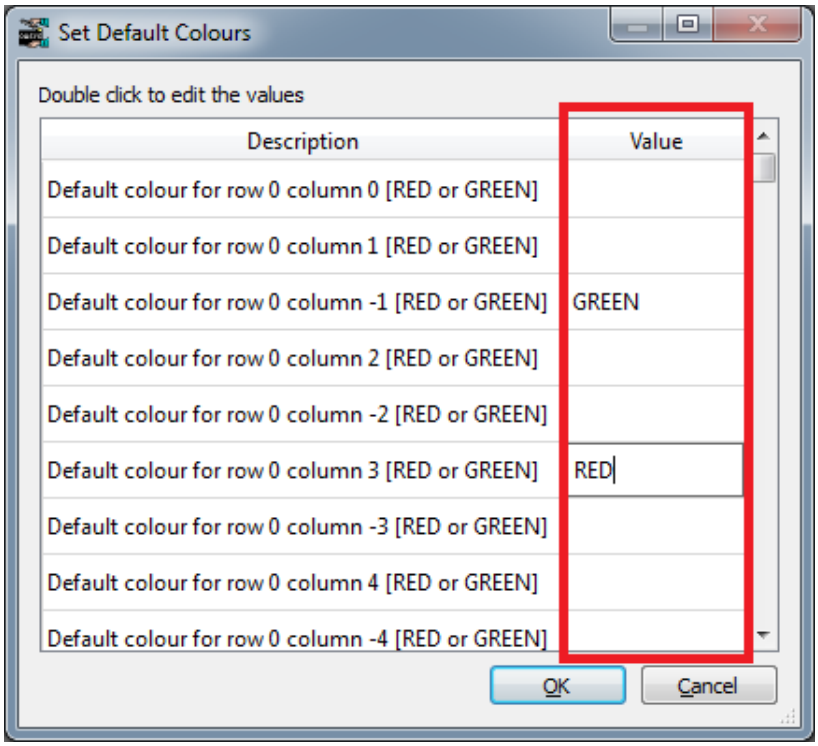
Set default scores

Some points do not need to be tested and can be defaulted either to Green (max score) or Red (min score).

You can manually set default scores on the second page by pressing the 'Set Defaults' button.



This will bring up a window where you can set the default values either to GREEN, RED or leave blank.



Once you have set the values and pressed 'OK' the scores will update automatically.

Alternatively, you can get the template to automatically default points to GREEN when the LST file is read in. This requires a special comment '**$DG:**' before each file location. e.g.

```
$DG:C:\Model\A_1_1\a_1_1.key
C:\Model\A_1_2\a_1_2.key
$DG:C:\Model\A_1_3\a_1_3.key
```

will default the score for the 1st and 3rd model to GREEN and will not attempt to read any results for them.

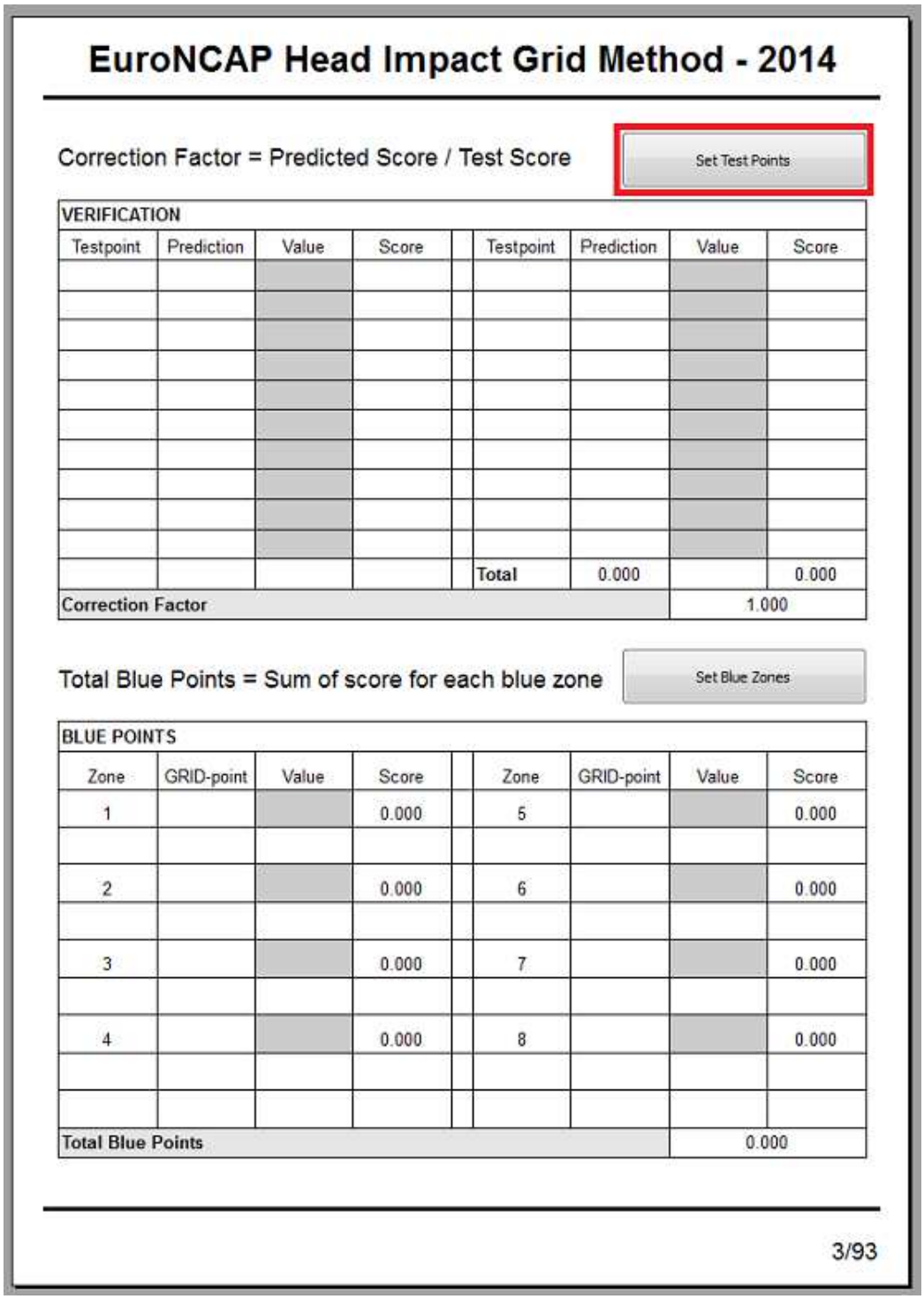
This can be done automatically if you use the pedestrian markup script in PRIMER. You can select an area where you want the points to default to green and PRIMER will add the 'DG:' comments to the correct lines in the LST file.

Test points

The results from the analyses are scaled using a correction factor, which is calculated based on results from a number of real world verification tests. The correction factor is calculated by dividing the actual tested total score of the verification points by the predicted total points of these verification points.

The correction factor is then applied to all points execept for defaulted and blue points.

To specify the test points press the 'Set Test Points' button on the third page:



EuroNCAP Head Impact Grid Method - 2014

Correction Factor = Predicted Score / Test Score

Set Test Points

VERIFICATION								
Testpoint	Prediction	Value	Score		Testpoint	Prediction	Value	Score
					Total	0.000		0.000
Correction Factor							1.000	

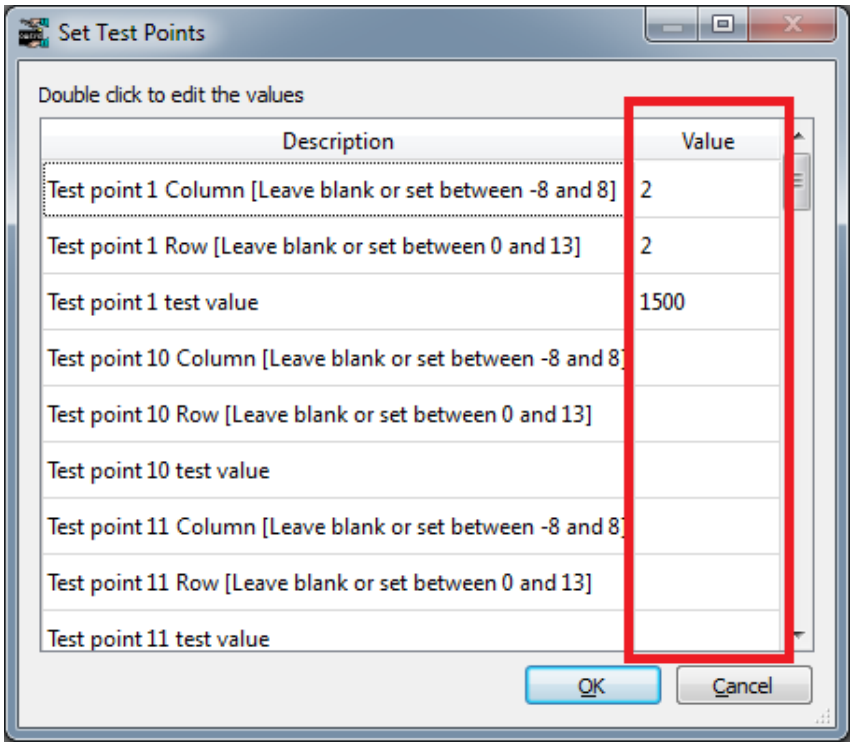
Total Blue Points = Sum of score for each blue zone

Set Blue Zones

BLUE POINTS								
Zone	GRID-point	Value	Score		Zone	GRID-point	Value	Score
1			0.000		5			0.000
2			0.000		6			0.000
3			0.000		7			0.000
4			0.000		8			0.000
Total Blue Points							0.000	

3/93

This will bring up a window where you can enter the test point row, column and value (HIC) for up to 20 test points:



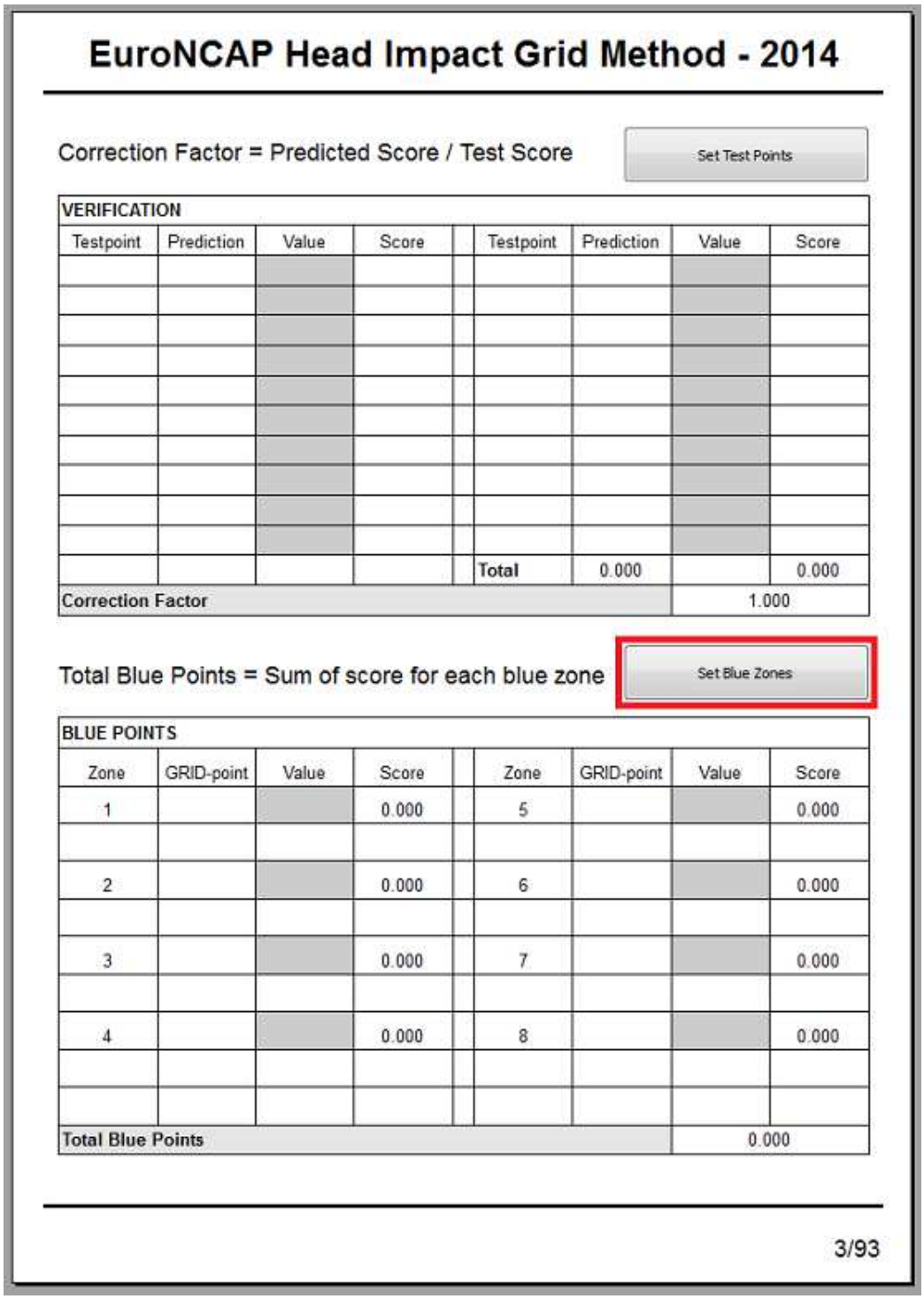
Once you have set the values and pressed ’OK’ the scores will update automatically.

If no test points are specified a correction factor of 1.0 is used.

Blue zones

Some impact point locations may give unpredictable results when analysed and in these cases test data can be used instead. These are specified as blue points, either singly or grouped together in adjacent pairs to form a blue zone. Up to 8 blue zones can be specified. The test results of the blue points are applied to each point in the zone.

To specify blue points press the ’Set Blue Zone’ button on the third page:



EuroNCAP Head Impact Grid Method - 2014

Correction Factor = Predicted Score / Test Score

Set Test Points

VERIFICATION								
Testpoint	Prediction	Value	Score		Testpoint	Prediction	Value	Score
					Total	0.000		0.000
Correction Factor							1.000	

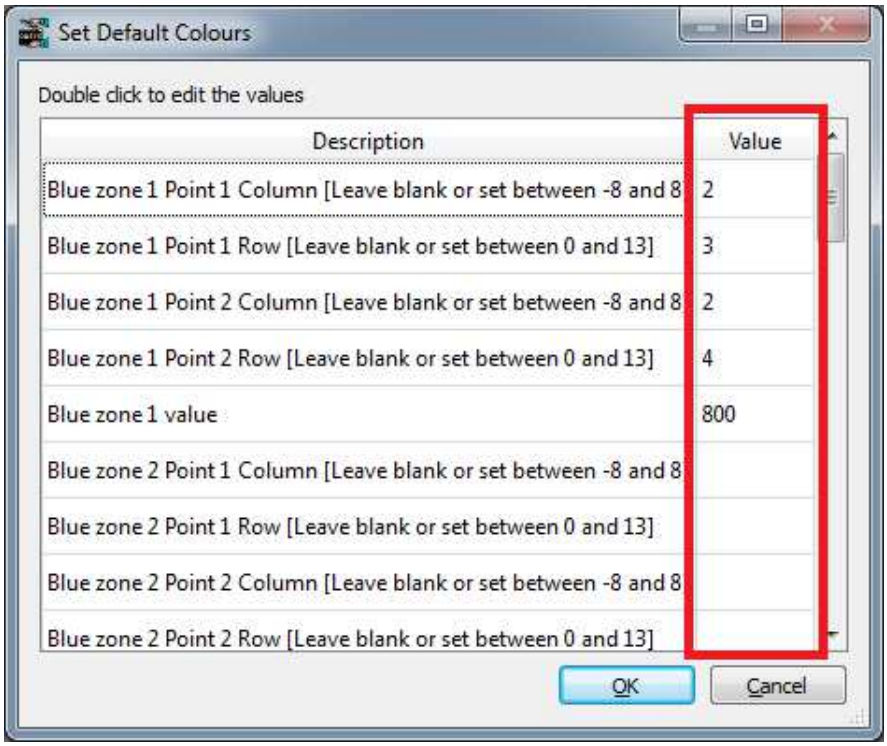
Total Blue Points = Sum of score for each blue zone

Set Blue Zones

BLUE POINTS								
Zone	GRID-point	Value	Score		Zone	GRID-point	Value	Score
1			0.000		5			0.000
2			0.000		6			0.000
3			0.000		7			0.000
4			0.000		8			0.000
Total Blue Points							0.000	

3/93

This will bring up a window where you can enter the test point row, column and value (HIC) for up to 8 blue zones:



Once you have set the values and pressed ’OK’ the scores will update automatically.

Pedestrian legform

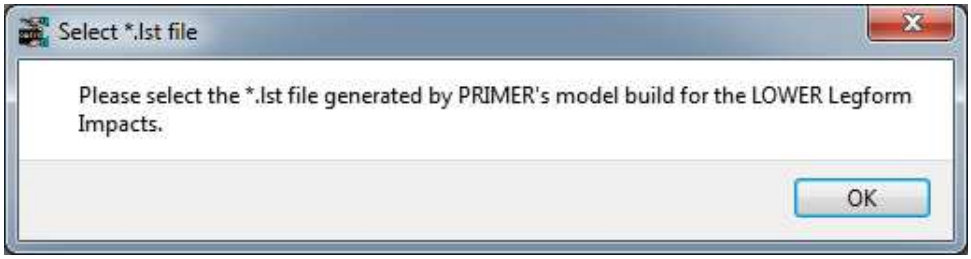
Select the template

Use the **File → Open Library Template** menu and select a template from the **Automotive** tab (see 3.2 Reading an existing template or report for more details).

Generate the template

After selecting the template REPORTER should prompt you to select the *.lst* file for the **lower** leg impacts.

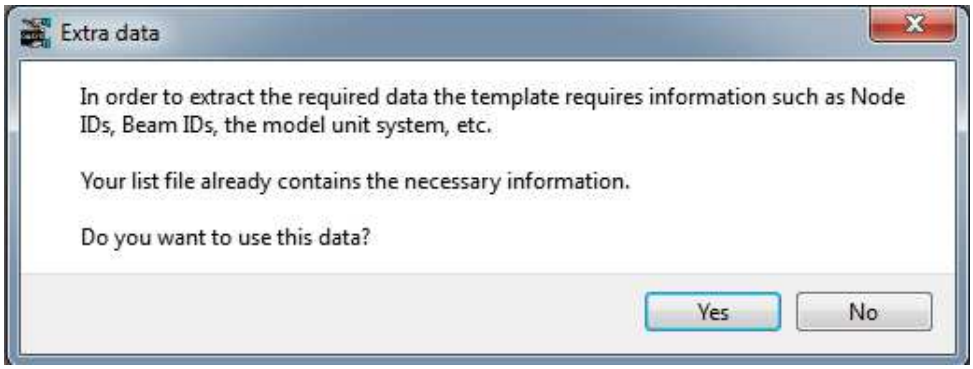
The *.lst* file is a simple text file that lists the file locations of each model keyword file.If you use PRIMERs model build process to create the models it is created automatically. If not, you can create it manually or as part of your own process for building the models. The names of the models are important as they tell the template the location/zone of the impactor. If you have used the pedestrian markup script in PRIMER, they will be named correctly.



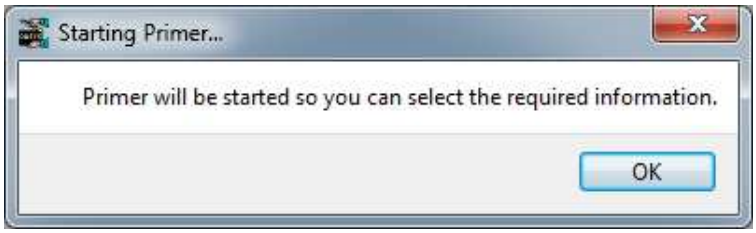
After pressing ’OK’ a file selector is mapped for you to select the *.lst* file. If you do not want to process the lower leg impacts or you don’t have a *.lst* file, press cancel in the file selector.

In order to correctly extract the results needed for the protocol the template needs model information such as Node IDs and the unit system etc. This needs to be supplied to the template from comments written in the *.lst* file and the template will help you to create this information.

The template will scan the *.lst* file to see if it contains the required information. If it does you will be asked if you want to use it:



If you press 'No' or the *.lst* file does not contain the required informtion REPORTER will inform you that it will start PRIMER so you can select the required information interactively:



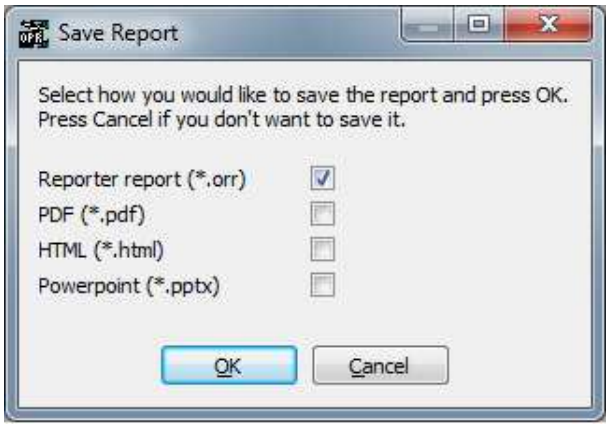
After pressing 'OK' PRIMER will start, the first model in the *.lst* file will be read in and a window will be open for you to interactively select the required information. Select the information in the same way as described in the single analysis template section.

On returning to REPORTER you should be prompted to select the LST file for the **upper** leg impacts. This follows the same process as for the lower *.lst* file. If you do not want to process the upper leg impacts or you don't have a *.lst* file, press cancel in the file selector. So long as you have selected at least one *.lst* file, the template should generate.

Finally you will be asked for a directory where REPORTER should write any images.

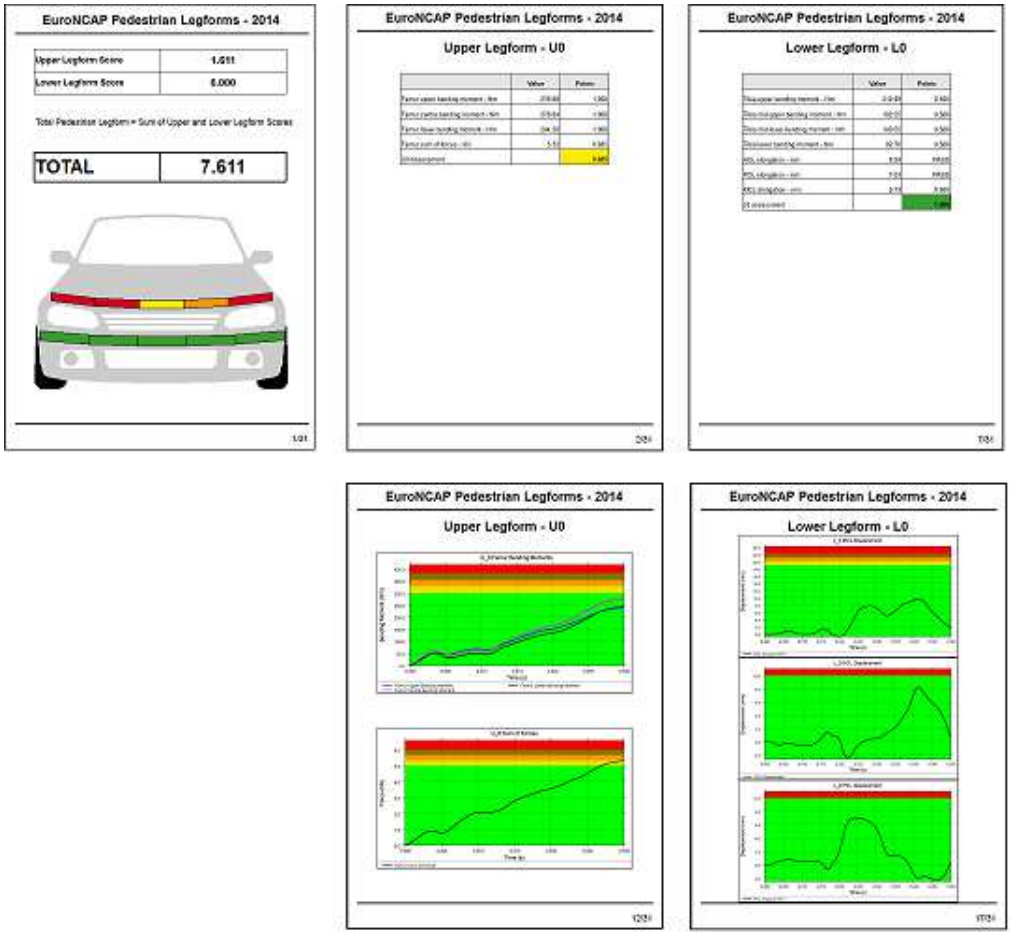
Press OK and select the directory.

REPORTER should now have all the information it needs. T/HIS will load and carry out the post-processingg according to the selected protocol, generating the required graphs. Once this is finished, REPORTER will ask a final question asking you how/if you want to save the report:



If you don't want to save the report, just press 'Cancel'. If you do, select the format(s) you want to save it in, press 'OK' and select where you want to save it in the file selector that will pop up.

The final report should look something like this, with a front summary page showing the protocol scores in tables and as an image and pages of tabulated results and graphs for each impact point:



B.8.3 Running the templates in batch mode

As well as running the templates interactively they can also be run in batch mode, in which case the user is not prompted with questions, but must supply the information through a command line argument.

To run a template in batch, type in the following at a command prompt:

reporter19 .exe -batch -file=*template_name* -varTEMPLATE_ARGS=*args_filename*

[Add the -pdf, -html, -pptx command line arguments to write the report out in the format you want].

Where:

template_ name	The full path and filename of the template you want to use, e.g. C:\oasys19 \reporter_ library\templates\EuroNCAP_Front_ODB_Impact_2017.ort Note that you should enclose it in "s if the path contains any spaces.
args_ filename	The full path and filename of the arguments file, e.g. C:\my_directory\arguments_file.txt Note that you should enclose it in "s if the path contains any spaces.

The *args_filename* is a CSV file containing the arguments to pass to the template in comma separated 'arg_name','arg_ value' pairs. For the EuroNCAP Front ODB Impact template the file can contain the following:

```
KEYWORD_FILE,<keyword_filename>
CSV_FILE,<csv_filename> [OPTIONAL]
RESULTS_DIR,<results_directory> [OPTIONAL]
IMAGES_DIR,<images_directory> [OPTIONAL]
```

As with the interactive case where the template behaves differently depending on the users response to the questions, the interactive case will work differently depending on what arguments are supplied, e.g.:

KEYWORD_FILE specified	Post *END data in keyword file	CSV_FILE specified	Outcome
No	-	-	Template will not run
Yes	No	No	Template will not run
Yes	Yes	No	Template will run using the post *END data
Yes	Yes	Yes	Template will run using the CSV file data
Yes	No	Yes	Template will run using the CSV file data

If `RESULTS_DIR` or `IMAGES_DIR` are not specified then they are set to the keyword file directory.

A description for each argument is given in the table below:

Argument	Description
CSV_FILE	Filename of the CSV file containing the extra data (entity IDs, etc). For single analysis templates, if CSV_FILE is not specified then the data needs to be specified in the keyword file as post-*END data.
JSON_FILE	Used instead of CSV_FILE by ICFD Assessment template. Note that ICFD Assessment template does not support post-*END data.
SIDE_ EXCURSION_ FILE	An additional CSV file required as an input by the Euro NCAP Far Side Impact template. Contains maximum intrusion data produced by the Euro NCAP Side MDB and Side Pole templates.
IMAGES_DIR	Model keyword filename.
RESULTS_DIR	Directory to look for results. If this is not specified the template will look for results in the same directory as the keyword file.
ADULT_LST_FILE	Adult *.lst* filename. For head impact templates, at least one of ADULT_LST_FILE, CHILD_ LST_FILE_FILE or CYCLIST_LST_FILE needs to be specified.
CHILD_LST_FILE	Child *.lst* filename. For head impact templates, at least one of ADULT_LST_FILE, CHILD_ LST_FILE_FILE or CYCLIST_LST_FILE needs to be specified.
CYCLIST_LST_ FILE	Cyclist *.lst* filename. For head impact templates, at least one of ADULT_LST_FILE, CHILD_ LST_FILE_FILE or CYCLIST_LST_FILE needs to be specified.
LOWER_LEG_ LST_FILE	Lower leg *.lst* filename. For leg impact templates, at least one of LOWER_LEG_LST_FILE or UPPER_LEG_LST_FILE needs to be specified.
UPPER_LEG_ LST_FILE	Upper leg *.lst* filename. For leg impact templates, at least one of LOWER_LEG_LST_FILE or UPPER_LEG_LST_FILE needs to be specified.

The list of arguments required for each template is given in the table below. Note that for the single analysis templates, CSV_FILE is required unless the input data is stored in your keyword file as post-*END data.

Template	Required arguments	Optional arguments
C-NCAP Front ODB Impact 2018	KEYWORD_FILE	CSV_FILE, IMAGES_DIR, RESULTS_DIR
C-NCAP Head Impact 2021	(At least one of ADULT_LST_FILE, CHILD_LST_FILE or CYCLIST_LST_ FILE)	ADULT_LST_FILE, CHILD_LST_ FILE, CYCLIST_LST_FILE, IMAGES_DIR
C-NCAP Leg Impact 2021	(At least one of LOWER_LEG_LST_ FILE or UPPER_LEG_LST_FILE)	LOWER_LEG_LST_FILE, UPPER_ LEG_LST_FILE, IMAGES_DIR

C-NCAP MPDB 2022-23 Compatiblity Assessment	DEFAULT_DIR, DEFAULT_JOB, CSV_FILE	IMAGES_DIR
C-NCAP MPDB 2022 Occupant Assessment	KEYWORD_FILE	CSV_FILE, IMAGES_DIR, RESULTS_DIR
Euro NCAP Far Side Impact 2022	KEYWORD_FILE, SIDE_EXCURSION_FILE	CSV_FILE, IMAGES_DIR, RESULTS_DIR
Euro NCAP Front FFB Impact 2017	KEYWORD_FILE	CSV_FILE, IMAGES_DIR, RESULTS_DIR
Euro NCAP Front ODB Impact 2017	KEYWORD_FILE	CSV_FILE, IMAGES_DIR, RESULTS_DIR
Euro NCAP Head Impact 2023	(At least one of ADULT_LST_FILE and CHILD_LST_FILE)	ADULT_LST_FILE, CHILD_LST_FILE, CYCLIST_LST_FILE, IMAGES_DIR
Euro NCAP Leg Impact 2023	(At least one of LOWER_LEG_LST_FILE or UPPER_LEG_LST_FILE)	LOWER_LEG_LST_FILE, UPPER_LEG_LST_FILE, IMAGES_DIR
Euro NCAP MPDB 2020-23 Compatiblity Assessment	DEFAULT_DIR, DEFAULT_JOB, CSV_FILE	IMAGES_DIR
Euro NCAP MPDB 2020 Occupant Assessment	KEYWORD_FILE	CSV_FILE, IMAGES_DIR, RESULTS_DIR
Euro NCAP Side MDB Impact 2022	KEYWORD_FILE	CSV_FILE, IMAGES_DIR, RESULTS_DIR
Euro NCAP Side Pole Impact 2022	KEYWORD_FILE	CSV_FILE, IMAGES_DIR, RESULTS_DIR
General LS-DYNA Vehicle Model	KEYWORD_FILE	CSV_FILE, IMAGES_DIR, RESULTS_DIR
GTR Head Impact 2020	(At least one of ADULT_LST_FILE and CHILD_LST_FILE)	ADULT_LST_FILE, CHILD_LST_FILE, IMAGES_DIR
GTR Leg Impact 2019	(At least one of LOWER_LEG_LST_FILE or UPPER_LEG_LST_FILE)	LOWER_LEG_LST_FILE, UPPER_LEG_LST_FILE, IMAGES_DIR
IIHS Front ODB Impact 2021	KEYWORD_FILE	CSV_FILE, IMAGES_DIR, RESULTS_DIR
IIHS Front ODB Impact 2021 – Structure Only	KEYWORD_FILE	CSV_FILE, IMAGES_DIR, RESULTS_DIR
IIHS Front SOB Impact 2021	KEYWORD_FILE	CSV_FILE, IMAGES_DIR, RESULTS_DIR
IIHS Front SOB Impact 2021 – Structure Only	KEYWORD_FILE	CSV_FILE, IMAGES_DIR, RESULTS_DIR
IIHS Side MDB Impact 2021	KEYWORD_FILE	CSV_FILE, IMAGES_DIR, RESULTS_DIR
IIHS Side MDB Impact 2021 – Structure Only	KEYWORD_FILE	CSV_FILE, IMAGES_DIR, RESULTS_DIR

JNCAP Leg Head Impact 2018	(At least one of LOWER_LEG_LST_ FILE or UPPER_LEG_LST_FILE)	LOWER_LEG_LST_FILE, UPPER_ LEG_LST_FILE, IMAGES_DIR
KNCAP Leg Head Impact 2019	(At least one of LOWER_LEG_LST_ FILE or UPPER_LEG_LST_FILE)	LOWER_LEG_LST_FILE, UPPER_ LEG_LST_FILE, IMAGES_DIR
USNCAP Front FFB Impact 2015	KEYWORD_FILE	CSV_FILE, IMAGES_DIR, RESULTS_DIR
USNCAP Side MDB Impact 2015	KEYWORD_FILE	CSV_FILE, IMAGES_DIR, RESULTS_DIR
USNCAP Side Pole Impact 2015	KEYWORD_FILE	CSV_FILE, IMAGES_DIR, RESULTS_DIR

C. FAQ

This section gives answers to some common questions which have been asked about REPORTER. Over time this FAQ will be extended. If the answer to your question is not here then contact Oasys Ltd for support.

C.1 Running REPORTER

1.1 Can I run REPORTER from the command line?
1.2 Do I need a license to run REPORTER?
1.3 How do I get REPORTER to run automatically after my LS-DYNA job finishes?
1.4 How do I run REPORTER in batch mode?

C.2 Generating output

2.1 None of my scripts/programs work on windows

C.3 Extending REPORTER

3.1 Can I write my own scripts?
3.2 Can I add new scripts/images/pages to the library?

C.4 Other questions

4.1 Text appears to be bigger/smaller on the screen than in a postscript/pdf file
4.2 REPORTER doesn't have xxxx capability. Can you add it?

Answers

1.1 **Can I run REPORTER from the command line?**
Yes you can. See appendix A for a list of command line options.

1.2 **Do I need a license to run REPORTER?**
To run REPORTER you need a valid license for REPORTER or alternatively a valid license for D3PLOT, T/HIS or PRIMER. To get maximum benefit from REPORTER, D3PLOT and T/HIS are required.

1.3 **How do I get REPORTER to run automatically after my LS-DYNA job finishes?**
Use the Oasys Ltd shell to submit your job which has options to allow you to run REPORTER automatically.

1.4 **How do I run REPORTER in batch mode?**
REPORTER does not have a batch mode which means that it requires a display to be able to draw things on. In reality this is not too much of a problem as D3PLOT will also need a display. You can give a DISPLAY that REPORTER can display back to. This can be a computer which is left logged in or a virtual display using xvfb. Additionally to stop REPORTER from pausing to ask for confirmations you should use the `-batch` command line argument.

2.1 **None of my scripts/programs work on windows**
1. Do you have perl, python, Tcl (or whatever your script is written in) installed on your machine?
2. Do you have the correct file extensions and associations for this type of file. e.g. for perl the script should be 'script.pl' and this should be associated with the perl executable on your machine.
3. Do any of the program arguments have spaces in them? If so you may need to quote them. For example:
```
%MYPATH%\scripts\title.pl "C:\my directory\my file with spaces.key"
```

3.1 **Can I write my own scripts?**
Yes. See chapter13 and appendix D for more details.

3.2 **Can I add new scripts/images/pages to the library?**
Yes. See appendix B for more details.

4.1 **Text appears to be bigger/smaller on the screen than in a postscript/pdf file.**
This can be a problem on Unix machines. Unlike windows machines which use true type fonts, fonts on unix are stored as bitmaps. Only certain sizes are actually available. If you request a size that is not available the one that is displayed could be the wrong size.
To get a list of the fonts (and sizes) on your unix machine use the command **`xlsfonts`**.
If you are trying to see how much space some text will take up in the presentation view try zooming into the page. This may help.

4.2 **REPORTER doesn't have xxxx capability. Can you add it?**
We will try. Please contact Oasys Ltd support to discuss it.

D. JavaScript class reference

This appendix documents the javascript classes that REPORTER uses for scripting. It is not an introduction to scripting. See chapter13 for that.

REPORTER extends the javascript interpreter with the following new classes.

Class	Description
Colour	The Colour class defines colours in REPORTER.
File	The File class allows you to read and write from text files in REPORTER.
Image	The Image class allows you to create bitmaps in REPORTER.
Include	The Include class from PRIMER is used in limited form in REPORTER.
Item	The Item class gives access to items in REPORTER.
Options	The Options class is used to get/set options for certain functions in REPORTER.
Page	The Page class gives access to pages in REPORTER.
Reporter	The Reporter class is the root class for objects, properties etc in REPORTER.
Template	The Template class gives access to templates in REPORTER.
Utils	The Utils class contains various useful utility functions in REPORTER.
Variable	The Variable class gives access to variables in REPORTER.
Window	The Window class gives access to windows for a graphical user interface in REPORTER.

In addition REPORTER also adds some new methods to the global Javascript object.

global class

The global class is the root object in Javascript. More...

The REPORTER JavaScript API provides many class constants, properties and methods. 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

Class functions

- Batch()
- Debug(string*[Any valid javascript type]*)
- Exit()
- GetCurrentDirectory()
- LogError(arg1*[Any valid javascript type], ...[Any valid javascript type]*)
- LogPrint(arg1*[Any valid javascript type], ...[Any valid javascript type]*)
- LogWarning(arg1*[Any valid javascript type], ...[Any valid javascript type]*)
- Output(string*[Any valid javascript type]*)
- SetCurrentDirectory(directory*[string]*)
- System(string*[Any valid javascript type]*)
- debug() [deprecated]
- exit() [deprecated]
- output() [deprecated]

global properties

Name	Type	Description
reporter	Reporter	This property is deprecated in version 12.0. It is only provided to keep old scripts working. We strongly advise against using it in new scripts. Support may be removed in future versions. [deprecated]

Detailed Description

When Reporter is started a **single** global class object is created. All of the standard JavaScript functions and properties are available from it.
In addition an instance of a Reporter class is available, from the global reporter property. The reporter object allows you to access the properties and templates used in Reporter.

Details of functions

Batch() [static]

Description

This method can be used to test whether REPORTER is running in batch mode or not.

Arguments

No arguments

Return type

true/false

Example

To check if REPORTER is running in batch mode

```
if (Batch()) { do something }
```

Debug(string*[Any valid javascript type]*) [static]

Description

Print a string to log file for debugging. Anything that you call the debug method on will be 'printed' to the log file window. **Note that a carriage return will automatically be added**.

Arguments

Name	Type	Description
string	Any valid javascript type	The string/item that you want to debug

Return type

No return value

Example

To print string "Hello, world!" to the debug log file

```
Debug("Hello, world!");
```

Exit() [static]

Description

Stop execution and exit from script

Arguments

No arguments

Return type

No return value

Example

Exit from script with

```
Exit();
```

GetCurrentDirectory() [static]

Description

Return the current working directory for REPORTER.

Arguments

No arguments

Return type

string

Example

To return the current directory

```
var dir = GetCurrentDirectory();
```

LogError(arg1*[Any valid javascript type]*, ...*[Any valid javascript type]*) [static]

Description

Print an error to log file. Anything that you print will be output to the log file window in bold red text. **Note that a carriage return will automatically be added**.

Arguments

Name	Type	Description
arg1	Any valid javascript type	The string/item that you want to print
...	Any valid javascript type	The string/item that you want to print

Return type

No return value

Example

To give error "Error: something has gone wrong" to the log file

```
LogError("Error: something has gone wrong");
```

Any number of arguments can be given. They will be concatenated. e.g.

```
LogError("The value of i is ", i, " elephants");
```

LogPrint(arg1*[Any valid javascript type]*, ...*[Any valid javascript type]*) [static]

Description

Print a string to log file. Anything that you print will be output to the log file window. **Note that a carriage return will automatically be added**.

Arguments

Name	Type	Description
arg1	Any valid javascript type	The string/item that you want to print
...	Any valid javascript type	The string/item that you want to print

Return type

No return value

Example

To print string "Hello, world!" to the log file
```
LogPrint("Hello, world!");
```

Any number of arguments can be given. They will be concatenated. e.g.
```
LogPrint("The value of i is ", i, " elephants");
```

LogWarning(arg1*[Any valid javascript type]*, ...*[Any valid javascript type]*) [static]

Description

Print a warning to log file. Anything that you print will be output to the log file window in red text. **Note that a carriage return will automatically be added**.

Arguments

Name	Type	Description
arg1	Any valid javascript type	The string/item that you want to print
...	Any valid javascript type	The string/item that you want to print

Return type

No return value

Example

To give warning "Warning: something has gone wrong" to the log file
```
LogWarning("Warning: something has gone wrong");
```

Any number of arguments can be given. They will be concatenated. e.g.
```
LogWarning("The value of i is ", i, " elephants");
```

Output(string*[Any valid javascript type]*) [static]

Description

Output a string from a script. **Note that a carriage return is not automatically added**.

Arguments

Name	Type	Description
string	Any valid javascript type	The string/item that you want to print

Return type

No return value

Example

To output string "Hello, world!" with a carriage return:
```
Output("Hello, world!\n");
```

SetCurrentDirectory(directory*[string]*) [static]

Description

Set the current working directory for REPORTER.

Arguments

Name	Type	Description
directory	string	The directory that you want to change to

Return type

true if successful, false if not

Example

To set the current directory to C:\temp

```
var status = SetCurrentDirectory("C:\\temp");
```

System(string*[Any valid javascript type]*) [static]

Description

Do a system command outside REPORTER.

Arguments

Name	Type	Description
string	Any valid javascript type	The system command that you want to do

Return type

integer (probably zero if command successful but is implementation-dependant)

Example

To make the directory "example"

```
System("mkdir example");
```

---

debug() [static] [deprecated]

This function is deprecated in version 12.0. It is only provided to keep old scripts working. We strongly advise against using it in new scripts. Support may be removed in future versions.

Description

Please use Debug() instead

Arguments

No arguments

Return type

No return value

exit() [static] [deprecated]

This function is deprecated in version 12.0. It is only provided to keep old scripts working. We strongly advise against using it in new scripts. Support may be removed in future versions.

Description

Please use Exit() instead

Arguments

No arguments

Return type

No return value

output() [static] [deprecated]

This function is deprecated in version 12.0. It is only provided to keep old scripts working. We strongly advise against using it in new scripts. Support may be removed in future versions.

Description

Please use Output() instead

Arguments

No arguments

Return type

No return value

Colour class

The Colour class gives access to colours in Reporter. More...

The REPORTER JavaScript API provides many class constants, properties and methods. 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

Class functions

- Black()
- Blue()
- Cyan()
- Green()
- Grey10()
- Grey20()
- Grey30()
- Grey40()
- Grey50()
- Grey60()
- Grey70()
- Grey80()
- Grey90()
- Magenta()
- None()
- RGB(red*[integer]*, green*[integer]*, blue*[integer]*)
- Red()
- White()
- Yellow()

Colour properties

Name	Type	Description
blue (read only)	integer	Colour blue component (0-255)
green (read only)	integer	Colour green component (0-255)
name (read only)	string	Colour name
red (read only)	integer	Colour red component (0-255)

Detailed Description

The Colour class is used to define colours, either by predefined colours or by RGB values. The easiest way to set the colour of something is to use the predefined colour methods. e.g. to set the text colour of item i to red:

```
i.textColour = Colour.Red();
```

For other colours use Colour.RGB().

Details of functions

Black() [static]

Description

Creates a black colour

Arguments

No arguments

Return type

Colour object

Example

To set the text colour of item i to black:

```
i.textColour = Colour.Black();
```

Blue() [static]

Description

Creates a blue colour

Arguments

No arguments

Return type

Colour object

Example

To set the text colour of item i to blue:

```
i.textColour = Colour.Blue();
```

Cyan() [static]

Description

Creates a cyan colour

Arguments

No arguments

Return type

Colour object

Example

To set the text colour of item i to cyan:
```
i.textColour = Colour.Cyan();
```

Green() [static]

Description

Creates a green colour

Arguments

No arguments

Return type

Colour object

Example

To set the text colour of item i to green:
```
i.textColour = Colour.Green();
```

Grey10() [static]

Description

Creates a 10% grey colour

Arguments

No arguments

Return type

Colour object

Example

To set the text colour of item i to 10% grey:
```
i.textColour = Colour.Grey10();
```

Grey20() [static]

Description

Creates a 20% grey colour

Arguments

No arguments

Return type

Colour object

Example

To set the text colour of item i to 10% grey:

```
i.textColour = Colour.Grey20();
```

Grey30() [static]

Description

Creates a 30% grey colour

Arguments

No arguments

Return type

Colour object

Example

To set the text colour of item i to 30% grey:

```
i.textColour = Colour.Grey30();
```

Grey40() [static]

Description

Creates a 40% grey colour

Arguments

No arguments

Return type

Colour object

Example

To set the text colour of item i to 40% grey:

```
i.textColour = Colour.Grey40();
```

Grey50() [static]

Description

Creates a 50% grey colour

Arguments

No arguments

Return type

Colour object

Example

To set the text colour of item i to 50% grey:

```
i.textColour = Colour.Grey50();
```

Grey60() [static]

Description

Creates a 60% grey colour

Arguments

No arguments

Return type

Colour object

Example

To set the text colour of item i to 60% grey:

```
i.textColour = Colour.Grey60();
```

Grey70() [static]

Description

Creates a 70% grey colour

Arguments

No arguments

Return type

Colour object

Example

To set the text colour of item i to 70% grey:

```
i.textColour = Colour.Grey70();
```

Grey80() [static]

Description

Creates a 80% grey colour

Arguments

No arguments

Return type

Colour object

Example

To set the text colour of item i to 80% grey:

```
i.textColour = Colour.Grey80();
```

Grey90() [static]

Description

Creates a 90% grey colour

Arguments

No arguments

Return type

Colour object

Example

To set the text colour of item i to 90% grey:

```
i.textColour = Colour.Grey90();
```

Magenta() [static]

Description

Creates a magenta colour

Arguments

No arguments

Return type

Colour object

Example

To set the text colour of item i to magenta:

```
i.textColour = Colour.Magenta();
```

None() [static]

Description

No colour

Arguments

No arguments

Return type

Colour object

Example

To set the fill colour of item i to none:

```
i.fillColour = Colour.None();
```

RGB(red*[integer]*, green*[integer]*, blue*[integer]*) [static]

Description

Creates a colour from red, green and blue components

Arguments

Name	Type	Description
red	integer	red component of colour (0-255).
green	integer	green component of colour (0-255).
blue	integer	blue component of colour (0-255).

Return type

Colour object

Example

To set the text colour of item i to red:

```
i.textColour = Colour.RGB(255, 0, 0);
```

Red() [static]

Description

Creates a red colour

Arguments

No arguments

Return type

Colour object

Example

To set the text colour of item i to red:

```
i.textColour = Colour.Red();
```

White() [static]

Description

Creates a white colour

Arguments

No arguments

Return type

Colour object

Example

To set the text colour of item i to white:

```
i.textColour = Colour.White();
```

Yellow() [static]

Description

Creates a yellow colour

Arguments

No arguments

Return type

Colour object

Example

To set the text colour of item i to yellow:

```
i.textColour = Colour.Yellow();
```

File class

The File class allows you to read and write from text files. More...

The REPORTER JavaScript API provides many class constants, properties and methods. 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

Class functions

- ConvertSeparators(filename*[string]*)
- Copy(source*[string]*, dest*[string]*)
- Delete(filename*[string]*)
- Directory(filename*[string]*)
- DriveMapFilename(filename*[string]*, format*[constant]*)
- Exists(filename*[string]*)
- FindFiles(directory*[string]*, pattern*[string]*, recursive*[boolean]*)
- IsAbsolute(filename*[string]*)
- IsDirectory(filename*[string]*)
- IsFile(filename*[string]*)
- Mkdir(name*[string]*)
- Move(source*[string]*, dest*[string]*)
- SimplifyName(filename*[string]*)
- Size(filename*[string]*)

Member functions

- Close()
- FindLineContaining(contain1*[string]*, contain2 (optional)*[string]*, contain3 (optional)*[string]*, ... containn (optional)*[string]*)
- FindLineMatching(regex*[RegExp]*)
- FindLineStarting(start1*[string]*, start2 (optional)*[string]*, start3 (optional)*[string]*, ... startn (optional)*[string]*)
- Flush()
- ReadChar()
- ReadLine()
- ReadLongLine()
- Seek(position*[integer]*)
- Write(string*[Any valid javascript type]*)

File constants

Name	Description
File.APPEND	Flag to open file for appending
File.EOF	Flag to indicate end of file
File.READ	Flag to open file for reading
File.WRITE	Flag to open file for writing

Detailed Description

The File class allows you to read text and write text to files. There are various functions available that allow to to find lines matching specific strings or regular expressions when reading.
Additionally, there are a number of utility functions to check if a file exists or is a directory etc.

Constructor

new File(filename*[string]*, mode*[constant]*)

Description

Create a new File object for reading and writing text files.

Arguments

Name	Type	Description
filename	string	Filename of the file you want to read/write. If reading, the file must exist. If writing, the file will be overwritten if it already exists
mode	constant	The mode to open the file with. Can be File.READ, File.WRITE or File.APPEND

Return type

File object

Example

To create a new file object to read file "/data/test/file.txt"

```
var f = new File("/data/test/file.txt", File.READ);
```

Details of functions

Close()

Description

Close a file opened by a File object.

Arguments

No arguments

Return type

No return value

Example

To close File object f.

```
f.Close();
```

ConvertSeparators(filename*[string]*) [static]

Description

Convert directory separators to the correct type for this operating system

Arguments

Name	Type	Description
filename	string	Filename you want to convert separators on.

Return type

string filename

Example

e.g. on windows the filename "c:/test/file.key" would be converted to "c:\test\file.key" by

```
var converted = File.ConvertSeparators("c:/test/file.key");
```

Copy(source*[string]*, dest*[string]*) [static]

Description

Copy a file

Arguments

Name	Type	Description
source	string	Source filename you want to copy.
dest	string	Destination filename you want to copy source file to. Note that if a file with the name dest already exists it will not be overwritten. Delete the file first with File.Delete().

Return type

true if copy successful, false otherwise.

Example

To copy the file "/data/test/file.txt" to "/data/test/file.txt_backup"

```
var copied = File.Copy("/data/test/file.txt", "/data/test/file.txt_backup");
```

Delete(filename*[string]*) [static]

Description

Delete a file

Arguments

Name	Type	Description
filename	string	Filename you want to delete.

Return type

true if successful, false if not

Example

To delete the file "/data/test/file.txt"

```
var deleted = File.Delete("/data/test/file.txt");
```

Directory(filename*[string]*) [static]

Description

Extract directory name from an absolute filename

Arguments

Name	Type	Description
filename	string	Absolute filename you want to extract directory from.

Return type

string directory

Example

To extract the directory "/data/test/" from file "/data/test/file.key"

```
var directory = File.Directory("/data/test/file.key");
```

DriveMapFilename(filename*[string]*, format*[constant]*) [static]

Description

Changes a filename or directory name to the correct format for a specific operating system using the directory mappings (if present).

Arguments

Name	Type	Description
filename	string	Filename you want to drive map.
format	constant	The format for the file/directory name. Can be Include.NATIVE, Include.UNIX, or Include.WINDOWS.

Return type

string containing drive mapped filename.

Example

If REPORTER has drive S:\ mapped to /data/ (by using the oasys*drive_s preference)

```
var mapped = File.DriveMapFilename("/data/test/file.ptf", Include.WINDOWS);
```

mapped will be "S:\test\file.ptf".

```
var mapped = File.DriveMapFilename("S:\\test\\file.ptf", Include.UNIX);
```

mapped will be "/data/test/file.ptf",

Exists(filename*[string]*) [static]

Description

Check if a file exists

Arguments

Name	Type	Description
filename	string	Filename you want to check for existance.

Return type

true/false

Example

To see if the file "/data/test/file.key" exists

```
if (File.Exists("/data/test/file.key")) { do something }
```

FindFiles(directory*[string]*, pattern*[string]*, recursive*[boolean]*) [static]

Description

Find any files in a directory (and subdirectories if required) matching a pattern

Arguments

Name	Type	Description
directory	string	Directory to look for files in
pattern	string	Pattern to use to find matching files
recursive	boolean	If Reporter should look for files recursively or not

Return type

array filenames

Example

To find all of the files matching the pattern "*.key" recursively from directory /data/test

```
var filelist = File.FindFiles("/data/test/", "*.key", true);
```

FindLineContaining(contain1*[string]*, contain2 (optional)*[string]*, contain3 (optional)*[string]*, ... containn (optional)*[string]*)

Description

Reads a line from a file which contains contain, opened for reading by a File object. To enable this function to be as fast as possible a maximum line length of 256 characters is used. If you expect a file to have lines longer than 256 characters then use ReadLongLine which allows lines of any length. If one argument is used then the line must contain that string. If more than one argument is used then lines which contain argument1 OR argument2 OR argument3 will be returned

Arguments

Name	Type	Description
contain1	string	String which matching lines must contain (maximum length of 256 characters).
contain2 (optional)	string	alternative string which matching lines must contain (maximum length of 256 characters).
contain3 (optional)	string	alternative string which matching lines must contain (maximum length of 256 characters).

... containn (optional)	string	alternative string which matching lines must contain (maximum length of 256 characters).

Return type

string read from file or `File.EOF` if end of file

Example

Loop, reading lines from File object f which contain 'example'.

```
var line;
while ( (line = file.FindLineContaining("example") ) != File.EOF)
{
}
```

FindLineMatching(regex*[RegExp]*)

Description

Reads a line from a file opened for reading by a File object. To enable this function to be as fast as possible a maximum line length of 256 characters is used. If you expect a file to have lines longer than 256 characters then use ReadLongLine which allows lines of any length. Note that this may be much slower than FindLineStarting or FindLineContaining, especially if the regular expression is very complicated.

Arguments

Name	Type	Description
regex	RegExp	Regular expression which matching lines must match with.

Return type

string read from file or `File.EOF` if end of file

Example

Loop, reading lines from File object f which contain digits.

```
var line;
var regex = new RegExp("\\d+");
while ( (line = file.FindLineMatching(regex) ) != File.EOF)
{
}
```

FindLineStarting(start1*[string]*, start2 (optional)*[string]*, start3 (optional)*[string]*, ... startn (optional)*[string]*)

Description

Reads a line from a file which starts with start, opened for reading by a File object. To enable this function to be as fast as possible a maximum line length of 256 characters is used. If you expect a file to have lines longer than 256 characters then use ReadLongLine which allows lines of any length. If one argument is used then the line must start with that string. If more than one argument is used then lines which start argument1 OR argument2 OR argument3 will be returned

Arguments

Name	Type	Description

start1	string	String which matching lines must start with (maximum length of 256 characters).
start2 (optional)	string	alternative string which matching lines must start with (maximum length of 256 characters).
start3 (optional)	string	alternative string which matching lines must start with (maximum length of 256 characters).
... startn (optional)	string	alternative string which matching lines must start with (maximum length of 256 characters).

Return type

string read from file or `File.EOF` if end of file

Example

Loop, reading lines from File object f which start 'example'.

```
var line;
while ( (line = file.FindLineStarting("example") ) != File.EOF)
{
}
```

Flush()

Description

Flushes a file opened for writing by a File object.

Arguments

No arguments

Return type

No return value

Example

To flush File object f.

```
f.Flush();
```

IsAbsolute(filename*[string]*) [static]

Description

Check if a filename is absolute

Arguments

Name	Type	Description
filename	string	Filename you want to test if absolute.

Return type

true/false

Example

To see if the file "/data/test/file.key" is absolute
```
if (File.IsAbsolute("/data/test/file.key")) { do something }
```

IsDirectory(filename*[string]*) [static]

Description

Check if a filename is a directory

Arguments

Name	Type	Description
filename	string	Filename you want to test to see if it is a directory.

Return type

true/false

Example

To see if "/data/test" is a directory
```
if (File.IsDirectory("/data/test")) { do something }
```

IsFile(filename*[string]*) [static]

Description

Check if a filename is a file

Arguments

Name	Type	Description
filename	string	Filename you want to test to see if it is a file (i.e. not a directory).

Return type

true/false

Example

To see if "/data/test" is a file
```
if (File.IsFile("/data/test")) { do something }
```

Mkdir(name*[string]*) [static]

Description

makes a directory

Arguments

Name	Type	Description
name	string	Directory you want to create.

Return type

true if successful

Example

To make directory "/data/test" if it does not exist:

```
if (!File.IsDirectory("/data/test")) File.Mkdir("/data/test");
```

Move(source*[string]*, dest*[string]*) [static]

Description

Move a file

Arguments

Name	Type	Description
source	string	Source filename you want to move.
dest	string	Destination filename you want to move (rename) source file to. Note that if a file with the name dest already exists it will not be overwritten. Delete the file first with File.Delete().

Return type

true if move successful, false otherwise.

Example

To move the file "/data/test/file.txt" to "/data/test/file.txt_backup"

```
var moved = File.Move("/data/test/file.txt", "/data/test/file.txt_backup");
```

ReadChar()

Description

Reads a single character from a file opened for reading by a File object.

Arguments

No arguments

Return type

character read from file or `File.EOF` if end of file

Example

Loop, reading characters from File object f.

```
var c;
while ( (c = f.ReadChar()) != undefined) { ... }
```

ReadLine()

Description

Reads a line from a file opened for reading by a File object. To enable this function to be as fast as possible a maximum line length of 256 characters is used. If you expect a file to have lines longer than 256 characters then use ReadLongLine which allows lines of any length.

Arguments

No arguments

Return type

string read from file or `File.EOF` if end of file

Example

Loop, reading lines from File object f.

```
var line;
while ( (line = file.ReadLine() ) != File.EOF)
{
}
```

ReadLongLine()

Description

Reads a line from a file opened for reading by a File object. The line can be any length. If your file has lines shorter than 256 characters then you may want to use ReadLine instead which is faster.

Arguments

No arguments

Return type

string read from file or `File.EOF` if end of file

Example

Loop, reading lines from File object f.

```
var line;
while ( (line = file.ReadLongLine() ) != File.EOF)
{
}
```

Seek(position*[integer]*)

Description

Sets the file position for reading a file

Arguments

Name	Type	Description
position	integer	Position you want to seek to.

Return type

No return value

Example

To seek to position 1000 in file object f:

```
f.Seek(1000);
```

SimplifyName(filename*[string]*) [static]

Description

Simplify the name of a file by removing //, /./ and /../

Arguments

Name	Type	Description
filename	string	Filename you want to simplify.

Return type

string filename

Example

To simplify the filename "/data/test//../file.key"

```
var simple = File.SimplifyName("/data/test//../file.key");
```

This simplifies to "/data/file.key"

Size(filename*[string]*) [static]

Description

Get the size of a file

Arguments

Name	Type	Description
filename	string	File you want to find the size of.

Return type

integer

Example

To find the size of file "/data/test"

```
var size = File.Size("/data/test");
```

Write(string*[Any valid javascript type]*)

Description

Write a string to a file opened for writing by a File object

Arguments

Name	Type	Description
string	Any valid javascript type	The string/item that you want to write

Return type

No return value

Example

To write string "Hello, world!" to File object f

```
f.Write("Hello, world!\n");
```

To write the title of model 2 to File object f

```
f.Write("The title of model 2 is " + models[2].title + "\n");
```

Image class

The Image class allows you to create bitmaps in Reporter. More...

The REPORTER JavaScript API provides many class constants, properties and methods. 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

Member functions

- Ellipse(x1*[integer]*, y1*[integer]*, x2*[integer]*, y2*[integer]*)
- Fill(x*[integer]*, y*[integer]*, tol (optional)*[integer]*)
- Line(x1*[integer]*, y1*[integer]*, x2*[integer]*, y2*[integer]*)
- Load(filename*[string]*)
- PixelCount(colour*[string]*, tol (optional)*[integer]*)
- Polygon(x1*[integer]*, y1*[integer]*, x2*[integer]*, y2*[integer]*, ... xn*[integer]*, ... yn*[integer]*)
- Polyline(x1*[integer]*, y1*[integer]*, x2*[integer]*, y2*[integer]*, ... xn*[integer]*, ... yn*[integer]*)
- Rectangle(x1*[integer]*, y1*[integer]*, x2*[integer]*, y2*[integer]*)
- Save(filename*[string]*, filetype*[constant]*)
- Star(x*[integer]*, y*[integer]*, r*[integer]*)
- Text(x*[integer]*, y*[integer]*, text*[string]*)

Image constants

Name	Description
Image.BMP	Save image as BMP
Image.JPG	Save image as JPG
Image.PNG	Save image as PNG

Image properties

Name	Type	Description
antialiasing	bool	Whether or not lines, shapes and text are drawn with antialiasing (true by default).
fillColour	string	Colour to use when filling shapes on the Image. Can be "none", a valid colour from the X colour database (see /etc/X11/rgb.txt) e.g. "Blue", or #RRGGBB (each of R, G and B is a single hex digit) e.g. "#0000FF" for blue.
font	string	Font to use when drawing text on the Image e.g. "Courier". Can be any font accessible by REPORTER.
fontAngle	integer	Angle (degrees) text is drawn at on the Image. Can be between -360 and 360 degrees.
fontColour	string	Colour to use when drawing text on the Image. Can be "none", a valid colour from the X colour database (see /etc/X11/rgb.txt) e.g. "Blue", or #RRGGBB (each of R, G and B is a single hex digit) e.g. "#0000FF" for blue.
fontJustify	constant	Justification to use when drawing text on the Image. Can be Reporter.JUSTIFY_CENTRE, Reporter.JUSTIFY_LEFT or Reporter.JUSTIFY_RIGHT
fontSize	integer	Size of font (in points) to use when drawing text on the Image

fontStyle	constant	Style of font to use when drawing text on the Image. Can be any combination of Reporter.TEXT_NORMAL, Reporter.TEXT_BOLD, Reporter.TEXT_ITALIC and Reporter.TEXT_UNDERLINE
height	integer	Height of the Image
lineCapStyle	constant	Style to use for the end of lines on an Image. Can be Reporter.CAP_FLAT, Reporter.CAP_SQUARE or Reporter.CAP_ROUND
lineColour	string	Colour to use when drawing lines on the Image. Can be "none", a valid colour from the X colour database (see /etc/X11/rgb.txt) e.g. "Blue", or #RRGGBB (each of R, G and B is a single hex digit) e.g. "#0000FF" for blue.
lineJoinStyle	constant	Style to use for the line join at vertices of polygons and polylines on an Image. Can be Reporter.JOIN_MITRE, Reporter.JOIN_BEVEL or Reporter.JOIN_ROUND
lineStyle	constant	Style to use when drawing lines on an Image. Can be Reporter.LINE_NONE, Reporter.LINE_SOLID, Reporter.LINE_DASH, Reporter.LINE_DOT, Reporter.LINE_DASH_DOT or Reporter.LINE_DASH_DOT_DOT
lineWidth	integer	Width to use when drawing lines on an Image value
width	integer	Width of the Image

Detailed Description

The Image class allows you to create, load and save bitmaps. There are various functions available that allow to to draw lines, rectangles, ellipses, text etc on a bitmap.

Constructor

new Image(width (optional)*[integer]*, height (optional)*[integer]*, backgroundColour (optional)*[string]*)

Description

Create a new Image object for creating an image. If no arguments are given a null (0 pixels wide by 0 pixels high) is made. If 2 arguments are given they are used as the width and height of the image. The third argument can be used to define the initial background colour (the default is white).

Arguments

Name	Type	Description
width (optional)	integer	Width of image
height (optional)	integer	Height of image
backgroundColour (optional)	string	Initial background colour for the image (default is white). Can be "none", a valid colour from the X colour database (see /etc/X11/rgb.txt) e.g. "Blue", or #RRGGBB (each of R, G and B is a single hex digit) e.g. "#0000FF" for blue.

Return type

Image object

Example

To create a new image object 100 pixels wide by 50 pixels high

```
var img = new Image(100, 50);
```

Details of functions

Ellipse(x1*[integer]*, y1*[integer]*, x2*[integer]*, y2*[integer]*)

Description

Draw an ellipse on an image

Arguments

Name	Type	Description
x1	integer	X coordinate of start position for ellipse
y1	integer	Y coordinate of start position for ellipse
x2	integer	X coordinate of end position for ellipse
y2	integer	Y coordinate of end position for ellipse

Return type

no return value

Example

To draw an ellipse with no fill and solid red border line width 2 pixels, on image ’idata’, starting at point 30, 20 and finishing at point 100, 50

```
idata.lineColour = "red";
idata.fillColour = "none";
idata.lineWidth = 2;
idata.lineStyle = Reporter.LINE_SOLID;
idata.Ellipse(30, 20, 100, 50);
```

Fill(x*[integer]*, y*[integer]*, tol (optional)*[integer]*)

Description

Fill an area in an image with a colour.

Arguments

Name	Type	Description
x	integer	X coordinate of start position for fill
y	integer	Y coordinate of start position for fill
tol (optional)	integer	Tolerance for colour matching (0-255). Default is 0. When filling a shape if the red, green and blue components are within tol of the colour of pixel (x, y) the pixel will be filled with the current fill colour.

Return type

no return value

Example

To fill an area of image ’idata’, starting at point 30, 20 with red:

```
idata.fillColour = "red";
idata.Fill(30, 20);
```

Line(x1*[integer]*, y1*[integer]*, x2*[integer]*, y2*[integer]*)

Description

Draw a line on an image

Arguments

Name	Type	Description
x1	integer	X coordinate of start position for line
y1	integer	Y coordinate of start position for line
x2	integer	X coordinate of end position for line
y2	integer	Y coordinate of end position for line

Return type

no return value

Example

To draw a blue, dashed line width 2 pixels, on image ’idata’, starting at point 30, 20 and finishing at point 100, 50

```
idata.lineColour = "blue";
idata.lineWidth = 2;
idata.lineStyle = Reporter.LINE_DASH;
idata.Line(30, 20, 100, 50);
```

Load(filename*[string]*)

Description

Load an image file (gif, png, bmp or jpeg)

Arguments

Name	Type	Description
filename	string	Imagename you want to load.

Return type

no return value

Example

To load the image file "/data/test/image.jpg" into the image object ’idata’

```
idata.Load("/data/test/image.jpg");
```

PixelCount(colour*[string]*, tol (optional)*[integer]*)

Description

Count the number of pixels in an image that have a specific colour.

Arguments

Name	Type	Description
colour	string	A valid colour from the X colour database (see /etc/X11/rgb.txt) e.g. "Blue", or #RRGGBB (each of R, G and B is a single hex digit) e.g. "#0000FF" for blue
tol (optional)	integer	Tolerance for colour matching (0-255). Default is 0. When looking at pixels if the red, green and blue components are within tol of the colour of pixel (x, y) the pixel will be counted.

Return type

Number of pixels (integer) with the colour.

Example

To count the number of red pixels in image ’idata’:

```
var nred = idata.PixelCount("red");
```

Polygon(x1*[integer]*, y1*[integer]*, x2*[integer]*, y2*[integer]*, ... xn*[integer]*, ... yn*[integer]*)

Description

Draw a polygon on an image. The last point is always connected back to the first point.

Arguments

Name	Type	Description
x1	integer	X coordinate of point 1
y1	integer	Y coordinate of point 1
x2	integer	X coordinate of point 2
y2	integer	Y coordinate of point 2
... xn	integer	X coordinate of point n
... yn	integer	Y coordinate of point n

Alternatively you can specify a single argument which is an array of coordinates to use.

Return type

no return value

Example

To draw a blue polygon with a solid red border line width 2 pixels, on image 'idata', connecting points (10,10) (20,10) (20,20) (10,20)

```
idata.fillColour = "blue";
idata.lineColour = "red";
idata.lineWidth = 2;
idata.lineStyle = Reporter.LINE_DASH;
idata.Polygon(10,10, 20,10, 20,20, 10,20);
```

or

```
idata.fillColour = "blue";
idata.lineColour = "red";
idata.lineWidth = 2;
idata.lineStyle = Reporter.LINE_DASH;
var a = new Array(10,10, 20,10, 20,20, 10,20);
idata.Polygon(a);
```

Polyline(x1*[integer]*, y1*[integer]*, x2*[integer]*, y2*[integer]*, ... xn*[integer]*, ... yn*[integer]*)

Description

Draw a line with multiple straight segments on an image

Arguments

Name	Type	Description
x1	integer	X coordinate of point 1
y1	integer	Y coordinate of point 1
x2	integer	X coordinate of point 2
y2	integer	Y coordinate of point 2
... xn	integer	X coordinate of point n
... yn	integer	Y coordinate of point n

Alternatively you can specify a single argument which is an array of coordinates to use.

Return type

no return value

Example

To draw a blue, dashed polyline width 2 pixels, on image 'idata', connecting points (10,10) (20,10) (20,20) (10,20)

```
idata.lineColour = "blue";
idata.lineWidth = 2;
idata.lineStyle = Reporter.LINE_DASH;
idata.Polyline(10,10, 20,10, 20,20, 10,20);
```

or

```
idata.lineColour = "blue";
idata.lineWidth = 2;
idata.lineStyle = Reporter.LINE_DASH;
var a = new Array(10,10, 20,10, 20,20, 10,20);
idata.Polyline(a);
```

Rectangle(x1*[integer]*, y1*[integer]*, x2*[integer]*, y2*[integer]*)

Description

Draw a rectangle on an image

Arguments

Name	Type	Description
x1	integer	X coordinate of start position for rectangle
y1	integer	Y coordinate of start position for rectangle
x2	integer	X coordinate of end position for rectangle
y2	integer	Y coordinate of end position for rectangle

Return type

no return value

Example

To draw a rectangle with no fill and solid red border line width 2 pixels, on image ’idata’, starting at point 30, 20 and finishing at point 100, 50

```
idata.lineColour = "red";
idata.fillColour = "none";
idata.lineWidth = 2;
idata.lineStyle = Reporter.LINE_SOLID;
idata.Rectangle(30, 20, 100, 50);
```

Save(filename*[string]*, filetype*[constant]*)

Description

Save an image to file (gif, png, bmp or jpeg)

Arguments

Name	Type	Description
filename	string	Imagename you want to save.
filetype	constant	Type you want to save as. Can be: Image.BMP, Image.JPG or Image.PNG

Return type

no return value

Example

To save the image object ’idata’ to file "/data/test/image.jpg" as a jpeg

```
idata.Save("/data/test/image.jpg", Image.JPG);
```

Star(x*[integer]*, y*[integer]*, r*[integer]*)

Description

Draw a star on an image

Arguments

Name	Type	Description
x	integer	X coordinate of centre of star
y	integer	Y coordinate of centre of star
r	integer	Radius of star

Return type

no return value

Example

To draw a blue star with yellow fill, on image ’idata’, centred at point 30, 20 with radius 10

```
idata.lineColour = "blue";
idata.fillColour = "yellow";
idata.Star(30, 20, 10);
```

Text(x*[integer]*, y*[integer]*, text*[string]*)

Description

Draw text on an image

Arguments

Name	Type	Description
x	integer	X position for text
y	integer	Y position for text
text	string	Text to write on image

Return type

no return value

Example

To write the text ’Test’ in Helvetica 12pt bold underlined, coloured red on image ’idata’, at point 30, 20

```
idata.fontColour = "red";
idata.fontSize = 12;
idata.fontStyle = Reporter.TEXT_BOLD | Reporter.TEXT_UNDERLINE;
idata.Text(30, 20, "Test");
```

Include class

The Include class allows you to access the include files in a model. More...

The REPORTER JavaScript API provides many class constants, properties and methods. 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

Include constants

Constants for Directory separators

Name	Description
Include.NATIVE	Use directory separators native to this machine when writing directory names.
Include.UNIX	Use unix directory separators when writing directory names.
Include.WINDOWS	Use windows directory separators when writing directory names.

Detailed Description

Originally developed for use in PRIMER, the Include class allows a user to create and query include files in a model. A stripped-back version of this class has been added to T/HIS, D3PLOT, and REPORTER for consistency between the programs. See File.DriveMapFilename for the current use of this class in REPORTER.

Item class

The Item class gives access to items in Reporter. More...

The REPORTER JavaScript API provides many class constants, properties and methods. 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

Class functions

- GetAll(page*[Page]*)
- GetFromName(page*[Page]*, name*[string]*)

Member functions

- DeleteColumn(column*[integer]*)
- DeleteRow(row*[integer]*)
- Generate()
- GetCellProperties(row*[integer]*, column*[integer]*)
- GetColumnProperties(column*[integer]*, header*[constant]*)
- GetColumnWidth(row*[integer]*)
- GetCondition(index*[integer]*)
- GetCondition(index*[integer]*, column*[integer]*)
- GetCondition(index*[integer]*, row*[integer]*, column*[integer]*)
- GetGeneratedData(row_index*[integer]*, column_index*[integer]*)
- GetRowHeight(row*[integer]*)
- InsertColumn(column*[integer]*)
- InsertRow(row*[integer]*)
- MergeCells(topLeftRow*[integer]*, topLeftColumn*[integer]*, rows*[integer]*, columns*[integer]*)
- SetCellProperties(properties*[object]*, row*[integer]*, column*[integer]*)
- SetColumnProperties(properties*[object]*, column*[integer]*, header*[constant]*)
- SetColumnWidth(column*[integer]*, width*[real]*)
- SetCondition(condition*[integer]*, properties*[object]*)
- SetCondition(condition*[integer]*, column*[integer]*, properties*[object]*)
- SetCondition(condition*[integer]*, row*[integer]*, column*[integer]*, properties*[object]*)
- SetRowHeight(row*[integer]*, height*[real]*)
- UnmergeCells(row*[integer]*, column*[integer]*)

Item constants

Name	Description
Item.ARROW	Arrow item
Item.AUTO_TABLE	Automatic table item
Item.D3PLOT	D3Plot item
Item.ELLIPSE	Ellipse item
Item.IMAGE	Image item
Item.IMAGE_FILE	Image file item
Item.LIBRARY_IMAGE	Library image item
Item.LIBRARY_PROGRAM	Library program item
Item.LINE	Line item

Item.NOTE	Note item
Item.PLACEHOLDER	Placeholder item
Item.PRIMER	Primer item
Item.PROGRAM	Program item
Item.RECTANGLE	Rectangle item
Item.SCRIPT	Script item
Item.SCRIPT_FILE	Script File item
Item.TABLE	Table item
Item.TEXT	Text item
Item.TEXTBOX	Textbox item
Item.TEXT_FILE	Text file item
Item.THIS	T/HIS item

Item properties

Name	Type	Description
active	logical	If item is active or not. Inactive items will be skipped during report/page/item generation.
autotableType	constant	Autotable type (whether the data is sourced from a file or a directory). Can be Reporter.AUTO_TABLE_DIRECTORY or Reporter.AUTO_TABLE_FILE. Valid for item type Item.AUTO_TABLE.
bottomCrop	integer	Bottom cropping value. Valid for item types Item.IMAGE, Item.IMAGE_FILE, Item.D3PLOT, Item.PRIMER and Item.THIS.
bottomMargin	real	Bottom margin width. Valid for item types Item.TEXTBOX, Item.TEXT_FILE, Item.TABLE and Item.AUTO_TABLE
columns (readonly)	integer	The number of columns in the table. Valid for item types Item.TABLE and Item.AUTO_TABLE
conditions (readonly)	integer	The number of conditions assigned to the item. Valid for item types Item.PROGRAM, Item.TEXT_FILE, Item.TEXT and Item.TEXTBOX
embed	logical	If image is embedded or not. Valid for item types Item.IMAGE
file	string	File or directory for item. Valid for item types: Item.AUTO_TABLE Item.D3PLOT Item.IMAGE Item.IMAGE_FILE Item.PRIMER Item.PROGRAM Item.SCRIPT_FILE Item.TEXT_FILE Item.THIS
filetype (read only)	string	Output file type. Read-only but can be updated by changing the file extension on the item property "file". Valid for item types Item.D3PLOT Item.PRIMER and Item.THIS.
fillColour	Colour object	Colour of fill for the item. Valid for item types Item.RECTANGLE, Item.ELLIPSE, Item.TEXTBOX, Item.PROGRAM and Item.TEXT_FILE
fontName	string	Font for the item e.g. "Courier". Can be any font accessible by REPORTER. Valid for item types Item.TEXT, Item.TEXTBOX, Item.PROGRAM and Item.TEXT_FILE

fontSize	integer	Font size for the item (6 <= fontSize <= 72). Valid for item types Item.TEXT, Item.TEXTBOX, Item.PROGRAM and Item.TEXT_FILE
fontStyle	constant	Font style for the item. Can be a combination of Reporter.TEXT_NORMAL, Reporter.TEXT_BOLD, Reporter.TEXT_ITALIC or Reporter.TEXT_UNDERLINE Valid for item types Item.TEXT, Item.TEXTBOX, Item.PROGRAM and Item.TEXT_FILE
generatedRowHeight	real	The height of each generated row in an Autotable. Valid for item type Item.AUTO_TABLE.
headerHeight	real	The height of the header in an Autotable. Valid for item type Item.AUTO_TABLE.
height	real	Height for "rectangular" items (absolute difference between y and y2)
job	string	Input job file. Valid for item types Item.D3PLOT and Item.THIS.
justify	constant	Text justification for the item. Can be Reporter.JUSTIFY_CENTRE, Reporter.JUSTIFY_LEFT or Reporter.JUSTIFY_RIGHT combined with Reporter.JUSTIFY_TOP, Reporter.JUSTIFY_MIDDLE or Reporter.JUSTIFY_BOTTOM Valid for item types Item.TEXT, Item.TEXTBOX, Item.PROGRAM and Item.TEXT_FILE
leftCrop	integer	Left cropping value. Valid for item types Item.IMAGE, Item.IMAGE_FILE, Item.D3PLOT, Item.PRIMER and Item.THIS.
leftMargin	real	Left margin width. Valid for item types Item.TEXTBOX, Item.TEXT_FILE, Item.TABLE and Item.AUTO_TABLE
lineColour	Colour object	Colour of outline for the item. Valid for item types Item.LINE, Item.ARROW, Item.RECTANGLE, Item.ELLIPSE, Item.TEXTBOX, Item.D3PLOT, Item.PRIMER, Item.THIS, Item.PROGRAM, Item.TEXT_FILE, Item.IMAGE_FILE, Item.TABLE and Item.AUTO_TABLE.
lineStyle	constant	Style of outline for the item. Can be Reporter.LINE_NONE, Reporter.LINE_SOLID, Reporter.LINE_DASH, Reporter.LINE_DOT, Reporter.LINE_DASH_DOT or Reporter.LINE_DASH_DOT_DOT Valid for item types Item.LINE, Item.ARROW, Item.RECTANGLE, Item.ELLIPSE, Item.TEXTBOX, Item.D3PLOT, Item.PRIMER, Item.THIS, Item.PROGRAM, Item.TEXT_FILE and Item.IMAGE_FILE.
lineWidth	real	Width of outline for the item in mm. Valid for item types Item.LINE, Item.ARROW, Item.RECTANGLE, Item.ELLIPSE, Item.TEXTBOX, Item.D3PLOT, Item.PRIMER, Item.THIS, Item.PROGRAM, Item.TEXT_FILE, Item.IMAGE_FILE, Item.TABLE and Item.AUTO_TABLE
name	string	Name of the Item
resolution	integer	Image resolution. Larger values yield a smaller image representation on screen. Valid for item type Item.IMAGE.
rightCrop	integer	Right cropping value. Valid for item types Item.IMAGE, Item.IMAGE_FILE, Item.D3PLOT, Item.PRIMER and Item.THIS.
rightMargin	real	Right margin width. Valid for item types Item.TEXTBOX, Item.TEXT_FILE, Item.TABLE and Item.AUTO_TABLE
rows (readonly)	integer	The number of rows in the table. Valid for item type Item.TABLE
saveCSV	bool	Whether or not a CSV file of the table contents is written when the item is generated. Valid for item types Item.TABLE and Item.AUTO_TABLE
saveCSVFilename	string	The path and filename of the CSV file written when the item is generated. Valid for item types Item.TABLE and Item.AUTO_TABLE

saveXlsx	bool	Whether or not a Excel file of the table contents is written when the item is generated. Valid for item types Item.TABLE and Item.AUTO_TABLE
saveXlsxFilename	string	The path and filename of the Excel file written when the item is generated. Valid for item types Item.TABLE and Item.AUTO_TABLE
script	string	The script source text for the item. Only valid for item type Item.SCRIPT. For Item.SCRIPT_FILE, use the *file* property.
text	string	The text for the item. Valid for item types Item.TEXT, Item.TEXTBOX, Item.PROGRAM, Item.TEXT_FILE and Item.SCRIPT
textColour	Colour object	Colour of text for the item. Valid for item types Item.TEXT, Item.TEXTBOX, Item.PROGRAM and Item.TEXT_FILE
topCrop	integer	Top cropping value. Valid for item types Item.IMAGE, Item.IMAGE_FILE, Item.D3PLOT, Item.PRIMER and Item.THIS.
topMargin	real	Top margin width. Valid for item types Item.TEXTBOX, Item.TEXT_FILE, Item.TABLE and Item.AUTO_TABLE
type (read only)	constant	type of the Item. Can be Item.LINE, Item.TEXT etc.
width	real	Width for "rectangular" items (absolute difference between x and x2)
x	real	X coordinate
x2	real	Second X coordinate for "rectangular" items
y	real	Y coordinate
y2	real	Second Y coordinate for "rectangular" items

Detailed Description

The Item class allows you to access the items in templates that Reporter currently has open.

Constructor

new Item(page*[Page]*, type*[constant]*, name (optional)*[string]*, x (optional)*[real]*, x2 (optional)*[real]*, y (optional)*[real]*, y2 (optional)*[real]*)

Description

Create a new Item. The name and coordinates arguments are optional. Item.TABLE items are constructed with two rows and two columns by default. If you require only one row or column, use DeleteRow and DeleteColumn.

Arguments

Name	Type	Description
page	Page	Page to create item in
type	constant	Item type. Can be Item.LINE, Item.ARROW, Item.RECTANGLE, Item.ELLIPSE, Item.TEXT, Item.TEXTBOX, Item.IMAGE, Item.PROGRAM, Item.D3PLOT, Item.PRIMER, Item.THIS, Item.TEXT_FILE, Item.IMAGE_FILE, Item.LIBRARY_IMAGE, Item.LIBRARY_PROGRAM, Item.TABLE, Item.AUTO_TABLE, Item.SCRIPT, Item.SCRIPT_FILE, Item.NOTE or Item.PLACEHOLDER.
name (optional)	string	Name of item
x (optional)	real	X coordinate

x2 (optional)	real	Second X coordinate for "rectangular" items
y (optional)	real	Y coordinate
y2 (optional)	real	Second Y coordinate for "rectangular" items

Return type

Item object

Example

To create a new blank Item object:

```
var i = new Item();
```

Details of functions

DeleteColumn(column*[integer]*)

Description

Delete a column from a table. Valid for item type Item.TABLE and Item.AUTO_TABLE.

Arguments

Name	Type	Description
column	integer	The index of the column to delete. Note that indices start from 0.

Return type

No return value

Example

To delete the second column from table item i:

```
i.DeleteColumn(1);
```

DeleteRow(row*[integer]*)

Description

Delete a row from a table. Valid for item type Item.TABLE.

Arguments

Name	Type	Description
row	integer	The index of the row to delete. Note that indices start from 0.

Return type

No return value

Example

To delete the second row from table item i:
```
i.DeleteRow(1);
```

Generate()

Description

Generate an item.

Arguments

No arguments

Return type

No return value

Example

To generate item i:
```
i.Generate();
```

GetAll(page*[Page]*) [static]

Description

Get all of the items on a page.

Arguments

Name	Type	Description
page	Page	Page to get items from.

Return type

Array of Item objects

Example

To get all of the items on page p:
```
var items = Item.GetAll(p);
```

GetCellProperties(row*[integer]*, column*[integer]*)

Description

Get the properties of the specified cell. Valid for item type Item.TABLE and Item.AUTO_TABLE.

Arguments

Name	Type	Description

row	integer	The row index of the cell of interest. Note that indices start from 0.
column	integer	The column index of the cell of interest. Note that indices start from 0.

Return type

Object with the following properties:

Name	Type	Description
bottomBorderWidth	real	Cell bottom border width. Can be 0.0, 0.1, 0.5, 0.75, 1.0, 1.5, 2.0, 2.5, 3.0, 4.0, 5.0. Other values will result in no border.
colspan	integer	Number of columns this cell spans (for merged cells). 1 if not merged. Use columnMergeOrigin to find top-left cell.
column (read only)	integer	The column index
columnMergeOrigin	integer	The column index of the top-left cell in this merge cell group (if cell not merged then == column).
conditions	integer	Number of conditions assigned to this cell.
fillColour	Colour object	Fill colour
fontName	string	Font name (e.g. "Courier").
fontSize	integer	Font size (between 6 and 72).
fontStyle	integer	Font style. See Text style constants for details.
height	real	Cell height. Modifying this property will modify the height of all cells in the row.
hyperlinkHTML	string	Hyperlink destination for HTML.
hyperlinkPDF	string	Hyperlink destination for PDF.
hyperlinkReport	string	Hyperlink destination for Report or page within Report.
justify	integer	Text justification for the item. Same rules as justify property of Item Class.
output	string	The output text from a Program or Library Program cell.
prefix	string	Prefix text to appear before Library Program output.
program	string	Path and filename for a Program cell, or the filename (e.g. *title.js*) for a Library Program cell.
programArgs	Array of strings	Program arguments
rightBorderWidth	real	Cell right border width. Can be 0.0, 0.1, 0.5, 0.75, 1.0, 1.5, 2.0, 2.5, 3.0, 4.0, 5.0. Other values will result in no border.
row (read only)	integer	The cell row index.
rowMergeOrigin	integer	The row index of the top-left cell in this merge cell group (if cell not merged then == row).
rowSpan	integer	Number of rows this cell spans (for merged cells). == 1 if not merged. Use rowMergeOrigin to find top-left cell.
suffix	string	Suffix text to appear after Library Program output.
text	string	The cell text. For Program and Library Program cells, use the **prefix**, **output** and **suffix** properties.
textColour	Colour object	Colour of text
topBorderWidth	real	Cell top border width. Can be 0.0, 0.1, 0.5, 0.75, 1.0, 1.5, 2.0, 2.5, 3.0, 4.0, 5.0. Other values will result in no border.
type	integer	Can be Item.TEXT,Item.LIBRARY_PROGRAM or Item.PROGRAM.

variable	string	REPORTER variable for library program output.
width	real	Cell width. Modifying this property will modify the width of all cells in the column.

Example

To get the properties of the top-left cell in a table:

```
i.GetCellProperties(0, 0);
```

GetColumnProperties(column*[integer]*, header*[constant]*)

Description

Get an autotable column properties. Valid for item type Item.AUTO_TABLE.

Arguments

Name	Type	Description
column	integer	The index of the column of interest. Note that indices start from 0.
header	constant	An argument to signify to get the properties of the header or the generated rows. Can be Reporter.AUTO_TABLE_HEADER or Reporter.AUTO_TABLE_ROWS.

Return type

Object with the following properties:

Name	Type	Description
conditions	integer	Number of conditions assigned to this cell.
fillColour	Colour object	Fill colour
fontName	string	Font name (e.g. "Courier").
fontSize	integer	Font size (between 6 and 72).
fontStyle	integer	Font style. Same rules as fontStyle property of
hyperlinkHTML	string	Hyperlink destination for HTML.
hyperlinkPDF	string	Hyperlink destination for PDF.
hyperlinkReport	string	Hyperlink destination for Report or page within Report.
justify	integer	Text justification for the item. Same rules as justify property of Item Class.
program	string	Path and filename for a Program cell, or the filename (e.g. *title.js*) for a Library Program cell.
programArgs	Array of strings	Program arguments
text	string	The cell text. For Program and Library Program cells, use the **prefix**, **output** and **suffix** properties.
textColour	Colour object	Colour of text
type	integer	Can be Item.TEXT,Item.LIBRARY_PROGRAM or Item.PROGRAM.
width	real	Cell width. Modifying this property will modify the width of all cells in the column.

Example

Returns the column properties of the header of the first column:
```
i.GetColumnProperties(0, Reporter.AUTO_TABLE_HEADER);
```

GetColumnWidth(row*[integer]*)

Description

Get the width of a table column. Valid for item types Item.TABLE or Item.AUTO_TABLE.

Arguments

Name	Type	Description
row	integer	The index of the column of interest. Note that indices start from 0.

Return type

Integer. The width of the specified column.

Example

To get the width of the first column in a table:
```
i.GetColumnWidth(0);
```

GetCondition(index*[integer]*)

Description

Get the conditional formatting data for an item. Valid for item types Item.TEXT_FILE, Item.PROGRAM, Item.TEXT or Item.TEXTBOX (for Item.AUTO_TABLE and Item.TABLE, see GetCondition functions with additional arguments below).

Arguments

Name	Type	Description
index	integer	The index of the condition to get. Note that indices start from 0. See conditions for the total number of comditions

Return type

Object with the following properties:

Name	Type	Description
fillColour	Colour object	Fill colour
fontName	string	Font name (e.g. "Courier").
fontSize	integer	Font size (between 6 and 72).
fontStyle	integer	Font style. See Text style constants for details.
justify	integer	Text alignment for the item. See Justification constants for details.
name	string	Condition name
textColour	Colour object	Colour of text

type	integer	See Condition types constants for details.
value	string	First condition value
value2	string	Second condition value (where relevant)

Example

To get the data for the 2nd condition in item i:

```
var condition = i.GetCondition(1);
```

GetCondition(index*[integer]*, column*[integer]*)

Description

Get the conditional formatting data for an Item.AUTO_TABLE item.

Arguments

Name	Type	Description
index	integer	The index of the condition to get. Note that indices start from 0.
column	integer	The column to get the condition from. Note that indices start from 0.

Return type

Object with the following properties:

Name	Type	Description
fillColour	Colour object	Fill colour
fontName	string	Font name (e.g. "Courier").
fontSize	integer	Font size (between 6 and 72).
fontStyle	integer	Font style. See Text style constants for details.
justify	integer	Text alignment for the item. See Justification constants for details.
name	string	Condition name
textColour	Colour object	Colour of text
type	integer	See Condition types constants for details.
value	string	First condition value
value2	string	Second condition value (where relevant)

Example

To get the data for the 2nd condition from the 3rd column in autotable item i:

```
var condition = i.GetCondition(1, 2);
```

GetCondition(index*[integer]*, row*[integer]*, column*[integer]*)

Description

Get the conditional formatting data for an Item.TABLE item.

Arguments

Name	Type	Description
index	integer	The index of the condition to get. Note that indices start from 0.
row	integer	The cell row to get the condition from. Note that indices start from 0.
column	integer	The cell column to get the condition from. Note that indices start from 0.

Return type

Object with the following properties:

Name	Type	Description
fillColour	Colour object	Fill colour
fontName	string	Font name (e.g. "Courier").
fontSize	integer	Font size (between 6 and 72).
fontStyle	integer	Font style. See Text style constants for details.
justify	integer	Text alignment for the item. See Justification constants for details.
name	string	Condition name
textColour	Colour object	Colour of text
type	integer	See Condition types constants for details.
value	string	First condition value
value2	string	Second condition value (where relevant)

Example

To get the data for the 2nd condition from the 4th row, 3rd column in table item i:

```
var condition = i.GetCondition(1, 3, 2);
```

GetFromName(page*[Page]*, name*[string]*) [static]

Description

Get an Item from its name.

Arguments

Name	Type	Description
page	Page	Page to get item from
name	string	Item name

Return type

Item object (or null if item cannot be found)

Example

To get the item with name test on page p:

```
var item = Item.GetFromName(p, "test");
```

GetGeneratedData(row_index*[integer]*, column_index*[integer]*)

Description

Get the text that appears in an autotable cell once generated. Valid for item type Item.AUTO_TABLE.

Arguments

Name	Type	Description
row_index	integer	The index of the row of interest. Note that indices start from 0.
column_index	integer	The index of the column of interest. Note that indicies start from 0.

Return type

String: the text displayed in the specified row and column.

Example

Get the data from the first cell in the first row and column in an autotable.

```
i.GetGeneratedData(0, 0);
```

GetRowHeight(row*[integer]*)

Description

Get the height of a table row. Valid for item type Item.TABLE.

Arguments

Name	Type	Description
row	integer	The index of the row of interest. Note that indices start from 0.

Return type

integer

Example

To get the height of the first row in a table:

```
i.GetRowHeight(0);
```

InsertColumn(column*[integer]*)

Description

Insert a column into a table. Valid for item types Item.TABLE and Item.AUTO_TABLE.

Arguments

Name	Type	Description
column	integer	The index of the position where the inserted column will end up. Note that indices start from 0. If no argument is given, a column will be added to the bottom of the table.

Return type

No return value

Example

To insert a column that will become the second column from the left of the table:

```
i.InsertColumn(1);
```

InsertRow(row*[integer]*)

Description

Insert a row into a table. Valid for item type Item.TABLE.

Arguments

Name	Type	Description
row	integer	The index of the position where the inserted row will end up. Note that indices start from 0. If no argument is given, a row will be added to the bottom of the table.

Return type

No return value

Example

To insert a row that will become the second row from the top of the table:

```
i.InsertRow(1);
```

MergeCells(topLeftRow*[integer]*, topLeftColumn*[integer]*, rows*[integer]*, columns*[integer]*)

Description

Merge specified cells in a table. Valid for item types Item.TABLE and Item.AUTO_TABLE.

Arguments

Name	Type	Description
topLeftRow	integer	The row index of the top-left cell in the group of cells to be merged. Note that indices start from 0.
topLeftColumn	integer	The column index of the top-left cell in the group of cells to be merged. Note that indices start from 0.
rows	integer	The number of rows of cells to be merged (measured from the topLeftRow position).
columns	integer	The number of columns of cells to be merged (measured from the topLeftColumn position).

Return type

No return value

Example

To merge the cells in first row and the first two columns in the table:

```
i.MergeCells(0, 0, 1, 2);
```

SetCellProperties(properties*[object]*, row*[integer]*, column*[integer]*)

Description

Set the properties of the specified cell. Valid for item type Item.TABLE.

Arguments

Name	Type	Description

<table>
<tr><td>properties</td><td>object</td><td>
<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>bottomBorderWidth (optional)</td><td>real</td><td>Cell bottom border width. Can be 0.0, 0.1, 0.5, 0.75, 1.0, 1.5, 2.0, 2.5, 3.0, 4.0, 5.0. Other values will result in no border.</td></tr>
<tr><td>fillColour (optional)</td><td>Colour object</td><td>Fill colour</td></tr>
<tr><td>fontName (optional)</td><td>string</td><td>Font name (e.g. "Courier").</td></tr>
<tr><td>fontSize (optional)</td><td>integer</td><td>Font size (between 6 and 72).</td></tr>
<tr><td>fontStyle (optional)</td><td>integer</td><td>Font style. See Text style constants for details.</td></tr>
<tr><td>hyperlinkHTML (optional)</td><td>string</td><td>Hyperlink destination for HTML.</td></tr>
<tr><td>hyperlinkPDF (optional)</td><td>string</td><td>Hyperlink destination for PDF.</td></tr>
<tr><td>hyperlinkReport (optional)</td><td>string</td><td>Hyperlink destination for Report or page within Report.</td></tr>
<tr><td>justify (optional)</td><td>integer</td><td>Text justification for the item. Same rules as justify property of Item Class.</td></tr>
<tr><td>prefix (optional)</td><td>string</td><td>Prefix text to appear before Library Program output.</td></tr>
<tr><td>program (optional)</td><td>string</td><td>Path and filename for a Program cell, or the filename (e.g. title.js) for a Library Program cell.</td></tr>
<tr><td>programArgs (optional)</td><td>Array of strings</td><td>Program arguments</td></tr>
<tr><td>rightBorderWidth (optional)</td><td>real</td><td>Cell right border width. Can be 0.0, 0.1, 0.5, 0.75, 1.0, 1.5, 2.0, 2.5, 3.0, 4.0, 5.0. Other values will result in no border.</td></tr>
<tr><td>suffix (optional)</td><td>string</td><td>Suffix text to appear after Library Program output.</td></tr>
<tr><td>text (optional)</td><td>string</td><td>The cell text. For Program and Library Program cells, use the prefix, output and suffix properties.</td></tr>
<tr><td>textColour (optional)</td><td>Colour object</td><td>Colour of text</td></tr>
<tr><td>topBorderWidth (optional)</td><td>real</td><td>Cell top border width. Can be 0.0, 0.1, 0.5, 0.75, 1.0, 1.5, 2.0, 2.5, 3.0, 4.0, 5.0. Other values will result in no border.</td></tr>
<tr><td>type (optional)</td><td>integer</td><td>Can be Item.TEXT,Item.LIBRARY_PROGRAM or Item.PROGRAM.</td></tr>
<tr><td>variable (optional)</td><td>string</td><td>REPORTER variable for library program output.</td></tr>
</table>
An object containing the cell properties. Object has the following properties:
</td></tr>
<tr><td>row</td><td>integer</td><td>The row index of the cell to be modified. Note that indices start from 0.</td></tr>
<tr><td>column</td><td>integer</td><td>The column index of the cell to be modified. Note that indices start from 0.</td></tr>
</table>

Return type

No return value

Example

To set the properties of the cell object to those of the object *cell_obj*:

```
i.SetCellProperties(cell_obj, 0, 0);
```

SetColumnProperties(properties*[object]*, column*[integer]*, header*[constant]*)

Description

Set the properties of an autotable column. Valid for item type Item.AUTO_TABLE.

Arguments

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>properties</td><td>object</td><td>Set the properties of an autotable column. Valid for item type Item.AUTO_TABLE. Object has
<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>fillColour (optional)</td><td>Colour object</td><td>Fill colour</td></tr>
<tr><td>fontName (optional)</td><td>string</td><td>Font name (e.g. "Courier").</td></tr>
<tr><td>fontSize (optional)</td><td>integer</td><td>Font size (between 6 and 72).</td></tr>
<tr><td>fontStyle (optional)</td><td>integer</td><td>Font style. Same rules as fontStyle property of</td></tr>
<tr><td>hyperlinkHTML (optional)</td><td>string</td><td>Hyperlink destination for HTML.</td></tr>
<tr><td>hyperlinkPDF (optional)</td><td>string</td><td>Hyperlink destination for PDF.</td></tr>
<tr><td>hyperlinkReport (optional)</td><td>string</td><td>Hyperlink destination for Report or page within Report.</td></tr>
<tr><td>justify (optional)</td><td>integer</td><td>Text justification for the item. Same rules as justify property of Item Class.</td></tr>
<tr><td>program (optional)</td><td>string</td><td>Path and filename for a Program cell, or the filename (e.g. title.js) for a Library Program cell.</td></tr>
<tr><td>programArgs (optional)</td><td>Array of strings</td><td>Program arguments</td></tr>
<tr><td>text (optional)</td><td>string</td><td>The cell text. For Program and Library Program cells, use the prefix, output and suffix properties.</td></tr>
<tr><td>textColour (optional)</td><td>Colour object</td><td>Colour of text</td></tr>
<tr><td>type (optional)</td><td>integer</td><td>Can be Item.TEXT,Item.LIBRARY_PROGRAM or Item.PROGRAM.</td></tr>
</table>
the following properties:</td></tr>
<tr><td>column</td><td>integer</td><td>The index of the column of interest. Note that indices start from 0.</td></tr>
<tr><td>header</td><td>constant</td><td>An argument to signify to set the properties of the header or the generated rows. Can be Reporter.AUTO_TABLE_HEADER or Reporter.AUTO_TABLE_ROWS.</td></tr>
</table>

Return type

No return value

Example

Sets the column properties of the header of the first column with the properties of the object *column_obj*.

```
i.SetColumnProperties(column_obj, 0, Reporter.AUTO_TABLE_HEADER);
```

SetColumnWidth(column*[integer]*, width*[real]*)

Description

Set the width of a table column. Valid for item type Item.TABLE.

Arguments

Name	Type	Description
column	integer	The index of the column of interest. Note that indices start from 0.
width	real	The column width.

Return type

No return value

Example

To set the width of the first column in a table to 10.0:

```
i.SetColumnWidth(0, 10.0);
```

SetCondition(condition*[integer]*, properties*[object]*)

Description

Set the specified condition for an item. Valid for item types Item.TEXT_FILE, Item.PROGRAM, Item.TEXT or Item.TEXTBOX (for Item.AUTO_TABLE and Item.TABLE, see SetCondition functions with additional arguments below).

Arguments

Name	Type	Description
condition	integer	The index of the condition you wish to set. Note that indices start at 0. If a condition already exists at the specified index, it will be replaced. To add a new condition, specify an index equal to the number of existing conditions.

properties	object	The index of the condition you wish to set. Note that indices start at 0. If a condition already exists at the specified index, it will be replaced. To add a new condition, specify an index equal to the number of existing conditions. Object has the following properties:

Name	Type	Description
fillColour (optional)	Colour object	Fill colour
fontName (optional)	string	Font name (e.g. "Courier").
fontSize (optional)	integer	Font size (between 6 and 72).
fontStyle (optional)	integer	Font style. See Text style constants for details.
justify (optional)	integer	Text alignment for the item. See Justification constants for details.
name (optional)	string	Condition name
textColour (optional)	Colour object	Colour of text
type (optional)	integer	See Condition types constants for details.
value (optional)	string	First condition value
value2 (optional)	string	Second condition value (where relevant)

Return type

No return value

Example

To set the conditions for the condition index 1 in item i to those of the object obj:

```
var obj = { name:"example", type:Reporter.CONDITION_EQUAL_TO, value:"Test",
textColour:Colour.Red() };
i.SetCondtion(1, obj);
```

SetCondition(condition*[integer]*, column*[integer]*, properties*[object]*)

Description

Set the specified condition for an Item.AUTO_TABLE item.

Arguments

Name	Type	Description
condition	integer	The index of the condition you wish to set. Note that indices start at 0. If a condition already exists at the specified index, it will be replaced. To add a new condition, specify an index equal to the number of existing conditions.
column	integer	The column to set the condition for. Note that indices start from 0.

properties	object	The column to set the condition for. Note that indices start from 0. Object has the following properties:

Name	Type	Description
fillColour (optional)	Colour object	Fill colour
fontName (optional)	string	Font name (e.g. "Courier").
fontSize (optional)	integer	Font size (between 6 and 72).
fontStyle (optional)	integer	Font style. See Text style constants for details.
justify (optional)	integer	Text alignment for the item. See Justification constants for details.
name (optional)	string	Condition name
textColour (optional)	Colour object	Colour of text
type (optional)	integer	See Condition types constants for details.
value (optional)	string	First condition value
value2 (optional)	string	Second condition value (where relevant)

Return type

No return value

Example

To set the conditions for condition index 1 in the third column in item i to those of the object obj:

```
var obj = { name:"example", type:Reporter.CONDITION_EQUAL_TO, value:"Test",
textColour:Colour.Red() };
i.SetCondtion(1, 2, obj);
```

SetCondition(condition*[integer]*, row*[integer]*, column*[integer]*, properties*[object]*)

Description

Set the specified condition for an Item.TABLE item.

Arguments

Name	Type	Description
condition	integer	The index of the condition you wish to set. Note that indices start at 0. If a condition already exists at the specified index, it will be replaced. To add a new condition, specify an index equal to the number of existing conditions.
row	integer	The row to set the condition for. Note that indices start from 0.
column	integer	The column to set the condition for. Note that indices start from 0.

properties	object	The column to set the condition for. Note that indices start from 0. Object has the following properties:

Name	Type	Description
fillColour (optional)	Colour object	Fill colour
fontName (optional)	string	Font name (e.g. "Courier").
fontSize (optional)	integer	Font size (between 6 and 72).
fontStyle (optional)	integer	Font style. See Text style constants for details.
justify (optional)	integer	Text alignment for the item. See Justification constants for details.
name (optional)	string	Condition name
textColour (optional)	Colour object	Colour of text
type (optional)	integer	See Condition types constants for details.
value (optional)	string	First condition value
value2 (optional)	string	Second condition value (where relevant)

Return type

No return value

Example

To set the conditions for condition index 1 in the fourth row, third column in item i to those of the object obj:

```
var obj = { name:"example", type:Reporter.CONDITION_EQUAL_TO, value:"Test",
textColour:Colour.Red() };
i.SetCondtion(1, 3, 2, obj);
```

SetRowHeight(row*[integer]*, height*[real]*)

Description

Set the height of a table row. Valid for item type Item.TABLE and Item.AUTO_TABLE.

Arguments

Name	Type	Description
row	integer	The index of the row of interest. Note that indices start from 0.
height	real	The row height.

Return type

No return value

Example

To set the height of the first row in a table to 10.0:

```
i.SetRowHeight(0, 10.0);
```

UnmergeCells(row*[integer]*, column*[integer]*)

Description

Unmerge the specified cell in a table. All cells merged to the specified cell will be unmerged. Valid for item types Item.TABLE and Item.AUTO_TABLE.

Arguments

Name	Type	Description
row	integer	The row index of the cell to be unmerged. Note that indices start from 0.
column	integer	The column index of the cell to be unmerged. Note that indices start from 0..

Return type

No return value

Example

To unmerge the top-left cell in a table:

```
i.UnmergeCells(0, 0);
```

Options class

The Options class enables you to access several options in REPORTER. More...

The REPORTER JavaScript API provides many class constants, properties and methods. 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

Options constants

Constants for Promises

Name	Description
Options.RUN_PROMISE_CONSTRUCTOR	Allow/run promises when an API constructor is called
Options.RUN_PROMISE_METHOD	Allow/run promises when an API method is called
Options.RUN_PROMISE_PROPERTY	Allow/run promises when an API property getter/setter is done
Options.RUN_PROMISE_SCRIPT	Allow/run promises when a script is run
Options.RUN_PROMISE_WINDOW_LOOP	Allow/run promises in a window event loop

Options class properties

Name	Type	Description
run_promises	constant	When any promise callbacks/handlers are allowed to run. Can be a bitwise OR of: Options.RUN_PROMISE_WINDOW_LOOP, Options.RUN_PROMISE_CONSTRUCTOR, Options.RUN_PROMISE_METHOD and Options.RUN_PROMISE_PROPERTY Options.RUN_PROMISE_SCRIPT The default is for all to be allowed. Promise handlers can also be run manually by using Utils.CallPromiseHandlers()

Detailed Description

The Options class is used to get/set options that REPORTER uses for certain functions. The options are available as **class** properties. See the documentation for more details. An example: Options.run_promises = Options.RUN_PROMISE_WINDOW_LOOP

Page class

The Page class gives access to pages in Reporter. More...

The REPORTER JavaScript API provides many class constants, properties and methods. 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

Member functions

- DeleteItem(index*[integer]*)
- Duplicate(index (optional)*[integer]*)
- Generate()
- GetAllItems()
- GetItem(index*[integer]*)
- ImportItem(filename*[string]*)

Page properties

Name	Type	Description
items (read only)	integer	The total number of items on the page
master (read only)	logical	true if this is a master page object.
name	string	Name of the Page

Detailed Description

The Page class allows you to access the pages in templates that Reporter currently has open.

Constructor

new Page(template*[Template]*, options (optional)*[object]*)

Description

Create a new Page..

Arguments

Name	Type	Description
template	Template	Template to create page in

options (optional)	object	Options specifying various page properties, including where the page should be created. If omitted, the default values below will be used. Object has the following properties:

Name	Type	Description
colour (optional)	Colour object	Page background colour (white if omitted)
index (optional)	integer	The page index at which the new page will be inserted (indices start from zero). You cannot create pages prior to the current page i.e. the index must be greater than the index of the current page. If omitted, the new page will be created immediately after the current page. Note that the current page continues to be the page that the Script item is running on (it does not change to the newly-created page).
name (optional)	string	Name for page (empty if omitted)

Return type

Page object

Example

To create a new blank Page object in template *t*:

```
var page = new Page(t);
```

To create a new red page named "Last page" as the last page in template *t*:

```
var page = new Page(t, {name:"Last page", colour:Colour.Red(),
index:t.GetAllPages().length});
```

new Page(template*[Template]*, name (optional)*[string]*) [deprecated]

This function is deprecated in version 17.0. It is only provided to keep old scripts working. We strongly advise against using it in new scripts. Support may be removed in future versions.

Description

Create a new Page.

Arguments

Name	Type	Description
template	Template	Template to create page in
name (optional)	string	Name for page (empty if omitted)

Return type

Page object

Example

To create a new blank Page object in template t:

```
var page = new Page(t);
```

Details of functions

DeleteItem(index*[integer]*)

Description

Deletes an item from a page.

Arguments

Name	Type	Description
index	integer	The index of the item that you want to delete. Note that indices start at 0.

Return type

No return value

Example

To delete the first item of page *p*:

```
p.DeleteItem(0);
```

Duplicate(index (optional)*[integer]*)

Description

Duplicate a page

Arguments

Name	Type	Description
index (optional)	integer	The page index that you want to insert the duplicate page at in the template. Note that indices start at 0. If omitted the duplicate page will be put after the one that you are duplicating.

Return type

Page object

Example

To duplicate page p:

```
var dp = p.Duplicate();
```

To duplicate page p putting the duplicate as the first page in the template:

```
var dp = p.Duplicate(0);
```

Generate()

Description

Generate a page

Arguments

No arguments

Return type

no return value

Example

To generate page p:

```
p.Generate();
```

GetAllItems()

Description

Gets all of the items from a page.

Arguments

No arguments

Return type

Array of Item objects

Example

To get all of the items on page p:

```
var items = p.GetAllItems();
```

GetItem(index*[integer]*)

Description

Get an item from a page.

Arguments

Name	Type	Description
index	integer	The index of the item on the page that you want to get. Note that indices start at 0.

Return type

Item

Example

To get the 1st item on page p:

```
p.GetItem(0);
```

ImportItem(filename*[string]*)

Description

Import an item from a file onto the page.

Arguments

Name	Type	Description
filename	string	File containing the object to import

Return type

Item

Example

To read an item from file "item.oro" and put it on page p:

```
p.ImportItem("item.oro");
```

Reporter class

The Reporter class contains constants for use in REPORTER. More...

The REPORTER JavaScript API provides many class constants, properties and methods. 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

Reporter constants

Name	Description
Reporter.CapFlat	This constant is deprecated in version 12.0. It is only provided to keep old scripts working. We strongly advise against using it in new scripts. Support may be removed in future versions. Please use Reporter.CAP_FLAT instead [deprecated]
Reporter.CapRound	This constant is deprecated in version 12.0. It is only provided to keep old scripts working. We strongly advise against using it in new scripts. Support may be removed in future versions. Please use Reporter.CAP_ROUND instead [deprecated]
Reporter.CapSquare	This constant is deprecated in version 12.0. It is only provided to keep old scripts working. We strongly advise against using it in new scripts. Support may be removed in future versions. Please use Reporter.CAP_SQUARE instead [deprecated]
Reporter.JoinBevel	This constant is deprecated in version 12.0. It is only provided to keep old scripts working. We strongly advise against using it in new scripts. Support may be removed in future versions. Please use Reporter.JOIN_BEVEL instead [deprecated]
Reporter.JoinMitre	This constant is deprecated in version 12.0. It is only provided to keep old scripts working. We strongly advise against using it in new scripts. Support may be removed in future versions. Please use Reporter.JOIN_MITRE instead [deprecated]
Reporter.JoinRound	This constant is deprecated in version 12.0. It is only provided to keep old scripts working. We strongly advise against using it in new scripts. Support may be removed in future versions. Please use Reporter.JOIN_ROUND instead [deprecated]
Reporter.JustifyBottom	This constant is deprecated in version 12.0. It is only provided to keep old scripts working. We strongly advise against using it in new scripts. Support may be removed in future versions. Please use Reporter.JUSTIFY_BOTTOM instead [deprecated]
Reporter.JustifyCentre	This constant is deprecated in version 12.0. It is only provided to keep old scripts working. We strongly advise against using it in new scripts. Support may be removed in future versions. Please use Reporter.JUSTIFY_CENTRE instead [deprecated]
Reporter.JustifyLeft	This constant is deprecated in version 12.0. It is only provided to keep old scripts working. We strongly advise against using it in new scripts. Support may be removed in future versions. Please use Reporter.JUSTIFY_LEFT instead [deprecated]

Reporter.JustifyMiddle	This constant is deprecated in version 12.0. It is only provided to keep old scripts working. We strongly advise against using it in new scripts. Support may be removed in future versions. Please use Reporter.JUSTIFY_MIDDLE instead [deprecated]
Reporter.JustifyRight	This constant is deprecated in version 12.0. It is only provided to keep old scripts working. We strongly advise against using it in new scripts. Support may be removed in future versions. Please use Reporter.JUSTIFY_RIGHT instead [deprecated]
Reporter.JustifyTop	This constant is deprecated in version 12.0. It is only provided to keep old scripts working. We strongly advise against using it in new scripts. Support may be removed in future versions. Please use Reporter.JUSTIFY_TOP instead [deprecated]
Reporter.LineDash	This constant is deprecated in version 12.0. It is only provided to keep old scripts working. We strongly advise against using it in new scripts. Support may be removed in future versions. Please use Reporter.LINE_DASH instead [deprecated]
Reporter.LineDashDot	This constant is deprecated in version 12.0. It is only provided to keep old scripts working. We strongly advise against using it in new scripts. Support may be removed in future versions. Please use Reporter.LINE_DASH_DOT instead [deprecated]
Reporter.LineDashDotDot	This constant is deprecated in version 12.0. It is only provided to keep old scripts working. We strongly advise against using it in new scripts. Support may be removed in future versions. Please use Reporter.LINE_DASH_DOT_DOT instead [deprecated]
Reporter.LineDot	This constant is deprecated in version 12.0. It is only provided to keep old scripts working. We strongly advise against using it in new scripts. Support may be removed in future versions. Please use Reporter.LINE_DOT instead [deprecated]
Reporter.LineNone	This constant is deprecated in version 12.0. It is only provided to keep old scripts working. We strongly advise against using it in new scripts. Support may be removed in future versions. Please use Reporter.LINE_NONE instead [deprecated]
Reporter.LineSolid	This constant is deprecated in version 12.0. It is only provided to keep old scripts working. We strongly advise against using it in new scripts. Support may be removed in future versions. Please use Reporter.LINE_SOLID instead [deprecated]
Reporter.TextBold	This constant is deprecated in version 12.0. It is only provided to keep old scripts working. We strongly advise against using it in new scripts. Support may be removed in future versions. Please use Reporter.TEXT_BOLD instead [deprecated]
Reporter.TextItalic	This constant is deprecated in version 12.0. It is only provided to keep old scripts working. We strongly advise against using it in new scripts. Support may be removed in future versions. Please use Reporter.TEXT_ITALIC instead [deprecated]
Reporter.TextNormal	This constant is deprecated in version 12.0. It is only provided to keep old scripts working. We strongly advise against using it in new scripts. Support may be removed in future versions. Please use Reporter.TEXT_NORMAL instead [deprecated]
Reporter.TextUnderline	This constant is deprecated in version 12.0. It is only provided to keep old scripts working. We strongly advise against using it in new scripts. Support may be removed in future versions. Please use Reporter.TEXT_UNDERLINE instead [deprecated]
Reporter.ViewDesign	This constant is deprecated in version 12.0. It is only provided to keep old scripts working. We strongly advise against using it in new scripts. Support may be removed in future versions. Please use Reporter.VIEW_DESIGN instead [deprecated]

Reporter.ViewPresentation	This constant is deprecated in version 12.0. It is only provided to keep old scripts working. We strongly advise against using it in new scripts. Support may be removed in future versions. Please use Reporter.VIEW_PRESENTATION instead [deprecated]

Constants for Autotable source and row types

Name	Description
Reporter.AUTO_TABLE_DIRECTORY	Autotable data is generated from a directory.
Reporter.AUTO_TABLE_FILE	Autotable data is generated from a file.
Reporter.AUTO_TABLE_HEADER	Represents the header row in an Autotable.
Reporter.AUTO_TABLE_ROWS	Represents the rows with generated data in an Autotable.

Constants for Condition types

Name	Description
Reporter.CONDITION_BETWEEN	Treats the value as a number. True if value is between the two condition values
Reporter.CONDITION_CONTAINS_STRING	Treats the vlue as a string. True if the value contains the string
Reporter.CONDITION_DOESNT_CONTAIN_STRING	Treats the vlue as a string. True if the value does not contain the string
Reporter.CONDITION_DOESNT_MATCH_REGEX	Treats the value as a regular expression. True if the regular expression does not match
Reporter.CONDITION_EQUAL_TO	Treats the value as a string. True if the strings are equal
Reporter.CONDITION_GREATER_THAN	Treats the value as a number. True if value is greater than the condition value
Reporter.CONDITION_LESS_THAN	Treats the value as a number. True if value is less than the condition value
Reporter.CONDITION_MATCHES_REGEX	Treats the value as a regular expression. True if the regular expression matches
Reporter.CONDITION_NOT_BETWEEN	Treats the value as a number. True if value is between the two condition values
Reporter.CONDITION_NOT_EQUAL_TO	Treats the value as a string. True if the strings are not equal

Constants for Justification

Name	Description
Reporter.JUSTIFY_BOTTOM	Bottom justification of text
Reporter.JUSTIFY_CENTRE	Centre justification of text
Reporter.JUSTIFY_LEFT	Left justification of text
Reporter.JUSTIFY_MIDDLE	Middle justification of text
Reporter.JUSTIFY_RIGHT	Right justification of text
Reporter.JUSTIFY_TOP	Top justification of text

Constants for Line cap style

Name	Description

Reporter.CAP_FLAT	A square line ending at the end point of the line
Reporter.CAP_ROUND	A rounded line ending
Reporter.CAP_SQUARE	A square line that extends beyond the end point of the line by half the line width

Constants for Line join style

Name	Description
Reporter.JOIN_BEVEL	The triangular notch where the line segments meet is filled
Reporter.JOIN_MITRE	The outer edges of the line segments are extended to meet at an angle and this is filled
Reporter.JOIN_ROUND	A circular arc between the two line segments is filled

Constants for Line style

Name	Description
Reporter.LINE_DASH	A dashed line (dashes separated by a few pixels)
Reporter.LINE_DASH_DOT	A line drawn with alternate dashes and dots
Reporter.LINE_DASH_DOT_DOT	A line drawn with one dash and two dots
Reporter.LINE_DOT	A dotted line (dots separated by a few pixels)
Reporter.LINE_NONE	Invisible line
Reporter.LINE_SOLID	A simple continuous line

Constants for Text style

Name	Description
Reporter.TEXT_BOLD	Text drawn in a bold font
Reporter.TEXT_ITALIC	Text drawn in an italic font
Reporter.TEXT_NORMAL	Text drawn in a normal font
Reporter.TEXT_UNDERLINE	Text drawn underlined

Constants for View

Name	Description
Reporter.VIEW_DESIGN	Show template in design view
Reporter.VIEW_PRESENTATION	Show template in presentation view

Reporter properties

Name	Type	Description
currentTemplate	Template	This property is deprecated in version 12.0. It is only provided to keep old scripts working. We strongly advise against using it in new scripts. Support may be removed in future versions. Please use Template.GetCurrent() instead [deprecated]

templates	array	This property is deprecated in version 12.0. It is only provided to keep old scripts working. We strongly advise against using it in new scripts. Support may be removed in future versions. Please use Template.GetAll() instead [deprecated]

Detailed Description

The Reporter class allows you to access constants used in REPORTER.

Template class

The Template class gives access to templates in Reporter. More...

The REPORTER JavaScript API provides many class constants, properties and methods. 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

Class functions

- GetAll()
- GetCurrent()

Member functions

- Close()
- DeletePage(index*[integer]*)
- DeleteTemporaryVariables()
- EditVariables(title (optional)*[string]*, message (optional)*[string]*, update (optional)*[boolean]*, variables (optional)*[array]*, columns (optional)*[constant]*, alphabetical (optional)*[boolean]*)
- ExpandVariablesInString(string*[string]*)
- Generate()
- GetAllPages()
- GetMaster()
- GetPage(index*[integer]*)
- GetVariableDescription(name*[string]*)
- GetVariableValue(name*[string]*)
- Html(filename*[string]*)
- Pdf(filename*[string]*)
- Ppt(filename*[string]*) [deprecated]
- Pptx(filename*[string]*)
- Print(printer*[string]*)
- Save()
- SaveAs(filename*[string]*)
- Update()

Template properties

Name	Type	Description
filename (read only)	string	Filename (without path) of the Template.
name (read only)	string	This property is deprecated in version 15.0. It is only provided to keep old scripts working. We strongly advise against using it in new scripts. Support may be removed in future versions. Name of the Template. This property has been preserved for compatability with older scripts. It either contains the absolute path and filename, or just the filename, depending on how the Template was opened. Please use the filename and path properties for consistent results. [deprecated]
pages (read only)	integer	Number of Pages in template
path (read only)	string	Absolute path (without filename) of the Template. If the Template is new and has not yet been saved, this property will be empty.

variables	array	This property is deprecated in version 12.0. It is only provided to keep old scripts working. We strongly advise against using it in new scripts. Support may be removed in future versions. Array of Variable objects for this template. Please use Variable.GetAll() and Variable.GetFromName() instead. [deprecated]
view	constant	Current view type (presentation or design view) for this Template. Can be: Reporter.VIEW_DESIGN or Reporter.VIEW_PRESENTATION.

Detailed Description

The Template class allows you to access the templates that Reporter currently has open.
Note that if you want to get a list of the current templates in Reporter you should see the templates array in the reporter object.
The currently active template is stored in the currentTemplate property of the reporter object.

Constructor

new Template(filename (optional)*[string]*)

Description

Create a new Template. The filename argument is optional. If present it is a file to open

Arguments

Name	Type	Description
filename (optional)	string	Name of template file to open

Return type

Template object

Example

To create a new blank Template object

```
var template = new Template();
```

Details of functions

Close()

Description

Close a template.
Note that if you call this function for a Template object, the Template data will be deleted, so you should not try to use it afterwards!.

Arguments

No arguments

Return type

no return value

Example

To close template data:
```
data.Close();
```

DeletePage(index*[integer]*)

Description

Deletes a page from a template.

Arguments

Name	Type	Description
index	integer	The index of the page that you want to delete. Note that indices start at 0.

Return type

No return value

Example

To delete the first page of template t:
```
t.DeletePage(0);
```

DeleteTemporaryVariables()

Description

Deletes any temporary variables from a template.

Arguments

No arguments

Return type

No return value

Example

To delete all the temporary variables from template t:
```
t.DeleteTemporaryVariables();
```

EditVariables(title (optional)*[string]*, message (optional)*[string]*, update (optional)*[boolean]*, variables (optional)*[array]*, columns (optional)*[constant]*, alphabetical (optional)*[boolean]*)

Description

Start a dialog to edit the template variables

Arguments

Name	Type	Description
title (optional)	string	Title for dialog. If omitted, null or an empty string is given then the default title will be used.
message (optional)	string	Message to show in dialog. If omitted, null or an empty string is given then the default message will be used.
update (optional)	boolean	Whether the variables in the template will be updated with the new values if OK is pressed. Setting this to be false allows you to check variable values before updating them from a script. If omitted the default is true
variables (optional)	array	A list of variables to show in the dialog. If omitted, null or an empty array, all variables will be shown
columns (optional)	constant	Columns to show in the dialog (as well as the variable value column). Can be a bitwise OR of Variable.NAME, Variable.TYPE, Variable.DESCRIPTION, Variable.FORMAT, Variable.PRECISION and Variable.TEMPORARY. If omitted columns will be shown for name and description
alphabetical (optional)	boolean	Whether to sort variables in the table by alphabetical order. If false, variables are listed in the order they are passed in the optional variables argument. If no variables are passed to the function, all template variables will be shown in alphabetical order. If omitted, the default value is true.

Return type

Object containing the variable names and values or null if cancel was pressed.

Example

To edit all of the variables in template:

```
var variables = template.EditVariables();
```

To edit variables TEST and EXAMPLE in template giving a title and a message, returning the edited values but **not** updating them in the template:

```
var variables = template.EditVariables("Edit variables", "Type in the values",
false, ["TEST", "EXAMPLE");
```

ExpandVariablesInString(string*[string]*)

Description

Replaces any variables in a string with their current values

Arguments

Name	Type	Description
string	string	The string you want to expand variables in.

Return type

String (string) with variables expanded. If a variable in a string does not exist it is replaced by a blank.

Example

If the variable FRED in template contains the value "test", then the following

```
var value = template.ExpandVariablesInString("This is a %FRED%");
```

will return "This is a test" in variable value.

Generate()

Description

Generate a template

Arguments

No arguments

Return type

no return value

Example

To generate template data:

```
data.Generate();
```

GetAll() [static]

Description

Get all of the open templates

Arguments

No arguments

Return type

array of Template objects or null if no open templates

Example

To get all of the templates open in REPORTER:

```
var templates = Template.GetAll();
```

GetAllPages()

Description

Gets all of the pages from a template.

Arguments

No arguments

Return type

Array of Page objects

Example

To get all of the pages from template t:
```
var pages = t.GetAllPages();
```

GetCurrent() [static]

Description

Get the currently active template

Arguments

No arguments

Return type

Template object or null if no active template

Example

To get the current template open in REPORTER:
```
var current_template = Template.GetCurrent();
```

GetMaster()

Description

Get the master page from a template.

Arguments

No arguments

Return type

Page object

Example

To get the master page of template t:
```
var m = t.GetMaster();
```

GetPage(index*[integer]*)

Description

Get a page from a template.

Arguments

Name	Type	Description

index	integer	The index of the page that you want to get. Note that indices start at 0.

Return type

Page object

Example

To get the first page of template t:

```
var p = t.GetPage(0);
```

GetVariableDescription(name*[string]*)

Description

Get the description for a variable

Arguments

Name	Type	Description
name	string	Variable name you want to get description for.

Return type

Variable description (string) or null if variable does not exist

Example

To get description for variable FRED in template:

```
var description = template.GetVariableDescription("FRED");
```

GetVariableValue(name*[string]*)

Description

Get the value for a variable

Arguments

Name	Type	Description
name	string	Variable name you want to get value for.

Return type

Variable value (string) or null if variable does not exist

Example

To get value for variable FRED in template:

```
var value = template.GetVariableValue("FRED");
```

Html(filename*[string]*)

Description

Save a template as HTML

Arguments

Name	Type	Description
filename	string	Filename you want to save.

Return type

no return value

Example

To save template data as file /data/test/template.html:

```
data.Html("/data/test/template.html");
```

Pdf(filename*[string]*)

Description

Save a template as Adobe Acrobat PDF

Arguments

Name	Type	Description
filename	string	Filename you want to save.

Return type

no return value

Example

To save template data as file /data/test/template.pdf:

```
data.Pdf("/data/test/template.pdf");
```

Ppt(filename*[string]*) [deprecated]

This function is deprecated in version 18.0. It is only provided to keep old scripts working. We strongly advise against using it in new scripts. Support may be removed in future versions.

Description

Save a template as PowerPoint. This function is deprecated. Use Template.Pptx instead.

Arguments

Name	Type	Description
filename	string	Filename you want to save.

Return type

no return value

Example

To save template data as file /data/test/template.pptx:

```
data.Ppt("/data/test/template.pptx");
```

Pptx(filename*[string]*)

Description

Save a template as PowerPoint

Arguments

Name	Type	Description
filename	string	Filename you want to save.

Return type

no return value

Example

To save template data as file /data/test/template.pptx:

```
data.Pptx("/data/test/template.pptx");
```

Print(printer*[string]*)

Description

Print template on a printer

Arguments

Name	Type	Description
printer	string	Printer you want to print to.

Return type

no return value

Example

To print template data on printer myprinter:

```
data.Print("myprinter");
```

Save()

Description

Save a template

Arguments

No arguments

Return type

no return value

Example

To save template data:

```
data.Save();
```

SaveAs(filename*[string]*)

Description

Save a template/report with a new name

Arguments

Name	Type	Description
filename	string	Filename you want to save. Note if you use the .orr extension the template will be saved as a report if generated.

Return type

no return value

Example

To save template data as file /data/test/template.opt:

```
data.SaveAs("/data/test/template.opt");
```

Update()

Description

Update/redraw a template

Arguments

No arguments

Return type

no return value

Example

To update template data:

```
data.Update();
```

Utils class

The Utils class contains various useful utility functions. More...

The REPORTER JavaScript API provides many class constants, properties and methods. 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

Class functions

- CallPromiseHandlers()
- HiResTimer()
- TimerResolution()

Detailed Description

The Utils class is used to provide various useful functions.

Details of functions

CallPromiseHandlers() [static]

Description

Manually call any promise handlers/callbacks in the job queue

Arguments

No arguments

Return type

no return value

Example

To run any queued promise handlers/callbacks:

```
Utils.CallPromiseHandlers();
```

HiResTimer() [static]

Description

A high resolution timer that can be used to time how long things take. The first time this is called the timer will start and return 0. Subsequent calls will return the time in nanoseconds since the first call. Note that the timer will almost certainly not have 1 nanosecond precision but, depending on the platform, should should have a resolution of at least 1 microsecond.

Arguments

No arguments

Return type

number

Example

To time how long something takes to nanosecond precision:

```
var start = Utils.HiResTimer();
do something that takes some time...
var end = Utils.HiResTimer();
Message("it took " + (end-start) + "nanoseconds");
```

TimerResolution() [static]

Description

Returns the resolution (precision) of the Utils.HiResTimer() timer in nanoseconds

Arguments

No arguments

Return type

number

Example

To find the resolution of the timer in nanoseconds:

```
var resolution = Utils.TimerResolution();
```

Variable class

The Variable class gives access to variables in Reporter. More...

The REPORTER JavaScript API provides many class constants, properties and methods. 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

Class functions

- GetAll(template*[Template]*)
- GetFromName(template*[Template]*, name*[string]*)

Member functions

- Remove()

Variable constants

Name	Description
Variable.DESCRIPTION	Show variable description when editing variables with Template.EditVariables()
Variable.FORMAT	Show variable format when editing variables with Template.EditVariables()
Variable.FORMAT_FLOAT	Variable has floating point number format
Variable.FORMAT_GENERAL	Variable has general format
Variable.FORMAT_INTEGER	Variable has integer format
Variable.FORMAT_ LOWERCASE	Variable has lower case format
Variable.FORMAT_NONE	Variable has no format
Variable.FORMAT_SCIENTIFIC	Variable has scientific format
Variable.FORMAT_UPPERCASE	Variable has upper case format
Variable.NAME	Show variable name when editing variables with Template.EditVariables()
Variable.PRECISION	Show variable precision when editing variables with Template.EditVariables()
Variable.READONLY	Show variable readonly status when editing variables with Template.EditVariables()
Variable.TEMPORARY	Show variable temporary status when editing variables with Template.EditVariables()
Variable.TYPE	Show variable type when editing variables with Template.EditVariables()
Variable.VALUE	Show variable value when editing variables with Template.EditVariables()

Variable properties

Name	Type	Description
description	string	Variable description

format	constant	Variable format. Can be Variable.FORMAT_NONE, Variable.FORMAT_FLOAT, Variable.FORMAT_SCIENTIFIC, Variable.FORMAT_GENERAL, Variable.FORMAT_INTEGER, Variable.FORMAT_UPPERCASE or Variable.FORMAT_LOWERCASE
name	string	Variable name
precision	integer	Variable precision for floating point numbers.
readonly	logical	If Variable is read only or not.
temporary	logical	If Variable is temporary or not.
type	string	Variable type. Predefined types are "Directory", "File(absolute)", "File(basename)", "File(extension)", "File(tail)", "General", "Number" and "String". Alternatively give your own type. e.g. "NODE ID"
value	string	Variable value

Detailed Description

The Variable class allows you to access the name, description and value of a variable inside Reporter.
Note that if you want to get a list of the variables used in a Template you should see the variables array in the Template object.
The name, description and value properties give access to the variable name, description and value respectively.

Constructor

new Variable(template*[Template]*, name*[string]*, description (optional)*[string]*, value (optional)*[string]*, type (optional)*[string]*, readonly (optional)*[boolean]*, temporary (optional)*[boolean]*)

Description

Create a new Variable. The template and name arguments MUST be given, all others are optional

Arguments

Name	Type	Description
template	Template	Template object to create variable in
name	string	Name of variable
description (optional)	string	Description of variable
value (optional)	string	Variable value
type (optional)	string	Type of variable. Predefined types are "Directory", "File(absolute)", "File(basename)", "File(extension)", "File(tail)", "General", "Number" and "String". Alternatively give your own type. e.g. "NODE ID". If omitted default is "General"
readonly (optional)	boolean	If variable is readonly or not. If omitted default is false.
temporary (optional)	boolean	If variable is temporary or not. If omitted default is true.

Return type

Variable object

Example

To create a new Variable object called TEST with description 'test variable', type of "Number" and value '10' which is not readonly for template, templ

```
var variable = new Variable(templ, "TEST", "test variable", "10", "Number",
false);
```

Details of functions

GetAll(template*[Template]*) [static]

Description

Returns an array of Variable objects for all of the variables in a Template.

Arguments

Name	Type	Description
template	Template	Template to get the variables from

Return type

Array of Variable objects

Example

To get all the variables in template t:

```
var v = Variable.GetAll(t);
```

GetFromName(template*[Template]*, name*[string]*) [static]

Description

Returns the Variable object for a variable name.

Arguments

Name	Type	Description
template	Template	Template to find the variable in
name	string	name of the variable you want the Variable object for

Return type

Variable object (or null if variable does not exist)

Example

To get the Variable object for variable EXAMPLE in template t:

```
var v = Variable.GetFromName(t, "EXAMPLE");
```

Remove()

Description

Remove a variable
Note that if you call this function for a Variable object, the Variable data will be deleted, so you should not try to use it afterwards!.

Arguments

No arguments

Return type

no return value

Example

To remove variable data:

```
data.Remove();
```

Window class

The Window class gives access to windows for a graphical user interface. More...

The REPORTER JavaScript API provides many class constants, properties and methods. 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

Class functions

- Error(title*[string]*, error*[string]*, buttons (optional)*[constant]*)
- GetDirectory(initial (optional)*[string]*)
- GetFile(extension (optional)*[string]*, allow new (optional)*[boolean]*, initial (optional)*[string]*)
- GetFiles(extension (optional)*[string]*)
- GetInteger(title*[string]*, message*[string]*)
- GetNumber(title*[string]*, message*[string]*)
- GetOptions(title*[string]*, message*[string]*, options*[object]*)
- GetString(title*[string]*, message*[string]*)
- Information(title*[string]*, info*[string]*, buttons (optional)*[constant]*)
- Message(title*[string]*, message*[string]*, buttons (optional)*[constant]*)
- Question(title*[string]*, question*[string]*, buttons (optional)*[constant]*)
- Warning(title*[string]*, warning*[string]*, buttons (optional)*[constant]*)

Window constants

Name	Description
Window.CANCEL	Show CANCEL button
Window.NO	Show NO button
Window.OK	Show OK button
Window.YES	Show YES button

Detailed Description

The Window class is used to define several standard windows that can be used to read data, give messages and provide feedback

Details of functions

Error(title*[string]*, error*[string]*, buttons (optional)*[constant]*) [static]

Description

Show an error message in a window.

Arguments

Name	Type	Description
title	string	Title for window.
error	string	Error message to show in window.

buttons (optional)	constant	The buttons to use. Can be bitwise OR of Window.OK, Window.CANCEL, Window.YES or Window.NO. If this is omitted an OK button will be used.

Return type

Button pressed

Example

To show error *Critical error!\nAbort?* in window with title *Error* with Yes and No buttons:

```
var answer = Window.Error("Error", "Critical error!\nAbort?", Window.YES |
Window.NO);
if (answer == Window.YES) Exit();
```

GetDirectory(initial (optional)*[string]*) [static]

Description

Map the directory selector box native to your machine, allowing you to choose a directory.

Arguments

Name	Type	Description
initial (optional)	string	Initial directory to start from.

Return type

directory (string), (or null if cancel pressed).

Example

To select a directory:

```
var dir = Window.GetDirectory();
```

GetFile(extension (optional)*[string]*, allow new (optional)*[boolean]*, initial (optional)*[string]*) [static]

Description

Map a file selector box allowing you to choose a file. See also Window.GetFiles()

Arguments

Name	Type	Description
extension (optional)	string	Extension to filter by.
allow new (optional)	boolean	Allow creation of new file.
initial (optional)	string	Initial directory to start from.

Return type

filename (string), (or null if cancel pressed).

Example

To select a file using extension '.key':
```
var file = Window.GetFile(".key");
```

GetFiles(extension (optional)*[string]*) [static]

Description

Map a file selector box allowing you to choose multiple files. See also Window.GetFile()

Arguments

Name	Type	Description
extension (optional)	string	Extension to filter by.

Return type

Array of filenames (strings), or null if cancel pressed.

Example

To select multiple files using extension '.key':
```
var files = Window.GetFiles(".key");
```

GetInteger(title*[string]*, message*[string]*) [static]

Description

Map a window allowing you to input an integer. OK and Cancel buttons are shown.

Arguments

Name	Type	Description
title	string	Title for window.
message	string	Message to show in window.

Return type

Integer. Value input, (or null if cancel pressed).

Example

To create an input window with title *Input* and message *Input integer* and return the value input:
```
var value = Window.GetInteger("Input", "Input integer");
```

GetNumber(title*[string]*, message*[string]*) [static]

Description

Map a window allowing you to input a number. OK and Cancel buttons are shown.

Arguments

Name	Type	Description
title	string	Title for window.
message	string	Message to show in window.

Return type

Real. Value input, (or null if cancel pressed).

Example

To create an input window with title *Input* and message *Input number* and return the value input:

```
var value = Window.GetNumber("Input", "Input number");
```

GetOptions(title*[string]*, message*[string]*, options*[object]*) [static]

Description

Map a window allowing you to input various options. OK and Cancel buttons are shown.

Arguments

Name	Type	Description
title	string	Title for window.
message	string	Message to show in window.
options	Array of objects	Array of objects listing options that can be set. If OK is pressed the objects will be updated with the values from the widgets. If cancel is pressed they will not. Object has the following properties: **Name** / **Type** / **Description** selected (optional) / boolean / If checkbox is selected or not text / string / Text to show next to option type / string / Type of option. Can be "label" (plain text), "text" (a one line text widget), "textbox" (a multi line text widget) or "checkbox" (a checkable option) value / string / Text to show for option

Return type

false if cancel pressed, true if OK pressed.

Example

To create a window with title *Options* , message *Please give the options* with label, text, textbox and checkbox widgets:

```
var options = [
                 { text:"Label example", type:"label", value:"banana" },
                 { text:"Text example", type:"text", value:"single line of text"
},
                 { text:"Textbox example", type:"textbox",
value:"Multiple\\nlines\\nof\\ntext" },
                 { text:"Checkbox example", type:"checkbox", value:"Do this?",
selected:true }
                ];
ok = Window.GetOptions("Options", "Please give the options", options);
```

GetString(title*[string]*, message*[string]*) [static]

Description

Map a window allowing you to input a string. OK and Cancel buttons are shown.

Arguments

Name	Type	Description
title	string	Title for window.
message	string	Message to show in window.

Return type

String. Value input, (or null if cancel pressed).

Example

To create an input window with title *Input* and message *Input string* and return the value input:

```
var value = Window.GetString("Input", "Input string");
```

Information(title*[string]*, info*[string]*, buttons (optional)*[constant]*) [static]

Description

Show information in a window.

Arguments

Name	Type	Description
title	string	Title for window.
info	string	Information to show in window.
buttons (optional)	constant	The buttons to use. Can be bitwise OR of Window.OK, Window.CANCEL, Window.YES or Window.NO. If this is omitted an OK button will be used.

Return type

Button pressed

Example

To show information *Information* in window with title *Example* with OK and Cancel buttons:

```
var answer = Window.Information("Example", "Information", Window.OK |
Window.CANCEL);
if (answer == Window.CANCEL) Message("You pressed the Cancel button");
```

Message(title*[string]*, message*[string]*, buttons (optional)*[constant]*) [static]

Description

Show a message in a window.

Arguments

Name	Type	Description
title	string	Title for window.
message	string	Message to show in window.
buttons (optional)	constant	The buttons to use. Can be bitwise OR of Window.OK, Window.CANCEL, Window.YES or Window.NO. If this is omitted an OK button will be used

Return type

Button pressed

Example

To show message *Press YES or NO* in window with title *Example* with YES and NO buttons:

```
var answer = Window.Message("Example", "Press YES or NO", Window.YES |
Window.NO);
if (answer == Window.NO) Message("You pressed No");
```

Question(title*[string]*, question*[string]*, buttons (optional)*[constant]*) [static]

Description

Show a question in a window.

Arguments

Name	Type	Description
title	string	Title for window.
question	string	Question to show in window.
buttons (optional)	constant	The buttons to use. Can be bitwise OR of Window.OK, Window.CANCEL, Window.YES or Window.NO. If this is omitted Yes and No button will be used.

Return type

Button pressed

Example

To show question *Do you want to continue?* in window with title *Question*:

```
var answer = Window.Question("Question", "Do you want to continue?");
if (answer == Window.NO) Message("You pressed No");
```

Warning(title*[string]*, warning*[string]*, buttons (optional)*[constant]*) [static]

Description

Show a warning message in a window.

Arguments

Name	Type	Description
title	string	Title for window.
warning	string	Warning message to show in window.

buttons (optional)	constant	The buttons to use. Can be bitwise OR of Window.OK, Window.CANCEL, Window.YES or Window.NO. If this is omitted an OK button will be used.

Return type

Button pressed

Example

To show warning *Title is blank\nSet to ID?* in window with title *Warning* with Yes and No buttons:

```
var answer = Window.Warning("Warning", "Title is blank\nSet to ID?", Window.YES
| Window.NO);
if (answer == Window.NO) Message("You pressed No");
```

E. Writing external programs/scripts

Programs or scripts for REPORTER that do some external function can be written in any language. It is up to you if you prefer to use a scripting language such as Perl, Python, Tcl etc or a compiled language such as C or Fortran.

Anything which a program prints to stdout (standard output) will be returned to REPORTER (the one exception to this is returning variables which is described below)

Returning variables from programs

To return a variable back to REPORTER output a line that take the form

```
VAR <NAME> VALUE="<value>" DESCRIPTION="<description>"
```
or
```
VAR <NAME> VALUE="<value>"
```

It will not inserted into the report as text but will be used to create a variable. See section 4.4 for more details.

Accessing existing variables in REPORTER

If you only want to use one or two variables from REPORTER then they can be passed as arguments to your program. However, if you want to access a lot of variables (or print all the variables to a file) this would not be possible.

To overcome this, REPORTER adds an extra argument to every program that it runs. This extra argument is a filename which contains lines of the form:

```
VAR <NAME> VALUE="<value>" DESCRIPTION="<description>"
```

You can read this file and pick up all the variables from REPORTER.

Example perl program to read variables file from REPORTER

The following example shows how you could read this file.

```
# Skeleton REPORTER Perl script showing extraction of variables fed to program
# The variable file REPORTER generates will be the LAST argument
#
# Variables are stored in a hash '%vars', each entry in the hash contains
# {value} and {description}.
#
# e.g. If REPORTER has a variable 'FRED' with value '1' and description
# 'Example variable' you can get at the variable value and description using:
#
# $vars{FRED}->{value}
# $vars{FRED}->{description}
#
# Arguments
# =========
# 1: Variables file
#
# Miles Thornton 23/5/2002
#
%vars = ();
if ($#ARGV >= 0)
{
    open (VAR, "< $ARGV[$#ARGV]") or die "Error: Cannot open variable file";
    while ( <VAR> )
    {
        chomp;
        &get_var_from_string($_);
    }
}
else
{
    die "Error: No variable file on the command line\n";
```

```
}
##########################################################
# START OF YOUR PROGRAM
#
# e.g. loop over variables and save them to a file
open (SAVE, "> varfile") or die "Error: Cannot open variables file";
foreach $var (sort keys %vars)
{
    print SAVE "Variable $var value=$vars{$var}->{value} ",
               "desc=$vars{$var}->{description}\n";
}
close (SAVE);
# END OF YOUR PROGRAM
##########################################################
exit;
#   ===================
sub get_var_from_string
#   ===================
#
# Tries to read a variable from the variable file
#
{
    my $string = shift;
    my ($var, $val, $desc);
    if ($string =~ /VAR\s+(\w+)\s+
                    VALUE\s*=\s*['"](.*?)['"]\s*
                    DESCRIPTION\s*=\s*['"](.*?)['"]
                   /x)
    {
        $var  = $1;
        $val  = $2;
        $desc = $3;
    }
    elsif ($string =~ /VAR\s+(\w+)\s+
                       DESCRIPTION\s*=\s*['"](.*?)['"]\s*
                       VALUE\s*=\s*['"](.*?)['"]
                      /x)
    {
        $var  = $1;
        $val  = $3;
        $desc = $2;
    }
    elsif ($string =~ /VAR\s+(\w+)\s+
                       VALUE\s*=\s*['"](.*?)['"]
                      /x)
    {
        $var  = $1;
        $val  = $2;
        $desc = undef;
    }
    if ($var)
    {
        $var = uc($var);
        $var =~ s/\s+/_/g;
        if (exists $vars{$var})
        {
            $vars{$var}->{value}       = $val;
            $vars{$var}->{description} = $desc;
        }
        else
        {
            my $variable = {};
            $variable->{value}       = $val;
            $variable->{description} = $desc;
            $vars{$var} = $variable;
        }
    }
}
```

Example program: Extracting the smallest timesteps (Text output)

These programs/scripts are designed to extract from the OTF file the 5 elements with the smallest timesteps, and write out the data as text to the standard output. They also output the smallest timestep as a REPORTER variable called **TIMESTEP**. Note that these programs/scripts are only simple examples and as such don't have all the necessary error checking that should be included.
They work by searching the OTF file for the text string "100 smallest timesteps" which appears towards the end of the model initialization section, and then reading in relevant element data from this list. An example of this section of an OTF file is shown below. The one argument for this program/script is the OTF filename (for example tube2.otf).

```
 The LS-DYNA time step size should not exceed 0.133E-05
 to avoid contact instabilities. If the step size is
 bigger then scale the penalty of the offending surface.
       0 t 0.0000E+00 dt 0.00E+00 flush i/o buffers
 100 smallest timesteps
 ----------------------
 element                        timestep
 shell     16620                0.66873E-06
 shell     16619                0.66873E-06
 shell     16612                0.66873E-06
 shell     16611                0.66873E-06
 shell     16572                0.66873E-06
 shell     16571                0.66873E-06
 shell     16564                0.66873E-06
 shell     16563                0.66873E-06
 shell     16520                0.66873E-06
 shell     16519                0.66873E-06
 shell     16512                0.66873E-06
 shell     16511                0.66873E-06
 shell     16504                0.66873E-06
 shell     16503                0.66873E-06
 shell     16472                0.66873E-06
```

Example programs to extract the data are shown in 4 languages:

- C
- C shell script
- Fortran
- Perl

C program/script

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define MAX_LEN 257
int main(int argc, char *argv[])
{
    char line[MAX_LEN], *ptr;
    int c, i, l, n = 5;
    float t, tmin;
    FILE *fp;
    if (argc < 2)
    {
         printf("No otf filename\n");
         exit(0);
    }
    if ( (fp = fopen(argv[1], "r")) == NULL)
    {
         printf("Cannot open otf file %s\n", argv[1]);
         exit(0);
    }
    while (fgets(line, MAX_LEN, fp))
    {
        if (strstr(line, "smallest timesteps"))
        {
             sscanf(line, "%d", &n);
             if (n > 5) n = 5;
             tmin = 1.0e+20;
```

```
            fgets(line, MAX_LEN, fp);
            fgets(line, MAX_LEN, fp);
            for (i=0; i<n; i++)
            {
                fgets(line, MAX_LEN, fp);
                printf ("%s", line);
/* Remove any trailing characters */
                l = strlen(line) - 1;
                while ( (c = line[l]) == ' ' || c=='\n' || c=='\r' || c=='\t')
                    l--;
                line[l+1] = '\0';
/* Find start of number */
                l = strlen(line) - 1;
                while ( (c = line[l]) != ' ')
                    l--;
                ptr = &line[l];
                sscanf(ptr, "%e", &t);
                if (t < tmin)
                    tmin = t;
            }
            printf ("VAR TIMESTEP VALUE=\"%e\"\n", tmin);
            exit(0);
        }
    }
    fclose(fp);
}
```

C Shell program/script

```
#!/bin/csh -f
#
# Script to extract the 5 smallest timesteps from otf file
#
# Arguments: 1: otf filename

# Test to see if there is an argument
if ($#argv < 1) then
    echo "No otf filename";
    exit;
endif
# test to see if the otf file exists
if ( !(-e $argv[1]) ) then
   echo "otf file $argv[1] does not exist";
   exit;
endif
# Use awk to extract the timesteps
awk '/smallest timesteps/ {                # search for smallest timestep \
                    n = $1;                # save how many found \
                    getline;               # skip a line \
                    getline;               # skip a line \
                    if (n > 5) n = 5;      # limit to 5 timesteps \
                    t = 1.0e+20;           # initialise smallest timestep \
                    for (i=0; i<n; i++)    # loop over lines \
                    {                      # \
                     getline;              # read the line \
                     print $0;             # print it \
                     if ($NF < t) t = $NF; # save timestep if smaller \
                    }                      # than current smallest \
                  }                        # \
 END {                                          # \
      printf ("VAR TIMESTEP VALUE=\"%e\"\n", t); # Print smallest timestep \
     }                                          # \
  ' $argv[1]
```

Fortran program/script

```
c
      character*80 fname,line
      integer elemno(5)
      real timestep(5)
```

```
      n=iargc(1)
c
c Read in model name argument
c
      call getarg(1,fname)
c
c Open model OTF file
c
      open (unit=25, file=fname, status='old')
c
c Scan file for line with the text string
c " 100 smallest timesteps"
c
 10   continue
      read (25,'(a)',end=900) line
      if (line(1:23).eq.' 100 smallest timesteps') then
         goto 20
      else
         goto 10
      endif
c
c Read in but ignore next 2 lines of data
c
 20   continue
      read(25,*)
      read(25,*)
c
c Read in the element no. and timestep data
c from the next five lines
c
 101  format(i10)
 102  format(e23.0)
c
      do 30 i=1,5
       read (25,'(a)') line
       read (line(7:16),101) elemno(i)
       read (line(20:42),102) timestep(i)
 30   continue
c
c Write out the data as a text output
c
 201  format (2x,i9,5x,e11.5)
c
      write (*,*) ' Element No.    Timestep  '
      do 40 i=1,5
       write (*,201) elemno(i),timestep(i)
 40   continue
c
c Also write out the smallest timestep as
c REPORTER variable
c
 301  format ('VAR TIMESTEP VALUE="',e11.5,'"')
      write(*,301) timestep(1)
      goto 999
c
 900  write(*,*) 'End of file reached'
c
 999  continue
      stop
      end
c
```

Perl program/script

```
# Perl Script to extract the 5 smallest timesteps from otf file
#
# Arguments: 1: otf filename
use strict;
# Test to see if there is an argument
if ($#ARGV < 0)
{
```

```
    print "No otf filename\n";
    exit;
}
# test to see if the otf file exists
if ( !(-e $ARGV[0]) )
{
    print "otf file $ARGV[0] does not exist\n";
    exit;
}
open (OTF, "< $ARGV[0]");
my $n;
my $t = 1.0e+20;
while ( <OTF>)
{
    if (/ (\d+) smallest timesteps/)
    {
         $n = $1;
         if ($n > 5) { $n = 5; }
         <OTF>;
         <OTF>;
         for (my $i=0; $i<$n; $i++)
         {
             $_ = <OTF>;
             print $_;
             my @f = split;
             if ($f[$#f] < $t) { $t = $f[$#f]; }
         }
         print "VAR TIMESTEP VALUE=\"$t\"\n";
         exit;
     }
}
close (OTF);
```

F. Unicode support

REPORTER has basic unicode (i.e. non-latin characters) support. This means that if you have the appropriate language kit and fonts installed on your computer you can input and use European accented, Japanese, Korean and Chinese characters. On Windows you can input unicode characters using the normal IME (global Input Method Editor).

The XML format that REPORTER uses to save files supports unicode.

As Japanese, Korean and Chinese have many common ideographs, but these may have different appearances depending on the font there is a preference in REPORTER which allows you to set the default language you want to use, `reporter*cjk_default` which can be either `Chinese`, `Japanese` or `Korean`.

Note that although REPORTER has unicode support, currently D3PLOT, T/HIS and LS-DYNA do not so you should not use unicode characters in filenames.

F.1 Output formats that support unicode

Currently only text objects and table headers can be output with unicode characters.

HTML

Unicode is fully supported in the HTML written by REPORTER. To view the HTML a user needs the appropriate fonts installed.

PowerPoint

Unicode is fully supported in the PowerPoint files written by REPORTER (as long as the appropriate language pack(s) are installed).

PDF

The PDF files created by REPORTER do not embed the fonts used in the document. However, newer versions of the acrobat reader will automatically detect that the document uses a Chinese, Japanese or Korean font and prompt the user to download the necessary fonts.

There are two preferences which affect what fonts are used in pdf files:
Firstly for Japanese the preference `reporter*japanese_font` indicates what font should be used for Japanese characters. It can be ’`Kozuka Mincho Pro`’ (a serif font) or ’`Kozuka Gothic Pro`’ (a sans serif font). The default is ’`Kozuka Gothic Pro`’.
For Chinese the preference `reporter*chinese_characters` indicates if traditional or simplified characters should be used. It can be `Traditional` or `Simplified`. the default is `Traditional`.

Installation organisation

The version19.0installation 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 their 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.

Version19.0.0 Installation structure

In version19.0.0 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. Version19.0.0 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`** =19.0for release19.0.0, thus **`OA_ADMIN_19.0`**) Admin level oa_pref file Other configuration files Timeout configuration file	

OA_ADMIN	*Optional*	Same as **OA_ADMIN_19.0**, 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_19.0** *is not defined then this non-release specific version is checked.*	
OA_ INSTALL_ xx	*Optional*	(**xx** =19.0for release19.0.0, thus **OA_ADMIN_19.0** All executables Installation level oa_pref file	**oasys*install_ dir: *<pathname>***
OA_ INSTALL	*Optional*	Same as **OA_INSTALL_19.0**. 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_19.0** *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_19.0" 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.

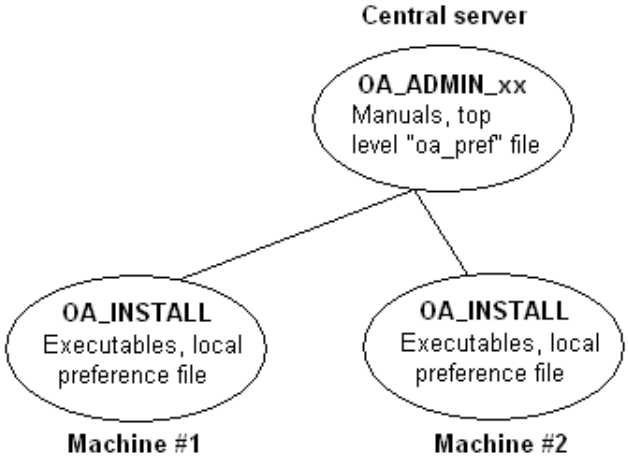
It is suggested that the **_xx** version suffix of **OA_INSTALL_xx** is used in order to keep parallel installations of different releases of the Oassys Ltd software separate on the machine.

b) A few machines on a small network, each user has their own machine

The top level administration directory can be installed on a network server, possibly also locating the manuals centrally.

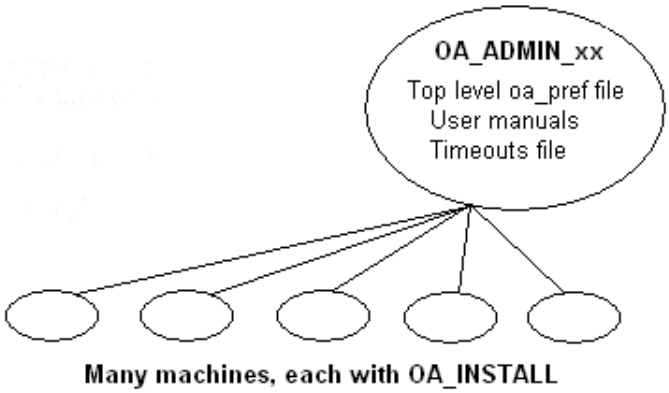
Each user's machine has its own 'installation' directory to give good performance, but there is no need to manage home or temporary directories centrally since each user 'owns' their machine.

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:

Release19.0.0	**Release19.0.1**
Top level directory **OA_ADMIN_19.0**	Top level directory **OA_ADMIN_19.01**
oa_pref file in **OA_ADMIN_19.0** contains: **oasys*install_dir:** *<pathname for **19.0.0** installation>* **oasys*manuals_dir:** *<pathname for **19.0.0** manuals>* **oasys*home_dir:** *<pathname for home directory>* **oasys*temp_dir:** *<pathname for temporary files>*	oa_pref file in **OA_ADMIN_19.01** contains: **oasys*install_dir:** *<pathname for **19.0.1** installation>* **oasys*manuals_dir:** *<pathname for**19.0.1** 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.

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

Apple Public Source

```
Copyright (c) 1999 Apple Computer, Inc. All rights reserved.
The contents of this file constitute Original Code as defined in and
are subject to the Apple Public Source License Version 1.1 (the
"License"). You may not use this file except in compliance with the
License. Please obtain a copy of the License at
http://www.apple.com/publicsource and read it before using this file.

This Original Code and all software distributed under the License are
distributed on an "AS IS" basis, WITHOUT WARRANTY OF ANY KIND, EITHER
EXPRESS OR IMPLIED, AND APPLE HEREBY DISCLAIMS ALL SUCH WARRANTIES,
INCLUDING WITHOUT LIMITATION, ANY WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT. Please see the
License for the specific language governing rights and limitations
under the License.
Copyright (c) 1992 NeXT Computer, Inc.  All rights reserved.
```

Note: the URL http://www.apple.com/publicsource cited above no longer exists, see instead https://spdx.org/licenses/APSL-1.1.html

Draco

google/draco is licensed under the Apache License:

```
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.
```

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 © <year> The FreeType
    Project (www.freetype.org).  All rights reserved.
   """
  Please replace <year> 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 licensed under the LGPL v2.1+. The exception to this is the x264 library used by FFmpeg, for which Arup have obtained a commercial license (see here).

License

Most files in FFmpeg are under the GNU Lesser General Public License version 2.1 or later (LGPL v2.1+). Read the file 'COPYING.LGPLv2.1' for details. Some other files have MIT/X11/BSD-style licenses. In combination the LGPL v2.1+ applies to FFmpeg.

Some optional parts of FFmpeg are licensed under the GNU General Public License version 2 or later (GPL v2+). See the file 'COPYING.GPLv2' for details. None of these parts are used by default, you have to explicitly pass '--enable-gpl' to configure to activate them. In this case, FFmpeg's license changes to GPL v2+.

```
Specifically, the GPL parts of FFmpeg are:
- libpostproc
- optional x86 optimization in the files
    - ‘libavcodec/x86/flac_dsp_gpl.asm‘
    - ‘libavcodec/x86/idct_mmx.c‘
    - ‘libavfilter/x86/vf_removegrain.asm‘
- the following building and testing tools
    - ‘compat/solaris/make_sunver.pl‘
    - ‘doc/t2h.pm‘
    - ‘doc/texi2pod.pl‘
    - ‘libswresample/tests/swresample.c‘
    - ‘tests/checkasm/*‘
    - ‘tests/tiny_ssim.c‘
- the following filters in libavfilter:
    - ‘signature_lookup.c‘
    - ‘vf_blackframe.c‘
    - ‘vf_boxblur.c‘
    - ‘vf_colormatrix.c‘
    - ‘vf_cover_rect.c‘
    - ‘vf_cropdetect.c‘
    - ‘vf_delogo.c‘
    - ‘vf_eq.c‘
    - ‘vf_find_rect.c‘
    - ‘vf_fspp.c‘
    - ‘vf_histeq.c‘
    - ‘vf_hqdn3d.c‘
    - ‘vf_kerndeint.c‘
    - ‘vf_lensfun.c‘ (GPL version 3 or later)
    - ‘vf_mcdeint.c‘
    - ‘vf_mpdecimate.c‘
    - ‘vf_nnedi.c‘
    - ‘vf_owdenoise.c‘
    - ‘vf_perspective.c‘
    - ‘vf_phase.c‘
    - ‘vf_pp.c‘
    - ‘vf_pp7.c‘
    - ‘vf_pullup.c‘
    - ‘vf_repeatfields.c‘
    - ‘vf_sab.c‘
    - ‘vf_signature.c‘
    - ‘vf_smartblur.c‘
    - ‘vf_spp.c‘
    - ‘vf_stereo3d.c‘
    - ‘vf_super2xsai.c‘
    - ‘vf_tinterlace.c‘
    - ‘vf_uspp.c‘
    - ‘vf_vaguedenoiser.c‘
    - ‘vsrc_mptestsrc.c‘
Should you, for whatever reason, prefer to use version 3 of the (L)GPL, then
the configure parameter ‘--enable-version3‘ will activate this licensing option
for you. Read the file ‘COPYING.LGPLv3‘ or, if you have enabled GPL parts,
‘COPYING.GPLv3‘ to learn the exact legal terms that apply in this case.
There are a handful of files under other licensing terms, namely:
* The files ‘libavcodec/jfdctfst.c‘, ‘libavcodec/jfdctint_template.c‘ and
  ‘libavcodec/jrevdct.c‘ are taken from libjpeg, see the top of the files for
  licensing details. Specifically note that you must credit the IJG in the
  documentation accompanying your program if you only distribute executables.
  You must also indicate any changes including additions and deletions to
  those three files in the documentation.
* ‘tests/reference.pnm‘ is under the expat license.
## External libraries
FFmpeg can be combined with a number of external libraries, which sometimes
affect the licensing of binaries resulting from the combination.
### Compatible libraries
The following libraries are under GPL version 2:
- avisynth
- frei0r
- libcdio
- libdavs2
- librubberband
- libvidstab
```

- libx264
- libx265
- libxavs
- libxavs2
- libxvid

When combining them with FFmpeg, FFmpeg needs to be licensed as GPL as well by passing ‘--enable-gpl‘ to configure.
The following libraries are under LGPL version 3:

- gmp
- libaribb24
- liblensfun

When combining them with FFmpeg, use the configure option ‘--enable-version3‘ to upgrade FFmpeg to the LGPL v3.
The VMAF, mbedTLS, RK MPI, OpenCORE and VisualOn libraries are under the Apache License
2.0. That license is incompatible with the LGPL v2.1 and the GPL v2, but not with
version 3 of those licenses. So to combine these libraries with FFmpeg, the license version needs to be upgraded by passing ‘--enable-version3‘ to configure.
The smbclient library is under the GPL v3, to combine it with FFmpeg,
the options ‘--enable-gpl‘ and ‘--enable-version3‘ have to be passed to configure to upgrade FFmpeg to the GPL v3.
Incompatible libraries
There are certain libraries you can combine with FFmpeg whose licenses are not compatible with the GPL and/or the LGPL. If you wish to enable these libraries, even in circumstances that their license may be incompatible, pass ‘--enable-nonfree‘ to configure. This will cause the resulting binary to be unredistributable.
The Fraunhofer FDK AAC and OpenSSL libraries are under licenses which are incompatible with the GPLv2 and v3. To the best of our knowledge, they are compatible with the LGPL.

HDF5

Copyright Notice and License Terms for
HDF5 (Hierarchical Data Format 5) Software Library and Utilities
---
HDF5 (Hierarchical Data Format 5) Software Library and Utilities
Copyright 2006 by The HDF Group.
NCSA HDF5 (Hierarchical Data Format 5) Software Library and Utilities
Copyright 1998-2006 by The Board of Trustees of the University of Illinois.
All rights reserved.
Redistribution and use in source and binary forms, with or without
modification, are permitted for any purpose (including commercial purposes)
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 materials provided with the distribution.
3. Neither the name of The HDF Group, the name of the University, nor the name of any Contributor may be used to endorse or promote products derived from this software without specific prior written permission from The HDF Group, the University, or the Contributor, respectively.

DISCLAIMER:
THIS SOFTWARE IS PROVIDED BY THE HDF GROUP AND THE CONTRIBUTORS
"AS IS" WITH NO WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED. IN NO
EVENT SHALL THE HDF GROUP OR THE CONTRIBUTORS BE LIABLE FOR ANY DAMAGES
SUFFERED BY THE USERS ARISING OUT OF THE USE OF THIS SOFTWARE, EVEN IF
ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

You are under no obligation whatsoever to provide any bug fixes, patches, or upgrades to the features, functionality or performance of the source code ("Enhancements") to anyone; however, if you choose to make your Enhancements available either publicly, or directly to The HDF Group, without imposing a separate written license agreement for such Enhancements, then you hereby grant the following license: a non-exclusive, royalty-free perpetual license to install, use, modify, prepare derivative works, incorporate into other computer software, distribute, and sublicense such enhancements or derivative

works thereof, in binary and source code form.

Limited portions of HDF5 were developed by Lawrence Berkeley National Laboratory (LBNL). LBNL’s Copyright Notice and Licensing Terms can be found here: COPYING_LBNL_HDF5 file in this directory or at http://support.hdfgroup.org/ftp/HDF5/releases/COPYING_LBNL_HDF5.

Contributors: National Center for Supercomputing Applications (NCSA) at the University of Illinois, Fortner Software, Unidata Program Center (netCDF), The Independent JPEG Group (JPEG), Jean-loup Gailly and Mark Adler (gzip), and Digital Equipment Corporation (DEC).

Portions of HDF5 were developed with support from the Lawrence Berkeley National Laboratory (LBNL) and the United States Department of Energy under Prime Contract No. DE-AC02-05CH11231.

Portions of HDF5 were developed with support from the University of California, Lawrence Livermore National Laboratory (UC LLNL).
The following statement applies to those portions of the product and must be retained in any redistribution of source code, binaries, documentation, and/or accompanying materials:

This work was partially produced at the University of California, Lawrence Livermore National Laboratory (UC LLNL) under contract no. W-7405-ENG-48 (Contract 48) between the U.S. Department of Energy (DOE) and The Regents of the University of California (University) for the operation of UC LLNL.

DISCLAIMER:

THIS WORK WAS PREPARED AS AN ACCOUNT OF WORK SPONSORED BY AN AGENCY OF THE UNITED STATES GOVERNMENT. NEITHER THE UNITED STATES GOVERNMENT NOR THE UNIVERSITY OF CALIFORNIA NOR ANY OF THEIR EMPLOYEES, MAKES ANY WARRANTY, EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY OR RESPONSIBILITY FOR THE ACCURACY, COMPLETENESS, OR USEFULNESS OF ANY INFORMATION, APPARATUS, PRODUCT, OR PROCESS DISCLOSED, OR REPRESENTS THAT ITS USE WOULD NOT INFRINGE PRIVATELY- OWNED RIGHTS. REFERENCE HEREIN TO ANY SPECIFIC COMMERCIAL PRODUCTS, PROCESS, OR SERVICE BY TRADE NAME, TRADEMARK, MANUFACTURER, OR OTHERWISE, DOES NOT NECESSARILY CONSTITUTE OR IMPLY ITS ENDORSEMENT, RECOMMENDATION, OR FAVORING BY THE UNITED STATES GOVERNMENT OR THE UNIVERSITY OF CALIFORNIA. THE VIEWS AND OPINIONS OF AUTHORS EXPRESSED HEREIN DO NOT NECESSARILY STATE OR REFLECT THOSE OF THE UNITED STATES GOVERNMENT OR THE UNIVERSITY OF CALIFORNIA, AND SHALL NOT BE USED FOR ADVERTISING OR PRODUCT ENDORSEMENT PURPOSES.

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.

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

COPYRIGHT NOTICE, DISCLAIMER, and LICENSE
==========
PNG Reference Library License version 2
---
 * Copyright (c) 1995-2019 The PNG Reference Library Authors.
 * Copyright (c) 2018-2019 Cosmin Truta.
 * Copyright (c) 2000-2002, 2004, 2006-2018 Glenn Randers-Pehrson.
 * Copyright (c) 1996-1997 Andreas Dilger.
 * Copyright (c) 1995-1996 Guy Eric Schalnat, Group 42, Inc.
The software is supplied "as is", without warranty of any kind, express or implied, including, without limitation, the warranties of merchantability, fitness for a particular purpose, title, and non-infringement. In no event shall the Copyright owners, or anyone distributing the software, be liable for any damages or other liability, whether in contract, tort or otherwise, arising from, out of, or in connection with the software, or the use or other dealings in the software, even if advised of the possibility of such damage.
Permission is hereby granted to use, copy, modify, and distribute this software, or portions hereof, for any purpose, without fee, 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 Copyright notice may not be removed or altered from any
    source or altered source distribution.
PNG Reference Library License version 1 (for libpng 0.5 through 1.6.35)
---------------------------------------------------------------------------
libpng versions 1.0.7, July 1, 2000, through 1.6.35, July 15, 2018 are
Copyright (c) 2000-2002, 2004, 2006-2018 Glenn Randers-Pehrson, are
derived from libpng-1.0.6, 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
    Mans Rullgard
    Cosmin Truta
    Gilles Vollant
    James Yu
    Mandar Sahastrabuddhe
    Google Inc.
    Vadim Barkov
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.
Some files in the "contrib" directory and some configure-generated
files that are distributed with libpng have other copyright owners, and
are released under other open source licenses.
libpng versions 0.97, January 1998, through 1.0.6, March 20, 2000, are
Copyright (c) 1998-2000 Glenn Randers-Pehrson, are derived from
libpng-0.96, 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, are derived from libpng-0.88,
and are 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
Some files in the "scripts" directory have other copyright owners,
but are released under this license.
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.
```

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
```

MPEG-LA

```
THIS PRODUCT IS LICENSED UNDER THE
AVC PATENT PORTFOLIO LICENSE FOR THE PERSONAL USE OF A
CONSUMER OR OTHER USES IN WHICH IT DOES NOT RECEIVE
REMUNERATION TO (i) ENCODE VIDEO IN COMPLIANCE WITH THE
AVC STANDARD ("AVC VIDEO") AND/OR (ii) DECODE AVC VIDEO
THAT WAS ENCODED BY A CONSUMER ENGAGED IN A PERSONAL
ACTIVITY AND/OR WAS OBTAINED FROM A VIDEO PROVIDER
LICENSED TO PROVIDE AVC VIDEO. NO LICENSE IS GRANTED OR
SHALL BE IMPLIED FOR ANY OTHER USE. ADDITIONAL
INFORMATION MAY BE OBTAINED FROM MPEG LA, L.L.C. SEE
HTTP://WWW.MPEGLA.COM
```

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.]
 */
```

PCRE2

```
PCRE2 LICENCE
-------------
PCRE2 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.
Releases 10.00 and above of PCRE2 are distributed under the terms of the "BSD"
licence, as specified below, with one exemption for certain binary
redistributions. The documentation for PCRE2, supplied in the "doc" directory,
is distributed under the same terms as the software itself. The data in the
testdata directory is not copyrighted and is in the public domain.
The basic library functions are written in C and are freestanding. Also
included in the distribution is a just-in-time compiler that can be used to
optimize pattern matching. This is an optional feature that can be omitted when
the library is built.
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) 1997-2018 University of Cambridge
All rights reserved.
PCRE2 JUST-IN-TIME COMPILATION SUPPORT
--------------------------------------
Written by:       Zoltan Herczeg
Email local part: hzmester
Email domain:     freemail.hu
Copyright(c) 2010-2018 Zoltan Herczeg
All rights reserved.
STACK-LESS JUST-IN-TIME COMPILER
--------------------------------
Written by:       Zoltan Herczeg
Email local part: hzmester
Email domain:     freemail.hu
Copyright(c) 2009-2018 Zoltan Herczeg
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 notices,
      this list of conditions and the following disclaimer.
    * Redistributions in binary form must reproduce the above copyright
      notices, 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 names of any
      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.
EXEMPTION FOR BINARY LIBRARY-LIKE PACKAGES
-----------------------------------------
The second condition in the BSD licence (covering binary redistributions) does
not apply all the way down a chain of software. If binary package A includes
PCRE2, it must respect the condition, but if package B is software that
includes package A, the condition is not imposed on package B unless it uses
PCRE2 independently.
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.
```

Treeview

```
Copyright (C) 2006 Conor O'Mahony (gubusoft@gubusoft.com)
All rights reserved.
This application includes the TreeView script.
```

You are not authorized to download and/or use the TreeView source code from this application for your own purposes. For your own FREE copy of the TreeView script,
please visit the http://www.treeview.net Web site.
THIS SOFTWARE IS PROVIDED "AS IS" AND WITHOUT ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, WITHOUT LIMITATION, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS
FOR A PARTICULAR PURPOSE.
If Customer is using the free version of SOFTWARE, Customer must ensure that the "JavaScript Tree Menu" link at the top of the TreeView is visible and readable in their
Web page or application.
Customer may not harm the GUBUSOFT intellectual property rights using any media or via
any electronic or other method now known or later discovered.
Customer may not use the GubuSoft name, the name of the TreeView author, or the names
of any source code contributors to endorse or promote products derived from this SOFTWARE without specific prior written permission.
Customer may not utilize the SOFTWARE in a manner which is disparaging to GUBUSOFT.

Turf

The MIT License (MIT)
Copyright (c) 2019 Morgan Herlocker
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.

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>

x264

The x264 software library is used under commercial license from x264, LLC

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