PRIMER
Version 10.0

Oasys Ltd
The Software house of Arup

For help and support from OASYS Ltd please contact:

UK

Arup Group Ltd
The Arup Campus
Blythe Gate
Blythe Valley Park
Solihull
West Midlands
B90 8AE
United Kingdom
Tel: +44 (0) 121 213 3399
Fax: +44 (0) 121 213 3302
Email: dyna.support@arup.com
Web: www.oasys-software.com/dyna

China

Arup
39/F-41/F
Huai Hai Plaza
Huai Hai Road (M)
Shanghai
China 200031
Tel: +86 21 6126 2875
Fax: +86 21 6126 2882
Email: china.support@arup.com
Web: www.oasys-software.com/dyna

India

Arup
Plot 39, Ananth Info Park
Opp. Oracle Campus
HiTec City
Madhapur Phase II
Hyderabad 500081
India
Tel: +91 40 4436 9797/98
Email: india.support@arup.com
Web: www.oasys-software.com/dyna

or contact your local Oasys Ltd distributor

Preamble

Abstract

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

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

PRIMER is designed to solve these problems. It is capable of reading, processing and writing out the entire LS-DYNA version 940 onwards keyword input deck, with no exceptions or omissions: no information is lost during processing. It will also read and write several other common formats.

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

In addition PRIMER 10.0 provides several specialist features for LS-DYNA analysis. These are features such as occupant positioning, belt fitting, airbag folding, spotwelding, massing-up and de-penetration of contacts.

Development status

This manual documents PRIMER 10.0. The code is still being developed, and this version provides full compatibility with LS-DYNA release 971R4; it also new keywords in the recent LS-DYNA release 971R5 as documented in May 2010.

Memory requirements

Memory is allocated dynamically, so the amount required rises in proportion to the amount of data being manipulated. In release 9.2 tests show that memory required to read in and display data is:

- **32 bit version**: Approximately 500MBytes per 1,000,000 elements in model
- **64 bit version**: Approximately 750MBytes per 1,000,000 elements in model (Larger because "pointers" are now 64 bits)

Operations such as Model Merge, Spotwelding and Contact checking can easily double these requirements, and as the code gets more complex so - unfortunately - its memory consumption rises. Therefore we would recommend that users contemplating the purchase of new equipment should give serious consideration to buying 64 bit hardware if their typical model size during the life of that equipment is likely to exceed 2,000,000 elements. Please contact Oasys Ltd if you would like advice about specifying a computer for PRIMER usage.

Output devices

The code supports the following graphical devices:

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

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

Revision history

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

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

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

Rev 0 October 1998 Initial release of version 7.1 software

- New Version 7.1 features;

- Manual reorganisation for improved layout

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

Rev 0 January 2000 Initial release of version 8.0 software

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

Rev 0 October 2000 Interim release of version 8.0a software

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

Rev 0 April 2001: Release of version 8.1 software

- New Version 8.1 features;

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

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

Rev 0 November 2003: Release of version 9.0 software

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

Rev 0 November 2004: Release of version 9.1 software

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

Rev 0 March 2006:

Released with Primer 9.3 rc2:

Rev 0 January 2008:

Released with Primer 9.4:

Rev 0 Nov 2009:

Released with Primer 10.0

Rev 0 May 2011:

Text conventions used in this manual

Typefaces

Three different typefaces are used in this manual:

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

Notation

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

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

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

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

Round To show optional items during input, for example:

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

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

```
Give new value (10) :

Give model number (12) :
```

Square To show advisory information at computer prompts, eg

```
Give filename: [.key] :

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

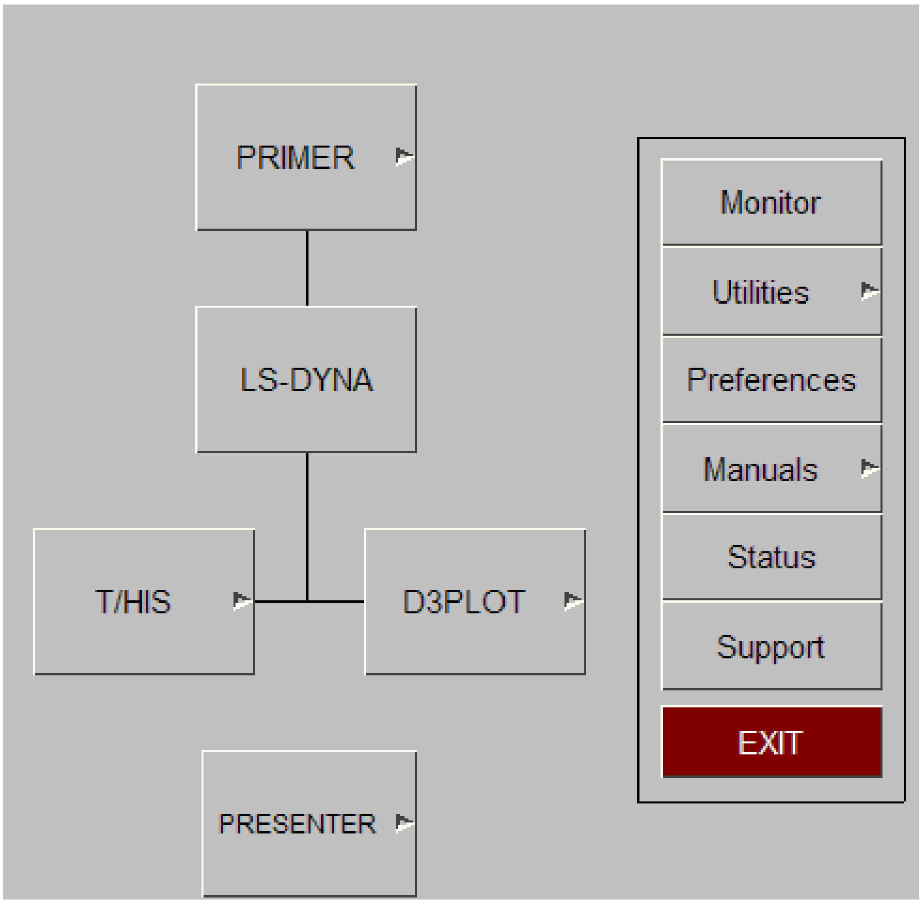
Also to show implicit commands, eg

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

1 Running PRIMER

1.1 Starting the code

PRIMER is run from the PRIMER button in the Shell:



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

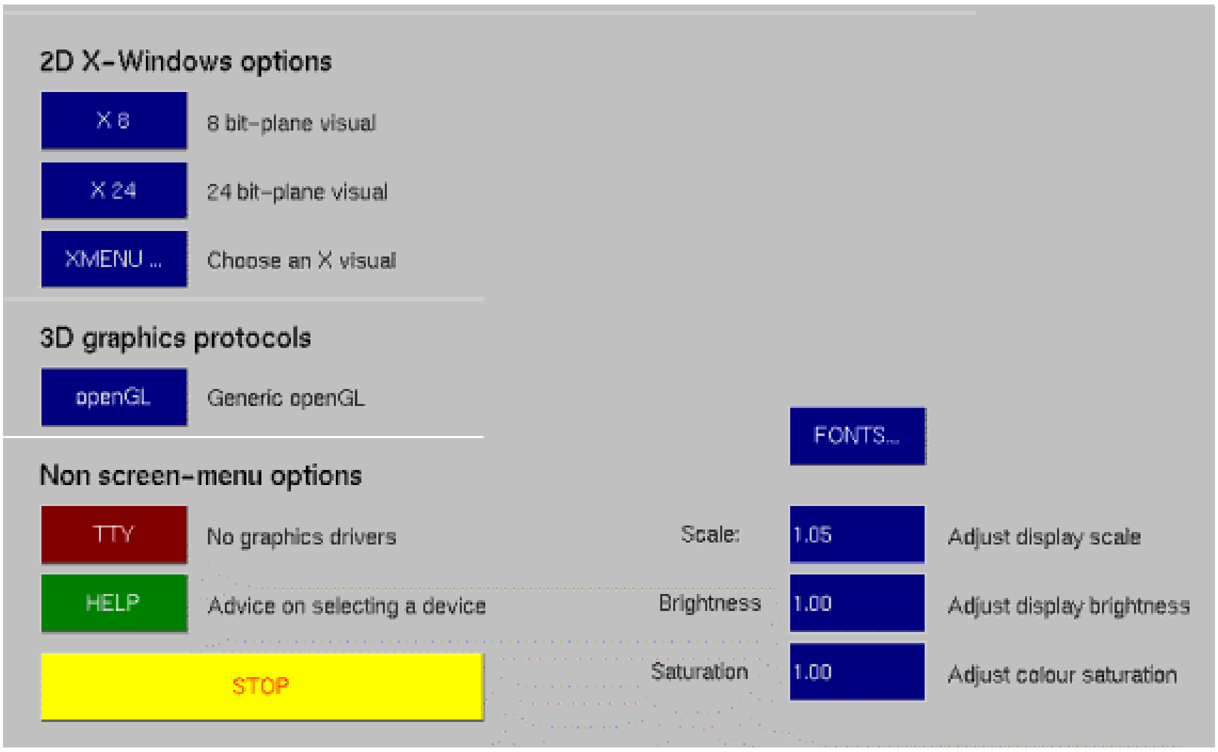
If your system has been customised locally you may have to use some other command or icon: consult your system manager in this case.

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

1.2 Selecting a graphics device

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

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



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

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

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

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

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

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

The other command options on this panel are:

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

TTY Invokes text-only mode with no graphics or menus

HELP Gives online advice about using this panel

STOP Stops the PRIMER session.

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

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

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

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

`setenv DISPLAY_FACTOR 1.2` (C shell syntax)

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

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

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

Brightness Controls screen menu colour brightness.

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

Saturation Controls screen menu colour saturation.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

```
xhost +
```

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

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

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

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

1.4 Running PRIMER in "Text-Only" mode.

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

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

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

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

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

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

1.4.1 Starting PRIMER in Text-Only mode.

This may be done in one of two ways:

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

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

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

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

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

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

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

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

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

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

1.4.3 Using "command files"

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

These are invoked using the "**-cf=<*filename*>**" option on the command line, and they operate as follows:

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

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

```
$<pathname>/primer93.exe -d=batch -cf=my_command_file -exit
```

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

1.5 Command Line arguments

Function	Format	Options
Setting the graphics device By default no graphics device is defined, and the device selection panel is mapped. These options are useful if you want to bypass the device selection panel and always start PRIMER with a particular graphics driver.	`-d=`*`<device>`*	**`-d=opengl`** - Use OpenGL 3D graphics **`-d=x24`** - 24 bit-plane X-Windows graphics **`-d=x8`** - 8 bit-plane X-Windows graphics **`-d=x`** - X24 if available, otherwise X8 **`-d=default`** - Whichever is available in the order OpenGL, X24, X8 **`-d=batch, -d=tty`** - No graphics - text-only mode
Specifying "full screen" mode on startup Normally PRIMER occupies about 70% of the display when it starts, the "maximise" argument changes this to become the full screen.	**`-maximise`**	
Specifying which screen PRIMER starts up in on a multi-screen display By default PRIMER will start in the top right quadrant of the desktop, occupying about 3/4 of the (whole) display, unless **-maximise** is used.	`-placement=`*`<where>`*	<*where*> may be **LEFT** or **RIGHT** On a dual screen display **LEFT** or **RIGHT** will start PRIMER on the relevant monitor. To start maximised on that monitor combine this with **-maximise**. (But see notes (1) and (2) below).
Defining a command file name By default no command file is assumed.	`-cf=`*`<filename>`*	<*filename*> can be any text file containing valid commands.
Requesting termination at the end of a command file This is ignored if no command file is defined	**`-exit`**	
Requesting batch creation of ZTF and group files This generates both <*filename*>.ztf and groups <*filename*>.bin files for subsequent post-processing in D3PLOT. When combined with "**`-d=batch`**" then: • ZTF and group (.bin) files are created, then PRIMER exits • No licence to run PRIMER is required	`-ztf=`*`<filename>`*	<*filename*> must be a valid LS-Dyna keyword (.key) file, with or without the ".key" extension.
Specifying the directory in which to start. PRIMER will make this your "current working directory" - all files which do not have explicit pathname prefixed are assumed to be in this directory.	`-start_in=`*`<directory>`*	<*directory*> must be a valid directory name on your system.
Specifying a custom "oa_pref" file This causes an extra, optional "oa_pref" file to be read.	`-pref=`*`<filename>`*	<filename> must be a valid "oa_pref" file. If it has no explicit pathname prefix it is assumed to be in the **`$OASYS`** directory. Any legal filename may be used.

Note 1: `-placement=<where>` may not be the best solution on Windows platforms.

Left/right placement works well when the desktop is a "Single Logical Screen" equal to the combined width of the two monitors, which is a typical configuration found on Linux and Unix platforms, since it stops the PRIMER window spanning the two screens on startup.

On Windows platforms there are often more sophisticated options available from the graphics card vendor, typically NVidia's "NView" product or ATI's equivalent, which allow more intelligent placement of application windows. For example the Window's desktop can be configured to open applications on a particular screen. In this case the **-placement** option may conflict with the Windows desktop management, and it will be best to rely on the latter instead.

Note 2: There are further placement options to cope with 2x2 matrices of displays.

There are in fact also **`TOP`** and **`BOTTOM`** placement arguments too, which deal with the case of monitors stacked vertically rather than horizontally, and these may be combined with **`LEFT`** and **`RIGHT`** using an underscore, for example **`-placement=LEFT_TOP`**, to place the initial PRIMER window on a particular screen of a 2x2 matrix of displays.

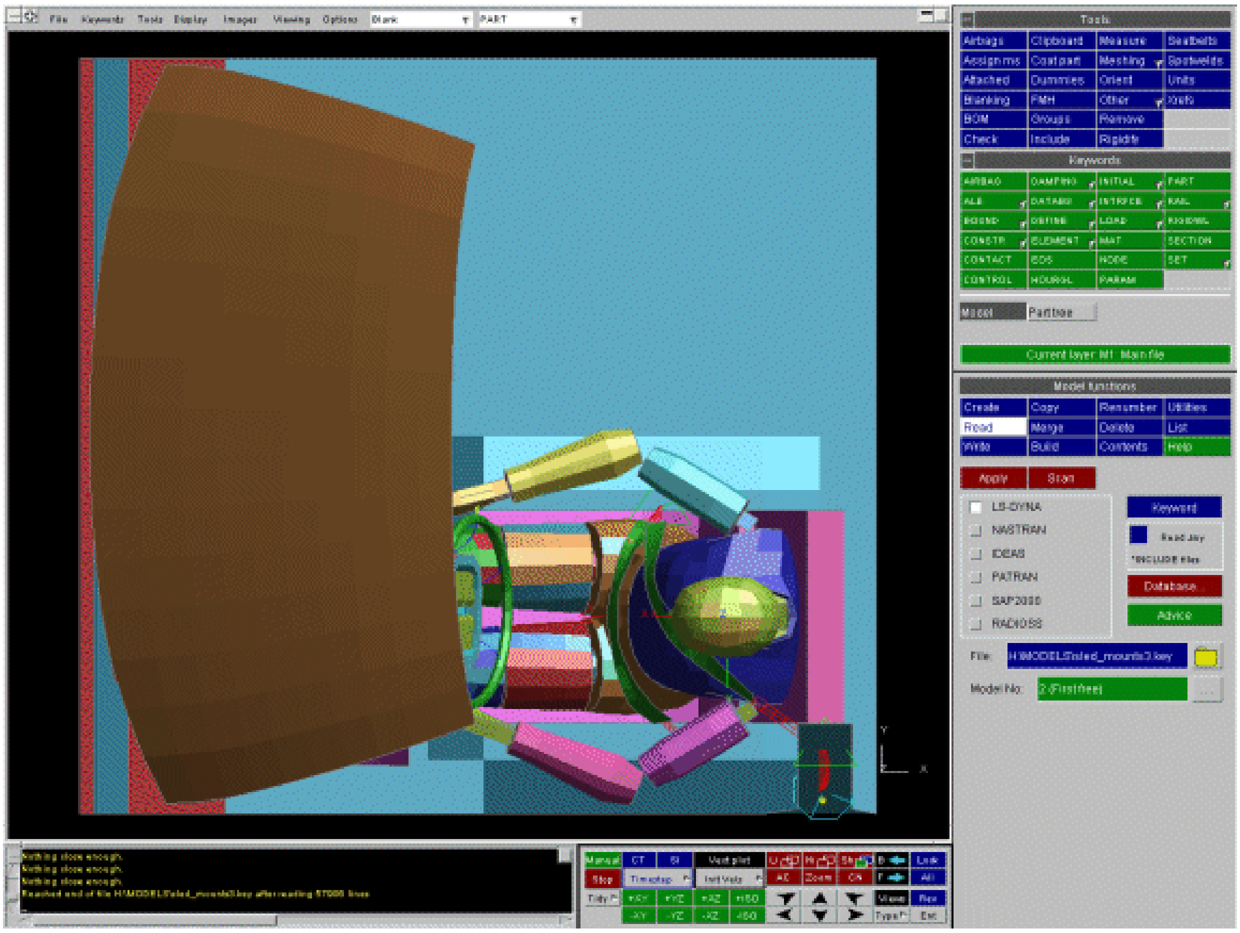
2 Using Screen Menus

2.0 USING THE PRIMER SCREEN MENU SYSTEM

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

2.1 Basic screen menu layout

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



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

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

<table>
<tr><td>Top menus</td><td colspan="2">Quick-pick control</td><td rowspan="2">Tools</td></tr>
<tr><td colspan="3" rowspan="4">Graphics area</td></tr>
<tr><td>Keywords</td></tr>
<tr><td>Menu tabs</td></tr>
<tr><td rowspan="2">Current menu panel</td></tr>
<tr><td>Dialogue and list area</td><td colspan="2">Viewing and Drawing Commands</td></tr>
</table>

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

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

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

2.2 Mouse and keyboard usage for screen-menu interface

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

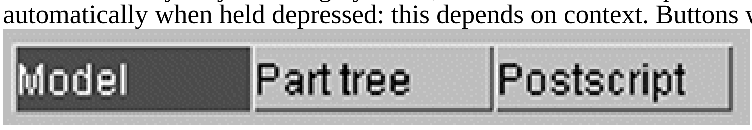
- "Popup" menus are invoked with the right mouse button; those in the Top menus, Tools and Keywords areas can also be invoked with the left mouse button.
- Text in the dialogue area and text boxes requires keyboard entry;
- Text strings saved in the cursor "cut" buffer may be "pasted" into dialogue areas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

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

2.2.1 Buttons

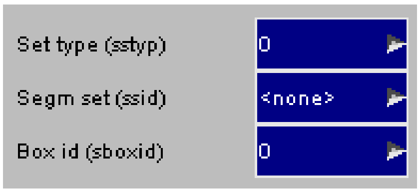
Screen buttons are depressed by clicking on them, but action only takes place when the mouse button is released, so it is safe to drag the (depressed) mouse around the screen.

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



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

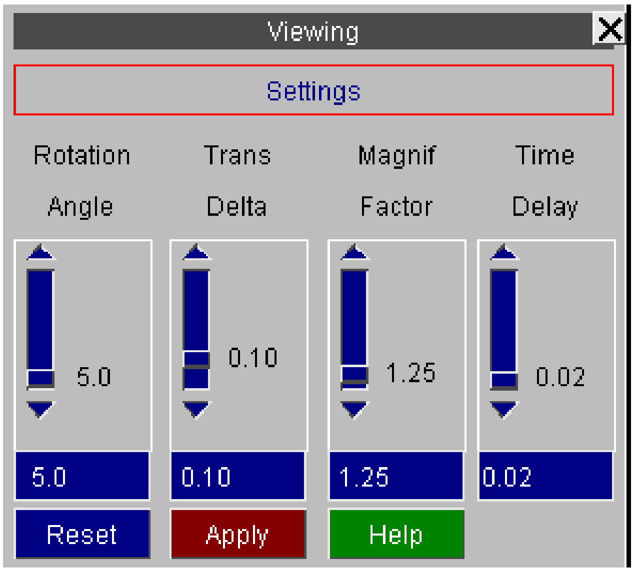
"Popup" window invocation: Buttons with an ">" symbol may be selected normally with the left mouse button, but if the *right* mouse button is depressed over them it will invoke a "popup" window. Move the cursor into this window to make a selection, or move elsewhere and click a button to deactivate the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



2.2.2 Sliders

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

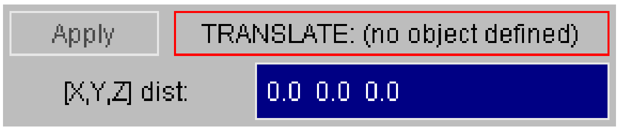
They may also be moved automatically by clicking on, and holding down, one of the arrows at either end. Using the left mouse button for this advances the slider by 1 unit, the middle button by 10, and the right button by 100.



2.2.3 Text boxes

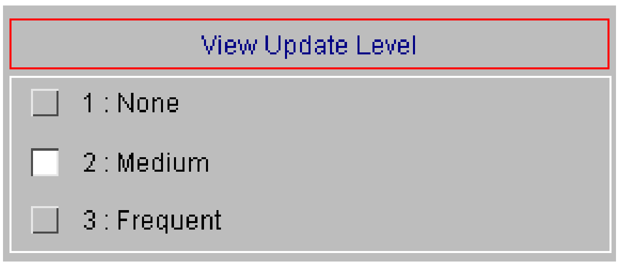
To enter text in a text box: first make it "live" by clicking on it, then type in text, then type `<return>` to enter the string. Clicking on a "live" box for a second time is exactly the same as typing `<return>`, so clicking twice on a box effectively enters its current contents. You can use the left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

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



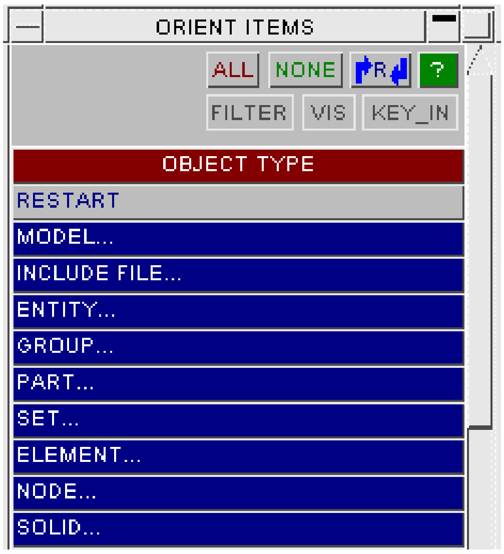
2.2.4 Radio boxes

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



2.2.5 Menu selections

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



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

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

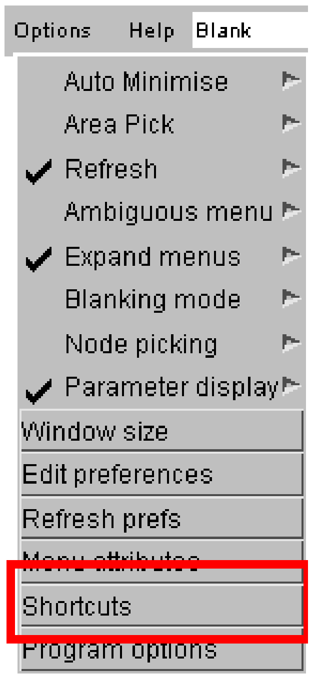
By default menus will expand horizontally when you move the mouse into them in order to show more of their contents. This is described in section 2.4.4.3 below

2.2.6 Shortcut keys

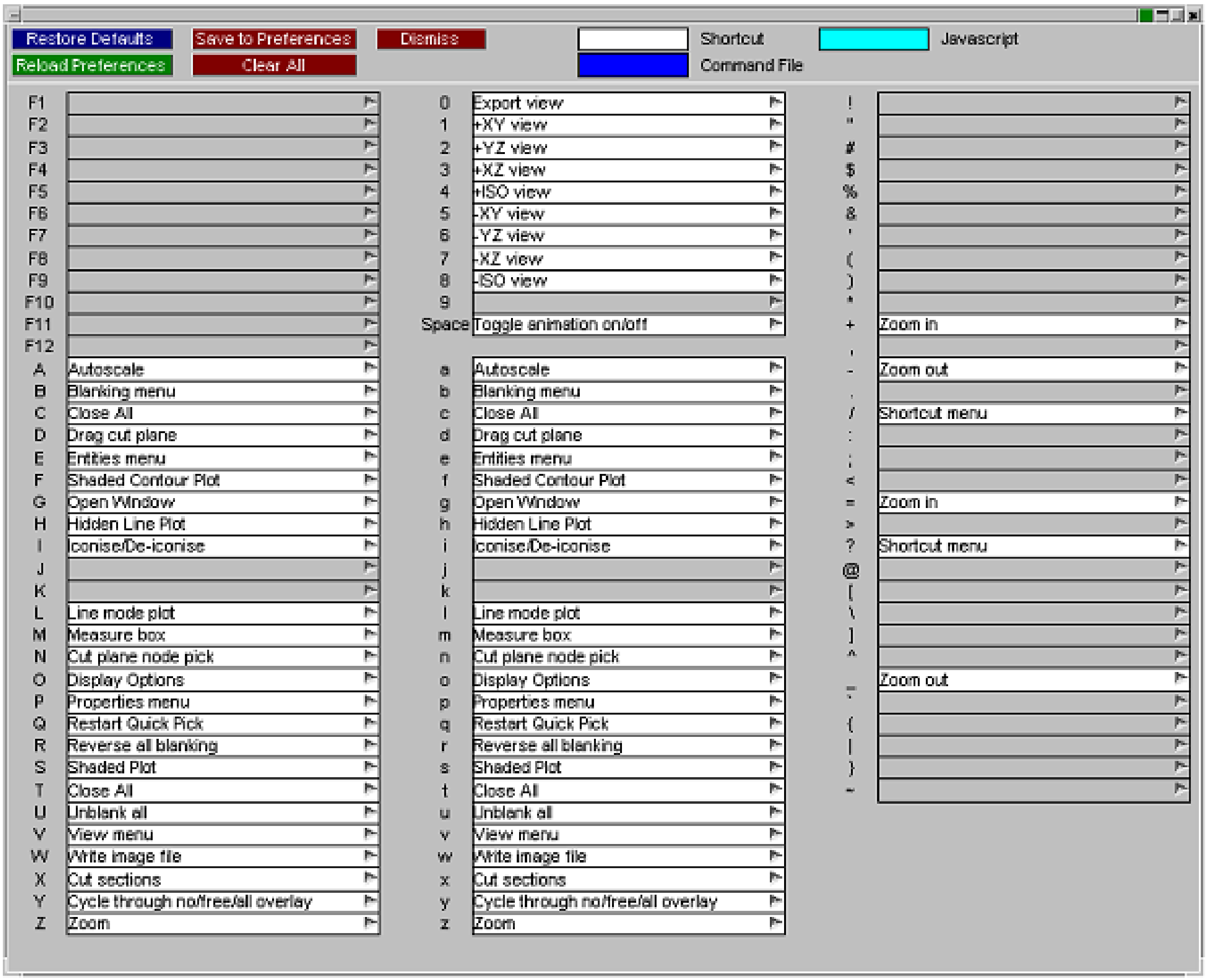
Some panels and actions can be accessed through pre-programmed shortcuts and from v9.4 the keys they are assigned to are customizable.

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

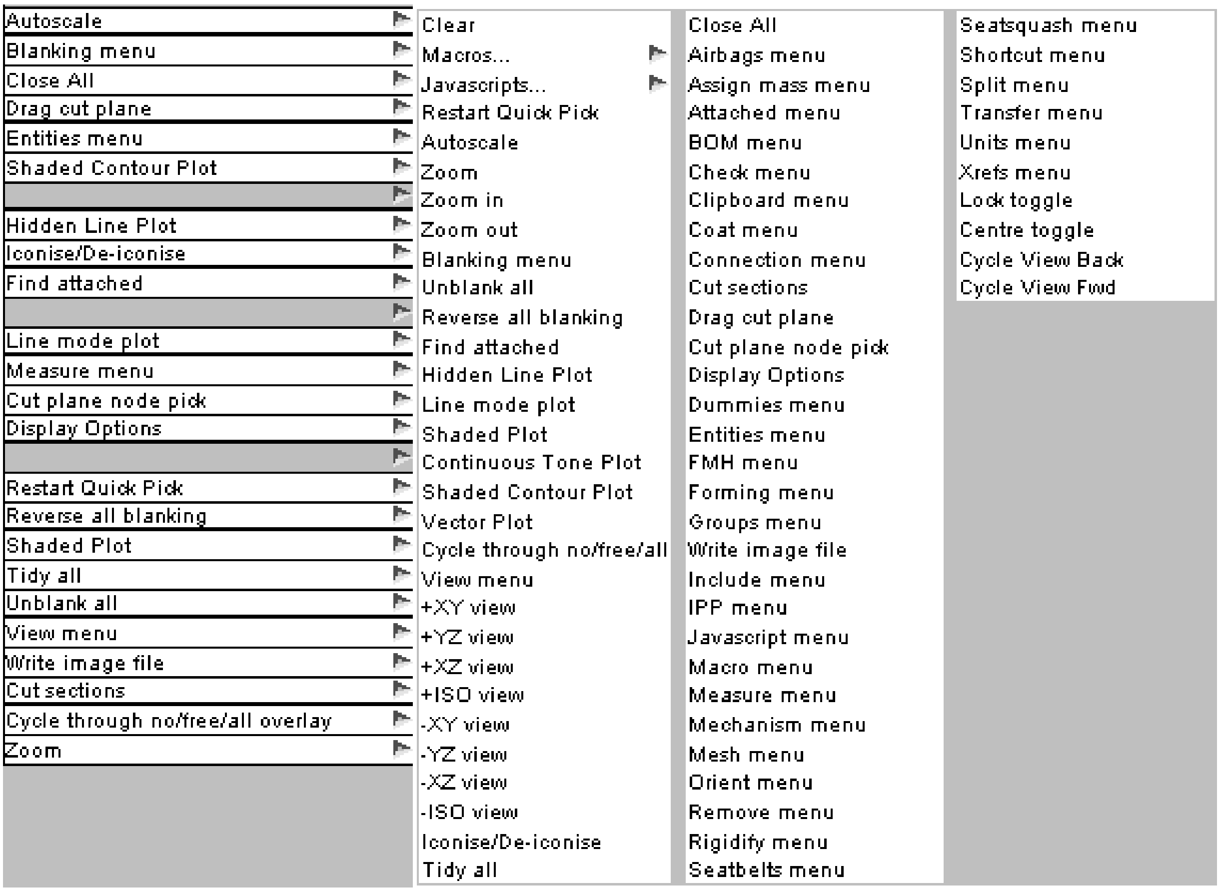
A listing of the available shortcuts and the keys they are assigned to can be brought up by pressing either the '?' key (by default) or accessing it through the Options top menu.



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


Restore Defaults
Save to Preferences
Dismiss
Shortcut
Javascript
Reload Preferences
Clear All
Command File
F1
F2
F3
F4
F5
F6
F7
F8
F9
F10
F11
F12
A Autoscale
B Blanking menu
C Close All
D Drag cut plane
E Entities menu
F Shaded Contour Plot
G Open Window
H Hidden Line Plot
I Iconise/De-iconise
J
K
L Line mode plot
M Measure box
N Cut plane node pick
O Display Options
P Properties menu
Q Restart Quick Pick
R Reverse all blanking
S Shaded Plot
T Close All
U Unblank all
V View menu
W Write image file
X Cut sections
Y Cycle through no/free/all overlay
Z Zoom
0 Export view
1 +XY view
2 +YZ view
3 +XZ view
4 +ISO view
5 -XY view
6 -YZ view
7 -XZ view
8 -ISO view
9
Space Toggle animation on/off
a Autoscale
b Blanking menu
c Close All
d Drag cut plane
e Entities menu
f Shaded Contour Plot
g Open Window
h Hidden Line Plot
i Iconise/De-iconise
j
k
l Line mode plot
m Measure box
n Cut plane node pick
o Display Options
p Properties menu
q Restart Quick Pick
r Reverse all blanking
s Shaded Plot
t Close All
u Unblank all
v View menu
w Write image file
x Cut sections
y Cycle through no/free/all overlay
z Zoom
+ Zoom in
- Zoom out
/ Shortcut menu
= Zoom in
? Shortcut menu
_ Zoom out

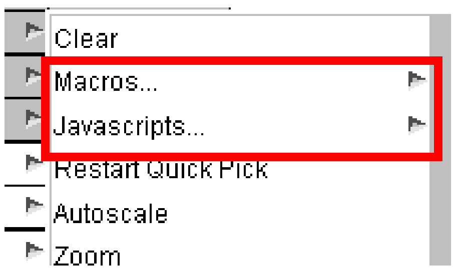
The image above shows the default key assignations, but each key may be programmed to use any of the predefined functions as shown in this screen-grab:



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

Macros are described in section 6

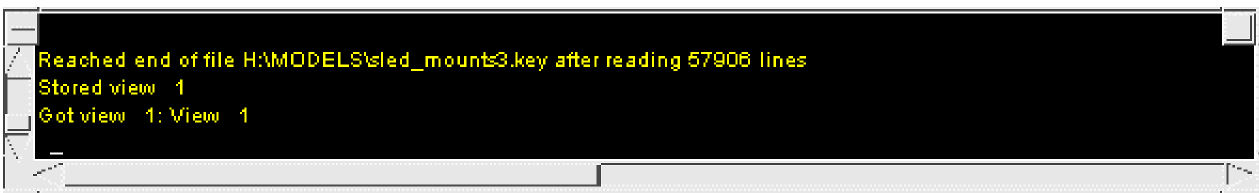
Javscripts are described in section 10



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

2.3 Dialogue input in the screen menu interface

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



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

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

2.4 Window management in the screen interface

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

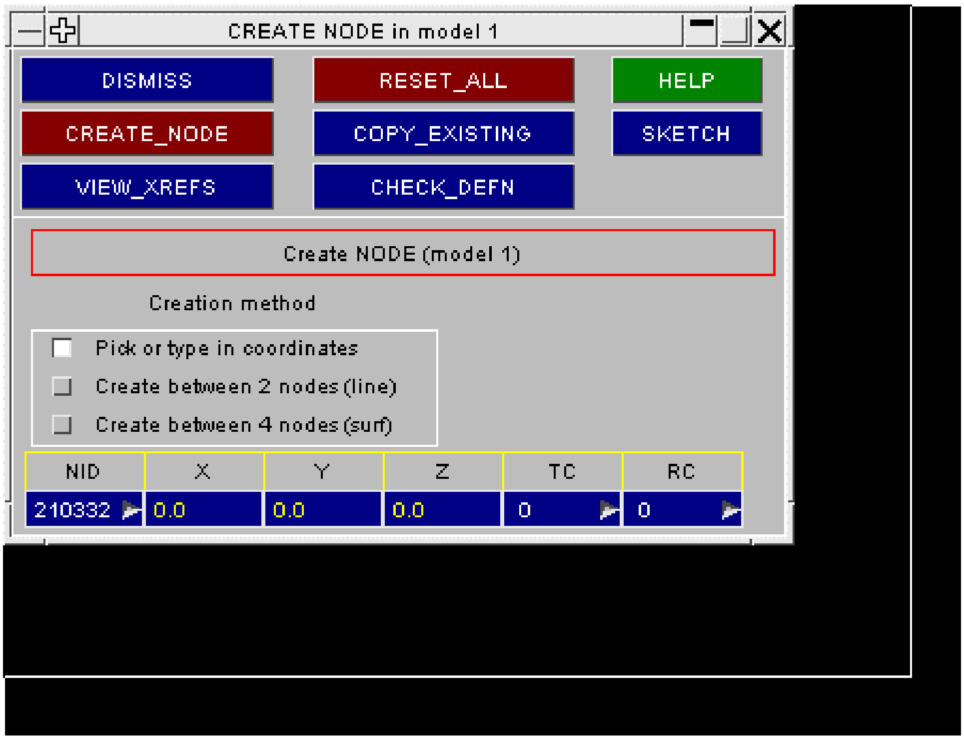
Moving, resizing and scrolling of windows is based on the conventions used in the Motif Window Manager.

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

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

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

This example shows a sub-window being resized:



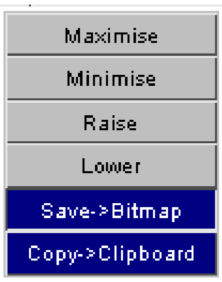
The user has chosen to drag the bottom right corner out.

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

CREATE NODE in model 1

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

2.4.1 Popup menus for window management:



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

MAXIMISE expands the window to its full size (in the case of the dialogue and graphic areas this is taken as the entire PRIMER window, for other sub-windows the minimum size such that no scroll bars are required).

MINIMISE collapses the window to a bar. This will be positioned where the top right -hand corner of the window was.

If a window has already been maximised the option to do so will be replaced by **RESTORE** or if minimised by **EXPAND**. These will undo the effect of maximisation and minimisation respectively.

RAISE raises the window to the front of the "stacking order", obscuring any others.

LOWER lowers the window to the bottom of the stacking order, allowing other to obscure it.

SAVE->BITMAP saves this window (and its borders) as a "bitmap" (.bmp) file.

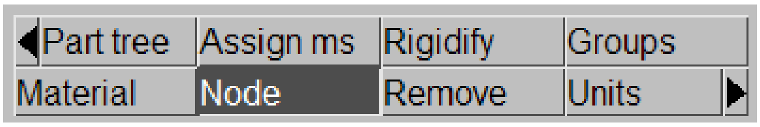
Copy->Clipboard For "text" windows, ie dialogue input box and listing boxes, the complete text in the window is copied to the clipboard

On Windows platforms only other windows are saved as a bitmap image on the clipboard. (This is not feasible on X11-absed window managers, typically Unix and Linux, because there is no common protocol for exchange of images.)

2.4.2 Use of Menu Tabs

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

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



2.4.3 Iconisation of Menus

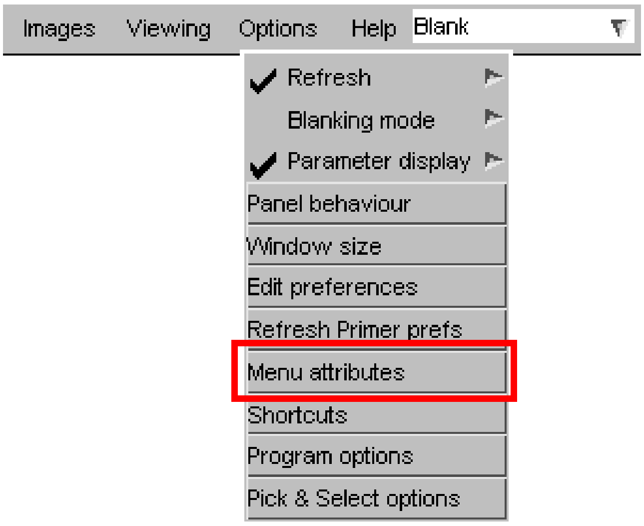
Menus can be iconised by clicking . Click  to restore them.
Alternatively a list of options is produced by clicking on the button in the top-left corner.
Pressing **I** will iconise all windows or if restore them all if they are all already iconised
It is also possible from **Options** > **Auto Minimise**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

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

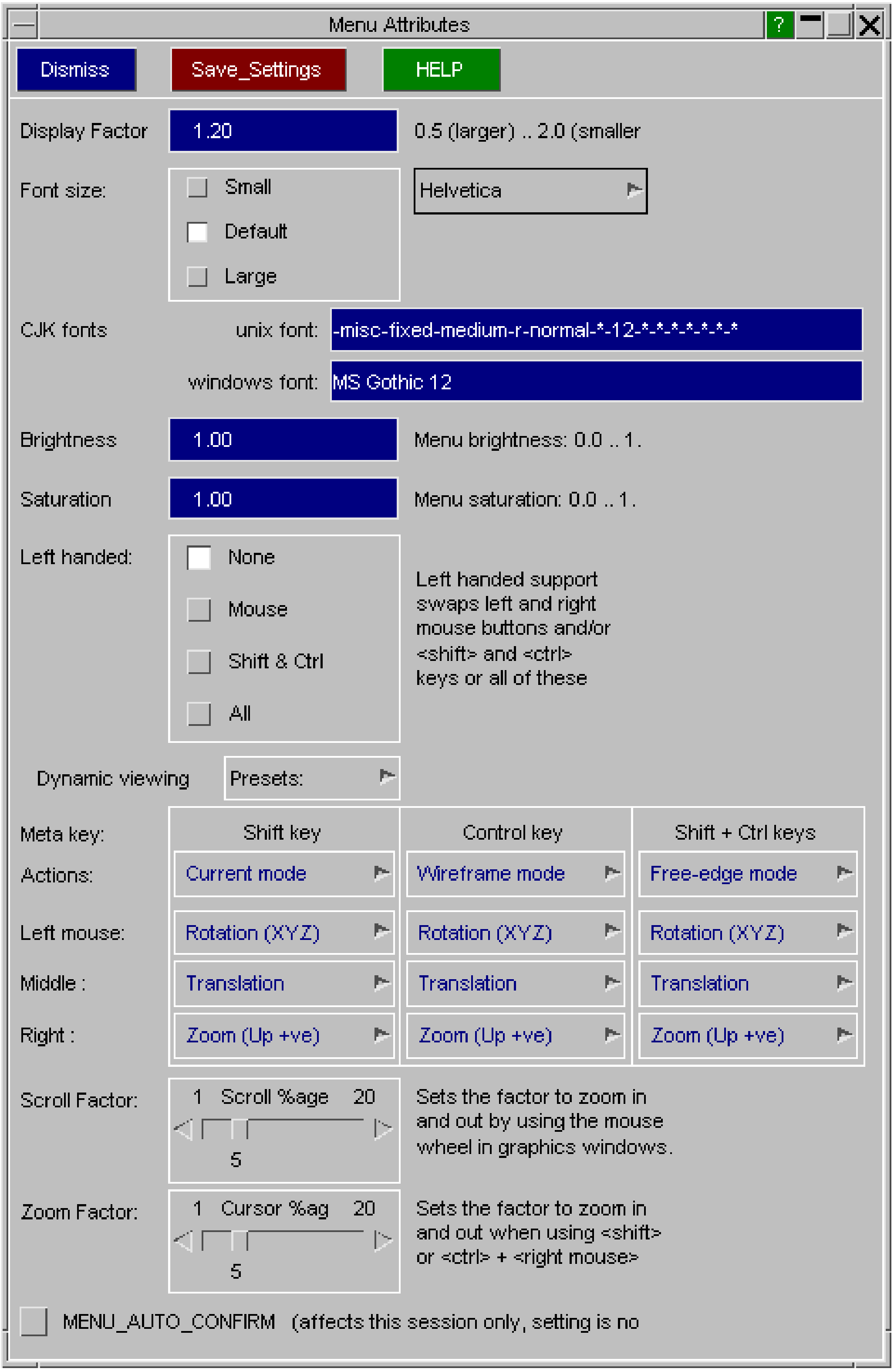
2.4.4 Customising the User Interface

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

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



Gives the menu attributes panel:


Menu Attributes
Dismiss
Save_Settings
HELP
Display Factor
1.20
0.5 (larger) .. 2.0 (smaller
Font size:
Small
Default
Large
Helvetica
CJK fonts
unix font:
-misc-fixed-medium-r-normal-*-12-*-*-*-*-*-*
windows font:
MS Gothic 12
Brightness
1.00
Menu brightness: 0.0 .. 1.
Saturation
1.00
Menu saturation: 0.0 .. 1.
Left handed:
None
Mouse
Shift & Ctrl
All
Left handed support swaps left and right mouse buttons and/or <shift> and <ctrl> keys or all of these
Dynamic viewing
Presets:
Meta key:
Shift key
Control key
Shift + Ctrl keys
Actions:
Current mode
Wireframe mode
Free-edge mode
Left mouse:
Rotation (XYZ)
Rotation (XYZ)
Rotation (XYZ)
Middle :
Translation
Translation
Translation
Right :
Zoom (Up +ve)
Zoom (Up +ve)
Zoom (Up +ve)
Scroll Factor:
1 Scroll %age 20
5
Sets the factor to zoom in and out by using the mouse wheel in graphics windows.
Zoom Factor:
1 Cursor %ag 20
5
Sets the factor to zoom in and out when using <shift> or <ctrl> + <right mouse>
MENU_AUTO_CONFIRM (affects this session only, setting is no

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

Dynamic viewing

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

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

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

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

Normal	will use the current display mode
Wireframe	only the line vectors in the current display mode
Free-edge	special "free edge lines only" display mode

Dynamic rotation options, assigned to mouse buttons

Rotate XYZ	traditional D3PLOT behaviour, rotates in XY if cursor's initial position is in centre 2/3rd of screen, otherwise about Z
Rotate XY	rotates about screen XY only, regardless of where the cursor's initial position
Rotate Z	rotates about screen Z only, regardless of cursor initial position
Rotate Sphere	free rotation about any of XYZ, like grabbing a point in a virtual sphere and dragging it

Dynamic translation options, assigned to mouse buttons

Translate	model follows cursor movement in screen XY plane

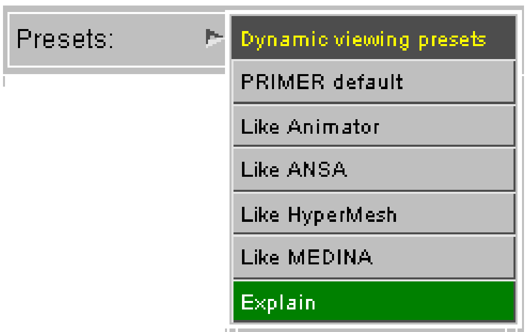
Zoom options, assigned to mouse buttons

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

Presets

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

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



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

Scroll factor

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

Zoom factor	Determines how rapidly the <meta key + mouse key> dynamic zoom operations above work. Again this is best set by trial and error.
MENU_ AUTO_ CONFIRM	This is a special setting designed mainly for "batch" style usage, and it controls how "popup" windows that normally wait for acknowledgement from the user should respond. If it is switched on then these windows will assume that the user has clicked the default action (usually "OK") and continue operation without waiting. This can be useful when replaying scripts, but it is not recommended for normal interactive usage.

Saving Menu Attributes settings

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

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

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

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

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

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

2.4.4.2 Refresh: Controlling backing store redraws

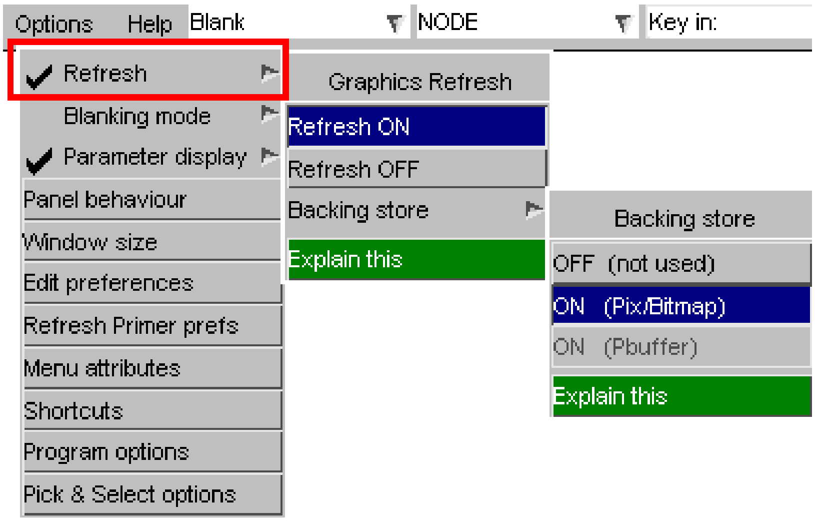
Graphics images in PRIMER may become quite complex, and therefore slow to redraw. On copmuters that do not support overlay planes in their graphics hardware (typically PCs), any window placed in front of the graphics window and then removed will leave a black hole behind, requiring the graphics window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

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

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

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

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

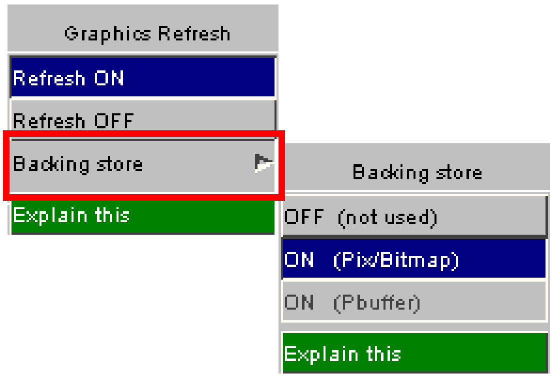


At the simplest level you can turn expose event refreshes **OFF**, using the second level menu shown here.

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

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

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



Two methods of providing backing store are available:

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

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

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

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

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

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

Saving Backing Store Redraw Settings

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

The backing store method is controlled by the oa_pref option:

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

Graphics refreshes themselves may be turned on or off by:

```
primer*graphics_refresh: off | on
```

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

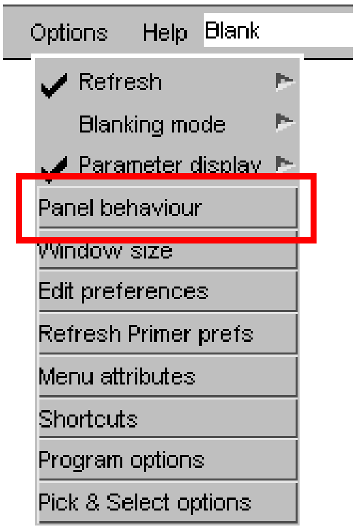
`PRIMER_NO_PBUFFER true` *which will turn off PBuffer usage*

and

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

All these options are listed in Appendix XIII

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

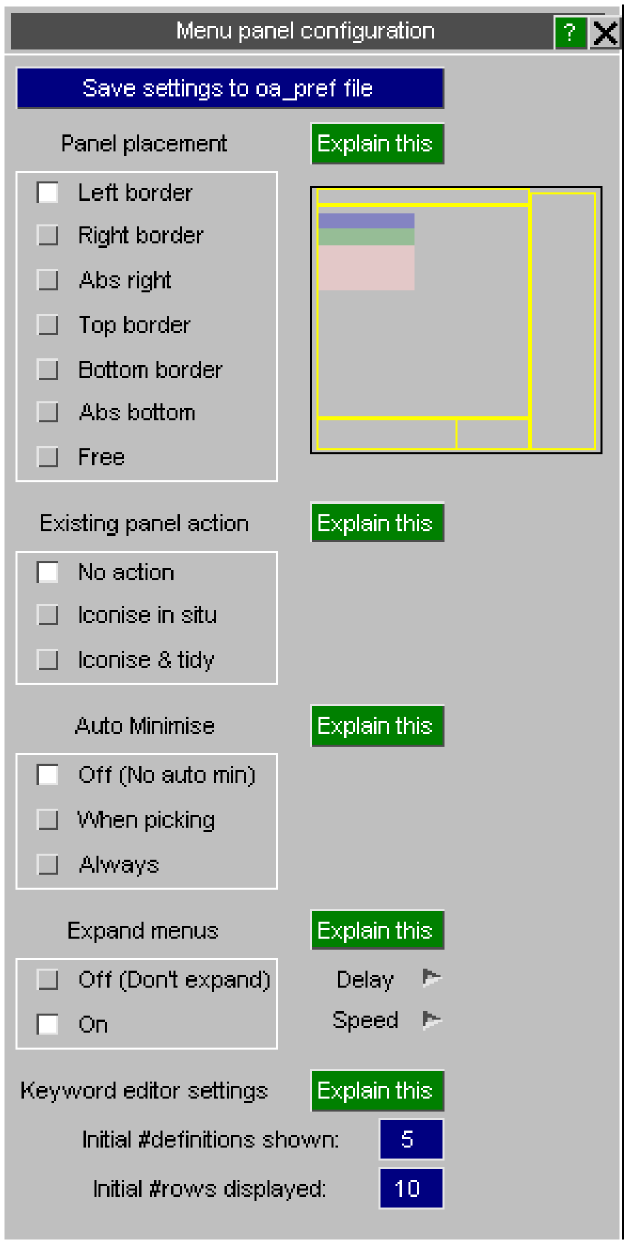


Selecting **[Options] Panel Behaviour** maps the **`Menu Panel Configuration`** panel, which controls the following:

Panel Placement	The placement of "floating" menu boxes on the display
Existing Panel Action	The action to be taken for existing floating menus when a new one is mapped
Auto Minimise	Whether to minimise floating panels when a picking operation is in force
Expand Menus	Whether to expand menu lists, the delay before doing this and their expansion speed
Keyword Editor settings	The initial state of keyword editor panels when first mapped

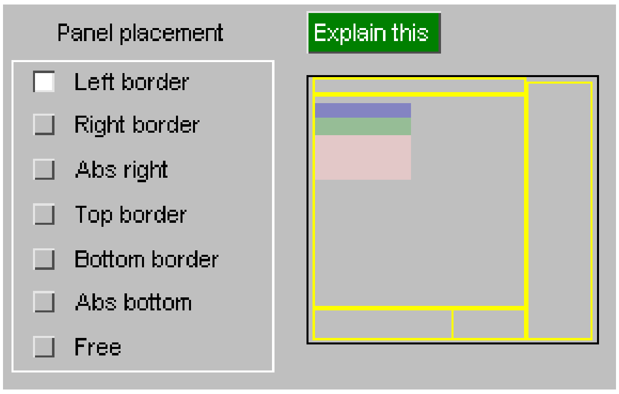
These options are described in more detail below.

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



Panel Placement

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



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

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

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

Existing panel action

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

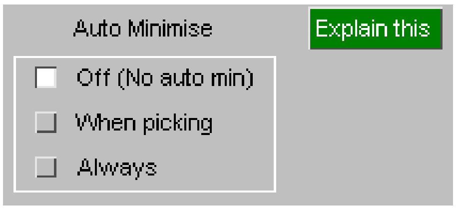
There are three options

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

Auto-minimise

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

There are three options



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

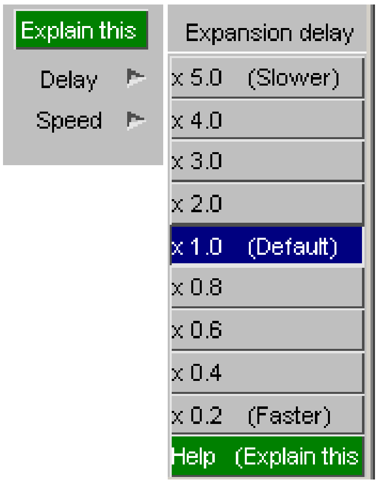
When picking When cursor picking (other than the global "quick pick" operation) is active then a panel will automatically minimise itself when you move the cursor out of it into the graphics window. The panel will be restored automatically if you move the mouse back over its icon, or when the picking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

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

Expand menus

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

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



You can also control:

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

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

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

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

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

Keyword editor settings

Controlling the initial appearance of the generic keyword editor.

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

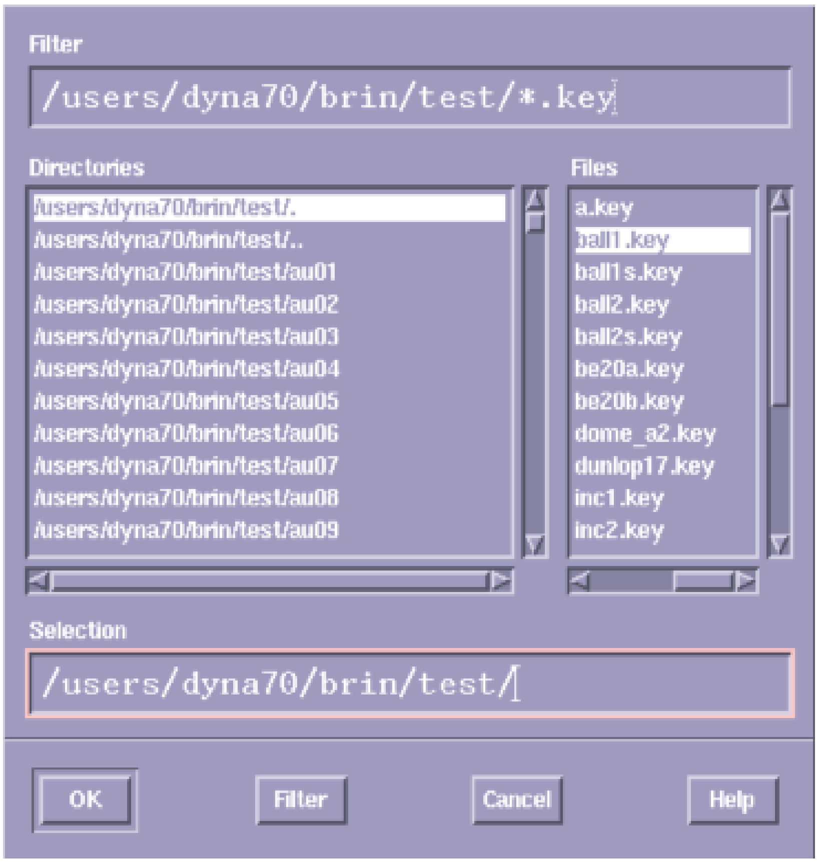
Initial #definitions shown	The maximum number of actual items that will be displayed.
Initial #rows displayed	The maximum number of rows of data that will be displayed. If the items being shown span several rows of data then this may limit the number of items actually shown. However there will never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

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

2.5 Using standard "file filter" boxes.

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

2.5.1 Standard "X11" (Motif) file filter box



The "filter" is the pathname and wildcard search pattern to be used. Here the pathname is

`/users/dyna70/brin/test` and the pattern is `*.key` (to look for a keyword input file).

The "directories" area lists those directories which exist under the current pathname. Here there are a number of directories eg "." (Unix for "this directory"), and "au01".

The "Files" area lists those files in the current directory which match the search pattern.

The "Selection" box shows the current selection.

You select a directory by clicking on it, then clicking on Filter in order to apply your selection. This updates the "Files" box accordingly. You then select a file by clicking on its name in the "Files" area, and finally on OK to make it your choice and return.

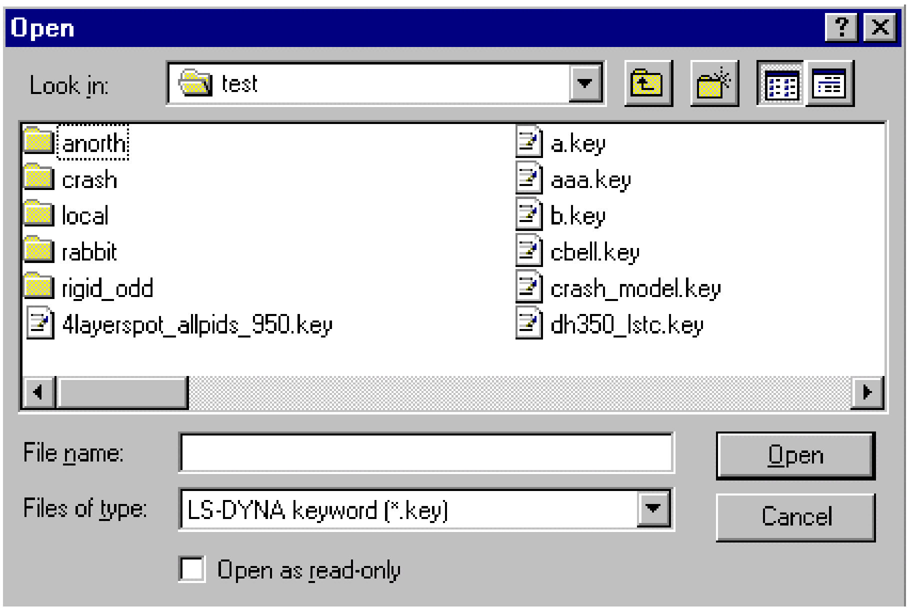
To go back up the directory tree you need to select the ".." (ie parent) directory. So to do this here you would have to select the **`/users/dyna70/brin/test/..`** line in the "Directories" area, then Filter to make it happen.

As an alternative to using Filter and OK you can double-click (quickly) on the relevant directory or pathname to make your selection.

Cancel Cancels this operation and returns with no file selected;

Help Provides context-dependent help and advice, then returns to file selection.

2.5.2 Standard "Windows" file filter box



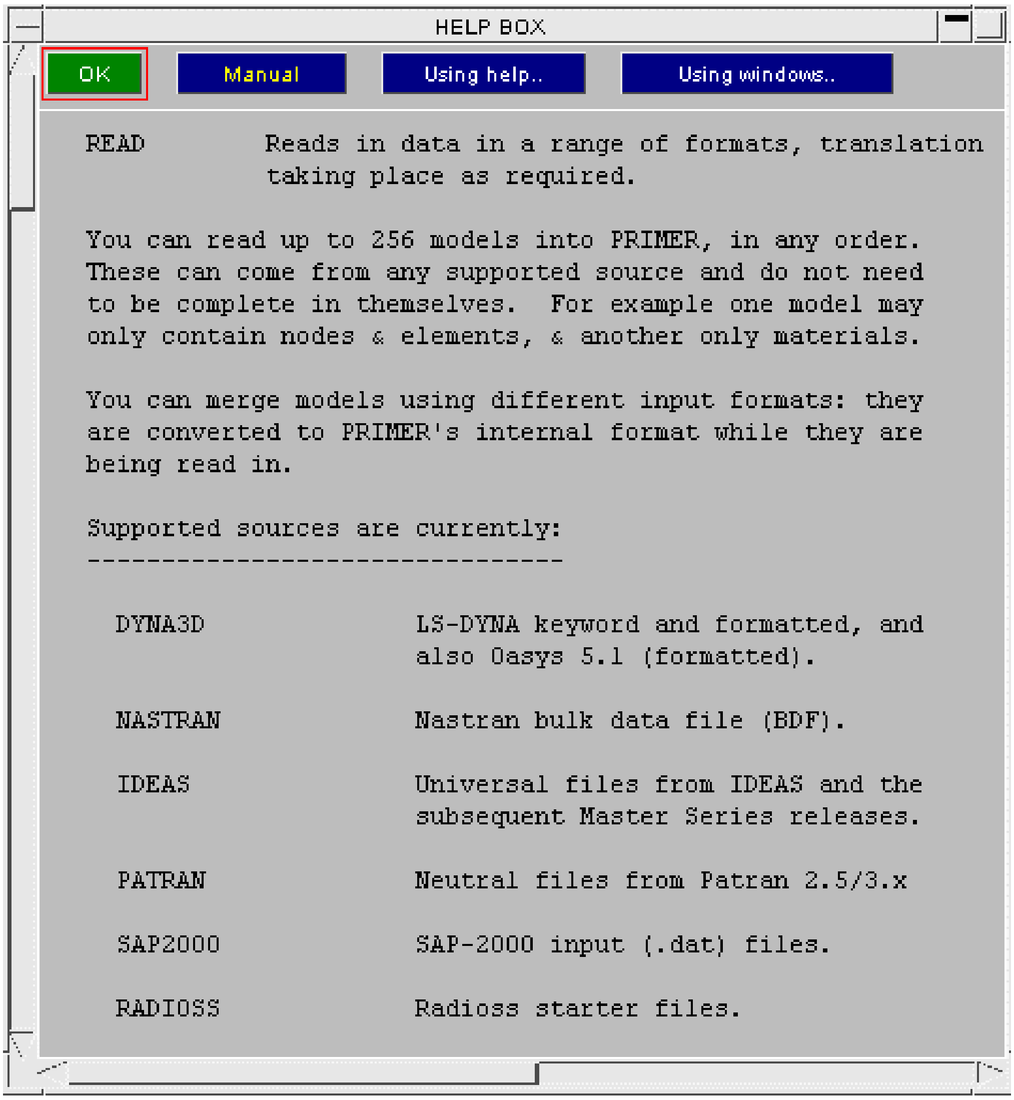
Double-click on the directory required, then on the filename you wish to open.

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

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

2.6 Obtaining Help and advice.

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



2.7 Error and Warning messages

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

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

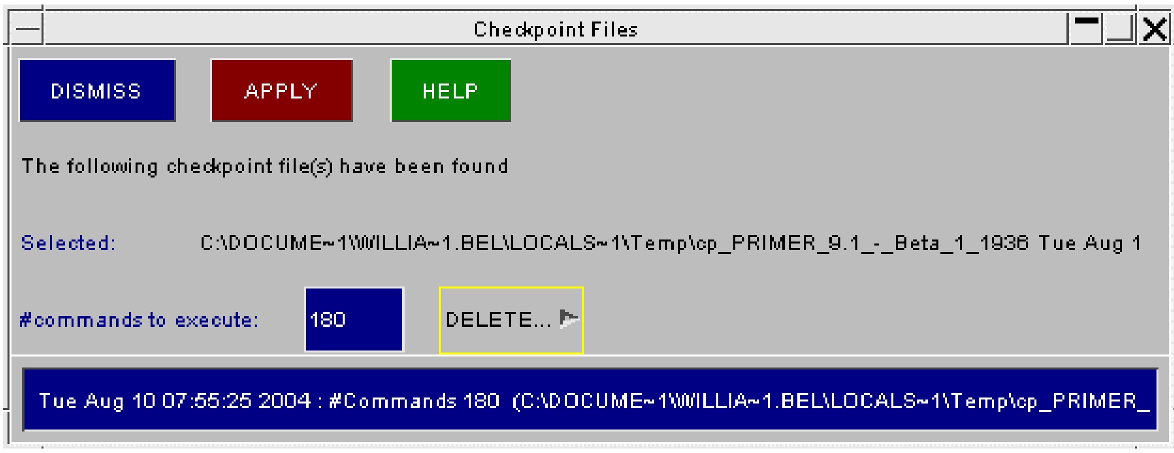
2.8 Checkpoint/Recovery files

All commands of your primer session are recorded in a binary (non-editable) checkpoint or recovery file (CP_PRIMER_9.1_xxx). If the session terminates normally the file is automatically deleted. If the software crashes, the file will be left behind.

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

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

In some cases, sending the checkpoint file and the input files to Oasys Ltd will assist in debugging the software.



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

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

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

2.9 Quick Pick Function

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

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

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

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

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

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

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

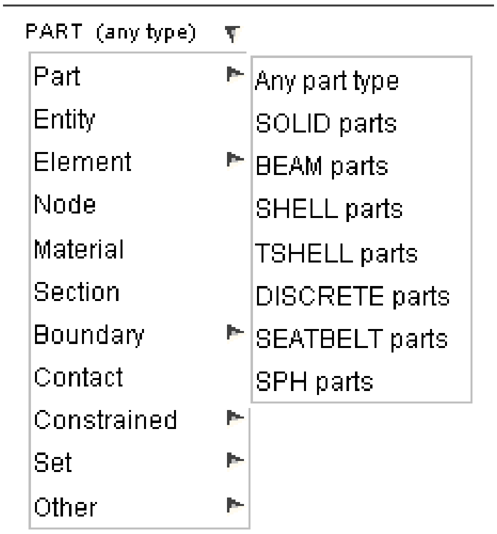
The first menu allows selection of what function is to be applied to left click operations. A summary of these is provided here:

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

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

Blank	Blanks the selected item(s). For more information on Blanking see section 4.5
Unblank	Unblanks the whole of the selected item(s) (only available for certain types)
Only	Blanks everything except the selected item(s)
Information	Provides a list of the item’s properties. (If multiple items are selected only the first is shown.)
Label	Labels the item on the screen dynamically, with a choice of attributes selected in the sub-menu
Edit	Maps the standard editing panel for the selected item(s). (To a maximum of 20 panels)
Keyword	Maps the standard Keyword Editor (see section 5.1.3), showing the selected item(s) only.
Colour	Sets the colour of the selected item(s) to the one set in the pop-up menu (accessed by >)
Transparency	Sets the transparency of the selected item(s) to the value set in the pop-up menu (accessed by >)
Plotting Mode	Sets the plotting mode (Shaded, Wireframe, etc) of the selected item(s) to that set in the pop-up menu (accessed by >)
Locate in Tree	Highlights the selected part(s) in the part tree. Add selects in addition to any currently selected in the part tree, only selects instead of any existing selection. See section 6.17 for more detail on the Part Tree.
Part Table	Produces a Part Table for the selected part. See section 6.16 for more detail.
Sketch	Sketches the selected item and locates cross-hairs at its centre (generally used via **Key in** <label>)
<item> **Details**	Opens the detailed information panel for that item.

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



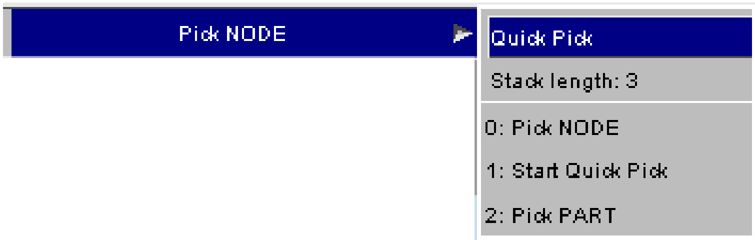
This shows the master popup menu of all possible types, and the second level menus under Element, Boundary, etc permit more detailed selection of type.

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

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

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

Whenever screen picking can be applied for a menu other than Quick Pick (see section 6.2) the menus in the top option box will be replaced by a box indicating what can be currently screen picked. Quick Pick control can be restored either by clicking the white cross in the top left of the graphics area, or from the drop-down in the Quick Pick control.



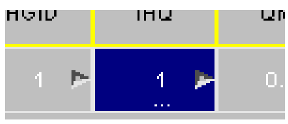
2.10 Using Parameters in Edit Panels.

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

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

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

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



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

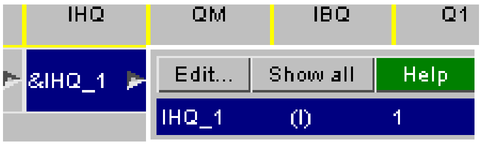
  In exactly the same way that you can type in numbers you can now also type in parameters using ***&NAME*** syntax.

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

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

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

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



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

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

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

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

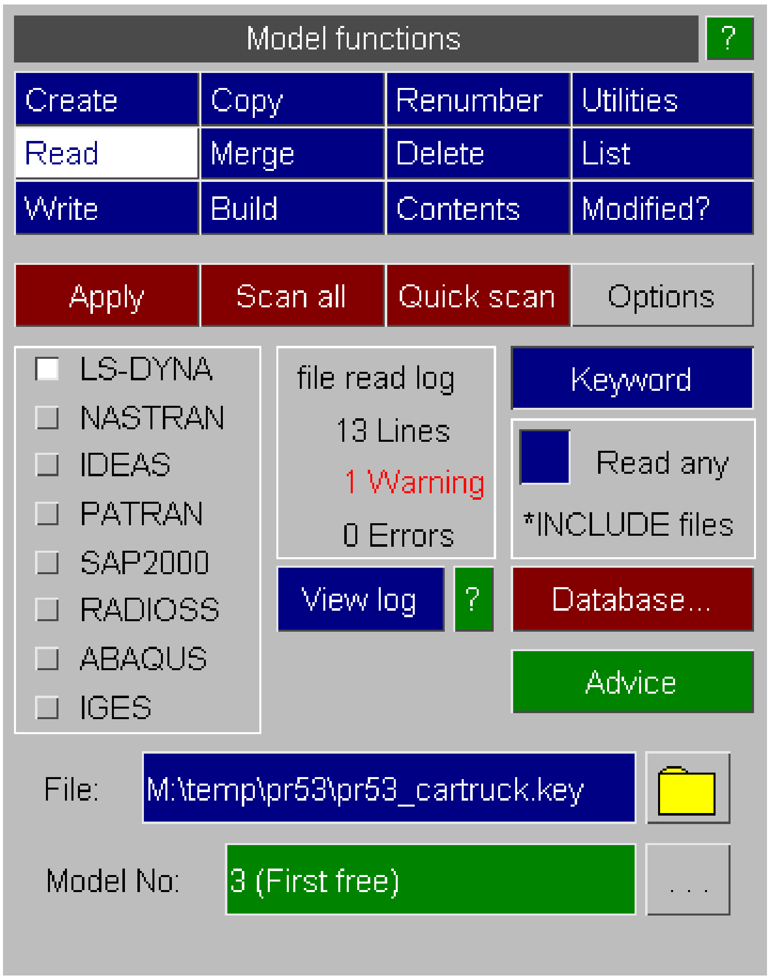
  These are evaluated on initial keyword input, and the correct value is used in the data field.
  They may also be created and edited interactively.

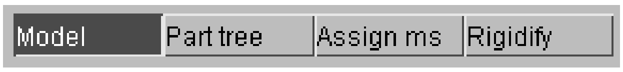
Full details of the processing and display of Parameters may be found in Section 5.

3 Model manipulation

Quick links to sections:

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



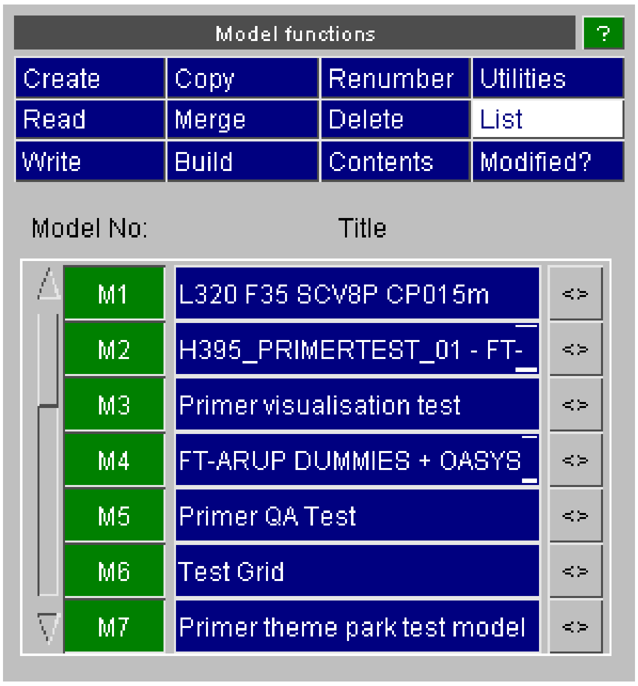


The model menu can always be accessed via the **MODEL** tab in the window control area.

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

3.0 How PRIMER treats "models"

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



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

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

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

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

The "Model No:" column:

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

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

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

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

- Automatically deselects their **M*n*** "tabs" in selection menus (see section 6.0.2), meaning that their contents are not shown by default.

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

The "Title" column:

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

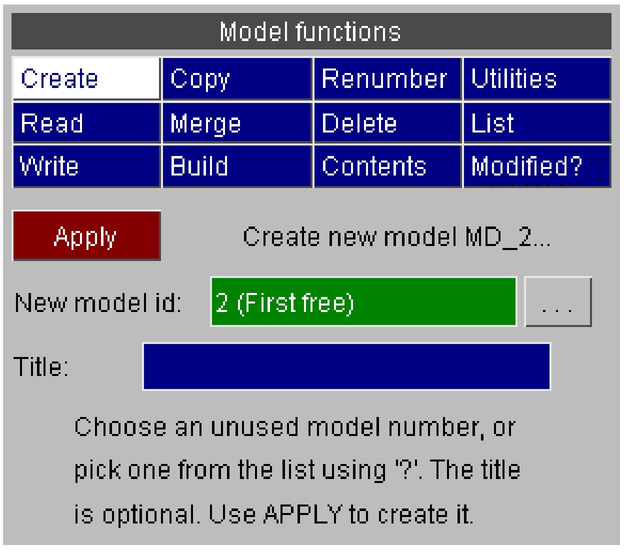
- A model's title may be changed by typing a new string into its "title" button. (It can also be changed in the **Keyword > Control** editing panel)

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

3.1 MODEL > CREATE

Creating a new model - an empty model is created.

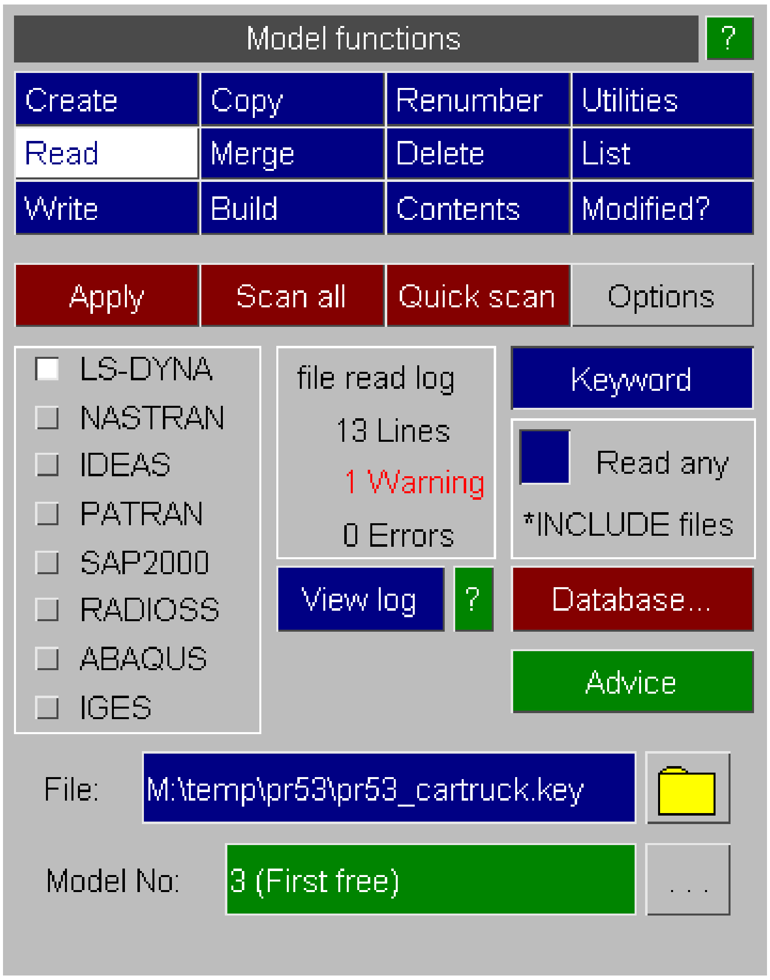
The **CREATE** command allows you to generate a new model without reading anything from disk. You must define its internal number, (which must not already be in use), and optionally give a title.



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

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

3.2 MODEL > READ



Reading in models from disk.

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

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

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

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

3.2.1 Options for LS-DYNA Keyword files only

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

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

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

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

Options: Read one entry per *keyword:

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

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

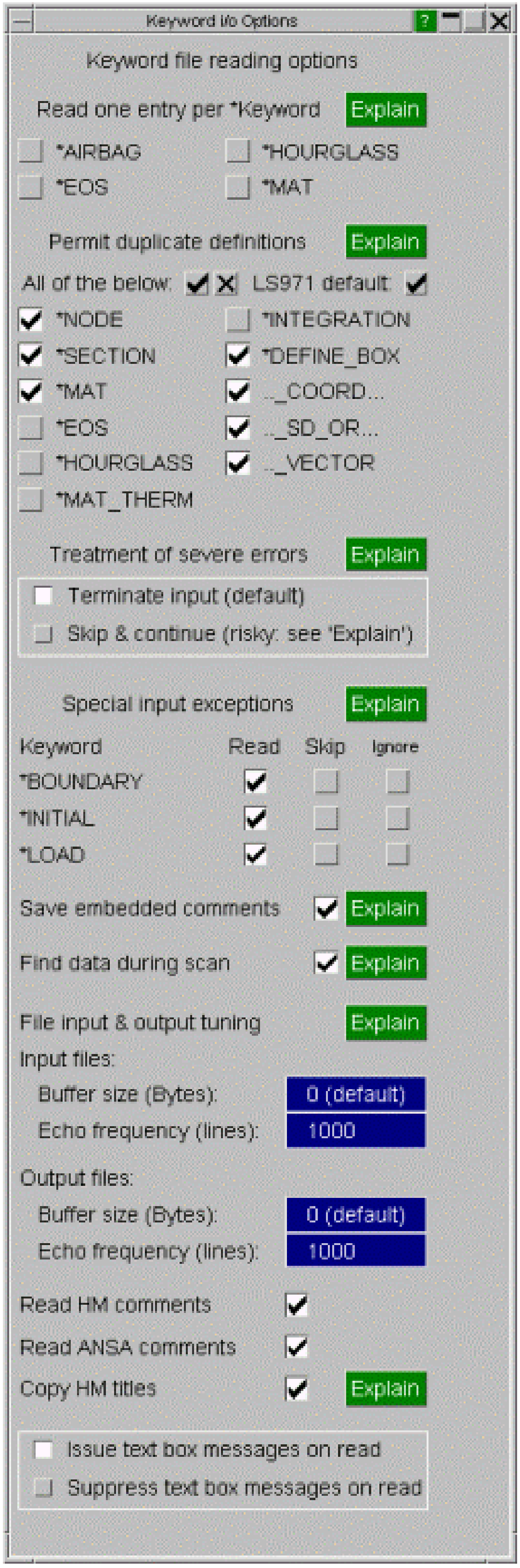
Options: Treatment of severe errors

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

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

Options: File input & output tuning.

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



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

Input and output Buffer size

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

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

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

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

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

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

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

Echo frequency (lines)

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

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

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

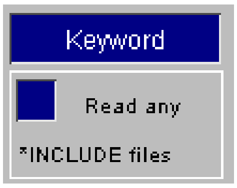
Where **nnnnnn** is the number of lines.

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

Options: Comment reading options.

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

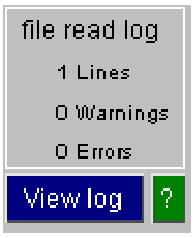
Processing of LS-DYNA include files.



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

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

View log: Viewing the input log



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

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

3.2.2 List of file formats read by PRIMER

Format	Description	What is read
DYNA3D	LS-DYNA "Keyword" format up to LS970	Complete input deck is supported. (No conversion required.)
NASTRAN	NASTRAN "Bulk Data" (.bdf) format	Nodes, elements, loads, properties, materials, SPCs, + others: see Appendix VI (1)
IDEAS	Master Series and IDEAS "Universal" (.unv) file format	See Appendix VI for a list of supported modules.
PATRAN	MSC Patran level 2.5 "Neutral" (.neu) file format.	Nodes and elements only are read.
SAP2000	SAP 2000 input deck	Most items are read, and some interpretation takes place - see Appendix VI (2)
RADIOSS	Mecalog RADIOSS "Starter" and "engine" files, fixed format v4.1	Most items are read, and some interpretation takes place - see Appendix VI (3)
ABAQUS	ABAQUS input deck	Nodes and some element types. Also, basic section, set and simple material data. See Appendix VI for details.
IGES	Geometry data in IGES 5.3 format	Most points, curves and surfaces. See Appendix VI for details.

To read a particular format:

- Select that format in the "Input file formats" area;
- Select any sub-type (here **KEYWORD** has been chosen);
- Select a "target" model id. Here the default of the next free model (#9) has been chosen, but you are free to use any model in the range 1 to 255.

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

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

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

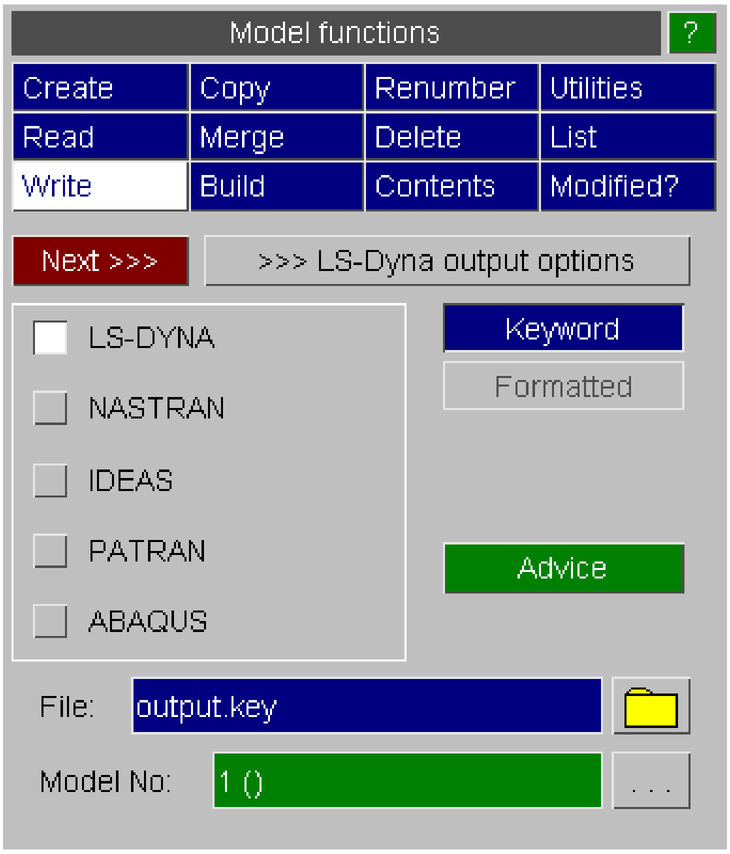
You will be warned if any errors are found, the action taken depending upon context:

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

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

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

3.3 MODEL > WRITE



Writing out models to disk.

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

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

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

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

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

Table 3.2: External data formats written by PRIMER

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

Before writing out the model in LS-DYNA keyword format the user is given a number of review functions. These are a summary of the main `*CONTROL` and `*DATABASE` cards and access to the **Clean up** (see section 6.25.2) and **Model checking** functions (see section 3.9)

CLEAN-UP and **CHECK MODEL** run the standard model tidying and checking functions, and their use is recommended if a deck is to be run in LS-DYNA.

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

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

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

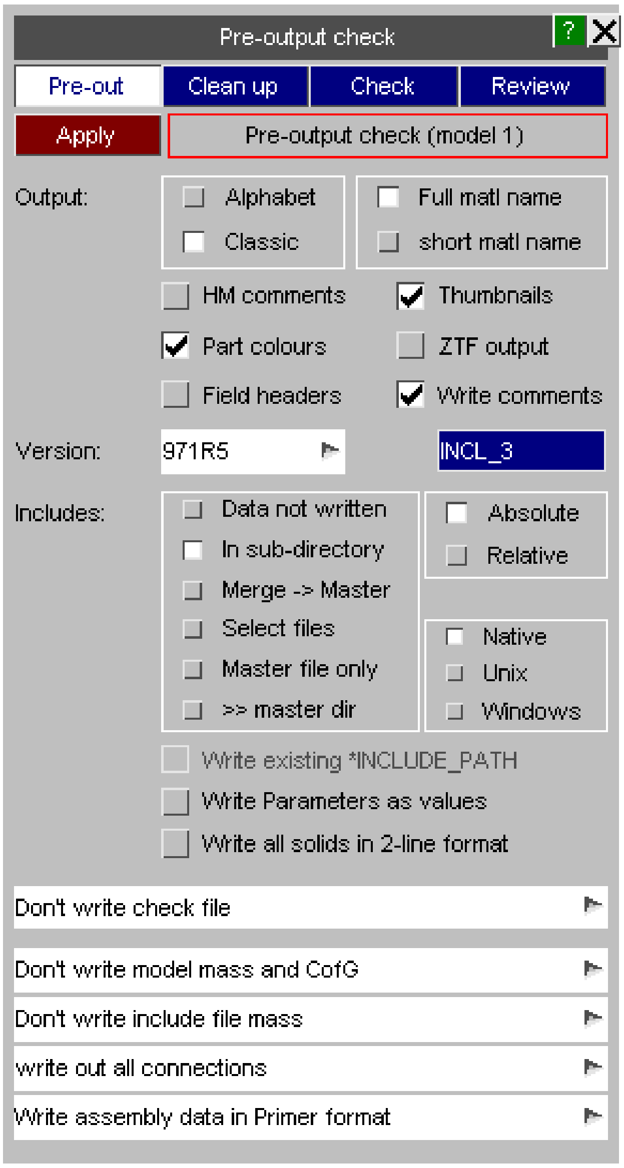
The **Thumbnails** option will write out any existing include file thumbnail images at the end of each keyword file.

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

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

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

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



Version allows you to modify or suppress keywords to be compatible with different versions of LS-DYNA. PRIMER release 10.0 supports the following LS-DYNA formats:

- `LS940`, `950` and `960` "legacy" formats.
- `LS960+` contains a limited subset of the `LS970` keyword format.
- `LS970 v3858`, `v5434` and `v6763` formats are fully supported.
- `LS971R2 (v7600)`
- `LS971R3`
- `LS971R4`
- `LS971R5`
  (Option xxx Dev is the current development version, as is intended for development use only)

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

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

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

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

Includes are quite a complex topic, described in more detail under **INCLUDE Files** below.

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

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

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

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

Model mass and C of G output has the options:

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

Include file mass output has the following options:

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

Connections output has the options:

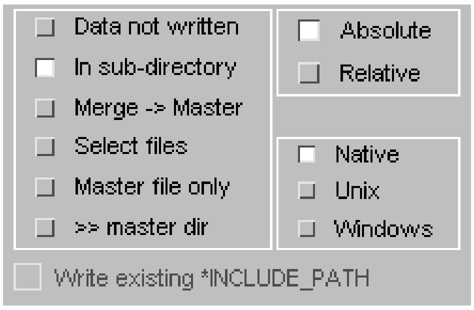
Write all connections	All connections in the model are output.
Suppress all connections	No connections are output
Suppress created by Model > Check	Only connections read in or created by the user are written. Those created during a Model Checking operation are omitted.

Connections are an innovation in PRIMER 9.3 which provide a common way to handle the processing of spotweld, bolts, adhesives and other connection methods. They are described in section 6.10

Assembley output has the options:

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

*INCLUDE Files: How "include" files are handled on output.



The various **INCLUDE files** options allow suppression of data in these files, or their merging into the master file, or separate output.

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

Select the radio button **Use *INCLUDE_PATH if possible** if you wish PRIMER to write out the include files with the `*INCLUDE_PATH` keywords (these are stored on read-in of the model).

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

- **Data not written:** It is not written out at all.
- **In sub-directory:** It is written out in separate files in a sub-directory - see below.
- **Merge->master:** It is merged into the master file, and no *include files are written.
- **Select files:** User selects files he wants to write, choosing overwrite/new-file/into sub-directory for each `*INCLUDE` file
- **Master file only:** Only master file written
- >>**master dir**: All include files are moved into the directory of the master file.

If no renumbering occurred, it is recommended that you select **Master file only** under the `*INCLUDE` heading.

The Master file will list all the Include files to be used, the data of any shifted connection files and (if this option is used) any rigid body merges, which are stored under the **MASTER CONNECTIONS** for each component file.

If any renumbering has occurred or you made any changes specific to the content of any individual Include file it is recommended that you select **Select files** under the *INCLUDE heading. This will allow you to save both the Master file and the modified Include files.

When you have completed all pre-output checks, press **APPLY**

If you select Master file only mode and *renumbering of items in a file has occurred* during model build, you will be prompted to use the Select file mode, which will then be invoked with the renumbered files automatically selected.

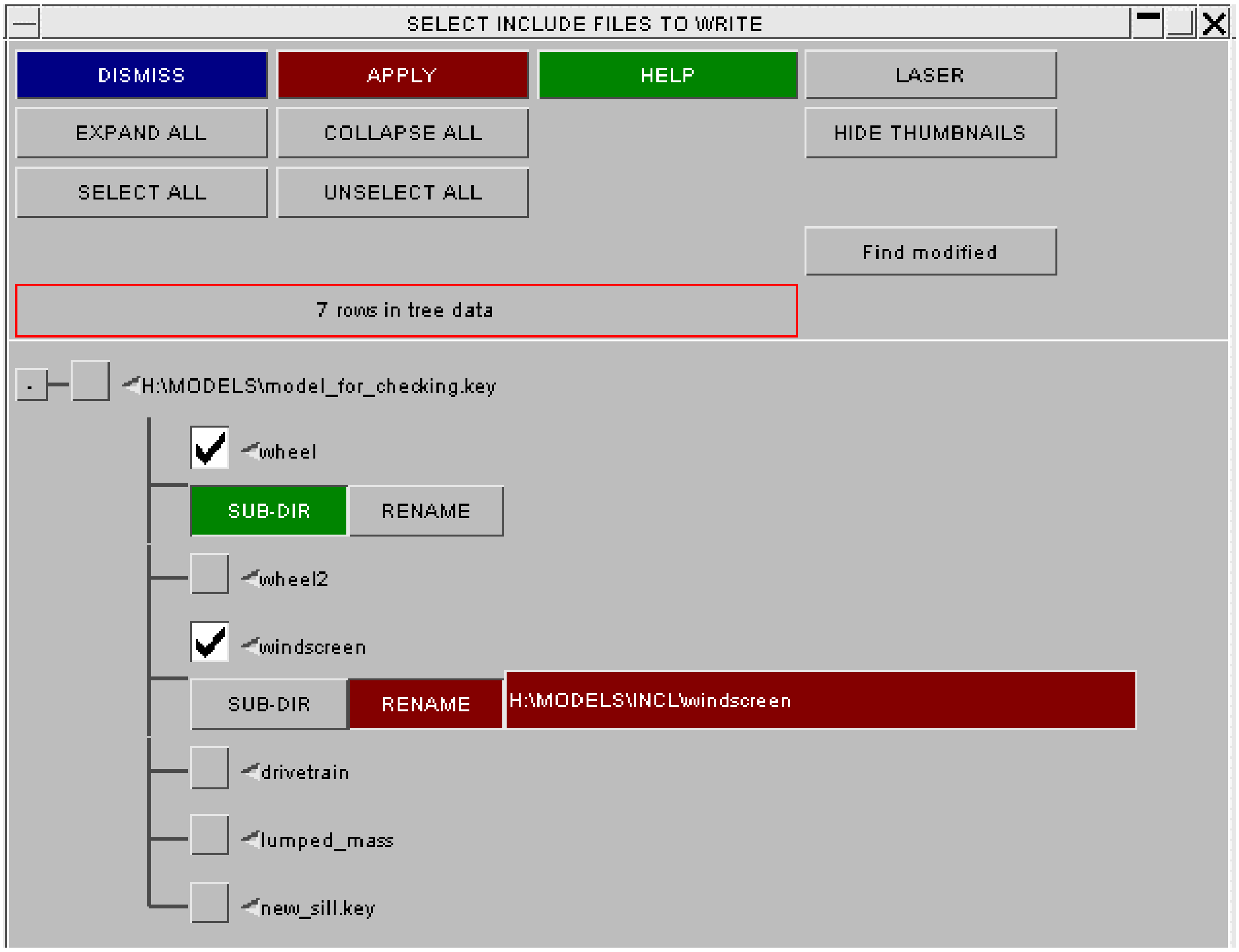
If you selected the Select files option, a new window will appear asking you to select and name which files you wish to save. It is recommended that you select the Master file and any Include files that have been modified or renumbered.

Select files by clicking on the grey box to their left. A tick will appear in selected boxes and the options **SUB-DIR** and **RENAME** will appear.

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

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

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



How master file references include files:

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

How "In sub-directory" output, the default, is handled:

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

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

Unix or Windows environment - Drive mapping:

In the oa_pref file, you may, for example, map the windows "`s`" drive to correspond to the unix directory "`/data`" thus: **primer*drive_s: /data**

Thus an include file "**/data/includes_1/inc.key**" (read in unix) will be referenced under the ***INCLUDE** as "**S:\includes_1\inc.key**" if the model in written out in **WINDOWS** format. Similarly files read in Windows versions may be converted to Unix format.

The default setting **NATIVE** writes in the format appropriate to the machine.

When writing out a model may change its representation in memory.

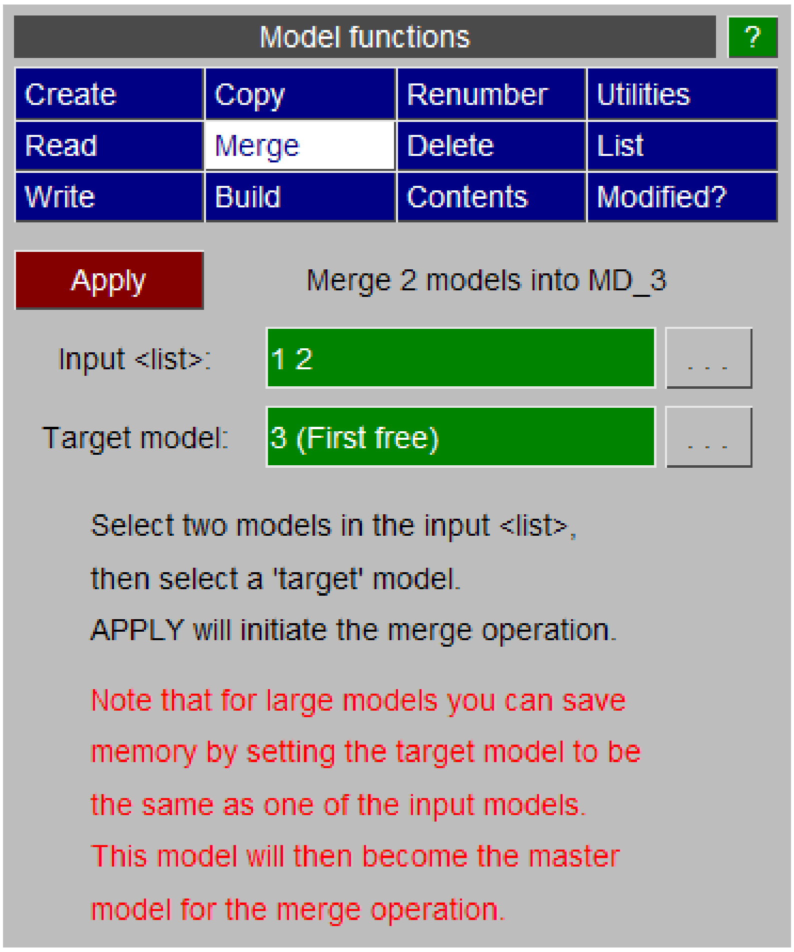
Writing out a model does not usually affect its contents in memory in any way, the exception being when label resequencing is required to prevent clashes in the external format. This usually arises because LS-DYNA permits different classes of item to have overlapping labels, whereas the external data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

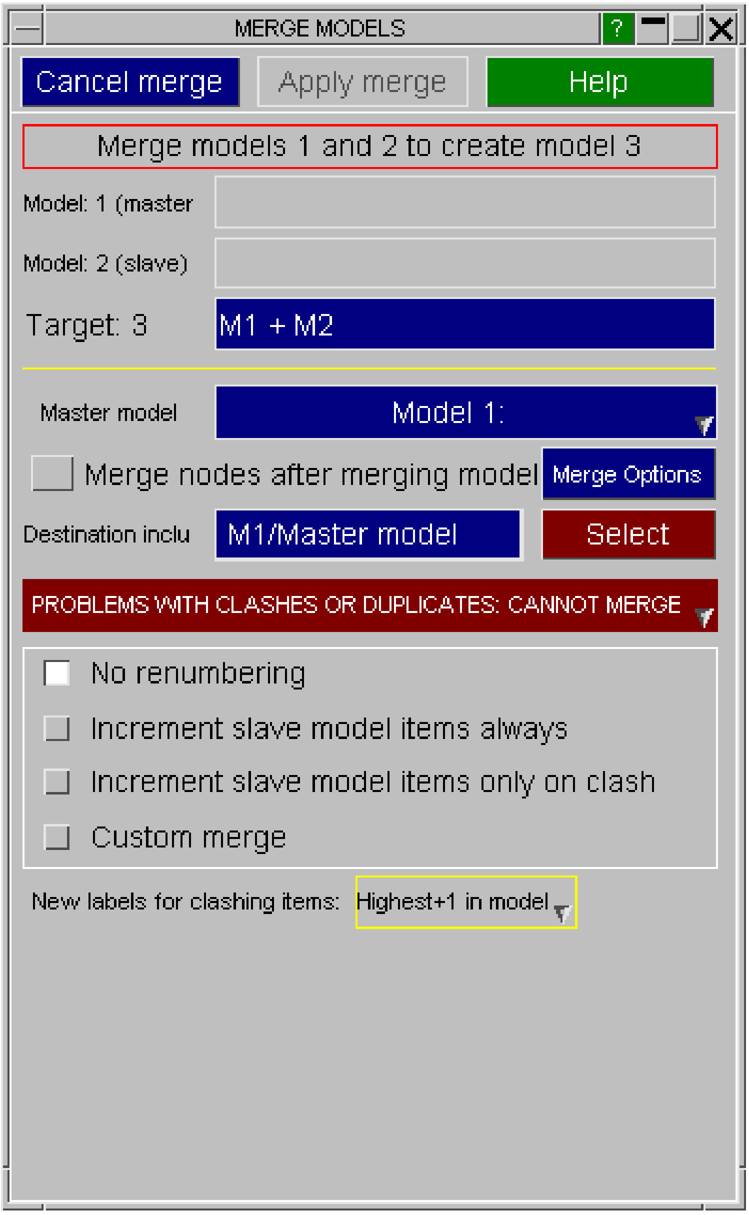
In this situation the labels in the internal model are permanently modified as required. If this is not acceptable **COPY** the model before you write it out, then **DELETE** the copy that has been modified during the write operation.

3.4 MODEL > MERGE

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

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





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

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

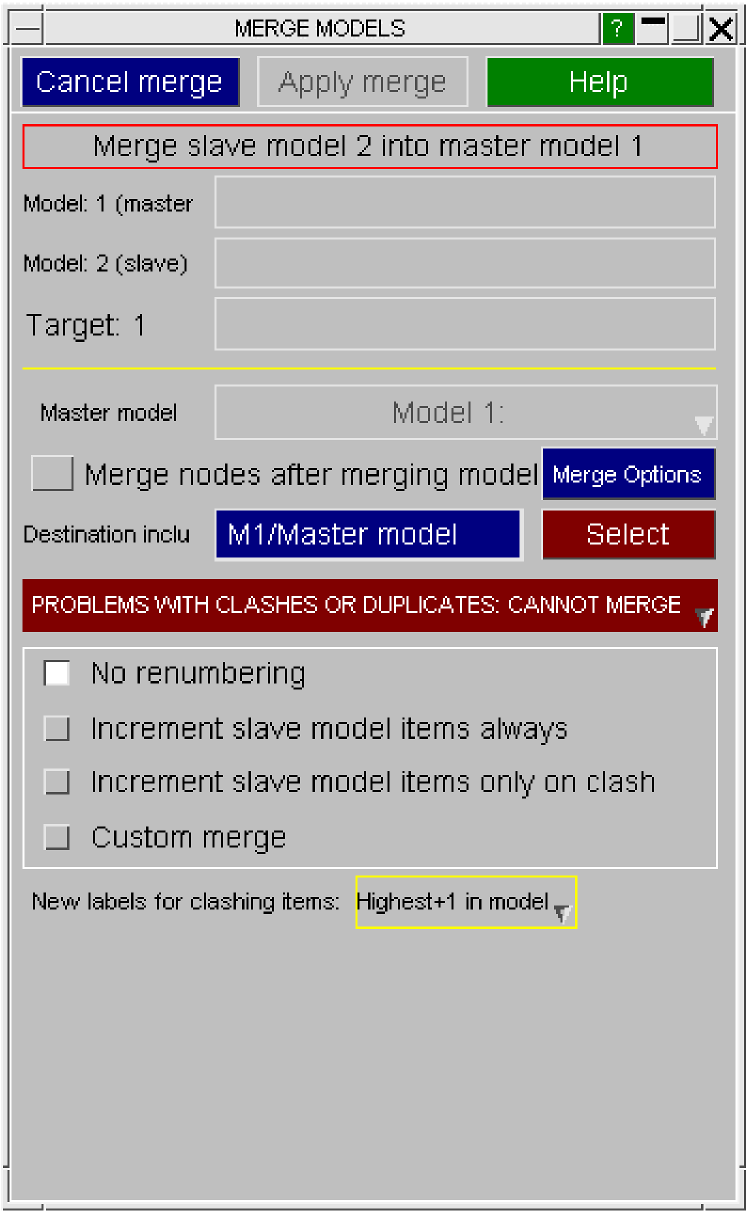
A title for the target model (in this example model 5) can be typed in and this will be used when the new model is created.

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

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

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

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



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

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

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

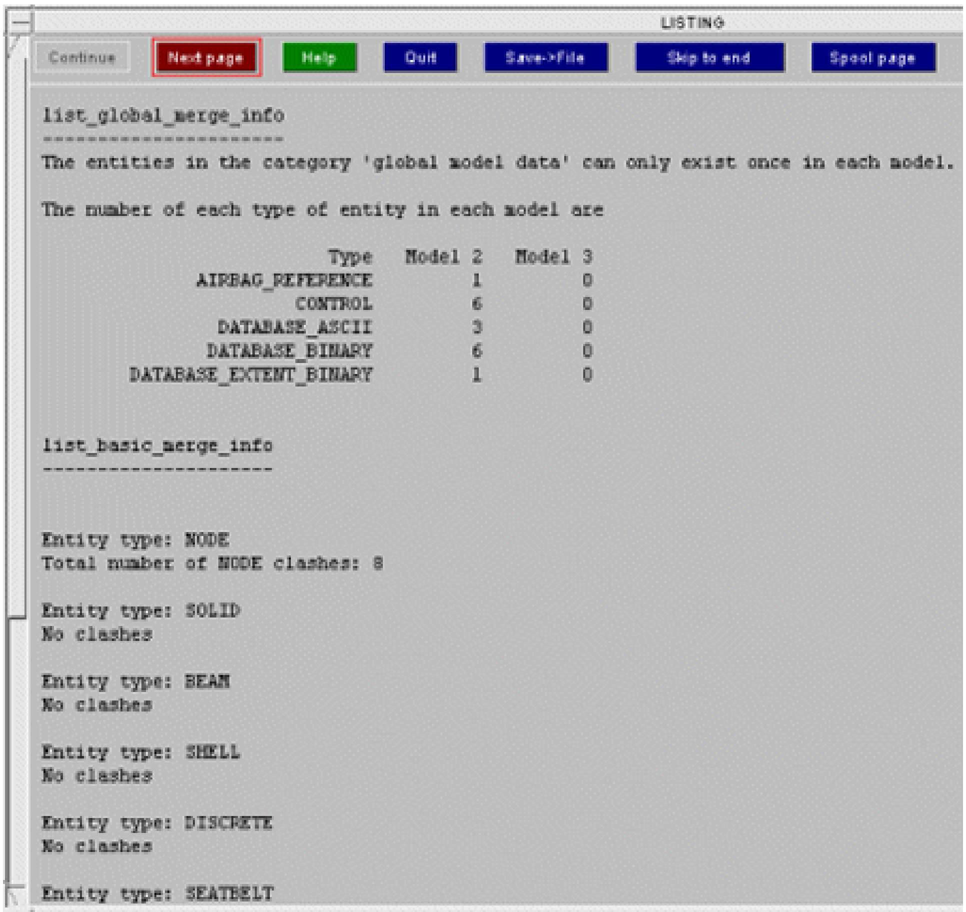
Additionally some of the options for fixing clashes in the custom merge case will not be available as they are not possible.

3.4.1 Options to fix clashes

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

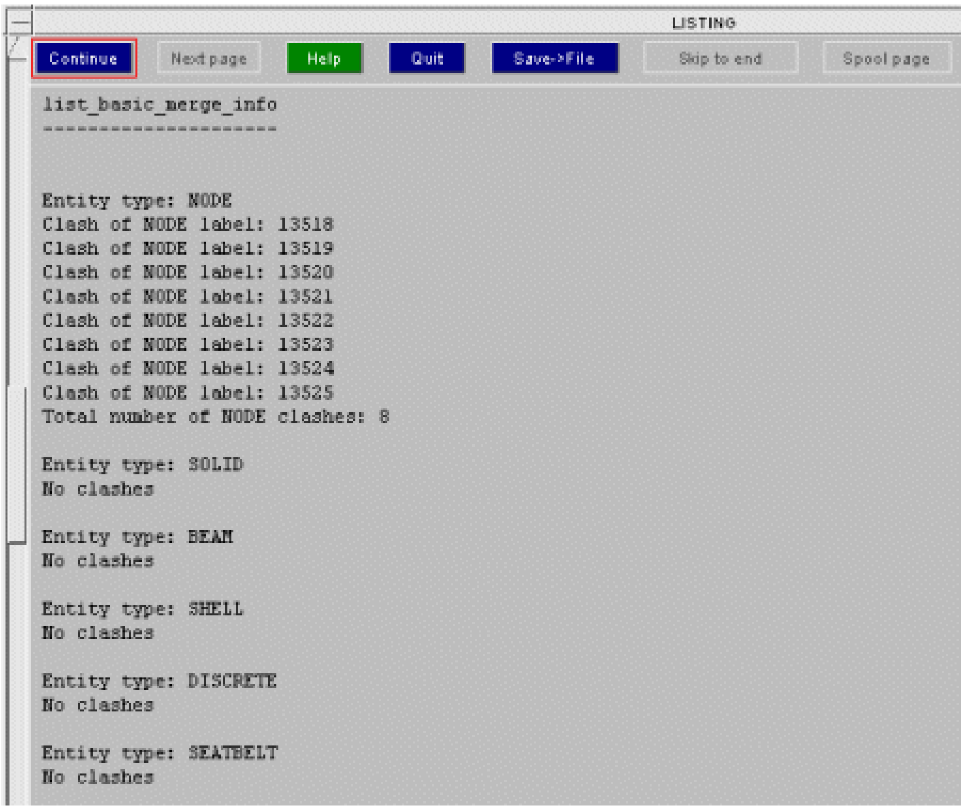
Determining the nature of clashes

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



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

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



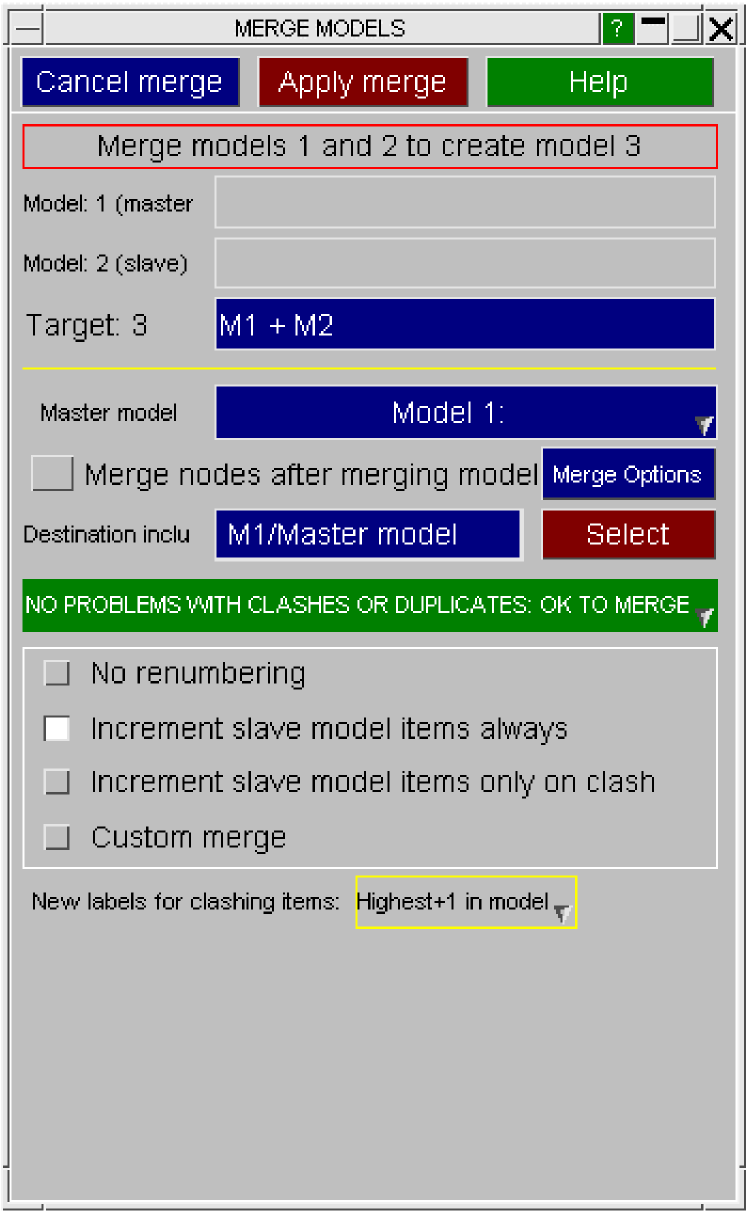
The problem with label clashes is easily solved by using either option 2 or option 3 above (or custom merge for greater control)

The problem with duplicate items such as control cards etc is solved by the following method in options 1, 2 and 3.

- If the item exists in the master model but not in the slave model it is taken from the master model.
- If the item exists in the slave model but not in the master model it is taken from the slave model.
- If it exists in the slave and master model it is taken from the master model.

Other items in the model which do not have labels such as constrained cards and boundary cards cannot clash so there is no problem with merging these entities. All of the cards from both models are used in the merge process.

Once you have chosen a method for merging the two models together which fixes any potential problems the **PROBLEMS WITH MERGE...** button in the main merge window will change message and turn green. The **APPLY MERGE** button will be ungreyed and the merge can proceed.



This method is the easiest way to merge two models together. The original models will not be deleted after merging so if the outcome of the merge is not what you wanted you do not lose your original models. However, it is good practice to save your models before attempting to merge them together.

For much greater control on how models are merged together the **Custom Merge** option can be used but this is only recommended for experienced users as there are potential problems. This is described in section 3.4.2.

New labels for clashing items

By default the labels for clashing items will be changed to be greater than the highest label in the master and slave model. e.g. if the labels for nodes in the slave model were in the range 100 - 1000 and in the range 200 - 2000 in the master model the clashing node labels would be changed to be 2001, 2002 etc. This can be changed with the **New labels for clashing items** popup. The default value of **Highest+1 in model** is the above behaviour. Instead a start label can be given for new labels by changing this to **Start at label** and giving a starting label in the text box. For example, with the above labels you may want clashing labels to start at 100000.

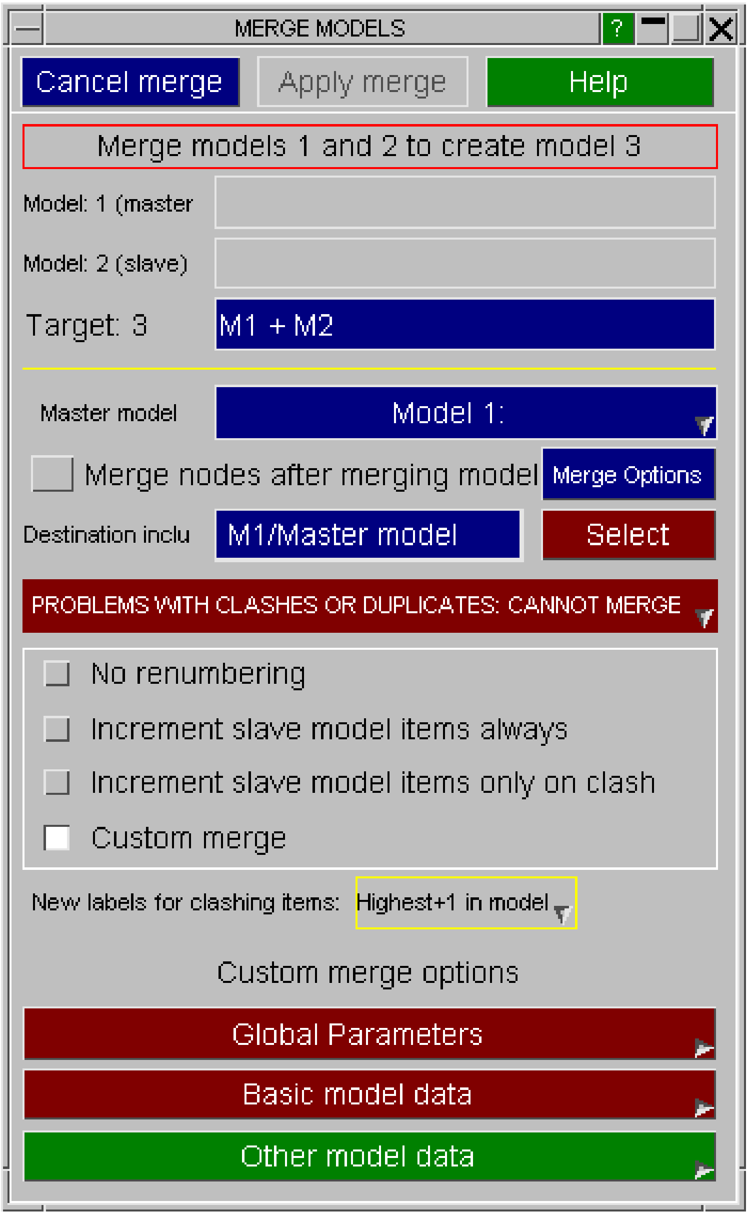
3.4.2 Custom Merge

More control over merging process

The custom merge option gives control of how each type of entity is merged from the two source files. For example, shells can be merged together in a different way to nodes and parts.

If custom merge is selected, three extra buttons are shown at the bottom of the window:

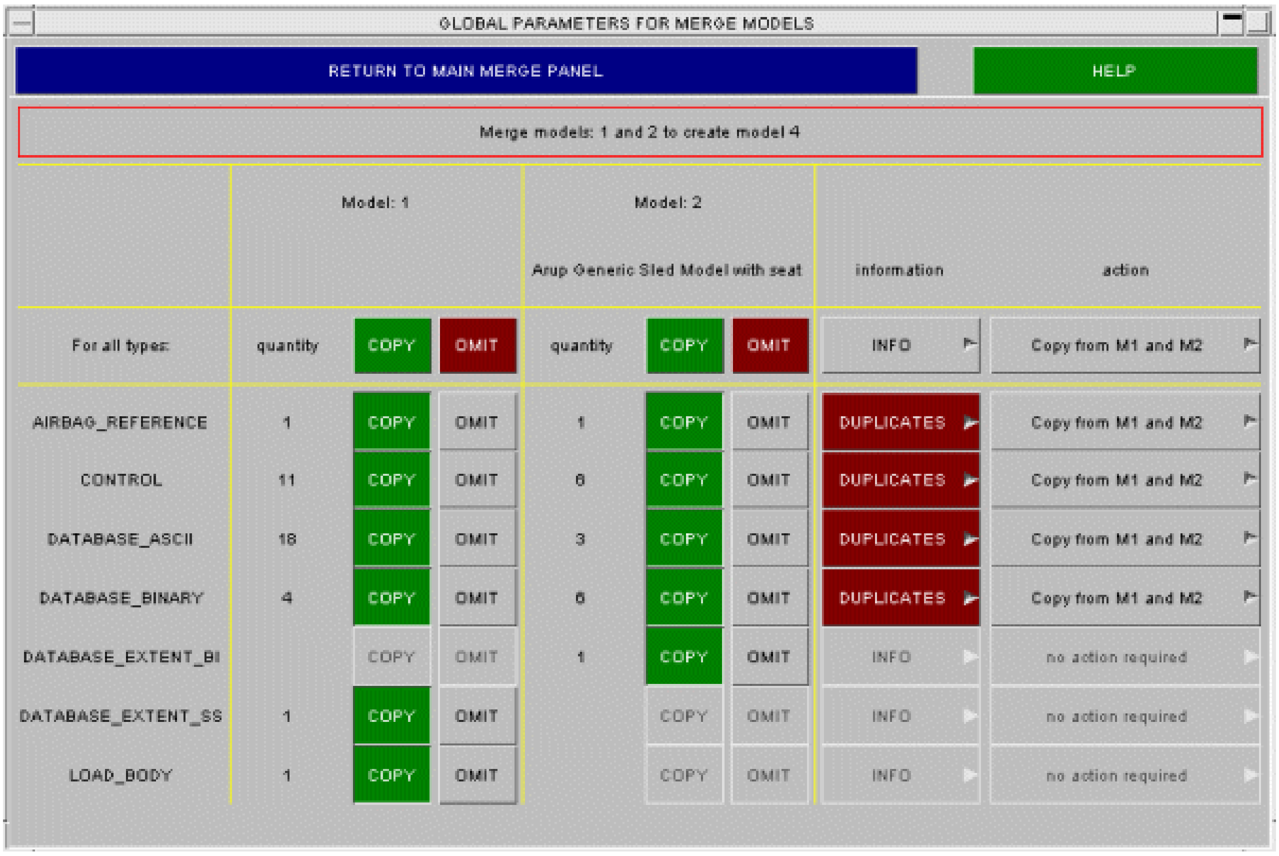
- **GLOBAL PARAMETERS** This is used for things that can only exist once in the model such as control and database cards, airbag reference geometry etc.
- **BASIC MODEL DATA**. The basic model data comprises entities which have labels and so can clash. E.g. shells, solids, nodes, loadcurves etc.
- **OTHER MODEL DATA**. The other model data comprises entities which do not have labels and so cannot clash. E.g. boundary cards, constrained cards.



If a button is red then there are problems with merging that type of data. For example in the figure above there are problems with the basic model data. The "other model data" button will always be green because there cannot be clashes as the entities have no labels. A summary or a detailed list of the problems for each category is available by using the popup on each button.

Clicking on each of the custom merge buttons will bring up a new window showing all the entity types in that category and the associated problems. Using these windows the problems can be solved and then the models can be merged together.

Custom Merging Global Parameters



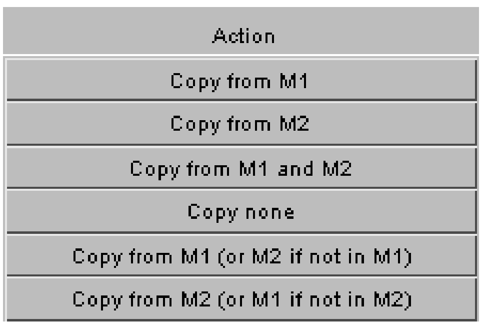
This figure shows the global parameters window for the example above. You can see that:

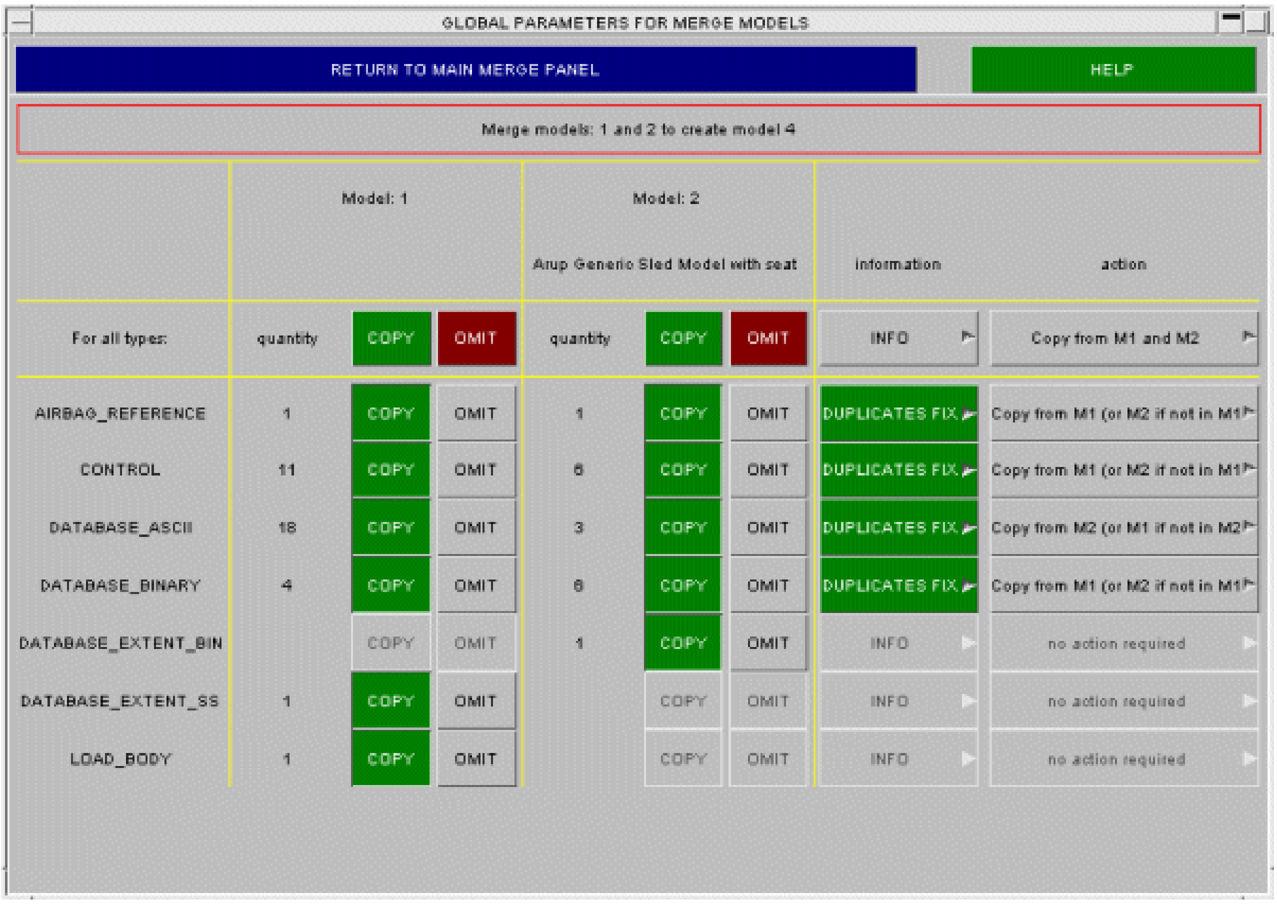
`DATABASE_EXTENT_BINARY` `DATABASE_EXTENT_SSSTAT, LOAD_BODY`	Cards only exist in one of the 2 models so there is no problem.

`AIRBAG_REFERENCE_GEOMETRY,`
`CONTROL,`
`DATABASE_ASCII`
`DATABASE_BINARY`
	Cards exist in both models and so there are duplicate cards which is causing a problem.

Resolving clashes using the global "Action" popup menu

The problems arise because the default action for all the types is to copy from both models (**Copy from M1 and M2**). This can be changed for all types, or for each individual type by using the action popup. The actions are self explanatory except for the last two. The **Copy from M1 (or M2 if not in M1)** card is only in the first model or is in both models. If will take a card from the first model if the the card is only in the second model it will be taken from the second model. This is useful if you want to make sure that all control cards are copied from both models. They will be taken from both but the control cards in the first model will take precedence over the control cards in the second model.

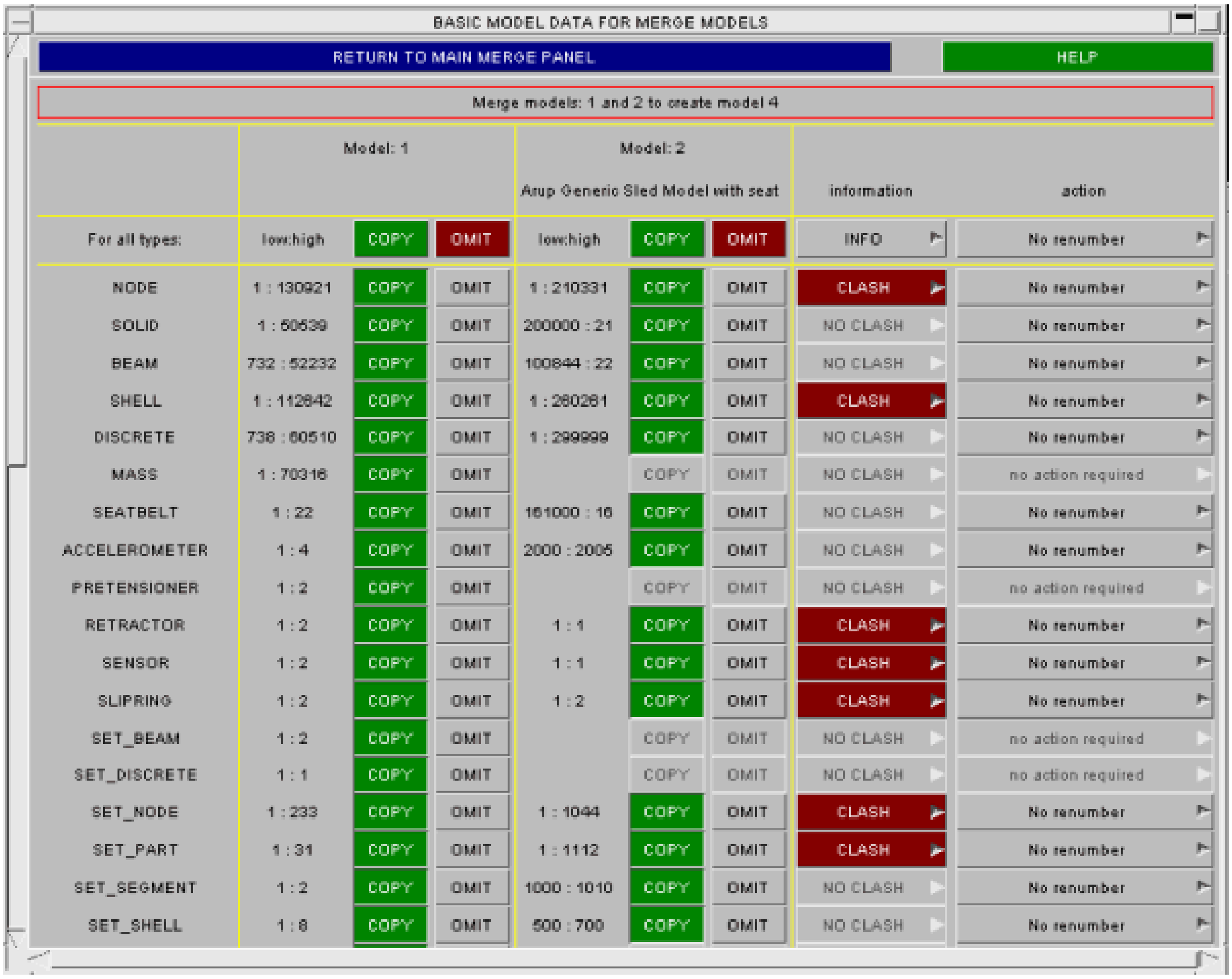




As appropriate actions are chosen for each type (or all types) the **DUPLICATES** will be replaced by **DUPLICATES_FIXED**. The figure above shows the same model after actions have been chosen to fix problems.

Summary or detailed information on a problem with each type is available by using the information popups.

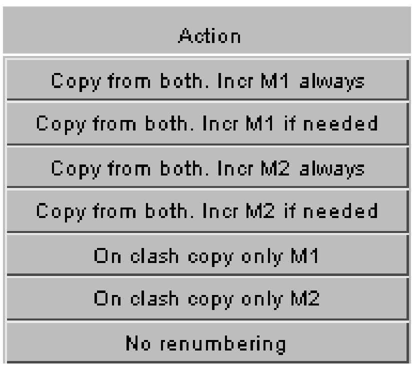
Custom Merging Basic Model Data



This figure shows the basic model data window. In this example there are problems with clashes in the types **`SHELL, SET_PART, PART, MATERIAL`** and **`CONTACT`**.

Resolving clashes using the basic "Action" popup menu

Just as in the global parameters window, actions can be used to solve the clash problems.



The actions available to resolve these, in the **CLASH** > popup menu, are, in more detail:

1	**Copy from both. Inc M1 always**.	The items will be taken from both models. The labels of all items in M1 (of this type) are renumbered to be above the item labels in M2

2 **Copy from both. Inc M1 if needed** The items will be taken from both models. The labels of items in M1 (of this type) will be renumbered to be above the item labels in M2 only if there is a clash. If there is no clash the original label will be used

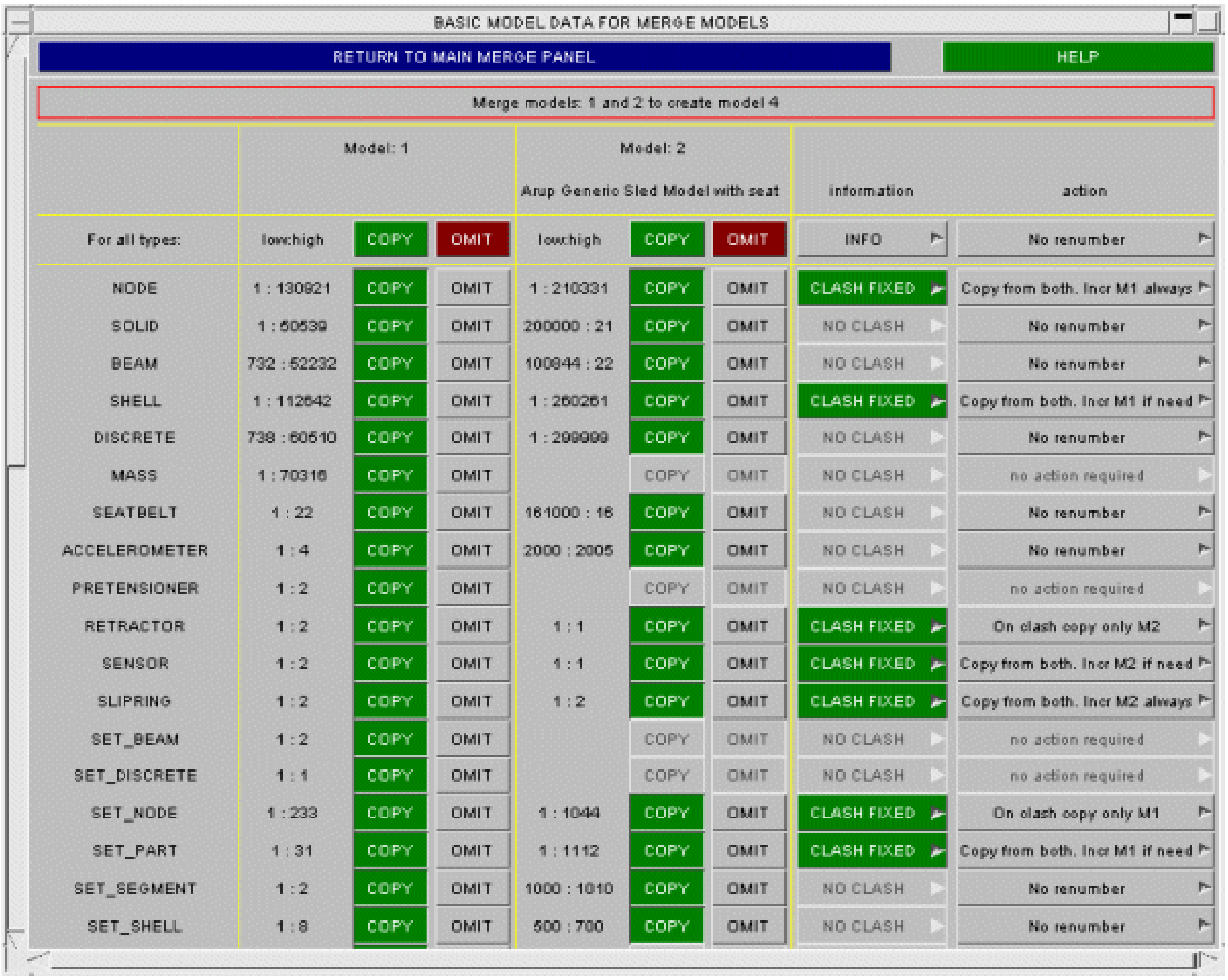
3 **Copy from both. Inc M2 always**. As 1. except models swapped.

4 **Copy from both. Inc M2 if needed**. As 3. except models swapped.

5 **On clash copy only M1** The items will be taken from both models except when there is a label clash. When this occurs only the item from M1 will be taken

6 **On clash copy only M2**. As 5. Except item will be taken from M2

7 **No renumbering**. Nothing will be renumbered. This is only possible if there are no clashes.


BASIC MODEL DATA FOR MERGE MODELS

RETURN TO MAIN MERGE PANEL | HELP

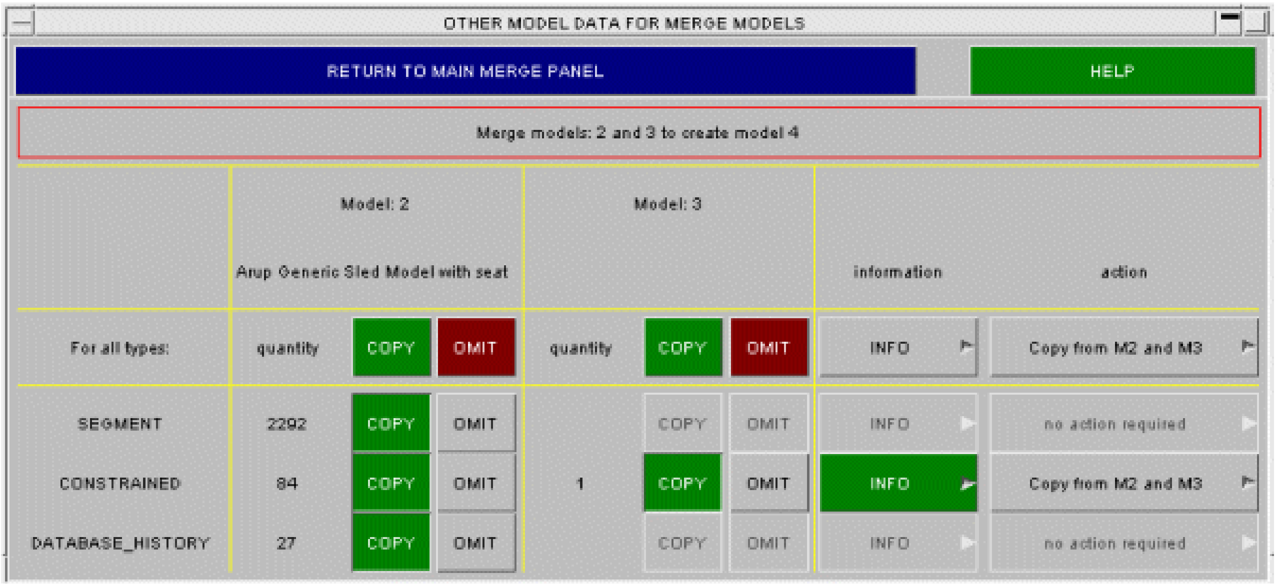
Merge models: 1 and 2 to create model 4

	Model: 1			Model: 2 Arup Generic Sled Model with seat			information	action
For all types:	low:high	COPY	OMIT	low:high	COPY	OMIT	INFO	No renumber
NODE	1 : 130921	COPY	OMIT	1 : 210331	COPY	OMIT	CLASH FIXED	Copy from both. Incr M1 always
SOLID	1 : 50539	COPY	OMIT	200000 : 21	COPY	OMIT	NO CLASH	No renumber
BEAM	732 : 52232	COPY	OMIT	100844 : 22	COPY	OMIT	NO CLASH	No renumber
SHELL	1 : 112642	COPY	OMIT	1 : 260261	COPY	OMIT	CLASH FIXED	Copy from both. Incr M1 if need
DISCRETE	738 : 60610	COPY	OMIT	1 : 299999	COPY	OMIT	NO CLASH	No renumber
MASS	1 : 70316	COPY	OMIT		COPY	OMIT	NO CLASH	no action required
SEATBELT	1 : 22	COPY	OMIT	161000 : 16	COPY	OMIT	NO CLASH	No renumber
ACCELEROMETER	1 : 4	COPY	OMIT	2000 : 2005	COPY	OMIT	NO CLASH	No renumber
PRETENSIONER	1 : 2	COPY	OMIT		COPY	OMIT	NO CLASH	no action required
RETRACTOR	1 : 2	COPY	OMIT	1 : 1	COPY	OMIT	CLASH FIXED	On clash copy only M2
SENSOR	1 : 2	COPY	OMIT	1 : 1	COPY	OMIT	CLASH FIXED	Copy from both. Incr M2 if need
SLIPRING	1 : 2	COPY	OMIT	1 : 2	COPY	OMIT	CLASH FIXED	Copy from both. Incr M2 always
SET_BEAM	1 : 2	COPY	OMIT		COPY	OMIT	NO CLASH	no action required
SET_DISCRETE	1 : 1	COPY	OMIT		COPY	OMIT	NO CLASH	no action required
SET_NODE	1 : 233	COPY	OMIT	1 : 1044	COPY	OMIT	CLASH FIXED	On clash copy only M1
SET_PART	1 : 31	COPY	OMIT	1 : 1112	COPY	OMIT	CLASH FIXED	Copy from both. Incr M1 if need
SET_SEGMENT	1 : 2	COPY	OMIT	1000 : 1010	COPY	OMIT	NO CLASH	No renumber
SET_SHELL	1 : 8	COPY	OMIT	500 : 700	COPY	OMIT	NO CLASH	No renumber

As appropriate actions are chosen for each type (or all types) the **CLASH** will be replaced by **CLASH_FIXED**. This figure shows the same model after actions have been chosen to fix problems. The **action** column shows what has been done in each case.

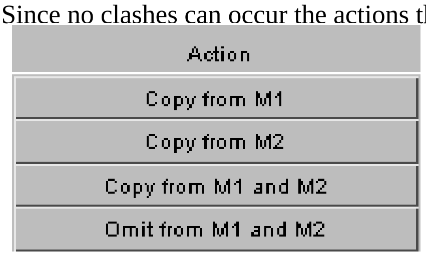
Custom Merging "Other" Model Data

"Other" means items that don’t have labels, so which can never clash, but which may require manual control.



In this example there are *AIRBAG_INTERACTION, *BOUNDARY and cards.

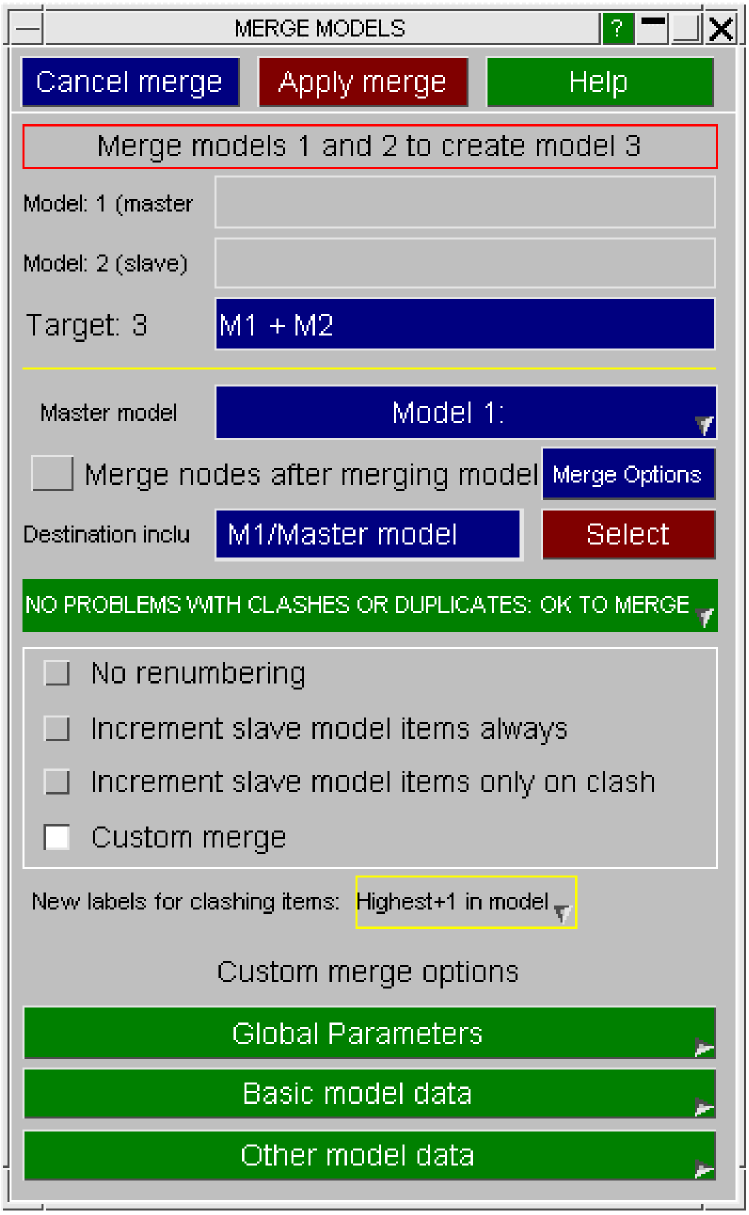
Since no clashes can occur the actions that can be taken are simply to copy or to leave each category.



Other Issues in Custom Merge

As the problems in the **GLOBAL PARAMETERS**, **BASIC MODEL DATA** and **OTHER MODEL DATA** are fixed the main merge window will be updated. When all the problems are fixed the merge can be done. The original models will not be deleted after merging.

Great care must be taken when using the custom merge options, especially when omitting some entity types from either model.



As an example imagine merging 2 models which both have a rivet from node 1 to node 2. When doing a custom merge you will be warned that there is a clash of nodes between the 2 models. If you choose an action to renumber the nodes everything will be OK. If instead you only take the nodes from one of the models then there is no clash as the rivets have no labels, but the merged model will have 2 rivets from node 1 to node 2. There are lots of similar situations which may occur which do not cause errors in the merging process but may give an unexpected result when the models are merged.

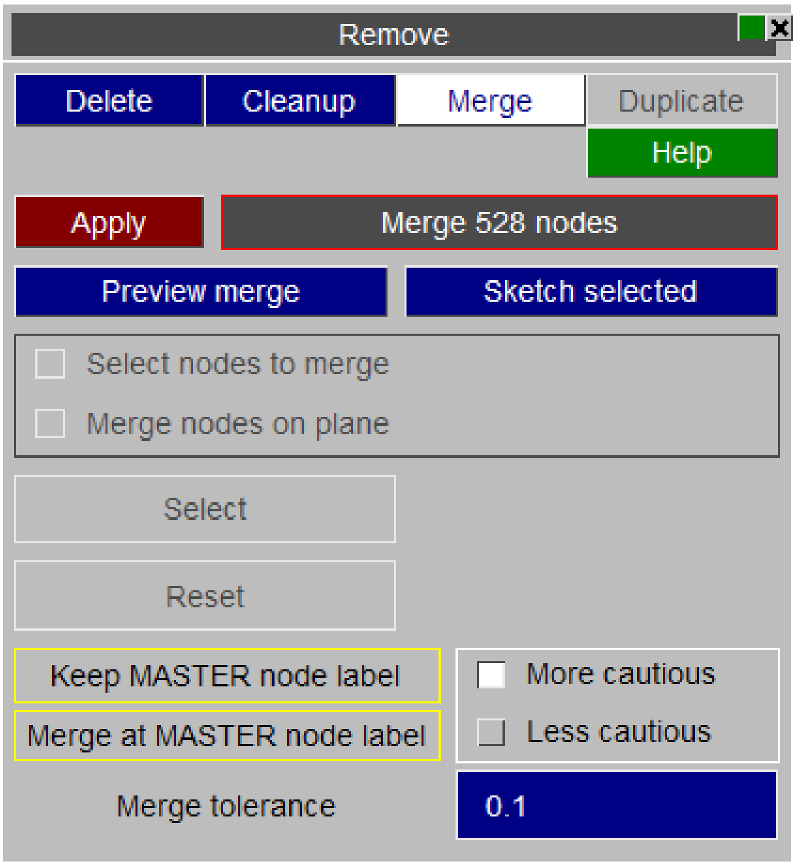
3.4.3 Merging nodes during model merge

If the **Merge nodes after merging model** checkbox on the main model merge panel is selected then PRIMER will perform a node merge on the target model after the models are merged together.

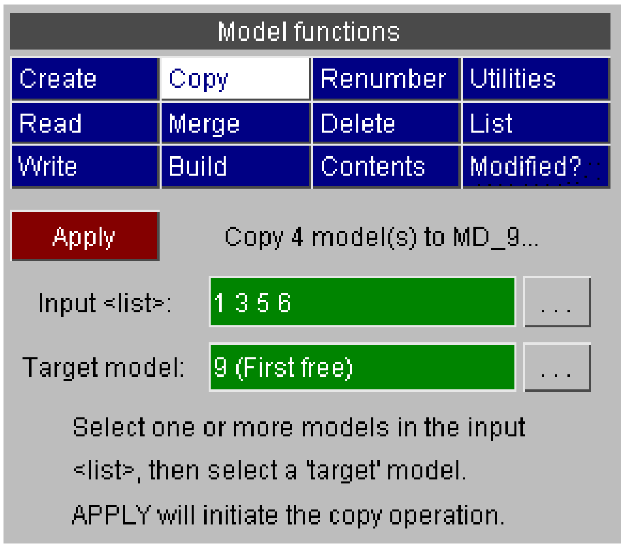
The merge nodes panel is started in a special mode where only nodes from the slave model can be replaced by nodes from the master model. In this mode:

- The node label from the master model is always retained.
- The position of the merged node will always be the position of the node in the master model.

For more details on merging nodes see the merge nodes section of the manual.



3.5 MODEL > COPY Copying models internally.



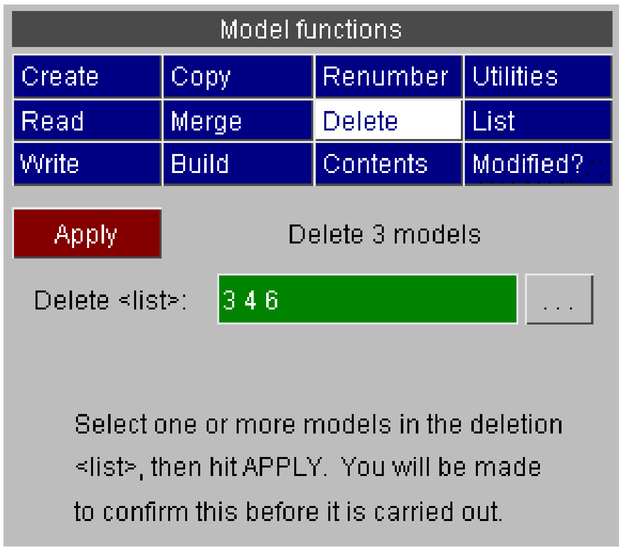
You can copy a <list> of ***n*** existing models to ***n*** new models starting at model ***i***.

The process is simple, as shown in this figure:

- Select a list of 1 or more input models (which must all exist).
- Select the first target model (which must not exist).
- Press **APPLY** to start the copy operation

The input models are copied in the order defined to new models starting at the target model id. New models are created in a contiguous sequence of free models: any existing ones are skipped over, not deleted.
Copying a model duplicates all the internal data; and the new model(s) created are totally separate from their originals. (Internally the model is effectively written out and read back into the new model, although this is carried out in memory and no disk i/o is performed.) For this reason a **COPY** operation may take a little time, although it is usually still much faster than re-reading from disk file.

3.6 MODEL > DELETE Deleting internal models



Entire models may be deleted from memory with the **DELETE** command (Models are only deleted from PRIMER, not from disk).

Deletion is carried out as shown in this figure:

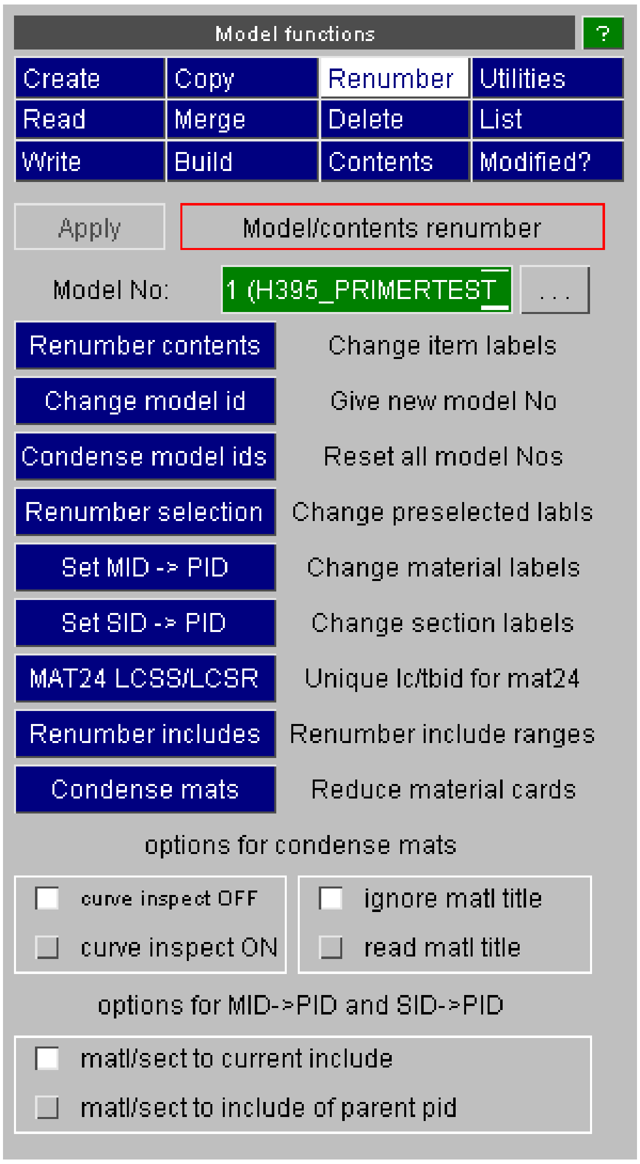
- Select a list of 1 or more existing models;
- Hit **APPLY**.
- You will be forced to confirm that you want to do this before they are actually deleted.

Once confirmed the complete model(s) will be removed from memory. **Deletion is irrevocable**: once it has been deleted a model cannot be restored.

The **MODEL** > **DELETE** command should only be used when you want to remove a ***complete model*** from memory. To delete a ***subset*** of a model (eg individual parts, elements, nodes, etc) you should use one of the following options:

Function **REMOVE**	**DELETE UNWANTED** **CLEANUP UNUSED**	Lets you select items of any type for deletion. "Cleans up" redundant items from a model.
Most **KEYWORD >** panels	**DELETE**	Lets you select items of that type for deletion
Generic Keyword Editor	**Row index > Delete**	Deletes item(s) on selected row(s)

3.7 MODEL > RENUMBER Renumbering models and/or their contents



This command is useful for renumbering models, or whole categories within models and, while you can use it to renumber individual items, it may be easier to edit them directly.

A brief description of each option is given below, follow the hyper-links for more information.

- **RENUMBER CONTENTS** lets you renumber the contents of a model.
- **CHANGE MODEL ID** lets you renumber the label of a model itself (not its contents).
- **CONDENSE MODEL IDS** renumbers <n> models from 1 to n. (Only the model ids themselves are renumbered, their contents are unchanged)
- **RENUMBER SELECTION** renumber items selected via an object menu

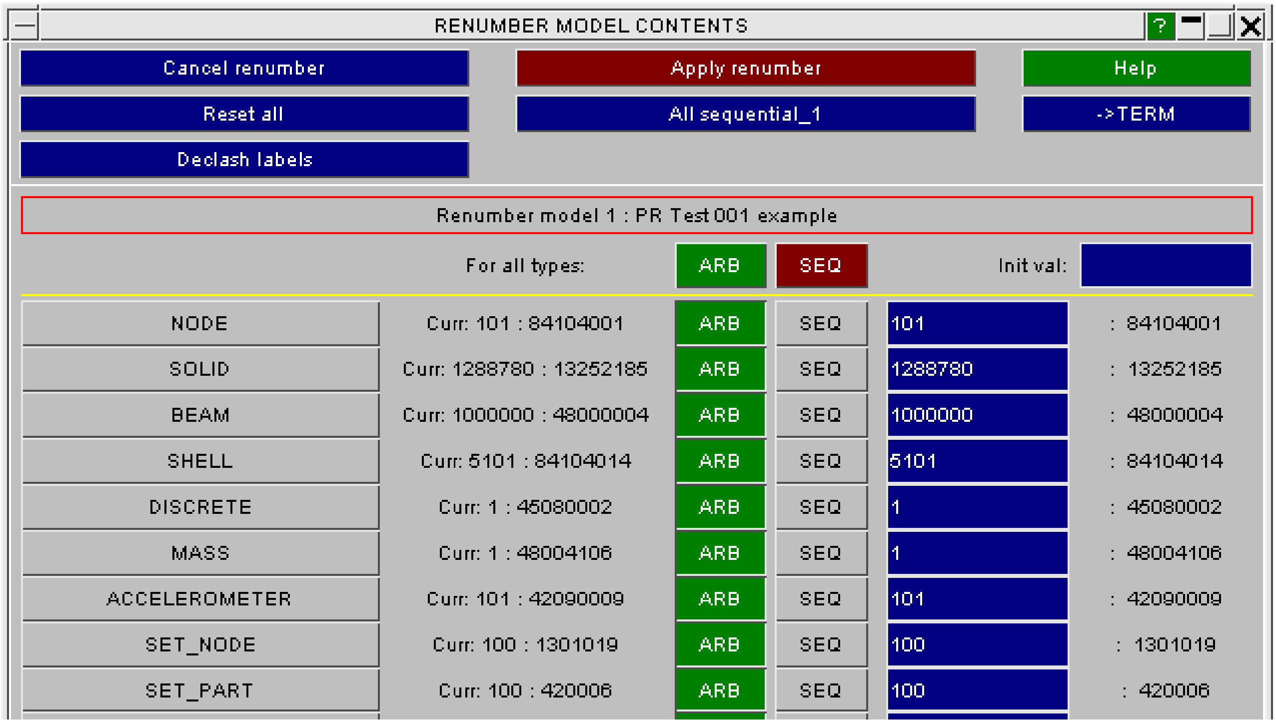
- **SET MID->PID** establishes a `*MAT` card for every part using the same label. If more than one part uses the same material, then a copy of the material is created. This does not apply for `*PART_COMPOSITE` parts that can refer to > 1 material.
- **SET SID->PID** establishes a `*SECTION` card for every part using same label. Again, if more than one part uses the same section, then a copy of the material is created
  The **Options for MID->PID and SID->PID** determine where the newly created `*MAT` and `*SECTION` cards are placed if the model contains include files:

Matl/Sect to current include	Places all newly created definitions in the current include file, regardless of where their referring `*PART` cards occur.
Matl/Sect to include of parent PID	Places each newly created `*MAT` and `*SECTION` card in the same include file as its referring `*PART` card

- **CONDENSE MATS** minimises the number of materials by condensing duplicate identical definitions into a single `*MAT` card. In order for materials to be condensed all data fields must match, subject to the exceptions defined in the options below.
- **RENUMBER INCLUDES** lets the user renumber ranges for general types and for types with explicitly relevant labels for the master file and for one or more include files.
- **MAT24 LCSS/LCSR** sets a unique load curve or table id on the material (MAT24) cards that use the same curve or table. Copies are made of the curve / table in order to achieve this.

3.7.1 RENUMBER CONTENTS Renumbering the item labels within a model

- Select a model.
- Press **RENUMBER CONTENTS** to get the renumbering panel



This figure shows a typical panel, but the actual appearance will depend upon the contents of your model.

An individual category (eg **NODE**) can be renumbered selectively by clicking on its category name button.

The model renumbering table has the following columns for each item category:

KEYWORD	Current range	ARB SEQ	<Initial>	Highest
Each item category in your model, defined by its LS-DYNA keyword.	The unmodified lowest : highest labels for this category	**ARB**itrary or **SEQ**uential labels for this category	The first (lowest) label of the category	What the highest label will become.

To renumber all the items in a category (eg all NODES):

You can control two aspects of labelling for any item category:

1: The spacing between item labels: which may be **ARB**itrary or **SEQ**uential.

ARBitrary Starts at the given initial value, but preserves the gaps between successive items. This is the default.

SEQuential Starts at the given initial value, and numbers items sequentially upwards from that with no gaps

2: The initial value.

The default is whatever the input model contained, but you may change this to any positive integer. Successive values will be adjusted in an "arbitrary" or "sequential" fashion from this value.

Each item category may be changed individually, or a complete column can be changed by using the "**For all types**" boxes at the head of the list.

Also there are some commonly used global options:

ALL_SEQUENTIAL_1 Renumbers everything sequentially starting from 1.
CANCEL_RENUMBERING Exits renumbering without making any changes to the model.
RESET_ALL Sets all the values of all boxes back to their original values.

The changes made in this box are volatile.
They are only permanently saved in this model when **APPLY_RENUMBERING** is used.

To renumber an individual category selectively

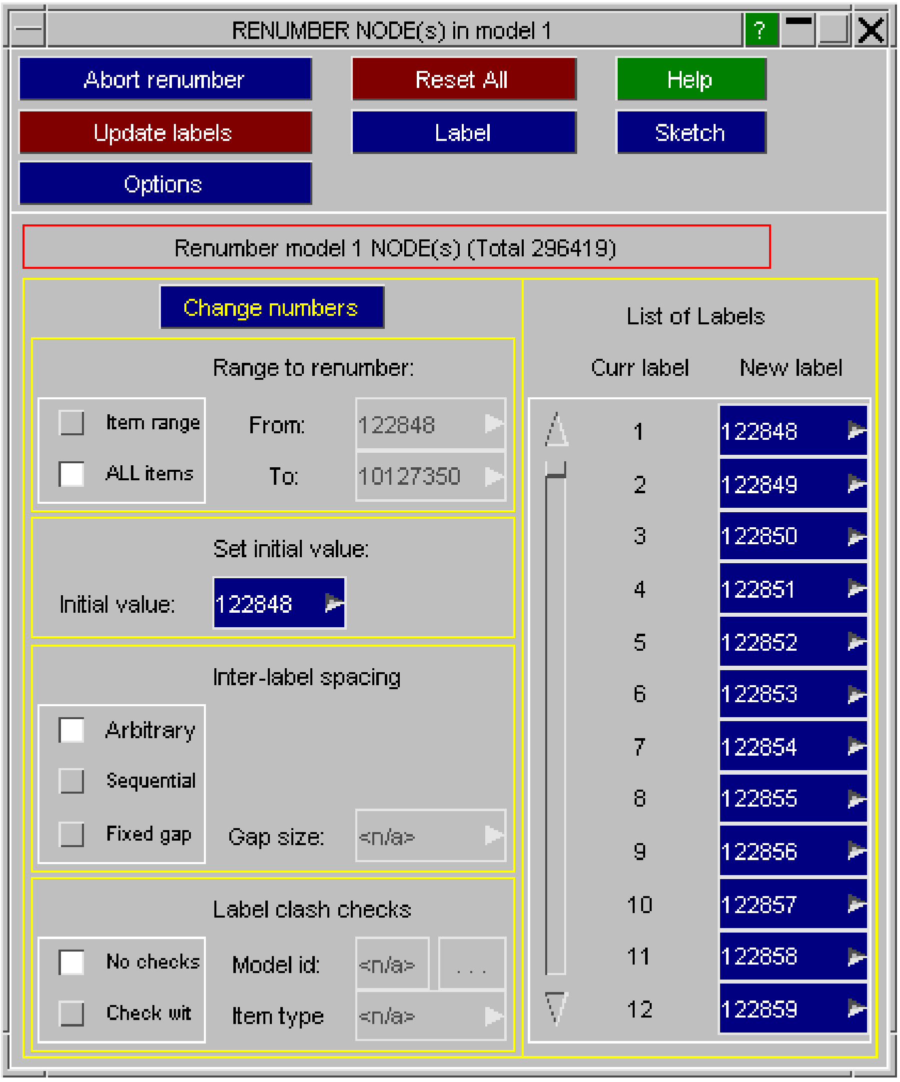
By clicking on a keyword button in the left hand column of the renumber contents panel, eg the **SOLID** button, you can invoke the standard item renumbering panel for that category, as shown in the adjacent figure.

Range to renumber: Select the range of items to be processed.

Set initial value: Choose the initial value for this range

Inter-label spacing: Set the gaps between labels

Label clash checks: Check for and eliminate clashes between categories



This panel is designed to let you change the labels of individual items, or a range of items, in this category (here solid elements have been chosen).

The left half of the panel allows you to select a range, and update any or all of:

- Its initial value. Default is the current start of the range;
- The gaps between adjacent labels. The default is the current ("arbitrary") gaps.
- Any clashes between these and other items. You can choose both the item type and the model id to check against, for example this user might check against solids in another model.

Clash checking against the following generic categories is also available:

- **ELEMENT** Any element type. Useful where no clashes are permitted between element numbers of any type.
- **SET** Any set type. Useful where no clashes are permitted between sets of different types.

When you have set up those changes you wish to make **APPLY_CHANGES** to see their effect. You may alter settings and repeat this operation as often as you like since you are only operating on a "scratch" definition.

The slider box on the right side of the panel both shows the current status and also allows you to renumber items individually: just type in a new label, or use the popup options.

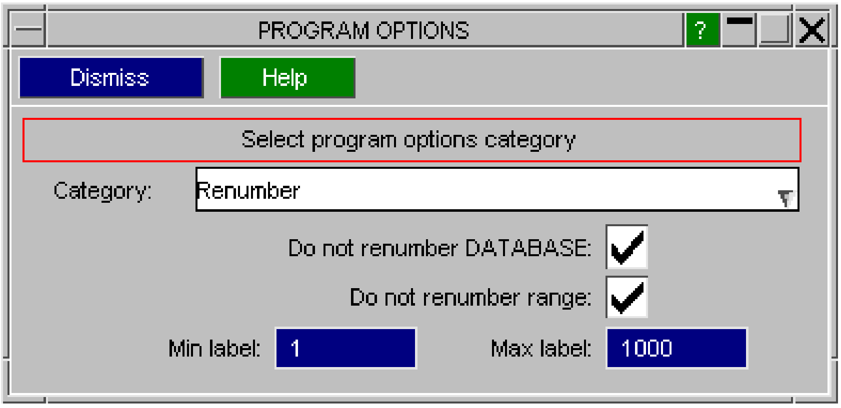
All operations within this panel operate on "scratch" labels.

To make these changes take effect in the permanent database you must use **UPDATE_LABELS**

Freezing entity labels during renumbering

Entity labels that lie within a user-specified range can be 'locked' during renumbering. This can be done by selecting the appropriate options in the Renumbering tab in the Program Options panel. The Renumbering options panel can be reached by clicking on the Options button either in the generic renumbering panel or in the category renumbering panel.

Likewise entity labels that are used by DATABASE_HISTORY cards can be 'locked' during renumbering.

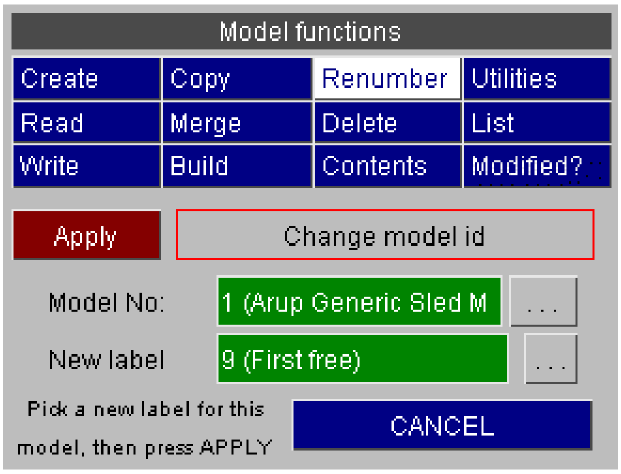


3.7.2 CHANGE MODEL ID

- Select the model number to change
- Select a new (unused) model label

Changing a model's id (number) does not affect its contents at all: its "id" is purely the number by which it is identified within PRIMER.

It is not possible to change a model id to that of another model which already exists, in order to achieve that you would first have to renumber or delete the target model.



3.7.3 CONDENSE MODEL IDS

Renumbers all current <n> model ids from M1 to Mn. (In effect from "arbitrary" model labelling to "sequential starting at 1".)

Condensing model ids has no effect on model contents.

3.7.4 RENUMBER SELECTION

Renumbering will only be applied to those items pre-selected through the object menu panel. Provisionally, the select items are to be renumbered **sequentially** starting at the defined start label ("START AT" option - the default) or to be **offse**t by a defined value ("OFFSET" option).

If, however, such renumbering would cause a clash of labels, some additional action must be taken.

Two options are offered for modifying labels of items **other** than the select items, to make sequential labels available for the select items:

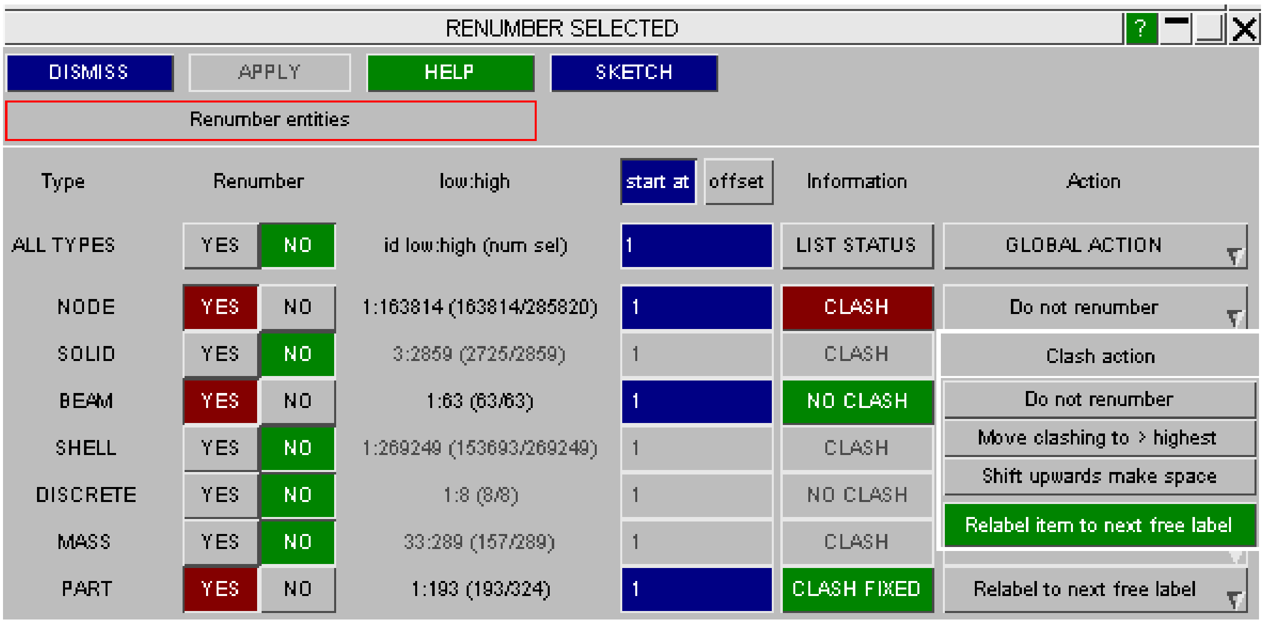
- Move any clashing labels to above the highest label for that type in the model
- apply an offset to all labels of the type, to shift all the labels clear of the sequential renumbering range

Alternately, rather than the select items being renumbered sequentially, one may:

- Renumber select items into the next available free label

In the case of "OFFSET" the only available clash fix is to move the offending labels to above the highest type label.

Latent items: Items which are referenced by a keyword but do not actually exist in a model are called latent. The renumber-selection function has been designed to avoid renumbering the labels of latent items. These labels are therefore reserved, and renumbering/clash fixing will always work around them. In the "OFFSET" case, if the required label for an item (current+offset) already belongs to a latent item, the item will not be renumbered.



The **SKETCH** function will sketch only those items that have been assigned for renumbering by activating "**YES**" button.

3.7.5 CONDENSE MATS

Models that contain multiple definitions of the same material, for example a material card for each part cards, may be tidied up with the **CONDENSE MATS** function.

Duplicate materials are detected by matching type, title(if any) and then comparing each entry (using a test of 5 significant figures for floating point values other than zero). If they are found the reference PID->MAT is adjusted to make them redundant. The user is then prompted to apply the deletion function to remove them.

The following **Options for Condense Mats** apply.

Curve inspection	**OFF**	Materials are only condensed together if the curves they refer to have the same labels.
Applies to fields LCSS and LCSR on MAT24 and MAT123 only	**ON**	If all other fields match, but the curve labels referred to are different, then the curves themselves are inspected. If the curve data points match, despite having different labels, then the materials are condensed.
Material titles	**Ignored**	Materials are condensed regardless of any mismatch between their title lines.
(Applies to all material types)	**Read**	Materials are only condensed if their titles match.

3.7.6 RENUMBER INCLUDES

- Select a model.
- Press **Renumber Includes** to get to the renumbering panel

New ranges can be specified for the master file, or for one or more include files using the appropriate text boxes. Ranges - for both general types and for nodes, elements, node sets and constrained nodal rigid bodies - can be generated for these files using the appropriate **Auto** button. An **Auto All** button is also available.

Upon selecting the **Apply** button, Primer evaluates the specified ranges to check whether renumbering would be necessary. If a given type has labels outside the specified range, Primer attempts to renumber those labels. Primer computes the number of labels of a particular type that exist outside the user-specified range. This is then compared with the number of unused labels available in the range (including the range labels). Users are warned if the specified range is not large enough to accomodate all labels. In that case, Primer renumbers as many labels as it can within the specified range. It then renumbers the remaining labels starting from the highest ID for that particular type. A warning is also issued if user-specified ranges for two or more files overlap. Nodes, elements, node sets, and constrained nodal rigid bodies are renumbered, as are general types that always carry a label. General types that support an optional ID are only renumbered if they carry an explicit label.

Additional information about overlapping ranges and about out-of-range items can be obtained using popups attached to two sets of status buttons (**Range unique?** and **In range?**). The list of include files can be sorted using a popup that is available at the top of the window. There is also a provision to copy node/element/nset/nrbc ranges into general type ranges using the **Copy ranges** button.

The **Read csv** button can be used to import user-defined ranges in the form of a .csv file. Likewise, current ranges can be exported to a .csv file using the **Write csv** button.

The **Range unique** button permits specification of generic renumbering options.

3.7.7 MAT24 LCSS/LCSR

This function will ensure that each material card of type 24/123 will have a unique table with unique curves. This will allow stochastic variation of material properties.

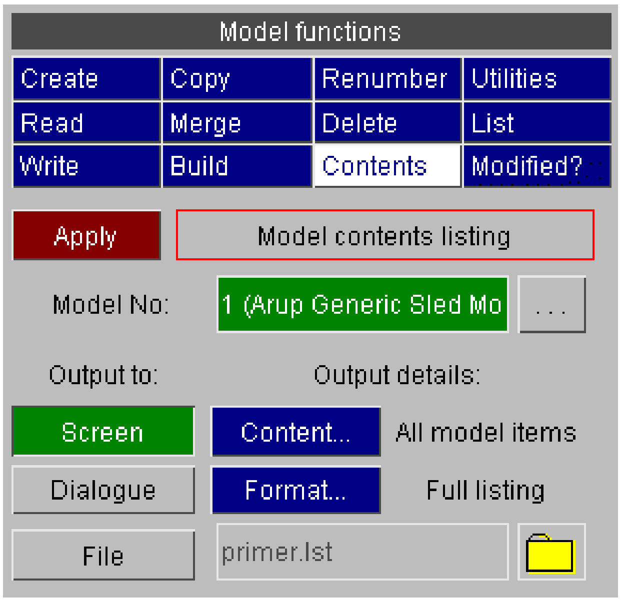
3.8 Model Contents

There are two ways of listing the contents of a model:

MODEL > CONTENTS	General listing of all model contents to screen and/or dialogue area and/or file
UTILITIES > WRITE_SUMMARY_FILE	More specialised list of Part, Material, Contact and Element properties

3.8.1 MODEL > Contents

It is possible to summarise the contents of a model to the screen and/or to file with this option.



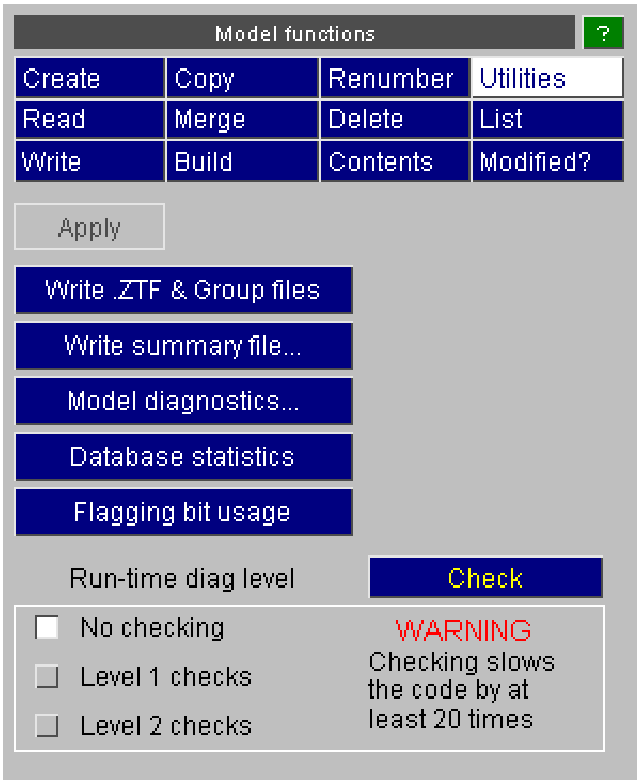
You can select any or all of the following "**Output to:**" locations:

- **SCREEN** A paged screen window.
- **DIALOGUE** The screen dialogue box
- **FILE** A disk file.

Output details: control what is written:

- **CONTENT...** May be all items (default), or just a listing of parts and contacts.
- **FORMAT...** May be a full listing, or a summary of the number of items of each type

3.8.2 UTILITIES



The **UTILITIES** menu allows you to:

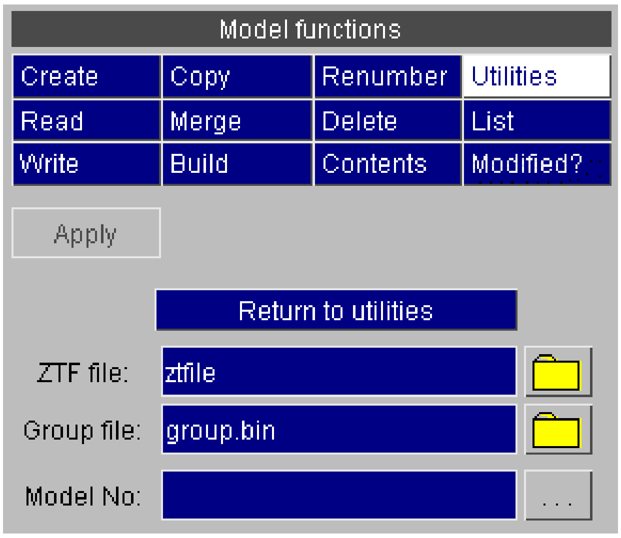
Write a summary file of the model

Write ZTF and group files

List flagging bit usage

The diagnostics and statistics options are described briefly below, but they are intended for use by Oasys Ltd developers only.

Writing ZTF and group files



The ZTF and group files are used by D3PLOT in order to visualise during post-processing information that is not available in the normal LS-DYNA results files. Only PRIMER can write these files.

The **ZTF file** contains information about nodal restraints, spotwelds, nodes on contacts, parts, sections, etc; which enables D3PLOT to draw and process extra information. This information is not available in the ptf, ctf or xtf files.

The binary **Group file** (.bin) is now superseded in D3PLOT, although it can still read it. Its output is preserved here for compatibility with older versions.

Writing a ZTF file

This may be carried out in any of three ways in PRIMER:

1. As shown in the panel here: fill in a file name (`<jobname>.ztf` is preferred) and **Apply** it.

2. If ZTF Output is selected in the **Model > Write** panel file `<jobname>.ztf` will be generated at the same time as the output .key file is written. See section 3.3.

3. If PRIMER is run in batch mode with the following command line arguments: (see Appendix XIII)

   `-d=batch -ztf=<ztf filename> <input filename>`

   . This method is used by the Shell to generate ZTF files automatically after an analysis has been run.

Writing a binary Group file (*deprecated*).

This file type was read by D3PLOT 9.1 and earlier, providing a method of exporting groups in PRIMER for use in post-processing. It had the disadvantage that, being a binary file, it was not only large, opaque and uneditable, but it was also "tightly" locked to the given analysis and could not be used with a different one.

Therefore from release 9.2 onwards an alternative way of exporting group information as an ASCII file via the **GROUPS->EXPORT** function was introduced: (see section 6.15 GROUPS)

- This is simply a copy of the **`*GROUP`** keywords (after **`*END`**) used by PRIMER to encode group information in the keyword output file.
- It format is compact, being typically less than 100 lines line.
- It is in a "human friendly" format and, being an ASCII file, it is manually editable if required.
- It is also flexible: any mis-match between the group file contents and the file being processed is simply ignored, making it usable over a wide range of broadly similar analyses.

ASCII groups file may also be read into PRIMER using **GROUPS->IMPORT** .

Although D3PLOT releases from 9.2 onwards will continue to read the binary groups file it is strongly recommended that you use the ASCII version instead. This feature may be withdrawm in future releases of PRIMER

WRITE_SUMMARY_FILE...

This command may also be used to generate a summary of the model contents giving more detail than **MODEL** > **LIST**.

It writes a file containing a part summary, material summary, contact summary, element summary and element quality summary. The default file name is **model_summary** but this can be changed at the prompt before being written out.

The part summary lists model contents by:
- Parts, in order of part ID, giving a break-down of material ID, material type, section type, gauge (if shell elements), part mass, timestep added mass and title for each part and part inertia.

The mass of each part is calculated as it by LS-Dyna on model initialisation. If nodes on deformable elements are attached to rigid parts, mass is lost by the deformable and gained by the rigid (or discarded if rigid is part-inertia). Lumped masses on rigid bodies (non part-inertia) are included in their part mass, whereas those on deformable bodies are added to the lumped mass total. If a rigid part is merged onto another, the slave loses its mass and this is added to the master (or discarded if master is part-inertia).

The mass contributions from parts, lumped masses on deformable, part inertias and nodal rigid bodies, the total model mass, and (if any) the timestep mass scaling as %age of model mass are summarised at the end of the part listing. Model mass is also written to the dialogue box.

The material summary contains a listing of:
- The commonly used materials in each model with a few useful details about each material e.g. E, and for each *MAT_ELASTIC.
- Note that not all material parameters are given for each material and not all material models are supported by the summary file. A list of materials which exist in the model for which no details are given is included at the end of the material summary.

The contact summary contains a list of:
- title
- contact type
- slave and master type

The element summary lists:
- parts by element type referenced, giving part ID, material ID, section ID, hourglass ID, thermal material ID and title.

The element quality summary lists:
- part ID
- percentage of elements failing one or more quality criteria
- minimum element length - worst value and the percentage of elements failing this criterion
- aspect ratio - worst value and the percentage of elements failing this criterion
- skew - worst value and the percentage of elements failing this criterion
- warpage - worst value and the percentage of elements failing this criterion
- minimum angle - worst value and the percentage of elements failing this criterion
- maximum angle - worst value and the percentage of elements failing this criterion

Model Diagnostics

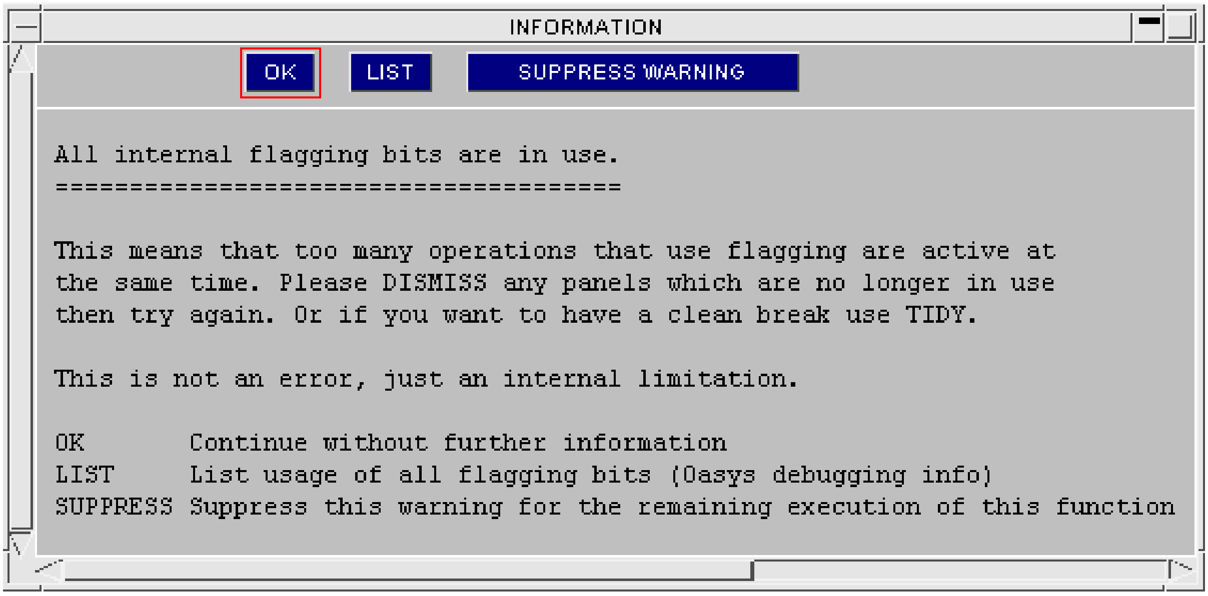
This option lists a summary of the main internal entity tables to the controlling terminal <stdout>. It is intended to help the developers and will not be of any help to ordinary users.

Database Statistics

This option is also intended for the developers. It writes to the controlling terminal <stdout> a list of memory allocation and usage by each internal data type, together with a summary of total usage. It may, with some guidance from Oasys Ltd, help with the debugging of memory-related problems.

Flagging bit usage

PRIMER makes heavy internal use of "flagging bits". These are literally bits in an internal word used when selecting items for processing and, unfortunately, they are a finite resource which can at times become exhausted. If you try to perform too many functions at the same time you may see the following message:



LIST will give a table of flagging bit usage by model, allowing you to work out what you need to close down in order to free some flagging bits for the wanted operation.

In order to avoid having to provoke this message in order to find out which bits are being used where, this command will list them on demand.

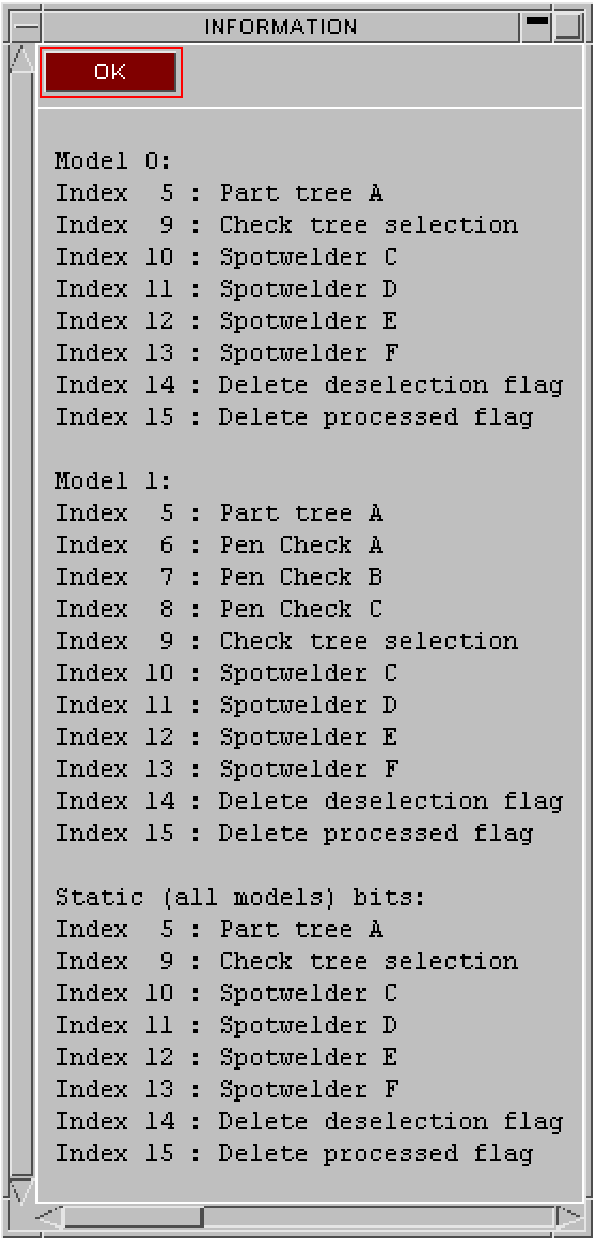
The example listing here shows that:

- The Part tree is using one bit. This is always the case and cannot be changed.
- The Contact penetration checker is using 3 bits in model #1.
- The Model Check panel is using 1 bit in model 1.
- The spotwelder in the Connections panel is using 4 bits in all models.
- The deletion panel is using 2 bits in all models.

PRIMER release 9.3 has a total of 19 dynamically allocatable flagging bits, numbered 5 to 23. As this example demonstrates some operations (eg DELETE) allocate bits in all models, whereas others (eg contact penetration check) are specific to a particular model.

Bit exhaustion can occur even if not all bits in a particular model are in use, since "all model" operations require that bits have the same number in each model.

Therefore when shutting down panels to free flagging bits it is usually most effective to close those which allocate "all model" bits.



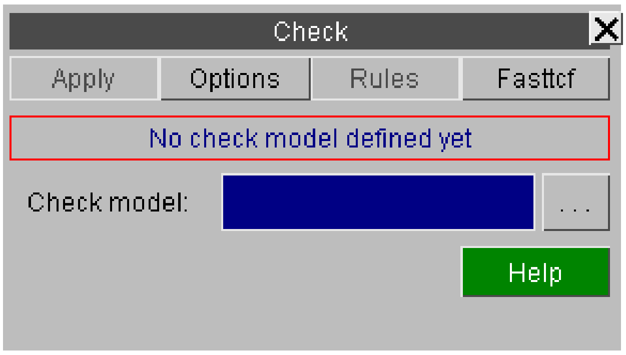
Run-time Diagnostics.

As with the other diagnostic options above the run-time diagnostic level is intended for use by the developers. It monitors internal memory allocation to detect "memory leaks". This would never normally be switched on by a user.

The **Check** option lists to the controlling terminal a snapshot of pointer allocation and integrity

.

3.9 MODEL > CHECK



PRIMER offers a powerful checking feature to validate models before the user submits the model in LS-DYNA. Over 3000 individual checks are perfomed to check a whole model for grammatical, contextual and other errors. In addition to flagging fatal errors (invalid fields, cards and so on), many of these checks will also improve the quality of the model analysis. PRIMER separates the check results into 2 types: errors and warnings. **Errors** usually indicate a model that will not initialise within LS-DYNA or will be of poor quality, **warnings** are indications of bad modelling practise and/or integrity. The results of the check can be viewed in a table format or navigated using an intuitive tree structured viewer similar to the part table.

In addition, a FASTTCF file can also be checked (for details on this type of file see the T/HIS manual - section 7). The **CHECK** option can be found in the tools menu.

TO CHECK A MODEL:

- Set up any checking **OPTIONS...** required
- Select a model
- Press **APPLY** to run the checker (more than 3000 checks are applied).
- Press **RULES** to run a set of user defined or custom checks (below) .

The whole model is checked, which may take a little time, and a summary of any errors found is given, together with any that can be "Auto-Fixed". A summary table and error tree viewer can then be utilised to navigate through the error categories and sub-categories.

A combination of **CHECK** and **REMOVE** > **CLEANUP_UNUSED** will go a long way to eliminating most errors in models.

Most **KEYWORD >** panels and **CREATE/EDIT** panels include a **CHECK** option for checking specific categories or items, which is the way to check individual items. The checking routines are the same in all cases.

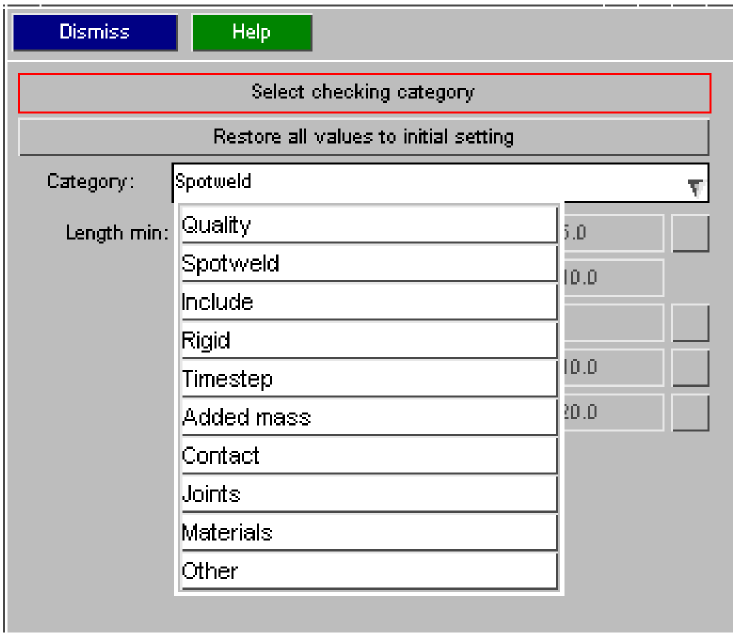
FASTTCF CHECK:

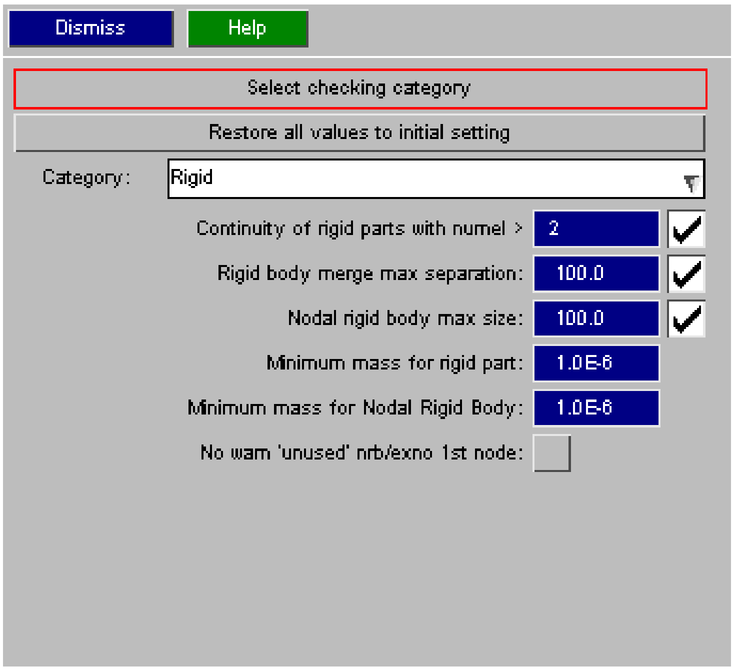
- Press **CHECK FASTTCF FILE** to continue to the FASTTCF check menu.

3.9.1 OPTIONS... Setting model check options

The check option panel is now divided into pages for ease of use.

Use the popup to go to the page you want.

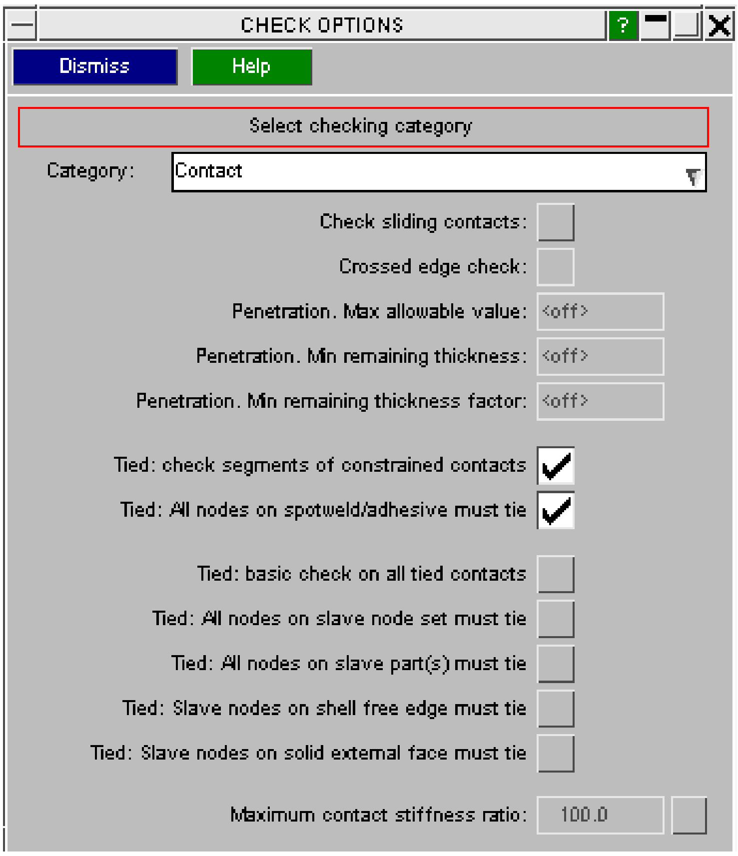




These are the options that control checks related to rigid entities

Options for which zero is a valid entry will have an ON/OFF tick box to control whether the option is active.

For others entry of zero (or blank) de-activates the option.



These are the options that control checks related to contacts

Sliding contact - by default the checks are **off**. If active, crossed edge checking is done. Additionally, penetrations may be reported if they fail one of 3 thresh-hold criteria.

- the max allowable value of penetration (typically 0.5mm)
- the minimum remaining unpenetrated segment thickness (typically 0.5mm)
- criterion B expressed as ratio of average segment thickness (typically 0.7)

Tied contact - by default constrained contacts will be checked for clashes with one another and connection contacts will be checked for any untied nodes. In the general case, there is no rule for which of slave nodes are supposed to tie so a variety of options (all off by default) have been added.

- basic check - will give contact error if no node is tied
- slave node set - error if any contact with slave side defined by node set does not tie all the nodes
- slave part -error if contact with slave set defined by part/part-set or shell-set does not tie all the nodes
- free shell edge - error if slave nodes on free edges of shell do not tie
- solid face - error if any external nodes on solid do not tie

The majority of PRIMER's checks are automatic and built into the software. These flag up fatal errors that would stop the model initialising in LS-DYNA and errors which could mean that the model is unreliable or may fail to run to completion. However, PRIMER enables the user to turn off/on many of its model integrity checks to allow the user to set a custom level of model warnings. The options available to control the level of checking have expanded considerably in recent versions of PRIMER. Most of these options can have an initial state set using the **oa_pref** file. For example:

Contact Penetration Checks Controls whether contacts are checked for penetrations and crossed edges. Set the following line in the oa_pref file: `primer*contact_penetration_checks: ON (default OFF)`

The following is a summary of the check options in PRIMER. Most options can also be set as user preferences.

LS-DYNA version number (LS970, etc)

- Because earlier versions of LS-DYNA imposed some restrictions on the valid label range for certain items (for example a maximum of 99999 load curves) this version number can sometimes be significant when checking.
- The version number is the same as that used in the keyword output section.

Element quality checks (ON/OFF/PREF) - by default each check set to PREF value (or OFF if this is unset)

- A master control for whether element quality checks are performed
- A single check may be set on or of by setting a switch and giving a value (if needed), for example: **primer*element_length_check**: ON and **primer*element_min_length**: 6.0
- by default element checks are at their individual settings. However, **primer*element_quality_checks**: ON (or OFF) activates/de-activates all checks and will over-ride any individual settings

Spotweld checks - by default these are ON

- The Length min/max refer to lengths of individual beams within a spotweld
- The minimum distance between welds and maximum number of panels to attempt to weld can also be set here

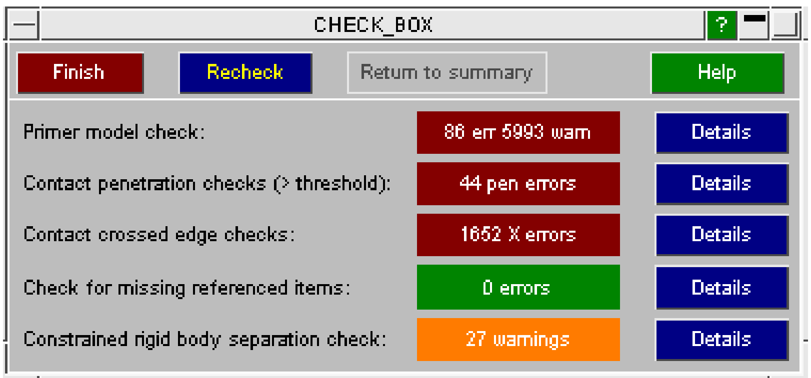
The other checks are self-explanatory, providing the user with several mass checks and model quality checks. A few worth noting are:

- **max separation for rigid body merge** checks the distance between merges to detect a merge that has accidentally been made across the model
- **nodal rigid body max size** warns if nodal rigid bodies as measured by the diagonal of an enclosing box exceed defined size
- **check mat24 curves to strain (FAIL=0)** it is an error for MAT24 load curves to cross the x axis before failure strain is reached, similarly the curves in a table should not cross over one another below such strain. By default (FAIL=0) the failure strain is very large (1e21) so the curve will get extrapolated and two curves may converge in theory. In reality extrapolating strain to this amount is unrealistic, users may prefer to limit the check to a realistic strain value (e.g. 100.0).
- **check for duplicated elems in same part** - elements which share the same nodes & part, also called overlap check
- **part quality** will warn if more thah a user defined %age of elements of a part have failed the user defined quality checks
- **minimum mass for rigid part** the default is <auto> which means Primer attempts to calculate the minimum mass of the rigid part required to keep the adjoining deformable elements stable under conditions where model has added mass. Such mass is effectively lost at the nodes which join the rigid part, hence its mass must be sufficient to compensate. Users may set their own minimum mass value.

The current element quality check settings apply to both standard checking and APPLY_RULES checking. To invoke the latter the rules checks must be activated.

Note: some checks may require intensive operations, e.g. Model %age added mass requires calculation of model mass. On very large models, this may make the checking process appear slow.

3.9.1.1 Rules check



The **CHECK** >**RULES** function applies a set of custom checks which can be controlled through the oa_pref file or by the **OPTIONS** panel.

The standard model check and element quality checks are as described above. The model checking is always run without contact checks in this mode.

Contact check - if these are active, for example by the oa_pref settings

- **primer*contact_penetration_rule**: ON
- **primer*contact_penetration_max_allowable_value**: 0.2

Rules check will run the contact checker directly and report penetration (above thresh-hold) and crossed edge count. In this case **Details** button will access Primer's penetration checker via an intermediate panel.

Rigid-body separation check - measures the distance between the centroids of merged rigid bodies and reports those that exceed the given value.

This check is now done as part of model checking if the option is active. For historic reasons, it is available as separate feature in rules check. **Details** will start a bespoke visualization panel. The oa_pref settings are:

- **primer*rigid_body_merge_check:** ON
- **primer*rigid_body_merge_max_separation:** 200.0

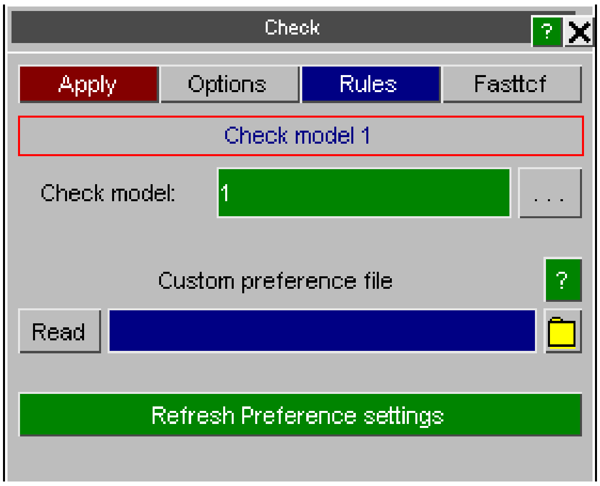
Missing item check is always done. If, for example, an element references a part, but the part card is absent from the model, the missing item check will detect it.

For model check, the **Details** button will list take the user to the standard checking panel.

Whenever a custom check is made a summary is dumped to the text file **apply_rules.txt** written in cwd.

3.9.1.2 Custom oa_pref file

On start up Primer reads the system oa_pref file (in the $OA_INSTALL directory) and then the home oa_pref file (in the home area) and finally any oa_pref file in the current working directory (which is an uncertain concept on windows machines). The last preference file may be considered to be a custom pref file. Primer now has a more effective way of handling these which disposes of the need to put the file into cwd.



Under **CHECK > OPTIONS** there is a function for selecting and reading the custom pref file. If you have changed any check options previously in this session, it is advisable to run **Refresh Preference settings** to restore the settings to their state at start up. Note that reading a preference file will set to ON or OFF any preferences that are present in the file, but omitting a preference **does not turn it OFF**. Therefore, you m ust ensure that your any settings which are activated by home or system preference file are actively turned OFF in the custom pref file if you do not want them.

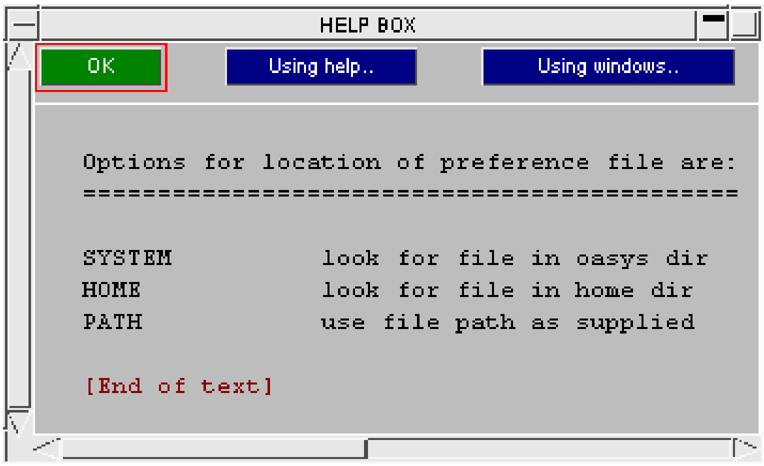
The custom pref file may be specified as a command line argument "-pref=xxx", see command line arguments.

A custom file may be applied for model checking by using batch file. For example, the following command file will read the custom oa_pref file "pr90.pref" from the system area

/pref system pr90.pref

/read dk pr90.key 1

/check full list model 1 apply



3.9.2 CHECK output

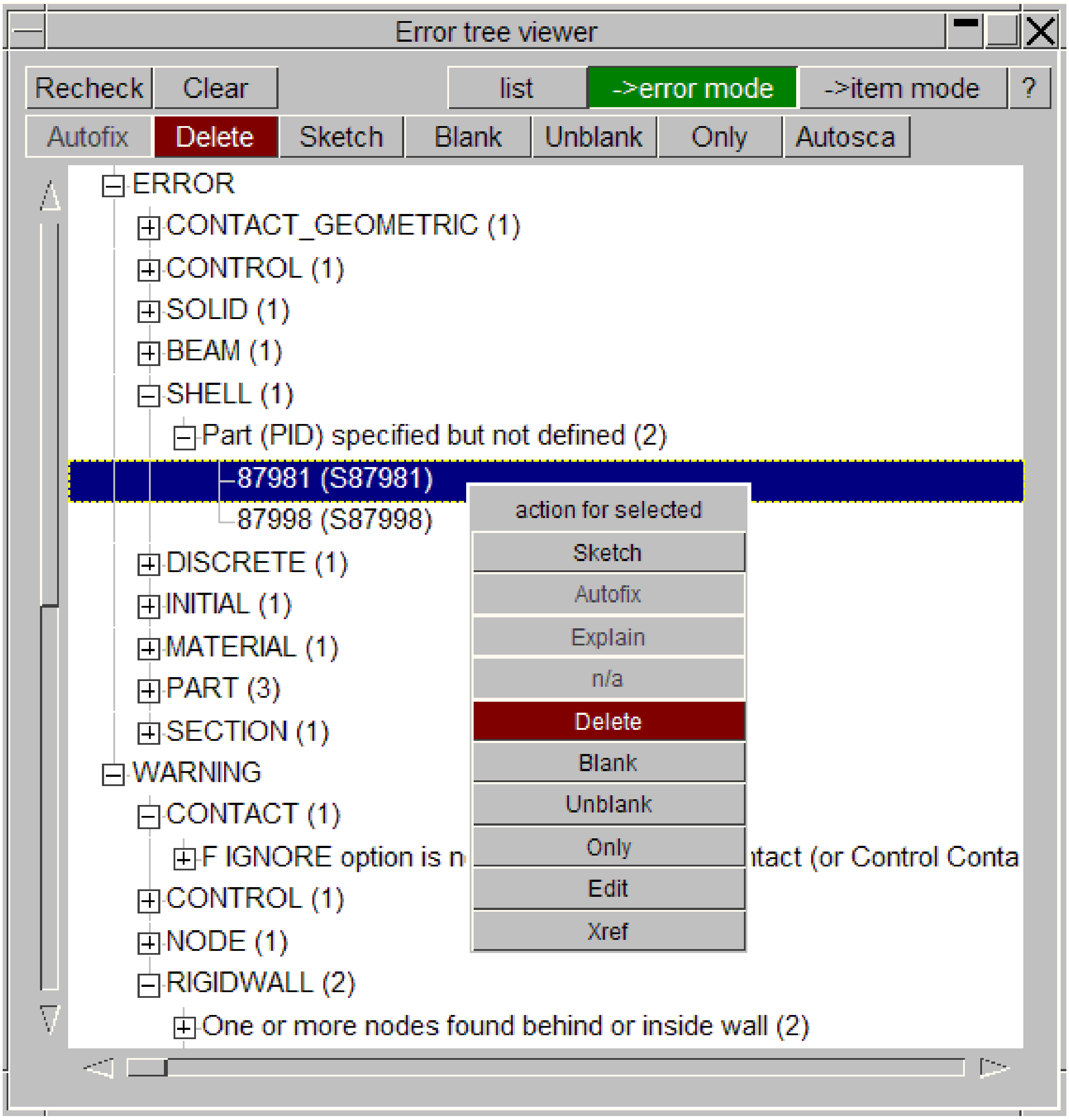
Two panels will appear during the model checking:

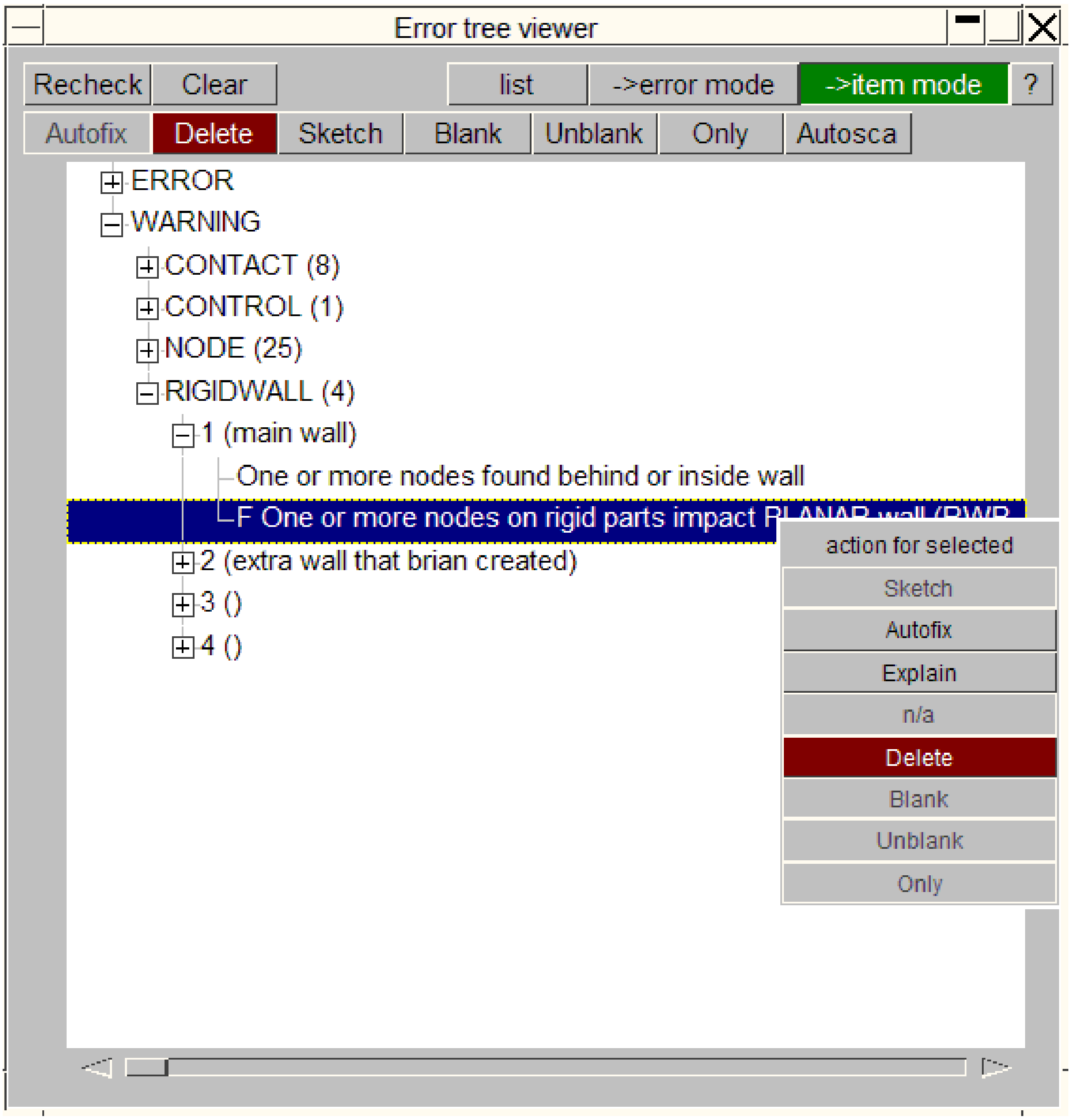
1. **Error tree viewer** this groups the errors and warnings into categories and sub-categories. Intuitive navigation through the checks is possible via a tree system
2. **Check model summary panel** this gives a compact summary of the errors and warnings found in a table format, with a popup from each entity type

3.9.2.1 Error tree viewer

When PRIMER has performed the model check, the results are grouped into errors and warnings and then sub-divided into entity categories. Each category has further divisions to group the checks together, depending on the mode selected:

1. **>error mode** lists under each error code the entities which exhibit the error. This is the default mode.
2. **>item mode** lists under each entity which has been found to have an error, the error or errors which pertain to it


Error tree viewer
Recheck
Clear
list
->error mode
->item mode
?
Autofix
Delete
Sketch
Blank
Unblank
Only
Autosca
ERROR
CONTACT_GEOMETRIC (1)
CONTROL (1)
SOLID (1)
BEAM (1)
SHELL (1)
Part (PID) specified but not defined (2)
87981 (S87981)
87998 (S87998)
DISCRETE (1)
INITIAL (1)
MATERIAL (1)
PART (3)
SECTION (1)
WARNING
CONTACT (1)
F IGNORE option is n
ntact (or Control Conta
CONTROL (1)
NODE (1)
RIGIDWALL (2)
One or more nodes found behind or inside wall (2)
action for selected
Sketch
Autofix
Explain
n/a
Delete
Blank
Unblank
Only
Edit
Xref



Navigation of the errors and warnings is very simple due to the hierarchy within the error tree. Categories can be expanded or contracted using the small box to the left of the headings. Once the categories have been expanded a number appears to the right of the heading - showing the number of sub-categories directly underneath this level.

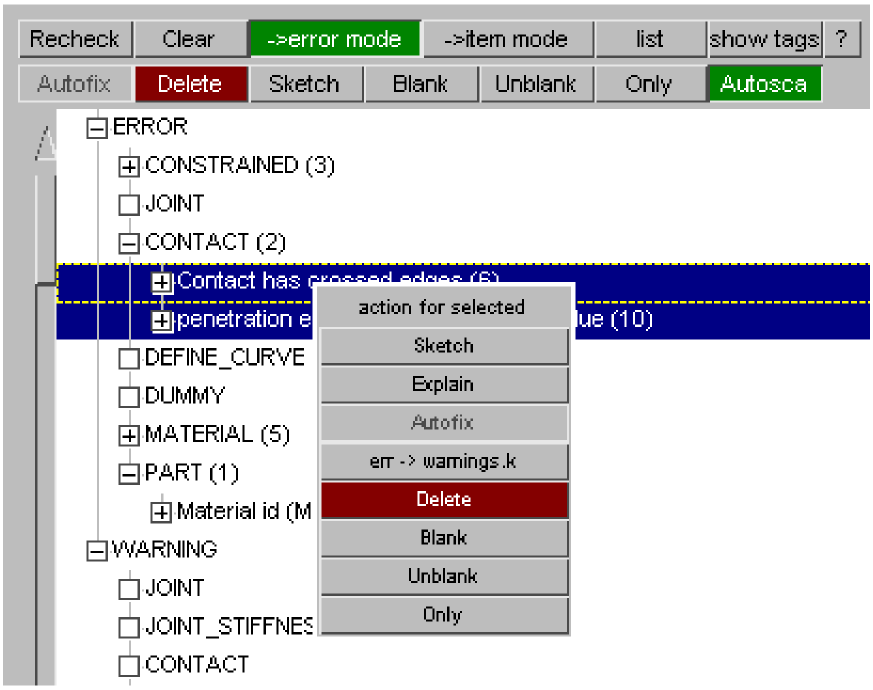
Various operations are available once an error(s) has been selected. Multiple selections of errors are warnings are possible using the shift and control keys (similar to the part tree behaviour). Viewing operations can be utilised through the buttons at the top of the panel to **sketch**, **unblank**, and manipulate the model display to help identify the problem.

List will give a concise summary of the errors and warnings in a text box.

Right-clicking on an selection brings up a popup (the figure shown above). Here, if possible the selection can be **sketched**, **autofixed** (this is explained below), **explained** (a quick summary of the problem), **deleted**, and viewed in different ways. The user can also **edit** the entity (if a single entity has been picked) or look at its **xrefs** within the model using the cross-reference viewer in a separate panel.

On completion of an autofix or deletion of item in error, in consideration of speed with larger models the recheck function will be run only on the type directly affected. In most cases this action will be suffiecient. However, sometimes fixing one error will fix other errors implicitly and occasionally it maycause new errors to appear. It is recommended, therefore, that you do a full recheck when you have completed you fixes by pressing **Recheck.**

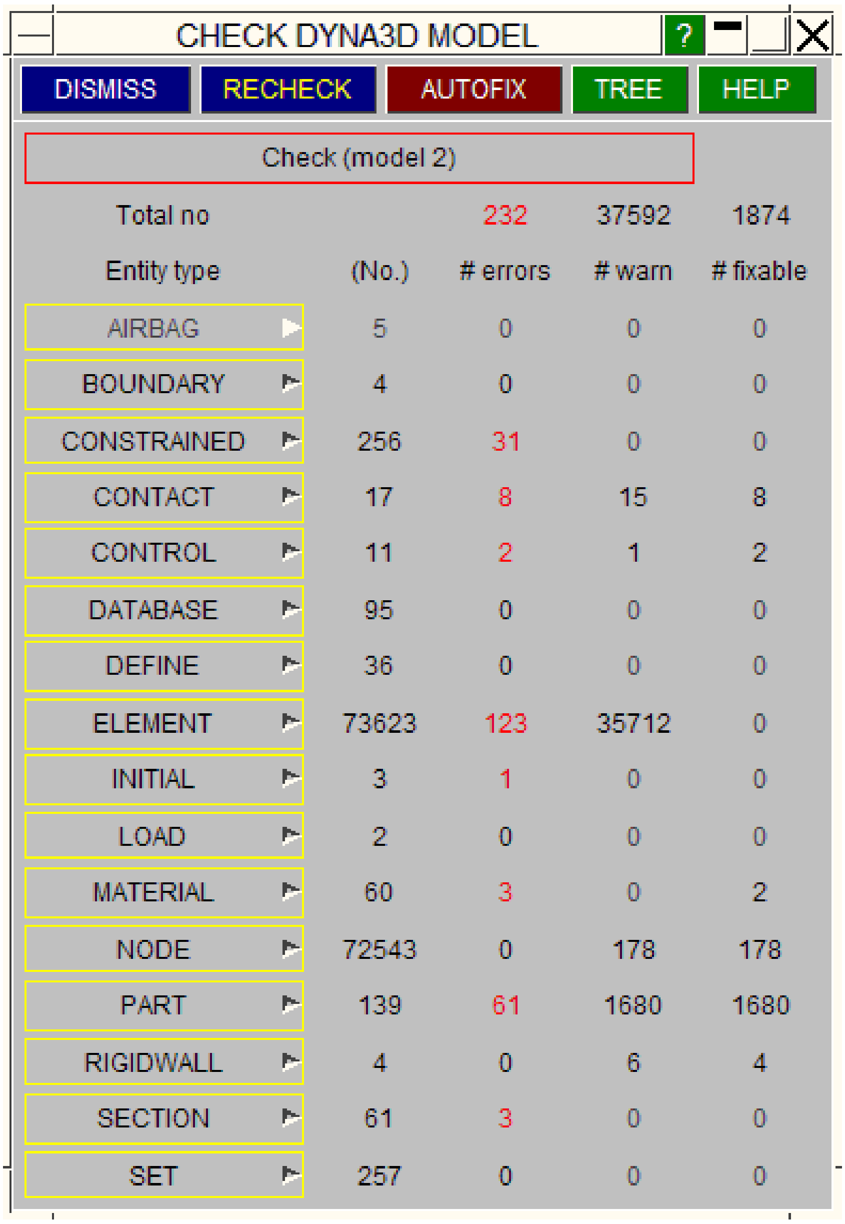
The drop-down from the check tree allows Nodes and Elements in error to be written to a set, parts to be put onto the table, and contact penetration and tie errors to be written to sets in a separate include (warnings.k) file. Pressing **err->warnings.k** will run the contact checker and generate node and segment sets each with a title corresponding to the error detected.



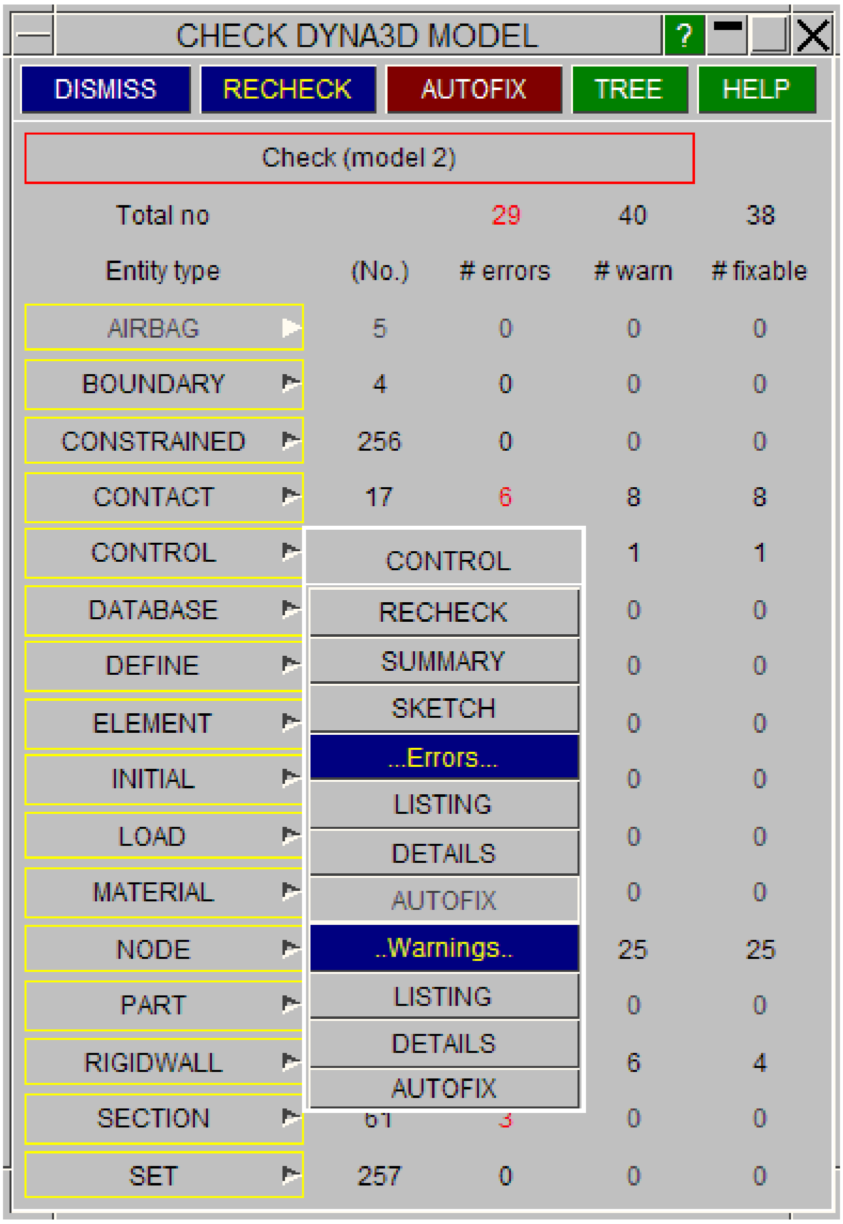
3.9.2.2 Summary table panel

The second check panel gives a concise summary of every master keyword category in the model, with a listing of:

- **(No.)** The total number of items of that keyword
- **#errors** The number of errors found
- **#warnings** The number of warnings found.
- **#fixable** The number of these errors/warnings that can be "Auto fixed"



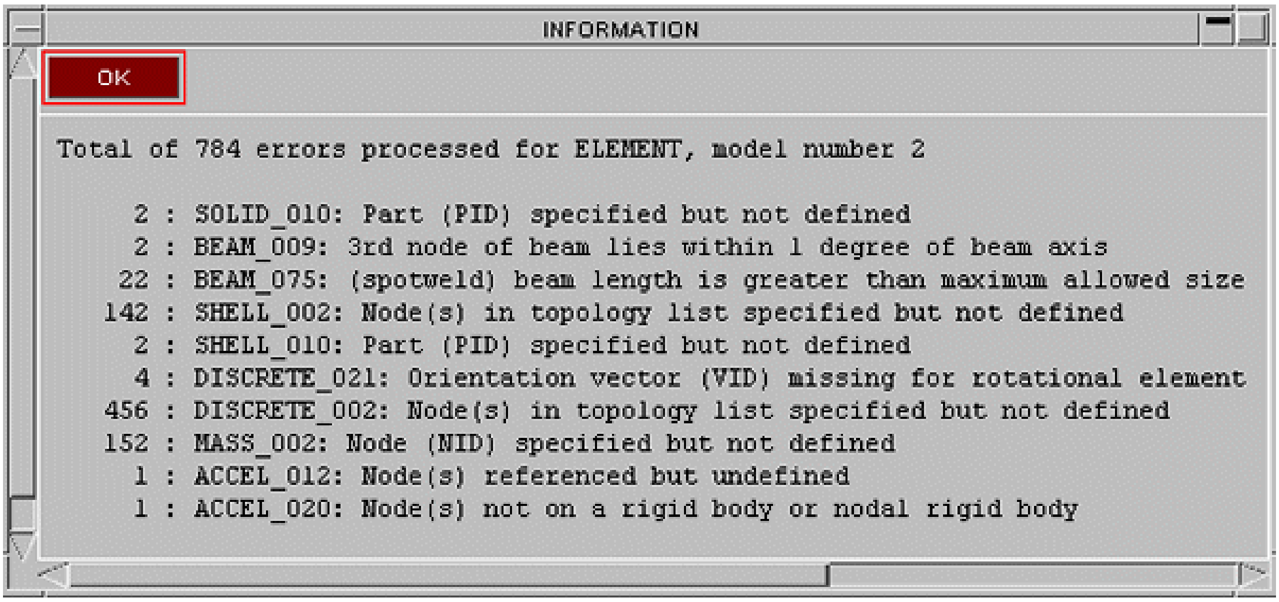
You can then use the popup menu of each category to examine problems in more detail. In this figure the user has used the popup menu for category **CONTROL**. The popup displays warnings and errors in separate areas.



Click on the links for descriptions of:

SUMMARY
LISTING
DETAILS
AUTOFIX
SKETCH

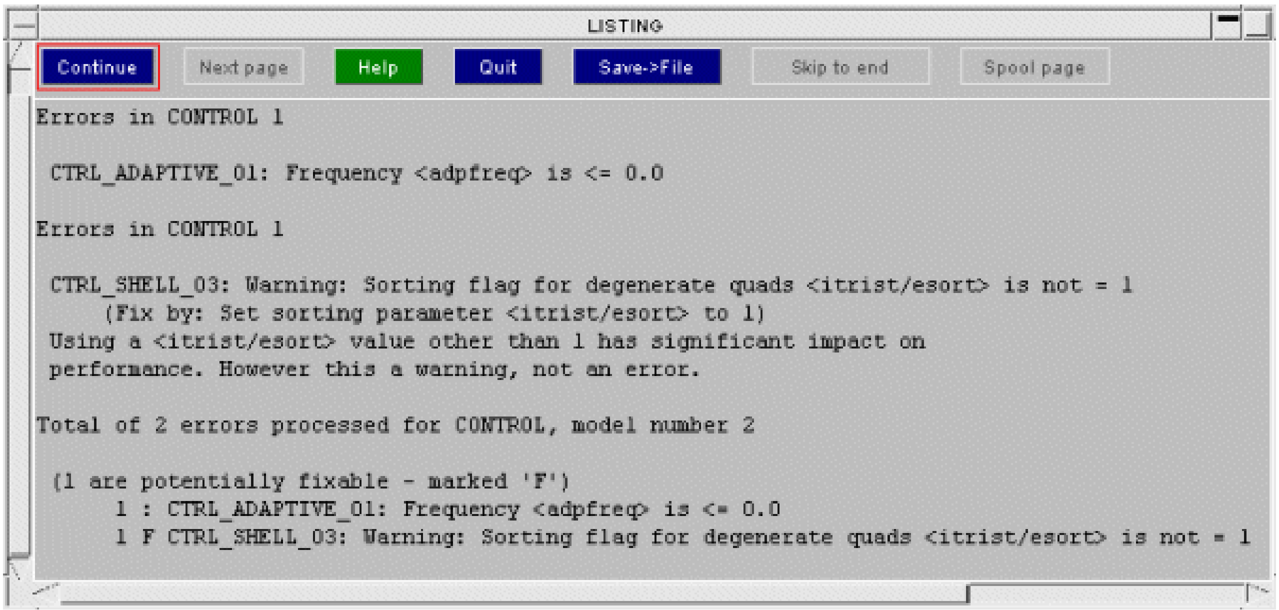
3.9.2.2.1 > SUMMARY



Lists only the number of occurrences of each class of error for this category.

Here ELEMENTS are listed.

3.9.2.2.2 > LISTING

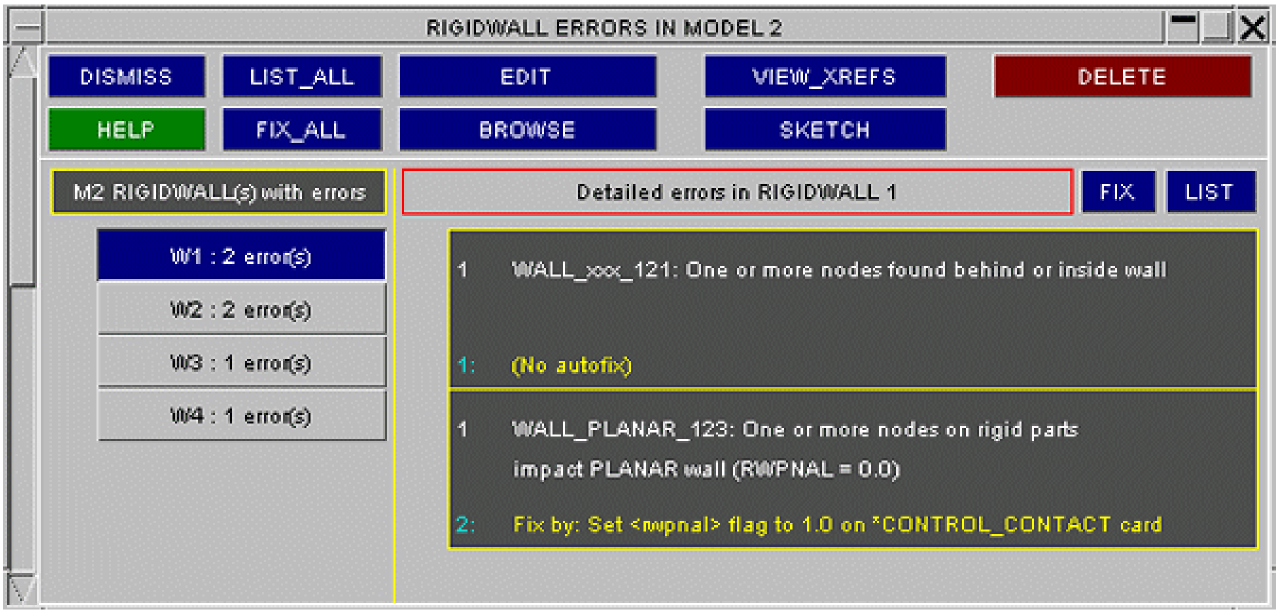


For each item with errors lists the details of the error and any extra information available.

If it can be "Auto-fixed" a description of how is also given (as here).

This can be a long listing, so it is paged and can be saved to disk file with **SAVE->FILE**.

3.9.2.2.3 > DETAILS

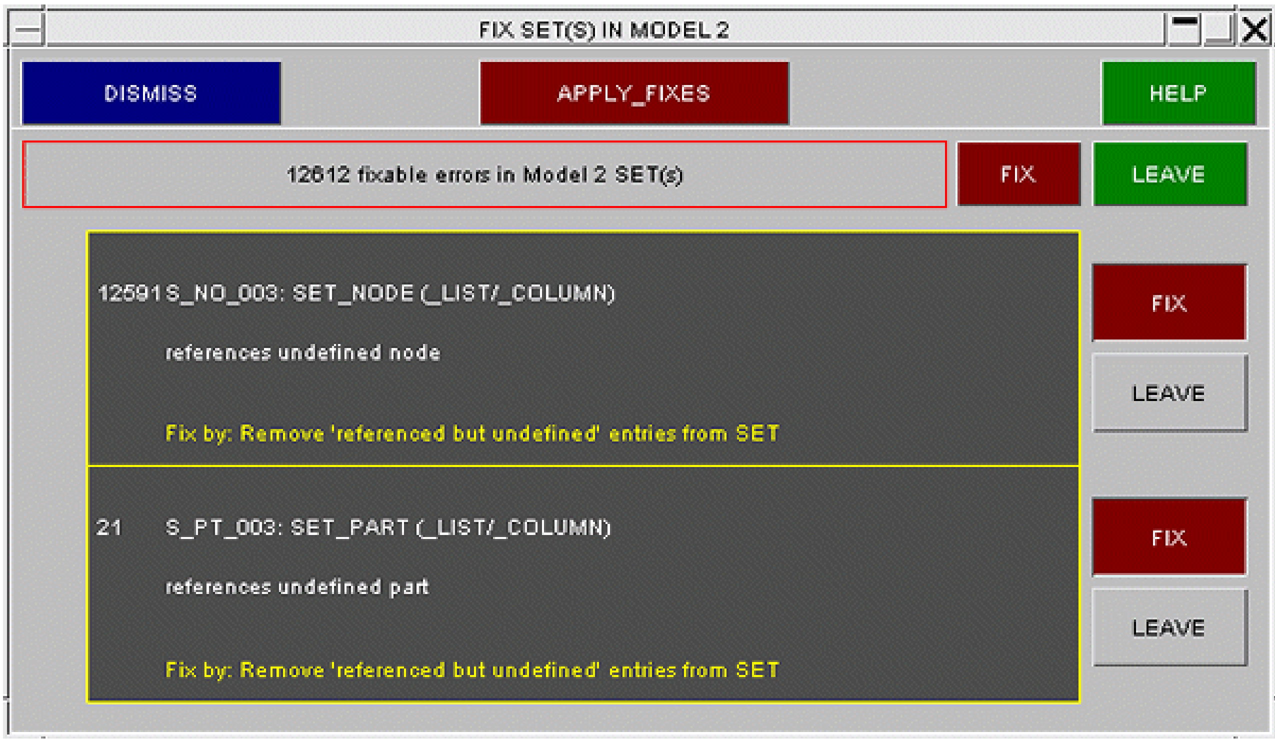


For each category builds a panel containing a list of the items with errors down the left hand side.

These can be selected in turn to view their specific errors, and to fix them (or not) by whatever means you wish.

Auto-fixes may be applied on a case-by-case basis.

3.9.2.2.4 > AUTOFIX



Builds a panel containing all fixable errors for this category.

You can choose to **FIX** or **LEAVE** each error.

APPLY_FIXES will then correct what you have selected.

In this way Auto-fixes may be applied to all errors in this category.

3.9.2.2.5 > SKETCH

The items in the category which contains errors may be sketched on the current image.

3.9.2.3 Auto-Fixing errors.

"Auto-fixing" is where PRIMER offers to correct errors it has detected for you. Generally PRIMER will only offer to make fixes where it can do no harm, so typical fixable errors would be:

- "Out of range" values, which would be reset to defaults.
- Duplicate entries in sets or time-history blocks, which would be eliminated.
- Reversing element topology to make area or volume +ve.
- Eliminating segments that don't lie on element faces.

. Auto-fixes can be applied:

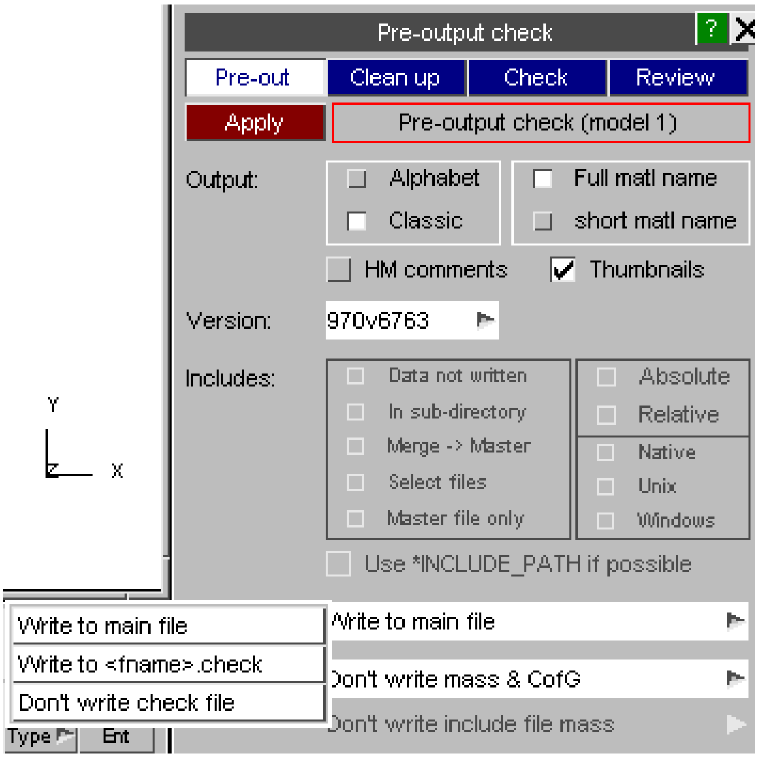
- On an entity by entity basis via the **DETAILS** popup in the check summary panel;
- On a whole category basis via the **AUTOFIX** popup in the check summary panel;
- Using the autofix option when right-clicking in the error tree viewer.
- To the whole model via the **AUTOFIX** button at the top of the check summary panel.

3.9.2.4 Notes on checking:

- All keyword categories present in the model are shown, but those for which checking functions are not available are greyed out. These will follow in the fullness of time.
- That no errors have been detected is no guarantee that there are none! The checking algorithms in PRIMER are not based on those in LS-DYNA, and they pick up some that DYNA misses, and vice-versa.
- Checking may be done at any time, but is good practice to check models prior to output, and in fact this is the default behaviour in the model **WRITE** command.
- A check that produces many errors is sometimes due to the consequences of deleting items leaving an "untidy" model. Try a **REMOVE**, **CLEANUP_UNUSED** operation before trying more detailed corrections, since it may sort out most of the problems.

3.9.2.5 Keyout error check

On keyout a listing of the results of model checking can be written either to the top of the keyword file or to a new file which will have the same name as the keyword file but with a **.check** extension. This may be set up on the pre-output check panel or by making appropriate **oa_pref** settings.



3.9.2.6 Batch error check and autofix

Checking may be called from the command line using the syntax: **CHECK MODEL <n> CHECKFILE <filename> APPLY** or **CHECK MODEL <n> APPLY.**

You may include the option **FULL_LIST** if you want a detailed listing of every item label in the file, otherwise labels will not be printed for the more populous items types (such as nodes and elements).

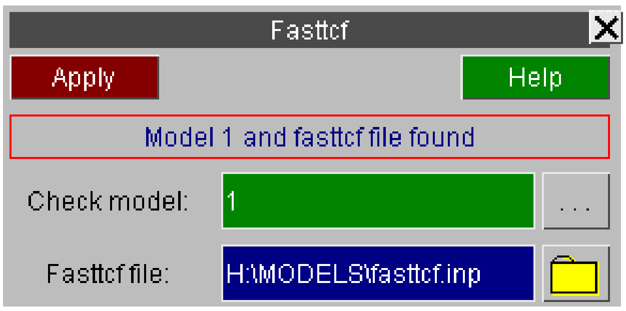
Similarly the Autofix all function may be called in command line using the syntax **AUTOFIX MODEL <n> APPLY**.

3.9.3 FASTTCF CHECK

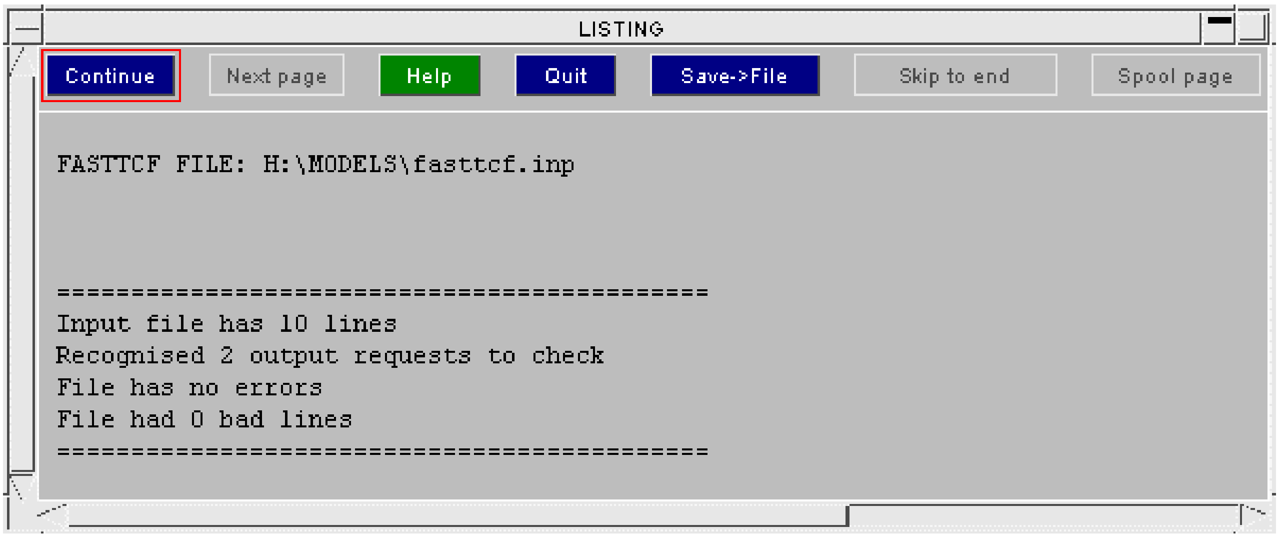
This section analyses a FASTTCF file and checks any data extraction requests for errors. The main checks are as follows:

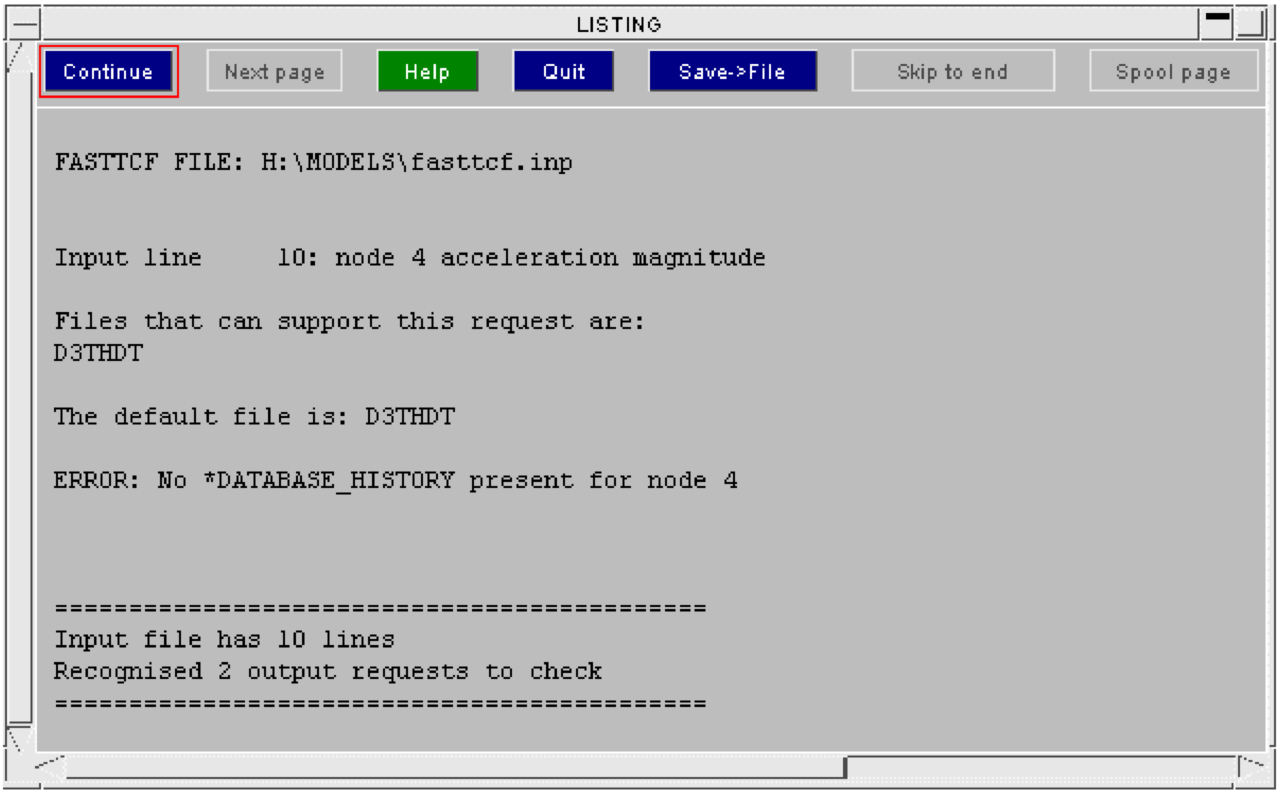
1. The request is supported by FASTTCF
2. The file requested for the extraction is being outputted from DYNA
3. The data extraction request is included in the *DATABASE_HISTORY output if it needs to be
4. If the entity i.d. exists or if it is latent

Choose the fasttcf file to check, and the model to check against. Then press the **APPLY** button.



Once the file has been checked a report text box appears to highlight any errors found. The following two images are examples of the output text. The first has no errors reported and the second has an error regarding the database history output. To solve the error in the second example the *DATABASE_HISTORY_NODE should have node i.d. 4 added.





3.9.4 Error vs Warning: User configuration

By default Primer offers two levels of configuration - error and warning. We have assigned "error" status to, cases where the error will prevent a model initializing, is likely to result in a model failing to run to completion or to give physically sensible results and those where a user defined criterion is infringed. We have assigned the less severe "warning" status to cases which will not stop a model running but could have an adverse affect on the result.

This is not an exact science. So users now have been provided with the ability to configure errors for themselves. Within Primer the any existing error can be given user status which will over-ride the default. Additionally a string may be attached to the error which can be identified in the output.

The implementation is as follows:

Set up a comma separated error config file which consists of one line per error in the format - error tag, user config, optional extra comment, e.g.

JNTC_13, ERROR, FATAL ERROR
JNTC_14, ERROR, FATAL ERROR

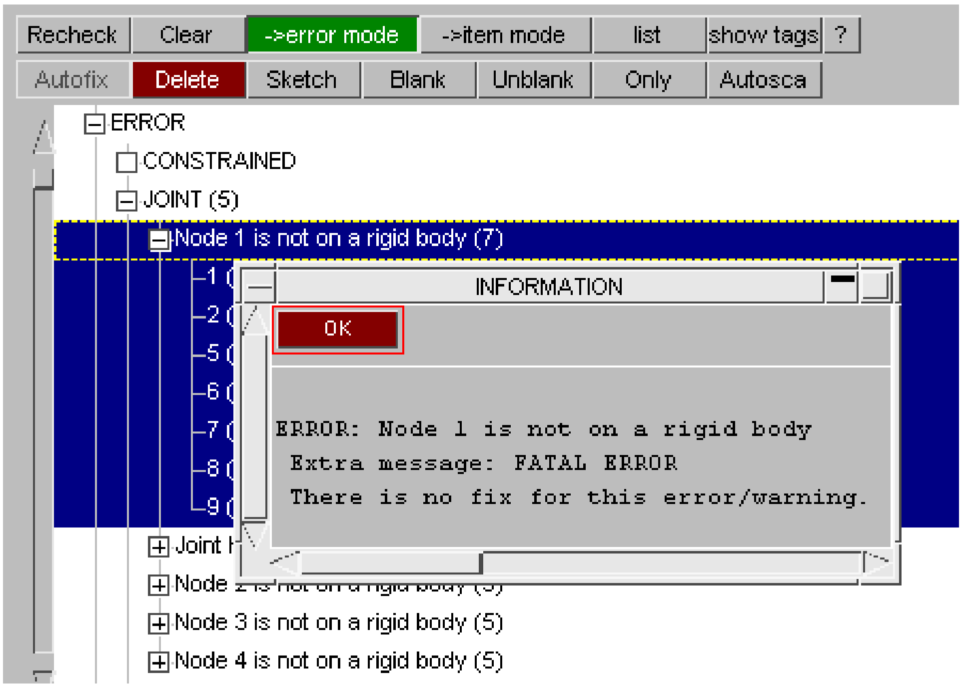
JNTC_15, ERROR, FATAL ERROR
JNTC_16, ERROR, FATAL ERROR

The **error tag** is the unique code which identifies the error and can be displayed on the error tree (you need to have a model in which the error exists and to activate **show tags** on the error tree). The **user config** may be ERROR, WARNING or IGNORE which will determine how the error will be displayed in Primer (if at all). The **optional extra comment** will be printed in the error summary output file and may be detected by your own process

Define the oa_pref setting: **primer*error_configuration_file: /path/filename.**

```
Drive mapping u: -> \u\mid
Error configuration file processed: c:\tmp\config.csv
User configuration will be applied.
Primer > _
```

If the error file exists Primer will read it on start up and report this in the dialogue box.



On the tree you can access the extra message via the "explain" popup on the main error message

```
Error/Warning listing for model: <untitled>


=============  ERRORS  ===========

CONSTRAINED ERROR
 RIGID_BODIES: Slave part <pids> referenced but not defined (1)
 RIGID_BODIES: Slave part <pids> is not rigid (1)

JOINT ERROR
 Node 1 is not on a rigid body (7)
 Extra message: FATAL ERROR
 Joint has massless node. (9)
 Node 2 is not on a rigid body (9)
 Extra message: FATAL ERROR
 Node 3 is not on a rigid body (5)
 Extra message: FATAL ERROR
 Node 4 is not on a rigid body (5)
 Extra message: FATAL ERROR
```

When you write an error file, the extra message is included and can be detected by the controller.

The check listing can be written by using using a command file and running Primer in batch mode. Users can then use their own process to detect the presence of significant strings (such as "FATAL ERROR" in this example) and take the appropriate action.

3.10 Operations on models

Once in memory models can be drawn; they can also be translated, rotated, reflected and scaled via the **ORIENT** command. Models may also be deleted, and their contents edited in a variety of ways. These and other operations are described elsewhere in the manual, in particular:

Section 5: **Keywords** Describes how to create, edit and process *keywords

Section 6: **Tools** Describes how to use the various "tools" in PRIMER

3.11 Viewing models

Models become visible as soon as they have been read in.

The default action when a model is input is to calculate its max/min dimensions, then to display it, autoscaled if necessary, in the current plotting mode: the default mode can be set in the oa_pref file (see appendix XIII for details).

There are a range of commands which affect what is visible, how it is drawn and what labelling takes place. These are described in section 4, but a summary is:

- Model visibility is controlled globally via the "**Mnnn**" buttons under **MODEL** > **LIST** (see section 3.0.1). When depressed (green) a model is potentially available for viewing, when up (red) the model is removed from the view list.
- Classes of entity (eg Shells, Nodes, Constraints, etc) may be made visible and, optionally, labelled using the Entity Visibility controls (see section 4.4). These flag an entity class for display and/or labelling across all models. This panel can be accessed by the shortcut key E, the top bar menu **DISPLAY** > **ENTITIES** or the button **ENT** from the viewing drawing window.
- Any item, or range of items, can be made visible or invisible using the **BLANK** command. "Blanking" may be applied in a hierarchical fashion to models, or subsets thereof, down to individual items. See Section 4.5 for more information on Blanking.

Basic drawing itself takes place in one of three modes: (see section 4.1)

- **LI**ne "Wireframe", with no hidden-surface removal.
- **HI**dden Also wireframe, but with hidden surface removal applied.
- **SH**aded 2D and 3D items are drawn shaded and lit, with 1D and other items superimposed in hidden mode.

In addition items may be **SKETCH**ed on top of the current image. "Sketching" superimposes a wireframe (unhidden) sketch of the relevant items on top of the current image, in an alternate colour and without clearing the current image. Sketching is not affected by the entity switches or blanking settings.

Data-bearing items may also be contoured or otherwise displayed using: (see section 4.2)

- Vector plots (**VECT PLOT**). Arrows or similar symbols, for example of initial velocity.
- Continuous Tone (**CT**) or Shaded Image (**SI**) contour plots, for example of timestep, shell thickness, etc.

3.12 Memory management and usage.

PRIMER stores all data in memory, therefore it must manage memory efficiently despite reads, deletes, merges, copies etc. This is done by allocating chunks of memory by data category as they are required, and returning these chunks to the relevant free list when that data is deleted. For example when a node is deleted the space required to store its data is returned to the "node data" free list for re-use the next time a node needs to be stored, and so on for all internal categories.

However there is some overhead associated with this and, like middle-aged spread, memory consumption tends to grow as more operations are carried out. In addition many create/delete operations will lead to greater memory fragmentation, and thus more page-faults, so the performance of the programme will degrade.

This is not usually a problem with small models, but when you start to manipulate larger models you may experience some performance degradation as you approach the memory limit of your computer. There are some things you can do to alleviate this:

- Don't have more models in memory than you require at any one time, and perhaps consider writings models temporarily out to disk (and then deleting them from memory!) before reading in new models.
- Try to avoid unnecessary read/delete/read/delete cycles. This will cause a steady build-up of memory consumption, and also increased fragmentation.

- If merging a succession of large models consider doing the job in stages: merge (say) 2 or 3 models, write out the result, then exit PRIMER and re-enter it to do the remainder. This will lead to memory being more organised, and hence give a faster response.
- Try not to run other memory-hungry processes on your computer at the same time as a large PRIMER session.
- There is a small saving to be made by using X-Windows graphics rather than a 3-D protocol, and also by trying to avoid drawing large images with lots of added labels, symbols, shading, etc.

Tests on this release of PRIMER (v9.3) suggest that just to read in and display a typical model each 1,000,000 nodes and elements requires:

- **32 bit version**: 600 MBytes of memory
- **64 bit version**: 800 MBytes of memory

Actually working with models, and in particular performing memory-hungry operations such a merging, contact penetration checking and spotwelding can push this requirement up to 1GByte per model; and this figure should be used when estimating the memory required for a workstation.

In practice, the memory needed for a workstation or desktop is more likely to be controlled by post-processing needs than by PRIMER.

The **UTILITIES** > **DATABASE_STATISTICS** button can be used to provide a summary of current memory usage. This is intended primarily for the programmer to use during debugging, but it provides a useful general guide to how efficiently PRIMER is managing its space. The total consumption it reports will always be an underestimate since it only lists memory used for data storage.

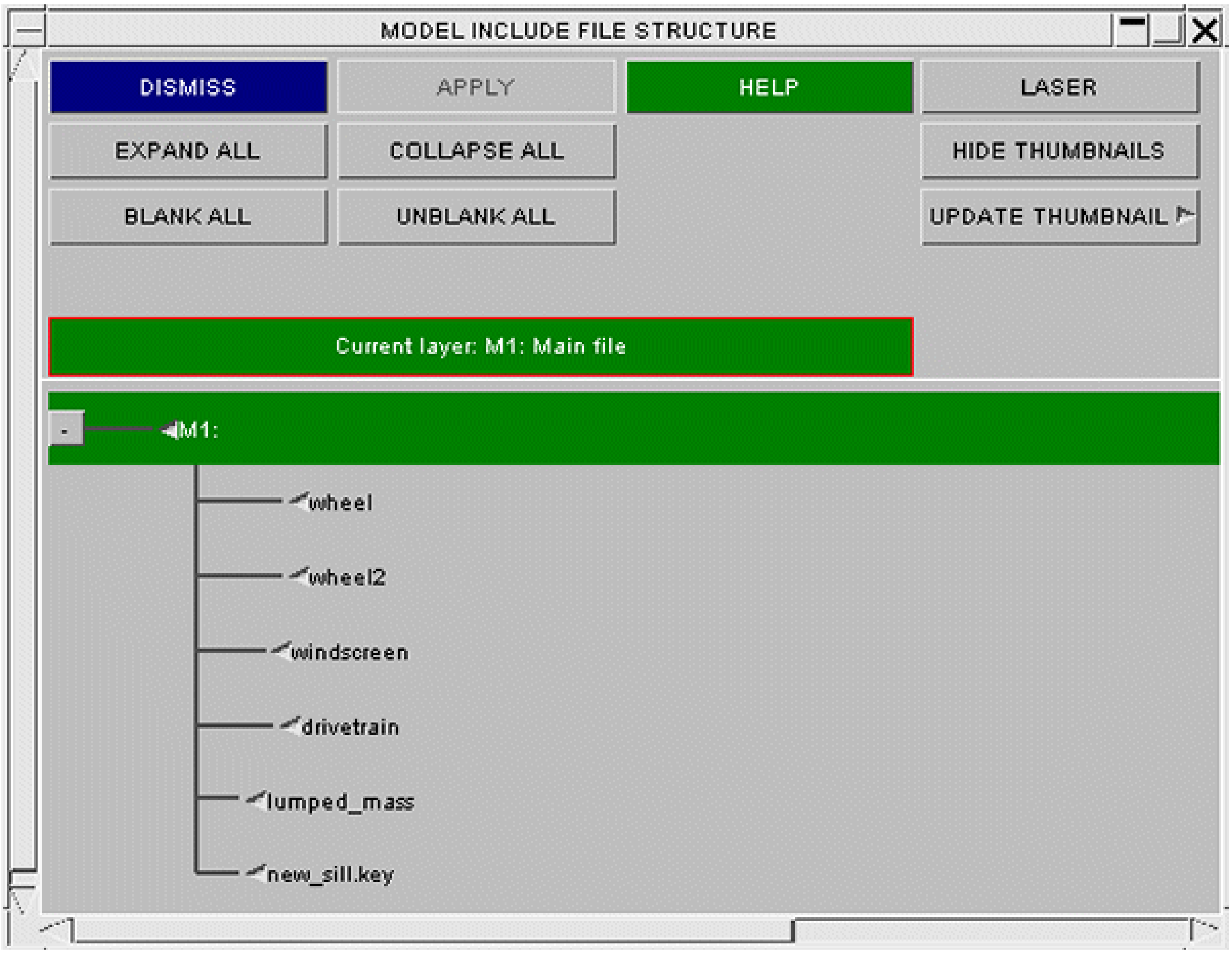
To see actual memory usage use:

Under Unix / Linux	The "ps" command. eg "ps -ealf | grep primer", or the "top" command
Under Windows	The Task Manager, Processes tab, Peak Mem Usage and VM size columns

3.13 Include Files

Include files provide a simple way to break up large analyses into smaller, more easily managed files

3.13.1 Include File structure



The "**INCLUDE**" button in the tools panel will invoke a tree diagram (shown above in **EXPAND ALL** mode) describing the include structure of the models in memory. Models built of include files are referred to as "**Master models**", the include files themselves are generally "**Component models**" (which will usually be valid LS-dyna models in their own right) and "**connection files**".

Include files offer an easy, robust method to organise your analyses into a file structure, represented above as a tree, of smaller (component) files which can be edited individually.

The "Master keyword file" or "Root file" (at the top of the tree) references all the include files which reside at the first layer of depth. These files themselves may then contain include files at second layer of depth, and so on.

The Include files can exist in different directories to the Master file allowing the user to organise the them with flexibility (e.g. under different directories in a database). On keying out the model each include file is referenced after its *INCLUDE statement, either with the **full path** or with the **path relative to the master file**. See **MODEL > WRITE**.

3.13.2 Reading in selected Include Files

If Master Model already exists

If Master Model is to be created

3.13.2.1 Master Model already exists

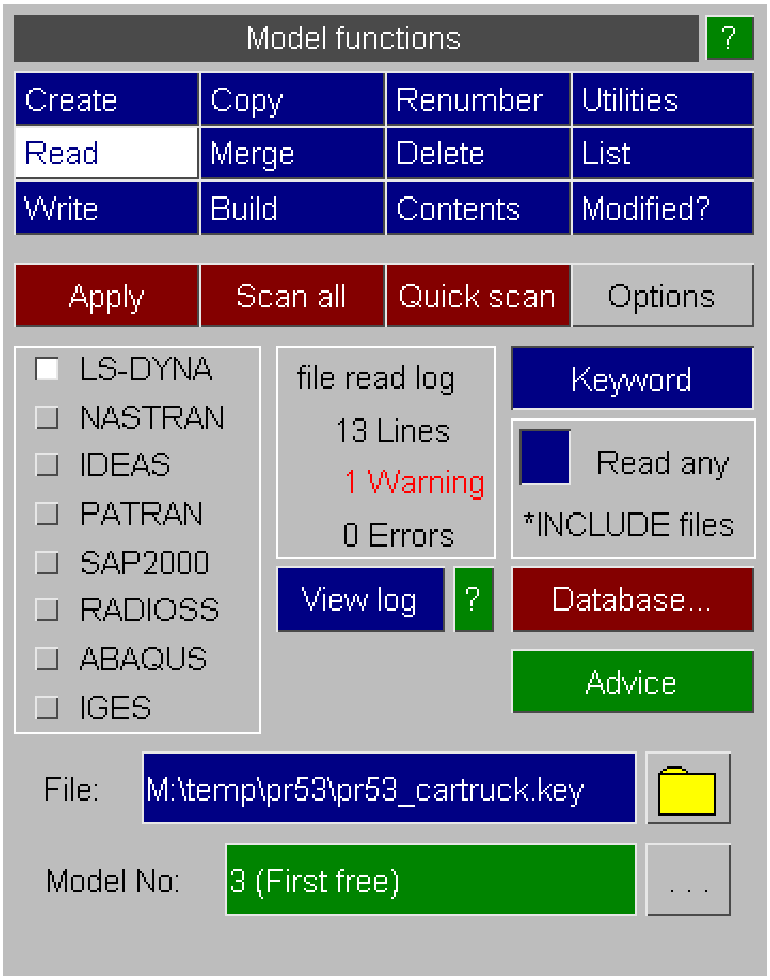
From the Menu, select **Model**, **Read** and use:

- **Scan all** to look for all include files, including those "nested" as include files within include files.
- **Quick scan** to look only for include files in the master file.

This scans the input deck. "Scan" in this context means look only for include file information, but don't actually import any normal keyword data into a model.

Scanning is much faster than reading since no internal computation or storage allocation is required; normally it is limited only by the disk or network speed of the machine.

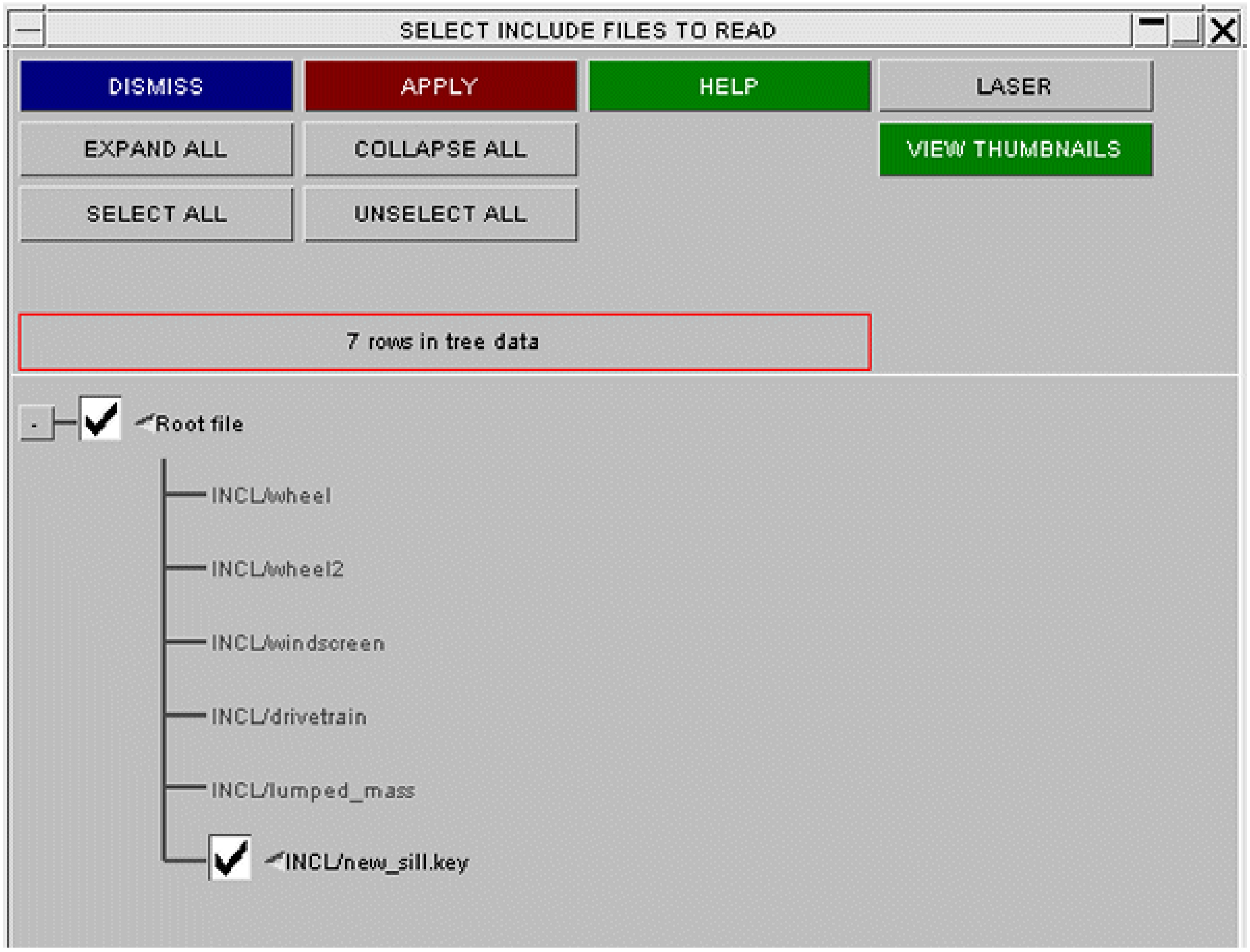
Once complete this operation maps the panel below which allows you to control what is to be read in.



In order to view all the Include files present in the Master Model, Select **EXPAND ALL**

Select the files you wish to read in (more than one include file can be read in at once) and press **APPLY**. Select options are invoked by a right mouse click on the popup.

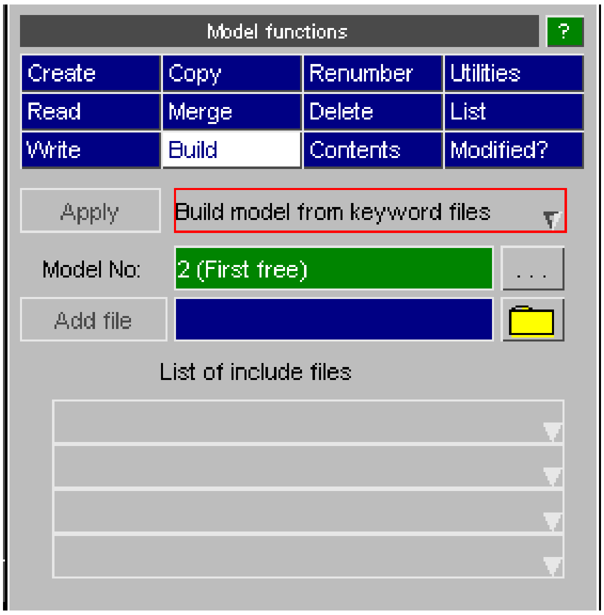
NOTE: When reading Include files into PRIMER it is important to ensure that you read them into the same Model in order to allow all the references across include files to operate successfully



3.13.2.2 Master Model is to be created

From the Main Menu, select **MODEL**->**BUILD**

Simple build from keyword files



If there are **no label clashes** between the include files, a simple build can be used.

The Include files you wish to read in are selected one by one, by inputting the name into the text box and pressing **ADD FILE**. The files will appear in the list.

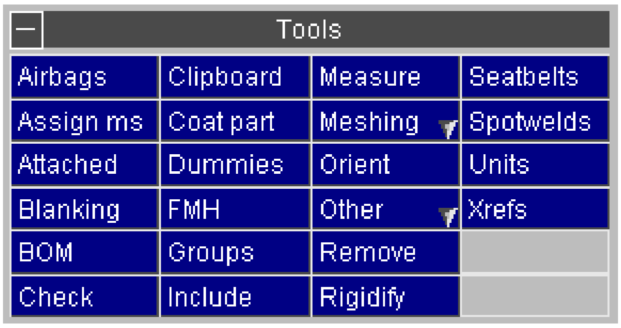
Once you have selected all the Include files you wish to read in, press **APPLY** and the model will be built with the added files at the first layer of depth. These files may themselves contain include files, thus the method may be used to create a multi-layered include file structure.

The more complex method of building a model from a database is described in section on Model database.

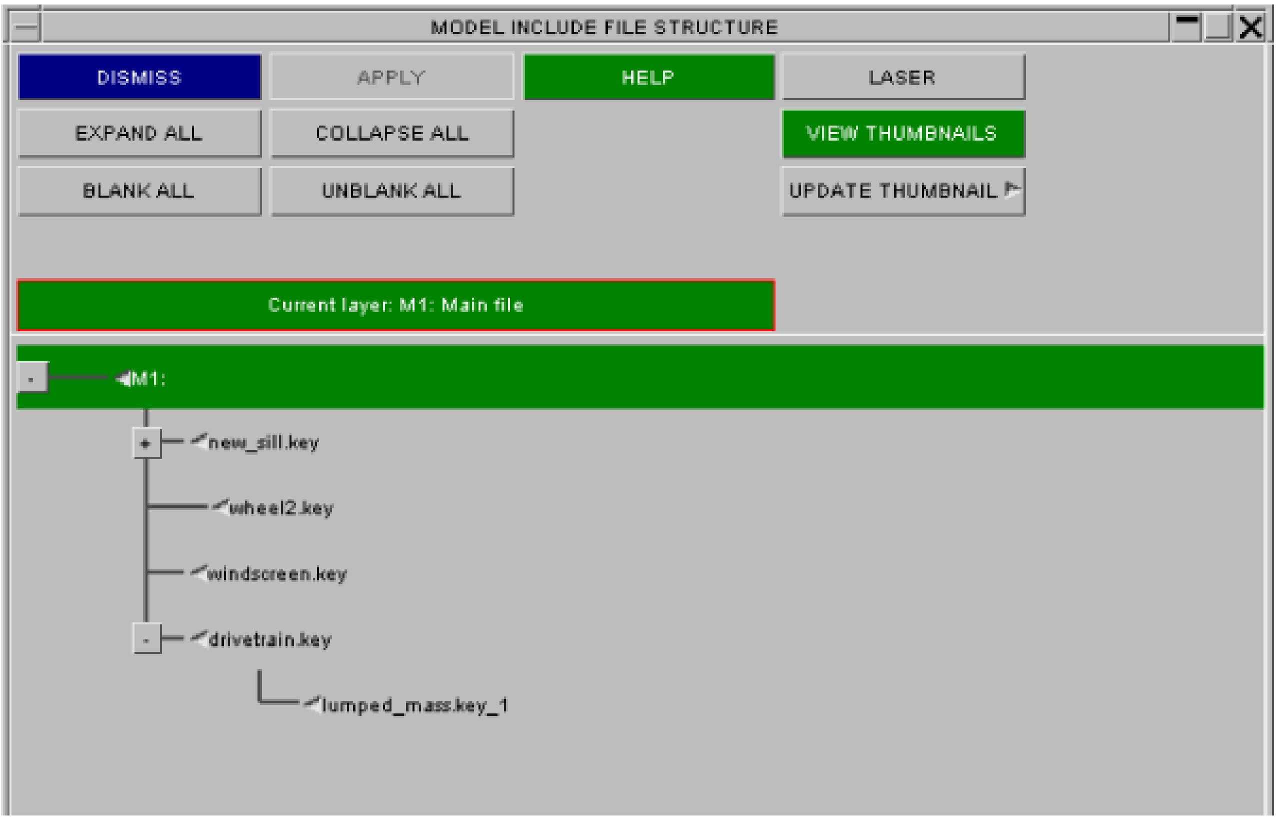
3.13.3 Viewing and managing the Include file structure

From the Tools Menu, select **INCLUDE** to access a diagram illustrating the tree structure of the current model.

- Viewing the contents of the Master Model
- Thumbnails
- Adding Include files to a model
- Controlling Include file location for newly created entities



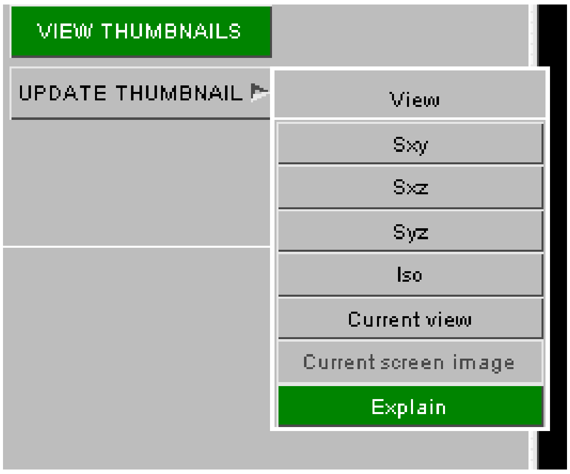
Viewing the contents of the Master Model



In order to display the names of the Include files in all layers of the file, click on the Expand all tab. In order to condense the window in order to just show the Master model, click on the Collapse all tab. In order to expand an individual Include file to see the Include files contained within, click on the grey square to the left of the Include file you wish to investigate

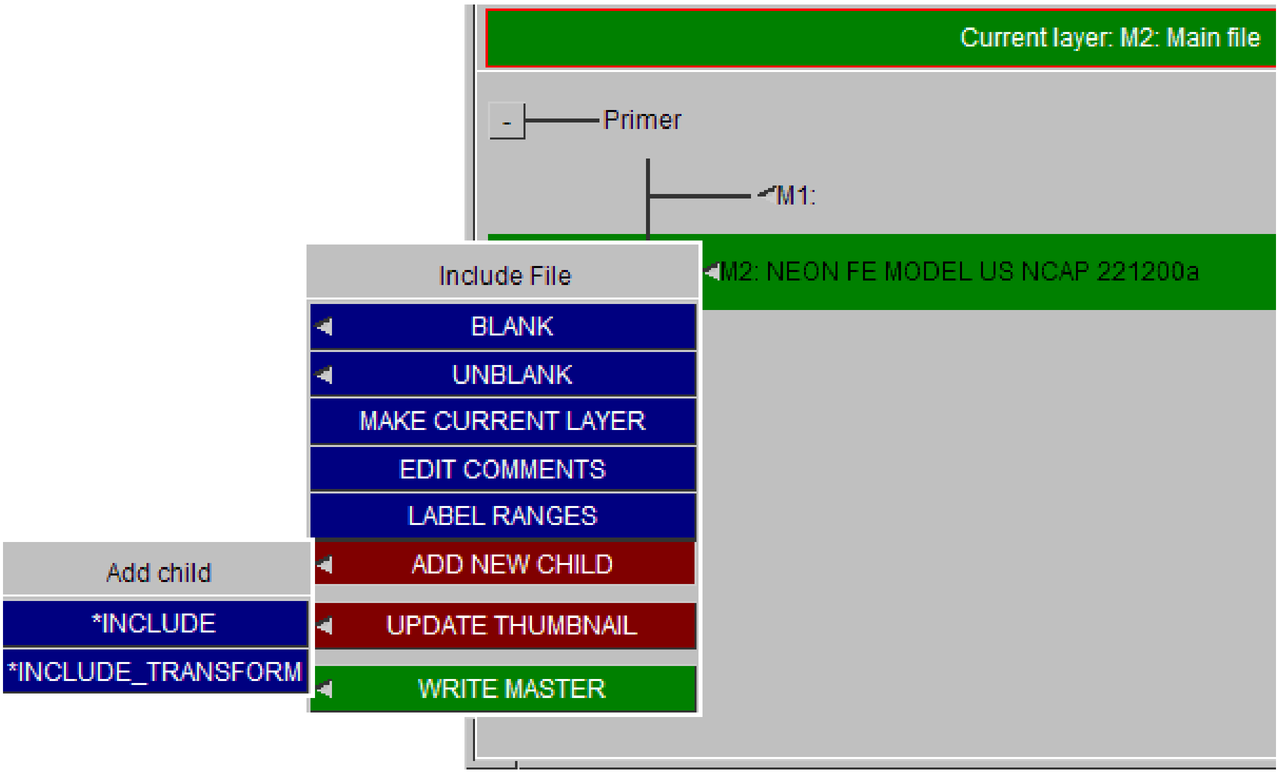
Thumbnails

Thumbnails can be inserted next to the include file names to provide a graphical illustration of the contents of individual include files. To access this capability, right click on the Update Thumbnails tab at the top of the window and a pop-up will come up asking you to select a view. The current option will display the contents of the include file as they currently appear on the screen. Individual thumbnails can be updated by right clicking on the name of the keyword file and selecting the Update Thumbnails tab in the pop-up window. If generated, thumbnails can be turned on and off by toggling the **VIEW THUMBNAILS** button



On keyout, unless **WRITE ANY THUMBNAILS** is clicked off, any thumbnails generated will be written below *END of their keyword file.

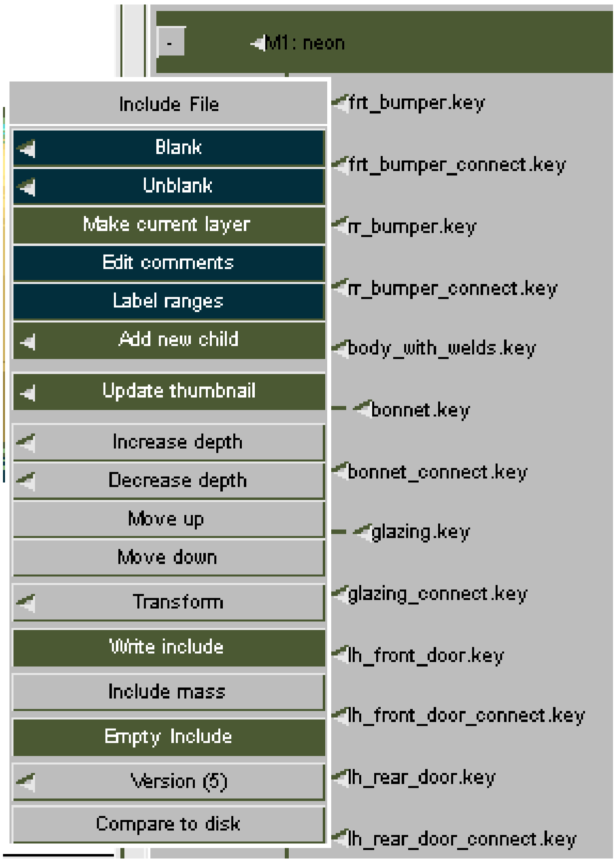
Adding Include files to a model



Include files can be added to a model by right clicking on the keyword file you wish to reference the new Include file in and selecting the **ADD NEW CHILD** option in the pop-up. Select the file you wish to add by either inputting the name and path into the box, using the search option by clicking on the question mark, or by accessing the model DATABASE by clicking on the Database tab. If you want to create a new Include file within the model, then input the name and path of the new file you wish to create.

Removing Include file

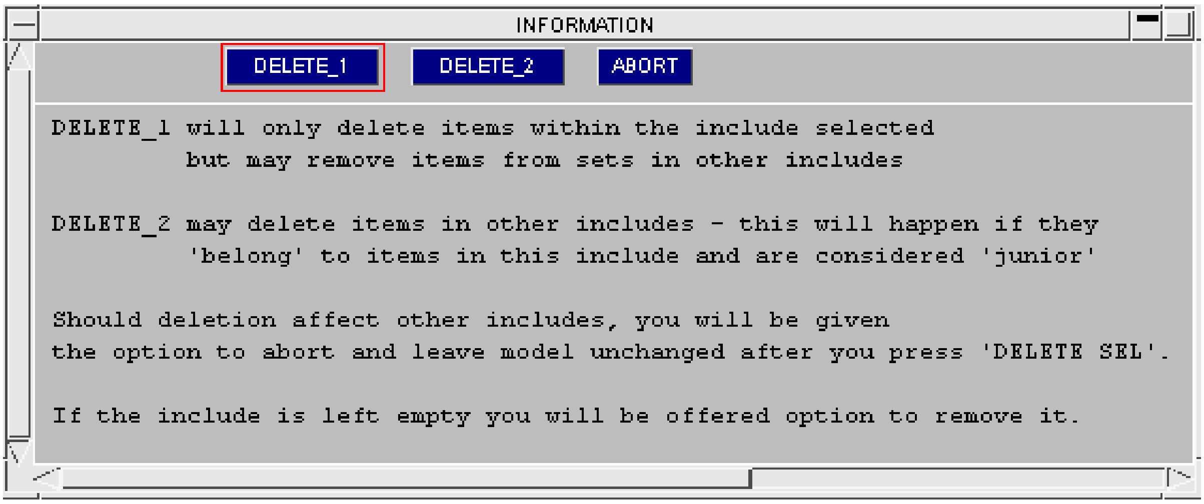
Off **TOOLS > INCLUDE** tree there is an option on the include file dropdown to **Empty Include**.



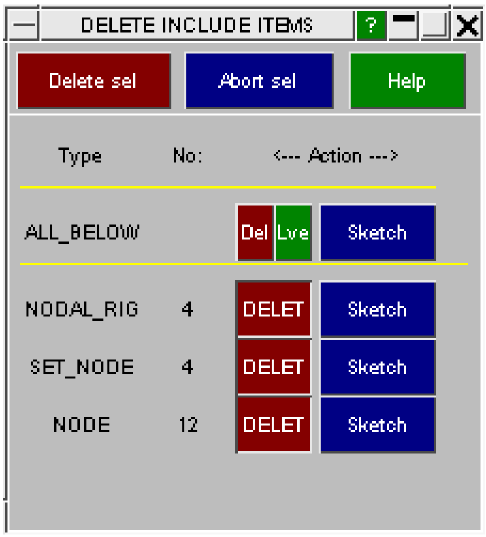
Two deletion methods are available.

DELETE_1 is strict deletion which will only try to delete items in the include file. However, if the **Remove from sets** option is active (by default this will be), the deletion may require the removal of items from sets in other include files.

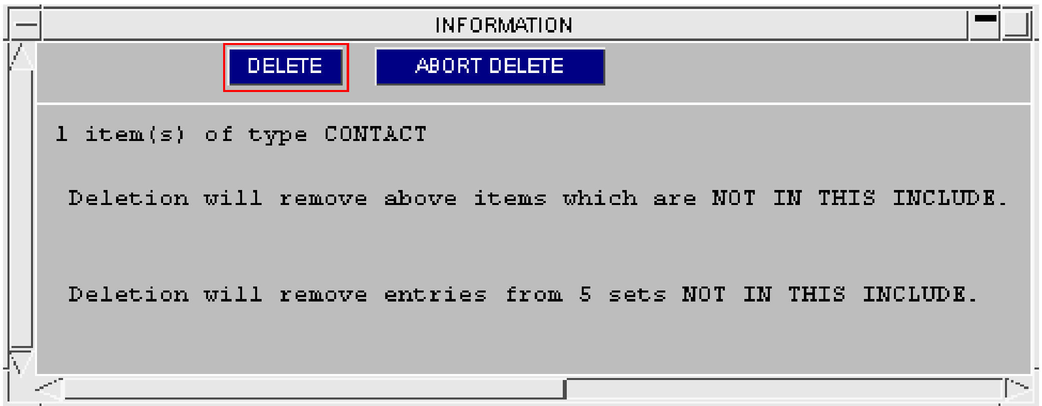
DELETE_2 is propagate deletion. This is analagous to selecting the contents of the include, putting them on the clipboard and applying delete. Any items which are junior to those selected (e.g. the nodes of an element) will be flagged for deletion without regard to their include file. This mode is more likely to want to delete items outside the include file



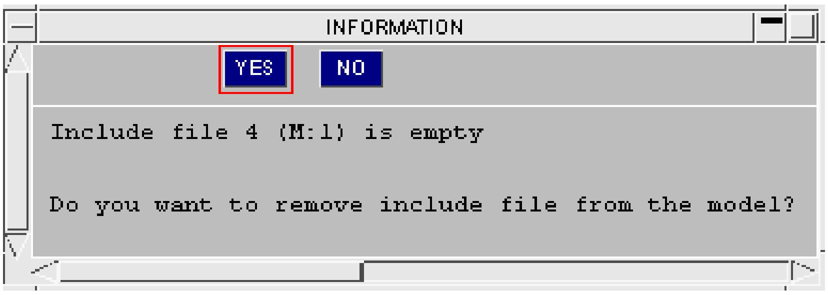
In both modes, the interactive deletion panel will then be activated. Items may be de-selected by using **Leave**. Pressing **Delete Sel** will initiate the deletion.



Before deletion is actually carried out, Primer will detect if any items to go are outside the include file. These will be reported and user given the option to abort the deletion operation. The model will be unchanged.



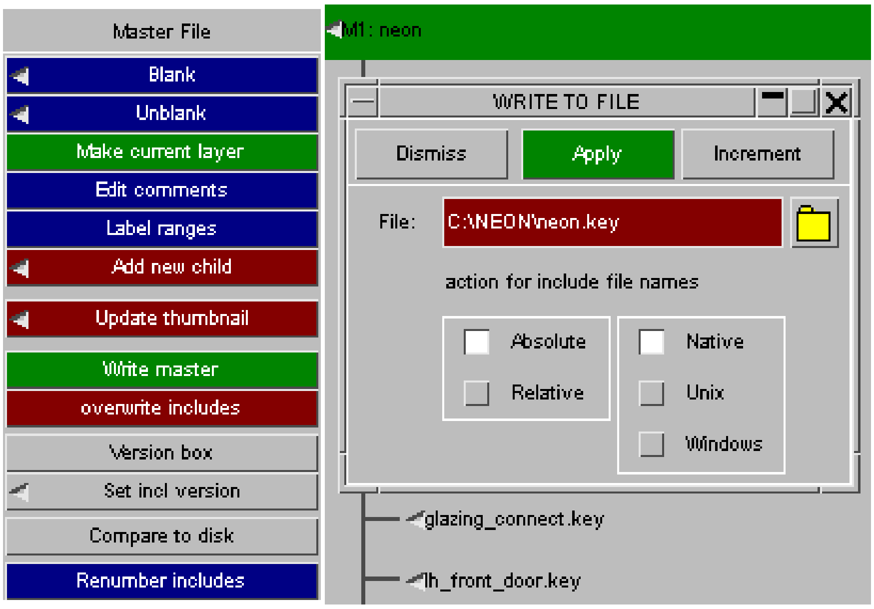
If deletion is applied, Primer will then check to see if the include file is empty. If so, you will be given the option to remove the include file from the model structure. Alternately you can leave it as an empty include, presumably for later population.



Writing Master file

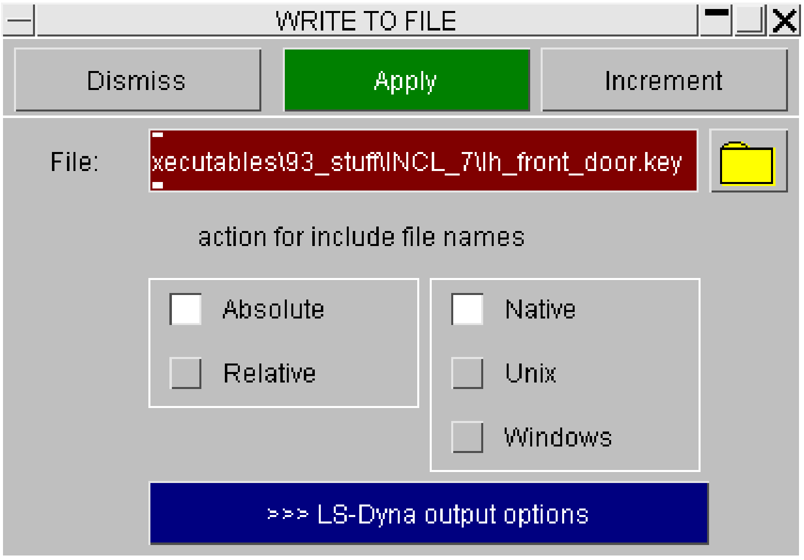
The master file can easily be written from the the include tree using **Write Master**. The panel allows you to set the options for include files.

Increment changes the file name from fred.key -> fred_001.key -> fred_002.key or from fred_1.key -> fred_2.key, etc.

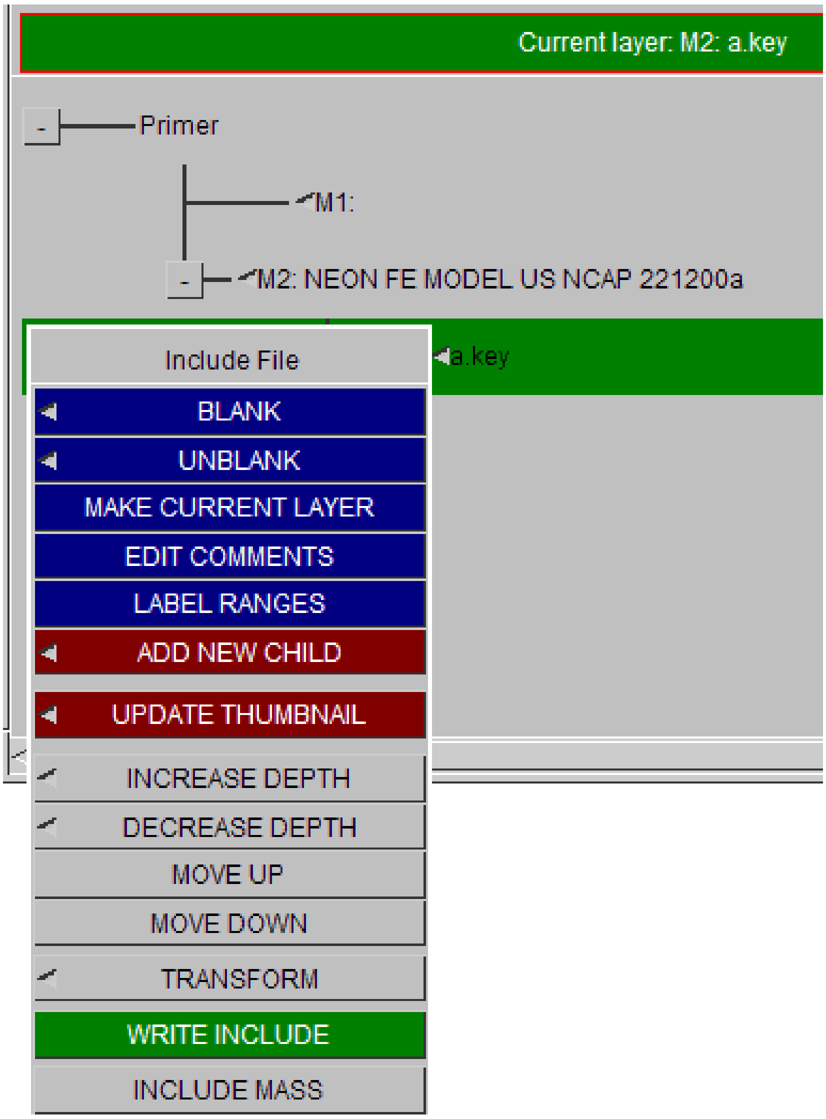


Writing Include file

An easy way to write out a single include file is provided by the **WRITE INCLUDE** option accessed from the include file tree. This initiates a WRITE TO FILE box ready to overwrite the existing file, shown by the red background. No further warning will be given if the user presses **APPLY**. Modifying the file name will set the background to green. Some output options are given on the WRITE TO FILE box. For all the output options, click on **>>> LS-Dyna output options**.



Controlling Include file location for newly created entities



A Green band highlights the **Current working layer** or **current include**.

Any new items made in the model will by default belong to this layer and will be written with this INCLUDE file. Items which are merely modified should always stay in their original layer, unless the user specifies a change of include layer through an edit panel. In order to change the current working layer, right click on the layer you wish to make current and select the **MAKE CURRENT LAYER** option in the pop-up.

BLANK/UNBLANK by include file (or use of the ONLY function in the Part tree) is a useful check for which items are in which layer.

Renaming an include file

To rename an existing include file right click on an include file and select **Rename Include** from the popup. This maps a window allowing you to choose a new name for the include file.

3.13.4 Editing comments in include files

The **EDIT COMMENTS** option allows you to change comments that are stored at the top of include files (or the main model). Using this option will start an external editor that allows you to change the comments.

On windows the editor defaults to notepad. On unix the default editor is vi (opened in a new xterm window). The editor that is used can be changed by either:

1. Setting an oa_pref option **`primer*text_editor`** to the editor you want to use.
2. Setting an environment variable **`EDITOR`** to the editor you want to use.

The oa_pref option is checked before the environment variable. Note that the oa_pref option/EDITOR variable **should be set to a filename** and the editor should always start another window.On windows this will always be the case. However, on unix if you just set the editor to **`/usr/bin/vi`** then this would try to open vi in the command/xterm window you started Primer (or the shell) from. This can cause problems (and it would definitely cause a problem if you had started primer by clicking on an icon as there is no xterm window!).

To get round this you can start vi in an xterm window by using a wrapper c-shell script and setting the oa_pref option/environment variable to this. e.g. the following c-shell script will start vi in a new sterm window.

```
#!/bin/csh -f
xterm -title "Edit Primer comments" -e vi $1
```

This technique can also be used if you need to pass other command line options to your editor.

Editors such as dtpad (/usr/dt/bin/dtpad) open a new window so that is OK.

Previous versions of PRIMER would not allow you to do other operations while editing a file. In version 10.0 this restriction has been removed.

3.14 INCLUDE transform

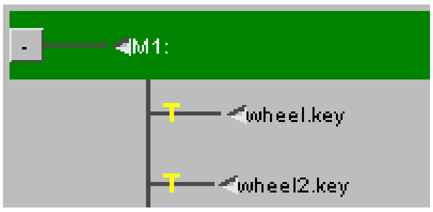
- Display of Include Transform
- Edit of include transform
- Edit of define transform
- converting include files
- keying out include transform

*INCLUDE_TRANSFORM in LS-Dyna provides a means of setting label offsets and applying units conversion to items of an include file.

Additionally the transform *may* reference a Define Transformation statement which will apply a geometric transformation to the include file.

3.14.1 Display of transform

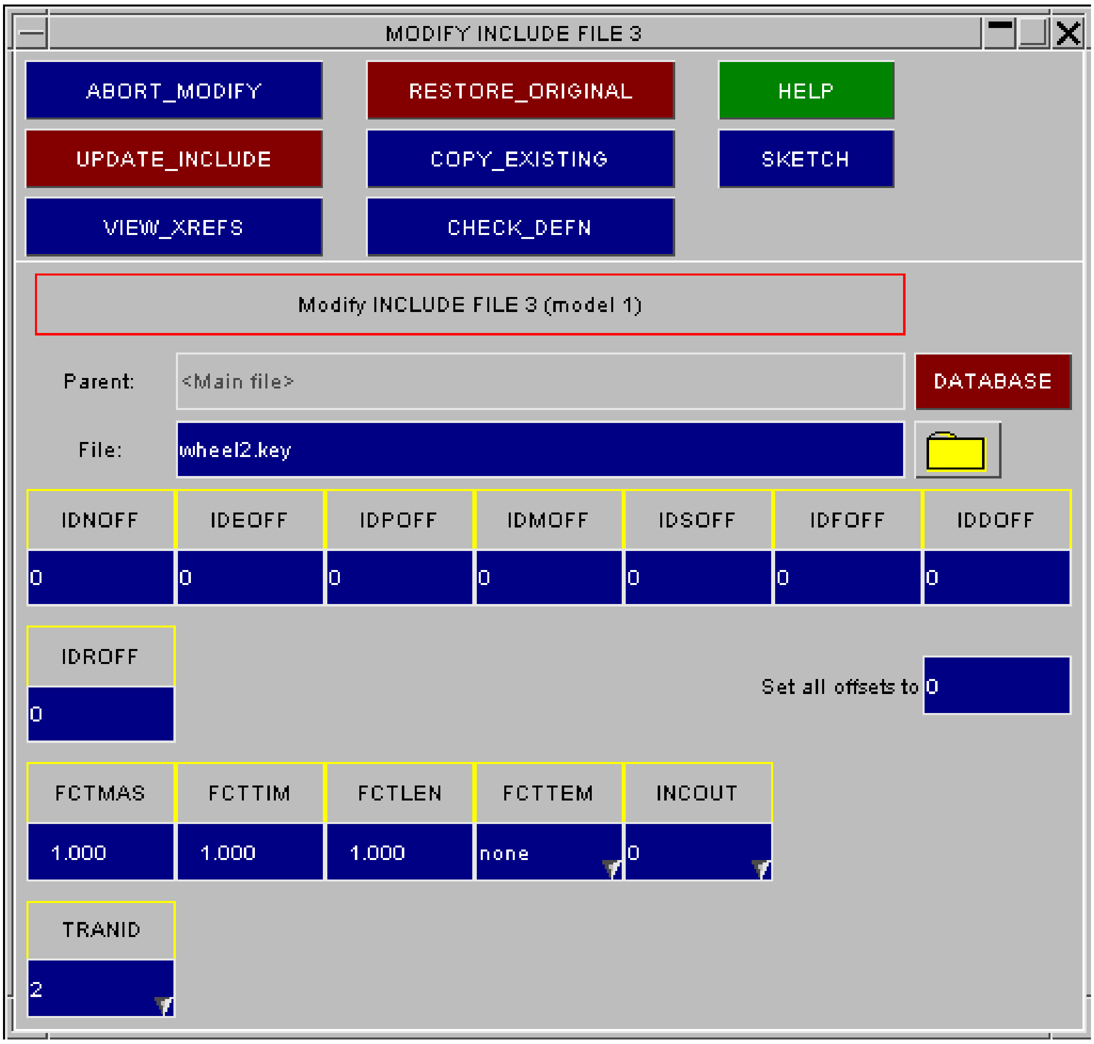
Include files which belong to include Transform statements will be displayed in the include file tree with a prominent "**T**" on their branch.



PRIMER applies the transformations to the model on read-in. Therefore, on write-out the transformations are usually reversed and the transformation definition data restored (but the user may prevent this).In the graphics window, the items of the include file will be displayed *in their transformed state.* That is with their labels offset and their geometry changed as necessary.

3.14.2. Edit of include transform

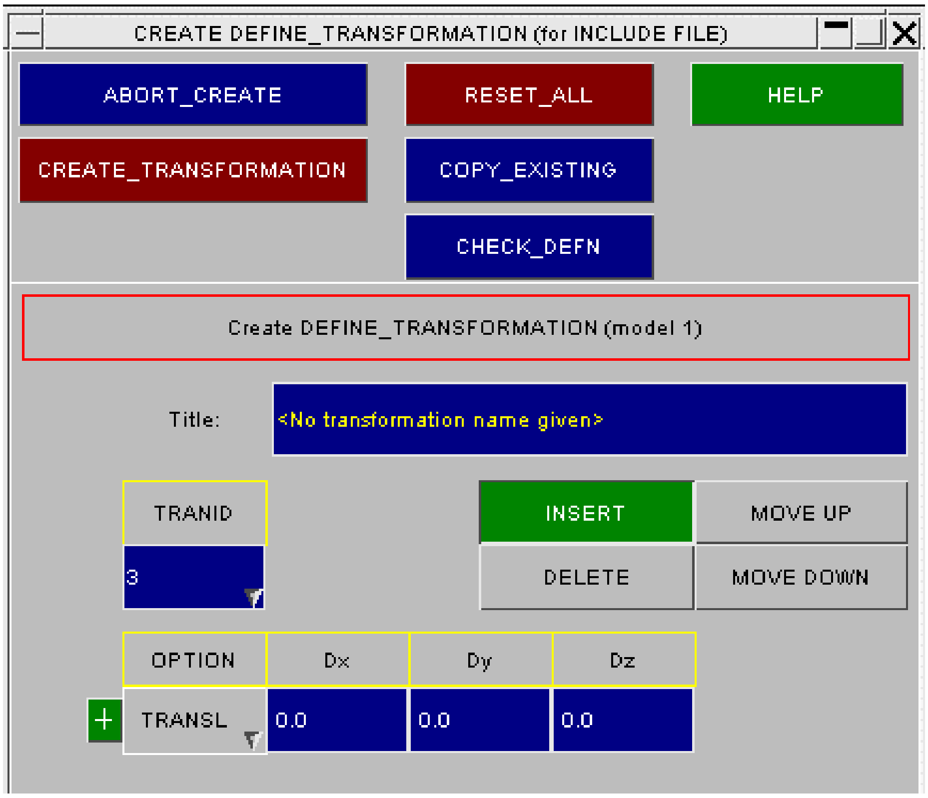
Using the popup from the tree diagram, select **TRANSFORM**->**EDIT** to invoke the edit panel.



This menu allows definition of label offsets and unit conversion to be applied on reading the model, and refers also to a geometric transformation (TRANID)

3.14.3. Edit of define transform

Using the **TRANID** popup, you may create or edit the Define Transformation statement. Any newly created will be be placed at the top of the include file.



Each definition can contain multiple transformations (the options available for which are **TRANSLATE, ROTATE** and **SCALE**. Whilst **INSERT** mode is selected clicking the green [+] button will add another in the row below. **MOVE UP** and **MOVE DOWN** allow the altering of the order in which the transformations are applied. A label in the TRANID filed must be defined before the **CREATE_TRANSFORMATION** button will become active.

3.14.4. Converting include files

The **ADD NEW CHILD** function on the include file tree may be used to add an Include Transform as well as an ordinary include file.

An ordinary include may be converted to transform type (**TRANSFORM**->**ADD**) and a transform type to an ordinary (**TRANSFORM**->**DELETE**).

In the latter case the option is offered of leaving the data in its transformed or untransformed condition .

3.14.5. Keying out include transform

The method of keyout will affect how the data transformation is handled.

- merge->master - the data is transformed
- in sub-directory - as all the transformation calls will be present the data is written in its native state
- select files - the user must select the mode as "**NO-CHANGE**" (leave the data in its native state) or "**MOVE**"(change data to its transformed state, i.e. as if for use with an ordinary include file).

When using the select file mode, it is unclear whether the applicable DEFINE_TRANSFORMATIONS are included or not, as they exist entirely separately from the INCLUDE_TRANSFORM statements. The user must decide.

3.15 MODEL > BUILD

- Building a model using Model Database
- Creating and managing a Model Database
- Reading files using a Model Database
- Managing Templates
- Build of multiple models
- Build from csv targeting files

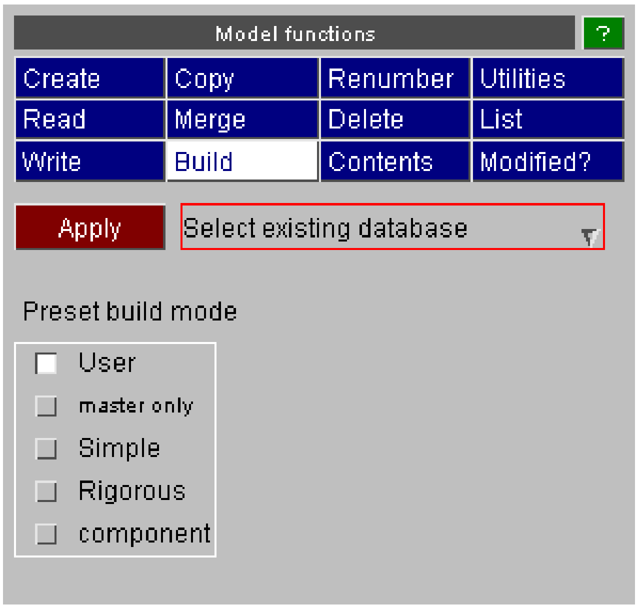
A Model Database is an xml listing of include files and other information that enables you to easily assemble a model from smaller files, typically component and connection files.

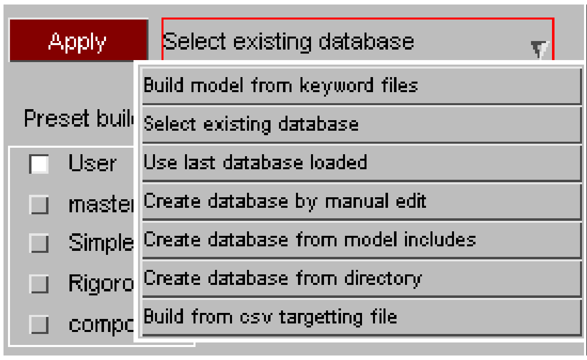
3.15.1 Building a model using Model Database

- Selecting/Creating a Database for model build
- Viewing the Model Database
- Selecting Include files to read
- Building the Model
- Orienting the include files
- Post model build panels
- Writing the Model

3.15.1.1 Selecting/Creating a Database for model build

In the menu, select **MODEL**->**BUILD.** The default is **use last database loaded** which will on its first call revert to **select existing database**.

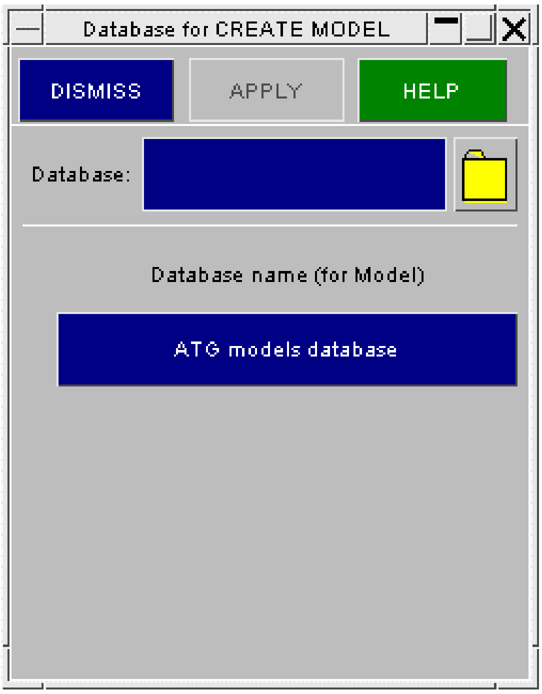




NOTE: this procedure is to be distinguished from **MODEL**>**READ**>**DATABASE** (see section **MODEL>READ**) in which each selected model is read into a separate file.

In order to select an existing database, select the **SELECT EXISTING DATABASE** (the default) option and press **APPLY**.

To create a new Database select the appropriate option and press **APPLY**. See **CREATE NEW DATABASE.**

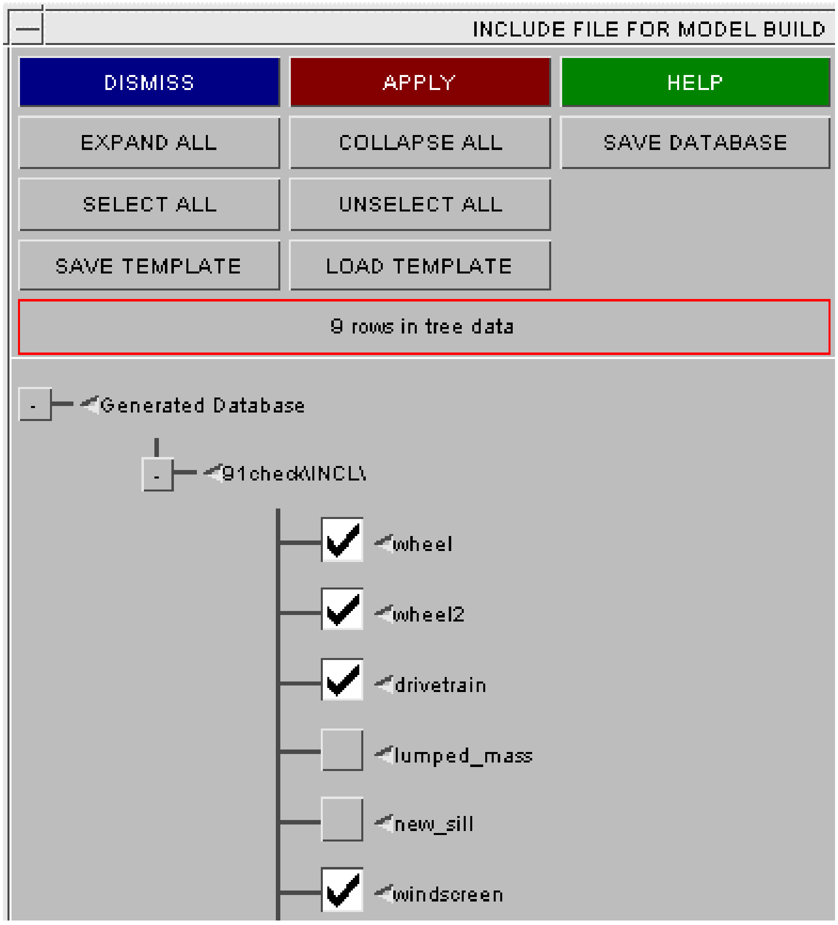


Select an existing database by inputting the name and path in the input box or using the search facility or by selecting one of the databases listed in the Database Name list.

Press **APPLY** to load the database. Once this is read in the following section applies.

3.15.1.2 Viewing the Model Database

In order to see descriptions of all the files available to read, click on the **EXPAND ALL** tab. In order to display just the top layer of the database select the **COLLAPSE ALL** tab. To see the thumbnails linked to each file, to give an idea of its contents, select the **VIEW THUMBNAILS** tab. To hide the thumbnails click on the **HIDE THUMBNAILS** tab.



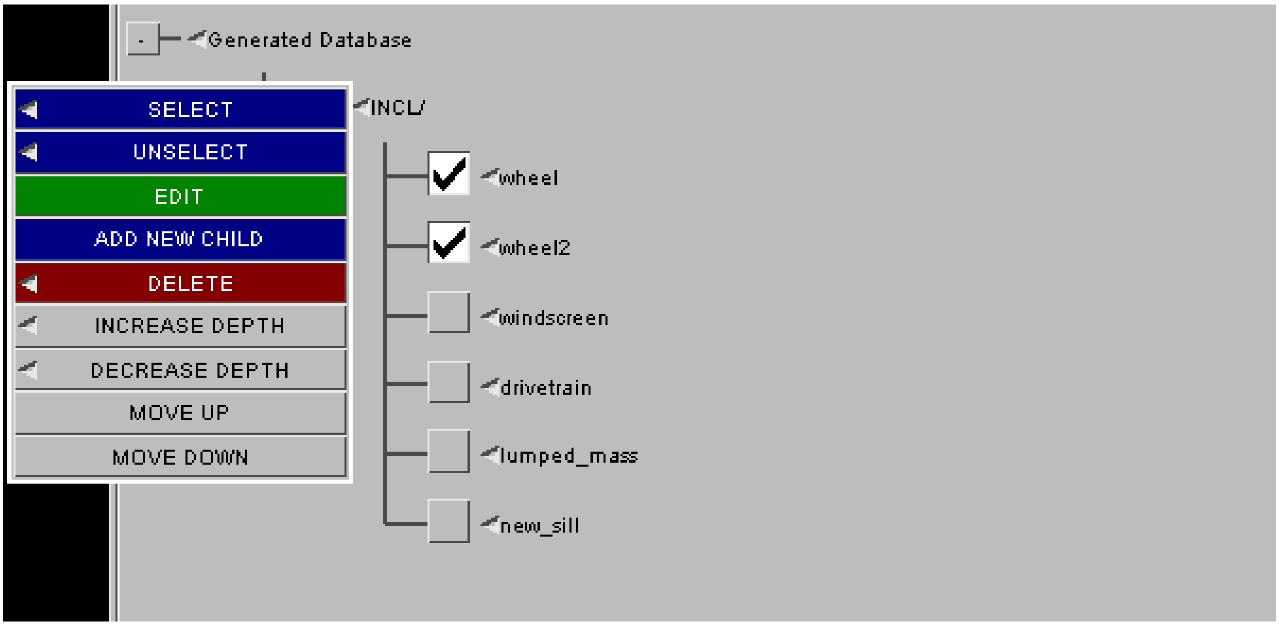
3.15.1.3 Selecting include files to read

In order to select a standard file from the database to read into PRIMER, select the file by clicking on the grey box to its left. The box will display a tick when selected to show which files are to be read in.

Changing the keyword file associated with an entry: To select a file not currently in the database you can select any row, right click on the small arrow to the left of the option and select **EDIT** on the pop-up window that appears. The file which is read in when this option is selected can now be changed by changing the name of the keyword file in the Keyword file input box.

Adding a new entry: Alternatively you can add a new entry into the database by right clicking any option and selecting the **ADD NEW CHILD** tab. You can then enter the file for your new entry.

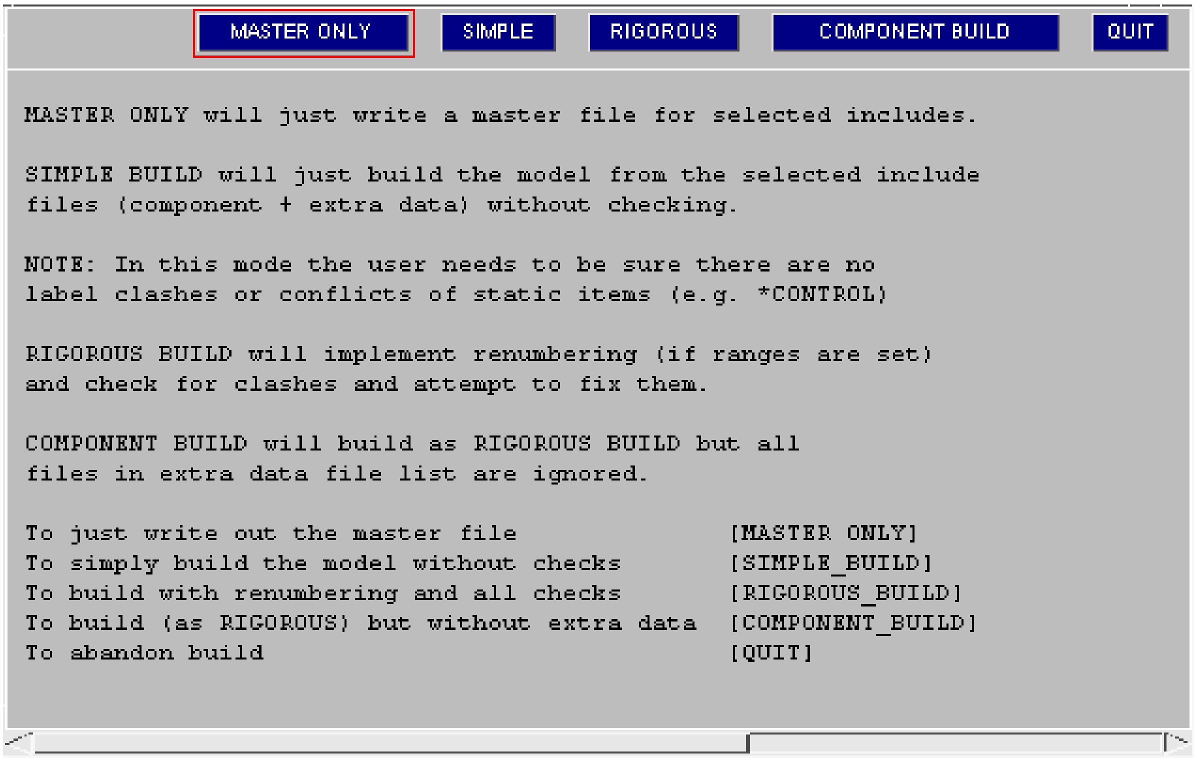
Using a template: If there is a particular combination of Include files you frequently wish to read in together, you can save a template listing these include files (described later in this section). Click on **LOAD TEMPLATE** in order to open an existing template and the include files listed in the template will be automatically selected.



3.15.1.4 Applying the Build

Once you have selected all the files you wish to read in, press the **APPLY** button. You will now select one of 4 build modes. **MASTER ONLY** will create a master file for the selected include files. **SIMPLE BUILD** is suitable when there is no renumbering scheme and no label clashes amongst the include files. If either of these two conditions is not applicable a **RIGOROUS BUILD** is required. This is considerably slower as each include file has to be read into a scratch model, checked against the existing model, perhaps re-labelled and then read into the existing model. **COMPONENT BUILD** will ignore all connection files during the build, which allows user to check that all component files are self contained (no references to items which don't exist) and numbered correctly within their ranges.

You can skip this panel by pre-setting the build mode to **SIMPLE BUILD** or **RIGOROUS BUILD** by setting the radio button option.



PRIMER will now take you through a number of stages to help ensure the model is built correctly.

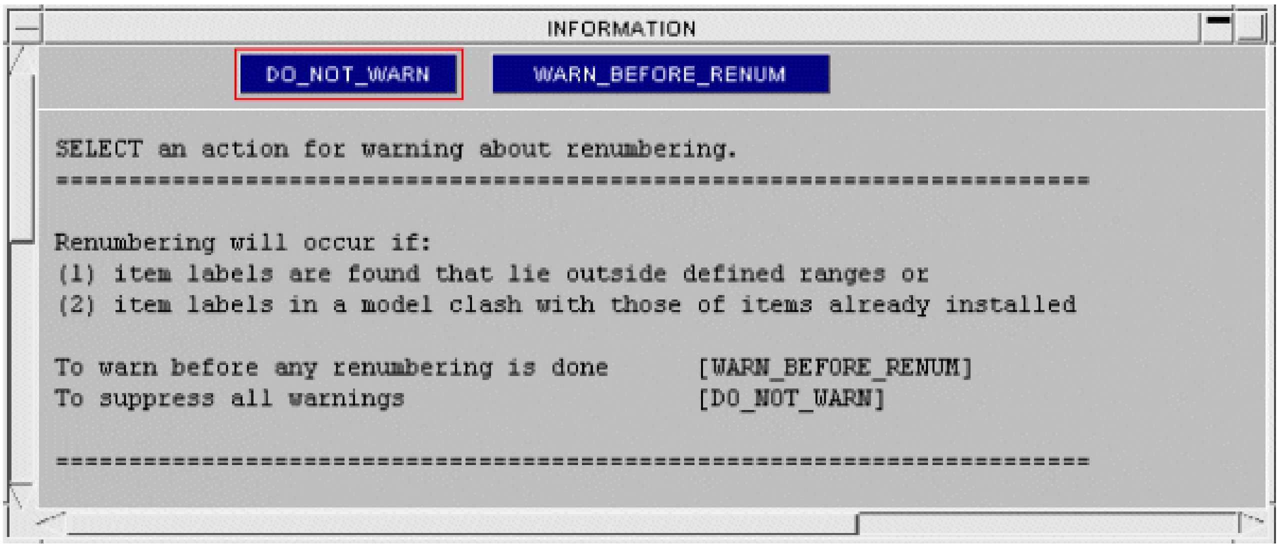
The next choice PRIMER will ask you to make concerns warning over model renumbering. Model renumbering may occur because either certain model labels lie outside the ranges specified in the database or because item labels in the include file are already taken up in the model.

The first pass involves setting item labels to meet the renumbering ranges. If the "**general type id**" range is set, part/section/material (see note below) labels must already be within this range. Other items will get renumbered into it. An **optional second range** may also be set for the more populous items (nodes, elements, nodal rigid bodies and node sets). Thus a node in the first or the second range will not have its label changed, but a node lying outside both will be renumbered into the second range. This node may subsequently be re-labelled for clash fixing. There is also a **FROZEN** range, intended for DATABASE_HISTORY items, the labels of which will never be changed.

Note: renumbering is NOT applied to latent items, so Materials that are referenced by a part but exist elsewhere (typically in a material database file) will not be affected.

The second pass involves checking the (perhaps re-labelled) items of the include file against those of the model and, if possible, re-labelling to avoid clashes. If this proves impossible, e.g. because a label range is exhausted or frozen items lie out of range, the read operation will trip an error.

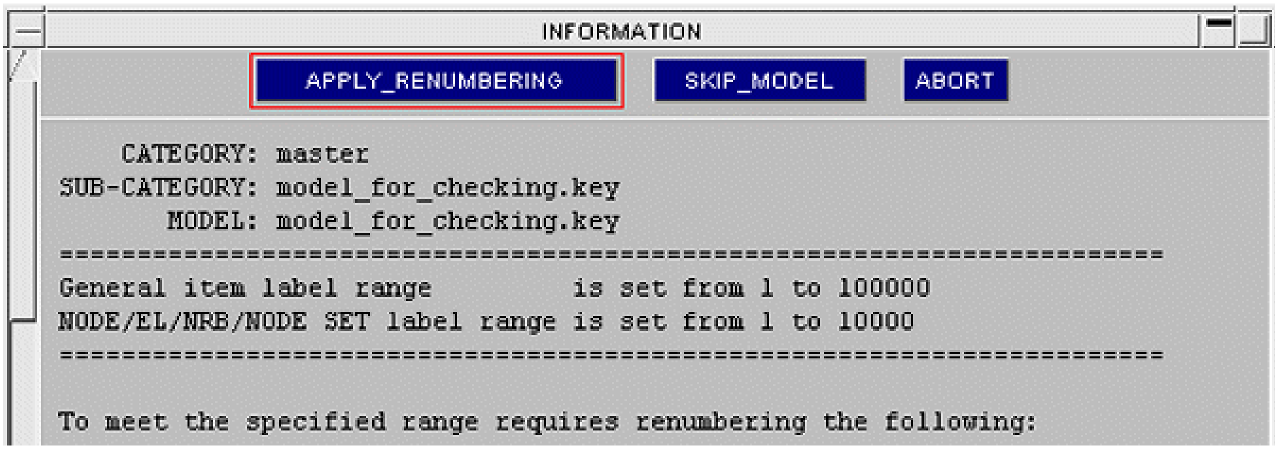
If you wish to be warned of any renumbering before it occurs, press the **WARN BEFORE RENUMBERING** button. If you wish PRIMER to renumber without notifying, press the **DO NOT WARN** tab. Then dialogue box will only be invoked in the event of a failure to renumber.



Renumbering into Range

If you select the option **WARN BEFORE RENUMBERING**, a window will pop up in PRIMER before any renumbering takes place providing a description of the renumbering necessary and a list of options.

If renumbering is achievable the options are **APPLY RENUMBERING**, **SKIP_MODEL** or **ABORT**. Otherwise, they are: **CONTINUE**, **SKIP_MODEL** or **ABORT**.

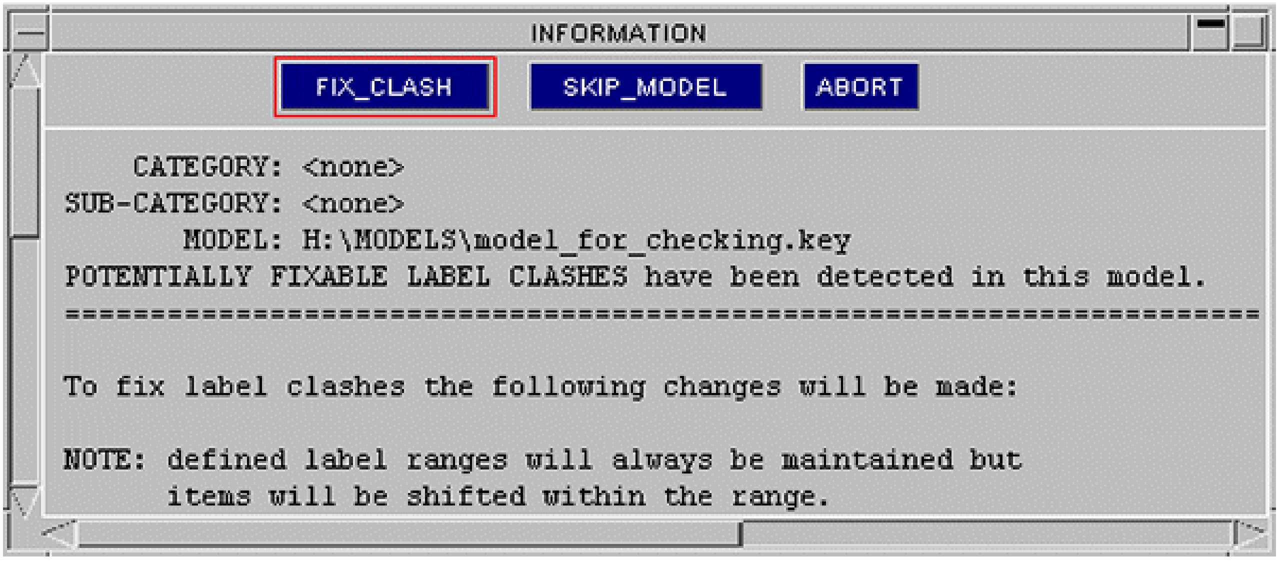


If you select **CONTINUE** the operation will proceed as usual. If you select **SKIP_MODEL**, the file that required renumbering will be skipped and PRIMER will continue to read all further files. If you select **ABORT**, PRIMER will stop the operation entirely.

Renumbering to fix clashes

Similarly, If renumbering is required because of a clash of labels, PRIMER will detail the fixing procedure automatically. Three options will be available: **FIX_CLASH**, **SKIP_MODEL** or **ABORT**.

If you select **FIX_CLASH** PRIMER will fix the numbering problem as detailed in the pop-up window. If you select **SKIP_MODEL**, the file that required renumbering will be skipped and PRIMER will continue to read all further files. If you select **ABORT**, PRIMER will stop the operation entirely.

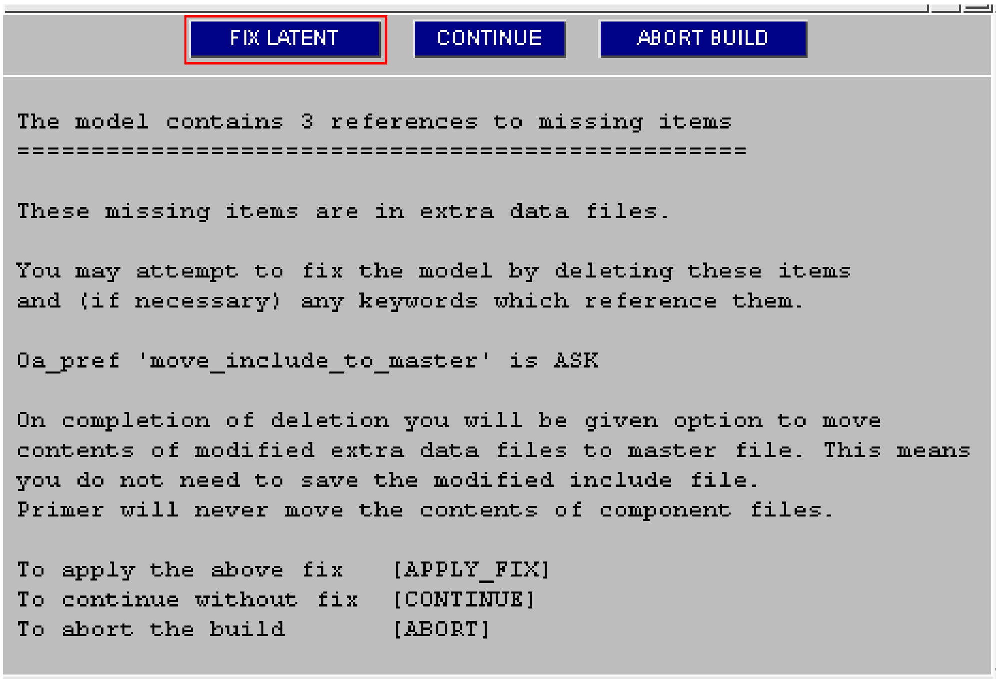


Once the model build has been completed, PRIMER will search the model for references to missing items. These will arise typically, where a connection file spans several components, but a choice was made to build the model with a subset of those components. For example, a connection file containing a contact which includes a vehicle dummy, an airbag and a steering wheel, but with only the dummy and the steering wheel read in as components. The airbag parts are now latent items in the connection file and must be removed before the job can run in LS-Dyna.

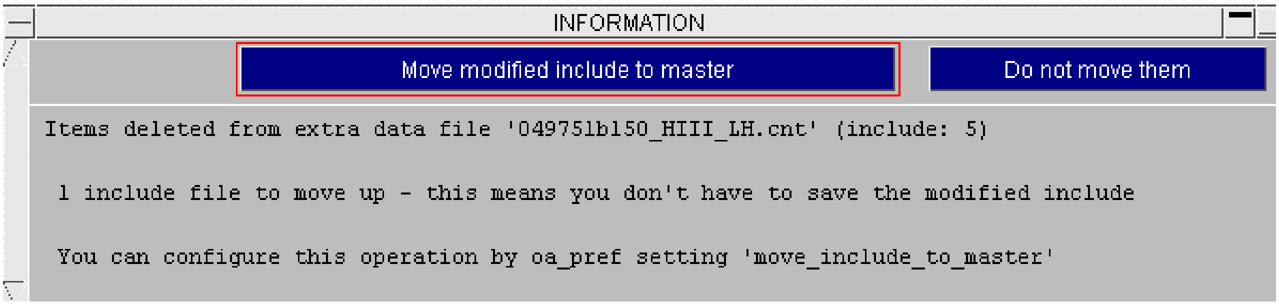
A window will pop up mentioning the number of missing items and offering option to **FIX LATENT.**

If you select **CONTINUE**, the missing items will remain in the model and it will require fixing before it can be run in LS-Dyna

If you select **FIX LATENT** Primer will run an auto-fix procedure to delete the offending items.



As the option ’move_include_to_master’ is set to ASK, you will get a 2nd information panel.



The deletion of latent items has modified the extra data file. If the option **move modified include to master** is taken, all the items of the file are moved up into the master file, and in the master file the keyout of the include file is itself suppressed. Thus the keyout of nthe master file alone is sufficient to represent the load case.

Fix for Missing Items

For more information on model cleanup look at section 6.4.2 Cleanup unused.

If you apply the fix Primer will also remove any unused items in the model (with the exception of parts and part sets which we deliberately keep) as this is implicit in the cleanup unused procedure.

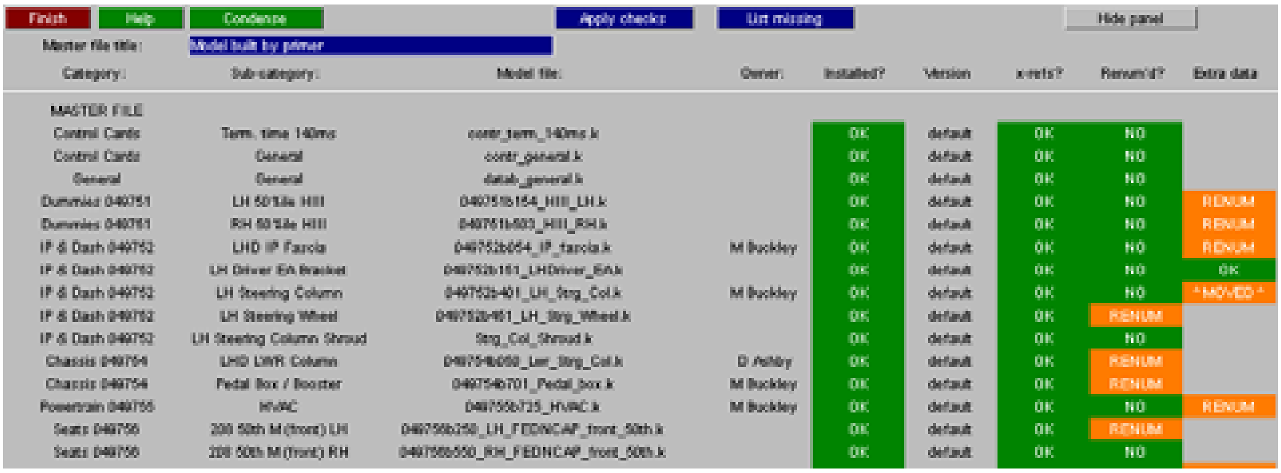
The removal of missing items may incur further deletion operations such as the removal of contacts with empty sets. For this reason the cleanup unused procedure is necessary to ensure that the model will run in LS-Dyna.

Primer will now summarize the build procedure in the Summary box and present the result of checks made on the model in the Check box. Once you have studied these boxes and wish to continue, click on the **FINISH** tab to complete the build procedure. The model Database will reappear. The information contained within the Summary and Check boxes will be written to a file called **build_status.txt.**

3.15.1.5 Post model build panels

The following panels appear if the **RIGOROUS** or **COMPONENT** build has been selected.

Summary Box



Listed down the left hand side of the box are the names of all the keyword files that you asked to be read in. In order to show only those keyword files that have been renumbered (orange) or that show an error (red), click on the **CONDENSE** tab.

Up to 7 pieces of information are available on each file under the headings:

- Owner?
- Installed?
- Standard?
- x-refs?
- Renum'd?
- Extra data?

A green box illustrates that no problems were encountered under the corresponding heading in the building procedure. An Orange box illustrates a Warning - items in the file were renumbered or a connection file was shifted into the master file (see below). A Red box illustrates a fatal problem indicating that the file was skipped or contains missing cross-references.

In order to list any missing parts that have been read into the model, press the **LIST_MISSING** button..

In order to display the model Database in its tree format, press the **SHOW_TREE** button.

To access the Check box if it isn't already present, press the **APPLY_CHECKS** button

If the check box is displayed, you may temporarily hide the summary box by pressing the **HIDE_PANEL** button. It will be restored by **RETURN_TO_SUMMARY**.

Owner

This category states who the owner of the corresponding keyword file is.

Installed

This category declares whether the keyword file was installed successfully (**OK**) or failed to install (**FAILED**).

Standard

This category reveals whether the installed keyword file was a standard file (**YES**) contained in the loaded database or a non-standard file (**N/S**), resulting from an edit to the the database.

X-refs

The **BAD XREF** error indicates that the include file contains references to items which do not exist in the model. It should only arise if the Deletion and Cleanup functions have been aborted during the model build.

Renum'd

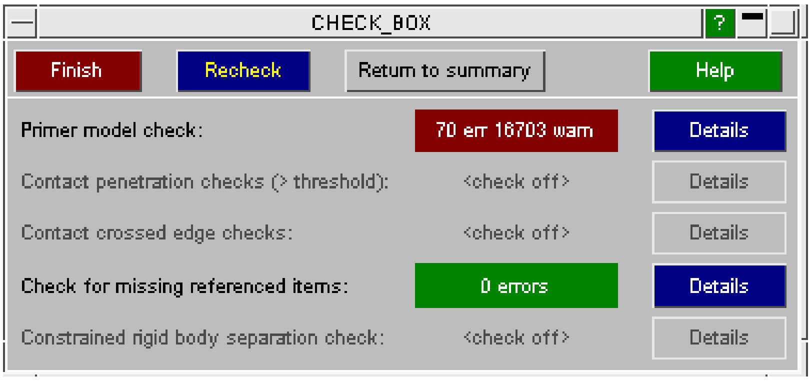
This category specifies whether or not any renumbering of the contents of the keyword file occurred in the building process.

Extra data

This category specifies whether the extra data files linked to the file were read in successfully (**OK**) or reports a warning, such as **FAILED** or **BAD XREF**.

If, during the build process, latent items of an extra data file have been deleted (see **APPLY FIX**), the file contents will have been shifted to the master file and the include file itself suppressed. Such files will bear the warning "^**MOVED**^".

Check Box



This window contains the result of a number of predefined checks specified as **CHECK>OPTIONS.**

In order to detail the results of any check that generated errors, press on the **DETAILS** button.

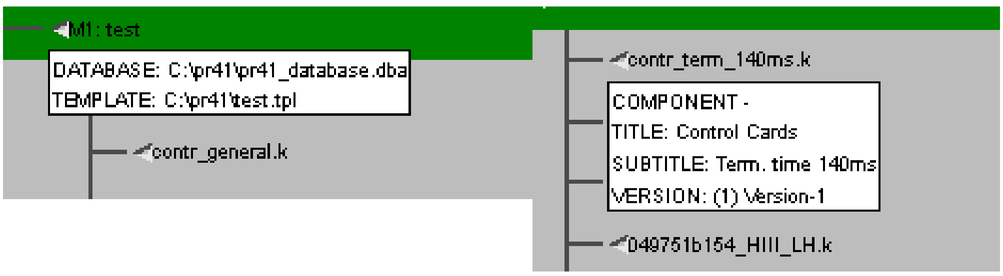
To close this window and return to the summary box, click on the **RETURN_TO_SUMMARY** tab. In order to rerun the checks, for example, after modifying the **CHECK**->**OPTIONS** settings, click on the **RECHECK** tab.

NOTE: The same check panel can be activated for any existing PRIMER model by the route **CHECK**->**APPLY**->**APPLY_RULES**.

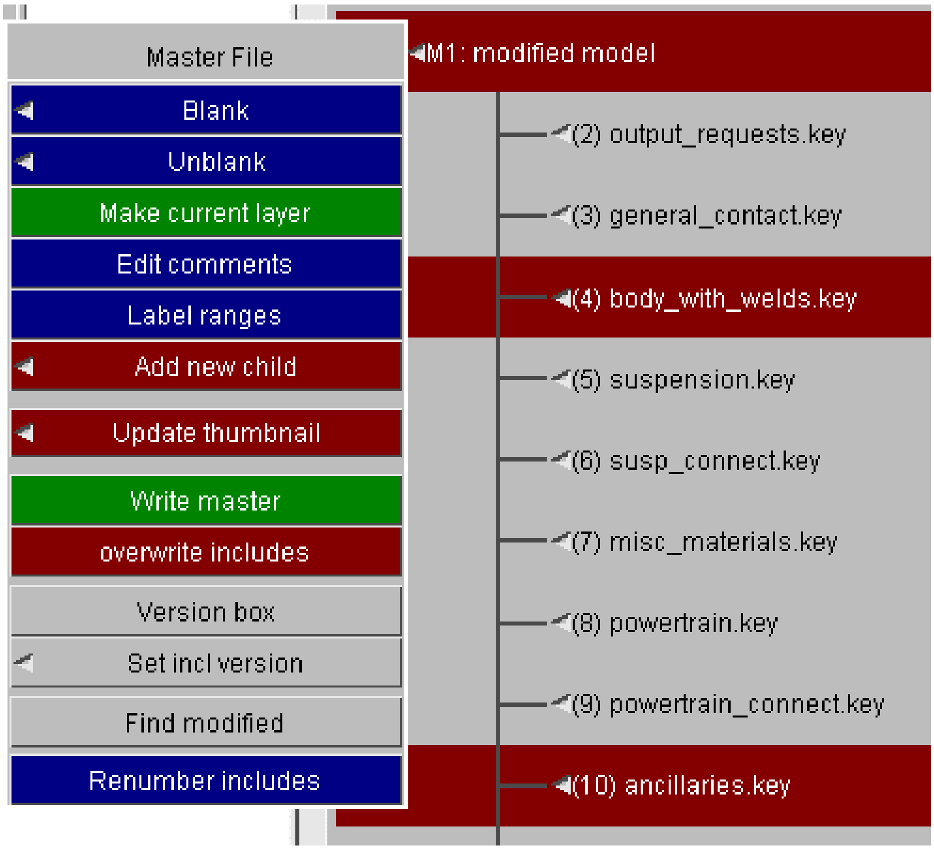
3.15.1.6 Modifying the built model

The built model now exists in Primer and may require modification, such as fix by auto-fix or otherwise of model errors, modification of properties, modification of boundary conditions, etc. On completion of the changes the updated include file must be saved. If it is overwritten the database requires no change, however, if it is written to a new filename the database entry requires update.

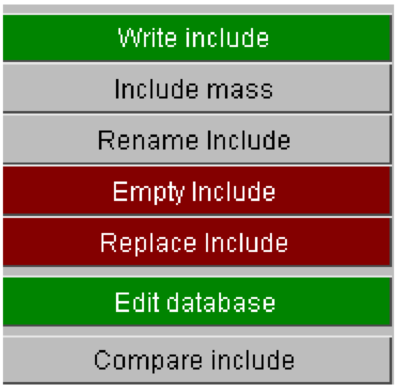
The include tree (TOOLS > INCLUDE) is now aware of the provenance of the model and its includes. This is displayed in hover text.



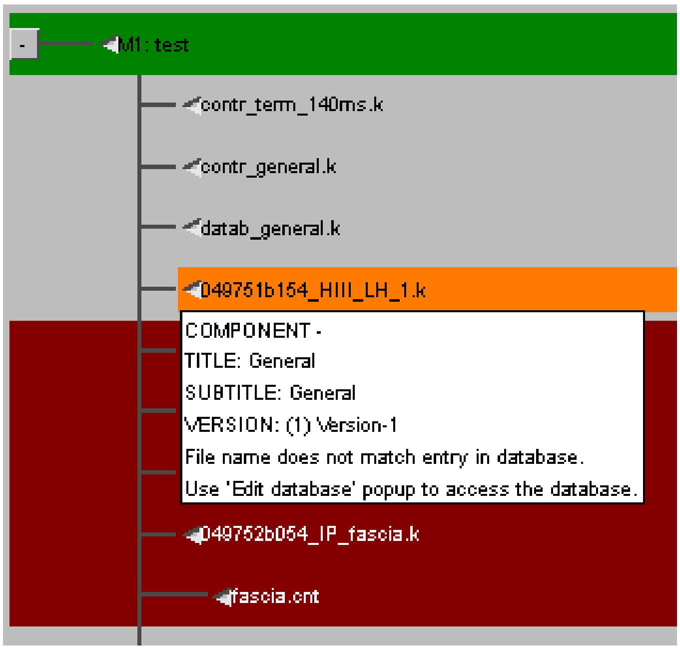
The Find modified function may be run on all the includes if accessed off the model popup. This will write the current include to a model and read the original include file into a model. These two models are compared and any differences reported.



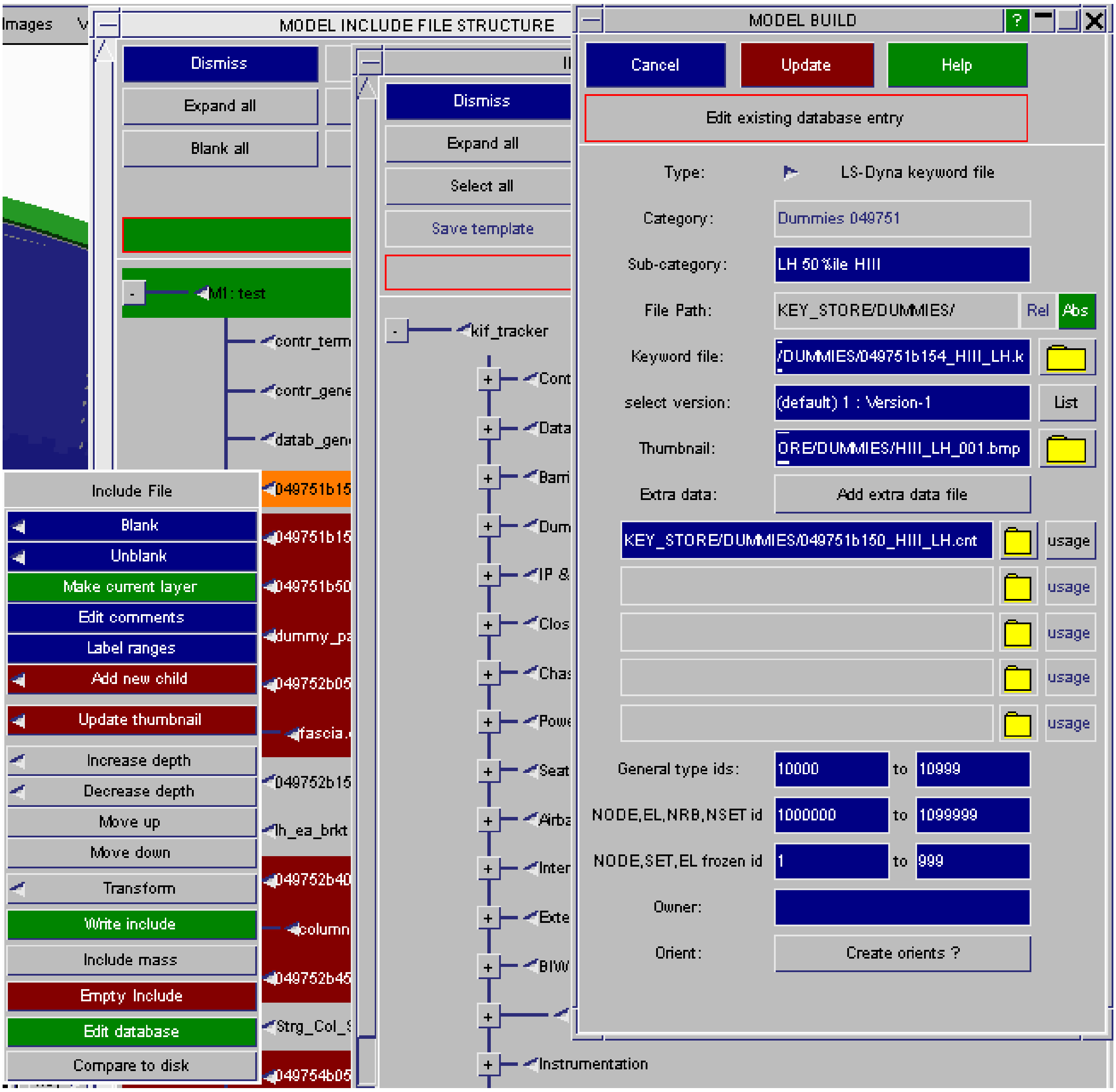
On completion of the function the modified includes will be marked by a red background as shown above. The details of the differences for an individual include may be reviewed by running compare include off the individual include popup.



Write include available off the include popup can then be used to save the modified include to a new name. The entry then becomes orange to warn the user that the database entry is out of date. The hover text is also modified.



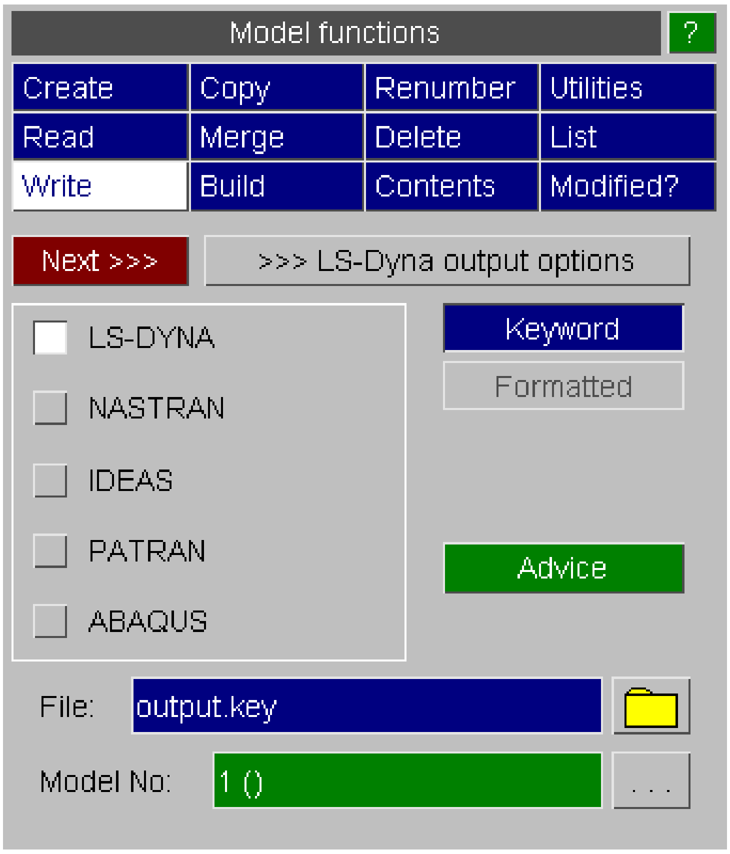
Edit database available off the include popup can be used to take you directly to the entry on the database. Here the keyword file (or extra data file) entry can be updated (at a newly created version number if you are using version tracking), the database entry updated and the database re-saved. The orange entry will then have its grey background restored, as the model and database are again consistent.



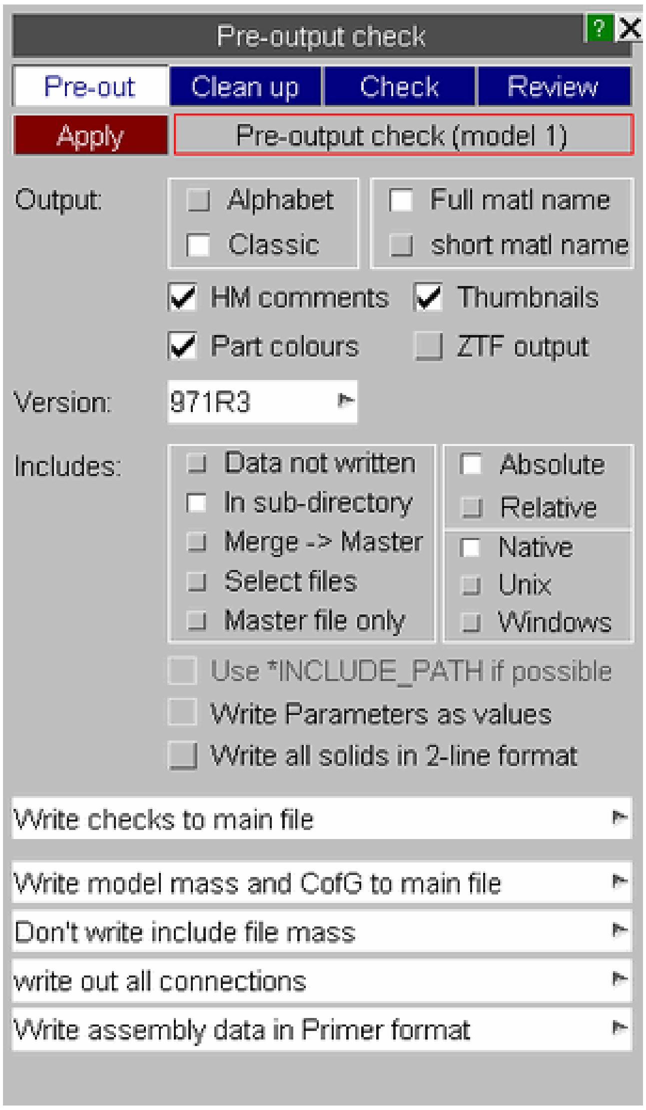
3.15.1.7 Writing the Model using select files

To save the model once it has been built, in the main menu right click on Model and select write in the pop-up.

In the Write to file window input the name you wish to save the master file under in the input box:



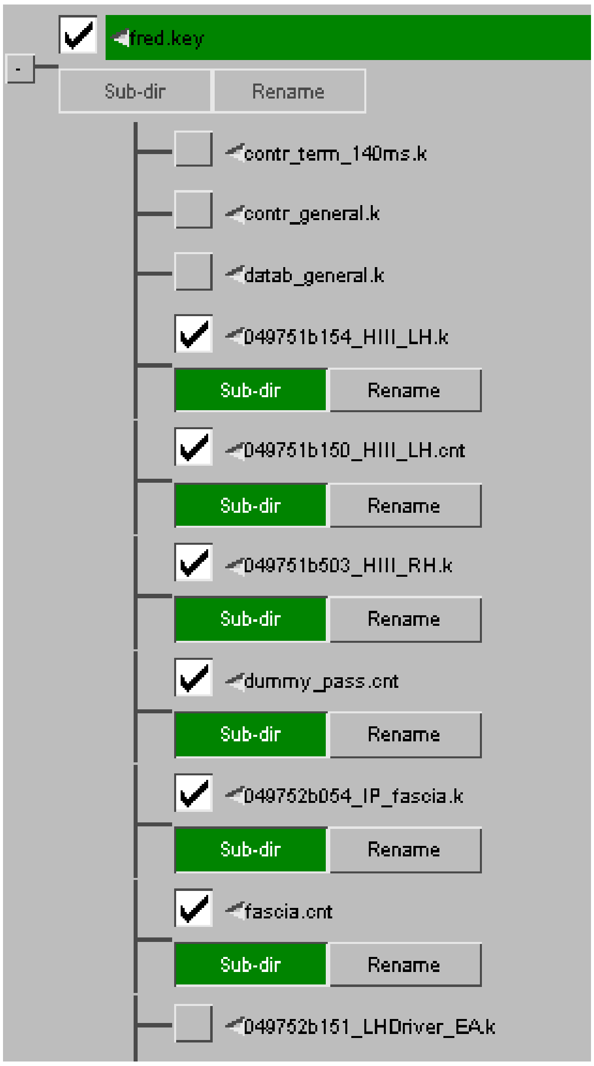
Select appropriate options in the Pre-Output check window:



If you select **Master File only** mode and any include files have been renumbered during the build process Primer will prompt you to change to **Select files** mode, where the renumbered includes will be preselected, putting up the following panel.


SELECT_FILES
CONTINUE
25 include file(s) were renumbered during model build
==
C:\pr41\KEY_STORE\DUMMIES\049751b154_HIII_LH.k
C:\pr41\KEY_STORE\DUMMIES\049751b150_HIII_LH.cnt
C:\pr41\KEY_STORE\DUMMIES\049751b503_HIII_RH.k
C:\pr41\KEY_STORE\DUMMIES\dummy_pass.cnt
C:\pr41\KEY_STORE\IP_AND_DASH\049752b054_IP_fascia.k
C:\pr41\KEY_STORE\IP_AND_DASH\fascia.cnt
C:\pr41\KEY_STORE\IP_AND_DASH\049752b401_LH_Strg_Col.k
C:\pr41\KEY_STORE\IP_AND_DASH\049752b451_LH_Strg_Wheel.k
C:\pr41\KEY_STORE\CHASSIS\049754b050_Lwr_Strg_Col.k
C:\pr41\KEY_STORE\CHASSIS\049754b701_Pedal_box.k
C:\pr41\KEY_STORE\POWERTRAIN\049755b725_HVAC.k
C:\pr41\KEY_STORE\POWERTRAIN\hvac.cnt
C:\pr41\KEY_STORE\SEATS\049756b250_LH_FEDNCAP_front_50th.k
C:\pr41\KEY_STORE\AIRB_AND_SEATB\driver_airb.cnt
C:\pr41\KEY_STORE\AIRB_AND_SEATB\Passenger_airbag.k
C:\pr41\KEY_STORE\AIRB_AND_SEATB\pass_airbag.cnt
C:\pr41\KEY_STORE\AIRB_AND_SEATB\seatb_frnt_lh.cnt
C:\pr41\KEY_STORE\AIRB_AND_SEATB\049757b501_Belt_RH_50th.k
C:\pr41\KEY_STORE\AIRB_AND_SEATB\seatb_frnt_rh.cnt
C:\pr41\KEY_STORE\INTERIOR_TRIM\049758b001_LH_door_trim.k
C:\pr41\KEY_STORE\INTERIOR_TRIM\int_trim_a_pillar.cnt
C:\pr41\KEY_STORE\INTERIOR_TRIM\049758b400_Header.k
C:\pr41\KEY_STORE\INTERIOR_TRIM\int_trim_header.cnt
C:\pr41\KEY_STORE\INTERIOR_TRIM\049758b450_LH_Carpet.k
C:\pr41\KEY_STORE\BIW\049760b500_vs2sled_v1.k
SELECT_FILES will switch to <select files mode> and preselect the
renumbered files.
To select renumbered files [SELECT_FILES]
To continue in master only mode [CONTINUE]

SELECT_FILES will then take you to the keyout selected panel.



Note - Primer stores a marker that the component has been renumbered, this is **not** a rigorous check for modified includes. To achieve that you need to use the compare to disk function described above.

3.15.1.6 Linking model to database

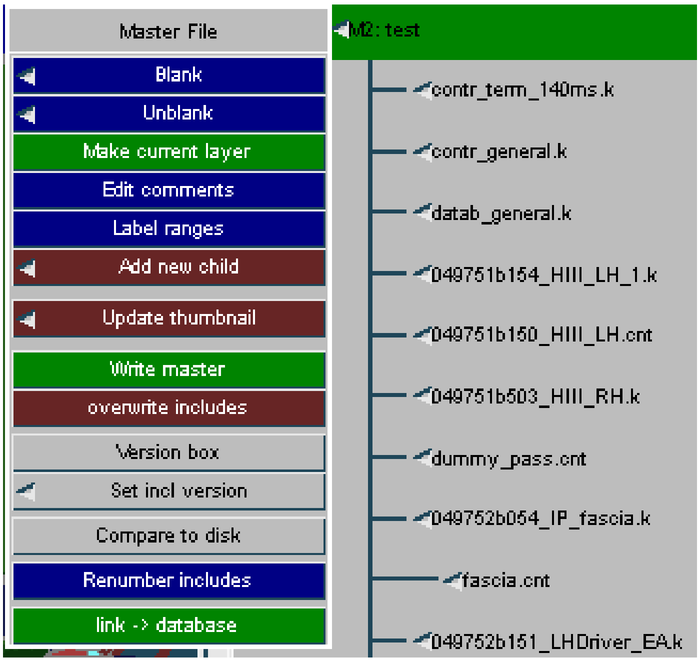
When a model has just been built, the include tree is linked implicitly to the database from which the model has been built as described above.

However, you may wish to write out the built model (probably just the master file) and come back to work on it later. This can be done in Primer by re-linking the read model to its database.

When you read in a model that has been built using the database/template method, a special comment in the master file enables Primer to recognize the origin of the model. Hover text on the model popup will show the database and template. By default the model is **not linked** to the database and may be treated as an independent entity.

By activating the **link -> database** option off the model popup, the include tree will behave as if the model has been built. It does not matter if the contents of model have been changed before linking. However, it is assumed that the include file structure has not been changed, i.e. includes have not been added/deleted or written to different names.

After linking the hover text that shows the provenance of each include will become active and **Edit database** will be available. Writing out includes (to different file names) is now handled as described above.



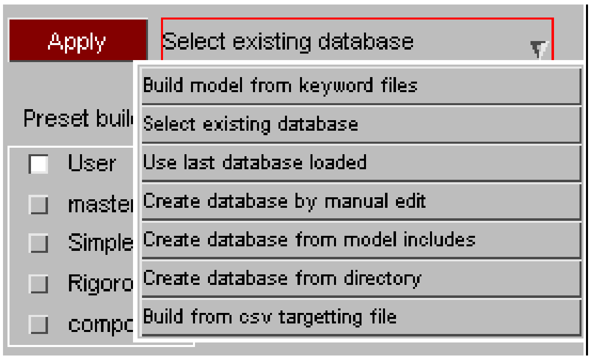
3.15.2 Creating and managing a model Database

- Creating a new model database
- Editing the model database
- Creating and editing database entries
- Templates

A Model Database consists of an hierarchical list of Include files and information regarding those Include files from which an individual can select a number of Include files to build a model.

3.15.2.1 Creating a new Model Database

Access the model Database creation options by clicking **BUILD** under the MODEL tab. Then select the appropriate build option.



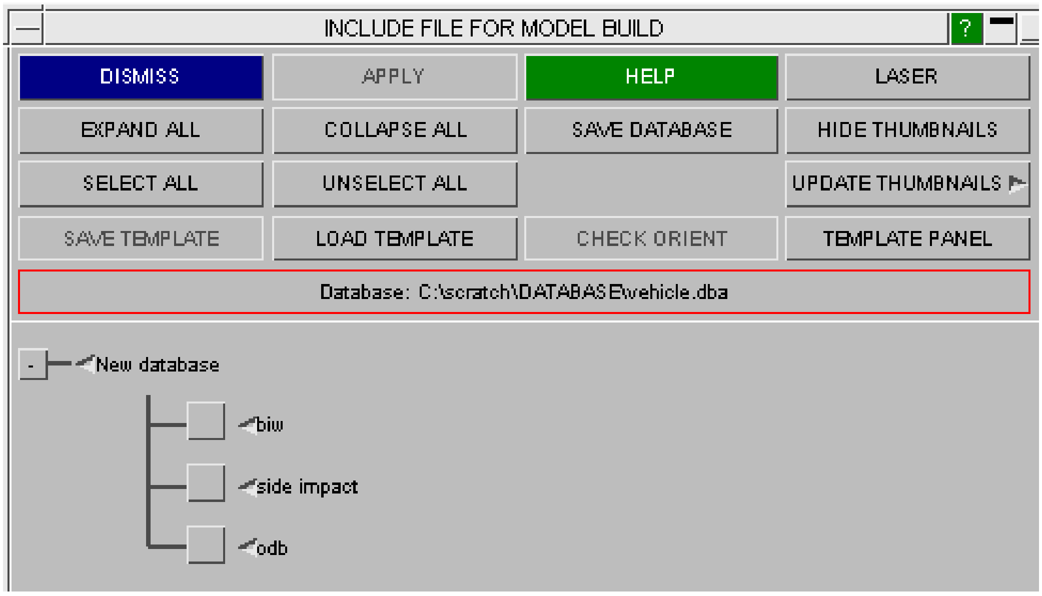
Create database from directory. If your files have been grouped under meaningful directory names (DUMMIES, BARRIERS, etc.) you can easily create a database with this option. You need only specify the start directory and Primer will locate all the ".key" files in sub-directories and create the database structure.

Create database from model includes. If you have a model in memory which contains an include file structure, you may create a database directly from the model. The database will group the keyword files for each directory, under a category named after the directory itself.

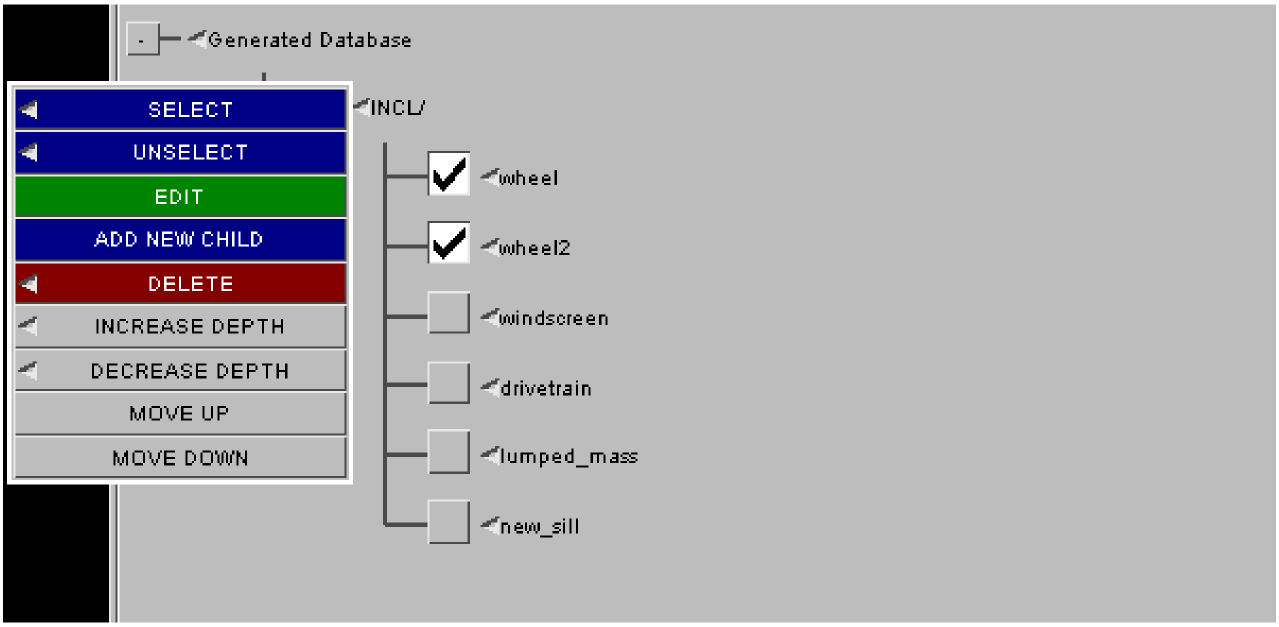
In both the above cases the new database should then be loaded by reverting to **select existing database** option.

Create database by manual edit. Apply will create a starter database with a single row. In order to add entries into the database, right click on the present entry and select **ADD NEW CHILD** in the popup menu and a window will appear asking you to provide information about the new entry. See the creating and editing database entries section. To save the database you have created, select the **SAVE_DATABASE** tab and input the path and name you wish to give to the database into the input box.

In order to write out a postscript file to provide a print out of the Database, select the **LASER** tab and fill in the required categories.



3.15.2.2 Editing a Model Database



In order to add entries into the database, right click on an entry in the layer above where you wish to create the new entry that would serve as a appropriate category for the entry and select **ADD NEW CHILD** in the popup menu. A window will appear asking you to provide information about the new entry. See the creating and editing database entries section for information about the contents of this window.
Current entries can be edited by right clicking on the entry and selecting the appropriate option.

- Edit
- Delete
- Increase Depth
- Decrease Depth
- Move up
- Move down

Edit

This option allows you to alter the information about the file inputted when first created. For more information see the creating and editing database entries section.

Delete

This option brings up a second pop up menu when you place your cursor over it. Selecting THIS will delete the entry selected, this option cannot be selected if the specified entry has any children. Selecting CHILDREN will remove any lower level files under this entry. Selecting THIS AND CHILDREN will remove both the selected file and any lower level files underneath it.

Increase Depth

This option brings up a second pop up menu when you place your cursor over it. Selecting THIS will move only the selected entry up a level in the database, this option cannot be selected if the specified entry has any children. Selecting THIS AND CHILDREN will move both the selected entry and all entries in lower levels up a level.

Decrease Depth

This option brings up a second pop up menu when you place your cursor over it. Selecting THIS will move only the selected entry down a level in the database, this option cannot be selected if the specified entry has any children. Selecting THIS AND CHILDREN will move both the selected entry and all entries in lower levels down a level.

Move up

Selecting this option will move the selected entry down an entry in their current level.

Move down

Selecting this option will move the selected entry down an entry in their current level.

3.15.2.3 Creating and editing database entries

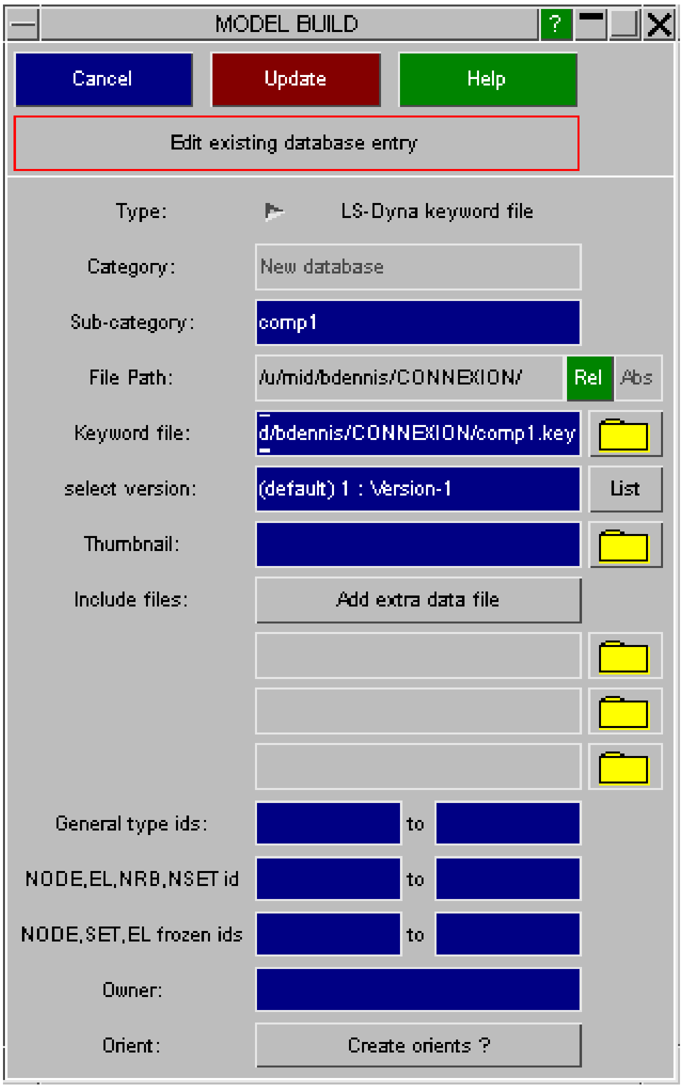
When adding a new child or editing an existing entry in a database the following information can be given;

- Category
- Sub-Category
- Model Path
- Keyword file
- Thumbnail
- Extra data files
- General type ids
- NODE,EL,NRB,NSET ids
- NODE,SET,EL frozen ids
- Owner
- Orientation data

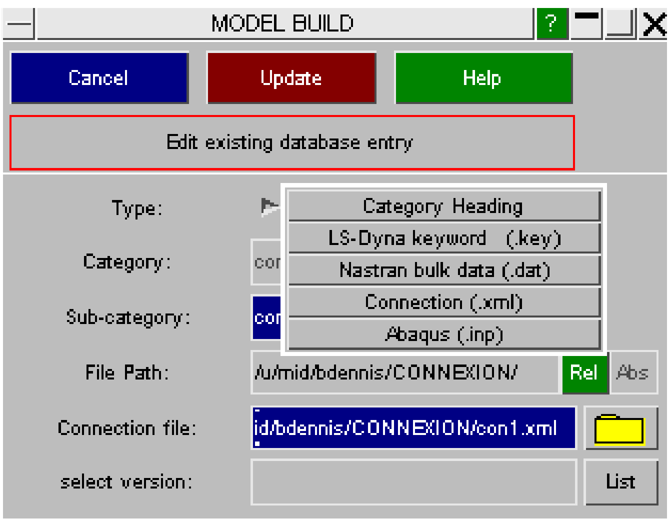
A category must always be specified if the entry is to exist in the Database. If the entry references a keyword file then a Sub-Category must also be present.

The combination of category and sub-category must be unique. This is how a database entry is referenced by a template, and is deliberately independent of the keyword file name. The user who is building the model is selecting the item category, not the keyword file directly. The person responsible for maintaining the database must ensure that the end user's selection gets the most up to date version of the component file.

Primer now allows you to store multiple keyword files for each component. The current version will determine which keyword file is actually used. See Version Control.



Component type



In addition to an LS-Dyna keyword file, the component may be described by a Nastran or Abaqus file (for which Primer supports a subset of keywords).

Also an xml connection file may be loaded as a component of the build. These may be managed using the same version control as keyword files.

On completion of the build a connection panel will be invoked to process the connection file(s). See build with connections.

Category

This option allows you to give the database entry a title by which you may identify it in the future. If no sub-category is given then the category will be displayed as the name of the entry.

Sub-category

The Sub-category acts as a second name to identify your the entry. If specified it will also be displayed as the name of the entry.

Model Path

This contains the path to the keyword file and is automatically generated from the full path when a file is selected from the selector box. If you type in the filename directly without a path, the file will be expected to be in the same directory as the database.

Usefully, the path may be set to Absolute or Relative using the **Abs** or **Rel** button.

Keyword File

If you wish the entry to contain a keyword file, enter the name of the file here.

Thumbnail

This option enables you to insert a bitmap image to act as a graphical representation of the contents of the entry. Enter the name and path of the bitmap file here. The image will be displayed alongside the name of the entry in the Database.

Extra data files

Press **ADD EXTRA DATA FILE** to specify the name(s) of any extra data file(s) which are to be associated with this component.

These will always be read as includes immediately after the main component file has been installed. Although the files may contain any LS-Dyna keywords, they would normally be expected to contain items such as contacts, rigid body merge connection, nodal rigid bodies, etc. which connect component files together. Therefore, unlike the component files, these files are not "stand alone" and will contain references to items in component files. If a referring component is omitted, this risks leaving the extra data file with reference to a missing item (e.g. part-set of contact refers to missing part). To handle this, Primer on completion of build will find all latent items and offer to delete them using FIX LATENT function. On completion of deletion, you will be given the option to move the contents of the modified extra data file up into the master file and suppress the keyout of the include file itself. If you take this option, you will not need to save the modified include.

General type ids

This option allows you to specify a number range into which PRIMER will renumber general items (excludes nodes, elements, nrbs, node sets).

Any items labelled outside the range specified will be renumbered to fit in the range. If the range is not set no renumbering of these items will occur.

Input the lowest boundary of the range to the first input box and the highest boundary of the range to the second input box.

If any parts, sections or materials lie outside the specified range an error will be reported. These will never be renumbered.

NODE,EL,NRB,NSET ids

As nodes, elements, node sets and nodal rigid bodies often require a larger range than other items, they have their own item range.

All nodes, element, node sets and nodal rigid bodies lying outside the node/el range will be renumbered into this range.

If the range is not set no renumbering of these items will occur.

Note on latent items: If an item in an include file is effectively latent, e.g. a material card that is referred to but not actually in the file, it will **not be renumbered**.

NODE,SET,EL frozen ids

This option allows you to specify a range of node, element, node set and element set numbers that will never be renumbered in the build process.

This can be useful if you wish to preserve your time history items or if you wish to protect items (other than parts) used for connection.

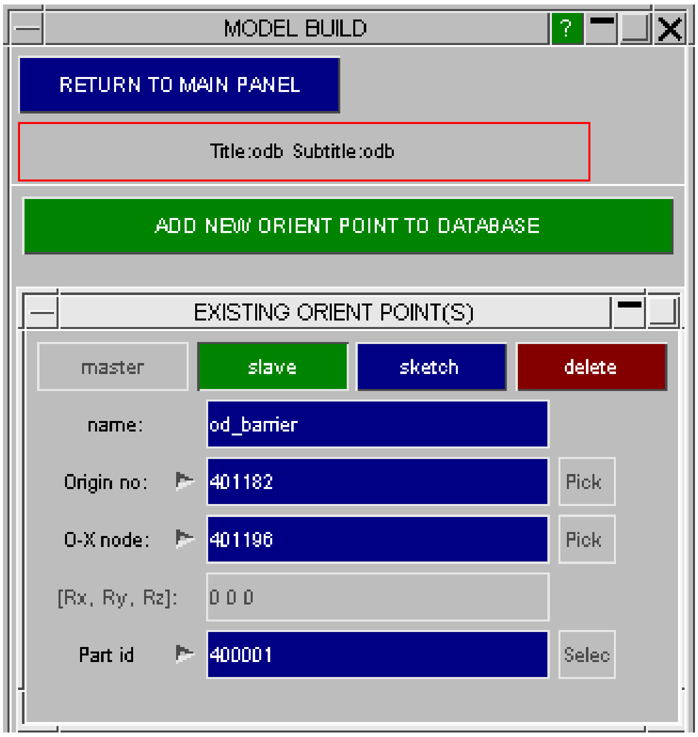
Owner

Enter the owner of the file here.

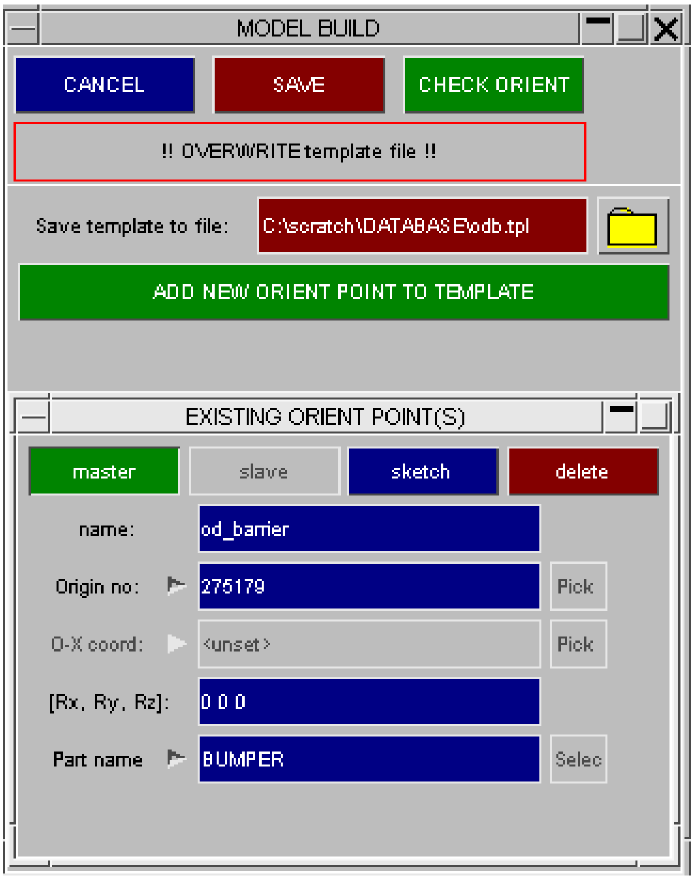
Orienting the include files

It is possible to orient include files during the model build process. This is achieved by generating *INCLUDE_TRANSFORM rather than plain *INCLUDE.

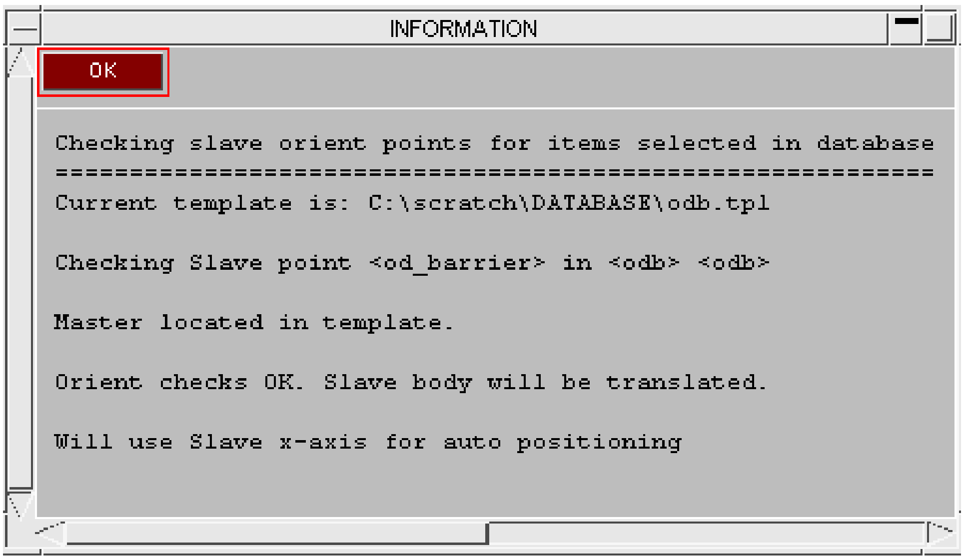
The Orient create/edit feature is accessed through the **Create/Edit slave/master Orients** button on the category edit panel.



The user creates each orient by adding a master and a slave point of matching name. The slave points will always reside in the database, under the include file to which the orient is to be applied. The master points may be stored either in the database or in the template as they apply for a particular load case.



After you have created/edited orients you need to save the database or template. It is recommended that you then run the **CHECK ORIENT** function, which will sketch the orient as well as report its status.



In the simplest orient case you need to define a master point and its co-ordinate (or node id) and a slave point of the same name and its co-ordinate. The build process will detect matching master & slave orient points, resolve any node ids into co-ordinates and calculate the necessary transform to bring the slave point to the master. A*INCLUDE_TRANSFORM will then be applied to the include file associated with the slave orient.

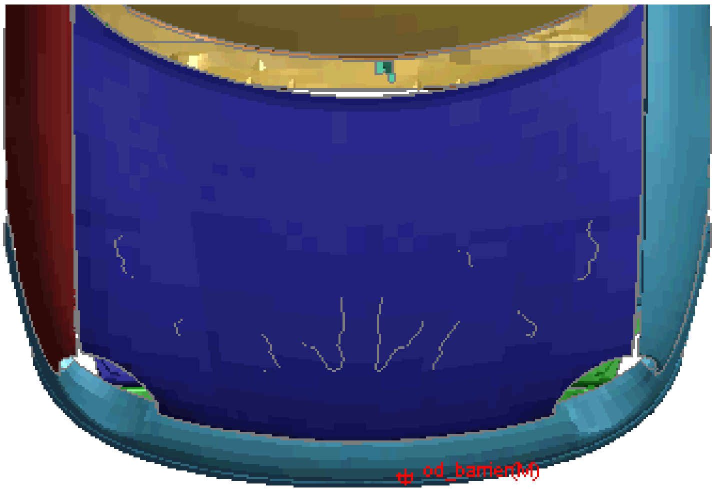
Rotation of component model: A master orient point may additionally include global rotations [Rx, Ry, Rz about the master point] which will be applied to the slave include file.

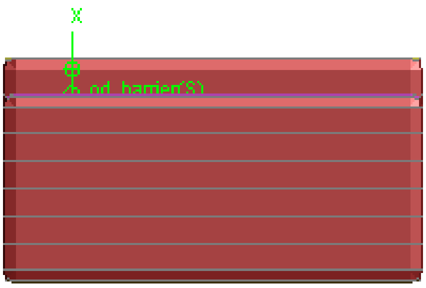
Moving into position: On the slave orient a second point (O-X) may be defined (co-ordinate or node id) which defines a "line of flight" vector. This defines the direction in which the impactor will be moved if it is *not penetrating* and against which it will move if it *is penetrating*. In the normal case where the impactor is initially positioned away from the vehicle the vector should point toward the vehicle (as shown below).

Additionally information is required so that a contact can be created and the slave body will be either depenetrated or advanced along the "line of flight" vector until it is on the point of contact. Thus an odb barrier can be set up so that it will be optimally positioned for different bumper designs with minimum wasted cpu time before the onset of impact.

Setting up the contact. The orient point may reference a contact that exists in the model directly. If so, this may be located on the master orient point or the slave point. In the event of both being defined, the one on the master orient will be used. Alternately, a part or part-set may be defined for both the master and slave orient points, these will be used directly to define a surface-surface contact. Alternately, a single part set may be defined for the slave side only. In this case, Primer will split the set up to form a surface-surface contact between the impactor and the vehicle. For part/part-set method the contact is disposable and will not appear in the model.

Definition of orient points. These may be defined as node id, node name (if *DATABASE_HISTORY_NODE_ID) or a co-ordinate. The node method has the advantage that if the component files get moved the orient points will still be in the correct position. If the node is defined by ID it must not be renumbered. This may required the node to be included in the frozen range if renumbering during model build is active. If using the co-ordinate method the orient data must be updated if component files are moved.

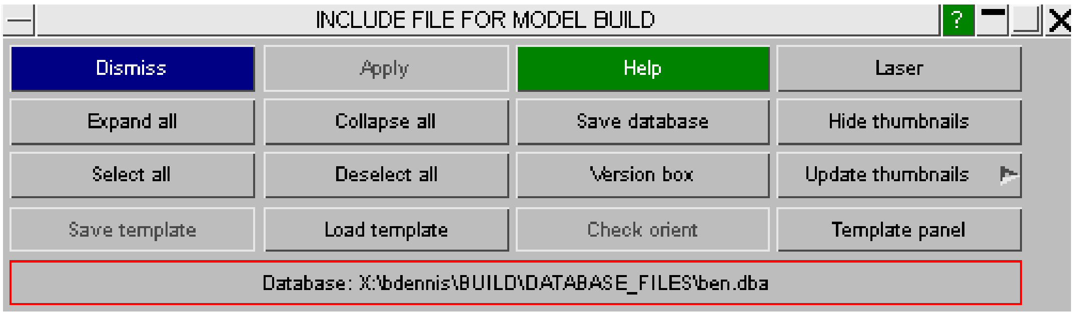




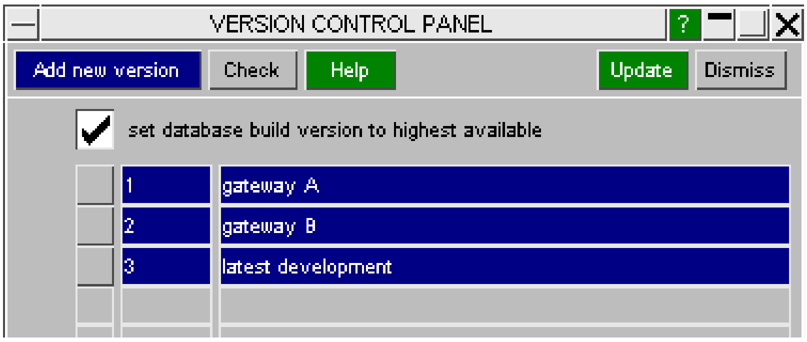
3.15.3 Version Control

Version control provides a powerful method of keeping track of the development of component models during a project. It also allows you easily to recreate a previous version of your model should you need to do so. The information is stored both in the database and a corresponding .history file which Primer updates.

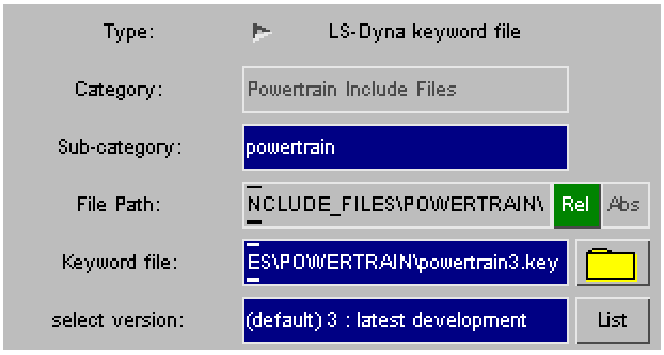
One way of managing version control would be this. At suitable point you will add a new version to the database using the "**Version Box**" function, e.g. version 3 for "latest development" build. You will freeze all the component files and copy them, e.g. copying the current development *powertrain.key* component file to *powertrain3.key*.



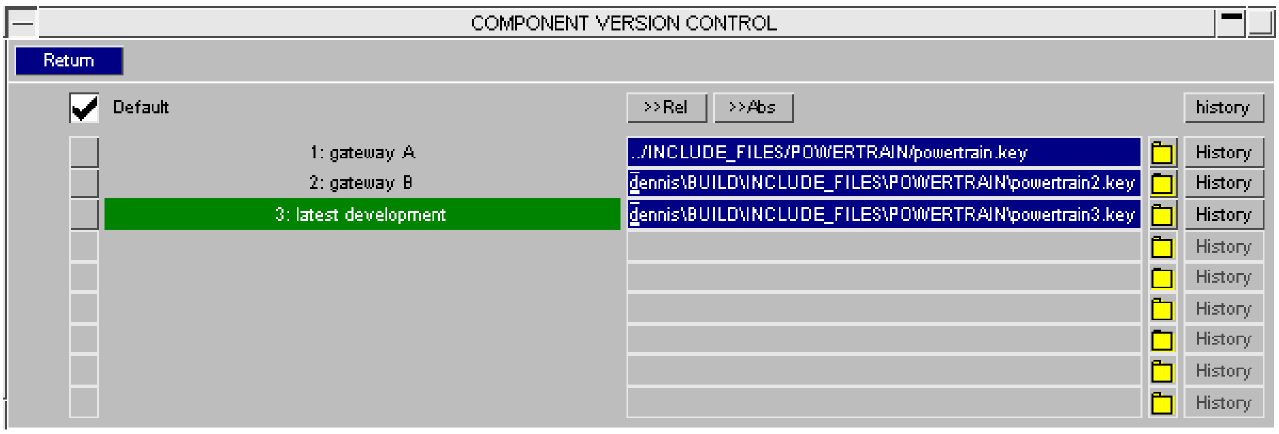
The version control panel by default will be set to use the latest version (v3 in this case).



Using the edit function on the build tree, select the component of interest and, instead of selecting a keyword file directly, press **LIST** to access the multiple keyword panel.

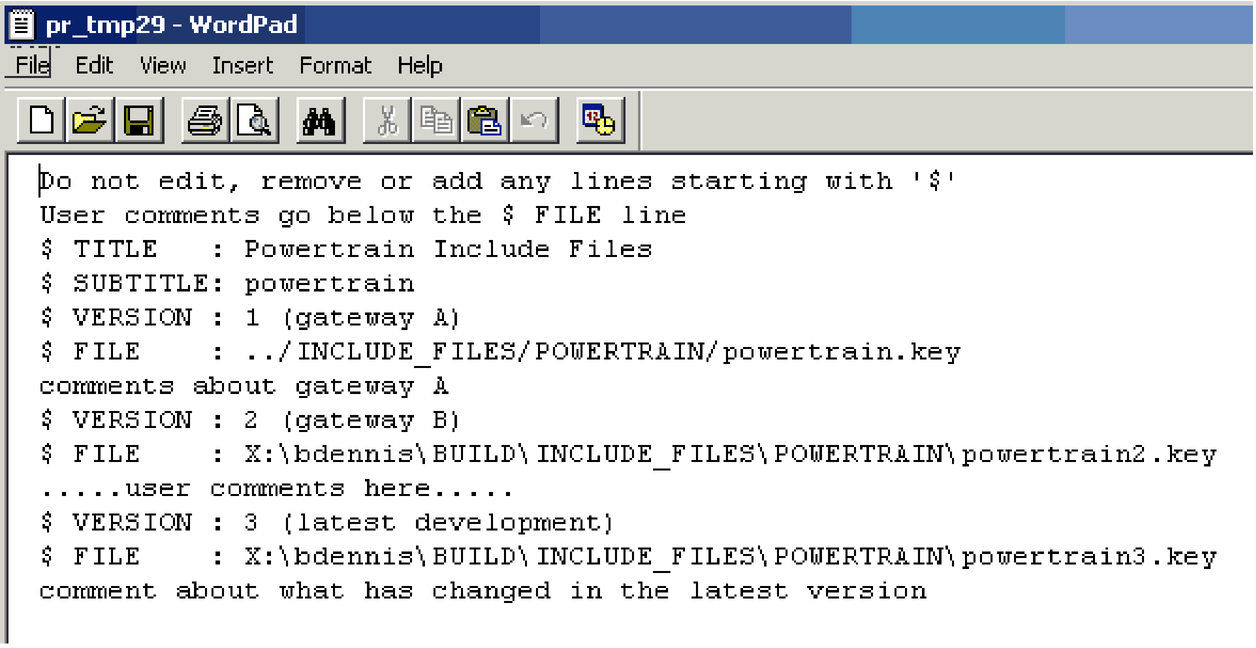


Add in the new file as the version 3 component. Return and update the component.



3.15.3.1 User defined history

By pressing the **HISTORY** button you can bring up a text editor to add history comments to the describe the version.



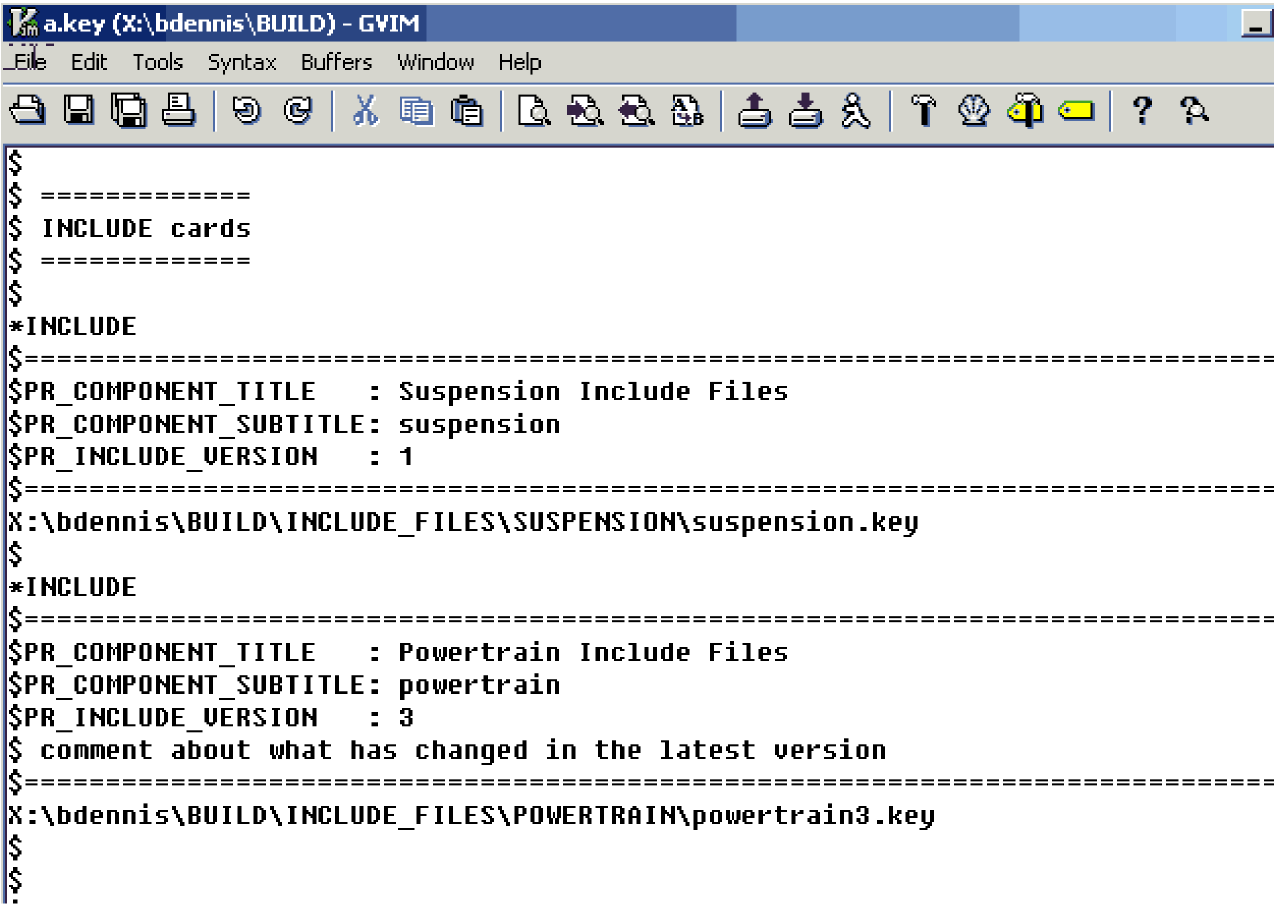
pr_tmp29 - WordPad

File Edit View Insert Format Help

```
Do not edit, remove or add any lines starting with '$'
User comments go below the $ FILE line
$ TITLE    : Powertrain Include Files
$ SUBTITLE: powertrain
$ VERSION : 1 (gateway A)
$ FILE     : ../INCLUDE_FILES/POWERTRAIN/powertrain.key
comments about gateway A
$ VERSION : 2 (gateway B)
$ FILE     : X:\bdennis\BUILD\INCLUDE_FILES\POWERTRAIN\powertrain2.key
.....user comments here.....
$ VERSION : 3 (latest development)
$ FILE     : X:\bdennis\BUILD\INCLUDE_FILES\POWERTRAIN\powertrain3.key
comment about what has changed in the latest version
```

Build of the model will now use version 3 if available for all components. If the version is not available it will use the highest previous version.

Version information and the relevant history comment will be written out with the master file.



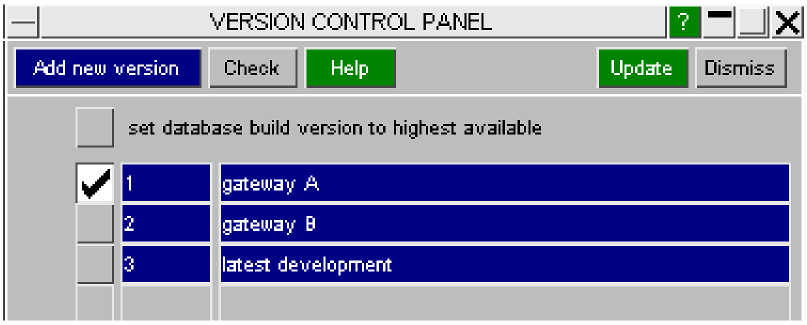
a.key (X:\bdennis\BUILD) - GVIM

File Edit Tools Syntax Buffers Window Help

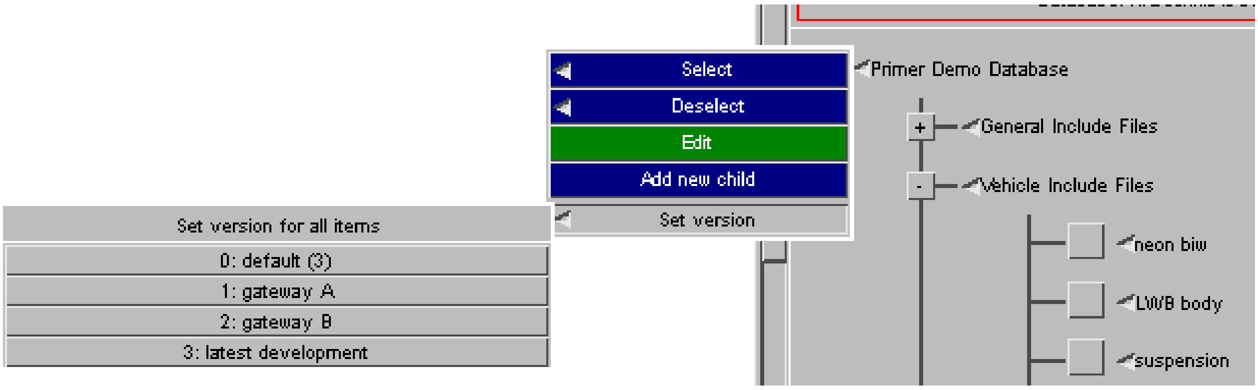
```
$
$ =============
$ INCLUDE cards
$ =============
$
*INCLUDE
$===============================================================================
$PR_COMPONENT_TITLE    : Suspension Include Files
$PR_COMPONENT_SUBTITLE: suspension
$PR_INCLUDE_VERSION    : 1
$===============================================================================
X:\bdennis\BUILD\INCLUDE_FILES\SUSPENSION\suspension.key
$
*INCLUDE
$===============================================================================
$PR_COMPONENT_TITLE    : Powertrain Include Files
$PR_COMPONENT_SUBTITLE: powertrain
$PR_INCLUDE_VERSION    : 3
$ comment about what has changed in the latest version
$===============================================================================
X:\bdennis\BUILD\INCLUDE_FILES\POWERTRAIN\powertrain3.key
$
$
```

3.15.3.2 Setting the applied version

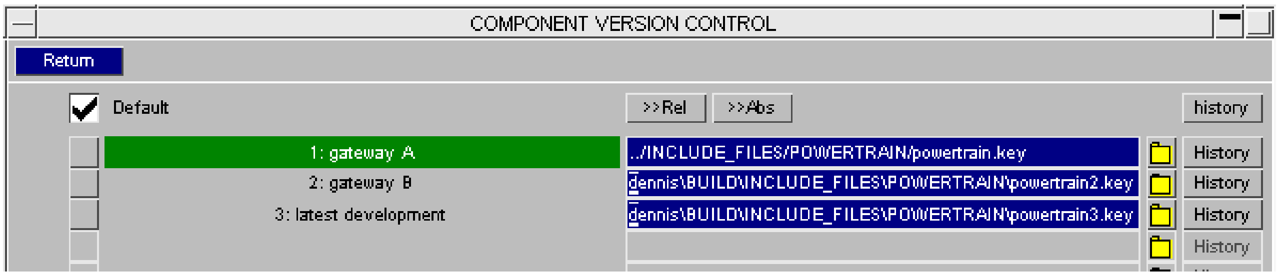
To build a previous version of the model, use the version box to select the one you want, e.g. gateway A (version 1)



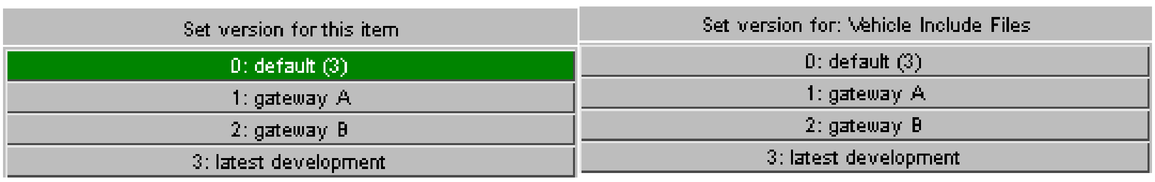
Use the popup of the head node of the build tree to set all component files to use default version (this is their default setting).



All component files will now use version 1 as shown by inspecting the edit panel.



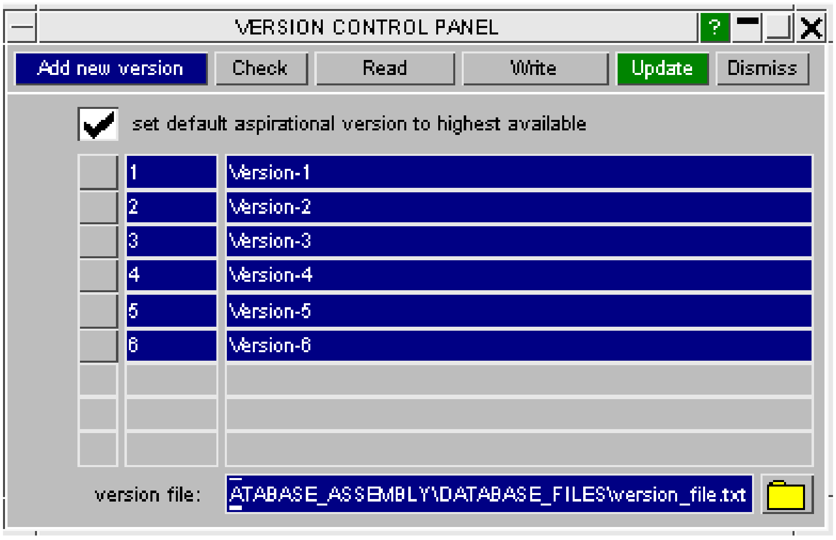
The version may be set on individual component files, or for a whole branch of the build tree, using the version popup.



3.15.3.3 Recording a snapshot of versions

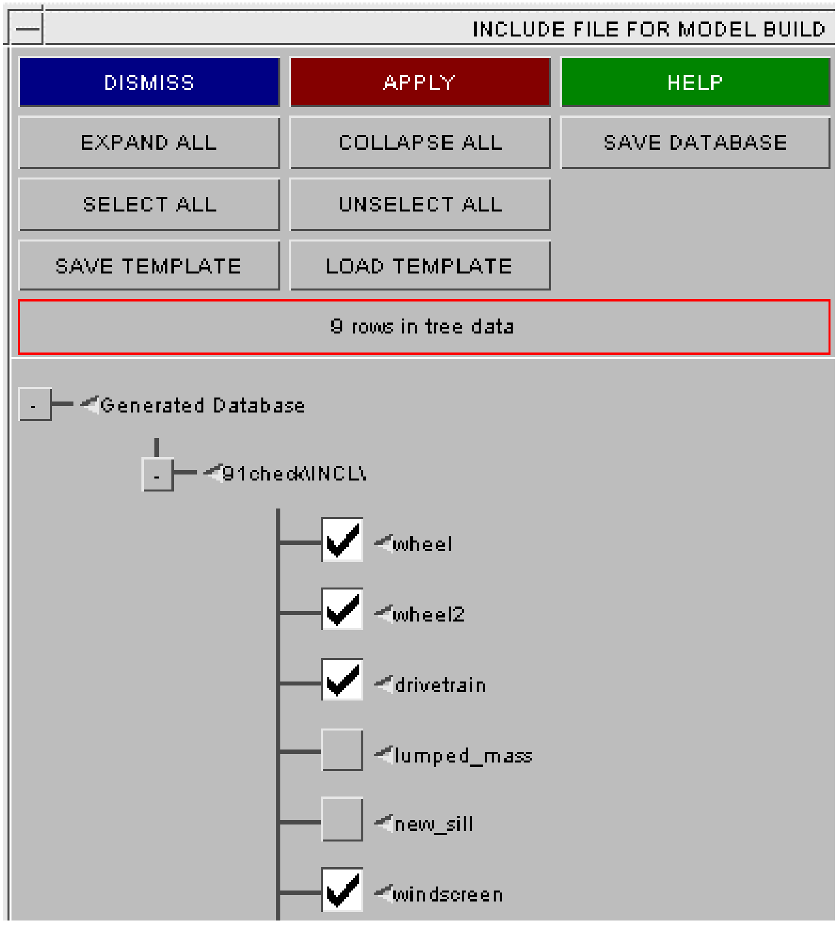
The method described above works on the assumption that the development of component files of the project marches progresses in a monotonic manner. Some users have requested a more exact control of which version applies to which component.

This can be achieved by the writing/reading a version control file. **Write** will dump a text file with the current version of each component file in the database. **Read** will apply the specified versions in the text file to each component. Thus a particular build can be constructed using the version setting tools described above and the file can be written to capture these. At a later stage of the project, the snapshot may be recaptured by loading the database and template and applying **Read** of the version file.



3.15.4 Templates

The template provides an easy way to select a set of include files with which to build a model.



Templates provide a way of saving particular combinations of include files in order to allow you to easily read in a particular, frequently used, pattern of files without having to select each file from the database every time you build a model. Generally, there will be one database for a vehicle programme, and one template for each load case or variant.

In order to save a particular combination of include files, select the desired combination in the Model Database window and press **SAVE TEMPLATE**.

When a standard keyword file is present, the template will reference *only the category and sub-category* of the mentioned file, hence if the keyword files of the database are externally updated, the model read in from the template will automatically be the latest one.

If a template is saved that lists a non-standard include file (i.e. the user has modified the original database entry - it shows in red), the name and path of the include file will be specified in the template. When this template is read in, a warning will be given that non-standard keyword files are being used.

In order to read a template file click on the **LOAD TEMPLATE** tab and the selection of files in the template will be selected in the model database.

3.15.5 Editing multiple templates

On vehicle programs there will be many variant load-cases to analyze and consequently many templates to handle. The TEMPLATE CONTROL PANEL accessed from the **TEMPLATE PANEL** button will display contents and allow modification of multiple templates.

Primer will locate all the templates that exist in the search directory, applying the filtering string if it is set.

Reread All will discard any current edits and reread templates from disk.

Add new tpl will create a new blank template which can be populated and saved.

Increment all will modify all loaded template names, such that *fred.tpl* -> *fred_001.tpl* or *fred_001.tpl* -> *fred_002.tpl*. This allows easy version control for templates. The renamed templates must then be saved. Special logic has been added so that *fred_1.tpl* will increment to *fred_2.tpl* (not *fred_002.tpl*).

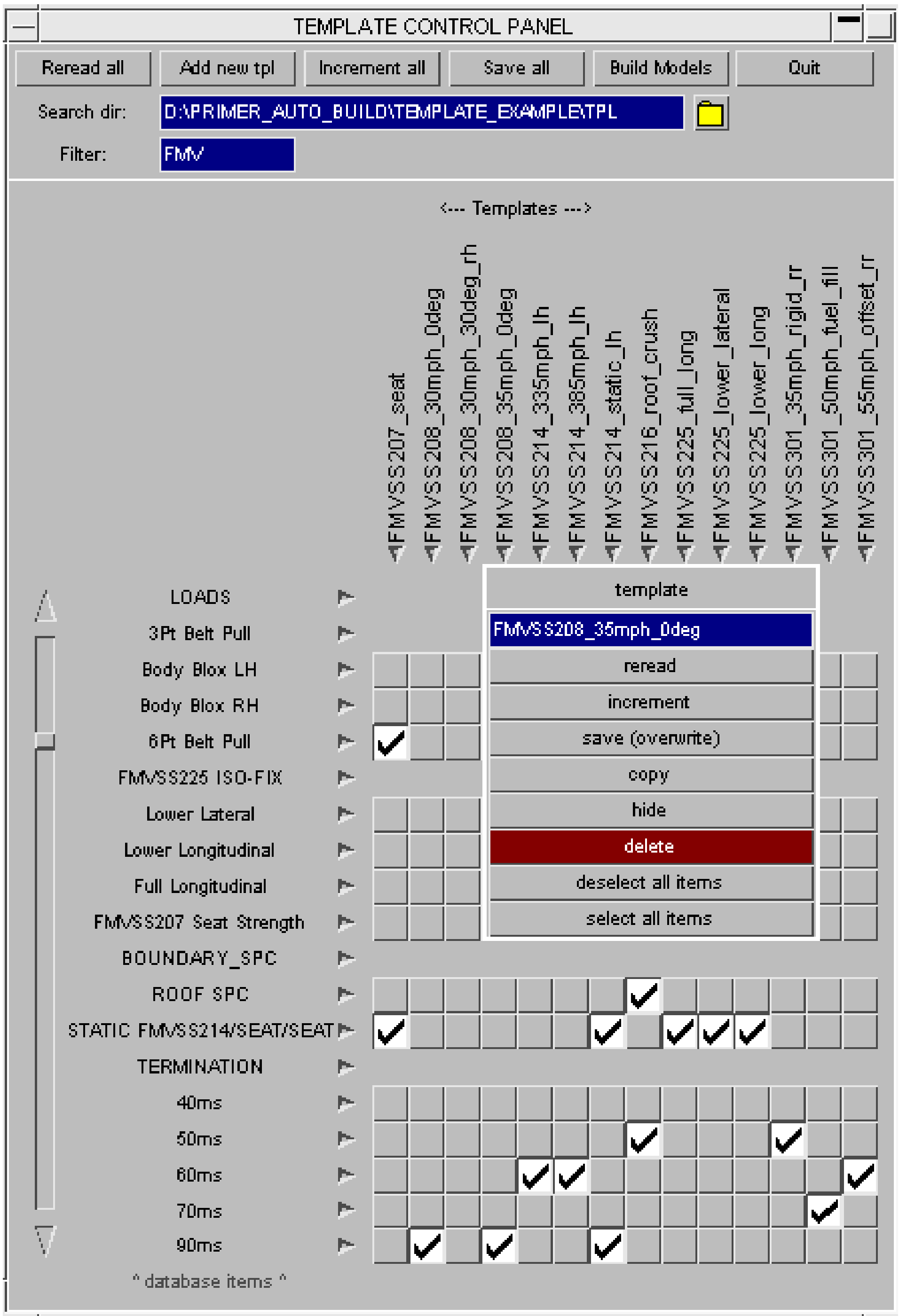
Save all will save all the loaded templates to disk in the search directory, overwriting (without warning) if necessary.

Build Models starts multiple model build panel, see below.

Quit returns to the database panel.

The above functions may be activated for a single template by using the drop down (as shown below). Additionally, this allows user to **Copy** an existing template.

The Database item popups allow selection of an item across all templates. They also access the same category edit panel that is available from the database panel.

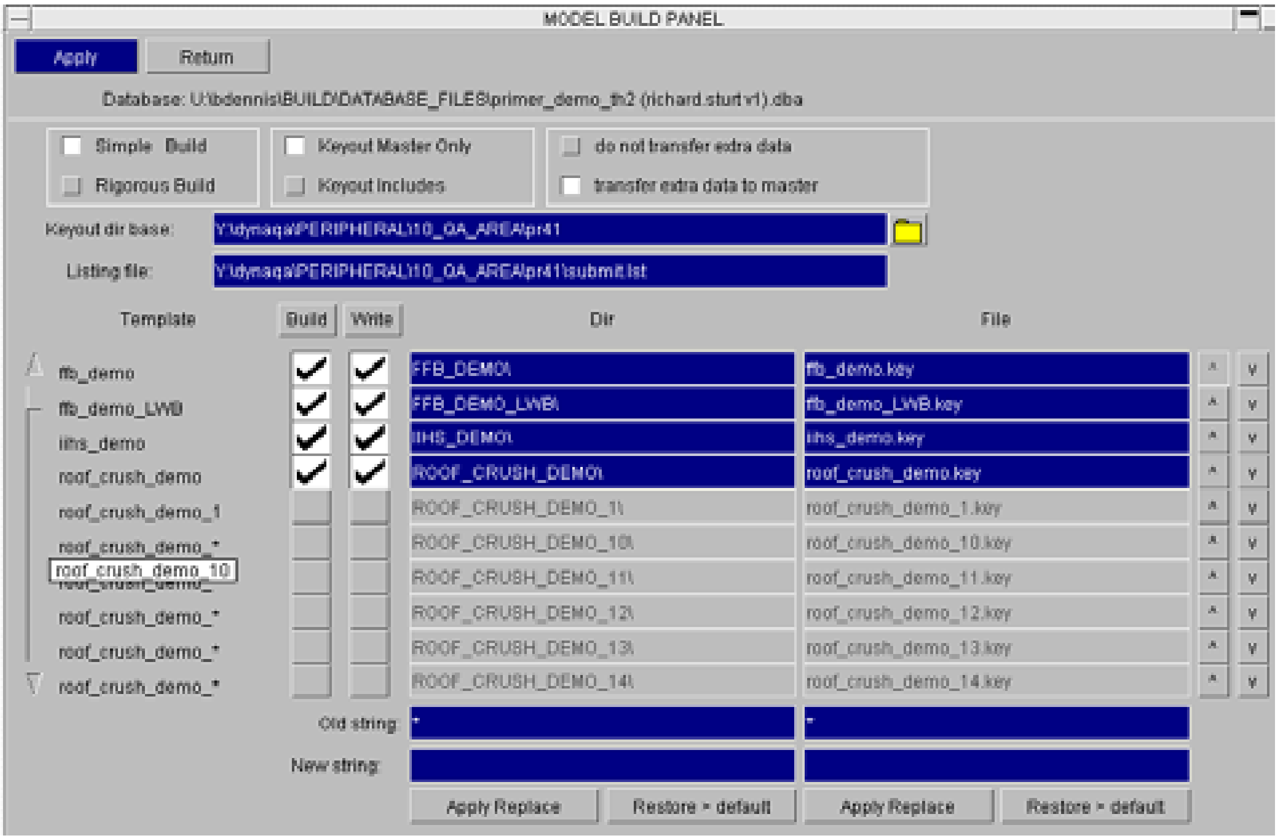


3.15.6 Multiple Build from template panel

The **Build Models** button will take you to a panel which enables the build process for all the active templates. As this process reads files from disk, it is essential that the database and all templates have been saved.

Build. Activating this will mean that the model will be built and retained as a model in memory in Primer.

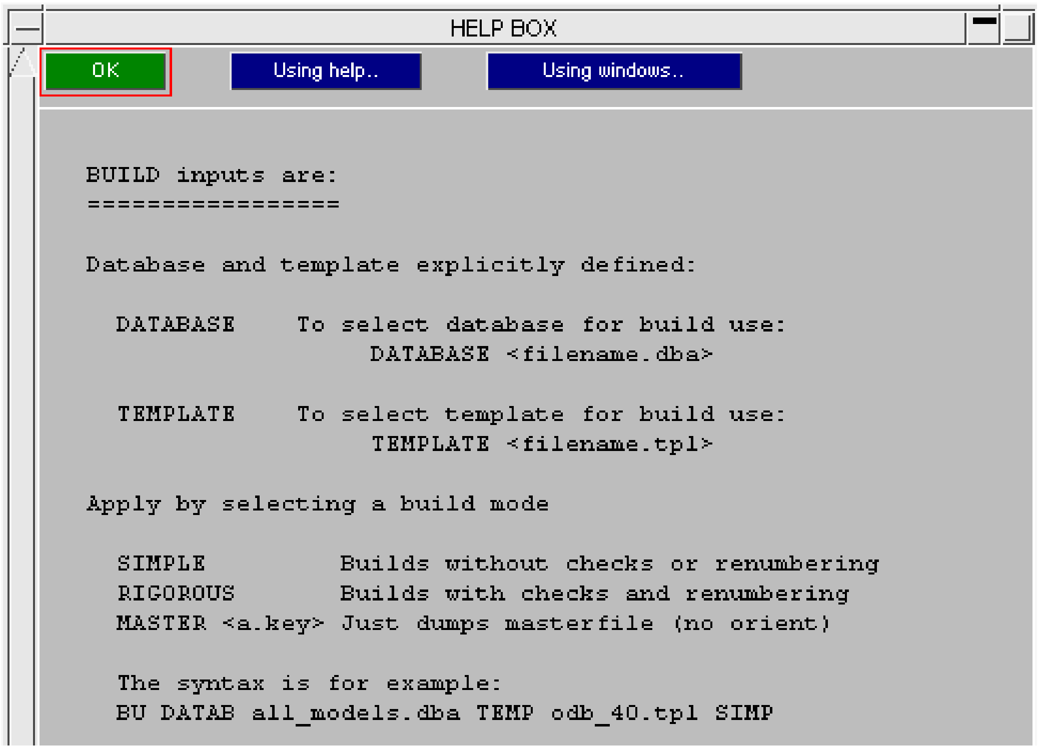
Write. Activating this without build will mean that the model will be built (if necessary), keyed out and then deleted from memory. If build is of simple mode and contains no orientation, it is not necessary for primer to build the model. The filename and directory are automatically generated, based on the template name, the directory being appended to the keyout directory base. A listing file will also be written which can be read by the Shell to submit a set of LS-Dyna jobs.



3.15.7 Single Build from command line

The database/template build may be run in simple, rigorous or master only mode from the command line or in batch mode using a command file.

Type **BUILD** on the command line to set the mode. Then **HELP** to get a description of the syntax.



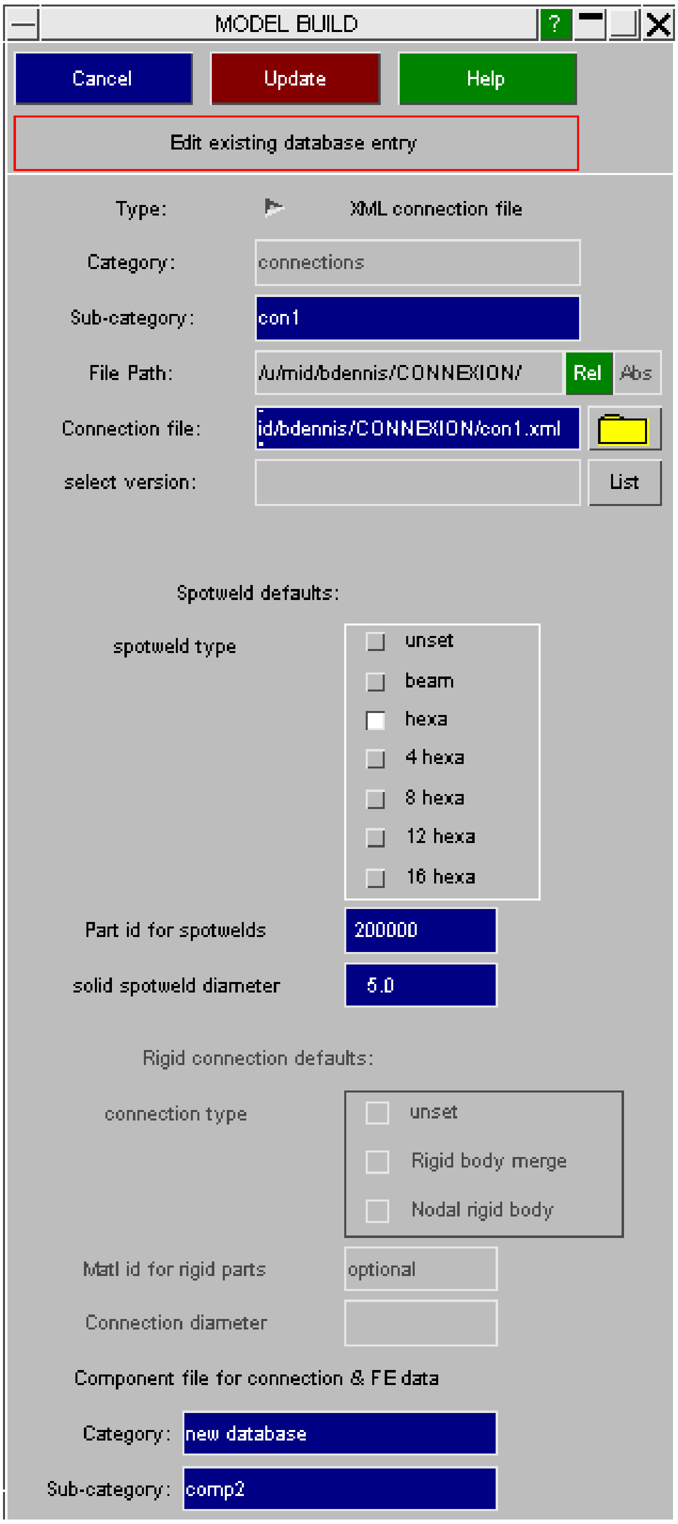
3.15.8 Connection file as component of build

Edit of component that is an xml connection file will bring up the panel below.

The contents of the xml file are parsed and default option are offered for the type we find. In this example we have only spotwelds. These default values will **only** be applied in cases where the data is missing in the xml file, e.g. no partid is specified for the spotwelds.

On completion of build the connections will be created in the include file which matches the category/sub-category set by the user "component file for connection" or, failing that, in the master file.

If you are using this method to make connections each time a model is built, you should **not** have the same connections stored as post-end data, although other connections may be.



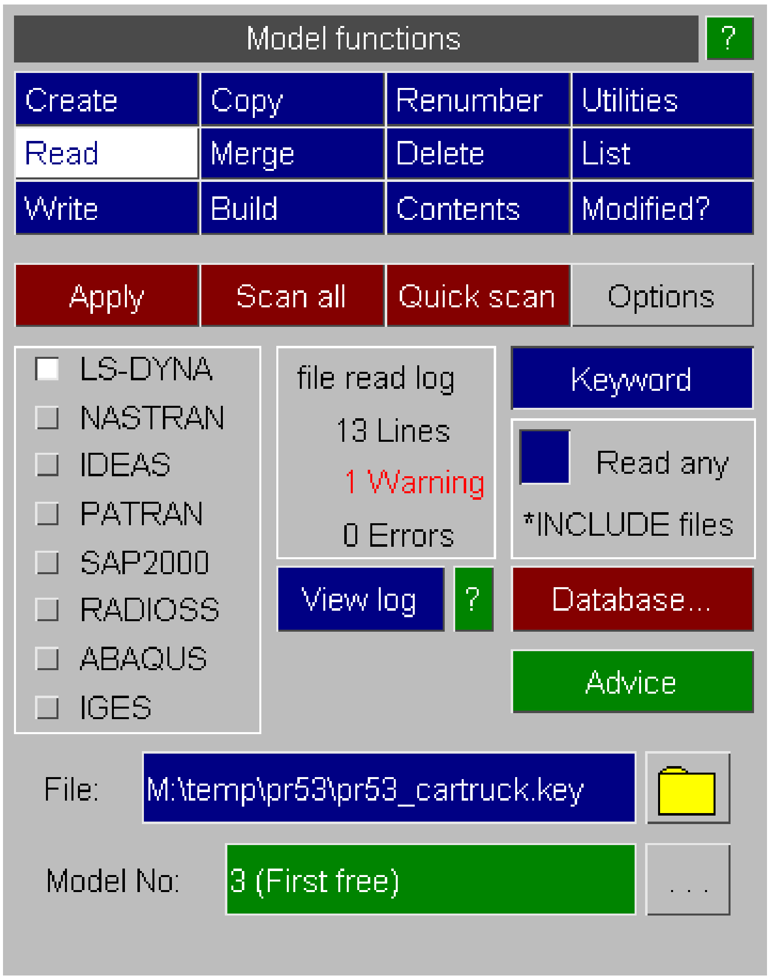
3.15.9 Reading files using a Model Database

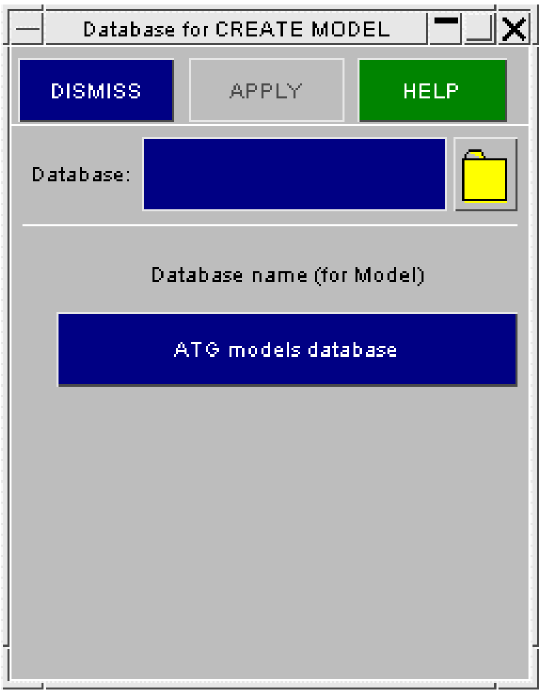
In the menu, click on **Model** and select **Read.**

In the **READ** menu press the **DATABASE** tab.

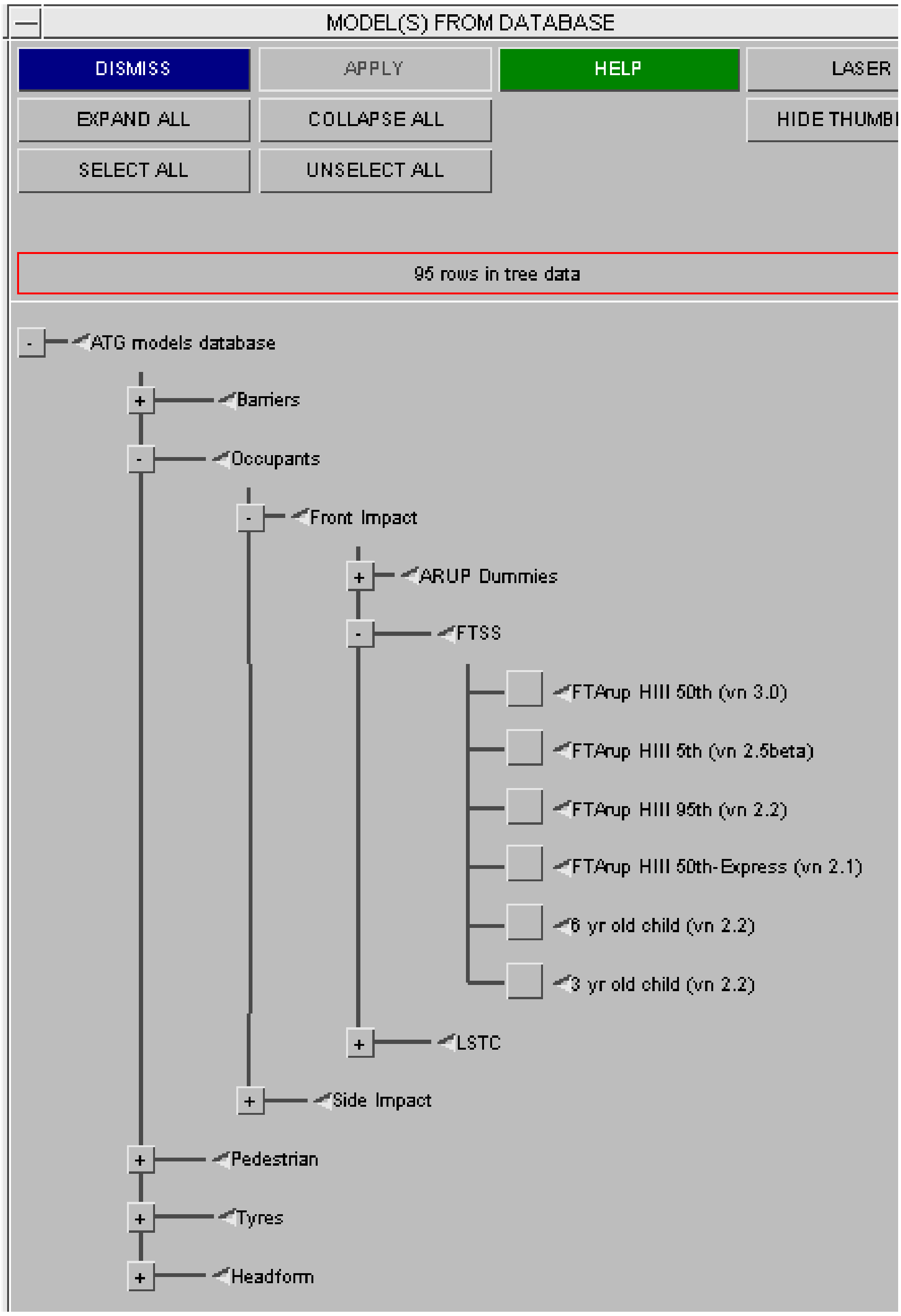
Select the database you wish to read in by inputting the name and path in the input box or using the search facility or by selecting one of the databases listed in the Database list.

Select **APPLY** to load the specified database



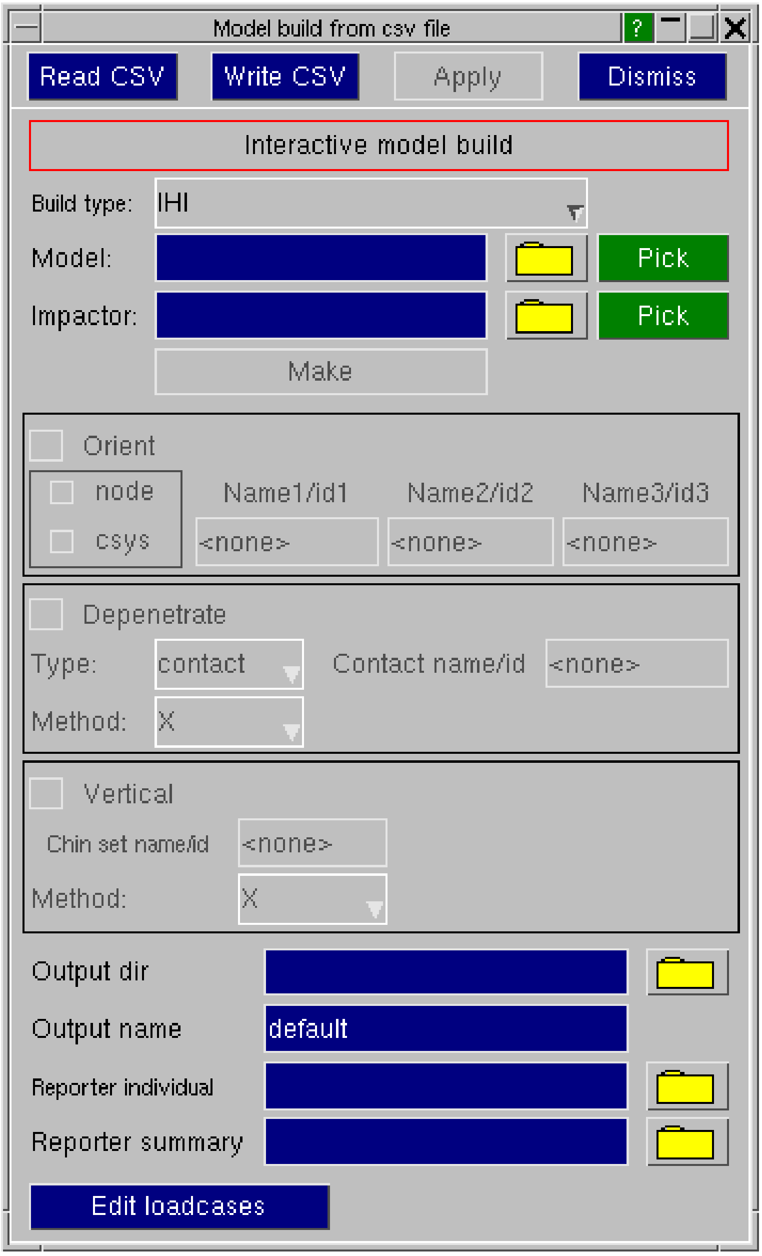


Select the file/files you wish to read in the model database window and select **APPLY**. The selected files will all be read in to **separate Models** in PRIMER. The files in this database may be compressed, as PRIMER will search for a compressed version of the file, if it fails to find the uncompressed one.



3.15.10 Building using csv targeting file

Select **MODEL**->**BUILD**. Choose the **Build from csv targeting file** option.



A model and an impactor can be read using the appropriate text boxes or file selectors. Alternatively, an existing CSV file can be read in. A model and an impactor are automatically located. Selecting the **Make** button will merge these into the active model.

The following templates are available, of which one can be chosen:

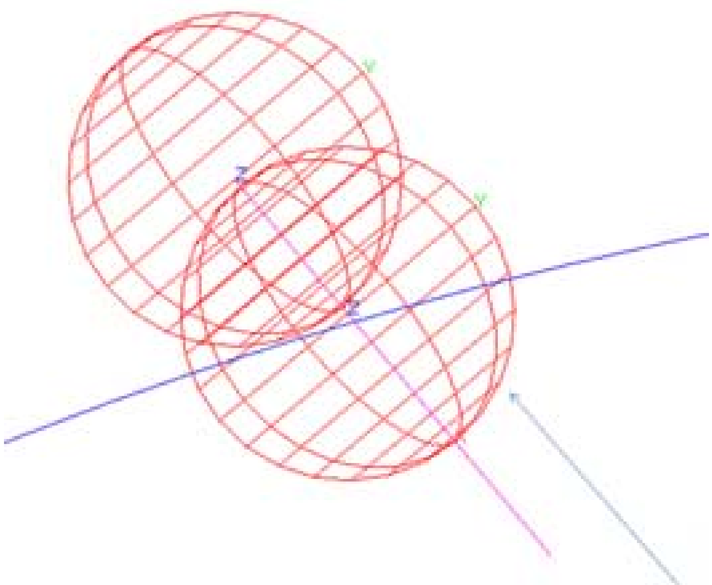
- IHI
- PEDHEAD
- PEDHEAD_ANGLE
- PEDLEG_LOWER
- PEDLEG_UPPER
- GENERAL_TRANSLATE
- GENERAL_TRANSLATE_ROTATE

Refer to Appendix IV for more information about these templates.

Orientation and depenetration options, root directory, output file name, reporter individual template, and reporter summary template can also be specified using appropriate text-boxes/selectors.

Note on IHI positioning. Rather than just positioning to a set vertical angle, Primer can now automatically position the IHI headform to the maximum vertical angle. The process is positioning the headform at zero vertical angle, rolling the headform down until the chin touches the trim, then rotating the headform back by a set back angle. The user needs to specify a shell set that represents the chin of the headform. The user can choose the method of head depenetration when rotating.

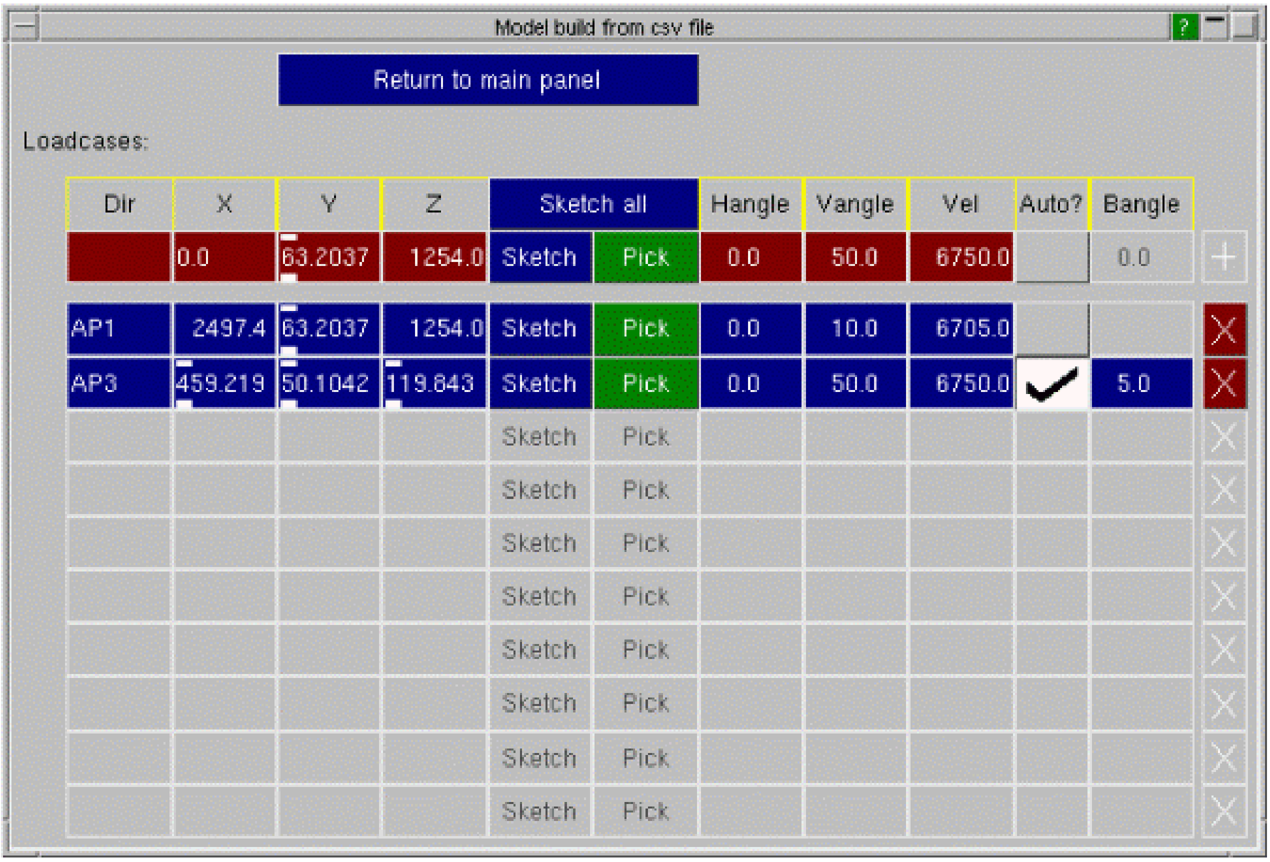
With the default depenetration method ’X’, the headform will roll off the target point as it would in reality.



Using the ’XZ’ or ’XYZ’ setting, Primer will attempt to move the headform back towards the target point after each rotation iteration.

The back angle (Bangle) is set on the loadcase panel for IHI (see below). On this panel the user must also specify that the loadcase uses the auto-vertical method. When the auto-vertical method is used, the vertical angle specified (Vangle) is the maximum angle the headform will rotate to when carrying out the automatic process.

Load-cases can then be specified by selecting the **Edit Load-case** button

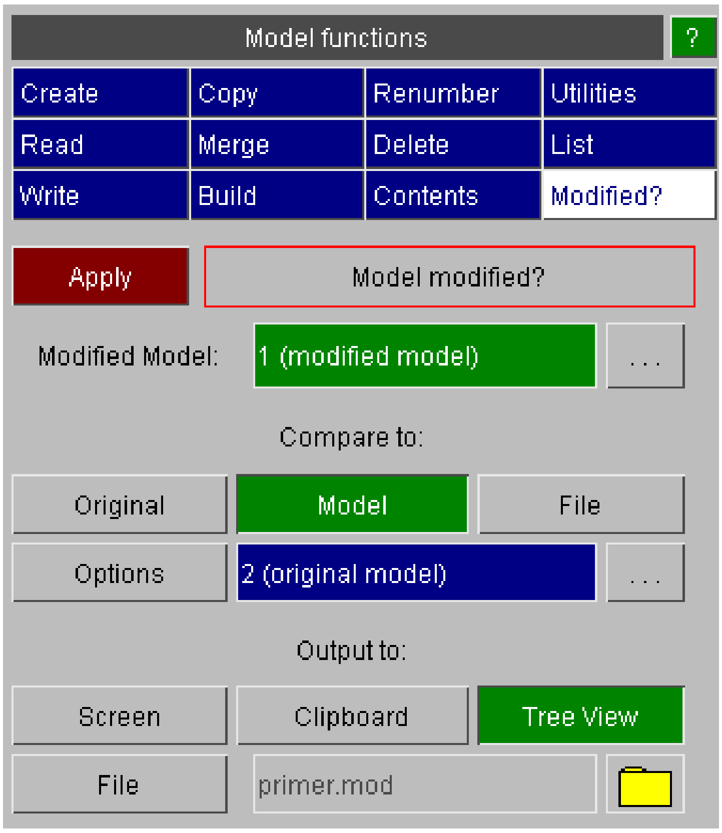


It is recommended that users write the CSV file out before proceeding with the model build. A model save operation might also be necessary in certain cases.

3.16 Model Modified

The model modified function allows you to:

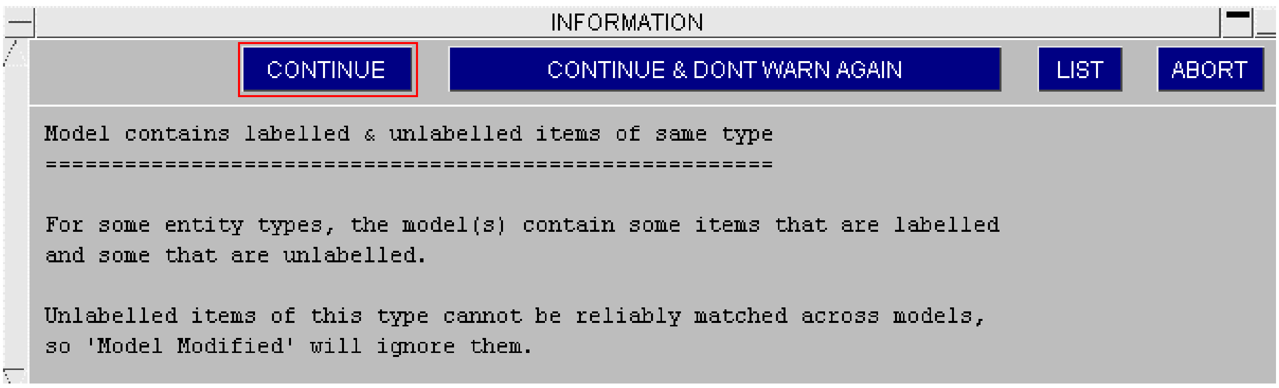
- see if a model has changed (compare to original)
- compare two models (modified model vs original model)
- compare a model to a file



By default the output is displayed in a **Tree View** as described below, but it can also be sent in the form of listing to the **Screen** or to a **File**.

Items which have been changed or created in the modified file can be put on to the **Clipboard** so you can view/modify them as required.

Primer will report items that have been created in the modified model (**only in modified model**), items that have been deleted from the original model (**only in original model**) and items that have been matched across models which have been modified (**differ**). For labelled items match across models is trivial. For unlabelled items this is done by trying to match the data on the cards. It is not always possible to tell whether an unlabelled item has been modified or created/deleted. There is a particular difficulty with types which admit of both labelled and unlabelled items (e.g. CONTACT). If all are labelled there is no problem. If they are all unlabelled, they will be treated the same as an unlabelled type. If some are labelled and some unlabelled, Primer will give a warning message and decline to treat the unlabelled items.

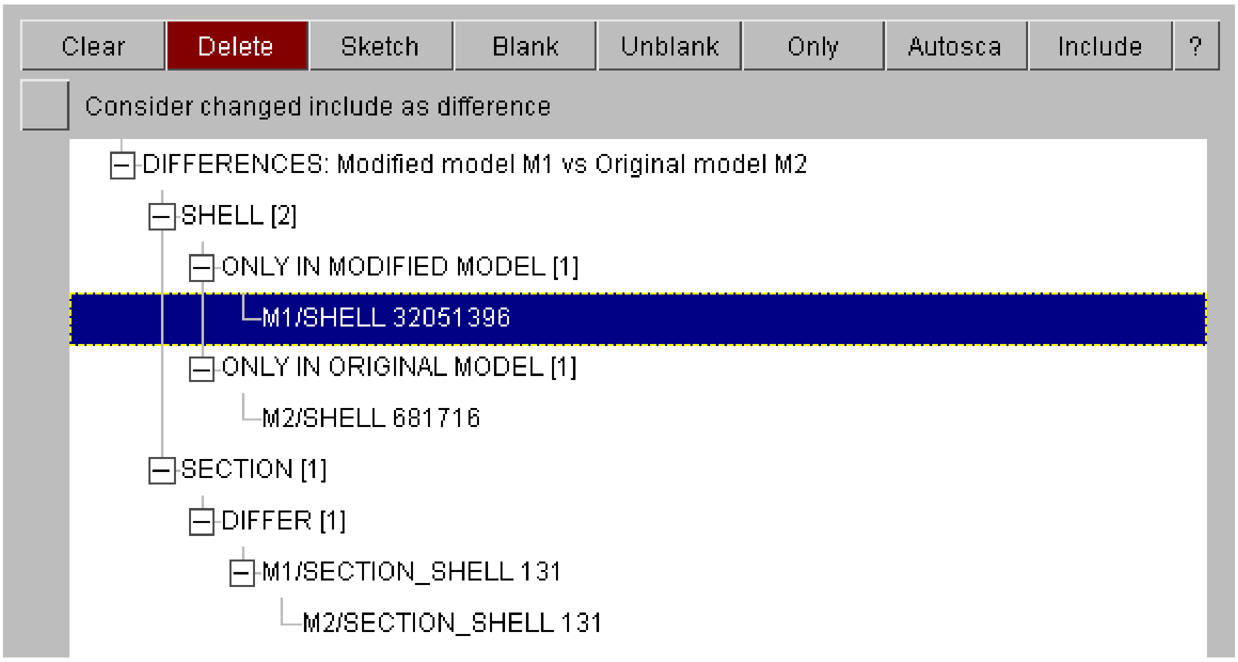


This function (in compare to original mode) is also available from the **Find modified** button in the **Include** tree (if >1model in memory use the drop-down off the model tag) or when selecting include files to write keyword files. Any include files which have changed are highlighted so they can be written out.

3.16.1 Comparing one model to another model

Comparing two models in memory is the preferred method as all items concerned (including those deleted from the modified model) are available for interrogation (and possible manipulation) on the modified tree.

In this example a shell has been created, a shell has been deleted and a section card has been changed.

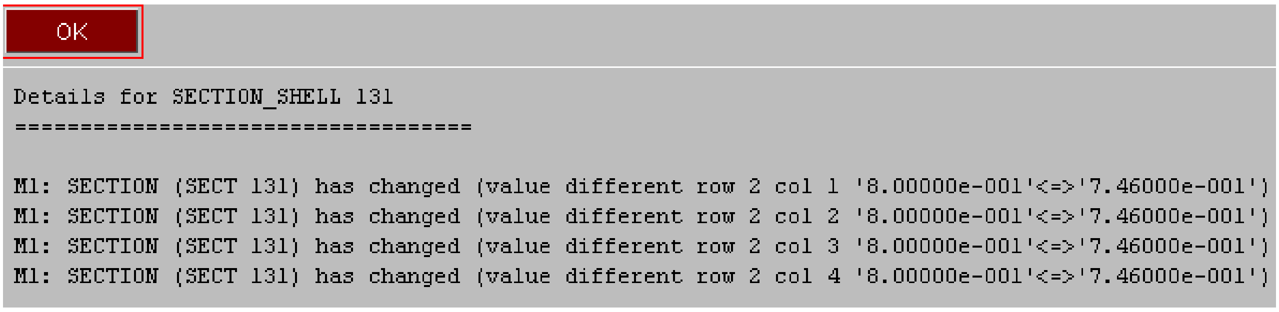


Options for sketch, edit, blank, etc. are available from the drop-down.

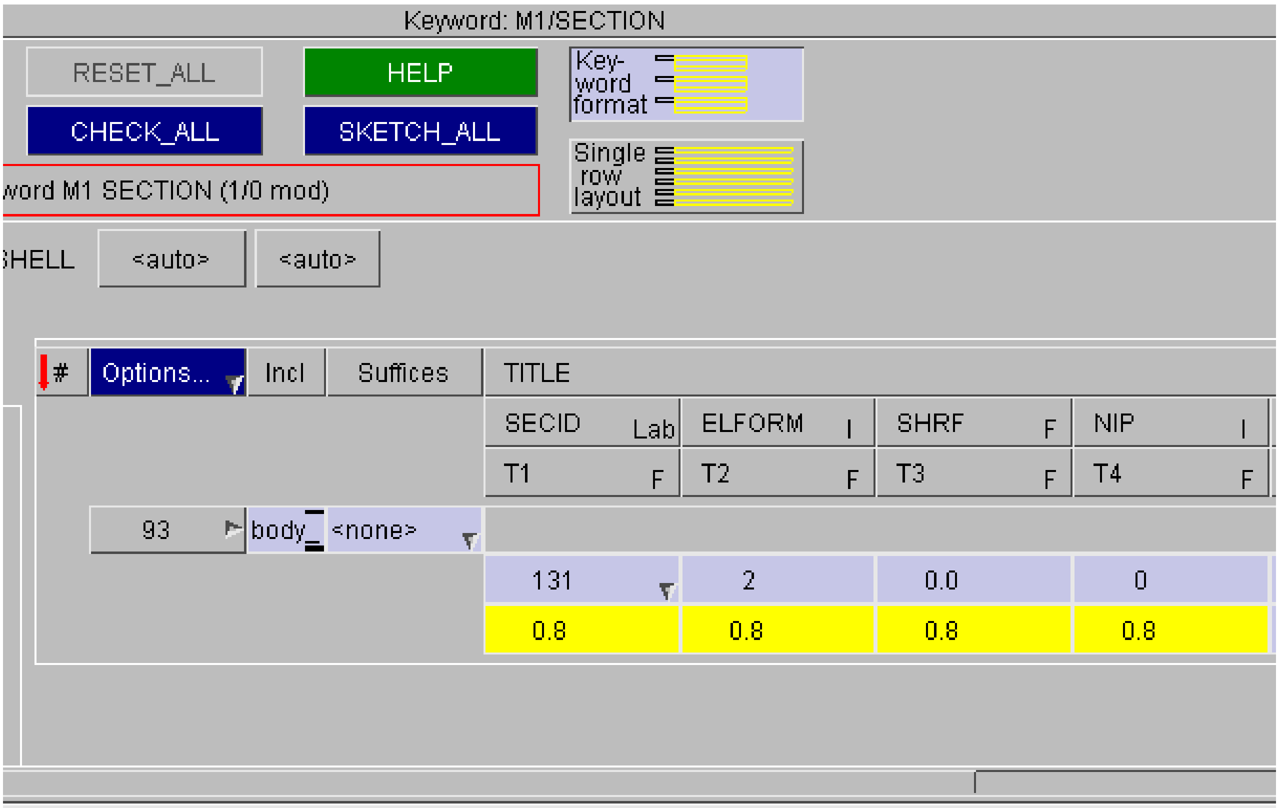
The drop-down from the created shell(s) in the modified model gives the option to **Delete** or put the shells into a set - **make_SET_SHELL**

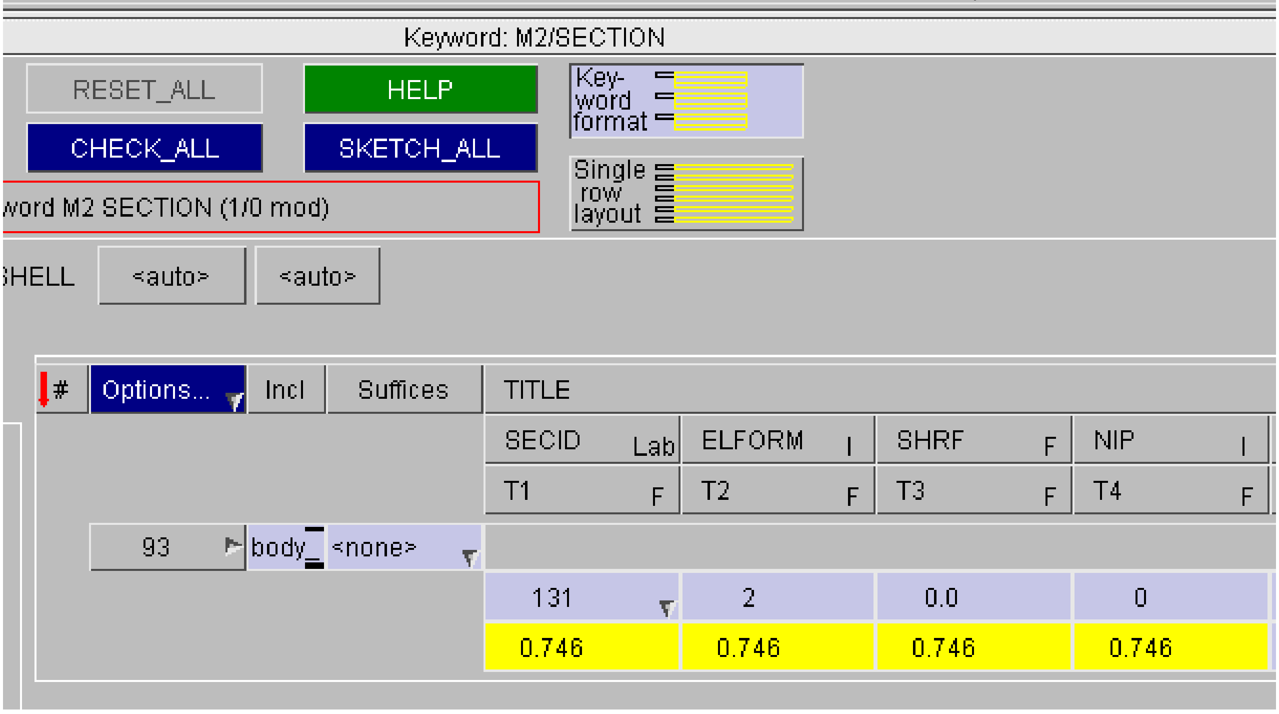
The drop-down from the shell(s) in the original model, deleted from the modified model gives the option to copy them back in **Copy_M2->M1**

For the modified section card, the **Details** drop-down shows what has changed.

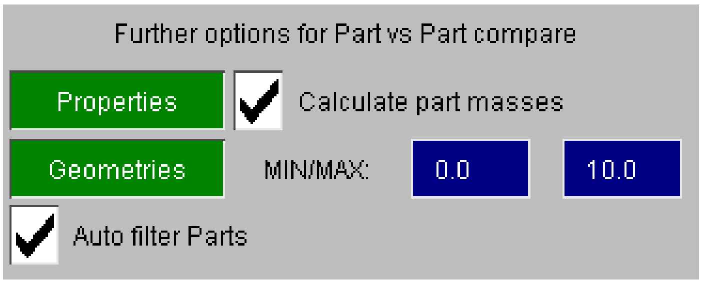


The keyword editor may also be invoked in a special mode which will highlight the changes.





Additional options for parts

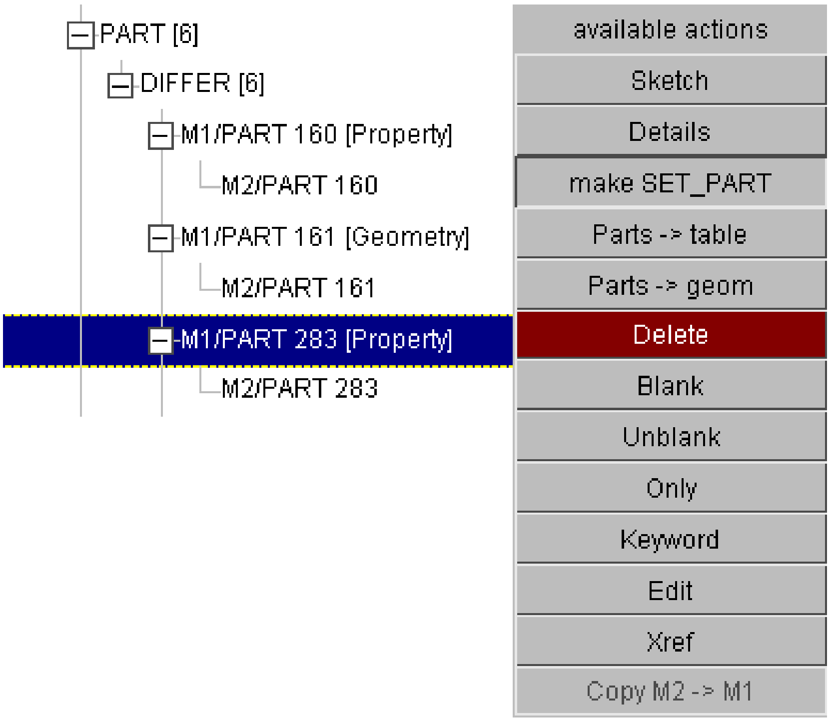


Additionally with this method part properties and geometries may be compared by using the part compare function, see section 7.3.2

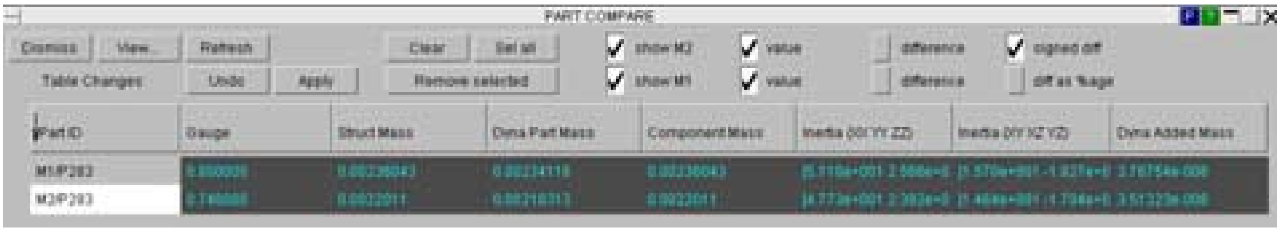
Properties If active, all properties available on part table (mass properties can be switched off) will be calculated for each pair of matched parts. Any parts for which properties differ will be reported on the tree.

Geometries This function will run a contact type check to detect gaps (using defined min/max values) between matched parts (of type shell only). The option **Auto filter parts** is recommended to block the test (which can take a few secs) for part pairs which are unlikely to be geometrically different (same element count, same geometric CofG and same surface area).

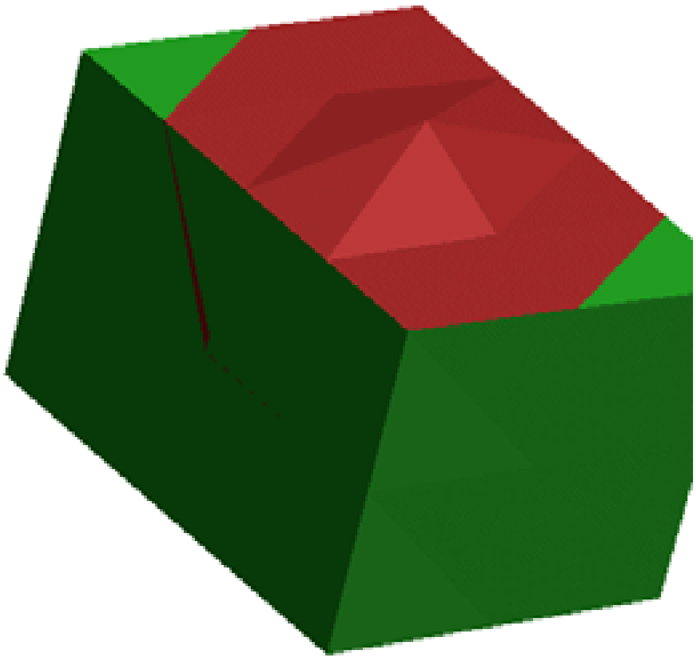
These options enable the user to readily identify parts which have been changed as result of change to another keyword, such as *SECTION or *NODE.



For parts with property differences, the **Parts -> table** function will give a detailed description.

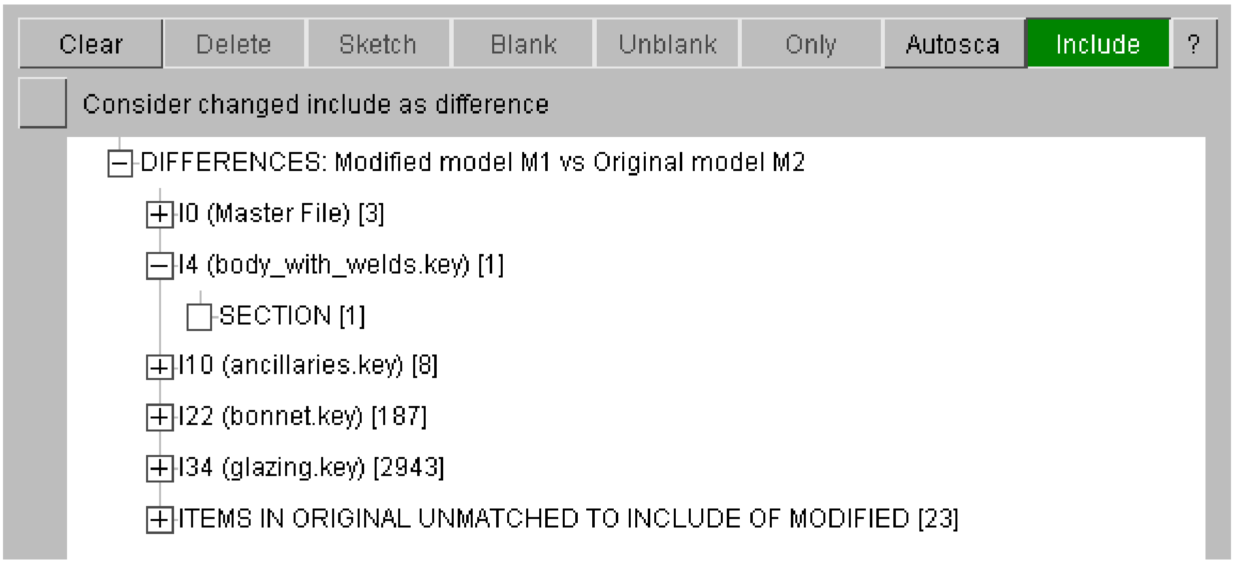


For parts with geometric differences, the **Parts -> geom** function will invoke a display where the difference can be contoured. See section 7.3.3. This must be dismissed to return to the tree viewer.



Displaying modified items by include

The modified items may be displayed under their include file if the **include** button is activated. This will be the list of includes derived from the modified file, so creation of include files presents no difficulty. If, however, an include has been deleted (or renamed) items will appear under the heading **ITEMS IN ORIGINAL MODEL UNMATCHED TO INCLUDE OF MODIFIED**.



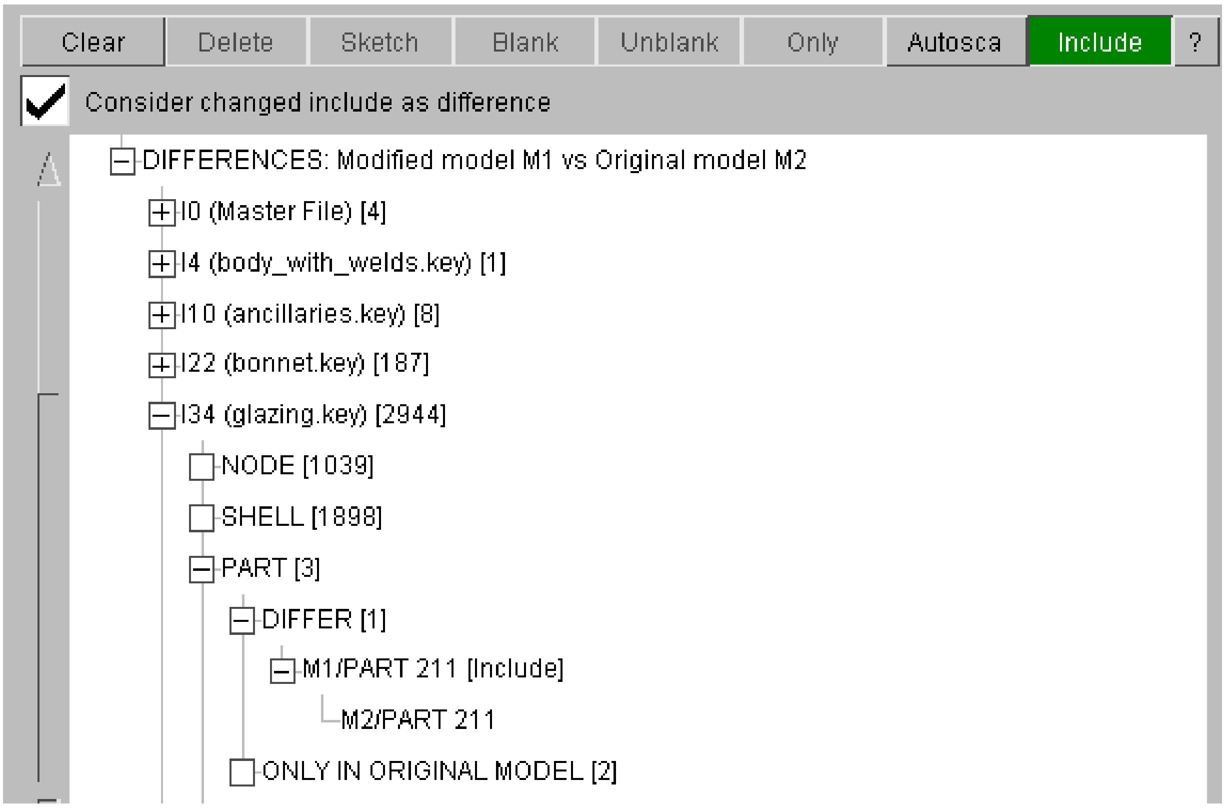
Item has changed include

By default Primer does not treat a change of include as a difference, so items which are the same but have simply been moved to another include will not appear on the tree.

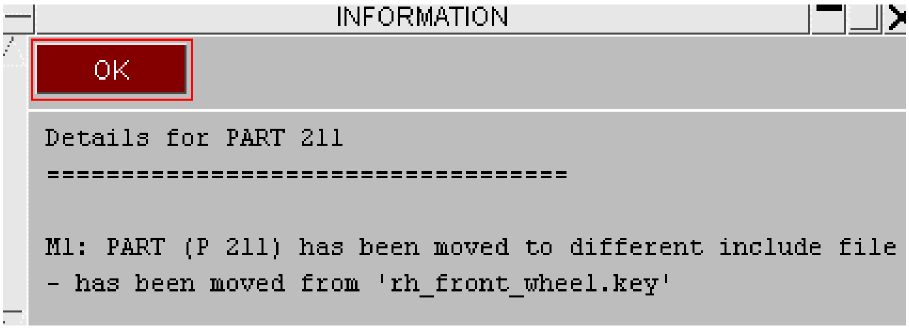
Thus a model with no includes may be reported as identical to one split into many includes.

By activating the option Consider changed include as difference, the tree view will show them (in addition to the changed items).

Primer matches includes across models by using the name, the order in which they have been read is irrelevant. Only if a model contains multiple includes of the same name (these will be *INCLUDE_TRANSFORM) will the order be significant and a change of order between the models may give rise to spurious reports of include difference.



The Details drop-down will give the include from which the item has been moved.



3.16.2 Comparing a model to original or to a file

To compare a model to the original version of the file (i.e. when you read the file into PRIMER) select **Original** or **File**.

PRIMER will reread the original model/selected file into the next free model and then compare the 2 models.

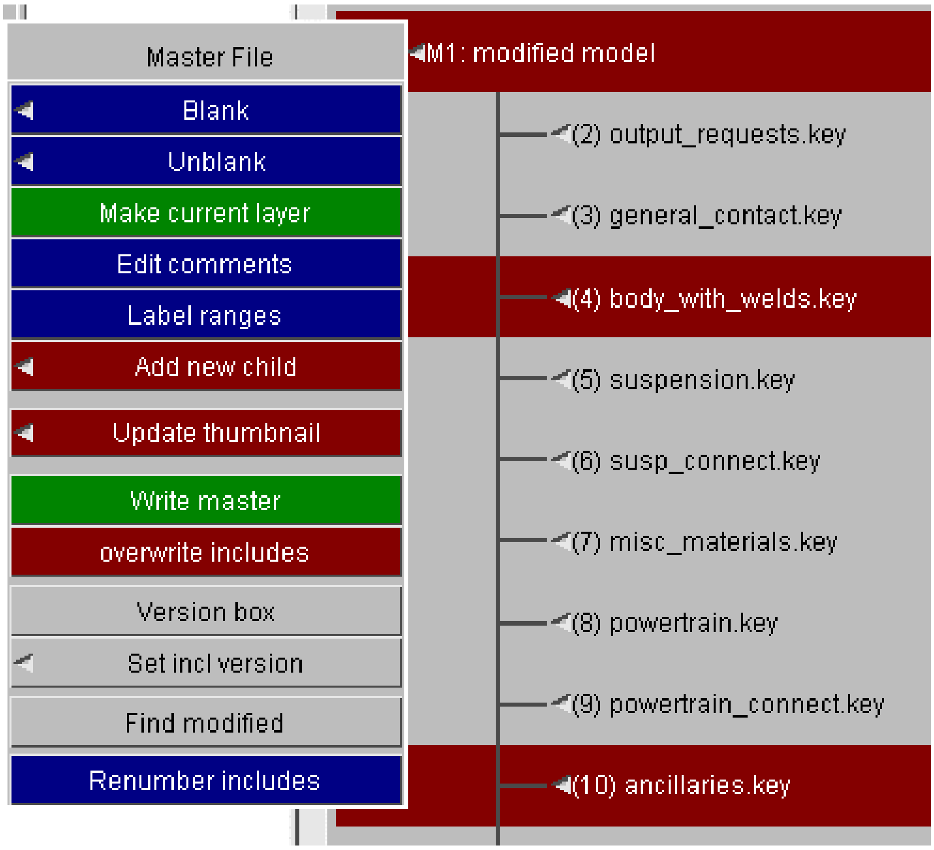
As in compare to file, the 2nd model is not plotted and deleted as soon as the comparison has completed. This means that the further options for part comparison are not available and deleted items can only be reported (as opposed to interrogated) on the modified tree. Otherwise the process is much the same as model vs model comparison.

3.16.3 Comparing individual include files

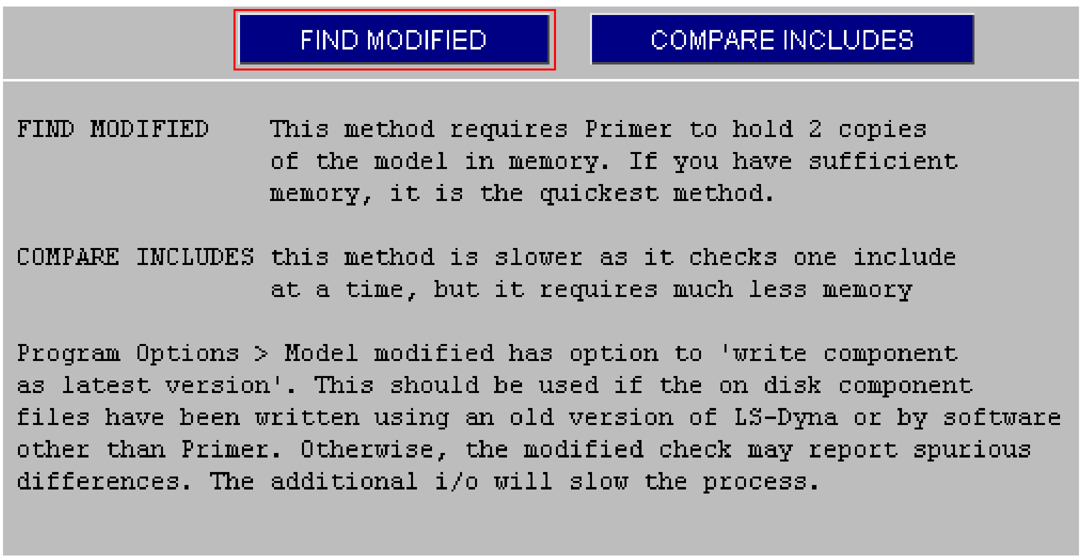
Find modified described above works by reading the original file and comparing it to the current model in memory. This methodology cannot be applied to models which have been **built in Primer**. We would need to record how each model is built so we could repeat the process to construct the original - this is not practical.

Find modified also requires that we hold 2 copies of the model in memory which may not be possible for very large models on machines with limited amount of memory.

Instead, we have an alternate function available off the model drop-down on the include tree. Note - the **Find modified** button always runs the normal function.



You will be given the option of running the normal **Find Modified** or **Compare Include**.

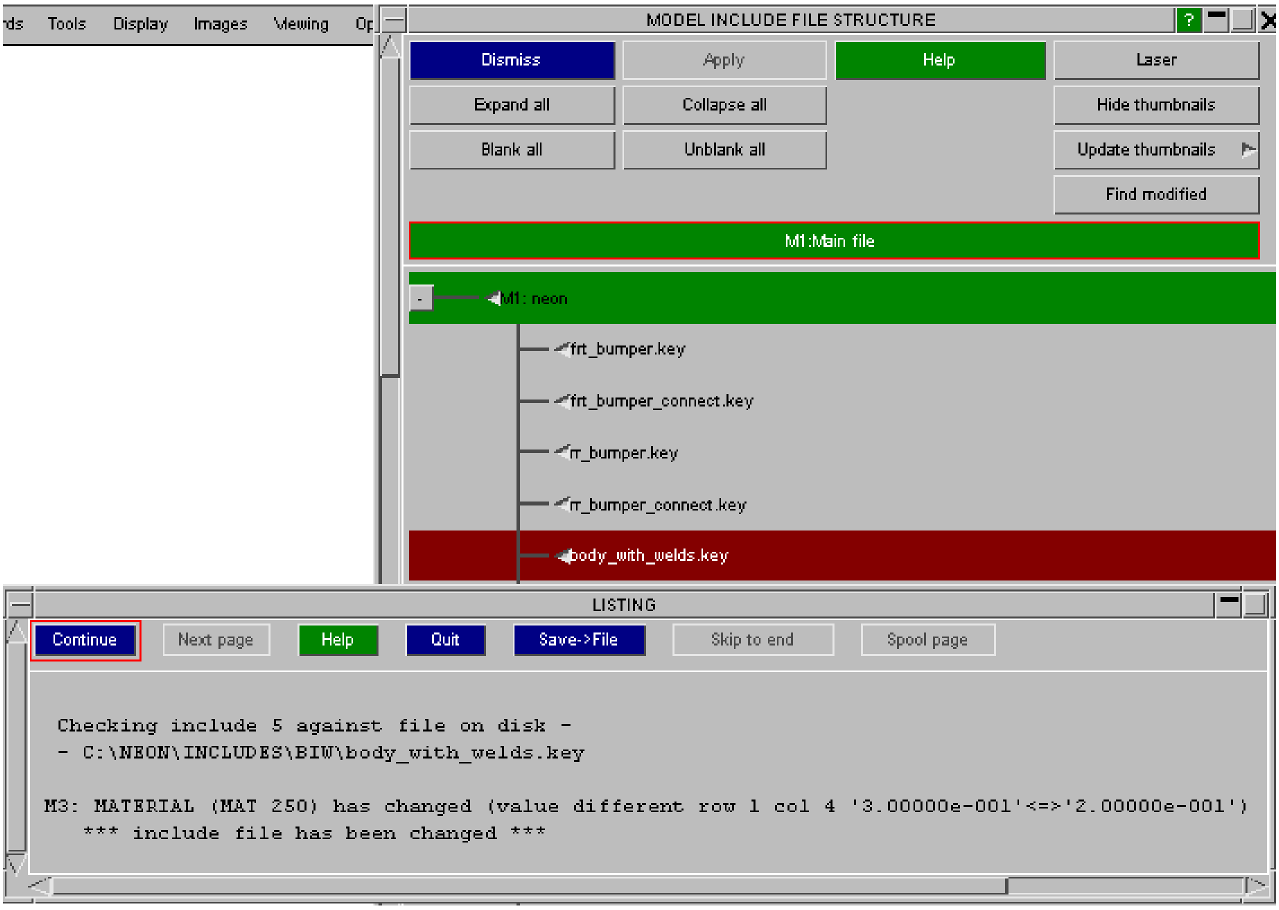


Additionally, **Compare include** may be run for an individual include (off the include drop-down).

Compare Include works by writing the current include to scratch area and reads it back in to form model A and then reads the original include file to form model B. Models A and B are then compared using the model modified function which effectively performs a dxdiff between them and reports any differences.

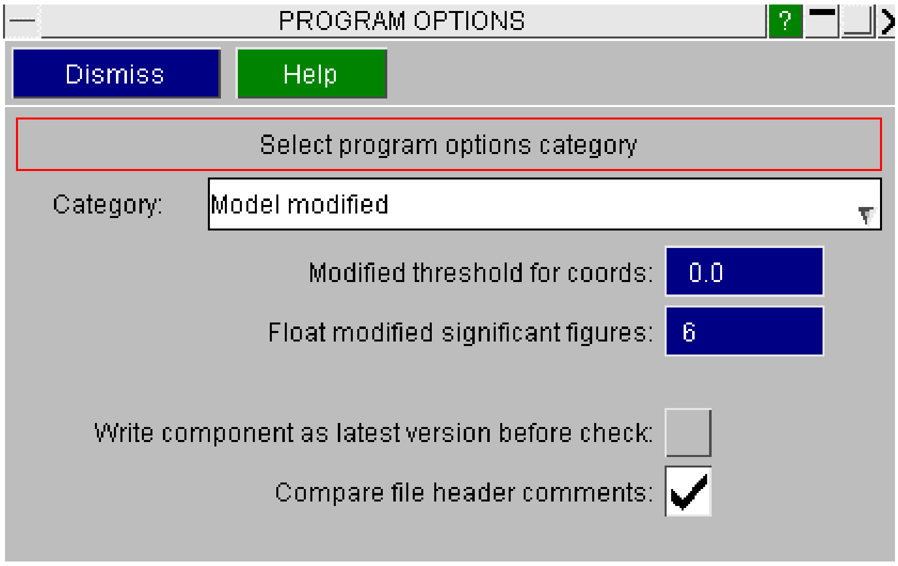
The function involves intensive disk I/O and so if run on all the includes of a large model may take a while to complete. The original model modified function will achieve the same result much more quickly (when it can be applied) so is recommended.

For all models this function is very useful for interrogating an individual include to see the details of what has changed. With a non-built model you can use **Compare include** to detect which include files have changed, and then extract the details for the include of interest using compare to disk. Although they work in different ways, both processes should always report the same differences.



3.16.4 Options for comparing models

The Options button on the modified panel will give direct access to the appropriate program options panel.



Comparing floating point values

By default, floating point numbers are compared to 6 sig fig. This can be reduced at the user's discretion and will affect the comparison of all floating point numbers.

Coordinates (on nodes, connections & airbag reference geometry) are special cases which admit of an absolute difference threshold which can be set by the user. This is particularly useful to remove the spurious differences when comparing models with *INCLUDE_TRANSFORM. These arise due to rounding when the transforms are applied and unapplied.

Avoiding spurious differences

Write component as latest version before check This option applies only when we are comparing include files. If this option is set Primer will read the component file, write it using the latest output version and then re-read it. The I/O overhead is considerable. If the component files are rather old or have been written by software other than Primer this may be worth doing to avoid spurious difference reports. If the component file has been written from Primer relatively recently there is no need to do this, hence the default if off.

File header comments

By default Primer will compare file header comments for the master file and the includes at the beginning of the model modified process and warn if these appear to differ. This includes a check on include file label range definitions. You may switch this off.

4 Model visualisation

4.0 Visualisation and labelling.

This section describes how to draw models, control what is drawn, and also add labels and associated data to plots. Viewing control is covered in section 9.

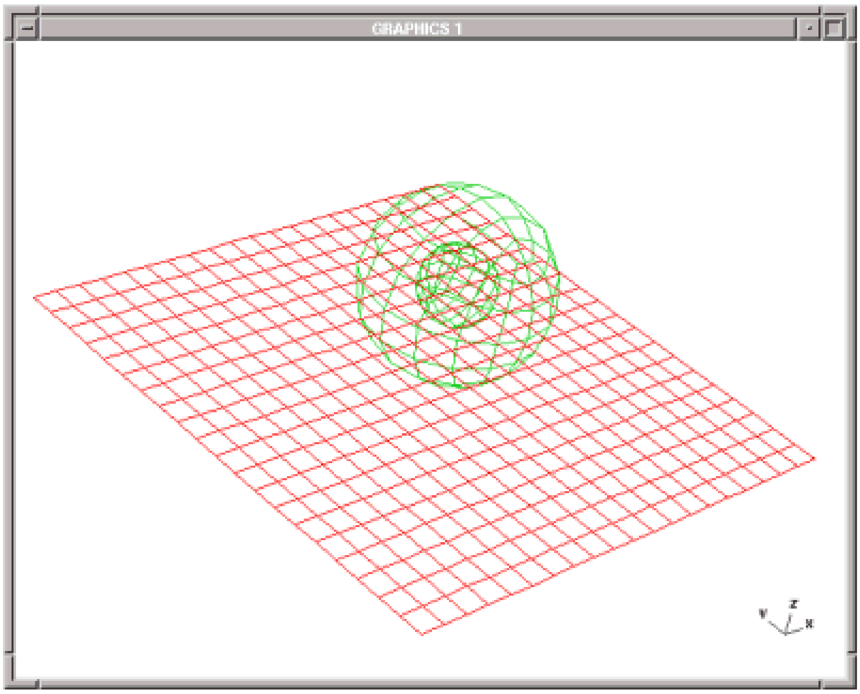
4.1 Basic drawing commands: LI(ne), HI(dden line), SH(aded image)

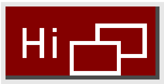
PRIMER is capable of drawing basic model geometry in three modes: "**LI**ne", "**HI**dden-line" and "**SH**aded".

"LINE" mode (**LI**) draws all element borders with no hidden surface removal. However the back faces of 3D elements are removed when graphics are in 2D mode (but not in 3D mode).

This figure shows an example of a line mode plot of a ball (made of solids) above a flat plate.

Note that no hidden surface removal has been carried out.

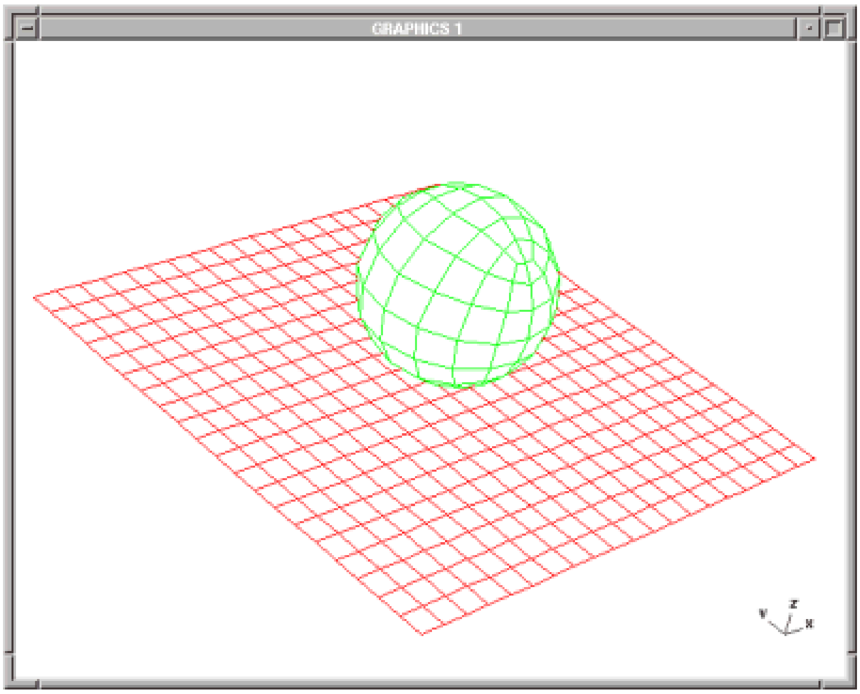


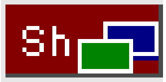


"Hidden-line" mode (**HI**) also draws element borders, but this time with hidden surfaces removed. (Back face removal is implicit in this.)
This figure shows an example of a hidden-line plot, with the same model

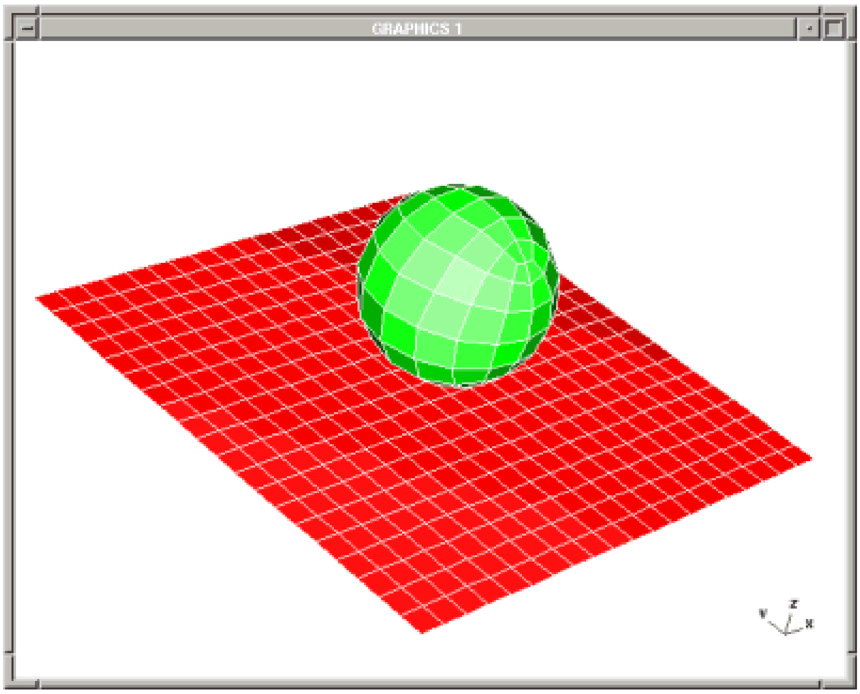
as above.
It is now obvious that the hidden surfaces and lines have been removed, and it is easy to tell that the ball lies above the plane.

Hidden surface removal requires more computation than a simple line mode plot, so it will be slower to generate. Most displays with 3D graphics protocols have a hardware "Z-buffer" which makes this process faster, but even so complex images may take an appreciable time to draw. For this reason the dynamic viewing modes, which permit real-time manipulation of the view, have a facility to drop back to line mode (or even "free-edge" mode) while the image is being moved, reverting to the normal display mode when the motion is complete. See section 9.4 for a description of how to do this.





"Shaded" (**SH**) mode also performs hidden-surface removal, but this time the surfaces of 2D and 3D elements are shaded and lit in the appropriate colours.

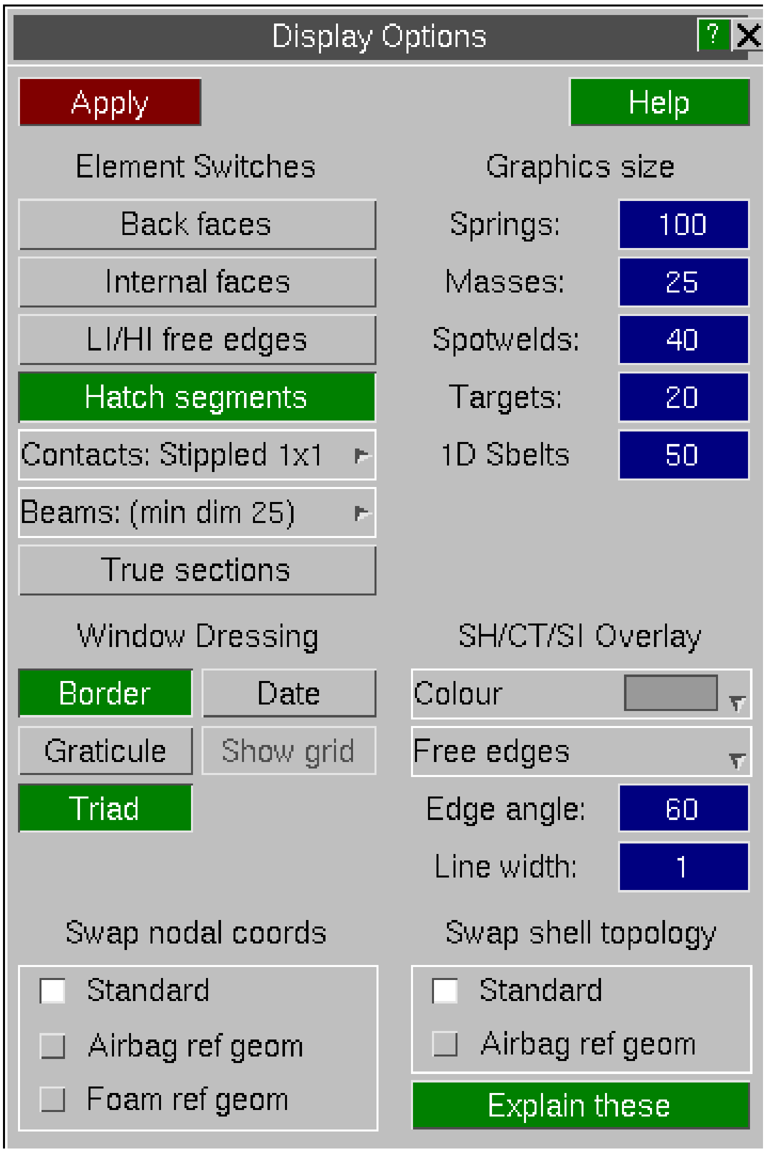


This figure shows an example of a shaded-image plot. Lighting and hidden-surface removal have both been applied, and the element borders have been overlaid on the resulting plot.

As with hidden-line plots, 3D devices with hardware assistance will generally produce these images much faster than the software alone (2D) method, but (in either mode) computation time will be longer and the ability to drop back to line or free-edge mode during dynamic viewing also applies.

Shaded, Line and Hidden plots may also be invoked with the shortcut keys S,L and H.

4.1.1 OPTIONS... Controlling plot parameters.



"Options" gives user control over a number of graphical features.

Some of these can be preset in the "oa_pref" file: see section 4.1.4 below.

Back faces Determines whether or not the back (ie facing away from you) faces of 3D (solid and thick shell) elements are drawn.

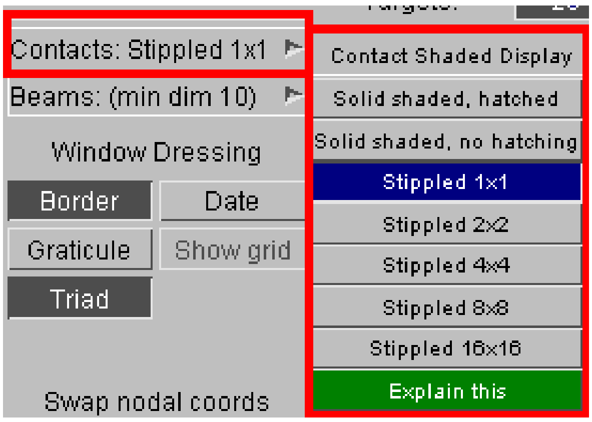
Internal faces Determines whether or not the internal faces of 3D elements are drawn. You should only use this if you need to see them, as it slows down drawing by a large factor.

LI/HI free edges Determines whether or not the internal faces of 3D elements are drawn on **LI**ne and **HI**dden-line plots. You should only use this if you need to see them, as it slows down drawing by a large factor.

The overlay on **SH**aded and data bearing plots is controlled in **SH/CT/SI Overlay**

Hatch segments Controls how segment sets are displayed. The default is to draw a "hatched" wireframe overlay on them in order to distinguish them from ordinary 2D elements, but this can slow down graphics considerably on some platforms so it is switchable.

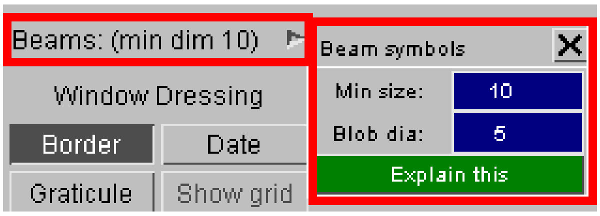
Contacts Historically PRIMER drew contacts using a hatched wireframe overlay but, as with segments above, this could be slow. Therefore the default in V9.3 has been changed to "**Stippled 1x1**" which is fast, and distinguishes them visually from shells by giving them a semi-transparent appearance since every other pixel is omitted.



This is much faster than hatching, but may not be to the taste of all users, so a range of options is given as shown here. **Solid shaded, hatched** will give the original (slow) appearance, and **Solid shaded, no hatching** will give an opaque result visually indistinguishable from shells.

The default appearance can be changed in the "oa_pref" file using

```
primer*contact_shaded_display:
```
<option>



Beams

PRIMER draws ordinary (not spotweld) beams as simple lines between the 2 nodes, but when these nodes get very close together the result can be a very small point of a single pixel which is almost impossible to see.

Therefore when the distance between the 2 nodes as drawn on the screen is less than **Min size** the symbol is changed to "blobs" on each node of **Blob dia** to make them easier to see. (Both these dimensions are in screen space units.)

With True Sections switched on beams will be drawn with their explicit sections dimensions and orientation.

For beams where only Area, Ixx and Iyy properties are available then a thin-walled rectangular section that matches these properties is synthesised. This should be approximately correct, but obviously it cannot represent I beams or rectangular sections with varying wall thicknesses, but it should give a reasonable representation of beam dimensions. If you use inconsistent or impossible properties you may get some strange looking sections!

Window Dressing

Controls the display of the plot border, and display of the current date.
The "**GRATICULE**" is tick marks around the edge of the plot which show the current window dimensions: useful for estimating distances on the screen (although **MEASURE** provides a more accurate method). If the graticule is on then you can also join up the tick marks with **SHOW GRID**.

Graphics size

Sets the display size of certain dimensionless symbols. (Springs get a "small spiral" symbol when their size gets too small to visualise as their "normal" symbol.)

SH/CT/SI Overlay

The element border overlay for **SH**(aded) plots, and also the contoured **CT** (continuous tone) and **SI** (shaded image) is separately controllable.

Colour	Is one of the standard PRIMER colours selected from the popup menu
Overlay edging mode	Is one of **No overlay**, **Free edges**, **Feature lines** or **All edges**. **Free edges** are defined where an element edge is not connected to any other element of the same type, or where the part ids of the elements at an edge differ. Therefore a topological plot of the boundaries of mesh zones is produced. **Feature lines** are a superset of "free edge" mode in which the angle between adjacent elements is considered. Where this angle is greater than the "Edge angle" defined below then a feature line edge is defined, and this is added (logically ORed) to the free edges. The effect is to give a better idea of the shape of the mesh than is available from free edges alone.
Edge angle	Sets the angle (in degrees) between adjacent element faces at which a "feature line" edge will appear.

Swap nodal coords allows you to swap the nodal coordinates used throughout PRIMER with:

Standard	Reverts to using the normal coordinates defined under the `*NODE` card.
Airbag ref geom	The coordinates of nodes defined under `*AIRBAG_REFERENCE_GEOMETRY`
Foam ref geom	The coordinates of nodes defined under `*INITIAL_FOAM_REFERENCE_GEOMETRY`

This is a straight swap: the values of the nodal coordinates used for all internal PRIMER operations are swapped over, and there are no interlocks or warnings to prevent you misusing this.

If you use this option it is your responsibility to manage it in the appropriate context(s) and to unset it when finished.

However, note that if this option is set when reading a model, PRIMER will automatically unset it first before reading the model to ensure that node data does not get corrupted.

Swap shell topology allows you to swap the shell topologies used throughout PRIMER with:

Standard	Reverts to using the shell topology defined under the `*SHELL` card.
Airbag ref geom	The topology of shells defined under `*AIRBAG_SHELL_REFERENCE_GEOMETRY`

This is a straight swap: the values of the shell topologies used for all internal PRIMER operations are swapped over, and there are no interlocks or warnings to prevent you misusing this.

If you use this option it is your responsibility to manage it in the appropriate context(s) and to unset it when finished.

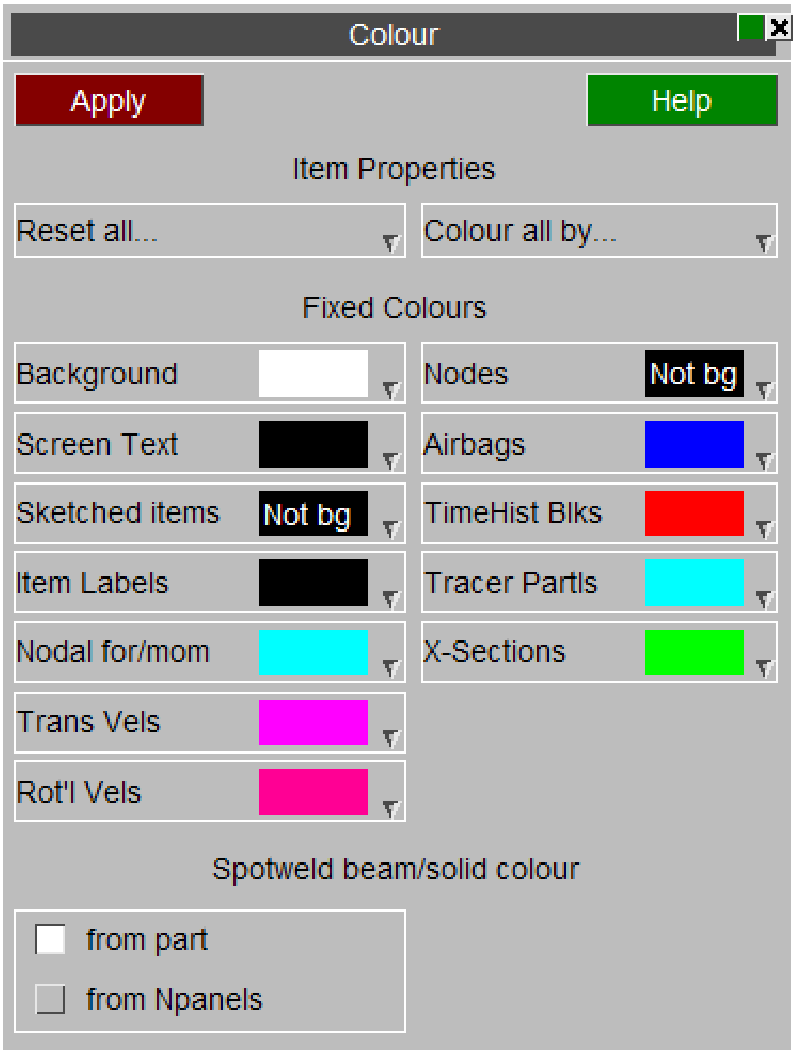
However, note that if this option is set when reading a model, PRIMER will automatically unset it first before reading the model to ensure that shell data does not get corrupted.

4.1.2 COLOUR... Setting item colours in plots.

All options in this panel have popup menus giving a range of colours, with the current selection being shown.

The special colour **Not bg** means "not the background". This is a colour guaranteed to show up well against the current graphics window background, and is the default for text, labels and sketched items.

This colour will change automatically as required if the background colour changes.



Background Sets the background colour of the graphics window. Default: black. This can be configured in the oa_pref file using:

```
primer*background_colour: <colour>
```

Screen Text Sets the colour for title, date, contour bar values, etc. Default: **Not background**. This can be configured in the oa_pref file using:

```
primer*text_colour: <colour>
```

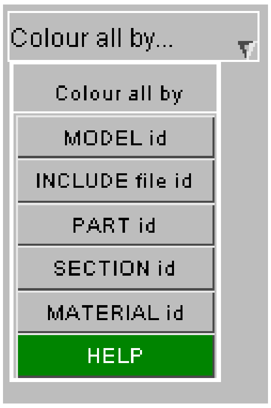
Sketched Items Sets the colour for anything sketched in any context. By default it is set to "**Not background**", the logical opposite of the current background setting so as to establish good contrast. It can also to be set to a fixed colour or **Use Text Colour** which will use the same setting as **Screen Text** is set to.

Item Labels Sets the colour for item (eg node) labels. Default: white.

Colours can also be set for several other types of item in PRIMER using the other popups.

Colour all by gives options of how to colour model items.

- MODEL id sets colour by model number.
- INCLUDE file id sets colour by which include file items belong to.
- PART id sets part-based element colours by part number
- SECTION id sets part-based element colours by section
- MATERIAL id sets part-based element colours by material.



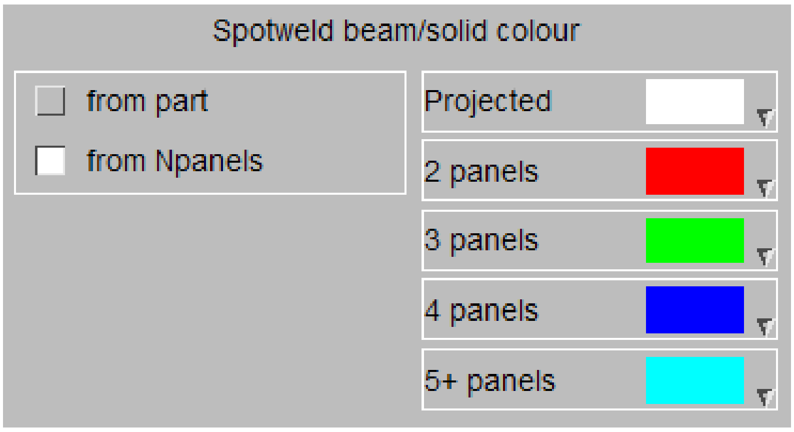
Using colour based on PART, SECTION or MATERIAL id:

For element types that use parts (solid, shell, beam, tk shell, discrete, seat-belt, sph) the colour may be based on one of these properties. The label of the property is used, for example all elements of part 1 will be the same colour. Where a property is undefined, for example no material defined on a *PART card, grey is used.

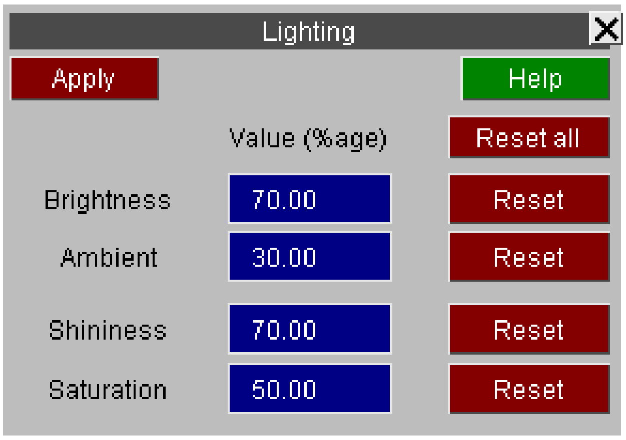
The default is for all such elements to be drawn by **PART** colour.

Setting user-defined colours for individual parts or groups of parts can be achieved using Quick Pick, Part Table or Part Tree. User -defined colours may also be defined for materials, elements and some other entity types using Quick Pick.

Spotweld beam/solid colour allows you to change the colours used when drawing spotweld beams or solids. The default is **from part** in which case the normal colour for the element is used. If **from Npanels** is chosen then the colour of the element will change depending on how many panels the spotweld connects. Popups allow you to change the colours as necessary.



4.1.3 Lighting



The light source position in PRIMER is fixed at approximately the viewer's right shoulder.

Only the attributes shown here can be changed.

Brightness There is a light source located approximately at the observer's right shoulder (this cannot be altered). This field controls how bright this source is. Facets normal to the observer reflect the maximum amount of light from this.

Ambient This control the level of "black-body radiation" which illuminates all facets equally regardless of orientation.

Shininess Low values will give a matt (dull) appearance, high values a shiny one.

Saturation This controls the 'depth' of colour in the range pure colour to grey.

4.1.4 Graphics setup via the "oa_pref" file.

The following parameters can be preset via the preferences file, either by manual editing or using the Preferences Editor. Syntax being:

```
primer*<keyword>: <argument>
```

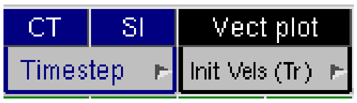
for example:

```
primer*initial_plot_model: SHAD
```

Keyword	**Possible arguments**	**Default value**
plot_border	**ON** or **OFF**	**ON**
overlay_mode	**OFF** or **FREE** or **ALL**	**ALL**
overlay_colour	**WHITE, GREY, BLACK, RED, GREEN, BLUE, CYAN, MAGENTA, YELLOW, ELEMENT**	**WHITE**
background_colour	**WHITE, GREY, BLACK, RED, GREEN, BLUE, CYAN, MAGENTA, YELLOW**	**BLACK**
initial_plot_mode	**LINE** or **HIDDen** or **SHADed**	**LINE**
contour_levels	**1** to **13**	**6**

Further "oa_pref" file options, and details of the interactive preferences editor, are given in Appendix XIII.

4.2 Data Plotting Commands:

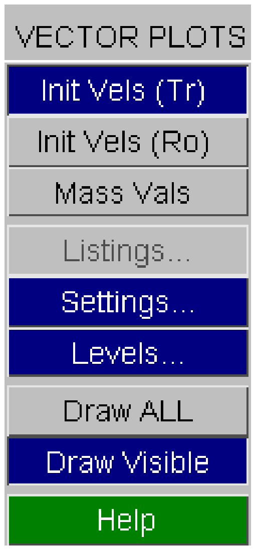


VEC(tor)
CT (continuous tone)
SI (shaded image)

Each command has a popup menu that gives some or all of the following options:

Data component:	Timestep, Shell thickness, etc.
Listings...	Written output of displayed data. For example lists of element timesteps sorted into ascending order.
Settings...	Unique panels for each data component that control what is drawn and how it is displayed.
Levels...	Control and display of the number of contour levels.

4.2.1 Vector plots.

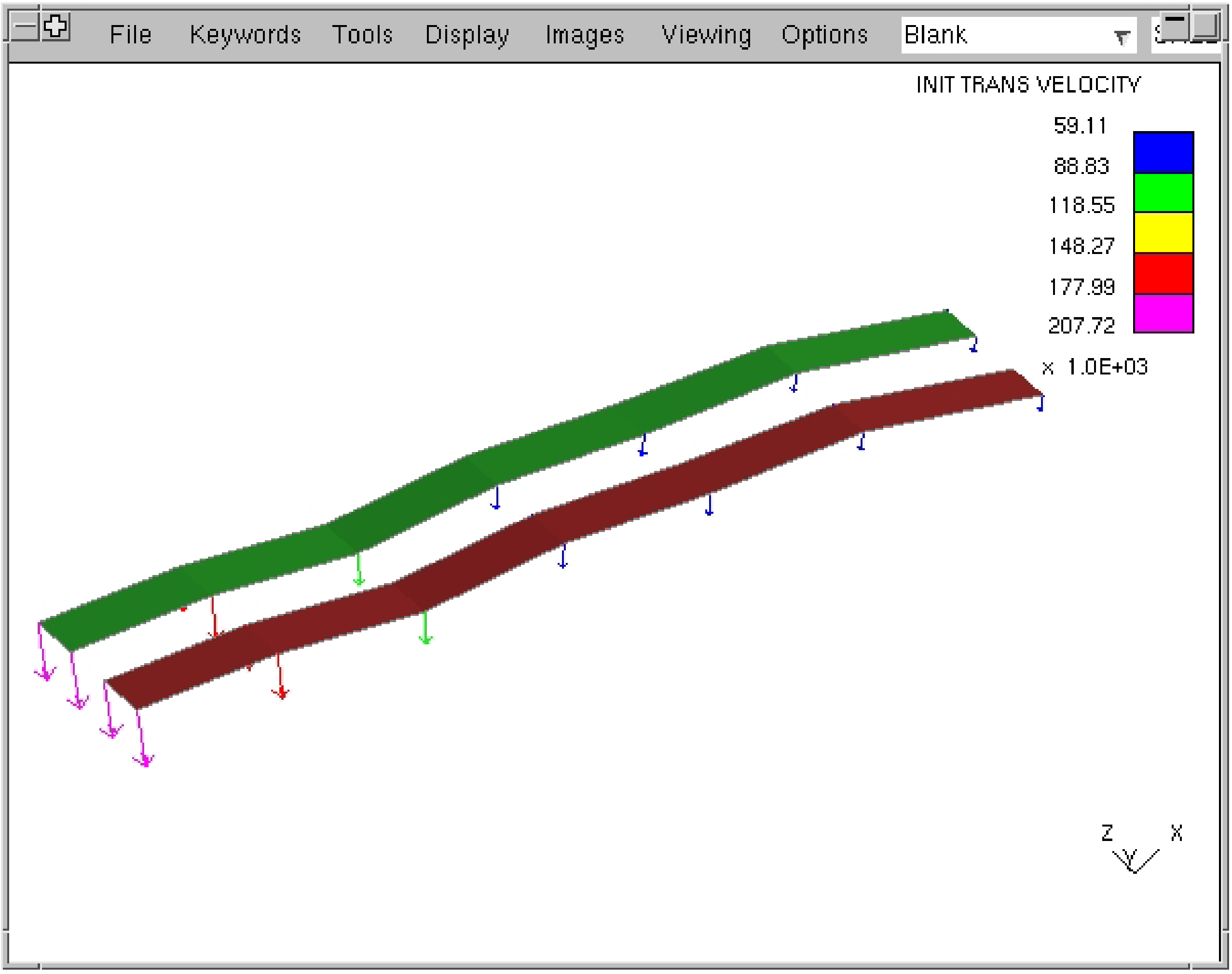


Vector plots superimpose nodal data on the current image display mode (**LI, HI** or **SH**). To display these data on a different mode draw it first, then repeat the **Vect plot** command.

At present vector plotting is only available for:

Init Vels (Tr)	Vectors of initial translational velocity
Init Vels (Ro)	Vectors of initial rotational velocity
Mass Vals	Lumped masses plotted by mass

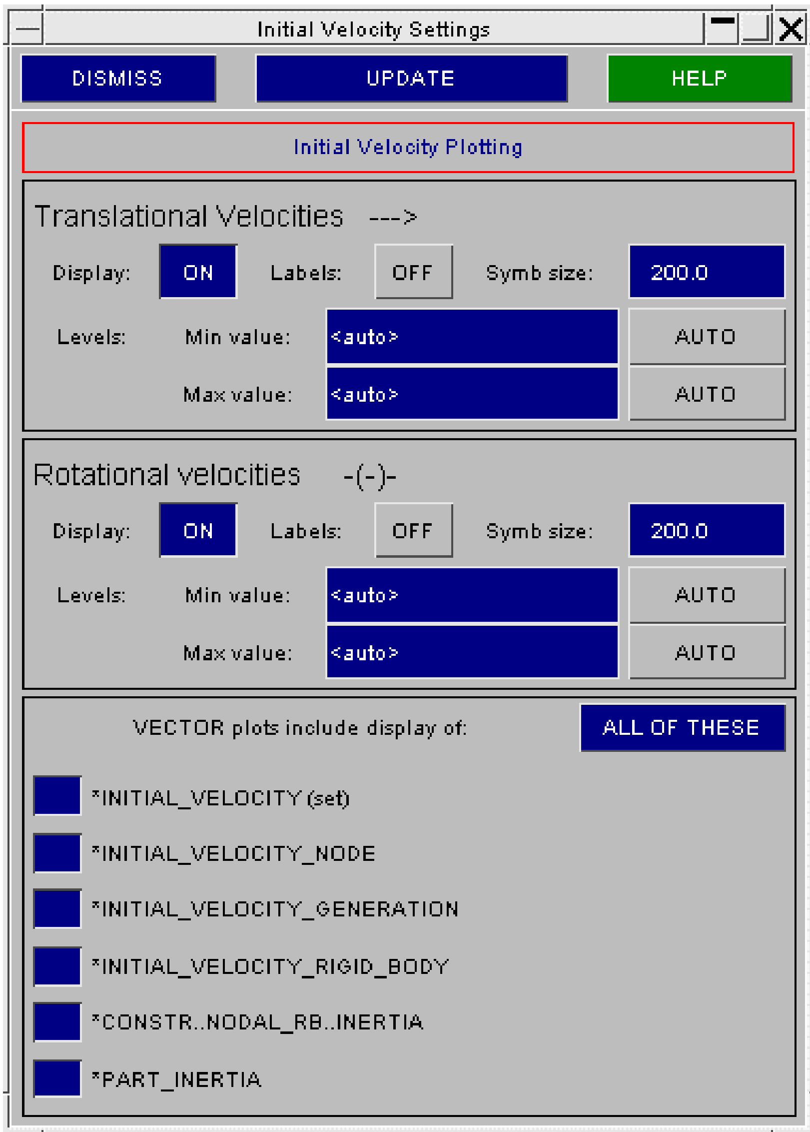
Vector plots of Initial Velocities.



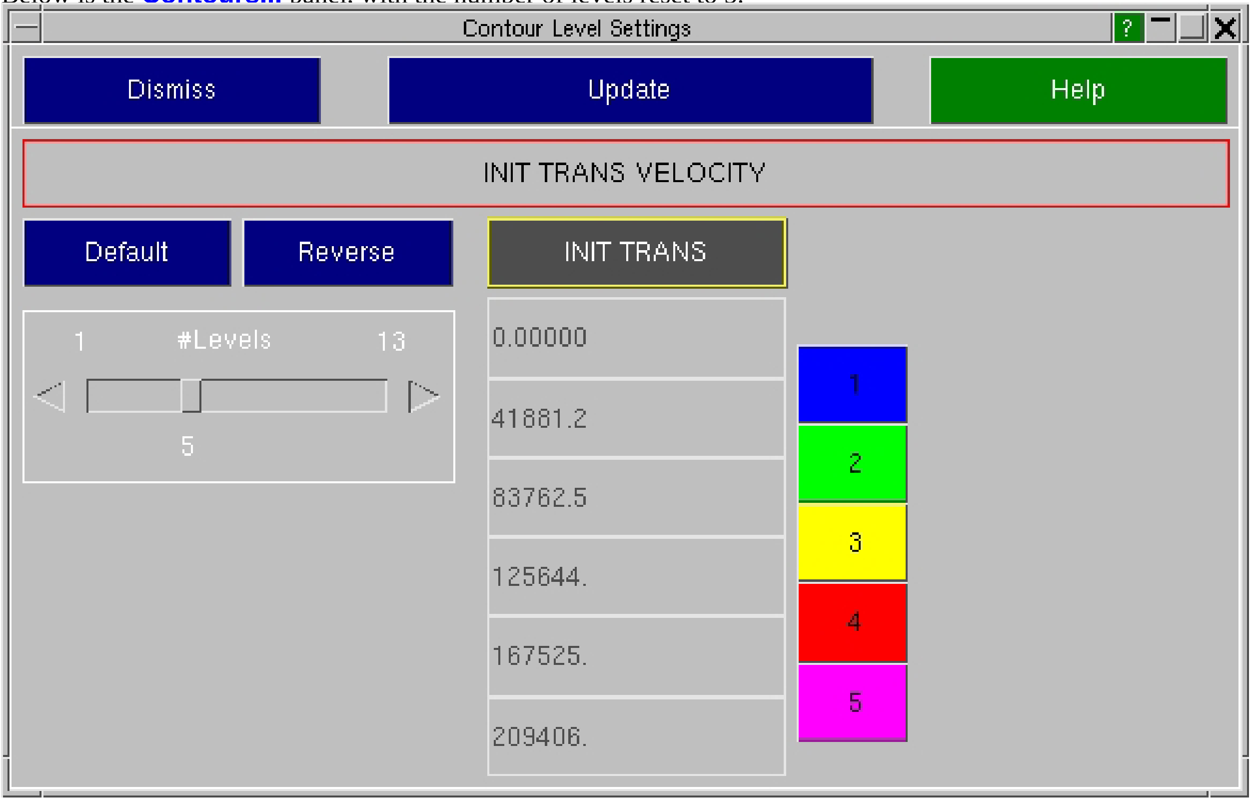
The figure above shows a typical plot of initial velocities for a simple structure. To the right is the **Settings...** panel for this plotting mode.

Note that:

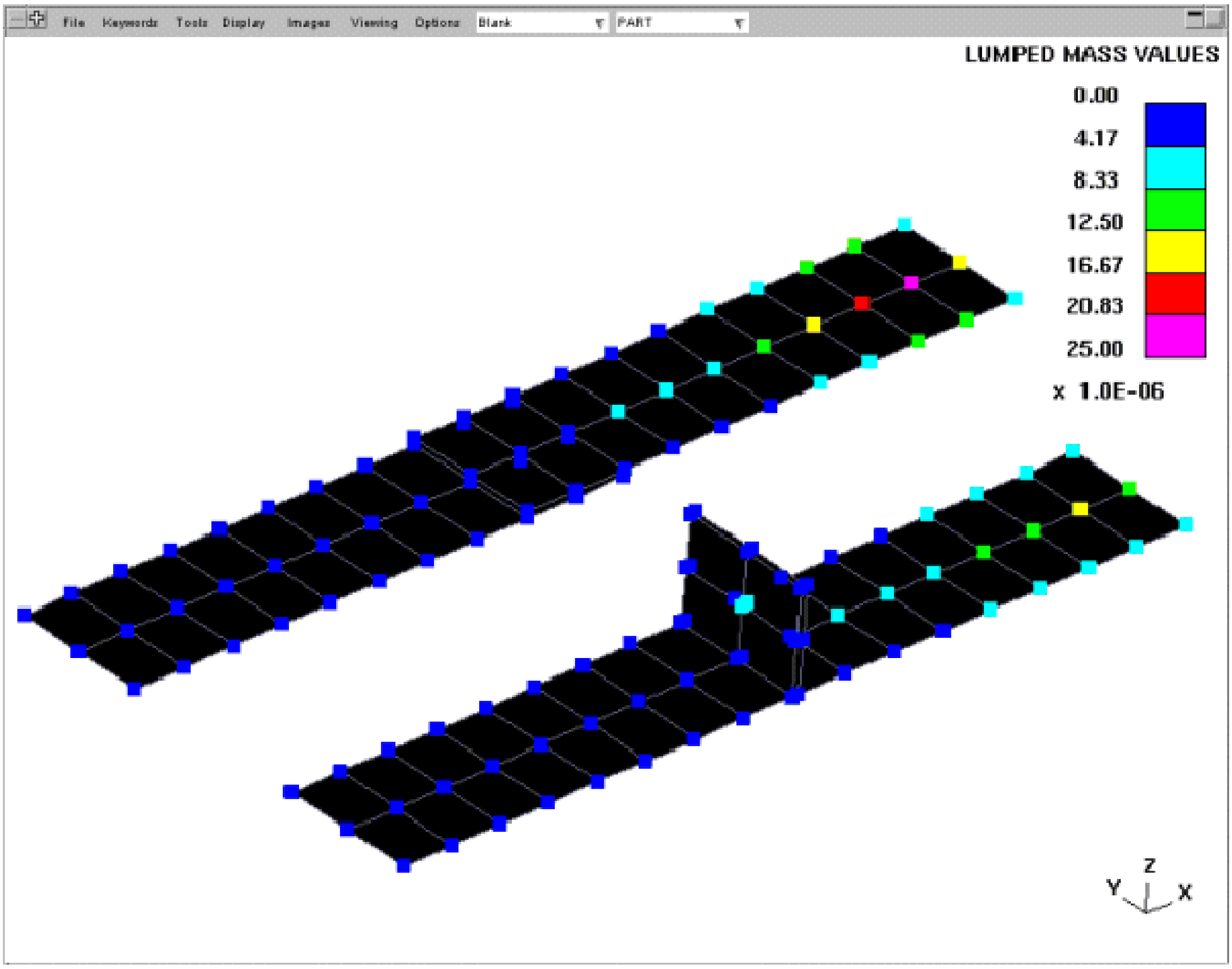
- Initial velocities in LS-DYNA can arise from five different definition methods. Display of each of these is independently switchable.
- Both translational and rotational initial velocities can be plotted, but as separate plots.
- The default contour bounds are automatic, but you may set any range you wish.


Initial Velocity Settings
DISMISS
UPDATE
HELP
Initial Velocity Plotting
Translational Velocities --->
Display: ON
Labels: OFF
Symb size: 200.0
Levels: Min value: <auto> AUTO
Max value: <auto> AUTO
Rotational velocities -(-)-
Display: ON
Labels: OFF
Symb size: 200.0
Levels: Min value: <auto> AUTO
Max value: <auto> AUTO
VECTOR plots include display of:
ALL OF THESE
*INITIAL_VELOCITY (set)
*INITIAL_VELOCITY_NODE
*INITIAL_VELOCITY_GENERATION
*INITIAL_VELOCITY_RIGID_BODY
*CONSTR..NODAL_RB..INERTIA
*PART_INERTIA

Below is the **Contours...** panel, with the number of levels reset to 5.



Vector plots of Lumped Mass values



This figure shows a "Vector" plot of lumped mass values for the structure above. (Following an "Assign Mass" operation to shift the centre of gravity in the +ve X direction, hence the concentration of mass towards the right.)

4.2.2 CT and SI plots.

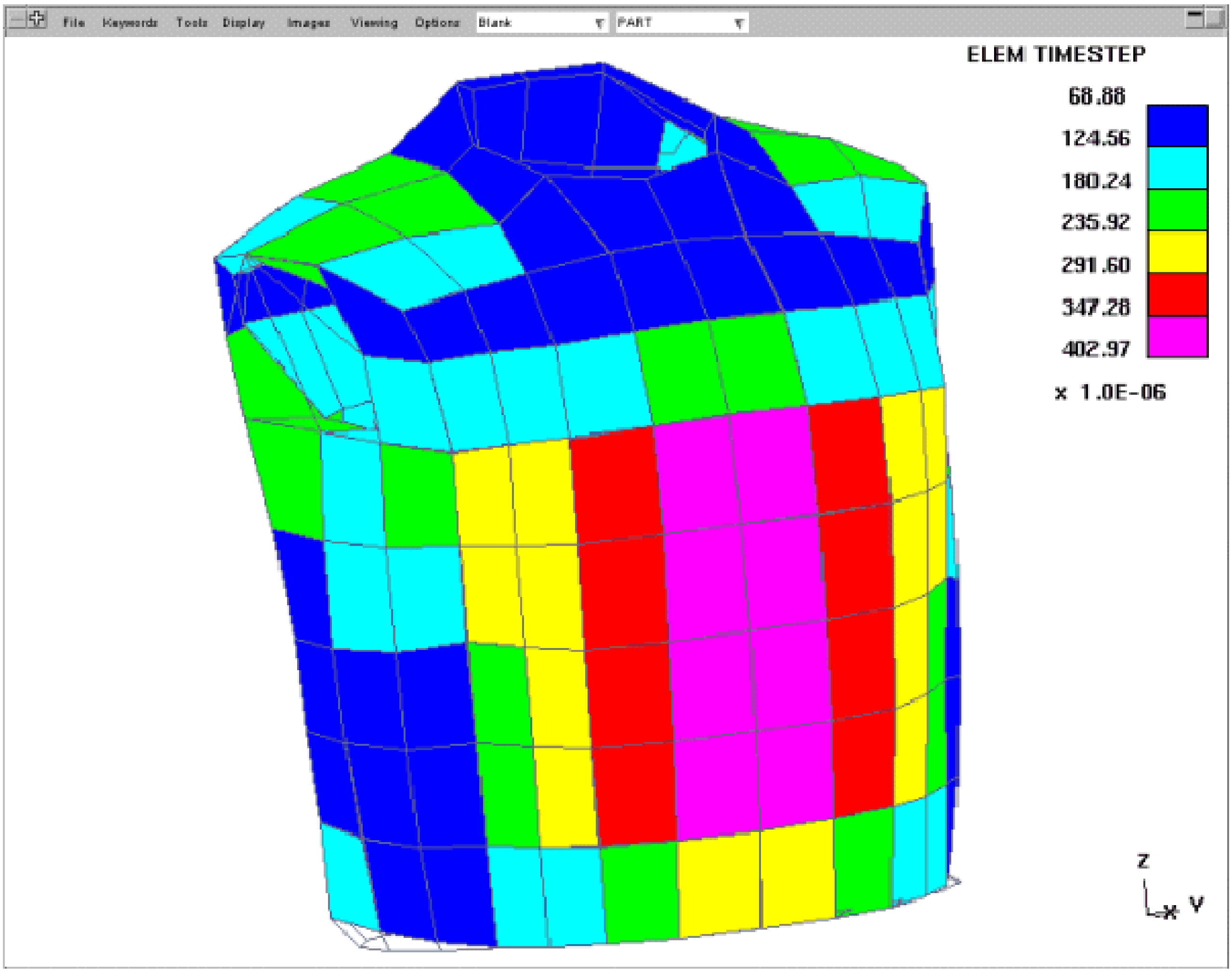
CT (continuous tone) and **SI** (shaded image) plotting modes both display the same data, but the former is unlit whereas the latter is shaded.

Both modes are used primarily to display data for 2D and 3D elements, so the underlying plotting mode is always "hidden surface with fill".

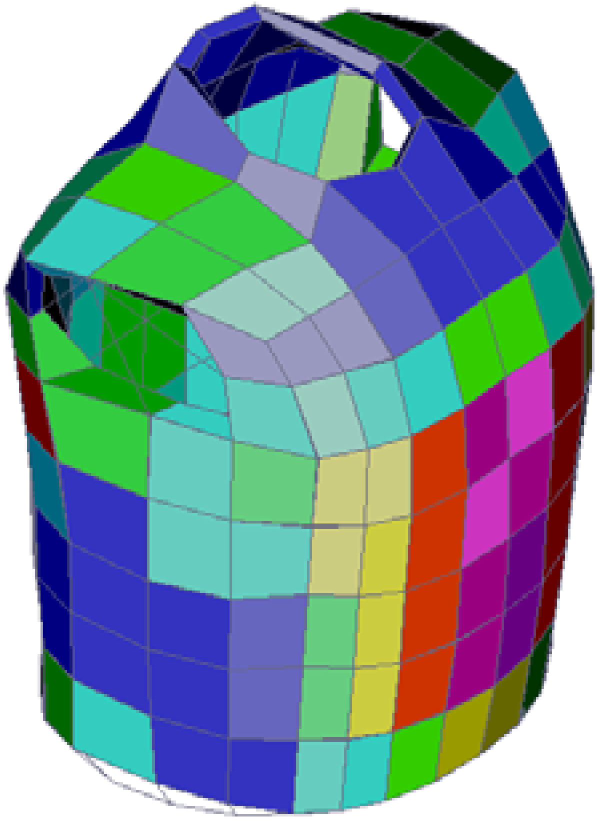
Current data components available are:

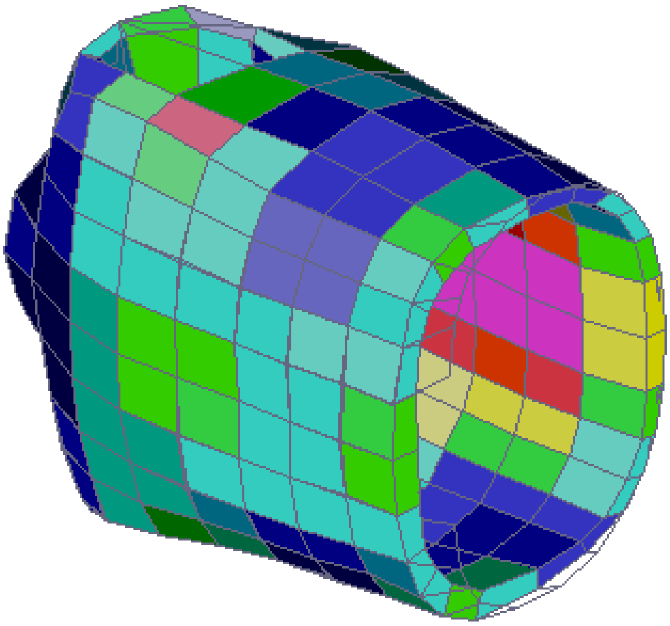
Timestep	Contours of timestep size in elements.
Shell Thk	Contours of (thin) shell thickness.
Mass Scale	Contours of mass added during mass scaling, both by elements and by parts.
Matl Props	Contours of Density, Yield stress, Poisson's ratio & Young's modulus.
Shell Normals	Contours of Shell normals (AWAY or TOWARDS).
Elem Props	Formulation, #Int points and plastic strain.
Elem Qual	Contour of element quality.
Init Vels	Contours of initial velocity components and resultant initial velocity.

CT plot of element timesteps.

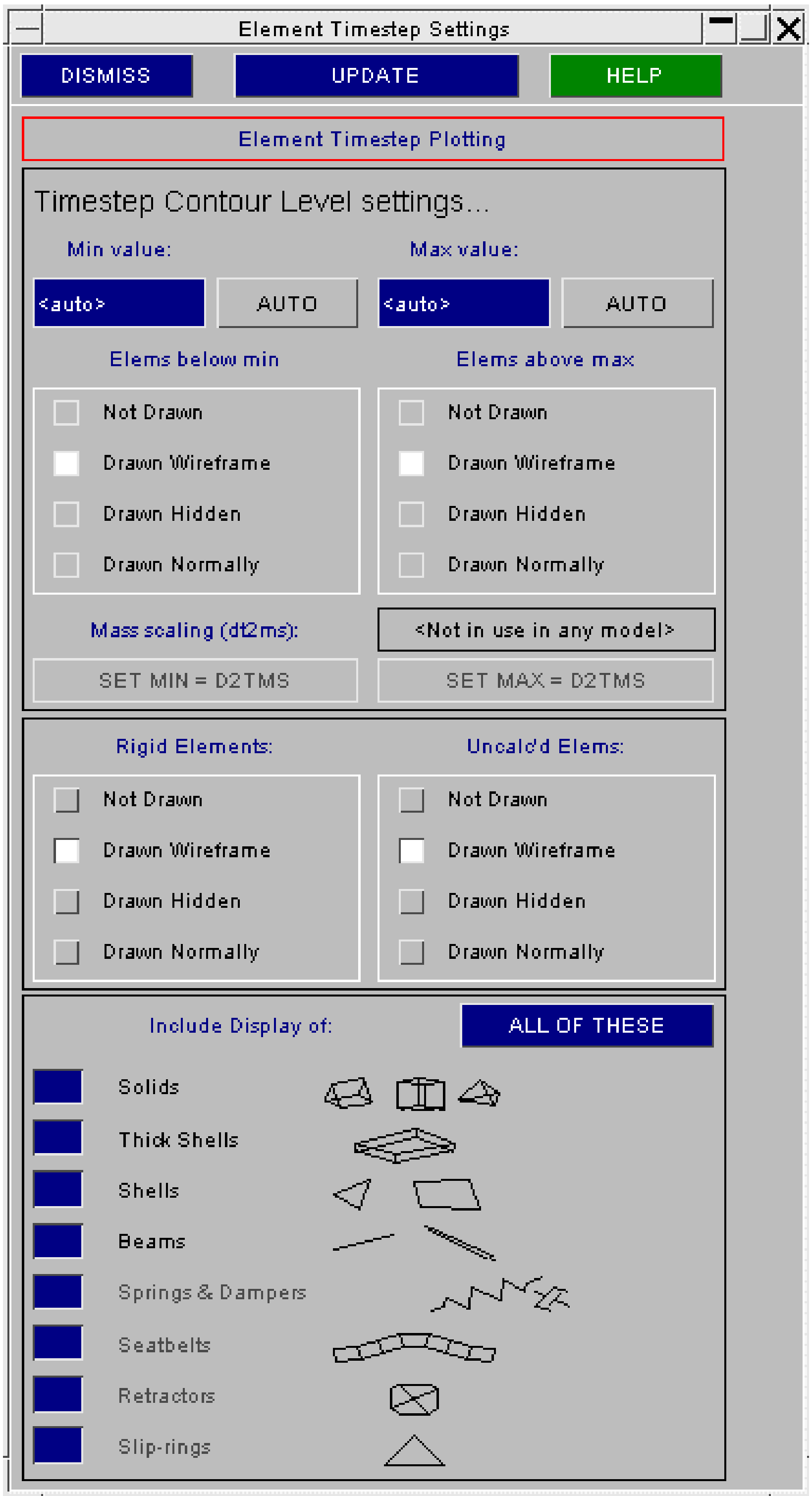


Here is the same image displayed from 2 different angles in **SI** mode, to show how lighting can be integrated with contouring to give a better idea of shape. This is a solid mesh of the torso section from a deformable crash dummy.





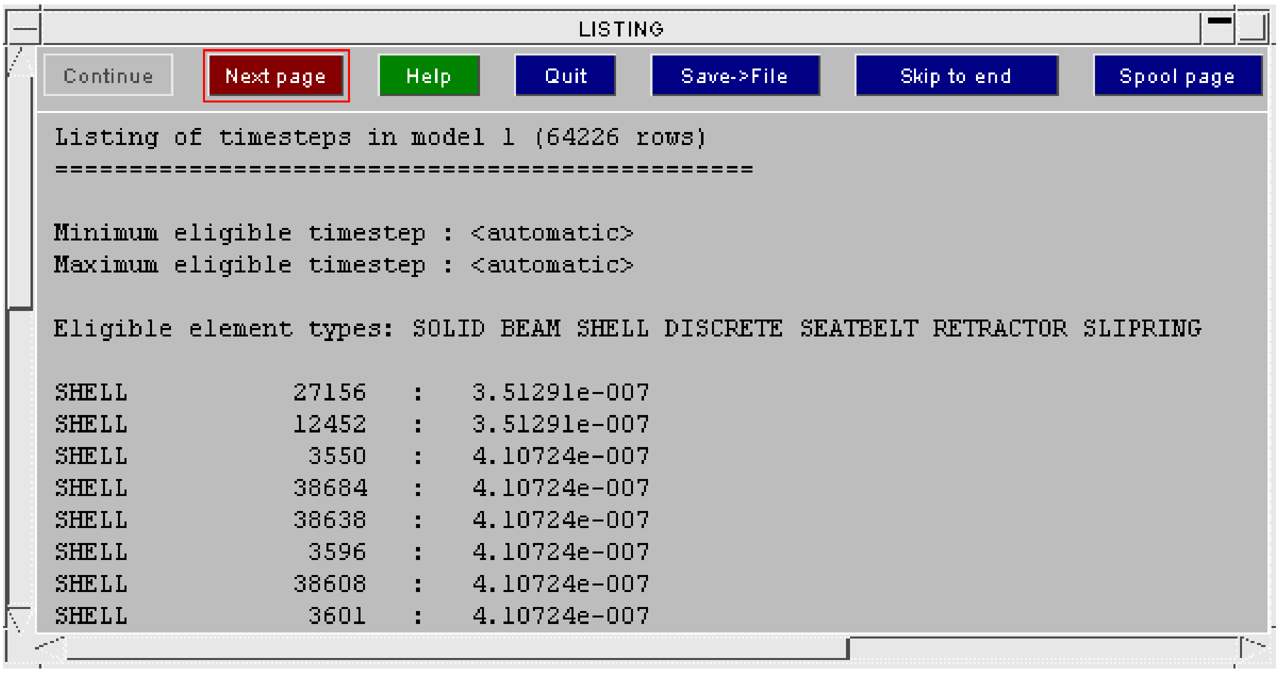
Here is the **Settings...** panel for element timestep plotting:



Note that:

- Timesteps are not computed for rigid elements, so their display is separately controllable.
- Timesteps are also not computed for elements that use more obscure material types. (Most commonly used materials are supported.) The display of uncomputed elements is also separately controllable.
- Timesteps are also currently not computed for discrete elements, or seatbelt types.

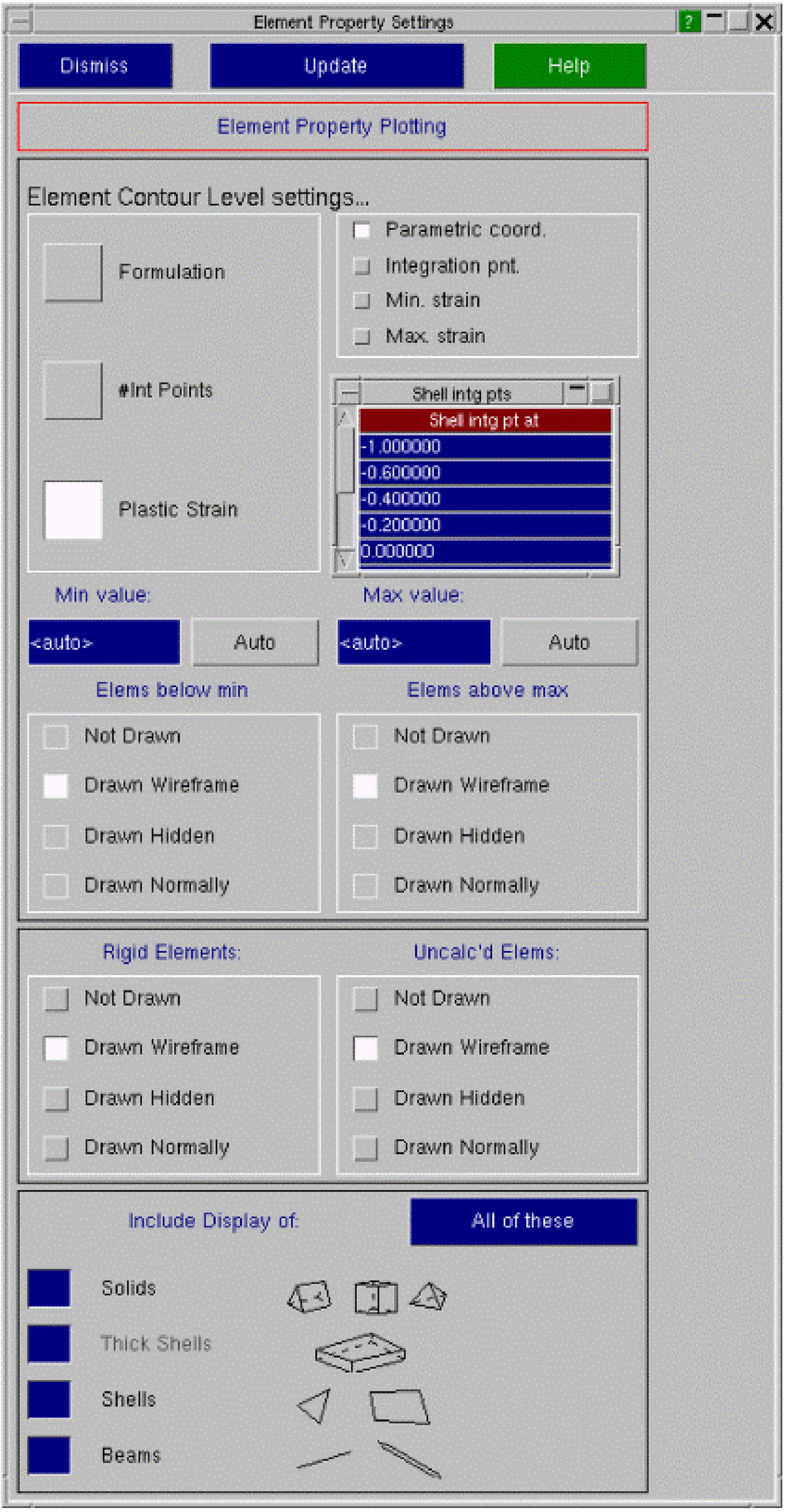
A **Listing...** of the smallest timesteps is shown below.



The remaining components that can be contoured in **CT** or **SI** mode are processed in a similar way, although the **Settings...** panel for each varies according to its context.

CT plot of plastic strains.

The **Settings...** panel for **Elem Props > Plastic Strain** is shown below:

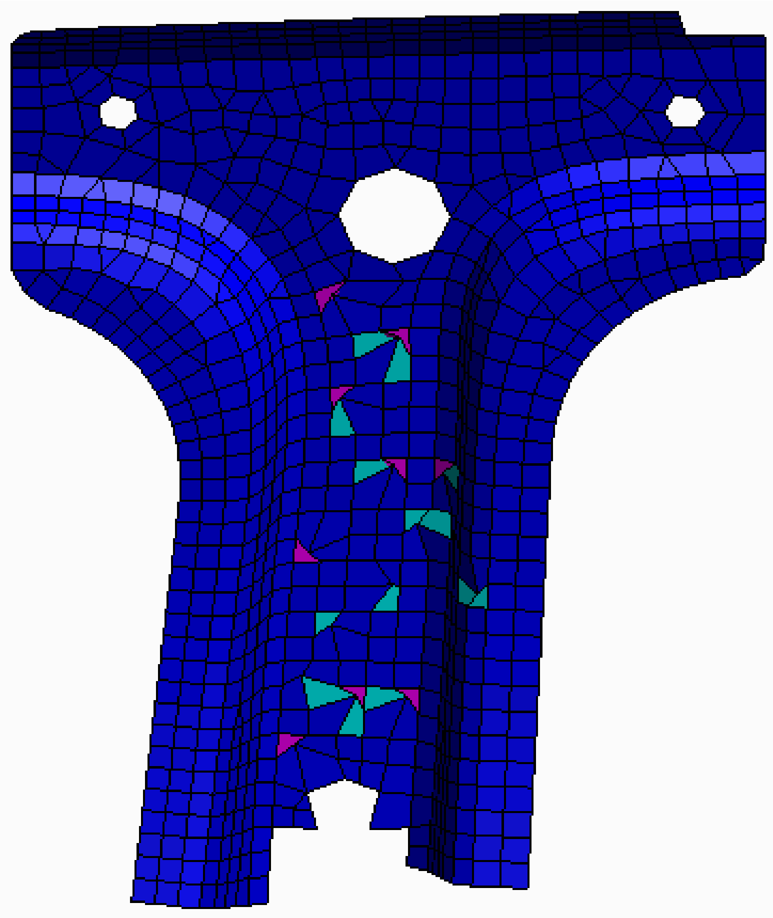


You are able to plot effective plastic strain values on initial stress cards in a number of ways. For parametric coordinates, the panel will contain a list of parametric coordinates of through-thickness shell integration points (-1 to 1 inclusive) sorted in ascending order. For integration points, the panel will contain a list of integration point numbers through the thickness. Finally, you can plot the maximum or minimum strain values for each shell. If the model(s) currently loaded in Primer does not contain effective plastic strain data, the list **"Shell intg pts"** shown above is blank.

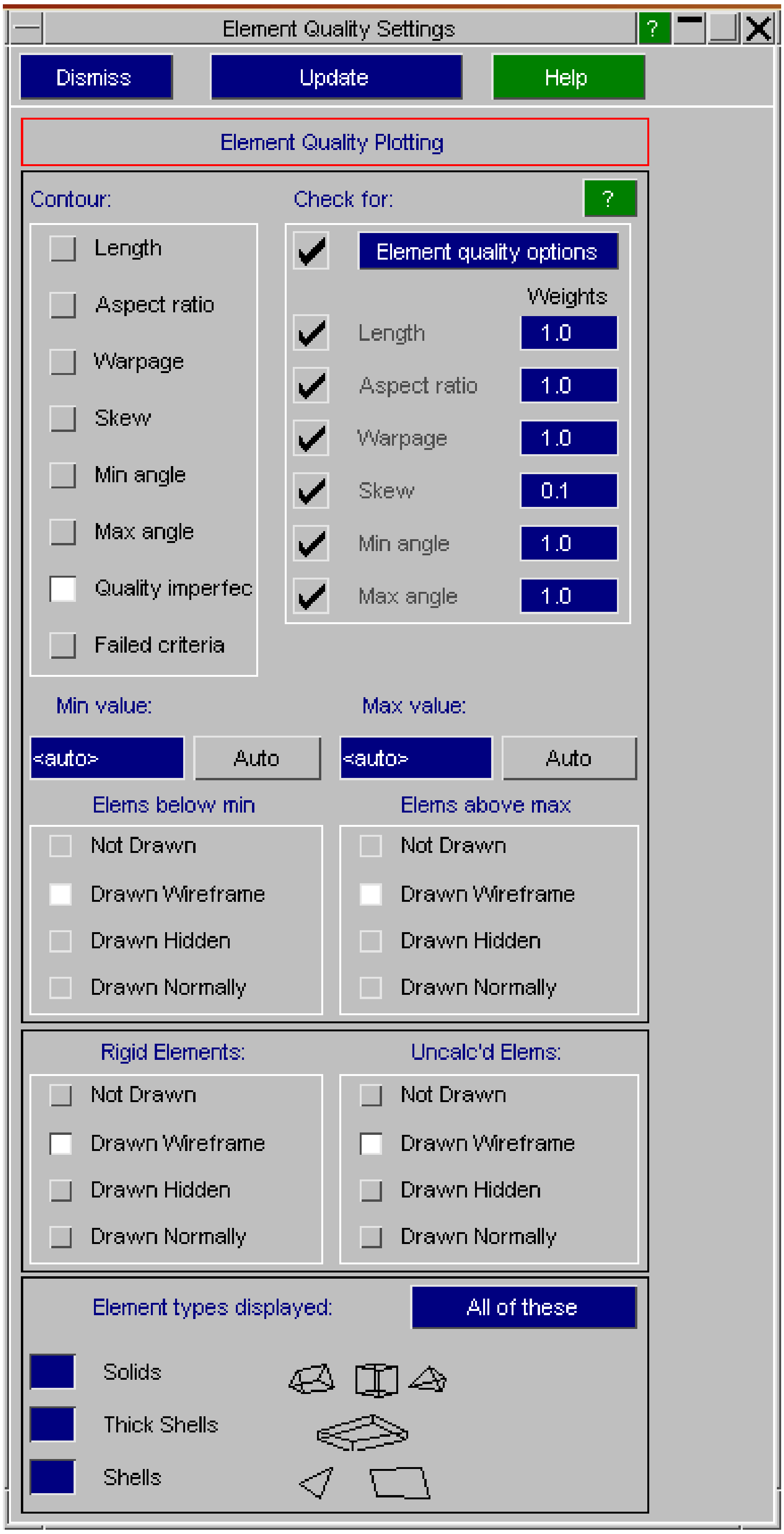
The effective plastic strains at any one of the listed locations can be plotted for all elements in the model(s) by selecting an item from the list and clicking **'Update'**.

CT plot of element quality.

A contour of Quality Imperfection is shown below:



The **Settings...** panel for **Elem Qual** is shown below:



Individual quality metrics such as aspect ratio can be contoured using appropriate radio buttons.

An overall penalty value can be contoured using the **Quality Imperfection** option. User can specify weighting factors used for this computation in the appropriate text boxes.

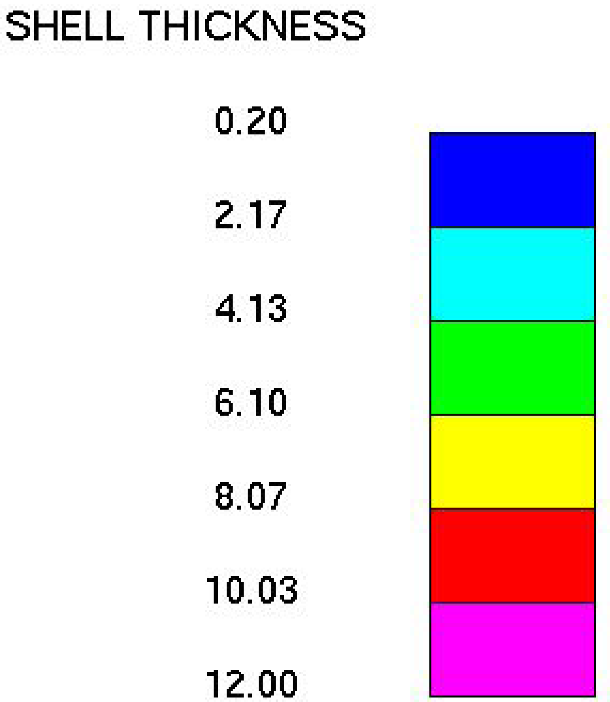
Elements failing one or more criteria can also be contoured using the **Failed Criteria** option. One or more metrics can be turned on or off using the appropriate tick buttons. Priority is given to that metric which has a higher overall penalty when an element fails multiple criteria.

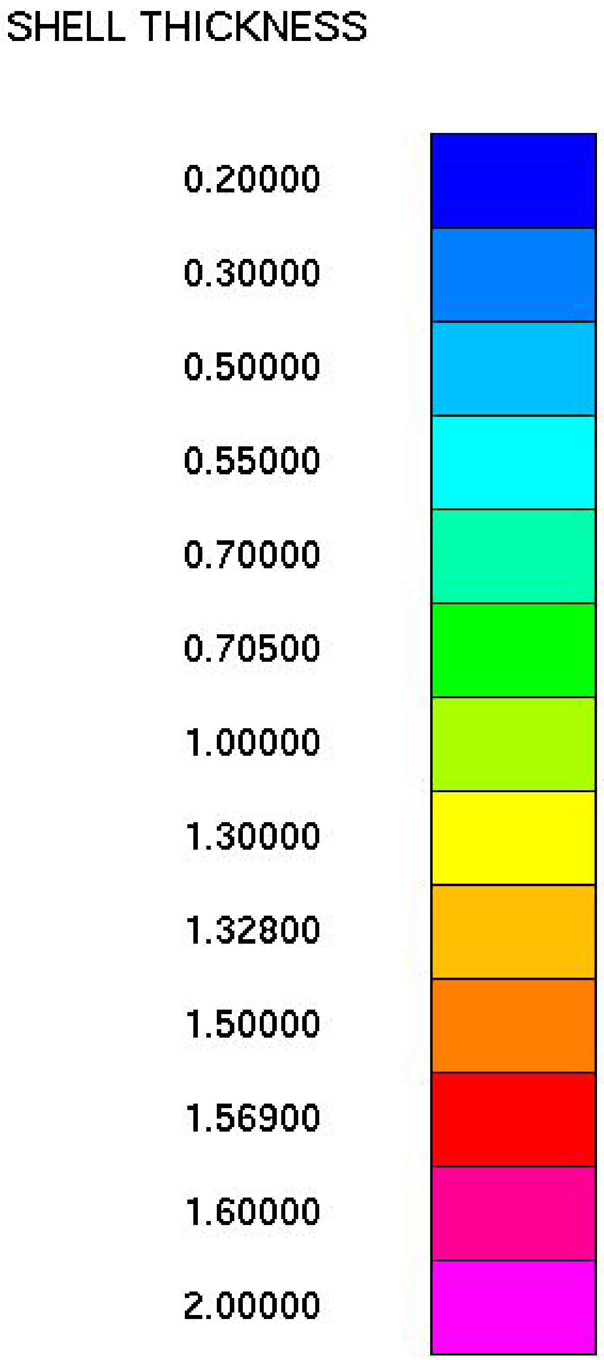
4.2.3 Contour levels on the contour ramp.

For all relevant Vector, CT and SI plots, the number of contour levels on the contour ramp can be set to any number between one and thirteen via the **Levels...** panel. When the number of distinct values being contoured is in excess of thirteen, the user-defined number of contour levels are displayed on the contour ramp with each colour representing a **range** of values being contoured.

However, if the number of distinct values being contoured in the visible model(s) is thirteen or less, each distinct value being contoured is allocated its own distinct colour on the contour ramp, and all values are automatically represented in it. Hence in such cases, the number of contour levels specified in the **Levels...** panel is ignored.

The following table illustrates the two different types of contour ramps just described. The first ramp is of a model containing shell elements with more than thirteen different thickness values. In this case, each colour represents a range of shell thicknesses, and the contour ramp contains six levels as specified via the **Levels...** panel. The second contour ramp is of a model containing shell elements with exactly thirteen distinct shell thickness values. In this case, each shell thickness is assigned its own colour in the contour ramp, thereby overriding the number of contour levels specified by the user.

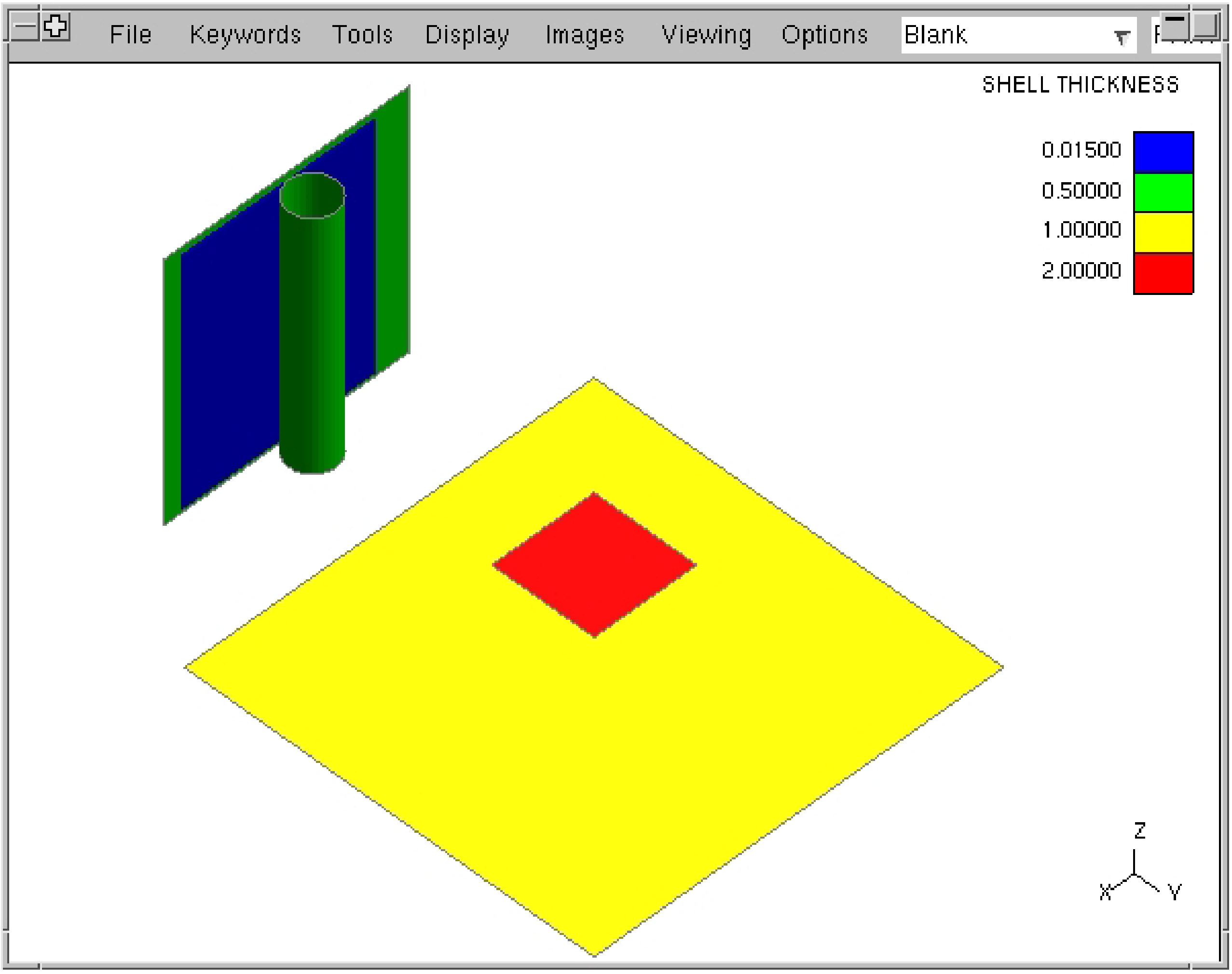




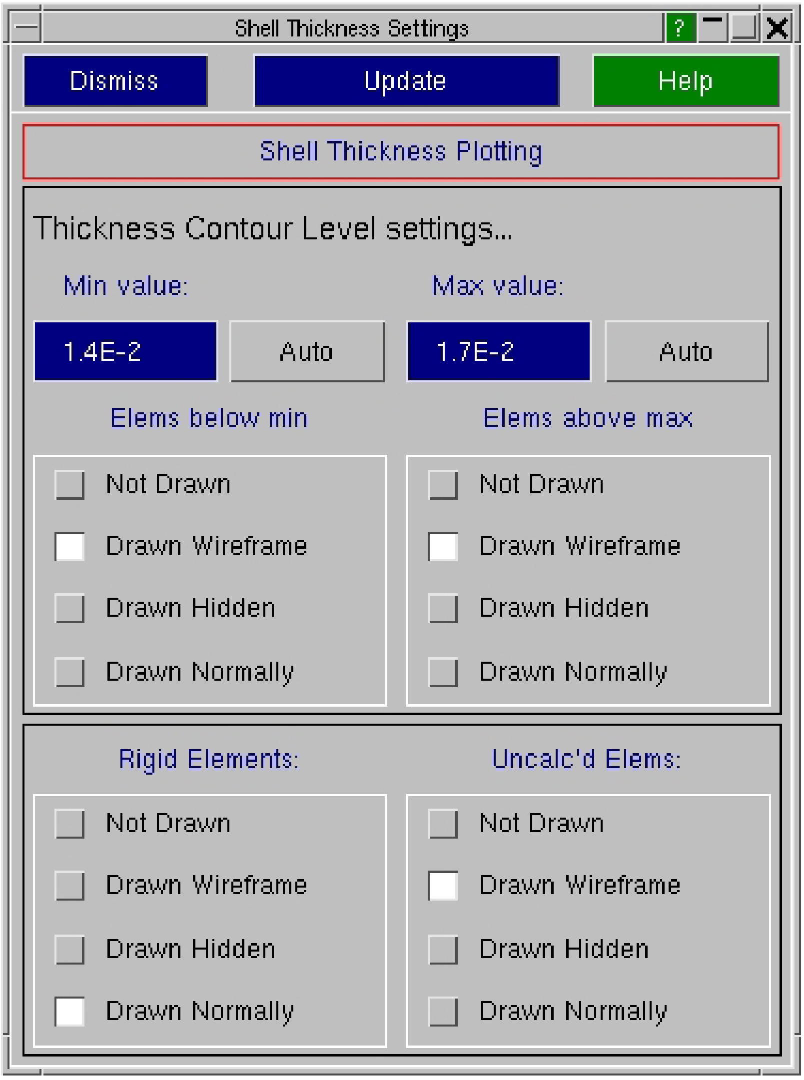
Contouring specific values in plots.

If required, a specific value of an entity can be contoured by specifying a narrow range of values in the **Settings...** panel.

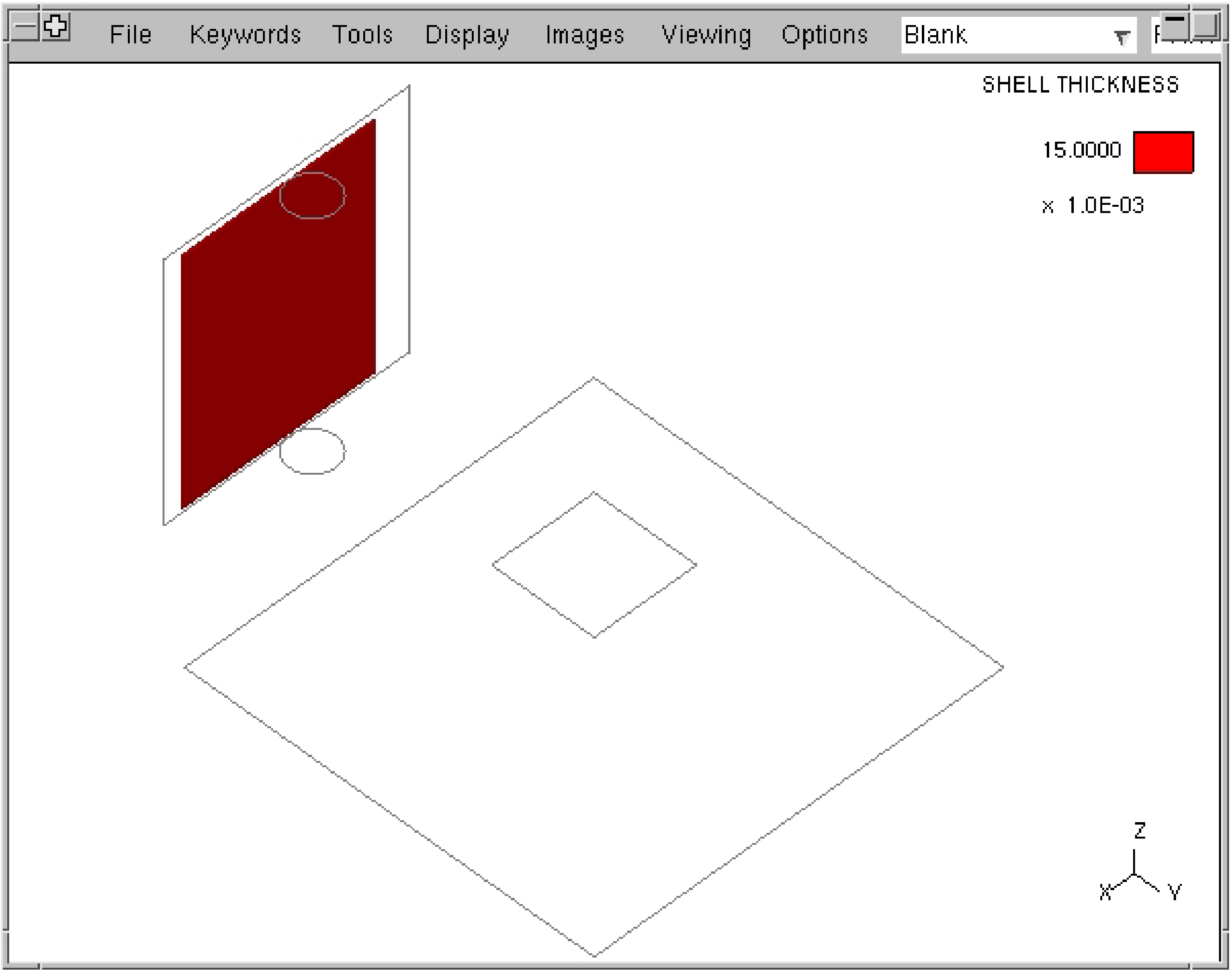
As an example, consider the following model containing shell elements of four distinct thickness values. The contour ramp thus contains four colours, each representing a distinct shell thickness value.



In order to visualize only those shell elements which are 0.015 units thick, a narrow range of values encompassing the desired value to be contoured is specified in the "**Min value**" and "**Max value**" boxes of the **Settings...** panel as shown below.


Shell Thickness Settings
Dismiss
Update
Help
Shell Thickness Plotting
Thickness Contour Level settings...
Min value:
Max value:
1.4E-2
Auto
1.7E-2
Auto
Elems below min
Elems above max
Not Drawn
Drawn Wireframe
Drawn Hidden
Drawn Normally
Not Drawn
Drawn Wireframe
Drawn Hidden
Drawn Normally
Rigid Elements:
Uncalc'd Elems:
Not Drawn
Drawn Wireframe
Drawn Hidden
Drawn Normally
Not Drawn
Drawn Wireframe
Drawn Hidden
Drawn Normally

Clicking the **UPDATE** button produces the following plot.



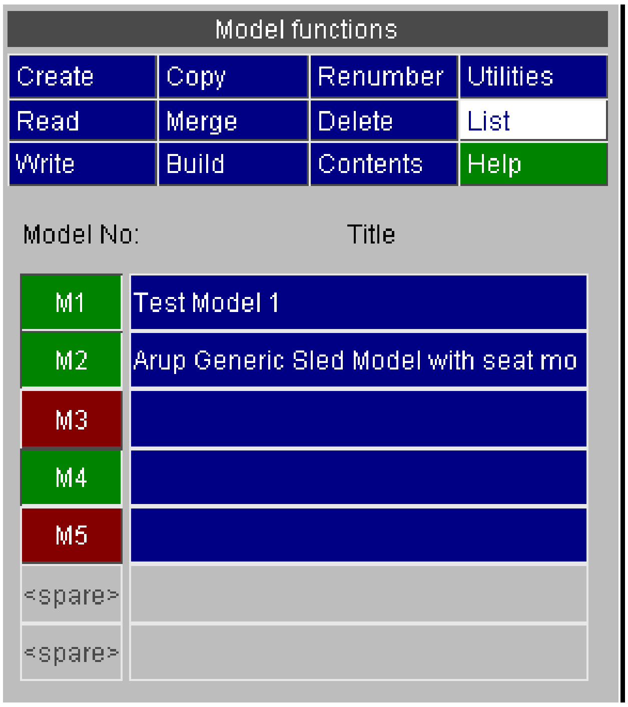
In the updated plot, shell elements with a thickness of 0.015 only are contoured as desired, while the remaining elements are drawn in the **wireframe** mode as per the options set in the **Settings...** panel.

Note that it is necessary to specify a range of values as opposed to the specific value to be plotted in the **Settings...** panel. If the exact value to be plotted is specified in both the "**Min value**" and "**Max value**" boxes, rounding errors that occur during computation might prevent the desired plot from being generated properly.

4.3 Controlling Model Visibility

Models can be enabled or disabled for display at will. This is carried out by setting them to "hidden" or "viewable": hidden models will not be drawn by any drawing command. By default a model is viewable when it is first read in, but thereafter its visibility is controlled by the user. Changing its status only takes effect the next time a drawing command is given.

Manipulating a model’s status is simple:

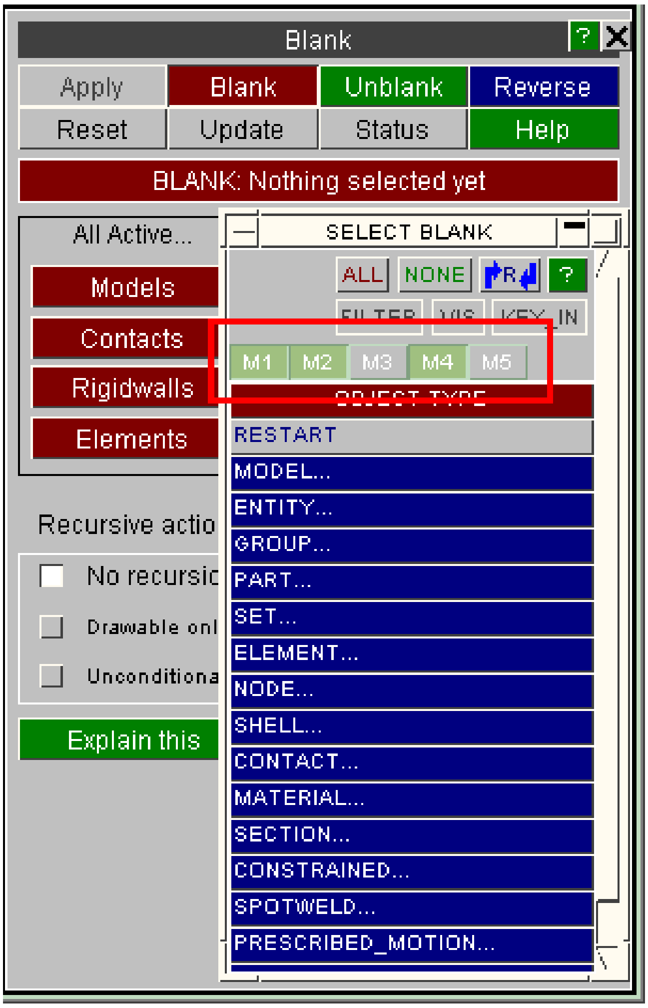


Under **MODEL** > **LIST** click on the **Mnnn** buttons in for the relevant models. A depressed button (green) is viewable, undepressed (red) is hidden.

In this example models 1,2 and 4 are viewable.

Setting a model’s visibility in this way has the highest priority when determining whether something should or should not be drawn. If the model is not viewable none of its contents will be, regardless of Entity switches or Blanking. However, making a model viewable does not cause its contents to be displayed if the entity types are not visible (section 4.4) or if the entities are blanked (section 4.5).

In addition turning off a model in the **MODEL** > **LIST** menu has the effect of turning off its "**Mn**" tab in all selection menus throughout the code. For example given the case above of five models, with M3 and M5 deselected, the **BLANK** panel will start off looking like this:



Note that the M3 and M5 tabs are deselected. You can still turn them on manually if you wish.

In other contexts, for example when creating items, if you only have one model "live" in the **MODEL** > **LIST** menu the question "which model do you want to create it?" will be omitted, saving one mouse click.

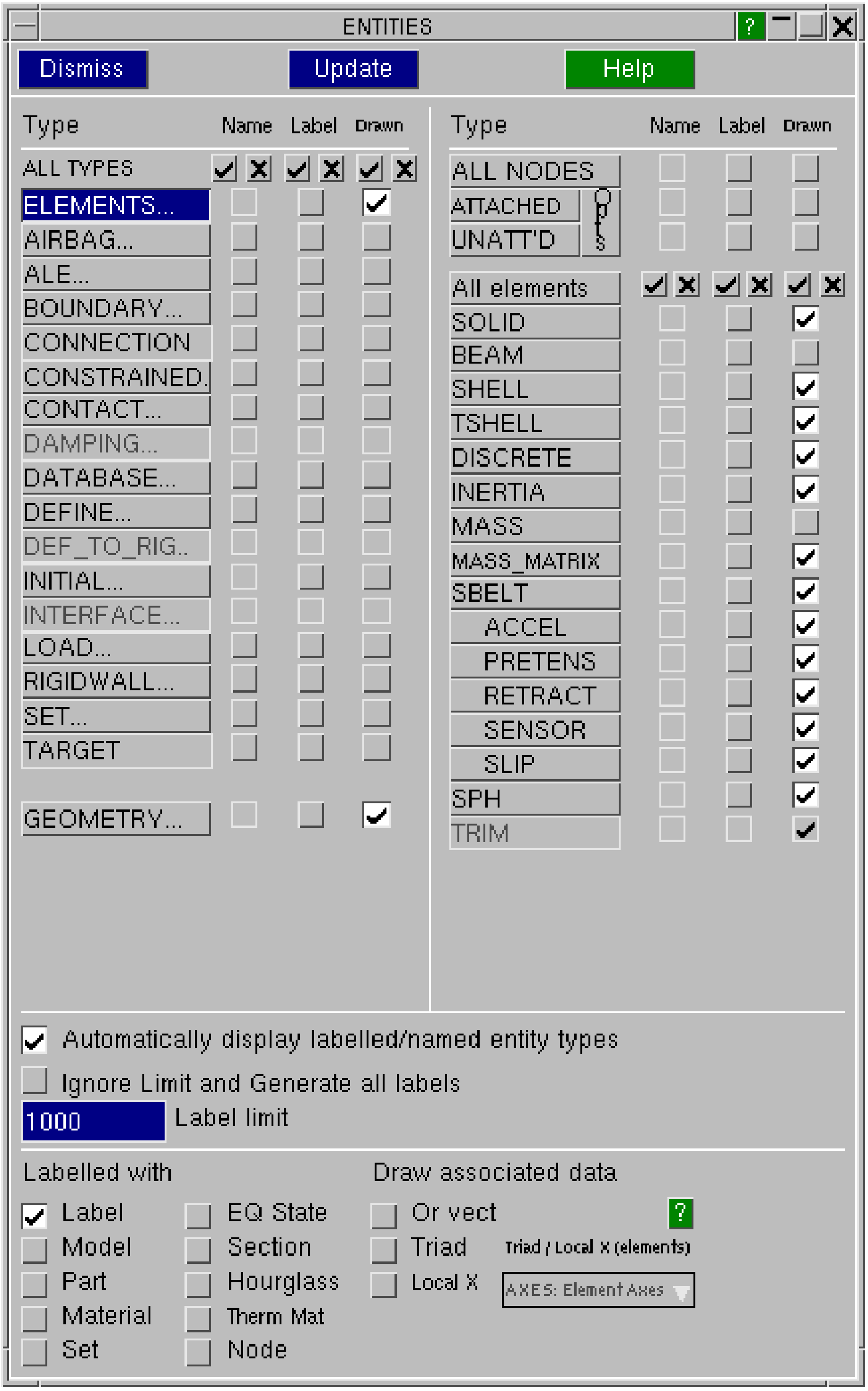
4.4 Controlling Entity visibility and labelling

By default only the elements in a model are drawn, with no labels, node symbols or other information appended to them.

You can add extra information to plots, control the display of classes of information and label items dynamically on the screen using the **ENT**ity Viewing panel. This can be accessed in 3 ways:

1. The keyboard shortcut key **E**.
2. The top bar menu **DISPLAY** > **ENTITIES**
3. The **ENT** from the viewing and drawing window.

This panel controls the display of elements and nodes, (ie basic "structural" items); also their symbols, labels and local direction triads as well as the display of "other" items, such as constraints, contacts, rigidwalls, etc; and also their labels, symbols and other related displayable data.



It must be stressed that these commands only permit or deny the display of *classes* of information, they do not control the visibility of individual items or models. However they do provide one means of accessing the "dynamic" labelling of items: see section 4.6.

For example they might be used to enable the display of nodes and of contact surfaces. This would permit nodes and contacts in any models to be displayed provided they were not made invisible by some other command.

The left hand column of the panel dictates the display of the right hand column. At any one time a "master" category will be selected from the left-hand column (in this example **Elements** is selected). The "master" categories each contain further "child" categories below them. The right hand column displays the appropriate "child" categories for the selected"master". The **Label** columns control whether or not the items will be labelled (with the information selected under **labelled with.** The **Drawn** columns control whether or not the items will be drawn. "Child"categories can be controlled individually (in the example shown the display of beams has been turned off), or all the child categories may be switched on/off together by switching on/off the master category or the ALL_<*category*> row.

Labelled with determines what is actually drawn as a "label" when labelling is selected for an element or node class.

Selecting multiple labelling categories will lead to compound labels being generated (eg **M1/H1001/P12/MAT12**) and plots will become very cluttered if too much information is displayed.

4.4.1 Elements and nodes (Structural Items).

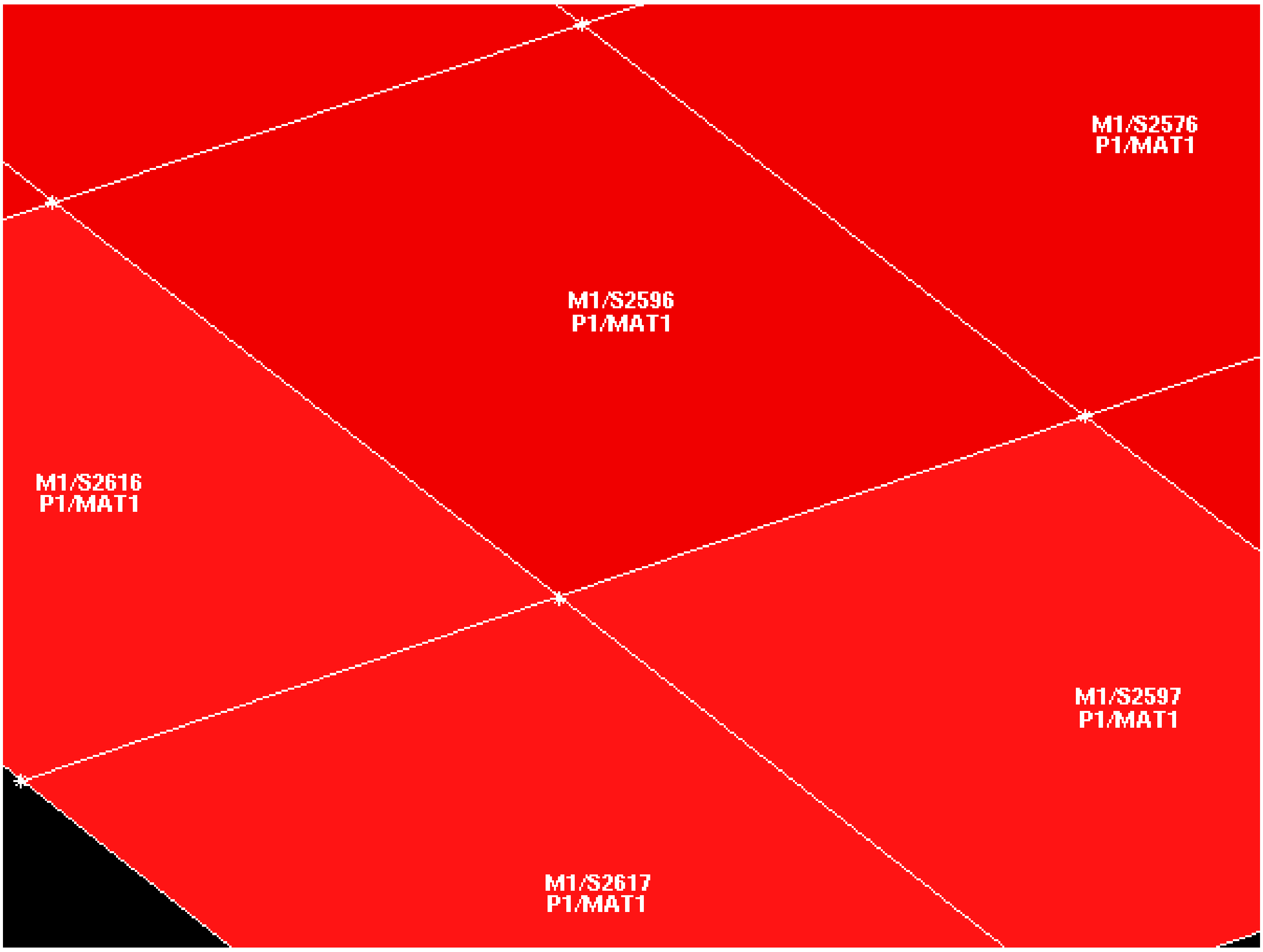
Nodes are treated as a special case:

- **ALL NODES** draws all nodes, regardless of attachments.
- **ATTACHED** draws only nodes attached to some other items currently displayed.
- **UNATTACHED** draws only nodes that are not attached to anything (visible or not).

Associated data: Local direction **TRIADS** are drawn for element types with coordinate systems, and **OR**(ientation)**_VECT**(or)**S** for springs and dampers.

4.4.2 How labelling on plots is handled for nodes and elements

The default label is a node or element number, but a variable amount of information can be generated to form a "label" which can run to multiple lines, as this example shows:



This figure shows an example of shells which have been labelled with:

MODEL	**Mnnn**	for ***M***odel number <nnn>
LABEL	**Snnn**	for ***S***hell <nnn>.
PART	**Pnnn**	for ***P***art <nnn>.
MATERIAL	**MATnnn**	for ***MAT***erial <nnn>

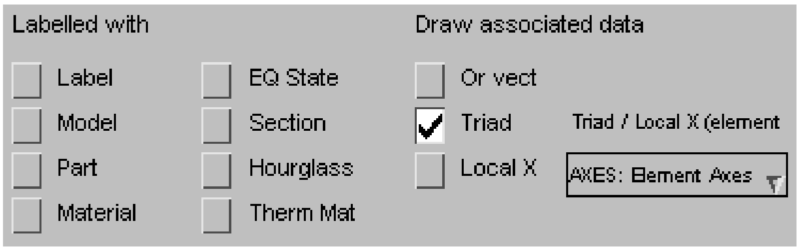
PRIMER attempts to group labels logically and to locate them so that they don’t overlap, but if you try to add too much information you will end up with a total mess on the page. This example, with four categories of data labelled on elements, is the sensible maximum; and even it starts to get messy when label numbers get large (> 5 digits).

Labelling uses the standard acronyms for entities, these are listed in Appendix 1.

The "attached" nodes in this figure have also been switched on: these are drawn as asterisks (*) at the relevant element vertices.

4.4.3 Triads (elements)

It is possible to draw triads on elements that would depict the local material orientation. Alternatively, the local X direction can be drawn by toggling the appropriate button "On".

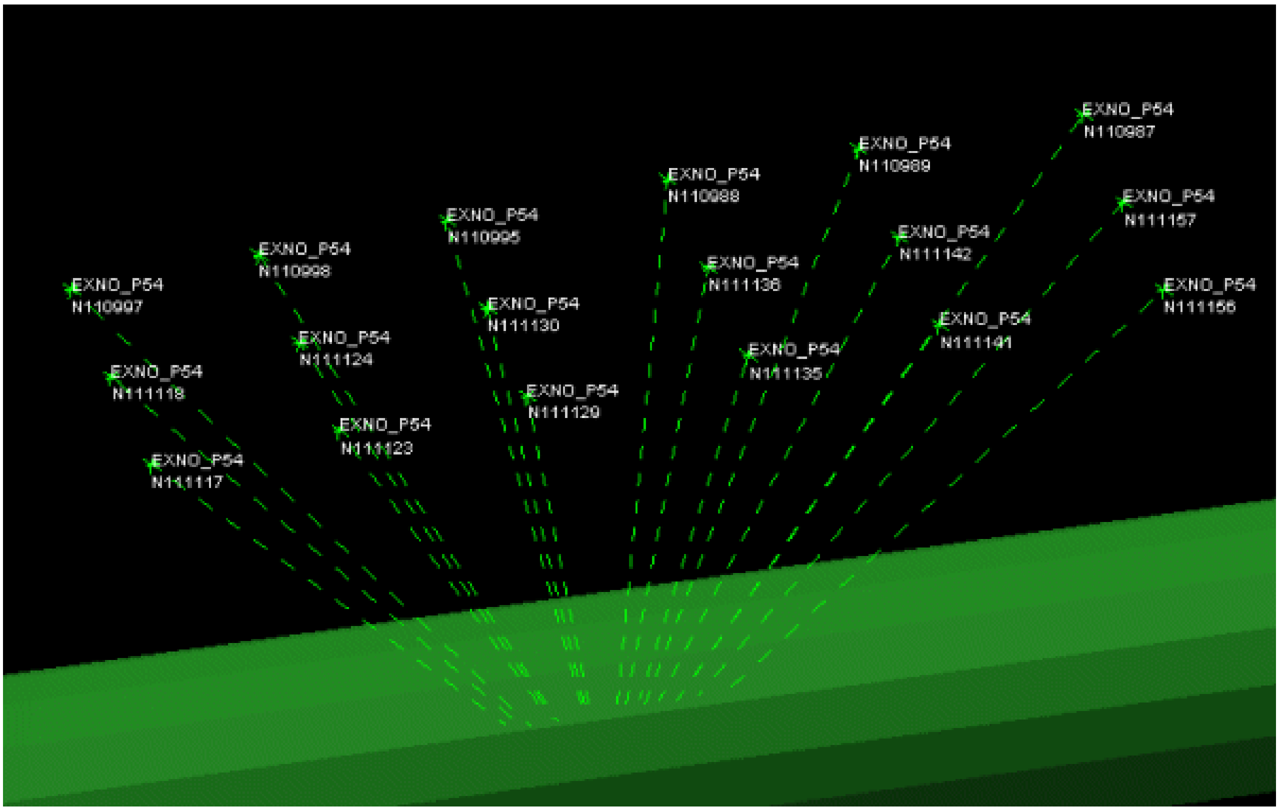


The following options are available for drawing element triads/local X direction and can be chosen using the popup:

Element Axes	This is the default option. Element orientation, as defined by its topology is drawn. Local angle specifications are disregarded.
Material Axes	Local angles as defined by MAT, ELEMENT_SHELL_BETA, ELEMENT_SHELL_MCID, ELEMENT_SOLID_ORTHO are computed. A suitable triad/local X is drawn on each element. However, layer-specific angles are not evaluated.
All layers	This option is only applicable to shells. Local angle calculation is carried out as in the "Material Axes" case. In addition, local direction specification is considered for each integration point. This can be defined using PART_COMPOSITE cards or using SECTION_SHELL cards in conjunction with INTEGRATION_SHELL or Gaussian or Lobatto integration rules. A triad/local X is drawn for each layer.
Top, bottom, middle layers	Local angle computation is carried out as in the "All layers" case. However, only the top, bottom, and middle layers are sketched. A sensible middle integration point cannot be identified if an INTEGRATION_SHELL is defined or if the element has even number of integration points. In such cases, only the top and bottom integration points are drawn.
Intg pt <n>	Local angle computation is identical to the "All layers" case. However, the triad/local X is drawn only for the specified integration point <n>.

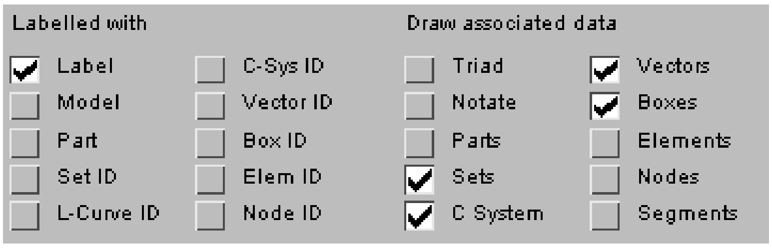
4.4.4 "Non-element" items

Most of the viewable entity types are shown by special symbols. For example, *CONSTRAINED_EXTRA_NODES are illustrated below.



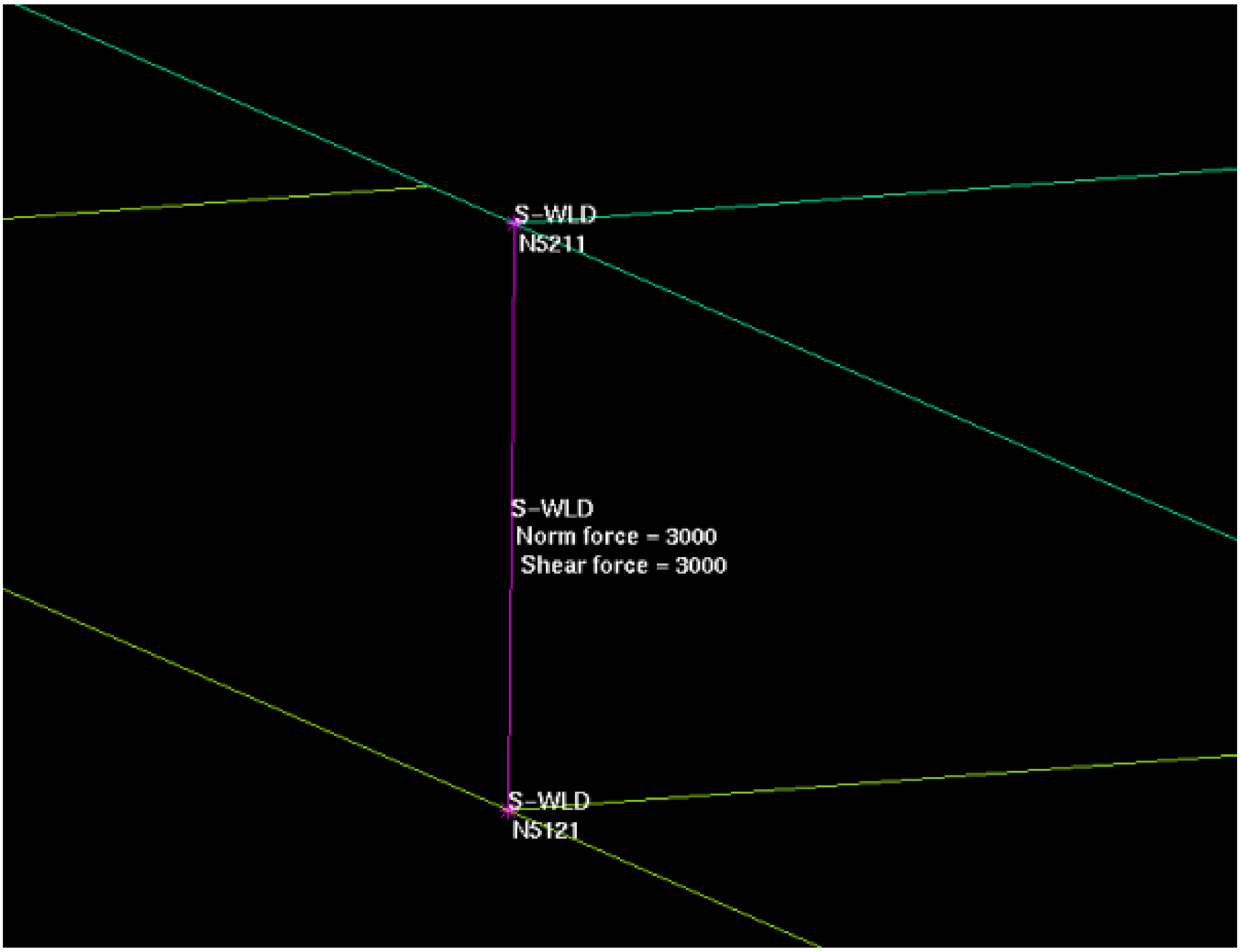
4.4.5 How "associated data" is drawn for non-element items

The addition of extra "associated" data to symbols is controlled by the lower half of its box:.



An example might be a contact surface, which is defined by sets of parts, sets of nodes, bounding boxes and which references a load-curve. It is possible to draw and label all these items (although the screen might get a bit cluttered!)

Therefore it is necessary to get used to the idea of primitive objects (nodes, elems, boxes, ...) being drawn because they make up part of some other higher order entity, and being labelled with that entity.



In this example, which shows a ***CONSTRAINED_SPOTWELD**, both the weld itself and the nodes at its ends have been drawn.

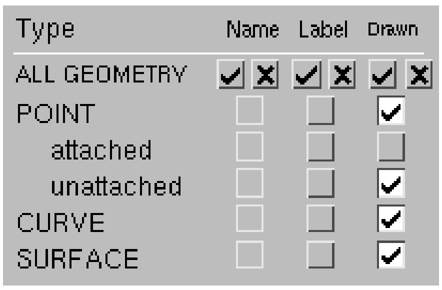
The labelling of associated nodes has been turned on, but note that the primary label on the nodes associates them with the spotweld since it is their "parent" entity in this context.

This example also demonstrates the use of the **NOTATE** function. This is a global function which adds "useful" data where possible to some visible items, in this example the normal and shear force values of the spotweld.

Other examples are the stiffness and damping factors of joints; motion and orthotropic data on rigidwalls; stiffness and stop angle data on generalised stiffnesses; and so on. Generally speaking **NOTATE** adds extra data to items where this can be expressed concisely enough to fit onto the screen.

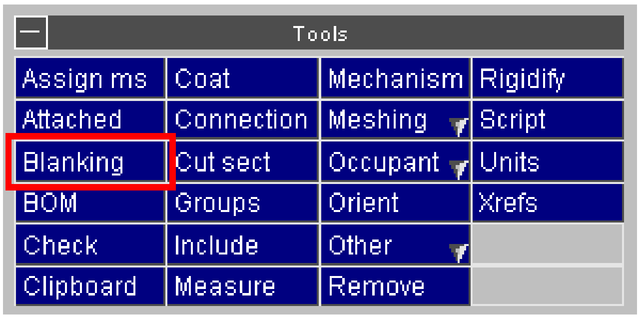
4.4.6 Geometry entities

From version 10.0 PRIMER can display basic geometry. Points, Curves (and lines) and Surfaces can be displayed. By default curves, surfaces and any points that are not attached to a curve or surface are drawn. This can be changed like any other entity type in PRIMER by using the **GEOMETRY...** button.



Note that the geometry engine in PRIMER is still in its infancy. There are currently no facilities to create or modify geometry. Additionally rendering of many surfaces will make PRIMER use considerably more memory and will make drawing slower. It is expected that the geometry engine will be developed over time. Please contact Oasys Ltd if you encounter any problems.

4.5 BLANKING Controlling entity visibility



Blanking allows the user to cut down what is displayed by controlling whether individual items are marked as drawable or not.

For an item in PRIMER to be drawn it must pass the following three tests:

Is the model visible?	=>	Is the entity type drawable?	=>	Is the entity unblanked?
(See section 4.3)		(See section 4.4)		(This section 4.5)

These represent increasingly more detailed levels of testing and the last of these checks, blanking, is performed on a per entity basis. Every drawable entity in PRIMER may be flagged as

- either "***blanked***" (not eligible for drawing)
- or "***unblanked***" (eligible for drawing)

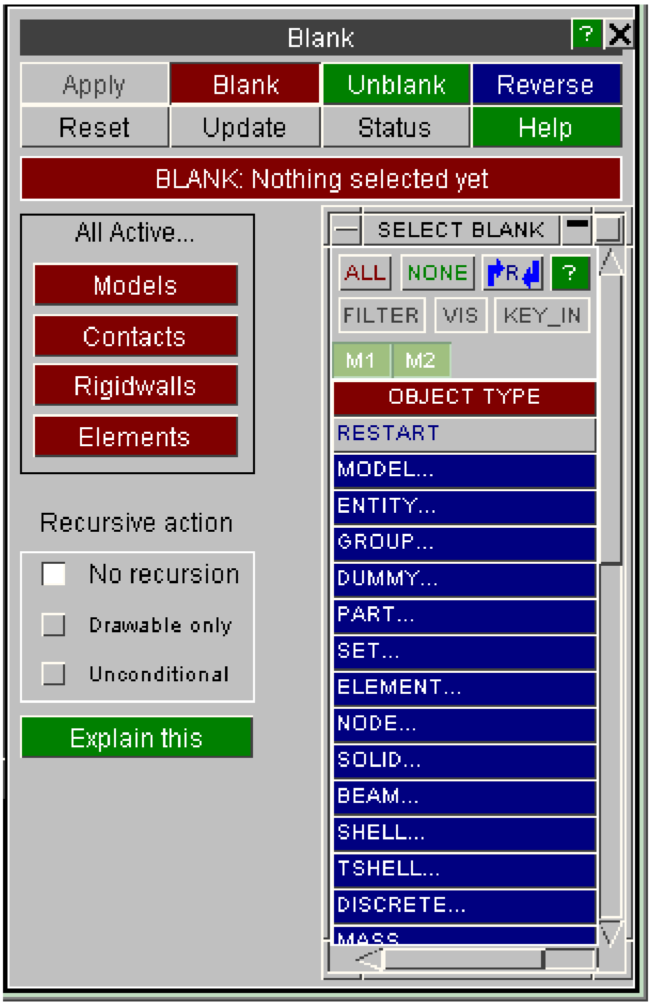
The default being ***unblanked***. Control of the blanking status can be exercised in the standard hierarchical fashion of models, sets, parts and finally down to individual items; thus it may be used to control exactly what is seen on the screen. The way that blanking selection propagates down through the model can be controlled by the **Recursive Action** setting - see section 4.5.1 below.

As well as the main **BLANK** menu described in this section, Blanking may be activated by:

- Quick Pick blanking (section 4.5.2 below)
- The Part Tree (section 4.5.3 below)
- Keyboard short-cut keys (section 4.5.4 below)
- Special keys in the View panel (section 4.5.5 below) which include "locking".

4.5.0 The BLANK menu

This figure shows the main **BLANKING** Menu:



It has three colour-coded states:

BLANK	(Red) makes the selected entities invisible.
UNBLANK	(Green) makes the selected entities visible again.
REVERSE	(Blue) inverts the status of the selected entities.

To use it: select a list of items, choose one of the three states above, and press **APPLY**.

The effect will be seen the next time the image is drawn, or when **UPDATE** is used.

The **ALL_xx** commands are to provide short cuts for commonly issued commands:

ALL_MODELS Means ***everything***! All the contents of all models currently in memory will be operated on.

ALL_CONTACTS Means all contact surfaces in all models.

ALL_RIGIDWALLS Means all rigid walls in all models.

ALL_ELEMENTS Means all elements (of all types) in all models.

These short cut commands will operate faster than the equivalent commands from the **SELECT** menu since they don’t have to perform the hierarchy propagation checks implicit in using the menu.

RESET Resets to null the contents of the **SELECT** menu. This may be used to delete any current selection and start again.

UPDATE Redraws the current image following a blanking change. This is necessary to see the effect of any changes (unless the Update Level in the View Control box has been set to "frequent", in which case changes take effect immediately).

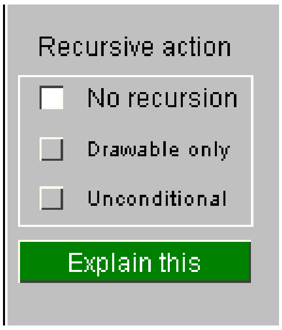
4.5.1 Recursive Blanking

Blanking in PRIMER has always been a contentious issue because of the way that selection propagates down through the hierarchy of items in a model. The increasing complexity of ls-dyna models has exacerbated this problem.

In earlier versions, where blanking propagation was unconditional, a user could accidentally blank a "junior" object (typically nodes) through propagation without being aware that this was happening. This would then give rise to "why won’t it draw xxxx?" questions, which could only be solved by unblanking things which the user didn’t think he had blanked in the first place!

In PRIMER V9.1 an on/off switch for blanking propagation was provided.

In PRIMER V9.2 yet more control has been provided over how blanking propagates through the structure by allowing the user to set the **Recursive Action** value, which controls how blanking (but not any other form of selection) is propagated down a model hierarchy.



"**No recursion**" means that only the selected items are blanked, with no propagation.

"**Drawable only**" means that blanking propagates downwards, but only affects items that are currently drawable (ie their Entity switch is on).

"**Unconditional**" propagates blanking unconditionally down through the model.

The "**No recursion**" case has some exceptions built into it as follows:

- Blanking an item that is not itself drawable, but which is drawn via its underlying items (eg MATERIAL, PART, SECTION, SET, etc), causes limited propagation downwards to the drawable items. Thus, even with "**No recursion**" set, blanking a MATERIAL will propagate downwards to blank the elements of that material, but not any further (ie not to the nodes on the elements).

- SETs are another special case when "**No recursion**" is selected:

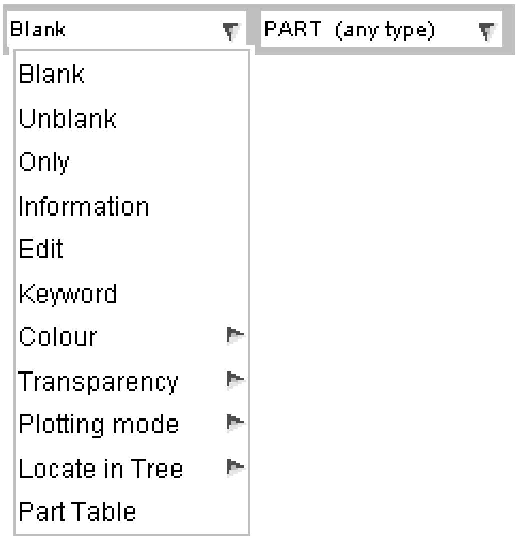
    - Normally SETs are ***not*** drawn explicitly, and if a SET is blanked then the blanking propagates down as above to the drawable items in the set.

    - However if SETs ***are*** drawn then blanking them stops them being drawn, but does ***not*** affect the visibility of the underlying items, meaning that the SETs will no longer be superimposed on the image.

The **Recursive Action** flag also affects how "Quick Pick" and Part Tree blanking propagate (internally BLANK, "Quick Pick" and Part Tree blanking are the same, simply using different selection methods.)

4.5.2 "Quick Pick" Blanking.

The "Quick Pick" functionality has been described in section 2.9. However it is so central to PRIMER usage that it merits a brief repeat here.



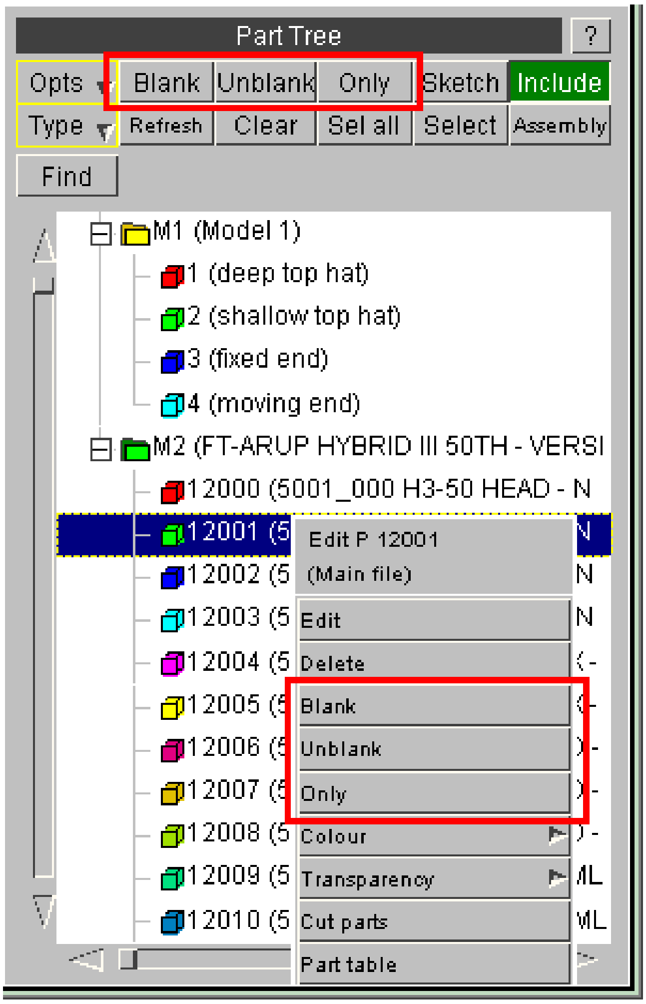
Blank Blanks the selected items.

Unblank unblanks them

Only draws only the selected items, effectively blanking everything else.

4.5.3 Part Tree Blanking

Blanking may also be performed from the Part Tree (see section 6.17).



The current operation of the Part Tree can be set to **Blank**, **Unblank** or **Only**.

Similarly a right-click popup on a given row also contains those options.

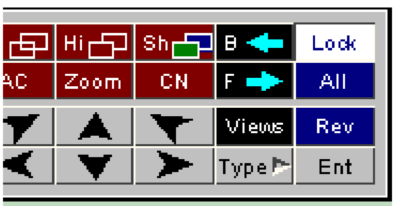
4.5.4 Blanking control using keyboard shortcut keys

The following keyboard "short cut" keys influence blanking:

U (**nblank all**)	Unblanks everything, ***unconditionally***, subject to "locking" (see below)
R (**everse all**)	Reverses the blanking status of everything, using the current **Recursive Action** logic.

4.5.5 "Locking" blanking in the "View" panel.

Blanking may also be "locked" to its current status via the following buttons in the View panel:



Lock "locks" the current blanking status so that keyboard short cut **U**(**nblank all**) returns to the "locked" visibility status. The Lock button toggles on / off.

All is exactly the same as keyboard shortcut **U**(**nblank all**) above

Rev is exactly the same as keyboard shortcut **R**(**everse all**) above

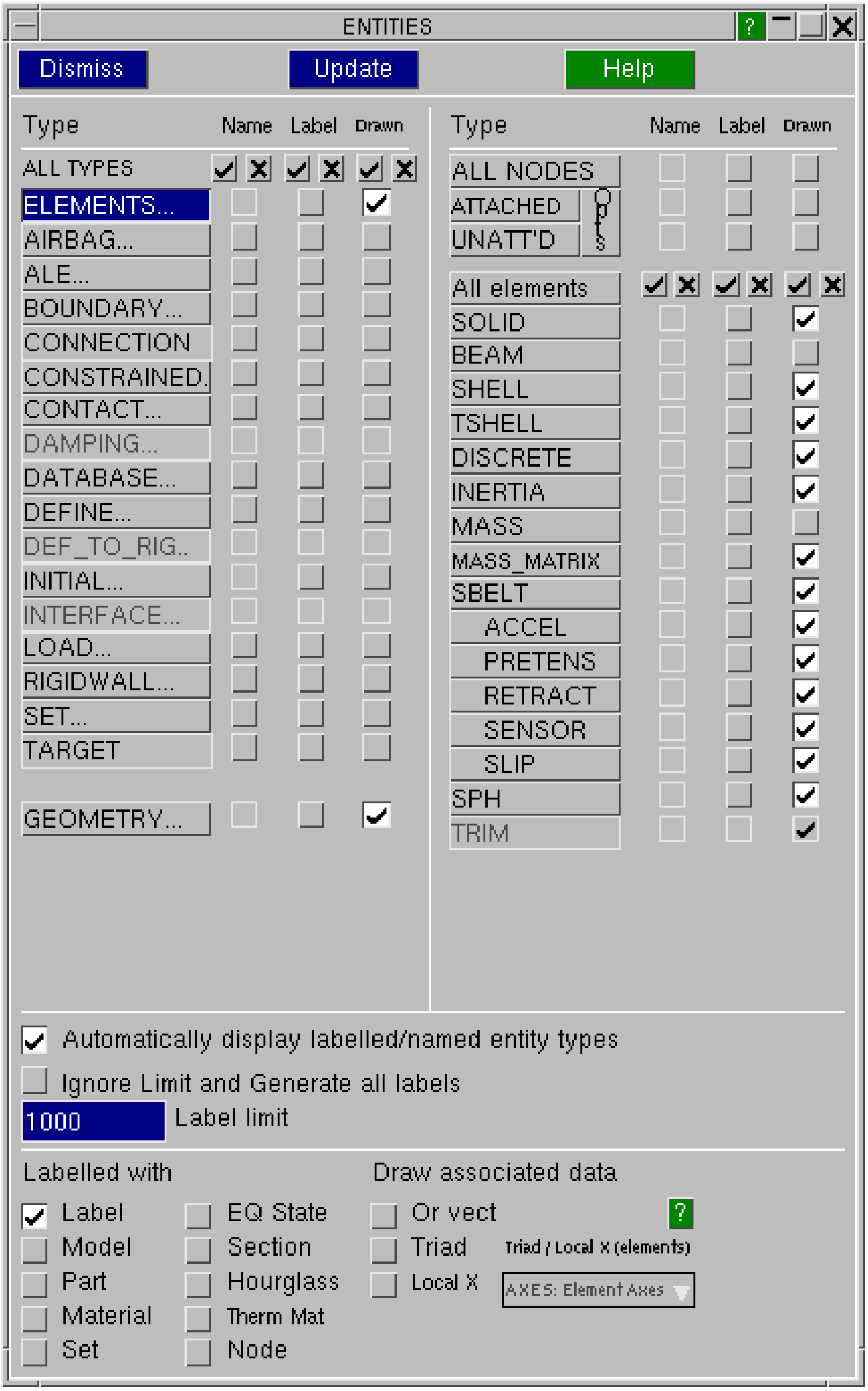
The purpose of "locked" blanking is to allow the user to return easily to a previous image.

"Locking" does ***not*** affect blanking carried out by the main Blank Panel, Quick Pick blanking, or blanking from the Part Tree.

4.6 Dynamic Labelling

Sketching labels and associated information on the existing plot.

4.6.1 Using the "type" Element and Node buttons in the Entities panel

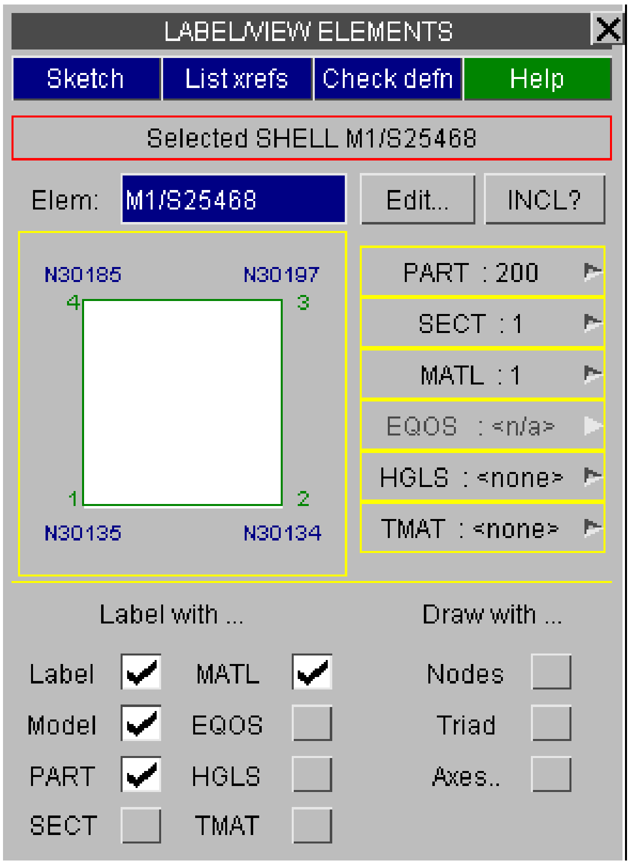


In section 4.4 above the use of the ENTity Viewing panel to control labelling on plots was described.

It is also possible to label nodes and elements "dynamically", which means "instantly on the existing image".

- Select**ELEMENTS** from the left hand "type" column
- Select a category from the right hand "type" column .
- This maps the labelling panel for that item type (below).
- Click on items of that type to label them immediately.

Alternatively, use Quick Pick, set the entity type to SOLID, SHELL, etc and the action to "Element Details".



This figure shows a typical dynamic labelling box for shell elements.

It is updated automatically as you click on elements, or you can type a new element number into the **Elem:** box.

More than one model is current in this example, so typed in elements must be prefixed with their model id. In this example shell element 25468 in model #3 (**M3/S25468**) has been selected.

The **EDIT** button invokes the detailed editing panel for this element. The **INCL?** button lists the elements position in the model's include file structure. **List Xrefs** invokes the cross-reference viewer for this element (see section 6.15 for more details).

Not only is the element in question labelled on the screen, but its major attributes are presented in this panel:

- The nodes on the element are drawn schematically. (Note that the schematic shape is idealised, here as a square, not the true shape of the element.)
- Its Part, Section, Material and other attributes are given.

By using the popup menus against the **PART, SECT**, etc boxes it is possible to view the details of these in their respective edit/browse panels.

The "**Label with ...**" buttons control how the selected items are labelled on the screen. The categories are the same as those in the main **ENT**ity Viewing panel, but apply only to these "dynamically" labelled items.

The "**Draw with ...**" buttons control what extra information is added to the selected items:

Nodes	Adds the labels of nodes connected to this element
Triad	Draws the local axes as a triad (if relevant for this class of element)
Axes	Draws other local axes where relevant: orientation vectors, etc

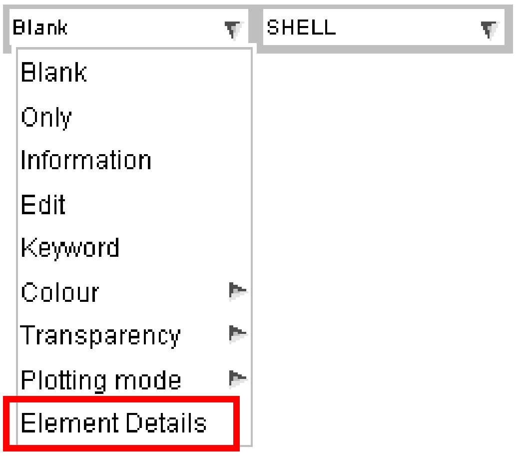
Selection will be limited to the class of item selected in the **ENT**ity Viewing panel. However selecting class

ALL_ELEMENTS permits any class of element to be selected for labelling.

The details of this **LABEL/VIEW** panel will vary with the class of object being shown: for example the panel for nodes doesn't show a diagram but rather lists coordinates, restraints, rigid connectivity (if any), etc for the node.

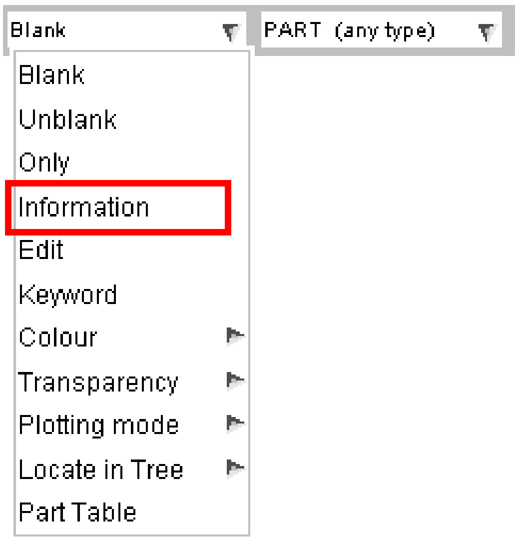
This panel can also be invoked anywhere in PRIMER from popup windows offering the **LABEL/VIEW** option

4.6.2 Using the "Quick Pick" Element and Node Details option



The same panel may also be invoked using the Quick Pick **Details** option for nodes and elements only.

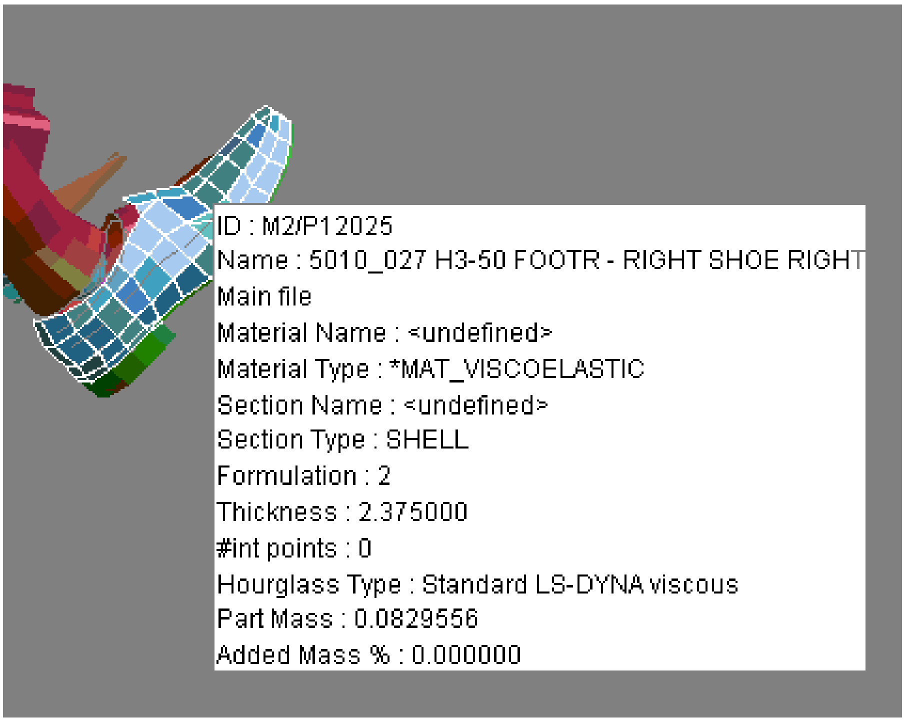
4.6.3 Using the "Quick Pick" Information option.



The "information" option in the quick pick menu gives type-specific data about any item picked.

This includes the label, but also a host of other information.

In the example below the user has clicked on the foot of a dummy to receive information:


ID : M2/P12025
Name : 5010_027 H3-50 FOOTR - RIGHT SHOE RIGHT
Main file
Material Name : <undefined>
Material Type : *MAT_VISCOELASTIC
Section Name : <undefined>
Section Type : SHELL
Formulation : 2
Thickness : 2.375000
#int points : 0
Hourglass Type : Standard LS-DYNA viscous
Part Mass : 0.0829556
Added Mass % : 0.000000

4.7 Predictive Picking and Menu "Hover Over"

"Predictive picking" highlights what would be picked were you to left-click with the mouse.

"Menu Hover Over" highlights items in menu lists, helping you to identify what they are.

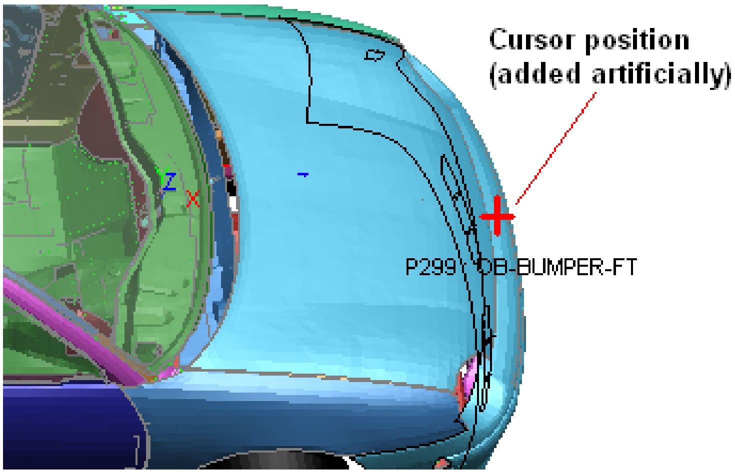
4.7.1 Description of Predictive picking.

From PRIMER 10.0 onwards all screen-picking operations have "predictive picking" enabled by default. This means that when you move the cursor into the graphics window and position it over something pickable in the current context, the item in question will be highlighted by sketching and labelling it, identifying what would be selected were you to perform a left mouse click at that position.

In this example the cursor (red cross added artificially here) has been hovered over the front bumper of a vehicle model.

The current mode is the default "Quick pick by part", so the part making up the bumper has been sketched in free edge mode, and labelled with its id and title, here "P299 OB-BUMPER-FT".

The sketching used to highlight items is transient: it will disappear as you move the cursor away from the object in question, and there is no need to refresh the graphics window to get rid of it.



In the example here the current pick mode was "Quick pick by part". Predictive picking is always associated with the current picking operation, so for example if you chose **[Keyword] Element Shell, Modify** then the current picking mode would be to select a shell, and predictive picking would change to highlighting shells under the cursor.

4.7.2 Controlling Predictive Picking

Most of the time Predictive Picking is helpful, but there are two situations in which you might want to turn it off:

1. If your computer is very slow, or you are displaying graphics over a network, you may find that the need to keep updating the display as the cursor position moves makes the response sluggish.

2. If your image is very complex, and you are picking items which generate a lot of extra graphics when they are highlighted (typically sets, or contacts defined by set) you may find that predictive pick highlighting becomes a nuisance.

In the first situation you might want to turn it off for all picking operations; but in the second you may just want to suppress it for the duration of the current pick operation, turning it back on when you revert to picking items that are less visually complex. Therefore two levels of control are provided:

Switching on/off temporarily for this picking operation only.

The **[PP]** button to the right of the "Quick Pick" selection buttons can be used to toggle predictive picking on/off ***for the current picking operation only.***

As an alternative you can use the "**p**" (note lower case) keyboard short-cut to have exactly the same effect.

This only affects the current picking operation, and the setting is "forgotten" once that operation ends.

Special case of Predictive Picking and contact surfaces

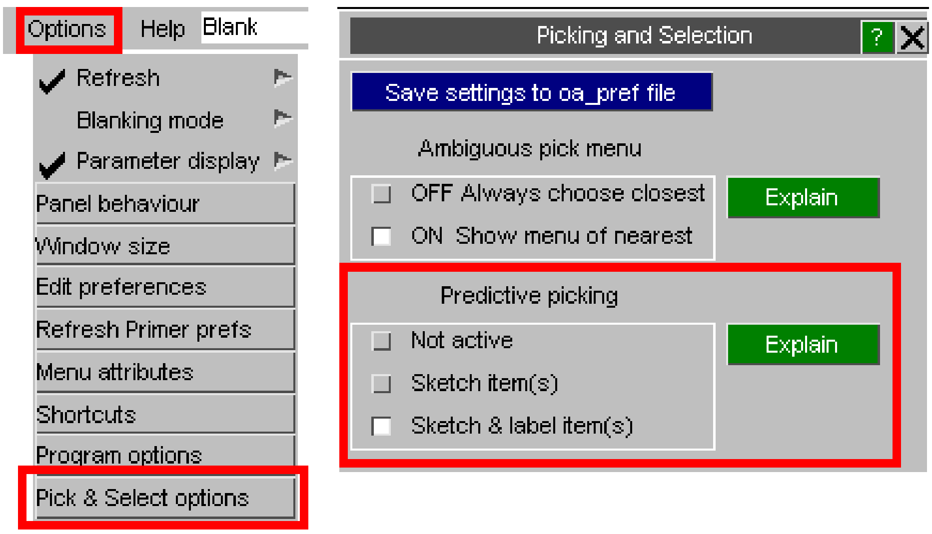
Experience has shown that the combination of Predictive Picking and contact surfaces is not helpful. Most contact surfaces are defined by sets, often sets of many parts, and it is sometimes the case that the whole model will have been placed in a global contact.

As a result Predictive picking of contacts tends to select many items, and if the whole model is in a contact it will always highlight the whole model - which is a hindrance and not a help!

As a consequence there is a special exception in the case of the [Keyword] **Contact** panel, where predictive picking is off by default. This is equivalent to disabling it temporarily by the **[PP]** button or the "**p**" shortcut whenever this panel is entered, and it can be re-enabled by either of these means if desired.

Switching on/off globally, and controlling what is displayed.

[Options]> Pick & Select options maps the **Picking and selection** panel:



Here you can choose from three possible modes for Predictive picking:

Not active	Turns Predictive Picking off globally for all picking operations.
Sketch item(s)	The item(s) in question are sketched; usually in free edge mode, but the exact sketching method depends upon what is being displayed.
Sketch and label item(s)	The item(s) are labelled as well as being sketched. Labelling is generally at the item's visual centre. (In this context "visual centre" means its average coordinate, which may not be its true centre of gravity.) Note that you can't have "label only", ie label without also sketching.

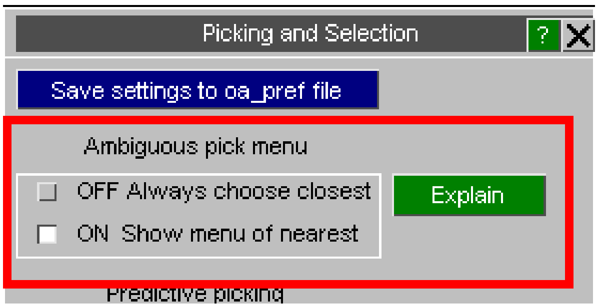
The current setting (along with all others in this panel) can be saved to the oa_pref file using **Save settings to oa_pref file** so that it is remembered for future sessions of PRIMER.

It is also possible to toggle programme-wide predictive picking on/off using the "**P**" (note upper case) keyboard short cut. This is equivalent to selecting Not active, or reverting to the current setting, in the panel above. However it is not "remembered" in any away, so a future PRIMER session will revert to the default behaviour as (possibly) modified in the oa_pref file.

4.7.3 Ambiguity and Predictive Picking

PRIMER has two possible ways of handling ambiguity during screen-picking operations. In the `Picking and Selection` panel shown above you can choose whether or not to map the Ambiguous selection menu using the options:

OFF: always choose closest	Picks always select the closest item without any further intervention from the user.
ON: Show menu of nearest	A list of possible candidates sorted by distance from the pick point is shown, and the user is invited to choose which is to be used.



Predictive Picking works with this setting as follows:

- When the ambiguous menu is turned **OFF** Predictive picking will only ever show the closest item.
- When the ambiguous menu is turned **ON** Predictive picking will show all possible candidates if the current cursor position would result in an ambiguous selection.

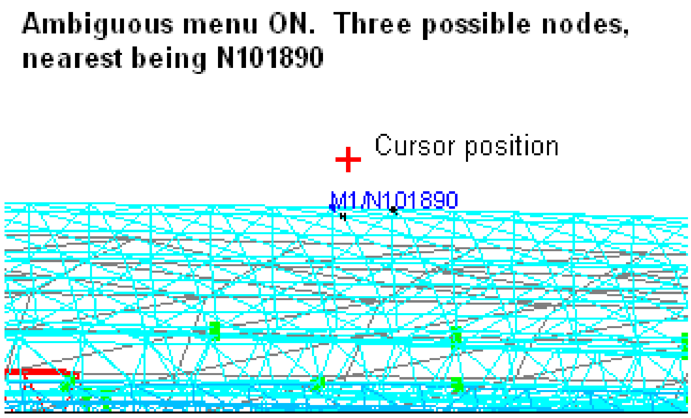
  In this second case only the closest item is labelled (assuming that labelling is active), and it is drawn in colour (yellow or blue depending on the background colour). All other potential candidates are only drawn, and in the current sketch colour (black or white depending on background colour).

This is illustrated in this example.

Here the display mode is LINE (ie no hidden surface removal), we are currently picking Nodes, and the cursor has been positioned just outside the mesh.

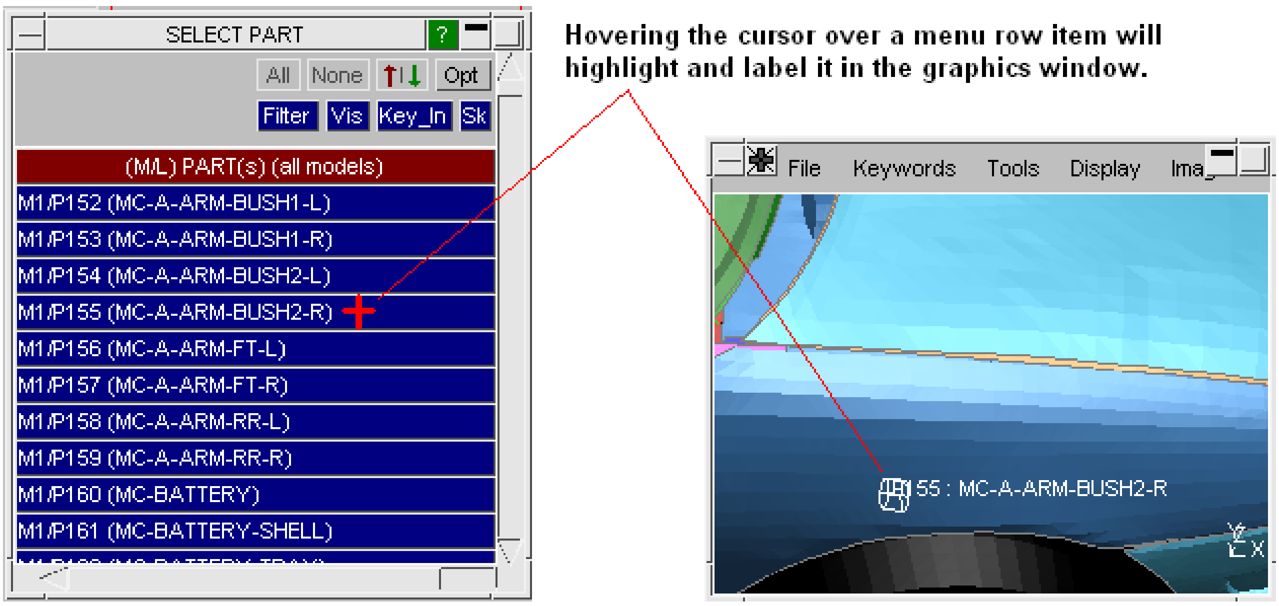
Three possible candidate nodes have been identified and highlighed, but only the nearest (M1/N101890) is labelled.

If the ambiguous menu was OFF then only N101890 would have been sketched and labelled.



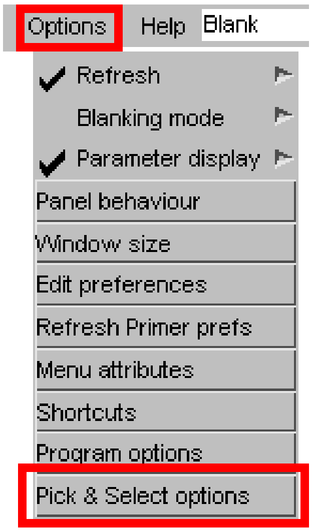
4.7.4 Description of Menu "hover over" highlighting

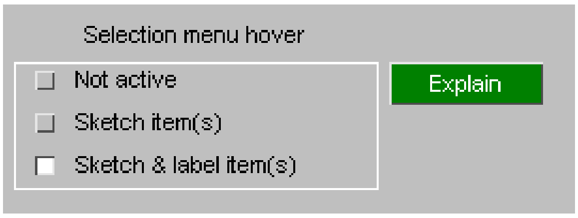
Menu "Hover over" highlighting is very similar to Predictive picking. Whenever PRIMER builds an "object menu" showing a list of items then hovering the cursor over a menu row will highlight and label that item on the screen.



Controlling menu "hover over" highlighting

[Options]> Pick & Select options maps the `Picking and selection` panel:





Here you can choose from three possible modes for menu hover over:

Not active	Turns hover over off
Sketch item(s)	The item(s) in question are sketched; usually in free edge mode, but the exact sketching method depends upon what is being displayed.
Sketch and label item(s)	The item(s) are labelled as well as being sketched. Labelling is generally at the item's visual centre. (In this context "visual centre" means its average coordinate, which may not be its true centre of gravity.) Note that you can't have "label only", ie label without also sketching.

The current setting (along with all others in this panel) can be saved to the oa_pref file using **Save settings to oa_pref file** so that it is remembered for future sessions of PRIMER.

5 Keywords

5.0 Index to keywords

***AIRBAG**	Airbags (CVs) and interactions
***ALE**	Arbitrary Lagrangian Eulerian
***BOUNDARY**	Boundary conditions
***CONSTRAINED**	Constraints
***CONTACT**	Contacts (Sliding Interfaces)
***CONTROL**	Control card processing
***DAMPING**	Damping cards
***DATABASE**	Database output control
***DEFINE**	Define cards
***DEFORMABLE_TO_RIGID**	Part state switching
***ELEMENT**	All element types
***EOS**	Equations of State
***HOURGLASS**	Hourglass control cards
***INITIAL**	Initial conditions
***INTEGRATION**	Integration rules
***INTERFACE**	Interface cards
***LOAD**	Applied loading
***MATERIAL**	Structural & thermal materials
***NODE**	Nodes (grid points)
***NODE_TRANSFORMATION**	Node transformations
***PARAMETER**	Parameters
***PART**	Parts
***PERTURBATION**	Perturbation
***RAIL**	Rail cards
***RIGIDWALL**	Rigid ("stone") walls
***SECTION**	Section processing
***SENSOR**	Sensor cards
***SET**	Set processing of Beam, etc types
***TERMINATION**	Termination settings

5.1 Keywords

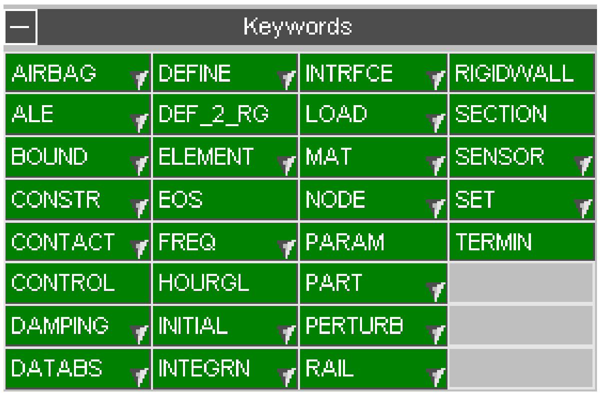
PRIMER allows you to create, modify, list and delete the constituent parts of an input deck, and this is done by object category, ie "Keyword".

Keywords are chosen from the **Keywords** menu which contains all major keywords.
Most keywords have sub-categories, the indicator that a popup menu can be invoked is: 

Once an item has been selected the top-level menu panel for that item will be invoked: see section 5.1.1 below.

There is no limit to the number of different keyword manipulation windows that can be current at any time: for example you can edit concurrently as many PARTs as you like.

Keywords present in the LS-DYNA manual but not in PRIMER's Keyword panel cannot be viewed or edited in PRIMER. However, they are read in, stored within PRIMER and written out, so no valid LS_DYNA data is lost. Further, the implications of actions such as renumbering or deleting parts of a model will be correctly applied to those "hidden" keywords.

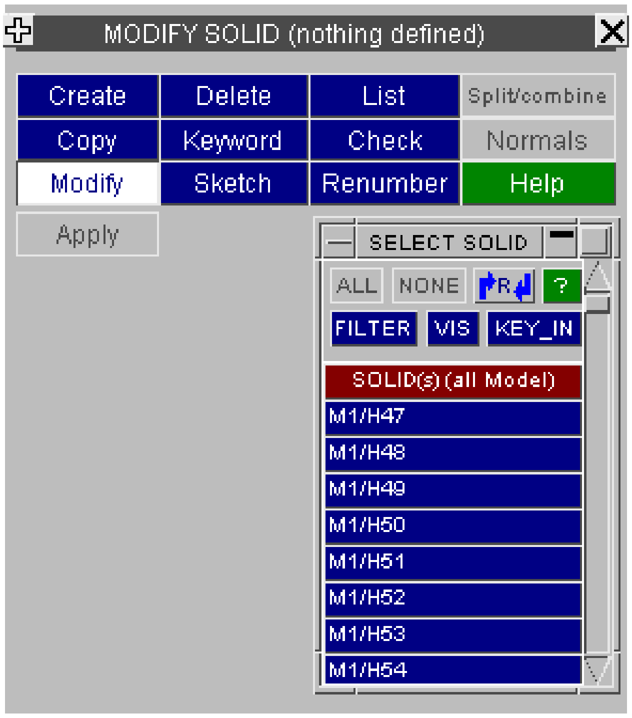


5.1.1 Standard Keyword top level menu options.

Most of the **KEYWORD** options have a set of standard options. The exact contents may vary slightly with context, but the basic functionality is the same in all cases.

This figure shows a standard display, here for **ELEMENT_SOLID** keywords, but it is the same for any item type. Currently Modify is the selected option. Selecting any of the other buttons will switch to that option.
These "standard operations" options are:

Create	Create a new item
Copy	Copy existing items
Modify	Edit (modify) an existing item.
Delete	Delete one or more existing items.
Keyword	Invoke the generic keyword editor
Sketch	Sketch existing items on top of the current image.
List	List a summary of the contents of existing items.
Check	Check items for errors.
Renumber	Change the labels of items.

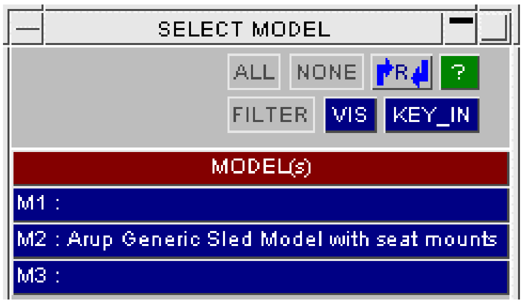


Not all options will be available in all contexts, for example **Create** and **Modify** will not be available where specific editing/creation functions do not yet exist in PRIMER, and not all types have a **Keyword** editor.

Operations requiring an explicit "Parent" model id.
If you have more than one model in memory, and the operation in question requires an explicit "parent" model id, then you will be forced to select a model prior to the operation taking place.

For example when you **CREATE** an element it must exist in one model only so, in the multiple model case, you will be asked to select which model to use.

If only one model is current it will be used automatically and this selection stage will be skipped

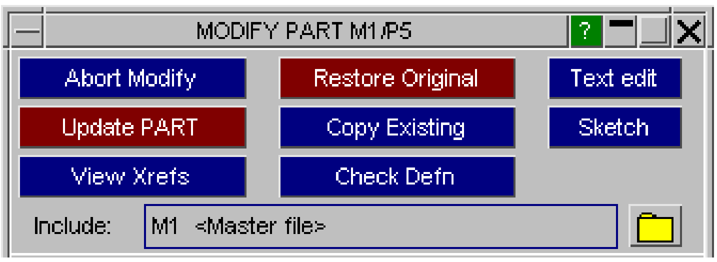


.

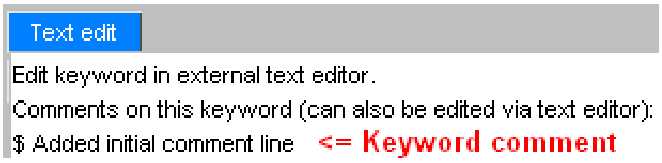
5.1.2 Standard "static" header for CREATE and EDIT functions

The Create/Edit panels for Keywords differ in detail, but share a common layout of the "static" functions at their top.

This example is taken from a Part editing panel, but the top buttons are the same in all cases.



The standard buttons act as follows. Note that some have vary between "Create" and "Edit" modes, whereas others are common to "Both":

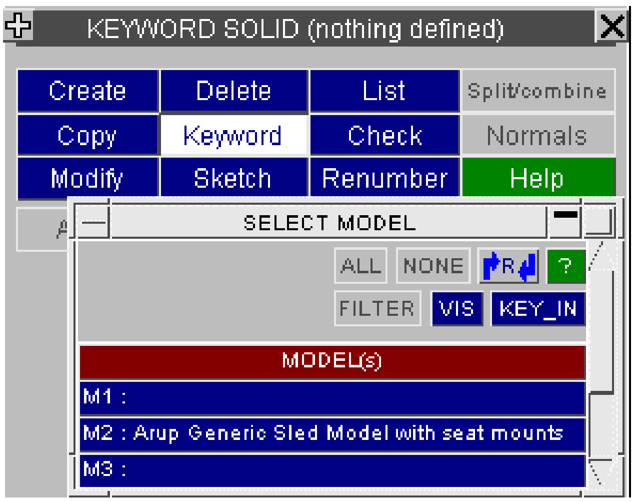
Abort_item	(Both)	Terminates the current operation, leaving the permanent definition unchanged (edit case) or undefined (create case).
Reset All **Restore Original**	(Create) (Edit)	Resets the definition to null, canceling any entries made so far. Restores the original, unedited definition (copied from the permanent definition), overwriting any changes made so far.
Text edit	(Both)	Writes a "mini keyword output" file of just this keyword, and opens the system standard text editor on this file. See Text Edit below for more information. This will also also display any "keyword comments" associated with this keyword as hover text when the mouse is hovered over the button. You can tell if any comments are present since the **Text Edit** button will be drawn in White on Light blue instead of the standard white on dark blue. 
Create item **Update item**	(Create) (Edit)	Creates a permanent entry from the scratch definition. Overwrites the "old" permanent definition with the revised entries from the scratch definition.
Copy existing	(Both)	Copies entries from an existing definition into this one, superseding any entries or changes made so far.
Sketch	(Both)	Sketches the currently defined scratch definition on top of the current graphics image.
View Xrefs	(Both)	Invokes cross-reference viewer.
Check Defn	(Both)	Checks the current scratch definition for errors

Remember: All Create and Edit functions ***always*** take place on a "scratch" copy of the current definition (if any). No changes are made to the permanent database until the **CREATE** or **UPDATE** buttons are used.

5.1.3 The generic KEYWORD editing panel.

There are many places in PRIMER where an explicit create/edit panel is not necessary, and a generic "keyword editor" will suffice. The "Keyword editor" has the additional advantage that it lists all items of a particular type, allowing multiple edits to be carried out with a single command.

This is invoked by the **KEYWORD** tab, as shown here, from the standard options available once a selections has been made from the Keywords panel.



The Keyword Editor has been completely rewritten for PRIMER release 9.3 to improve its functionality and make it easier to use. In particular:

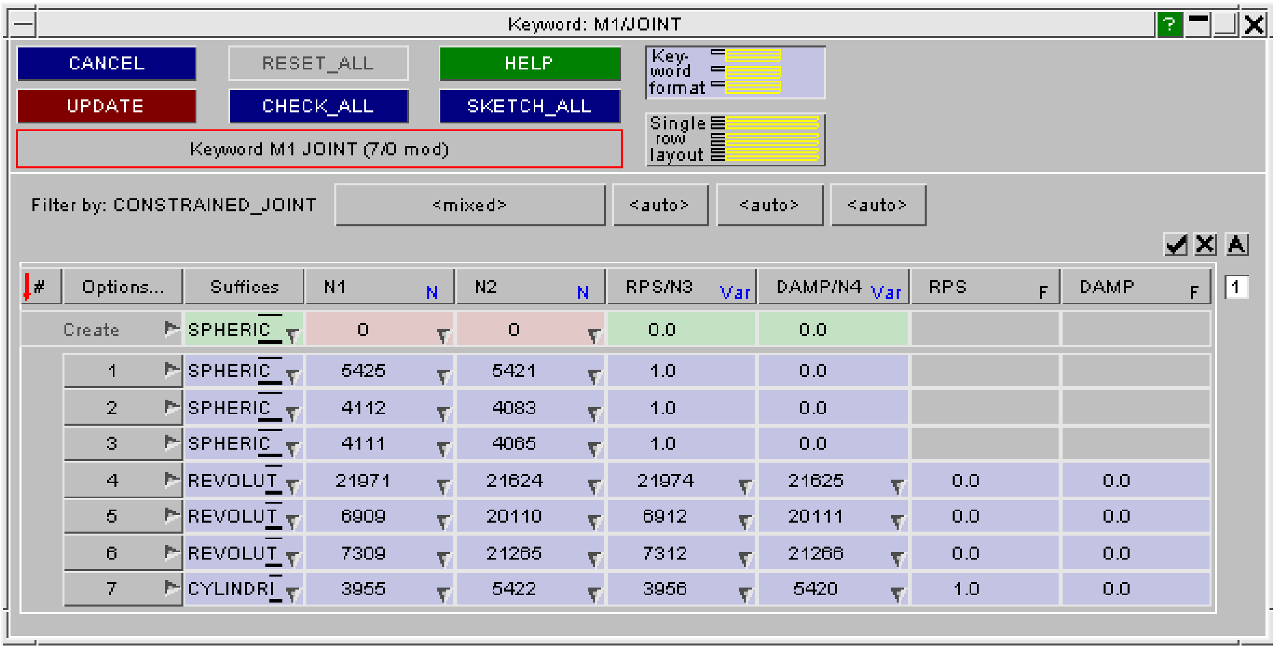
- The separate **Edit**, **Insert** and **Delete** modes of the original version have been superseded, with all this functionality now available in a single operating mode.

- The limitation that only a single keyword sub-type could be displayed at a time has gone: it is now possible to display entities of all sub-types of a master keyword together, combining any permutation of sub-keyword suffices.

- Following from this it is now possible to insert, remove or change the sub-keyword suffices for the entities shown.

- The display and functionality of the editor have both been improved, including sorting by column and "intelligent" editing of data fields over multiple entries.

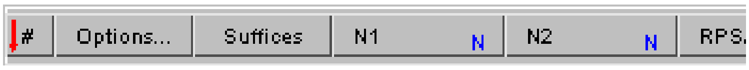
The following examples use the **`*CONSTRAINED_JOINT`** keyword to illustrate the new editor. In this model there are a variety of different joint types, and the figure below shows **`_SPHERICAL`**, **`_REVOLUTE`** and **`_CYLINDRICAL`** types simultaneously.



Terminology

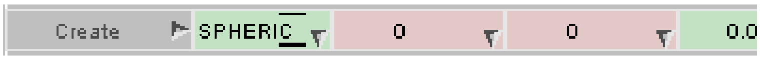
Various areas of this panel are referred to below:

The Acronym row: One or more rows of headers showing the acronyms for each field.



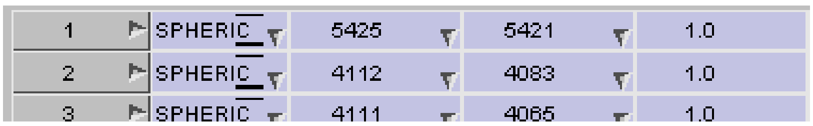
Hovering the mouse over a button will give more information about the data. Clicking on a button (other than **Options...**) will sort the data rows by that column.

The Entry row: One or more rows on a green background to enter new data.



This row will initially be blank: you must type in or select data to populate it, then use **Create** to create the definition and store it in the database.

The Data rows: Rows of existing data on a blue background.



Entries in these fields can be edited by over-typing them or by selecting new values from popup selection menus.

Multiple rows may be edited simultaneously by selecting the rows and then changing the required data field on any row to propagate its change to all other selected rows.

Use of button background colours

The keyword editor uses a variation on the standard button colour scheme used elsewhere in PRIMER in order to distinguish between inactive and active / selected.

In all cases the colours are dark text on a light background when either the data field or the whole row are not active, and invert to become light text on a dark background (the standard PRIMER colours) when they become active.

Green is used on the **Entry row** to denote data fields that are grammatically correct in their current state	Inactive	Active data entry
Blue is used on the **Data rows** to denote data fields that are grammatically correct in their current state	Inactive	Active data entry
Red is used on all rows to denote a field that is either invalid, or empty and requires population with data	Inactive	Active data entry
Cyan is used on all rows to denote references to latent (referenced but undefined) items	Inactive	Active data entry

Filter by: controlling what is shown in the panel.

`*CONSTRAINED_JOINT` has four possible sub-keywords:

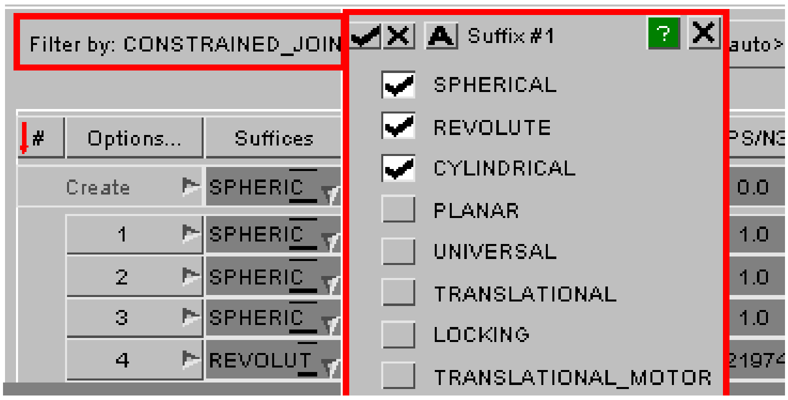
1: Joint type	2: Force output option	3: Failure option	4: Label & title option

`_SPHERICAL`
`_REVOLUTE`
etc
	<none>
	`_LOCAL`
		<none>
		`_FAILURE`
			<none>
			`_ID`

To see the keyword suffices for each of the buttons above hover the mouse over them, and a popup window will display the relevant options.

To control what is actually displayed in the editor click on the appropriate column and select the suffices to be shown.

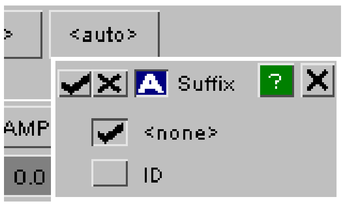
By default the `[A]` option, for "Automatic" will be selected for all suffices, causing all sub-types of this keyword in the model to be selected automatically for display.



This model contains the first three joint types, and also some `_TRANSLATIONAL` ones; but the last of these has been deselected meaning that these joints are not shown in the image above.

Many sub-keywords are optional, for example `_ID` in this context, and the alternative is for that sub-keyword to be omitted altogether.

In this situation you will be given the choice of that keyword or `<none>` as shown in the popup here for the `_ID` column.



This process of selection may be carried out for all sub-keyword columns, and what is shown in the editor rows below is the logical AND of the selected keyword suffices.

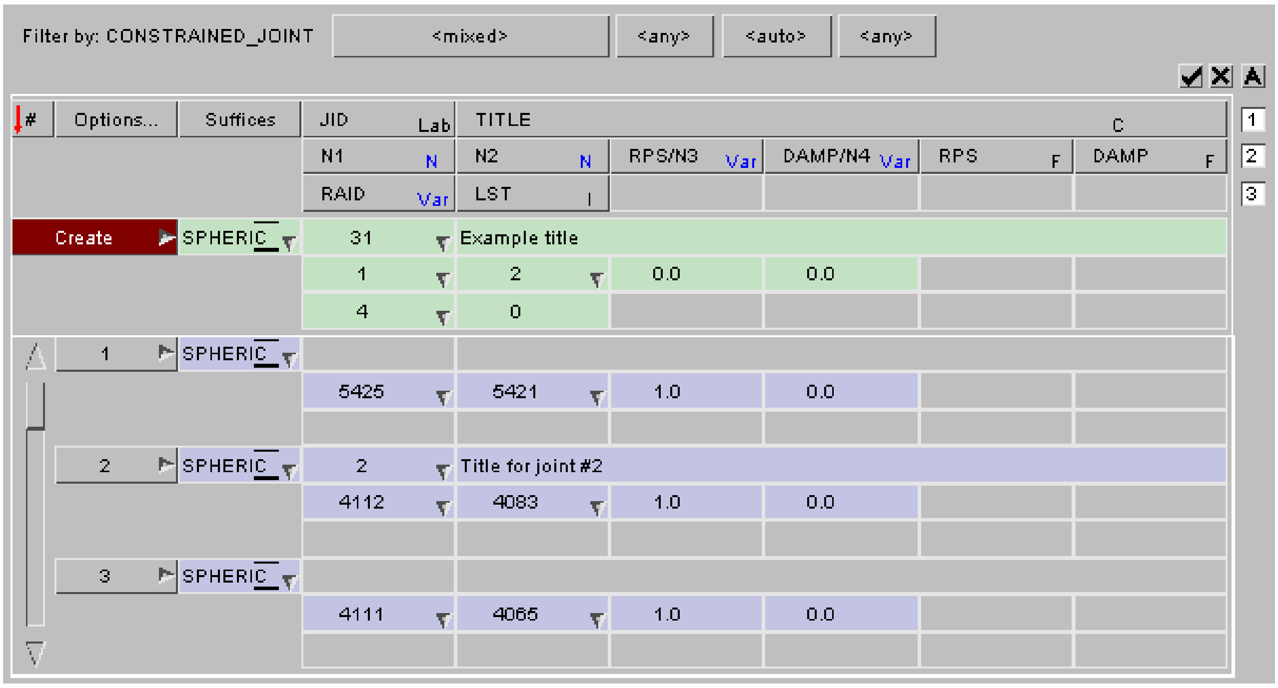
You can change what is shown at any time, and the effect is only to change what is shown in the editor rows below. No change is made to the actual keyword definitions themselves.

Displaying data for different sub-types

The display options above make it possible to display entries requiring different numbers of rows, and for row/column fields to have different contents and data types - or even to be absent.

The example above has been modified so that:

- Both **_LOCAL** and **_ID** options are displayed
- The top entry (green) row has been given notional values
- The second existing joint (blue row 2) has been given the **_ID** suffix and an explicit label and title



This presents several display problems:

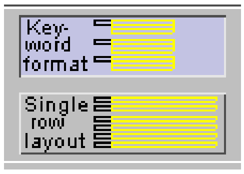
- The **_ID** row (**JID** and **TITLE**) row is undefined for existing joints #1 and #3
- The **_LOCAL** row (**RAID** and **LST**) is undefined for all existing joints.

It can be seen from the figure above that this problem is solved by simply "greying out" the data fields that are not relevant.

Changing the display layout

As the example demonstrates shows turning on lots of suffix options can results in the table rows expanding and containing a lot of empty grey rows as a consequence. It is often the case that the data fields of interest are on the early rows of a keyword, for example Joint nodes appear on the first line, and while it may be necessary to display lots of suffices in order to see all variants one is only interested in a few rows of data.

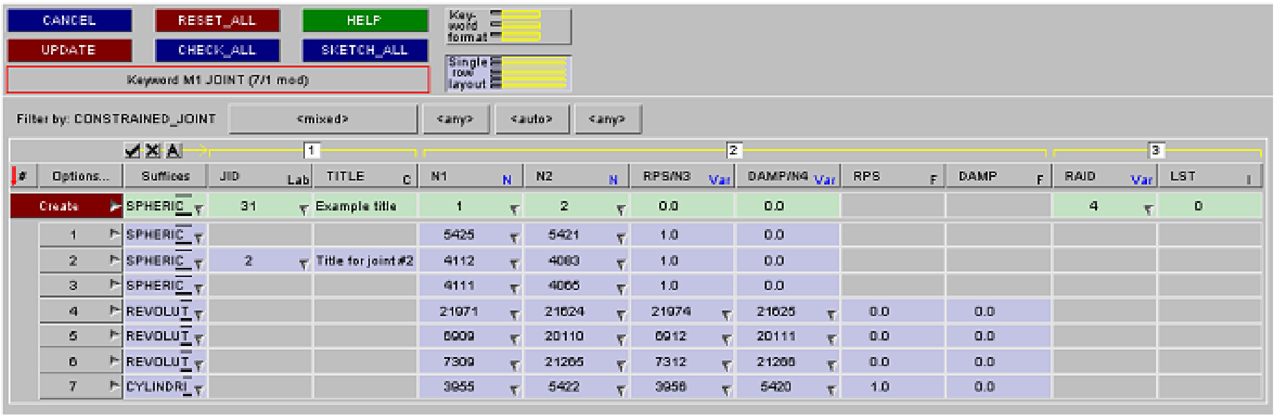
One solution to the problem is to change the layout of the editor rows:



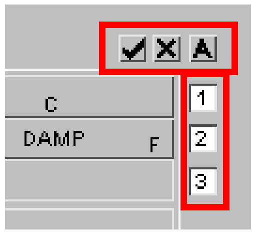
Keyword Format is the default, shown in the figures above, that mimics the LS-DYNA keyword row and column layout.

Single Row Layout is the alternative which condenses each item onto a single line by concatenating the rows.

The image here, reduced in size to fit the page, shows how the three rows have been concatenated so that each item only uses a single line. Empty fields are still greyed out, but they no longer affect what is displayed so severely. Note also that "wide" fields such as **`TITLE`** (column #2) are compressed into a standard with column in this display mode.

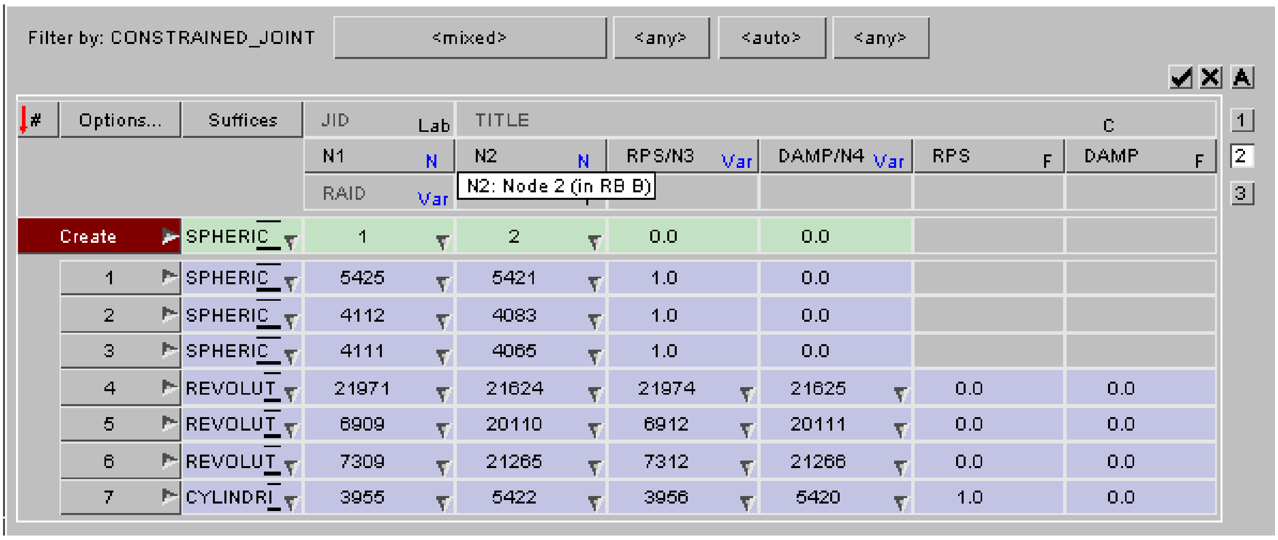


Limiting the rows displayed



An alternative solution to the "too much empty space" problem is to turn off the display of selected rows. In Keyword Format there is a numbered button to the right of each acronym row at the top of the panel, and this may be used to de-select a given row for display in the table below.

In the example below rows #1 (**`ID`** and **`TITLE`**) and #3 (**`RAID`** and **`LST`**) have been turned off leaving only row #2 containing nodes etc.



This demonstrates that the "acronym" rows are still present, albeit greyed out, but that only the selected data rows are shown thus condensing the display.

The **[A]** button, "Automatic", will turn back on all rows containing active data.

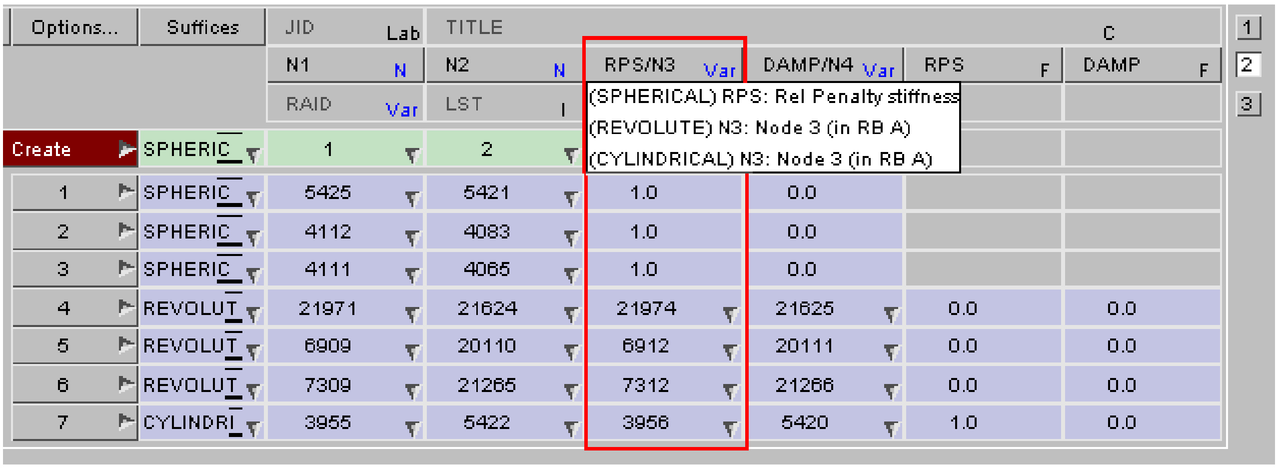
These row selection buttons are also available in **Single Row Layout** format where they work in the same way by eliminating the columns of the selected rows.

Mismatched data in row/column fields.

In the examples above we have:

Entity type	**1**	**2**	**3**	**4**	**5**	**6**
SPHERICAL joints, data fields	**N1**	**N2**	**RPS**	**DAMP**		
REVOLUTE joints, data fields	**N1**	**N2**	**N3**	**N4**	**RPS**	**DAMP**
CYLINDRICAL joints, data fields	**N1**	**N2**	**N3**	**N4**	**RPS**	**DAMP**

Therefore column #3 may contain **RPS** or **N3**, and column #4 may contain **DAMP** or **N4**. This is handled as follows:

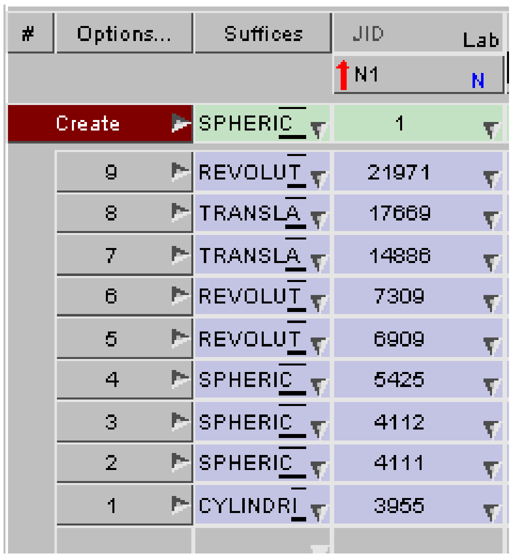


The acronym header button shows the various entries, here **RPS / N3**. In addition hovering the mouse over that button, as shown here, displays the details of that row/column contents by suffix type.

The data rows contain the relevant data. Here the first three entries for **SPHERICAL** joints contain an **RPS** value of 1.0, and the remaining rows show **N3** node values.

The keyword editor always "knows" the type of the data in a given field, and processes it accordingly. This is significant when multiple rows are edited as described below.

Sorting rows by data field

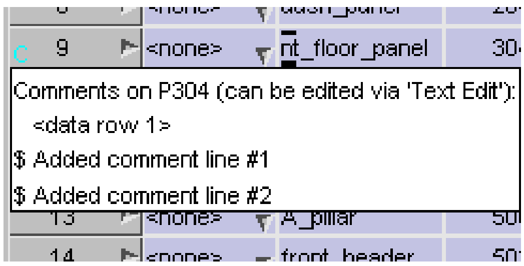


By default editor rows are sorted in ascending order by index (column # at the top left) but it is possible to sort using any data field by clicking on its acronym button. The first click on a button will sort by ascending order and subsequent clicks will reverse the sort order, the current direction being shown by a small red arrow on the button.

In this example rows have been sorted in descending order by node **N1**.

To revert to the conventional "sort by index" click on the **[#]** button at the top left.

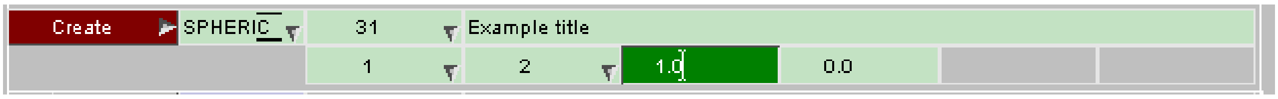
Comments on keywords



If a keyword contains embedded comments a light blue C will be shown on its row button, and hovering the cursor over that button will list the comments on that keyword.

Comments can be added, modified and deleted using the **Text edit** capability.

Creating a new definition in the Entry row

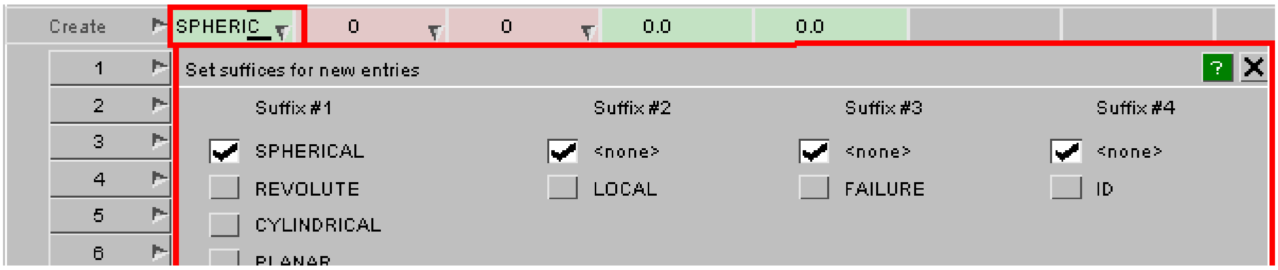


The green row at the top of the list of existing items is where data is inserted to create a new definition.

- Type in, or select from popup menus, sufficient data fields to populate the definition.
- Use **Create** to make the definition. This installs it in the database, and it will also appear in the **Data rows** below.

Changing the Suffix of the Entry row definition.

To change one or more keyword suffices right click on the field in the **Suffix** column (here `[SPHERIC v]`), and choose the revised suffices. (The image below has been truncated vertically.)

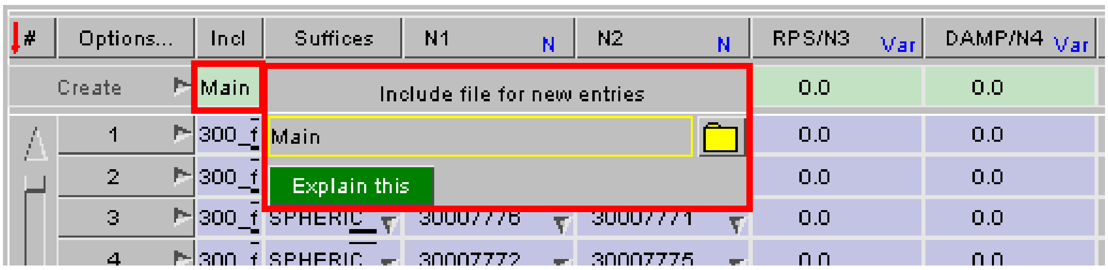


In this example there are four columns of suffices to choose from, other keywords will be different. You can only tick one entry in each column since, obviously, entries within a column are mutually exclusive.

In most cases column suffices are independent but there are a few cases where changing one suffix may affect what is legal in other columns. One example is `*RIGIDWALL` where further suffices on the `_PLANAR` suffix are not compatible with the `_GEOMETRIC` suffix. In these situations any illegal combinations will be removed, and the relevant buttons greyed out.

Setting the Include file for the new entry

If include files are present in the model then there will be an extra **Incl** column between **Options...** and **Suffices**, and each entry in the table will show its include file name.



By clicking on the entry (here the `Main` file) a small sub-panel for selection of an alternative include file will be mapped.

The Keyword editor will always initialise itself to create new entries in the current include file for this model, but if you change this then subsequent new entries will be in the selected file. Include files in PRIMER are described in more detail in section 3.13

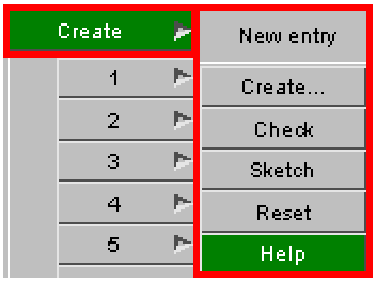
Using **Create** to make the entry.

The **Create** button will be one of three colours:

Greyed out	This means that the row fails the "grammatical" check and the entry cannot be created. One or more the data fields will be red and must be populated or corrected.
Red	This means that the row passes the "grammatical" check, ie no red data fields, but that the standard "Check" function has found one or more errors. The **Check** option on its popup menu can be used to list these errors. The entry can still be created, but you will be warned about the errors and may have to correct them later.
Green	The row passes both grammar and contents check, and can be created with no error or warning messages.

The popup options on the **Create** button

These options may be use to manipulate the **Entry row** as follows:



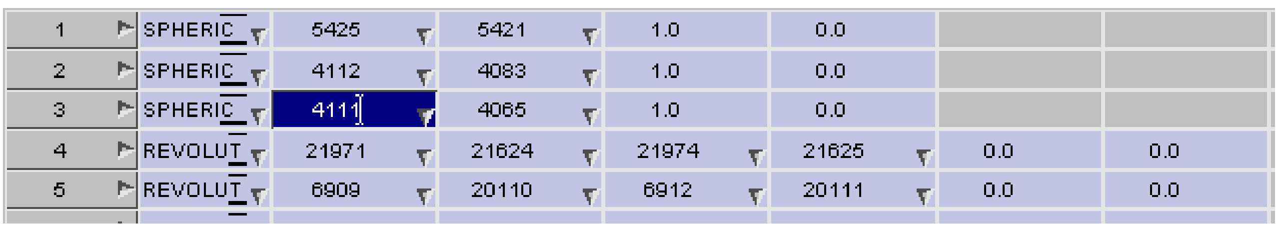
Create... Maps the standard Create/Edit panel for this item. When you exit from this the saved definition will be used to populate the **Entry row**. (This option will be greyed out if a create/edit function has not been written for the current data type.)

Check Runs the standard check function on this definition and reports any errors.

Sketch Sketches the definition in its current form on the model

Reset Resets the **Entry row** to its default (empty) state.

Changing an existing definition in the Data rows



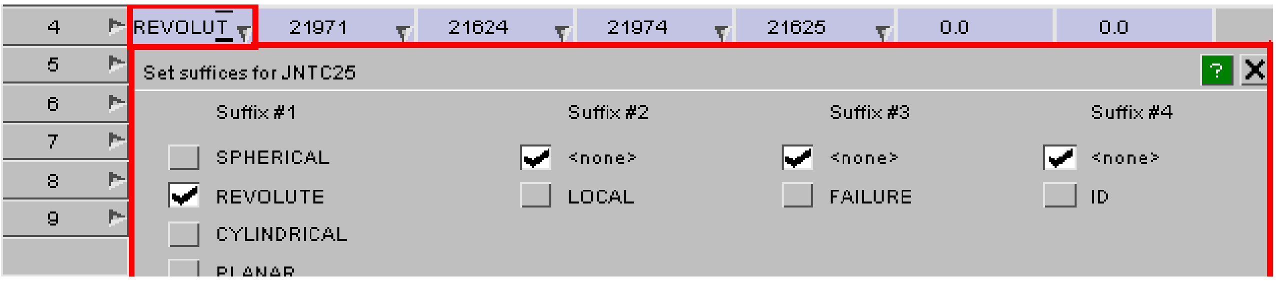
Simply overtype the existing entry, or select a new value from the standard popup selection menu.

Changing a definition in the **Data rows** does two things:

- A backup of the existing definition (before the change) is made, so that the original definition can be restored if required.
- The current definition in the database is changed immediately

Changing the Suffix of a Data row definition.

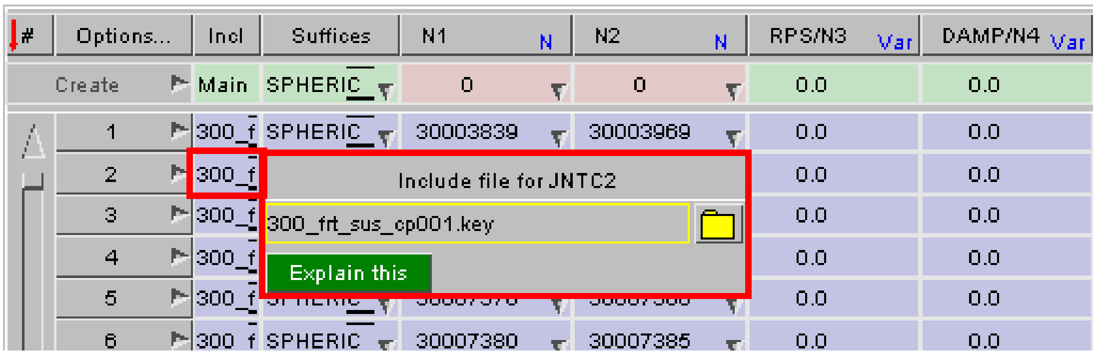
To change one or more keyword suffices right click on the field in the **Suffix** column (here **`[REVOLUT v]`**), and choose the revised suffices. (The image below has been truncated vertically.)



This popup and its usage are identical to that described under the **Entry Row** above.

Changing the Include file of a Data row definition

If include files are present in the model then an extra **Incl** column will be shown, and the name of each entry's include file will be listed. This is truncated to a narrow column width to save space, but if you hover the mouse over an entry the full include file details will be given.

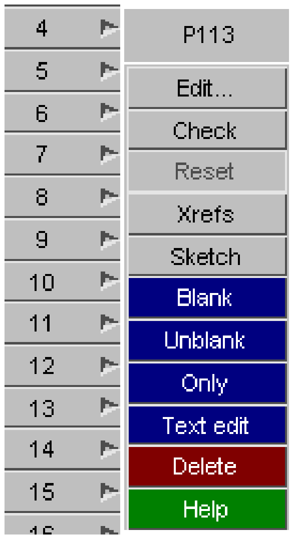


To change an entry's include file click on its data field, and the selection popup shown above will be mapped allowing you to select a different include file.

Warning: Include file changes are ***not*** undone by a **Reset** command

Because of the way include file membership is handled in PRIMER moving a **Data row** entry to a new include file is not reversible by a **Reset** command. The only way to revert to its original include file is to reset it explicitly using the <click> + <select new> process above.

The popup options on Data row Index buttons.



Each **Data row** has the following popup menu options:

Edit... Maps the standard Create/Update panel for the current definition. When the edits are saved the Data row will be updated.

Check Runs the standard check function on this Data row

Reset Resets this Data row back to its original condition (before any edits, not just the most recent one.)

Xrefs Maps the standard cross-reference viewer panel for this item.

Sketch Sketches this item on the current model.

Manipulating blanking using these options

These three commands act immediately, there is no need to update the display to see the changes.

Blank Blanks this item from the current display.

Unblank Unblanks this item in the current display.

Only Makes this the only item visible in the current display

Text edit: editing definitions in an external editor

The external text editor works in exactly the same as for scalar editing panels (see description above) in that it performs "mini keyword output" operation to write a keyword file containing data for the selected row(s), and then performs a "mini keyword input" to read the file back in again and to update the model.

Because the keyword editor permits operations on multiple items Text edit in this context does the same, reading in all definitions found in the edited file that match the current keyword type. Definitions with the same label as existing items replace these, and items with new labels are added to the model.

Using **Text Edit** to import new keywords

From V10.1 onwards **Text Edit** in the context of the keyword editor may also be used to import new definitions of the type being edited, as well as modifying existing ones. Therefore it is legal to edit a given row (or rows), and within the external editor to create or import new definitions of this keyword type. On exit from the external editor the new items will be imported into PRIMER exactly as if they had been read by conventional keyword input, and added to the current model using the current include file. The new items are added to the permanent model database, and will appear in the current keyword editor panel immediately, so long as they are not filtered out by current (sub-)keyword selection.

The addition of new keywords in this way is not reversible: a **Reset** in the keyword editor, or an **Abort** of the editor panel, will ***not*** remove these items; so if you change your mind about importing them you will need to **Delete** them explicitly.

Delete: Deleting the current definition

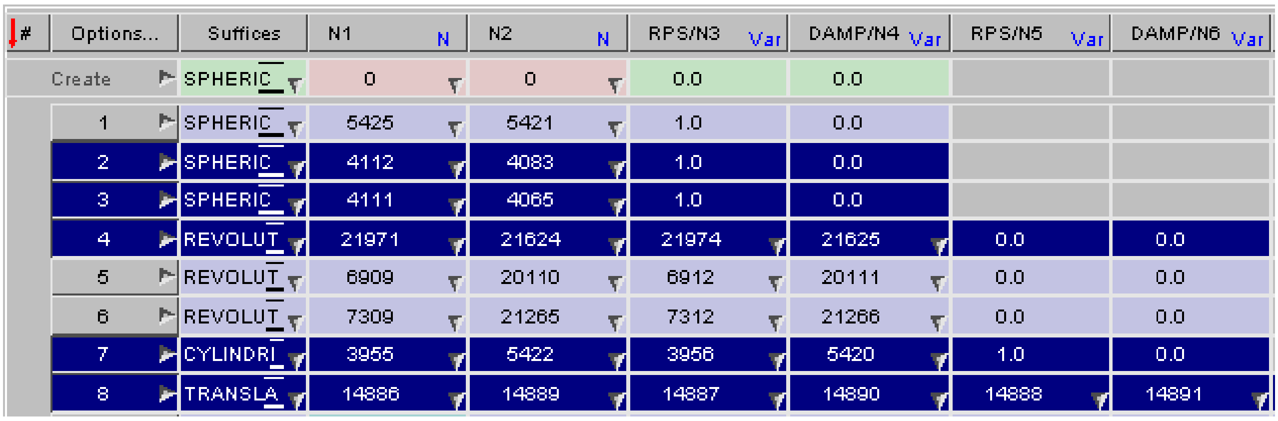
Using **Delete** will map a cut-down deletion confirmation menu for this item. If you choose to **Delete Items** the standard PRIMER deletion confirmation dialogue will be mapped and the item will be deleted.

Deletion is not reversible - once an item has been deleted it cannot be recovered.

Working with multiple rows

It is possible to select a range of **Data rows** and to change their properties, or their suffices, or to delete them in a single operation as described above for a single row.

Rows that have been selected invert their colour to a dark background, and become active for "multiple" operations. In the figure below rows 2 to 4, and 7 to 8 have been selected.

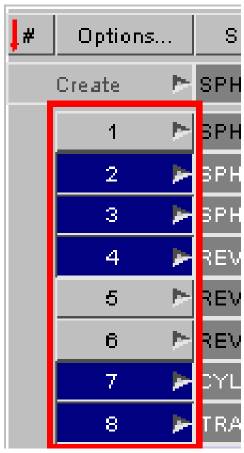


#	Options...	Suffices	N1 N	N2 N	RPS/N3 Var	DAMP/N4 Var	RPS/N5 Var	DAMP/N6 Var
	Create	SPHERIC	0	0	0.0	0.0		
	1	SPHERIC	5425	5421	1.0	0.0		
	2	SPHERIC	4112	4083	1.0	0.0		
	3	SPHERIC	4111	4065	1.0	0.0		
	4	REVOLUT	21971	21624	21974	21625	0.0	0.0
	5	REVOLUT	6909	20110	6912	20111	0.0	0.0
	6	REVOLUT	7309	21265	7312	21266	0.0	0.0
	7	CYLINDRI	3955	5422	3956	5420	1.0	0.0
	8	TRANSLA	14886	14889	14887	14890	14888	14891

Selecting a range of rows

Rows may be selected by a range of methods, which may be combined in any order:

By clicking on the row index buttons:



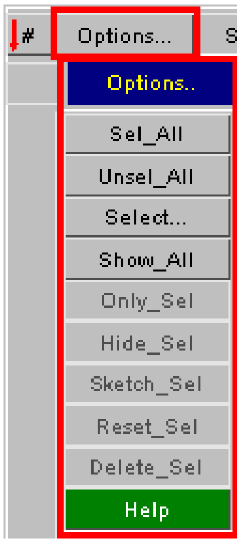
`<Click>`	Selects that row only, deselecting any others.
`<Shift + click>`	Selects all rows between the most recently clicked on and the current row.
`<Control + click>`	Inverts the selection status of the current row, leaving other rows' selection status unchanged.

From the Popup menus on the **Options ...** button

SEL_ALL Selects all rows

UNSEL_ALL Deselects all rows

Select ... Maps the standard PRIMER object menu allowing you to select items in the normal way.



Actions on selected rows

Show_All Shows all Data rows. Needed if only a subset has been displayed using the options below.

Only_Sel Shows only those **Data rows** which have been selected.

This can be useful if you have selected a small and diverse subset of a large number of items

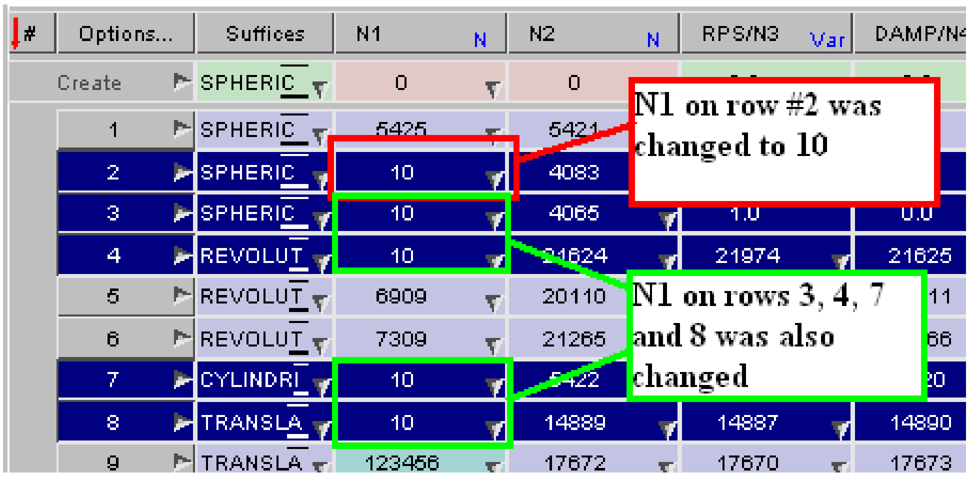
Hide_Sel The opposite of the above: shows only those **Data rows** which have ***not*** been selected.

Sketch_Sel Sketches the currently selected **Data rows** on the current model

Reset_Sel Peforms a **RESET** of all selected **Data rows**, restoring them to their original unedited state.

Delete_sel Deletes the selected **Data rows**, going through the same selection and confirmation procedures described above for deleting a single row.

Editing entries on multiple rows.



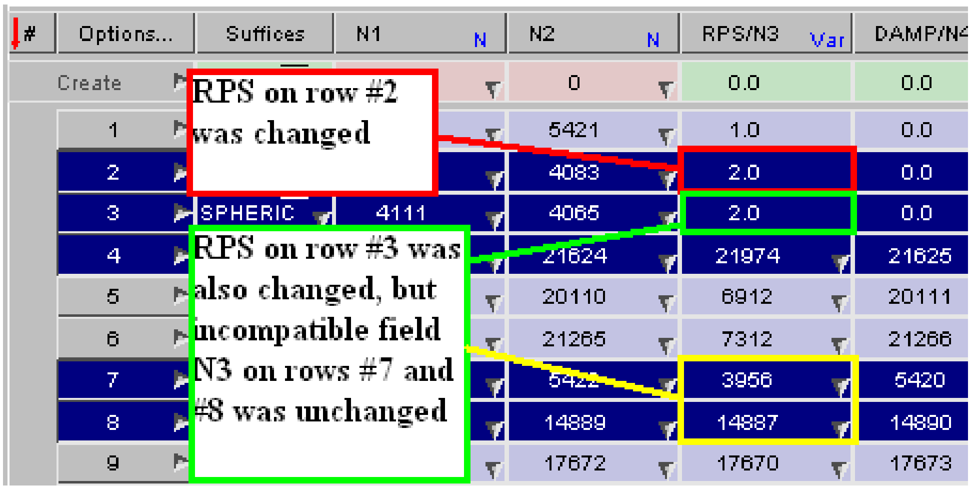
When multiple rows have been selected then editing any field on any selected row will result in the same field on all other selected rows, if compatible, being changed to the same value.

For example taking the image above, if field N1 on row 2 is changed to 10 (ie node 10), then N1 on rows 3, 4, 7 and 8 will also be changed.

Only "compatible" data are changed

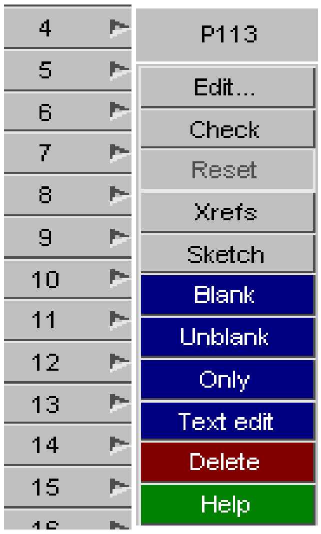
If the data in the field that is changed does not match that in the same field on another selected row, then the latter is unchanged.

In this example RPS on row #2 matches RPS on row #3, but not N3 on rows #7 and #8, so only row #3 is changed.



Popup menu actions on multiple rows.

When multiple rows are selected the popup menu on any index button works in exactly the same way as for a single row, except as described below.



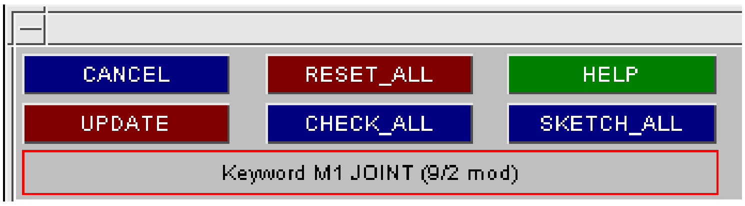
The following two options act on **this row only**

Edit...	Maps the Create/Update panel.
Xrefs	Maps the standard cross-reference viewer

The remaining actions operate **on all selected rows**

Check	Runs the standard check function
Reset	Resets to original unedited state
Sketch	Sketches on the current model.
Blank	Blanks them
Unblank	Unblanks them
Only	Draws only them
Delete	Deletes them

Saving and discarding changes



Changes made to **Data rows** in the keyword editor update the current database definitions immediately, but these changes only become permanent if and when you **UPDATE** to end an editing session. However any new entries that you have **CREATEd** will remain in the database regardless of how you exit the editor.

In more detail:

CANCEL	Undoes all edits, and exits the editor leaving all original definitions unchanged
UPDATE	Exits the editor making all changes permanent
RESET_ALL	Undoes all edits (equivalent to a **Reset** on every modified **Data Row**), returning all rows to their original state.
CHECK_ALL	Runs the standard checking function on all **Data rows** and reports the results.
HELP	Provides a text summary of how the keyword editor works.
SKETCH_ALL	Sketches all **Data rows** on the current model

General rules in the Keyword editor:

- Only one instance of a Keyword editor may be active at any one time on a given Model/Keyword combination. This is because changes made act upon the true definition in the database, not a scratch copy, therefore multiple instances would permit changes to conflict with one another.

  A Keyword editor may be used as the output of another command, for example to list results of a **Check**, or from the **Quick Pick** selection menu. Such usage counts as an "instance", and will also prevent a second panel being mapped.

- If you open a separate editing panel on an item in the **Data rows** outside the keyword editor (eg by **Keyword, *<item type>*, Modify**), make changes in that panel and then save it, the current row in the editor will only be updated the next time it is drawn. There is no interlock between these two methods of editing, and in particular the keyword editor does not "know" that one of its **Data rows** may have been changed externally, so it will not make a backup definition if one does not already exist.

  This method of working will not produce conflicts within PRIMER, but it does have the potential to cause confusion for you, the user. If you want to invoke a standard editing panel on an item in the **Data rows** it is better to use the **Edit...** option on the popup menu attached to its row index button. This will map exactly the same editing panel, and if the definition is changed it will also update the relevant **Data row** and create a backup, making it possible to undo changes.

- There are a few specific instances where a keyword editor panels retains something of the "old" layout which restricted display to certain keyword suffices. These have been retained for ergonomic reasons, and are:
  - **`*ELEMENT_SOLID`** There are options to restrict display by number of nodes on element (eg tetrahedra, wedges, hexahedra) since experience has shown that it makes sense to restrict display in this way.
  - **`*ELEMENT_SHELL`** There are similar options to restrict display, and a further option to segregate 4 noded **`SEATBELT`** elements, since these are really shells in disguise.
  - **`*INITIAL`**_xxx_**`STRESS/STRAIN`** There are options to limit display to specified numbers of integration points, both through the thickness and on plan.
  - **`*SECTION_xxx`** is split up into the different section types (**`_BEAM`**, **`_SHELL`**, etc) since it would create a total mess trying to display all section types in a single panel.

- The ***MAT**erial keyword editor limits the keyword suffices you can choose to only those material types currently in the model. If you want to create a material of a new type you won’t be able to select its keyword suffix from the sub-keyword popup, instead you will have to use [Right click on row index button] **Create** (to make a new material) or Edit (to change an existing one) and use the standard scalar editing panel to do this. Once you update from that panel the keyword editor will include the new material type in the list of keyword suffices in the keyword editor. A moment’s thought explains this limitation: with over 150 material models to choose from showing them all in the sub-keyword suffix popup menu would be totally impractical.

5.1.4 "Daisy-chain" functionality via Popup menus

The previous sections describe how to invoke and process data via Keywords.

This constitutes "ab initio" invocation of entity creation, modification, etc; where no particular context is implied. For example creating a new material in isolation (without any external references to it) means that the material may be of any type, and is not limited to types suitable for a particular class of element.

However the "Create", "Edit" and "Browse" PRIMER capabilities for all the following keywords may also be called from a given context. For example:

```
ELEMENT --(edit)---+
                   |
                 PART -- (edit) -------+
                                       |
                                       +-- MATERIAL
                                       +-- SECTION
                                       +-- EQUATION OF STATE
```

What this implies that while processing an element you might edit its Part definition and, through that, its Material, Section and Equation of State.

In this situation, referred to as "Daisy-Chaining", those items in the chain below the element have an implied context. For example the Material and Section definitions will be constrained to the element type of the Part, and you will not be permitted to add other types of element to the part.

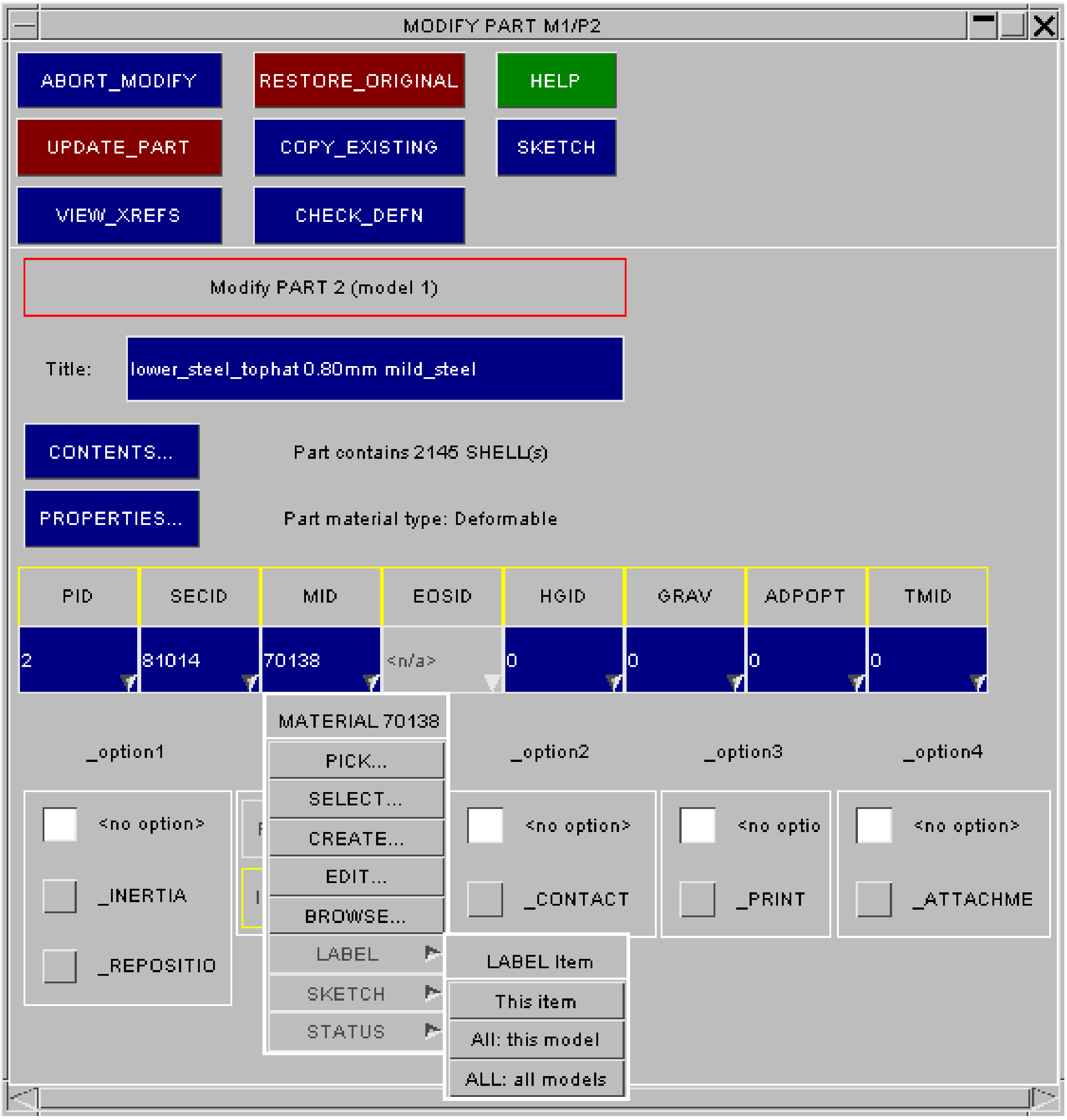
This process is invoked by "popup" menus, as shown in this example here.

The user is editing a Part, and wants to Label everything that uses this Part's material.

Thus he has used:

[Popup]-->**Label-->Sketch**

The **Edit** and other daisy-chain functions are also available. These are described in the table below.



"Daisy-Chain" function	Description
PICK...	Select an item for this location by picking directly from the screen with the mouse.
SELECT...	Select an item for this location via the standard selection menu, which offers filtering, screen-picks by various methods, typing in of labels, and so on.
CREATE...	Create a new item for this location. This maps the standard creation panels as described in the following sections.
EDIT...	Edit the current selection, using the standard Create/Edit panel as above. Only available if the selection actually exists already.
BROWSE...	Browse the current selection. This lets you look at it using the standard Create/Edit panel, but any options that would allow changes are disabled. Therefore this is a "safe" way of viewing data.
LABEL > **or** **VIEW/LABEL >**	The "Label" function allows you to label either the single item, or the whole range of these, on the current image. The "View" option is only available for those classes of item for which a viewing function (see section 4.5) exists.
SKETCH >	Lets you sketch this item, or the whole range of them, on the screen. (Sketching is always uses "line" mode, in white, to superimpose the items on the current image.)
STATUS >	Provides status listings at different degrees of complexity.

Daisy-chained panels may have "children" and "grand-children" and so on to any level of complexity, but all are owned by the original "parent". Dismissing the parent panel will destroy all its child panels, to the <nth> generation, equivalent to dismissing each one individually: they do not exist autonomously.

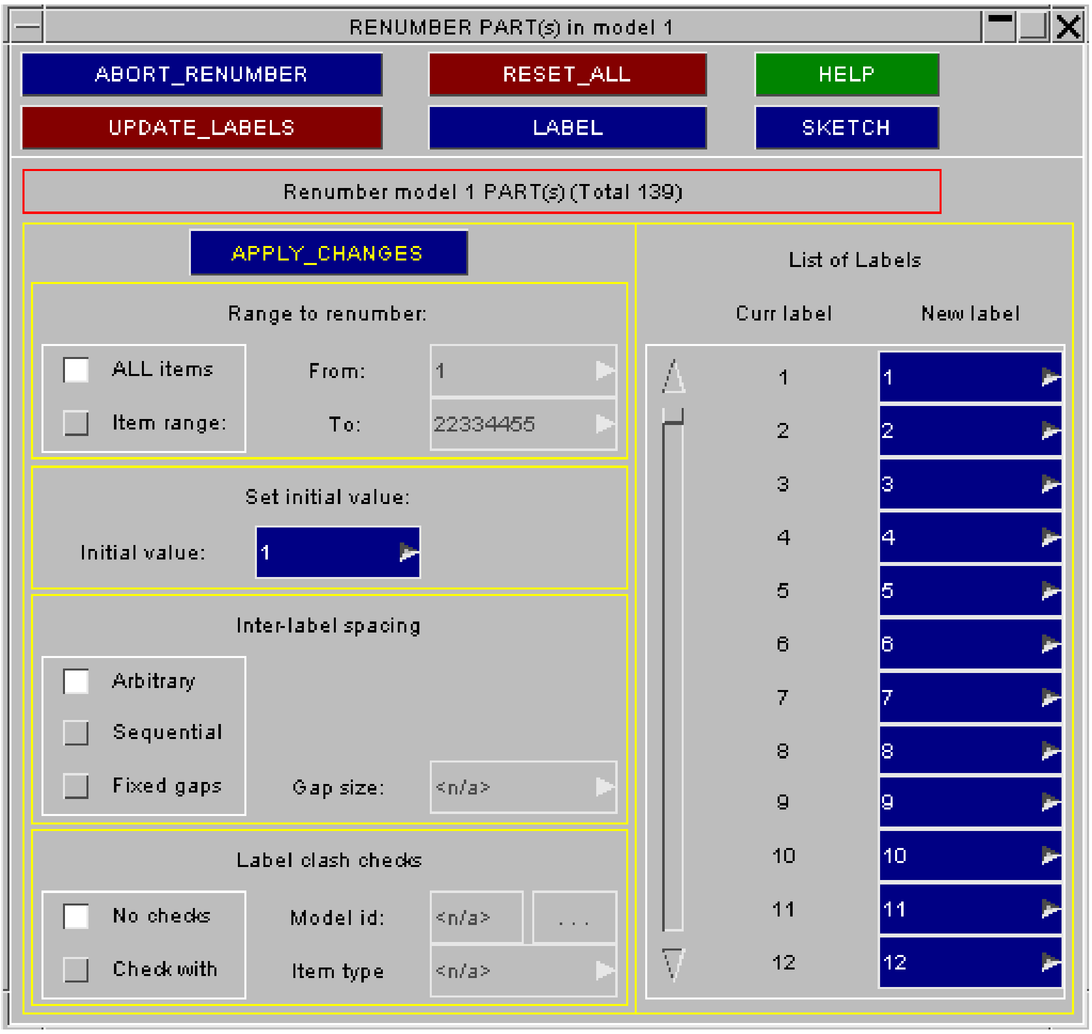
This makes is possible to be performing the same function on an item in two or more places simultaneously, for example you might be editing a part in two separate panels. There is no conflict, since all Create and Edit operations always take place on a "scratch" copy of the original item, and the original is only modified when the user explicitly requests this by using the **UPDATE/CREATE** button.

Therefore the permanent definition of the part in this example is always defined by whichever panel most recently updated it.

Clearly multiple Browse operations, being "read-only" in nature, never constitute a threat to the original definition since they will never update it, which is why the facility is provided.

5.1.5 Standard category renumbering panel.

Wherever a **RENUMBER** option is available for an item category this will invoke the standard renumbering panel for that item



This panel is the same as that described in section 3.7.1: refer there for usage details.

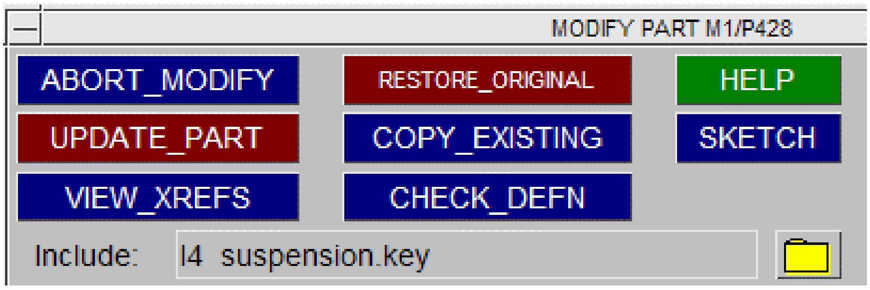
As with editing functions it is possible to have two renumbering panels live for a given model/item combination: one opened from its create/edit panel, and one from the main Model>Renumber command.

This is not a good idea, but should it happen the one from which the most recent **UPDATE_LABELS** command is issued will "win" in determining the final labels for that item category.

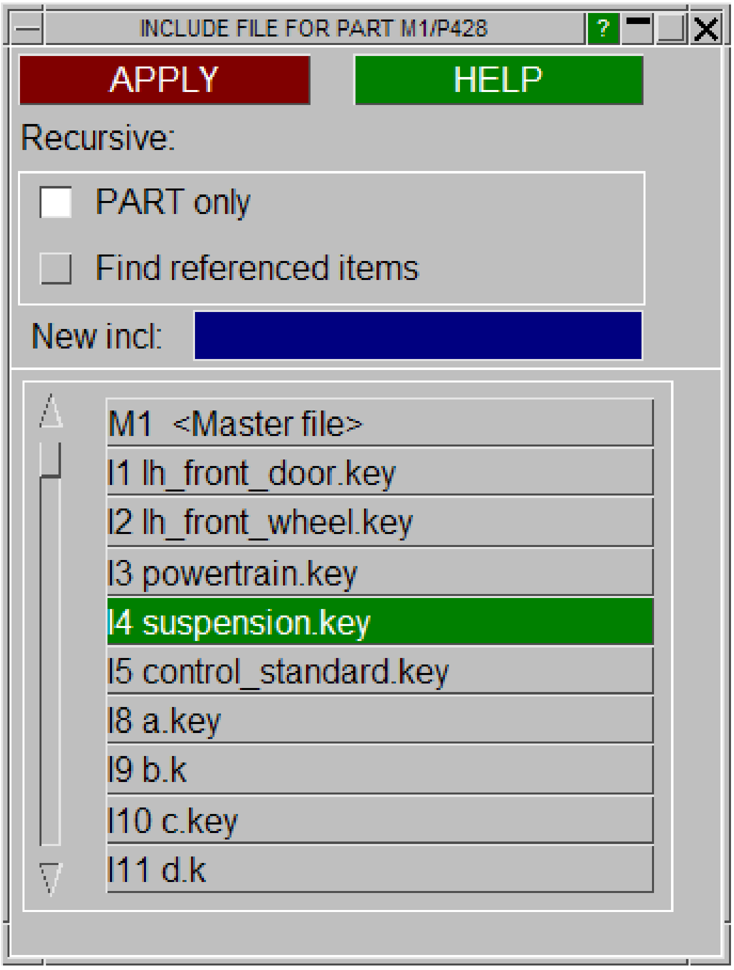
Renumbering is normally carried out using the **RENUMBER** option from the **MODEL** menu (see section 3.7). This section describes the standard category renumbering panel available for specific keywords.

5.1.6 INCLUDE file selection on edit panels

Edit panels now display which INCLUDE file the entity is located in. The INCLUDE file can also be modified in the edit panel when creating or modifying an entity.



The INCLUDE file is displayed in the box below the main edit panel buttons. If the INCLUDE file has an **INCLUDE_TRANSFORM** applied to it, **(T)** will be displayed next to the INCLUDE name. By clicking on the folder icon next to the include name the include selection panel opens up:



At the bottom of the panel all the INCLUDE files in the model are listed. The INCLUDE file that the entity being modified/created in is highlighted in green. To change the INCLUDE file, click on another file in the list and click **APPLY**. New INCLUDE files can also be created in the include selection panel. To do this, enter the name of a new INCLUDE next in the blue box next to **New incl**. The INCLUDE file will then be created. By default, only the calling entity will be moved to a newly selected INCLUDE file. By selecting **Find referenced items** a new panel is opened when clicking **APPLY** which allows you to choose any referenced items you may wish to move as well.

5.1.7 Character labels

LS-DYNA 971 R2 allows 8 letter character labels to be used for the following keywords

- EOS
- HOURGLASS
- MATERIAL (structural and thermal)
- SECTION

PRIMER fully supports these. In the editing panels for the above keywords (and the PART editing panels) character labels can be used instead of integer labels. Additionally the character labels will be shown in object menus.

If you write a model which contains character labels to an older version keyword file, the character labels will be replaced by integers which do not clash with the 'normal' labels.

Merging models containing clashing character labels

If clashing character labels are found when merging models PRIMER will try to fix the clash by:

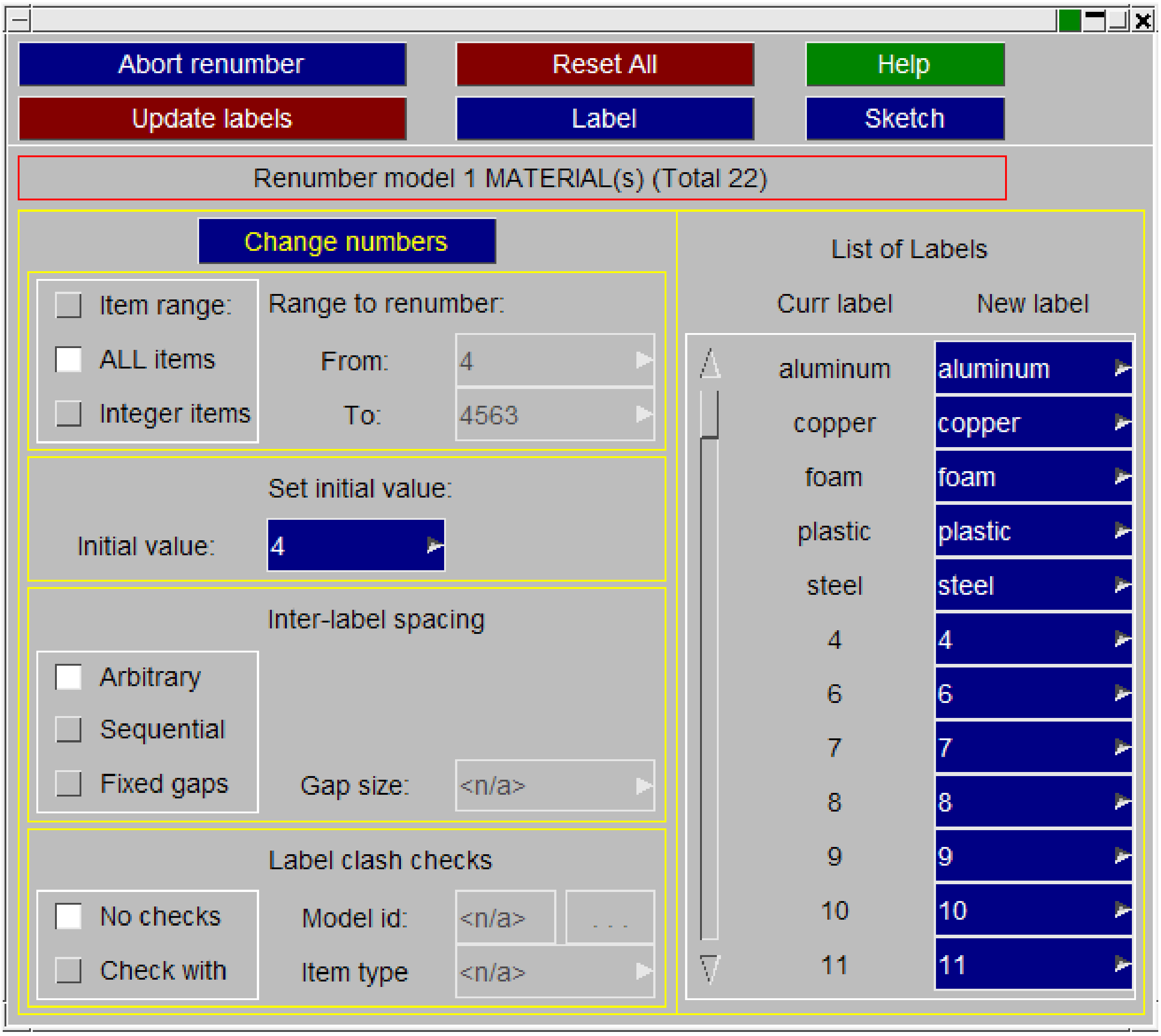
- if the name ends with '_<number>' then read the number, increment by one and replace '_<number> 'with '_<number+1>'.
- otherwise append '_1' onto the name
- repeat until a non clashing name is found
- if the name is longer than 8 characters, strip off the last character in the name (after stripping off any '_<number>') and repeat the process.

More details on model merge are available in section 3.4.

Renumbering models containing character labels

If a model contains character labels then they will not be included when renumbering using the main Renumber contents panel.

However they can be viewed changed/renumbered etc using the individual category panel. e.g. the following figure shows the renumbering panel for a model that contains materials with character labels.



If the category contains character labels then the **Range to renumber** section will have an additional option, **Integer items**. If **ALL items** is selected then the character labels will be included in the renumbering. If **Integer items** is selected then they will not be included.

5.1.8 PGP Encrypted keyword data

LS-DYNA release 971 onwards permits data encrypted with "Pretty Good Privacy" (PGP) to be embedded anywhere in an input deck using the sequece:

-----BEGIN PGP MESSAGE-----
[Any number of lines of encrypted data]
-----END PGP MESSAGE-----

Typically this used to encode ***MAT** and ***DEFINE_CURVE** data in order to protect the intellectual copyright inherent in the values used, but it can be used for any keyword in a completely arbitrary fashion.

PGP uses "public/private key" encryption, in which anyone with the public key can encrypt data, but only those in possession of private key can decrypt it. LS-DYNA has this private key, but PRIMER does not, which means that the information in the encrypted sections is completely opaque to PRIMER. Therefore blocks of encrypted data within an input deck are handled as follows:

Free-standing PGP encrypted data blocks.

When a block of PGP encrypted data is encountered ***outside*** the specific cards of a keyword it is referred to here as "free-standing", meaning independent of any keyword. PRIMER handles such blocks as follows:

- They are copied verbatim to a "saved pgp data" file.
- Separate files are used for the master keyword file and any include files.
- When the model is written back out any saved PGP data blocks are written out immediately before the *END card in the relevant file(s).

This treatment implicitly assumes that free-standing blocks of data are independent, and that ***the position and order in which they appear in the keyword file are not important***.

If the order or position of the encrypted data in your deck ***are*** important then it is recommended that you place them in an include file, since PRIMER will treat such a file verbatim and it does at least reduce the problem of positioning the data to the insertion of a single ***INCLUDE** statement at the desired position in the master input deck.

PGP data within a *MAT definition.

A special case is that PGP encrypted data with a material card is treated "intelligently" if the following format and rules are applied:

***MAT_*xxx*_TITLE**	Where *xxx* is the material type. A new *MAT header is required for each material
<title>	Must be (I10,A70) format, with the material label in the first 10 columns
-----BEGIN PGP MESSAGE-----	The encrypted data must be the whole of the rest of this material's data only.
[Encrypted data]	
-----END PGP MESSAGE-----	

If these rules are followed then PRIMER "knows" what the material label and type are, even if it doesn't know anything about the details of the data.

Given this information it can deal with the material in a reasonably intelligent fashion, for example if it is rigid then it can apply the normal rules about rigid bodies. However:

- PRIMER cannot "know" what the material's yield stress or other properties are (perhaps used in timestep calculations), so
  A nominal density or stiffness is inserted as appropriate
  A minimal nominal set of remaining required data is also inserted.

- PRIMER also cannot "know" about other items, typically loadcurves, that this material might reference.

PGP data within a *DEFINE_CURVE definition.

Another special case occurs when the (x,y) points data for a loadcurve are encrypted using the following rules and format:

`*DEFINE_CURVE`	A new *DEFINE_CURVE header is required for each curve
`<lcid> <sidr> ... <dattyp>`	This line must be "in clear"
`-----BEGIN PGP MESSAGE-----`	The encrypted data must be all the (x,y) curve points for this curve only
[Encrypted data]	
`-----END PGP MESSAGE-----`	

If these rules are followed PRIMER "knows" the material label, and therefore can deal with references to this curve from other items. However:

- PRIMER cannot "know" what the (x,y) data points of this curve are, so
  A nominal pair of points (0,0) (1,1) is inserted.

Further considerations affecting encrypted *MAT and *DEFINE_CURVE definitions

Locking encrypted loadcurves against deletion.

It is usually the case that when encrypted materials and loadcurves occur in the same model that the materials refer to the curves. Unfortunately it is not possible to tell which, if any, materials do actually refer to which loadcurves, and if no action is taken it is likely that some or all loadcurves would appear to be unreferenced, and hence unused, and would be deleted by the standard PRIMER "cleanup" operation.

To prevent this PRIMER automatically assumes during deletion or cleanup operations that all encrypted loadcurves are referred to by all encrypted materials, which has the effect of locking all such curves against deletion so long as there are any encrypted materials in the model. This can be a nuisance, but it is hard to see how else the problem can be dealt with and, in practice, most encrypted models will be effectively "read only" anyway.

Controlling curve labels

If loadcurves (encrypted or not) are referred to from within encrypted materials then these references are effectively frozen within the encrypted data block. Therefore if the labels of the loadcurves referred to are changed PRIMER will not "know" that this may cause problems, and when the input deck is run subsequently in LS-DYNA problems may occur because the "old" labels within the encrypted material will not find the "new" loadcurve labels.

Therefore take great care when changing labels in encrypted decks, since PRIMER cannot protect you from potential errors. The following strategies are recommended:

- The vendor of the encrypted deck may provide some mechanism for adjusting labels. Sometimes this is a simplifed (in clear) include file that can be used during model assembly, with the proper (encrypted) include file being substituted for the analysis.

- Another alternative is to include encrypted include files within an **`*INCLUDE_TRANSFORM`** definition, and to use this to offset labels. This is usually a better solution since it effectively moves the point at which the actual renumbering occurs to somewhere inside the LS-DYNA keyword reader, where the encrypted data is then available in clear.

Editing encrypted definitions

The normal editing panels will work for encrypted curves and materials, and they will show that a minimal primitive amount of data have been inserted. If you edit such definitions the edits will be accepted and remembered, but when the deck is written out the original encrypted data block will be re-inserted and any edited data ignored.

5.1.9 "Embedded" Keyword Comments

From release 10.0 onwards PRIMER reads, stores, edits and writes out comment lines embedded within keyword data. The following rules are used to define what is meant by "embedded", and also to limit and control which comments are stored.

Rules for comment storage

PRIMER faces the problem that while it stores keyword data, and knows which include file this occupies, it does not "remember" where (at which line) a particular keyword occurred. In fact it may often re-order the data in a keyword deck when it writes it out. Therefore in order to be stored comments need to be associated with a keyword, and PRIMER takes this a stage further by remembering exactly which line in a keyword each comment is associated with.

Consider the following fragment of a keyword deck, which has been artificially separated into sections:

<table>
<tr><td>These comments are "free-standing", coming after the previous keyword.</td><td><pre>$
$ The following parts refer to the seat.
$ These were added on 11th May 2011
$</pre></td></tr>
<tr><td>The comments here are "embedded" within the *PART definitions</td><td><pre>*PART
sill_swan_neck
$:   pid   secid   mid   eosid
hgid
$ This part used to be steel, but is now
aluminium
$ Thas change was made on 2nd January 2010
         5   3   5   0
0
$ This part uses the original steel
material
         6   3   6   0
0</pre></td></tr>
<tr><td>These comments are "free-standing"</td><td><pre>$
$ Ten further parts were moved to include
file "floor.key"
$</pre></td></tr>
<tr><td>The comment here is embedded within the *SECTION keyword</td><td><pre>*SECTION_SHELL
$ This section has been increased from 1.0
to 1.2mm thickness
         1   2   0.0   0
       1.2   1.2   1.2   1.2</pre></td></tr>
</table>

There are three sorts of comment lines here:

"**Embedded**"
 	
Shown in **red**
 	
Saved by PRIMER
	These comments lie between the *Keyword header and a data line, or between sucessive data lines.
	
	PRIMER associates these with the line of data that ***immediately follows*** the comment, and saves them in that location on that keyword.
	
	For example the comments
	
	`"$ This part used to be steel ..."`
	and
	`"$ This change was made ..."`
	
	Are both associated with line 1 of Part 5. And the comment
	
	`"$ This part uses the original ..."`
	
	is associated with line 1 of Part 6. Any number of comments may be associated with any data lines, and they are stored as text strings in the order in which they appear.
	
	Comments stay with their definitions so if, for example, Part 5 in this deck was moved to a different include file it would take its embedded comments with it, and they would appear above the relevant data line.
	
	If the keyword data is modified so that its data lines are reformatted then comments still associate themselves with line N of the definition, regardless of what that might contain. In other words association with a given line of a definition is "dumb" and is not related in any way to the data that line contains.
	
	These comments can be visualised in PRIMER, and modified using the Text edit capability.

"**Special**" Shown in **blue** Ignored by PRIMER	If a comment line starts with a dollar followed immediately by a colon, "**$:**", then PRIMER treats this as a special comment that does not need to be remembered. This syntax is used for data field headers and comment lines such as those listing the items referencing a loadcurve, since these will be regenerated automatically (and may change) each time a new keyword output file is created, so it does not make sense to store them.
"**Free standing**" Shown in **dark green** Ignored by PRIMER	These comments lie after the last data line of the previous keyword, and before the *Keyword header of the next one. PRIMER ignores these comments since they have no fixed association with a particular keyword. (An exception is made for comments at the top of a keyword deck / include file, which are stored as a property of that file and may be edited.)

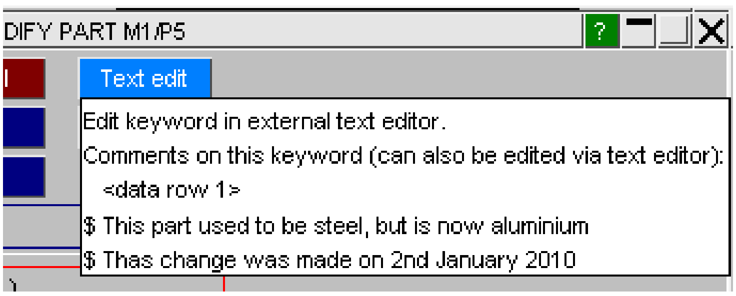
Visualising embedded comments

PRIMER allows you to visualise and edit comments in two ways:

On a scalar editing panel

The **Text edit** button will be shown in light blue, and hovering the cursor over that button will list comments for that keyword.

This example shows the result for Part 5 above.



On a Keyword editor row

If the definition on a row contains comments then a light blue C will be shown on its row button, and hovering the cursor over that button will list them.

In both cases you can add, remove and edit comments with the external text editor by using the **Text edit** button. (In the keyword editor case this is one of the options in the right-click popup menu associated with the row button

5.1.10 **Text Edit** Editing keyword data externally.

You can export data to file in keyword form and then use a system text editor to view, modify and update one or more keywords. This also gives the ability to import data from other decks by "cut and paste" into the text editor session and then import back into PRIMER. Embedded keyword comments can also be visualised and edited using this capability.

Editing a single item

In any scalar (single item) editing panel you can use the **Text Edit** button to generate a mini-keyword deck containing just this keyword. It is generated from the scratch data currently populating the editing panel, and will contain any comments embedded in the card. (It is also possible to edit multiple items using Text edit on the keyword editor, this is described below.)

An example is given below for a PART, and the default output is split into three sections as annotated here

Section	File Contents (continuous text in an actual file)

(1) Header	`$ Edit/view data and comments.`
	`$ ----------------------------`
	`$`
	`$ Only lines after *keyword below will be reread by PRIMER. See RULES`
	`$ at the end of this file for details on data and comment editing.`
	`$`

(2) Data	`*PART`
	`sill_swan_neck`
	`$:      pid     secid       mid     eosid      hgid      grav    adpopt`
	`          5         3         5         0         0         0         0`
	`$`

<table>
<tr><td>(3) Rules</td><td><pre>
$ RULES for editing data and comments in this file.
$ ---
$
$ Make any changes you wish to the data and/or comments and then save
$ the file under its existing name to update the PRIMER definition.
$
$ Rules when saving changes:
$
$ If you simply 'quit' from this edit this file will be deleted and
$ no changes will be imported back into PRIMER.
$
$ If you make changes to either data or comments then save this file
$ under its current name your changes will be imported back into the
$ current edit panel in PRIMER, and the file will then be deleted.
$
$ If you save this file under a different name, with or without any
$ edits, that new file will remain on disk. Any changes will only be
$ imported back into PRIMER if you also saved them to the original
$ filename, and that original file will be deleted as above.
$
$
$ Rules when editing comments for import back into PRIMER:
$
$ Only comments inserted between the *keyword header and data rows,
$ or in between multiple data rows, will be 'remembered'. Comments
$ before *keyword, and 'trailing' comments below data rows (such as
$ these rules) will be ignored.
$
$ Comment lines (anywhere) starting '$:' will be ignored.
$
$
$ To suppress the initial explanation and these RULES set the
preference
$
$ primer*text_edit_show_rules: FALSE
$
$ To turn off data field headers in the file to be edited (setting
does
$ not affect the output of headers in normal keyout files) use
$
$ primer*text_edit_show_names: FALSE
</pre></td></tr>
</table>

Controlling the output in the file

This default output is verbose and is intended for a 1st-time user. You can control the output using the following preferences:

primer*text_edit_show_rules:	**true** or **false**	Whether or not to show sections (1) and (3) above. Most users who have become accustomed to the rules explaining how this works will wish to use this preference to cut output down to just section (2) only.
primer*text_edit_show_names:	**true** or **false**	Whether or not the field headers (here `pid, secid`, etc) are shown above each row of data.

Controlling which text editor is used.

By default PRIMER will attempt to use the default system editor.

- On a Unix/Linux system this may be an editor built into the Window manager system, or if the environment variable $EDITOR is set then this will be used.
- On a Windows system PRIMER will try Notepad or Wordpad if no explicit editor is defined.

On any operating system you can override these defaults by setting the following preference:

primer*text_editor:	<*pathname of editor*> for example **C:\Program Files\Vim\Vim70\gvim.exe**

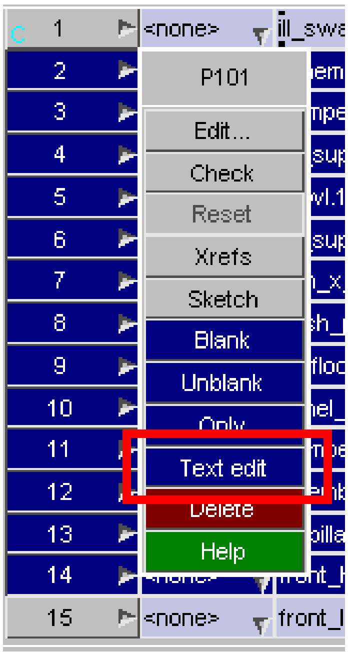
Rules for using the text editor

The text editor operates as follows:

- A temporary filename containing the keyword output is generated in the directory containing the model. If this cannot be opened, for example because that directory is read-only, then the file is created in a temporary directory instead.
- The relevant editor is launched to read this file. This is done in a separate thread so that the editor process is autonomous and does not "lock" the PRIMER session in any way.
- You are free to modify the temporary file in any way you please, or to read new content into it to replace the existing. You can also save copies of the file under different names, and generally do anything you would normally do in an editing session, including aborting the session without changing anything.
- When you finally exit from the editor the following happens:
    - Any files other than the temporary file that you might have created (by "Save as...") are left unchanged on disk, and are not considered or read in any way.
    - If the original temporary file is unchanged - specifically if its "last modified" time has not changed since its creation - then no further action is taken and it is deleted.
    - If the original temporary file has changed then it is read back into PRIMER via a "mini keyword read" process, and the contents of the file will overwrite the scratch definition in the editing panel. The temporary file is then deleted.

      Inside PRIMER this is done by reading all the file contents into a scratch model, then extracting the first definition of the relevant type (here *PART) from that scratch model and using it to replace the definition being edited. So if you import more than a single definition, or perhaps load different keywords from some other file, all except the first definition encountered of the relevant keyword will be ignored. If there are no keywords of the relevant type (here *PART) in the file, then PRIMER will issue a warning message, the file contents will be discarded and no changes will be made to the definition currently being edited.

      Therefore you can use this method as an alternative way to update keywords, perhaps by cutting and pasting text in the editor from other files. However it is restricted to changing only the current keyword definition, and cannot be used as a generalised method of importing unrelated keywords into the model.



Editing multiple items using the Keyword Editor

The example above describes how a single definition can be edited from a scalar PRIMER editing panel. It is also possible to **Text edit** multiple items by using the same function from the generic Keyword Editor described above.

This works in exactly the same way, and is subject to the same rules and limitations below, except that

- Multiple items can be exported to the external keyword file by selecting several rows for editing. In this example rows 2 to 14 inclusive will be exported for editing.

- Multiple items can be re-imported back into the model. Where labels in the external file match existing items in the PRIMER database then those items' definitions will be replaced, where external labels do not match an existing definition then a new definition will be created. The number of items read in does not have to match the number exported.

  Therefore it is possible to import multiple new entries into the model by this method, however the limitation that only those items of the current *Keyword will be considered still stands so, once again, this is not a generalised way of importing new model data.

Consequences of Text Edit being an autonomous process

Because the text editing session is an autonomous process in a separate thread it does not "lock" (freeze) the PRIMER session in any way, in computer-speak the PRIMER process and its child text editing session are "asynchronous", and this has some consequences:

- If you open a PRIMER editing panel, launch the text editor, and then close the editing panel while the editor is still running there is no longer a "scratch" definition to be updated when you finally close the editor. PRIMER detects this situation, warns you and simply throws away any changes in the edited file without updating the model in any way.

- If you record a Macro that includes Text Editing sessions the macro will not "remember" when a particular session ended, and in fact it has no way of knowing what - if anything - happened in the text editor, or when. This means that when the macro is played back there is no way of synchronising text editor sessions with operations inside PRIMER, so a macro that implicitly relies on the sequence:

  - User starts editor panel in PRIMER
  - User launches text editor to modify the scratch definition
  - User exits text editor and changes are imported back into PRIMER

  The macro will not work as expected, since the 3rd step of editing text editor and importing changes will not take place (because the macro cannot replace the user and drive the external editor).

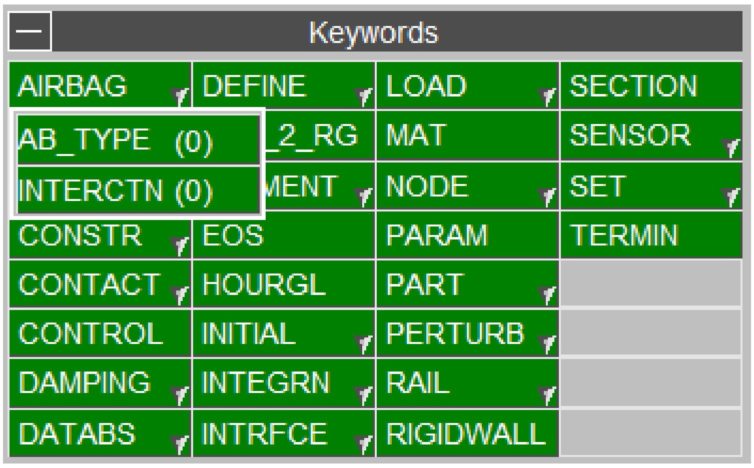
  For this reason it is recommended that you do not embed text editing sessions in macros.

AIRBAG: Airbag (Control Volume) definitions.

- **AIRBAG_<type>** sub level menu
- Creating a new definition
- Copying a definition
- Editing an existing definition
- Deleting airbag definitions
- Other operations
- Visualising airbags.
- **<_INTERACTION>** menu

"Airbags" are definitions which determine the gas pressure inside an enclosed volume by applying one of a range of gas expansion laws. This pressure then acts upon the elements that form the volume, generating pressure loads. The gas pressure is a function of incoming and outgoing gas flow, its thermodynamics and the size of the volume; the pressure is updated during the analysis as these parameters change and interact.

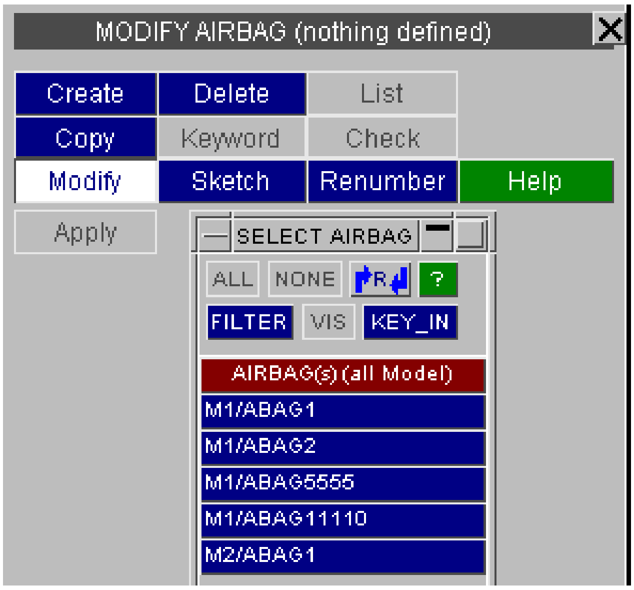
The name "airbag" is really a misnomer as they don't have to be used solely for vehicle airbags (which don't contain "air" anyway). They can, for example, be used to model the pressure inside tyres or indeed any structure where changes in volume may affect internal pressure. The term "control volume" is better, and is in fact used in formatted LS-DYNA input.



Select the sub menu desired using the AIRBAG popups in the KEYWORD menu.

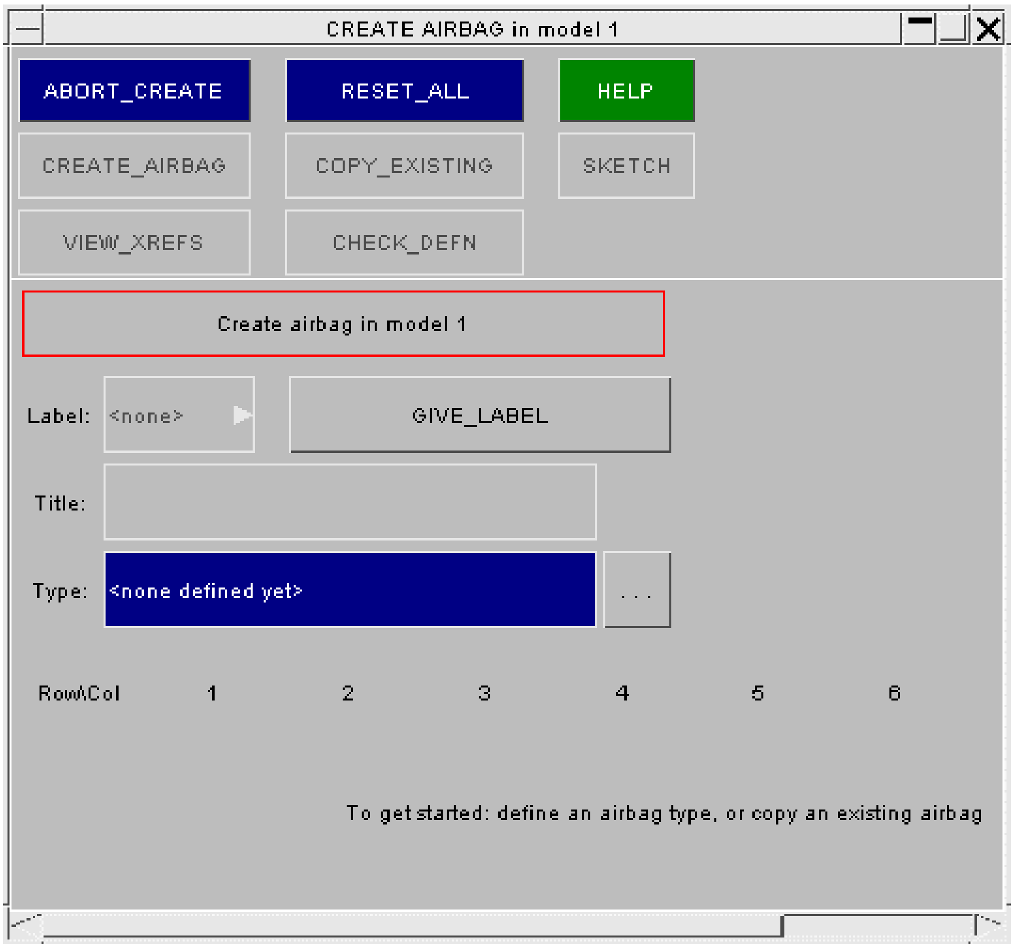
AIRBAG

This figure shows the main **AIRBAG** menu as selected from the Keywords panel. The functions currently available have their standard meanings (see section 5.0.1). Greyed out functions are not currently available:



CREATE Making a new airbag definition.

This figure shows the standard **CREATE**/**EDIT** panel for airbags. Here **CREATE** has been used, so a blank airbag creation panel is displayed.



The static buttons in the top section of the panel have functions which are common to the other editing panels within PRIMER.

Only once sufficient data has been input by the user will the **CREATE_AIRBAG** button become active. Until that time it will remain greyed out.

The data on the panel is as follows:

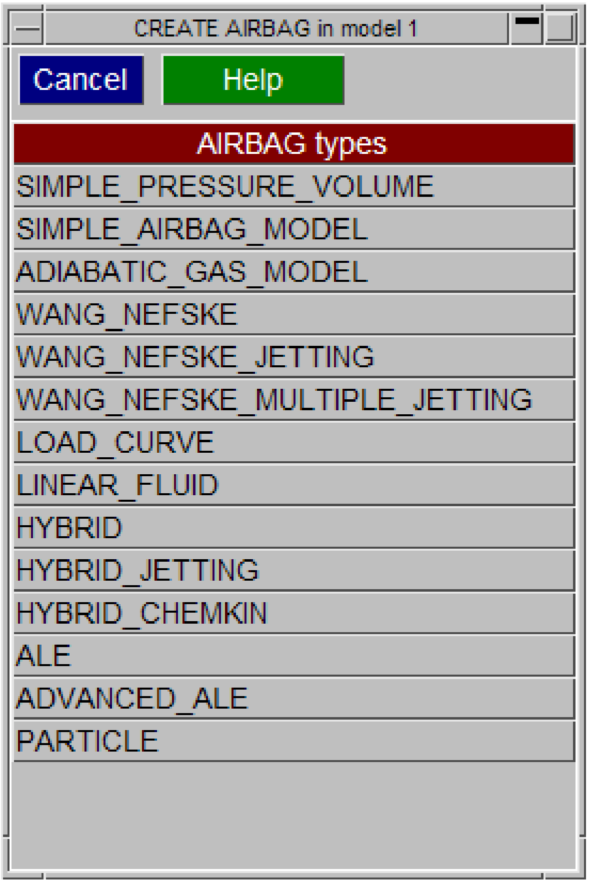
LABEL This is an internal label used by PRIMER and is not written out in the keyword deck. It is set to the first free airbag label within the model as a default. This value can be changed if necessary though existing labels cannot be overwritten. The popup window allows the **FIRST-FREE** label or the **HIGHEST + 1** label to be automatically selected.

GIVE_LABEL This button allows the optional numeric ID to be used. (Airbag definitions in LS-DYNA do not have to have an explicit number, although PRIMER requires this.) If an internal **LABEL** is already set then this is used by default. A new **LABEL** is selected as described above.

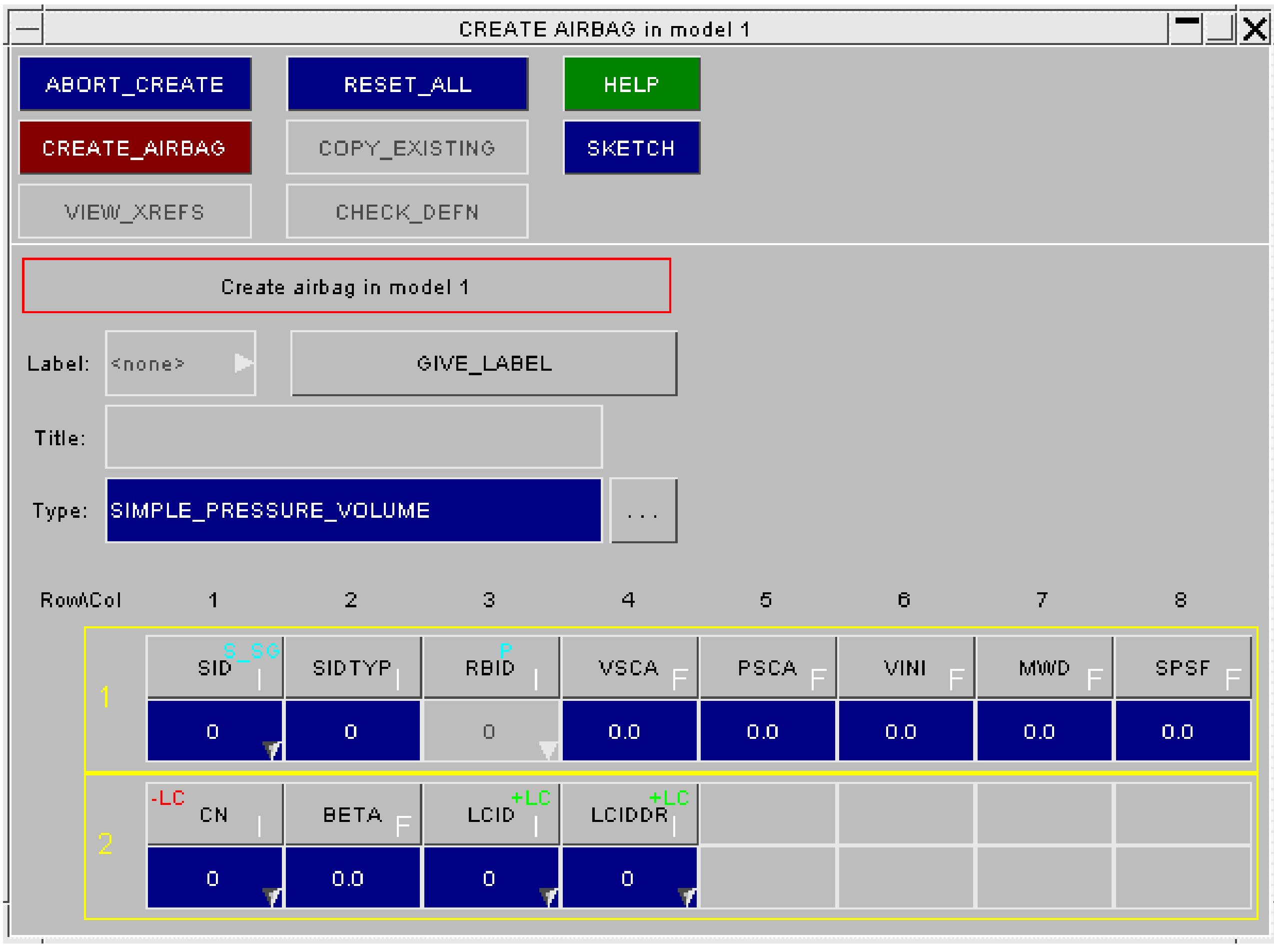
TYPE The airbag type is defined using this button. The [...] Shortcut button can be used to browse through a list of airbag types. The browse panel is shown in figure below.

ROW/COL The data relevant to each airbag type is displayed in row and column format identical to that of DYNA keywords

To start creating an airbag you must first define the type. You can type in a standard keyword if known, or invoke the selection menu in this figure with the [...] button.



Once the material type has been defined the keyword data will be displayed on the panel, as shown in figure below. Initially all values will be set to zero.

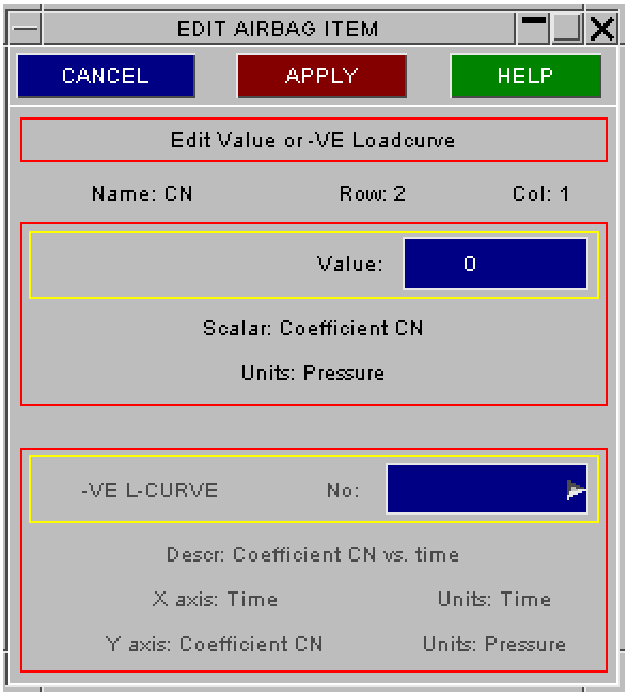


The airbag data can then be typed into the relevant boxes. The expected data type is indicated on the grey button, which also shows the acronym for that data value:

F	White **F** floating value
I	White **I** integer value
+LC	Green **LC** +ve loadcurve
-LC	Red **LC** -ve loadcurve

(Note that the component **'RBID'** is not available for editing. This is required for user defined activation subroutines which are not yet enabled from this panel.)
Information about each individual data component can be requested by pressing the grey data component button. For example; to request information about data component '**CN**' (2nd row, 1st column) press the grey button with **CN**. This will create a new sub-window with detailed information about the data component showing:

- A one-line description of it;
- Its current units type
- Its current value.



Once all of the data has been input on the airbag card, **CREATE_AIRBAG** installs the airbag permanently in the model.

COPY Copy existing airbag(s) to make a new airbag(s).

COPY makes new airbags, in the same model(s), that are identical to their originals apart from their labels. By default only the airbag definitions themselves are duplicated.

Where **RECURSIVE COPY** is requested, all items associated with that airbag (i.e. elements, parts, etc.) are also copied.

MODIFY Modifying the attributes of an existing airbag.

MODIFY functions in the same way as **CREATE**. Obviously, the airbag type will already have been selected so the panel will resemble that shown above.

Any modifications made to the airbag definition will not be made permanent until the **APPLY_MODIFY** button is pressed. At this point a the local copy which has been updated is used to overwrite the version in the model.

DELETE Delete existing airbag definitions.

The **DELETE** function deletes the selected airbag. Two switches may be used:

DELETE_RECURSIVE	Also elects for deletion any segment sets, their segments and associated nodes that are used uniquely to define this airbag. Where airbags are defined by Part or other non-unique "structural" methods these items are not selected for deletion.
REMOVE_FROM_SETS	Is needed if segments are to be deleted as it stops their definition in sets "locking" them against deletion.

SKETCH Sketch elements used by an airbag on the current image.

SKETCH sketches on top of the current image the parts and elements that are referenced by the selected airbags.

LIST Not yet active

CHECK Not yet active

RENUMBER Renumbering airbag labels.

RENUMBER lets you change any or all airbag labels within a given model using the standard renumbering panel.

To change the label of an individual airbag it may be simpler just to **MODIFY** it.

Visualising Airbags

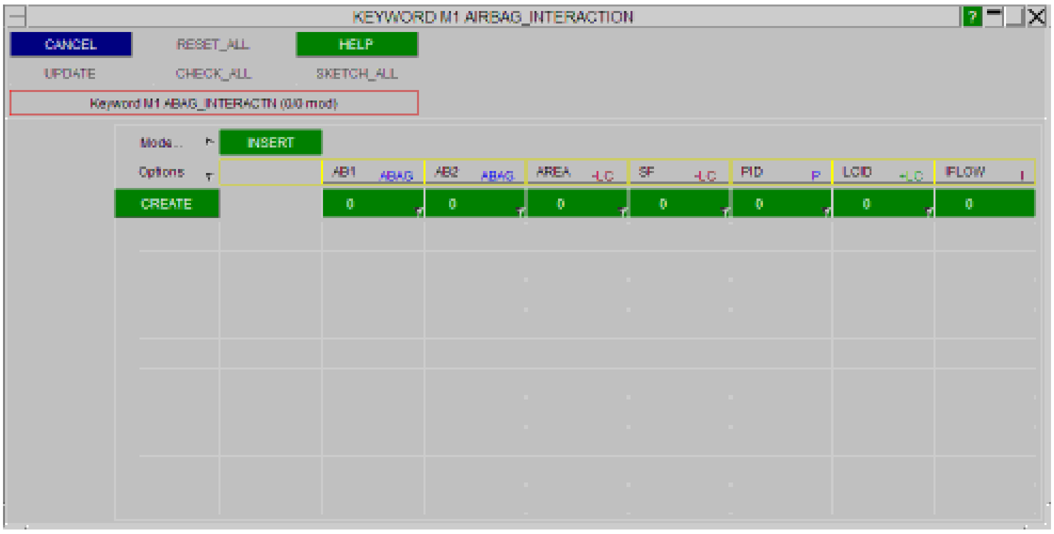
Airbags can be drawn (activate in the **ENT**ity Viewing panel). Alternatively,

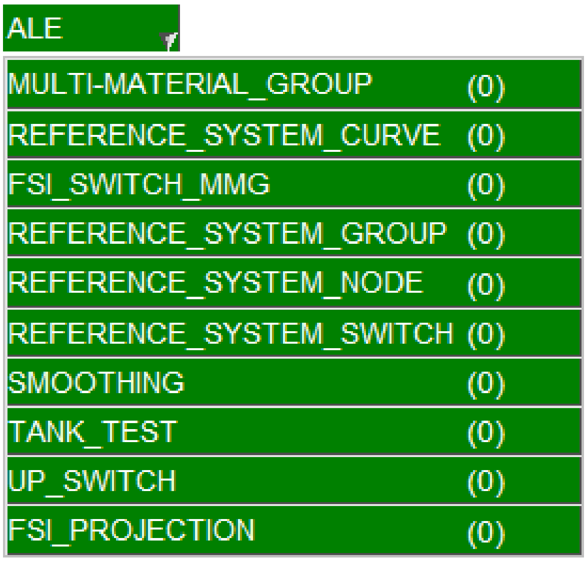
- **SKETCH** it, or
- **MODIFY** it and sketch it.

It will be drawn in terms of the parts and elements, or segments if defined by **SET_SEGMENT**, that defines it.

*AIRBAG_INTERACTION

This is available through the standard keyword editing panel in section_5a.



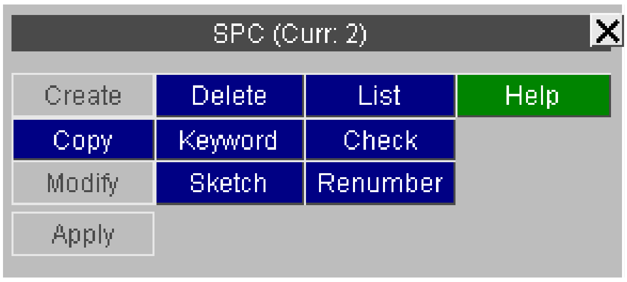


There are currently seven sub-keywords available as shown on the left. Currently these can be edited through the generic Keyword Editor.

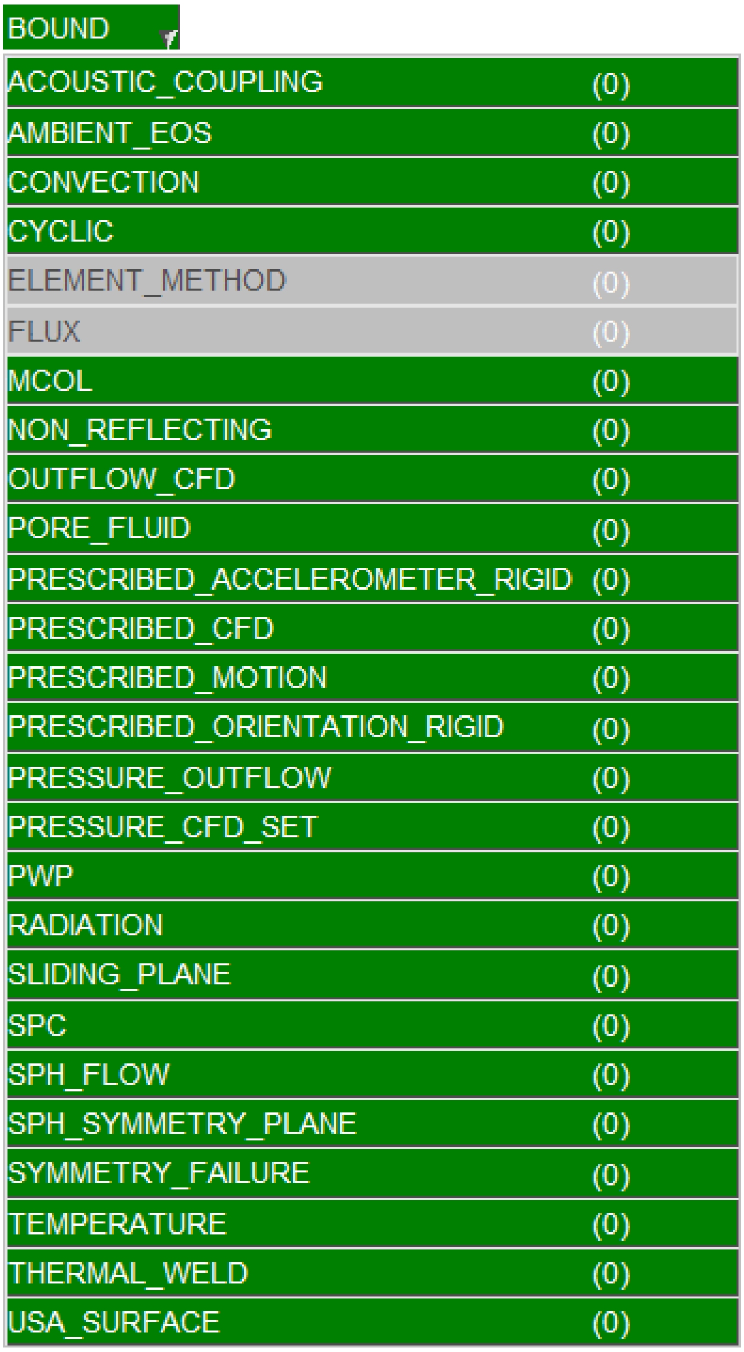
BOUNDARY: Defining boundary conditions.

Boundary conditions within LS-DYNA apply a range of restraints and other imposed conditions to models.
All *BOUNDARY sub-keywords except *BOUNDARY_ELEMENT are editable within PRIMER. (Boundary elements do not logically belong here: really they merit their own section since they imply a totally different type of analysis.)

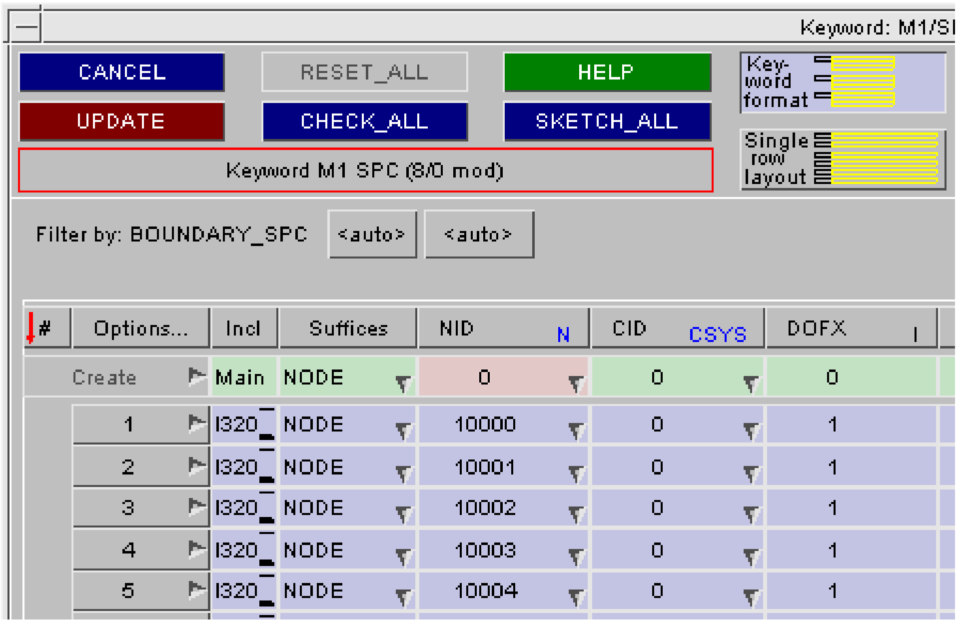
***BOUNDARY** cards can at present be edited only with the generic "Keyword" editor: no specific Create/Edit panels have been written yet.
All ***BOUNDARY** keywords except **_ELEMENT_METHOD** may be edited in this way.



The other commands (**COPY, DELETE,** ...) function in the standard manner described in section 5.0.1.



All of the boundary keywords are selected from the pop-up menu produced after Boundary is selected in the Keywords panel.



This shows an example of the Keyword editor for ***BOUNDARY_SPC**.
There are two sub-keywords: **_NODE** and **_SET**, with different formats.

The **_SET** variant is being edited here.

Drawing *BOUNDARY items.

These definitions can be viewed using the **ENT**ity viewing > **BOUNDARY** options. At present only the following keywords are fully visualised:

`*BOUNDARY_SPC` Restraints ("single point constraints") at nodes.

Restraint codes are drawn as vectors in the relevant X, Y or Z directions, using the colour scheme:
X Red

Y Green

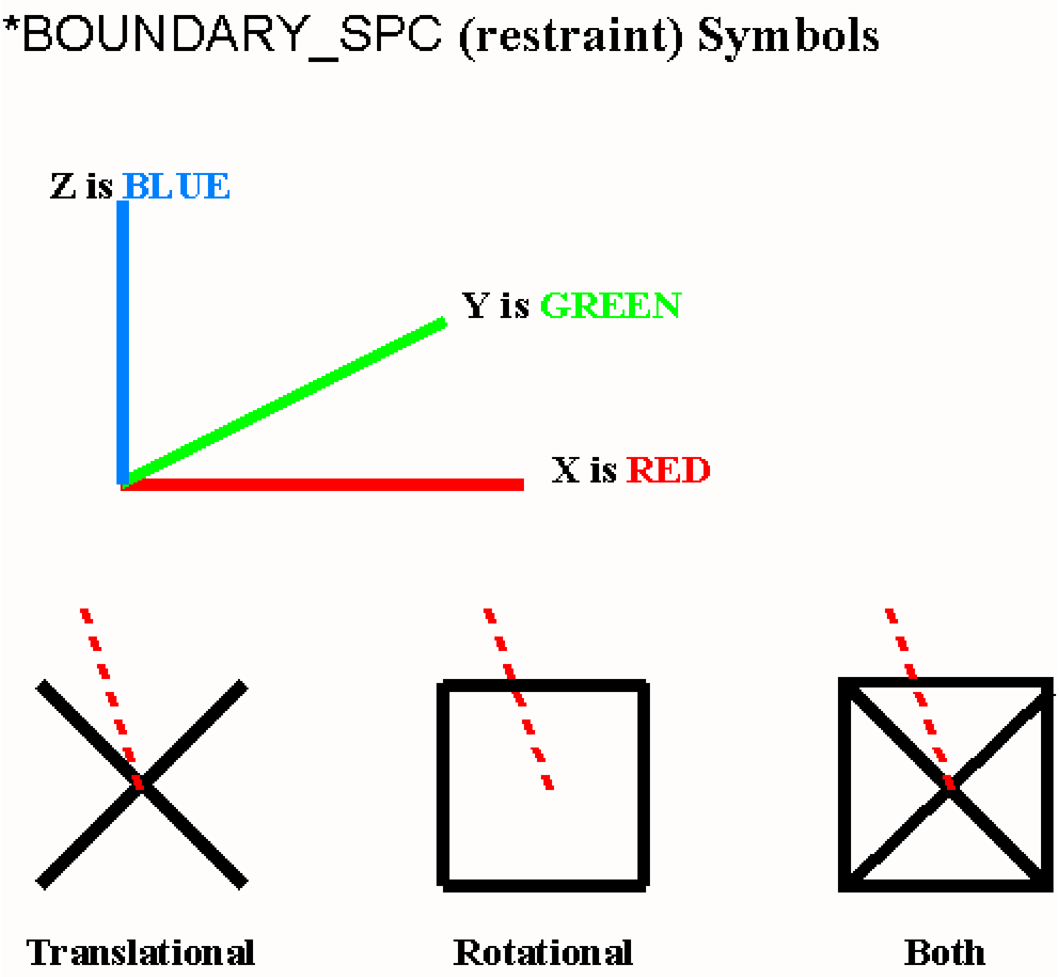
Z Blue

The symbols used at vector ends are:

Trans: Cross

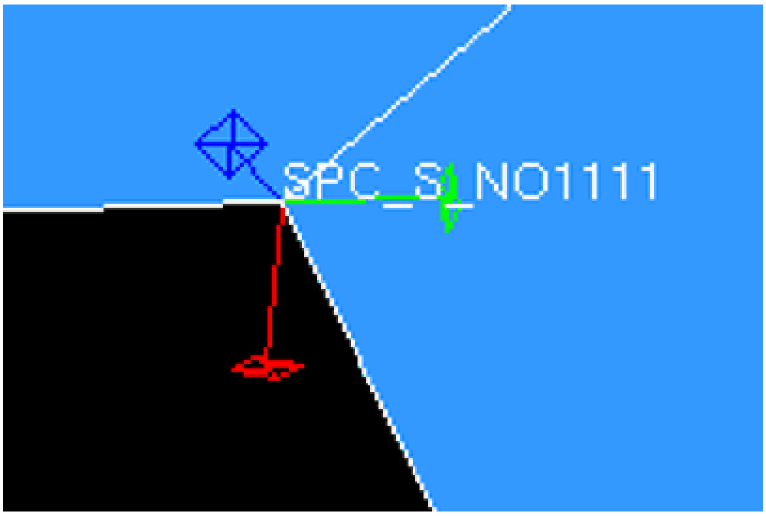
Rot'l: Square

Both: Cross + Square



Symbols are drawn at every restrained node.

This example shows a node which has been fully restrained in all of X, Y and Z, both in translation and rotation.
Labels have been turned on for this, and they show that this node is restrained via node set 1111.

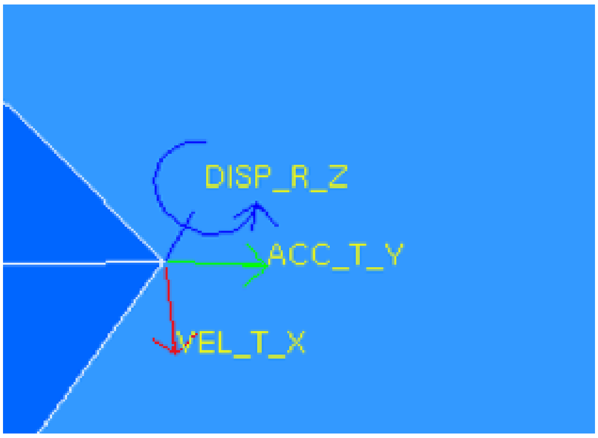


*BOUNDARY_PRESCRIBED_MOTION

This is visualised as

- An arrow in the relevant direction, coloured (X=red, Y=green, Z=blue). For rotational motion an arrow circling the relevant vector is used.
- A description at the arrow head, eg "**VEL_T_Z**" for Z translational velocity.

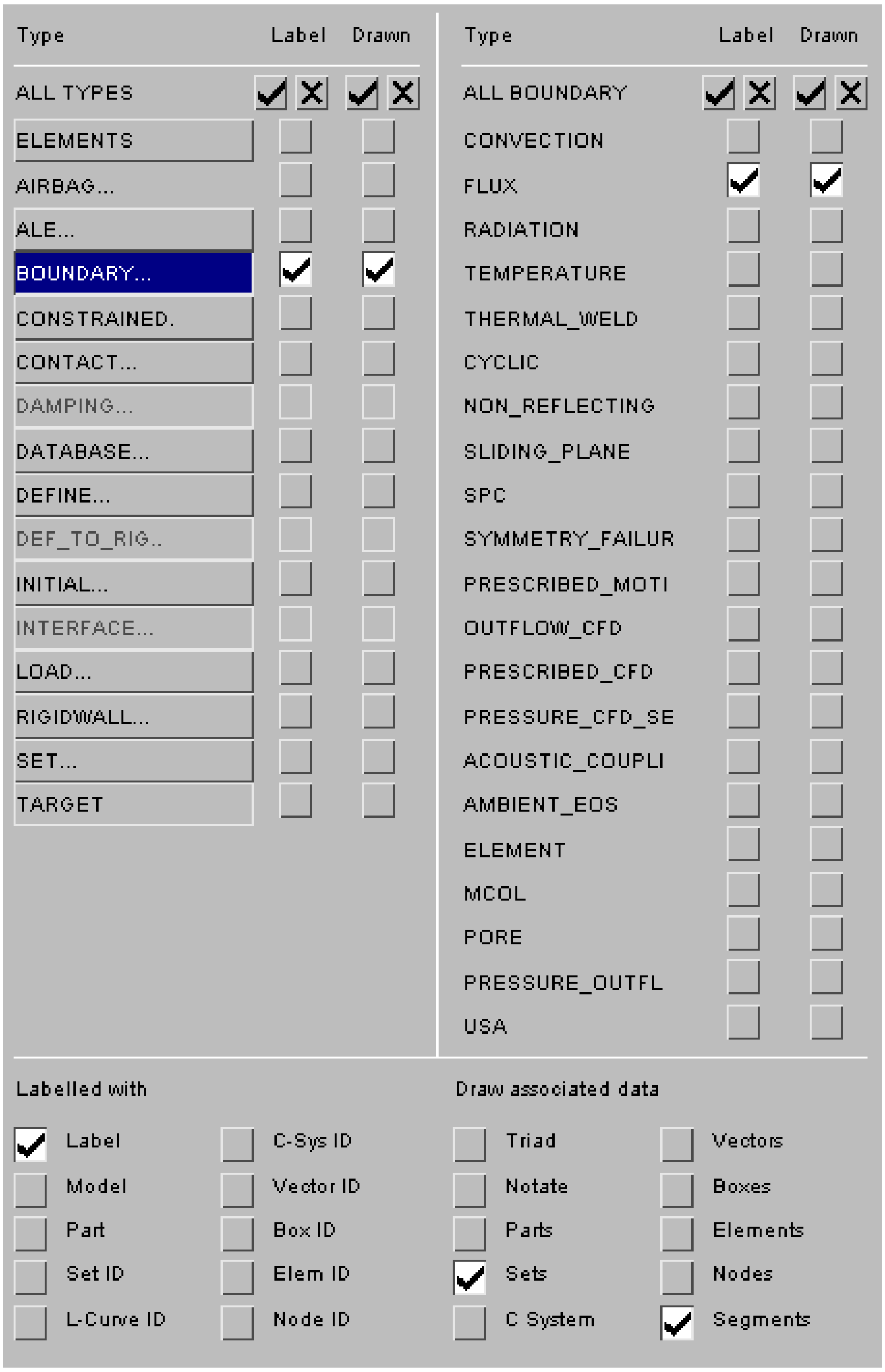
This example shows:
- Translational Acceleration in Y
- Translational Velocity in X
- Rotational Displacement in Z



All other ***BOUNDARY** sub-keywords:

Are visualised only in terms of the components that they reference: sets, elements, nodes, etc. Turn on the relevant items in **ENT**ity viewing to see these.

Turning on the relevant *BOUNDARY sub-keyword labels will annotate them correctly.

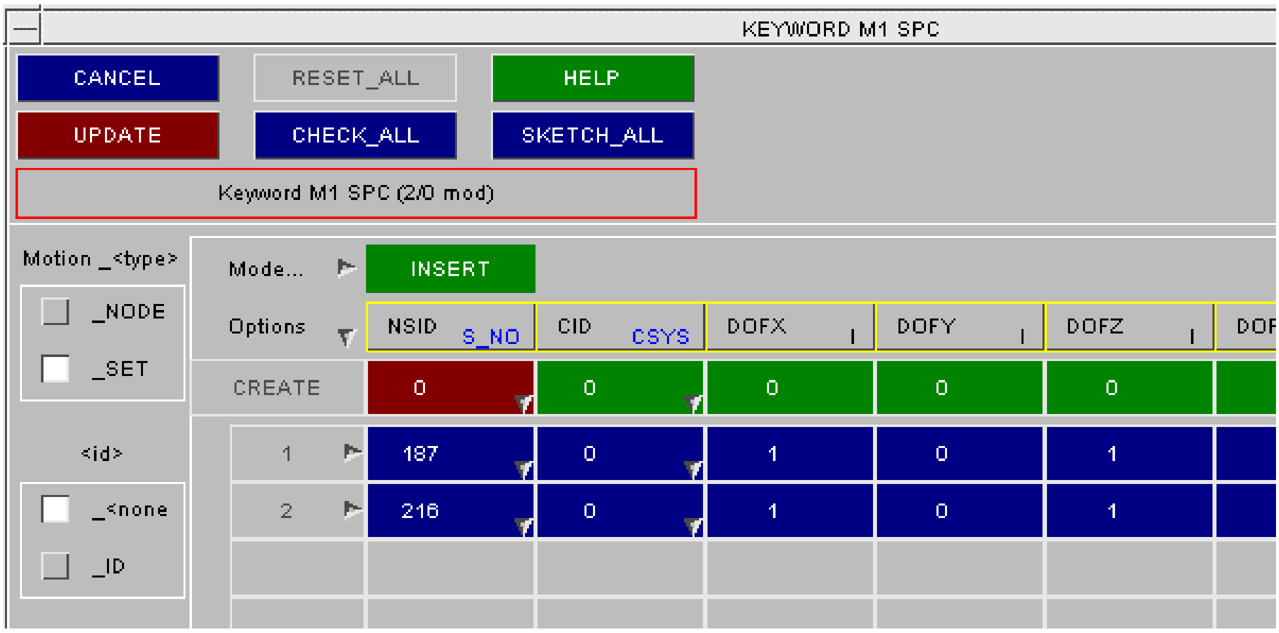


Labelling of ***BOUNDARY** items within PRIMER.

LS-Dyna keyword input has optional labels for *BOUNDARY items: the conversion from "keyword" to "formatted" input that precedes every LS-Dyna analysis converts them from discrete definitions to attributes applied to other items.

For internal consistency, for items not already labelled in the input model, and for items that cannot be labelled in LS_DYNA, PRIMER assigns new labels to everything that can be defined "once or many times", so *BOUNDARY definitions are given labels based on their order of appearance in the keyword input file.

Where LS_DYNA offers optional labels, (e.g. *BOUNDARY_SPC_ID versus *BOUNDARY_SPC), the labelling option is invoked in the keyword editor by selecting option "ID" (see below).



PRIMER’s new labels; where generated:

- May safely be ignored - you don’t have to worry about them if you don’t want to!
- Are treated sequentially, starting at 1. (Thus **BNDY_1, BNDY_2, ... BNDY_n**)
- Are not grouped by sub-type: **BNDY_1** might be an **SPC, BNDY_2** a prescribed motion - they are based solely on the order in which they appear in the input deck. Each *BOUNDARY definition encountered gets the next label in the sequence.
- Are used in selection menus (eg for blanking, deletion, etc). Are also used in the output deck when defining what is referenced by what.

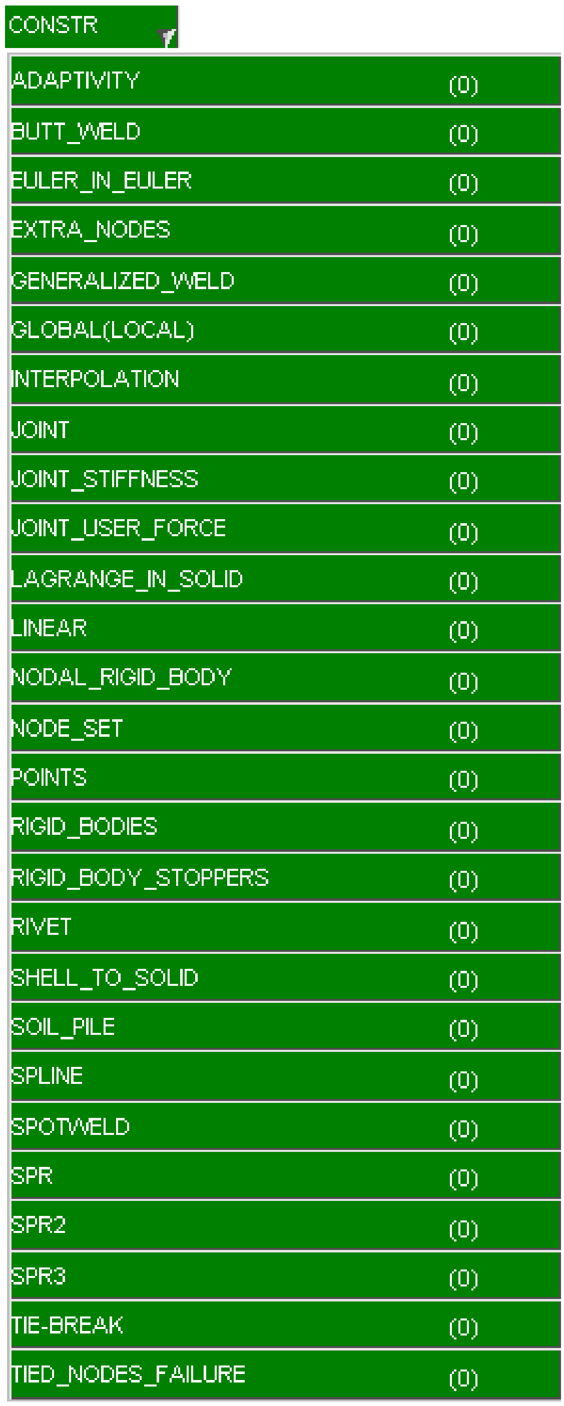
CONSTRAINED: Imposed constraints: joints, welds, etc

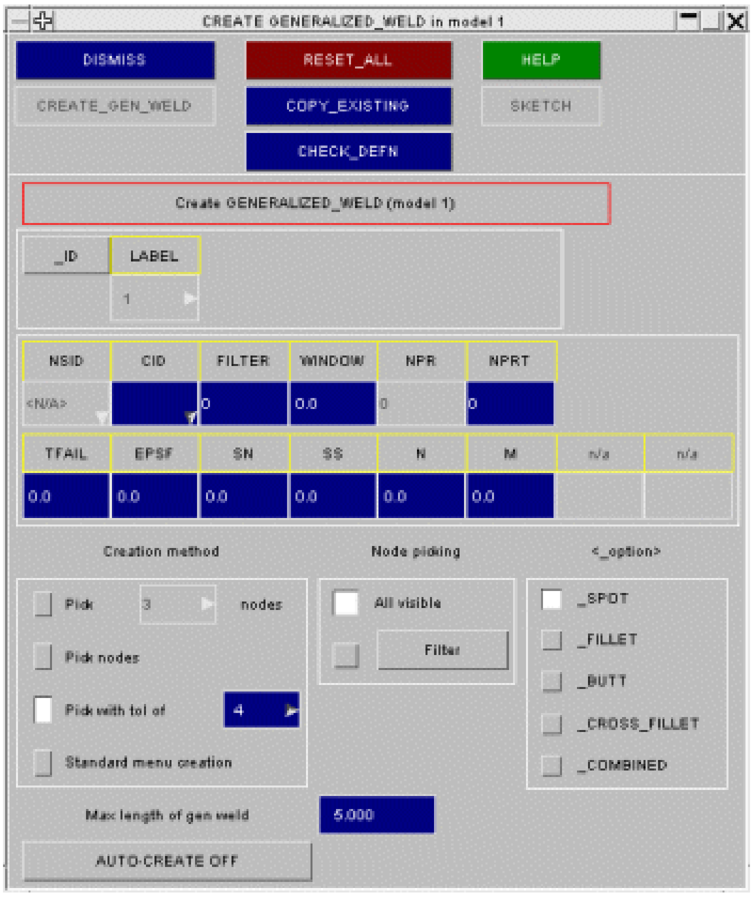
- Selecting the *CONSTRAINED sub-keyword
- "Scalar" editing panels
- "Edit range" editing panels
- Visualisation
- Labelling

Constrained conditions within LS-DYNA apply a range of constraints to models, and several ***CONSTRAINED** keywords are linked closely to rigid bodies.
All ***CONSTRAINED** sub-keywords are editable within PRIMER.

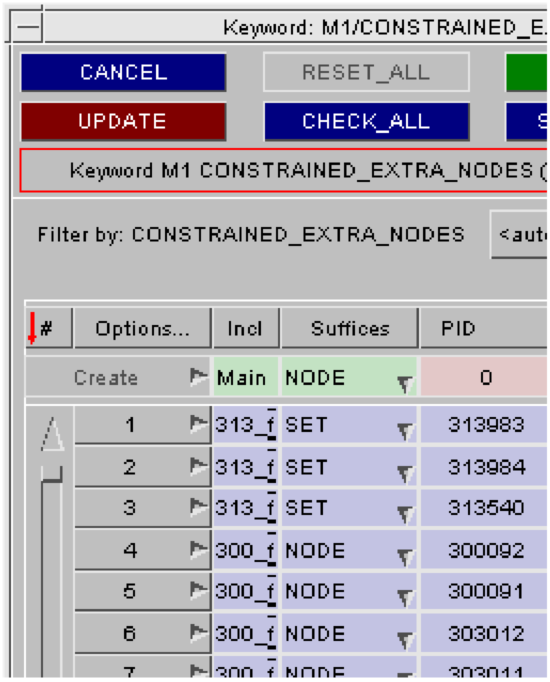
The ***CONSTRAINED** keyword has 26 sub-categories.
Some may be created and modified using standard Create/Edit panels and some with the standard keyword editor. The table below defines which.

Keyword	Create/Edit panel	Keyword editor
ADAPTIVITY		✓
BUTT_WELD		✓
EULER_IN_EULER		✓
EXTRA_NODES	✓	✓
GENERALIZED_WELD	✓	✓
GLOBAL		✓
INTERPOLATION	✓	
JOINT		✓
JOINT_STIFFNESS		✓
JOINT_USER_FORCE	✓	
LAGRANGE_IN_SOLID		✓
LINEAR	✓	
NODAL_RIGID_BODY	✓	✓
NODE_SET	✓	✓
POINTS		✓
RIGID_BODIES	✓	✓
RIGID_BODY_STOPPERS		✓
RIVET	✓	✓
SHELL_TO_SOLID		✓
SPLINE	✓	
SPOTWELD	✓	✓
SPR		✓
SPR2		✓
SPR3		✓
TIE-BREAK		✓
TIED_NODES_FAILURE		✓





Example of the **CREATE/EDIT** panel (**GENERALIZED_WELD**)



Example of the keyword editor (**EXTRA_NODES_NODE**)

The methods for creating and modifying constrained entities falls into two categories.

- **EXTRA NODES, NODAL RIGID BODIES, NODE SETS** and **RIGID BODIES**
  These are "scalar" panels, in which a single definition is created or edited.

- **GENERALIZED WELDS, RIVETS** and **SPOTWELDS**
  These provide "scalar" creation and editing as above.
  Also "quick create" functionality to create a sequence of items.
  Also "edit range" functionality to permit edits to apply to a range of items.

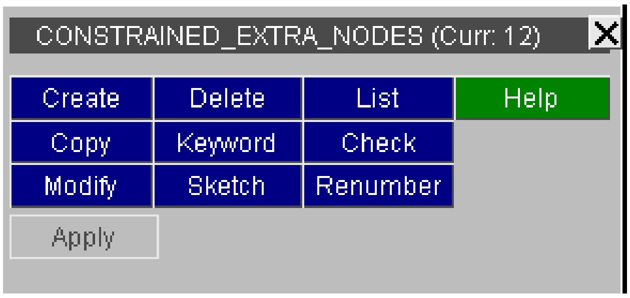
To illustrate the two categories **EXTRA_NODES** and **GENERALIZED WELDS** are presented as examples

CONSTRAINED_EXTRA_NODES Extra nodes on rigid bodies

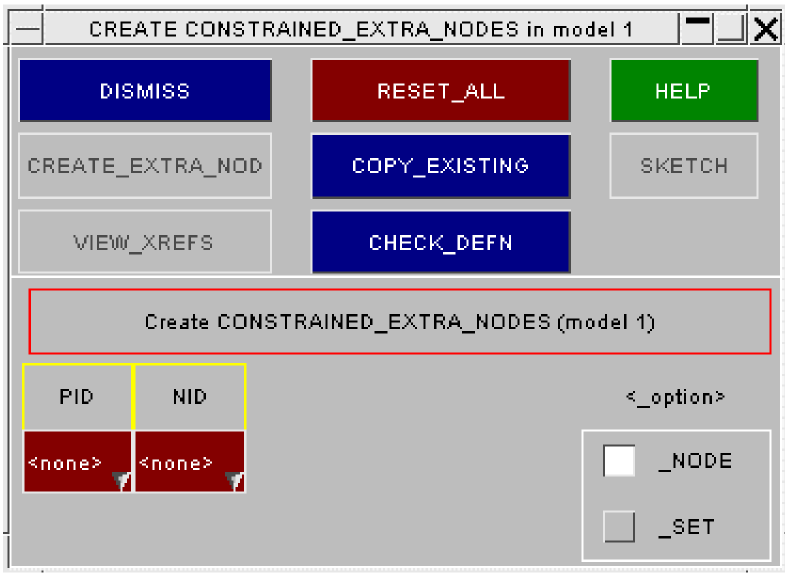
Constrained extra nodes allow a single node (**`EXTRA_NODES_NODE`**) or a group of nodes in a node set (**`EXTRA_ NODES_SET`**) to be attached to a rigid body in LS-Dyna.
This figure shows the main extra nodes menu.

The functions currently available have their standard meanings. (See 5.0.1)

CREATE and **MODIFY** apply only to single definitions ("scalar" editing)



CREATE Making a new extra node.



This figure shows the initial state of the extra nodes creation panel: no part or node has been given yet, so both are highlighted red.

The **<_option>** radio buttons can be used to change whether a **EXTRA_NODES_NODE** or a **EXTRA_NODES_SET** is created.

The part and the node (or node set) numbers can be typed directly into the text boxes. If the value is valid (for example the part must be rigid) the box will turn blue, otherwise an error message will be displayed indicating what is wrong. Alternatively, the popup menus can be used to pick a part, node or node set off the screen, or to select a part, node or node set from a list.

Once the required fields are filled in the **SKETCH** and **CREATE_EXTRA_NODE** buttons will become active.

CREATE_EXTRA_NODE saves the new definition permanently.

COPY Copying existing extra nodes(s) to make a new one(s).

You can **COPY** any number of extra node definitions, in multiple models.

For each model the <n> extra nodes chosen in that model are copied using labels <previous highest + 1> to <previous highest +n>, there is currently no control available over the new labels assigned.

MODIFY Modifying the attributes of an existing extra node.

This functions in exactly the same way as **CREATE**, using the same panels as in the figure above. The only difference is that the initial state of the panels is already set with the attributes of the extra node to be modified.

KEYWORD Invoking the standard keyword editor.

The standard keyword editing panel is set up.

DELETE Deleting existing constraints

The **DELETE** operation deletes the **EXTRA_NODES** definitions.

- If **DELETE_RECURSIVE** is switched on any nodes, node sets and parts, referenced by the extra nodes to be deleted are marked for deletion.
- If recursive deletion is not used only the extra node definitions themselves are removed.

Note also that the standard deletion rules described in Section 6.4.1 still apply: parts, nodes and node sets will only be deleted if nothing else (which is to remain) depends on them.

SKETCH Sketch the chosen extra node on the current image

SKETCH allows the user to select and sketch individual extra nodes on the current graphics image. Extra nodes are drawn with a dashed line from the node (or dashed lines from each node in the node set) to the centre of the rigid body.

CHECK Checking for errors

Runs the standard checking function on the selected extra nodes. Each extra node will be listed either as "OK", or a summary of the errors encountered will be printed. (This is the same as the **CHECK_DEFN** command during extra node editing.)

RENUMBER Changing labels.

Raises the standard renumbering panel for constraints in the chosen model, allowing you to renumber some or all of them.

As constraints do not have labels in LS-DYNA the usefulness of this is limited.

END_CONSTRAINED returns the user to the main **CONSTRAINED** box.

CONSTRAINED_GENERALIZED_WELD "Generalised" welds of various types.

Generalized welds in LS-Dyna are used to represent spotwelds between more than 2 nodes and fillet welds. At present only the creation and modification of spotwelds is implemented.

The main generalized weld menu has identical options to the extra nodes menu and the functions currently available have their standard meanings. (See section 5.0.1)

CREATE Making a new generalized weld.

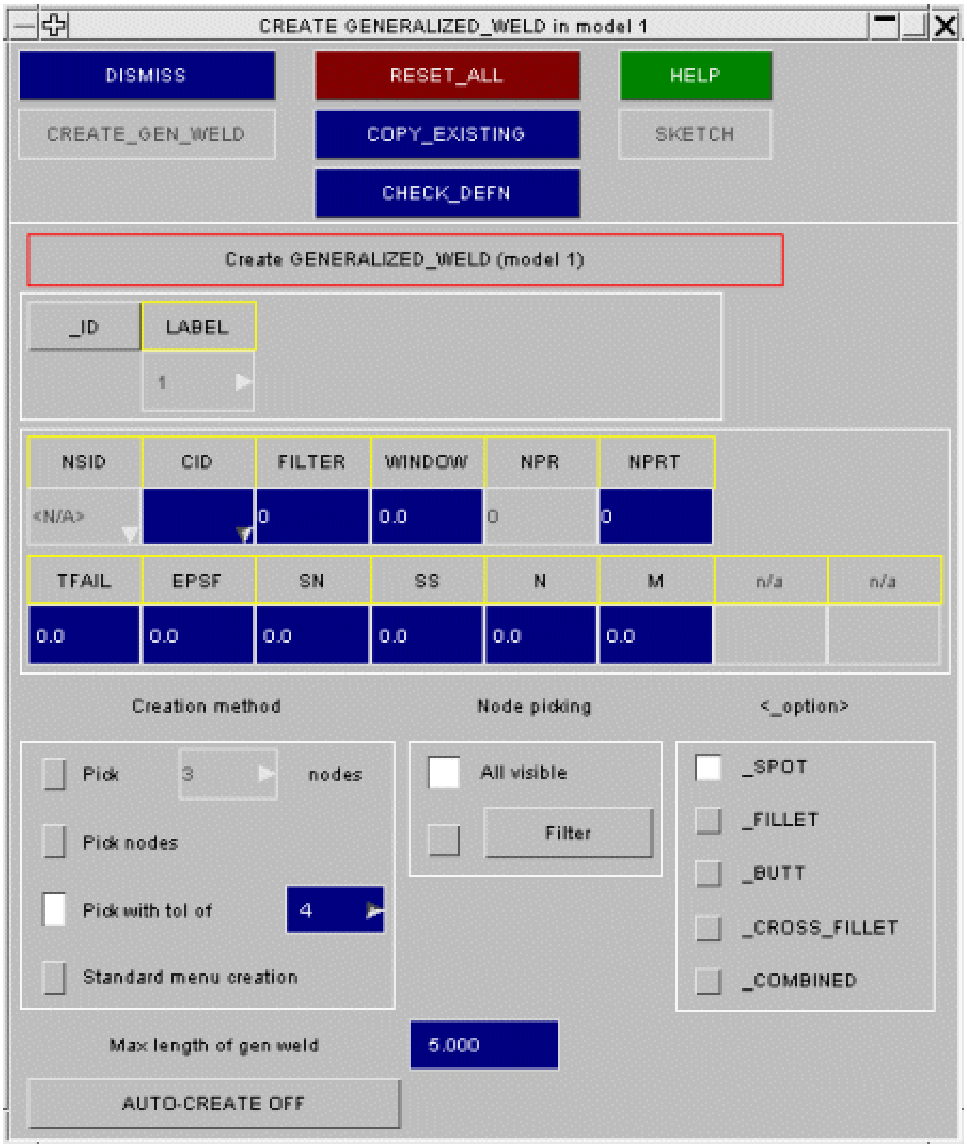
This figure shows the initial state of the generalized weld creation panel.

No node set has been given yet, so it is labelled <**N/A**>.

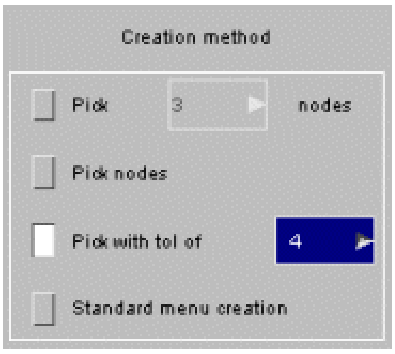
The **<_option>** radio buttons are greyed out as at present only **GENERALIZED_WELD_SPOT** can be created.

The various parameters for the generalized weld spot can be typed directly into the text boxes (eg **FILTER, SN, SS** etc). If the value is valid it will be displayed in the text box, otherwise an error message will be displayed indicating what is wrong.

Popup menus can be used to pick a coordinate system.

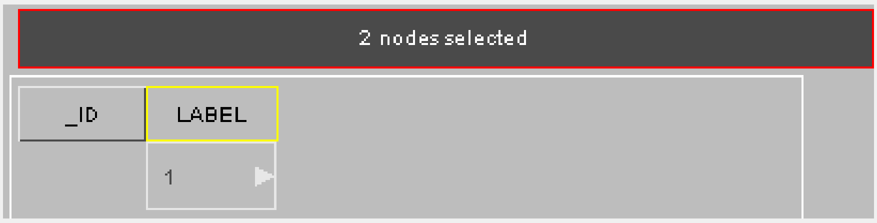


There are 4 methods available for creating the welds. For all methods except 'Pick nodes', there is an **AUTO-CREATE** button which will automatically create the generalized weld once the necessary information has been given. There is also a maximum length of generalized weld button which sets the maximum permissible length of weld. If you try to create a weld greater than this length, a warning will be given and creation stopped. If you had the **AUTO-CREATE** option on it will be turned off to give you a chance to do something about the problem such as changing the nodes or increasing the tolerance.



(1) Pick n nodes

If this option is selected you can pick nodes directly off the screen. The default number of nodes is 3 but you can easily change this by typing in a new number in the box (in the range 2 to 100) or by using the popup menu to select commonly used values. Once you have reached the number of nodes the **CREATE_GEN_WELD** and **SKETCH** buttons will be ungreyed, or if you have **AUTO-CREATE** on, the weld will automatically be created. If **AUTO-CREATE** is off and you try to pick more nodes they will be ignored and a warning given.
As you pick nodes the feedback button on the creation panel changes to indicate how many you have picked.



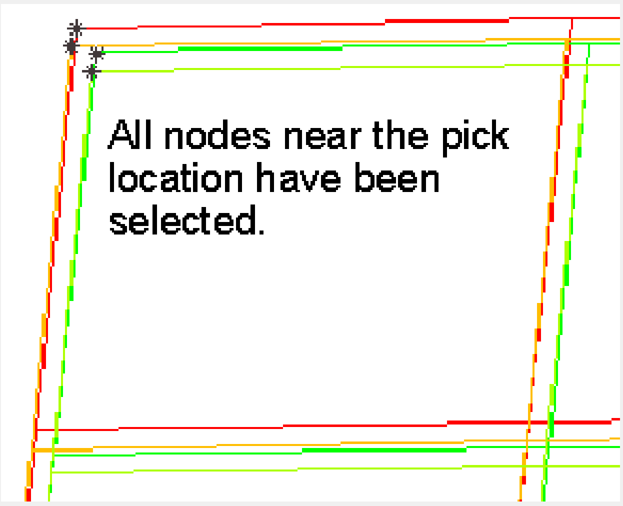
(2) Pick nodes

This is similar to method 1 but there is no limit on the number of nodes. Once you have picked 2 nodes the **CREATE_GEN_WELD** button will be ungreyed.
The **AUTO-CREATE** option cannot be used with this method. As you pick nodes the feedback button on the creation panel changes to indicate how many you have picked.

(3) Pick with tolerance of n

This method can be used to select all the nodes within a certain tolerance of a screen pick. The tolerance can be changed by typing in a number (in the range 1 to 7) or using the popup.
The nodes which you selected are sketched on the screen and the feedback button on the creation panel changes to indicate how many you have picked.

Care must be taken with this option to ensure that the weld geometry in a tightly meshed area is sensible.

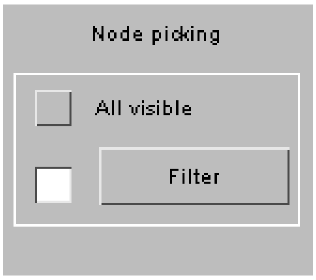


(4) Standard menu creation.

This method is the usual (scalar) method in PRIMER for creating an entity. Either type the values into the text boxes and/or use the popup menu to create or select a node set for the generalized weld.

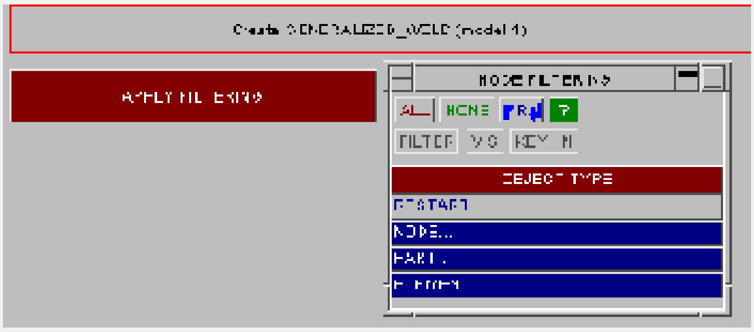
Unlike other editing panels in PRIMER which are closed when the entity is created, the generalized weld creation panel will remain on the screen until the **DISMISS** button is pressed. Additionally all the values which you type in for the failure parameters are remembered so that when creating multiple generalized welds the information only has to be typed in once. This information is also remembered when you dismiss the window.

Node picking: filtering the nodes that are picked.
By default any visible nodes in the currently selected model can be picked for use in the generalized weld. This can be changed by using the **Node picking** option. If this is set to:



then If the filter option is chosen then

All visible	Any visible nodes in the currently selected model can be picked.
Filter	A sub-menu allows you to filter which nodes can be picked. The most useful option is to select the **PART(s)** from which you want to pick the nodes. This permits you to limit selection within a dense mesh to just the panels you want to weld. For example you could filter the nodes so that only nodes on 2 panels can be picked. All the other panels are still visible on the screen, but they will be ignored when picking nodes.



MODIFY Modifying the attributes of an existing generalized weld.

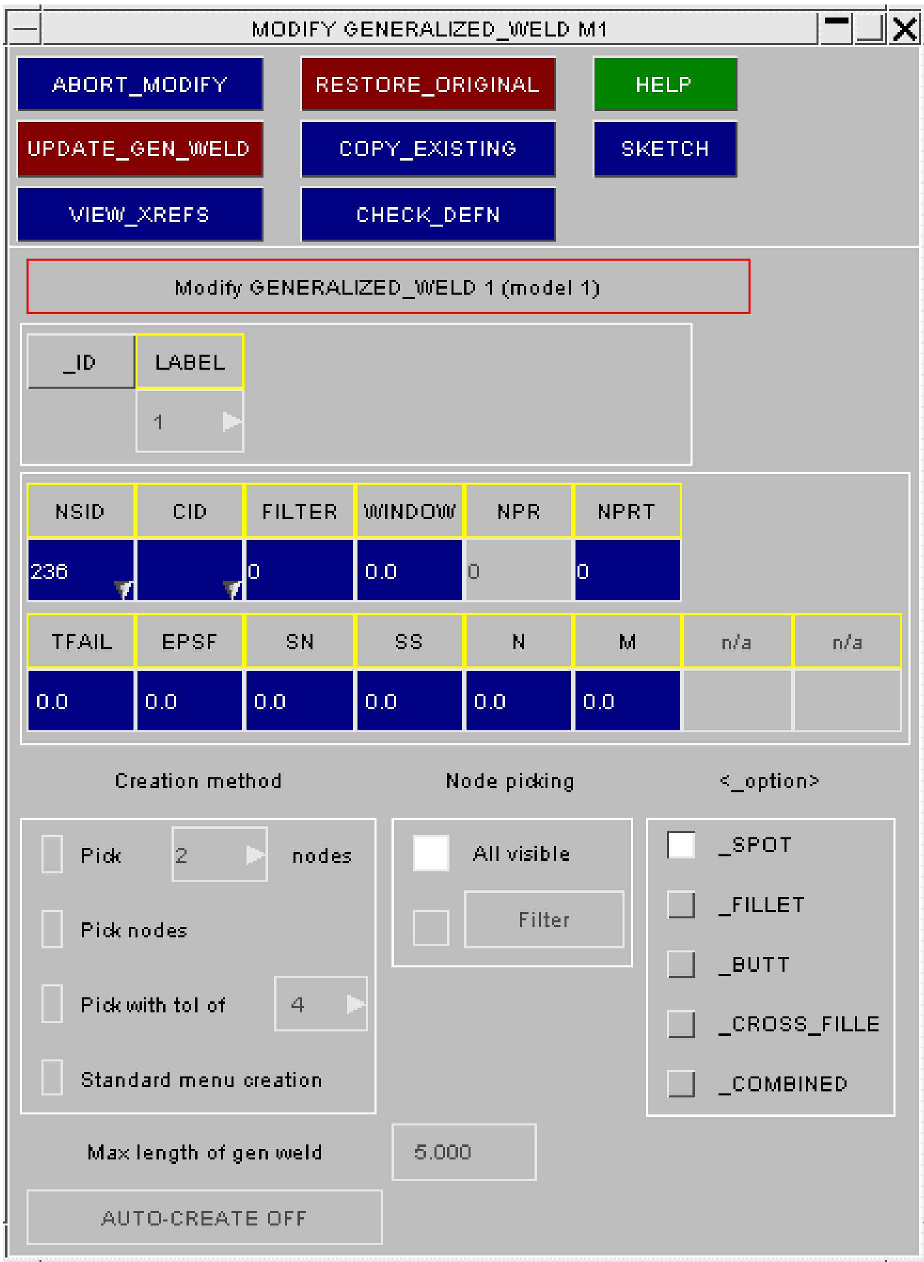
Unlike most editing panels in PRIMER it is possible to modify more than one generalized weld at a time. If only one is being modified then all attributes of the generalized weld including the failure criteria and the node set can be modified (see figure above).

None of the creation options are valid when modifying generalized welds so they are all greyed out. The node set can be changed or modified using the popup menus.

However when a range of > 1 welds has been selected then: (fig to right)

- The node set (**NSID**) is unavailable for editing.
- The default properties are taken from the first weld chosen, and will be applied to all welds, possibly modified, when you **UPDATE** the panel.
- The welds may be selected from multiple models, since only attributes, which are not model-specific, are editable.

Once all the modifications are complete the **UPDATE_GEN_WELD** button saves the new values into the database.



KEYWORD Editing welds using the generic Keyword editor.

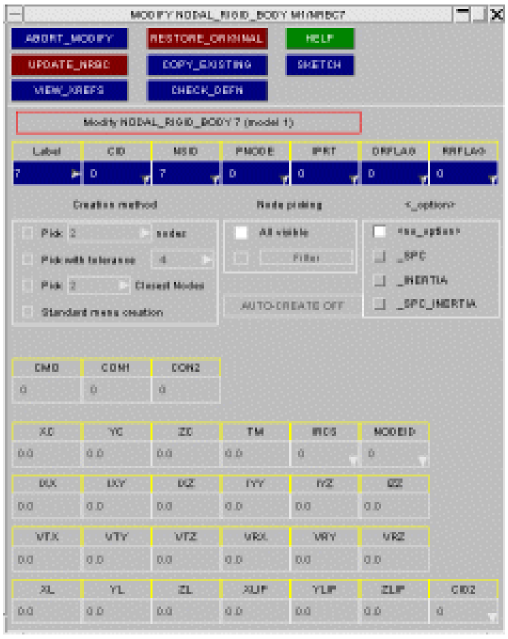
All weld types may be edited using the **KEYWORD** editor, but for the **CROSS_FILLET** and **COMBINED** types only the initial "attributes" rows are editable since the remaining rows are open-ended in length.

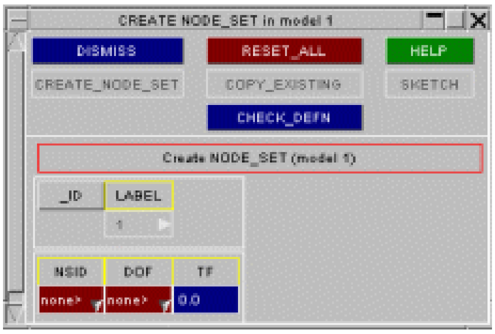
The keyword editor also provides a more selective way than **MODIFY** <range> above of changing properties over a range of welds

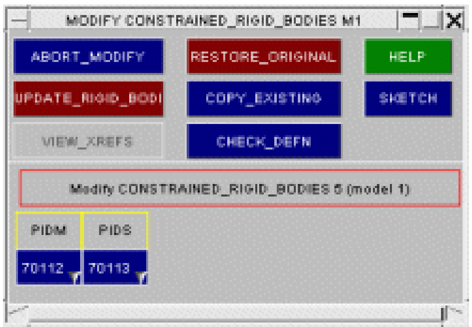
COPY	**Copying existing generalized weld(s) to make a new one(s).**	These functions work in exactly the same way as for **EXTRA_NODES**
DELETE	**Deleting existing generalized welds**	
SKETCH	**Sketching welds on the current image**	
CHECK	**Checking for errors**	
RENUMBER	**Renumbering welds**	

CREATE/EDIT panels for other constrained types

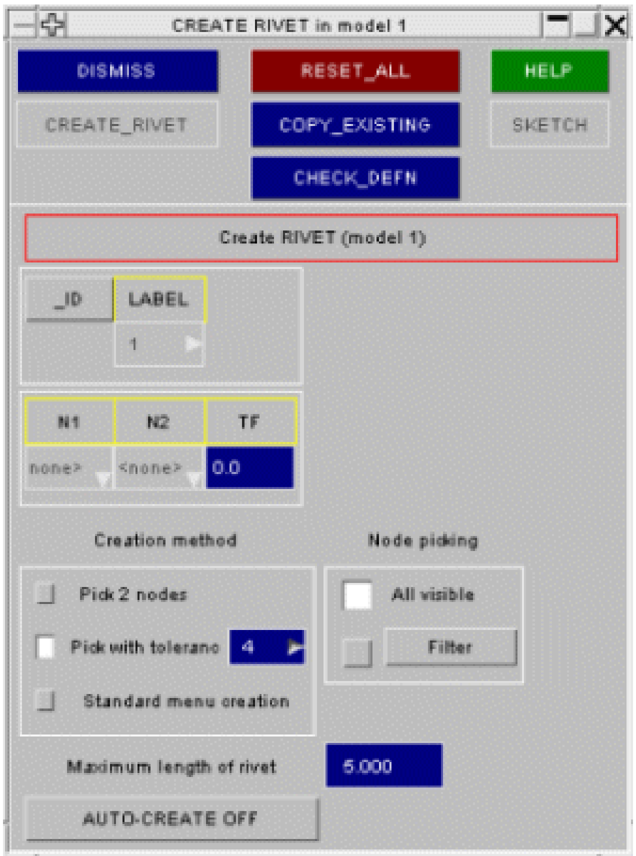
The **NODAL_RIGID_BODY, NODE_SET** and **RIGID_BODIES** creation menus are similar to the **EXTRA_NODES** menu and use the same principles.

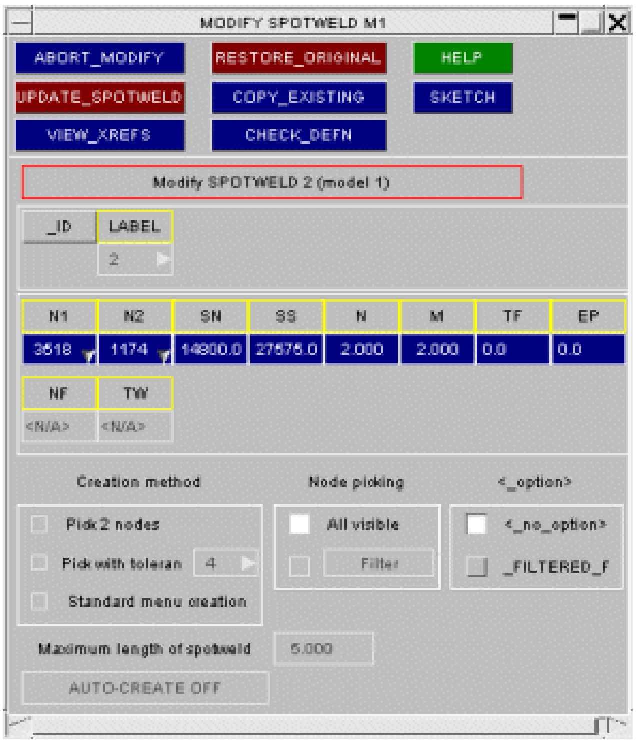




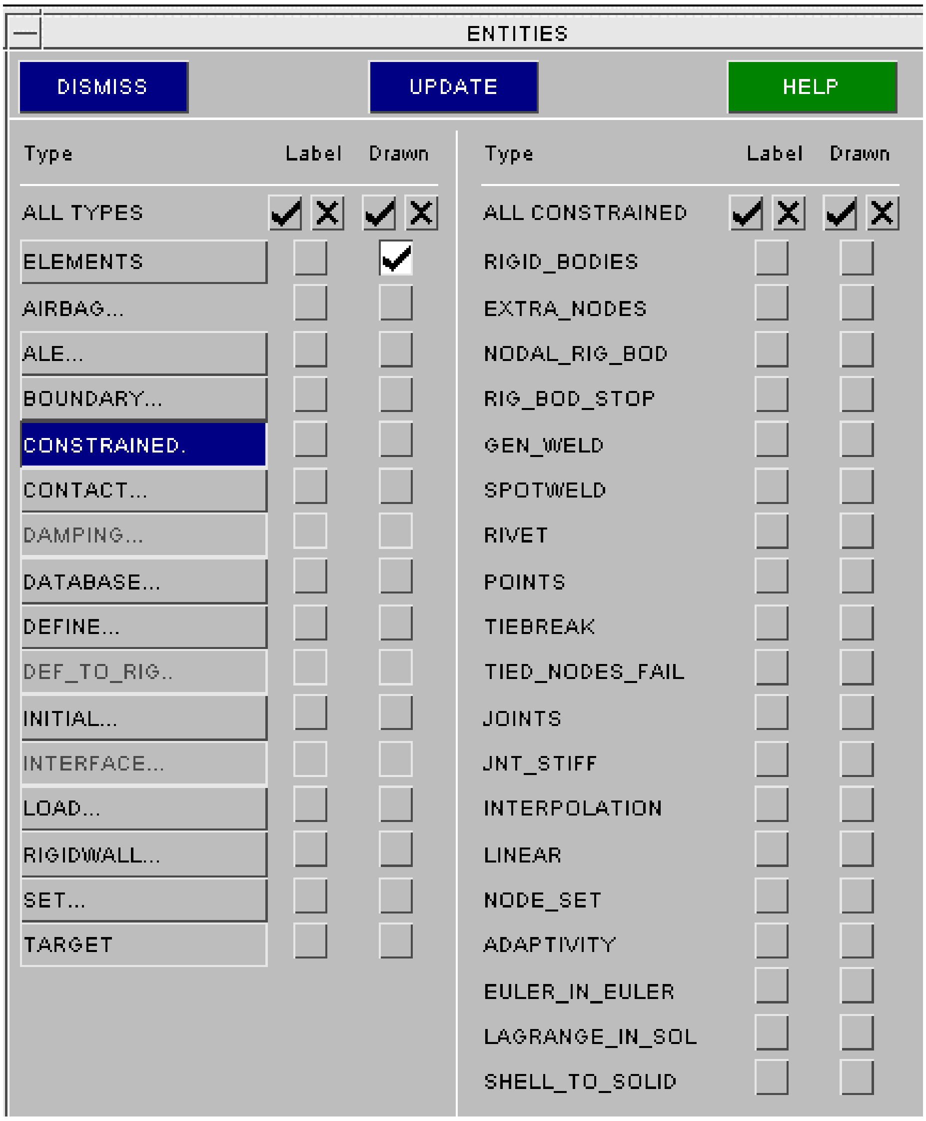


The **RIVET** and **SPOTWELD** creation menus are similar to the **GENERALIZED_WELD** menu and use the same principles.





Visualisation of ***CONSTRAINED** items



All **CONSTRAINED** items except **_LINEAR, _RIGID_BODY_STOPPERS** and **_LAGRANGE_IN_SOLID** are explicitly drawn and labelled, and all sub-types can have their constituent sets, parts or whatever displayed. Visibility is controlled by the **ENT**ity Viewing , **CONSTRAINED** panel.

Labelling of ***CONSTRAINED** items within PRIMER.

LS-Dyna has optional labels for some *CONSTRAINED items (e.g. *CONSTRAINED_NODE_SET_ID): the conversion from "keyword" to "formatted" input that precedes every LS-Dyna analysis converts them from discrete definitions to attributes applied to other items.

For internal consistency, PRIMER assigns new labels to everything that does not already have a label and that can be defined "once or many times", so *CONSTRAINED definitions are given labels based on their order of appearance in the keyword input file.

PRIMER's labels:

- May safely be ignored - you don't have to worry about them if you don't want to!
- Are treated sequentially, starting at 1. (Thus **CNST_1, CNST_2, ... CNST_n**)
- Are not grouped by sub-type: **CNST_1** might be a **NODE_SET, CNST_2** a **JOINT** - they are based solely on the order in which they appear in the input deck. Each *CONSTRAINED definition encountered gets the next label in the sequence.
- Are used in selection menus (eg for blanking, deletion, etc). Are also used in the output deck when defining what is referenced by what.

Because PRIMER groups ***CONSTRAINED** definitions by type when they are written out or copied (all **JOINT**s together, etc), and because labels are assigned in order of appearance, the labels assigned to these items may change when decks are written out and read in again, unless the _ID option is used.

***CONSTRAINED_JOINT** specific annotate tools.

There is a specific tool for joint constained types called **Annotate**. This is available through the main CONSTRAINED_JOINT panel or on individual CONSTRAINED_JOINT edit panels. This tool annotates the joints with nodal positions and rigid body information. It can be useful when creating or checking joints to ensure nodes and rigid bodies are defined in the correct order.

CONTACT: Defining Contact Surfaces.

CONTACT(_sliding)
- Top level menu
- Creating a new contact
- Copying
- Editing
- Deleting

Penetration Checking Visualisation

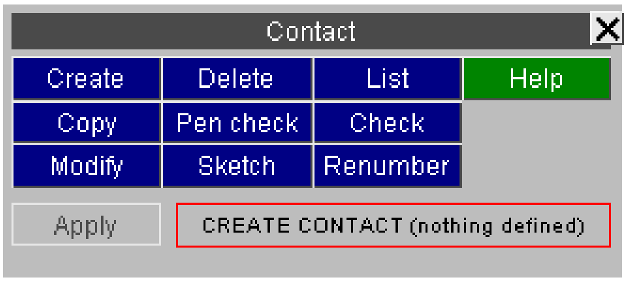
The ***CONTACT** keyword has 10 sub-categories. The following can be edited in Primer:

***CONTACT**	Traditional LS-DYNA contact types.
***CONTACT_1D**	One-dimensional "slidelines" for reinforcing bars in concrete.
***CONTACT_ENTITY**	Geometric definitions of rigid bodies. Generally used when DYNA is coupled to other codes.
***CONTACT_GUIDED_CABLE**	A sliding contact that guides 1D elements.
***CONTACT_INTERIOR**	Internal contact for solid elements. Used to prevent crushable materials collapsing in on themselves and turning inside-out.
***CONTACT_RIGID_SURFACE**	A rigid surface contact definition.

The following contact sub-types cannot be edited in PRIMER.

***CONTACT_AUTO_MOVE**	Moves the master surface in acontact definition to close the initial gap between the master and slave surfaces.
***CONTACT_COUPLING**	Defines a coupling surface for MADYMO to couple LS-DYNA with deformable and rigid parts within MADYMO
***CONTACT_GEBOD**	Contact between a "Gebod" dummy and the structural FE mesh.
***CONTACT_2D**	Two dimensional contact for use with 2D and Axisymmetric elements.

This figure shows the main contact menu.

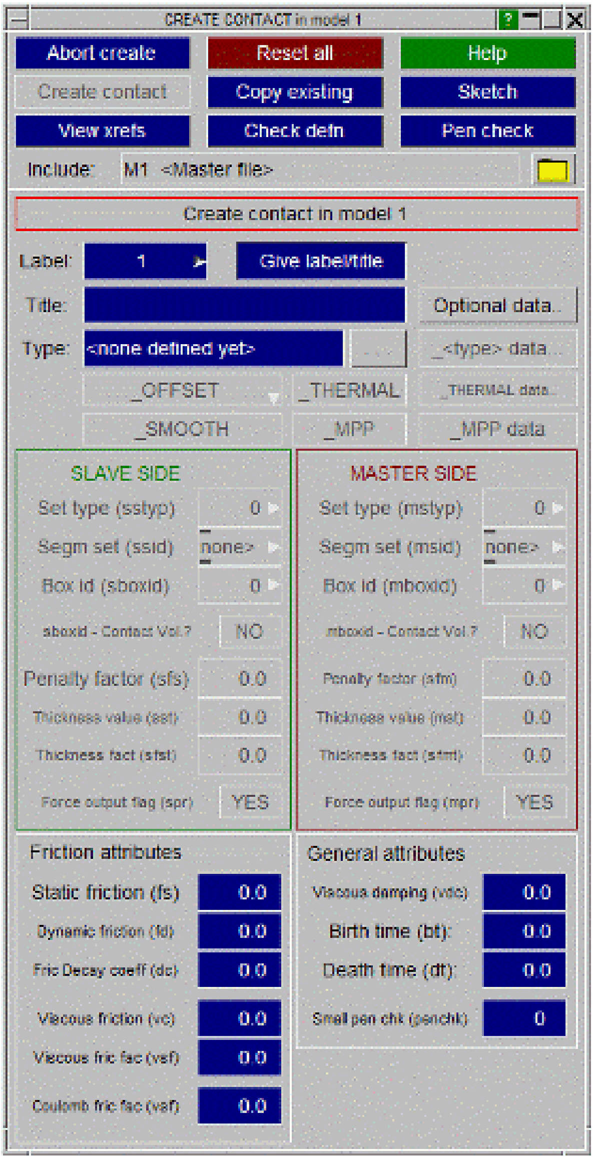


PEN_CHECK Runs the penetration checker for contacts.

> The penetration checker detects initial penetrations and crossed edges. It displays them graphically and also generates "null beams" on the crossed edges to identify them during external remeshing. Its use is summarised below.

The functions currently available have their standard meanings. (See section 5.0.1)

CREATE Making a new contact surface.

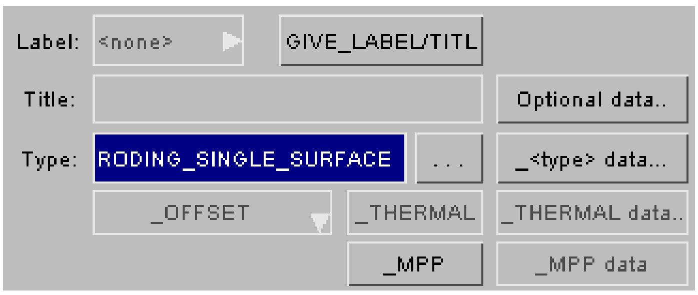


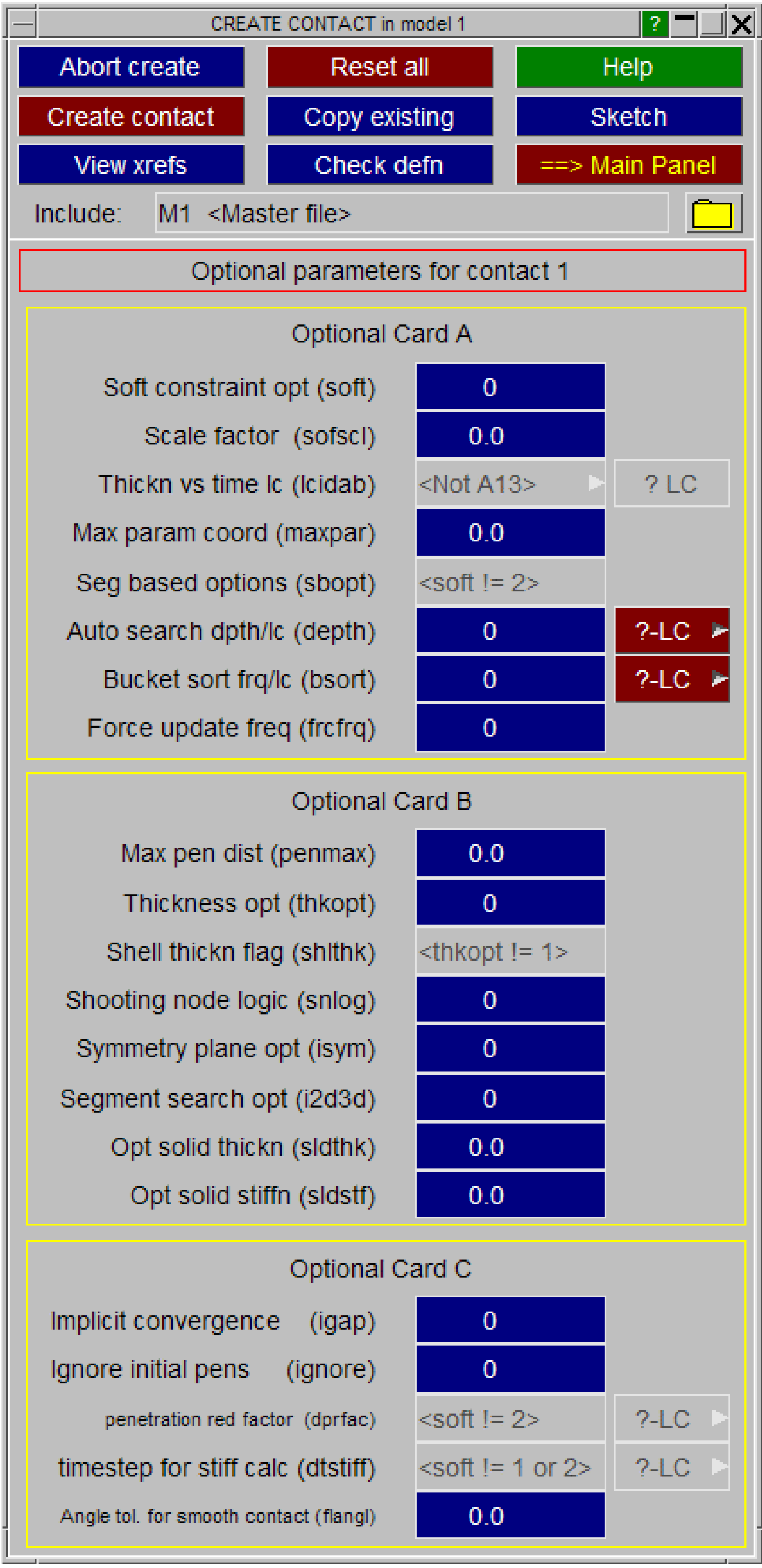
The figure on the left shows the initial state of the contact creation panel: no type has been given yet, so both master and slave side definition areas are greyed out

The figure on the right shows the same panel now that an **ERODING_SINGLE_SURFACE** contact type has been defined. This is a single surface contact, so only the slave side is available; had it been a two-sided contact type then both master and slave sides would have been available for input.

The data on the front panel here is required (even if zero) for all "sliding" contact types, but there are extra, optional data entries at the top of the panel that may be required:

GIVE_LABEL/TITLE Allows you to define the name and label.
LS-Dyna has an optional **_TITLE** suffix to the ***CONTACT_<type>** keyword that is enabled if this is defined.





Optional data...

Defining data on optional *CONTACT cards A, B and C

All contact types may have additional data for optional cards A, B and C in LS-DYNA.

The exact options available will depend upon the contact type, for example the Airbag thickness vs time loadcurve is not valid for the (eroding) contact type used in this example.

The red [**?-LC**] symbol on this panel is used to denote the fact that a loadcurve value is implied in this context if a negative number is input, whereas zero or a positive number is a simple constant. Pressing this button will give a selection menu of possible loadcurves.

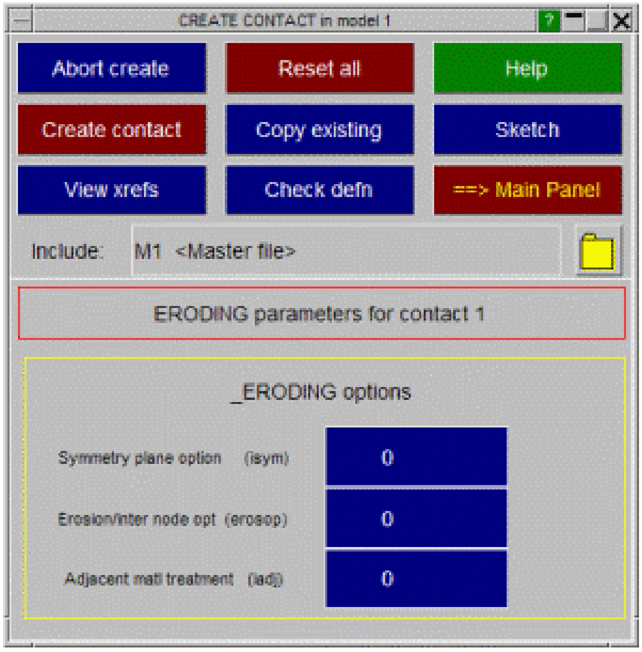
(A green [**?LC**] means that a positive value implies a loadcurve.)

When you have completed entries on this panel **==> MAIN PANEL** returns you to the master panel shown above.

_<type> data... Special data input for this contact surface type.

A few contact surface types have mandatory <type> data cards. If this is the case the [_<type>_data...] button on the main panel will be live.
In this example the `_ERODING_...` contact type requires the extra data shown here; but other contact types will require different data.

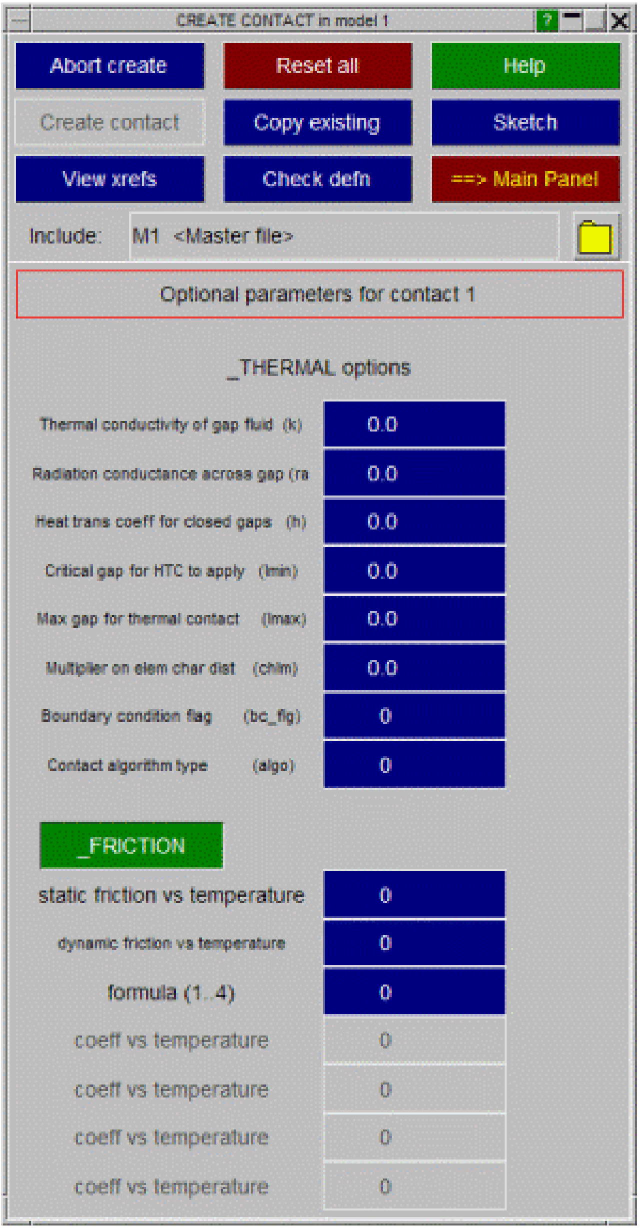
As with the optional data cards, **==> MAIN PANEL** will return you to the main contact input panel.



_THERMAL data... Special input for thermal contact types.

Where a contact surface type supports the **_THERMAL** or **_THERMAL_FRICTION** qualifier, and this is turned on, the **_THERMAL data...** button may be used to enter the thermal parameters.

As with the other optional card panels the **==>MAIN PANEL** button will return you to the master contact definition panel.



CREATE_CONTACT Saving the contact definition.

Once you have entered the minimum amount of data required to define this contact the **CREATE_CONTACT** button will be made live, and this permits you to save this definition. (If it is not live the missing fields will be highlighted in red.) The definition will be checked and any errors listed, an then it will saved permanently in this model.

Until you press this the definition remains volatile, and will be lost if you exit this panel in any other way.

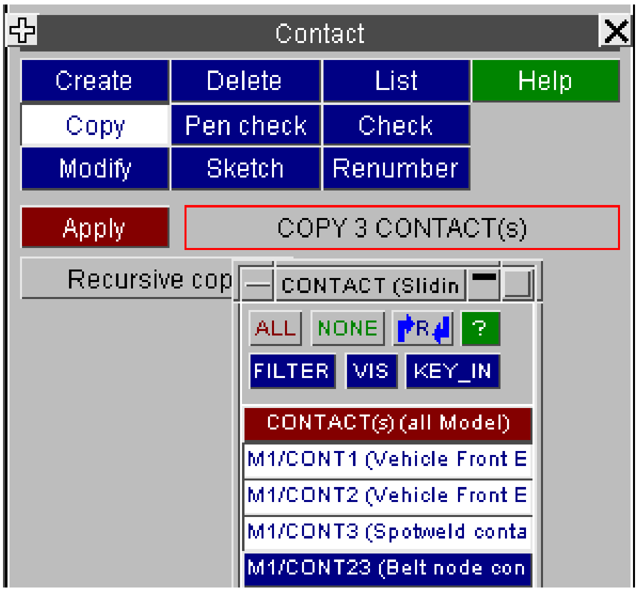
COPY Copying existing contact surface(s) to make a new one(s).

You can **COPY** any number of contacts, in multiple models.

Selection of subject models is from the menu shown in this figure.

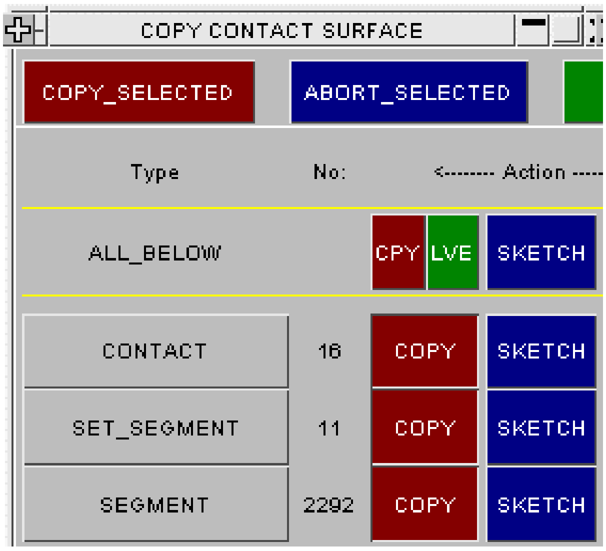
When **APPLY** is pressed you are asked to confirm what is to be copied, and then the operation is carried out.

For each model the <n> contacts chosen in that model are copied using labels <previous highest + 1> to <previous highest +n>, there is currently no control available over the new labels assigned.



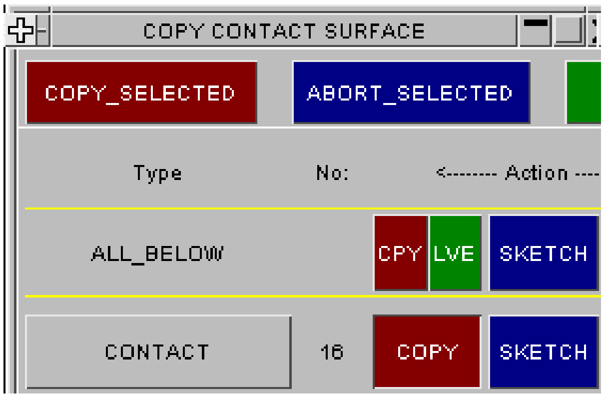
RECURSIVE_COPY Controlling the extent of a copy operation for segment based contacts.

If "recursive" copying is used then any segment sets, (and thus their associated segments), in the subject contacts are duplicated; and the newly created contacts will reference these duplicates. Without "recursive" copying the new contacts will reference the same segment sets as the old ones. This is illustrated in the examples below, using the same 3 contacts:



With **RECURSIVE_COPY** selected.

In the left box recursive copying has picked up segment sets and their constituent segment.



With no recursive copying.
In the right box, without recursive copying, only the contact definitions are copied.

Note that for contacts recursive copying only affects segment sets and their segments. A contact defined by parts, elements or nodes will not attempt to copy these in the "recursive" case.

MODIFY Modifying the attributes of an existing contact surface.

This functions in exactly the same way as **CREATE**, using the same panels as above. The only difference is that the initial state of the panels is already set with the attributes of the contact surface to be modified.

DELETE Deleting existing contact surfaces

The **DELETE** operation works exactly the same way as **COPY** described above, except that the chosen surfaces are deleted. The **DELETE_RECURSIVE** switch also operates in a similar fashion:

- If **DELETE_RECURSIVE** is switched on any segment sets, plus their segments, referenced by the contacts to be deleted are marked for deletion.
- If recursive deletion is not used only the contact definitions themselves are removed.

Note also that the standard deletion rules described in Section 6.4.1 still apply: contacts and/or segment sets will only be deleted if nothing else (which is to remain) depends on them.

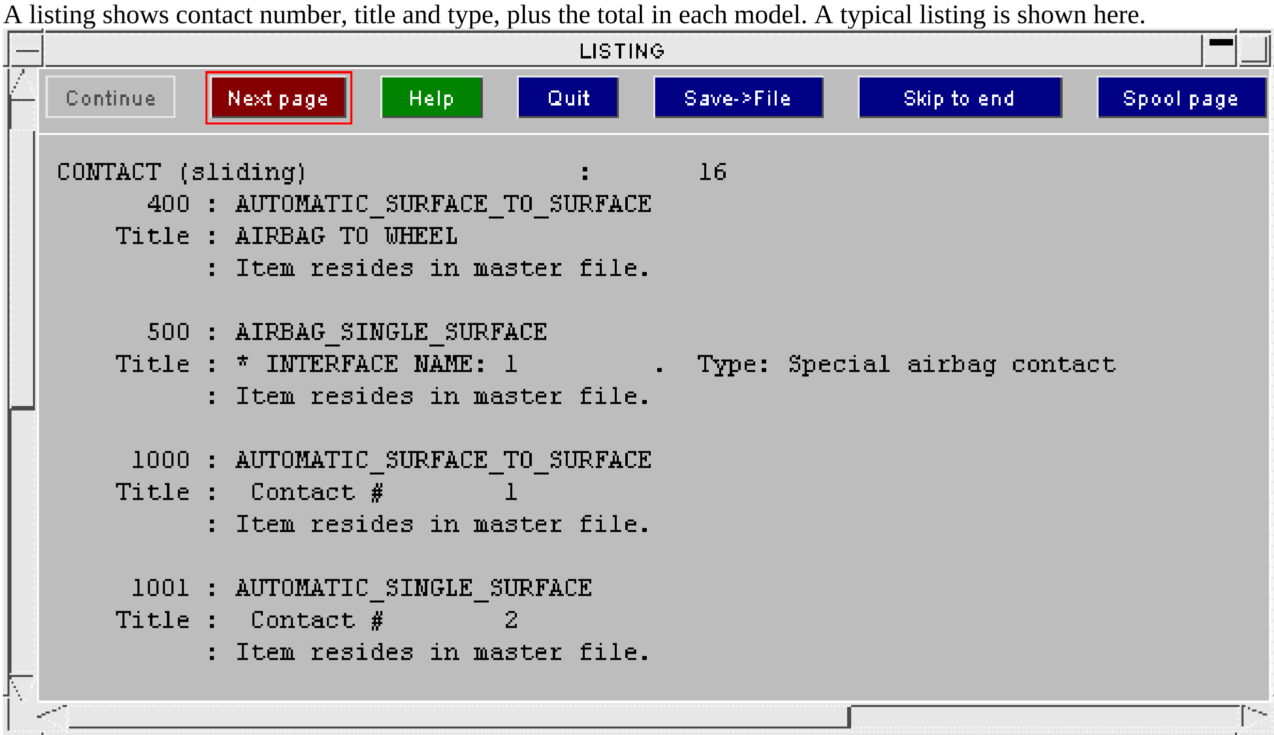
SKETCH Sketch the chosen contact surfaces on the current image

SKETCH allows the user to select and sketch individual contact surfaces on the current graphics image. Contacts are drawn in terms of the parts, elements, segments and nodes that they include.

LIST Summarise the contents of contacts

LIST allows the user to select any or all contacts and to list a summary of their attributes to the screen.

A listing shows contact number, title and type, plus the total in each model. A typical listing is shown here.



CHECK

Runs the standard checking function on the selected contacts. Each contact will be listed either as "OK", or a summary of the errors encountered will be printed. (This is the same as the **CHECK_DEFN** command during contact editing.)

RENUMBER

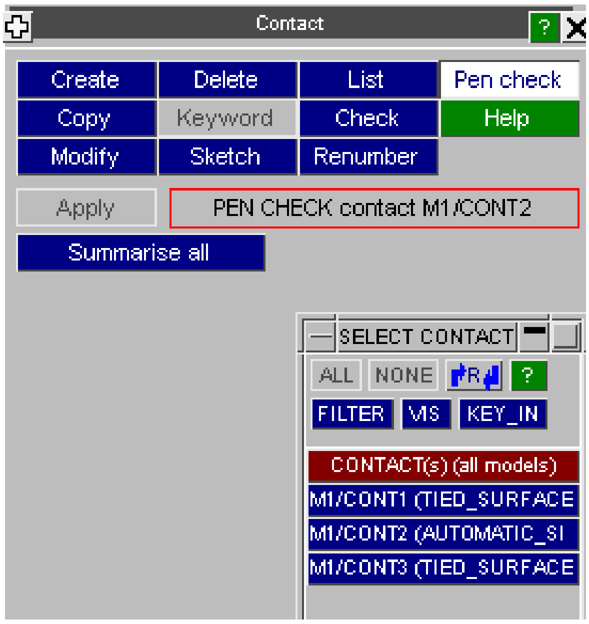
Raises the standard renumbering panel for contacts in the chosen model, allowing you to renumber some or all of them. To renumber a single contact it may be easier to **MODIFY** it and update its label.

PEN_CHECK Checking initial penetrations and crossed edges

The contact penetration checker may be run from:

- The **PEN_CHECK** command in the top level menu here
- The **PEN_CHECK** command in any contact create/edit panel
- The main **MODEL > CHECK** command as part of model checking.

Calling the penetration checker from the top menu panel.



SUMMARISE_ALL

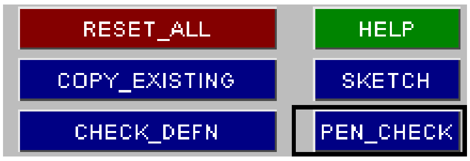
Runs the checker on all contacts in turn in "summary" mode, producing a listing of penetrations and crossed edges for each one, but no details. Note - the number of penetrations may exceed the number of penetrating nodes (which is reported in other contexts) as there may be >1 penetration on a node.

This is useful as an initial scan to look for problems meriting further attention.

<Explicit selection from menu>

Runs the checker in detailed mode on the contact selected.

Calling the penetration checker from the Create/Edit panel

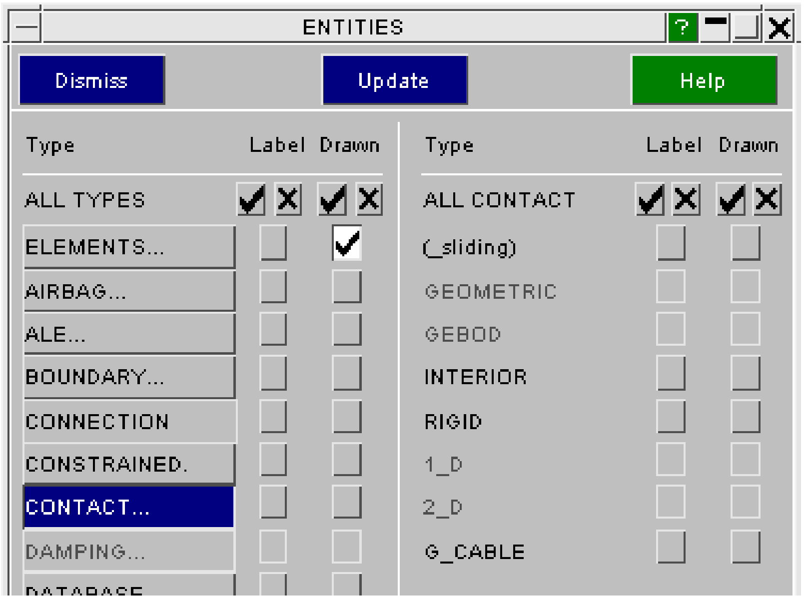


The **PEN_CHECK** command in this context runs the checker on the current (scratch) contact definition currently being edited.

Contact penetration checking is covered in more detail in section 5.3

Visualising contact surfaces

Contacts may be selected for explicit display in the **ENT**ity Viewing panel panel.
They are drawn in terms of the parts, elements, nodes and segments that define them. Where boxes are used to delimit contacts these are displayed if switched on in the "Associated data" section.



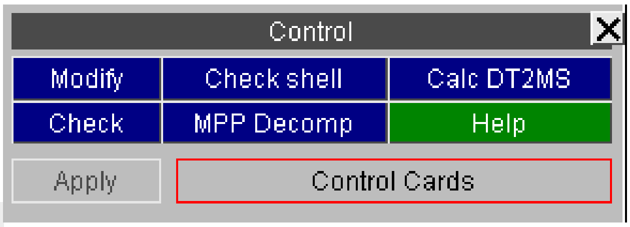
They may also be drawn via the **SKETCH** functions above.

CONTROL: Defining Analysis Control Cards.

The ***CONTROL** keyword in LS-DYNA refers to the unique keywords which control the main parameters of an analysis.
Each control card occurs either once or not at all, and none are labelled, therefore the PRIMER control and editing panels are slightly non-standard. Merging control cards presents some special problems - see the notes on this below.

Note for users of PRIMER prior to release 8.2: The layout of the control card editor has been totally revised, since the increasing number of control cards (44 in LS960) made the original panel unwieldy. In addition the distinction between "create", "modify" and "copy" modes has been removed and replaced by a single panel that performs all these functions.

The main **CONTROL** panel.



Since each control card can only exist once or not at all in a model the concept of separate "create", "modify", "copy", etc modes has been removed.

MODIFY	Maps the control card editing panel, in which cards can be created, modified, deleted and copied from other models.
CHECK	Runs the standard check routines on control cards.
CALC DT2MS	calculate relationship between %age added mass and timestep

As with all PRIMER **MODIFY** functions edits and other changes only take place on a "scratch" definition, which is only made permanent when explicitly **UPDATE**d.

CONTROL_CHECK_SHELL (**CHECK SHELL**) and **CONTROL_MPP_DECOMPOSITION_TRANSFORMATION** (**MPP DECOMP**)are open ended cards with their own edit panels which can be accessed directly or via **MODIFY**

CALC DT2MS is only active if DT2MS on the *CONTROL_TIMESTEP is set to 0 or less.

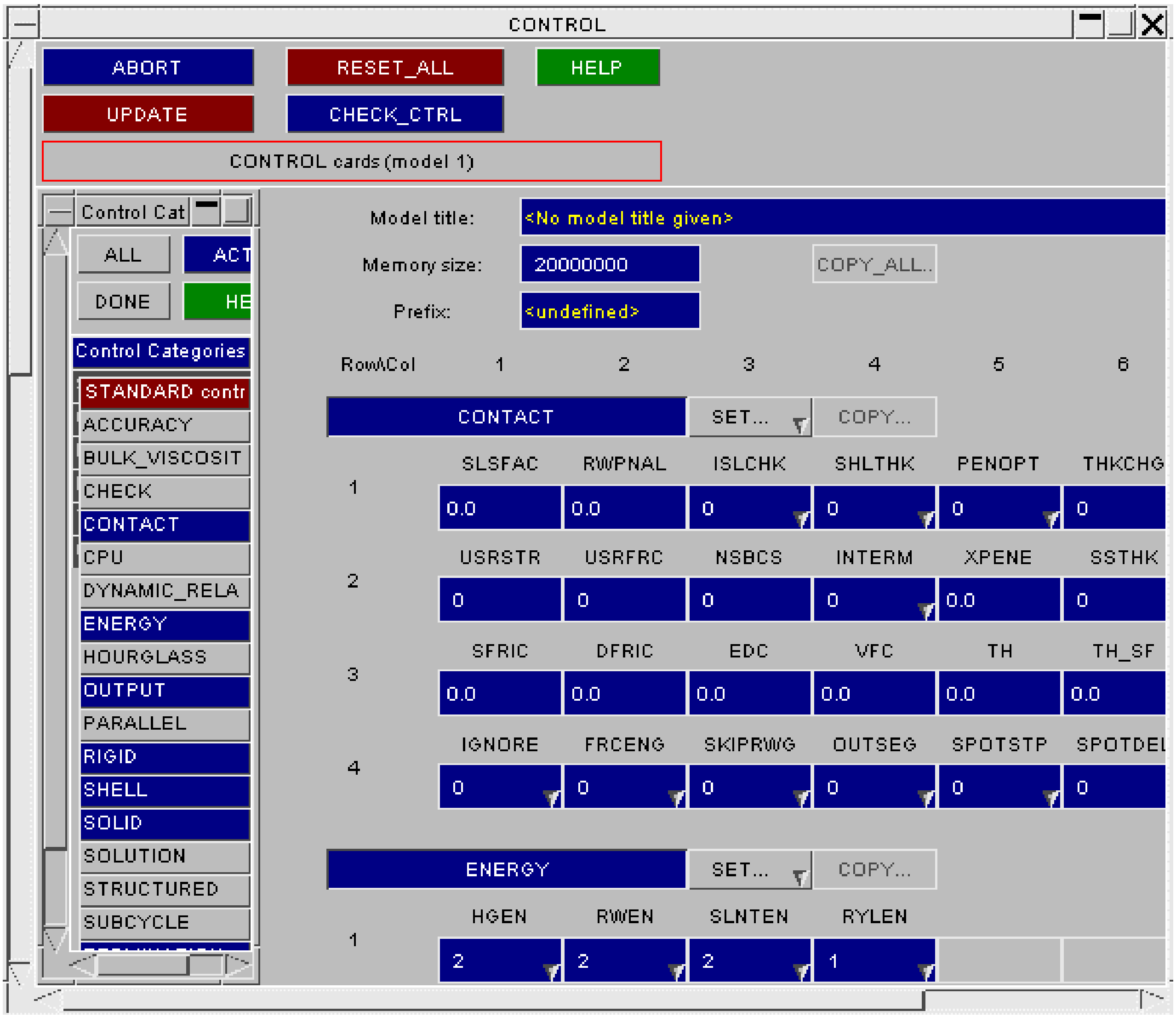
MODIFY: Creating, editing, deleting and copying control card definitions.

The single panel shown below is used to carry out all these operations.

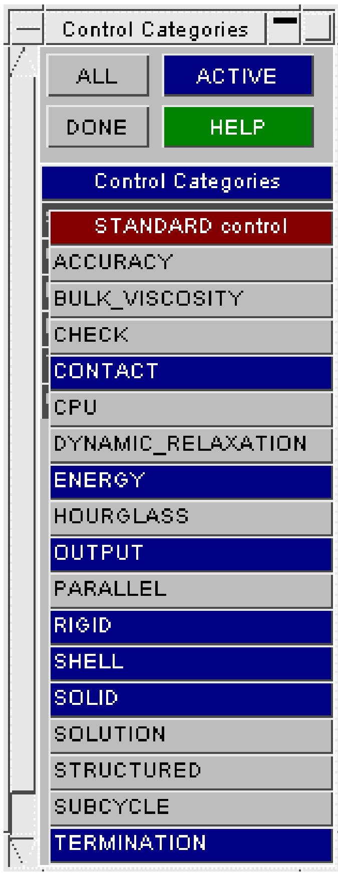
For ease of selection, Control card are now grouped into 6 categories.

ALL available Control options will be displayed (but not activated) by pressing **ALL** and all the active ones by pressing **ACTIVE**

All changes in this panel are performed on a "scratch" definition, and changes only become permanent in the database when **UPDATE** is pressed.


CONTROL
ABORT
RESET_ALL
HELP
UPDATE
CHECK_CTRL
CONTROL cards (model 1)
Control Cat
ALL
DONE
Control Categories
STANDARD contr
ACCURACY
BULK_VISCOSIT
CHECK
CONTACT
CPU
DYNAMIC_RELA
ENERGY
HOURGLASS
OUTPUT
PARALLEL
RIGID
SHELL
SOLID
SOLUTION
STRUCTURED
SUBCYCLE
Model title:
<No model title given>
Memory size:
20000000
COPY_ALL..
Prefix:
<undefined>
Row\Col 1 2 3 4 5 6
CONTACT
SET...
COPY...
SLSFAC RWPNAL ISLCHK SHLTHK PENOPT THKCHG
USRSTR USRFRC NSBCS INTERM XPENE SSTHK
SFRIC DFRIC EDC VFC TH TH_SF
IGNORE FRCENG SKIPRWG OUTSEG SPOTSTP SPOTDEL
ENERGY
HGEN RWEN SLNTEN RYLEN

Selecting which control cards are displayed.



The scrolling menu on the left lists all control cards of current category (in this case STANDARD), using the following colour convention:

White on blue	Control card is present (active) in model
Black on Grey	Control card is not defined in the model (inactive)

To select a card (active or inactive) for display toggle it on/off using its row in this menu. Note: Selection makes the card active, whereas deselection just removes the card from the display panel. Thus deactivation of a card must be done explicitly.

You may also select:

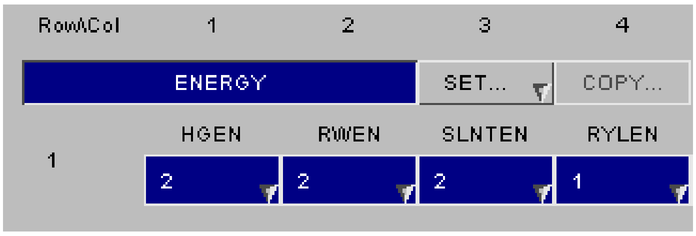
ALL All control cards, both active and inactive, are displayed

ACTIVE All active cards are displayed

DONE Return to card categories

Whenever a card is displayed and active its data fields can be updated by the normal text entry method. In addition all pre-defined lists of integers have popup menus giving the legal list of entries.

Making individual control cards active and inactive



Individual (visible) rows are toggled between active and inactive by clicking on their "name" button.

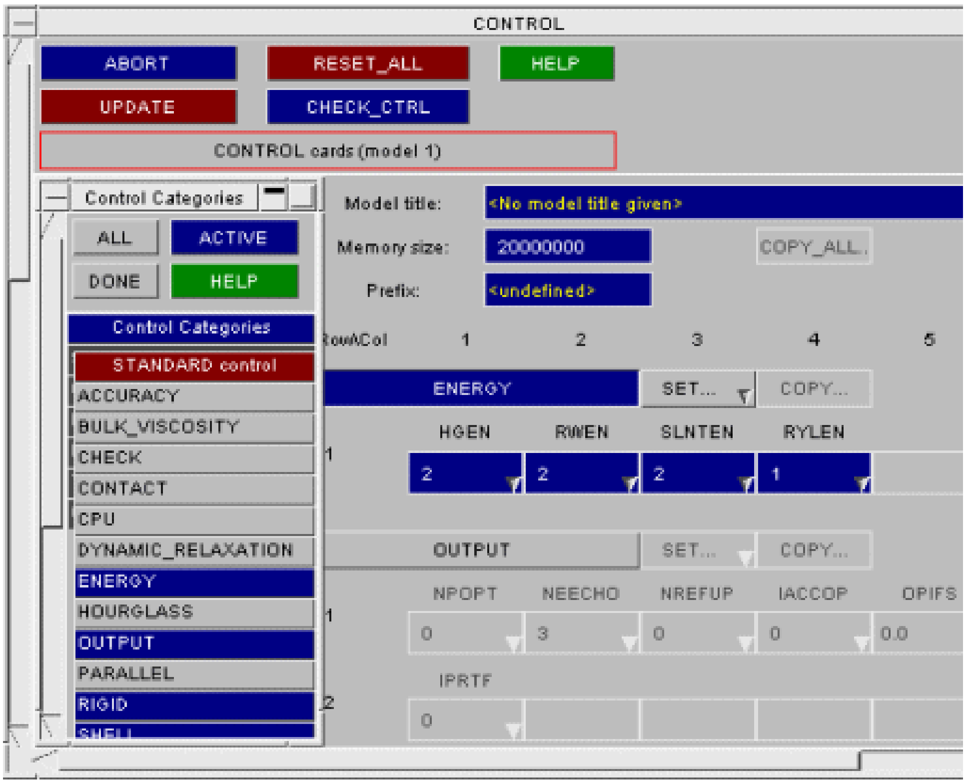
In this example:

ENERGY	Has been made active
OUTPUT	Has been made inactive

It can be seen that the (in)active status is also reflected in the selection menu on the left.

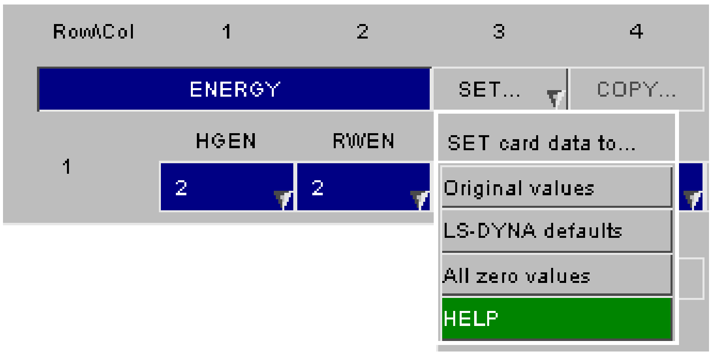
Note that inactive cards are "greyed out", and that entries cannot be made to them unless they are made active again.

Inactive cards will ***not*** be saved in the database following an **UPDATE** even if they contain (greyed out) values as here.



SET... (re)setting values for a control card.

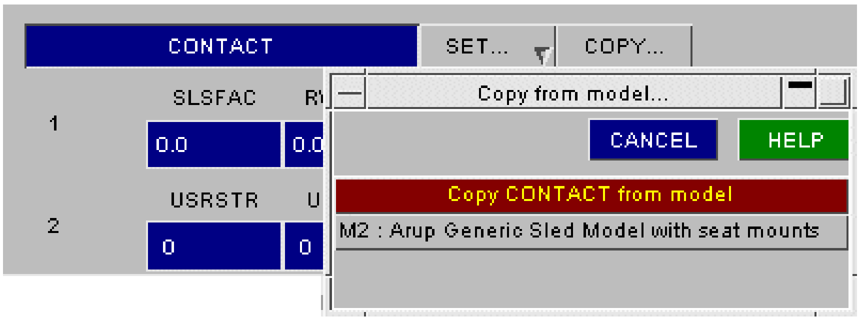
The data fields in a card may be (re)set back using the popup menu to:



Original values	The values in the current database, prior to any edits
LS-DYNA defaults	The default values quoted in the LS-DYNA keyword manual
All zero values	All fields are set to zero

Note that the local **SET...** option only affects this card. To reset ***all*** cards back to their original, unedited values use the **RESET_ALL** button at the top of the panel.

COPY... Copy data into a control card from another model

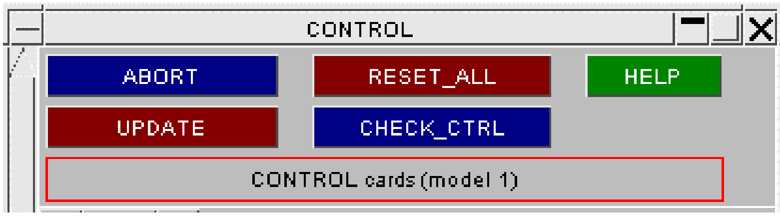


If one or more other models are present which also contain this control card then the **COPY...** button will be made live. This will give a list of possible models from which to copy this card's data. If no other models containing a definition for this card exist then the button will be greyed out.

Data copied in from another model supersedes the current data.

The **COPY...** option only affects this card. To copy data in from all cards in another model use the **COPY_ALL...** button at the top of the panel.

RESET_ALL: Restoring all control cards to their initial values.



RESET_ALL cancels the effect of all edit, copy, set and (in)activate operations by restoring all cards to their initial state as in the database.

UPDATE: Making control card edits permanent

All changes above are carried out on a "scratch" definition. Changes are only saved permanently in the database when **UPDATE** is pressed.

To exit leaving the control cards unchanged use **ABORT**.

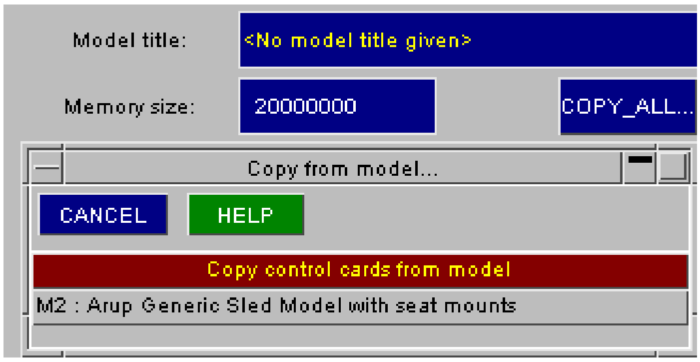
CHECK Running the standard checking function on control cards.

The **CHECK_CTRL** command runs the standard syntax and context checker. Most errors checking for control cards is based on detecting "out of range" parameters, but some interactions with data defined elsewhere in the model are also checked.

COPY_ALL: Copying all control cards from another model

COPY_ALL copies in all cards from another model, superseding any such definitions in this model.

Cards that are not active in the origin (copied from) model are not changed in the current model.



Setting model title and memory size.

In addition to editing the contents of the control cards:

- A **MODEL TITLE** of up to 80 characters can be specified within this panel.
- The **MEMORY SIZE** can also be set here. (This is optional, it is expressed in words of memory.)

CALC_DT2MS Calculates relationship of timestep to %age added mass

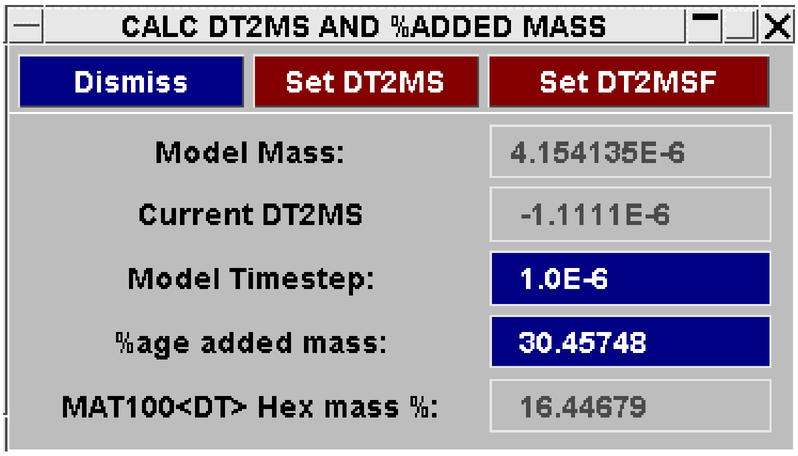
If DT2MS on the ***CONTROL_TIMESTEP** cards is set to less than zero, timestep added mass is active.

The **CALC DT2MS** function will report the %age added mass for the current timestep (expressed as DT2MS x TSSFAC).

A different value of timestep or %age added mass may be entered and the corresponding value will be calculated.

SET DT2MS and **SET DT2MSF**will update the values of DT2MS and DT2MSF on the timestep control card.

Note: TSSFAC is used for element timestep calculation and **should never exceed 0.9**. **CALC DT2MS** will never change TSSFAC.



LS-Dyna has a special method of adding mass to spotweld elements using the setting **DT** on the MAT100 card.

For beam spotwelds this added mass is included in the normal total and will appear in the **%age added mass** box.

For solid spotwelds, LS-Dyna calculates the added mass differently and reports it separately. This added mass will appear in the **MAT100<DT> Hex mass %** box (it is also included in **%age added mass** given).

See note in Appendix 17 for more details.

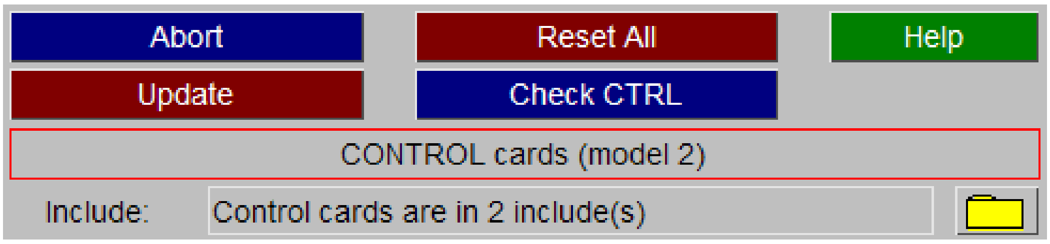
Notes on Control Cards and the model MERGE operation.

In common with other "static" (occurring either once or not at all) data in PRIMER, control cards may conflict when models are merged. The model merger allows you to select globally from source model #A or #B when cards exist in both models, but this may be too unselective for some cases.

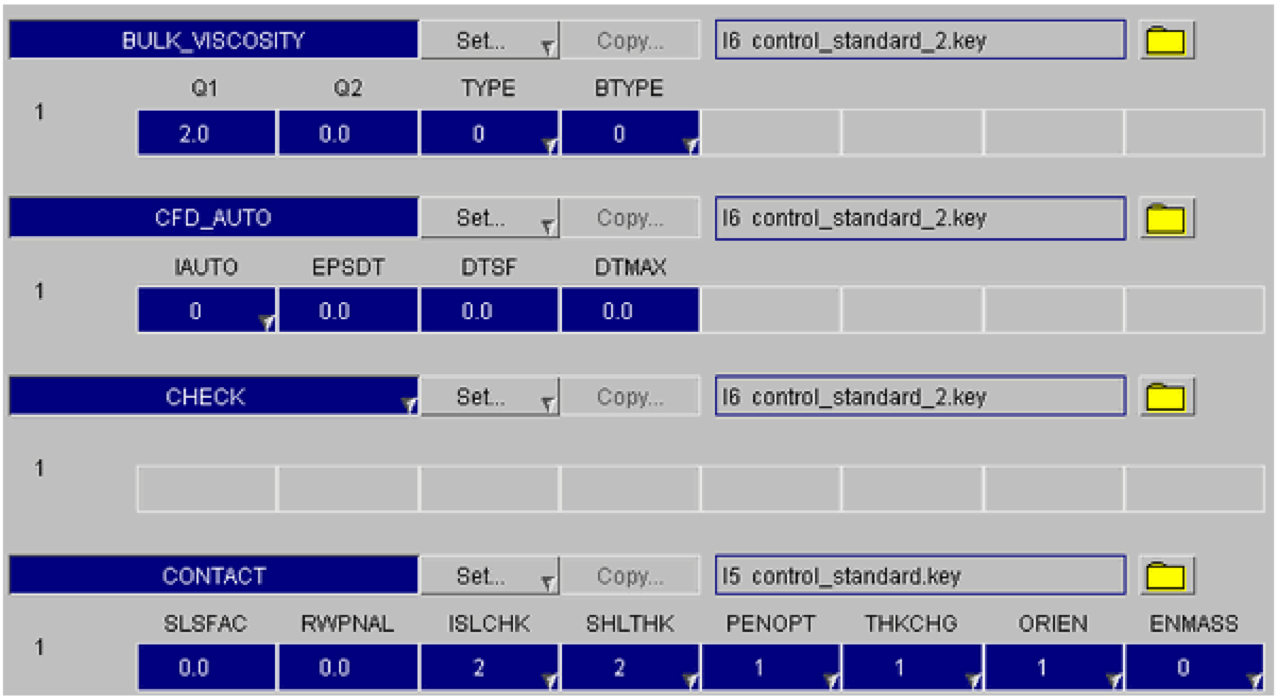
It is recommended that you review the control cards in the destination model generated by a merge operation, and make selective use of the **COPY** function above where required.

Notes on Control Cards and the include selection operation.

As in other editing panels, ***CONTROL** cards can be moved to a chosen include file using the include selection buttons at the top of the panel. All ***CONTROL** cards will be moved when this operation is carried out. When the ***CONTROL** cards are in more than one inlcude file, the include display within the editing panel will tell the user this is the case. Positioning the mouse over the include display area will print hover text to the screen listing all the include files the control cards are in.



The include file for each individual control card is displayed along side the control card information.



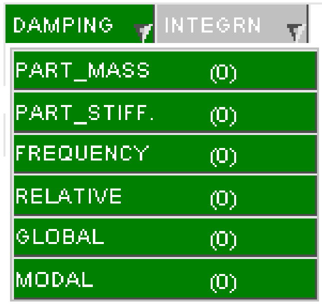
The include file can be modified by clicking on the folder button next to where the include name is displayed. Note the control card is only moved into the newly selected include file when **Update** is pressed on the top of the control card panel.

DAMPING: Defining Damping Cards.

LS-Dyna offers damping control via global damping, relative damping, frequency range damping, part damping by mass and part damping by stiffness. The CEAP version additionally has modal damping.

All six types may be created/edited in primer using the **DAMPING** function from the Keywords panel.

DAMPING MAIN MENU



This figure shows the main **DAMPING** menu.

The total number of damping cards for all models is reported.

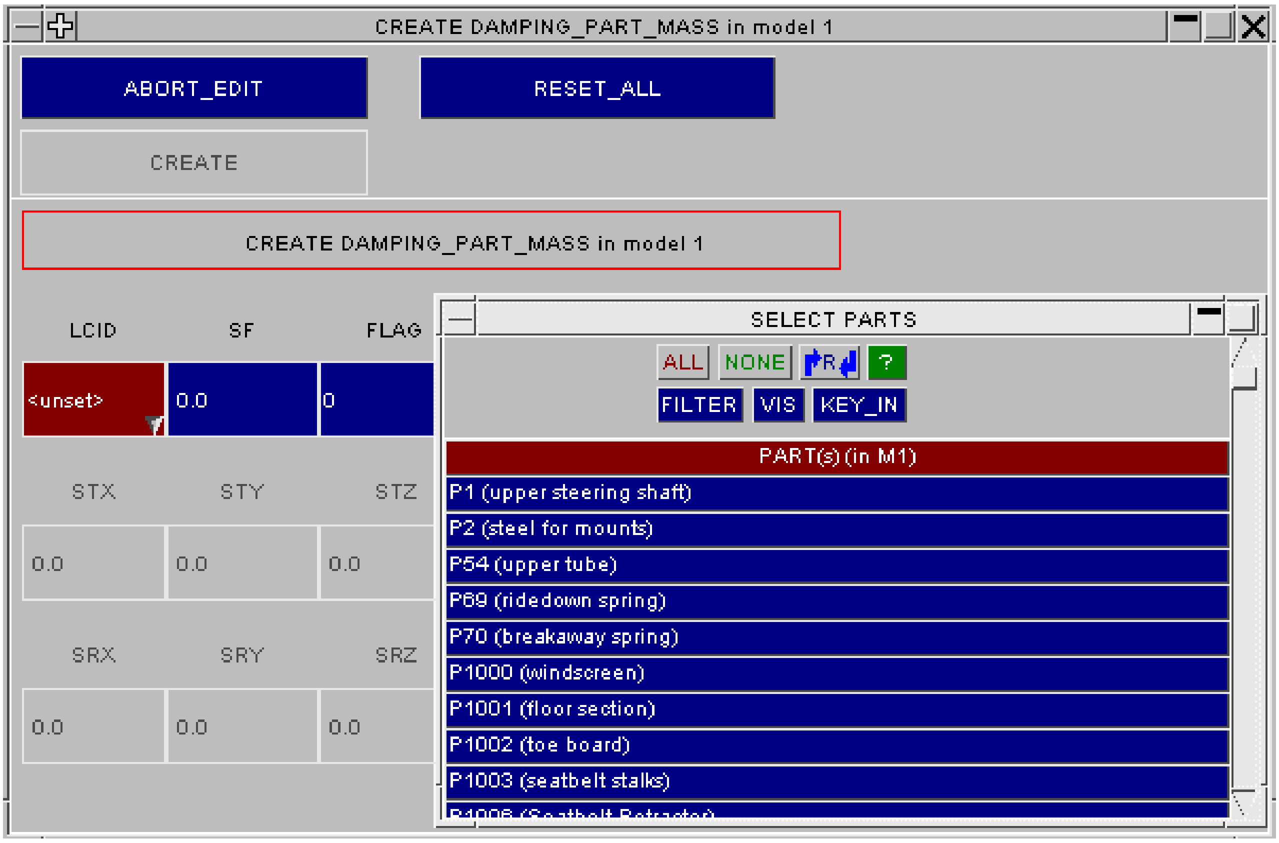
The panel will allow simultaneous editing of multiple damping cards. In the cases of Global and Modal damping there is no point in editing more than one card per model.

The static damping cards have dedicated editing panels. The others access the generic keyword editor.

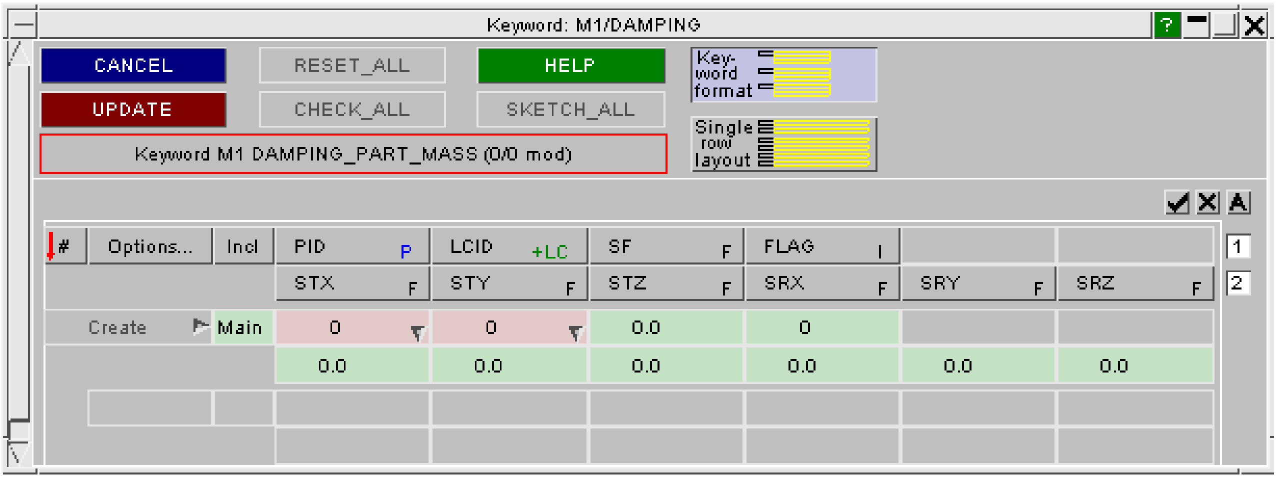
PART DAMPING

To create for a few parts, to delete and to edit, the keyword reader is perfectly sufficient. However, to enable the user to create rather larger numbers of damping cards, creation panels exist for damping part **MASS** and damping part **STIFFNESS** which allow parts to be selected from the object menu.

CREATING MULTIPLE DAMPING_PART_MASS



CREATING FROM KEYWORD READER (see section 5.0.3 for more details)



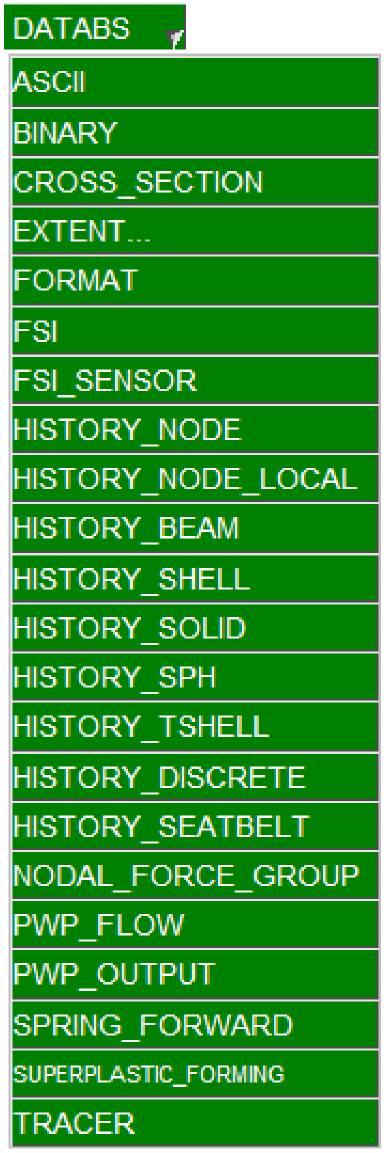
DATABASE: Defining Database Options.

- Selecting the *DATABASE sub-keyword
- Visualisation

The *DATABASE keyword in LS-DYNA is used to control what is written out to the BINARY and ASCII results files.
In addition to controlling what data is written out the *DATABASE cards also define how often the data is written out during the analysis and the format that is used to write the binary data files.

The main **DATABASE** pop-up menu allows any of the sub-categories to be selected. After a category has been selected a separate menu will be displayed allowing items to be created, modified and deleted.
The links below take you to the relevant sub-keyword sections:

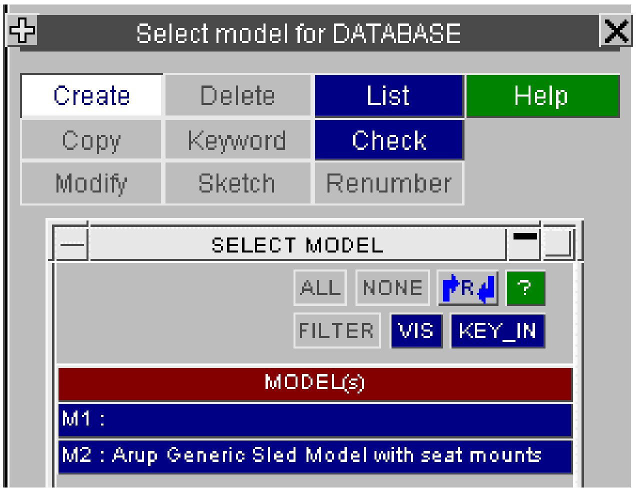
DATABASE_ASCII_xxx
DATABASE_BINARY_xxx
DATABASE_CROSS_SECTION_xxx
DATABASE_EXTENT_xxx
DATABASE_FORMAT
DATABASE_FSI
DATABASE_FSI_SENSOR
DATABASE_HISTORY_xxx **(this describes all sub-options).**
DATABASE_NODAL_FORCE_GROUP
DATABASE_SPRING_FORWARD
DATABASE_SUPERPLASTIC_FORMING
DATABASE_TRACER



If multiple models are open the main database panel will open and you will be asked to select a model. If only 1 model is open this stage will be automatically skipped and the selected sub-option window opened.

In addition the functions:

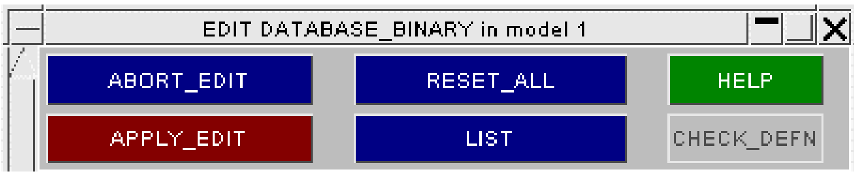
LIST	Generates a summary of all the ***DATABASE** cards in the model.
CHECK	Runs the standard check function on all ***DATABASE** definitions.



Common *DATABASE Options

Because some Keywords are "one-off" (e.g. **_BINARY**), some are "open-ended" (e.g. **_HISTORY_xxx**) and some reference normal "list" types with labels (eg **_CROSS_SECTION**) there is no standard editing panel for them.

However each of the sub-category menus contains a standard set of buttons:.



ABORT_EDIT	Returns to the main ***DATABASE** menu and discards any changes that have been made since the menu was invoked.
APPLY_EDIT	Returns to the main ***DATABASE** menu and keeps any modifications that have been made or any items that have been created. If in some of the sub-menus items have been marked for deletion this option will remove them from the model.
RESET_ALL	Resets all items back to the original values/settings they had when the menu was invoked. Any new items that have been created since the menu was invoked will be deleted.
LIST	Displays a list on the screen giving details about the status and values defined for all the items in the sub-category.
HELP	Displays a help screen giving information on what the sub-category is used for and what the different input parameters for each sub-category mean.

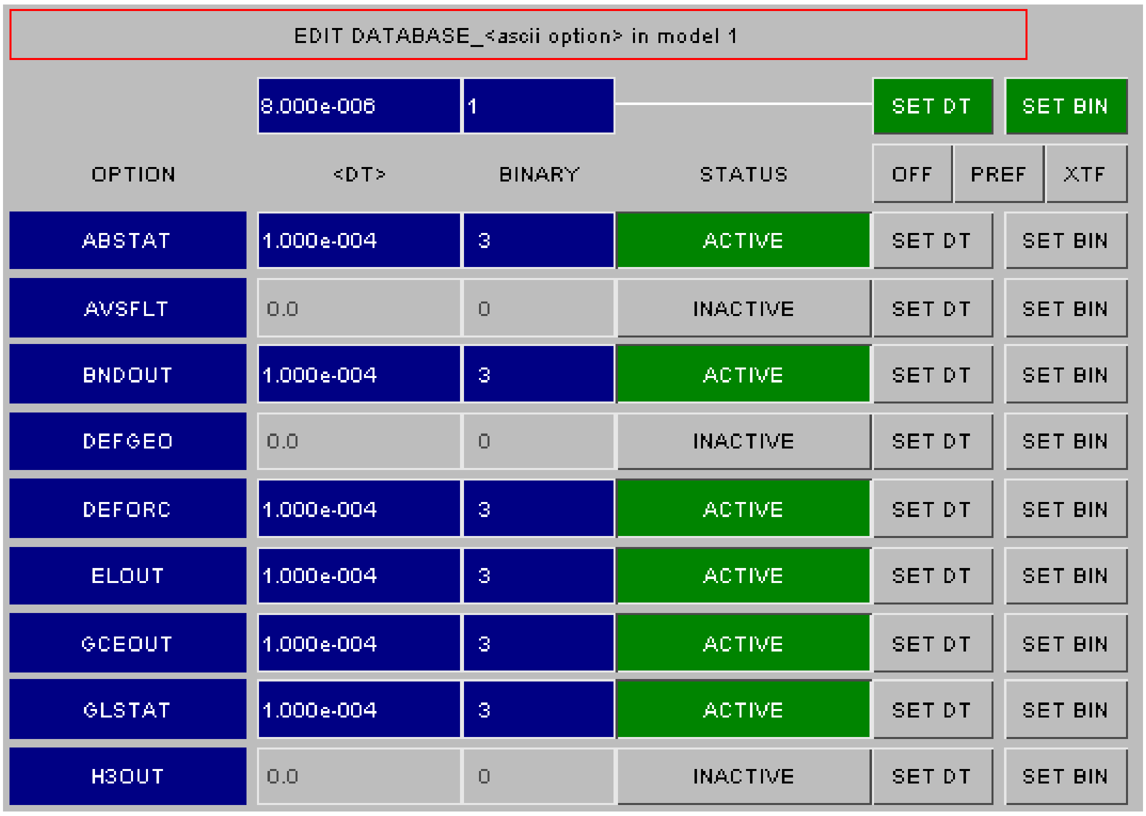
In addition to the standard set of buttons a number of the ***DATABASE** sub menus also contain the following two options.

ACTIVE / INACTIVE	Switch which can be used to include/exclude the DATABASE card in the model when it is written out. An INACTIVE card will have all parameters greyed out, whilst the ACTIVE card will allow parameters to be modified.
SET DEFAULTS	Reset all the variables within the card to the recommended LS-DYNA default values.

DATABASE_ASCII_xxx Control of "ASCII" database file output.

The **DATABASE_ASCII** menu displays a list of all the additional ASCII database files that may be created during an LS-DYNA analysis.

These files may now also be written in the more compact binary format.



DEFAULT TIME/ BINARY OPTION — Define a default output frequency and mode (1=ASCII, 2=BINARY, 3=BOTH), that can be applied to either individual ASCII cards (**SET DT** & **SET BIN**) or all the 'ASCII' cards (using green buttons at top).

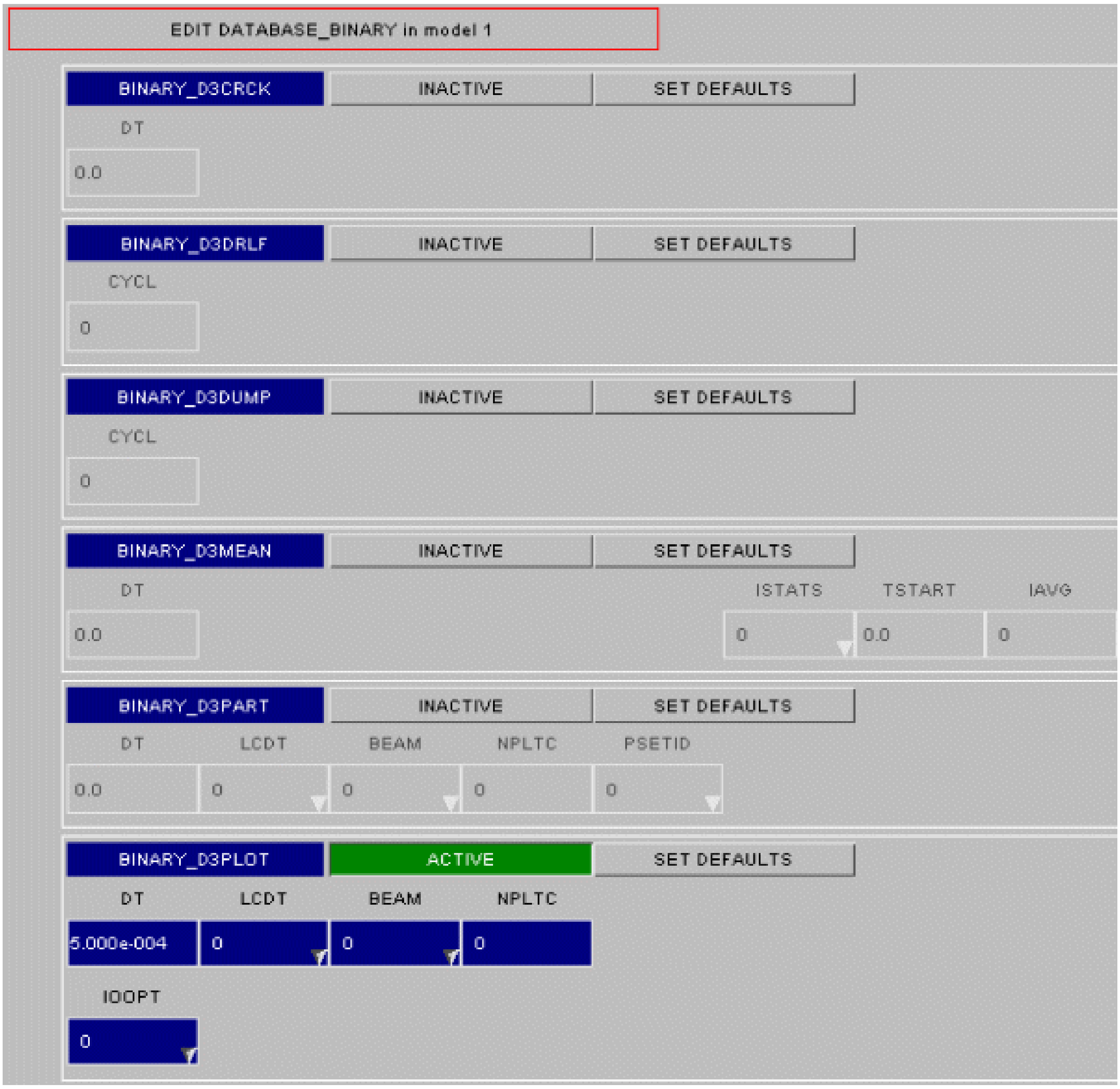
SET DT/SET BIN — Resets the output frequency/mode for 'ASCII' database file to that specified by the **Default Time/Binary option**. This function operates irrespective of an active or inactive status

ACTIVE / INACTIVE — Switch which can be used to include/exclude the `*DATABASE` card in the model when it is written out. An **INACTIVE** card will have all parameters greyed out, whilst the **ACTIVE** card will allow parameters to be modified

OFF/PREF/XTF — **OFF** makes all cards inactive.**PREF** makes those active which have been set in the oa_pref file. **XTF** activates those that contain information written to (soon to be obsolete) xtf file.

DATABASE_BINARY_xxx Control of binary database file output

The **DATABASE_BINARY_** menu displays a list of all the binary result and restart files that may be created during an LS-DYNA analysis



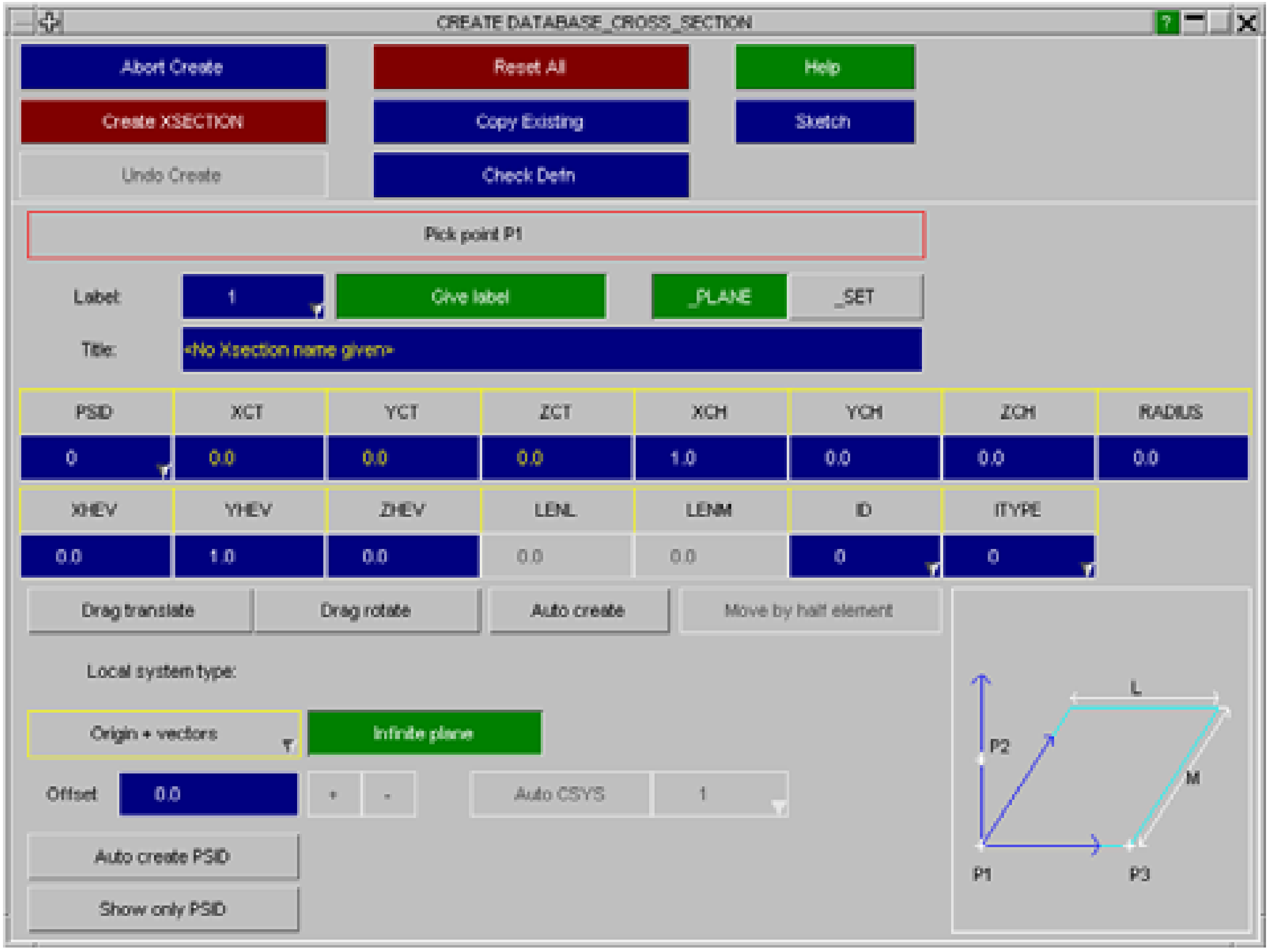
Output frequencies may be set for each database file, (note that some use time and some #cycles), and each may be made (in-)active. Some cards contain other fields which can also be set.

DATABASE_CROSS_SECTION_xxx Editing cross-sections for force extraction.

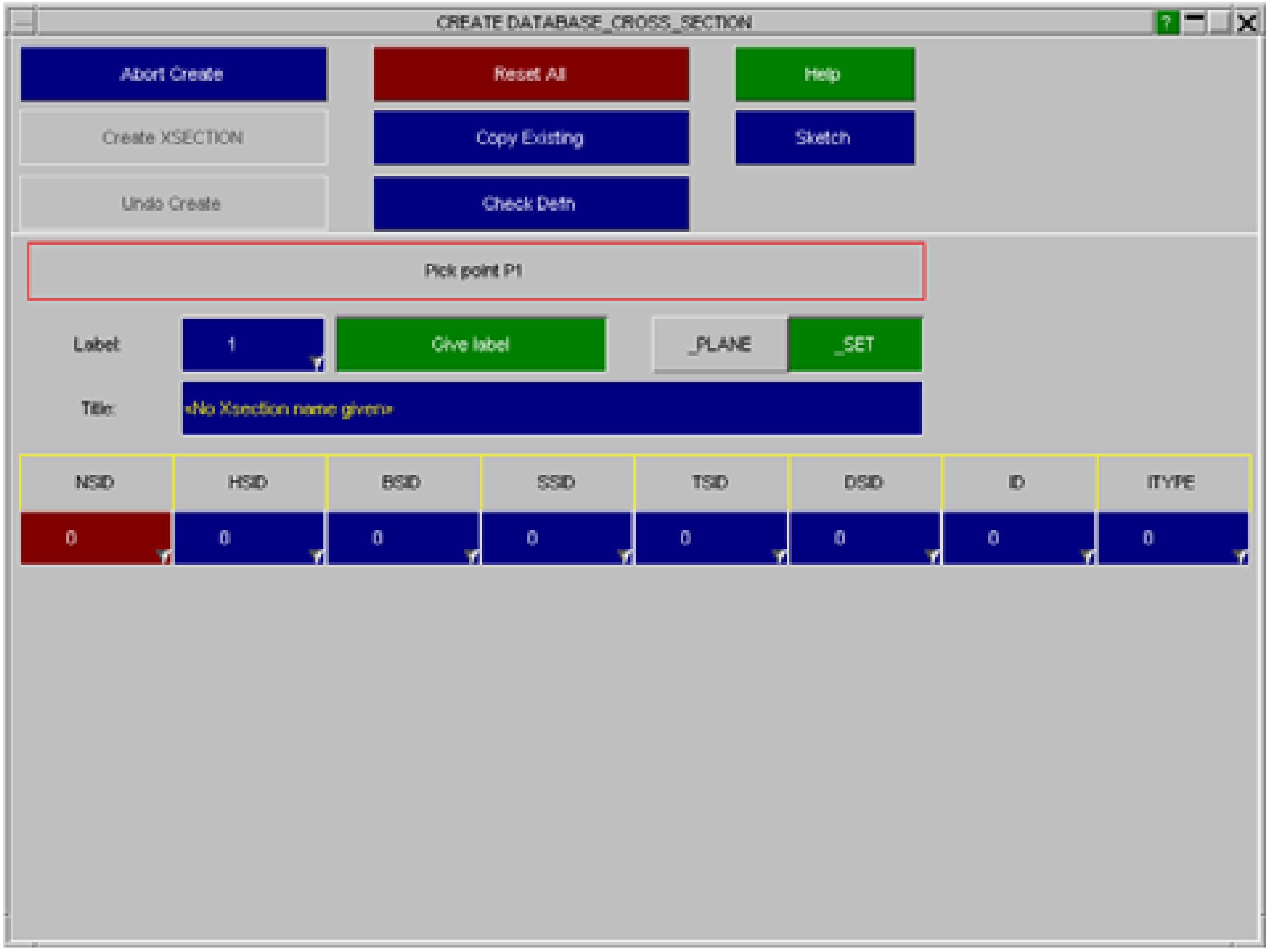
The ***DATABASE_CROSS_SECTION_PLANE** option defines a cross section for force output by defining a plane and then summing up the forces in the elements that the plane cuts through. (During the analysis the force is only summed for those elements that were cut by the plane at the start of the analysis).

The ***DATABASE_CROSS_SECTION_SET** option defines a cross section for force output by defining sets of nodes and elements for which the force should be summed.

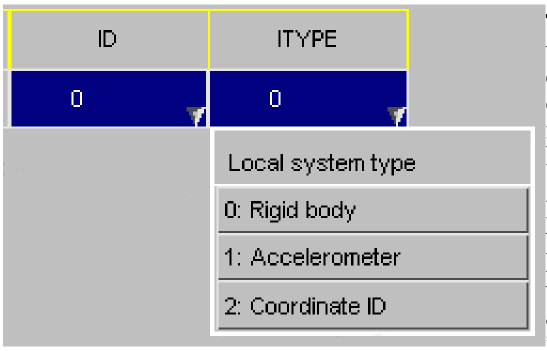
The figure below shows the **_PLANE** option.

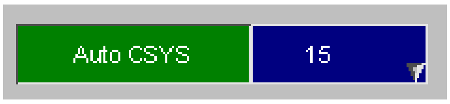


The figure below shows the **_SET** option.



The **Give label** button allows you to explicitly define a label for the ***DATABASE_CROSS_SECTION** card being created.



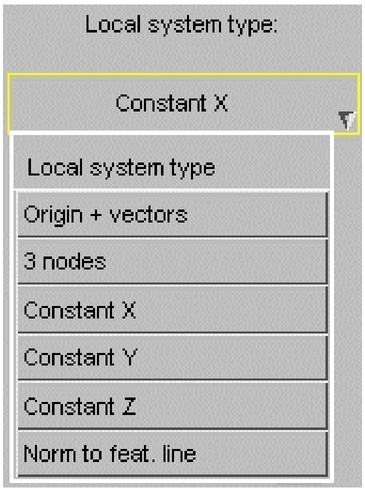


The <**itype**> data field which is common to both the **_PLANE** and the **_SET** options; controls whether the local system type <**id**> (if defined) is a rigid body or an accelerometer or an explicitly defined coordinate system. This is necessary because the type of the <**id**> field has to change, and therefore cannot remain "generic" within the keyword editor.

During the analysis, the user may want the extracted forces across the defined cross-section plane to be computed in the section plane's local coordinate system. This can be achieved quickly by means of the **Auto CSYS** button (not relevant in the _SET case) which is available when parameter <**itype**> is set to **2: Coordinate ID**. If a ***DATABASE_CROSS_SECTION_PLANE** card is created with this option switched on, a new ***DEFINE_COORDINATE_SYSTEM** card is automatically created using the **L-M-N** axes of the cross-section plane as its basis vectors. Parameter <**id**> of the created ***DATABASE_CROSS_SECTION_PLANE** card is then automatically set to be the label of this newly created coordinate system.
The user can define the label of the coordinate system to be created by entering it in the text box next to the **Auto CSYS** button.

Graphical definition of single **_PLANE** geometry. (Not relevant in the **_SET** case)

Primer allows you to quickly and easily create ***DATABASE_CROSS_SECTION_PLANE** cards in a number of ways. A particular method of plane creation can be chosen by invoking the **Local system type** pop-up shown below:

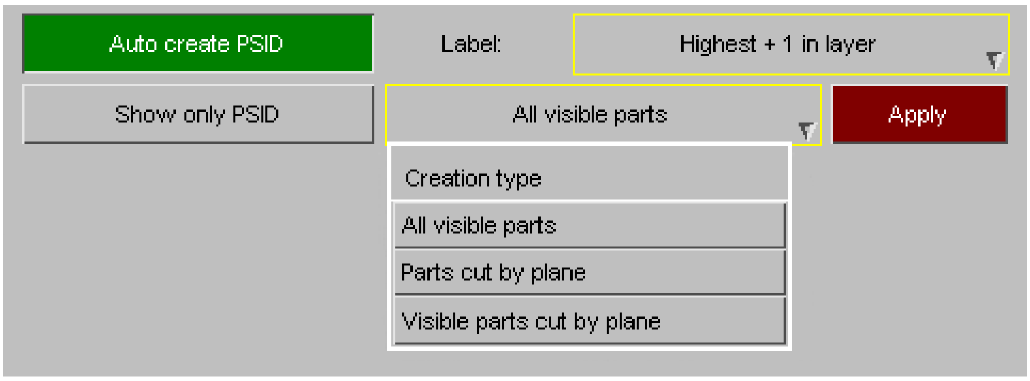


- **Origin + vectors**: Pick three nodes from the graphics window. They will define the origin, the unit normal vector **N** and the edge vector **L**, respectively, for the plane to be constructed.
- **3 nodes**: Pick three nodes from the graphics window. The first two nodes will define the origin and the edge vector **L**, respectively, for the plane to be constructed. The third lies on the **L**-**M** surface of the plane to be constructed, and hence defines it.
- **Constant X**: Pick one node from the graphics window. A cross-section plane passing through this node and normal to the global X-axis is computed immediately. The parameters <**lenl**> and <**lenm**> are computed automatically based on the dimensions of the model.
- **Constant Y**: Same as **Constant X**, except that the cross-section plane passes through the picked node, and is normal to the global Y-axis.
- **Constant Z**: Same as **Constant X**, except that the cross-section plane passes through the picked node, and is normal to the global Z-axis.
- **Norm. to feat. line**: Pick one node lying on a feature line or a free edge from the graphics window. A cross-section plane passing through this node and normal to the feature line is computed immediately.

The **Auto create** button (not relevant in the _SET case) automatically creates a new ***DATABASE_CROSS_SECTION_PLANE** card for every cross-section plane that is computed without the user having to explicitly click the **Create XSECTION** button. A cross section plane that is computed, but not created, is sketched in a suitable colour (depending on the background colour of your Primer's graphics window), while a cross-section plane that has been created and added to the model as a ***DATABASE_CROSS_SECTION_PLANE** card is always sketched in green.

The **Drag translate** and **Drag rotate** buttons (not relevant in the _SET case) allow the user to drag or rotate a computed cross-section plane in the graphics window about its local axes.

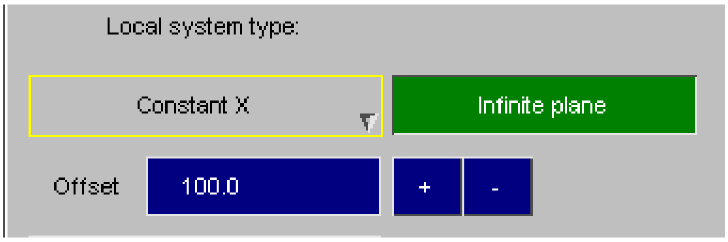
If a cross-section plane cuts along a mesh boundary, the results may be unreliable, as the forces from elements on both sides of the plane may be included. The **Move by half element** button (not relevant in the _SET case) attempts to remedy this problem by re-positioning the cross-section plane halfway along the edge of the element that lies on the positive side of the computed cross-section plane. If the feature line does not extend in the positive direction of the computed cross-section plane, the plane is automatically re-positioned halfway along the edge of the element that lies on the negative side of the cross-section plane.
This button is activated only when **Local system type** is set to **Norm. To feat. Line**



The **Auto create PSID** and related buttons (not relevant in the _SET case) shown on the left can be used to automatically create a new part set in the model and associate it with the ***DATABASE_CROSS_SECTION_PLANE** card being created (by automatically setting parameter <**psid**> on the ***DATABASE_CROSS_SECTION_PLANE** card to be the label of the newly created part set).

Once the part set is created by selecting an appropriate method from the pop-up box and clicking the **Apply** button, the parts contained in the set just created can be quickly visualized by clicking the **Show only PSID** button.

Graphical definition of multiple offset _**PLANE** geometry. (Not relevant in the _**SET** case)



Once a cross-section plane is computed using one of the methods described above, it can be used as a reference plane to quickly create another plane **OFFSET** units away from it. The offset plane is computed along the unit normal vector of the reference plane using the method selected in **Local system type**.
The buttons shown here are activated as soon as a reference plane is computed and a valid offset is specified in the **OFFSET** text box. Clicking the **+** button computes an offset plane along the positive direction of the reference plane's unit normal vector, while clicking the **-** button computes an offset plane along the negative direction of the reference plane's unit normal vector. The computed offset plane is then sketched using a suitable colour. The parameters of the computed plane are used to create the ***DATABASE_CROSS_SECTION_PLANE** card in the model by clicking the **Create XSECTION** button.
The **+** or **-** buttons can be clicked repeatedly to compute multiple planes each at a distance of **OFFSET** units away from the previously computed one.
If option **Auto create** is switched on, a new ***DATABASE_CROSS_SECTION_PLANE** card is automatically added to the model every time a new offset plane is computed without the user having to explicitly click the **Create XSECTION** button.

*DATABASE_CROSS_SECTION specific annotate/display tools.

There are specific tools for DATABASE_CROSS_SECTION's called **Annotate** and **Display**. **Annotate** is available through the main DATABASE_CROSS_SECTION panel or an individual DATABASE_CROSS_SECTION edit panel. This tool annotates the cross sections with the label associated with the cross section, and if you choose to annotate a number of cross sections, they are all sketched in different colours. **Display** is available through the main DATABASE_CROSS_SECTION panel. Display will show an individual cross section, and only the parts referenced by the cross section.

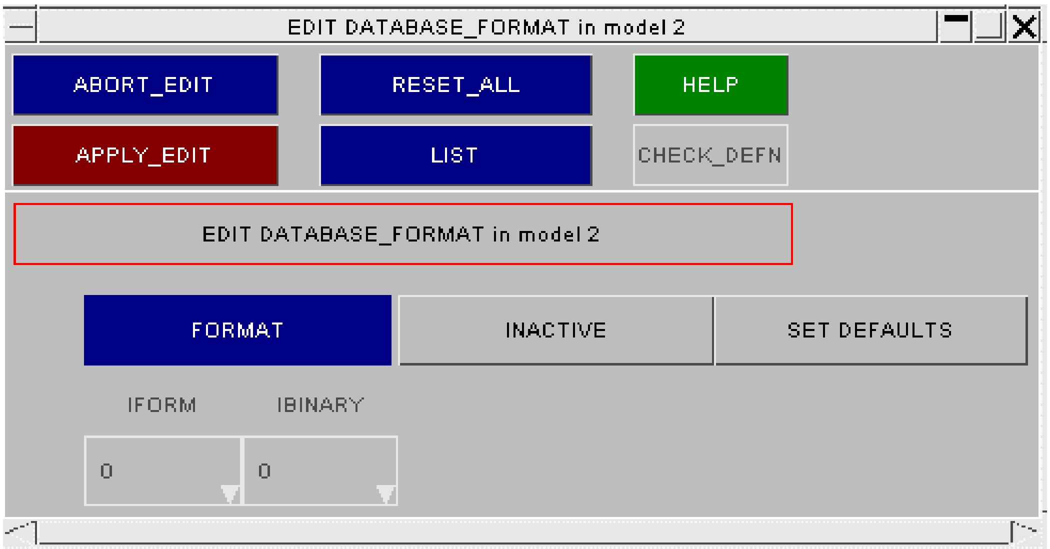
DATABASE_EXTENT_xxx

The ***DATABASE_EXTENT** menu has 2 sub options

***DATABASE_EXTENT_BINARY**	Controls a number of optional data components that can be written out to the binary results files.
***DATABASE_EXTENT_SSSTAT**	Defines a number of PART sets (see *SET) that are used to generate model sub system energies and contact forces.

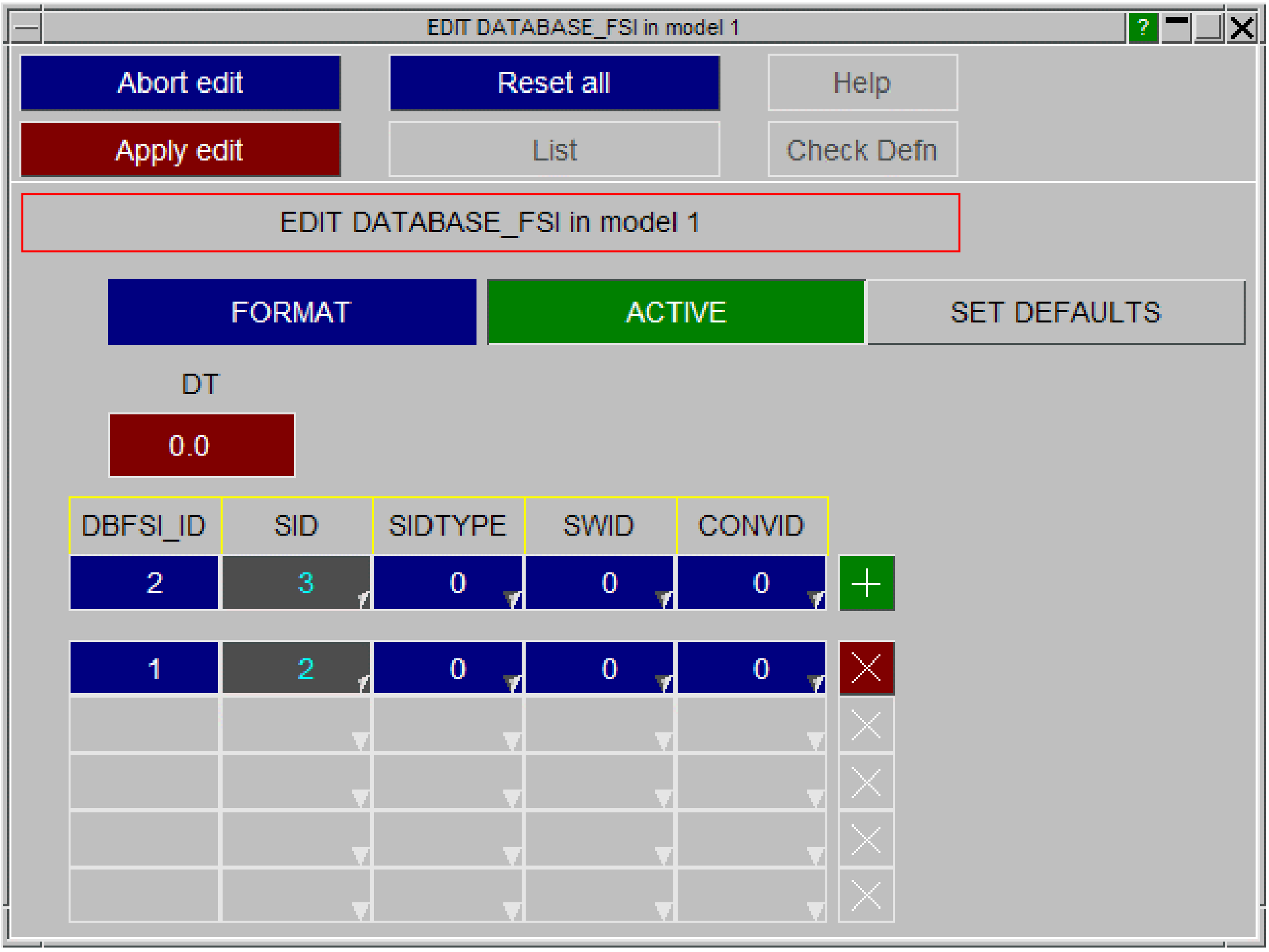
DATABASE_FORMAT Controlling binary database output format.

The *DATABASE_FORMAT menu controls the format the binary results files are written in.



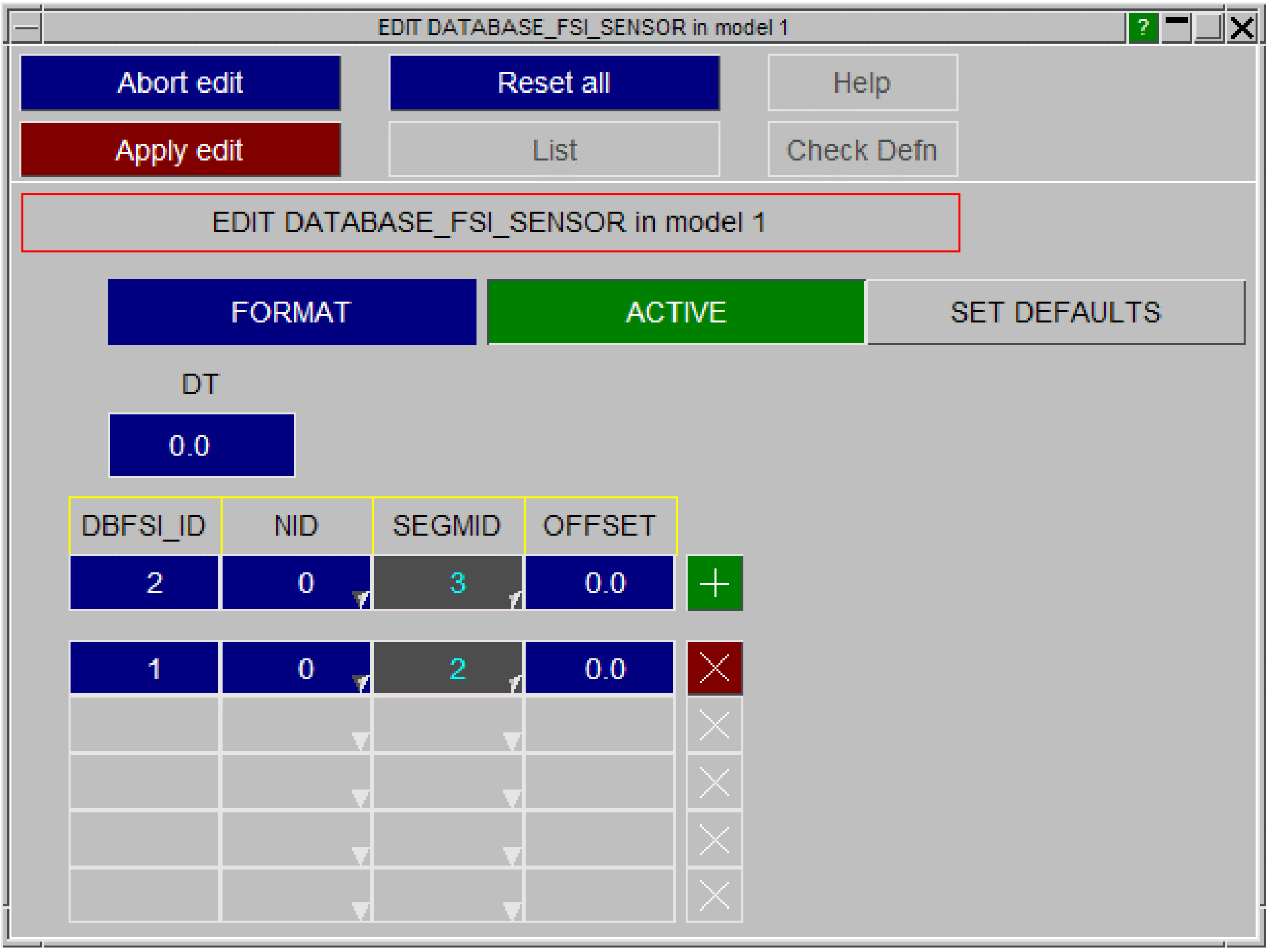
DATABASE_FSI Output information about certain Lagrangian surfaces.

The *DATABASE_FSI menu controls the output information about certain Lagrangian surfaces.



DATABASE_FSI_SENSOR Output of an ASCII file called "dbsensor".

The *DATABASE_FSI_SENSOR menu controls the output of an ASCII file called "dbsensor". The input defines the pressure sensors' locations which follow the positions of some Lagrangian segments during the simulation.

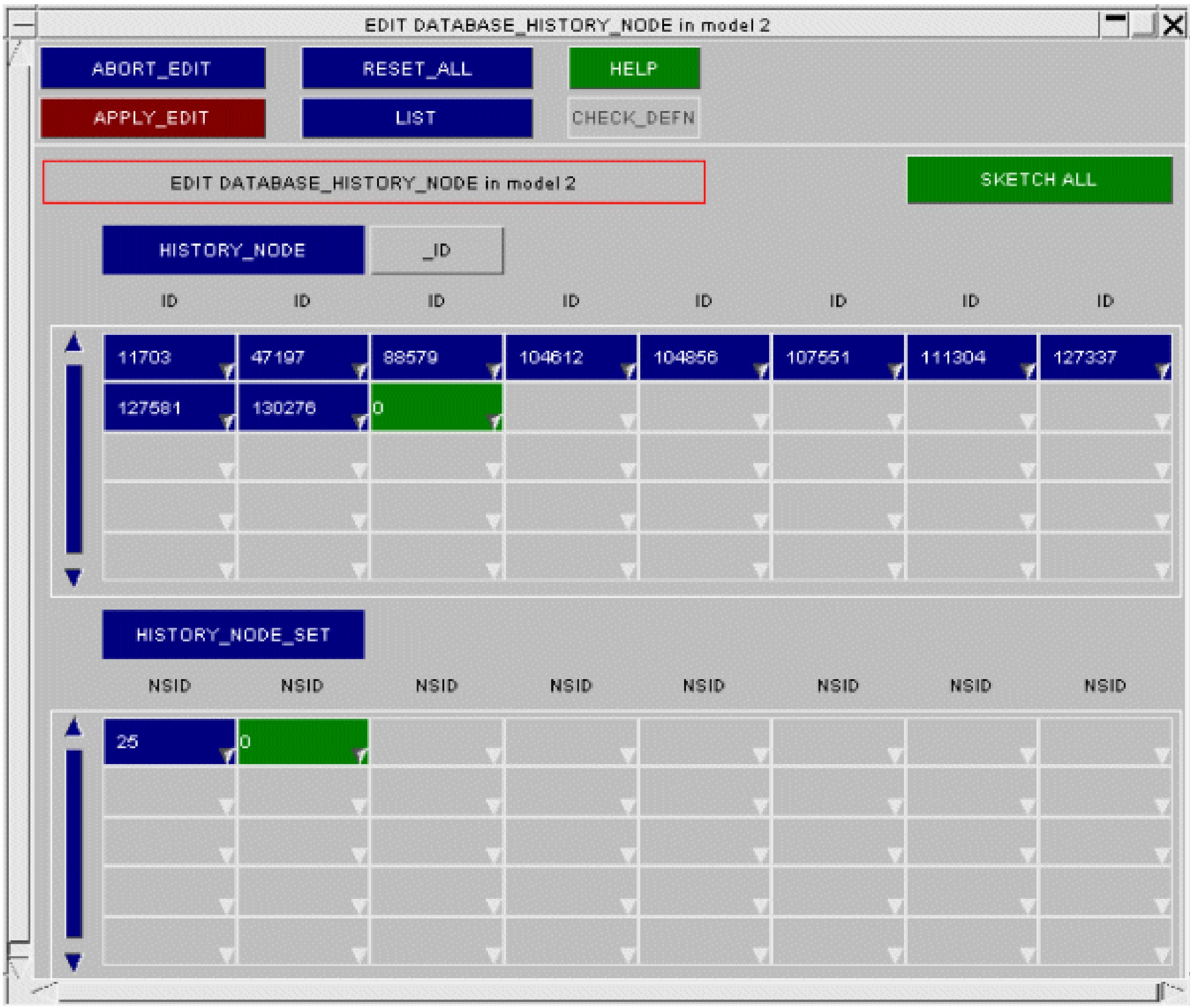


DATABASE_HISTORY_xxx Selecting nodes and elements for "time history block" output

***DATABASE_HISTORY** contains a number of sub-options to allow nodes, solids, beams, shells, SPH elements, thick shells, discretes and seatbelt elements to be defined for output to the time history (.thf) files.

For nodes only there is an extra suffix _LOCAL that permits a coordinate system to be associated with each node/set which controls the frame of reference for the output for that item. This uses a different panel, which is selected from the main ***DATABASE** pop-up.

Within each sub-option sets of entities may be selected/removed or individual items may be added and deleted.



For every ***DATABASE_HISTORY <type>** you can define items by:

Explicit **<item type>**	Here **_NODE**. A list of explicit items is defined, in any order.
SET_<item type>	Here **SET_NODE**. A list of node sets is defined.

_ID OPTION: By activating the _ID option, one may create database history items with a title.

NOTE: The SPH option in this context uses the following:

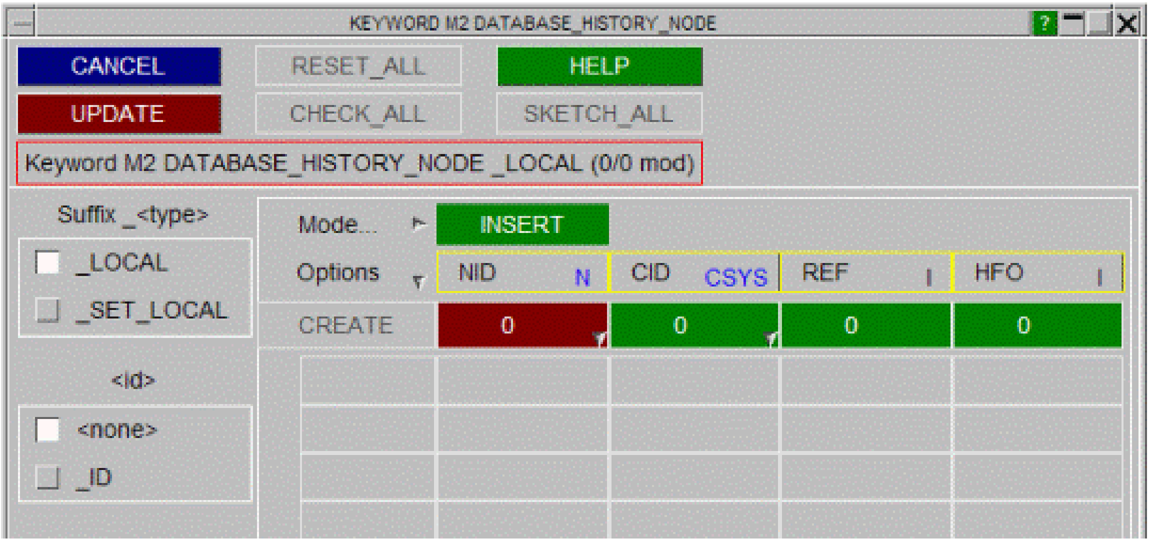
HISTORY_SPH	The "element" number of the SPH elements (which is the same as the node ids).
HISTORY_SPH_SET	The sets are in fact node sets (*SET_NODE) since there is no *SET_SPH keyword in LS-DYNA.

SKETCH_ALL: Sketches all of the entities that have been selected for time-history output or the currently selected entity type.

The special case of ***DATABASE_HISTORY_NODE(_SET)_*LOCAL*** uses the generic keyword editor instead of the layout above:

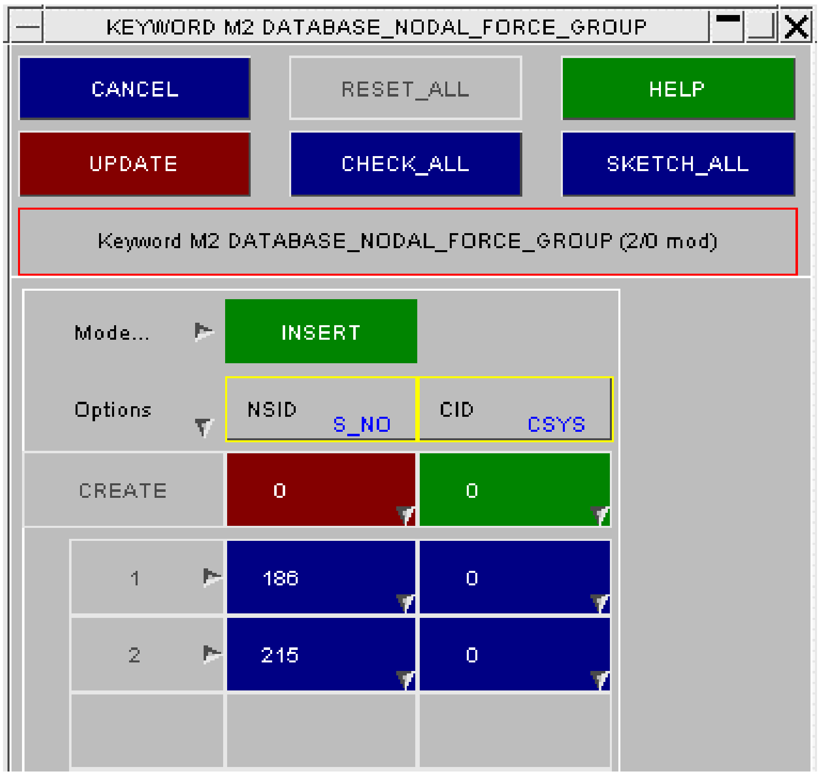
Toggle between **_NODE_LOCAL** and **_NODE_SET_LOCAL** using the radio box on the left of this panel.

Nodes should not appear more than once in a time-history block, however defined. The **CHECK** function will detect duplicate definitions and, when invoked from model checking, autofix them by removing them if requested.



DATABASE_NODAL_FORCE_GROUP

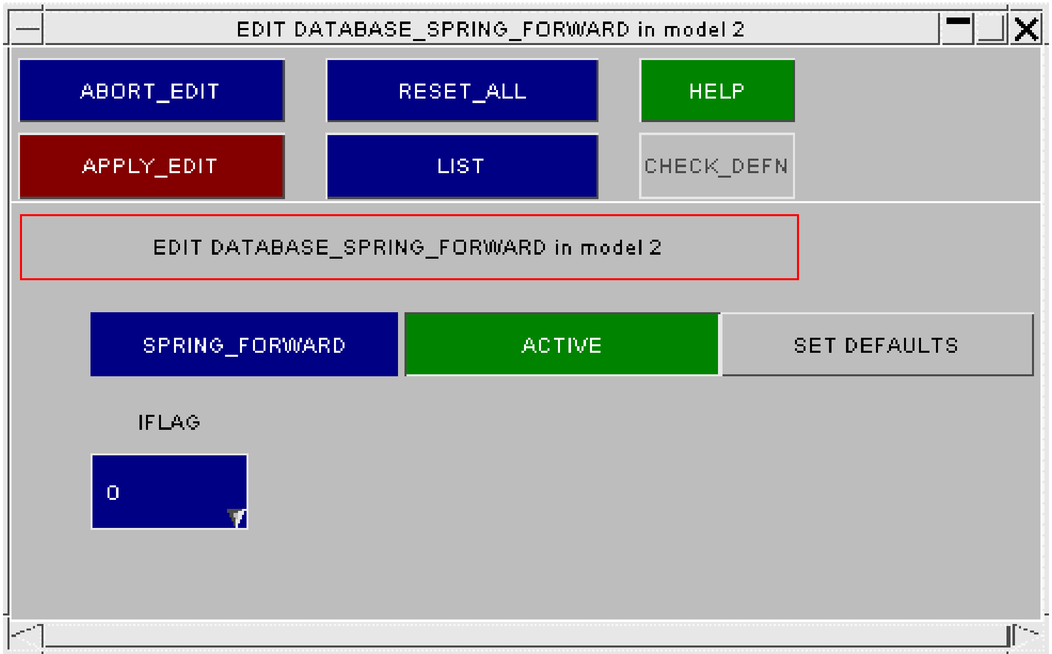
Definition of **SET_NODEs** that write summary force output.



DATABASE_SPRING_FORWARD

Output for implicit "spring forward" calculation after a metal-forming analysis

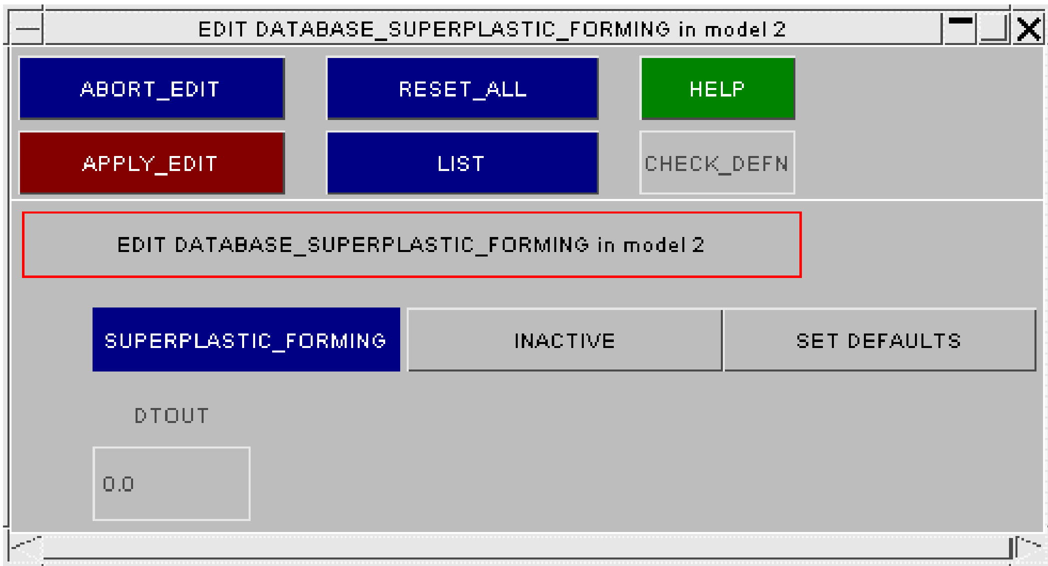
The ***DATABASE_SPRING_FORWARD** menu allows the nodal force components at the end of a metal stamping simulation to be output to an ASCII file for input to a subsequent spring forward calculation.



DATABASE_SUPERPLASTIC_FORMING

Output frequency to Super-plastic forming database files.

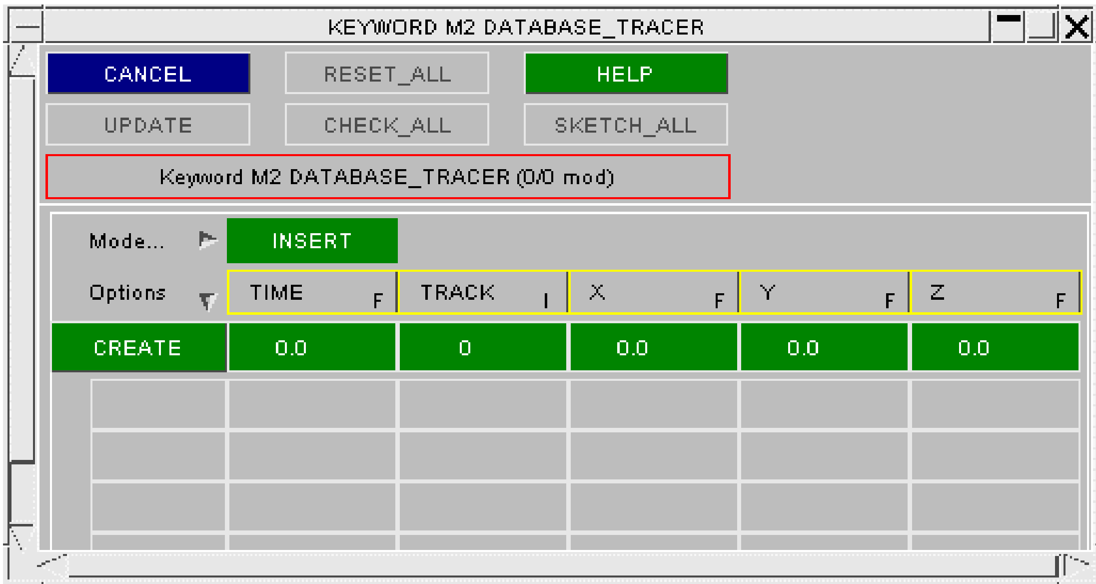
The ***DATABASE_SUPERPLASTIC_FORMING** menu allows the output frequency to be specified for output to the Superplastic Forming files.



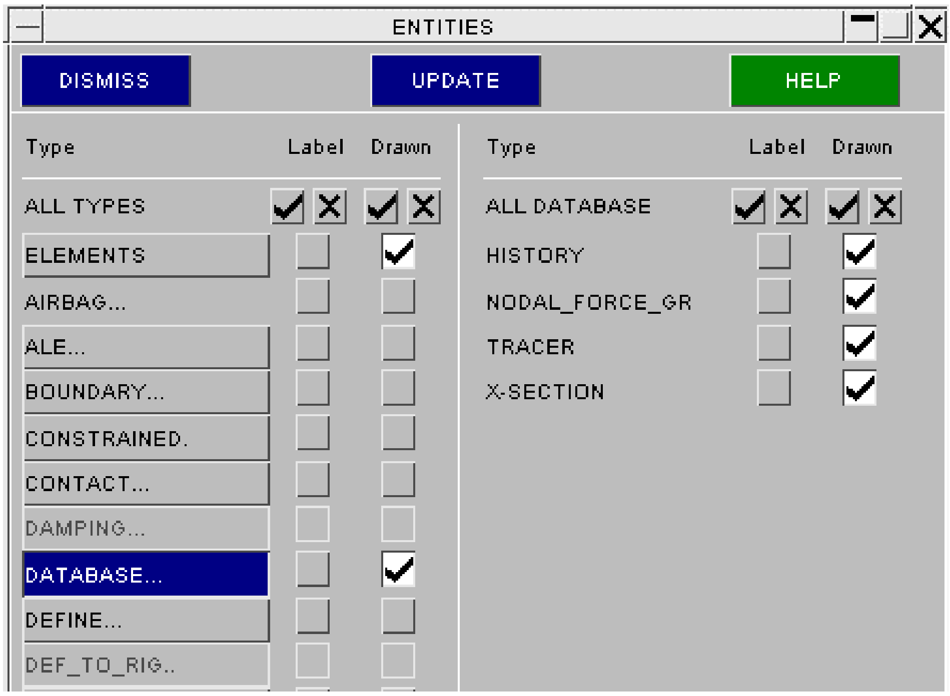
DATABASE_TRACER

Specification of "tracer" particles for Eulerian analyses

The ***DATABASE_TRACER** menu allows a series of initial points to be defined for which the position, velocity and stress components can be written out to an ASCII file.



Visualisation of *DATABASE items.



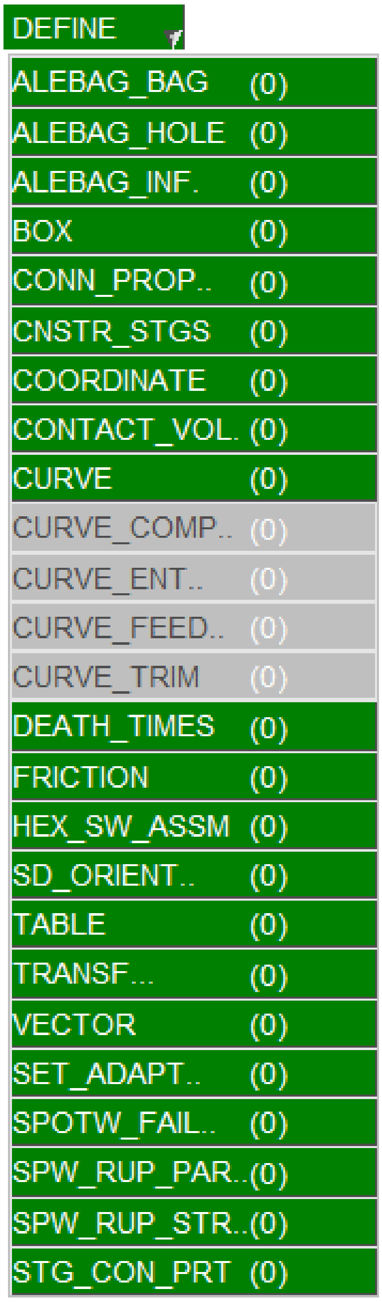
Those ***DATABASE_<type>** options which are sensibly drawable may be displayed via the **ENT**ity Viewing panel.
The **SKETCH** buttons in the relevant editing panels will also draw them.

DEFINE: Defining Define Options.

The ***DEFINE** keyword in LS-DYNA is used to define various things including boxes, load curves, tables, coordinate systems and orientation vectors.

The main **DEFINE** pop-up menu allows any of the sub-categories to be selected. After a category has been selected a separate menu will be displayed allowing items to be created, modified and deleted.
Some **DEFINE** keywords are edited in Primer using the generic Keyword editor. These are:

- **DEFINE_ALEBAG_BAG**
- **DEFINE_ALEBAG_HOLE**
- **DEFINE_CONSTRUCTION_STAGES**
- **DEFINE_SET_ADAPTIVE**
- **DEFINE_SPOTWELD_RUPTURE_PARAMETER**
- **DEFINE_STAGED_CONSTRUCTION_PART**



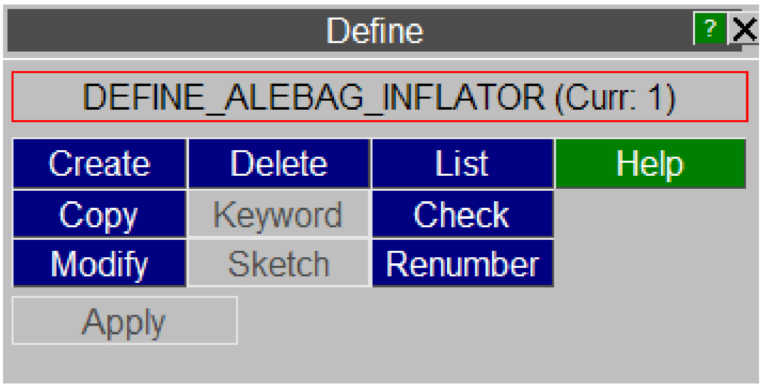
The other **DEFINE** keywords are edited in Primer using specific editing panels. The links below take you to the relevant sub-keyword sections:

- **DEFINE_ALEBAG_INFLATOR**
- **DEFINE_BOX**
- **DEFINE_CONNECTION_PROPERTIES**
- **DEFINE_CONTACT_VOLUME**
- **DEFINE_COORDINATE_xxx**
- **DEFINE_CURVE_xxx**
- **DEFINE_DEATH_TIMES**
- **DEFINE_FRICTION**
- **DEFINE_HEX_SPOTWELD_ASSEMBLY**
- **DEFINE_SD_ORIENTATION**
- **DEFINE_SPOTWELD_FAILURE_RESULTANTS**
- **DEFINE_SPOTWELD_RUPTURE_STRESS**
- **DEFINE_TABLE**
- **DEFINE_TRANSFORMATION**
- **DEFINE_VECTOR**

(DEFINE_) ALEBAG_INFLATOR:

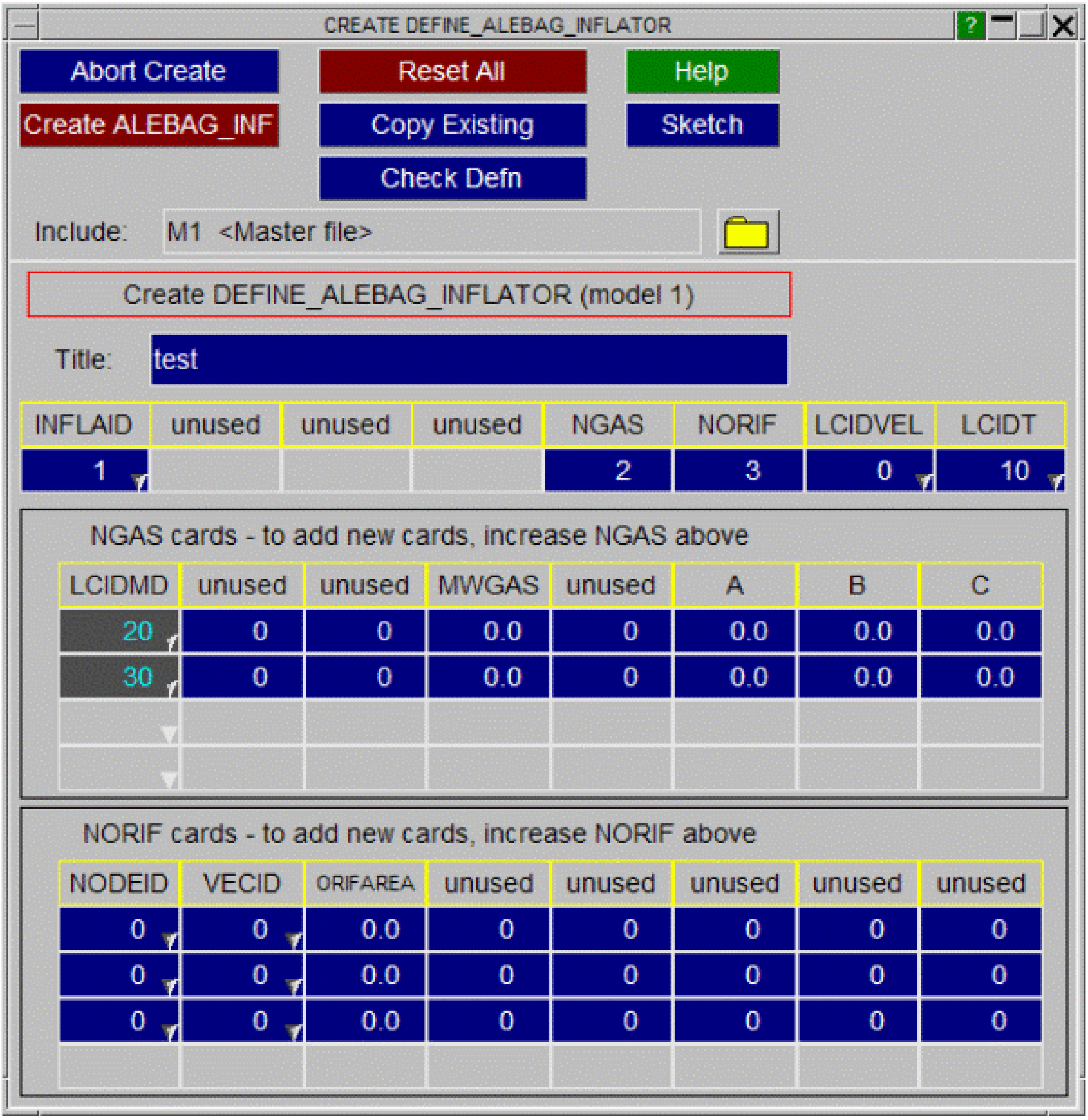
- These can be edited through their own specific editing panel (see below).
- Main Menu
- Creation
- Copying
- Editing
- Deletion
- List
- Check
- Renumber

This figure shows the main menu for the editing of alebag inflator definitions.
All functions have their standard meanings as given in section 5.0.1



CREATE Making a new alebag inflator definition.

This shows the create/edit panel for alebag inflators. New 'NGAS' or 'NORIF' rows can be added to this card by typing the required value into the **NGAS** or **NORIF** fields on the first line.



COPY Copy existing alebag inflator(s) to make a new alebag inflator(s).

The selected alebag inflators are copied. (alebag inflators do not "own" anything, so the concept of recursive copying does not apply.)

MODIFY Modifying the attributes of an existing alebag inflator.

MODIFY functions in the same way as **CREATE**, except that an initial definition will be present. Any modifications made to the alebag inflator definition will not be made permanent until the **UPDATE_ALEBAG_INF** button is pressed. At this point the local copy which has been updated is used to overwrite the version in the model.

DELETE Delete existing alebag inflator definitions.

The selected alebag inflators are deleted.

Alebag inflators do not "own" anything, so the concept of recursive deletion does not apply, however an alebag inflator that is referred to (ie "owned") by some higher order item will not be deletable unless that item is deleted too, or its reference to the alebag inflator removed.

LIST List alebag inflator summaries to screen

The selected alebag inflators are summarised on the screen.

CHECK Check alebag inflator definitions for errors

The selected alebag inflator definitions are run through the standard checking routines.

RENUMBER Change alebag inflator labels

RENUMBER lets you change any or all alebag inflator labels within a given model using the standard renumbering panel. To change the label of an individual alebag inflator it may be simpler just to **MODIFY** it.

(DEFINE_) BOX: Defining Boxes

- Main Menu
- Creation
- Copying
- Editing
- Deletion
- Visualisation

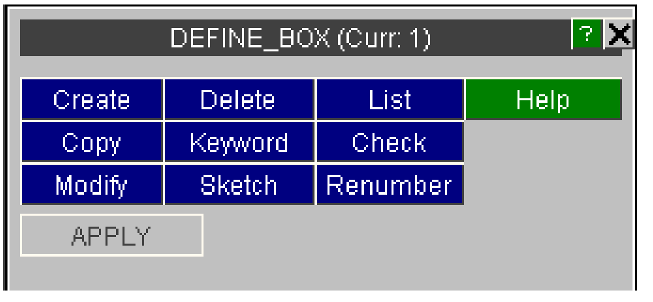
Box suffices:

_ADAPTIVE
_COARSEN
_DRAWBEAD
_SPH

The ***DEFINE_BOX** keyword is used to create "boxes". These are rectilinear volumes of space, defined by the min and max [x,y,z] coordinates, in global space units, of diagonally opposite corners. This imposes limitations upon their orientation when rotated - see below.
Boxes are used to bound the limits of other items, for example a contact may be defined by those elements which lie within a box. However they are not structural items and play no direct role during the actual analysis.

Boxes use unique labels and, although part of the ***DEFINE** keyword, their labels do not clash with other ***DEFINE_xxx** entities. For example it is legal to have **(*DEFINE_)BOX #1** and **(*DEFINE_)CURVE #1.**

This figure shows the main menu for the editing of box definitions.
All functions have their standard meanings as given in section 5.0.1



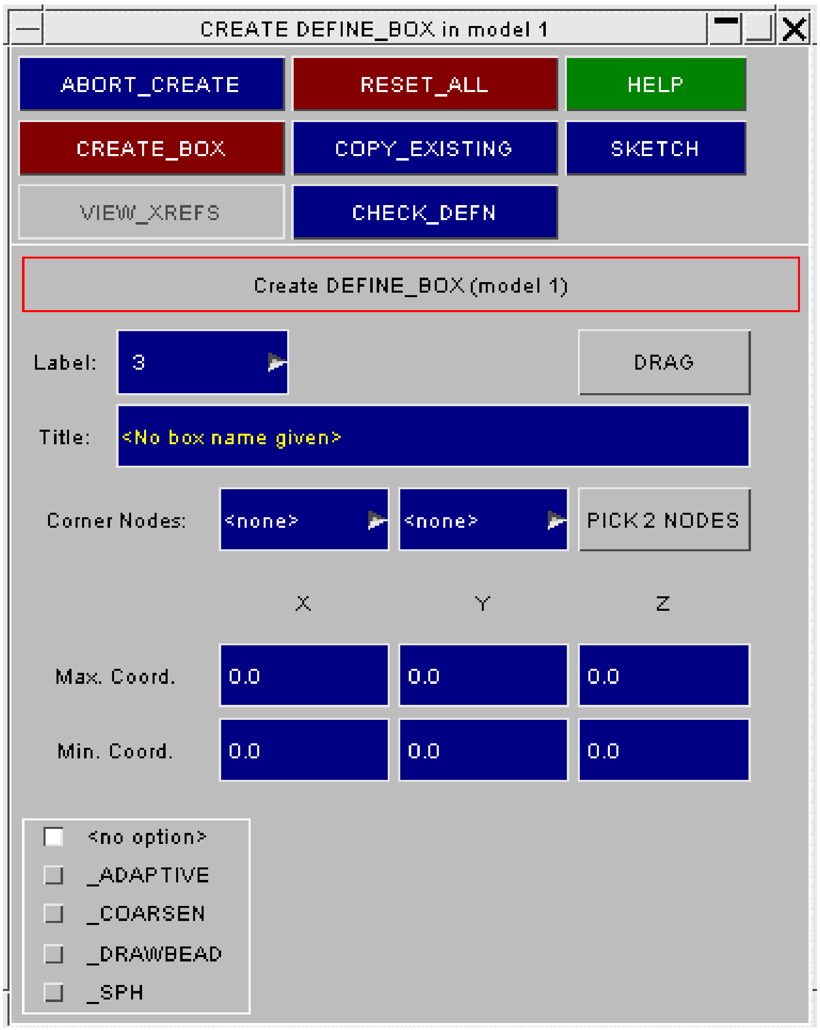
:

CREATE Making a new box definition.

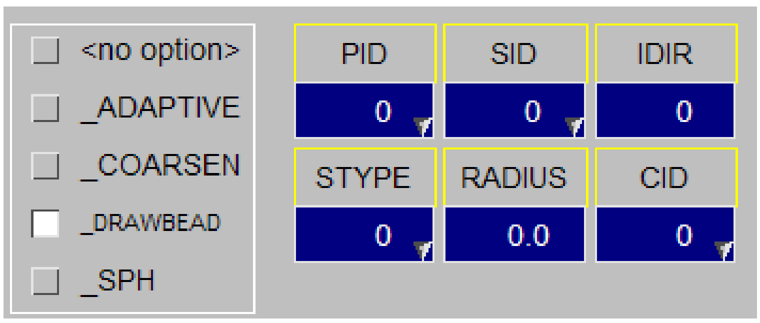
This figure shows the basic **CREATE** / **UPDATE BOX** panel.
Methods of defining box coordinates:

Corner Nodes:	The box can be defined in terms of corner nodes in the current target model. In this example the box does not use corner nodes: **<none>** is displayed in the button field. A value for each node can either be typed in directly or chosen via the associated popup window (i.e. screen picking).
PICK 2 NODES	Instead of defining each node separately, both nodes can be screen-picked together.
MIN / MAX COORDINATES	The minimum and maximum X, Y and Z coordinates are displayed at the bottom of the basic editing panel. These numbers can be typed in directly if required.

Note that the coordinates of a box are independent of the methods used to define them. For example using nodes, by either method above, only extracts the coordinates of the nodes, and they do not become part of the box definition.



*DEFINE_BOX options: _ADAPTIVE, _COARSEN, _DRAWBEAD, _SPH

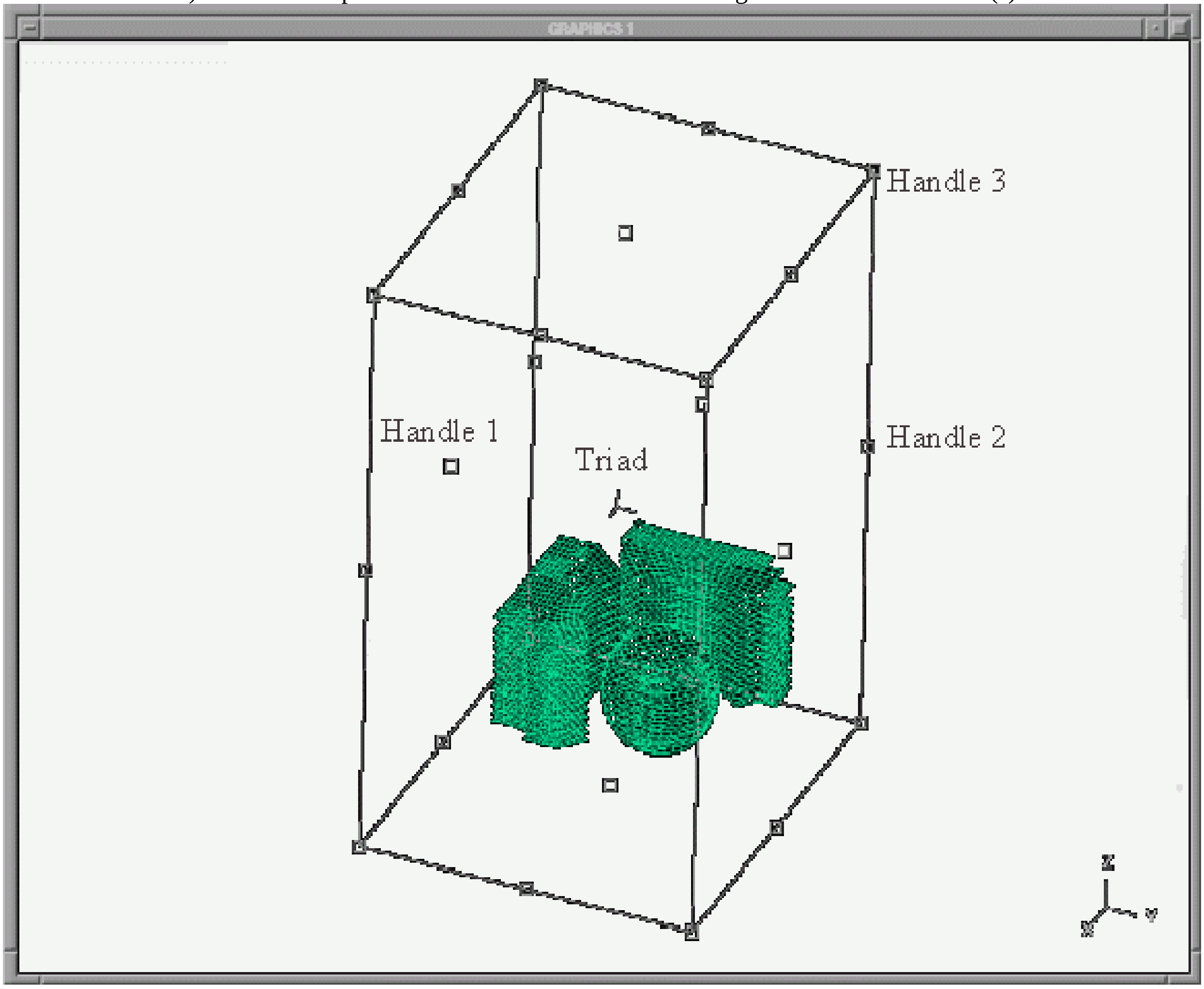


The various sub-types of box may be selected in the editor above.

The data entry rows will change accordingly. This example shows the _DRAWBEAD data.

DRAG "Dragging" a box size and shape interactively with the cursor.

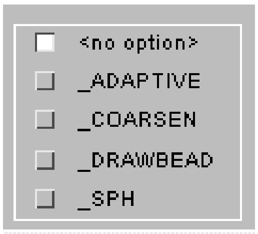
Once a box has been given some initial dimensions the cursor can be used to modify the dimensions and position of the box.
When **DRAG** is selected the current box definition is sketched, and 27 "handles" are added to it (8 corners, 12 edges, 6 faces and 1 centre). Each can be picked with the cursor and used to drag the relevant dimension(s).



The dragging "handles" are:

TRIAD (1 @ centre)	LEFT mouse button operates the X-coordinate position of the box; translating the whole box along the X-axis. Note that the X-coordinate buttons are turned green while the mouse button is depressed. MIDDLE mouse button translates the box through the Y-axis; Y-coordinate buttons are turned green. RIGHT mouse button translates the box through the Z-axis; Z-coordinate buttons are turned green.
HANDLE_1 (6: 1 @ each box face)	Any mouse button will allow the FACE of the box to translated along a vector normal to the face. Note that the coordinate buttons that are locked out turn red.
HANDLE_2 (12: 1 @ each box edge)	Any mouse button will allow an edge of the box to translated in the plane normal to the edge line. The coordinate box parallel to the edge will be locked out and turned red.
HANDLE_3 (8: 1 @ each box vertex)	Any mouse button will allow the corner of a box to be moved in any of the three axis directions

BOX_<options>
The radio buttons allow the selection the options **_ADAPTIVE**, **_COARSEN**, **_DRAWBEAD** and **_SPH**.



COPY Copy existing box(es) to make a new box(es).

The selected boxes are copied. (Boxes do not "own" anything, so the concept of recursive copying does not apply.)

MODIFY Modifying the attributes of an existing box.

MODIFY functions in the same way as **CREATE**, except that an initial definition will be present.

Any modifications made to the box definition will not be made permanent until the **UPDATE_BOX** button is pressed. At this point a the local copy which has been updated is used to overwrite the version in the model.

DELETE Delete existing box definitions.

The selected boxes are deleted.

Boxes do not "own" anything, so the concept of recursive deletion does not apply, however a box that is referred to (ie "owned") by some higher order item will not be deleteable unless that item is deleted too, or its reference to the box removed.

KEYWORD Creation / editing in the generic keyword editor

Boxes may be created, edited and deleted as a whole category in the generic keyword editor.

SKETCH Sketch box definitions on the current image.

SKETCH allows the user to select box definitions and superimpose a white sketch of them over the currently displayed image. The box and its contents (in the context in which it is being used) will be sketched if the **WITH CONTENTS** button is active.

LIST List box summaries to screen

The selected boxes are summarised on the screen.

CHECK Check box definitions for errors

The selected box definitions are run through the standard checking routines.

RENUMBER Change box labels

RENUMBER lets you change any or all box labels within a given model using the standard renumbering panel.

To change the label of an individual box it may be simpler just to **MODIFY** it.

Limitations of the box definition method that prohibit non-global orientations

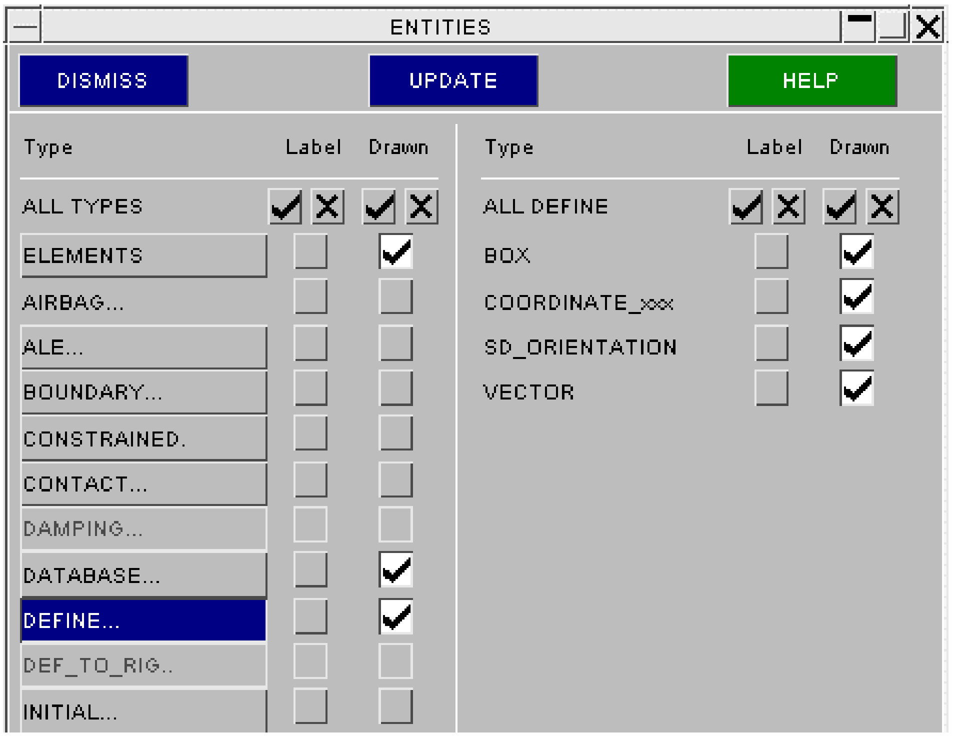
PRIMER allows you to create and edit boxes by both numerical (type in coordinates) and screen-based (pick and drag) methods. You can also transform them via the **ORIENT** function.

Note on ROTATION OF BOXES: The way boxes are specified in LS_DYNA (coordinates of two corners in global units) means that they are always aligned with the global [x,y,z] axes. If a box is rotated by any angle other than a multiple of 90 degrees it will be expanded to a global box which surrounds the actual rotated shape of the original box. Thus a box rotated by by 45 degrees will expand by a factor of 1.4. In such cases, the user will be warned and the contents of the new box sketched.

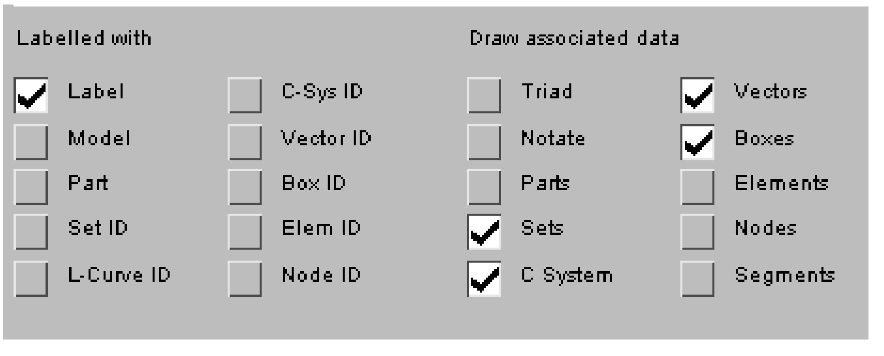
Visualising Boxes.

Boxes may be drawn by turning their display on in the **ENT**ity Viewing menu.

They can also be drawn via the **SKETCH** options above.



They may also be drawn in other contexts (for example contacts) if their display as "associated data" in the **ENT**ity Viewing menu is selected.

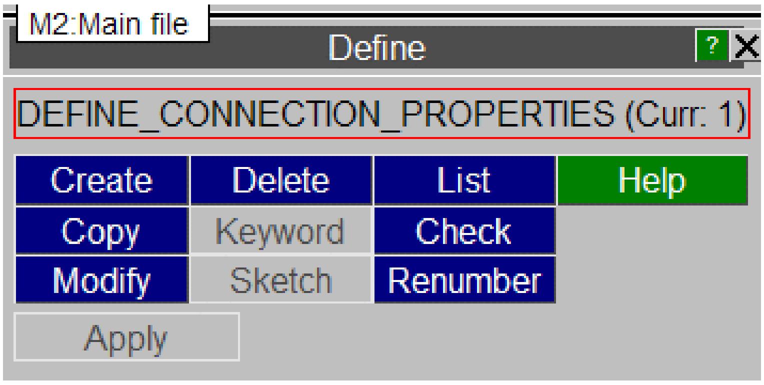


(DEFINE_) CONNECTION_PROPERTIES:

These can be edited through their own specific editing panel (see below).

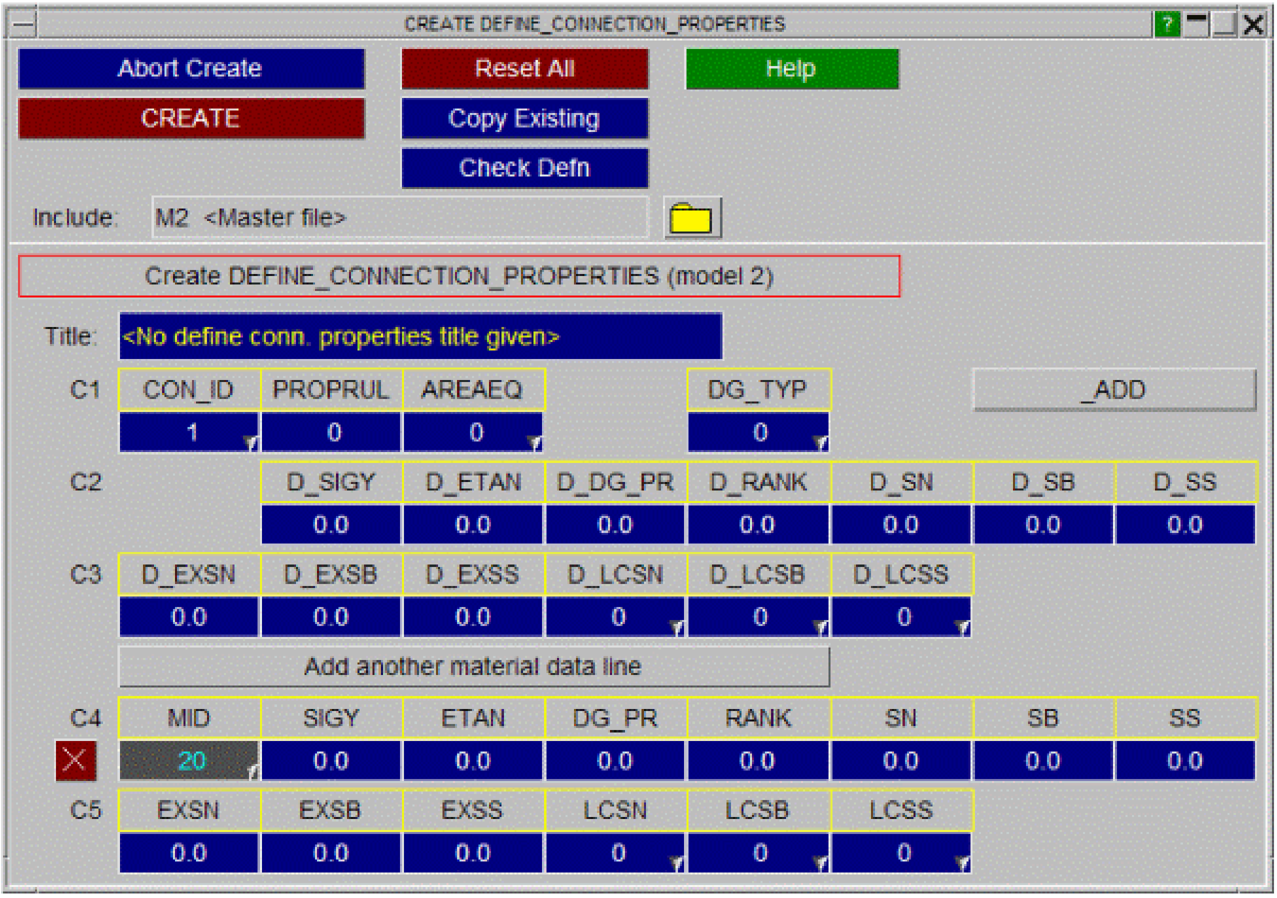
- Main Menu
- Creation
- Copying
- Editing
- Deletion
- List
- Check
- Renumber

This figure shows the main menu for the editing of connection properties definitions.
All functions have their standard meanings as given in section 5.0.1



CREATE Making a new connection properties definition.

This shows the create/edit panel for connection properties. New material data lines can be added by clicking on the **Add another material data line** button. The _ADD option can be activatied by clicking on the **_ADD** button. With the _ADD option active, cards 2 and 3 are greyed out.



COPY Copy existing connection properties(s) to make a new connection properties(s).

The selected connection properties are copied. (connection properties do not "own" anything, so the concept of recursive copying does not apply.)

MODIFY Modifying the attributes of an existing connection properties.

MODIFY functions in the same way as **CREATE**, except that an initial definition will be present. Any modifications made to the connection properties definition will not be made permanent until the **UPDATE** button is pressed. At this point the local copy which has been updated is used to overwrite the version in the model.

DELETE Delete existing connection properties definitions.

The selected connection properties are deleted.

Connection properties do not "own" anything, so the concept of recursive deletion does not apply, however a connection property that is referred to (ie "owned") by some higher order item will not be deletable unless that item is deleted too, or its reference to the connection propertie removed.

LIST List connection properties summaries to screen

The selected connection properties are summarised on the screen.

CHECK Check connection properties definitions for errors

The selected connection properties definitions are run through the standard checking routines.

RENUMBER Change connection properties labels

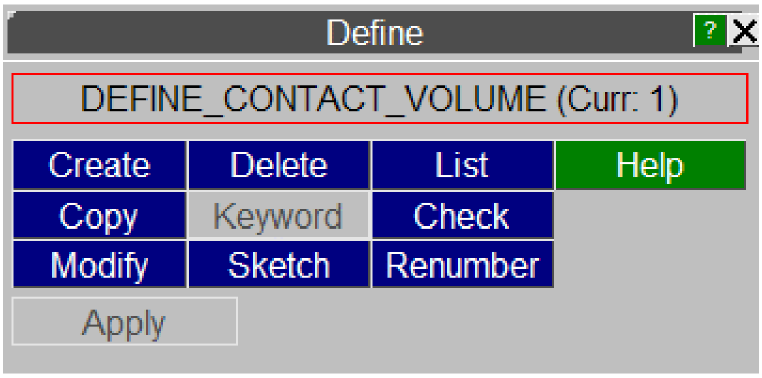
RENUMBER lets you change any or all connection properties labels within a given model using the standard renumbering panel. To change the label of an individual connection properties it may be simpler just to **MODIFY** it.

(DEFINE_) CONTACT_VOLUME:

These can be edited through their own specific editing panel (see below).

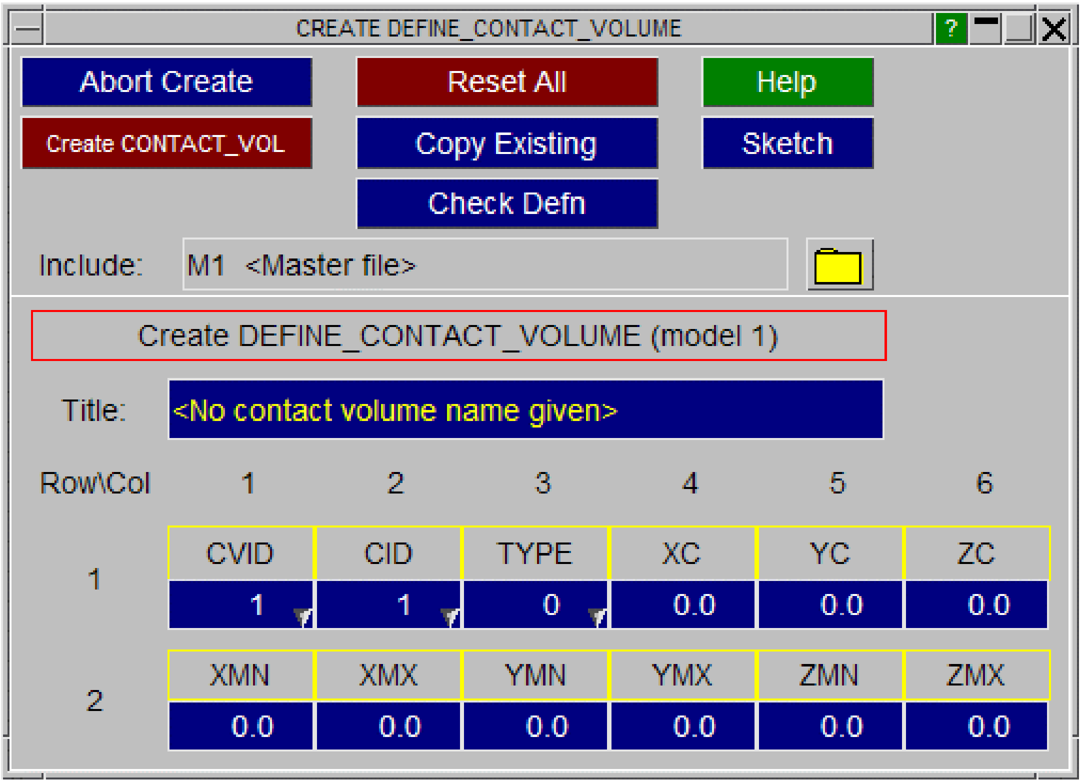
- Main Menu
- Creation
- Copying
- Editing
- Deletion
- Sketching
- List
- Check
- Renumber

This figure shows the main menu for the editing of contact volume definitions.
All functions have their standard meanings as given in section 5.0.1



CREATE Making a new contact volume definition.

This shows the create/edit panel for contact volumes. The second row of the card will change depending on the value chosen for **TYPE.**



COPY Copy existing contact volume(s) to make a new contact volume(s).

The selected contact volumes are copied. (contact volumes do not "own" anything, so the concept of recursive copying does not apply.)

MODIFY Modifying the attributes of an existing contact volume.

MODIFY functions in the same way as **CREATE**, except that an initial definition will be present. Any modifications made to the contact volume definition will not be made permanent until the **UPDATE_CONTACT_VOL** button is pressed. At this point the local copy which has been updated is used to overwrite the version in the model.

DELETE Delete existing contact volume definitions.

The selected contact volumes are deleted.

Contact volumes do not "own" anything, so the concept of recursive deletion does not apply, however a contact volume that is referred to (ie "owned") by some higher order item will not be deletable unless that item is deleted too, or its reference to the contact volume removed.

SKETCH Sketch contact volume definitions.

SKETCH draws the contact volume on top of the current graphics image.

LIST List contact volume summaries to screen

The selected contact volumes are summarised on the screen.

CHECK Check contact volume definitions for errors

The selected contact volume definitions are run through the standard checking routines.

RENUMBER Change contact volume labels

RENUMBER lets you change any or all contact volume labels within a given model using the standard renumbering panel. To change the label of an individual contact volume it may be simpler just to **MODIFY** it.

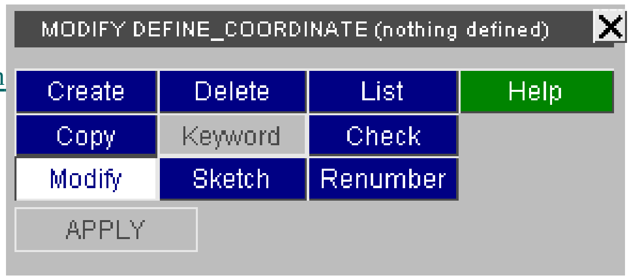
(DEFINE_) COORDINATE: Defining Coordinate Systems.

- Main Menu
- Creation
- Copying
- Editing
- Deletion
- Visualisation

The ***DEFINE_COORDINATE** keyword is used to create local coordinate systems. Three points in space, which form two vectors are required. The coordinate system is then computed from the cross product of these vectors. They are used when a system is required that is not orthogonal to the global axes. For example boundary conditions, orthtropic materials and beam orientations.

Coordinate systems use unique labels and, although part of the ***DEFINE** keyword, their labels do not clash with other ***DEFINE_xxx** entities. For example it is legal to have (***DEFINE_)BOX #1** and (***DEFINE_)COORDINATE #1**.

This figure shows the main menu for the editing of co-ordinate systems.
All functions have their standard meanings as given in section 5.0.1:



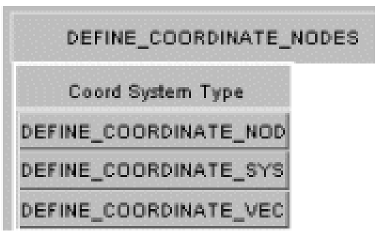
CREATE Making a coordinate definition.

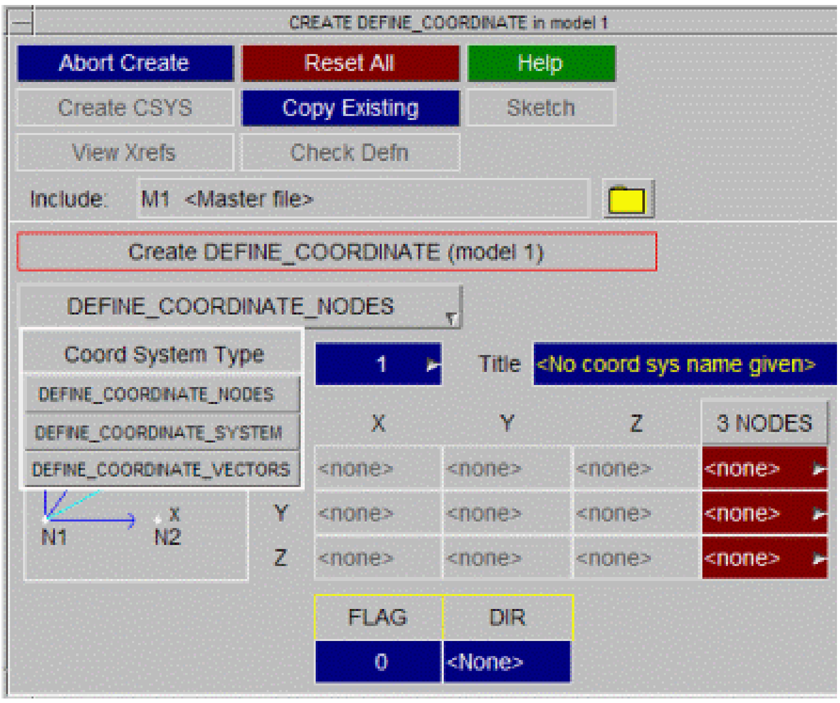
This figure shows the basic **CREATE** / **UPDATE COORDINATE_SYSTEM** panel.

There are three ways in LS-Dyna of defining a coordinate system:

DEFINE_COORDINATE_NODES
_SYSTEM
_VECTORS

The popup menu gives these options:



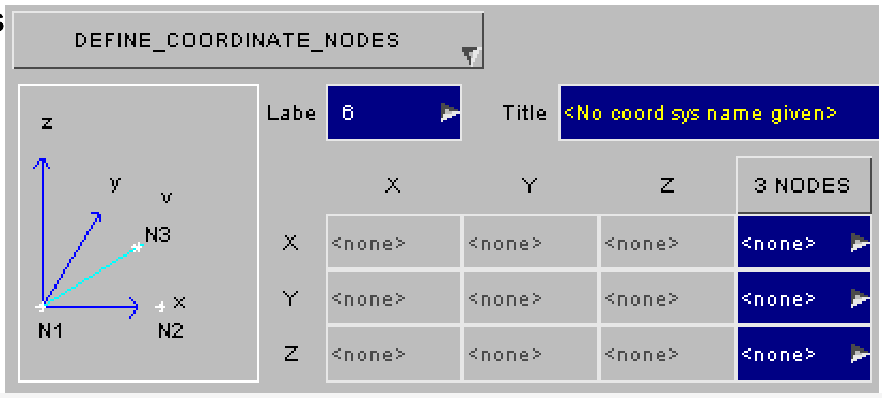


The detailed layout of the panels and definition methods vary slightly as shown below.

DEFINE_COORDINATE_NODES
Is defined by three nodes:

N1 : origin
N2 : Gives local X axis from N1N2
N3 : forms the local XY plane N1N2N3

Methods of defining the nodes:



3 NODES
Instead of defining each node separately, all nodes can be screen-picked together. Simply screen pick three nodes in the order:

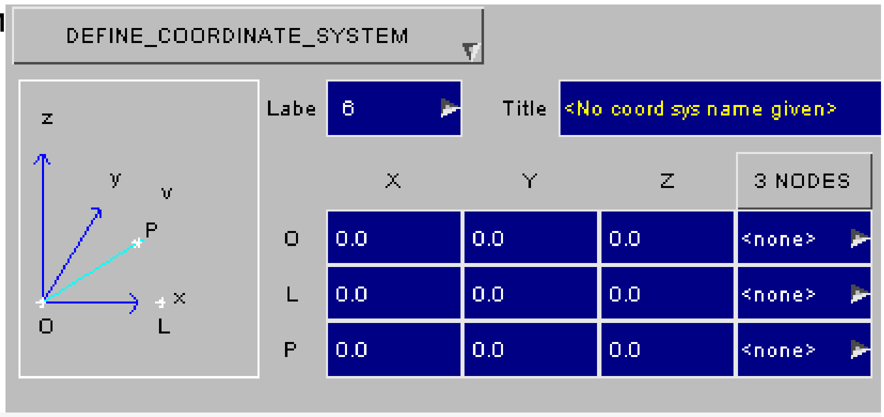
- N1 (origin)
- N2 (local X vector)
- N3 (lies on local XY plane)

<Individually>
Alternatively use the individual popup menus to select nodes, or simply type in their labels.

DEFINE_COORDINATE_SYSTEM
Is defined by three points:

P1 : origin
P2 : Gives local X axis from P1P2
P3 : forms the local XY plane P1P2P3

Methods of defining the points:



3 NODES
(Only their coordinates are used)
Instead of defining each node separately, all nodes can be screen-picked together. Simply screen pick three nodes in the order:

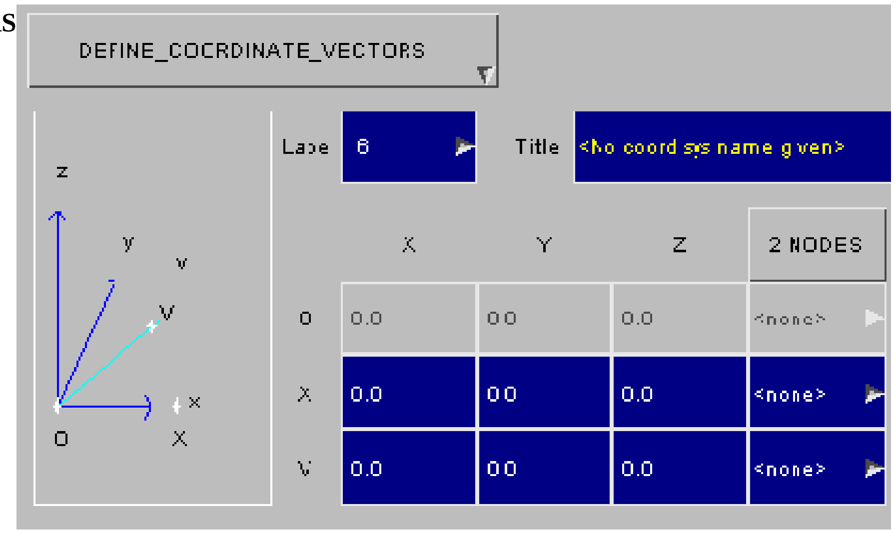
- N1 (origin)
- N2 (local X vector)
- N3 (lies on local XY plane)

<Individually>
Alternatively use the individual popup menus to select nodes, or simply type in their labels.
Or simply type in the coordinates explicitly.

DEFINE_COORDINATE_VECTORS
Is defined by the origin and 2 points:

Or : origin
P1 : Gives local X axis from OrP1
P2 : forms the local XY plane OrP1P2

Methods of defining the points:



2 NODES (Only their coordinates are used)	Instead of defining each node separately, both nodes can be screen-picked together. Simply screen pick three nodes in the order: • N1 (local X vector) • N2 (lies on local XY plane)
<Individually>	Alternatively use the individual popup menus to select nodes, or simply type in their labels. Or simply type in the coordinates explicitly.

COPY Copy existing coordinate(s) to make a new coordinate.

The selected coordinates are copied. (Coordinates do not "own" anything, so the concept of recursive copying does not apply.)

MODIFY Modifying the attributes of an existing coordinate.

MODIFY functions in the same way as **CREATE**, except that an initial definition will be present.

Any modifications made to the section definition will not be made permanent until the **UPDATE_CSYS** button is pressed. At this point a the local copy which has been updated is used to overwrite the version in the model.

DELETE Delete existing coordinate definitions.

The selected coordinates are deleted.

Coordinate definitions do not "own" anything, so the concept of recursive deletion does not apply, however a coordinate that is referred to (ie "owned") by some higher order item will not be deletable unless that item is deleted too, or its reference to the coordinate removed.

SKETCH Sketch coordinate definitions on the current image.

Allows the user to select coordinate systems and superimpose a white sketch of them over the currently displayed image.

LIST List coordinate summaries to screen

The selected coordinate definitions are summarised on the screen.

CHECK Check coordinate definitions for errors

The selected coordinate definitions are run through the standard checking routines.

RENUMBER Change coordinate labels

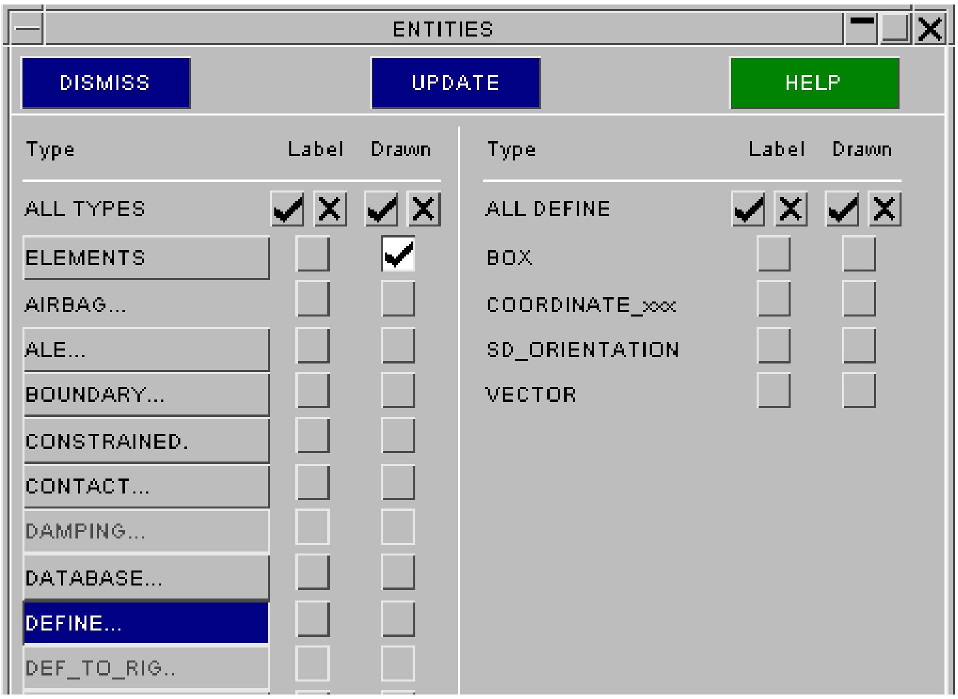
Lets you change any or all coordinate labels within a given model using the standard renumbering panel.

To change the label of an individual coordinate it may be simpler just to **MODIFY** it.

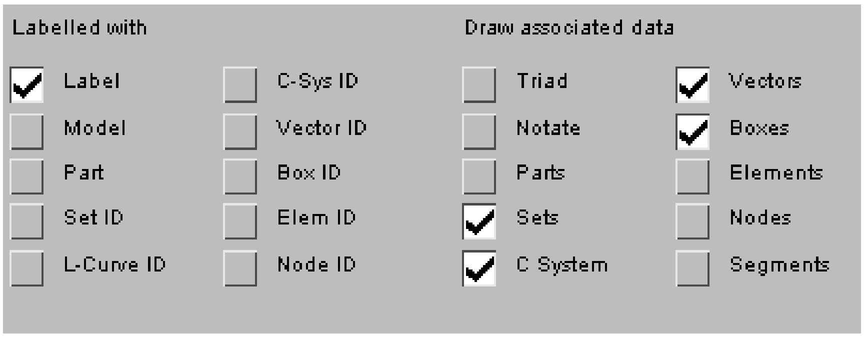
Visualising Coordinate systems.

Co-ordinate systems may be drawn by turning their display on in the **ENT**ity Viewing menu.

They can also be drawn via the **SKETCH** options above.



They may also be drawn in other contexts (for example contacts) if their display as "associated data" in the **ENT**ity Viewing box is selected.



(DEFINE_) CURVE/TABLE: Defining Load Curves.

- Main Menu
- Creation
- Copying
- Editing
- Deletion

Other curve suffices:

_COMPENSATION
_ENTITY
_FEEDBACK
_FUNCTION
_SMOOTH
_TRIM

The ***DEFINE_CURVE** keyword is used to create "loadcurves". These are lists of two or more (x, y) data points which are used extensively for defining loading (e.g. force vs. time), material properties (e.g. stress vs. strain) and other varying data in an LS-DYNA analysis.

The ***DEFINE_TABLE** keyword defines a table of loadcurves. A table is an ordered set of data pairs consisting of a value and a loadcurve id, typically a strain rate and a stress:strain characteristic. It is an unfortunate quirk of the LS-Dyna keyword input that the loadcurves belonging to a table **must follow it in sequential order**. PRIMER endeavours to maintain this ordering but care must be taken if decks are edited manually, or split into *include files, to ensure that this order is adhered to.

Loadcurves do not have any explicit data types or units associated with them, this is implied by the items which reference them. It is legal, but generally not sensible, for any number of unrelated items to use the same loadcurve. It may cause problems for Unit change operations.

PRIMER keeps track of what references each loadcurve, and hence the implied data types and units for each axis, which makes it possible to detect and correct conflicting usage.

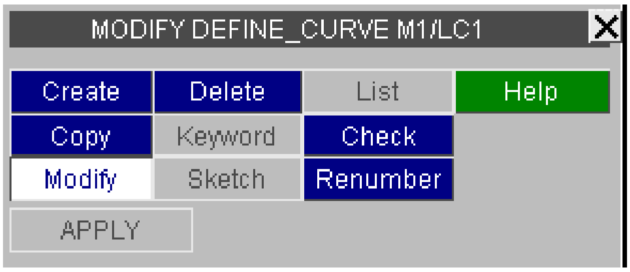
Loadcurves use unique labels and, although part of the ***DEFINE** keyword, their labels do not clash with other ***DEFINE_xxx** entities ***except tables*** (see below). For example it is legal to have **(*DEFINE_)BOX #1** and **(*DEFINE_)CURVE #1.**

LOADCURVES.

NOTE: TABLE and **CURVE** definitions occupy the same labelling space, and are interchangeable in some contexts. Thus it is ***not*** legal to have **TABLE#1** and **CURVE#1**.

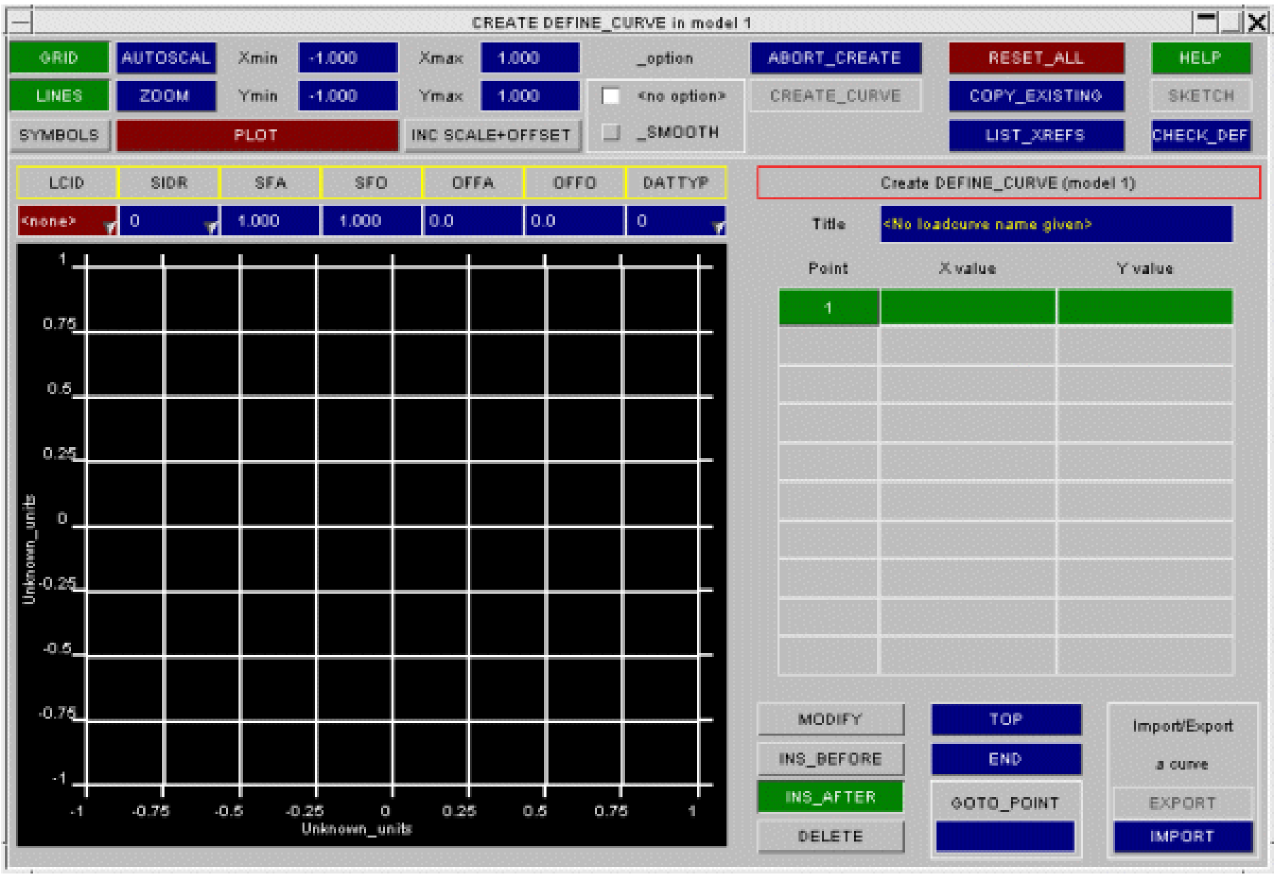
This figure shows the main menu for the editing of curves. All functions have their standard meanings as given in section 5.0.1

The table and curve main menu panels are similar.



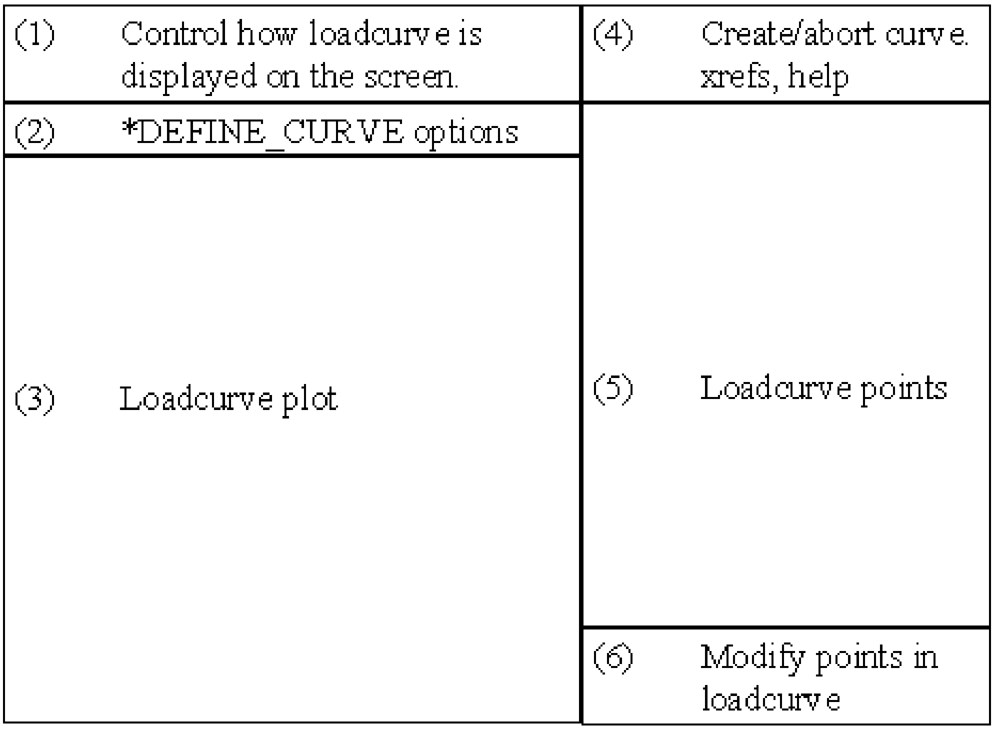
CREATE Making a new loadcurve definition.

This figure shows the basic **CREATE** / **UPDATE CURVE** panel.



The loadcurve editing panel layout is shown in this figure.
There are six main areas in the panel, each area grouping together buttons of similar function.

1. **Loadcurve display buttons**
2. ***DEFINE_CURVE options.**
3. **Loadcurve plot.**
4. **Create/abort loadcurve.**
5. **Loadcurve points.**
6. **Loadcurve modification.**



(1) Loadcurve display buttons

GRID, LINES & SYMBOLS

These buttons toggle whether the grid, the curve line and the curve symbols are drawn on the plot.

AUTOSCALE

Resets the scaling on the loadcurve plot so the curve just fits the screen and replots the loadcurve.

ZOOM

Two points are selected using the left mouse button. **Xmin, Xmax, Ymin** and **Ymax** are updated and the curve is replotted

PLOT

The curve is replotted as the scale currently selected by **Xmin, Xmax, Ymin** and **Ymax**.

Xmin, Xmax, Ymin & Ymax

Typing in a value changes the limits for plotting the curve.

By default when creating a curve **Xmin** and **Ymin** are -1. **Xmax** and **Ymax** are 1. When modifying an existing curve they are set so the curve just fits on the screen (equivalent to **AUTOSCALE**)

INC SCALE+OFFSET

By default when a curve is plotted on the screen the offsets (**SFA, SFO**) and scale factors (**OFFA, OFFO**) are not included. If this button is pressed then they are included in the plot. To ensure that the user is aware of this the **SFA, SFO, OFFA** and **OFFO** text boxes turn green (by default they are blue) and the curve line and symbols are plotted in green. Pressing the button again toggles the inclusion off.

Loadcurve values are scaled after the offsets are applied.

Abcissa value = SFA x *(Defined value + OFFA)*

Ordinate value = SFO x *(Defined value + OFFO)*

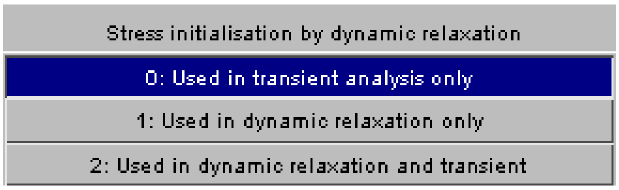
(2) *DEFINE_CURVE options.

LCID

Label for loadcurve. If there is no label then the label is shown as **<none>** and the box is red rather than the default blue. A new label can be typed in the box or the right mouse button pressed to get the standard label popup box.

SIDR

Sets whether the loadcurve will be used in a transient or dynamic relaxation analysis. Either type in the value or use the right mouse button to bring up a popup menu.

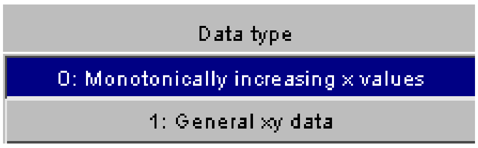


SFA, SFO, OFFA & OFFO

Scale factors and offsets for the loadcurve abcissa (x) and ordinate (y) values.

DATTYP

Sets the type of data in the loadcurve: generally this is set to zero. Either type in a value or use the right mouse button to bring up a popup menu.



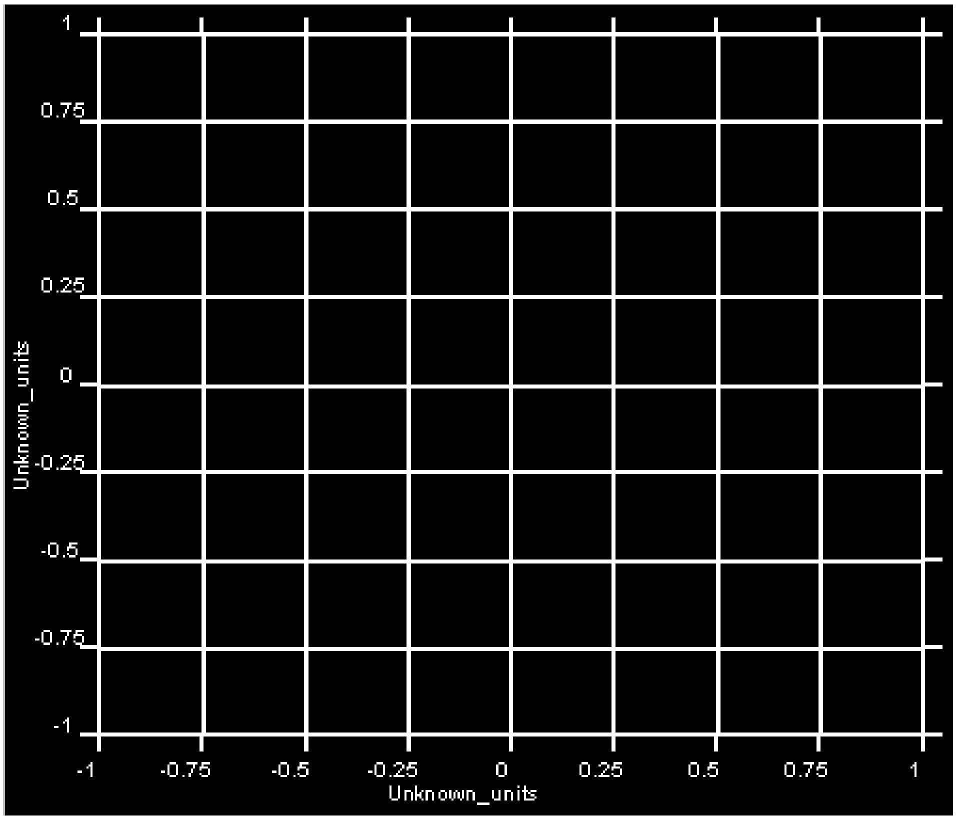
(3) Loadcurve plot.
The loadcurve currently being created or modified is plotted in the bottom left of the loadcurve panel.

If the curve has no cross references then the units for the X and Y axes are shown as Unknown units. If there are cross references, the first reference that PRIMER finds is used and the units displayed on the X and Y axes. All the cross references for the curve can be displayed with the **LIST_XREFS** button.

The visibility of the curve lines, symbols and the grid is controlled by the **GRID, LINES** and **SYMBOLS** buttons.

If the **INC_SCALE+OFFSET** button is selected, the curve is drawn in green instead of the default yellow to inform the user that the scale factors and offsets are included in the plot.

The plot can be updated at any time by pressing the **PLOT** button.



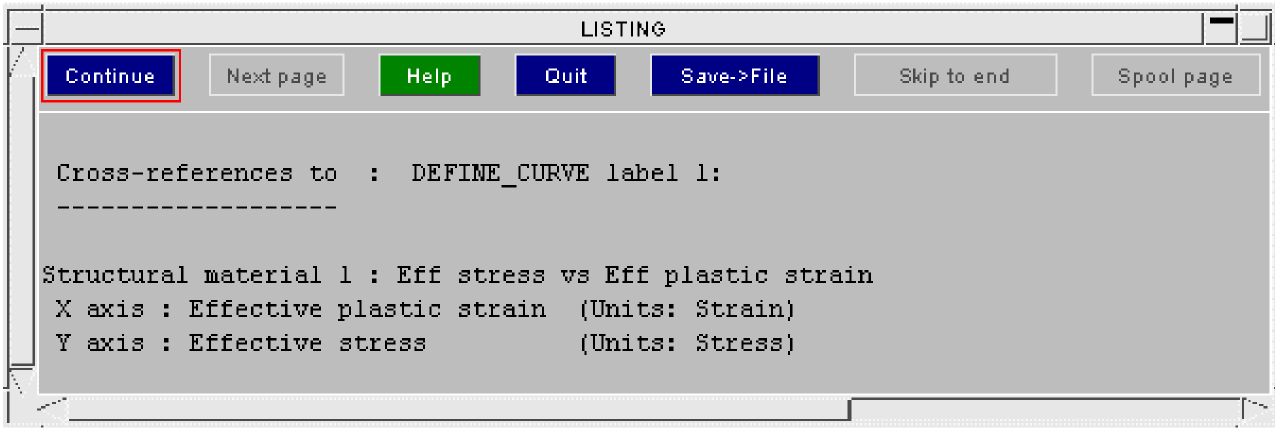
(4) Create/abort loadcurve.

ABORT_CREATE
Aborts from the current loadcurve creation without saving any of the modifications.

CREATE_CURVE
This will exit the current loadcurve creation, saving the curve in the database. This button will be inactive (greyed out) until a label (**LCID**) is given for the loadcurve and there are at least two points in the curve.

LIST_XREFS

If the loadcurve has any cross references in the database they are displayed in a dialogue window. If there are no cross references to this curve `[no cross references found]` will be displayed.



RESET_ALL

Resets the curve back to its initial state. Any points which have been added or modified are lost.

COPY_EXISTING

Copies the data from an existing loadcurve into the loadcurve currently being created. Any points which have been added since starting the create will be lost.

CHECK_DEFN

Checks the loadcurve currently being created for any errors.

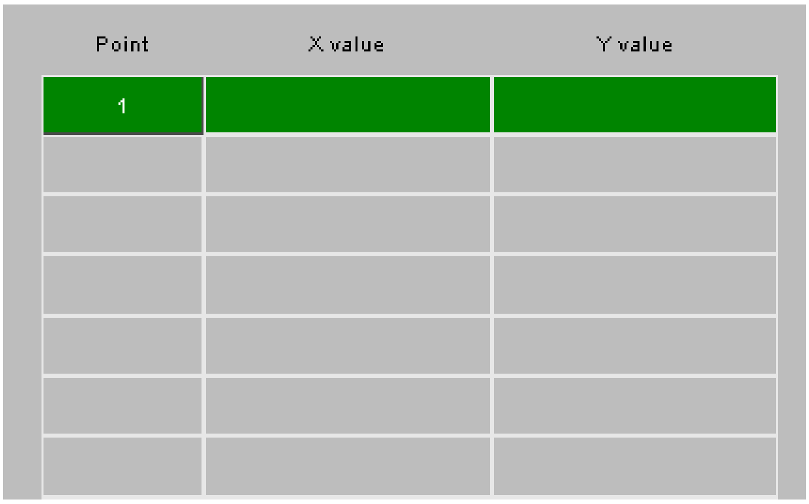
HELP

Displays the help pages for loadcurves

SKETCH

Sketch is currently inoperative.

(5) Loadcurve points.
When a curve is created the user is forced into **INS_AFTER** mode until a point is created. The user will not be able to change to another mode (**MODIFY, INS_BEFORE** or **DELETE**) until this point is created.



Point	X value	Y value
1		

The **X** and **Y value** boxes for this initial point are blank and coloured green to indicate that a number is required. If a number is typed into one of the boxes the box turns blue.

When numbers are present for X and Y the line for point 1 becomes blue and point 2 becomes green. As many points as necessary can be added using this method.

Point	X value	Y value
1	-70.00	0.0
2	-55.00	0.0
3	-53.00	-500.0
4	3.000	31.00
5	4.000	60.00

When there is more than one point the current mode can be changed at any time by pressing the **MODIFY, INS_BEFORE** or **DELETE** buttons.
Any incomplete points (ie if either the X or Y values [or both] are blank) will be deleted when changing mode. If the number of points in the loadcurve is greater than 10 a sliding bar appears by the side of the points. The mouse can be used to select which points are visible in the text box. Drag the bar up and down with the left mouse button to move between the points. Alternatively clicking on the up (or down) arrow with the left, middle or right mouse button, increases (or decreases) the points shown by 1, 10 or 100 respectively. The value of a point can be changed in any mode by clicking on the X or Y value box and typing in a number.

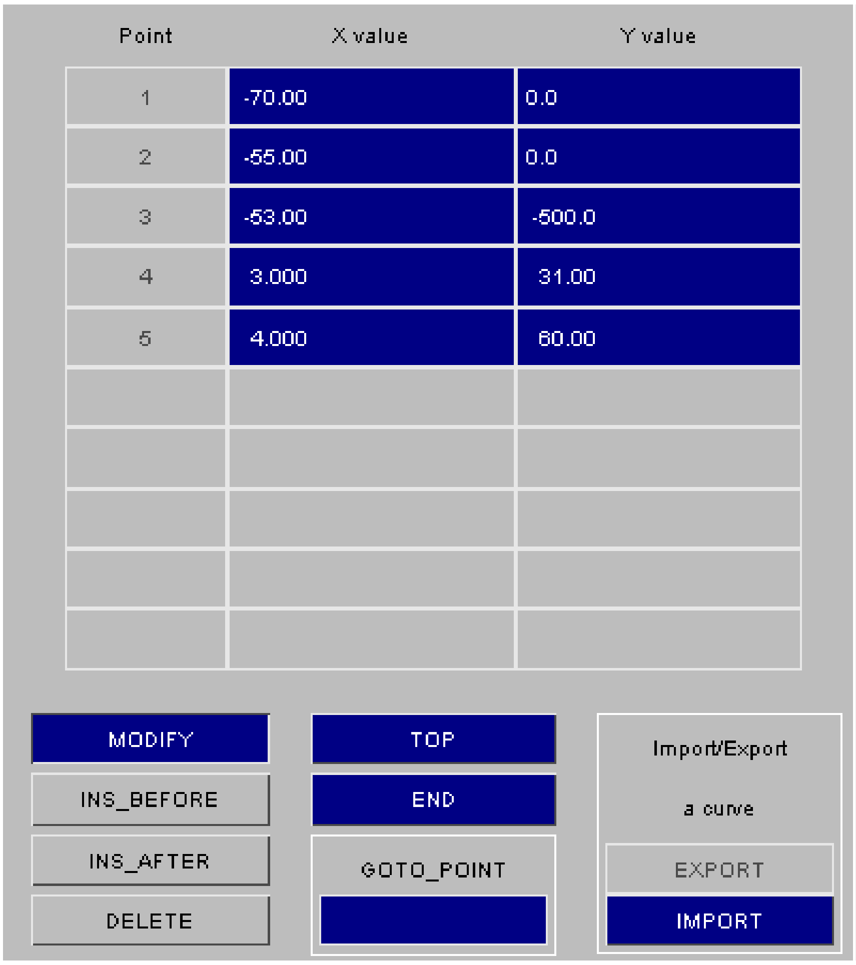
(6) Loadcurve modification.

MODIFY

Selects modify mode for loadcurve point editing. In this mode only the values of the points can be changed. No points can be added or deleted.

When in this mode the point buttons are greyed out so they cannot be selected.

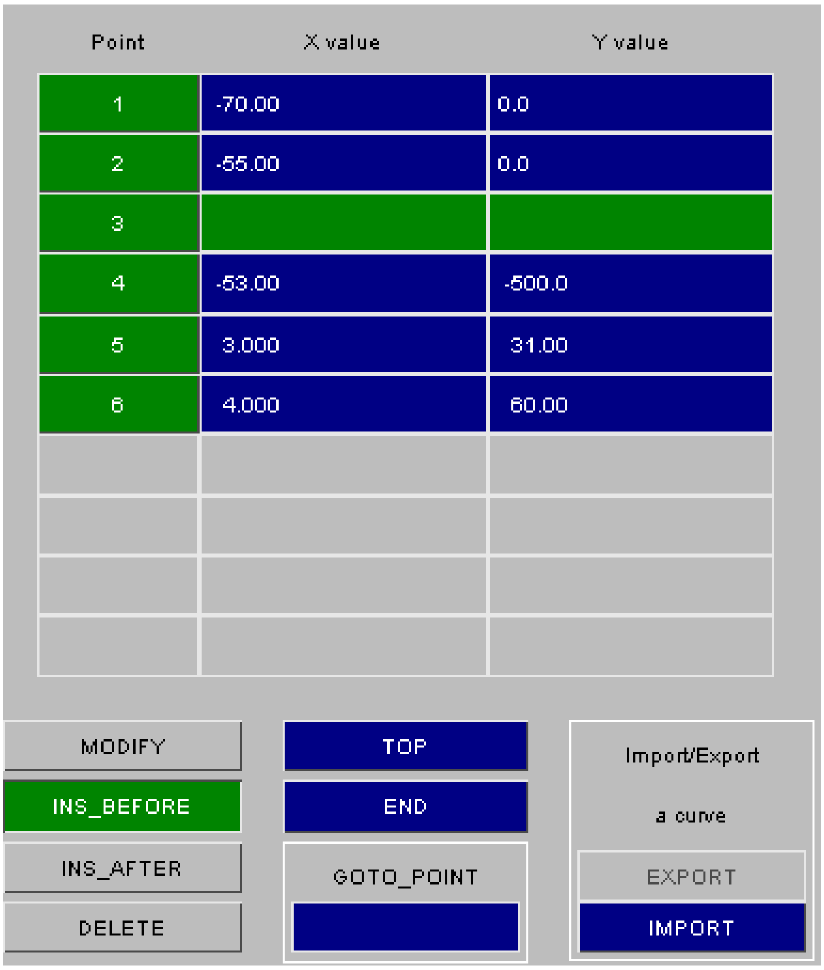
If a point is currently being edited in **INS_BEFORE** or **INS_AFTER** mode it is deleted before the modify mode is selected.



INS_BEFORE & INS_AFTER

Select either **INS_BEFORE** or **INS_AFTER** mode for loadcurve point editing. This allows points to be added to the curve.

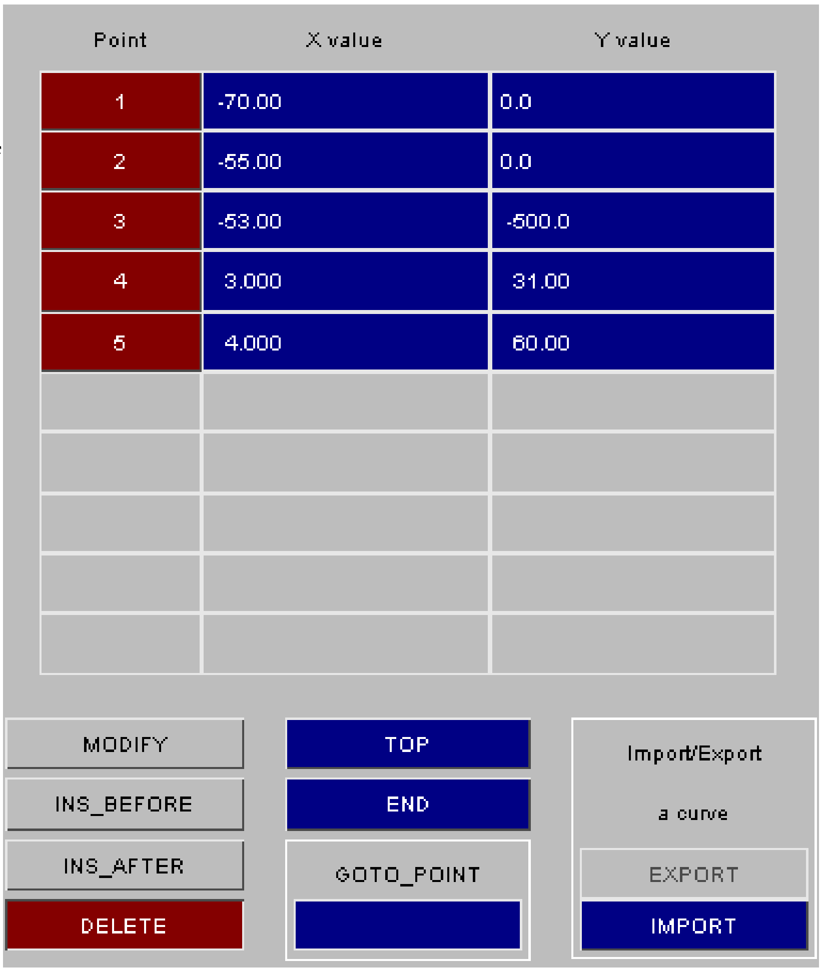
When in these modes the point buttons turn green. If a point is selected by clicking with the mouse a new point is created either before or after (depending on which mode) the selected point. If a point is currently being added in this mode and another point is selected the current point is deleted and the new point added.



DELETE

Selects delete mode for loadcurve point editing. In this mode points can be deleted as well as being able to change the values of the points.

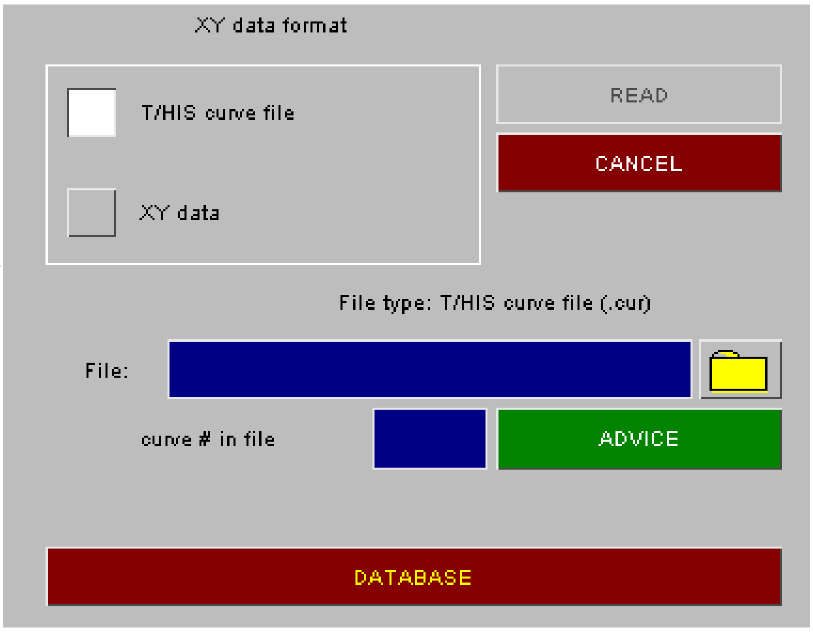
When in this mode the point buttons turn red. If a point is selected by clicking with the mouse it is deleted. If a point is currently being edited in **INS_BEFORE** or **INS_AFTER** mode it is deleted before the delete mode is selected.



TOP & **END**
: Moves the slider automatically to the top or end of the points for the loadcurve

GOTO_POINT
: Moves the slider so that the point number which is typed in is visible.

IMPORT
: Allows a loadcurve to be read from an external file or from a database in PRIMER. Pressing the **IMPORT** button brings up a new set of buttons instead of the loadcurve points.

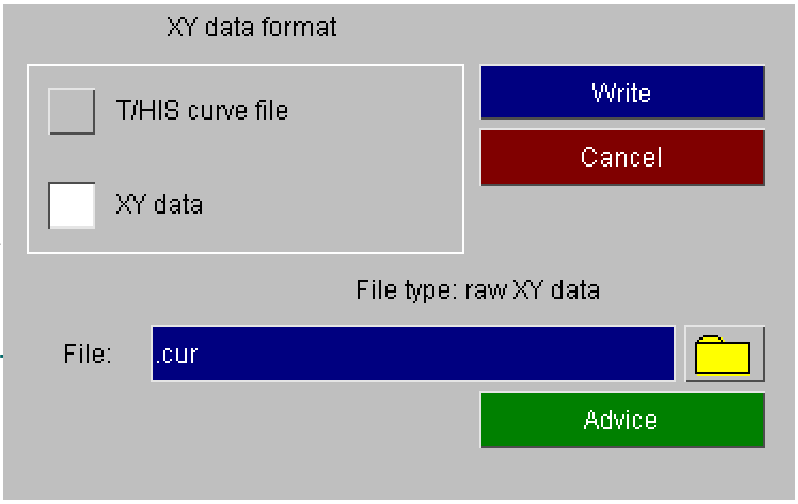


Two types of file can be read into the loadcurve editor. T/HIS curve files and raw x,y data. The formats of these files is given in Appendix VIII. The format of the file to import is selected by using the RADIO buttons. The filename can then either be typed in the file text box or selected by browsing using the ? button. T/HIS curve files can contain multiple curves in one file. In this case the curve number in the file to read should be given. If no number is given the first curve in the file will be read.

EXPORT

Allows a loadcurve to be written to an external file fromPRIMER. Pressing the **EMPORT** button brings up a new set of buttons instead of the loadcurve points.

Two types of file can be written from the loadcurve editor. T/HIS curve files and CSV x,y data. The formats of the T/HIS curve data is given in Appendix VIII. The format of the file to import is selected by using the RADIO buttons. The filename can then either be typed in the file text box or selected by browsing using the ? button.



READ

Reads the selected file into the loadcurve editor and plots the curve. Any modifications to the current curve will be lost when importing a file.

CANCEL

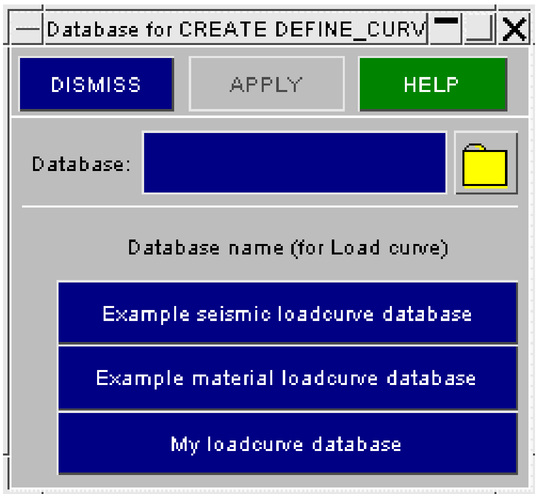
Aborts the import and returns to the normal loadcurve editor window.

DATABASE

The **DATABASE** button starts the loadcurve database function in PRIMER.

A list of the available loadcurve databases will be shown on the screen. When one is selected a curve can be read from the database.

For further details on databases see section 5.2 and Appendix IX.



COPY Copy existing loadcurve(s) to make a new loadcurve(s).

The selected loadcurves are copied. (Loadcurves do not "own" anything, so the concept of recursive copying does not apply.)

MODIFY Modifying the attributes of an existing loadcurve.

MODIFY functions in the same way as **CREATE**, except that an initial definition will be present. Any modifications made to the loadcurve definition will not be made permanent until the **UPDATE_CURVE** button is pressed. At this point a the local copy which has been updated is used to overwrite the version in the model.

DELETE Delete existing loadcurve definitions.

The selected loadcurves are deleted.

Loadcurves do not "own" anything, so the concept of recursive deletion does not apply, however a loadcurve that is referred to (ie "owned") by some higher order item will not be deletable unless that item is deleted too, or its reference to the loadcurve removed.

SKETCH Sketch loadcurve definitions.

SKETCH is currently inoperative.

LIST List loadcurve summaries to screen

The selected loadcurves are summarised on the screen.

CHECK Check loadcurve definitions for errors

The selected loadcurve definitions are run through the standard checking routines.

RENUMBER Change loadcurve labels

RENUMBER lets you change any or all loadcurve labels within a given model using the standard renumbering panel. To change the label of an individual loadcurve it may be simpler just to **MODIFY** it.

TABLES

CREATE/MODIFY Making/modifying a table definition.

This figure shows the **CREATE/UPDATE TABLE** panel.

The functionality of the table editing panel is similar to the panel for loadcurves. The following features are briefly described.

Adding a new entry: when a value and loadcurve id (select, create, or type in) are entered into the top box **ADD_ENTRY** will become active. The new entry will automatically be added to the correct row.

Deleting an entry: a row is deleted from the table using popup of the row id button. The curve itself is NOT deleted.

Sketching an entry: the same popup may be used to sketch a single curve

Blanking/Unblanking an entry: clicking on the green (unblanked) row id button will toggle it to red (blanked)

Editing an entry: any row may be modified by typing in a new value and new loadcurve id. The latter may also be selected through the **SELECT** popup. Further, a loadcurve itself may be created/edited as the loadcurve editing panel can be accessed directly via the **CREATE** & **EDIT** popup.

Reordering entries: If entries become out of order the **REORDER** button will sort them.

Exporting a table: exports all the table curves to a t/his format (2e20) curve file. The table values are output as a step diagram.

*DEFINE_CURVE_COMPENSATION

This definition defines a curve for local compensation..

PRIMER will read them in and write them out, but no interactive editing of them is provided

*DEFINE_CURVE_ENTITY

This definition defines a curve of straight line segments and circular arcs that defines an axisymmetric surface.

PRIMER will read them in and write them out, but no interactive editing of them is provided

*DEFINE_CURVE_FEEDBACK

These definitions do not create a loadcurve, rather they add special metal-forming attributes to an existing curve definition.

PRIMER will read them in and write them out, but no interactive editing of them is provided.

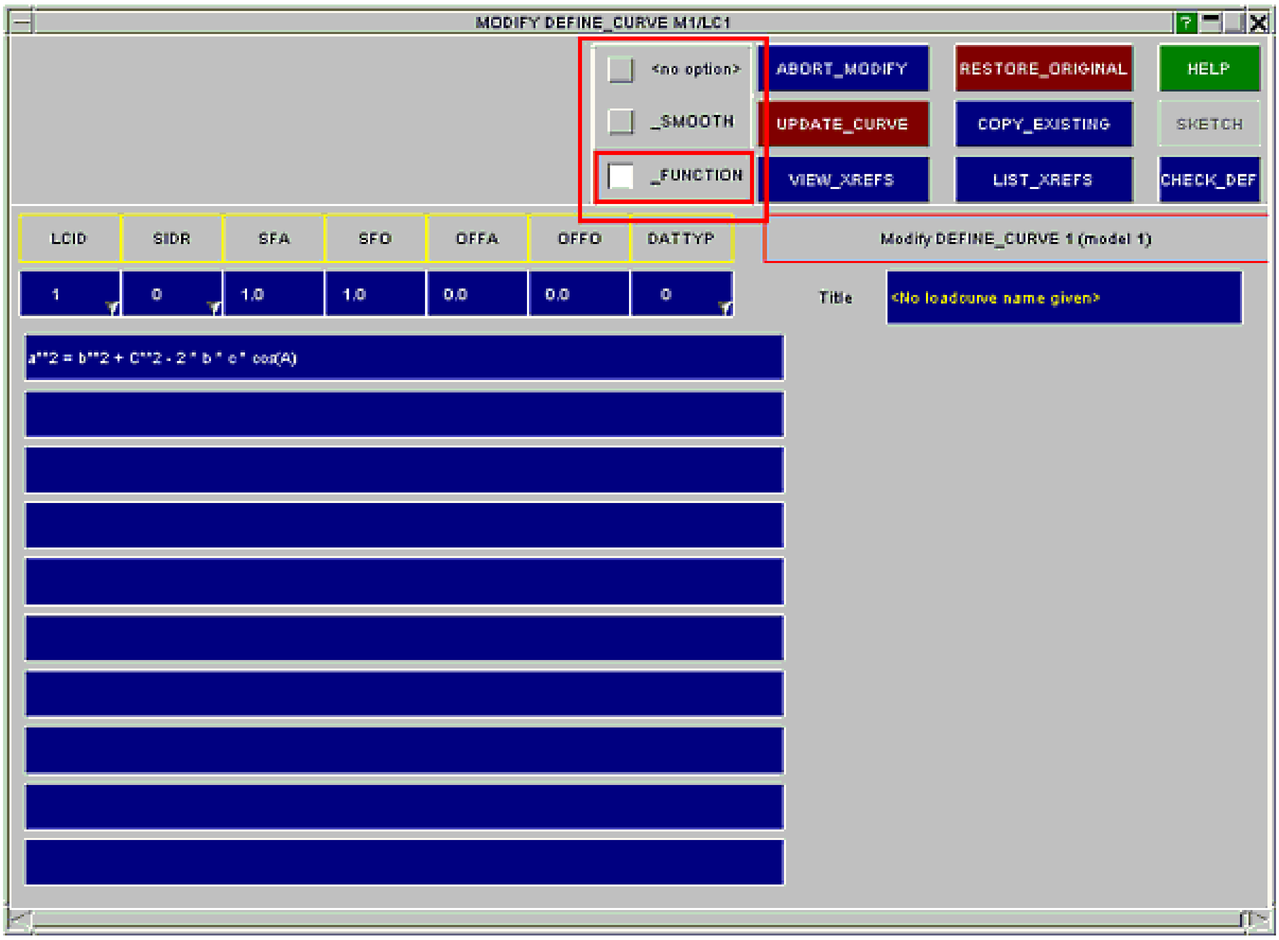
*DEFINE_CURVE_FUNCTION

These definitions are an alternative way of creating a loadcurve. Instead of explicit <x,y> data the user enters up to 10 rows of pseudo-fortran syntax, which may also contain references to functions that return the current state of ls-dyna entities during an analysis.

PRIMER reads these in and writes them out, but does not attempt to evaluate them. As a consequence only very limited checking is provided for them.

These definitions may be edited in the normal curve editor as follows:

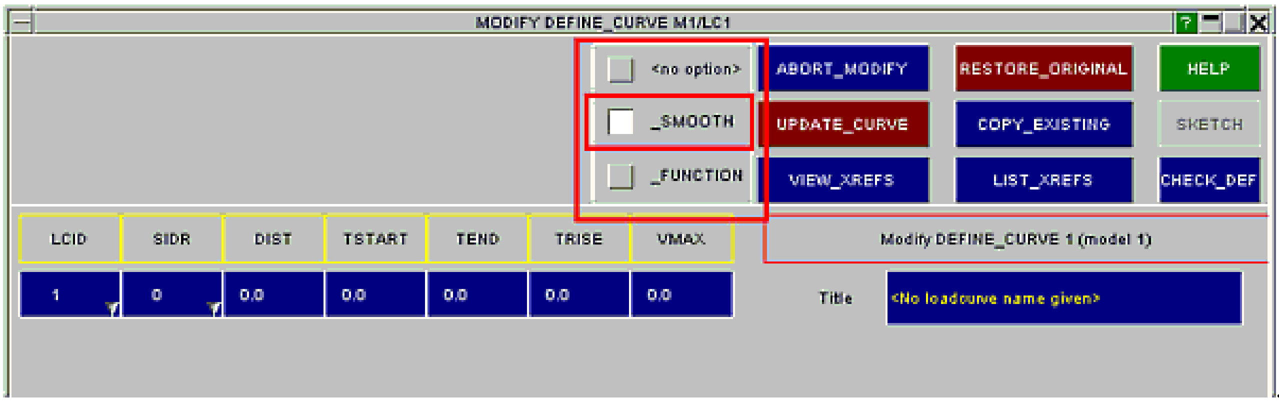
- Select **_FUNCTION** in the options box of the curve editor
- Enter up to 10 rows of data.



*DEFINE_CURVE_SMOOTH

This is yet another way to define a loadcurve. In this case a smoothly varying "hump" function is defined in terms of its parameters.

These definitions may be edited in the normal curve editor as follows:
- Select **_SMOOTH** in the options box of the curve editor
- Enter the relevant parameters



*DEFINE_CURVE_TRIM

These are not loadcurves at all (the name is misleading), rather they are geometrical definitions used with *ELEMENT_TRIM during springback analyses. Their label sequence is totally separate from that of conventional loadcurves, and there is no relationship between the two types.

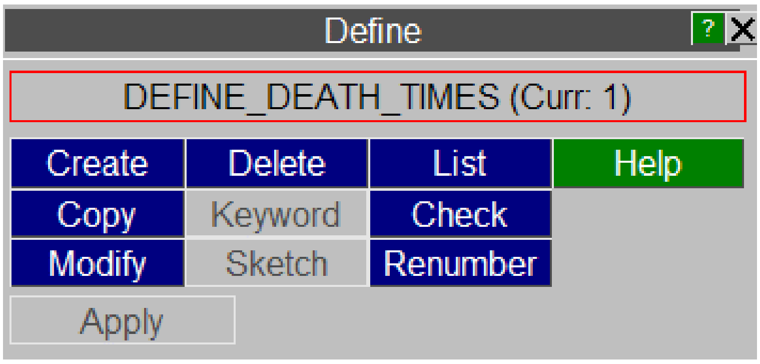
PRIMER reads in and writes out these definitions, but does not provide any interactive editing or visualisation of them.

(DEFINE_) DEATH_TIMES:

These can be edited through their own specific editing panel (see below).

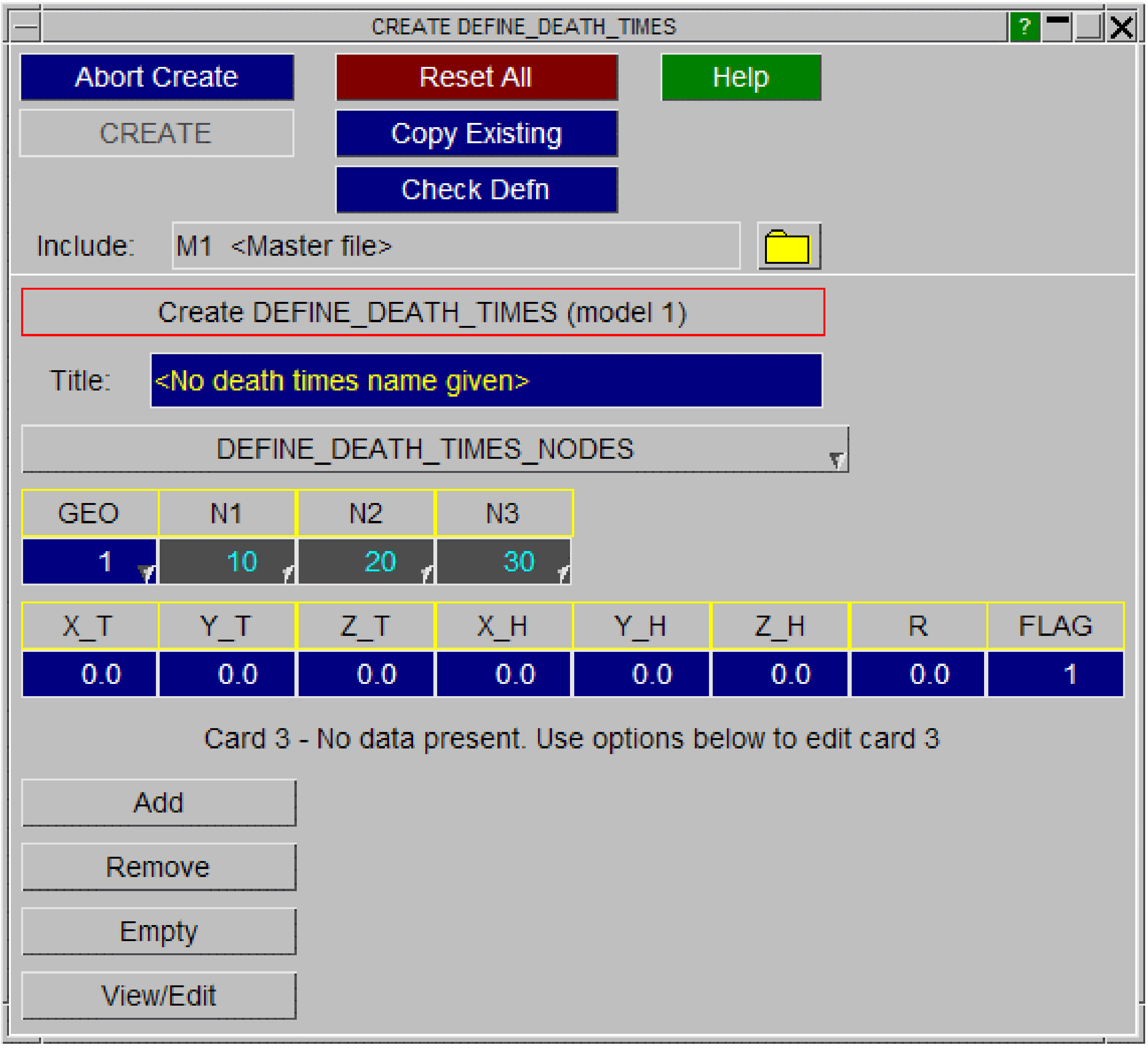
- Main Menu
- Creation
- Copying
- Editing
- Deletion
- List
- Check
- Renumber

This figure shows the main menu for the editing of death times definitions.
All functions have their standard meanings as given in section 5.0.1



CREATE Making a new death times definition.

This shows the create/edit panel for death times. Clicking on the **DEFINE_DEATH_TIMES_...** button will cycle through the _NODES, _SET and _RIGID options. Once the desired option is chosen, the information for card 3 can be added/modified using the **Add**, **Remove**, **Empty** and **View/Edit** buttons at the bottom of the panel.



COPY Copy existing death times(s) to make a new death times(s).

The selected death times are copied. (death times do not "own" anything, so the concept of recursive copying does not apply.)

MODIFY Modifying the attributes of an existing death times.

MODIFY functions in the same way as **CREATE**, except that an initial definition will be present. Any modifications made to the death times definition will not be made permanent until the **UPDATE_ALEBAG_INF** button is pressed. At this point the local copy which has been updated is used to overwrite the version in the model.

DELETE Delete existing death times definitions.

The selected death times are deleted.

Death times do not "own" anything, so the concept of recursive deletion does not apply, however a death times that is referred to (ie "owned") by some higher order item will not be deletable unless that item is deleted too, or its reference to the death times removed.

LIST List death times summaries to screen

The selected death times are summarised on the screen.

CHECK Check death times definitions for errors

The selected death times definitions are run through the standard checking routines.

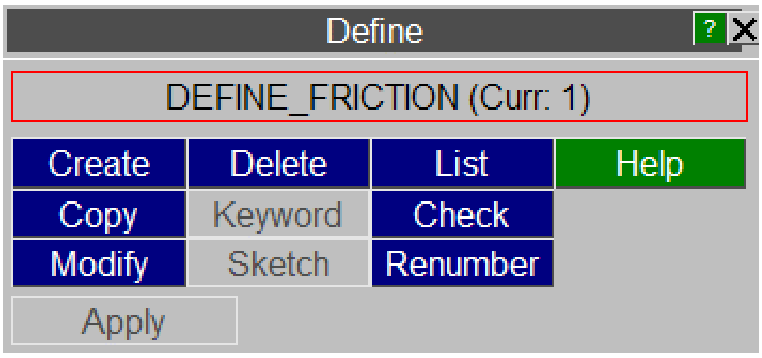
RENUMBER Change death times labels

RENUMBER lets you change any or all death times labels within a given model using the standard renumbering panel. To change the label of an individual death times it may be simpler just to **MODIFY** it.

(DEFINE_) FRICTION:

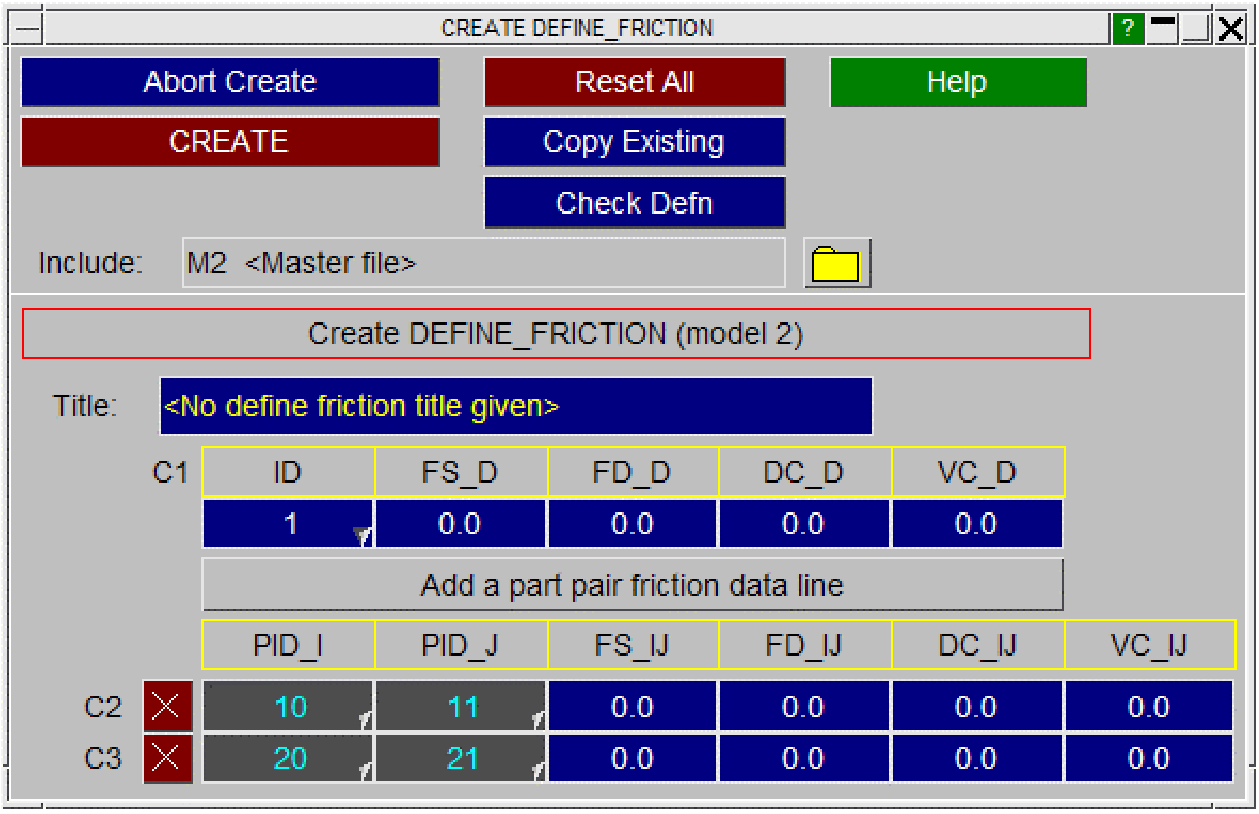
- These can be edited through their own specific editing panel (see below).
- Main Menu
- Creation
- Copying
- Editing
- Deletion
- List
- Check
- Renumber

This figure shows the main menu for the editing of friction definitions.
All functions have their standard meanings as given in section 5.0.1



CREATE Making a new friction definition.

This shows the create/edit panel for friction. New friction data lines can be added by clicking on the **Add a part pair friction data line** button.



COPY Copy existing define friction(s) to make a new define friction(s).

The selected define frictions are copied. (Define frictions do not "own" anything, so the concept of recursive copying does not apply.)

MODIFY Modifying the attributes of an existing friction.

MODIFY functions in the same way as **CREATE**, except that an initial definition will be present. Any modifications made to the define friction definition will not be made permanent until the **UPDATE** button is pressed. At this point the local copy which has been updated is used to overwrite the version in the model.

DELETE Delete existing define friction definitions.

The selected define frictions are deleted.

Define frictions do not "own" anything, so the concept of recursive deletion does not apply, however a connection property that is referred to (ie "owned") by some higher order item will not be deletable unless that item is deleted too, or its reference to the define friction removed.

LIST List friction summaries to screen

The selected define frictions are summarised on the screen.

CHECK Check define friction definitions for errors

The selected define friction definitions are run through the standard checking routines.

RENUMBER Change define friction labels

RENUMBER lets you change any or all define friction labels within a given model using the standard renumbering panel. To change the label of an individual define friction it may be simpler just to **MODIFY** it.

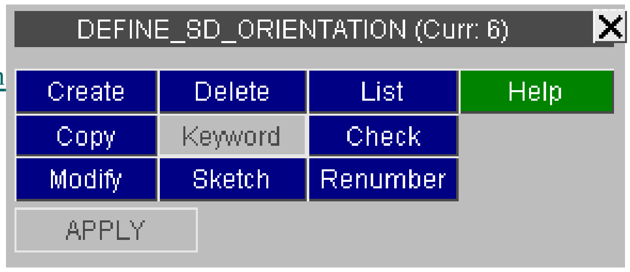
(DEFINE_) SD_ORIENTATION: Defining Spring & Damper Orientation Vectors.

- Main Menu
- Creation
- Copying
- Editing
- Deletion
- Visualisation

The ***DEFINE_SD_ORIENTATION** keyword is used to create orientation vectors for springs and dampers. These are vectors used to define the direction of the element, and are used primarily for rotational elements - which are generally of zero length.
However orientation vectors can also be used for translational springs/dampers of finite length, and they define the direction in which the elements acts, which can be different to the "natural" vector defined by its topology.

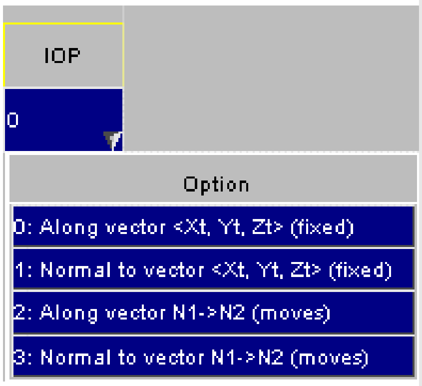
Orientation vectors use unique labels and, although part of the ***DEFINE** keyword, their labels do not clash with other ***DEFINE_xxx** entities. For example it is legal to have **(*DEFINE_)SD_ORIENTATION #1** and **(*DEFINE_)CURVE #1**.

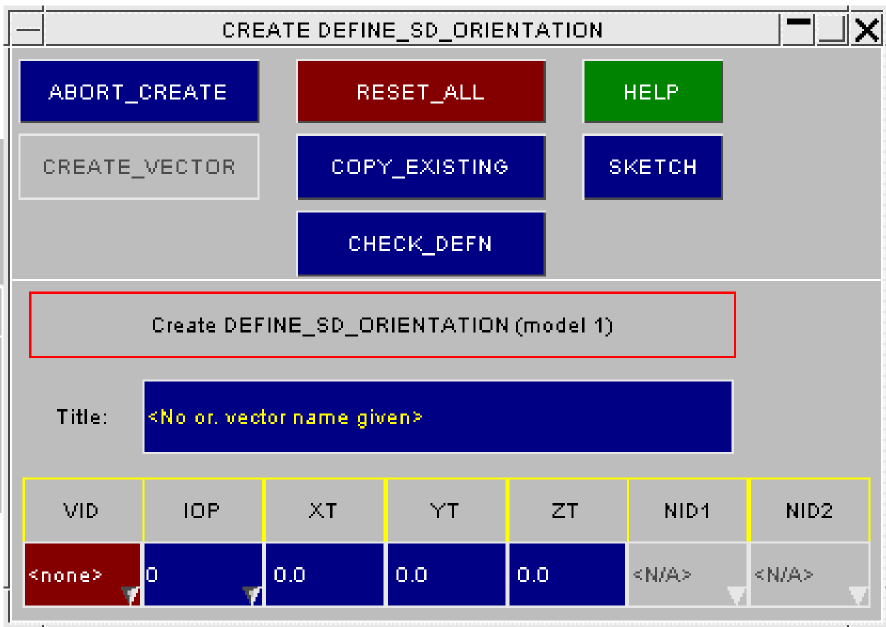
This figure shows the main menu for the editing of orientation vector definitions.
All functions have their standard meanings as given in section 5.0.1



CREATE Making a new orientation vector definition.

This shows the create/edit panel for orientation vectors.
<**IOP**> defines the orientation vector definition method.





Methods #2 and #3 make the **NID1** and **NID2** boxes "live" for selection.

COPY Copy existing orientation vector(s) to make a new vector(s).

The selected orientation vectors are copied. (orientation vectors do not "own" anything, so the concept of recursive copying does not apply.)

MODIFY Modifying the attributes of an existing orientation vector.

MODIFY functions in the same way as **CREATE**, except that an initial definition will be present. Any modifications made to the orientation vector definition will not be made permanent until the **UPDATE_VECTOR** button is pressed. At this point a the local copy which has been updated is used to overwrite the version in the model.

DELETE Delete existing orientation vector definitions.

The selected orientation vectors are deleted.

Orientation vectors do not "own" anything, so the concept of recursive deletion does not apply, however a orientation vector that is referred to (ie "owned") by some higher order item will not be deletable unless that item is deleted too, or its reference to the orientation vector removed.

SKETCH Sketch orientation vector definitions.

SKETCH draws the vector on top of the current graphics image.

LIST List orientation vector summaries to screen

The selected orientation vectors are summarised on the screen.

CHECK Check orientation vector definitions for errors

The selected orientation vector definitions are run through the standard checking routines.

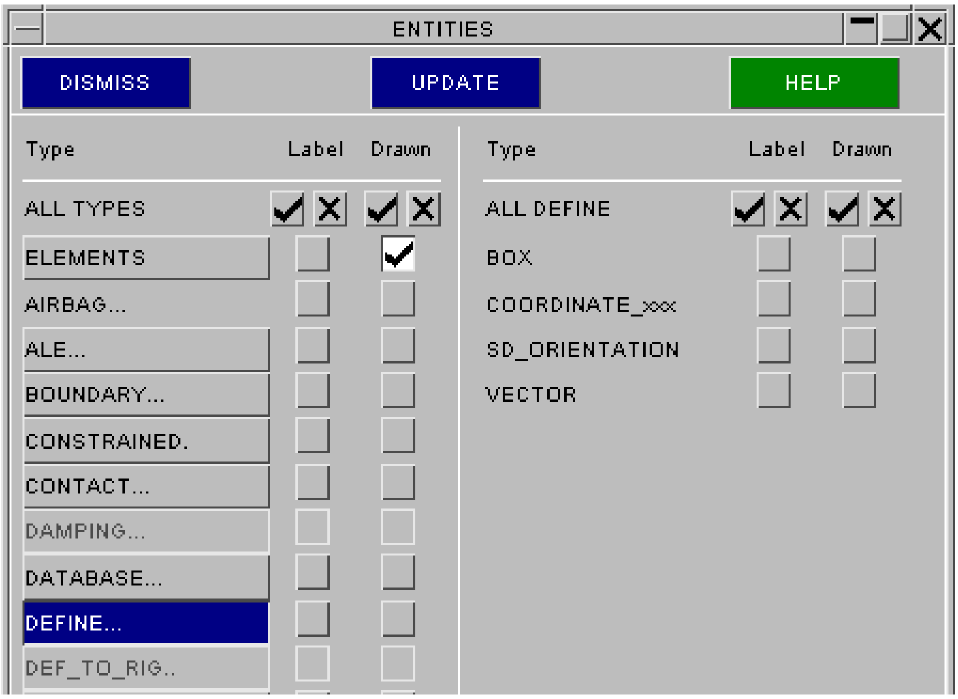
RENUMBER Change orientation vector labels

RENUMBER lets you change any or all orientation vector labels within a given model using the standard renumbering panel. To change the label of an individual orientation vector it may be simpler just to **MODIFY** it.

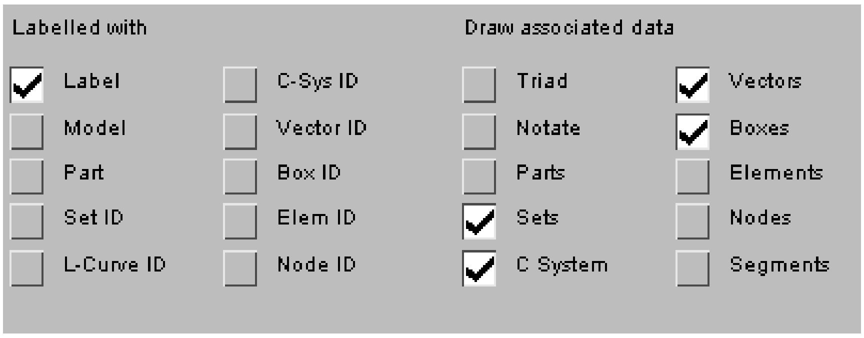
Visualising Orientation Vectors

Orientation Vectors may be drawn by turning their display on in the **ENT**ity Viewing menu.

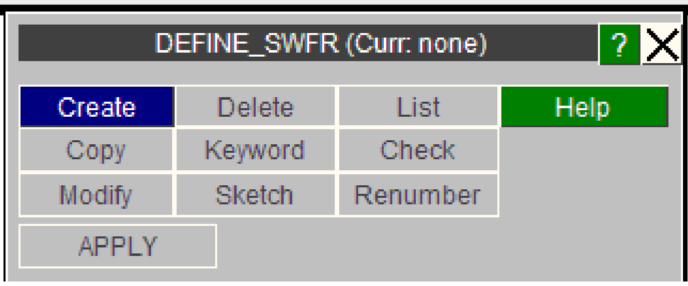
They can also be drawn via the **SKETCH** options above.



They may also be drawn in other contexts (for example contacts) if their display as "associated data" in the **ENT**ity Viewing box is selected.

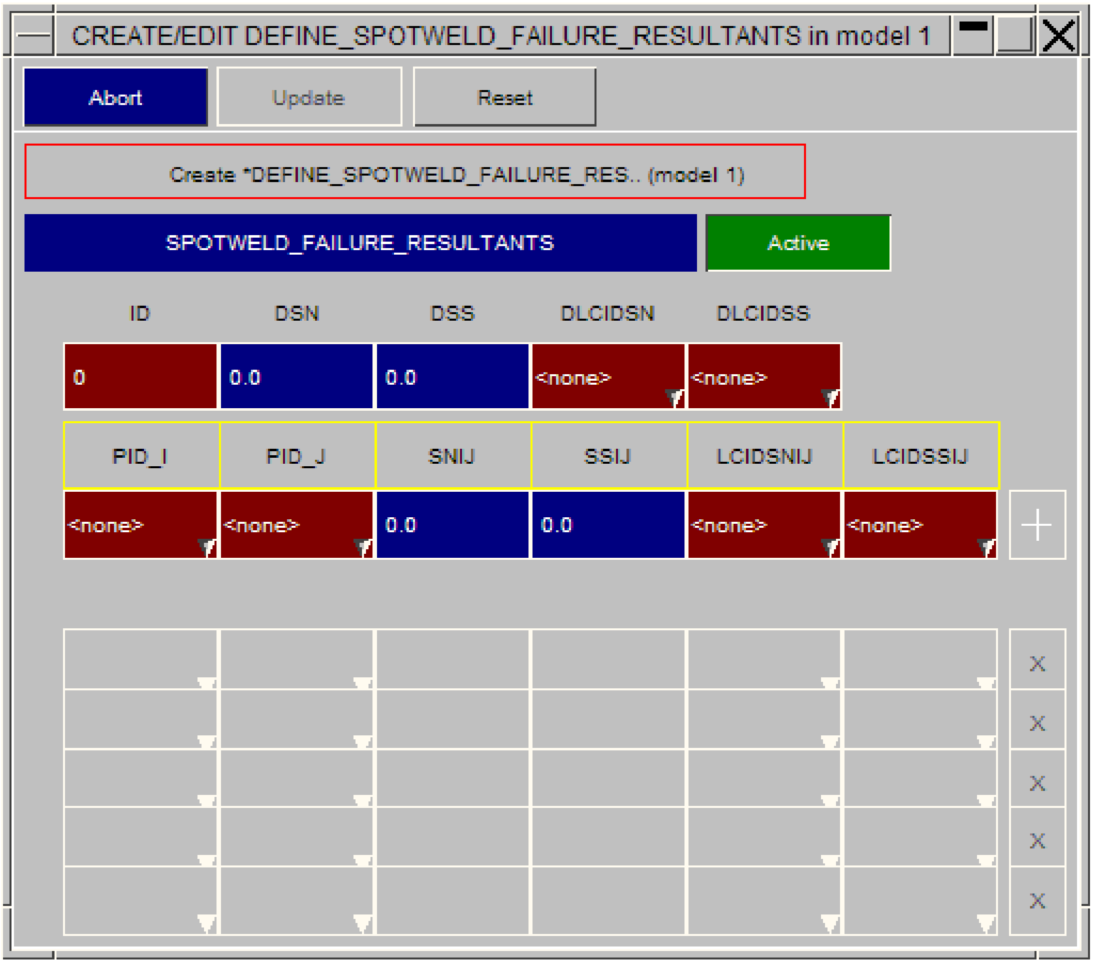


(DEFINE_) SPOTWELD_FAILURE_RESULTANTS:



This can be edited through the create and modify buttons shown above.

There is only one occurance of the DEFINE_SPOTWELD_FAILURE_RESULTANTS in each model. The following panel is used to edit the keyword:



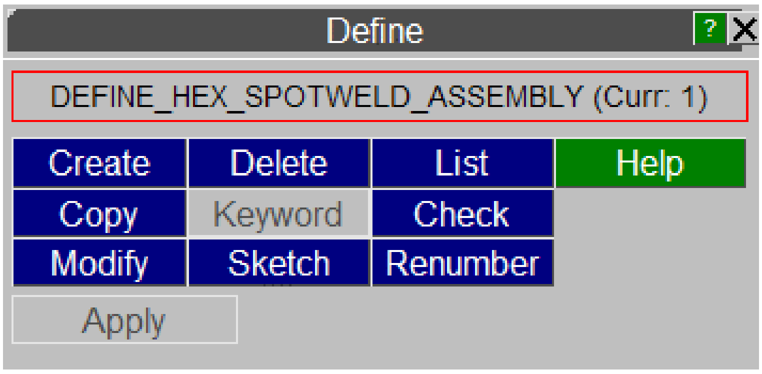
Enter the values in the relevant boxes, and hit the "+" button to add a row, or the "X" button to delete a row.

If you wish to deactivate the card from your model, press the green "Active" button to turn it grey (off).

(DEFINE_) HEX_SPOTWELD_ASSEMBLY:

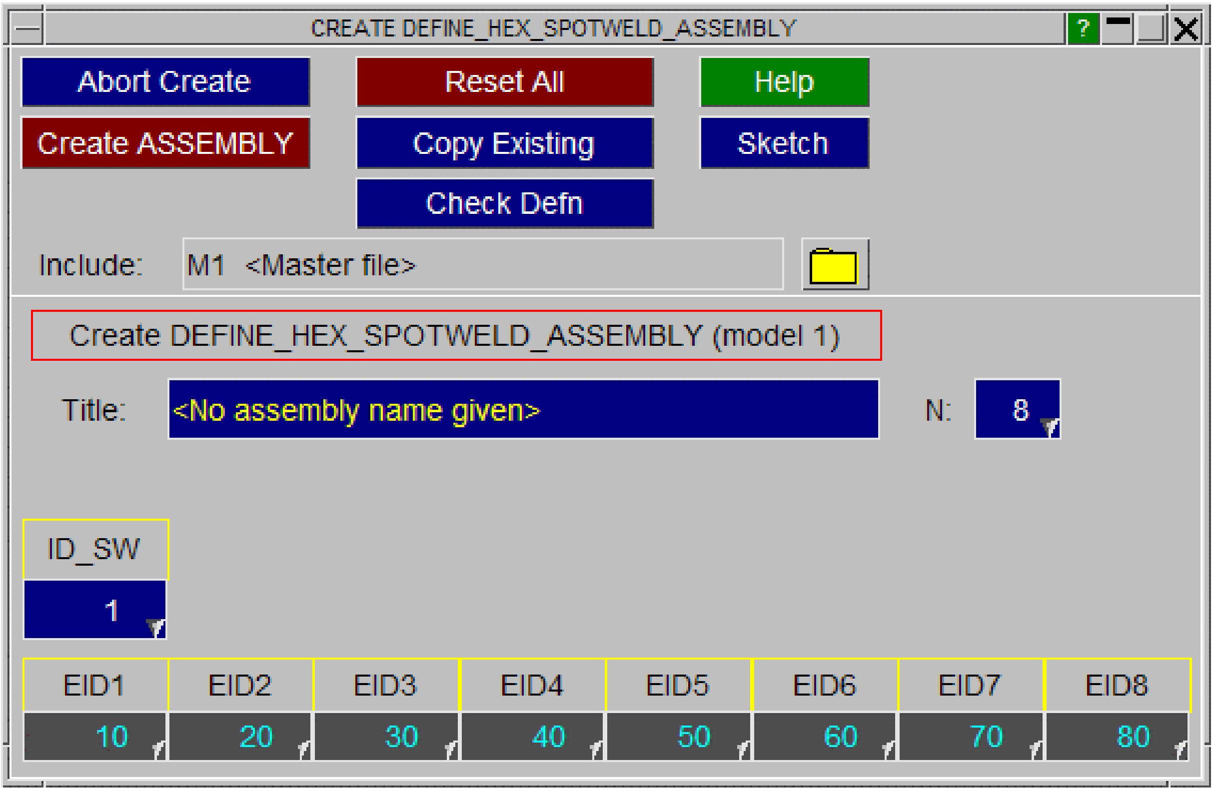
- These can be edited through their own specific editing panel (see below). Main Menu
- Creation
- Copying
- Editing
- Deletion
- Sketching
- List
- Check
- Renumber

This figure shows the main menu for the editing of hex spotweld assembly definitions.
All functions have their standard meanings as given in section 5.0.1



CREATE Making a new hex spotweld assembly definition.

This shows the create/edit panel for hex spotweld assemblies. The second row of the card will change depending on the value chosen for **'N'. 'N'** can be set to 4, 8 or 16 from a drop down list for that button.



COPY Copy existing hex spotweld assembly(s) to make a new hex spotweld assembly(s).

The selected hex spotweld assemblies are copied. (hex spotweld assemblies do not "own" anything, so the concept of recursive copying does not apply.)

MODIFY Modifying the attributes of an existing hex spotweld assembly.

MODIFY functions in the same way as **CREATE**, except that an initial definition will be present. Any modifications made to the hex spotweld assembly definition will not be made permanent until the **UPDATE** button is pressed. At this point the local copy which has been updated is used to overwrite the version in the model.

DELETE Delete existing hex spotweld assembly definitions.

The selected hex spotweld assemblies are deleted.

Hex spotweld assemblies do not "own" anything, so the concept of recursive deletion does not apply, however a hex spotweld assembly that is referred to (ie "owned") by some higher order item will not be deletable unless that item is deleted too, or its reference to the hex spotweld assembly removed.

SKETCH Sketch hex spotweld assembly definitions.

SKETCH draws the hex spotweld assembly on top of the current graphics image.

LIST List hex spotweld assembly summaries to screen

The selected hex spotweld assemblies are summarised on the screen.

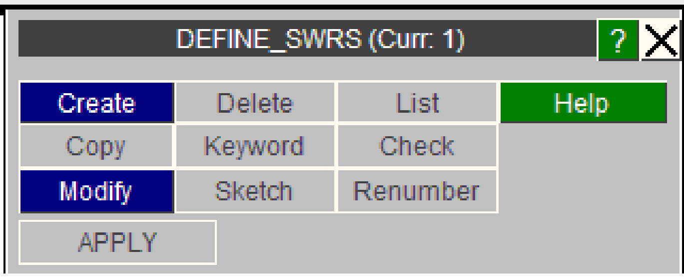
CHECK Check hex spotweld assembly definitions for errors

The selected hex spotweld assembly definitions are run through the standard checking routines.

RENUMBER Change hex spotweld assembly labels

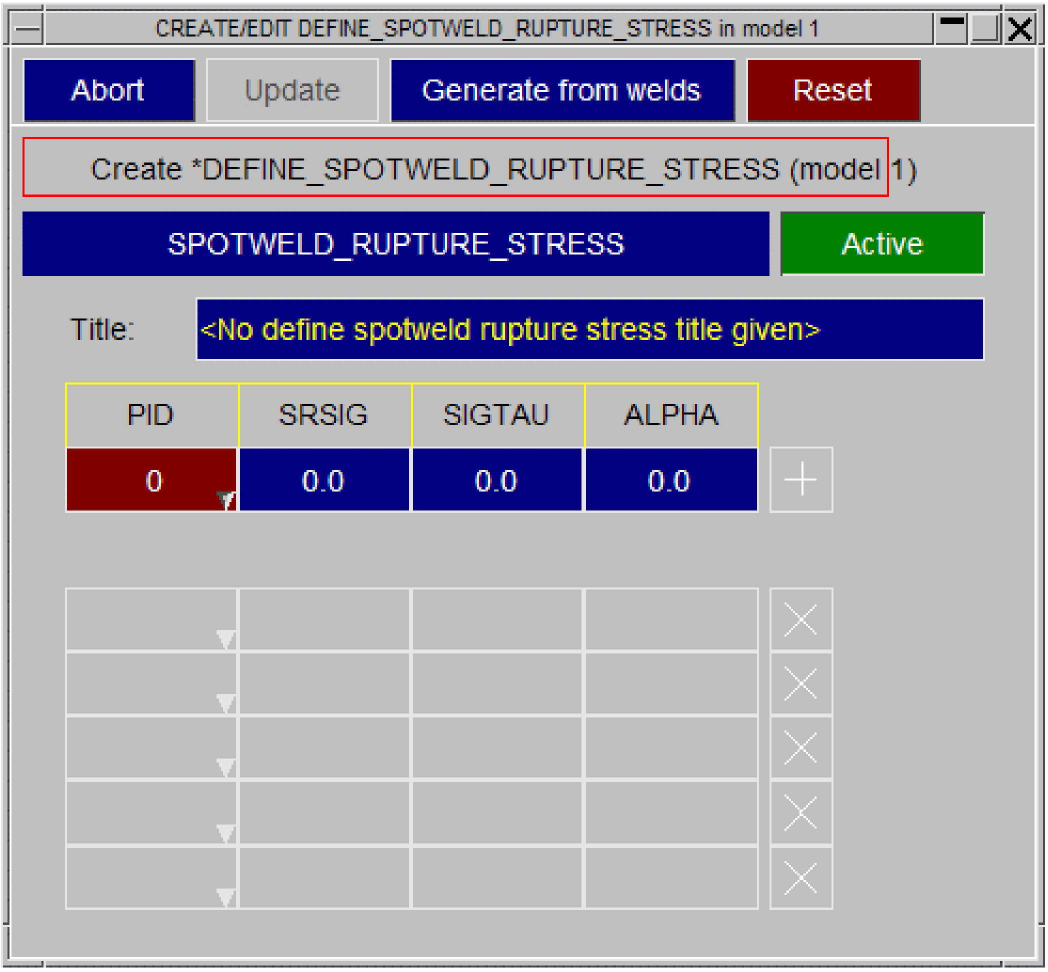
RENUMBER lets you change any or all hex spotweld assembly labels within a given model using the standard renumbering panel. To change the label of an individual hex spotweld assembly it may be simpler just to **MODIFY** it.

(DEFINE_) SPOTWELD_RUPTURE_STRESS:

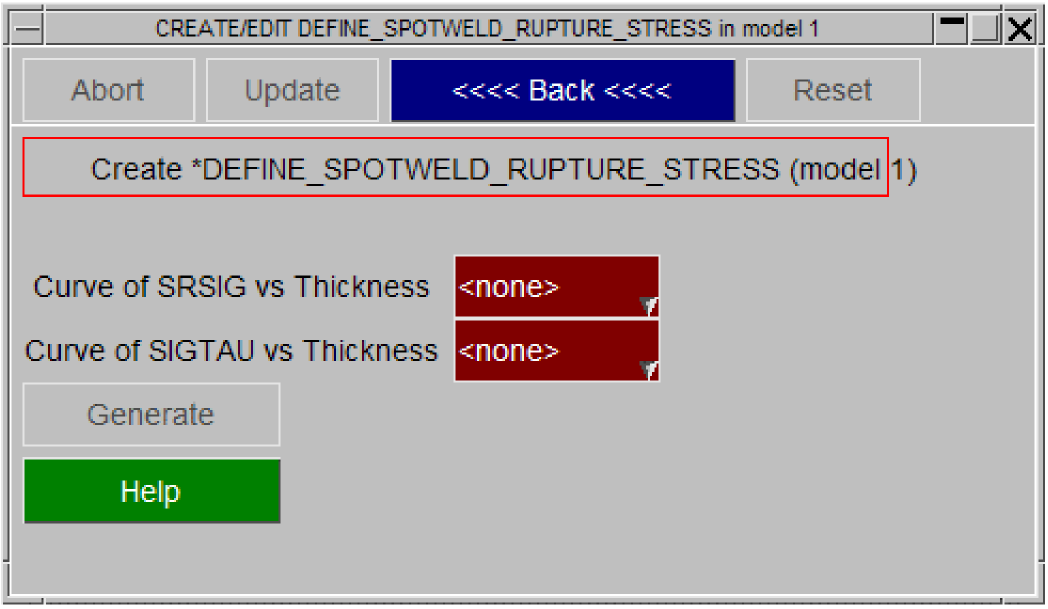


These types can be edited through the **create** and **modify** buttons shown above. There is only one occurance of the keyword in each model.

To deactivate the keyword from your model, click on the green **Active** button. Enter the relevant details in the card fields, then click on the **+** button to add a row, or the **X** button to remove a row.



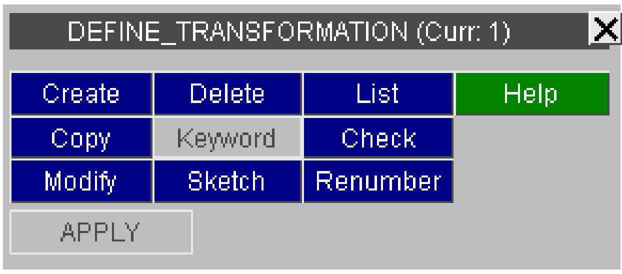
PRIMER has the ability to generate the rupture cards automatically. Click on the **Generate from welds** button and the following menu appears:



Enter a curve defining SRSIG vs thickness, and one defining SIGTAU vs thickness. PRIMER then looks at the thickness of each beam spotweld part in the model and enters the correct values in the keyword automatically.

(DEFINE_) TRANSFORM:

- Main Menu
- Creation
- Copying
- Editing
- Deletion
- List
- Check
- Renumber



This figure shows the main menu for the editing of co-ordinate systems.
All functions have their standard meanings as given in section 5.0.1:

CREATE Making a transformation definition.

See section 3.14.3. for more details about the **Create** /**Modify** panel for Transformations.

COPY Copy existing transformation(s) to make a new transformation.

The selected coordinates are copied. (Transformations do not "own" anything, so the concept of recursive copying does not apply.)

MODIFY Modifying the attributes of an existing transformation.

MODIFY functions in the same way as **CREATE**, except that an initial definition will be present.

Any modifications made to the section definition will not be made permanent until the **UPDATE_TRANSFORMATION** button is pressed. At this point a the local copy which has been updated is used to overwrite the version in the model.

DELETE Delete existing transformation definitions.

The selected transformations are deleted.

Transformation definitions do not "own" anything, so the concept of recursive deletion does not apply, however a transformation that is referred to (ie "owned") by some higher order item will not be deletable unless that item is deleted too, or its reference to the transformation removed.

SKETCH Sketch transformation summaries

The selected transformation definitions are sketched on the screen.

LIST List transformation summaries to screen

The selected transformation definitions are summarised on the screen.

CHECK Check definitions for errors

The selected transformation definitions are run through the standard checking routines.

RENUMBER Change transformation labels

Lets you change any or all transformation labels within a given model using the standard renumbering panel.

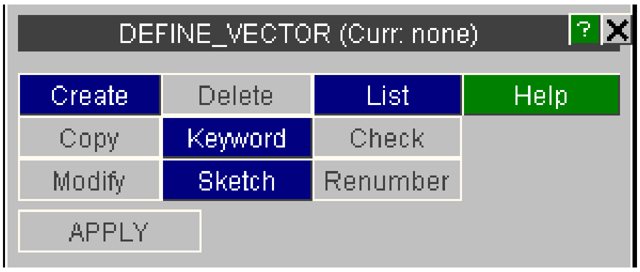
To change the label of an individual transformation may be simpler just to **MODIFY** it.

(DEFINE_) VECTOR:

These can be edited through their own specific editing panel (see below) and using the generic Keyword Editor.

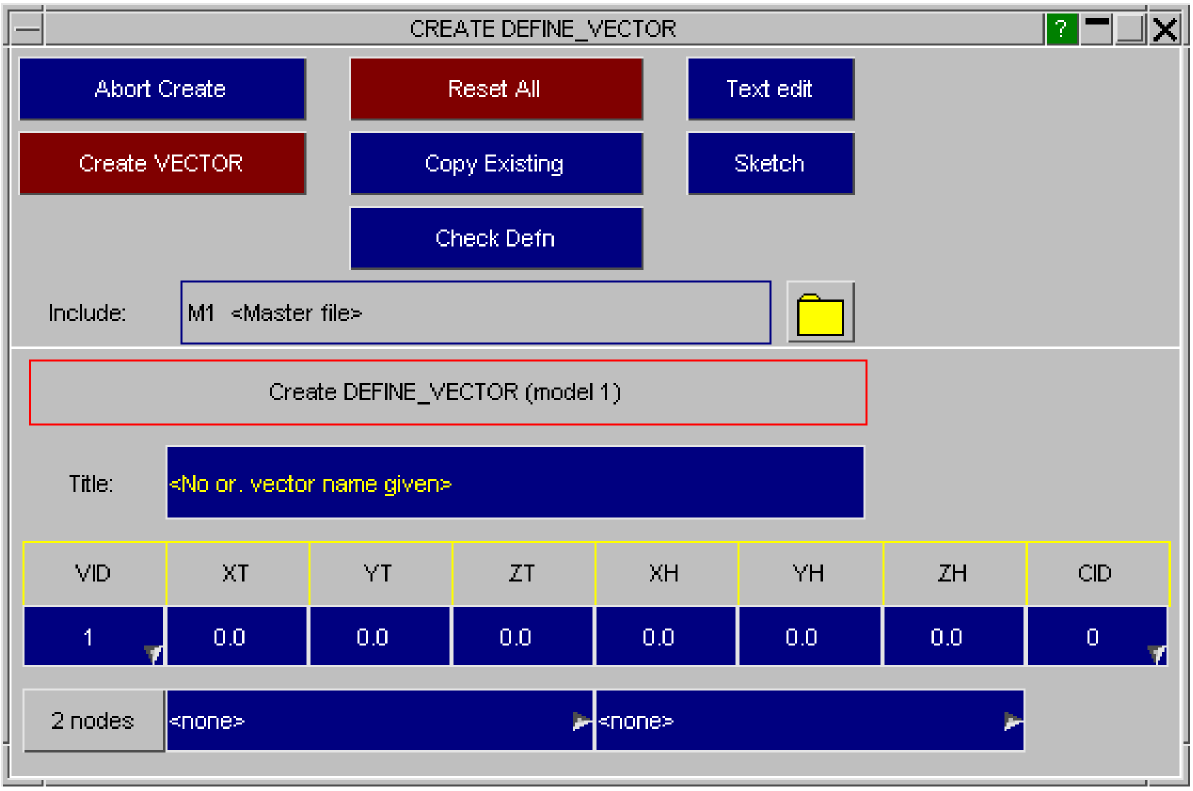
- Main Menu
- Creation
- Copying
- Editing
- Deletion

This figure shows the main menu for the editing of vector definitions.
All functions have their standard meanings as given in section 5.0.1



CREATE Making a new vector definition.

This shows the create/edit panel for vectors.



COPY Copy existing vector(s) to make a new vector(s).

The selected vectors are copied. (vectors do not "own" anything, so the concept of recursive copying does not apply.)

MODIFY Modifying the attributes of an existing vector.

MODIFY functions in the same way as **CREATE**, except that an initial definition will be present. Any modifications made to the vector definition will not be made permanent until the **UPDATE_VECTOR** button is pressed. At this point the local copy which has been updated is used to overwrite the version in the model.

DELETE Delete existing vector definitions.

The selected vectors are deleted.

Vectors do not "own" anything, so the concept of recursive deletion does not apply, however a vector that is referred to (ie "owned") by some higher order item will not be deletable unless that item is deleted too, or its reference to the vector removed.

SKETCH Sketch vector definitions.

SKETCH draws the vector on top of the current graphics image.

LIST List vector summaries to screen

The selected vectors are summarised on the screen.

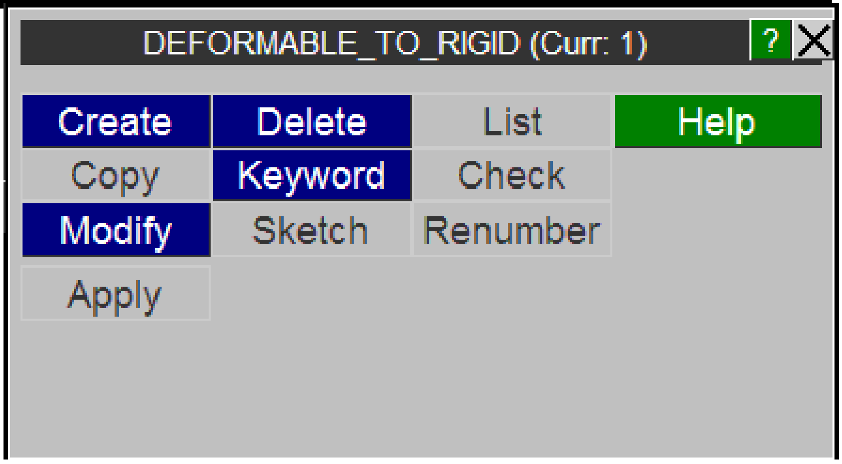
CHECK Check orientation vector definitions for errors

The selected vector definitions are run through the standard checking routines.

RENUMBER Change orientation vector labels

RENUMBER lets you change any or all vector labels within a given model using the standard renumbering panel. To change the label of an individual vector it may be simpler just to **MODIFY** it.

DEFORMABLE_TO_RIGID: Switching parts.



*DEFORMABLE_TO_RIGID and DEFORMABLE_TO_RIGID_INERTIA
DEFORMABLE_TO_RIGID_AUTOMATIC

This submenu panel is slightly different to the standard panels in PRIMER because there are 2 separate editing panels for DEFORMABLE_TO_RIGID - one for _AUTOMATIC, and one for _NONE and _INERTIA. To get to the _NONE and _INERTIA use the **keyword** button. To get to the _AUTOMATIC use the **create** and **modify** button.

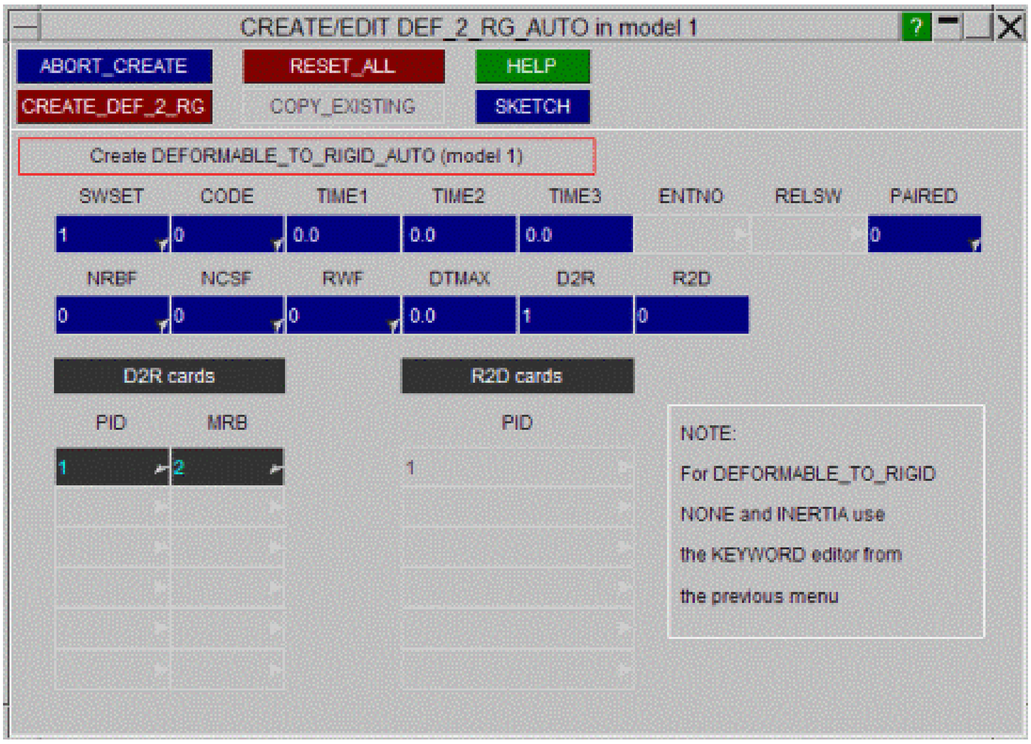
---

*DEFORMABLE_TO_RIGID and _INERTIA

This is a standard keyword panel, described in section_5a. Hit the radio buttons to switch between the two types.

*DEFORMABLE_TO_RIGID_AUTOMATIC

The _AUTOMATIC case has its own menu because it doesn’t fit into the standard keyword layout. Define the number of D2R and R2D conversions, and the rows beneath will become active, allowing the user to enter part ids.

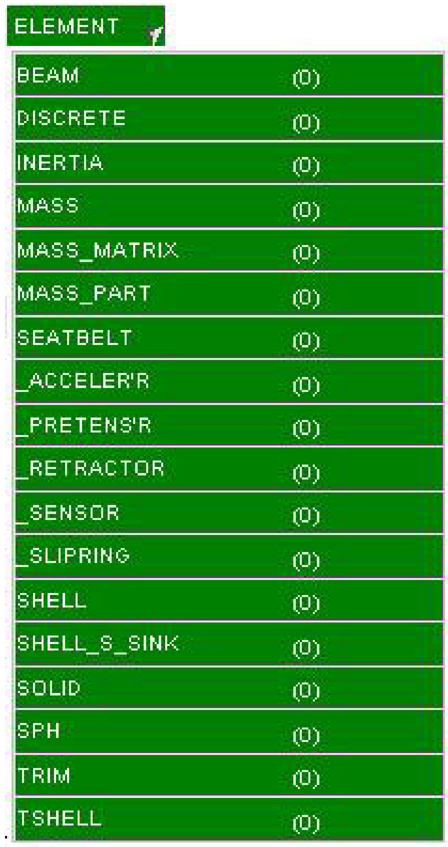

CREATE/EDIT DEF_2_RG_AUTO in model 1
ABORT_CREATE
RESET_ALL
HELP
CREATE_DEF_2_RG
COPY_EXISTING
SKETCH
Create DEFORMABLE_TO_RIGID_AUTO (model 1)
SWSET CODE TIME1 TIME2 TIME3 ENTNO RELSW PAIRED
1 0 0.0 0.0 0.0 0
NRBF NCSF RWF DTMAX D2R R2D
0 0 0 0.0 1 0
D2R cards
R2D cards
PID MRB
PID
1 2
1
NOTE:
For DEFORMABLE_TO_RIGID
NONE and INERTIA use
the KEYWORD editor from
the previous menu

ELEMENT: Defining Structural Elements.

LS-Dyna has 15 classes of structural element types, all of which are fully editable in PRIMER in individual create/edit panels. Generic keyword editing of all element types is also provided.
All element types except *TRIM are fully drawable, and there are a range of contouring options for different types of element data.

Each class of element has its own independent label sequence, thus it is legal to have shell #1 and solid #1, etc in the same model.

The elements menu enables you to create, modify and delete all the element types that are available in LS-Dyna.
This figure shows the main element menu.



The ***ELEMENT** keyword in LS-DYNA supports the following sub-types of structural element:

BEAM
DISCRETE
INERTIA
MASS
MASS_MATRIX
MASS_PART
SHELL
SHELL_SOURCE_SINK
SOLID
SPH
TSHELL
TRIM
SEATBELT
SEATBELT_ACCELEROMETER
SEATBELT_PRETENSIONER
SEATBELT_RETRACTOR
SEATBELT_SENSOR
SEATBELT_SLIPRING

All of the element types with the exception of ***ELEMENT_TRIM** can be created by quickly picking the nodes from the screen and setting the extra data. For all element types, when an element is created, default settings are saved so that the next element will use the same defaults. For example if you create a shell and select part 1000 for the PID, then the next element you create will automatically have the PID set to 1000. Obviously, you can change the part if needed but hopefully this will speed up creation of lots of elements in the same part.

A ‘quick create’ option is also available. Once the necessary information needed for the element is defined the element will automatically be created.

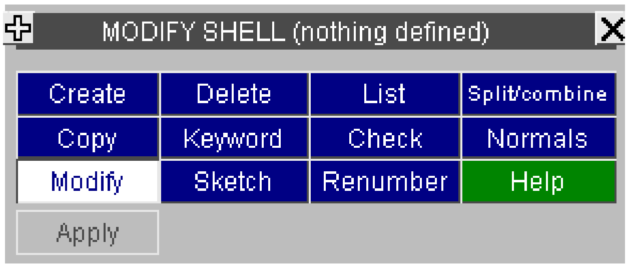
When an element is created the panel automatically remaps itself with the default values.

As the method for creating the different element types is very similar the generic method will be described in detail for shells. Any major differences in other element types will be stated

ELEMENT_SHELL

This figure shows the main element shell menu. The functions currently available have their standard meanings. (See section 5.0.1)

As with all classes of element the Generic Keyword editor may be used instead.



CREATE Making a new shell.

This figure shows the initial state of the element shell creation panel: no part has been given yet, so it is highlighted red.

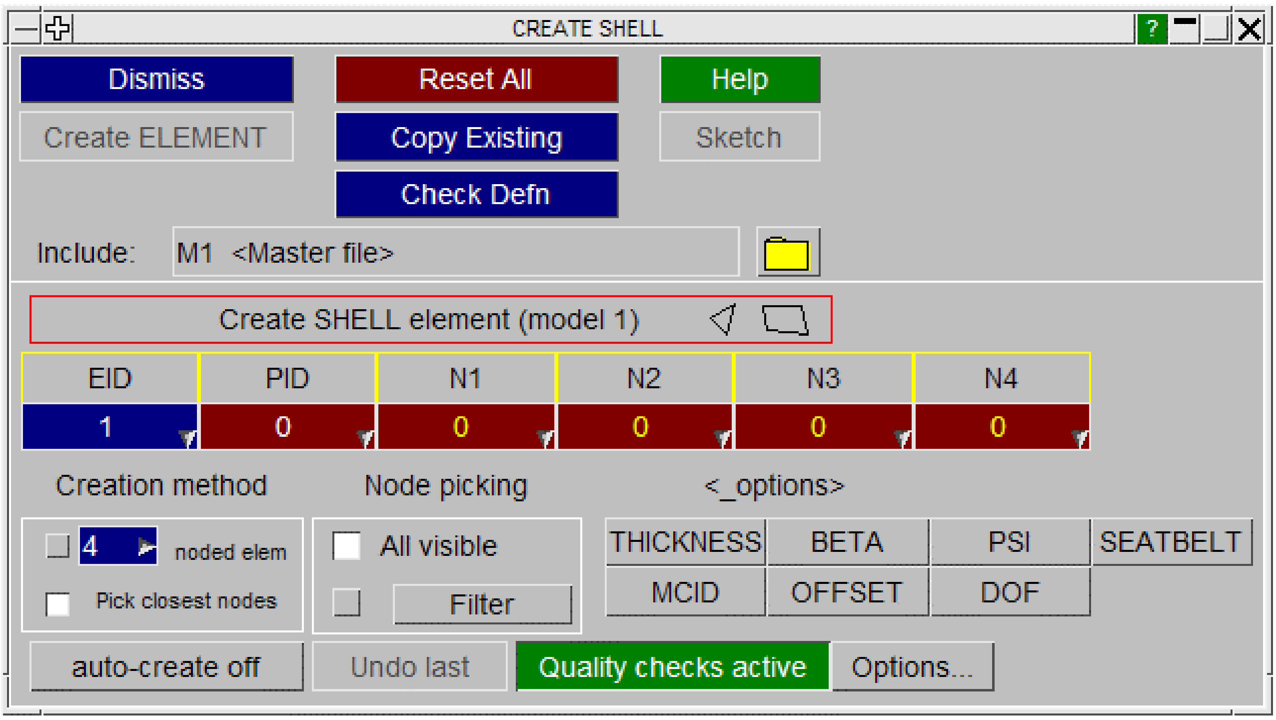
The **<_option>** buttons can be used to select among sub-keyword suffices:

ELEMENT_SHELL
ELEMENT_SHELL_THICKNESS or
ELEMENT_SHELL_BETA
etc

The **SEATBELT** option is a special case: see below.

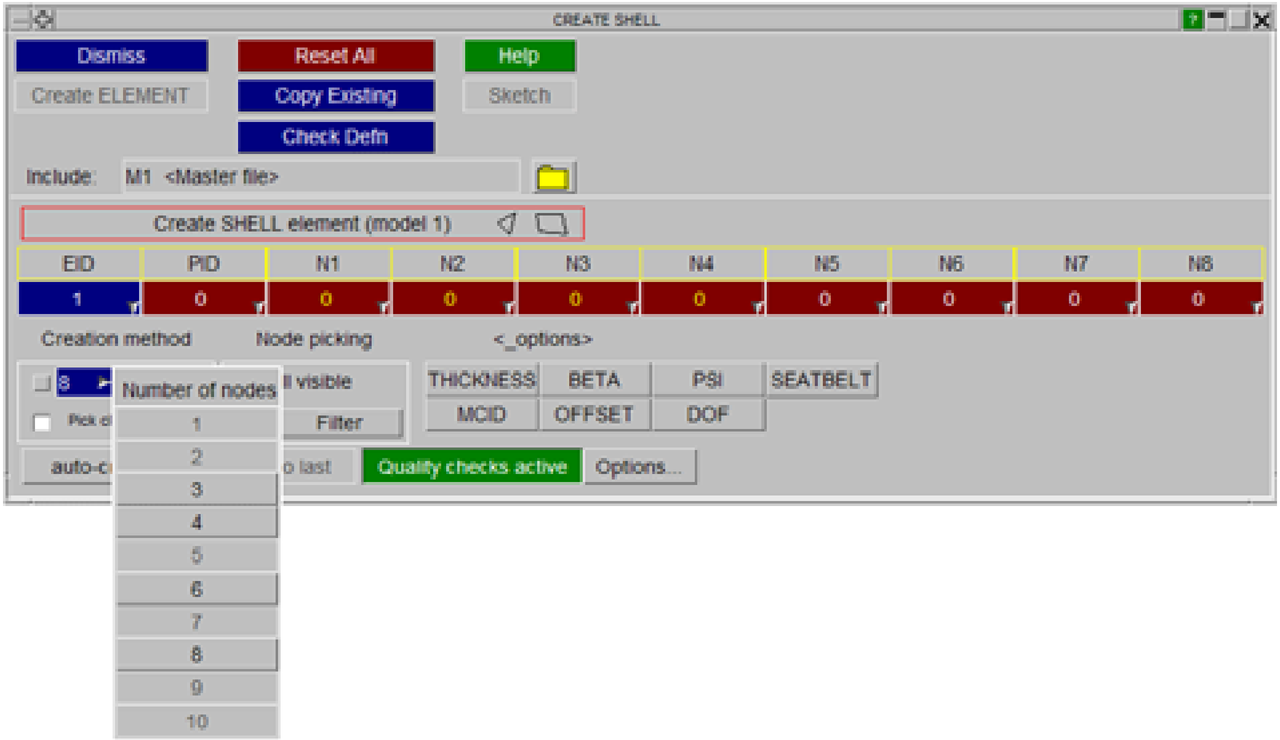
The part and the node numbers can be typed directly into the text boxes. The default element label used is the highest node label in the model + 1. This can be changed if needed. Alternatively, the popup menus can be used to pick a part, and the nodes from the screen, or to select a part, or node from a list.

To choose creation of tria or quad elements the nodes popup can be used to select the number of nodes. For a shell only 3 or 4 nodes can be chosen.



6 and 8 noded (parabolic) shell elements

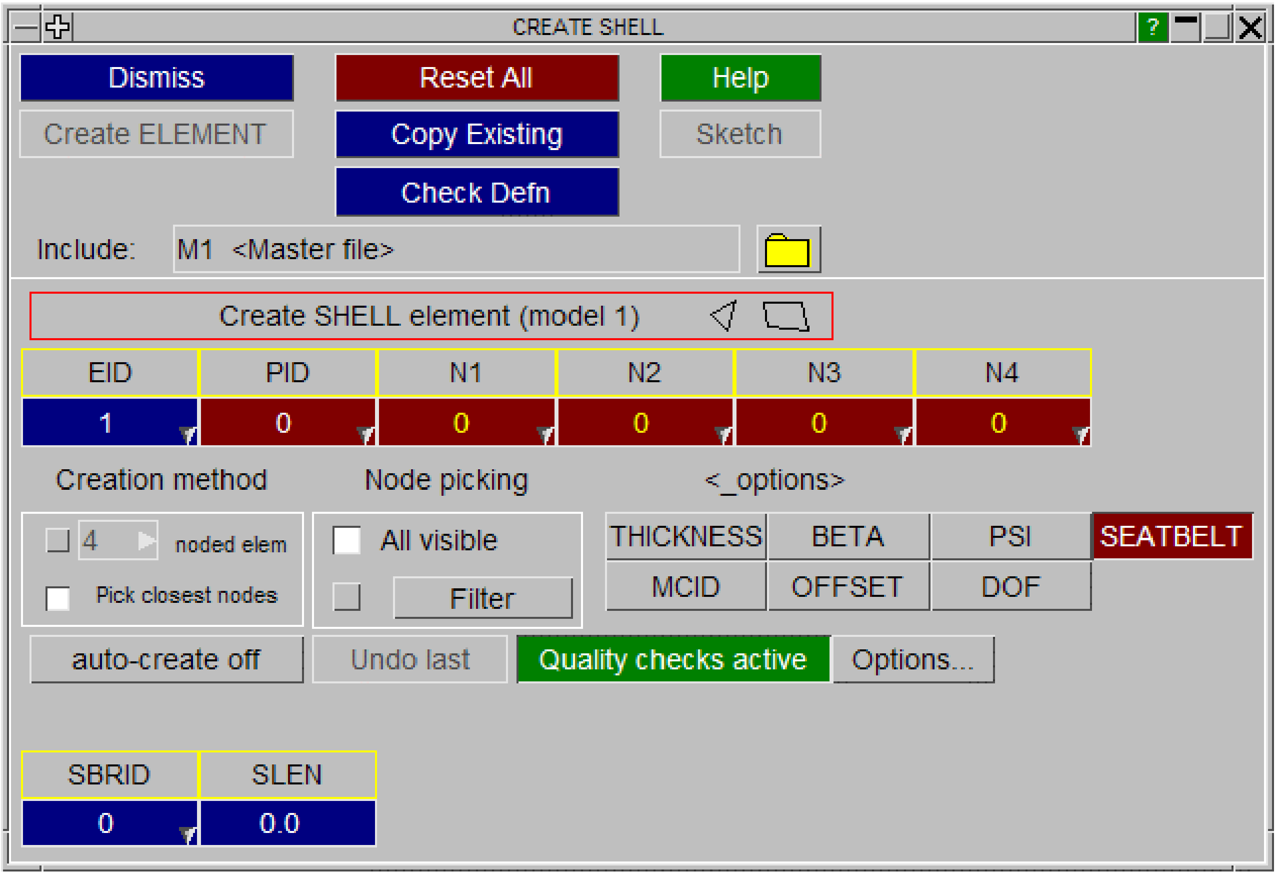
To edit parabolic shells used the "Pick >" popup to select the number of nodes on the element. The example below shows the 8 noded case:



The **SEATBELT** option: creating 4 noded seatbelt shells

In LS-DYNA release 971 4 noded "shell" seatbelt elements are introduced under the *ELEMENT_SEATBELT keyword. However these are in fact shell elements, sharing the same numbering scheme as shells, so for consistency within PRIMER they are edited under the SHELL keyword.

These elements should belong to shell parts, referencing *SECTION_SHELL cards, however they can use the *MAT_SEATBELT material definition.



The quickest method for creating a shell is to use one of the 'quick creation' methods:

Pick closest nodes

This is the default method for creating shells. When you pick a point on the screen the 4 (or 3 if you are creating a tria) closest nodes to the point are automatically selected. The order of the nodes will be automatically calculated for you so there is no danger of creating a shell with a negative area. The shell that will be created is sketched on the screen **but will not be created yet**. Picking a second point on the screen will update the display with the 4 closest nodes to that point. You can carry on picking a point until you have the nodes you require. When all nodes and the part are filled in the **SKETCH** and **CREATE_ELEMENT** buttons will be ungreyed.

By default all visible nodes in the model can be used for this method. The filtering option under 'Node picking' allows you to limit the nodes which can be used. If for example you only wanted to use nodes which are on part 1000 this option can be used.

Individual nodes can still be edited by either typing in a new value or using the popups.

Pick individual nodes

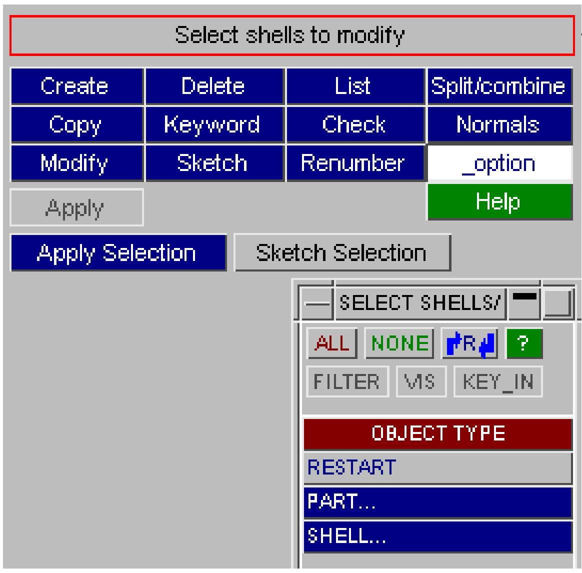
In this mode the nodes are picked from the screen in the order 1, 2, 3, 4, 1, 2… The node which will be picked is indicated by the colour of the node text in the panel. The node which will currently be picked has yellow text. All the others will have white text. You can also edit individual nodes by either typing in a new value or using the popups. With this method the filtering option for node picking is not available.

Auto create

Instead of having to press **CREATE_ELEMENT** each time you want to create a shell the **AUTO_CREATE** option can be used. When this option is set, as soon as the required data is set the element is created. Using this in conjunction with 'pick closest nodes' enables creation of a shell with a single click of the mouse.

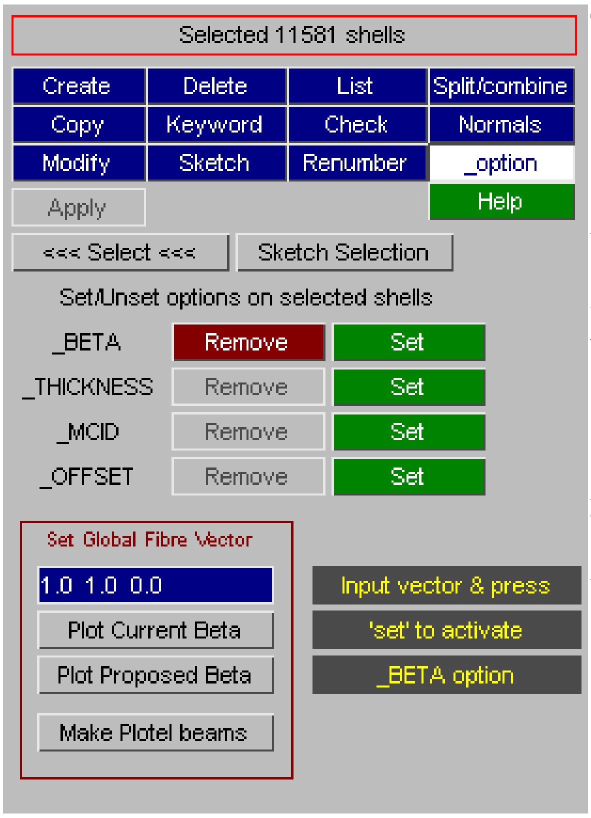
Once the element has been created the **UNDO LAST** button will be made live. This can be used to delete the element if is not what you wanted.

Shell options



_BETA _MCID _THICKNESS and **_OFFSET**

Options for *ELEMENT_SHELL can be set on shells selected directly or by part by using the **_option** function.



The option is set by pressing **Set** or removed by pressing **Remove**

Plot Current Beta shows the current beta angles of any shell_beta amongst the selection

Plot Proposed Beta shows prospective beta angles corresponding to the global input vector. Any shells onto which the vector does not project properly will be highlighted with the (dubious) angle line plotted in red and the user warned.

Use **Set** to set proposed beta angle on all selected shells. If the vector is [0,0,0] the beta angles will be set to zero. The effective angle may be viewed by pressing **Plot N1->N2 beams**.

Make Plotel creates plotel elements to show all current beta angles. These may be removed later by using ELEMENT->BEAM->DELETE->DELETE ALL PLOTEL if you wish.

For other options, the option alone is activated and you need to use the shell Keyword editor to set the values.

Other shell creation commands.

DISMISS	Aborts the current definition and returns to the main element shell menu.
RESET_ALL	Resets all attributes to <null> for this definition: all data entered will be lost, and the panel will return to its initial default state.
COPY_EXISTING	Copies the attributes of an existing shell definition (in the current model). This may then be modified as required.

SKETCH	Sketches the current definition on top of the current image.
LIST_XREFS	Lists everything that references the current shell definition.
CHECK	Performs a check of the current definition, listing any errors

CREATE_ELEMENT Saving the element definition.

Once you have entered the minimum amount of data required to define this shell the **CREATE_ELEMENT** button will be made live, and this permits you to save this definition. (If it is not live the missing fields will be highlighted in red.) The definition will be checked and any errors listed, an then it will saved permanently in this model.

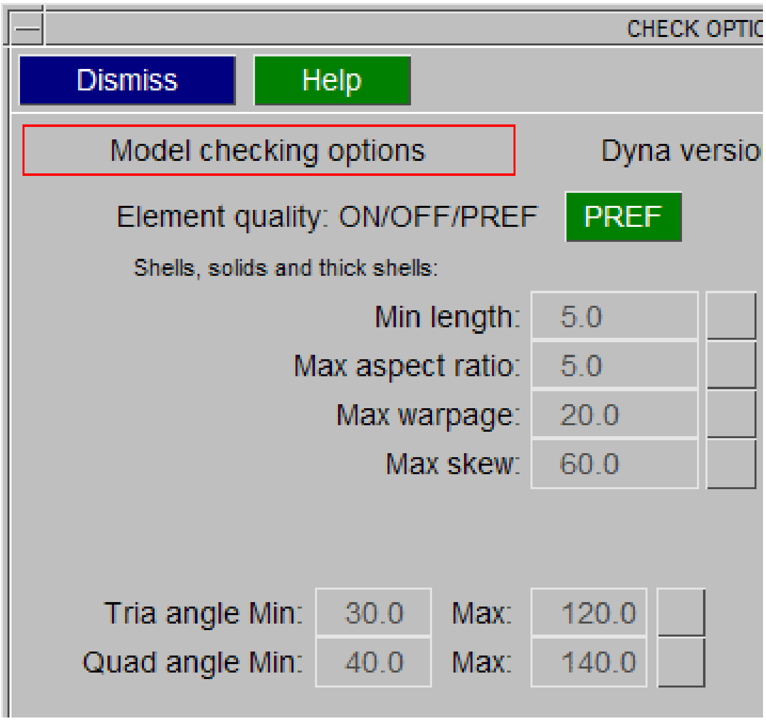
Until you press this, the definition remains volatile, and will be lost if you exit this panel in any other way.

When the shell element is created the part number and the number of nodes are saved as the defaults. When the panel refreshes for you to create another shell these defaults are automatically used to speed up element creation.

Quality checks

By default quality checks are done on shell elements when they are created to ensure that there are no badly defined elements. If you want to bypass the quality checks then they can be turned off by using the **QUALITY CHECKS** button. It is recommended that you keep the quality checks on.

The values that are used for the quality checks can be changed by using the **OPTIONS** button next to quality checks. This brings up the main check options panel.



Checks are done for:

- Element length
- Warpage
- Aspect ratio
- Skew
- Minimum and maximum internal angles

The values used for checking can easily be changed.

COPY Copying existing shell(s) to make a new one(s).

You can **COPY** any number of shells, in multiple models.

When **APPLY** is pressed you are asked to confirm what is to be copied, and then the operation is carried out.

For each model the <n> shells chosen in that model are copied using labels <previous highest + 1> to <previous highest +n>, there is currently no control available over the new labels assigned.

MODIFY Modifying the attributes of an existing shell.

This functions in exactly the same way as **CREATE**, using the same panels as in figure Elem_3. The only difference is that the initial state of the panels is already set with the attributes of the shell to be modified.

DELETE Deleting existing elements

The **DELETE** operation works exactly the same way as **COPY** described above, except that the chosen elements are deleted.

- If **DELETE_RECURSIVE** is switched on any nodes, restraints and loads on the elements to be deleted are

marked for deletion.
- If recursive deletion is not used only the elements themselves are removed.

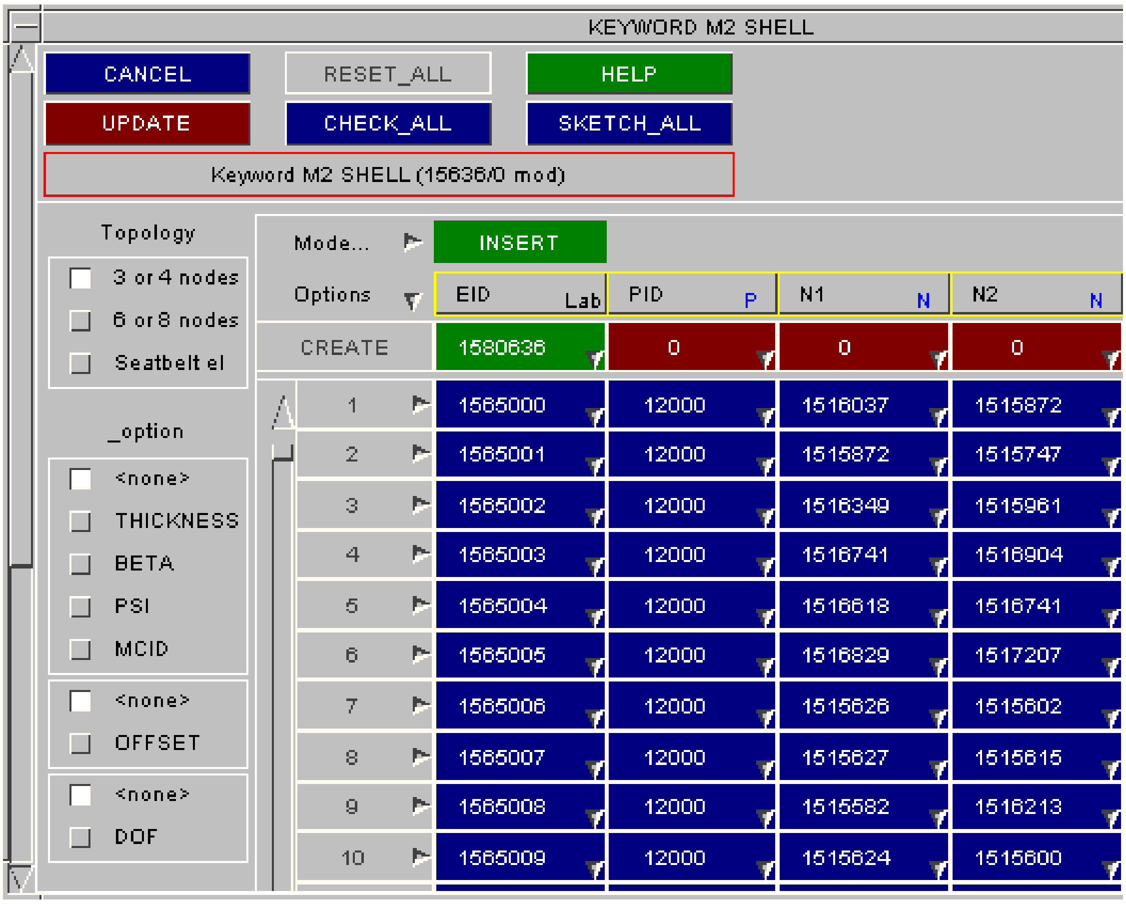
Note also that the standard deletion rules described in Section 6.4.1 still apply: nodes, loads, restraints, etc will only be deleted if nothing else (which is to remain) depends on them.

KEYWORD Generic keyword editor

KEYWORD starts the generic keyword editor which allows creation, deleting and modification of multiple shells. This is useful for modifying multiple shells in a single operation.

This example shows the **SHELL** keyword editor.

Note that the **_THICKNESS, _BETA, _PSI,** etc options require separate layouts since they have a different number of rows of data.



SKETCH Sketch the chosen shell on the current image

SKETCH allows the user to select and sketch individual shells on the current graphics image.

CHECK

Runs the standard checking function on the selected shells. Each shell will be listed either as "OK", or a summary of the errors encountered will be printed. (This is the same as the **CHECK_DEFN** command during shell editing.)

LIST

Writes a summary list of the selected shells to the screen.

RENUMBER

Raises the standard renumbering panel for shells in the chosen model, allowing you to renumber some or all of them.

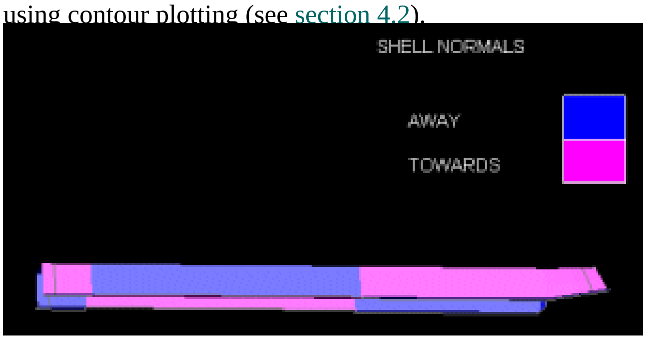
How to move shells from one part to another:

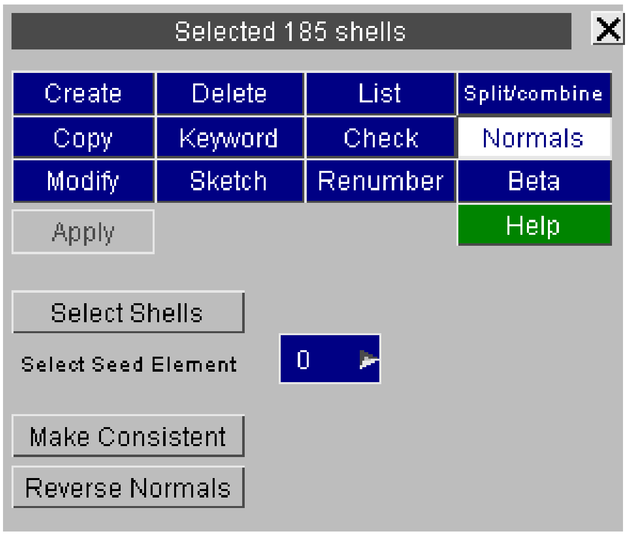
- For a single element, use the **EDIT** panel.
- Otherwise, use the Keyword editor in EDIT mode, select the shells, ill in the new part ID instead of * and press

Apply.
- Or, select **PART** (in Keywords) > **MODIFY** (select the new part) > **CONTENTS...** > **ADD_ITEMS** and add the shells.

NORMALS

The direction of shell normals can be showed in PRIMER using contour plotting (see section 4.2).

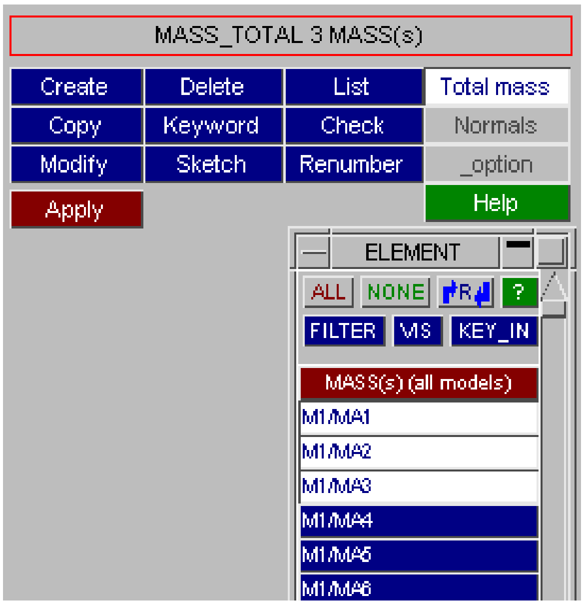




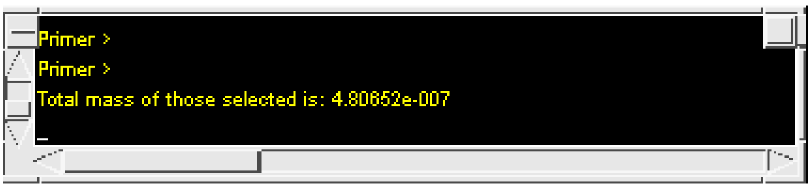
The direction of normals is indicated by the colour the shell is plotted as. Blue represents the normal heading away, pink towards.

The shell normals menu is shown on the right. **Select Shells** invokes an object menu whereby the shells can be selected. The selection can be sketched or **Apply Selection** can be used to return to the initial menu. The other options will now be "live" instead og greyed out. It is now possible to either reverse all the shell normals in the selection simply by clicking **Reverse Normals**. The other option is to make all the normals consistent with a selected one, the **Seed Element**.

SUM OF SELECTED MASSES



- The ELEMENT_MASS panel offers a function to sum the mass of mass elements selected from the object menu.



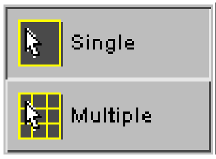
- The total mass of the selected mass elements is reported in the dialogue box when you press **APPLY**.

SPLIT/COMBINE

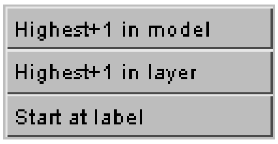
The split/combine panel allows you to manipulate shell elements. You can:

- Split shells using predefined patterns
- Split shells by drawing a line
- Find warped quads and split into 2 trias.
- Fix transitions between adjacent shells
- Detach shells from a mesh
- Combine shells together into one shell

To change the mode use the popup on the top left of the panel. The options will then change accordingly.

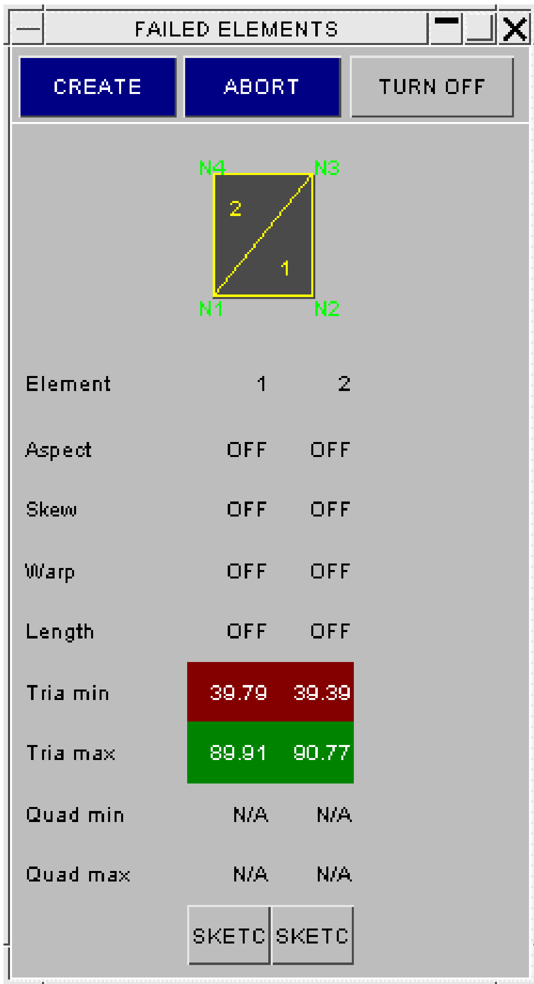


Some modes allow you to work on a single shell or multiple shells. The default, is **Single** mode. In this mode ’quick picking’ is activend. Alternatively, to operate on many shells at the same time, select **Multiple** mode. The standard object menu is mapped to allow you to choose the shells you want to modify. Press **APPLY** to change them.



You can choose what labels to use for any new nodes and shells that are created. Use the popup to select which option you require.

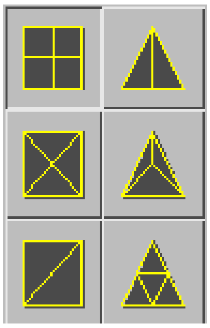
If you choose **Start at label** then give a label number to start from. PRIMER will try to use that number. If a node or beam already exists with that label it will revert to **Highest+1 in model**.



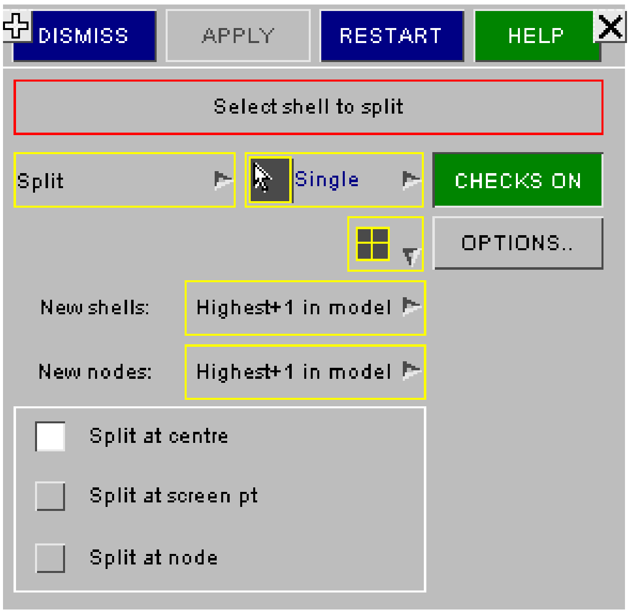
The **checks** button allows you to trap creating elements which do not pass specific quality checks. With **CHECKS ON** this checking is done. The values and types of checks done can be cahnged with the **OPTIONS...** button.

In the example on the right a quad is being split into 2 trias. However the minimum angle for the tria (38.66) is less than the allowed angle. PRIMER is warning you of this. **CREATE** will force the creation of the shell, **ABORT** will stop this operation. **TURN OFF** will turn the checks off. This is useful if you are splitting lots of shells.

Predefined split patterns



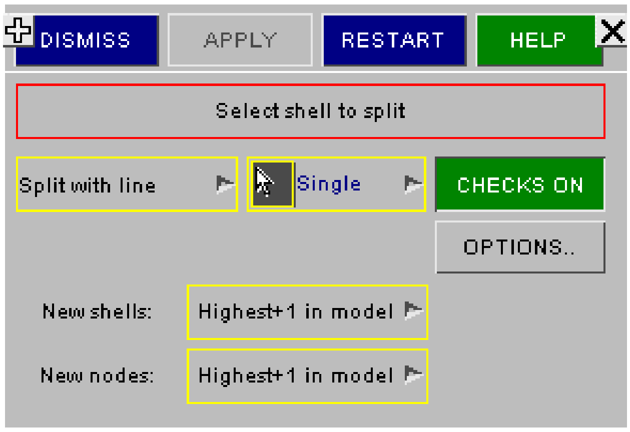
There are several predefined split patterns. To change the pattern use the popup.



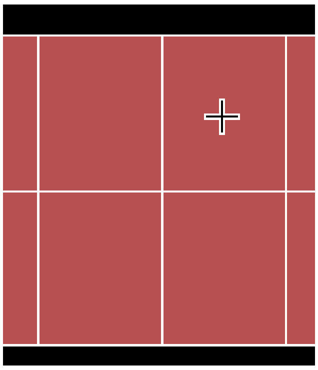
In **Single** mode just click on a shell to split it. You can split the shell at the centre of the shell, at the point you click on the screen (projected onto the shell) or at an existing node location.

In **Multiple** mode select the shells you want to split using the object menu and press **APPLY**.

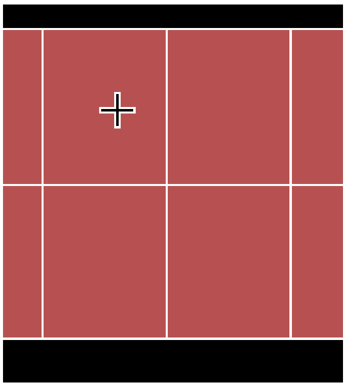
Splitting by line



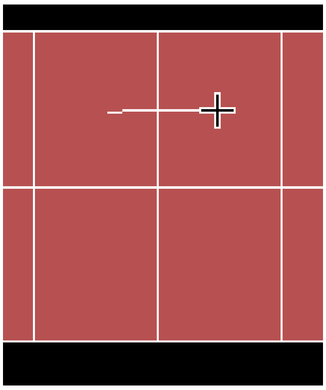
Single mode



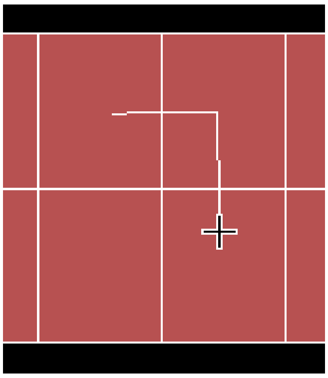
1. Select the shell to split



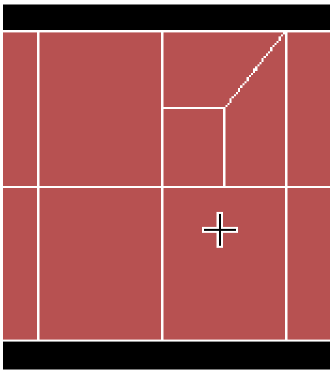
2. Click the first point on the line



3. Click the seond point on the line

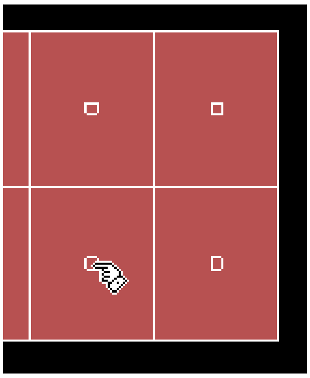


4. Click the third point on the line (if required)

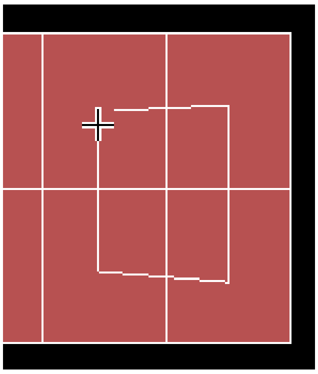


5. The shell is split

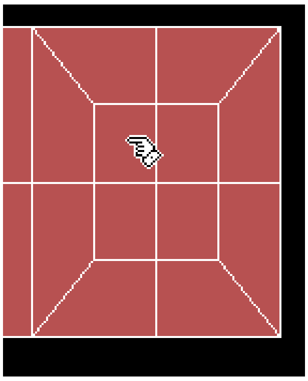
Multiple mode



1. Select the shells to split and press **DRAW LINE**.



2. Draw the line by clicking with the mouse.



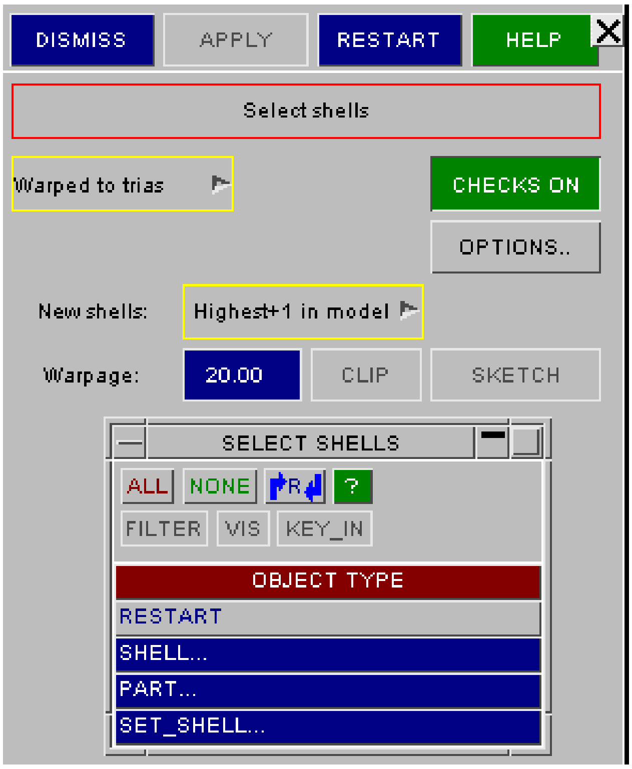
3. Press **APPLY**. The shells are split

Splitting warped quads

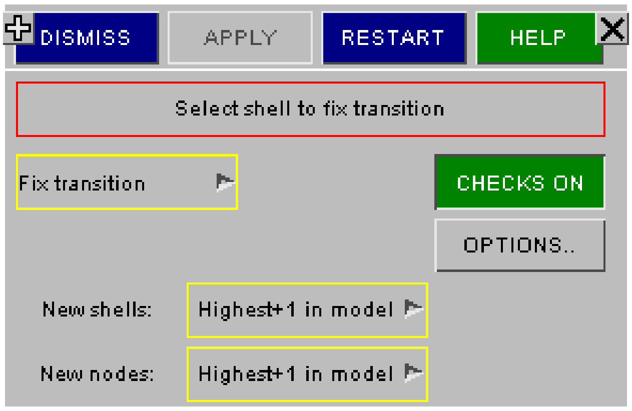
Select the shells you want to check/split by using the object menu. Give a value for the maximum warpage.

You can sketch the shells that are warped by pressing the **SKETCH** button. They can be placed on the clipboard by pressing **CLIP**.

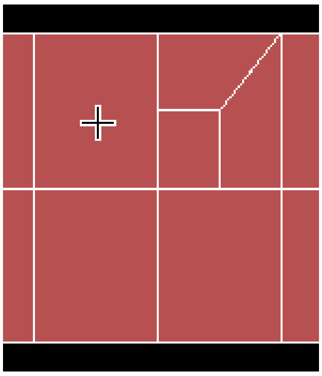
To split the shells into trias press **APPLY**.



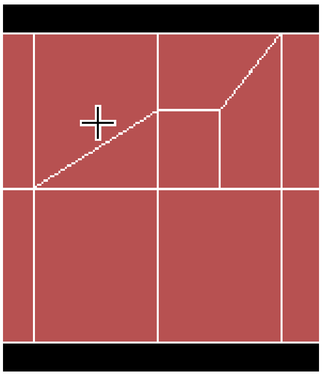
Fixing mesh-transitions



Fix transitions looks at neighbouring elements to see if the mesh is continuous. If it is not, the element is split to make it continuous.



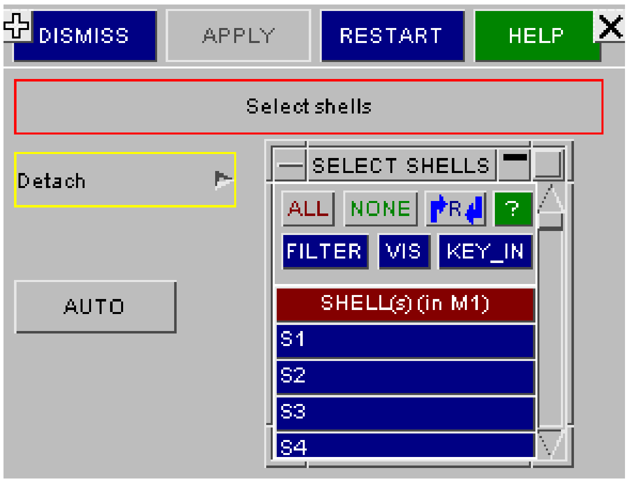
1. Click on the shell you want to split.



2. The shell is split to make a continuous mesh.

Detaching shells

Combining shells



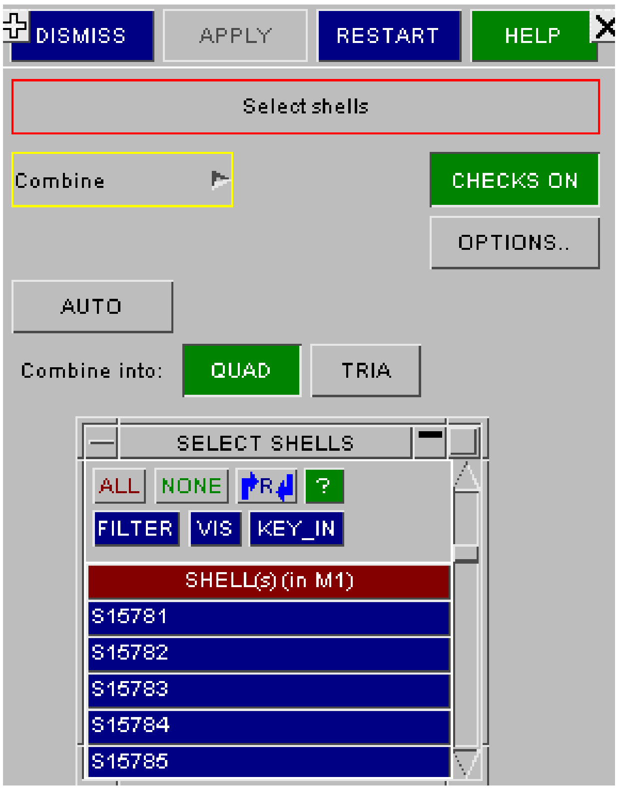
To detach one or more shells from a mesh use the **Detach** function. Select the shells you want to detach by either clicking on the screen or using the object menu.

If **AUTO** is enabled, once you have the correct number of shells they will be detached, otherwise press **APPLY** to detach them.

To combine two or more shells together use the **Combine** function. Select the shells you want to combine by either clicking on the screen or using the object menu.

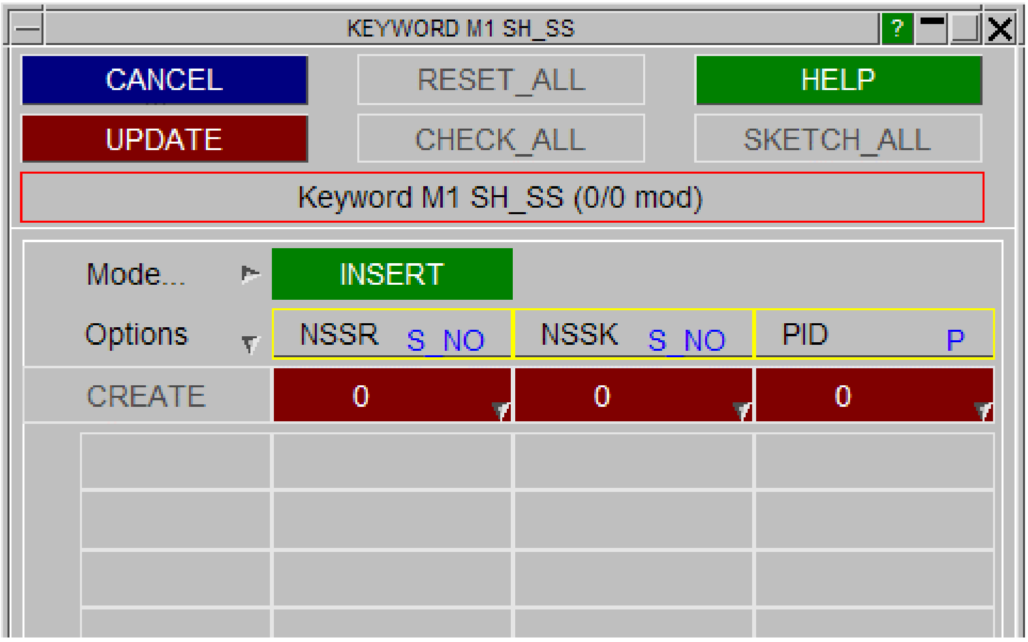
If **AUTO** is enabled, once you have the correct number of shells they will be combined, otherwise press **APPLY** to detach them.

The **QUAD** and **TRIA** buttons can be used to choose what to combine the shells into.



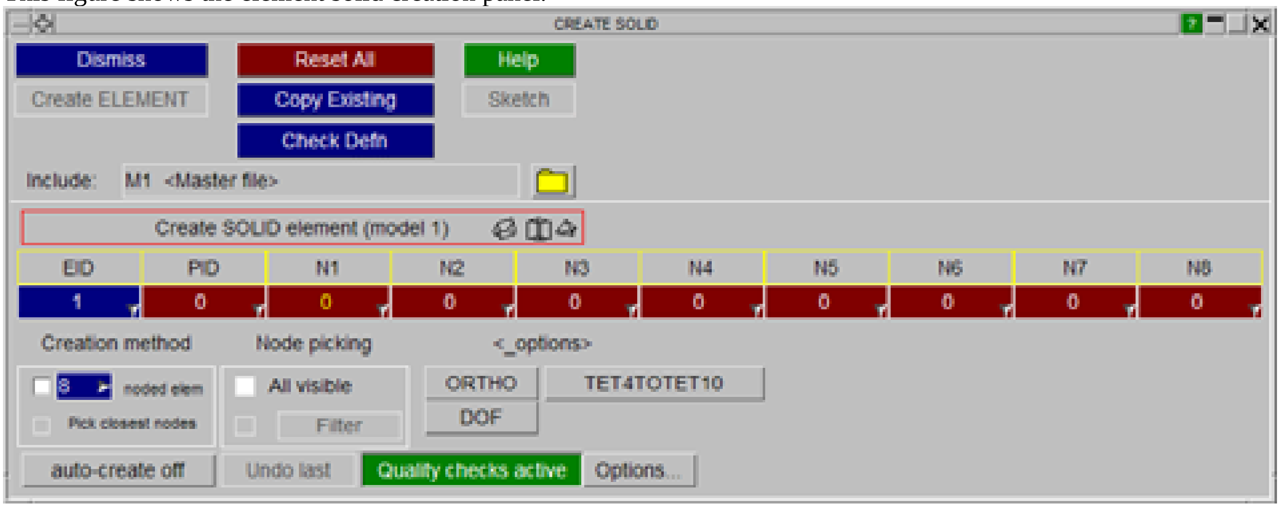
ELEMENT_SHELL_SOURCE_SINK

This figure shows the shell source sink keyword editing panel.
This keyword defines a strip of shell elements of a single part ID to simulate a continuous forming operation



ELEMENT_SOLID

This figure shows the element solid creation panel.



The **ORTHO** <_option> button can be used to change whether an **ELEMENT_SOLID** or an `ELEMENT_SOLID_ORTHO` is created.

Tetra, penta and hexa solid elements can be created by changing the number of nodes by typing in the number of nodes or using the popup.

Alternatively, if the number of nodes is left set at eight and the nomal LS-Dyna method for creating either penta elements (N1, N2, N3, N4, N5, N5, N6, N6) is used a penta element will be created, or tetra elements (N1, N2, N3, N4, N4, N4, N4, N4) a tetra element will be created.

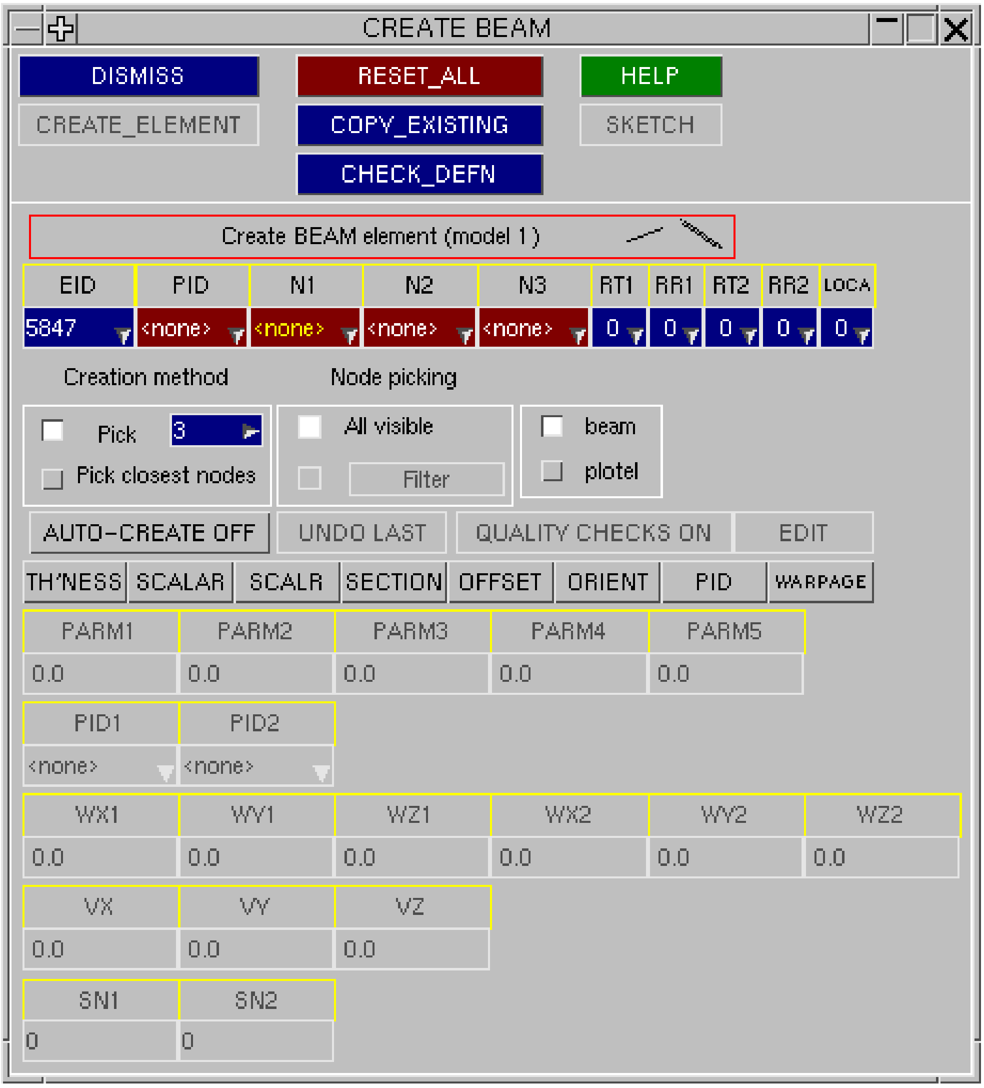
The '**Pick closest nodes**' and '**Node picking**' options are not available for solid elements.

When a solid element is created the part number and the number of nodes are remembered as defaults for the next element.

ELEMENT_BEAM

CREATE

This figure shows the element beam creation panel.



There is an option to create a **plotel** beam which is a 2 noded beam used for display purposes only.

The row of buttons **THICKNESS, SCALAR, SCALR, SECTION, OFFSET, ORIENT, PID, WARPAGE** can be used to select beam options, with proviso that THICKNESS and SCALAR are exclusive.

ELEMENT_BEAM
ELEMENT_BEAM_option1_option2

Three, two (discrete and spotweld), and one (spotweld) beam elements can be created by changing the number of nodes by typing in the number of nodes or using the popup.

There are no element quality checks available for beam elements at present.

When a beam element is created the part number and the number of nodes are remembered as defaults for the next element.

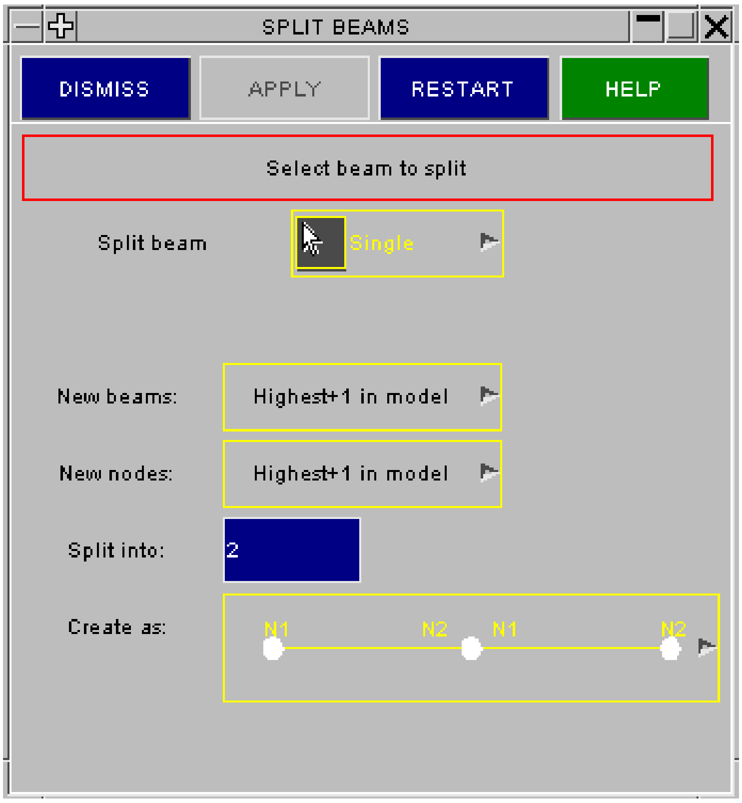
Additionally if a three noded beam is created the third node is also remembered as a default.

SPLIT

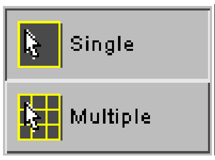
The SPLIT panel allows you to split beams into 2 or more beams.

- Any beams which are created will have the same 3rd node as the original beam.
- Release conditions on the original beam nodes will be retained.
- Thickness parameters for beams will be correctly calculated (e.g. if the original beam is tapered)
- If the beam is in a set, the new beams will automatically be added to the set.

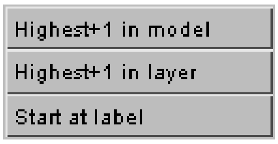
If the beam has a segment (2 noded)on it or a ***DATABASE_HISTORY_BEAM**, ***LOAD_BEAM** or ***INITIAL_STRESS_BEAM** card on it the beam cannot be split.



By default, 'quick picking' is activated in **Single** mode, and each beam you pick will be split. Alternatively, to split many beams at the same time, select **Multiple** mode. The standard object menu is mapped to allow you to choose the beams you want to split. Press **APPLY** to split them.

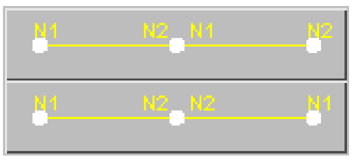


You can choose what labels to use for any new nodes and beams that are created. Use the popup to select which option you require.



If you choose **Start at label** then give a label number to start from. PRIMER will try to use that number. If a node or beam already exists with that label it will revert to **Highest+1 in model**.

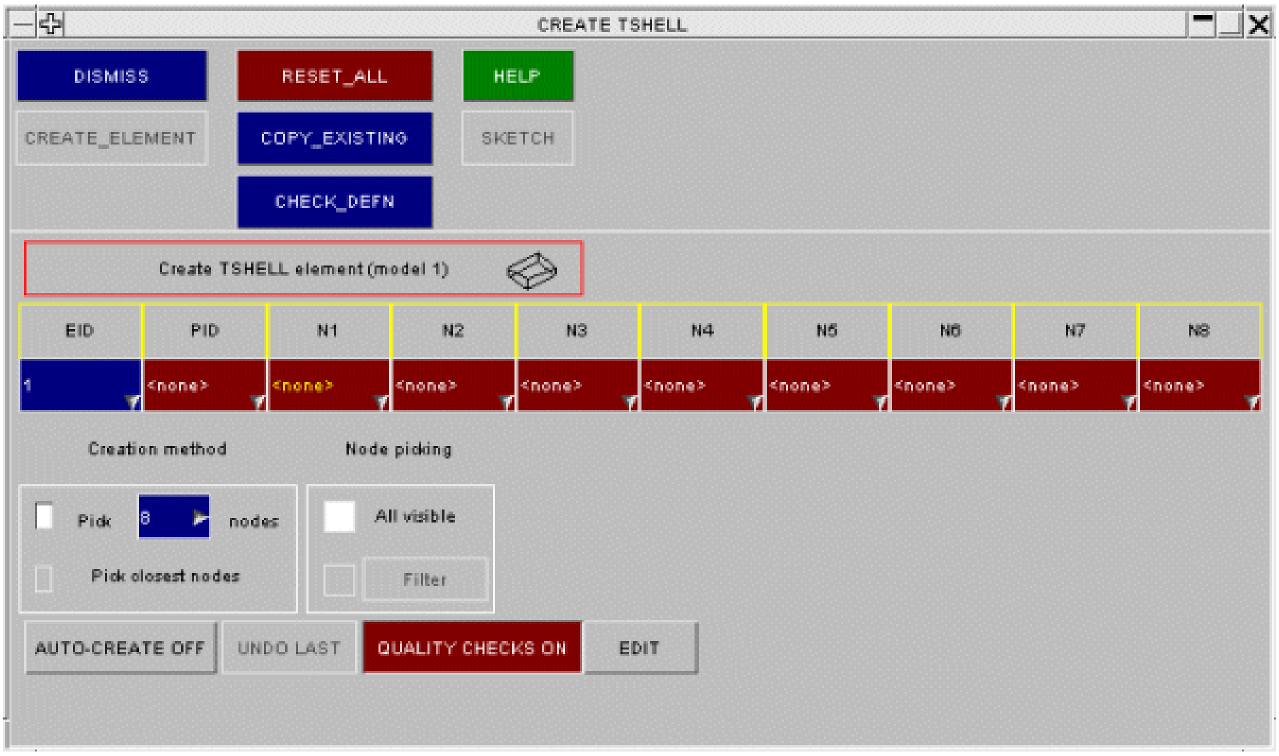
If you split a beam into two, you have the option of choosing the direction of the second beam. This can be useful if using the beam with ***MAT_SEISMIC_BEAM** as a plastic hinge can only be formed at one end of the beam. In this case N1-N2,N2-N1 should be used. For other analyses N1-N2,N1-N2 should be used so that forces and moments are plotted correctly in post-processing.



ELEMENT_TSHELL

This figure shows the thick shell element creation panel.
The thick shell creation method is identical to the solid element method except that:

- Only triangular and quadrilateral elements are allowed.
- There is no **ORTHO** option



ELEMENT_DISCRETE

This figure shows the element discrete creation panel.
Two and one (grounded) discrete elements can be created by changing the number of nodes by typing in the number of nodes or using the popup.

There are no element quality checks available for discrete elements at present.

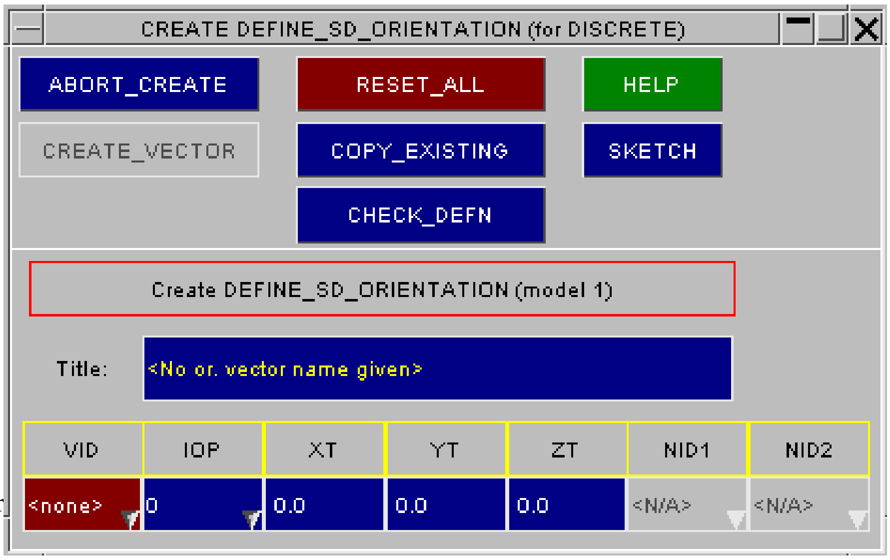
When a discrete element is created the part number and the number of nodes are remembered as defaults for the next element.

If an orientation vector (VID) is required:

If the discrete element needs to use an orientation vector then one can be created/ edited by using the **VID** popup.

The panel that the create option brings up can be seen in figure Elem_9.

In this panel the label can be defined as usual by typing in the number or using the popup. The type of orientation vector can be set by using the **IOP** option and either the vector set using the **XT**, **YT** and **ZT** fields or the 2 nodes defined by typing in or using the popups for the **NID1** and **NID2** fields



ELEMENT_INERTIA

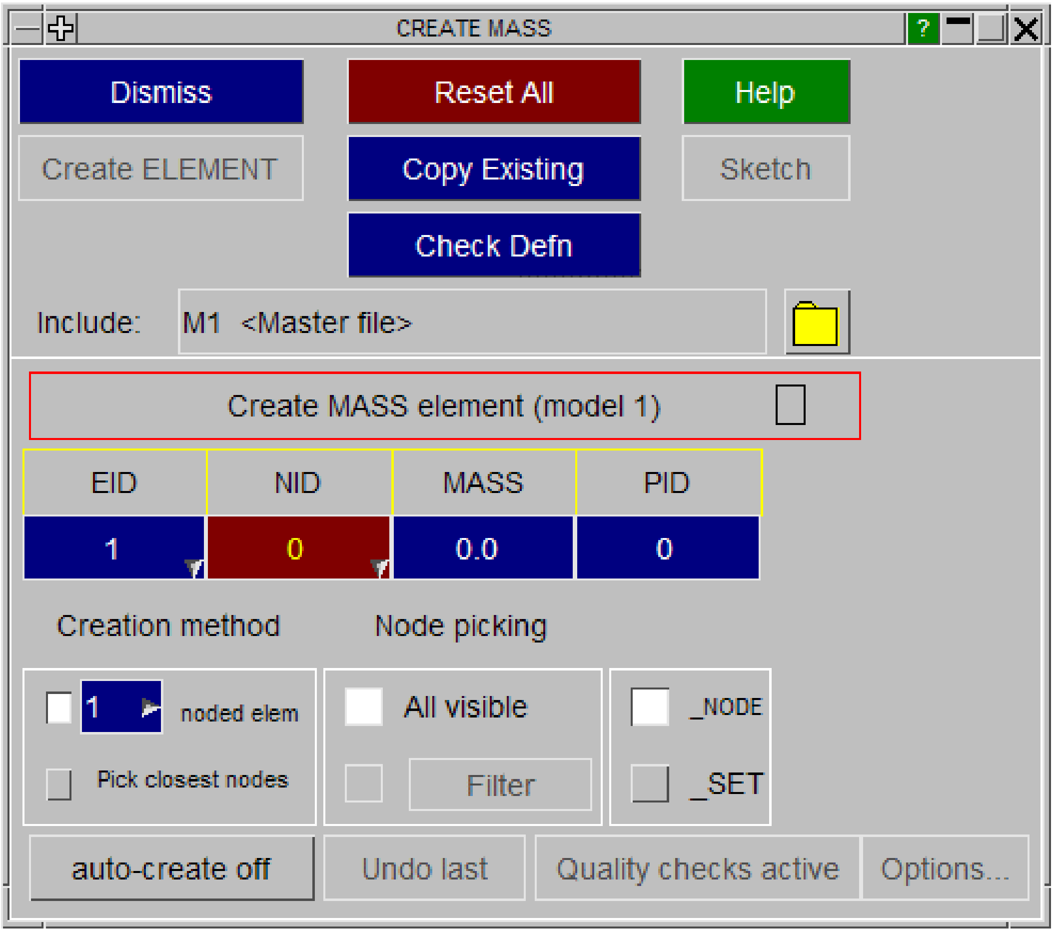
This figure shows the inertia element creation panel.
Only 1 node needs to be picked for the inertia element. The components of the inertia tensor and the coordinate system are saved as defaults when an element is created.

There are no element quality checks available for inertia elements at present.

ELEMENT_MASS

This figure shows the mass element creation panel.
Only 1 node needs to be picked for the mass element. The mass is saved as a default when an element is created.

There are no element quality checks available for mass elements at present.

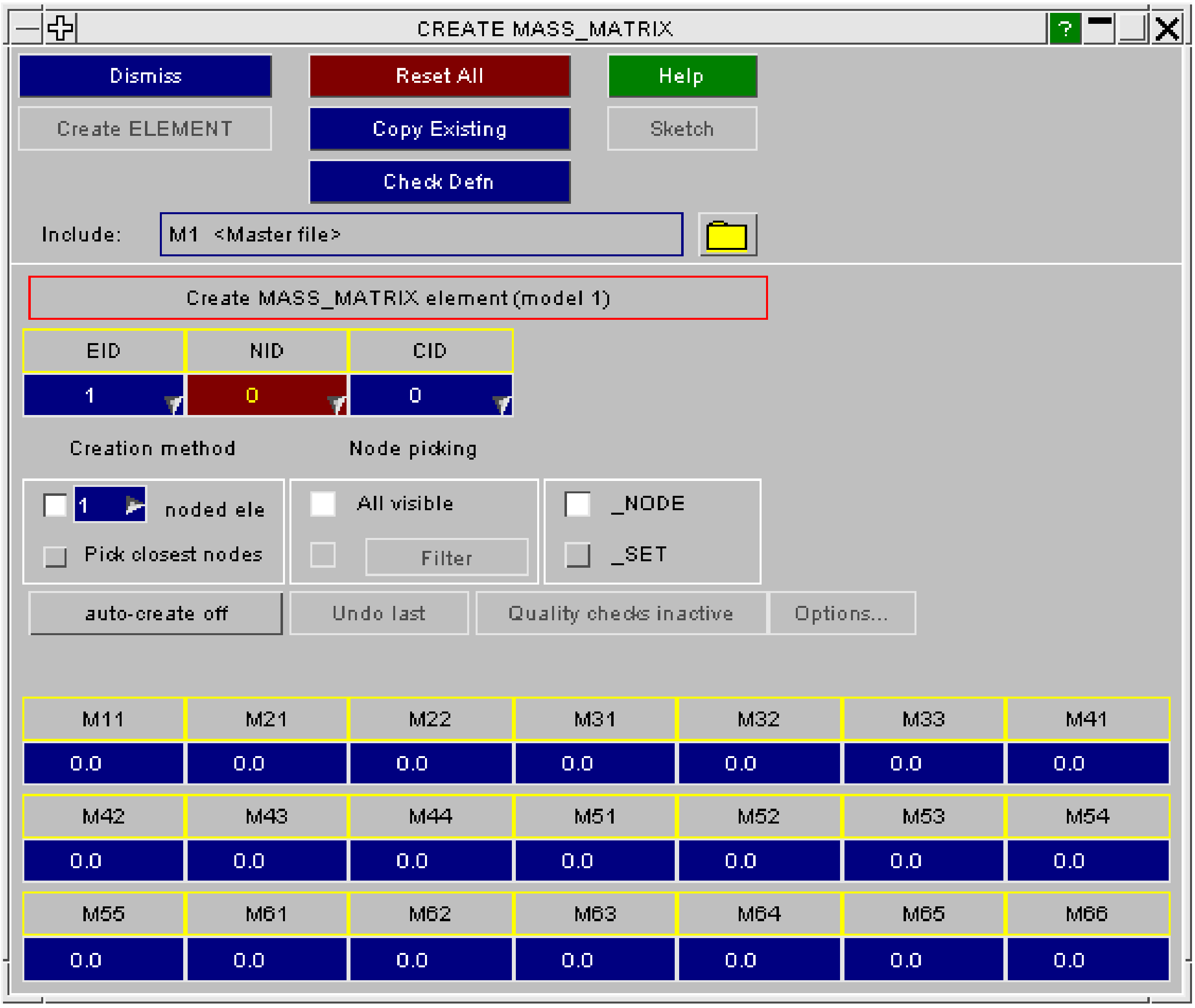


ELEMENT_MASS_MATRIX

This image shows the Create/Modify panel for *ELEMENT_MASS_MATRIX

One node, or node set, is selected at a time and a 6x6 mass matrix is defined for it. This is described in more detail below.

No quality checks are carried out for this element.



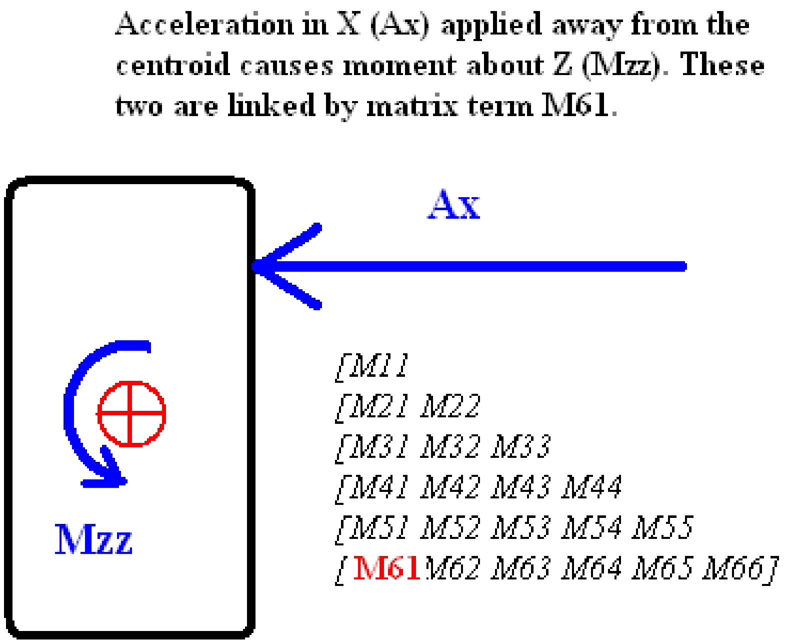
The 6x6 mass matrix [M]: what it means and how PRIMER handles it.

This element reads a symmetric 6x6 mass matrix which populates all possible terms of the classic **F = M.A** equation as follows:

Force	**=**	**Mass**						**x**	**Accel**			
Fx	[	M11	*m12*	*m13*	*m14*	*m15*	*m16*	]	Tx	Where:	Fx/y/z	are translational forces in [X,Y,Z]
Fy	[	M21	M22	*m23*	*m24*	*m25*	*m25*	]	Ty		Mxx/yy/zz	are rotational moments about [X,Y,Z]
Fz	= [	M31	M32	M33	*m34*	*m35*	*m36*	] x	Tz		Tx/y/z	are translational accelerations
Mxx	[	M41	M42	M43	M44	m45	m46	]	Rxx		Rxx/yy/zz	are rotational accelerations
Myy	[	M51	M52	M53	M54	M55	m56	]	Ryy			
Mzz	[	M61	M62	M63	M64	M65	M66	]	Rzz			

The matrix is symmetric, with only the lower triangle defined, so upper triangle terms M12 etc (in italics above) are identically equal to M21 etc.

A good way to think of term **Mij** is that it links **force** in direction <**i**> with **acceleration** in direction <**j**>. For example:



Extracting Mass and Inertia from the 6x6 matrix [M]

This raises two interesting questions:
- What is the (scalar) mass of this element?
- What is the inertia of this element?

These properties are needed when PRIMER calculates the mass and inertia of a model.

To understand how PRIMER calculates these values it is necessary to consider the full 6x6 [M] matrix above as the following block matrix of 3x3 sub-matrices:

[	**[A]**	|	*[B]*	**]**	Where: **[A]**	is a symmetric tensor describing mass
[	**----**	+	**----**	**]**	**[C]**	is an unsymmetric tensor describing cross-linking terms (*[B]* is the transpose of **[C]**)
[	**[C]**	|	**[D]**	**]**	**[D]**	is a symmetric tensor describing inertia

The "cross-linking" terms in sub-matrix [C] will be non-zero if the matrix has been set up to describe the result of accelerations not acting through the centroid, as in the image above, and - strictly - the full [M] matrix should be reworked to reduce the terms in [C] to zero before [A] can be considered to be "pure mass" and [D] "pure inertia". The physical equivalent of this reworking would be to shift the matrix so that it described accelerations acting through the element centroid.

However this is very hard to do and, given the unconstrained nature of the input, it would be easy for a user to define an unrealistic (non-physical) matrix in which this is not possible, so PRIMER adopts the following simplified approach:

MASS is taken to be the 1st invariant of sub-matrix [A]

That is the average of the leading diagonal terms: **mass = (M11 + M22 + M33) / 3.0**

INERTIA is taken to be the symmetric tensor in sub-matrix [D]:

[	**Ixx**	*Ixy*	*Ixz*	**]**		**[**	**M44**	*M45*	*M46*	**]**
[	**Iyx**	**Iyy**	*Iyz*	**]**	**=**	**[**	**M54**	**M55**	*M56*	**]**
[	**Izx**	**Izy**	**Izx**	**]**		**[**	**M64**	**M65**	**M66**	**]**

The "cross-linking" terms in sub-matrix [C] <u>are ignored</u>.

If the terms in sub-matrix [C] are large, ie the matrix [M] has been written to describe behaviour remote from the element centroid, then the Inertia (which includes "distance squared" terms) and - to a lesser extent - the Mass calculated using the methods above will be wrong.

Rotating matrix [M]

PRIMER rotates a *ELEMENT_MASS_MATRIX as follows:

The 6x6 matrix [M] is treated as a block matrix of three 3x3 matrices:

[	**[A]**	|	*[B]*	**]**
[	**----**	+	**----**	**]**
[	**[C]**	|	**[D]**	**]**

Each sub-matrix [A], [B], [D] is rotated separately as a 3x3 tensor.

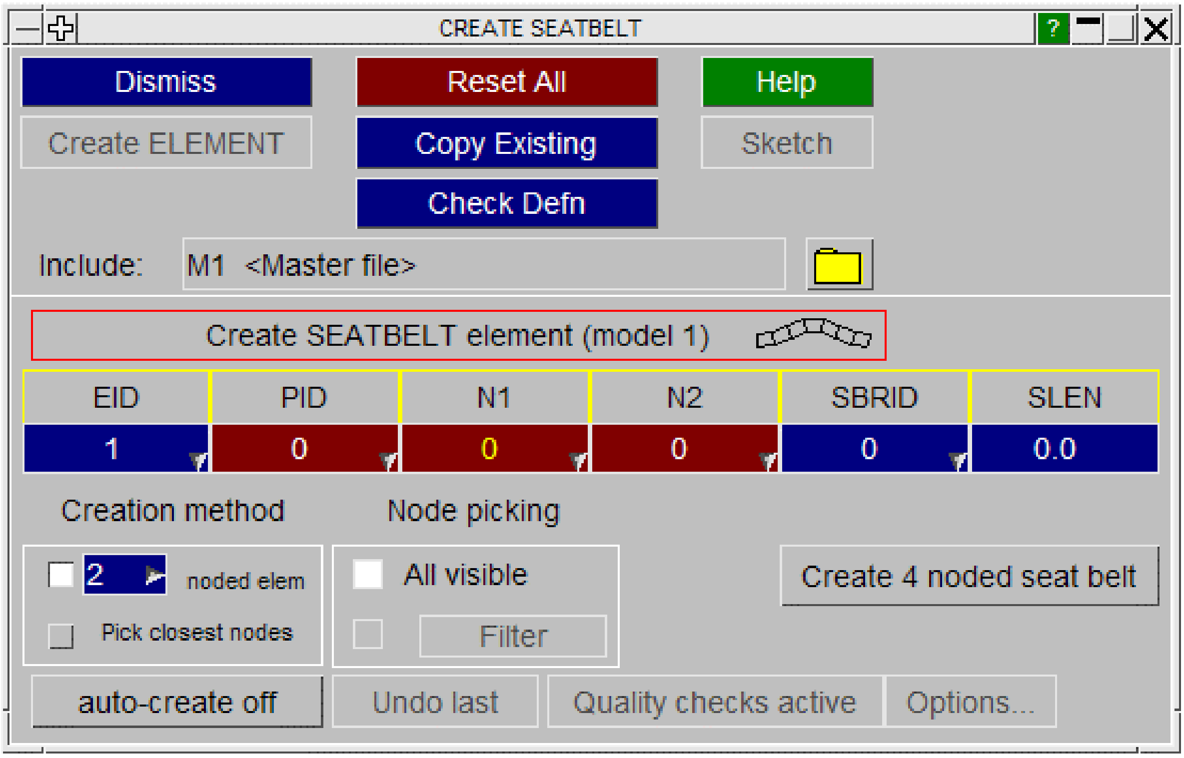
ELEMENT_SEATBELT

This figure shows the seatbelt element creation panel.
Two nodes need to be picked for the seatbelt element. The part number and slack length (**SLEN**) are saved as defaults when an element is created.

There are no element quality checks available for seatbelt elements at present.

This panel is suitable for creation and editing of individual seatbelt elements.

For generating and fitting a line of seatbelt elements to an occupant model PRIMER provides a "seatbelt fitting" capability: see section 6.



Four noded seatbelt elements (introduced in LS971)

The four noded seatbelt elements introduced in LS-DYNA release 971 are in fact shell elements, and share the same numbering sequence as conventional shells, so within PRIMER they are edited under shell elements.

On output they will be written correctly under the *ELEMENT_SEATBELT header. The user can click on **Create 4 noded seat belt** to open up a shell creation panel with the seatbelt option active.

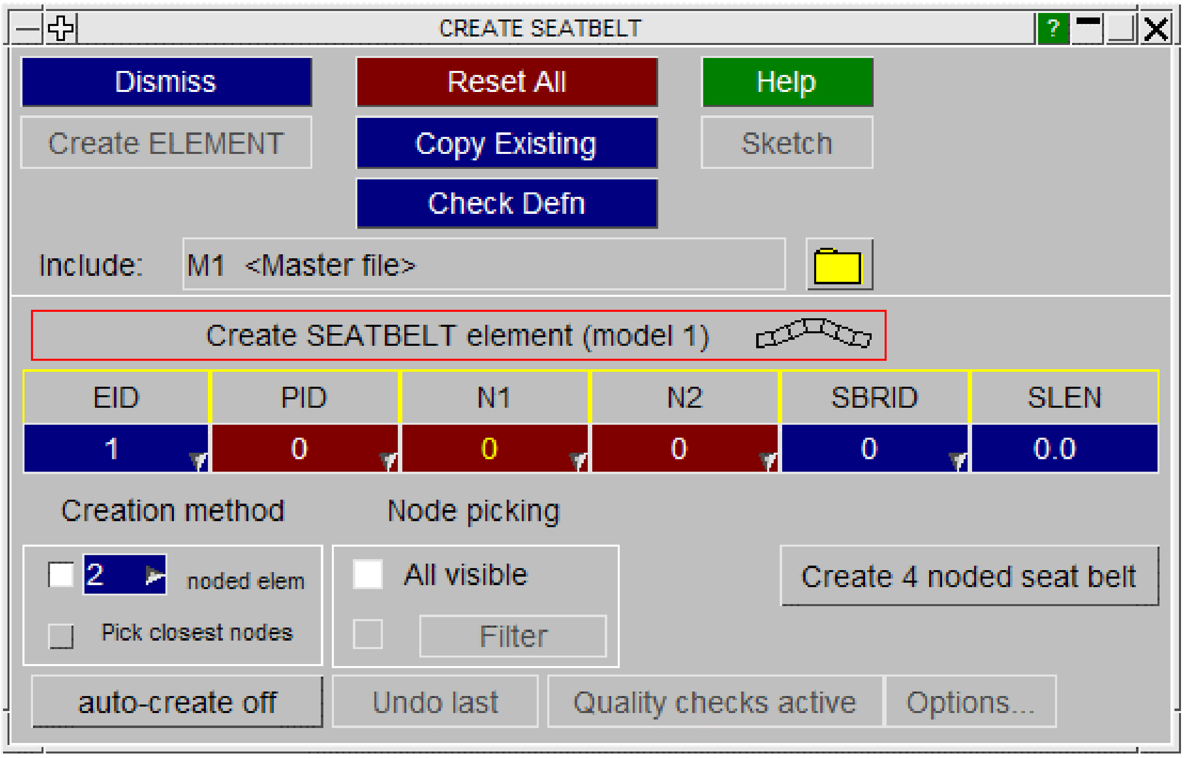
ELEMENT_SEATBELT_ACCELEROMETER

This figure shows the accelerometer create/edit panel.
Three nodes must be selected:

- N1: Origin
- N2 : Local X axis from N1N2
- N3 : Local XY plane from N1N2N3

All three nodes should be on the same rigid part.

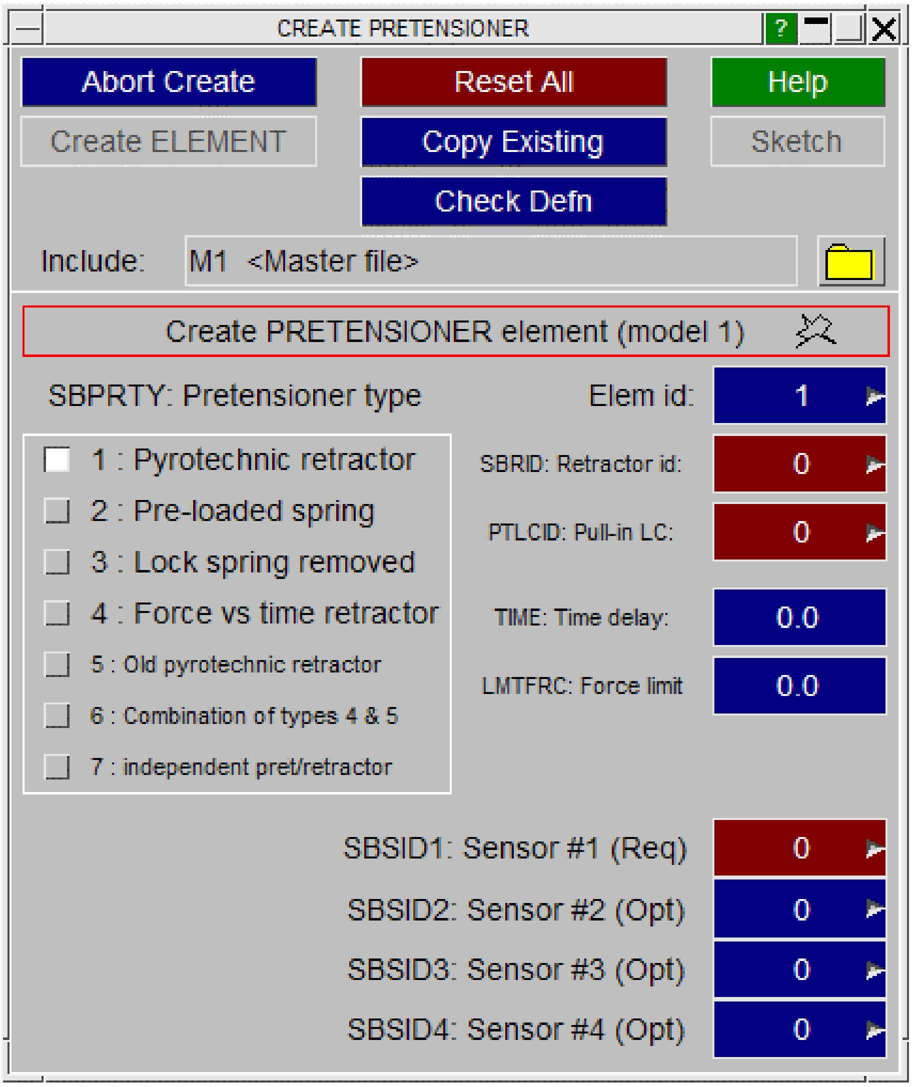
Accelerations will be output in the local coordinate system of the accelerometer.



ELEMENT_SEATBELT_PRETENSIONER

This figure shows the pretensioner create/edit panel.
Pretensioners are active devices that tighten a seatbelt in the event of a crash.

Three different types are provided, which may be triggered by up to four **SENSOR**s.

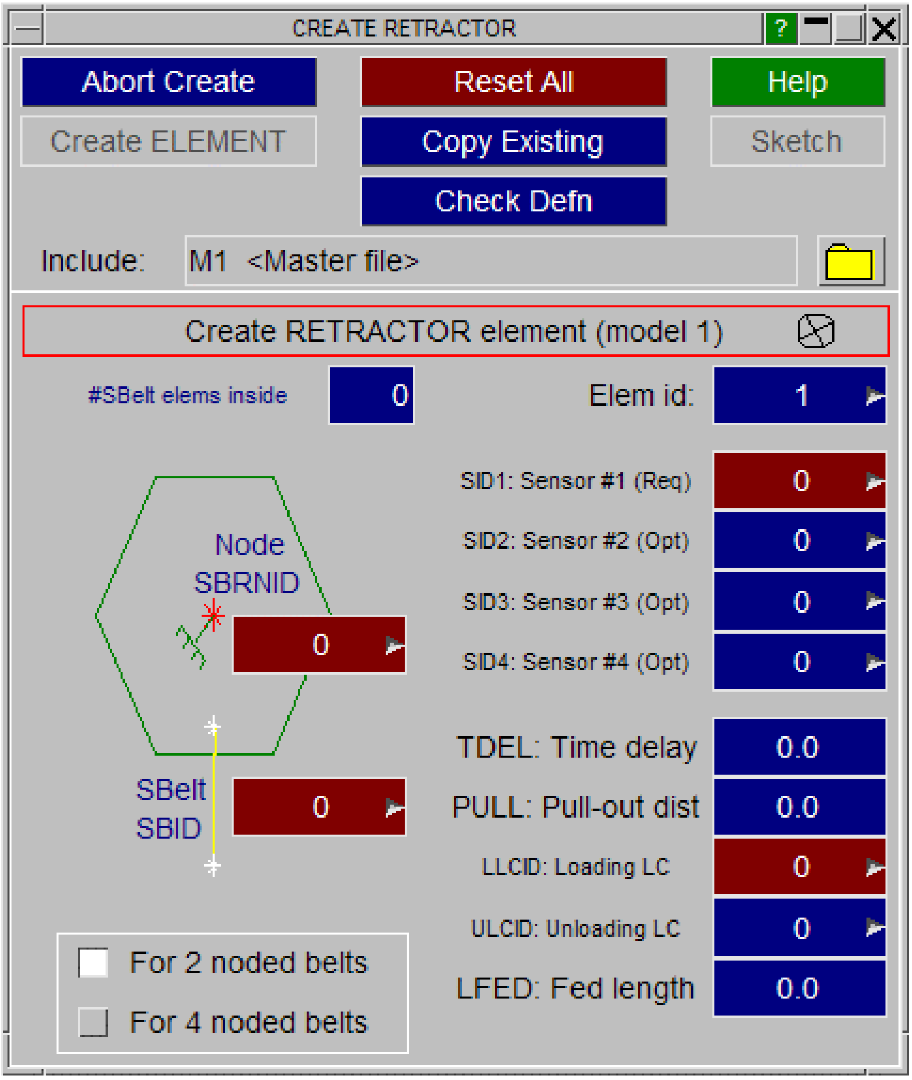


ELEMENT_SEATBELT_RETRACTOR

This figure shows the retractor create/edit panel.
Retractors are the "inertia reel" part of the seatbelt system.

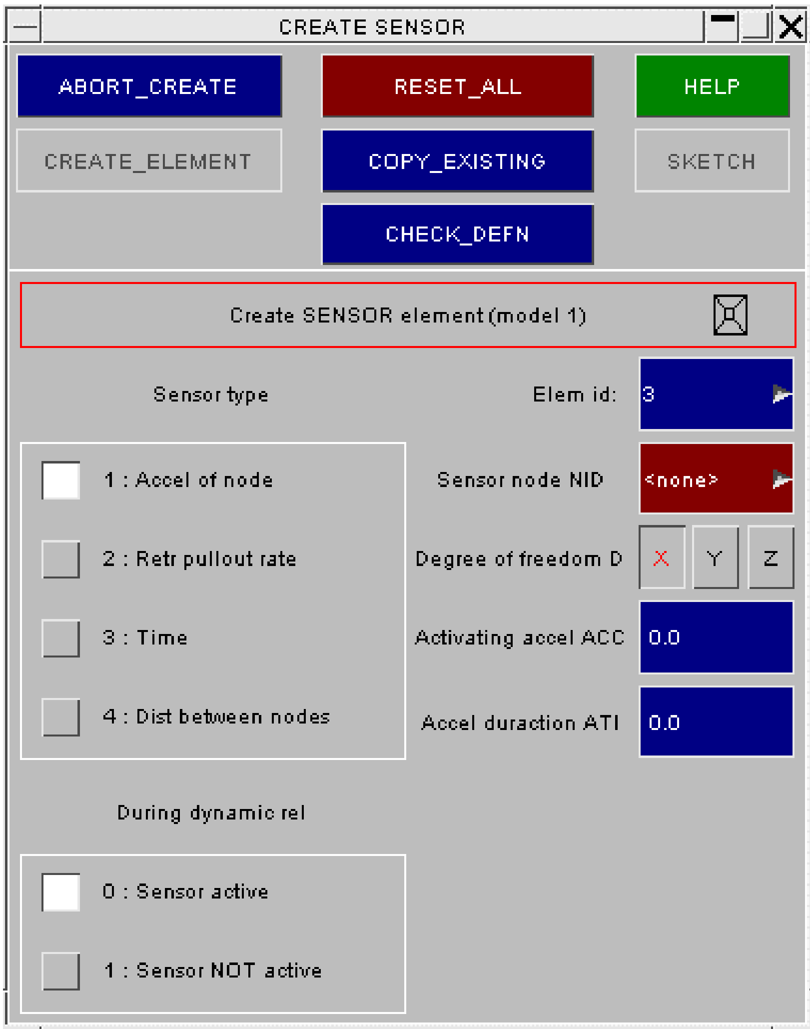
They are assumed to have some number of seatbelt elements curled up inside them, and to apply an constant tension to the belt to take up any slack.

When a collision occurs, denoted by a **SENSOR** activating, they can be programmed to "lock up".



ELEMENT_SEATBELT_SENSOR

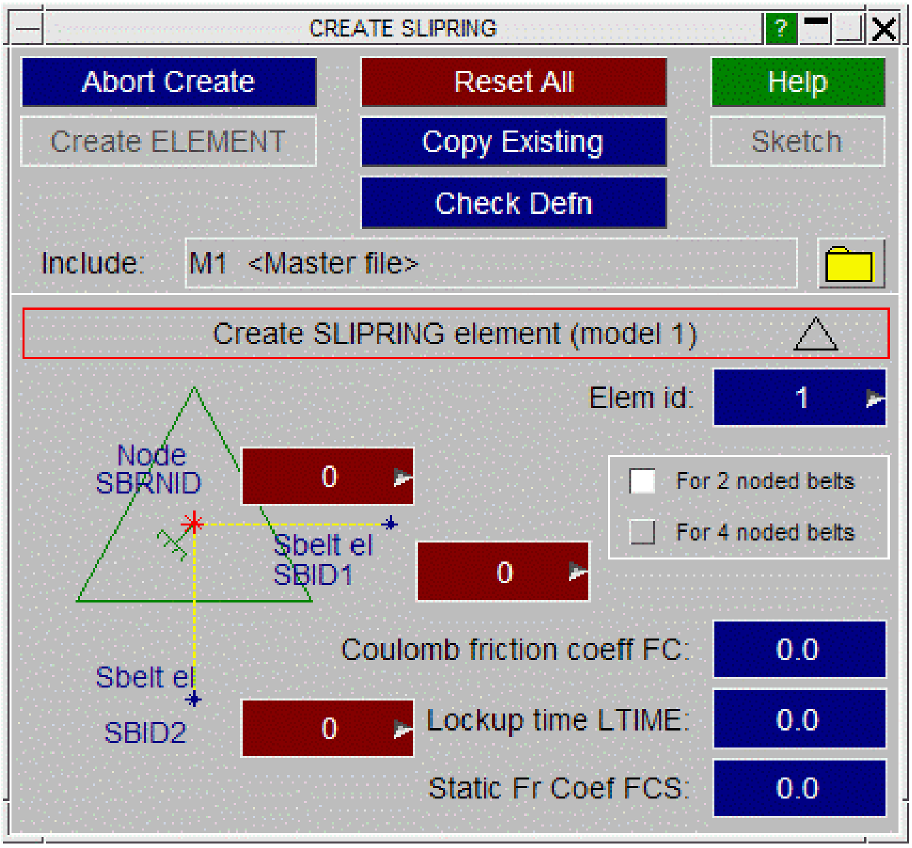
This figure shows the sensor create/edit panel.
Sensors are not really elements in the structural sense (forgive the pun!) of the word. Their role is to detect one of four types of event and then to trigger other **ELEMENT_SEATBELT_xxx** elements.



ELEMENT_SEATBELT_SLIPRING

This figure shows the slipring create/edit panel.
Sliprings permit seatbelt elements to "feed through" from one side to the other, emulating the real behaviour in a crash event.

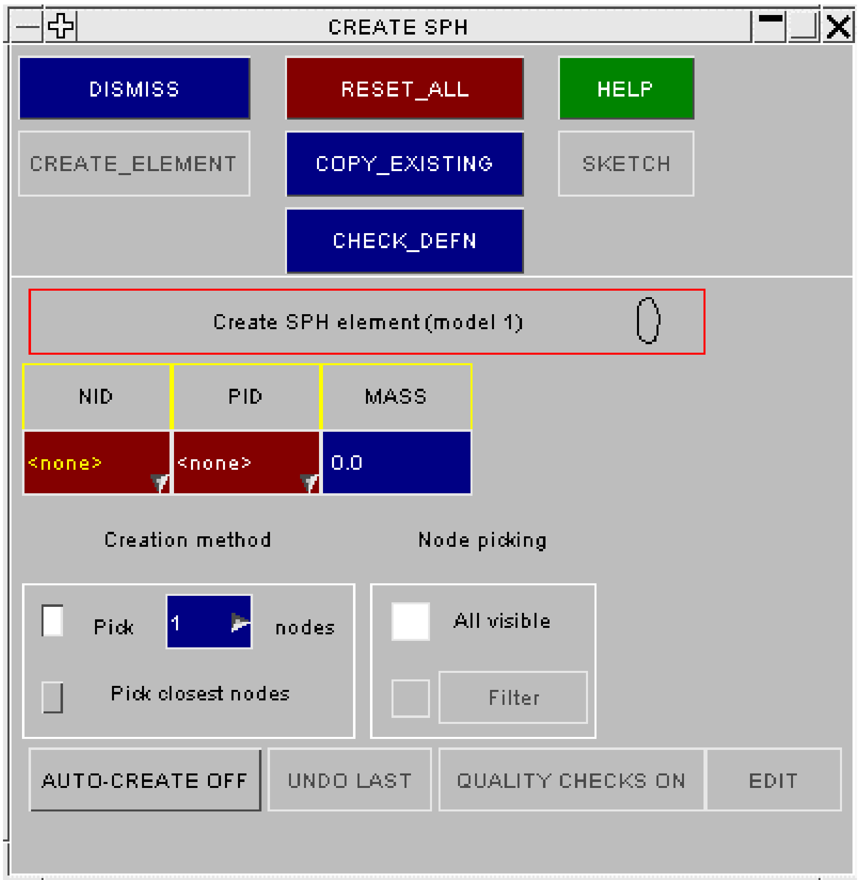
They require two contiguous seatbelt elements **SBID1** and **SBID2** to be defined, whose common node must initially be coincident with the slipring node **SBRNID.**



ELEMENT_SPH

This figure shows the sph element creation panel. This is virtually identical to the mass element except a part also needs to be chosen.
Only 1 node needs to be picked for the sph element. The mass and part number are saved as defaults when an element is created.

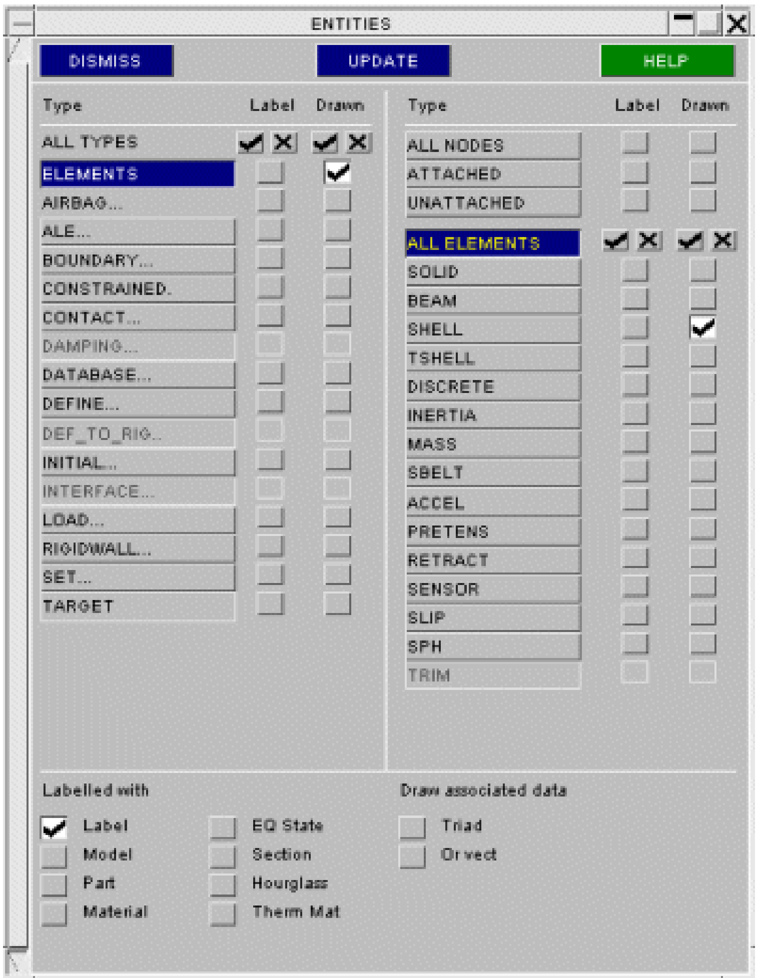
There are no element quality checks available for sph elements at present.



ELEMENT_TRIM

This figure shows the trim element creation panel. This element does not have any node picking associated with it. The only thing that needs to be defined is a part set.
Trim elements are used in conjunction with ***DEFINE_CURVE_TRIM** in metal forming analyses.

Visualisation and labelling of elements

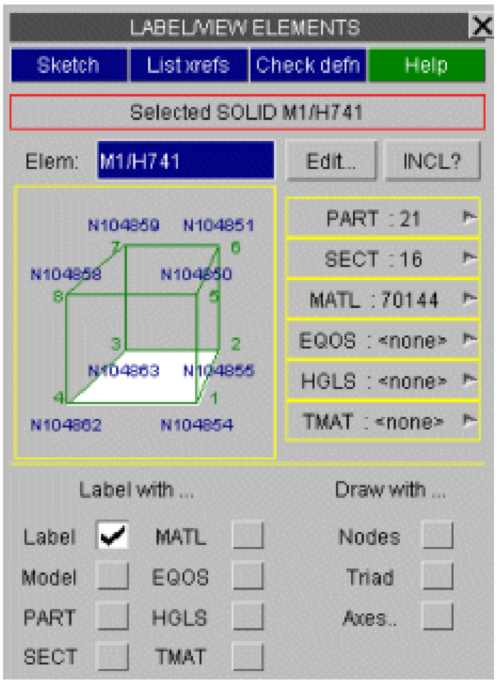


PRIMER draws and labels all element types (except TRIM), display being controlled in the

ENTity Viewing panel. (See section 4.3)
Details of an individual element of a specific type can be obtained by clicking on its category in this panel (eg **SOLID**) and then screen-picking or typing in the element label.

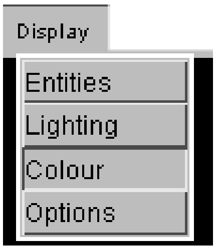
To visualise elements of any type use **ALL ELEMENTS**.

The details panel for a solid element is shown below:



This is accessed either from Quick Pick (click on the element in the graphics window) or by clicking on the word SOLID (or BEAM, SHELL, etc) in the Entities panel and typing in the label of the element.

Controlling the colour in which elements are drawn



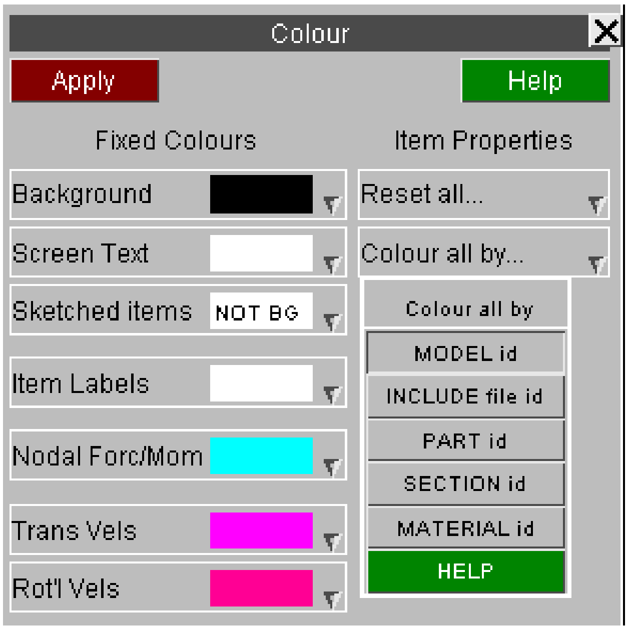
The **COLOUR** panel menu controls the display of elements that reference ***PART** cards.

Such elements may be drawn in colours based on:

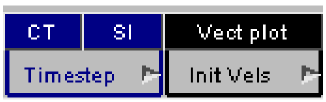
PART id (default)
MODEL id
INCLUDE file id
SECTION id
MATERIAL id

Colour can also be changed for parts and individual elements using Quick Pick.

See section 4.1.2 for more detail.



Data display and contouring



A range of different quantities can be displayed as data contours on elements via the **CT** (Continuous Tone) and **SI** (Shaded Image) commands. The **Vect or plot** command can also display element data.
These are described in section 4.2: Data Plotting Commands.

Special capabilities for seatbelts and related element types

Seat-belt elements can be fitted to occupants using the **OCCUPANT, SEAT-BELTS** menu

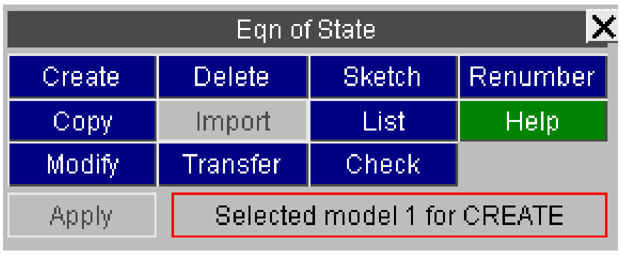
This involves form-finding to establish the line of the belt, a fitting process to pull it onto a dummy, meshing, element property definition and contact creation. It is described in section 7.2.3.7.

EQOS: Equation of State

- Top level menu
- Creating
- Copying
- Editing
- Deleting
- Sketch
- List
- Check
- Renumber

Equations of State are used to define material properties for special or typical fluids or null material types. Consequently, they are controlled in a similar fashion to the ***Material** keywords.

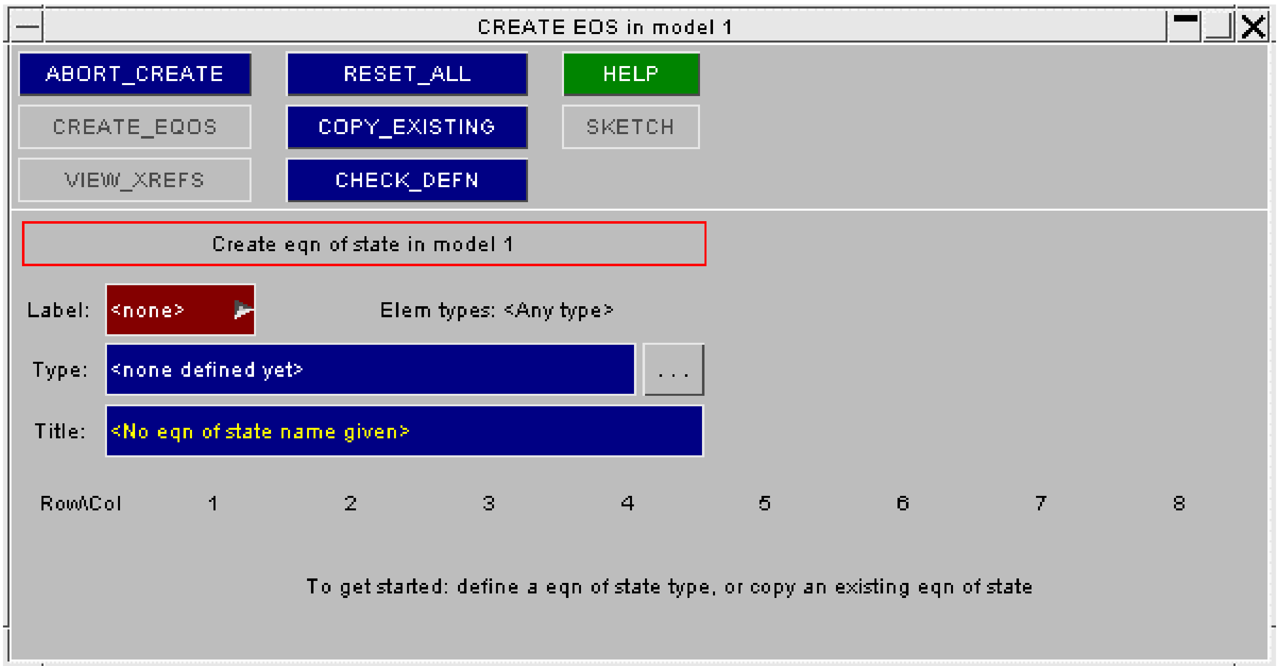
This figure shows the main equation of state editing panel.



TRANSFER opens the main window for the transfer data function (see section 6.7 for more detail)

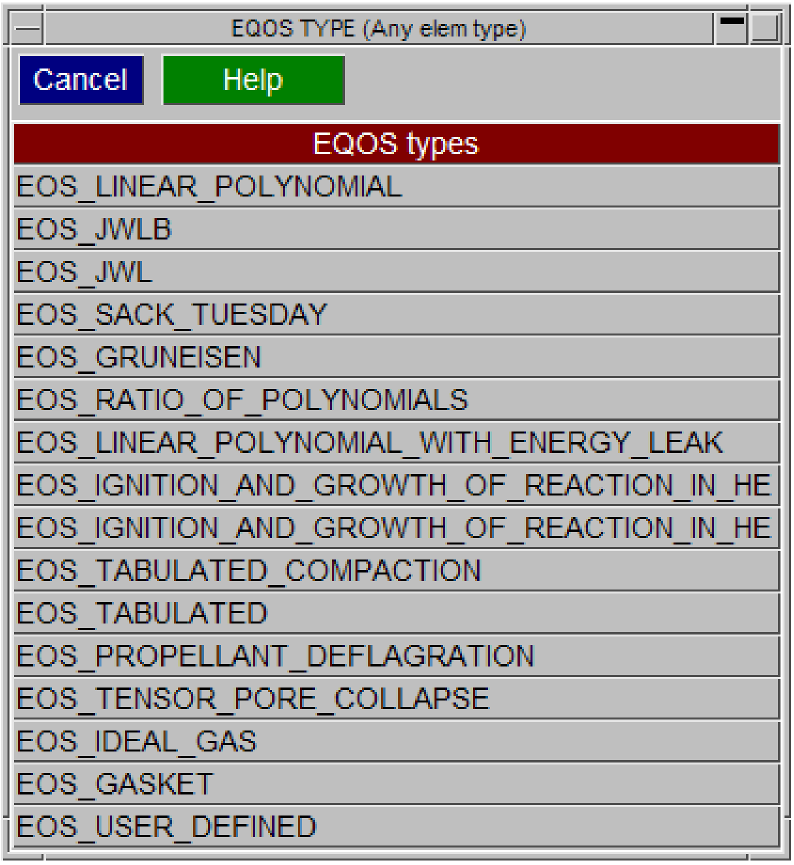
The other functions currently available have their standard meanings. (See 5.0.1).

CREATE Making a new equation of state definition.



CREATE produces this blank equation of state creation panel, since no equation of state type has been defined yet.

Type:



The equation of state type can be defined from this button.

The [...] Shortcut button can be used to browse through a list of equation of state types as shown here.

Note on selecting an Equation of State:

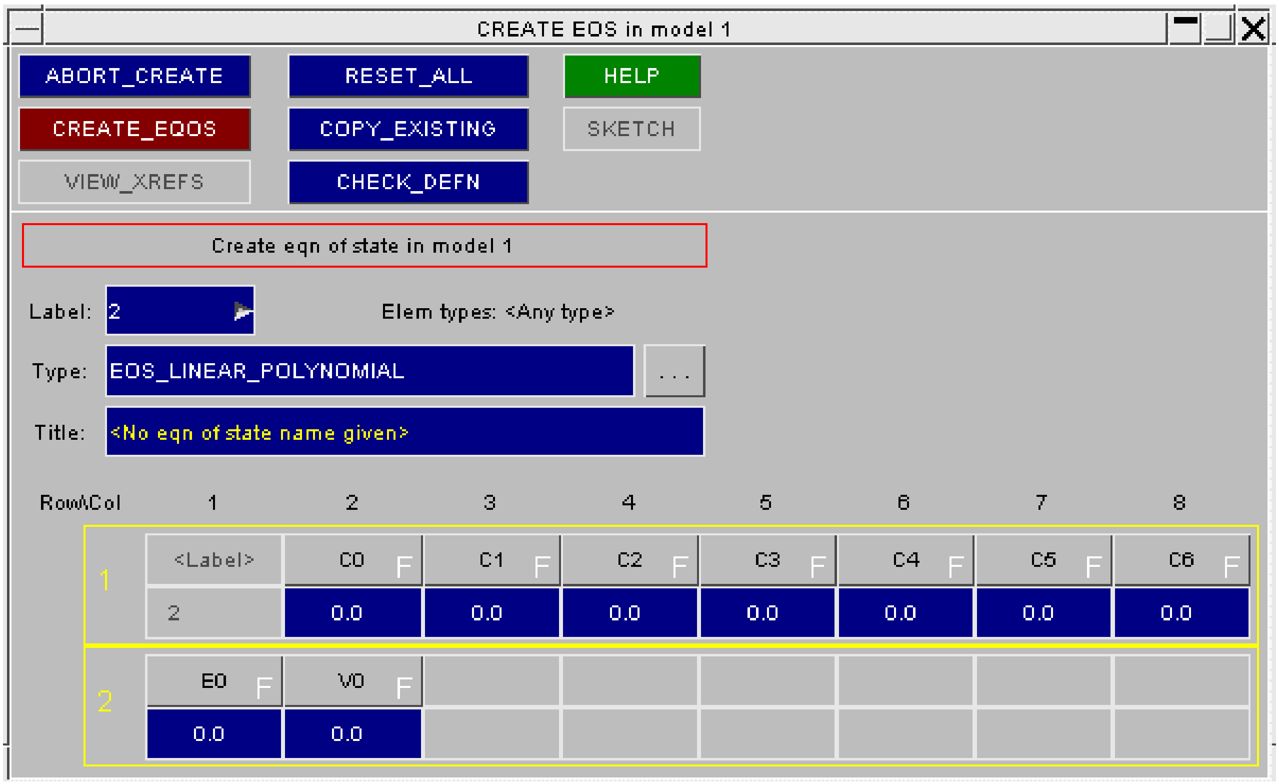
An equation of state may be selected by one of two ways:

- by invoking the browse [...] button and selecting the equation of state with the mouse from the list
- by typing in the equation of state name to the "Type" box, e.g. "ideal_gas" for "***EOS_IDEAL_GAS**"

ROW/COL

The data relevant to each equation of state type is displayed in row and column format identical to that of DYNA keyword.

Once a equation of state type has been defined the panel will become populated with that equation of state's format. For example the type ***EOS_LINEAR_POLYNOMIAL** has been chosen here:

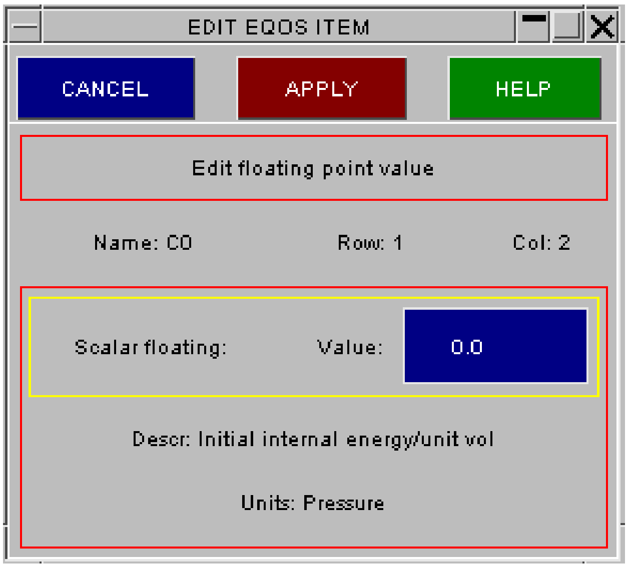


The data can then be typed into the relevant boxes. The expected data type is indicated on the grey button, which also shows the acronym for that data value:

F	White **F** floating value
I	White **I** integer value
+LC	Green **LC** +ve loadcurve
-LC	Red **LC** -ve loadcurve

Information about each individual data component can be requested by pressing the grey data component button. For example; to request information about data component '**C1**' (1^{st} row, 3^{rd} column) press the grey button with the C1.

This will create a new window with detailed information about that data component showing:



- A one-line description of it;
- Its current units type
- Its current value.

Once all of the data has been input , press **CREATE_EQOS** to install the equation of state permanently in the model.

As with other creation/editing functions a standard check is made of the new definition prior to saving it, and you are warned about errors found.

COPY Copy existing equation of state(s) to make a new equation of state(s).

COPY makes new equation of states, in the same model(s), that are identical to their originals apart from their labels. By default only the equation of state definitions themselves are duplicated.

Where **RECURSIVE COPY** is requested, all items associated with that equation of state (i.e. elements, parts, etc.) are also copied.

MODIFY Modifying the attributes of an existing equation of state.

MODIFY functions in the same way as **CREATE**. Obviously, the equation of state will already have been selected so the panel will resemble that shown in "populated" figure above.

If the equation of state is in use by a `PART` which has elements then the "element type" of the equation of state is locked to those elements, which will restrict the range of equation of state types you can change it to. For example a equation of state used by springs cannot be changed to a shell type.

Any modifications made to the equation of state definition will not be made permanent until the **APPLY_MODIFY** button is pressed. At this point a the local copy which has been updated is used to overwrite the version in the model.

DELETE Delete existing equation of state definitions.

The **DELETE** function deletes the selected equation of states. However you cannot delete a equation of state that is in use by a `PART` unless you remove it from the part definition, or delete that too. To help with this the following two switches may be used:

DELETE_RECURSIVE Will select for deletion the parts, associated elements, and so on that reference thisequation of state.

REMOVE_FROM_SETS Is often also needed if parts are to be deleted, as their elements, the connected nodes, and often the parts themselves may be included in sets.

A good way of getting rid of surplus (unused) equation of states is to turn these two switches off, then select all equation of states for deletion. Only those which are not used by anything will actually get deleted.

SKETCH Sketch elements using an equation of state on the current image.

SKETCH sketches on top of the current image the parts and elements that reference the selected equation of states.

LIST Summarise the attributes of selected equation of states.

LIST allows the user to individually select equation of states and display a summary listing of their attributes.

CHECK Check selected equation of states for correctness

CHECK runs the standard checking function on the selected equation of states, summarising any errors.

WARNING: Checking equations of state thoroughly is a mammoth task, which PRIMER does not at present attempt. Most equations of state are currently only checked for a positive density. Therefore that an equation of state"Checks OK" does not mean that it contains no errors!

RENUMBER Renumbering equation of state labels.

RENUMBER lets you change any or all equation of state labels within a given model using the standard renumbering panel.

To change the label of an individual equation of state it may be simpler just to **MODIFY** it.

HOURGLASS: Hourglass control cards.

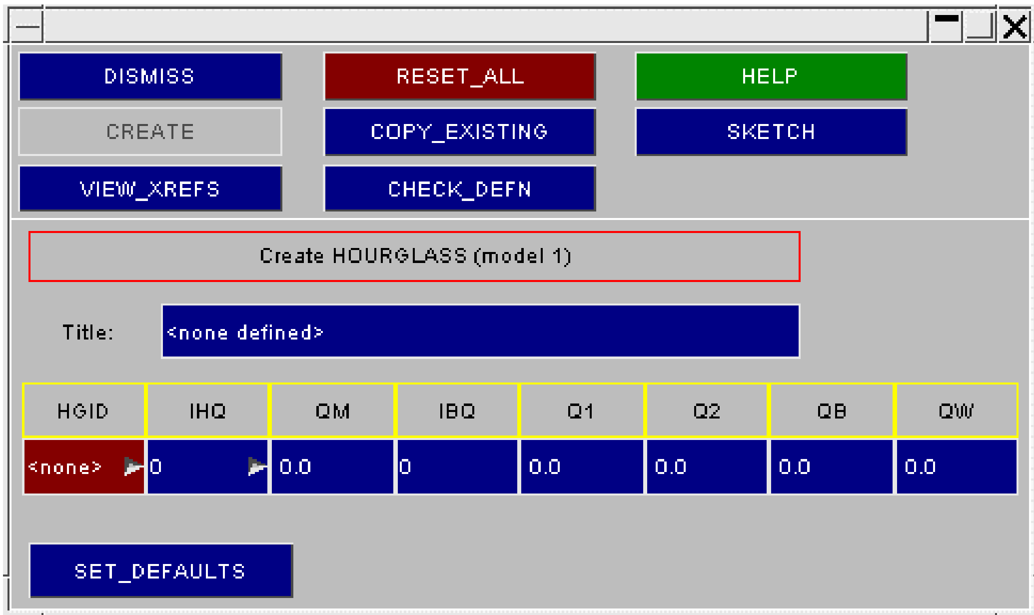
Hourglass cards are used in LS-DYNA to control the zero energy "hourglass modes" that occur with single integration point elements.
They are also used to specify bulk viscosity coefficients.

HOURGLASS MAIN MENU
This figure shows the main **HOURGLASS** menu.

The functions currently available have their standard meanings (see section 5.0.1).

Greyed out functions are not currently available:

CREATE Making a new hourglass definition.



This figure shows the standard **CREATE**/**EDIT** panel for hourglass cards. Here **CREATE** has been used, so a blank hourglass creation panel is displayed.The static buttons in the top section of the panel have functions which are common to the other editing panels within PRIMER.

Once all of the data has been input on the airbag card, **CREATE** installs the hourglass card permanently in the model.

The **SET_DEFAULTS** button will put the **current default values** into the fields. These will be taken from the `CONTROL_HOURGLASS` and the `CONTROL_BULK_VISCOSITY` settings. If the LS-DYNA default settings are active, after pressing the button the values are set thus:

HGID	IHQ	QM	IBQ	Q1	Q2	QB	QW
<none>	1	0.100	1	1.500	0.0600	0.100	0.100

COPY Copy existing hourglass card(s) to make a new card(s).

COPY makes new hourglass, in the same model(s), that are identical to their originals apart from their labels.

RECURSIVE COPY has no effect as hourglass cards do not 'own' anything else.

MODIFY Modifying the attributes of an existing hourglass card.

MODIFY functions in the same way as **CREATE**.

Any modifications made to the hourglass definition will not be made permanent until the **APPLY_MODIFY** button is pressed. At this point a the local copy which has been updated is used to overwrite the version in the model.

DELETE Delete existing hourglass definitions.

The **DELETE** function deletes the selected airbag. The **DELETE_RECURSIVE** and **REMOVE_FROM_SETS** switches have no effect here as hourglass cards do not 'own' anything and are not in sets.

SKETCH Sketch elements used by an hourglass on the current image.

SKETCH sketches on top of the current image the parts and elements that are referenced by the selected hourglass card.

LIST List hourglass cards

LIST gives a list of the selected hourglass cards showing the cross references to each card.

CHECK Check hourglass attributes

CHECK checks one or more hourglass cards for errors.

RENUMBER Renumbering hourglass labels.

RENUMBER lets you change any or all hourglass labels within a given model using the standard renumbering panel.

To change the label of an individual hourglass card it may be simpler just to **MODIFY** it.

DISMISS terminates hourglass processing.

Visualising hourglass cards

Hourglass cards are not explicitly drawn, or selectable for drawing. To view an hourglass definition:

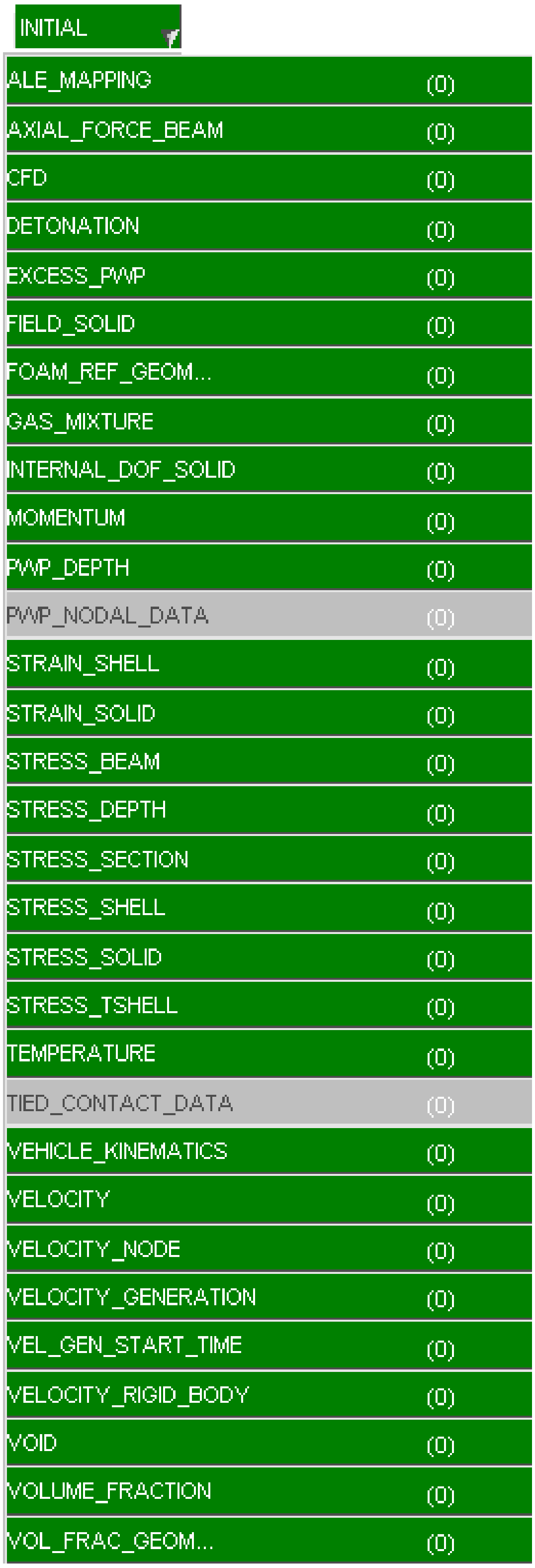
- **SKETCH** it, or
- **MODIFY** it and sketch it.

It will be drawn in terms of the parts and elements that use it.

Note that Hourglass data can also be created or modified using PART TABLE.

INITIAL: Defining Initial Conditions.

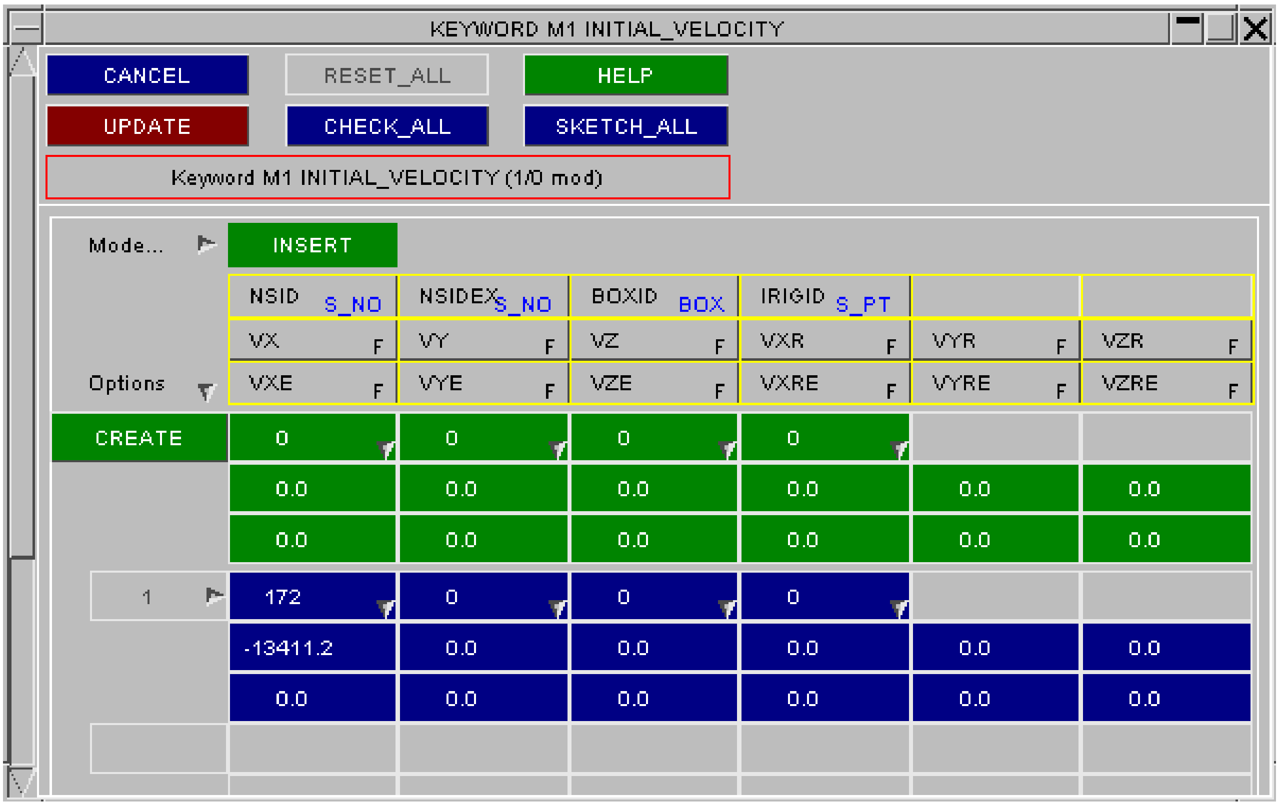
Initial conditions may be defined for a range of items with LS-Dyna.


INITIAL
ALE_MAPPING (0)
AXIAL_FORCE_BEAM (0)
CFD (0)
DETONATION (0)
EXCESS_PWP (0)
FIELD_SOLID (0)
FOAM_REF_GEOM... (0)
GAS_MIXTURE (0)
INTERNAL_DOF_SOLID (0)
MOMENTUM (0)
PWP_DEPTH (0)
PWP_NODAL_DATA (0)
STRAIN_SHELL (0)
STRAIN_SOLID (0)
STRESS_BEAM (0)
STRESS_DEPTH (0)
STRESS_SECTION (0)
STRESS_SHELL (0)
STRESS_SOLID (0)
STRESS_TSHELL (0)
TEMPERATURE (0)
TIED_CONTACT_DATA (0)
VEHICLE_KINEMATICS (0)
VELOCITY (0)
VELOCITY_NODE (0)
VELOCITY_GENERATION (0)
VEL_GEN_START_TIME (0)
VELOCITY_RIGID_BODY (0)
VOID (0)
VOLUME_FRACTION (0)
VOL_FRAC_GEOM... (0)

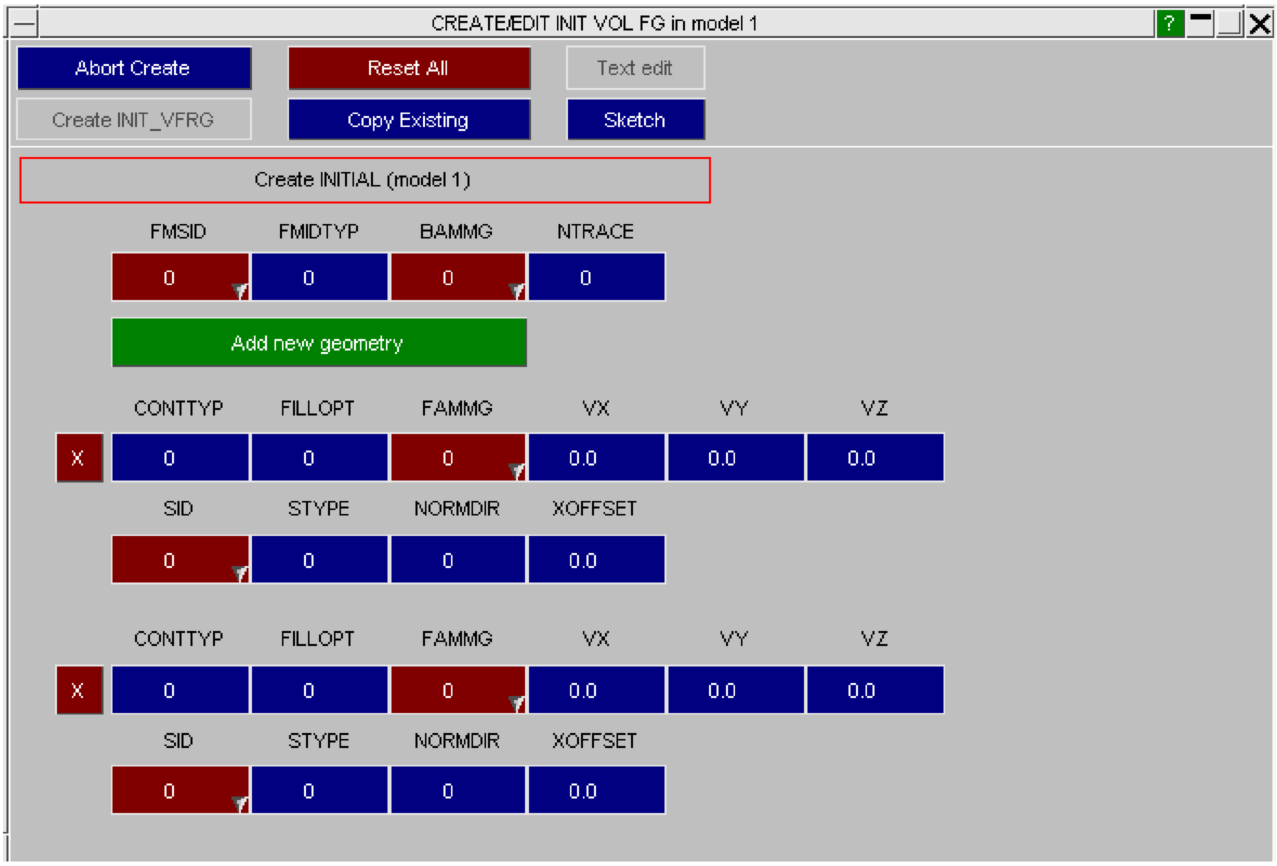
*INITIAL cards can be edited with the generic "Keyword" editor. All ***INITIAL** subtypes except **VOLUME_FRACTION_GEOMETRY** and **VELOCITY_GENERATION** (which have their own specific Create/Edit panel) can be processed in this way.

The other commands (**COPY, DELETE,** ...) function in the standard fashion as defined in section 5.0.1.

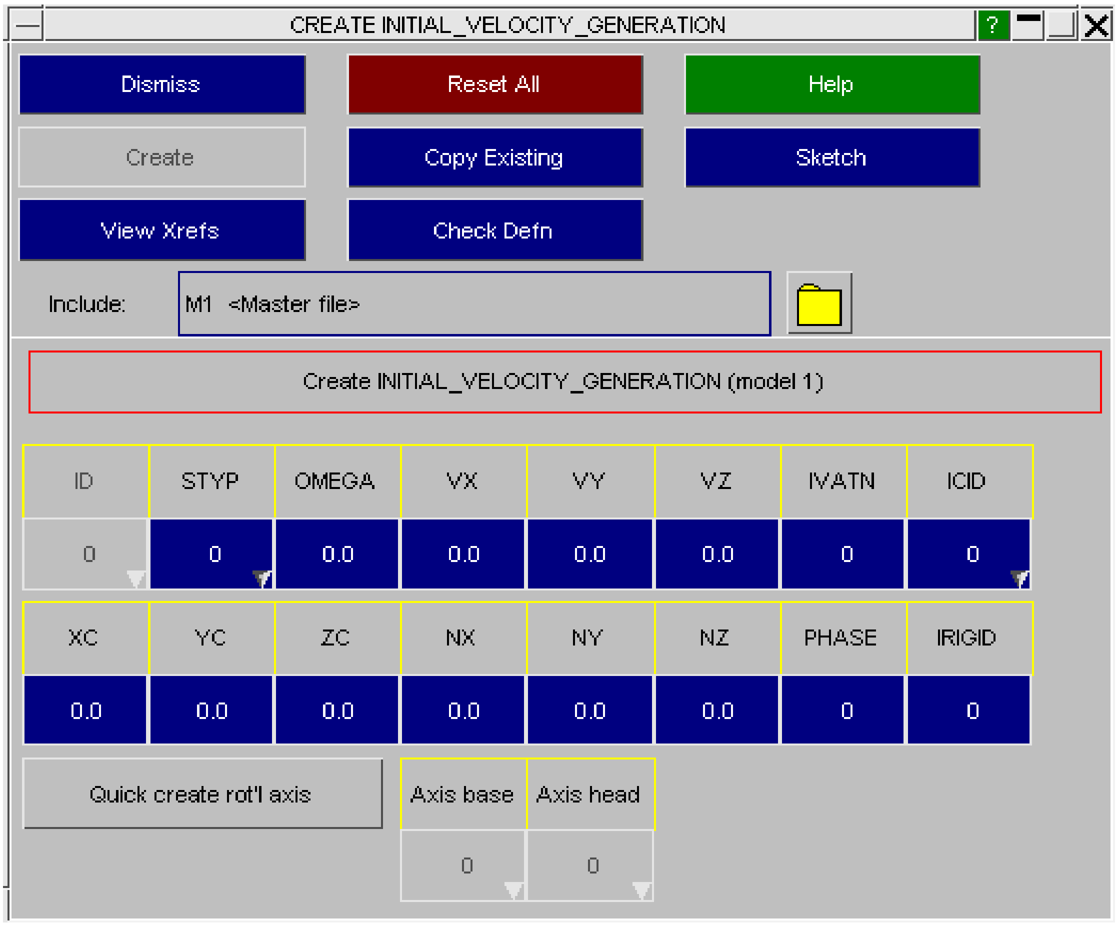
This shows an example of the Keyword editor for ***INITIAL_VELOCITY**



***INITIAL_VOLUME_FRACTION_GEOMETRY** items are created using a specific editing panel. New geometries can be added by clicking on the **Add new geometry** button.

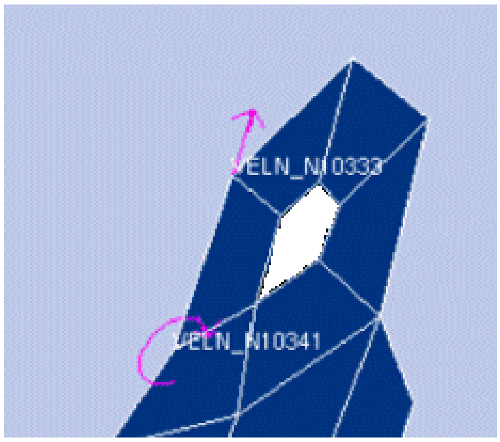


***INITIAL_VELOCITY_GENERATION** items are created using a specific editing panel.


CREATE INITIAL_VELOCITY_GENERATION
Dismiss
Reset All
Help
Create
Copy Existing
Sketch
View Xrefs
Check Defn
Include:
M1 <Master file>
Create INITIAL_VELOCITY_GENERATION (model 1)
ID
STYP
OMEGA
VX
VY
VZ
IVATN
ICID
0
0
0.0
0.0
0.0
0.0
0
0
XC
YC
ZC
NX
NY
NZ
PHASE
IRIGID
0.0
0.0
0.0
0.0
0.0
0.0
0
0
Quick create rot'l axis
Axis base
Axis head
0
0

Visualising *INITIAL items

*INITIAL items are not displayed by default, but can be selected for display and labelling in the **ENT**ity Viewing panel.
At present only *INITIAL_VELOCITY(_<type>) cards are drawn fully.



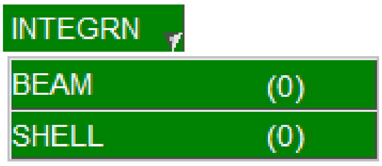
These are shown as arrows in the direction of the velocity. In this example there is a translational velocity at node 10333, and a rotational one at 10341.

If multiple initial velocities are defined at a node (an error) then all will be drawn separately.

Using the VEC plotting mode to display initial velocities
Initial velocities from any source (not just *INITIAL cards) can also be visualised in "vector contour" form via the **Vect plot** command.

Other *INITIAL items are not drawn explicitly, but can still be visualised by turning on (in **ENT**ity Viewing) the display of the items they reference: sets, nodes, elements, etc. Labels referring to the initial velocity type are generated correctly.

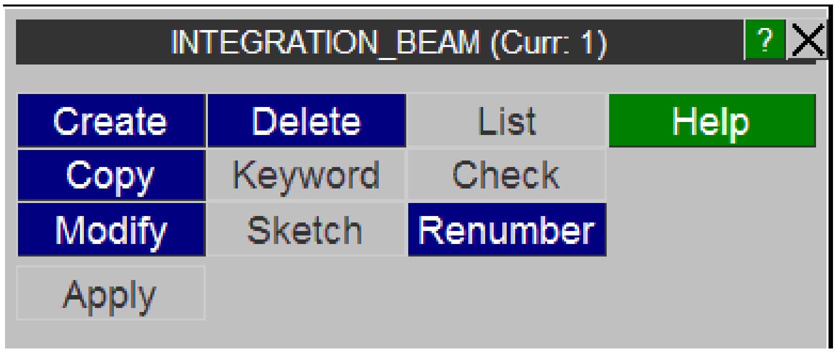
INTEGRATION: Defining Integration rules.



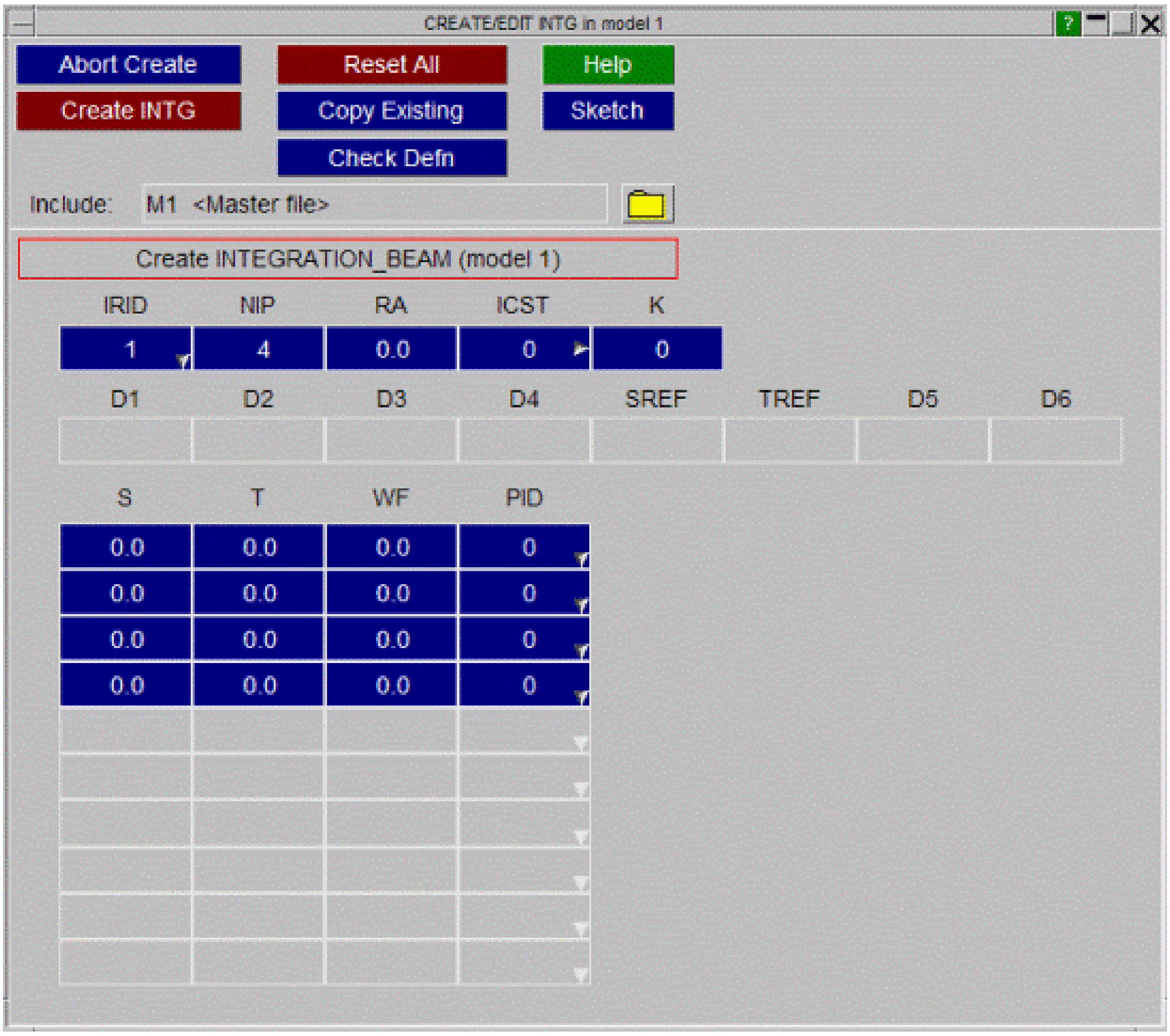
*INTEGRATION_BEAM
*INTEGRATION_SHELL

Select the integration rule to create from the keyword main panel, and the standard sub-menu will appear below.

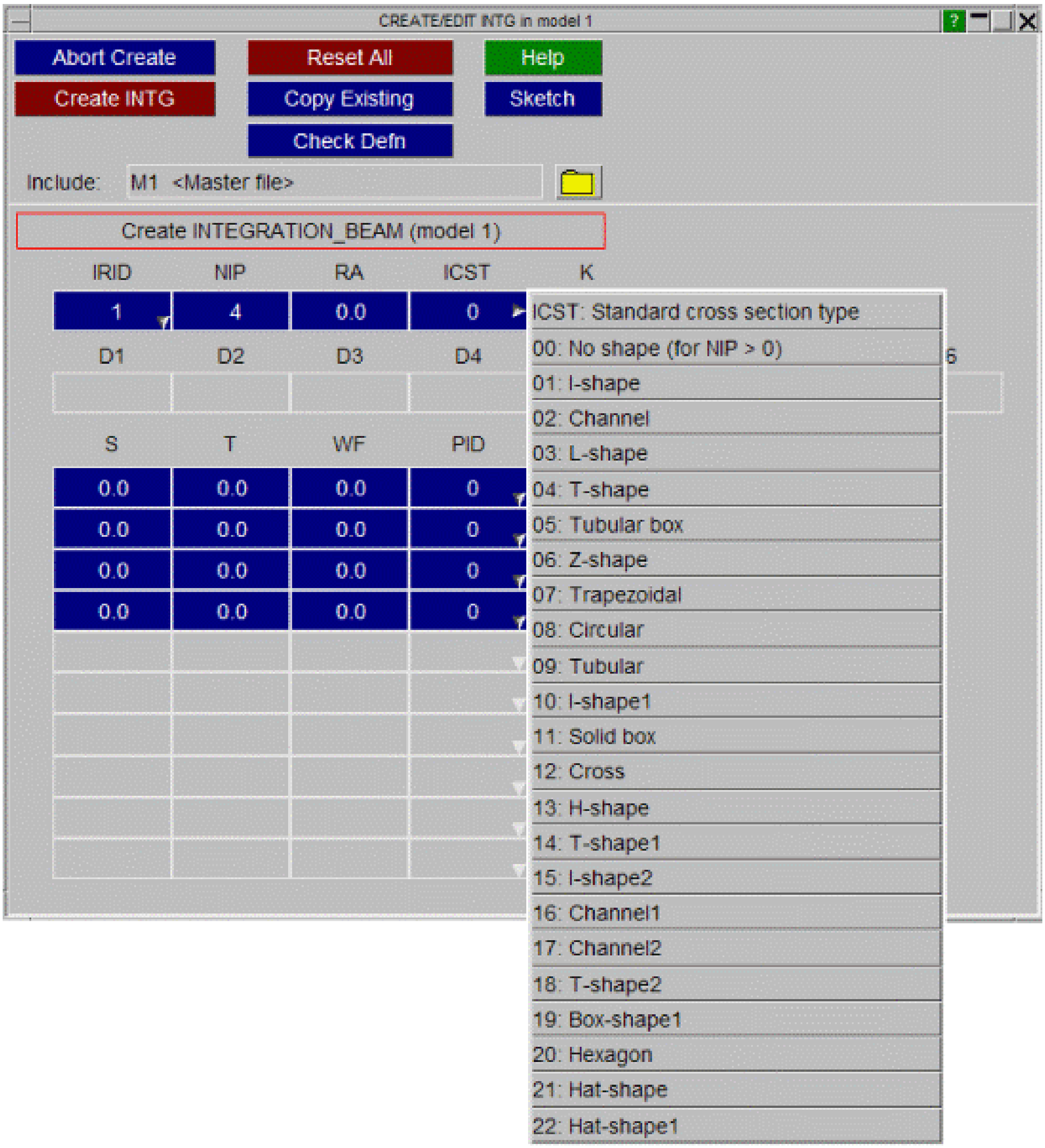
*INTEGRATION_BEAM



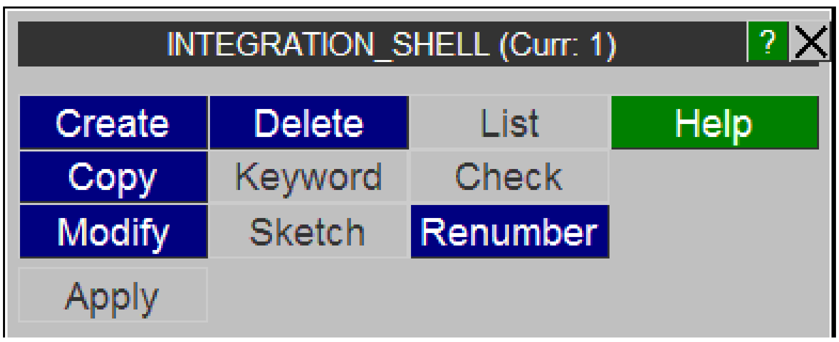
Enter the field to define the NIP for the INTEGRATION_BEAM, and ICST will be set to zero and the rows for each layer will appear below.



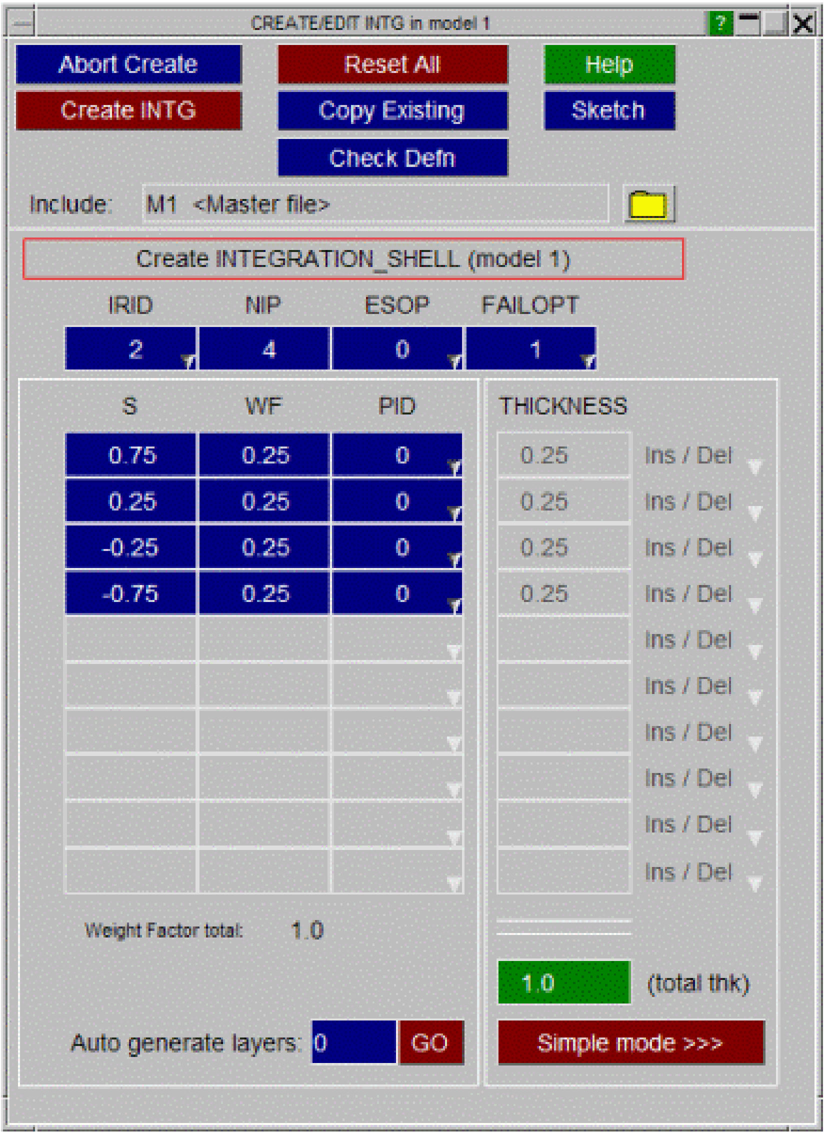
If the user wants to have a predefined shape, select the ICST type from the popuip and the NIP will be set to zero automatically.



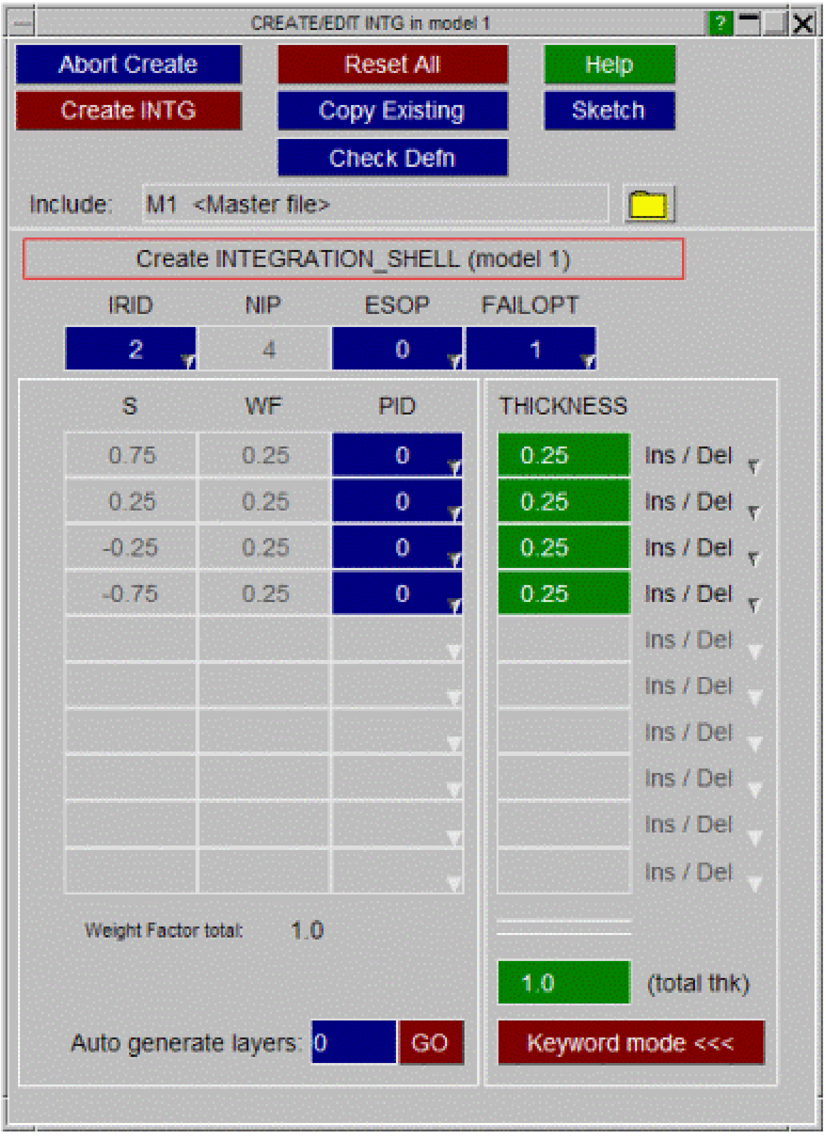
*INTEGRATION_SHELL



The standard panel is below, the user has to define the NIP and in each row below the weighting factor and coordinate have to be input manually. There is an **auto-generate layers** option that generates equally spaced layers for the user.



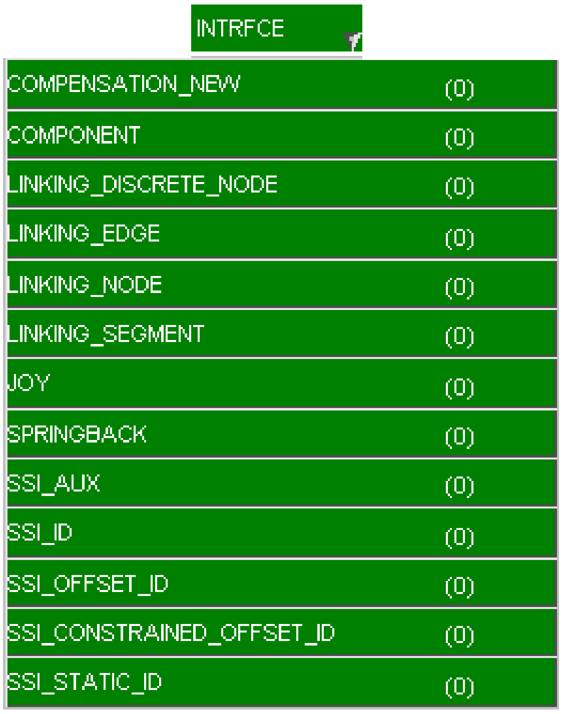
If the user wishes to have a more intuitive input panel, then select the **simple mode >>>** button. The following panel is displayed. It allows the user to define an overall thickness and insert or remove layers through the popups shown. The layers can be defined with thicknesses rather than a coordinate, and the panel automatically calculates the coordinate and weighting factor. Select **keyword mode <<<** to return to the basic panel.


CREATE/EDIT INTG in model 1
Abort Create
Reset All
Help
Create INTG
Copy Existing
Sketch
Check Defn
Include:
M1 <Master file>
Create INTEGRATION_SHELL (model 1)
IRID
NIP
ESOP
FAILOPT
2
4
0
1
S
WF
PID
THICKNESS
0.75
0.25
0
0.25
Ins / Del
0.25
0.25
0
0.25
Ins / Del
-0.25
0.25
0
0.25
Ins / Del
-0.75
0.25
0
0.25
Ins / Del
Weight Factor total:
1.0
1.0
(total thk)
Auto generate layers:
0
GO
Keyword mode <<<

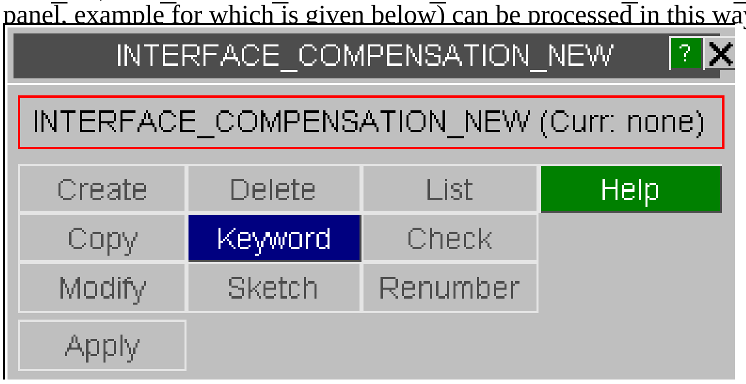
INTERFACE: Defining Interface Cards.

- Selecting the *INTERFACE sub-keyword
- Editing panels
- Special create/edit panel for **SSI_ID**

The ***INTERFACE** keyword has 13 sub-categories.
These can be selected from the drop down list available under **INTRFCE** keyword listing.

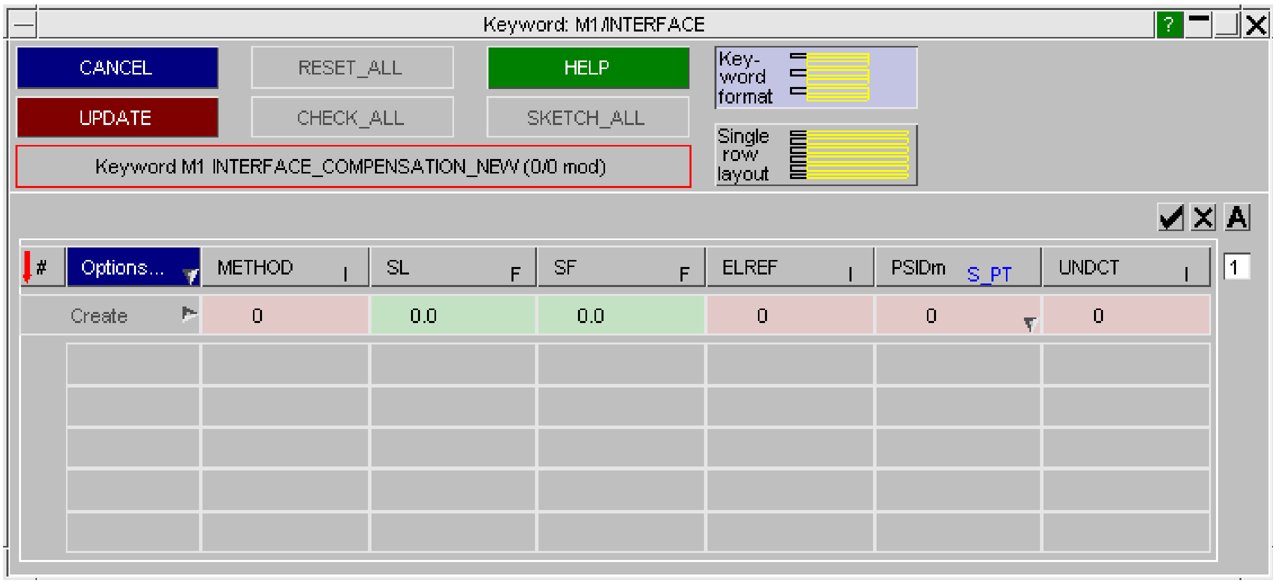


***INTERFACE** cards can be edited with the generic "Keyword" editor. All ***INTERFACE** sub-keywords except **SSI_ID**, **SSI_OFFSET_ID** and **SSI_CONSTRAINED_OFFSET_ID** (which have their own specific Create/Edit panel, example for which is given below) can be processed in this way.



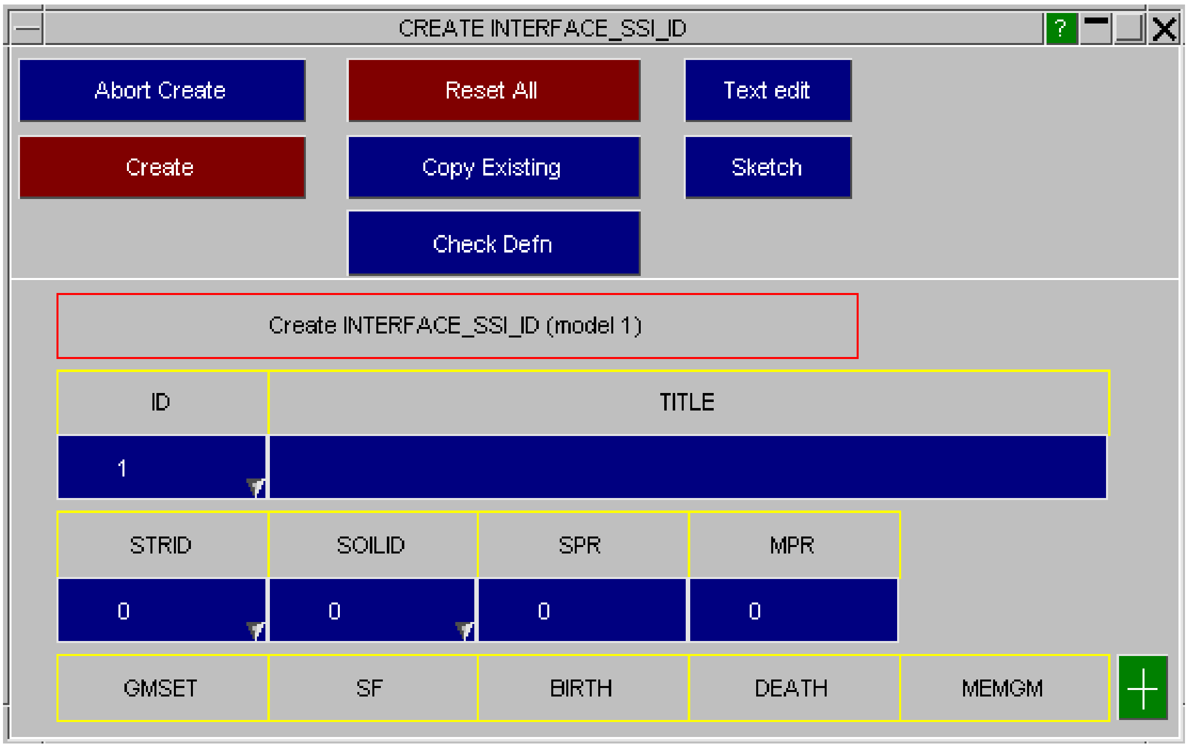
The other commands (**COPY, DELETE,** ...) function in the standard fashion as defined in section 5.0.1..

This shows an example of the Keyword editor for ***INTERFACE_COMPENSATION_NEW**.



INTERFACE_SSI_ID

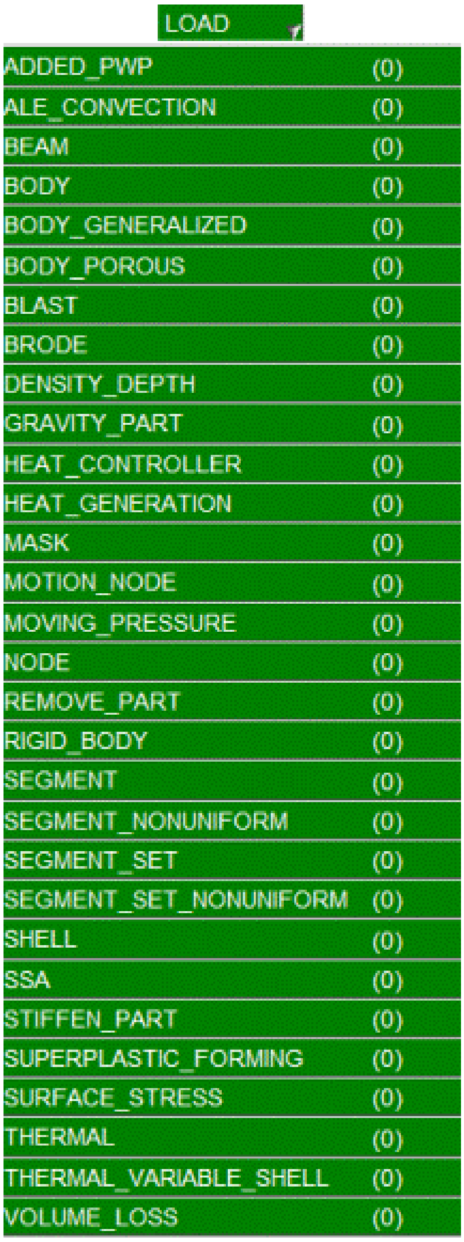
***INTERFACE_SSI_ID** items are created using a specific editing panel. New data rows can be added by clicking on the + button.



LOAD: Defining Loading Conditions.

- Selecting the *LOAD sub-keyword
- Visualisation

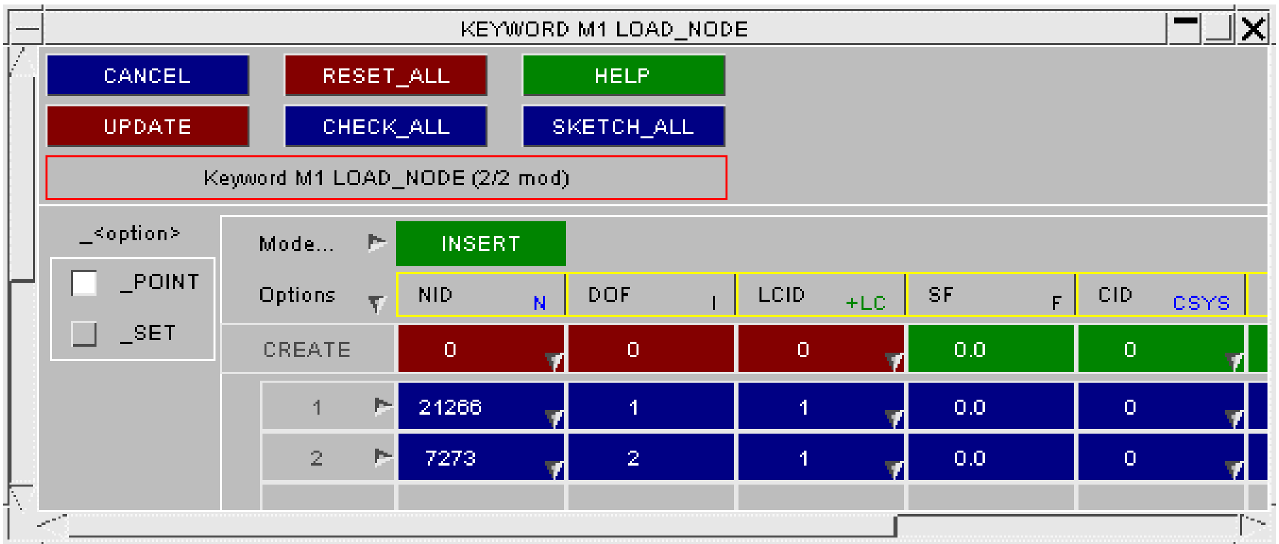
A range of different loading types can be defined in LS-Dyna
All *LOAD sub-keywords are editable within PRIMER.



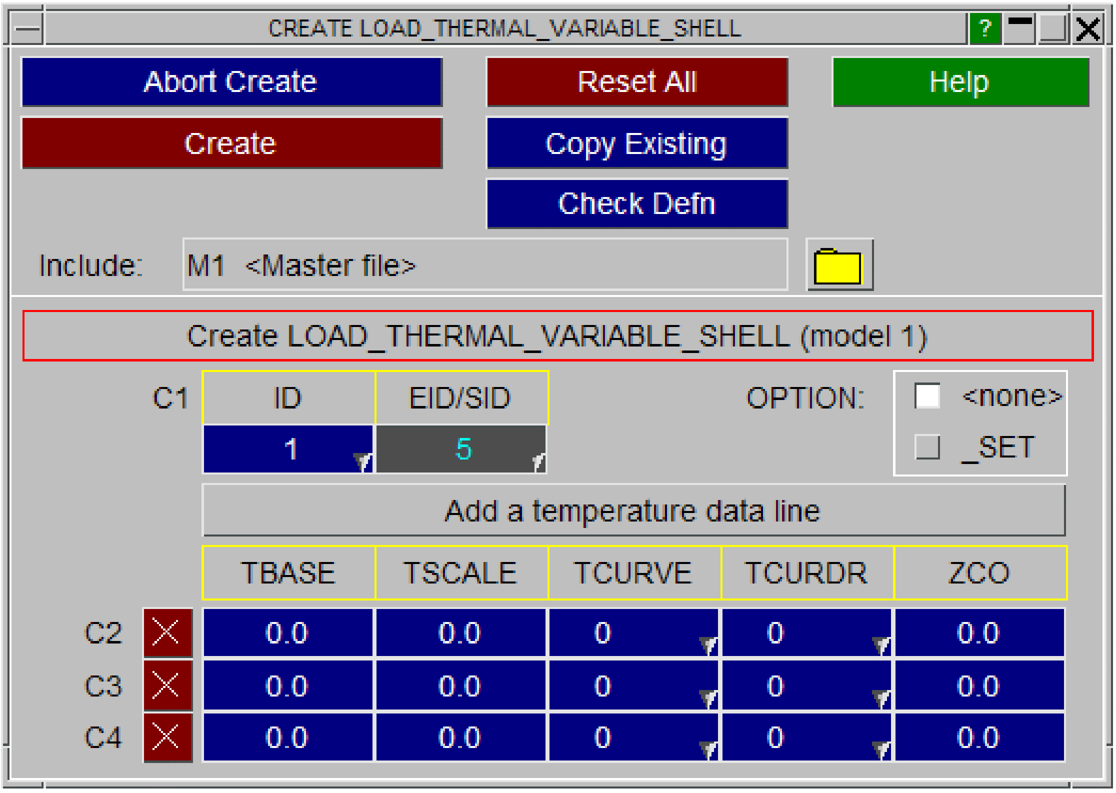
Most ***LOAD** cards can be edited only with the generic "Keyword" editor.

The other commands (**COPY, DELETE,** ...) function in the standard fashion described in section 5.0.1.

This shows an example of the Keyword editor for ***LOAD_NODE**. The **_POINT** variant has been chosen here.



Specific editing panels exist for ***LOAD_MOVING_PRESSURE** and ***LOAD_THERMAL_VARIABLE_SHELL** due to the nature of those keywords. These cards are similar. They both allow the user to add any number of additional lines of data per card. The ***LOAD_MOVING_PRESSURE** card is shown here.



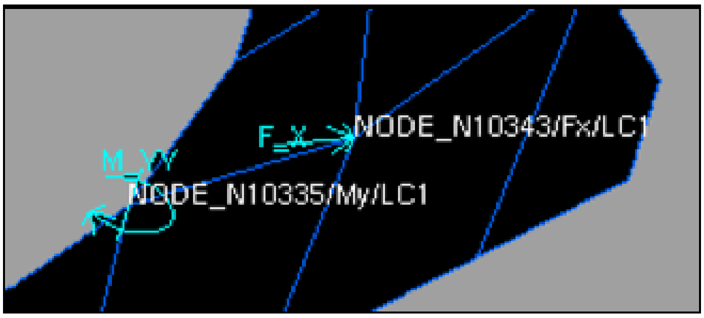
Additional rows of data can be added by clicking on the **Add a data line** button.

Visualising *LOAD items

***LOAD** items are not displayed by default, but can be selected for display and labelling in the **ENT**ity Viewing panel.

Only ***LOAD_NODE** symbols are drawn explicitly as arrows acting in the direction of the load/moment. Note that load symbols can be distinguished from ***INITIAL_VELOCITY** ones both by colour and because loads point to a node whereas velocities point away from it.

This example shows a force in X acting on Node 10343, and a moment about the YY axis acting on node 10335.



Other ***LOAD** sub-keywords are visualised only in terms of the sets, segments, elements or nodes upon which they act by turning on these additional items in **ENT**ity Viewing. Labels are generated correctly for all sub-types.

MATERIAL: Defining Structural and Thermal Materials.

The ***MATERIAL** keyword in LS-DYNA includes both structural and thermal materials, however they are treated separately within PRIMER since their roles are different:

- Structural materials are required for all structural analyses. (In keyword format all types of structural material, including those for discrete and seatbelt elements, form part of the same numbering sequence. The distinction between "structural", "discrete" and "seatbelt" material numbering found in formatted input decks does not exist.)
- Thermal materials are used for "thermal only" analyses, and may also be defined for combined structural and thermal analyses.

They use separate labelling sequences, so it is possible to have structural material #1 and thermal material #1.

STRUCTURAL MATERIALS

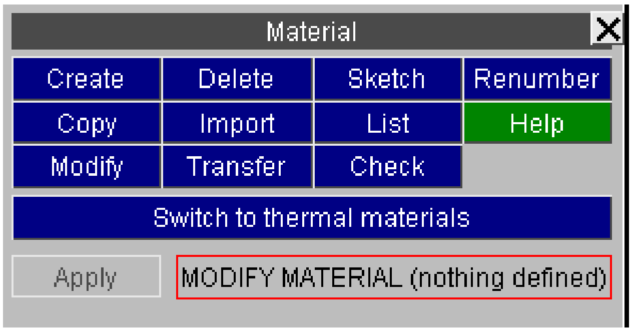
The structural material editing functions allow all structural material types to be processed.

PRIMER does not draw materials directly, but material labels may be appended to the graphics of structural items, see **VIS_1**; and Parts, Elements, etc may be selected by material for subsequent graphics operations such as **SKETCH** and **BLANK**

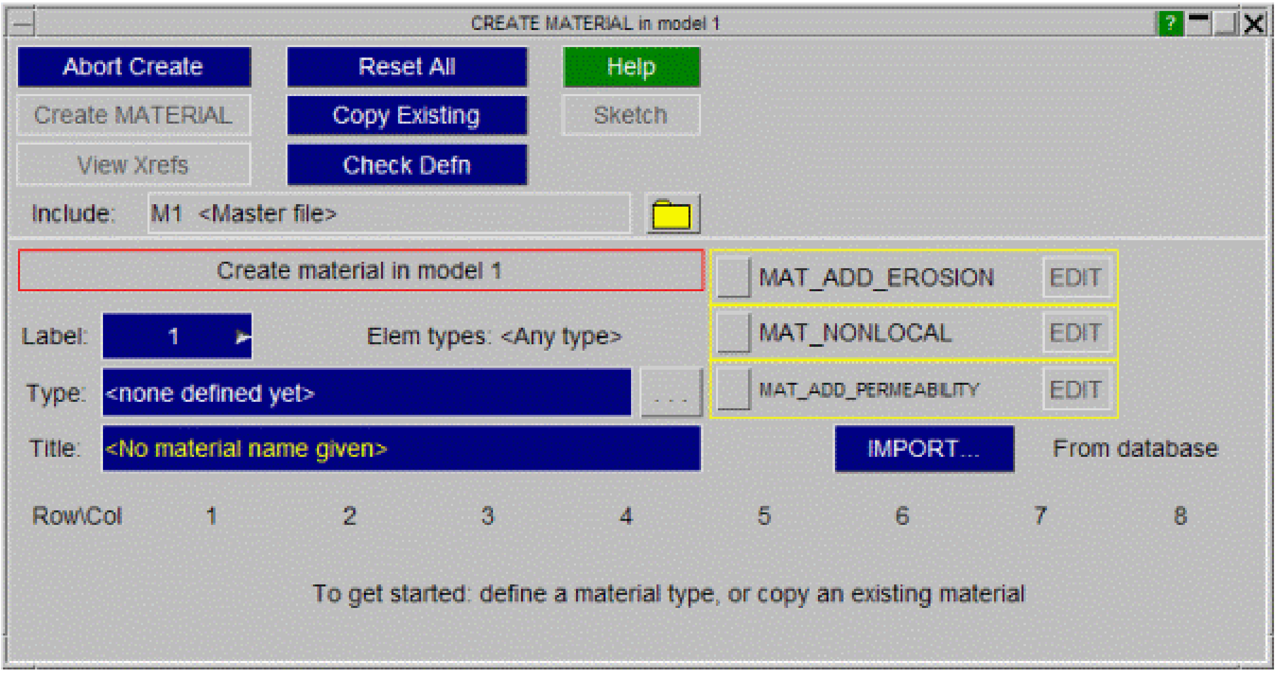
This figure shows the main structural material editing panel. **IMPORT** permits material definitions to be "imported" from databases of material definitions to populate undefined materials in a model.
TRANSFER opens the main window for the transfer data

function (see section 6.7 for more detail)
The other functions currently available have their standard meanings. (See 5.0.1).

Switch to thermal materials toggles the editing panel (for more detail on thermal materials see below).



CREATE Making a new material definition.



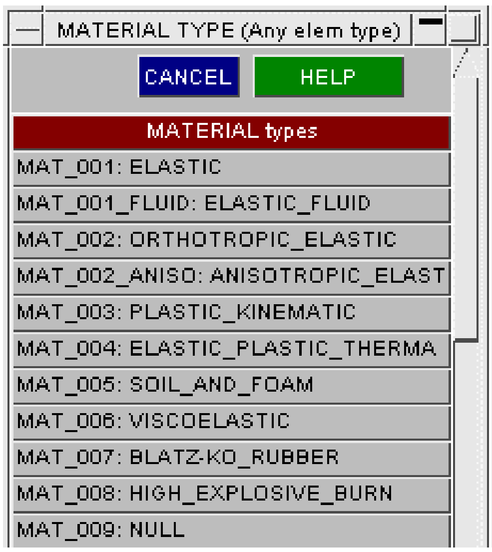
CREATE produces this blank material creation panel, since no material type has been defined yet.

Elem types:

Indicates the types of elements which are applicable to the currently defined material type.

This is a brand new material definition with no previous context, therefore **<Any type>** is shown. Had this material been created from a **PART** of known element type the relevant type would be shown here, and only materials valid for this element type would be selectable.

Type:



The material type can be defined from this button.

The [...] browse button can be used to browse through a list of material types as shown here.

Each material is listed with its LS-dyna material number.

Note on selecting a Material:

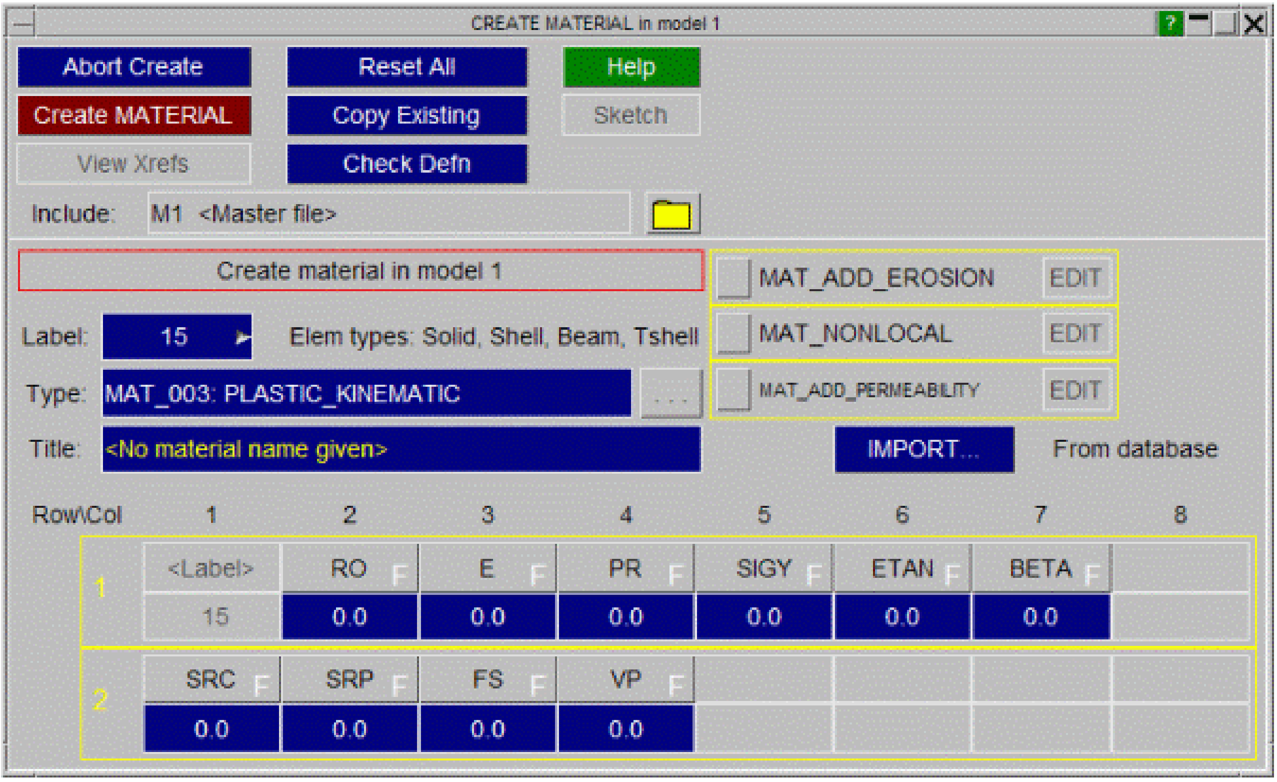
A Material may be selected by one of three ways:

- by invoking the shortcut button and selecting the material with the mouse from the list
- by typing in the material number to the "Type" box, e.g. "1" for **`*MAT_ELASTIC`**
- by typing in the material name to the "Type" box, e.g. "rigid" for "**`*MAT_RIGID`**"

ROW/COL

The data relevant to each material type is displayed in row and column format identical to that of DYNA keyword.

Once a material type has been defined the panel will become populated with that material's format. For example the type ***MAT_PLASTIC_KINEMATIC** has been chosen here:



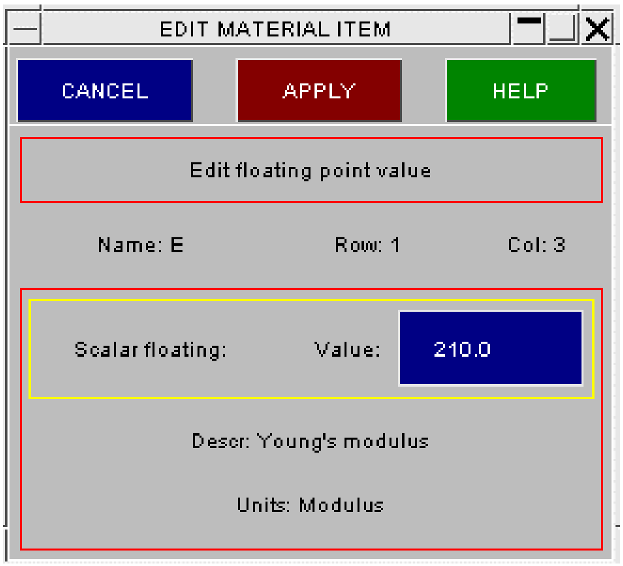
The material data can then be typed into the relevant boxes. The expected data type is indicated on the grey button, which also shows the acronym for that data value:

F	White **F** floating value
I	White **I** integer value
+LC	Green **LC** +ve loadcurve
-LC	Red **LC** -ve loadcurve

Information about each individual data component can be requested by pressing the grey data component button. For example; to request information about data component '**E**' (1^{st} row, 3^{rd} column) press the grey button with the E.

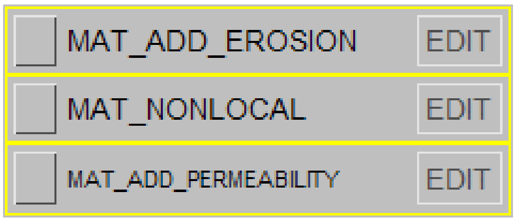
This will create a new window with detailed information about that data component showing:

- A one-line description of it;
- Its current units type
- Its current value.



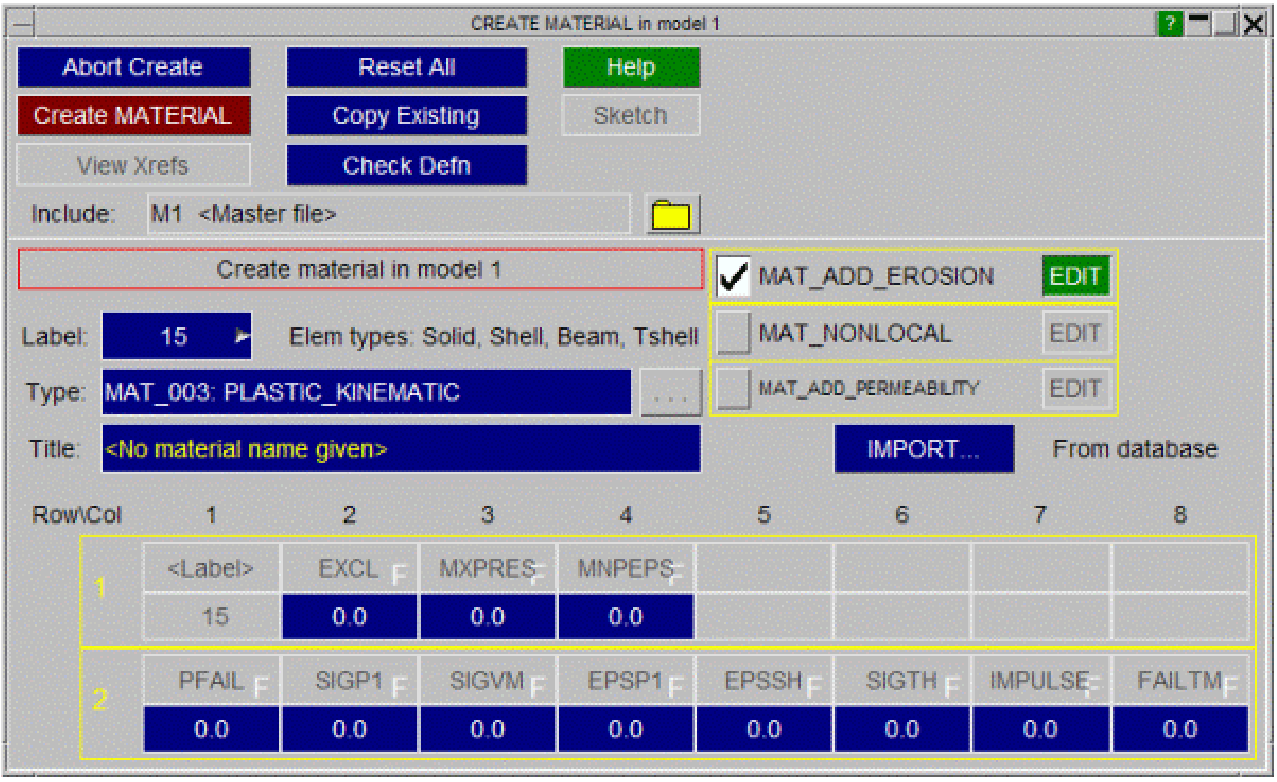
MAT_ADD_EROSION

From LS-Dyna version 950 onwards any structural material type can have "erosion" properties defined for it. This provides a range of failure parameters that can be used to delete ("erode") the elements using the material.



By default erosion properties are not defined for a material, and this option defaults to **Inactive**.

If it is made **Active** then you can **EDIT** it:



***MAT_NONLOCAL** and ***MAT_ADD_PERMEABILITY** are also available to edit in the same way as ***MAT_ADD_EROSION.**

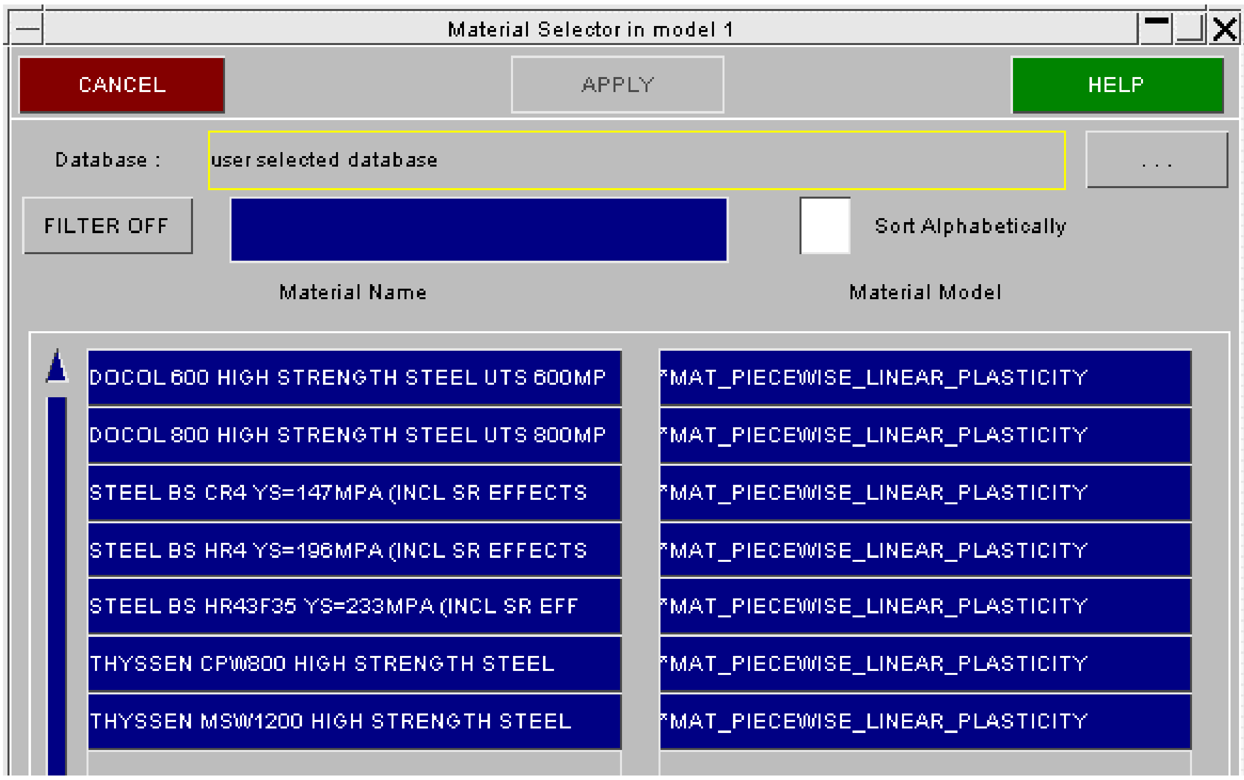
IMPORT... Importing pre-defined material data from a database
Material data can be stored in a database, and "imported" to populate a definition.

The information imported is:

- All the data fields in the material definition, except for its label, are overwritten.
- If the material in the database is of a different type to the current material, the type is also changed.
- If the database definition refers to loadcurves (***DEFINE_CURVE**) or tables (***DEFINE_TABLE**) then these will be imported too. They will be given labels <highest current curve/table number + 1> onwards.

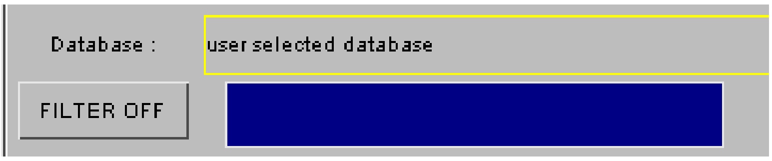
Note: **IMPORT** in this context differs slightly from the same command in the top material panel:

- Here: it imports data unconditionally for a single material definition.
- In the top panel it imports definitions for a range materials by matching up the names of the database definitions against those in the model.



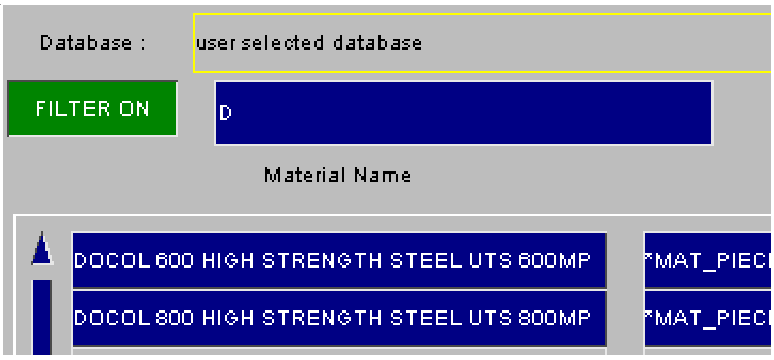
Here is an example of a database containing seven materials. The stored Material Name and the LS-Dyna **Material Model** type for each are listed.
To import a material click on its **Name** or **Model** definition, either will do, and press **APPLY**. This will overwrite the definition in the current create/edit panel with the imported data (only the label is left unchanged).

FILTER OFF/ON Filtering the material list
By default the filter is **OFF**, and all materials in the database are shown.



If the filter is **ON** then only those materials with names containing the character string given will be shown.
In this example "D" has been chosen, restricting the list to just two materials.

The filter is case-sensitive: "D" and "d" are treated as distinct.



Sort Alphabetically
By default materials appear in the order in which they are defined in the database.

Turning **Sort Alphabetically** on sorts them alphabetically by name. The sort is not case-sensitive.

Once all of the data has been input on the material card, press **CREATE_MATERIAL** to install the material permanently in the model.

As with other creation/editing functions a standard check is made of the new definition prior to saving it, and you are warned about errors found. However note that comprehensive checking of materials is nearly impossible, and for most types PRIMER simply ensures that a positive density has been used.

MAterial database format and instructions for setting up a database are given in appendix 9.

COPY Copy existing material(s) to make a new material(s).

COPY makes new materials, in the same model(s), that are identical to their originals apart from their labels. By default only the material definitions themselves are duplicated.

Where **RECURSIVE COPY** is requested, all items associated with that material (i.e. elements, parts, etc.) are also copied.

MODIFY Modifying the attributes of an existing material.

MODIFY functions in the same way as **CREATE**. Obviously, the material type will already have been selected so the panel will resemble that shown in "populated" figure above.

If the material is in use by a `PART` which has elements then the "element type" of the material is locked to those elements, which will restrict the range of material types you can change it to. For example a material used by springs cannot be changed to a shell type.

Any modifications made to the material definition will not be made permanent until the **APPLY_MODIFY** button is pressed. At this point a the local copy which has been updated is used to overwrite the version in the model.

DELETE Delete existing material definitions.

The **DELETE** function deletes the selected materials. However you cannot delete a material that is in use by a `PART` unless you remove it from the part definition, or delete that too. To help with this the following two switches may be used:

DELETE_RECURSIVE Will select for deletion the parts, associated elements, and so on that reference this material.

REMOVE_FROM_SETS Is often also needed if parts are to be deleted, as their elements, the connected nodes, and often the parts themselves may be included in sets.

A good way of getting rid of surplus (unused) materials is to turn these two switches off, then select all materials for deletion. Only those which are not used by anything will actually get deleted.

IMPORT Matching up and restoring material definitions from a database.

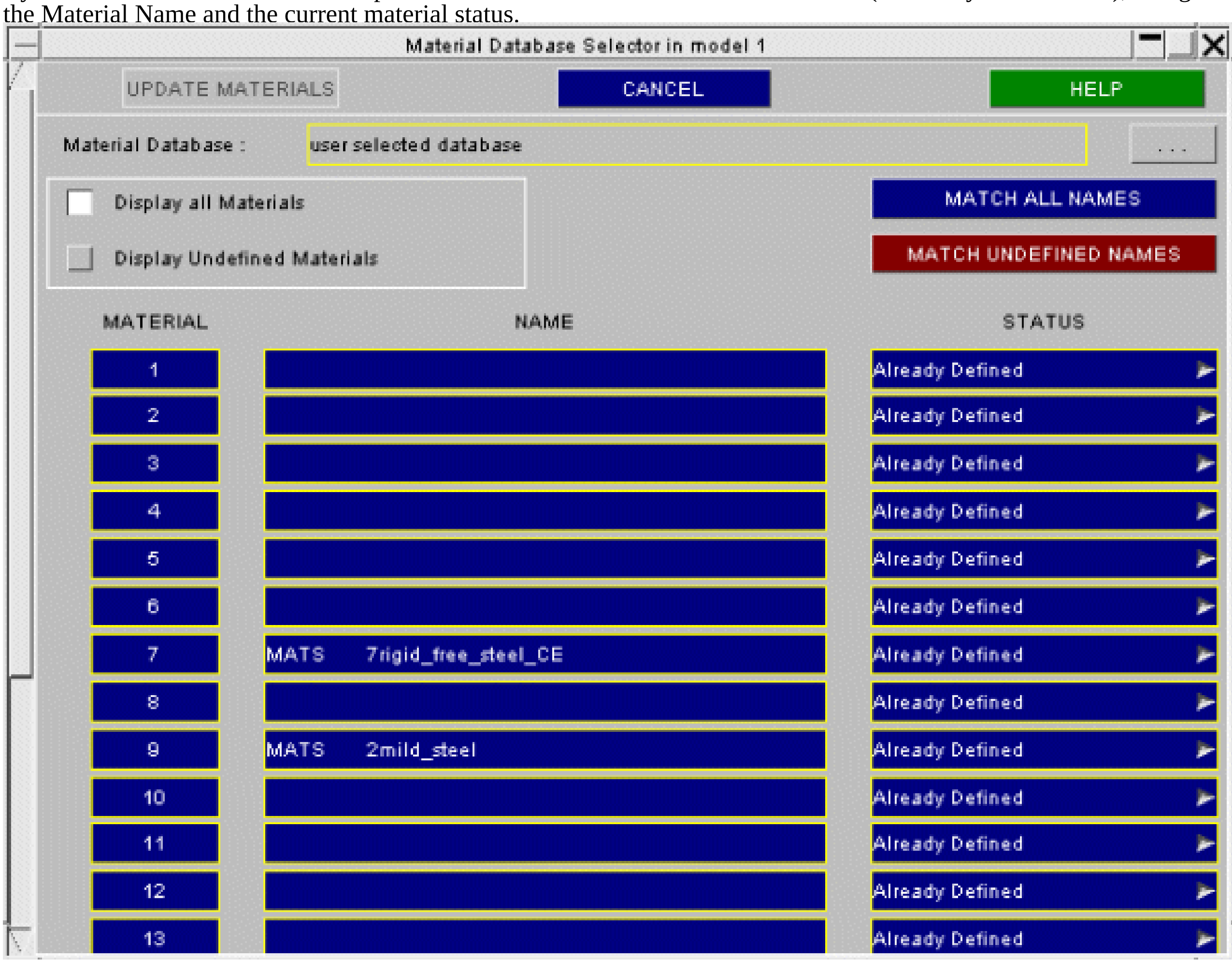
When a LS-Dyna model is taken into a 3rd-party pre-processor for modification it often loses most of its material definitions, as they are too complex for "general" software. This frequently happens when models are partially remeshed.

A model that comes back into PRIMER from such a source is referred to as being "stripped" of its materials.

This option allows you to maintain a separate database of material definitions, and to "marry" them up with the "stripped" model by matching their names.

This figure shows the material **IMPORT** menu.
At the top of the menu the current material database is displayed along with an option to select an alternative material database. If this option is greyed out then PRIMER has been unable to locate any material databases.

By default the menu contains a complete list of all the materials the model contains (sorted by Material ID), along with the Material Name and the current material status.



MATCH ALL NAMES

This option will search through **ALL** of the materials in the current model (ignoring the current material status) and will attempt to match them with the material definitions in the current material database.

MATCH UNDEFINED NAMES

This option will search through only the <<Undefined>> materials in the current model and will attempt to match them with the material definitions in the current material database.

If the current material database is changed after selecting either of the 2 options above and then the MATCH ALL NAMES option is selected then any material that was matched in the first database and also matches an entry in the second database will be replaced with the entry from the second database.

Material Names

The material name displayed for each material can come from a number of sources.

If the model being edited has been read in from a KEYWORD input deck then the material name is taken from one of the following sources (listed in order of preference)

- *`MAT_..._TITLE` card (LS-960 input files and above)
- `$HMNAME` - Hypermesh material name comment card.
- `$PRTITLE` - PRIMER material name comment card.

If the model has been read in from an IDEAS Master Series Universal file then the material names are taken from the material database cards in the universal file.

Material Status

Each material is displayed using one of the following statuses.

<< Undefined >>	The material is a Latent material without any material properties defined for it yet. (RED)
Automatically Found in Database	The material has been automatically matched with a material in the current material database. (GREEN)
User Selected From Database	A material definition in the current material database has been selected by the user for this material. (BLUE)
Defined by User	The material definition has been edited by the user. (BLUE)
Already Defined	The material is already defined in the model and has not been modified by the user. (BLUE)

In addition to the material status the entries in the list are colour coded using the colours above.

Automatic Material Import

The Material name can be used to automatically locate a material definition in the current material database. To locate a material in the database the following rules are applied.

- The material name in the current model is converted to upper case.
- The name is compared to each name in the material database (also converted to upper case) to see if there is an exact match. If the name matches more than one entry in the material database then the first entry in used.
- If none of the database names match exactly then a second search through the database is carried out to see if the material name is a subset of one of the names in the database of if the database name is a subset of material name.

If for example the current material database contained the following names

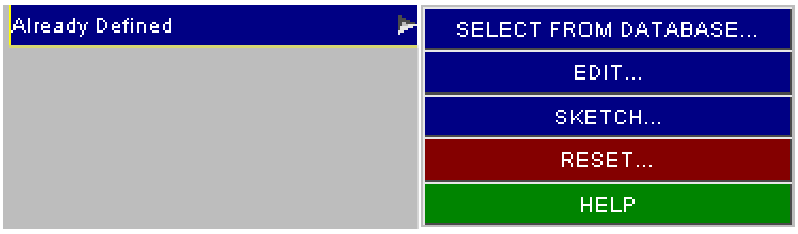
Database Names	: STEEL H350 - 1
	: STEEL H350
	: STEEL H420

Then the following materials would be matched as follows

Material Name	Database Material	
STEEL H350	STEEL H350	*(Exact match)*
NEW STEEL H350	STEEL H350	(Database subset of material name)
STEEL H	STEEL H350 - 1	(Material name subset of database 1st match)

Manual Material Import

In addition to automatically selecting a material from the database the user can manually select a material from the database by using the POPUP menu attached to the status button of each material.



SELECT FROM DATABASE... This option will display a list of all the materials in the current material database.

EDIT... This option will bring up the standard create (undefined material) or edit material panel.

SKETCH Sketch elements using a material on the current image.

SKETCH sketches on top of the current image the parts and elements that reference the selected materials.

LIST Summarise the attributes of selected materials.

LIST allows the user to individually select materials and display a summary listing of their attributes.

CHECK Check selected materials for correctness

CHECK runs the standard checking function on the selected materials, summarising any errors.

WARNING: Checking materials thoroughly is a mammoth task, which PRIMER does not at present attempt. Most structural materials are currently only checked for a positive density. Therefore that a material "Checks OK" does not mean that it contains no errors!

RENUMBER Renumbering material labels.

RENUMBER lets you change any or all material labels within a given model using the standard renumbering panel.

To change the label of an individual material it may be simpler just to **MODIFY** it.

`END_STRUCTURAL_MATERIAL` terminates material processing.

THERMAL MATERIALS

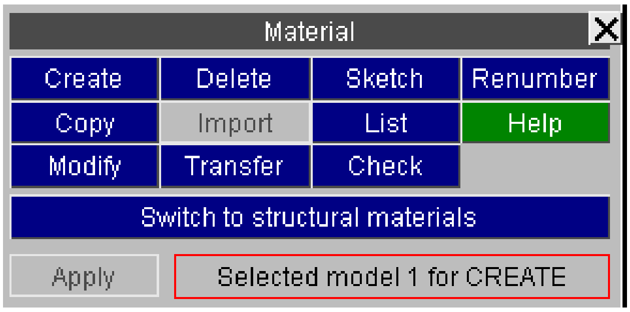
The thermal material editing functions allow all thermal material types to be processed.

This is done in exactly the same way as structural materials, save that:
- The panels are somewhat simpler, reflecting the less complex nature of thermal materials.
- The concept of "Erosion" does not exist for thermal materials.
- There is no database "Import" capability for thermal materials.

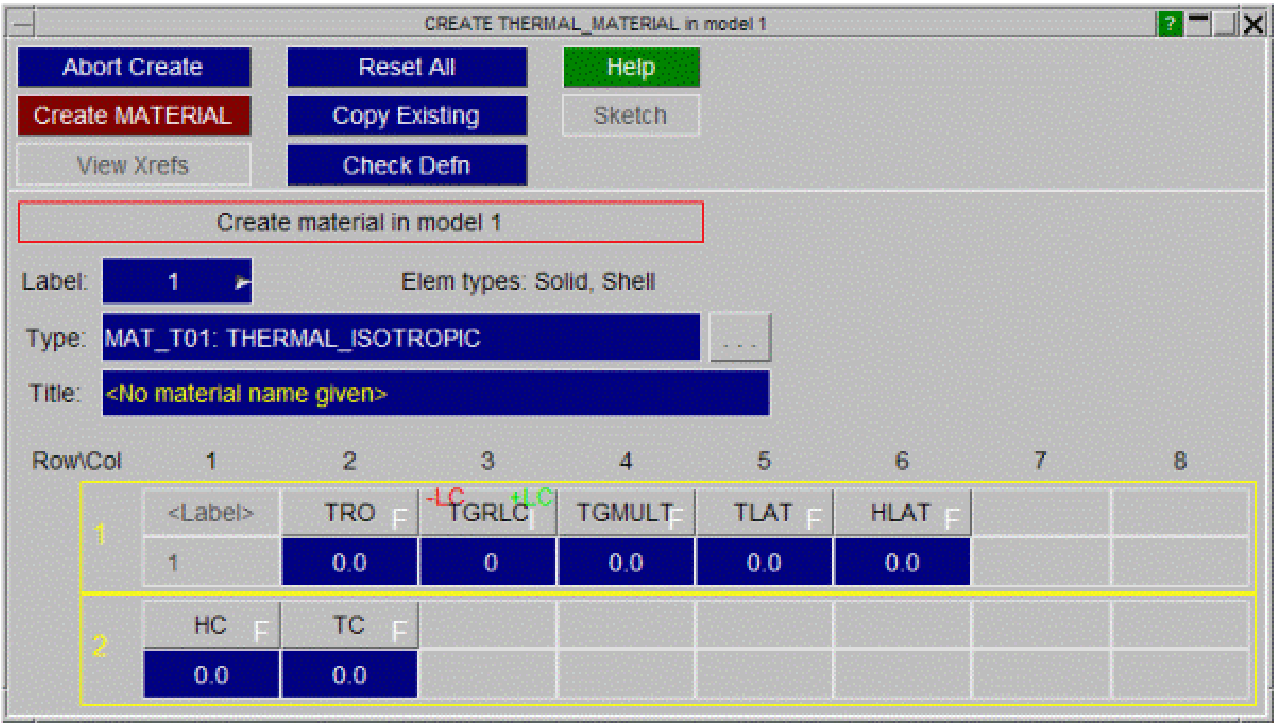
PRIMER does not draw thermal materials explicitly. Like structural materials they can be **SKETCH**ed by drawing the parts and elements that reference them.

This is the main panel for thermal material editing. The functions currently available have their standard meanings. (See 5.0.1).

(There is no **IMPORT** facility for thermal materials.)



This is a typical thermal material editing panel. It functions in exactly the same way as structural materials.



Visualisation of materials

Materials cannot be drawn explicitly (there is no **ENT**ity Viewing setting for them), however they can be visualised in the following ways:

- By a **SKETCH** command in the top menu or editing panels above;
- By selecting **MATERIAL** as the native colour for part-based elements: see "controlling the colour of elements".

Selected material properties can also be contoured: see section 4.2: "Data plotting commands"

NODE: Defining Nodes.

[Top level menu](#)
- Creating
- Copying
- Editing
- Deleting

Visualisation
Labelling

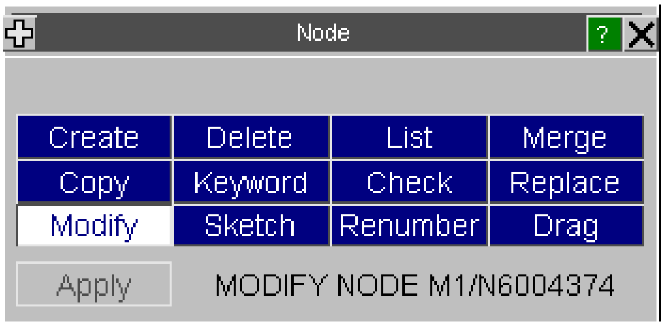
Screen-picking

Nodes ("grid points") form the vertices of elements and wide range of other structural purposes: they are the "glue" that holds a finite element model together. Within an LS-Dyna analysis virtually all mass is lumped at nodes, providing a simple mass vector that doesn't require matrix inversion: the basis of the "explicit" solution method.

Special rules apply to the eligibility of nodes for screen-picking: they do not have to be drawn explicitly to be pickable: see the "Rules for screen-picking nodes" below.

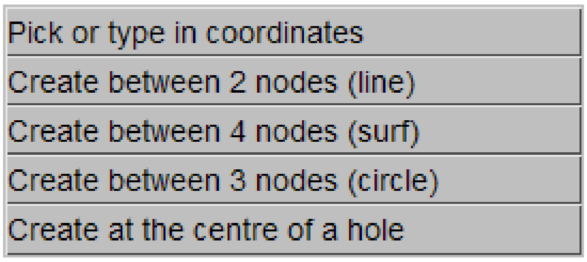
The nodes menu allows the creating, modification etc. of nodes in a keyword deck.
The functions currently available have their standard meanings. (See section 5.0.1)

Generic keyword editing is also available.



CREATE Making new node(s).

There are five possible ways of creating nodes. These are selected by right clciking below **Method**. The following options are available



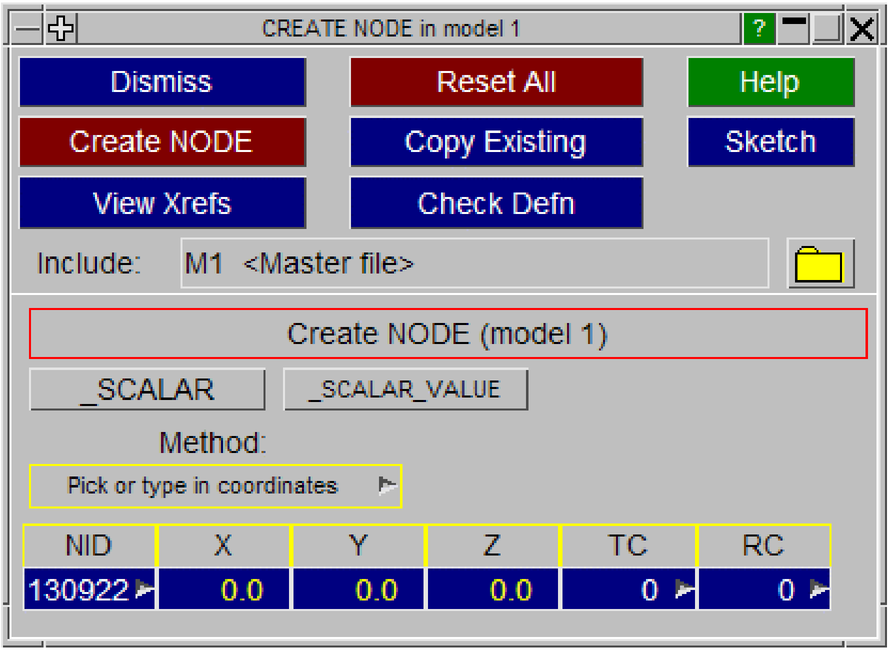
(1) Pick or type in coordinates

This figure shows the initial state of the nodes creation panel.
The default node label used is the highest node label in the model + 1.

The default coordinates used are (0.0, 0.0, 0.0).

Picking a node from the screen will set the X, Y and Z fields for the node you are creating to be the coordinates of the picked node. Alternatively you can just type in a new value for the X, Y or Z coordinate in the boxes.

The node label can be changed by typing in a new value or using the popup. If needed translational and rotational restraints can be applied by typing the value into the TC and RC fields or by using the popups.



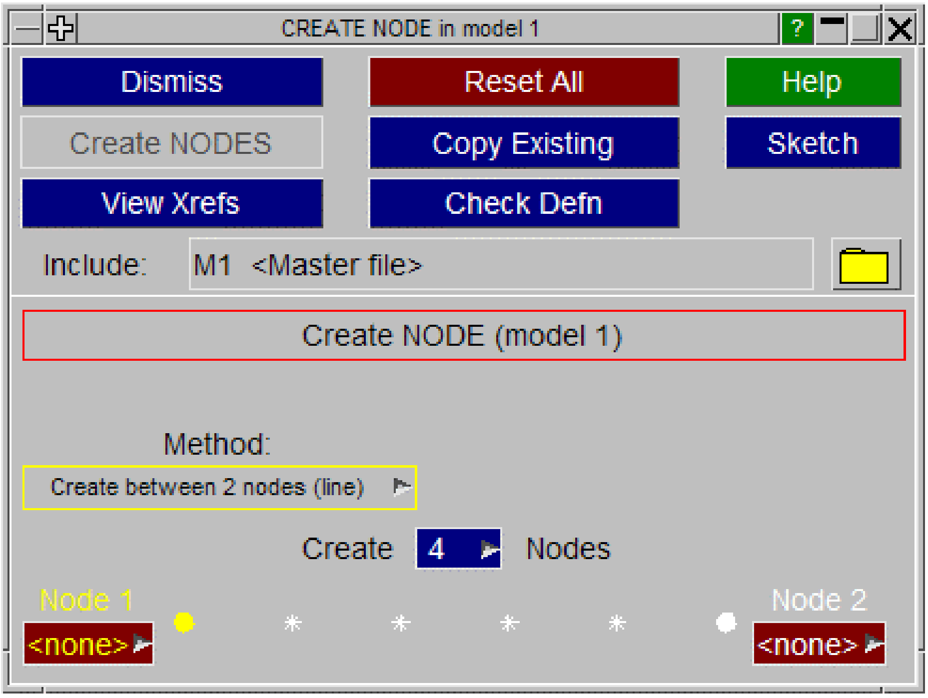
(2) Create between 2 nodes (line)

Instead of creating a single node, you can create any number of nodes in a line between 2 existing nodes. Initially the two end nodes are undefined so will be shown as <none> on a red background. Picking a node from the screen will update one of the two end nodes. The one which will be updated is shown in yellow instead of white, so in the figure to the right, Node 1 will be updated if a node is picked. Once this is picked then node 2 will be highlighted and can be picked from the screen. The two nodes can alternately be picked from the screen in this way.

Alternatively you can type in a node number or use the popups to create, select, sketch etc a node.

Any number of nodes can be created between the 2 end nodes. Either use the popup or type in a number to select how many to create. In this figure 4 nodes will be created.

Once both end nodes have been defined the **CREATE_NODES** button will become active and can be used to create the nodes. The display will then refresh for you to create another line of nodes. Once you have finished **DISMISS** will close the window.

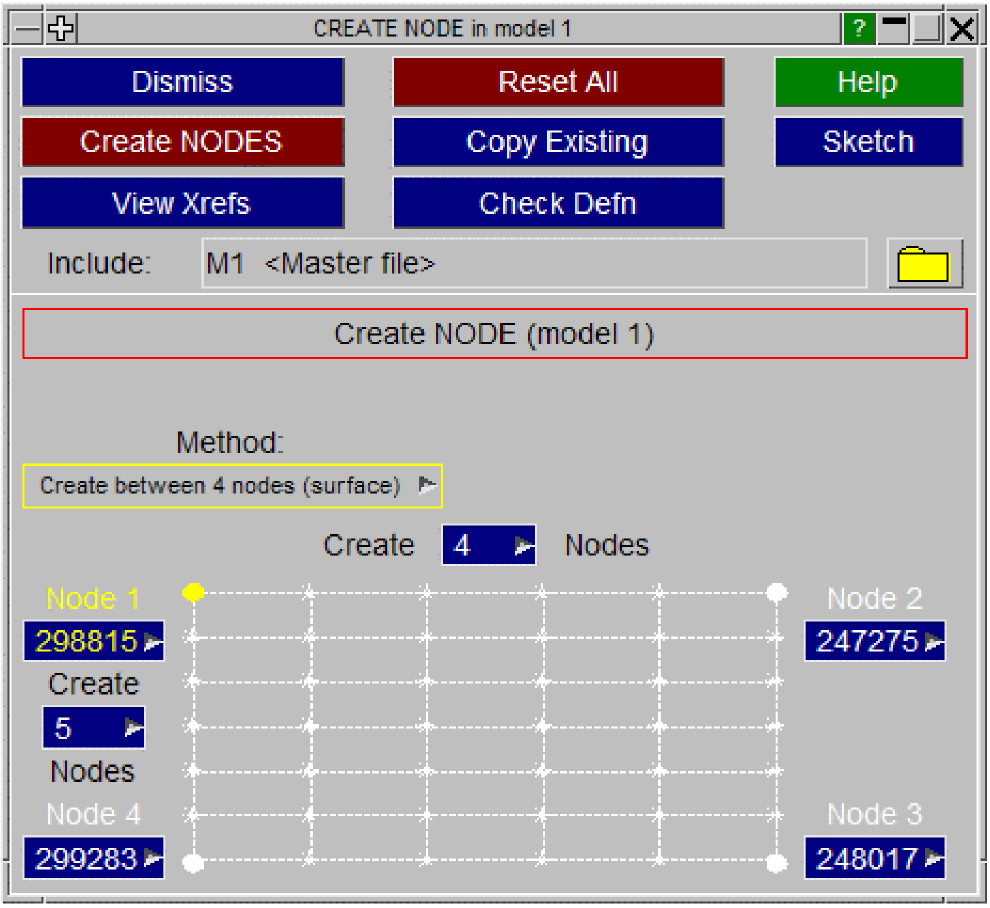


(3) Create between 4 nodes (surf)

Creating a surface of nodes works in an identical way to creating a line of nodes except that 4 nodes need to be defined

(one at each corner) and the number of nodes to create can be varied in both directions.
In this figure node 1 is the node currently highlighted for picking, and 4 nodes will be created in one direction, 5 in the other.

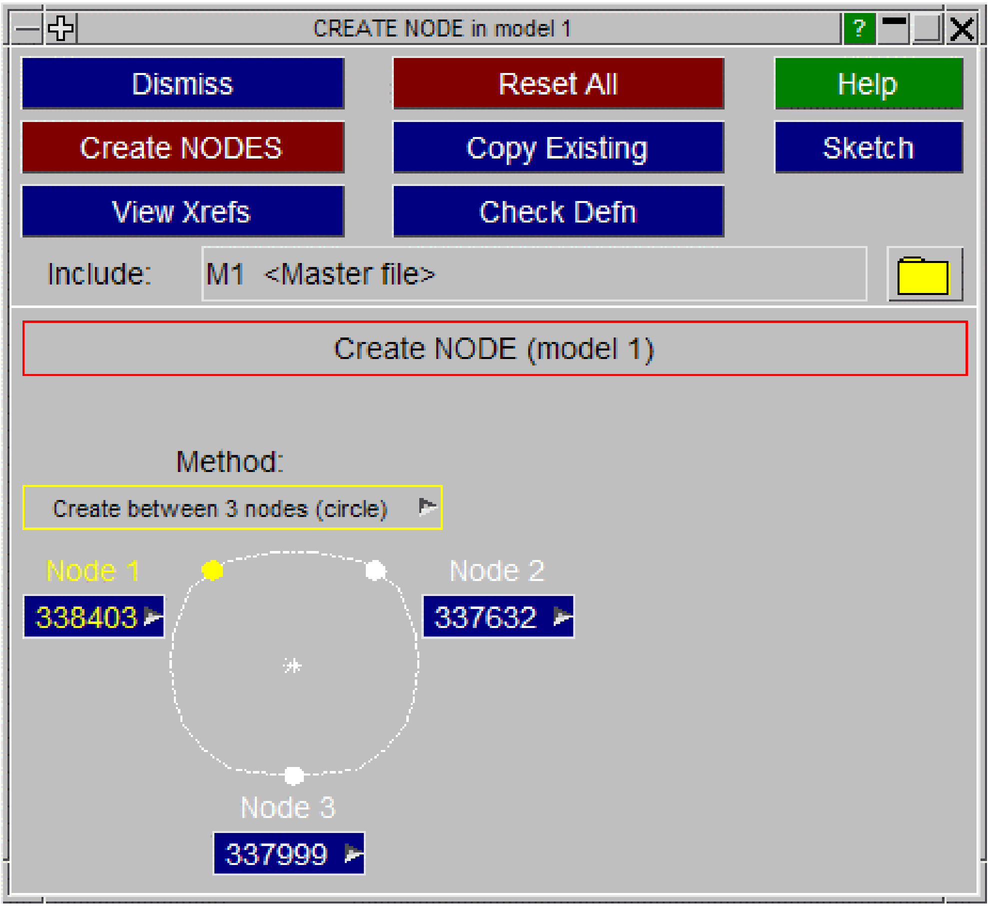
The nodes do not need to be on a plane. If the nodes are not then the nodes will be generated on a curved surface between the 4 nodes.



(4) Create between 3 nodes (circle)

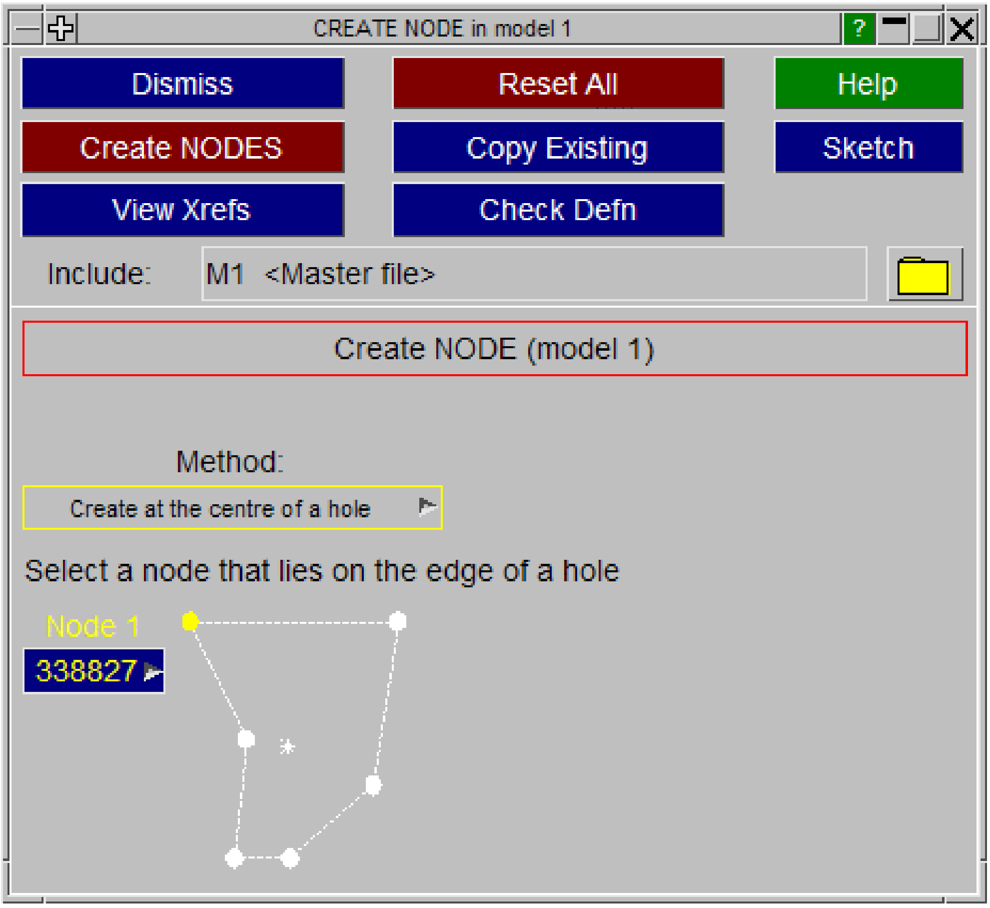
Creating a node at the centre of a circle works in an identical way to creating a line of nodes except that 3 nodes need

to be defined.
In this figure node 1 is the node currently highlighted for picking. The 3 nodes define a circle, and the node created will be at the centre of that circle.



(5) Create at the centre of a hole

This method of creating a node only requires the user to select one node. This node must be on the free edge of a hole. Primer will determine the centre of the hole and create the node there when clicking on **CREATE_NODES**.



Other node creation commands:

DISMISS	Aborts the current definition and returns to the main nodes menu.
RESET_ALL	Resets all attributes to <null> for this definition: all data entered will be lost, and the panel will return to its initial default state.
COPY_EXISTING	Copies the attributes of an existing node definition (in the current model). This may then be modified as required.
SKETCH	Sketches the current definition on top of the current image.
LIST_XREFS	Lists everything that references the current node definition.
CHECK_DEFN	Performs a check of the current definition, listing any errors.
CREATE_NODE	Saving the node definition. Once you have entered the node information the **CREATE_NODE** button will save this definition. The definition will be checked and any errors listed, an then it will saved permanently in this model. Until you press this the definition remains volatile, and will be lost if you exit this panel in any other way. Once the node(s) has been defined the panel will refresh with the default values to speed up node creation.

COPY Copying existing nodes(s) to make a new one(s).

You can **COPY** any number of nodes, in multiple models.

When **APPLY** is pressed you are asked to confirm what is to be copied, and then the operation is carried out.

For each model the <n> extra nodes chosen in that model are copied using labels <previous highest + 1> to <previous highest +n>, there is currently no control available over the new labels assigned.

MODIFY Modifying the attributes of an existing node.

This functions in exactly the same way as **CREATE**, using the same panels as in the figures above. The only difference is that the initial state of the panels is already set with the attributes of the node to be modified. In the modify mode the options to create nodes along a line or surface are not available. Only a single node can be modified.

If you want to change values on multiple nodes then you should use the **KEYWORD** option instead

DELETE Deleting existing nodes

The **DELETE** operation works exactly the same way as **COPY** above, except that the chosen nodes are deleted.

- If **DELETE_RECURSIVE** is switched on any loads, constraints etc. referenced by the nodes to be deleted are marked for deletion.
- If recursive deletion is not used only the node definitions themselves are removed.

Note also that the standard deletion rules described in Section 6.4.1 still apply: nodes will only be deleted if nothing else (which is to remain) depends on them.

KEYWORD Generic keyword editor

KEYWORD starts the generic keyword editor which allows creation, deleting and modification of multiple nodes. This is useful for modifying multiple nodes in a single operation.

SKETCH Sketch the chosen nodes on the current image

SKETCH allows the user to select and sketch individual nodes on the current graphics image. Nodes are drawn with a star symbol.

CHECK

Runs the standard checking function on the selected nodes. Each node will be listed either as "OK", or a summary of the errors encountered will be printed. (This is the same as the **CHECK_DEFN** command during node editing.)

LIST

Writes a summary list of the selected nodes to the screen. This is just the total number of selected nodes in each model.

RENUMBER

Raises the standard renumbering panel for nodes in the chosen model, allowing you to renumber some or all of them.

MERGE

MERGE allows you to merge coincident nodes (or nodes within a specific tolerance) together. For more information see the **MERGE NODES** option in the **REMOVE** window.

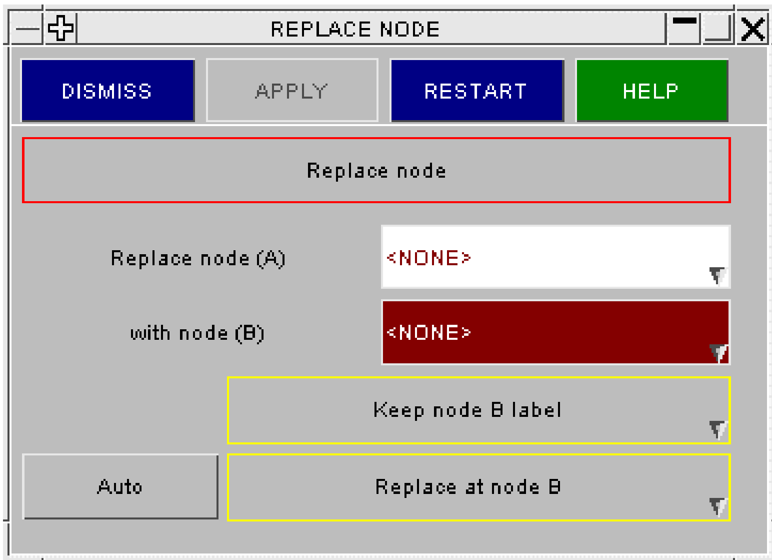
REPLACE

REPLACE works in an identical way to **MERGE** but only allows you to replace a single node with another node.

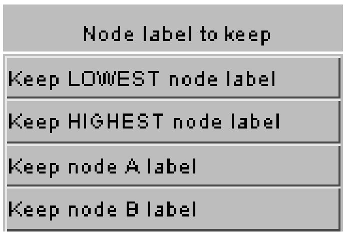
'quick picking' is enabled in the replace node panel so you can just click on the screen to select the 2 nodes. The node which you are currently picking is shown by the colours being inverted (i.e. in the figure on the right, node A is currently being picked).

Alternatively you can type in the node numbers or use the popup to select the nodes

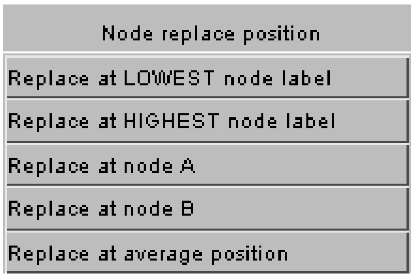
If **Auto** is selected then the node will be replaced as soon as both nodes are given. Alternatively press **APPLY** to replace the node.



By default the label of the second node you pick (B) will be kept. You can change this by using the popup. You can choose to keep either label or the highest or lowest label.



By default the node will be replaced at the location of the second node you pick (B). This can be changed by using the popup. You can force the node location to be at the position of either node A or B, the average position, or the position of the node with the lowest or highest label.



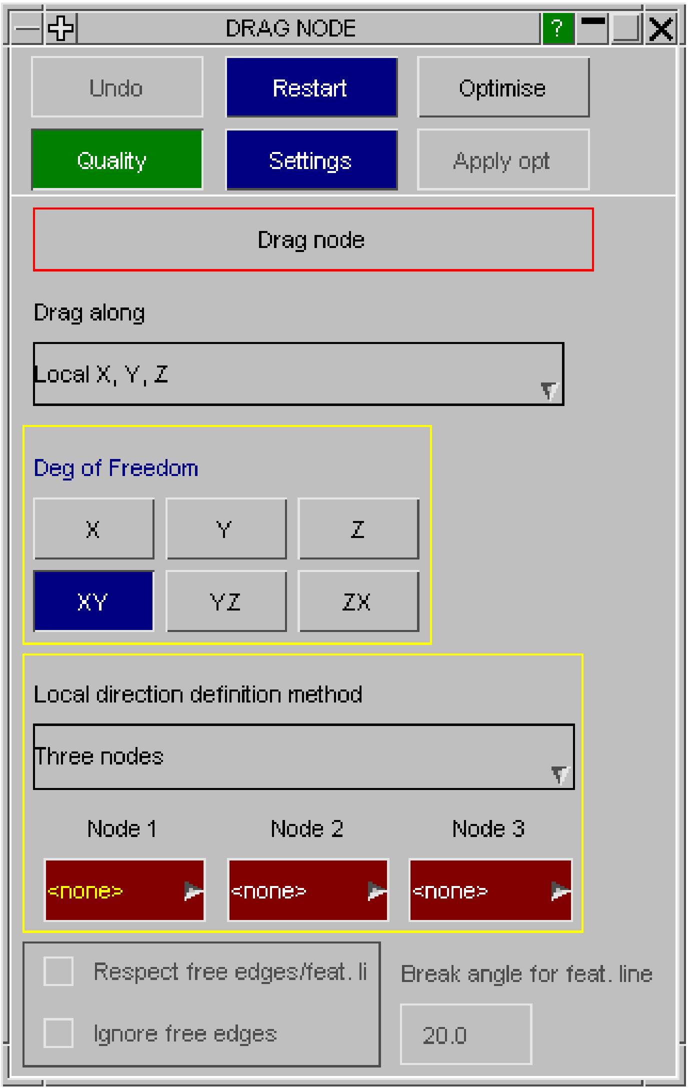
DISMISS returns the user to the main PRIMER window

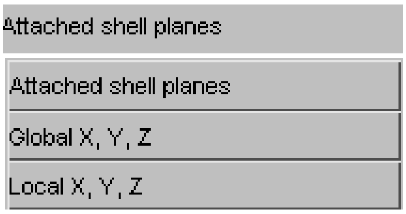
DRAG

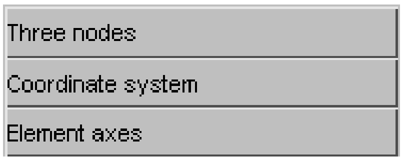
DRAG permits users to drag nodes based on certain constraints. The impact of such an operation on element quality can be viewed using the **Quality** button. Various individual quality metrics, as well as overall quality imperfection can be viewed using the **Settings** button.

Three methods are currently available for node dragging:

- The **Attached shell planes** popup option facilitates dragging along the planes of attached shells.
- The **Local X, Y, Z** option permits dragging along local X, Y, Z axes or along local XY, YZ, ZX planes. Appropriate degrees of freedom can be defined using the attached **X**, **Y**, **Z**, **XY**, **YZ**, **ZX** option buttons. A local coordinate system can be defined using an element, a coordinate system or a set of three nodes.
- The **Global X, Y, Z** option, likewise, permits dragging along global X, Y, Z axes or along global XY, YZ, ZX planes.





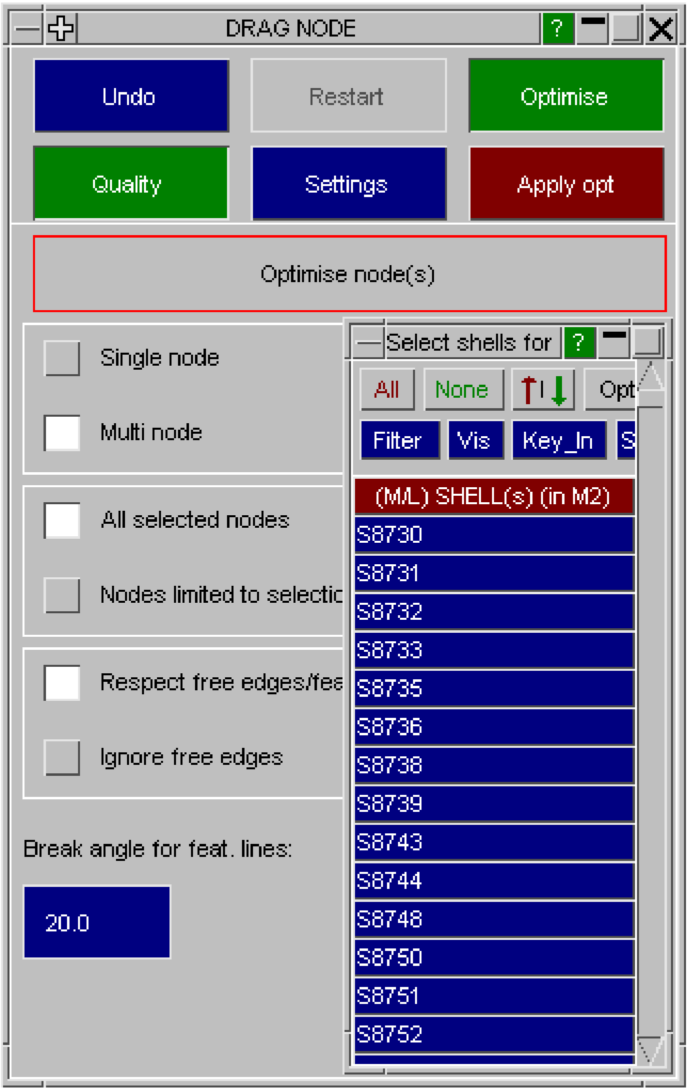


The **Optimise** button will instruct Primer to automatically reposition nodes for improved quality.

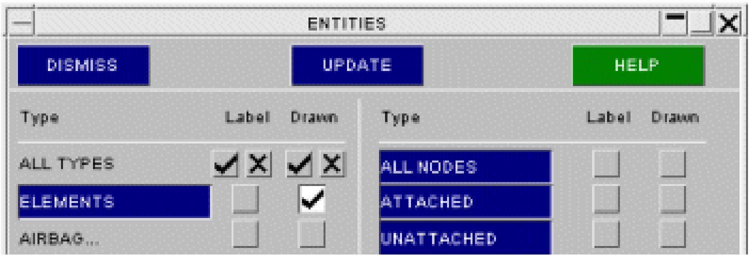
Two optimisation modes are available - single and multi node. The latter permits selection of one or more elements. Primer will then reposition attached nodes so that overall quality of the selected elements is improved.

Nodes that lie on a free edge or feature line can be restrained using an appropriate option.

Movement of nodes that also lie on unselected elements can also be restricted using an option.



Controlling the visibility and labelling of nodes.



Node visibility and labelling is controlled from the **ENT**ity Viewing menu.

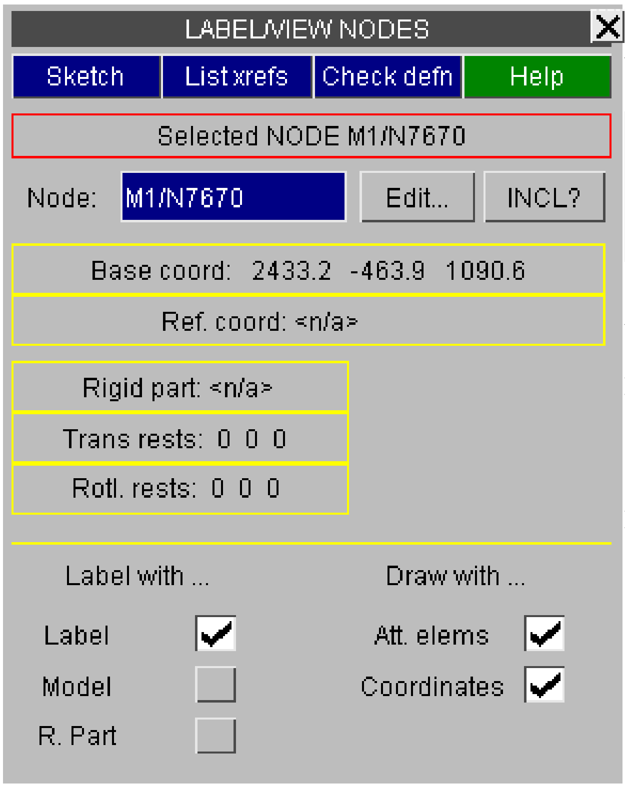
However its treatment is different to that applied to all other item types.

ALL_NODES	Draws all nodes, regardless of attachment
ATTACHED	Draws only nodes attached to items currently visible on the screen
UNATTACHED	Draws nodes that are ***not*** attached to any item.

Labelling of nodes is handled in much the same as their drawing.

Nodes do not have to be drawn explicitly in order to be labelled: for example selecting **ATTACHED** labels will display the node labels on visible elements etc.

Dynamic labelling and details of nodes.



As with elements, nodes have a special "pick to label and display details box" that is invoked by clicking on one of the

ALL_NODES	Buttons in **ENT**ity Viewing (doesn't matter which)
ATTACHED	
UNATTACHED	

or from Quick Pick.

You can control how nodes are labelled and drawn using:

Label with...
- **Label : The node's label**
- **Model : Prefixes the Mnnn model id**
- **R. Part : The part id of any "parent" rigid part.**

Draw with...
- **Att. elems : The elements attached to the node**
- **Coordinates : The node's global coordinates.**

Rules for screen-picking nodes.

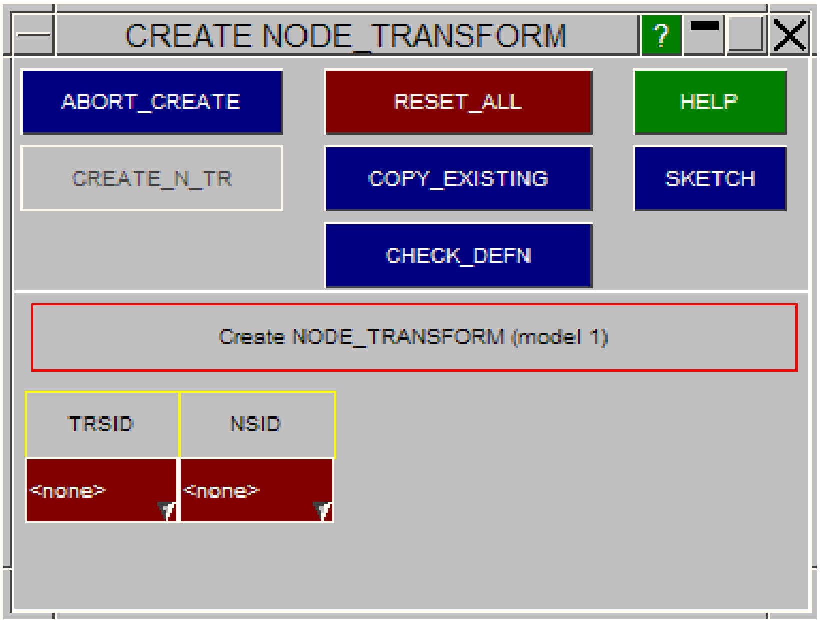
Nodes are treated in a non-standard way for screen-picking purposes. A node is pickable if:

- It is drawn explicitly (using the * symbol)
- Or if an item which uses it (element, joint, restraint, ...) is visible.

The second condition may be thought of as "**ATTACHED**", without actually having to draw the nodes.

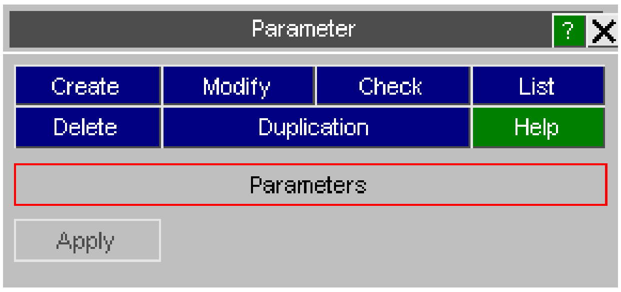
The reason for this anomaly is that it is extremely useful to be able to screen-pick nodes, but very annoying if they have to be drawn to make this possible: node symbols tend to obscure other useful information, and also slow down graphics.

NODE_TRANSFORMATION



Select or create the TRSID, refer to section 3.14 for more information. The node set ID can also be selected or created using the usual methods.

PARAMETERS



The figure on the left is the ***PARAMETER** main menu in PRIMER.

From release 970 onwards LS-DYNA supports parameters. These are Integer or Real (floating point) values defined by name at the top of the input deck which can be used in any data field of the relevant type.

In addition the ***PARAMETER_EXPRESSION** card allows a parameter to be defined by an arbitrary mathematical expression which may contain references to other previously defined parameters.

From release 9.3RC2 onwards PRIMER provides full support for parameters:

- Both ***PARAMETER** and ***PARAMETER_EXPRESSION** are evaluated on input.
- The association between a parameter and the data field in which it is used is "remembered" across all operations, and rewritten on keyword output.
- Parameters, including the _EXPRESSION variant, may be created, edited and deleted interactively.
- When a parameter's value is changed all affected parts of the model will be rebuilt to reflect the change.
- Either parameters or their underlying values can be displayed in all editing panels, and parameters may be typed into any valid field on these.

From release 10.0 PRIMER adds support for the LS971R5 keywords:

- ***PARAMETER_LOCAL**, which permits multiple parameters with the same name but different values in different include files
- ***PARAMETER_DUPLICATION**, which tells LS-DYNA how to process **_LOCAL** name conflicts

Create	Create a new parameter
Modify	Alter Parameter cardsl
Check	Check the parameter cards for errors and completeness.
List	Summarise parameter cards across all models including usage cross-references.
Delete	Delete parameters
Usage in panels	How to use parameters in editing panels and the generic keyword editor.
Mis-matches	What happens when the true data value is changed so that it no longer matches the parameter's value.
***Include_Transform**	Special rules for dealing with parameters used inside *INCLUDE_TRANSFORM definitions
Use for labels	Why you should be very careful about using parameters for item labels
Definition order	How PRIMER copes with the order in which parameters are defined.
Units change	What happens to parameter data when a units change is applied to fields in which they are used
Text strings	How "&name" is processed when encountered in a title or other text string
8 character names	Using the undocumented ability of LS-DYNA to handle 8 character parameter names
Duplication	How PRIMER handles the *PARAMETER_DUPLICATION card.

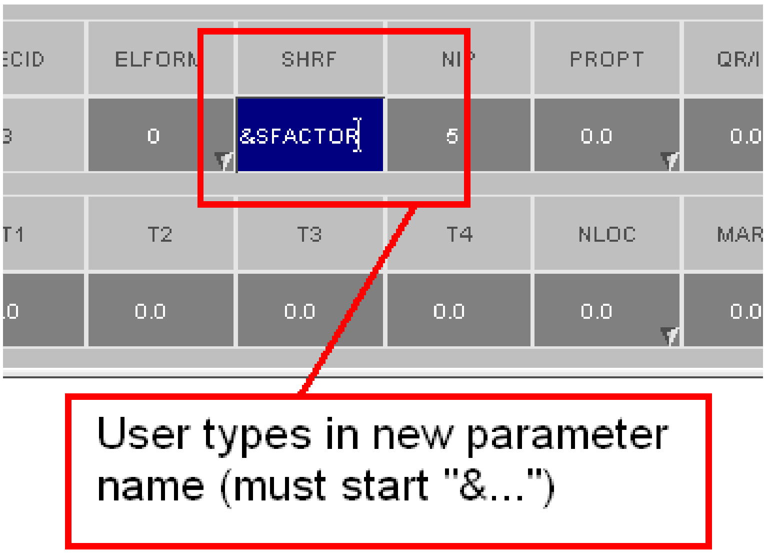
Creating a Parameter

Parameters may be created in the following ways:

(1) By typing a new parameter name into any text entry box in an editing panel

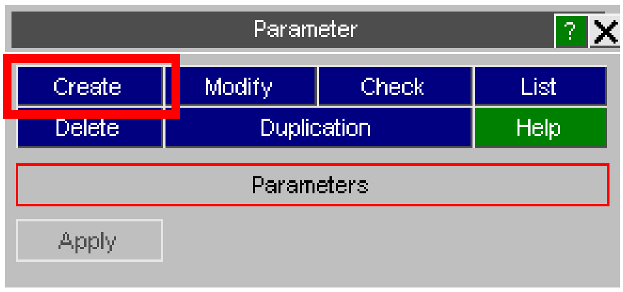
This will map a creation panel populated with:

- the parameter name
- the type (Integer or Real) as deduced from the context
- a sensible default value for this context.



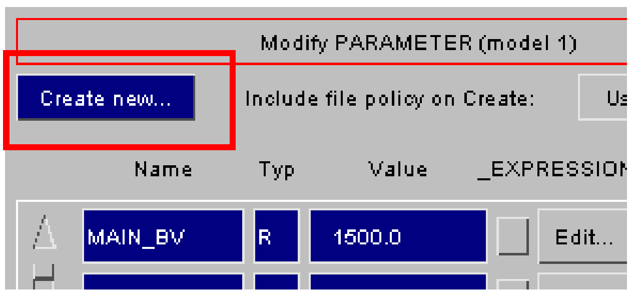
(2) From the Create button in the main Parameter panel

This will map an empty creation panel (see below).

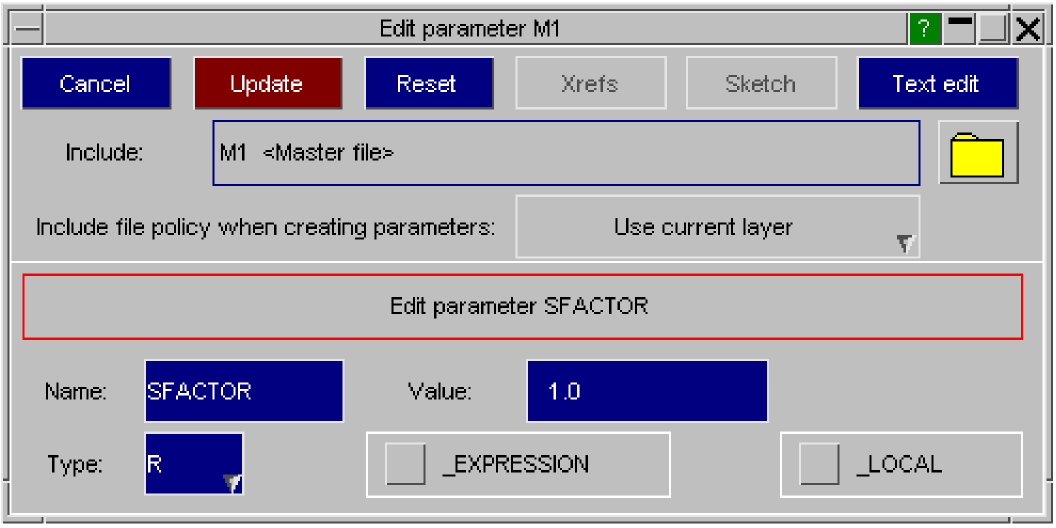


(3) From the "**Create new...**" button on the Modify Parameter panel

As for (2) above this will map an empty creation panel.



The Create/Edit parameter panel



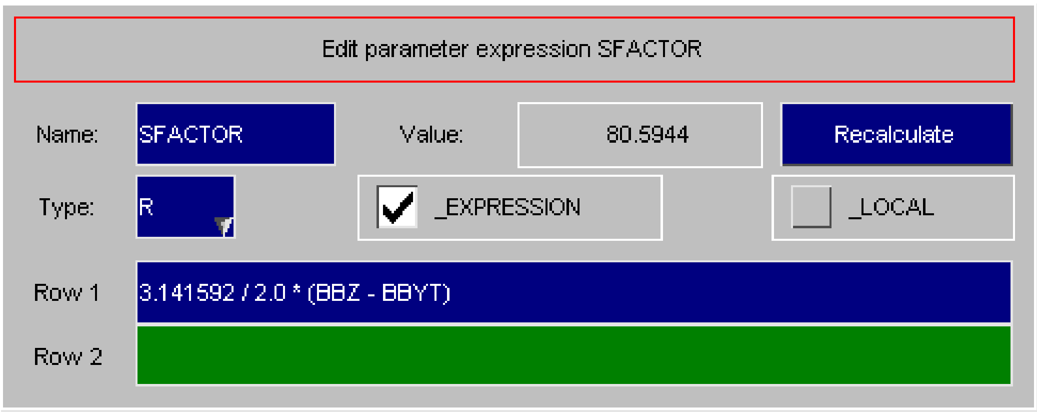
The panel allows you to edit all attributes of the parameter:

- Its name: up to 7 characters without the initial "&". (Although 8 character names seem to be accepted by LS-DYNA, see "allowing the use of 8 character names" below, so PRIMER will also accept this)
- Its type: **I**(nteger) or **R**(eal)
- Its value
- Whether or not it is an **_EXPRESSION** parameter
- Whether or not it is a **_LOCAL** parameter.

The value is simply typed into the "Value" box.

As with other editing panels this is a scratch definition, and the saved attributes of the parameter will only be updated when you use **Update**.

Using the **_EXPRESSION** option.



In the **_EXPRESSION** case the panel extends to provide editing rows to contain the expression.

As many rows as necessary to define the expression may be used. PRIMER will reformat the expression over as many lines as necessary to make it fit the 80 column width of the LS-DYNA input deck.

Its value is calculated when you hit <return>, but if a parameter referred to in the expression (for example BBZ or BBYT here) is changed externally you can use **RECALCULATE** at any time to force a recalculation.

Arithmetic rules used to evaluate PARAMETER_EXPRESSION

PARAMETER_EXPRESSION uses syntax and standard operators common to most programming languages. The standard rules of arithmetic evaluation and operator precedence are obeyed, and built-in functions common to most programming languages are supported.

Precedence of operators

Brackets `(..)`	are senior to
Unary `+/-` (eg -10, +3.141592)	are senior to
Raising to the power of (`^` or `**`)	are senior to
Multiply, divide, modulo (`*, /, %`)	are senior to
Add and subtract (`+, -`)	

Built-in functions

Trigonometric functions (all arguments in radians, not degrees)		**Hyperbolic functions**	
`sin(x)`	`asin(x)`	`sinh(x)`	`asinh(x)`
`cos(x)`	`acos(x)`	`cosh(x)`	`acosh(x)`
`tan(x)`	`atan(x)`	`tanh(x)`	`atanh(x)`
`sec(x)`	`atan2(x, y)`		
`csc(x)`			
`cot(x)`			
General maths functions			
`exp(x)` (e to the power of x)	`abs(x)` (absolute value of x)	`int(x)` (Integer part of x, truncated)	`float(x)` (x as a floating point value)
`log(x)` (natural log of x)	`mod(x,y)` (remainder when x divided by y)	`aint(x)` (Absolute value of integer part of x, truncated)	`ceil(x)` (nearest integer (as a float) larger than x)
`log10(x)` (log base 10 of x)	`max(x,y)` (maximum of x and y)	`nint(x)` (Nearest integer to x, rounded)	`floor(x)` (nearest integer (as a float) less than x)
`sqrt(x)`	`min(x,y)` (minimum of x and y)	`anint(x)` (Absolute value of nearest integer to x, rounded)	

All functions return a floating point result except for:

`int(x), nint(x), aint(x), anint(x)`	All return integers
`max(x,y), min(x,y), mod(x,y)`	Return integer if both X and Y are integers, otherwise floating point
`abs(x)`	Returns an integer if X is integer, otherwise floating point

Integer vs Floating evaluation of expressions

Programming languages make a distinction between an integer (eg 2) and a floating point value (eg 2.0). The same is true of expressions which give a result of the same type of their component parts, however when an expression is mixed the result is "promoted" to floating point. Generally this has little influence on the outcome, but there is one significant exception:

WARNING: integer division truncates!!

Consider the following examples

Expression	**Gives result**	**Because**
`5.0 / 2.0`	`2.5`	Both parts are floating point, giving a floating result
`5 / 2`	`2`	Both parts are integer, so result is truncated to an integer
`5 / 2.0`	`2.5`	Expression is promoted to float because ot has mixed types
`5.0 / 2`	`2.5`	Expression is promoted to float because ot has mixed types

These rules apply to subsets of expressions as well as the expression as a whole, for example:

Expression	**Gives result**	**Because**
`1.0 + 5 / 2`	`3.0`	The part (5 / 2) evaluates to 2, so result is 1.0 + 2 = 3.0 (result promoted to highest type)
`1 + 5 / 2`	`3`	All integer, so 1 + 2 = 3
`1 + 5.0 / 2.0`	`3.5`	1 (integer) + 2.5 (float) = 3.5 (float result)

Because it is easy to write an "all integer" expression by mistake when a floating point result is desired PRIMER performs a check for this on all **`PARAMETER_EXPRESSION`** definitions. It evaluates the parameter twice: once using the rules described above and a second time using "all floating point" arithmetic. If the results are different by more than 1 part in 1e6 of the magnitude of the result then a warning is issued stating that the result depends upon integer truncation. Almost always this is due to integer division as given in the examples above, but it is theoretically also possible as a result of the **`mod(x,y)`** function (or the expression **`x % y`**, which means the same thing) returning an integer result.

It is recommended that "real" expressions are always written using floating point numbers, ie write 2.0 rather than 2, as this reduces the chances of accidental integer truncation and the meaning is clear to anyone reading the expression.

Integer expressions may deliberately utilise integer truncation, but it is recommended that this is made explicit by using the **`int(x)`** function. It will make no difference to the outcome, but it will make it clear to the reader that truncation is intended.

When expressions refer to other parameters the type of each parameter's value is retained: integer parameters evaluate as integers, and real parameters as floats.

Conversion of results by Parameter type

You will see from the above that the results of parameter expressions may be floating point or integer, however parameters themselves may also be declared as being "real" (floating point) or integer.

Primer converts results as follows:

Result of evaluating expression	Destination parameter type	How data is converted
Floating point	Real	Takes result directly
	Integer	Truncates result to integer value (ie 2.5 becomes 2)
Integer	Real	Converts result to float (ie result 2 becomes 2.0)
	Integer	Takes result directly

Precision of Parameters

`Integer` parameters use single precision signed integers, giving results in the range +/- 2**31 - 1; in decimal this is +/-2,147,483,647 or just over 9 significant figures.

`Real` parameters use single precision floating point value, giving a decimal mantissa of about 7.5 significant figures and a decimal exponent range of about 1.0e+/-38.

`PARAMETER_EXPRESSION`s are evaluated as follows:

- All calculations are performed using double precision floating point variables, so all arithmetic is performed with a mantissa precision of about 16 significant figures and an exponent range of 1.0e+/-308.

- Where integer truncation is applied as described above ***within*** an expression the result is a "whole number" stored as a double precision float, retaining this precision.

- The result of a **`PARAMETER_EXPRESSION`** evaluation is converted to the intrinsic type of the parameter, thus:

  `Real` parameters convert the double precision floating result to a single precision floating value, in other words a mantissa of 16 sig figs is truncated to 7.5 sig figs, and the exponent is truncated from 1.0e+/-308 to 1.0e+/-38.

  `Integer` parameters truncate the double precision floating result to a single precision integer, ie a value in the range +/-2,147,483,647. Truncation to integer includes a small tolerance, the actual expression being:

  <Integer result> = "Integer part of" (<double precision floating result> + <tolerance>)

  where <tolerance> is +1.0e-6 for a +ve result, and -1.0e-6 for a -ve result.

  When a **`PARAMETER_EXPRESSION`** is used to calculate an integer result, typically for a label, it is advisable to use "all integer syntax" in the expression to avoid any possibility of ambiguity or rounding errors.

Parameters and Include files.

Early versions of LS-DYNA required all **`*PARAMETER`** statements to occur after **`*KEYWORD`** and before any other input card. This was quite restrictive, and the limitation was soon removed - in fact LS-DYNA 971R3 doesn't care what order parameters arrive in, of where they are written - see "the order in which parameters are defined" below.

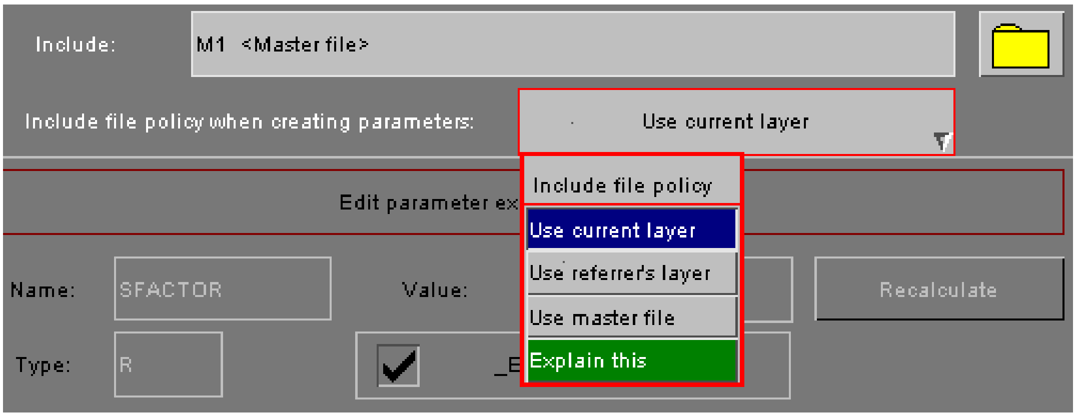
PRIMER adopts the following strategy:

- It doesn't care where in the deck a **`*PARAMETER`** statement occurs.
- From release 9.3 onwards it no longer needs to be defined before it is used, although not doing so is deprecated.
- It doesn't care if a **`*PARAMETER`** statement occurs in an include file, and it will keep it in that include file.
- It permits parameters to be moved between include files and master file, just like any other keyword.

The use of parameters inside include files manipulated by `*INCLUDE_TRANSFORM` is a special case, please see below for more details.

The use of the **`_LOCAL`** suffix to the **`*PARAMETER`** card, introduced in LS971R5, permits parameters to be multiply defined with different values in different include files, and this affects the scope of their applicability. This is explained under "Using the _LOCAL option" below.

The policy used for the destination Include file of new parameters.



There are three options:

1. **Create the parameter in the current layer (include file).**

   This is the default, and is consistent with the creation policy for other items in PRIMER.

2. **Use referrer's layer.**

   This is only relevant in the case of a parameter being created via a new name (&xxx) being typed into a text entry box (option 1 above).

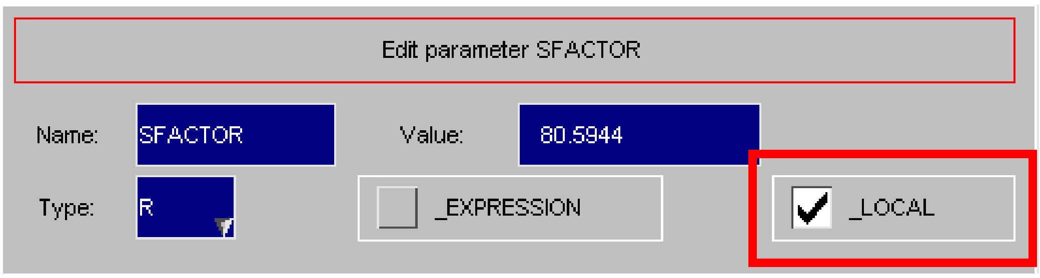
   In this mode the new parameter will "inherit" the include file of the item being edited. Parameters created by other means will use the default "current layer" to determine their include file.

3. **Use master file**.

   All new parameters are forced to reside in the master file. If the strict ls-dyna rules are applied this is the safest option, however where parameters are used in an include file it may result in "decoupling" of the parameter definition and its subsequent usage in a different context.

The include file of any parameter may be changed at will using the standard "include" editing buttons just as for any other in PRIMER.

Using the _LOCAL option



The suffix **_LOCAL** was introduced in LS-DYNA 971R5 (early 2011) and it is defined in the LS-DYNA user manual as working as follows:

- A conventional "global" parameter, plain or **_EXPRESSION**, is normally applicable throughout the whole model, including all include files, regardless of where or in which include file it is defined.
- A parameter with the **_LOCAL** suffix is only applicable in the include file in which it is defined.
- Multiple parameters of the same name, using different values, may exist in different include files in a deck so long as all definitions, or all but a global one, use the **_LOCAL** suffix.
- In a given include file file a parameter with the **_LOCAL** suffix defined in that file will "mask" all other parameters of the same name defined elsewhere in the deck, so that its local value is used.
- The ***PARAMETER_DUPLICATION** keyword defines how LS-DYNA will handle parameter **_LOCAL** and global name conflicts.

*PARAMETER_LOCAL scope rules used in PRIMER

The LS971R5 manual leaves certain details of this usage ambiguous, for example:

- If an include file containing ***PARAMETER_LOCAL** has child include files, does this "local" definition propagate down to usage in these include files, or will they use any globally defined (non_**LOCAL**) definitions of the parameter using the same name found elsewhere in the deck?
- If a child include file in the case above contains a plain (non_**LOCAL**) parameter of the same name will this be globally applicable to all other include files other than its parent? And what would be used for references within that file and any children it might have?

Pending resolution of these questions PRIMER uses the following rules to define the scope of parameters.

1. A global (non_**LOCAL**) parameter defined anywhere in the input deck, ie in master file or any include file, is also valid anywhere in the deck except when a **PARAMETER_LOCAL** definition of the same name occurs in an include file, in which case this local definition supersedes the global one:
   - In that include file
   - And in any children of that include file to any number of generations.
2. A **PARAMETER_LOCAL** definition in an include file not only supersedes any global parameter or the same name, but also any **_LOCAL** definitions of the same name "above" it in parent, (and grand-parent, etc to any number of generations) include files.
3. If a parent file contains a **PARAMETER_LOCAL** definition, and a child of that parent contains a global (non_**LOCA**L) definition of the same name, then the **_LOCAL** definition from the parent will be used for any references to that parameter in that file. (This is really an amplification of rule 1 above.)

   However the global definition will still be used for any references "above" that parent file elsewhere in the model, subject to any local definitions that they may have.

Another way to think of these rules is that PRIMER always prefers a **_LOCAL** definition from this file, or from its closest ancestor, over a global definition or a **_LOCAL** one from a more distant ancestor.

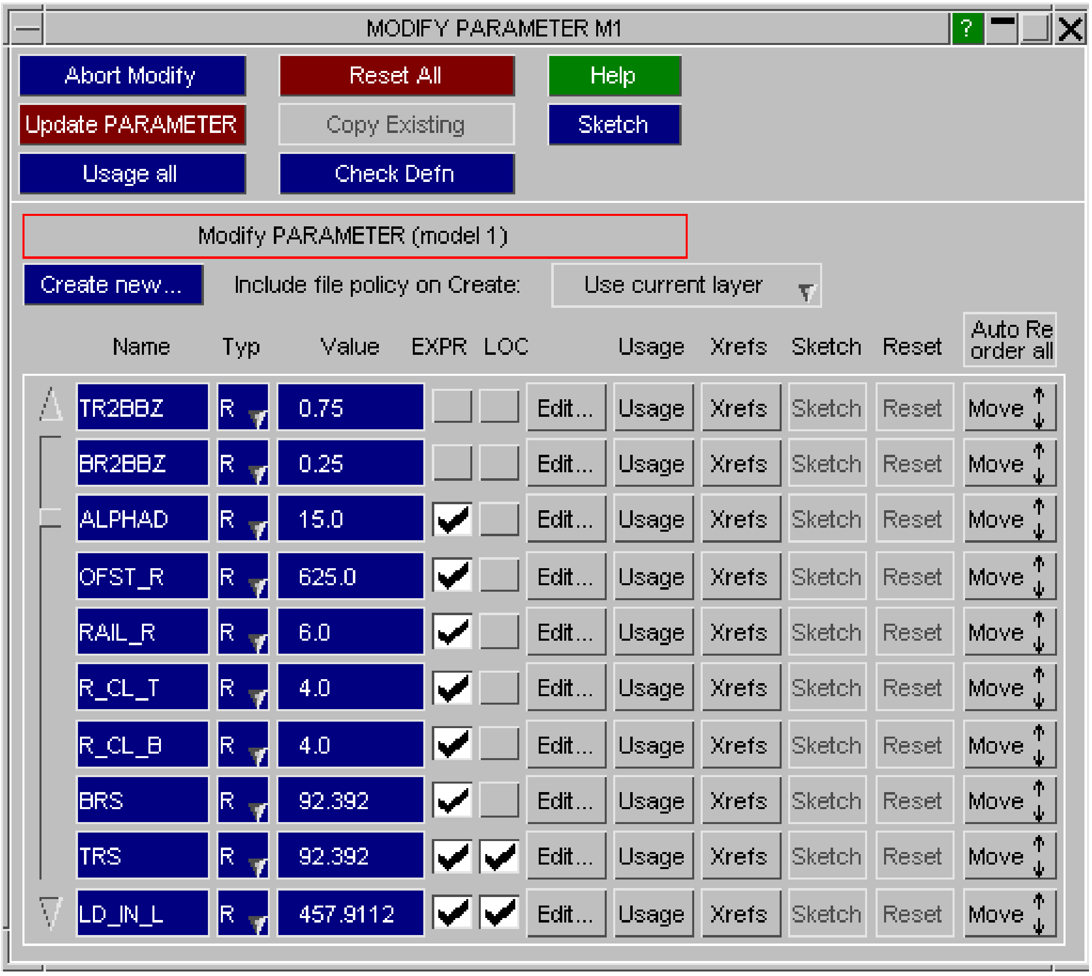
Although PRIMER reads and processes the ***PARAMETER_DUPLICATION** card (see below), it ignores any settings on it for internal purposes and treats definition of a global and a **_LOCAL** parameter of the same name in a single include file as an error. Existing decks containing this conflict will be read and a Model Check will generate errors. When editing existing parameters or creating new ones this "no duplication" rule will be enforced, and you will not be permitted to define two parameters of the same name in the same include file.

Modifying Parameters

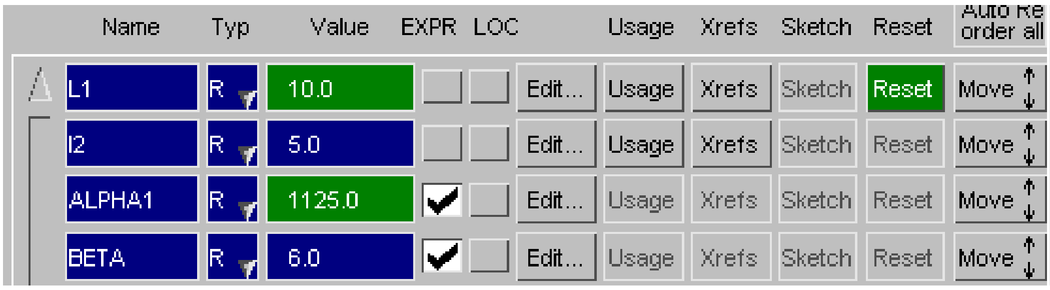
Parameter modification can be carried out at two levels:

The top level, shown here, lists all parameters in the model showing a summary of their attributes.

You may type in a revised name, type (I or R) or Value, or switch between plain and **_EXPRESSION** variants. For more detailed editing, necessary when changing an EXPRESSION, use the **Edit...** button to map the detailed editing panel above. Changes made in this panel are applied to a scratch definition, and the permanent parameter definitions will not be updated until **Update PARAMETERS** is used.



Changing a Parameter's value



When a parameter's attributes are changed the revised values are shown on a green background. In this example **L1** has been changed from 4.0 above to 10.5. In addition because the **EXPRESSION** in parameter **ALPHA** also refers to **L1** it too has changed.

Changes are "scratch", as explained above, and can be undone using the **Reset** button.

Dealing with errors caused by changes.

Sometimes changes to one parameter will create errors in an **`EXPRESSION`** that uses them.

In this example the parameter **`ALPHA`** has been changed to include "`1.0 / L1`" and the value of **`L1`** has been changed to zero. This gives a divide by zero error when evaluating **`ALPHA`**, and it is shown on a red background to emphasise this.

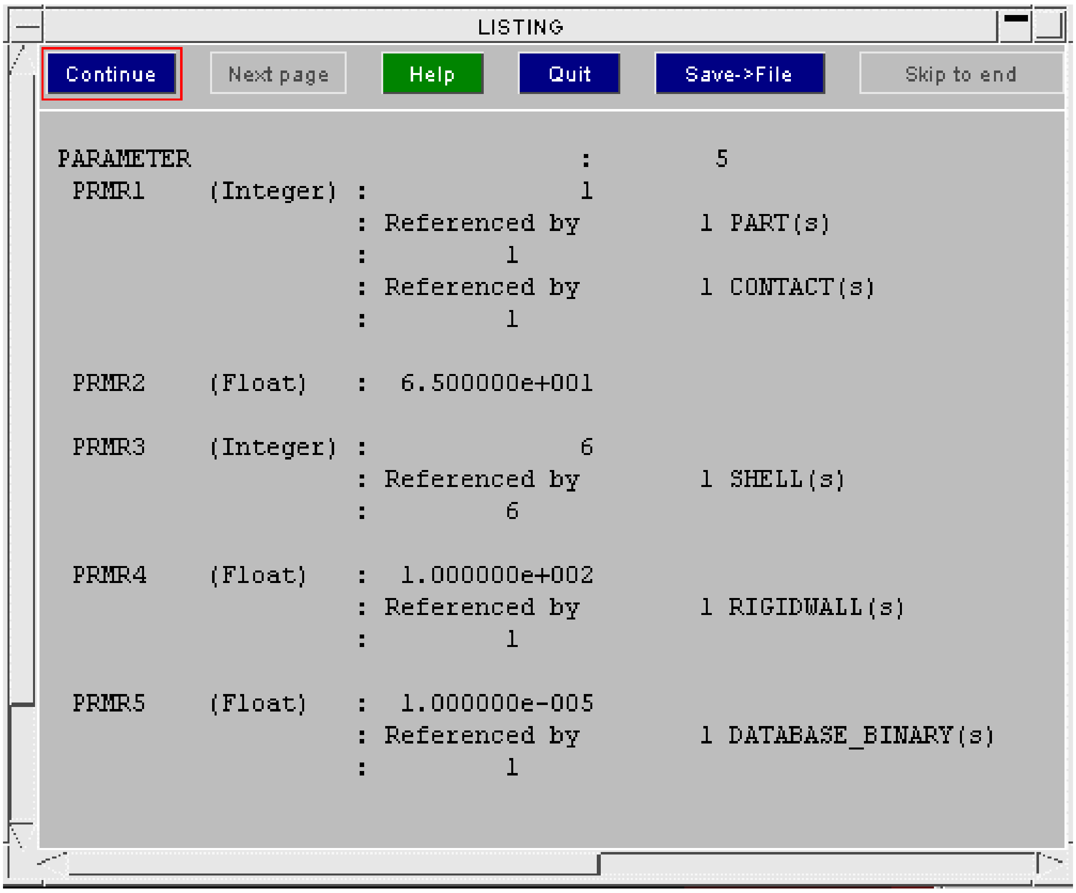
PRIMER can tolerate arithmetic errors in expressions, and substitutes a value of zero. However there is no guarantee that the analysis code will handle this and you will be warned if you attempt to update and save any errors.

Check

This checks all parameter cards for errors. It is can also be accessed from the **CHECK_DEFN** button in the **Modify** window.

List

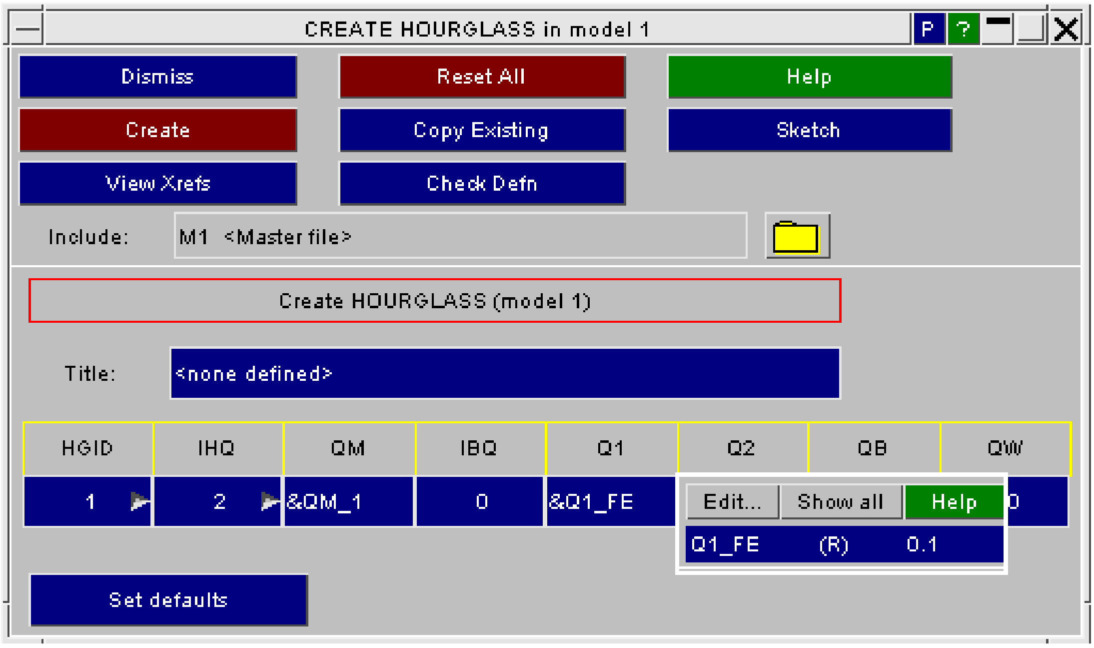
This provides a listing of all the parameters in the model and displays their numeric type, Value and what they are referenced by. The same listing can be accessed from **USAGE_ALL** in the **Modify** window.



DELETE deleting parameters.

Parameters may be deleted using the normal deletion logic in PRIMER. If they are referred to anywhere in the model, or within another **PARAMETER_EXPRESSION** statement, they will be locked against deletion.

Parameters in Editing Panels

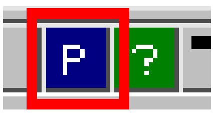


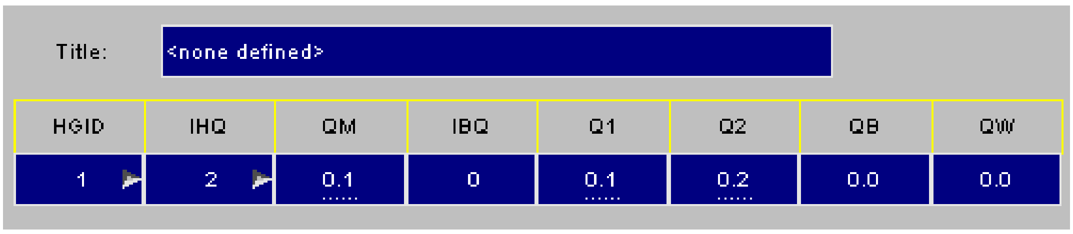
From PRIMER 9.3RC2 onwards editing panels, including generic keyword editor ones, display parameters either by name or by value wherever they are used.

By default they are displayed in exactly the same format as that used in keyword output files: **&NAME**, but this can be toggled to show values instead - see below.

Hovering the mouse over such a parameterised field will display a popup box giving the parameter name, type and value. In this example the user has hovered over data field **Q1** which uses the parameter **&Q1_FE**. The popup box reveals that its value is **0.1**.

To edit this parameter use **Edit...**

Using  to toggle Parameter display mode.



An alternative to **&NAME** symtax is to show the value of the parameter, but underlined to emphasise that the value shown is parameterised. You can toggle between the two modes using the "**P**" button at the top right hand corner of any panel showing parameters.

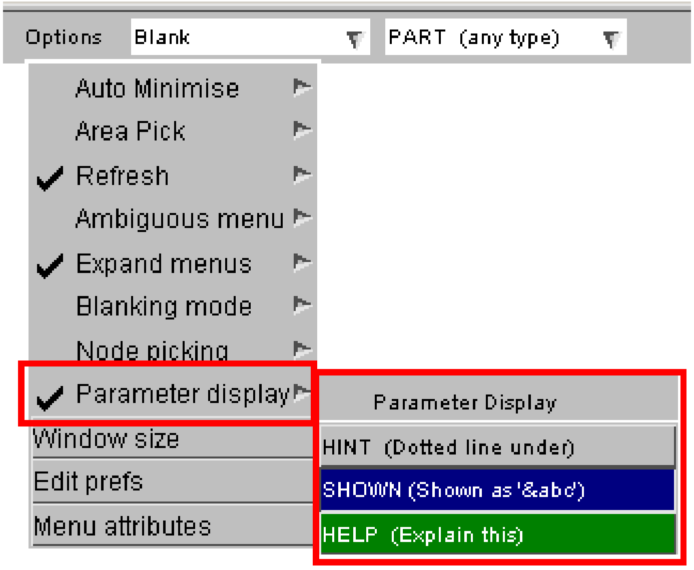
This image shows the panel above displayed in this alternative format. Hovering the mouse above an underlined field will map the parameter popup box as shown above, giving information about the parameter name.

Setting programme-wide parameter display mode.

The "**P**" button on an editing panel only controls display for that panel.

To set the default display mode for the whole programme use **Options, Parameter Display** and select:

HINT	Numeric values underlined with dots
SHOWN	Shown in **&NAME** format



Entering parameters in edit panel data fields.

In any data field where a parameter could be used in an LS-DYNA input deck the equivalent edit panel field may also use one. To enter a parameter do the following:

- Type an ampersand "**&**" into the text box field
- This will immediately map a popup box showing all currently defined parameters (if any exist)
- To choose an existing parameter simply select it from the popup.
- To narrow down the choice from many parameters type its initial letter(s), the list will contract to show only those that match.
- To enter a new, previously undefined, parameter type in its full name.

In the last case, entering a new parameter, the following happens:

- A new parameter definition using that name is created. It will be a plain parameter, ie not _EXPRESSION.
- It is given a type (Integer or Real) appropriate to the context
- It is given an initial value of zero, unless that would be illegal in that context in which case the smallest legal value is used.
- This value is entered into the data field, and the edit panel is updated.

The editing panel for the new parameter is then mapped, and you are invited to update its attributes. When you save these then its value on the "parent" editing panel will also be updated.

How to remove a parameter association.

To change a parameterised value to a plain number simply rub out the **&NAME** (or underlined) field, and type in a simple number. This will break the association between parameter and data field, even if the number you type in is the same as the parameter's value.

What happens when the value in a parameterised field is changed by some other means.

It is possible for the value in a parameterised field to be changed indirectly by some other operation in PRIMER, breaking the association between the field's true value and the associated parameter value. Consider the following sequence:

1. Parameter **&X1** is used as the X coordinate of a node.
2. The user chooses to **ORIENT** the model, translating it in X

After step (2) the the node's X coordinate will no longer match the value of parameter &X1. Other examples of such changes are "auto-fixing" in Check, contact de-penetration, mechanism/dummy positioning, item re-labelling, and there are many more such cases.

In these situations PRIMER will (silently) remove the association between the parameter and its usage in this context, so that if the deck is written out or the node displayed in an editing panel only the plain numeric value will be shown.

The test for whether or not there is a mismatch between data field and parameter value is as follows:

Parameter type	Description of test	Test as an algorithm
`Integer`	The test is absolute: any mis-match breaks the association	if(value != parameter)
`Real`	If the values differ by more than the absolute value of the parameter x 1.0e-6	if (abs(value - parameter) > abs(parameter) * 1.0e-6)

The special case of using Parameters in *INCLUDE_TRANSFORM files

A special situation arises when parameters are used in **`*INCLUDE`** files that are subject to transformation by the **`*INCLUDE_TRANSFORM`** keyword. When dealing with such files there are two requirements:

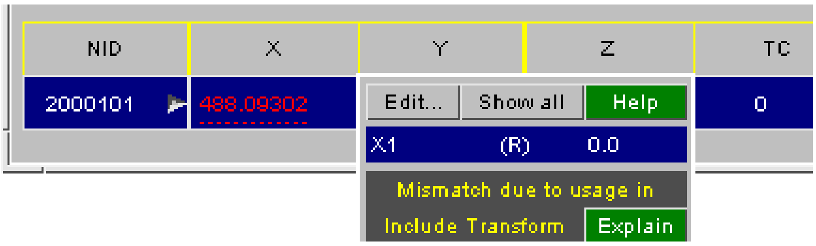
1. PRIMER must apply any transformations when it reads the input deck, so that it can display the model in its "as transformed" state.
2. PRIMER must "undo" the transforms on keyword output so that the original files are written out unchanged.

If parameters are used inside the **`*INCLUDE`** files this can lead to conflicts, since (1) may require the underlying values to change, invalidating the association between parameter and true values; yet (2) requires that the association be "remembered" in order that it can be restored on keyword output.

PRIMER deals with these conflicting requirements by setting a special internal flag against associations occurring inside **`*INCLUDE_TRANSFORM`** files. If the true value and the parameter value do not match in these cases then the following is done:

- The true value is used (as normal) for all calculations and display.
- In editing panels the true value is always displayed, but in red and underlined to designate it as a special parameter.
- On keyword output the inverse transformation is applied to the true value, and the parameter **`&NAME`** written if they still match (to within a small tolerance).

A typical editing panel field looks like this: here parameter **`&X1`** is the node's X coordinate. The parameter popup mapped in response to hovering over the panel shows the parameter's actual value and gives a warning message about the mis-match.



The effect of editing parameterised items used inside ***INCLUDE_TRANSFORM** files is as follows:

- Entering an explicit number in a parameterised field will destroy the association between data field and parameter - as in the normal case above.
- Entering a parameter into a field will result in the as-transformed value of that field changing to the parameter's current value. This may or may not result in the parameter definition appearing in the keyword output deck, as this will depend upon whether or not the pre-output inverse transformation changes its value so that it no longer matches the underlying value.

If the value of a parameter definition used inside a **`*INCLUDE_TRANSFORM`**, or a **`*DEFINE_TRANSFORMATION`** to which is refers, is changed then:

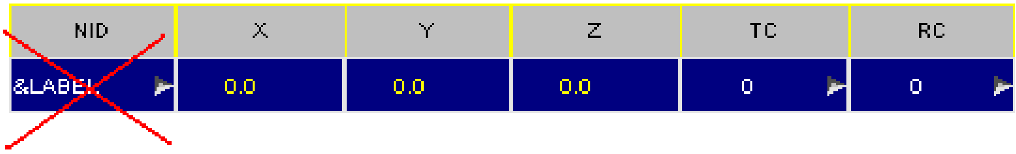
- The contents of the **`*INCLUDE_TRANSFORM`** are "untransformed".
- Any data fields using that parameter are changed to the new parameter value.
- The include file contents are "retransformed".

For large include files this may take a significant amount of time, and the "untransform, modify, retransform" cycle may result in small numerical errors in the result. The process works, but it is better if it can be avoided!

This is a complex problem, which only gets worse if the same include file is used multiple times in different **`*INCLUDE_TRANSFORM`** statements to create multiple instances of parameterised data fields! Please contact Oasys Ltd for help and advice if you are having problems with this.

Do not use parameters for the label fields of items!!!

(Or use them only if you won't change their values interactively during a PRIMER session.)



Do NOT use parameters for item labels!!!

It is ***STRONGLY*** recommended that you do ***not*** use parameters to define the labels of items. (ie the labels of nodes, elements, parts, etc).

PRIMER will try to cope if you do, but changing the value of such a parameter could lead to all sorts of conflicts and problems. For example if you accidentally change the parameter value so that you generate duplicate labels the internal "rebuild" of the affected items may fail because of errors, and this may lead to items disappearing - or indeed the whole model being deleted to avoid internal inconsistencies.

PRIMER will read in and process decks that use parameters in this way, and all will be well so long as their values are not changed, but you should regard this as extremely dangerous modelling practice which will, if treated as anything other than "read only", cause severe problems.

If you choose to use PARAMETER_EXPRESSION to build up and define labels remember that the arithmetic rules used to calculate expression values will truncate integer expressions to an integer result. This may have unexpected consequences if your expression includes integer division, and you should read the section "Integer vs Floating evaluation of expressions" above very carefully.

The order in which parameters are defined.

PRIMER 9.3RC2 required that parameters should be defined before they were used, since this was a requirement of contemporary LS-DYNA versions.

PRIMER 9.3 onwards permits parameters to be defined anywhere, including after where they are used, since this is now supported in LS-DYNA 971R3. However ***this practice is deprecated*** because:

- On keyword input PRIMER has to assign a default value of zero to undefined parameters, and then have a final pass to repopulate the internal definitions once the parameters' true values are known. This can slow down input, and if parameters are used inside **`*DEFINE_TRANSFORMATION`** definitions an "untransform, modify, retransform" cycle may be required which can lead to a build-up of small numerical errors.

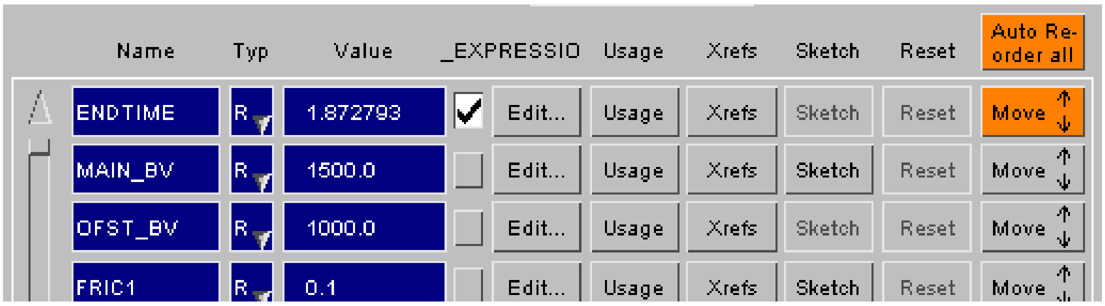
- It is possible, albeit unlikely, that an input deck may fail to read into PRIMER if the temporary substitution of zero in a parameterised data field results in an invalid card definition. This is most likely to be the case with integer values, but there are a few cards, especially among the materials, where the detailed format depends upon floating point data field values.

- Versions of LS-DYNA prior to 971R3 may not read input decks in which parameters are used before they are defined. Documentation of earlier releases suggested this would be the case, and it is not clear exactly when this limitation was removed.

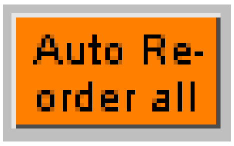
- Other 3rd party software may also have difficulties reading keyword decks in which parameters are used before they are defined.

Re-ordering parameter definitions inside PRIMER

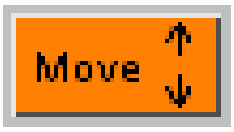
Once inside PRIMER the order in which parameters are stored is irrelevant, since the code can access internal data in any order. However it contains logic to re-order parameters to deal with the problem of a **`*PARAMETER_EXPRESSION`** referring to another parameter that is defined after rather than before it.



In this example **ENDTIME** is a parameter expression which refers to other parameters, and it has been moved to become the first entry in the list so that it is "out of order", resulting in its "**Move**" button being coloured orange as shown here. This can be resolved as follows:



Auto Re-order all will attempt to resort the parameter list automatically to resolve any ordering conflicts. It moves all plain (non **_EXPRESSION**) parameters to the top of the list, and then tries to re-order any remaining **_EXPRESSION** definitions so that they only refer to their predecessors.



Move provides a manual alternative to the above. Click on this button, and then use the keyboard up or down arrow keys to move this row up or down. Once the ordering conflict has been resolved the button will go green.

As explained above it is not strictly necessary to re-order parameters since both PRIMER 9.3 onwards and LS-DYNA 971R3 onwards can cope with them in any order. The re-ordering logic described here was written to deal with the earlier situation when this was not the case, but its use is still recommended for the reasons given above.

What happens to Parameters during a Units change operation

If a units change is applied to scale parameterised data fields, but the parameters themselves do not also have their values changed, then the association between the parameters and the data fields will be lost. PRIMER tries to handle this problem by inferring scale factors for parameters and hence scaling their values; this process is described in section 6.6.2 How Units change affects Parameters.

Parameters used in titles and other text strings.

PRIMER does not attempt to evaluate parameters used in titles or other text strings, simply leaving them as "&*name*". It also does not create a cross-reference between such usage and the parameter, meaning that they do not "know about one another" and a parameter used only in text strings will not be locked against deletion.

LS-DYNA handles "&*name*" in titles and text strings as follows:

- Release 971R4 (mid 2009) onwards will replace valid definitions with their numeric values. If "&*name*" is not a valid parameter then it will issue an error message and exit.
- Releases prior to this had inconsistent behaviour: some simply ignored "&*name*", some attempted to substitute numeric values with varying degrees of success, some generated errors if "&*name*" was not a valid parameter definition and some did not seem to care.

Therefore from release 9.4 onwards PRIMER deals with "&*name*" in titles and text strings as follows:

- During normal interactive usage (editing, copying, etc) these definitions are simply reproduced verbatim and are treated as ordinary text strings with no special significance.
- During keyword output ***only*** for "&*name*" in a title or text string behaviour depends upon the **primer*parameter_in_string** preference:

  This preference can have one of four values:

<table>
<tr><th>Preference value</th><th>Effect during output</th></tr>
<tr><td>AUTOMATIC
(default)</td><td>Processing on keyout depends on the LS-DYNA output version chosen:
<table>
<tr><th>LS-DYNA version selected for output</th><th>"&name" is a valid parameter name</th><th>"&name" does not match any parameter</th></tr>
<tr><td>971R4 and later</td><td>The string is output unchanged, ie as "&name"</td><td rowspan="2">A "@" character replaces "&" in all cases.</td></tr>
<tr><td>Pre 971R4</td><td>The numeric value of the parameter is substituted into the text string, replacing "&name" with the appropriate floating point or integer number.</td></tr>
</table></td></tr>
<tr><td>REPLACE_AMPERSAND</td><td>All "&" symbols in text and title strings are replaced unconditionally with "@", regardless of the LS-DYNA version chosen for output.</td></tr>
<tr><td>INSERT_VALUE</td><td>Where "&name" is a valid parameter its numeric value is inserted into the text string, regardless of the LS-DYNA version chosen for output. If it is invalid "@" is substituted for "&".</td></tr>
<tr><td>VERBATIM</td><td>The text string is unchanged, with no substitutions of any sort being made.</td></tr>
</table>

The default **AUTOMATIC** output policy may not suit all possible circumstances, but it has the merit that it should prevent initialisation in LS-DYNA failing due to malformed text strings in titles.

It is recommended that you do not use the "&" character in titles and text strings unless you are certain that it prefixes a valid parameter name, and even then it may cause problems in all but the most recent versions of LS-DYNA.

Allowing Parameters to use 8 character names

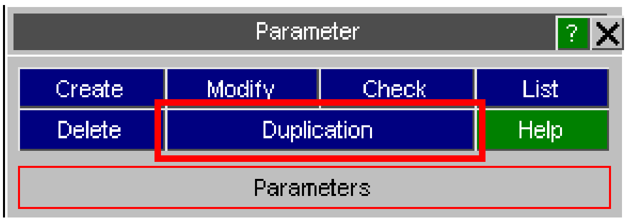
The LS-DYNA user manual states explicitly that parameter names are limited to 7 characters; however experimentation by users has revealed that LS-DYNA will in fact accept and process 8 character parameter names, and there is some evidence to suggest that the 7 character limit stipulated in the user manual is a misprint.

Since PRIMER needs to be able to read and process input decks that run in LS_DYNA it has been modified (in version 10.0) to increase the maximum length of parameter names from 7 to 8 characters.

A model "check" will issue a warning that this undocumented capability is being used if any 8 character parameter names are found, but this warning can be suppressed by setting the preference

primer*warn_param_8_chars: off

The *PARAMETER_DUPLICATION keyword



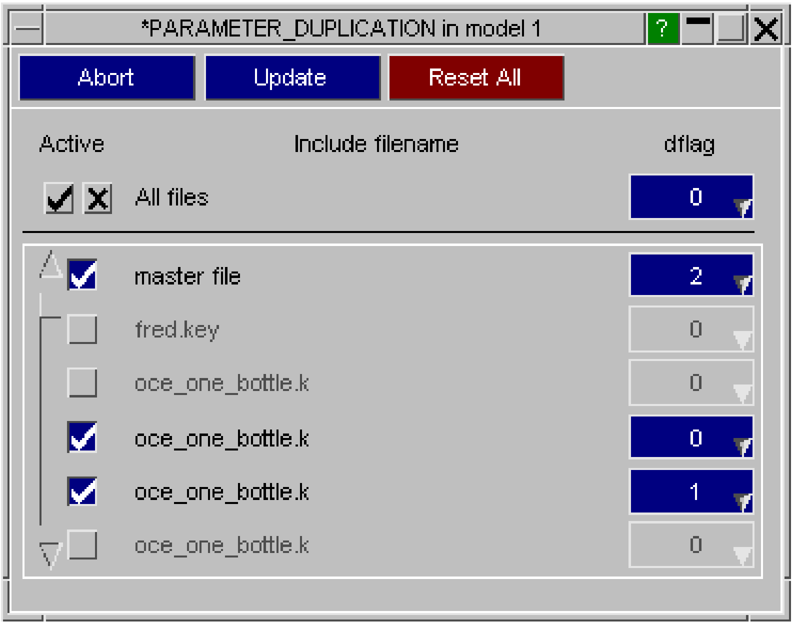
LS-DYNA 971R5 introduced the concept of "local" parameters, and these are described above. This creates the possibility of errors when both global and _LOCAL parameters are accidentally defined in the same include file, and the behaviour of LS-DYNA on input is determined by the settings on this card.

The LS-DYNA manual does not make it clear whether this is a "once only" keyword, that should only be defined once and will control behaviour throughout all include files in the model; or whether it can be defined multiple times in different include files, possibly with different values and permitting alternative behaviour in these files.

PRIMER permits each include file (including the master file) to have a separate, possibly different, definition of this keyword. It also permits these definitions to be created, edited and deleted using the **Duplication** option.

As this example shows you choose whether to have a **`*PARAMETER_DUPLICATION`** card in each include file by using the Active tick box.

In files where it is present you can set the **`DFLAG`** value accordingly.



*PARAMETER_DUPLICATION and error checking in PRIMER

As explained under _LOCAL above PRIMER does not consider the settings on this card when checking for errors in parameters. It treats duplication of parameter names within a file as an unconditional error regardless of any settings here, or indeed whether this card is present or not.

PART: Defining Parts.

[Top level menu](#)
- Creating
- Copying
- Editing
- Deleting

Rigid Parts
Visualisation

Options:
- _INERTIA
- _CONTACT
- _REPOSITION
- _PRINT
- _ATT.._NODES

Sub-keywords:
- _ADA.._FAIL..
- _COMPOSITE
- _MODES
- _SENSOR
- _MOVE

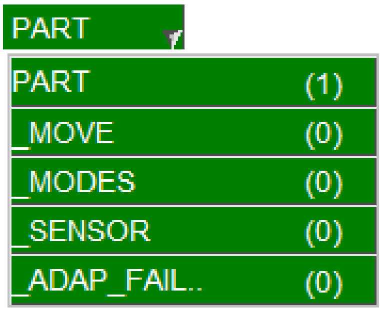
Moving elems in/out

Calculating properties

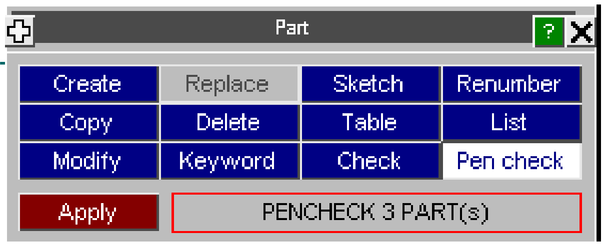
The `*PART` keyword is central to LS-DYNA usage: it acts as a "collector" of material, section property and other data for groups of elements. It is also commonly used as a means of splitting up large models into manageable components which can be given part names.

Parts in LS-DYNA are only used by **BEAM, DISCRETE, SEATBELT, SHELL, SOLID** and **TSHELL** element types, and elements of these types must reference exactly one part. Parts do not have an element type defined explicitly but, since LS-DYNA only permits a part to be referenced by a single element type, there is an implicit type associated with them. PRIMER honours this by associating an internal "type" with each part, which is defined by the first encountered element that references the part, and only material, section and other definitions valid for that element type are permitted.

***PART_(*OPTION1*)_(*OPTION2*)_(*OPTION3*)_(*OPTION4*)** cards and ***PART_COMPOSITE** cards are defined by choosing **PART** from the drop-down list in the Keywords panel in Primer. The other ***PART** types (_MOVE, _MODES, _SENSOR and _ADAPTIVE_FAILURE) are defined from their own individual selection button from the part dropdown list.



This figure shows the main **PART** control panel.
All options have their standard meanings as described in section 5.0.1.



CREATE Creating a new part

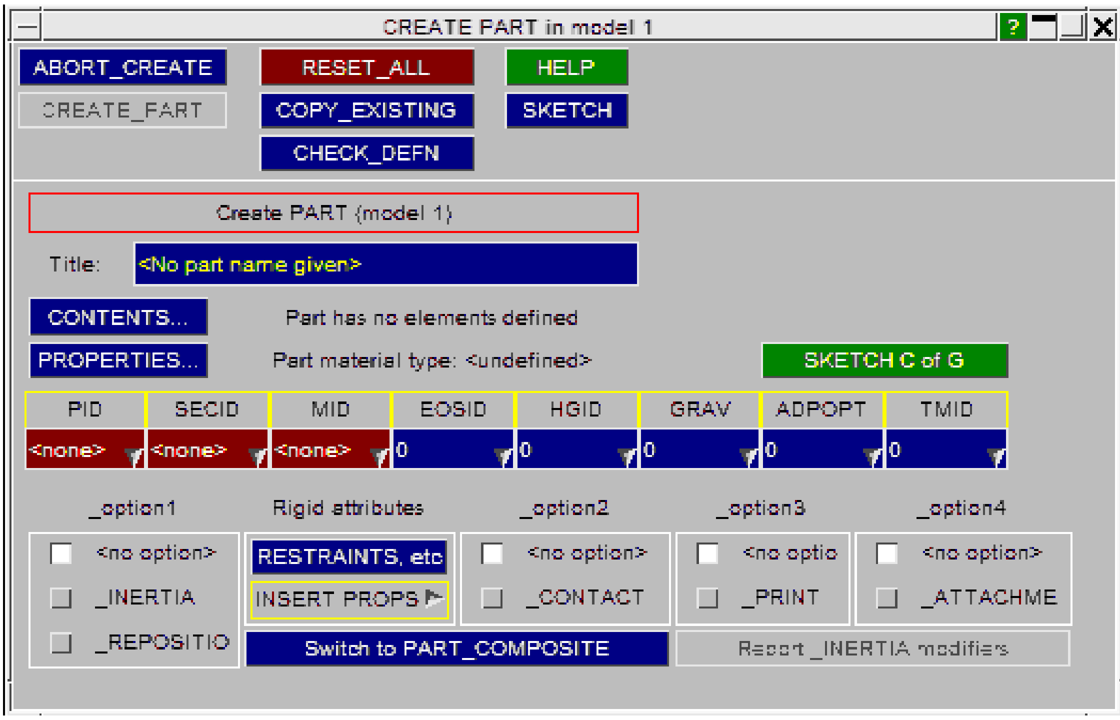
This figure shows the initial creation panel, in which nothing is defined.

In order to create the part you must define the following:

PID	The part id (label), which must be unique.
SECID	The section id. The section type must match the type of the elements in this part.
MID	Material id. Again, this must be a formulation valid for the elements in this part

This represents the bare minimum required to create a part, and you can now use **CREATE_PART** to make the definition permanent.

The remaining data fields (hourglass id, etc) are optional: limited on-line help is provided in their popup menus, but you should refer to an LS-DYNA manual for details of their meaning.



CONTENTS... The elements which make up the part. This is described below under section PART_CONTENTS.

PROPERTIES... Calculates and lists the structural properties (mass, C of G, inertia tensor) of this part - see PART_PROPERTIES

The optional sub-keywords (**_INERTIA, _REPOSITION, _CONTACT, _COMPOSITE, _PRINT**) are described in sections **PART_INERTIA** to **PART_PRINT** below.

COPY Copying selected parts.

The selected parts will be copied to the next free labels in the relevant models.

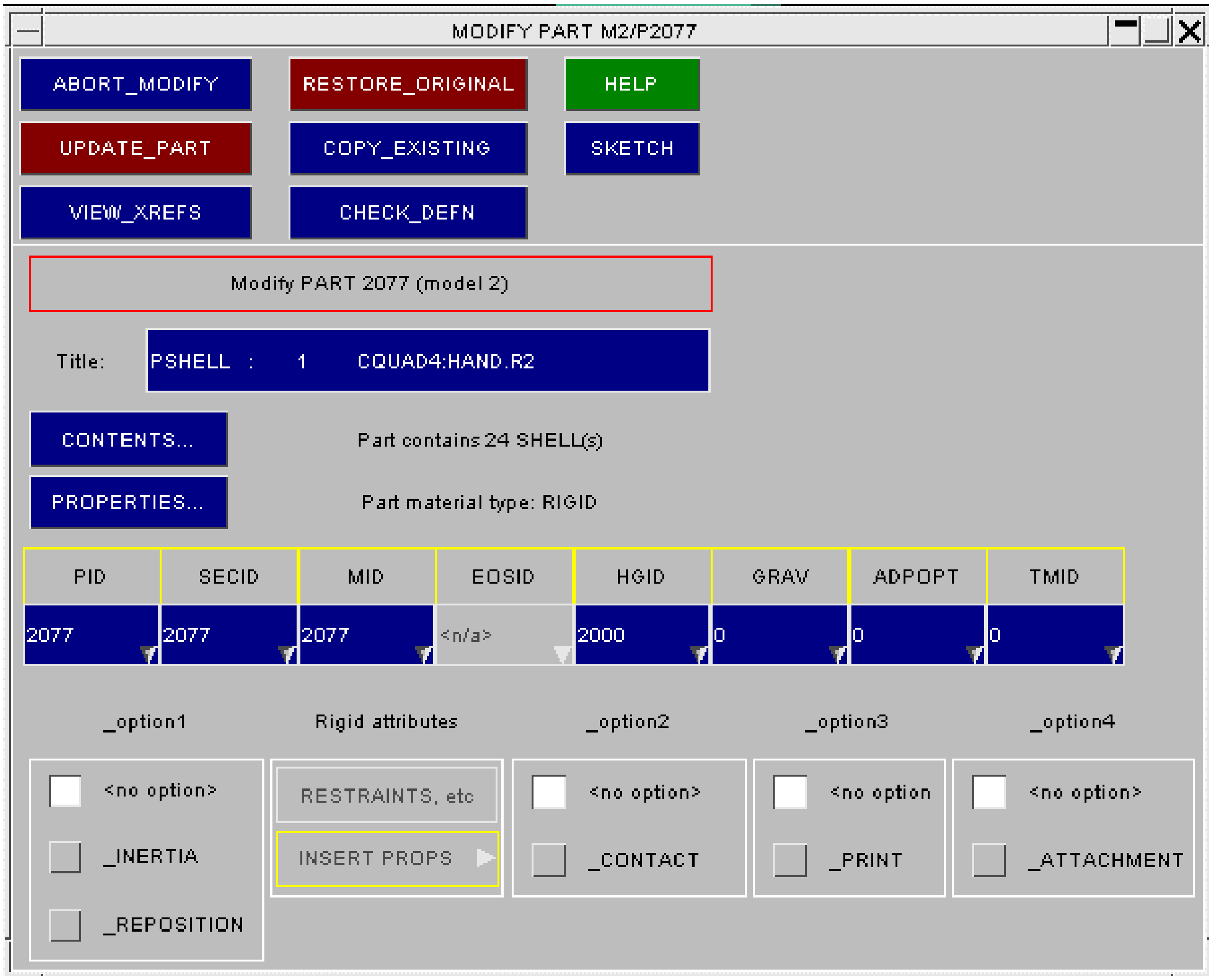
RECURSIVE_COPY is an extremely important flag in this context:

- When **OFF** Only the actual ***PART** card and the data on it are copied. The new definition contains no elements, but has the same properties as the original part.
- When **ON** Both the ***PART** card, and everything "owned by" (ie lower in the hierarchy than) the part are copied. Thus the elements and nodes will be duplicated, and the new part will refer to a set of new elements which are identical to but separate from the originals.

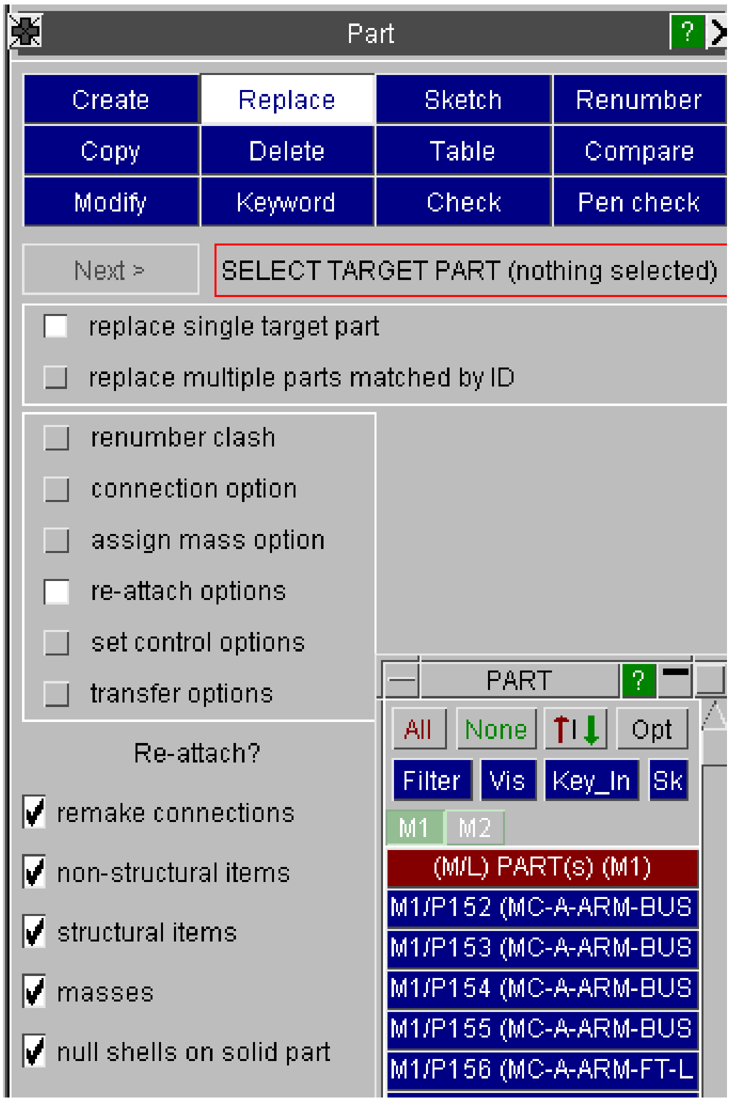
Note that the section and material definitions are ***not*** copied, since they are above parts in the programme hierarchy.

MODIFY Editing existing parts

The part editing panel is identical to the creation one, except that it will be populated with data when mapped. This figure shows a typical example, here for a part containing 24 **SHELL** elements. Changes are made as for **CREATE** above.


MODIFY PART M2/P2077
ABORT_MODIFY
RESTORE_ORIGINAL
HELP
UPDATE_PART
COPY_EXISTING
SKETCH
VIEW_XREFS
CHECK_DEFN
Modify PART 2077 (model 2)
Title:
PSHELL : 1 CQUAD4:HAND.R2
CONTENTS...
Part contains 24 SHELL(s)
PROPERTIES...
Part material type: RIGID
PID
SECID
MID
EOSID
HGID
GRAV
ADPOPT
TMID
2077
2077
2077
<n/a>
2000
0
0
0
_option1
Rigid attributes
_option2
_option3
_option4
<no option>
_INERTIA
_REPOSITION
RESTRAINTS, etc
INSERT PROPS
<no option>
_CONTACT
<no option
_PRINT
<no option>
_ATTACHMENT

REPLACE Replacing part(s) with those from another model



The process of replacing part(s)

The **REPLACE** function offers a powerful method of replacing a part or multiple parts in the target model with part(s) taken from **another** model (the source), typically a similar part remeshed or reshaped.

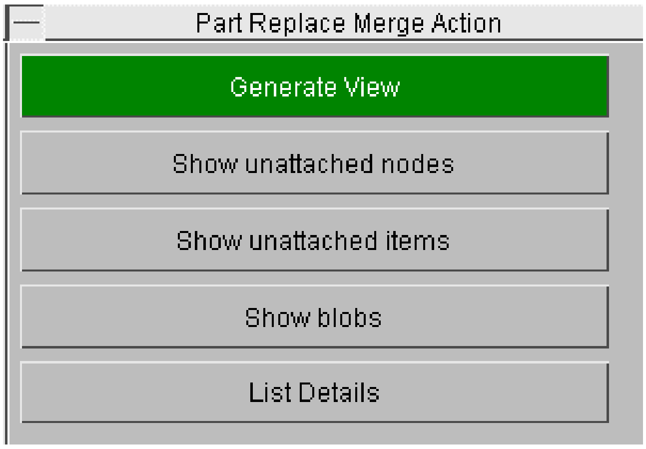
If replacing a single part, there is no need for the Part id to match.

If replacing multiple parts the target and source part ids must be the same. The element type (if specified) of the target and source parts must be the same. For multiple parts, all parts selected must have the same element type.

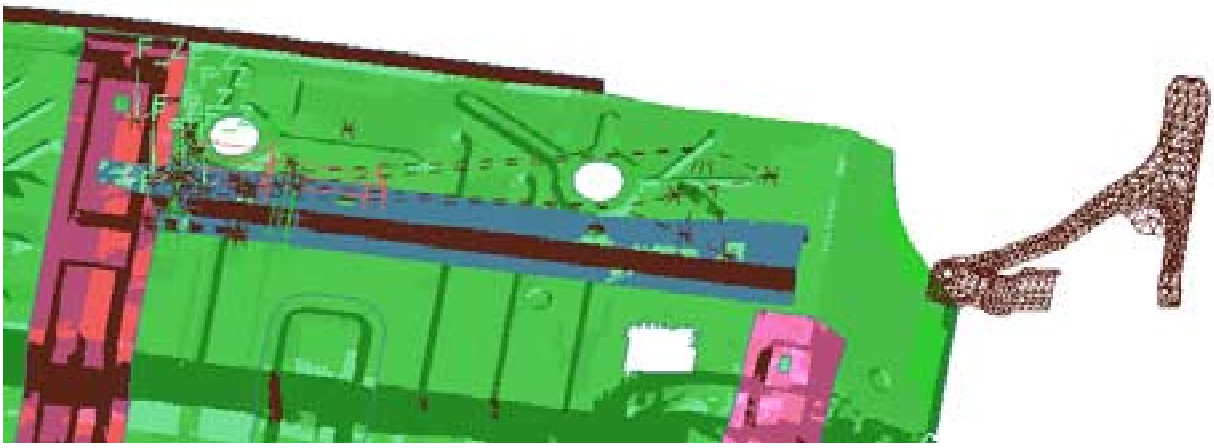
For single part, the replacement is achieved by simply selecting the target part and then the source part from the object menu.

For multiple parts, the target model is selected and then the parts from the source model which are to be imported. If the id of one or more selected source parts does not match any part is in the target model, the user is given the option of ignoring such parts or creating new ones in the target model.

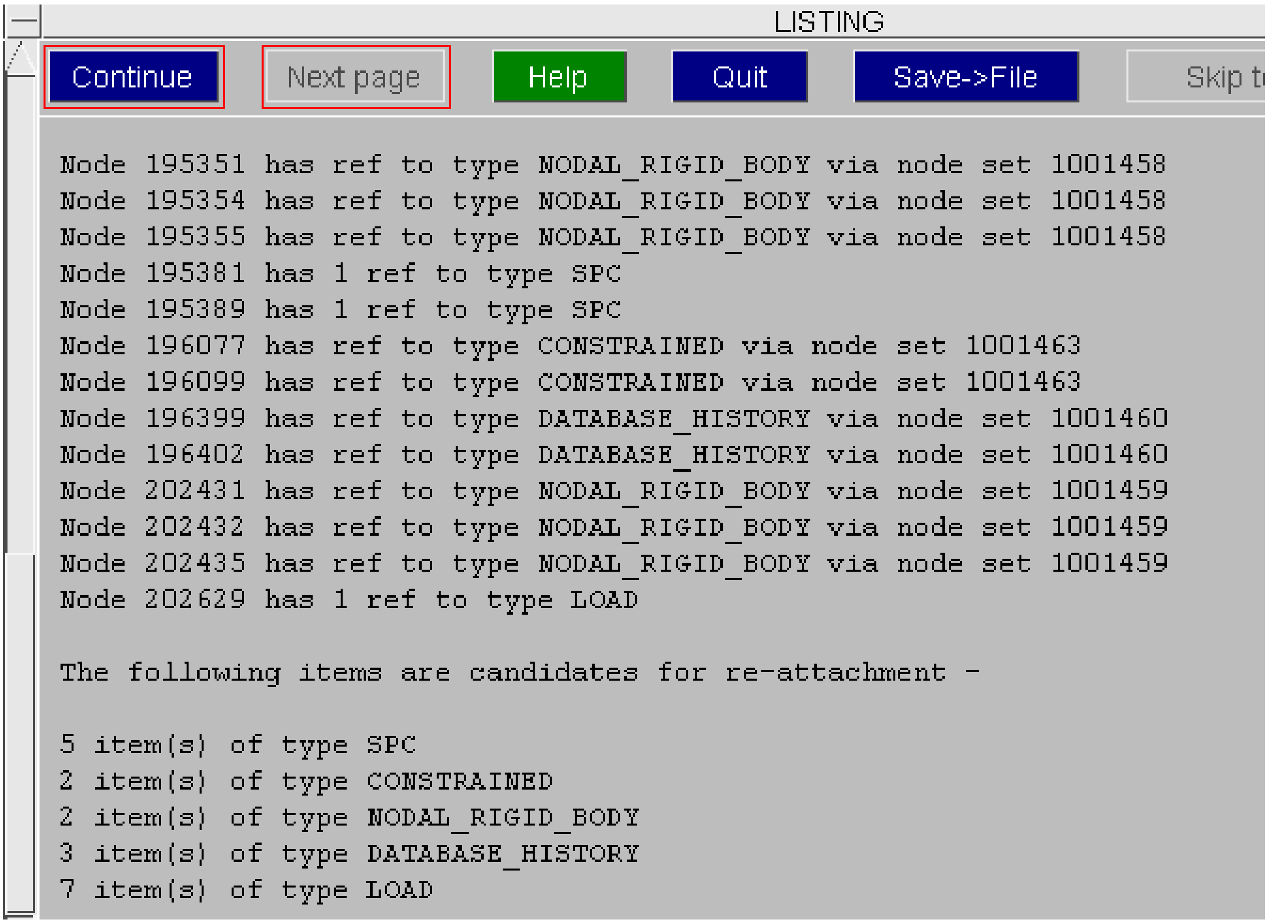
If any of the **re-attach** option are active on completion of transfer of elements & nodes into the target model, Primer will offer you a panel which allows you to view details of the possible merge options available and options on how to proceed.



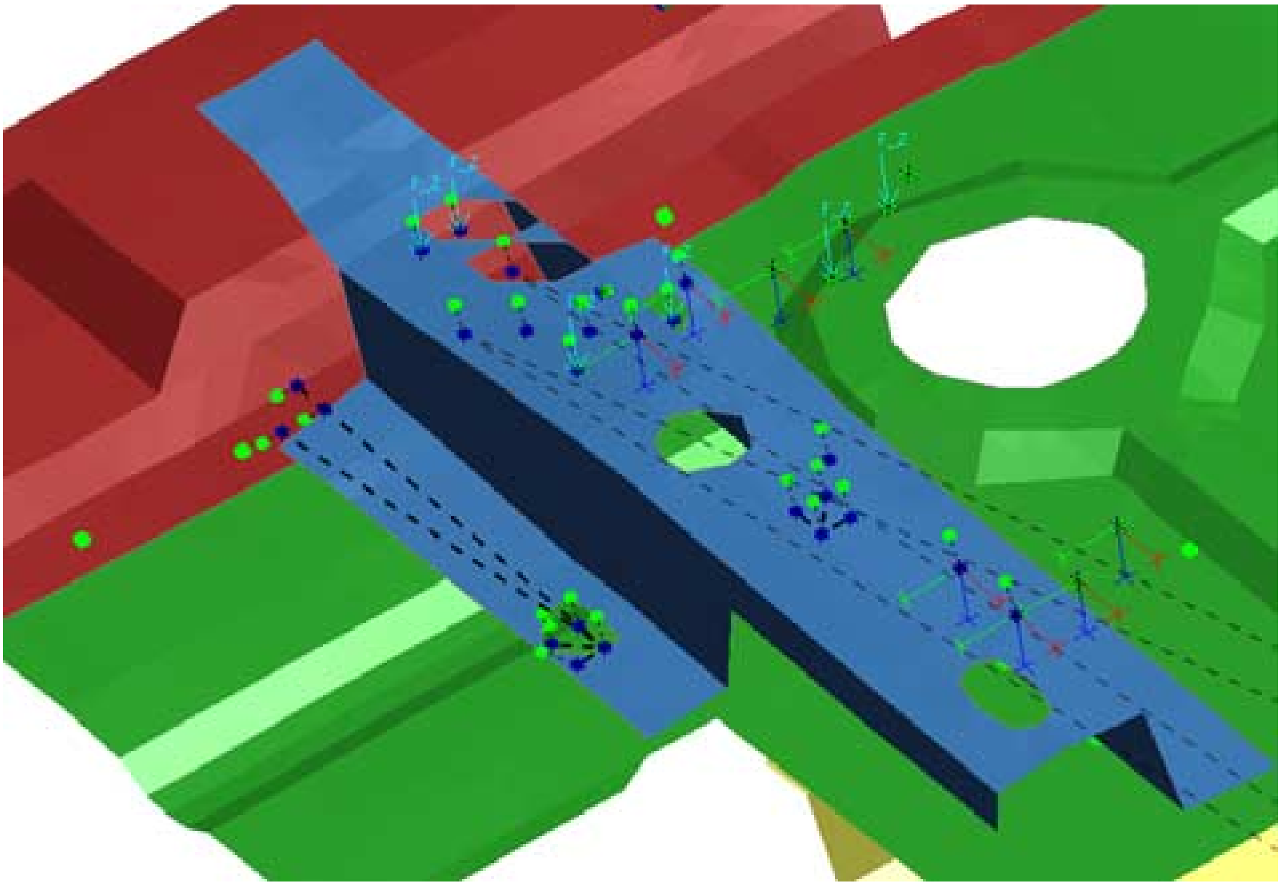
Show unattached nodes and **Show unattached items** will sketch the items concerned.



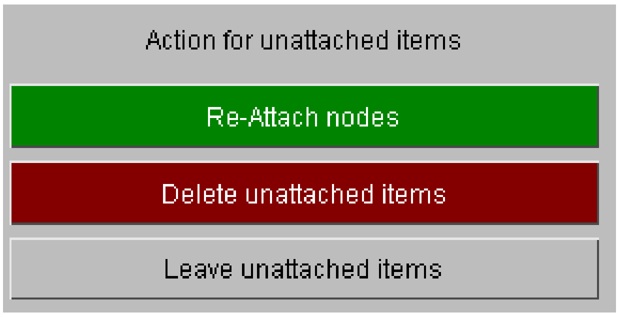
List Details will provide a detailed listing of what may require re-attachment.



Generate View will apply blanking automatically and give a blob plot to show how the nodes will be re-attached. The old nodes are plotted in blue and the corresponding new node in green. Any nodes which cannot be re-attached will be plotted in red.

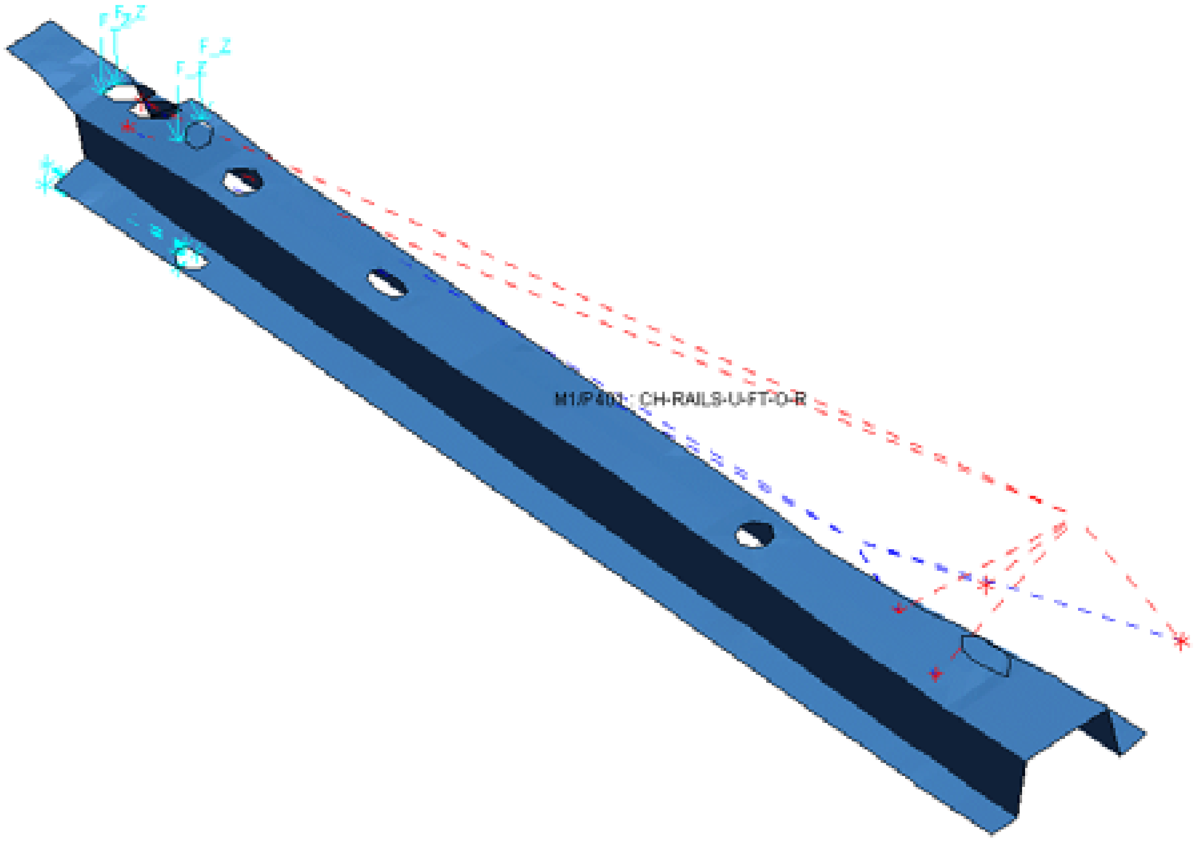


The user now has the option to **Re-Attach** the sketch items by merging the blue node to the green node, **Leave unattached** (for attention later) or **Delete unattached** items.

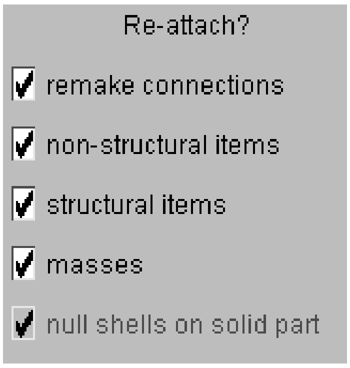


Delete unattached will bring up the interactive deletion panel, giving the user control over what is deleted.

If **Re-Attach** is applied, the user may wish to Only the parts concerned and run a Find Attached operation to check that the process has been done correctly.



Re-attach options for part replace



Remake connections Primer will remake any connections attached to the target parts, one the part replace operation is completed. Any that fail to remake will be displayed on the connection table.

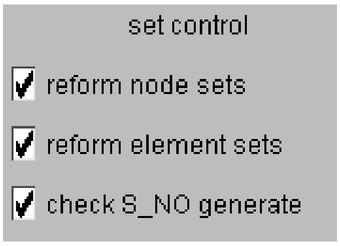
Non-structural items The nodal merge action will be offered for items such as *Initial_velocity, *Database_history_node or *Load_node (referenced directly or by conventional node set see below)

Structural items The nodal merge action will be offered for items such as node attached to structral element, node on a *Nodal_Rigid_Body or a *Boundary_spc (referenced directly or by conventional node set see below)

Masses Preserve lumped masses (excluding Assign mass) on the nodes of the target part.

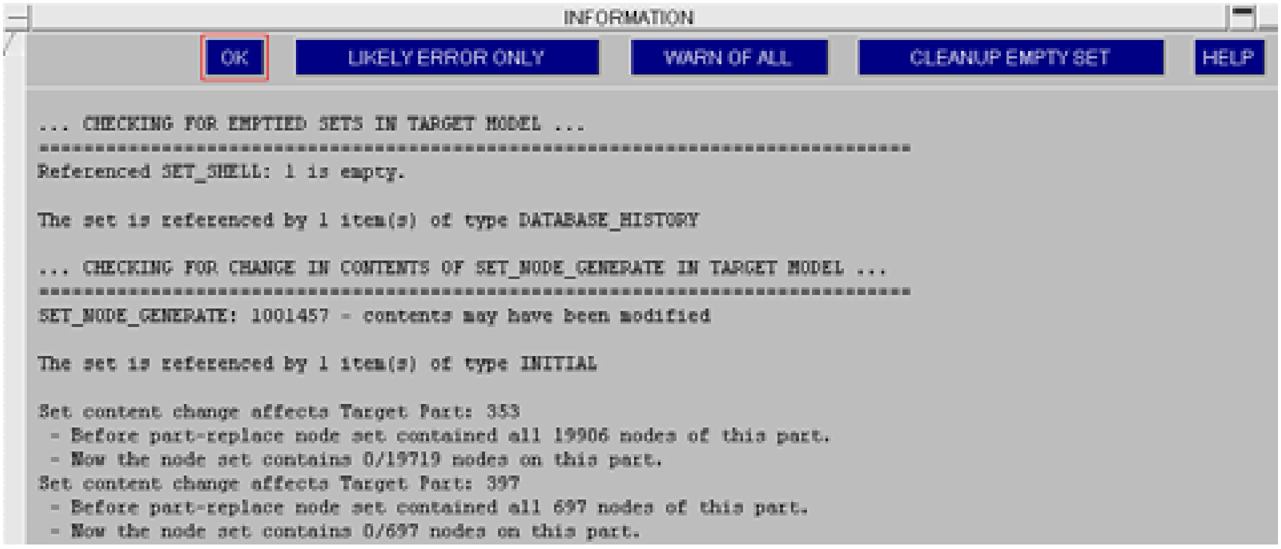
Null shells on solid part This is available for single part replace only. If active, when replacing a solid part coated with null shells, Primer will delete the old null shells and import those on the source part (if present) into the original null shell part in the target model.

Set control options for part replace

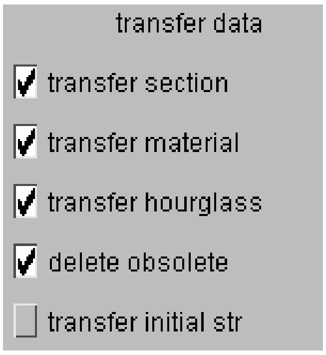


Reform node sets and **Reform element sets** These options apply only to conventional sets. This option ensures that if all nodes of a target part are in a node/element set, all nodes of the new part replacing it will also be in the set. Thus node sets for initial velocity or load can be maintained when the source part is more densely meshed than the target part.

Check S_NO generate As users may be using this type of set to deliberately control the contents Primer will make no attempt to modify the definition. It will however check the contents both before and after the part replace operation and report if there appears to be a discrepancy. Likely modelling errors are if all the nodes of a part were in the set and now some are missing, no nodes of a part were in the set and now some are, or some nodes of a part were in the set and now none are. Primer also warns of empty sets and offers to remove them.



Transfer options for part replace

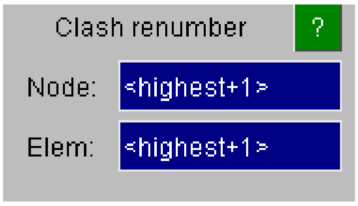


Transfer section / Transfer Material / Transfer hourglass Primer will import the section card, etc from the source model and change the target part to reference it. In the case of the Material card the load curves will be imported also.

Delete obsolete On completion of transfer, Primer will remove the original section, material, etc from the target model if it is unused by anything else.

Transfer Initial Primer will import any initial stress (or strain) cards which refer directly to elements of the source parts. This will not apply if the _SET definition has been used in the source model.

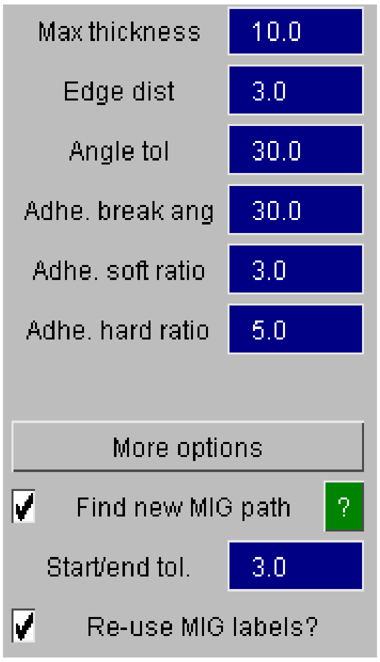
Renumber clash options for part replace



By default if any of the nodes/elements of the source part(s) clash with labels that already exist in the target model Primer will put these to highest label + 1. In models with set_generate, where an upper bound is above the highest label of any item, then the upper bound value will count as the highest label.

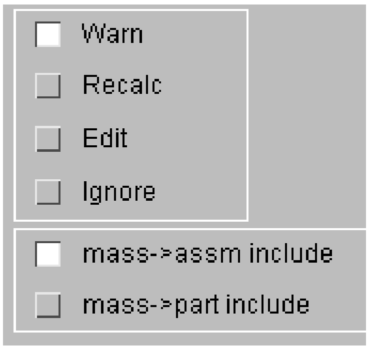
You may instead set a seed label. Note that if include label ranges are set on completion of part replace the generic relabelling routine will be applied to ant items that are out of their designated include range.

Connection options for part replace



You can set options for remaking welds and adhesive which will apply if the remake connections option is active. Additionally, you can activate/de-activate the option to reform the free edge geometry for MIG type spotwelds.

Assign mass options for part replace



If the target parts are subject to Assign mass statement , Primer will (by default) warn of this and offer the option to recalculate/modify/ignore the assign mass. This option can be pre-configured.

The include for the mases created on the imported part will by default be that of the assign mass definition (as it is if we create an Assign mass), you can, however, switch this to be the same include as the nodes of the part.

DELETE Deleting parts

The selected parts are deleted. As with **COPY** the choice of flags is very important:

The **DELETE_RECURSIVE** flag determines whether or not the items "owned" by (lower in the programme hierarchy than) the part are handled:

- When **OFF** Only the part definition itself will be deleted, and this will only happen if it is not referenced by anything else, eg elements, that depend on it. If it has any dependants then it will not be deleted.
- When **ON** Its dependants (elements, nodes, ...) will also be marked for deletion. The part itself will still only be deleted if *all* its dependent items are themselves deleted.

The **Remove from sets** flag determines whether or not a part is barred from deletion by virtue of being part of ("owned by") a **SET_PART**.

- When **OFF** If the part is in a set it will not be deleted, even if it has no dependants, although its dependants may be deleted.
- When **ON** If the only thing stopping the part being deleted is its membership of a set then it will be removed from the set and deleted.

SKETCH Sketching parts

The selected parts are sketched in white on top of the current graphics image. Parts are drawn by drawing their constituent elements

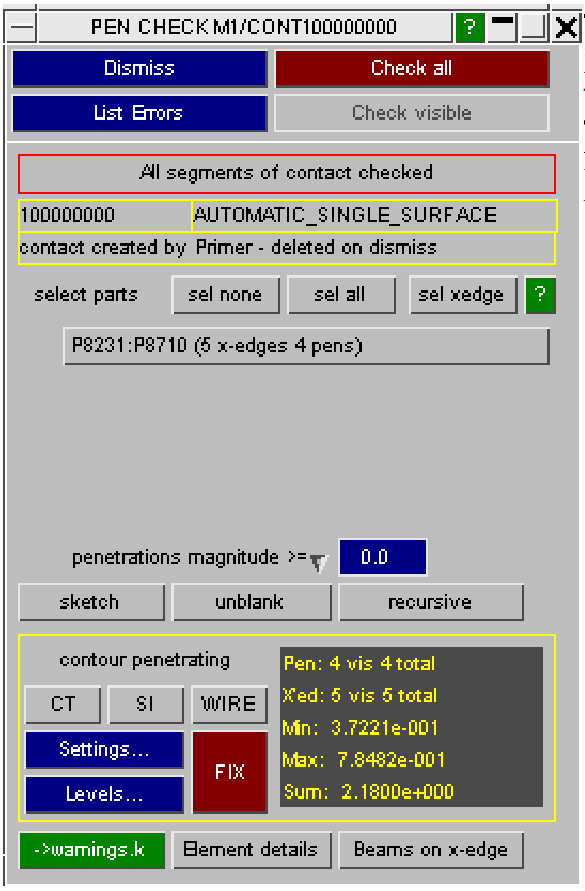
LIST Listing a summary of parts

A summary of the selected parts (name, material type, element type and #elements) is listed to the screen.

CHECK Checking parts

The selected parts are run through the standard checking routines, and any errors found are summarised on the screen.

PENCHECK Checking parts for penetrations



The selected parts will be put into a "private" automatic single surface contact and checked for penetrations. The standard penetration check panel is displayed.

This will allow you to contour any penetrations between the panels or if you switch mode, to contour gaps between panels.

When you dismiss the panel the vapid contact will be deleted.

RENUMBER Renumbering part labels

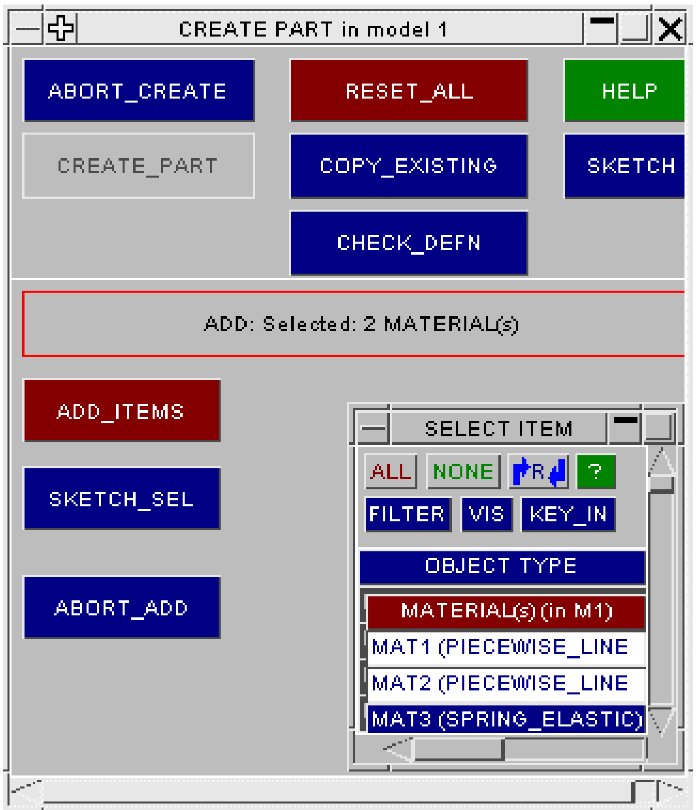
The standard renumbering panel is mapped for the relevant model, allowing part labels to be updated. To update a single part label it may be easier just to **MODIFY** it.

PART_CONTENTS... Adding and removing elements to and from parts

Both the **CREATE** and **MODIFY** functions use the **CONTENTS...** option to list part contents, and also to add and remove elements.
This maps the content editing sub-window, in which you can

ADD_ITEMS	To specify elements to be added to the part;
REMOVE_ITEMS	Specify elements to be removed from the part.
EMPTY_PART	Empty the part of all elements.

In both the **ADD** and **REMOVE** cases the standard object selection menu is shown, and you select (by element, part. model, or anything else) what is to be added or removed.



For example in this **ADD** case the user has selected two materials.

ADD_ITEMS	Adds the elements in these materials to the list of elements for this part.
SKETCH_SEL	Sketches the items that have been selected, to confirm they are correct.
ABORT_ADD	Aborts the ADD operation.

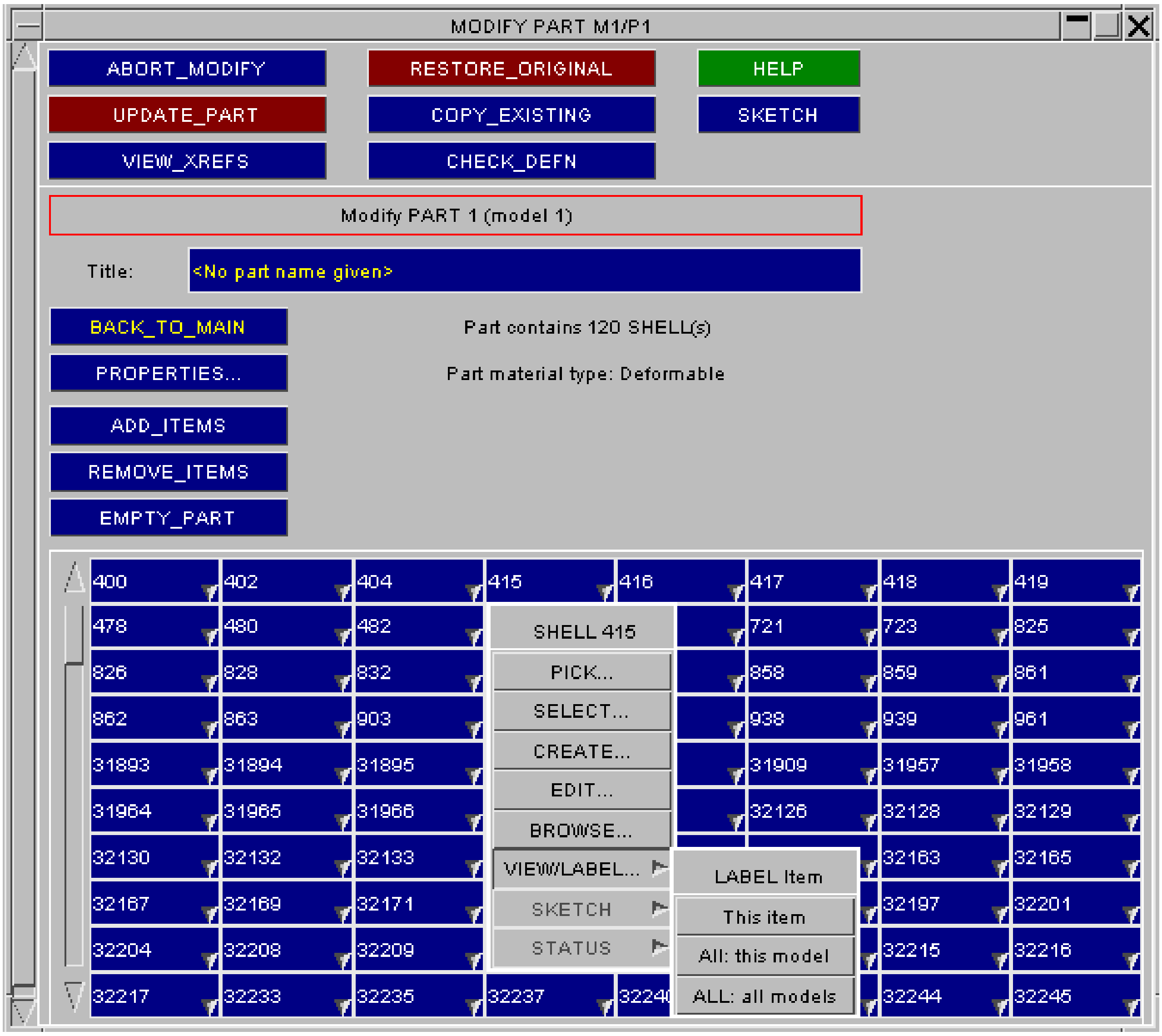
If the part already contains elements then its implicit type is already defined, and only elements of the relevant types are selected for processing.

If the part is empty then the first element type encountered in the addition list is used to determine its type.

Therefore when creating a new part take care to choose elements of the correct type. However when adding to or subtracting from an existing part you can choose broader categories, eg materials as here, knowing that only the relevant items will be selected for addition or removal.

It is legal to select for addition elements that are already in the part: they will not be duplicated since the outcome of an **ADD** operation is a logical (inclusive) OR of the existing and new elements.

Once the part contains some elements these become visible in the **CONTENTS...** sub-panel as shown below, and can be scrolled through individually.



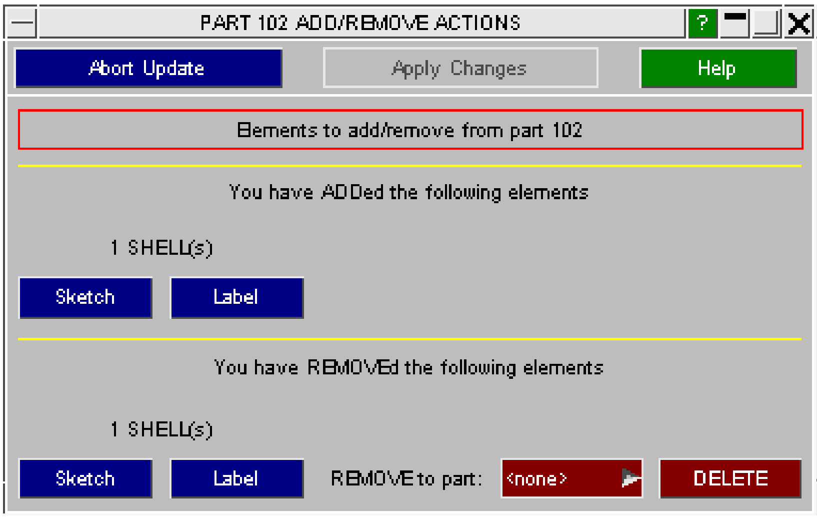
Popup menus are available for every element, giving a list of options that allow you to examine the elements in more detail. Here the user is about to look in detail at shell 415 using the **VIEW/LABEL** option.

> Note that while creating or editing a part, as elsewhere in PRIMER, you are always working with a "scratch" definition of the part.
> Elements are not actually transferred to or from this part at this stage, as its permanent definition is not updated until you **CREATE** or **UPDATE** it explicitly. At that time you will be required to confirm the transfer of elements between parts, and to determine how elements deleted from this part are to be disposed of. This is described in section PART_UPDATE below.

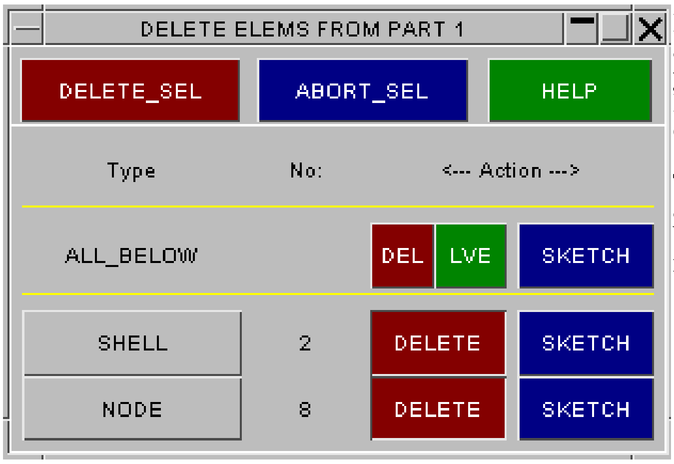
PART_UPDATE How element addition and deletion is processed on update/create

When part operations have added elements to or removed them from a part this only takes place in the scratch definition, so some extra processing is required when the permanent part definition is updated on **CREATE** or **UPDATE**.

This is done via the **ADD/REMOVE ACTIONS** panel, as shown here, which is automatically displayed.



In this example one shell has been added to this part and one has been removed from it.
Before user can apply the change he must determine what to do with the elements flagged for removal. These must be deleted or moved into another part.

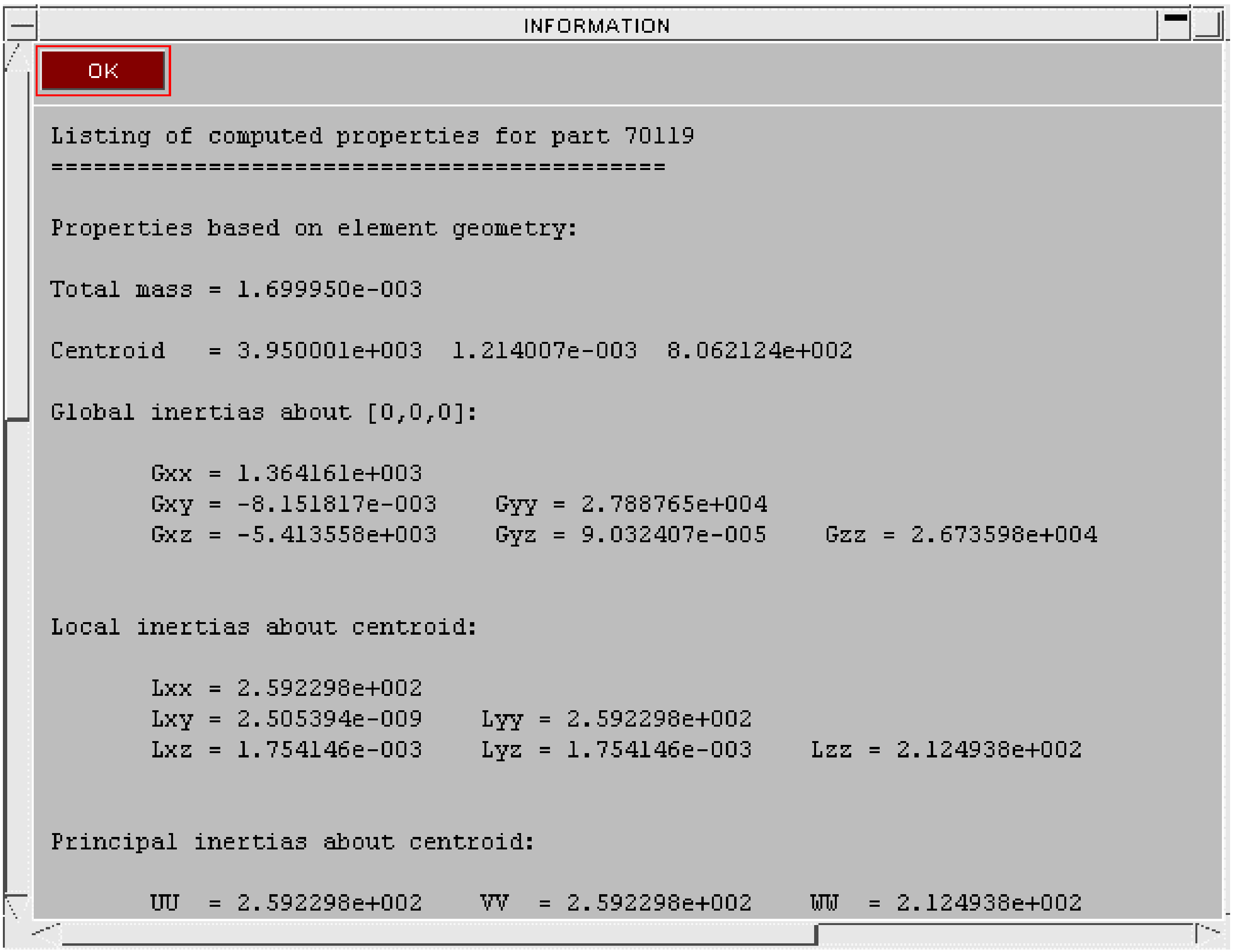


If the deletion option **DELETE** is chosen the elements and any associated items will be marked for deletion, and you will get the standard deletion confirmation panel.
In this example 2 shells are flagged for removal, and they own 8 nodes.

These will be deleted from the part once you confirm the action. If you abort the deletion the part update will be blocked as it cannot proceed without corrupting the model.

PART_PROPERTIES Calculating and displaying structural properties.

The **PROPERTIES...** command calculates the mass, C of G and Inertias of the part based on its known element properties. A typical output is shown:



"Known element properties" means:

- Element topologies must be defined, and node coordinates specified.
- Section definitions for shells and beams must be defined.
- Material definitions must be defined for all types to give density values.

Note the following exclusions from property calculations:

Any contributions due to items not in this part, but which are connected to it, are ***ignored.*** An example would be **`*ELEMENT_MASS`** and **`*ELEMENT_INERTIA`** elements which are connected to nodes on elements in this part, but not formally of it.

"Rigid" parts (see **PART_RIGID** below) for which mass and/or centroid and/or inertia have been externally specified still have their properties calculated on the basis of element properties, ***not*** on the stipulated values, and also do ***not*** include the effects of any rigid body merges.

PART_RIGID Parts using rigid materials (*MAT_RIGID)

"Rigid" parts in LS-DYNA are those which use the special rigid material **`*MAT_RIGID`,** and they have a whole range of special attributes which make them computationally efficient, but which can complicate their use. First a bit of theory - skip this if you already know about rigid bodies:

Rigid vs Deformable parts - some theory.

"***Deformable***" elements in LS-DYNA have their mass lumped at their nodes, and at each timestep the forces acting at each node are determined and Newton's 2nd law:

F = M.a (Force = Mass x Acceleration)

is used to determine the acceleration vector of each node, then by integration the velocity and displacement vectors are also found. Strains in elements arise due to differential displacements at nodes, and from these stresses are calculated leading to further forces for the next time-step. Each element can deform independently.

"***Rigid***" elements are treated quite differently: the total mass, C of G and inertia for each rigid part is calculated. Then at each timestep the forces acting on the centroid of this part are summed from all sources, and the resultant displacement vector calculated from its total mass and inertia terms. The resulting motion is then extrapolated to each node such that the whole part moves as a "rigid body", with no relative displacement between nodes. All element strains and stresses are zero.

This leads to the following special properties of rigid bodies:

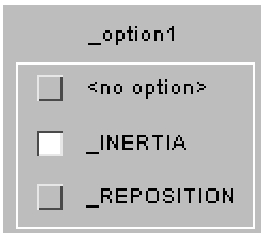
- Because in the "rigid" case motions are imposed at each node (the "a" part of Newton's equation above) by the rigid body formulation, all six degrees of freedom at each node on a rigid body are effectively constrained. Therefore no other conflicting constraints may act upon them.
- Although the default is for the properties (mass, etc) of a rigid part to be computed from its constituent elements this need not be the case, and these properties can be overwritten. The **`_INERTIA`** keyword appended to a part allows some or all of these properties to be defined.
- Rigid parts can be merged together such that one or more "slave" parts become subsumed into a "master", and all have their motion updated together during an analysis. The ***CONSTRAINED_RIGID_BODIES`** card defines these merges.

This is not a complete treatment of rigid bodies in LS-DYNA, and you are referred to the theory, example and user manuals for more information, but it suffices for the purposes of explaining the options below.

PRIMER provides some special facilities for processing and checking rigid parts.

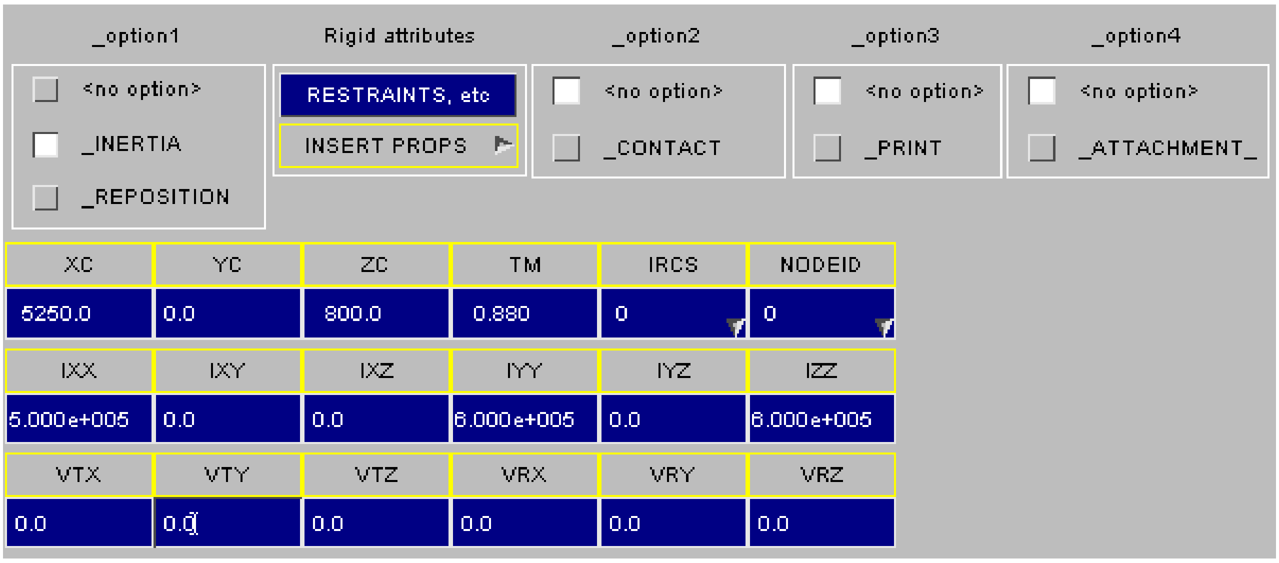
PART_INERTIA Overriding calculated properties for rigid parts.

The sub-keyword **`_INERTIA`** appended to a rigid part definition means that some or all of its mass, centre of gravity and inertia tensor terms are externally defined. For such a part no mass, C of G or inertia properties are calculated and the externally supplied values are used.



Initial velocities may also be defined: if they are not then the part is assumed to have no initial velocity.

PRIMER reports the rigid property fields in the create/edit panel, as shown below, and you are free to update any of these.



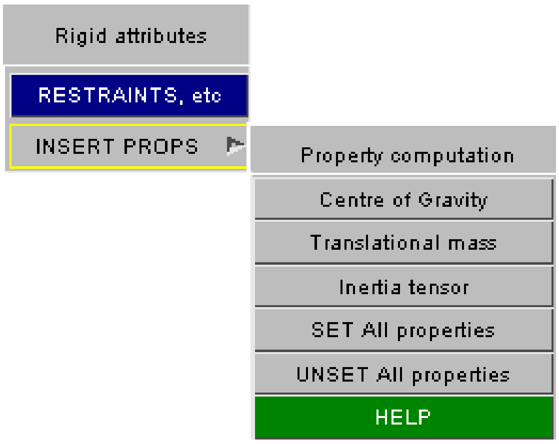
In this example the C. of G. (**XC, YC, ZC**), mass (**TM**) and inertia tensor (**Ixx .. Izz**) have all been defined. The tensor here is in the global system: a local system can be defined by setting **IRCS** to 1, then defining local axes.

Rigid attributes Calculating and (re-)setting rigid properties and restraints.

(At present **RESTRAINTS etc**, which will help with the definition of restraints and constraints for this rigid part, is not available. This is required since restraints and constraints applied directly to the nodes of rigid parts do not work, and these must instead be applied to the part itself. In addition rigid body merges, prescribed motion, "stoppers", and so on all merit special attention.)

INSERT PROPS calculates properties from the elements of this part, (exactly as described in PART_PROPERTIES above), and allows you to insert some or all of these as the "externally" defined terms above.

Use the popup menu, as shown here, to select which properties to overwrite.



Notes on using "rigid" parts

The definition of a "rigid" part is one that uses material type ***MAT_RIGID**.

- You should still give sensible density, Young's modulus and section properties for rigid parts. This is because these values are used when computing stiffness and geometry (for shells) for contact, and also mass, C of G and inertia properties if these are not externally defined.

- **_INERTIA** definitions are optional for rigid parts, and if omitted the properties will be calculated from the part's elements as described above.
- **_INERTIA** definitions *cannot* be defined for non-rigid parts. PRIMER will allow you to specify them in the editing panel on the premise that you will subsequently change the material type. However checking on **UPDATE**, or when using **CHECK_DEFN**, will flag this as an error.
- The rule that parts can only contain one type of element still holds true for rigid parts. To assemble a rigid body containing different element types create one (or more) parts for each element type, then merge them together using ***CONSTRAINED_RIGID_BODIES**. The parts need not be physically connected in any way.
- Rigid parts may have common nodes with other non-rigid parts, but they may not share common nodes with other rigid parts *unless* the parts have been merged as above. (Otherwise the common nodes would be subject to multiple constraints.)
- "Extra" nodes, not necessarily attached to any elements, may be added to rigid parts using ***CONSTRAINED_EXTRA_NODES**. These have their positions updated by the rigid body equations and can be extremely useful for connecting to rigid bodies.
- The LS-DYNA manual claims that rigid parts for which mass, C of G and inertia properties are defined need not contain any elements. In practice this seems not to work, and it is recommended that you have at least one element in a rigid part, even if it is a dummy that serves no purpose.

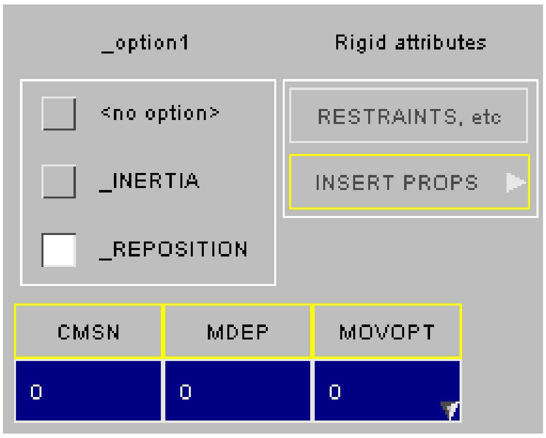
The part checking functions detect many errors associated with the use of rigid bodies, but for a comprehensive check it is necessary also to run the ***NODE** and ***CONSTRAINED** checking functions. These will, for example, detect multiple constraints on nodal degrees of freedom, and attempts to apply "rigid" constraints to non-rigid parts.

The **MODEL** > **CHECK** command, which runs all checking routines, will perform these checks for you.

PART_REPOSITION Special options for coupled analyses

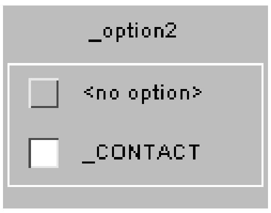
This is a specialist option which applies only to deformable parts that are to control the motion of rigid components in a coupled (CAL3D, MADYMO) analysis.

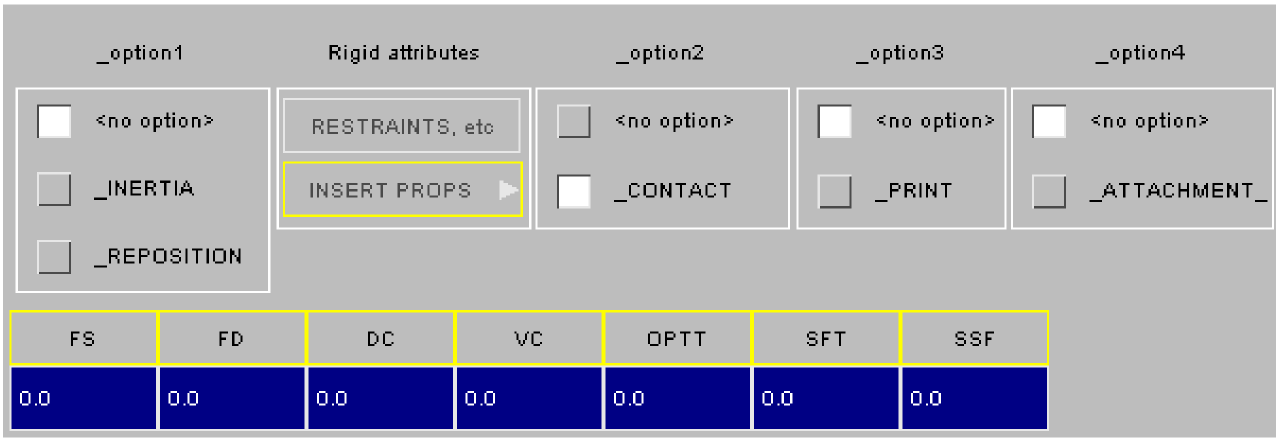
Refer to the LS-DYNA user manual for more information.



PART_CONTACT Specifying part-specific contact parameters for part-based contact.

When part-based contact (using parts or part sets) is used for the "automatic" contact types it can be convenient to have specific contact parameters for each part which supersede the default ones on the ***CONTACT** card.





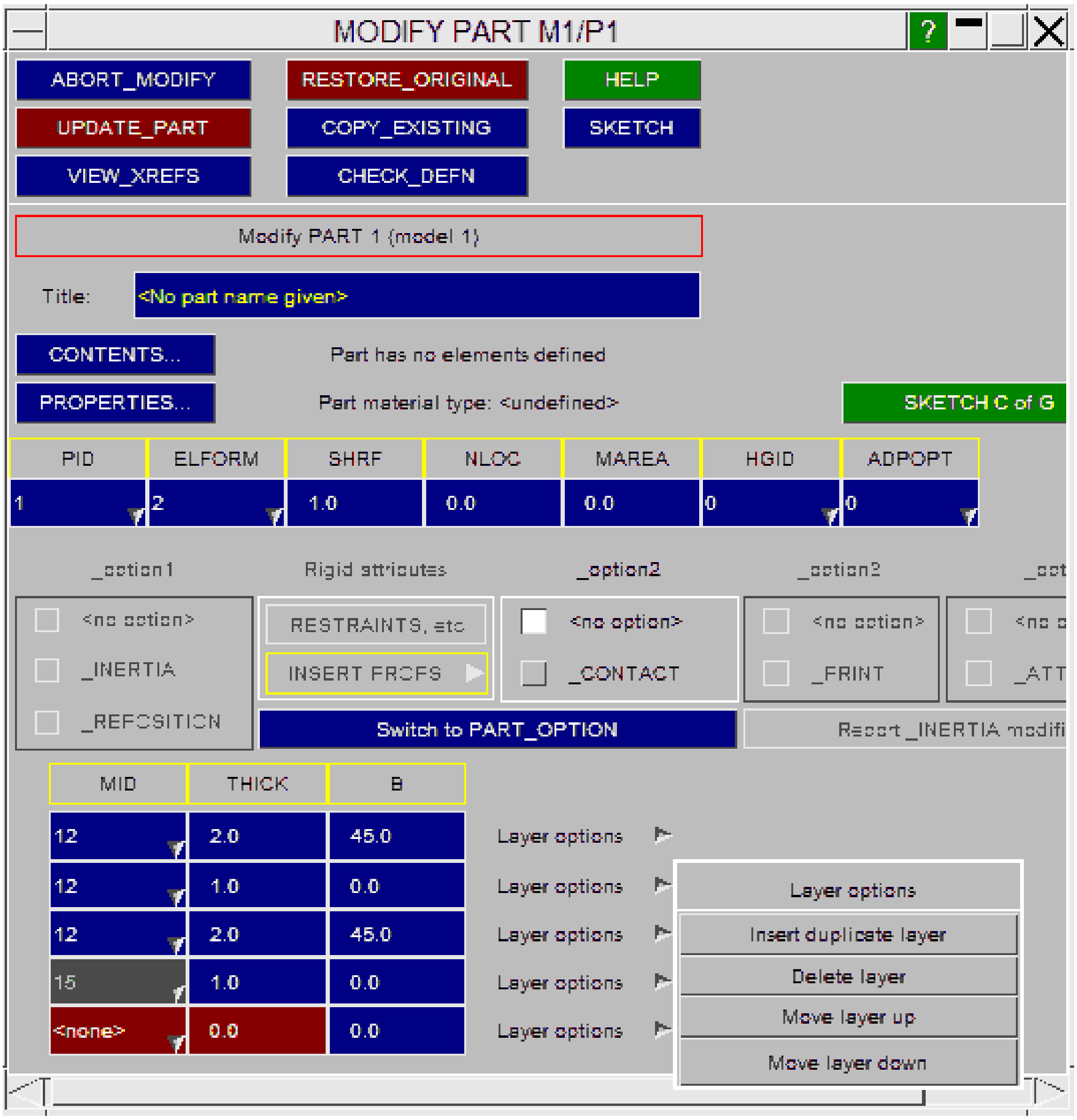
PRIMER allows you to set the **_CONTACT** parameter and define its values:

Refer to the LS-DYNA manual for the exact meaning of these parameters, and also the contact types to which they apply.

PART_COMPOSITE Specifying composite layers within a part.

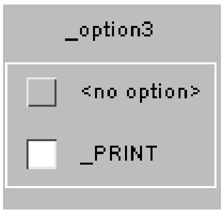
The part composite option changes the part panel to allow the user to create layers within the part. The MID and SECID are removed because it is all contained within the part cards

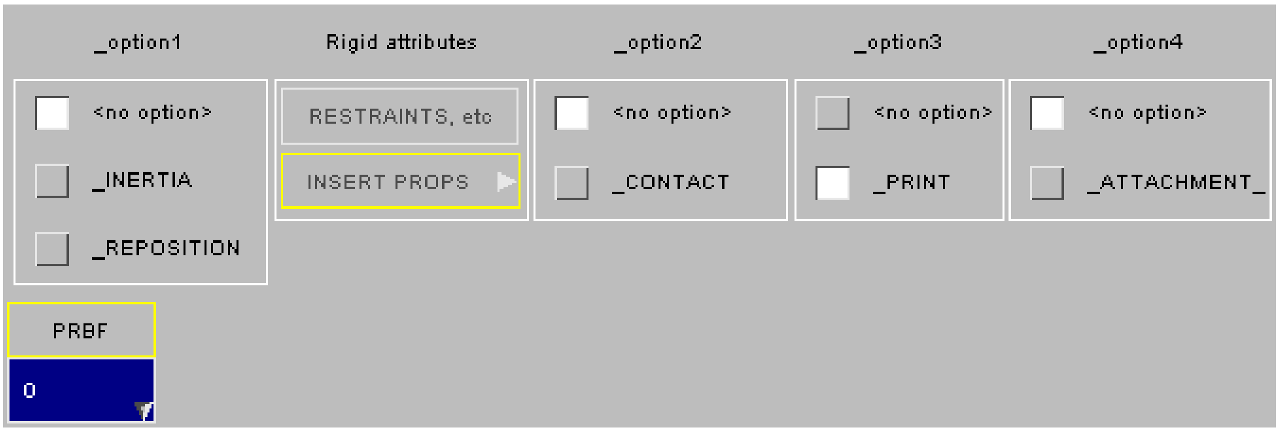
The user can enter layer information at the bottom of the panel, and it is possible to move layers up / down and create duplicate layers using the right-click popup to the right of each layer.



PART_PRINT Allows user control over whether output is written into the ASCII files MATSUM and RBDOUT.

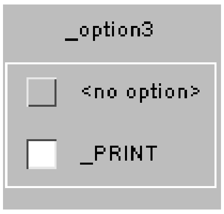
PRIMER allows you to set the `_PRINT` parameter and define its options. Refer to the LS-DYNA manual for the exact meaning of these.

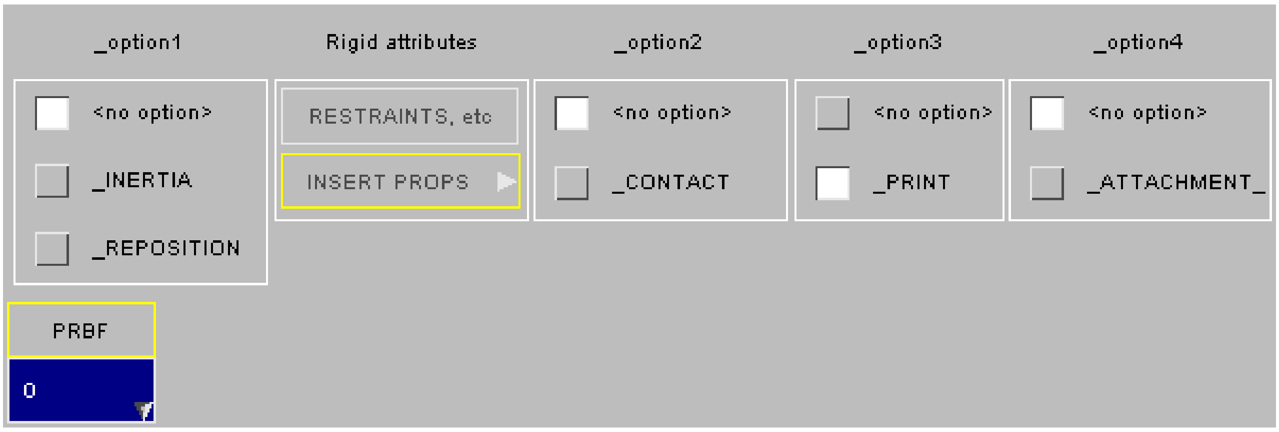




PART_ATTACHMENT_NODES Allows user control over which nodes are treated as attachment nodes.

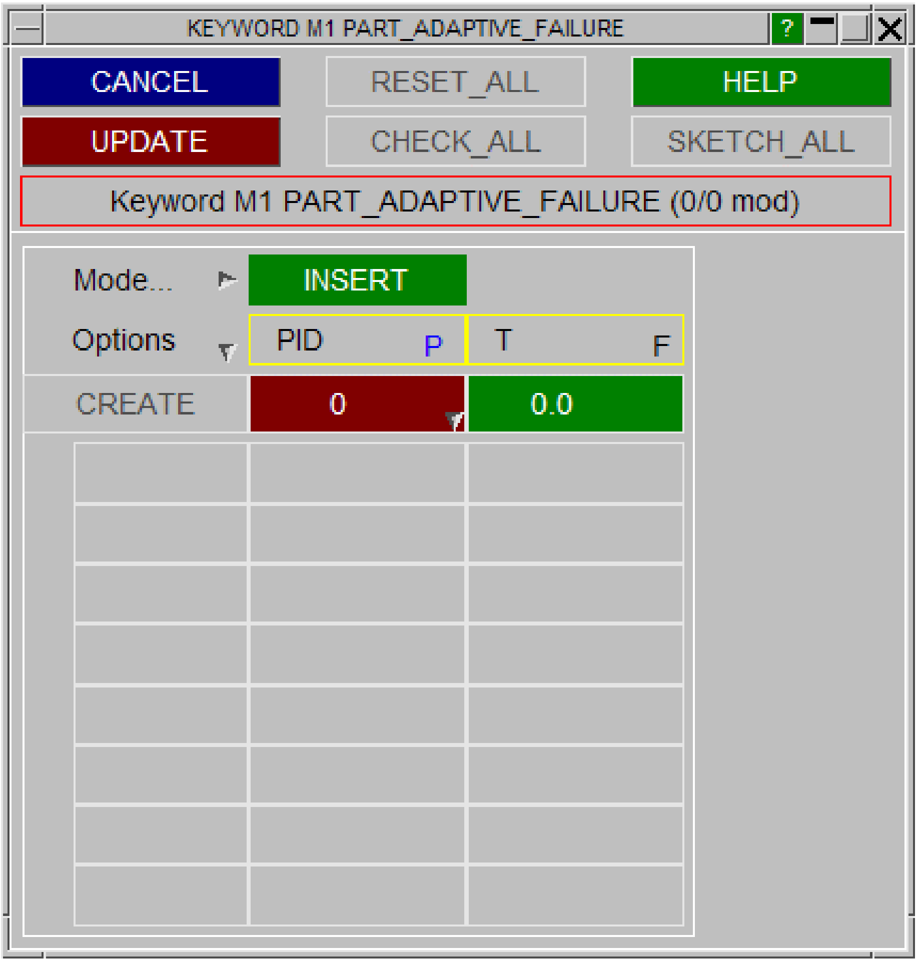
PRIMER allows you to set the **`_ATTACHMENT_NODES`** parameter and define its options. Refer to the LS-DYNA manual for the exact meaning of these.





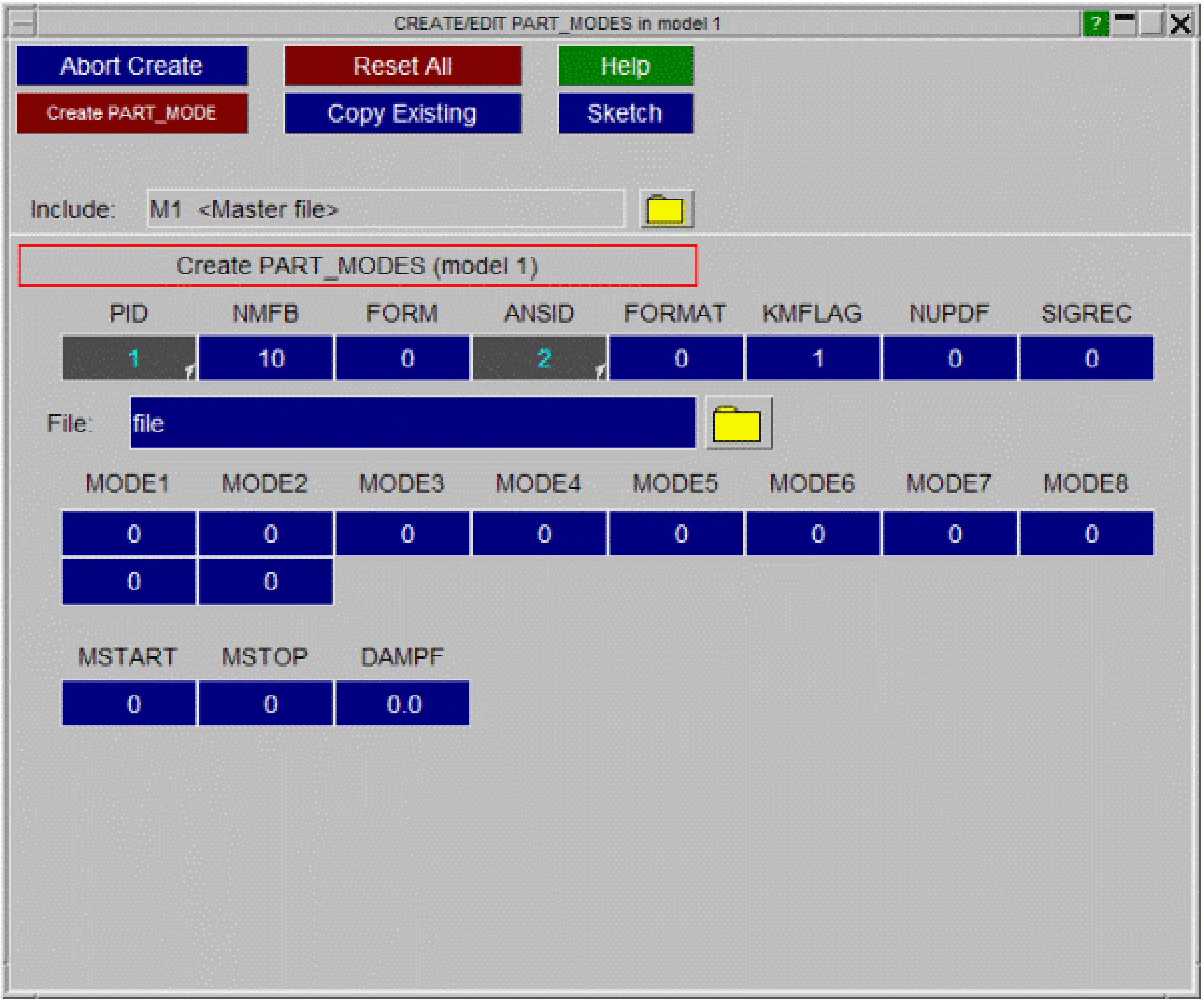
PART_ADAPTIVE_FAILURE Allows user control over which nodes are treated as attachment nodes.

PRIMER allows you to set up a **_ADAPTIVE_FAILURE** card. Refer to the LS-DYNA manual for the exact meaning of this card.



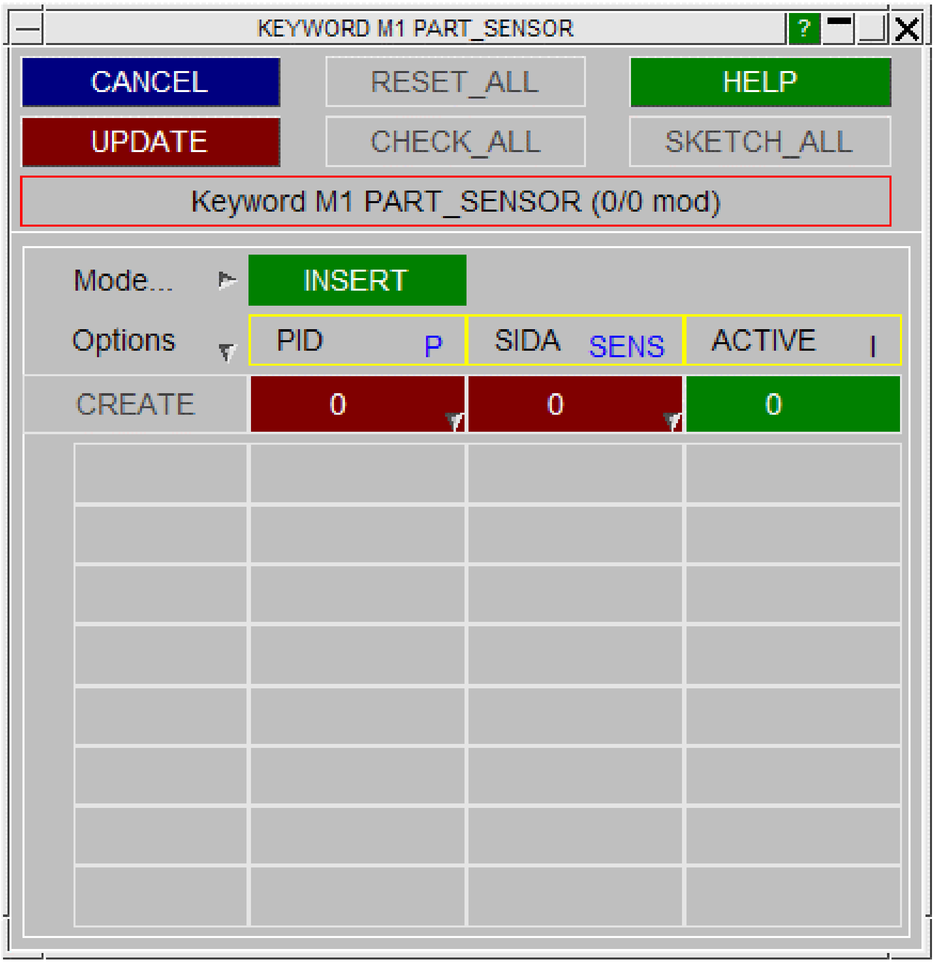
PART_MODES Allows user control over which nodes are treated as attachment nodes.

PRIMER allows you to set up a **_MODES** card. Refer to the LS-DYNA manual for the exact meaning of this card.



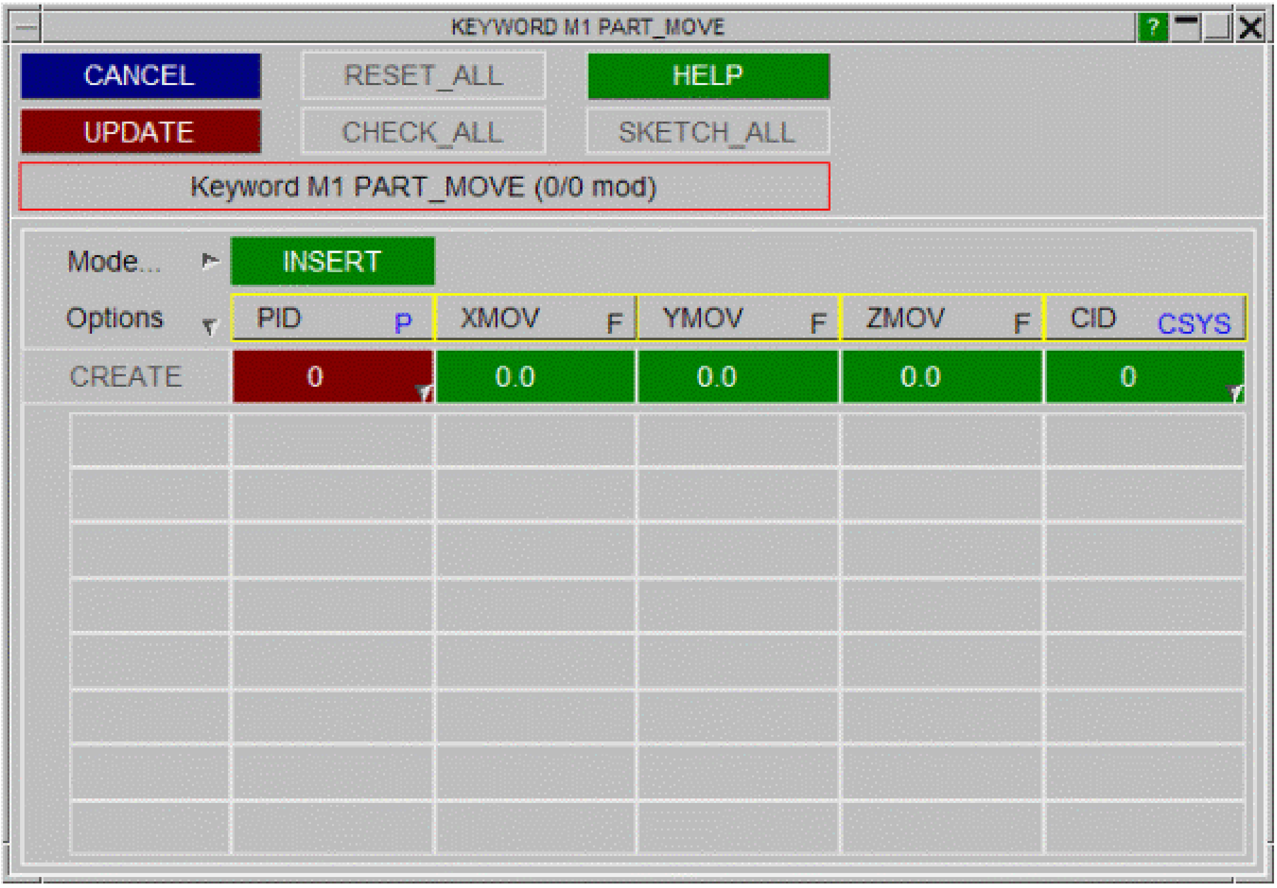
PART_SENSOR Allows user control over which nodes are treated as attachment nodes.

PRIMER allows you to set up a _**SENSOR** card. Refer to the LS-DYNA manual for the exact meaning of this card.



PART_MOVE Allows user control over which nodes are treated as attachment nodes.

PRIMER allows you to set up a **_MOVE** card. Refer to the LS-DYNA manual for the exact meaning of this card.

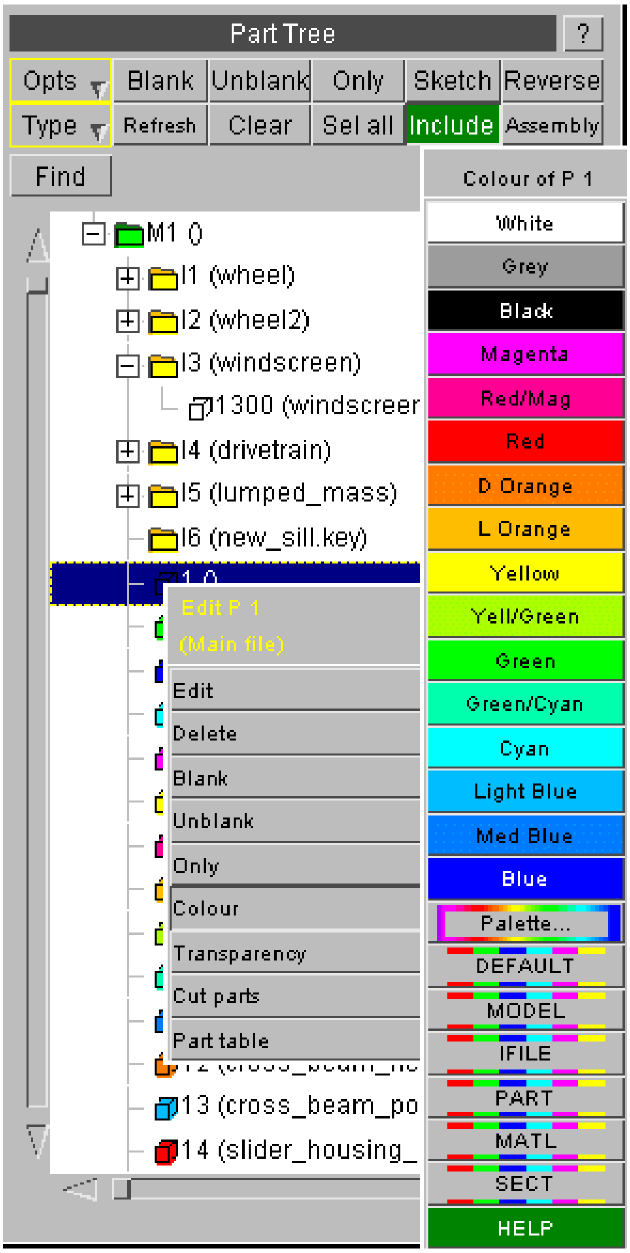


Visualising and Labelling parts

Parts are not drawn explicitly, but PRIMER can draw parts, by drawing their constituent elements, and can also label them by adding part ids to element label strings. Part visibility and labelling is therefore inherent in element visibility, controlled in **ENT**ity Viewing.

Parts may also be sketched in wireframe mode on the current image using the main and create/edit **SKETCH** options above.

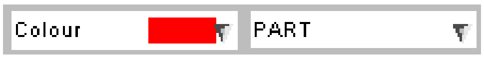
COLOUR Colouring parts



Colours of parts can be set in two ways. The first method (as shown to the right) is to locate the part in the part tree, right click to bring up the edit pop-up menu and then select the desired colour option. You can select:

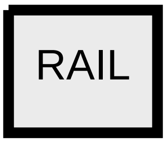
- A constant colour from the predefined range
- An arbitrary constant colour using **Palette**. This allows mixing of a colour from the full range of shades supported by the hardware.
- The **DEFAULT** colour for this item. The actual colour(s) are a function of item type, i.e. an element with a part id will inherit the colour of its part. Other items are based on their constituent sets or perhaps their labels.
- **MODEL** will apply the colour based on **Model** number modulo 13 (standard primer sequence red, green, blue etc).
- **IFILE** will apply the colour based on **Include file** number modulo 13 (standard primer sequence red, green, blue etc).
- **PART**, **MATL** (material) and **SECT** (section) colours only apply to element types with a

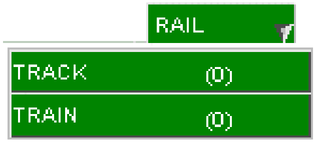
Alternatively, it is possible to use the Quick Pick tool. The same options as above are available which will then be applied to selected parts:

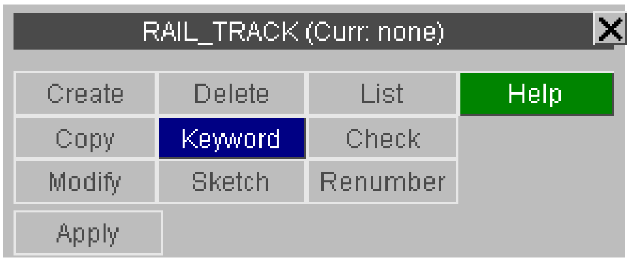


PERTURBATION

There are currently two sub-keywords available, NODE and SHELL_THICKNESS, as shown on the left. These can be edited through the generic Keyword Editor.







There are currently two sub-keywords available, TRACK and TRAIN, as shown on the left. These can be edited through the generic Keyword Editor.

RIGIDWALL: Defining Rigid ("stone") Walls.

Top level menu
Create/Edit
Keyword edit
Dragging walls
Editing planes
Visualisation

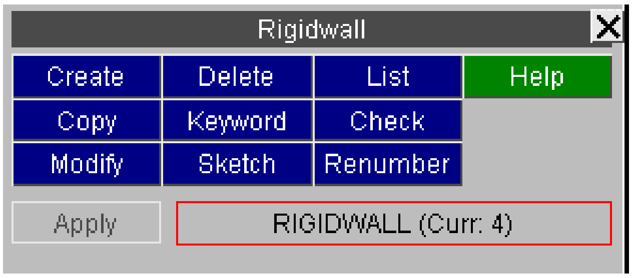
Rigidwalls in LS-DYNA are convenient and computationally cheap rigid geometrical shapes against which nodes on a deformable body can impact.

The ***RIGIDWALL** keyword in LS-DYNA supports the following sub-types:

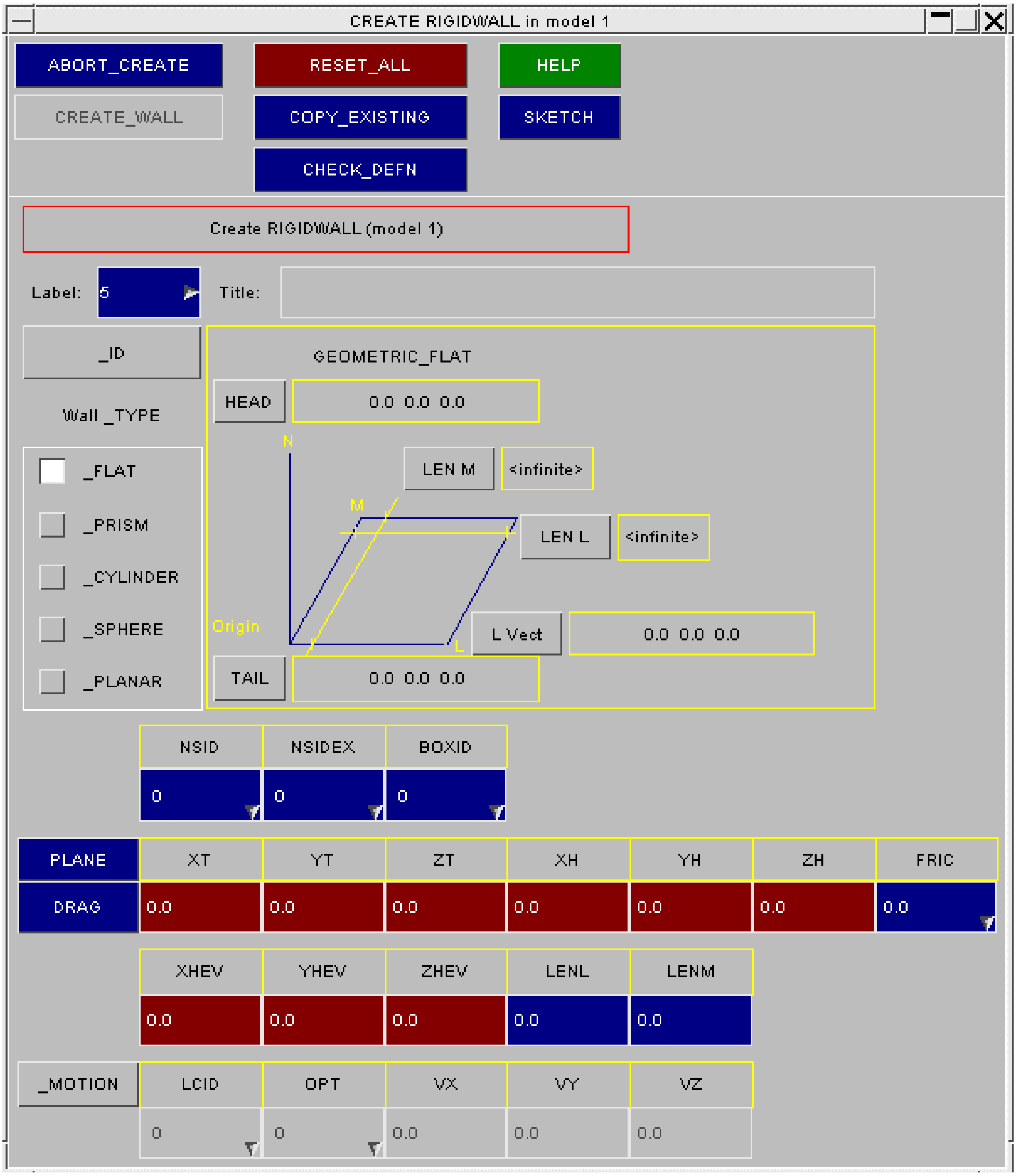
```
RIGIDWALL_GEOMETRIC_FLAT      RIGIDWALL_PLANAR
                  _PRISM
                  _CYLINDER
                  _SPHERE
```

This figure shows the top-level RIGIDWALL menu.
All rigidwall sub-types may be edited both explicitly (via Create/Edit panels) and via the generic Keyword editor.

Options have their standard meanings as defined in section 5.0.1.



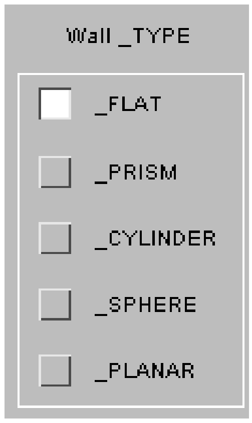
Create and Edit functionality



This figure shows the standard wall create/edit panel.

Its detailed layout changes with wall type: this example shows _**GEOMETRIC_FLAT**, although _PLANAR is the most commonly used option.

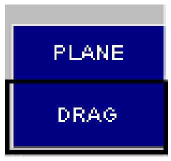
Selecting a different wall subtype



The detailed layout of the panel above changes as the different wall sub-types are selected.

In particular note that the ***RIGIDWALL_GEOMETRIC** types may only have the optional suffix **_MOTION**; whereas ***RIGIDWALL_PLANAR** may have a wider range of suffices. The LS-DYNA manual pages on the subject describe the various combinations of type and suffices available.

DRAG: Using the mouse to drag a wall into position.



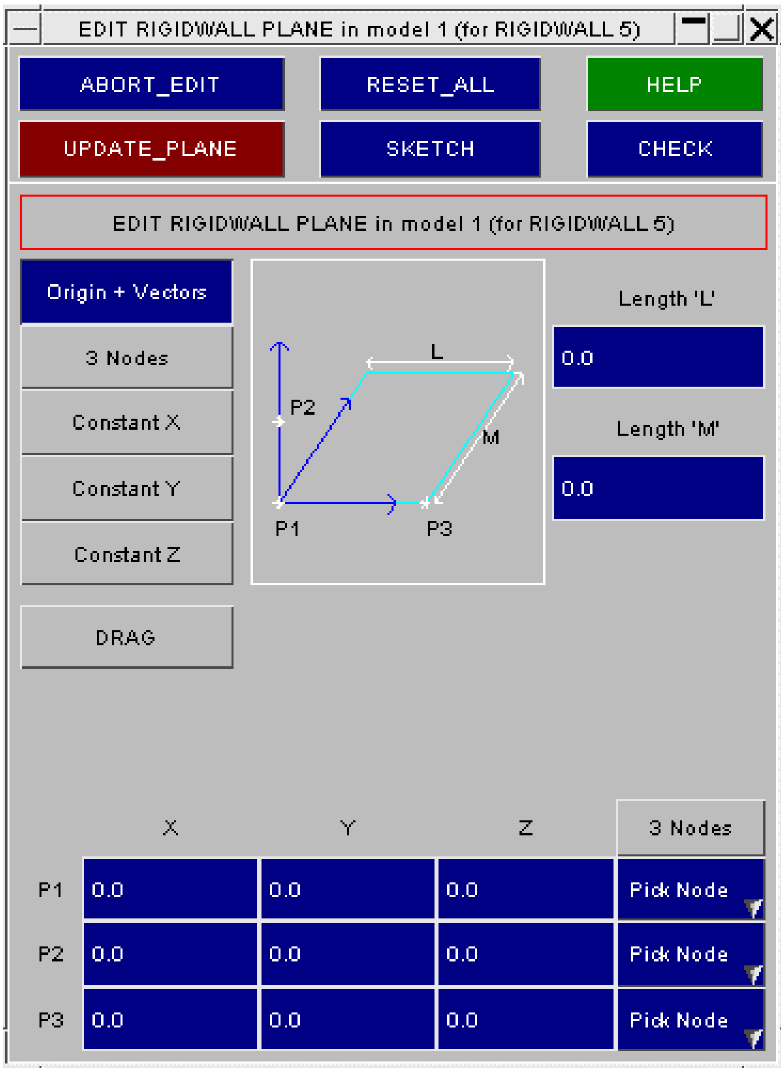
All rigidwall types can be dragged into position on the screen using the mouse. The mouse button determines the global axis along which it moves:

- X : Left mouse button
- Y : Middle
- Z : Right

END_DRAG terminates the dragging operation.

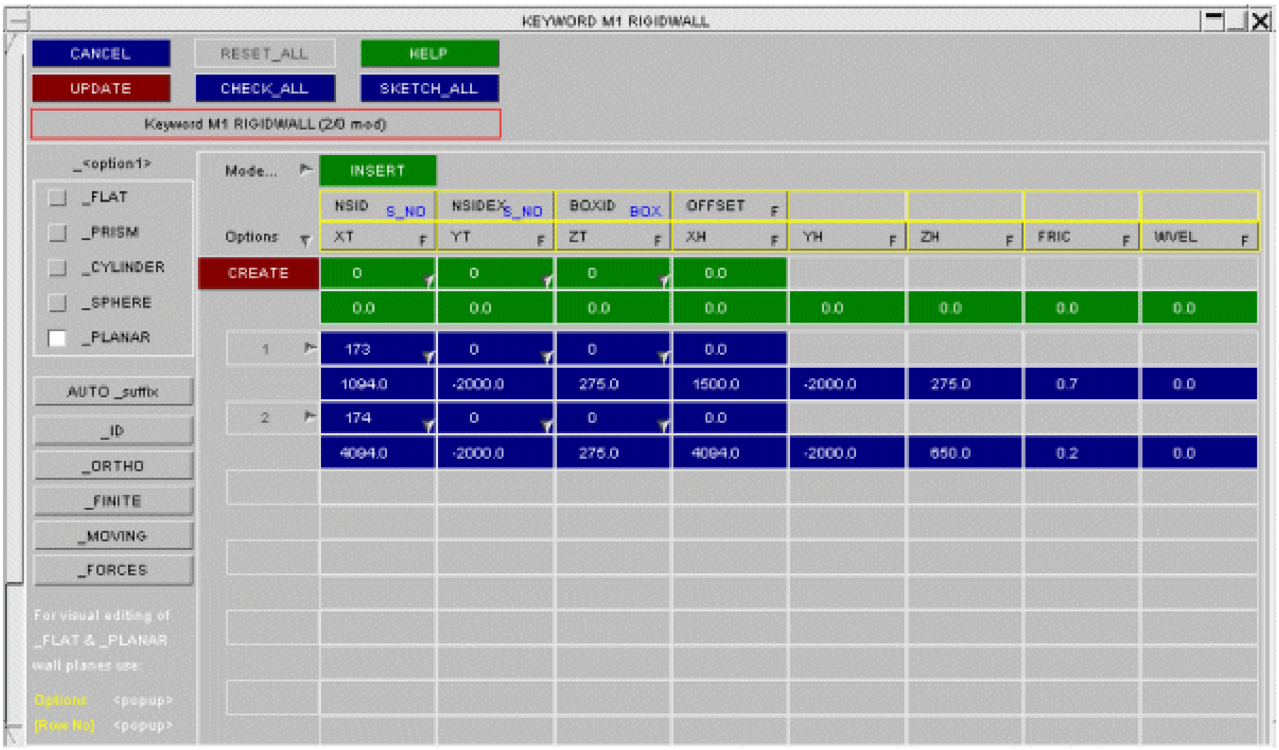
PLANE: For _GEOMETRIC_FLAT and _PLANAR wall types only

For walls defined by a flat plane the standard "plane" editor may be used.
This allows graphical definition of the plane geometry via a range of methods.



Rigidwall Keyword editing panel

All rigidwall sub-types can also be processed using the generic Keyword editor panel an example of which is shown below.

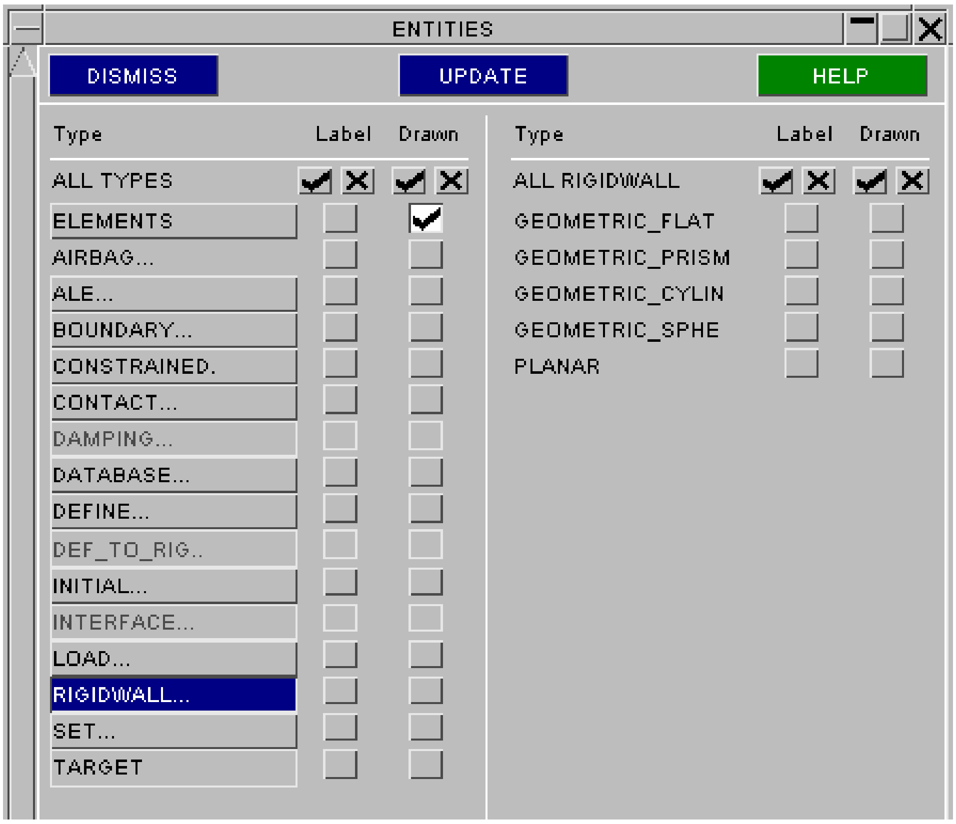


The Keyword editor "AUTO _suffix" Displaying all _PLANAR suffices simultaneously
Because there are so many suffices to the _**PLANAR** rigidwall type, which may be used in many permutations, the **AUTO** suffix allows all such types to be displayed at the same time.

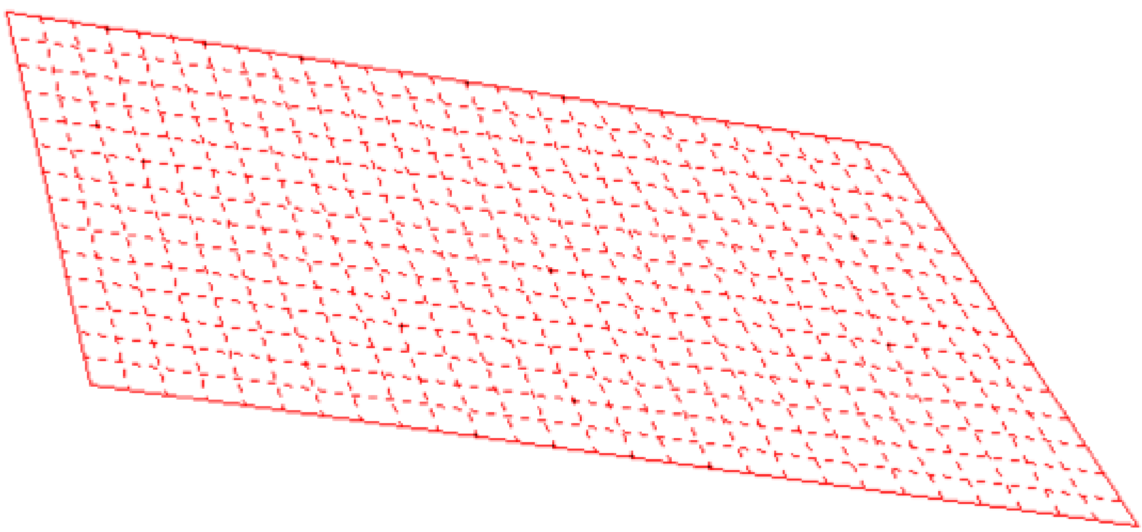
When **UPDATE** saves the editor status walls will only have a given suffix appended if the data fields for it are non-zero.

Visualising RIGIDWALLS

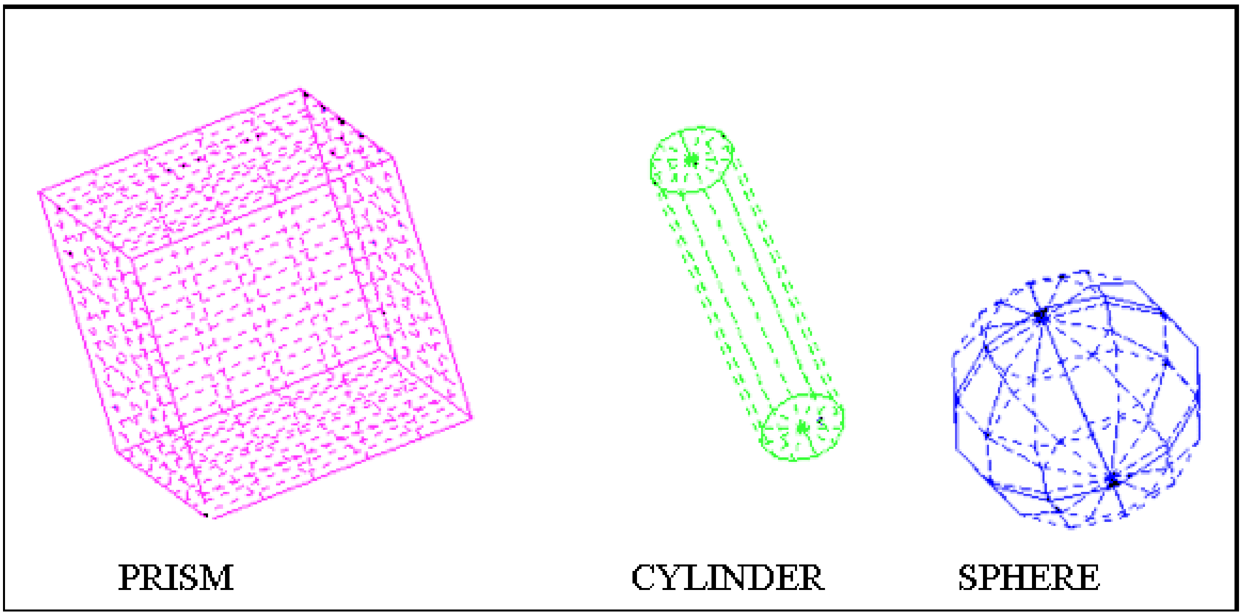
All rigidwall types may be visualised in **ENT**ity Viewing, also by the **SKETCH** functions above.



When **_FLAT** and **_PLANAR** walls have infinite side lengths then a dimension of approximately three times the diagonal of a box enclosing the model is used for graphical purposes. (Drawing an infinite object on a finite computer screen requires some compromise!). If sets and nodes are turned on as "extra" objects in **VIS_2** then the slave nodes for the walls will be drawn as well.



_GEOMETRIC_FLAT and _PLANAR



Note on scaling of finite RIGIDWALLS during **ORIENT** operations.

Prior to release 9.3RC2 PRIMER did not apply any **ORIENT** scale factors to the "finite" dimensions of rigidwalls. This was in keeping with the general policy of not applying Orient scale factors to "scalar" length dimensions since, for the most part, this is inappropriate.

However the finite dimensions of Rigidwalls are a special case, and the following scaling logic is now applied:

The local axis system of the rigidwall is calculated, that is:

- **N** is the normal axis (from tail to head)
- **L** is the first in-plane axis (defined by vector <hev>)
- **M** is the second in-plane axis, determined from the cross-product **N x L**

Any non-zero (ie non-infinite) "length" dimension is projected along the appropriate axis from the wall origin (tail coordinate) and the resulting vector is scaled by the [**Sx,Sy,Sz**] scale factors specified in the **ORIENT** operation. The length of the resulting vector is calculated and, corrected for sign if necessary, this becomes the new finite length.

The details of which dimension is projected where are as follows:

Wall type	**Dimension**	**Projected onto**	**Notes**
GEOMETRIC_FLAT PLANAR	LENL LENM	L axis M axis	
GEOMETRIC_PRISM	LENL LENM LENP	L axis M axis N axis	
GEOMETIC_CYLINDER	RADCYL LENCYL	L axis N axis	Asymmetric scale factors (Sx != Sy != Sz) will influence wall radius according to the wall's orientation, since radius operates in both L and M axes, but only L is used here.
GEOMETRIC_SPHERE	RADSPH	N axis	As for _CYLINDER case above

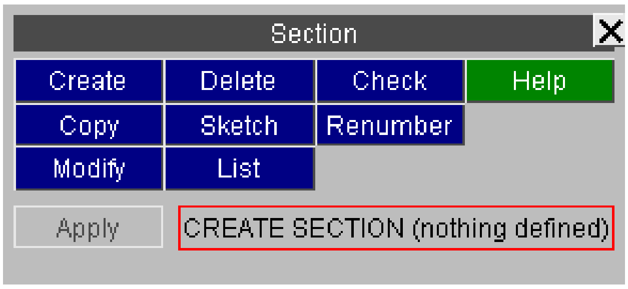
SECTION: Defining Element Sections.

Top level menu
- **Create**
- **Copy**
- **Edit**
- **Delete**
- **Visualisation**

The ***SECTION** keyword in LS-DYNA are used to define the section properties of elements. Sections are referred to from ***PART** cards. For the possible options see the type pop-up menu in Making a new section definition below.

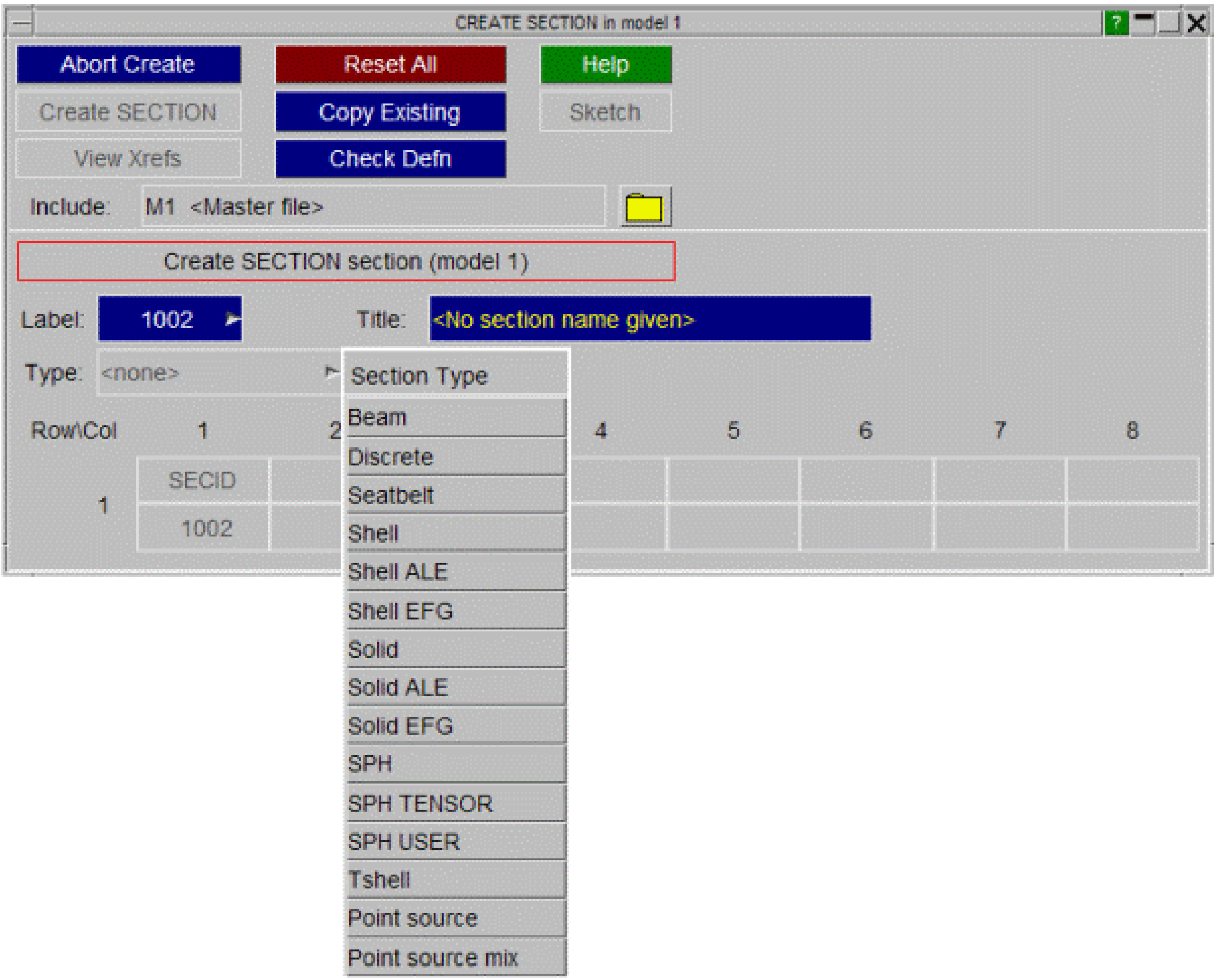
Sections of all types share a common numbering sequence (thus you cannot have ***SECTION_SHELL #1** *and* ***SECTION_SOLID #1**).

This figure shows the main section create/edit panel.
All functions have their standard meanings as described in section 5.0.1.

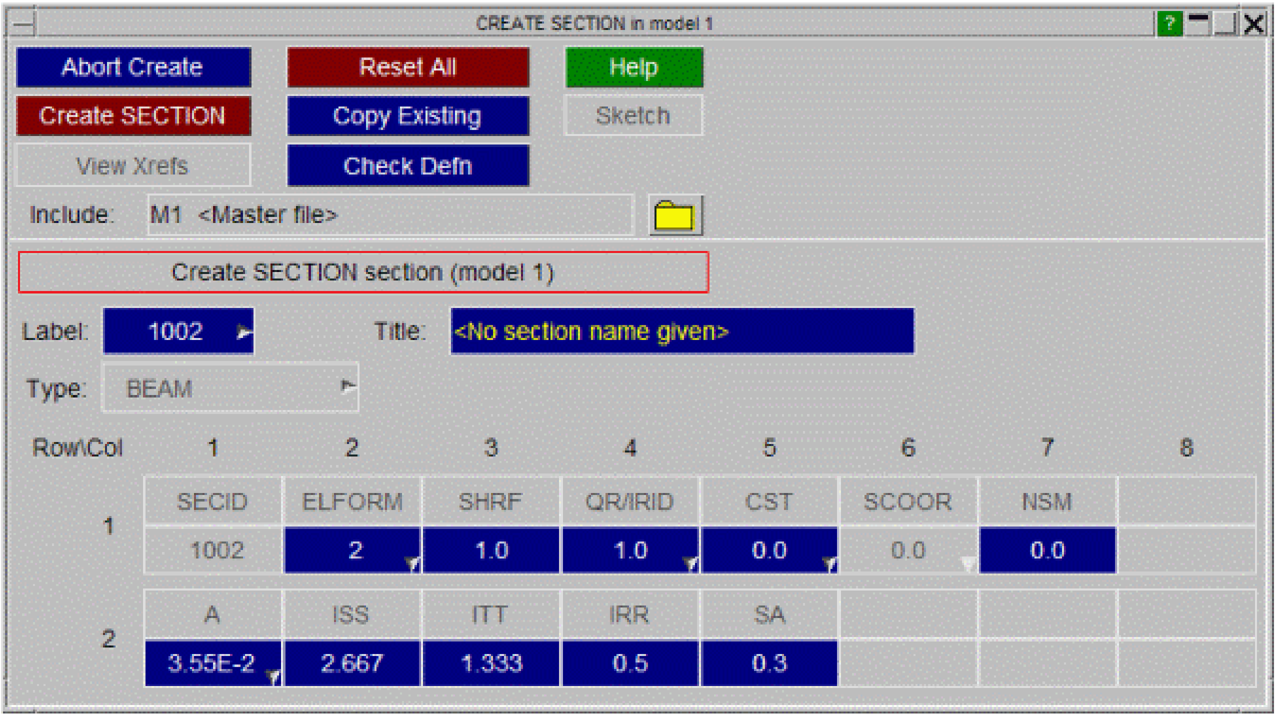


CREATE: Making a new section definition
Initially a new section has no **_type** defined, and it is necessary to define one.

Use the Type: popup menu, as shown in this figure to define an element type.



Once the section type has been defined, the relevant keyword cards appear on the editing panel, organised as shown in the LS-DYNA manual.
In this example the user has selected type **_BEAM**, and filled in the basic data for a section.



COPY Copy existing section(s) to make a new section(s).

When sections are copied the default is only to copy the section definitions themselves.

When **RECURSIVE COPY** is used all parts, elements, etc using the sections are also copied.

MODIFY Modify the attributes of an existing section.

Existing sections may be edited using **MODIFY**, which maps the same panel as above, except that data fields are already populated.

Where the section is referenced by a Part which contains elements, the topological type of the section cannot be altered. For example if a section is associated with a part containing shell elements, PRIMER will not allow the section type to be changed to **_SOLID**.

Any modifications made to the section definition will not be made permanent until the **UPDATE_SECTION** button is pressed. At this point a the local copy which has been updated is used to overwrite the version in the model.

DELETE Delete existing section definitions.

The **DELETE** function deletes the selected sections. However you cannot delete a section that is in use by a **PART** unless you remove it from the part definition, or delete that too. To help with this the following two switches may be used:

- **DELETE_RECURSIVE** Will select for deletion the parts, associated elements, and so on that reference this section.
- **REMOVE_FROM_SETS** Is often also needed if parts are to be deleted, as their elements, the connected nodes, and often the parts themselves may be included in sets.

A good way of getting rid of surplus (unused) sections is to turn these two switches off, then select all sections for deletion. Only those which are not used by anything will actually get deleted.

SKETCH Sketch elements belonging to sections on the current image.

SKETCH sketches on top of the current image the elements of the parts which reference the selected sections.

LIST List a summary of selected sections

The main attributes of the selected sections are listed to the screen.

CHECK Check selected sections for correctness

The selected sections are processed through the section checking functions, and any errors found are summarised to the screen.

Section checking is comprehensive: it detects both numerical errors (eg -ve area) and parameter errors (eg <type> indices out of range).

Individual sections may also be checked during creation and modification by the **CHECK_DEFN** command on those panels.

RENUMBER Change the labels of sections in a model

RENUMBER lets you change any or all section labels within a given model using the standard renumbering panel.

To change the label of an individual section it may be simpler just to **MODIFY** it.

END_SECTION terminates the section editing process.

Visualising *SECTION definitions

Sections are not drawn explicitly, but may be displayed by **SKETCH**ing the parts and elements that reference them.

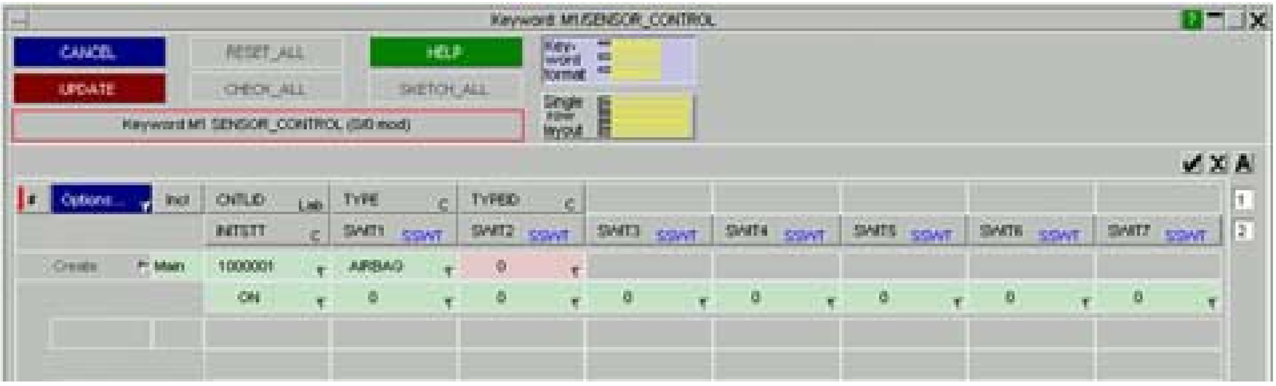
In addition the colour of part-based elements may be related to section id using the **Display > Colour > Colour all by...**. method as described in section 4.1.2

SENSOR

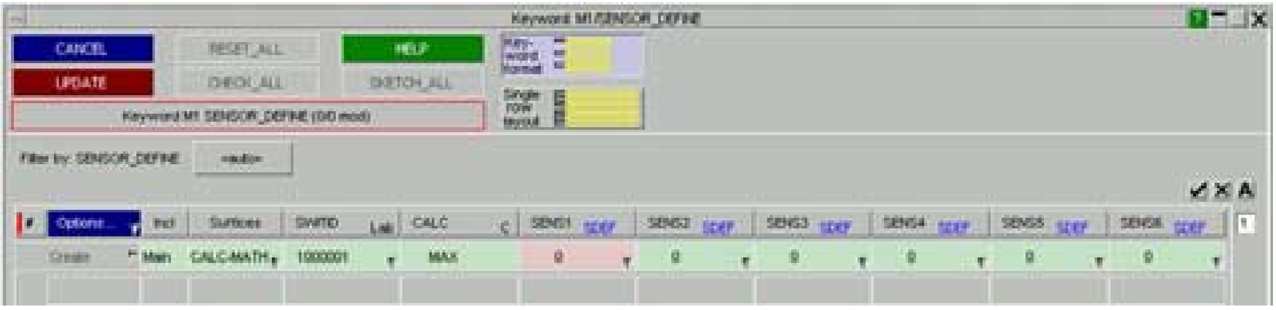
There are currently seven sub-keywords available, _CONTROL, _DEFINE_CALC-MATH, _DEFINE_ELEMENT, _DEFINE_FORCE, _DEFINE_NODE, _SWITCH and _SWITCH_CALC-LOGIC.

These can be edited through the generic Keyword Editor. There are 3 keyword editors used to create and edit these cards. One for the _CONTROL option, on for the _DEFINE options, and a third for the _SWITCH options.

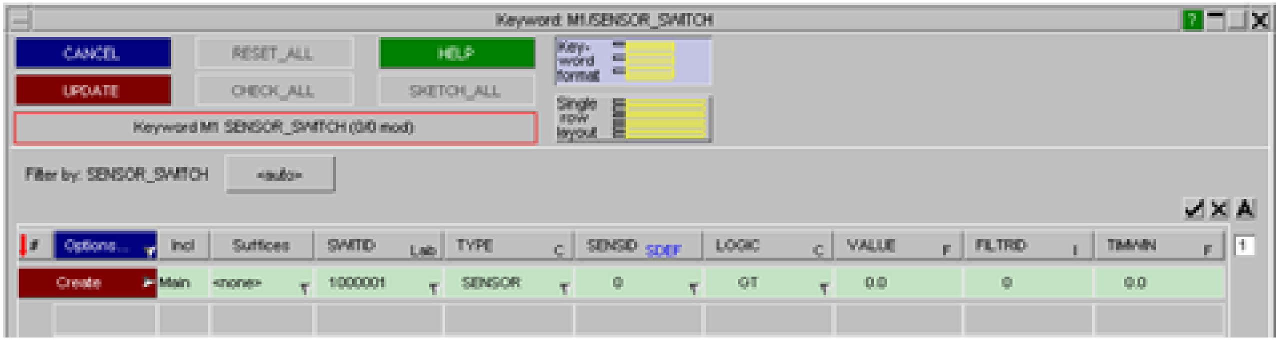
*SENSOR_CONTROL



*SENSOR_DEFINE_...



*SENSOR_SWITCH_...



SET: Defining Sets.

Top level menu
- **Create**
- **Copy**
- **Edit**
- **Delete**
- **Visualisation**

Set type:
- _LIST
- _COLUMN
- _GENERATE
- _GENERAL
- _ADD
- _COLLECT

Set defaults

Segment sets

Locking set contents

The ***SET** keyword in LS-DYNA is used to define groups of items that can be used in many different contexts. Valid set types are:

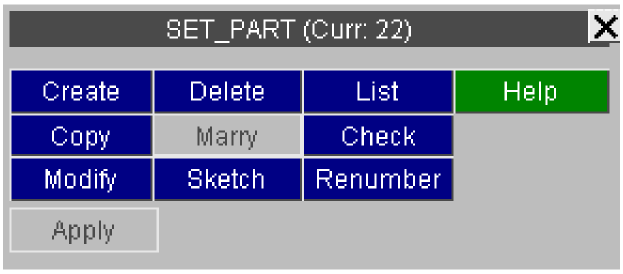
```
SET_BEAM
   _DISCRETE
   _MULTI_MATERIAL_GROUP
   _NODE
   _PART
   _SEGMENT
   _SHELL
   _SOLID
   _TSHELL
```

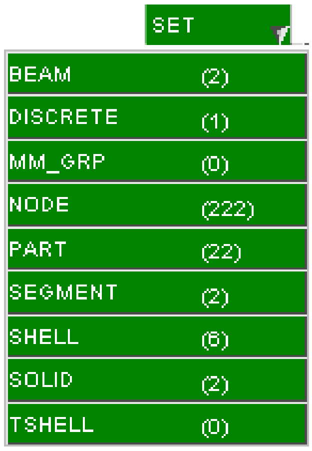
Each set types has its own numbering sequence, thus you can safely have ***SET_SHELL #1**, ***SET_SOLID #1,** etc.

With the exception of **SEGMENTS** sets simply reference other structural items, and do not themselves constitute "structure". **SEGMENTS** are a special case which are dealt with separately below.

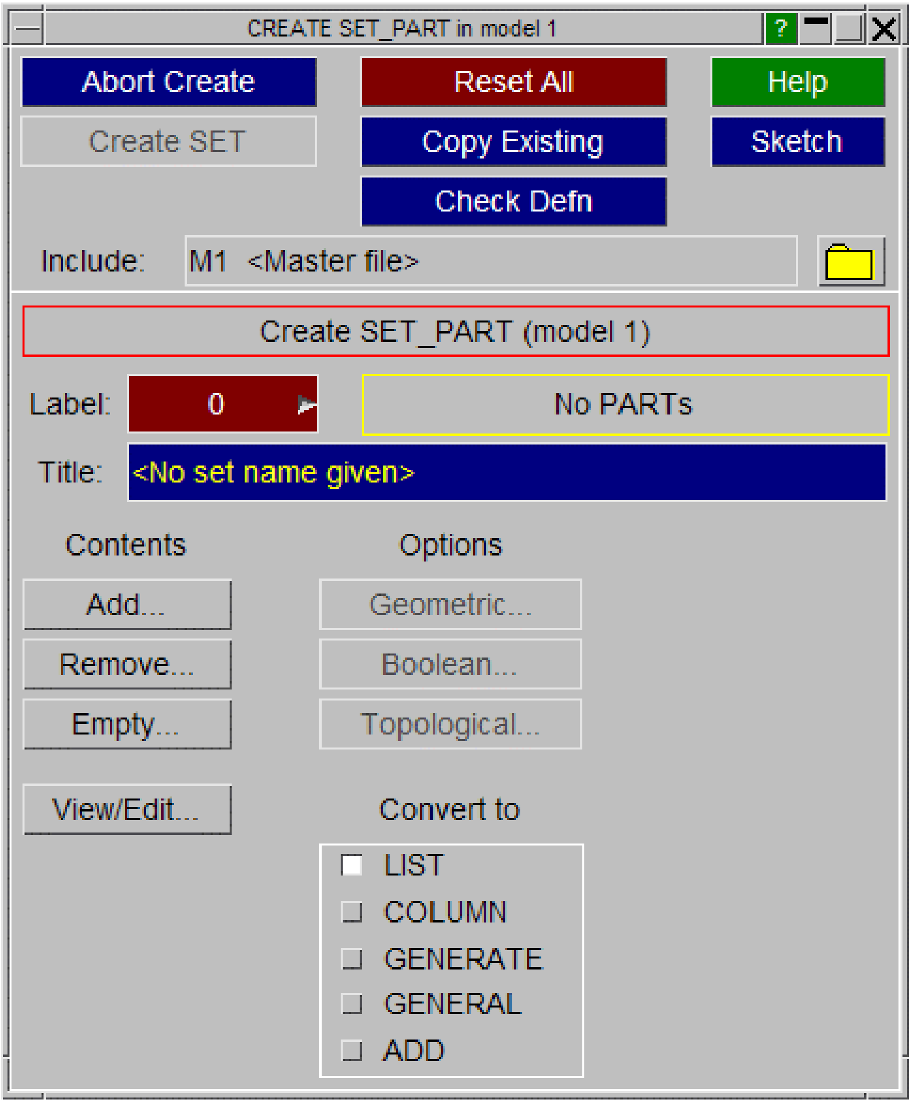
This figure shows the common top-level menu for all set types
On the right is shown the pop-up menu when **SET** is selected in the Keywords panel. The figure below shows a typical set main control panel, in this case for **SET_PART** definitions, but all are the same.

Commands have their standard meanings as described in section 5.0.1.





CREATE Creating a new set

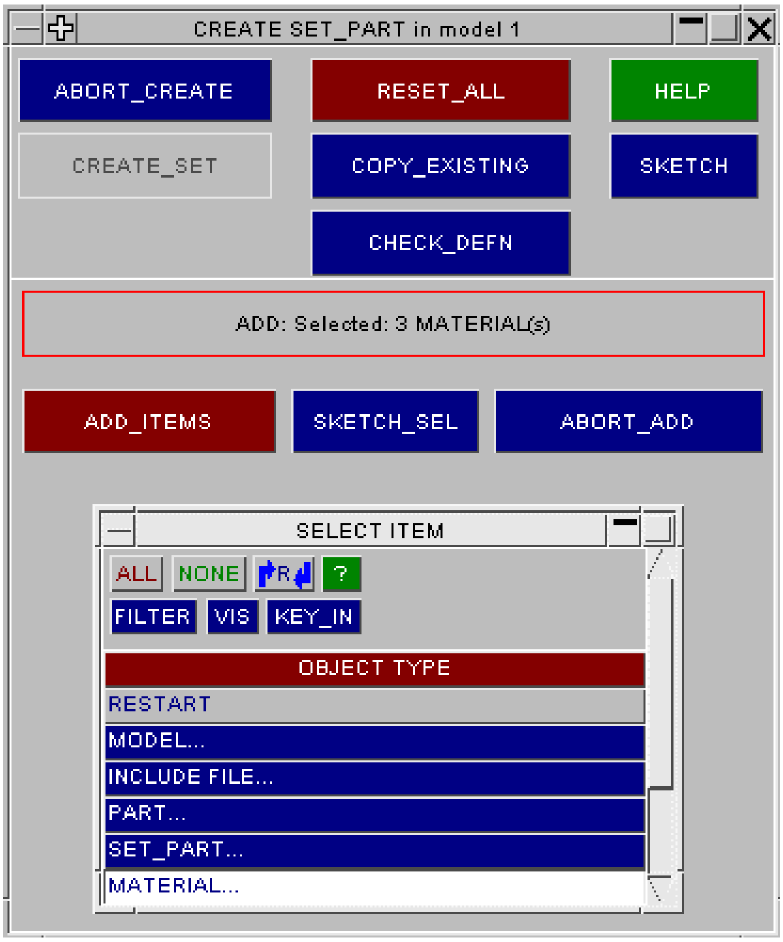


This figure shows the initial empty set creation panel. Items, here **PARTs**, but the same applies to all valid set types, can be added to or removed from the set using:

ADD...	Inserts items into the set.
REMOVE...	Removes them from the set.
EMPTY...	Completely empty the set of all its contents.

Items are added or removed using the standard selection menu, as shown in figure below.

The set can also be converted to different formats by using the **Convert to** radio buttons. GENERAL sets have a different editing panel.



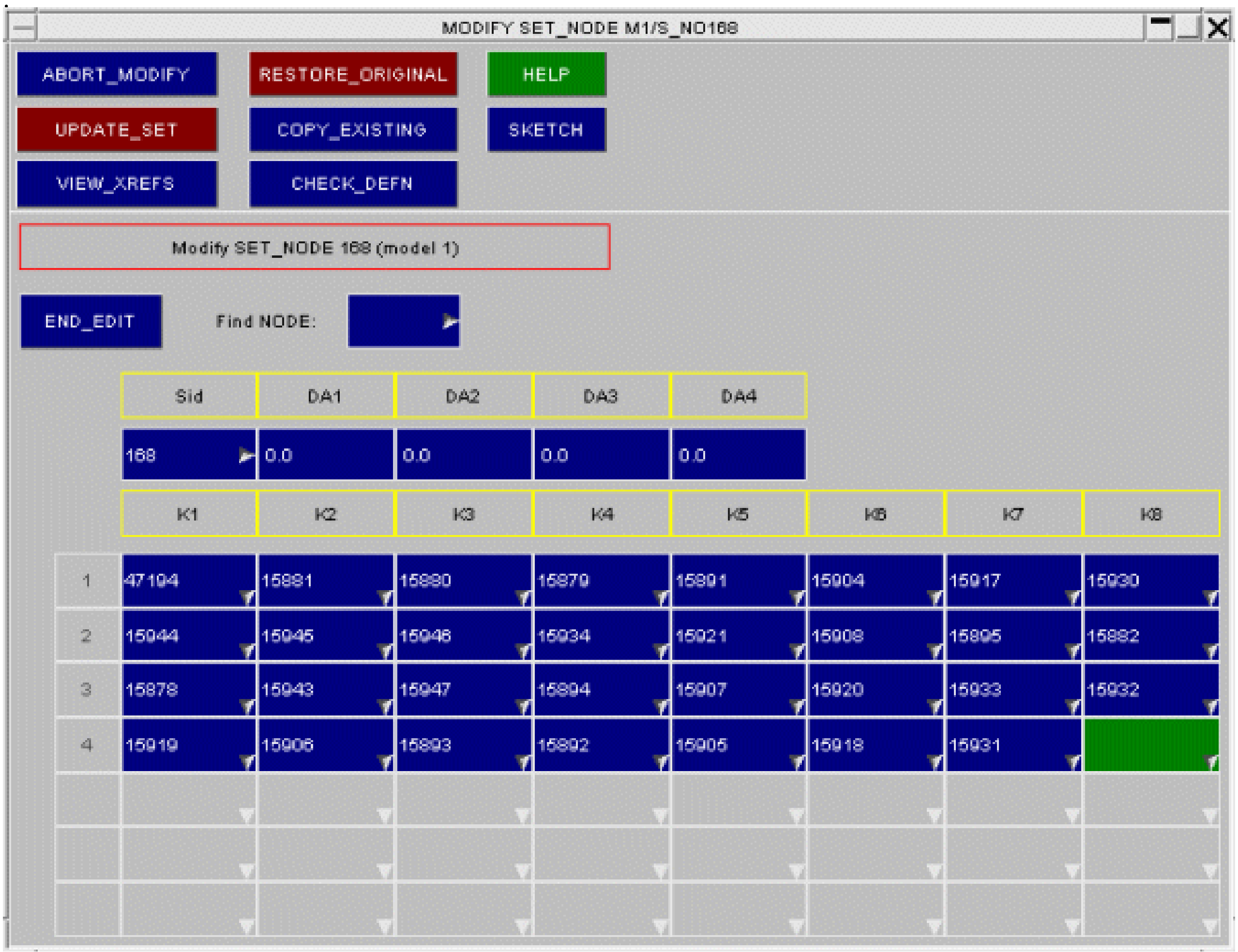
In this example the user is selecting parts by material: he has selected three materials and **ADD_ITEMS** will load the parts which reference these into the set.
Only items of the correct type, here parts, will be selected, therefore it is safe to select a super set of the objects required.

For example to load all parts in a model into a set you could just select the whole model.

It is legal to select items that are already in the set: they will not be duplicated as the addition operation performs a logical (inclusive) OR between the incoming items and the existing set contents.

Item removal operates in exactly the same way, except in reverse.

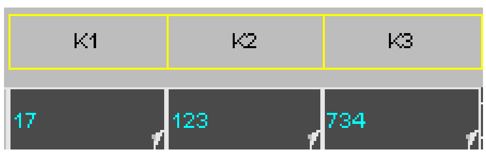
Once the set contains something you can use **VIEW/EDIT** to view the detailed contents of the set



This example shows the **VIEW/EDIT** panel for a set of nodes.

The popup options against each entry may be used to view details of that item, and different labels can be typed in to change the set's contents.

Defining "Latent" items in a set.



If a "latent" (referenced, but not yet defined) item is included in a set, its colour in the editing table changes to blue text on a dark background to warn you, as shown here.

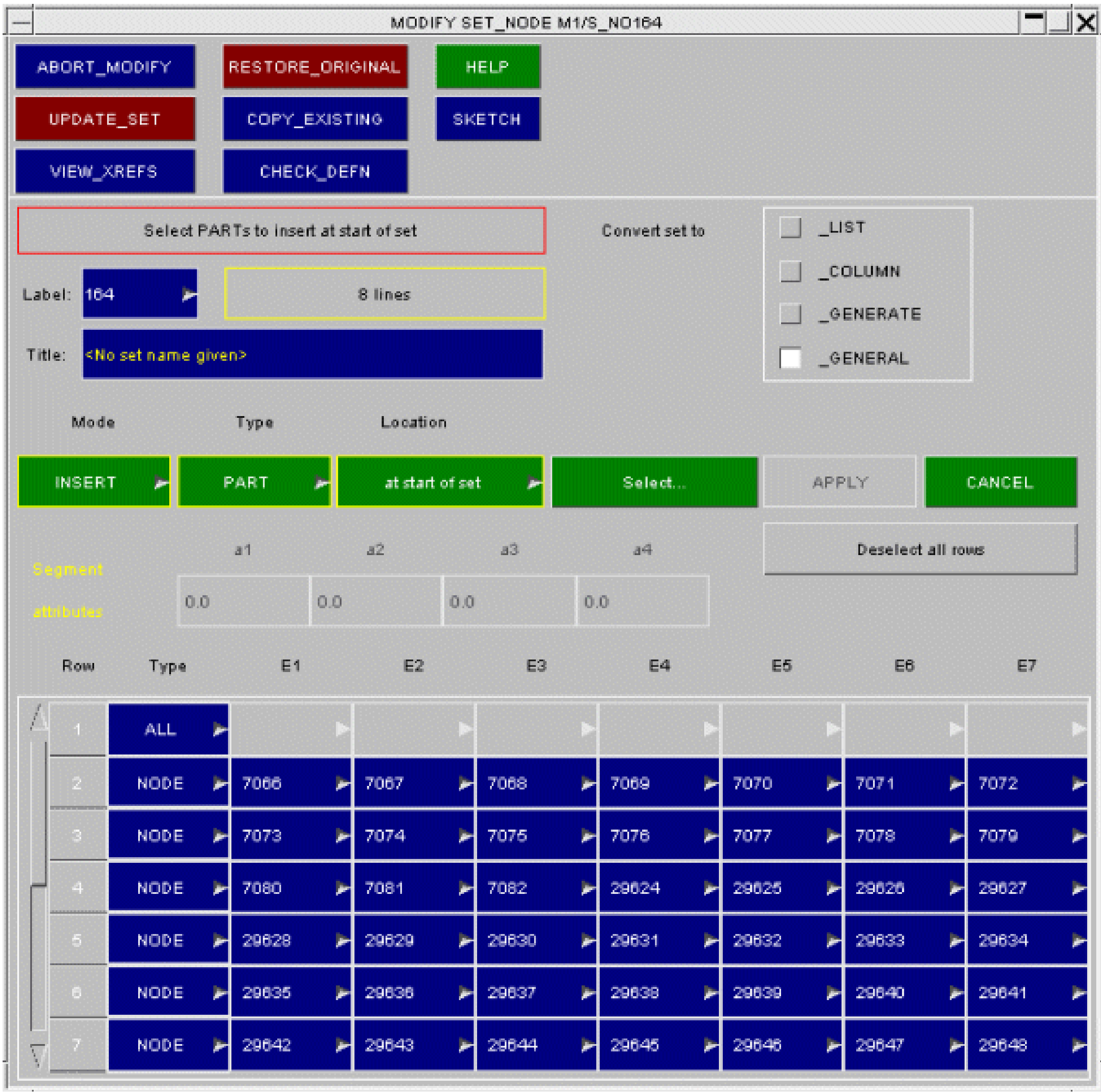
Including latent items in sets is legal, although they will show as an error when the set is checked. You must deal with this before running the analysis: by deleting them explicitly from the set, or by performing a **CLEANUP_UNUSED** operation on the model.

Defining set type and default parameters

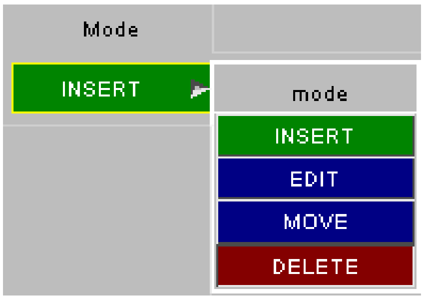
Sets may be of **_LIST** (default), **_COLUMN ,_GENERATE,** `_GENERAL` or **_ADD** types, default parameters may be defined, and individual parameters given in **_COLUMN** cases. These options are described in section SET_OPTIONS below.

Creating/editing _**GENERAL** sets

In LS-DYNA 960 _GENERAL sets have been introduced. These allow flexibility in how the set is defined. For example in a node set they can be used to add all nodes from part 10 and then remove all nodes inside box 1. As the way in which _GENERAL sets are defined is very different a separate panel is used for creating/editing them. The figure below shows a set with several rows already defined in it. Any of the existing rows in the set can be edited by using the usual popups and object menus in PRIMER or by typing in new labels for the items.



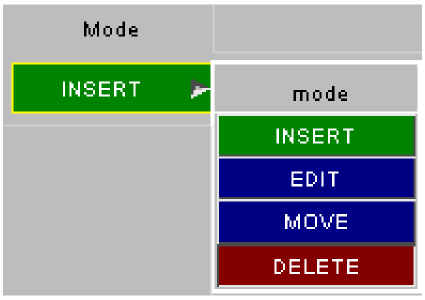
There are also 4 main functions that allow the user to insert, edit, move or delete rows in the set. The modes are selected by using the Mode popup.



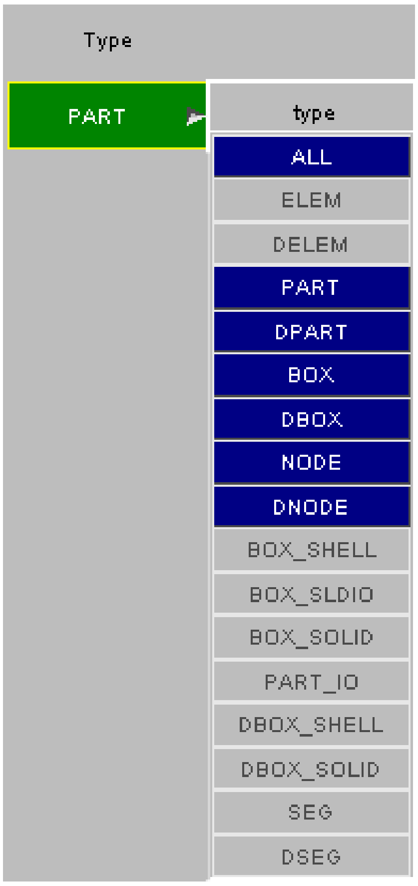
Inserting rows into a _GENERAL set

Example: inserting PARTS 4,5 and 6 to the set after row 4

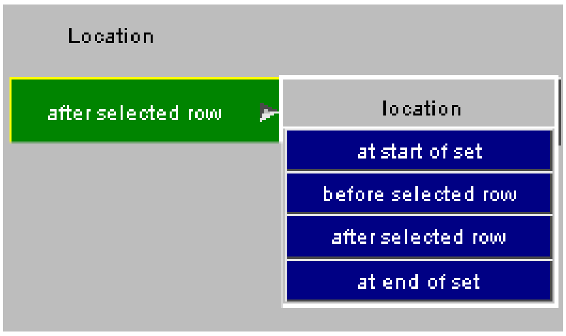
Make sure **INSERT** mode is selected by using the **Mode** popup.



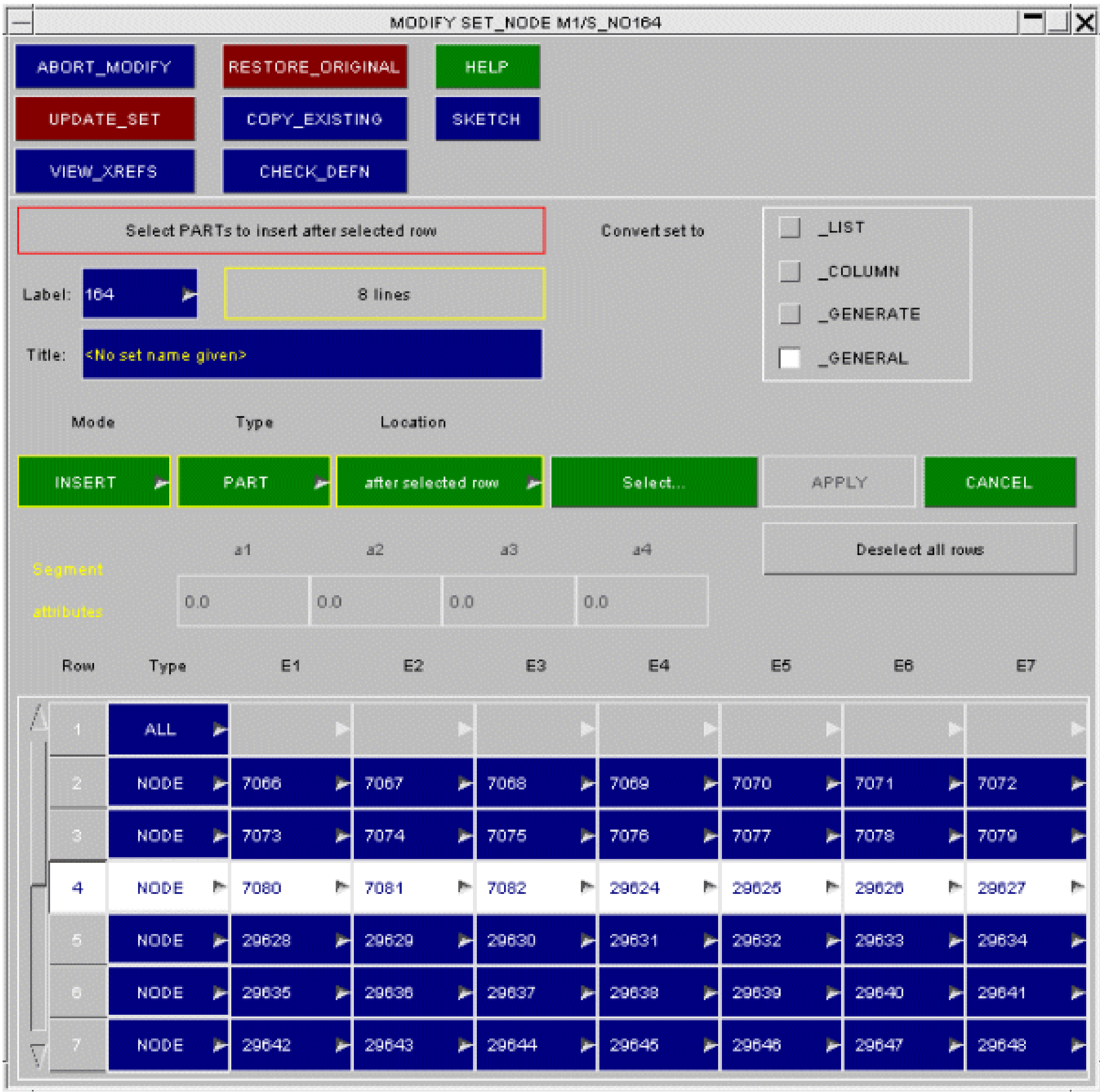
Select **PART** from the **Type** popup.



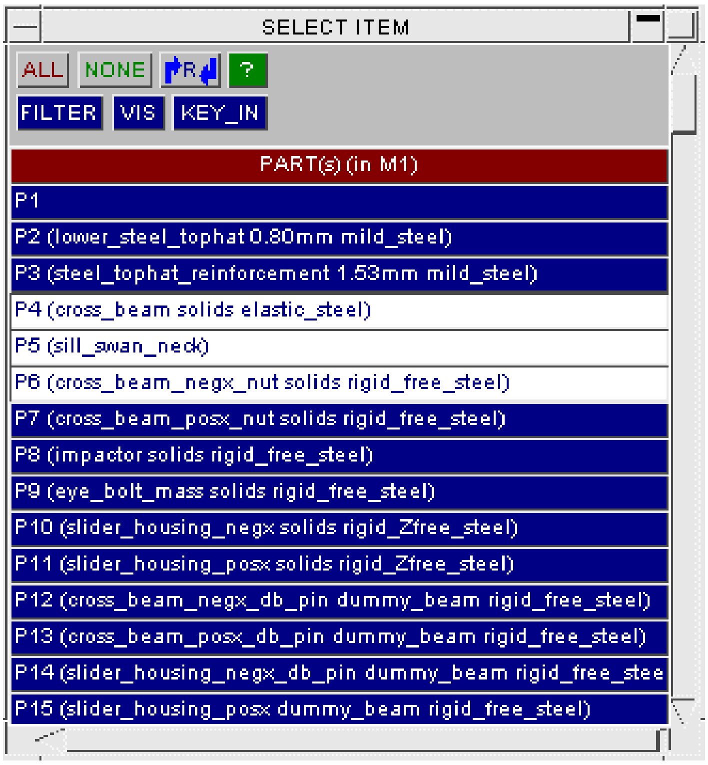
Choose **after selected row** from the **Location** popup



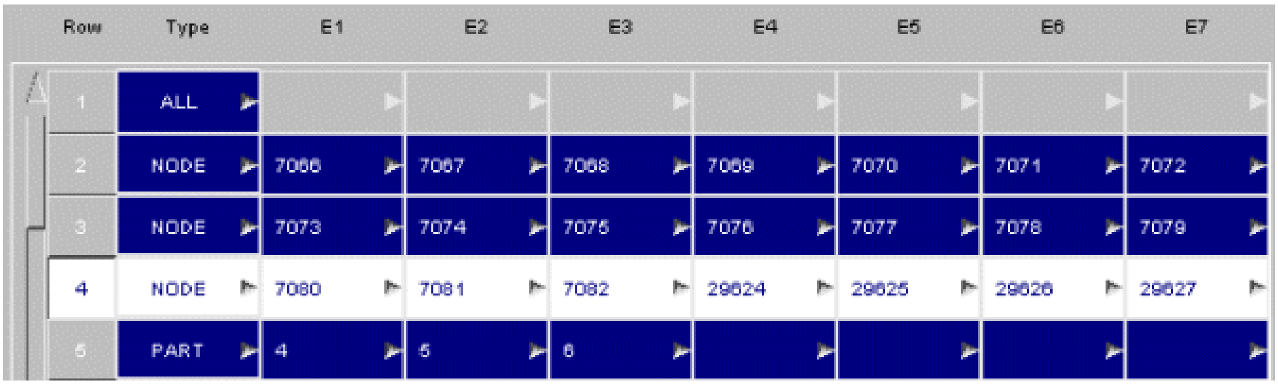
We want to insert the parts after row 4 so select row 4.



Press the **Select...** button and select the parts to add by either picking them or using the object menu.



Press **Apply** to add the parts to the set.



The parts have been added on row 5. All of the higher rows have been shifted up by one.

This method can be used to add a line (or more than one line if needed) to the set at:

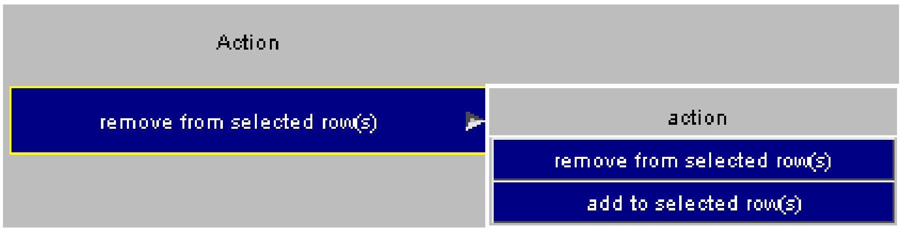
- the top (first row) of the set
- the bottom (last row) of the set
- before a selected row
- after a selected row

Editing rows in a _GENERAL set

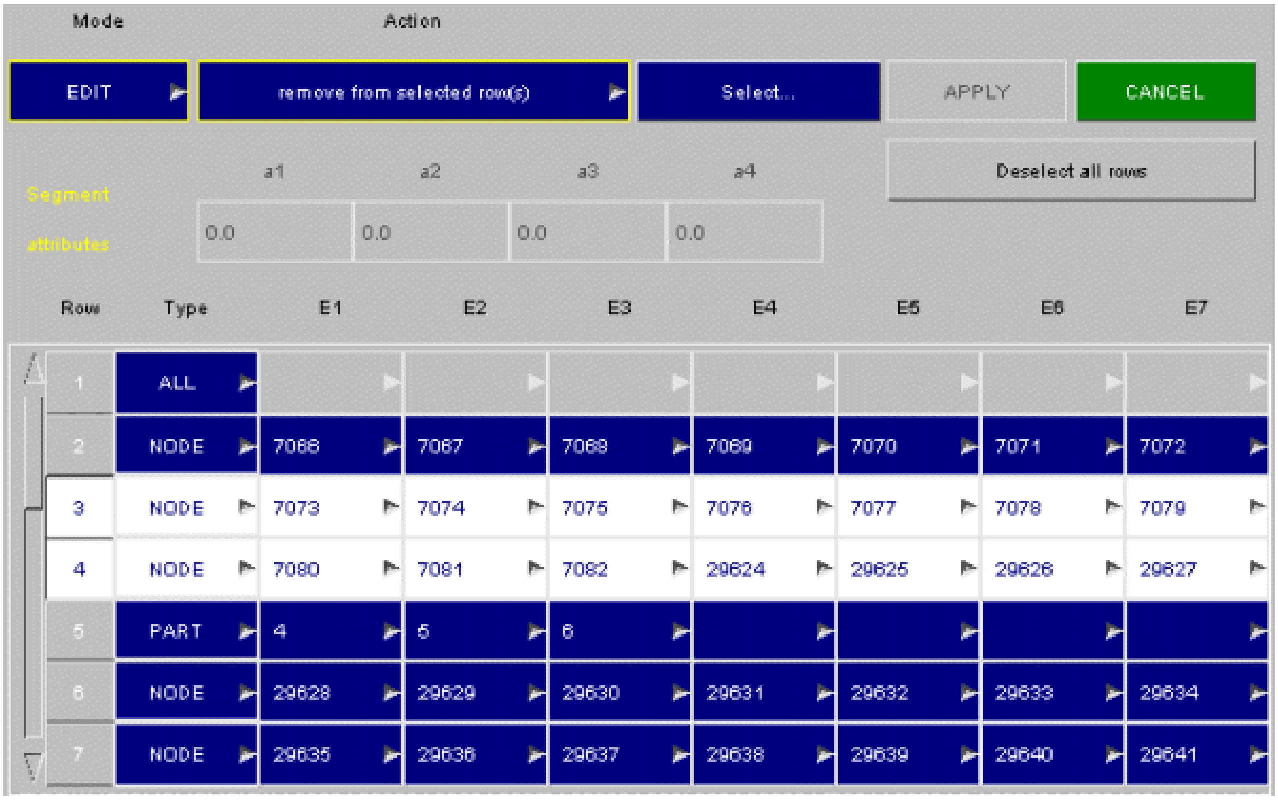
Example: removing BOXES from an existing row

Make sure **EDIT** mode is selected by using the **Mode** popup.

Select **remove from selected row(s)** from the **Action** popup.



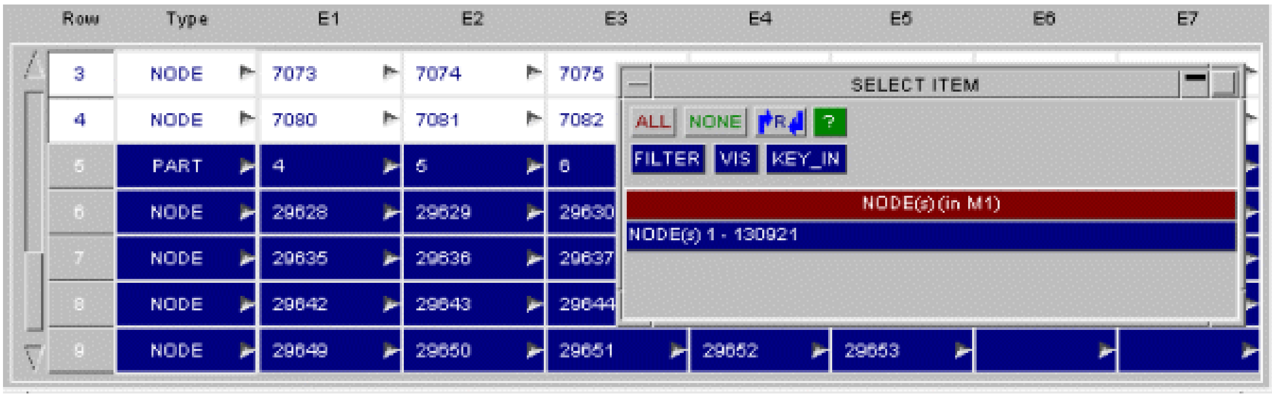
Select the rows that we want to edit (in this example; rows 3 and 4).



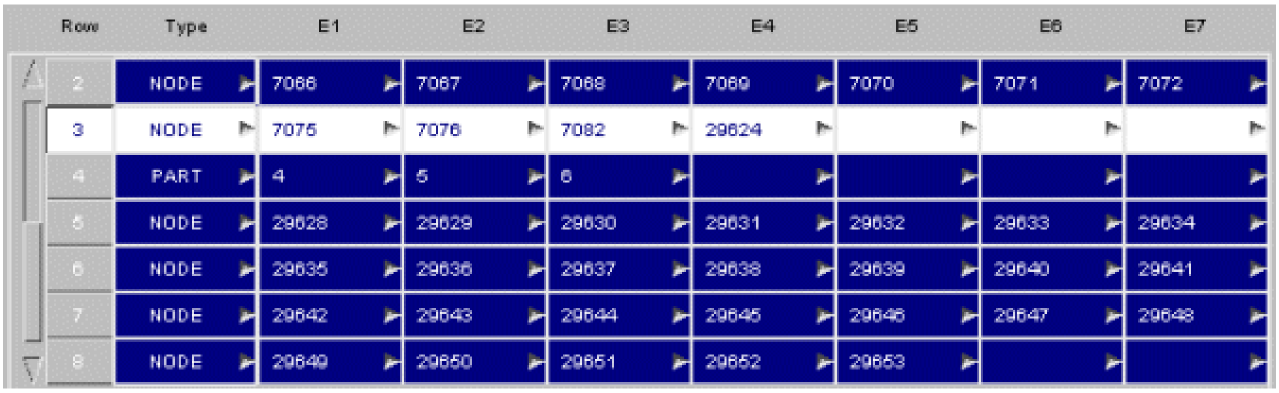
When selecting more than one row they must be:
- Of the same type. E.g. BOX, PART...
- Adjacent rows. E.g. selecting rows 3 and 4 is OK, selecting rows 3 and 7 is not.

To select a range of rows:
- Select the first row by left clicking
- Move to the last row and shift, left click to select the range

Press the **Select...** button and select the boxes to remove from the rows by either picking them or using the object menu.



Press **Apply** to remove the boxes from the selected rows in the set.



The boxes have been removed from the rows. As the entry now only requires one row (there are now only 4 boxes) the empty row is deleted and the higher rows have been shifted up by one.

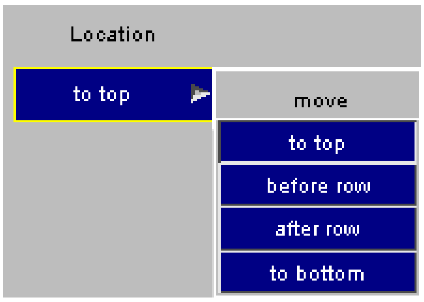
This method can also be used to add extra items to an existing line instead of removing items by choosing **add to selected row(s)** from the action popup. Extra rows will automatically be added if they are needed.

Moving rows in a _GENERAL set

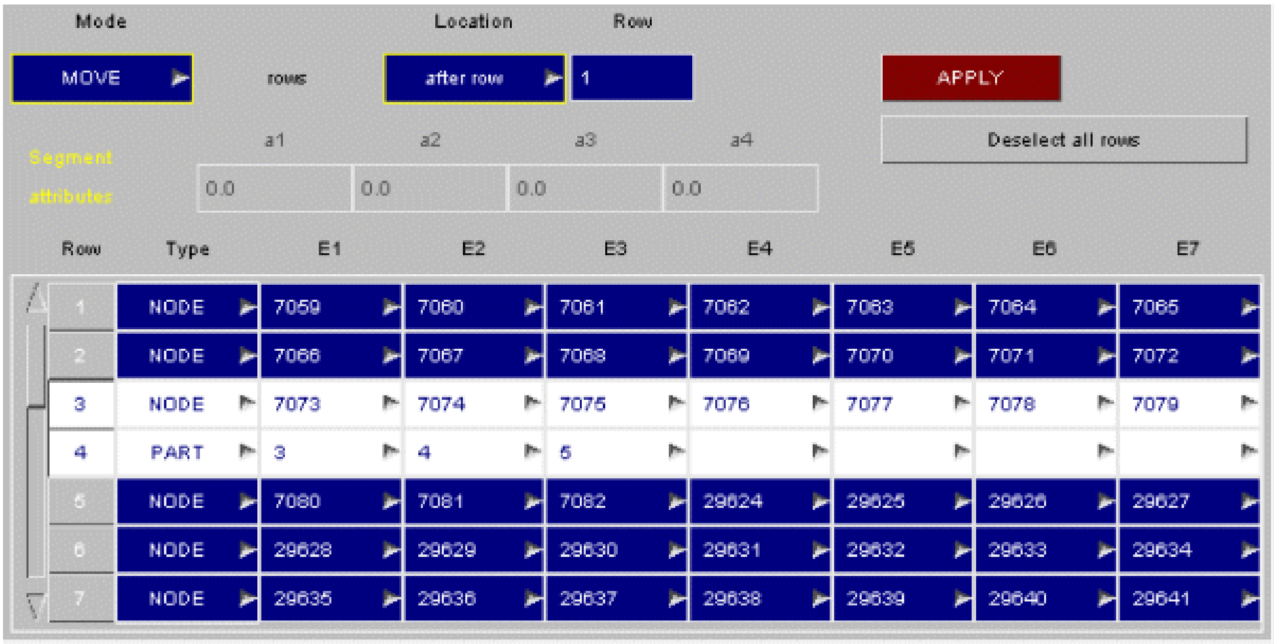
Example: Moving selected rows to be after row 1

Make sure **MOVE** mode is selected by using the **Mode** popup.

Select **after row** from the **Location** popup.



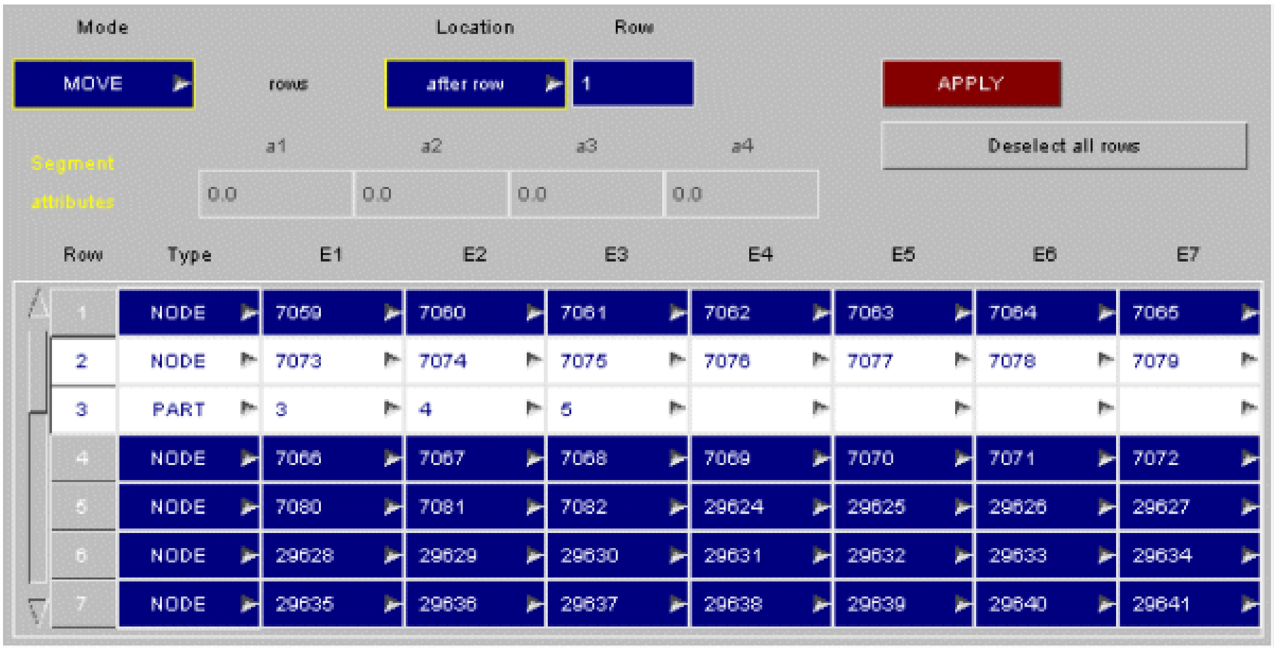
Select the rows that we want to move (in this example; rows 4 and 5) and type in row 1 for the **Location Row**.



When selecting more than one row they must be adjacent rows. E.g. selecting rows 3 and 4 is OK, selecting rows 3 and 7 is not.

To select a range of rows:
- Select the first row by left clicking
- Move to the last row and shift, left click to select the range

Press **Apply** to move the rows in the set to be after row 1.



The rows have been moved.

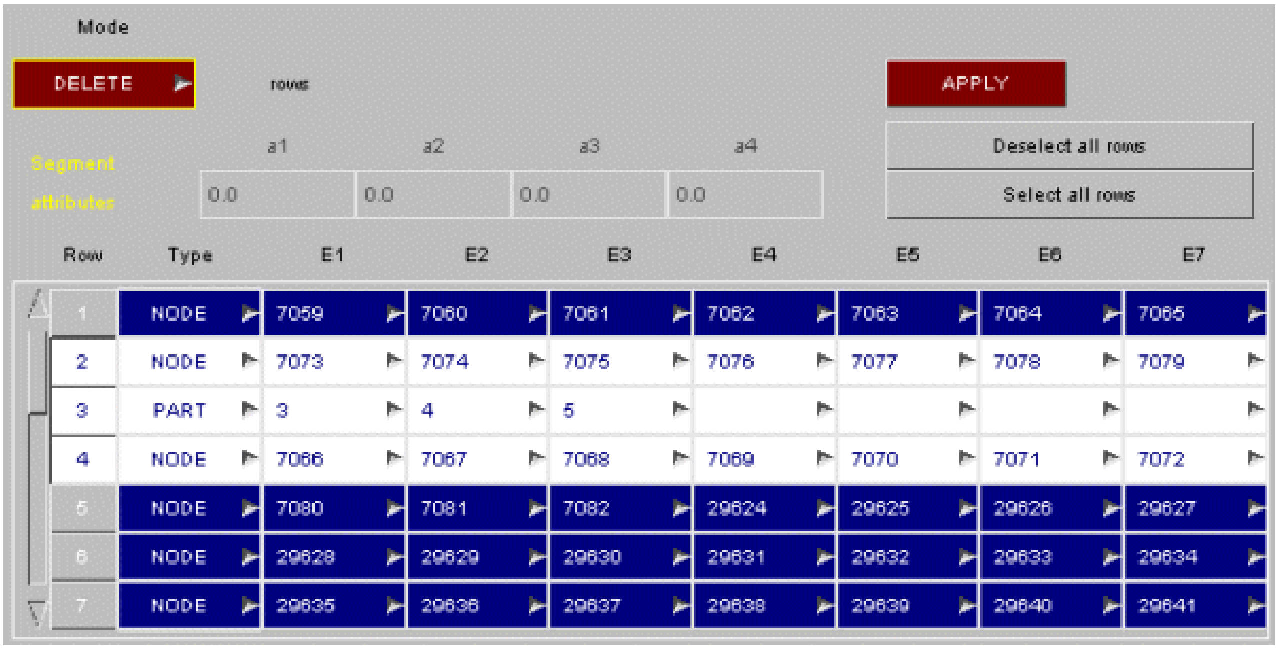
This method can be used to move a line (or more than one line if needed) to:
- the top (first row) of the set
- the bottom (last row) of the set
- before a selected row
- after a selected row

Deleting rows from a _GENERAL set

Example: Deleting selected rows

Make sure **DELETE** mode is selected by using the **Mode** popup.

Select the rows that we want to delete (in this example; rows 2, 3and 4)..

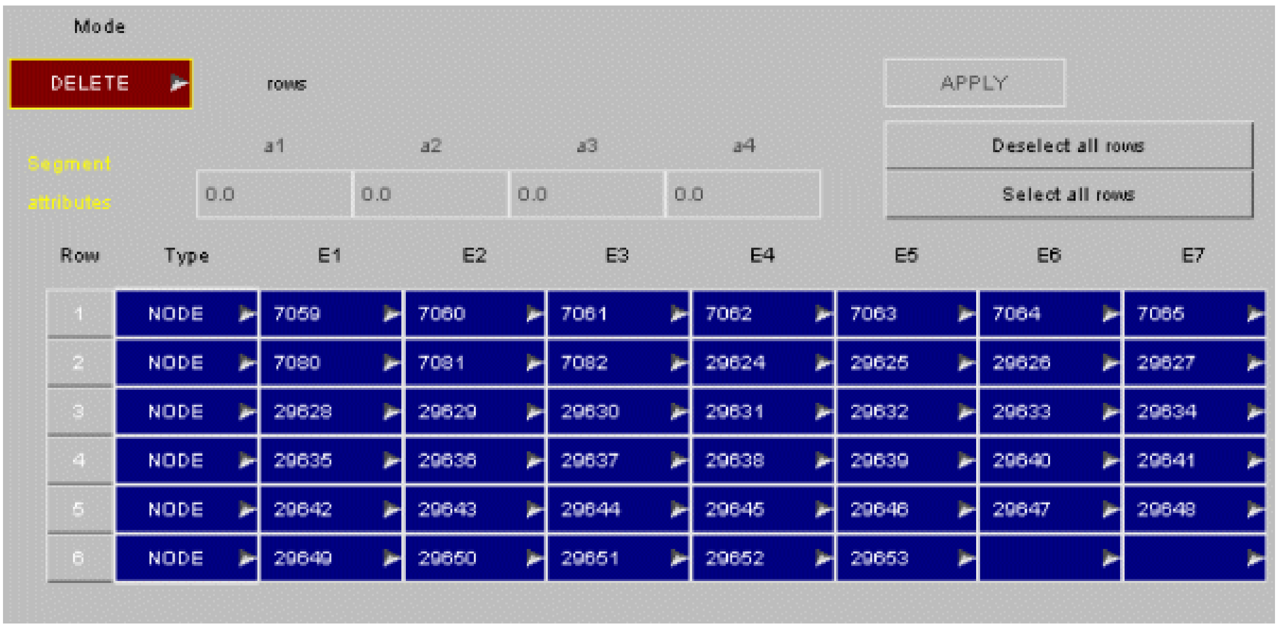


Any rows can be selected. They do not need to be adjacent.

To select a range of rows:

- Select the first row by left clicking
- Move to the last row and shift, left click to select the range

Press **Apply** to delete the rows.



The rows have been deleted.

_ADD option

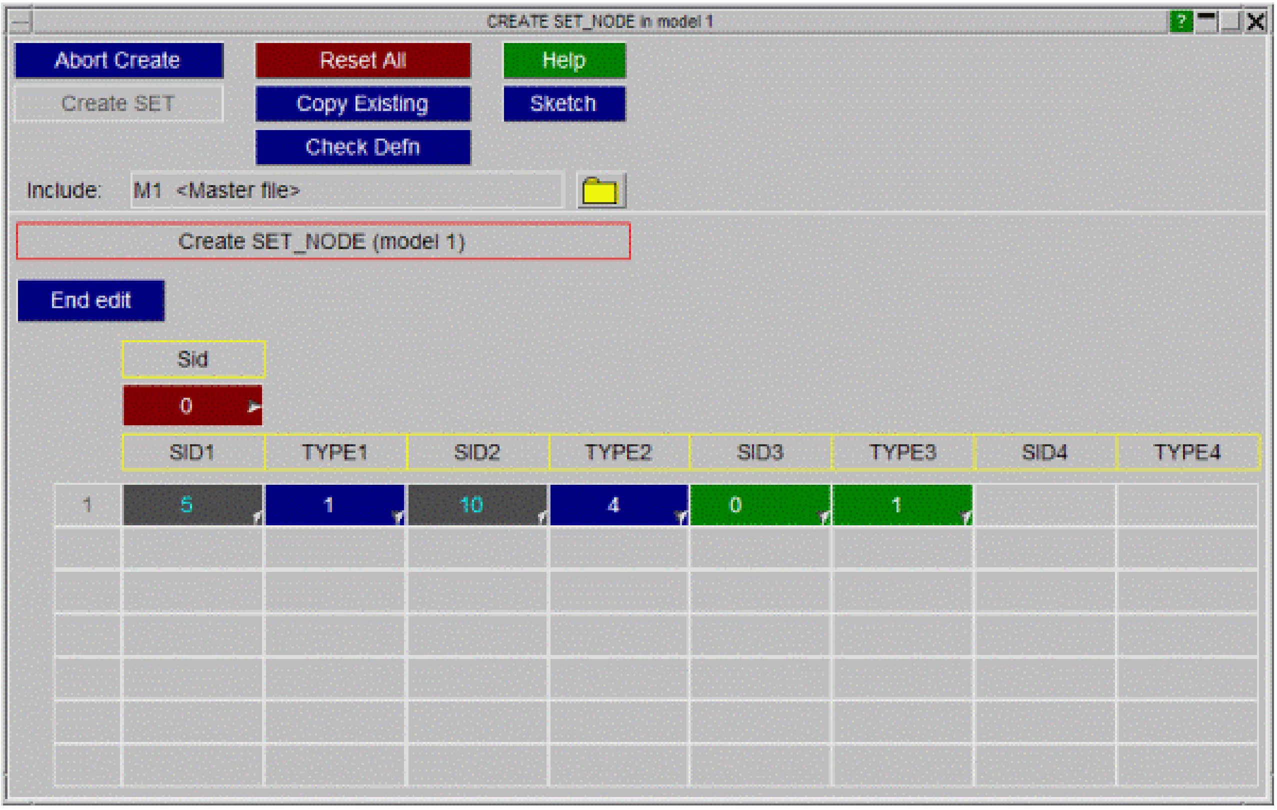
LS971 introduces the _ADD option for sets of type BEAM, DISCRETE, NODE, PART, SHELL and SOLID. Keyword *SET_PART_ADD is viable in LS970v6763 onwards. The _ADD option allows creation of a *SET that is itself composed of *SET definitions. For example a *SET_PART_ADD would list any number of other *SET_PART definitions. These sets may be created, edited, drawn, etc in exactly the same way as any other set types, with the limitation that LS-DYNA limits them implicitly to "list" syntax.

It is not clear from the LS-DYNA documentation whether or not these definitions can be nested to many levels (ie a *SET_PART_ADD contains sets that are themselves of type *SET_PART_ADD). So on the assumption that this is possible PRIMER permits them to be nested to any level, and does not treat this as an error.

PRIMER checks for recursive definitions (ie a *SET_PART_ADD contains a reference to itself, possibly several levels lower down) and flags these as errors. PRIMER handles such definitions by ignoring the 2nd definition, and anything below it.

A *SET_PART_ADD definition may easily contain multiple references to an underlying *PART. Again LS-DYNA does not state whether or not this is legal, although from past experience it probably is, so PRIMER also assumes that this is acceptable practice and does not mark this as an error. When using the set for internal purposes, for example graphics, PRIMER detects duplicate PART definitions and simply ignores the 2nd and subsequent ones.

For *SET_NODE_ADD the _ADVANCED option is available. This allows addition of other *SET types. NODE, SHELL, BEAM, SOLID, SEGMENT, DISCRETE and THICK SHELL sets can all be added. The keyword format for this option is a set ID followed by a type. The types available are NODE (type 1), SHELL (type 2), BEAM (type 3), SOLID (type 4), SEGMENT (type 5), DISCRETE (type 6)and THICK SHELL (type 7). Pressing **VIEW/EDIT** allows creation and modification of these keywords.



If a deck containing *SET_..._ADD is written out in a format pre-dating LS971R2 (LS970v6763 for *SET_PART_ADD), PRIMER will decompose the definition to a conventional *SET_... containing all underlying entities.

_COLLECT option

LS971 release 5 introduces the `_COLLECT` suffix for sets. This allows multiple definitions of a set to exist, all using the same label. These may be in the same file or (more probably) in different include files. During the LS-DYNA analysis the contents of all the set definitions are "collected" together to form a single set.

This presents problems for PRIMER because it must handle the conflicting requirements of:

- Treating the set as a single entity for graphics, checking and when referenced from other cards.
- Maintaining the individual set definitions as separate for the purposes of editing, and for ultimate output to the keyword file.

PRIMER handles the problem as follows:

1. When a **`*SET_xxx_COLLECT`** definition of label N is first encountered the following internal definitions are created

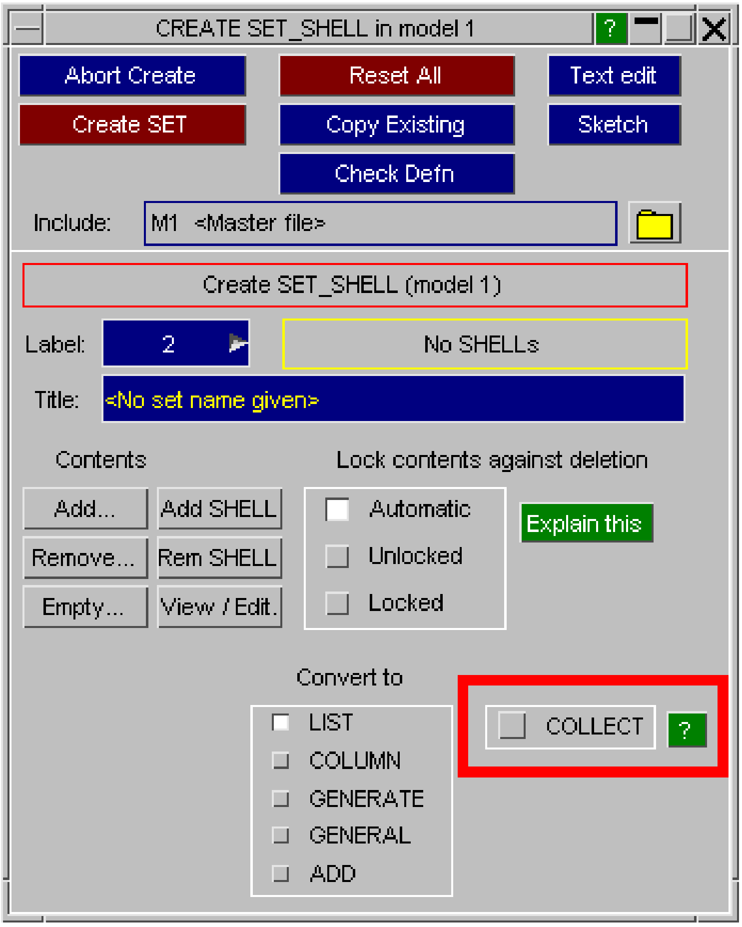
    - A "parent" set definition which has the label N, but has no contents.
    - A "child" definition that holds the contents of this portion of set N, and this is associated with the "parent".

2. For each subsequent **`*SET_xxx_COLLECT`** definition of label N that is found a new "child" set definition is created, and associated with "parent" set N

For example if a model contains three definitions of *SET_SHELL_COLLECT all using label 10 then PRIMER will store:

Parent *SET_SHELL 10		This is the parent set, and does not contain any elements
	Child definition #1	This contains the elements of the 1st set definition
	Child definition #2	The contains the elements of the 2nd set definition
	Child definition #3	The contains the elements of the 3nd set definition

Creating a *SET_xxx_COLLECT definition



For the first set in the collection:

- Create the set in the normal way, giving it a normal label
- Tick the **COLLECT** box to designate it as the start of a collection
- **CREATE** it normally.

PRIMER will warn you about what it is doing, and will create both the parent set and its first child. The parent set will automatically be given the title "All collected sets", but you can change this is you wish.

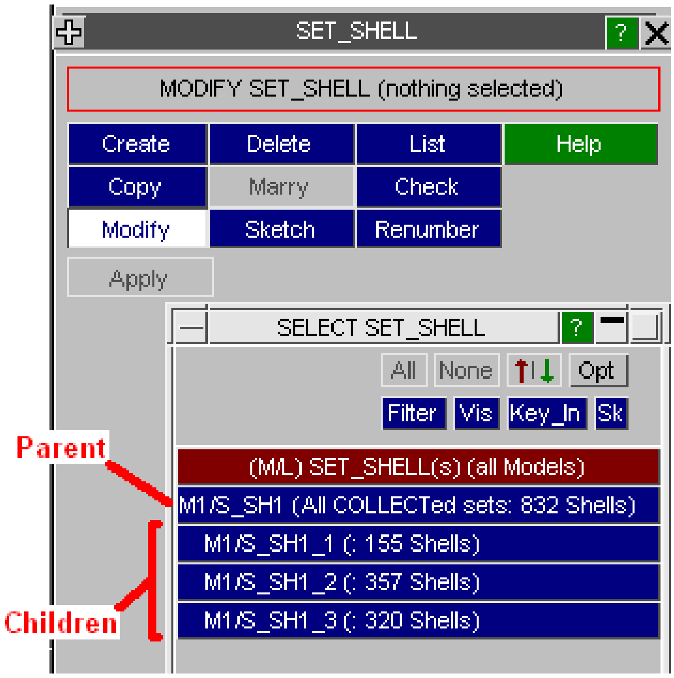
For the 2nd and subsequent sets in the collection:

- Create the set in the normal way
- Tick the **COLLECT** box as above
- Type in, or select, the label of the set used above.

PRIMER will automatically associate this set with the parent above, and this set definition will become its next child. Each child set may be given a separate title, and this is recommended as it will help to identify which is which.

There are no rules inside PRIMER about which include files the various sets "live" in: any set in the collection, including the parent, may be in any include file.

Editing a *SET_xxx_COLLECT definition



Firstly you must decide whether you want to edit the "parent" set, or one of its "children".

The menu from which you make this selection makes this parent/child relationship clear by indenting the children, and giving them label suffices _1, _2, etc as shown in this example.

Editing a ***child*** set is performed in exactly the same way as editing a normal set: you can add or remove contents, change its type, and so on, with two exceptions:

1. You can only change its label to that of a different *SET_xxx_COLLECT definition.

   PRIMER will not allow you to relabel a child set unless the new label is that of an existing (different) *SET_xxx_COLLECT definition.

2. You can remove it from the collection by unticking its **COLLECT** box.

   This will give it a new label, remove it from its parent, and turn it into a normal set. Note that is will not be referenced by anything else in the model as all existing references will be to its ex-parent.

Editing a ***parent*** *SET_xxx_COLLECT definition

The only operations you can perform on the parent set are:

- Change its label. This relabels the whole collection meaning that when the keyword deck is written out all child definitions will use this new label.

- Change its title. This title is only used within PRIMER, as the parent definition is not written out to the keyword file, so changing it will be of limited value.

- **Merge** the collection into a single, ordinary (non _COLLECT) set definition.

  This operation modifies the contents of this parent set definition to be an ordinary set, using _LIST format, that combines the entire contents of all its child sets. It then deletes all the child sets and converts this parent set into an ordinary (non _COLLECT) set.

  Once updated this merge operation is not reversible.

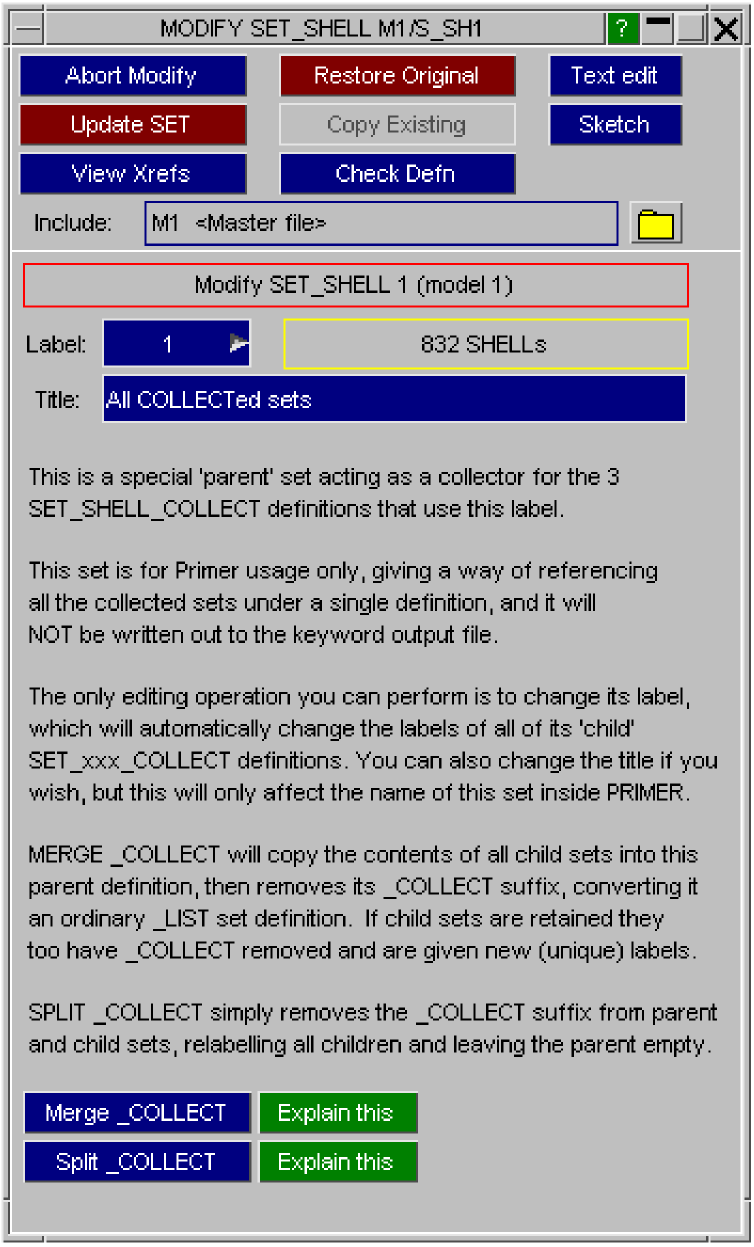
- **Split** the collection in N separate, ordinary (non _COLLECT) sets.

  Each child set is given a new (individual) label, and its _COLLECT status is turned off. This leaves it as an ordinary set, but note that it will not be referenced by anything else.

  The parent set remains, using its existing label and with its _COLLECT status also turned off. However it will have no contents. All references to the original collection will still refer to this set.

  Following a Split it will be necessary for you to sort out manually the consequences of having an empty set that is (possibly) referred to elsewhere in the model, and a series of child sets that are populated but not referenced.

  Once updated this split operation is not reversible.



Referencing and using a *SET_xxx_COLLECT definition elsewhere in PRIMER

From the point of view of the rest of PRIMER a _COLLECT set is seen as a single item, just like a normal set, and it will be referred to by the label of the parent definition. In other words no special actions need to be taken when dealing with a *SET_xxx_COLLECT definition, and it can be dealt with just like any other set. For example sketching the parent will automatically sketch all its children.

Object menus will only show the parent set, unless they are in a context in which it would make sense to offer selection of a child.

Keyword output of *SET_xxx_COLLECT definitions

During keyword output PRIMER will write out each child set definition in its appropriate include file, using the _COLLECT suffix and the label of its parent.

The parent definition used inside PRIMER is not written out and will be lost when the model is deleted or PRIMER exits. It will be recreated automatically when the keyword deck is reread, or when _COLLECT sets are created interactively.

COPY Copying sets.

The selected sets are copied. The **RECURSIVE_COPY** flag has an important influence on this:

- When **OFF** Only the set itself is copied. A new set referencing all the items in the original set is created.
- When **ON** All the items "owned" by the set are recursively marked for copying. This can select a considerable number of items: use with care.

Generally recursive copying will only be sensible for **`SET_SEGMENT`** definitions, refer to the special notes on segment sets below.

MODIFY Modifying sets

The set modification panel is identical to the **CREATE** one, except that it will already be populated when entered, and usage is exactly the same.

DELETE Deleting sets

The selected sets, and possibly their contents, are marked for deletion. What is actually deleted, and whether deletion of the set actually takes place, depends on the following switches:

DELETE_RECURSIVE Whether or not items "owned" by sets are marked for deletion.

- When **OFF** Only the set itself is so marked, its contents are not affected.
- When **ON** The contents, and anything they "own" are also marked for deletion.

REMOVE_FROM_SETS Whether flagged items can be removed from other sets.

- When **OFF** Items (marked recursively) will not be deleted if they are referred to by other sets. (But that won't stop this set being deleted.)
- When **ON** Items will be removed from any other sets in which they are referred to.

Deletion can only take place if the items referred to are not referenced elsewhere in the model. So deleting a set may fail if it is still in use somewhere, even though its contents may have been deleted leaving it empty! This will be picked up by global checking, and can be corrected with **CLEANUP_UNUSED** which gives the option of eliminating empty sets.

SKETCH Sketching sets

The selected sets will be sketched on the current graphics image. Sketching is performed by drawing the constituents of a set.

LIST Listing set summaries

A summary of the selected sets is listed to the screen.

CHECK Check sets for errors

The selected sets are processed through the standard checking routines, and the results summarised to the screen.

RENUMBER Renumber set labels

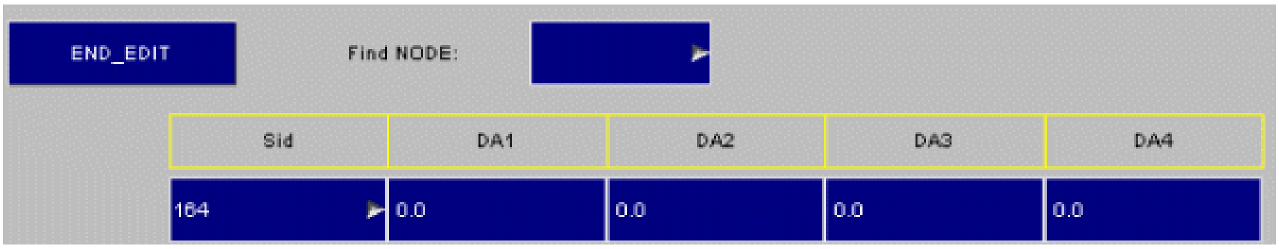
The standard item renumbering panel is mapped for the chosen model, and any or all labels can be changed. To change a single set label it may be easier just to **MODIFY** it.

SET_DEFAULTS Defining the default parameters for sets.

Set types **_NODE, _PART, _SEGMENT** and **_SHELL** can all be used in contexts where additional information may be required to complete input. Examples are:

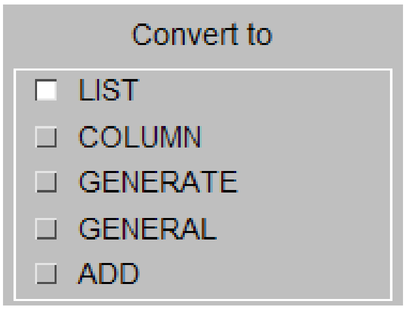
- In ***CONTACT_TIEBREAK** definitions you can define tiebreak failure parameters for a whole set, or on a per node/per element basis.
- In many other (see the LS-DYNA user manual) ***CONTACT_**... definitions it is possible to vary friction across a surface by defining individual friction parameters for segments or shells.
- In ***CONSTRAINED_TIE-BREAK** definitions individual failure parameters can be defined for each node.

In all cases default parameters may be defined for the whole set by filling in the **DA1 ... DA4** fields in the **VIEW/EDIT** panel as shown below.



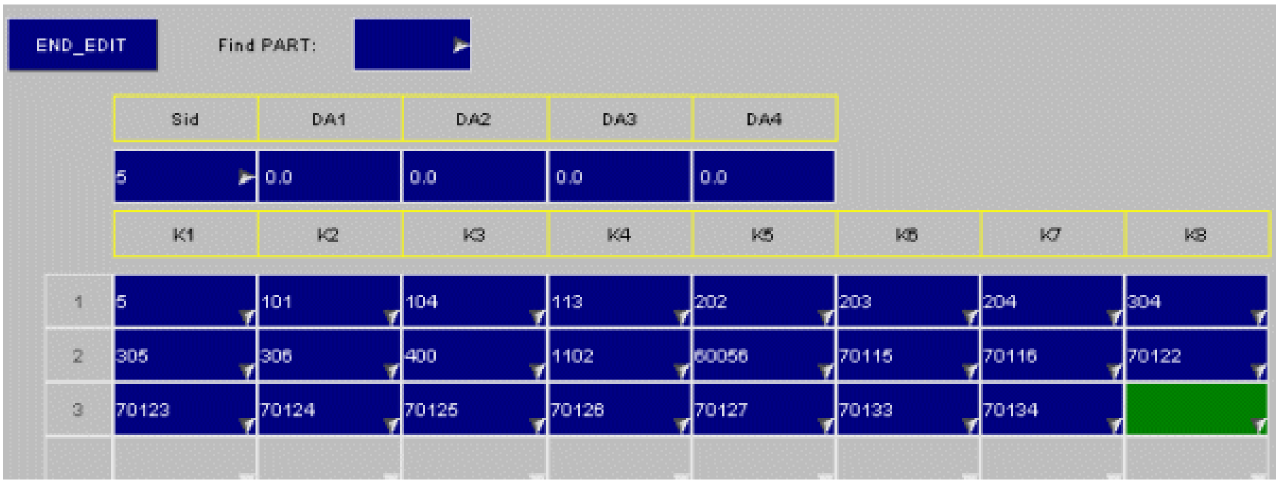
These will apply to every set entry except where individual entries have been made in **_COLUMN** mode.

SET_OPTIONS Using the _LIST, _COLUMN _GENERATE, _GENERAL and _ADD sub-keywords to change set layout.



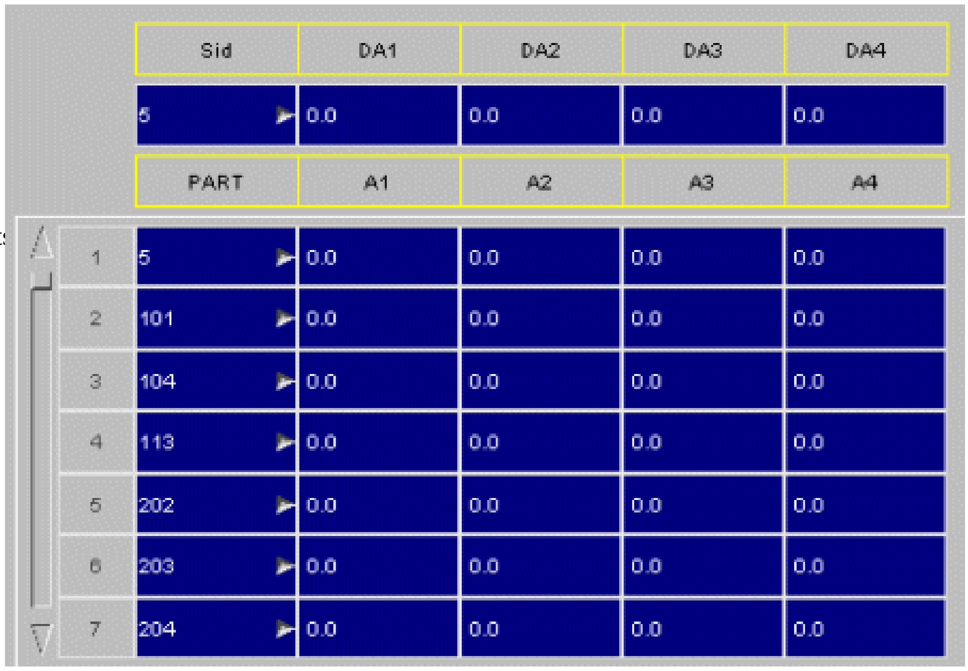
The simplest, default set layout is **SET_LIST** in which a simple list of constituent items is given 8 to a line, and in which no optional data is given for any item.
This example shows a typical **_LIST** layout for 23 parts.

Default attributes (**DA1 .. 4**) can be given, but no individual values can be defined.



Changing this definition to _**COLUMN** results in the revised layout shown here. It is now possible to give individual values for each item in the set.

When you change from _**LIST** to _**COLUMN** formats in PRIMER the code automatically fills in all the new row entries with zeros, ie the default. You can then overwrite any specific values as required.



The third method of defining sets is to use _**GENERATE**, in which item labels are given in [<start><end>] pairs.

Converting between _LIST, _COLUMN, _GENERATE, _GENERAL and _ADD set formats

PRIMER will convert between any of these ***except to*** _**GENERATE** and _ADD format.

- When a set is initially converted *to* _**COLUMN** format zeros are inserted for all optional data.
- When it is converted *from* _**COLUMN** format the optional data is still stored, but not displayed or output, and is still available if you revert back to that format. So you can convert to and from _**COLUMN** format without losing information.

Data type and units of "default" and item-specific data

Because a set can be used in so many different contexts PRIMER does not attempt to determine the meaning of default or item-specific data.

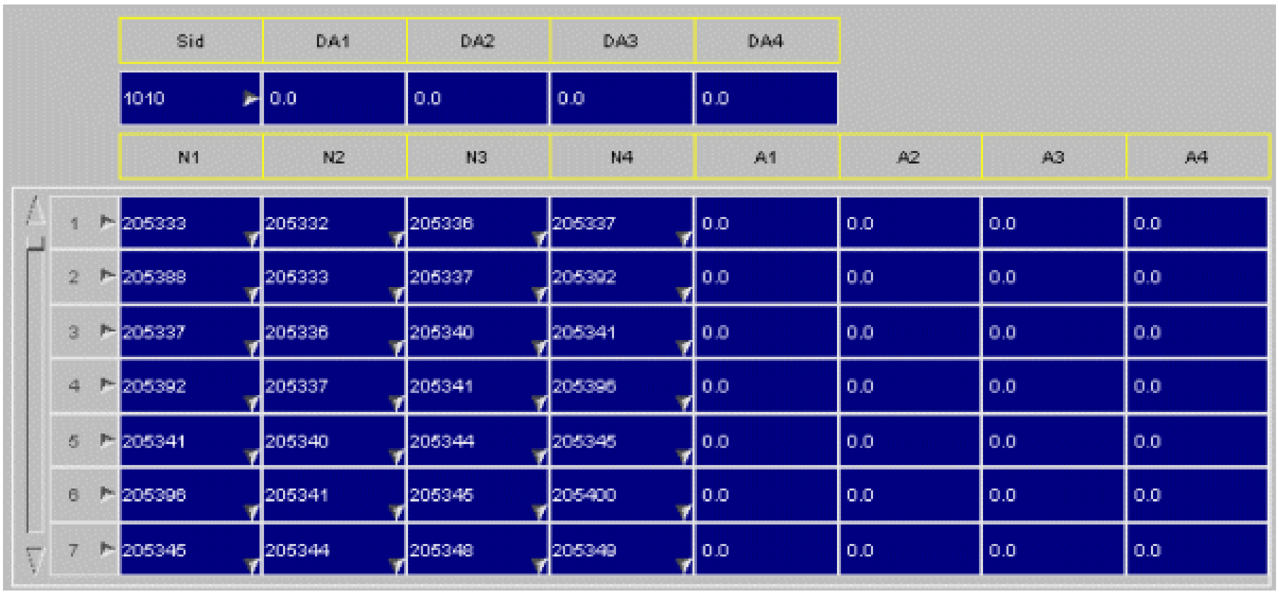
Therefore the data are treated as simple numbers, and no units conversion or checking is applied to them.

SET_SEGMENT The special case of segment sets

Segment sets are a special case for two reasons:

- A "segment" is not a structural item, although it may be drawn to look like one, and has no existence outside the context of its parent set definition (or load etc). It has to lie on top of a shell element, or the face of a solid or thick shell.
- Therefore segment sets actually create their constituent segments, rather than just referring to them, which makes the rules for their definition and deletion different to those of other set types.

As a consequence the layout and operation of the **`SET_SEGMENT`** editing panel is slightly different to that of the others.



Sid	DA1	DA2	DA3	DA4
1010	0.0	0.0	0.0	0.0

	N1	N2	N3	N4	A1	A2	A3	A4
1	205333	205332	205336	205337	0.0	0.0	0.0	0.0
2	205388	205333	205337	205392	0.0	0.0	0.0	0.0
3	205337	205336	205340	205341	0.0	0.0	0.0	0.0
4	205392	205337	205341	205396	0.0	0.0	0.0	0.0
5	205341	205340	205344	205345	0.0	0.0	0.0	0.0
6	205396	205341	205345	205400	0.0	0.0	0.0	0.0
7	205345	205344	205348	205349	0.0	0.0	0.0	0.0

Each segment is defined in terms of 4 nodes (n3 = n4 for a triangle), and the extra item-specific data occupies columns 5 to 8.

There is no choice of set format type.
Segments may be added to a set by duplicating those elsewhere in the model, or created using the COAT function.

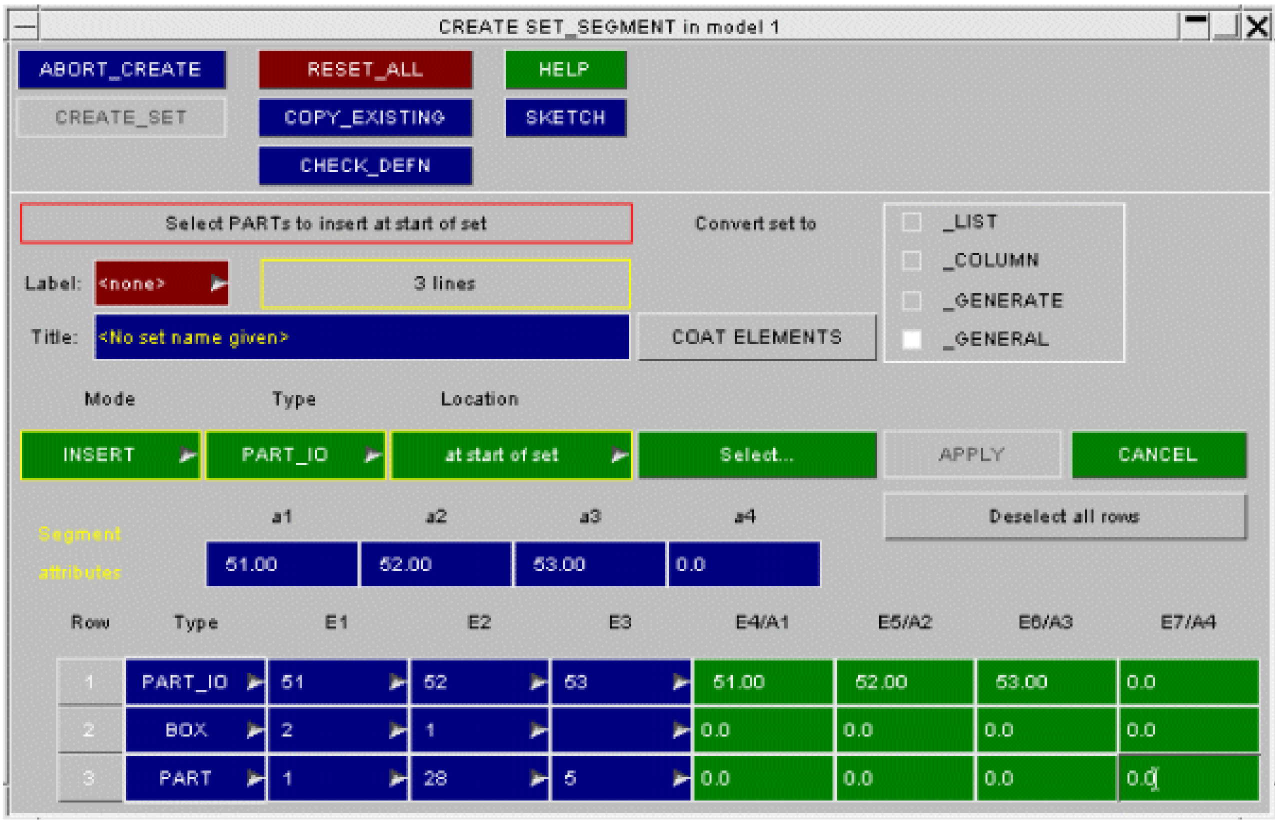
Creating/editing **`SET_SEGMENT_GENERAL`** sets

General segment sets are edited in the same way as other general sets. If needed segment attributes can be defined when inserting rows by using the **a1** to **a4 Segment attributes** boxes.

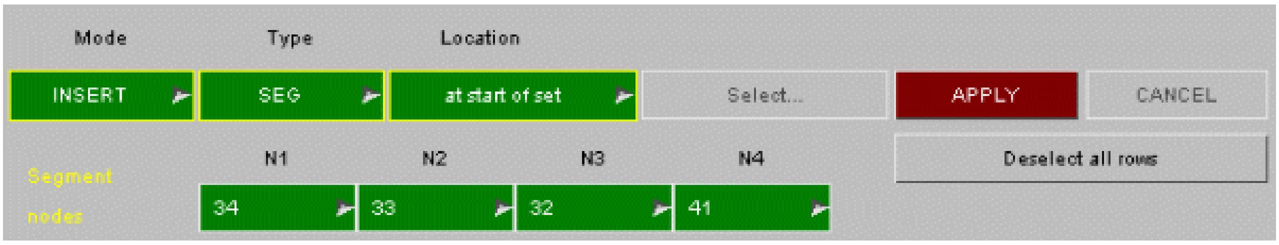
SEG and **DSEG** rows are added in a different way. These are inserted by selecting the four nodes.

Example: adding a **SEG** row to a **SET_SEGMENT_GENERAL**

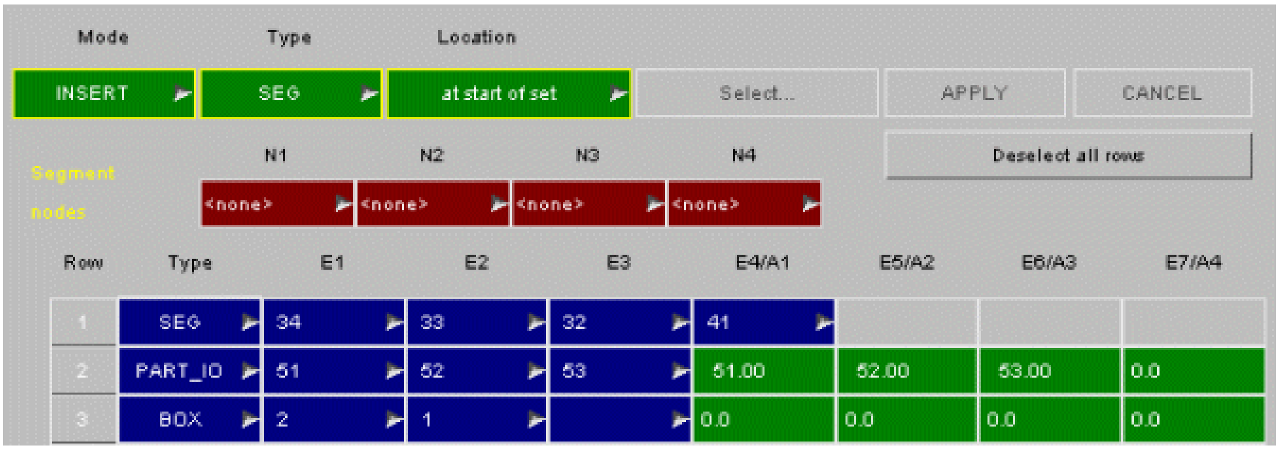
We want to add a SEG row to the top of the **SET_SEGMENT_GENERAL** set shown below:



Select mode **INSERT** , type **SEG** and location **at start of set** from the popups.

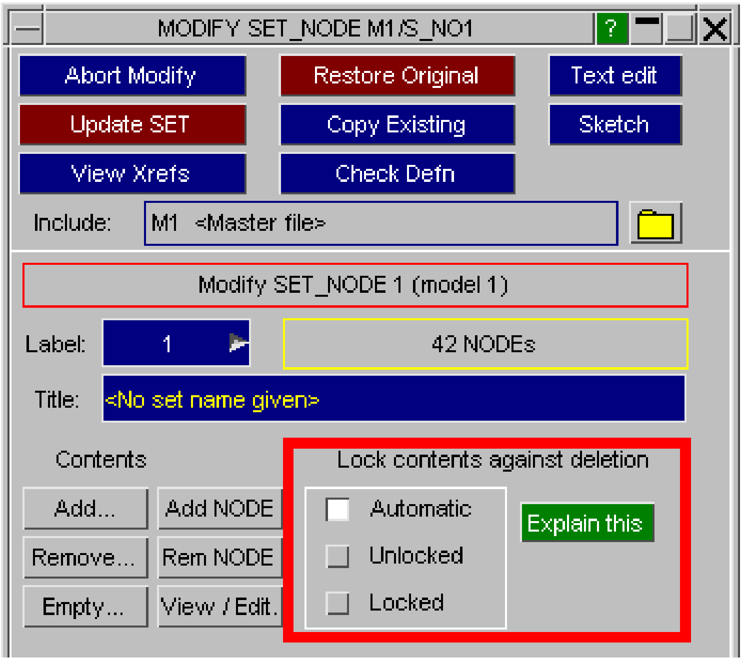


As we have selected **SEG** the Segment nodes popups are made live to be able to select the 4 nodes on the segment (for a triangular segment N3 = N4). Once the nodes are selected press **APPLY** to insert the row.



The row has been inserted at the top of the set.

Locking set contents against deletion



When items are deleted in PRIMER it has historically been the case that membership of a set does not - by default - "lock" an item against being deleted. This can be controlled globally on the deletion panel via the **Remove from sets** switch .

However there are a few cases where this logic can cause problems:

- Sliprings using 2D seatbelt elements rely on the contents of node and shell sets to define their geometry and connectivity
- Retractors using 2D seatbelt elements have a similar reliance

  ... and other cases may arise in the future.

Therefore the concept of "locking" set contents against deletion on a "per set" basis has been added in PRIMER release 10.1.

Contents locking has three possible settings:

Automatic (default)	This is the default PRIMER behaviour that membership of a set does not lock contents against deletion ***unless*** the set is referenced by something known to be sensitive to this problem. At present such items are: • *ELEMENT_SEATBELT_SLIPRING when the slipring is for 2D seatbelt elements • *ELEMENT_SEATBELT_RETRACTOR when the retractor is for 2D seatbelt elements If either if thse types reference the set then its contents will be locked against deletion. This list of items may be added to in the future as the keyword format evolves.
Unlocked	This is the original PRIMER behaviour that membership of a set never locks contents against deletion, regardless of what references the set.
Locked	This is a new option, and if selected membership of a set always locks its contents against deletion

As stated above removal from sets during deletion also depends upon the **Remove from sets** switch being turned on.

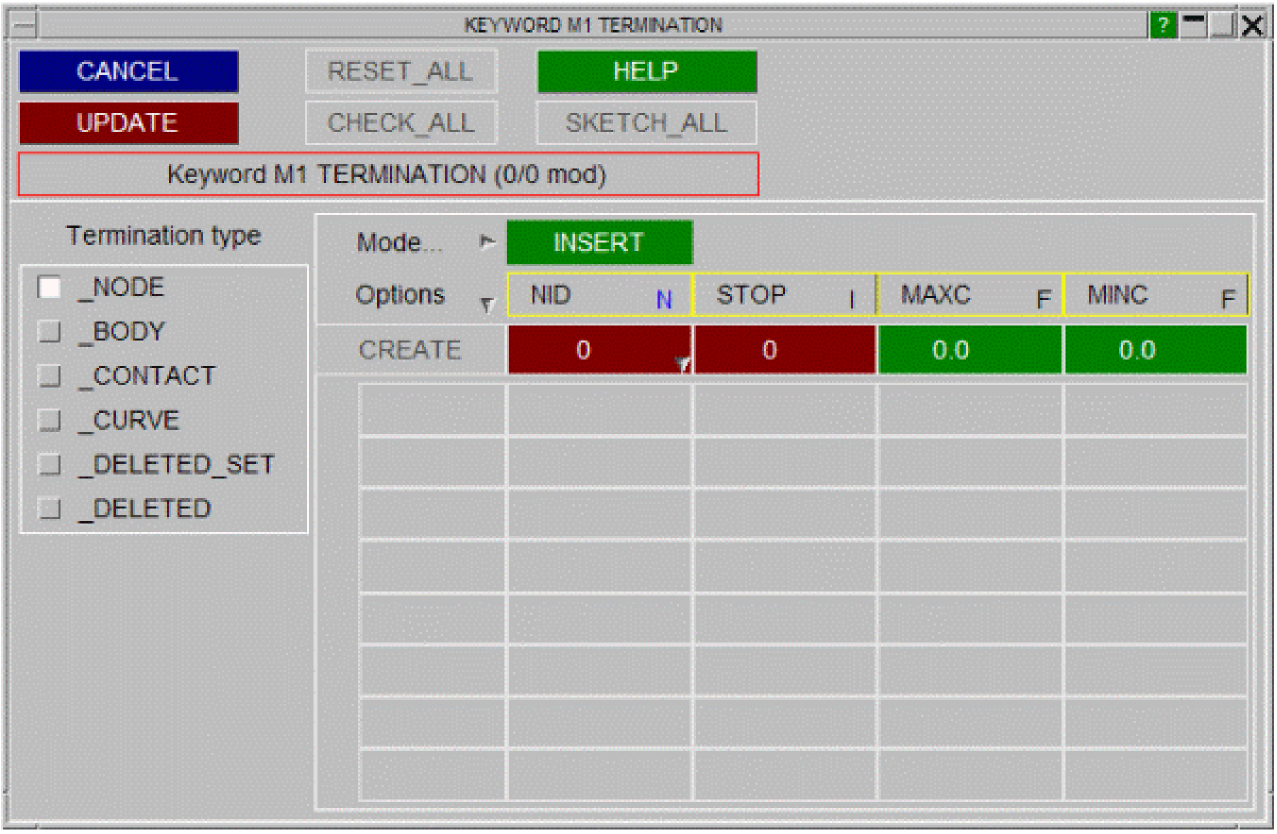
This setting is not "remembered" in the keyword output deck, so it will be lost when a model is deleted or a PRIMER session is terminated. It is hard to think of a situation in which the default **Automatic** setting will not be appropriate, and the use of this default is recommended, however the other two options are provided for completeness.

Visualising Sets.

Sets are not drawn explicitly, rather they are displayed by drawing their constituent parts, elements, nodes and segments. They may be **SKETCH**ed via the commands above, and in most contexts within PRIMER where sets are used it is possible to sketch them via their "daisy chain" popup menus.

TERMINATION

There are currently five sub-keywords available, _BODY, _CONTACT, _CURVE, _DELETED_SHELLS, and _NODE. These can be edited through the generic Keyword Editor.



5.2 Databases: Importing data from Pre-defined database files

Database files allow you to access predefined data of any type. Typical uses are material, section and loadcurve data; but anything which can be stored in tabular form may be accessed in this way.

The format of databases in PRIMER is described in appendix IX.

This section shows an example database and how it can be used in PRIMER.

An example of how to set up a database and how to use it.

A user 'guest' has set up:

- A **.database** file in his home directory (**$HOME**) which is **/guest**.
- There is also a global **.database** file in directory **$OASYS**.

These files are:-

in $OASYS

```
$ .database file for PRIMER in $OASYS
$
$ Any databases which are defined in here will be available to all users
$
$======================================================================
$
$ type directory to find .index file in  database name
$ ==== ================================  =============
$
LCUR*  /disk/database/loadcurve/seismic  example seismic loadcurve database
LCUR*  /disk/database/loadcurve/material example material loadcurve database
MATL*  /disk/database/material example   material database
SECT*  /disk/database/section example    section database
```

in $HOME

```
$ .database file for PRIMER, user 'guest' in $HOME (/guest)
$
$ Any databases defined here will only be available to the user 'guest'
$
$=====================================================================
$
$ type directory to find .index file in database name
$ ==== ================================ =============
$
LCUR*  /guest/my_database/loadcurve     my loadcurve database
MATL*  /guest/my_database/material      my material database
SECT*  /guest/my_database/section       my section database
```

Index files

In this example when creating a material in PRIMER, we will import a material stress strain curve from a database. Note that this example describes a loadcurve database, while the more common database type is the material database. The material definitions then include the curve data (*DEFINE_CURVE as well as *MAT).The database we will import the curve from is 'example material loadcurve database'. This database is in the directory '**/disk/database/loadcurve/material**'. In this directory there MUST be a **.index** file which contains the database information.

The **.index** file located in this directory is shown on the next page.

The database contains 8 fields per entry and there are 10 entries (4 cold reduced steels and 6 hot rolled steels). Each entry refers to a T/HIS curve file (as this is a **LCUR** database) in the directory (**/disk/database/loadcurve/material**)

```
$ Material stress-strain
curves
$

$==============================
$ NUMBER OF COLUMNS IN
DATABASE
8

$==============================
$ THE COLUMN NAMES (FILENAME
FIRST)
Filename
Material
Grade (GB)
Grade (Germany)
Grade (Japan)
Grade (USA ASTM)
Units
Description

$==============================
$ THE DATABASE ENTRIES
$
$ COLD REDUCED STEELS
$ -------------------
cr4_steel.cur
Steel
CR4
St12
SPCC
A366
MPa
Cold Reduced - Forming and
Drawing
$
cr3_steel.cur
Steel
CR3
St13
SPCD
-
MPa
Cold Reduced - Forming and
Drawing
$
cr2_steel.cur
Steel
CR2
-
SPCE
A619
MPa
Cold Reduced - Forming and
Drawing
$
cr1_steel.cur
Steel
CR1
RRSt14
SPCEN
A620
MPa
Cold Reduced - Forming and
Drawing
$
```

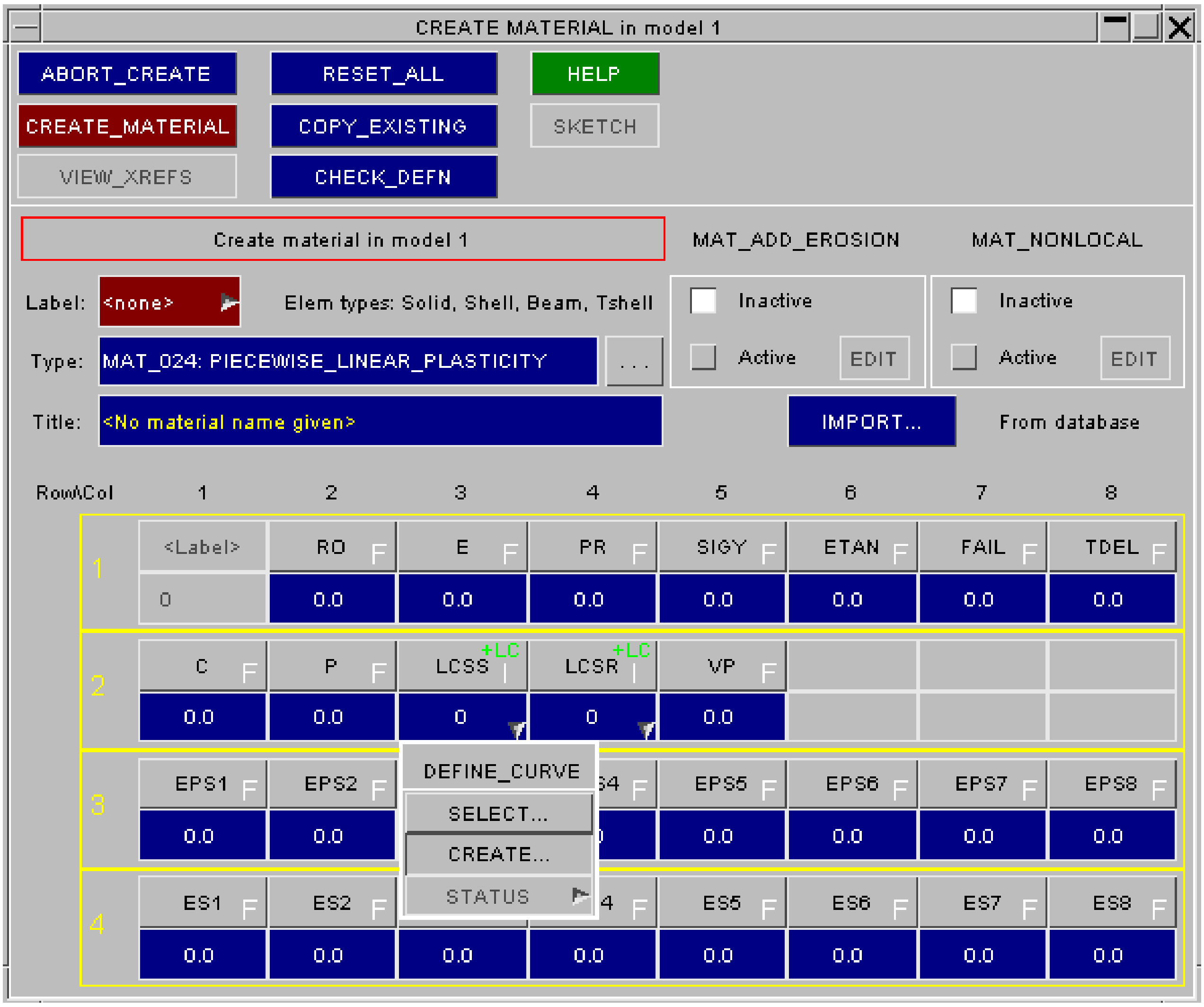
[continued on next column]

[from previous column]

```
$-----------------------------------
$ HOT ROLLED STEELS
$ -----------------
hr15_steel.cur
Steel
HR15
-
-
-
Mpa
Hot Rolled - Forming and
Drawing
$
hr14_steel.cur
Steel
HR14
-
-
A569
Mpa
Hot Rolled - Forming and
Drawing
$
hr4_steel.cur
Steel
HR4
-
SPHC
-
Mpa
Hot Rolled - Forming and
Drawing
$
hr3_steel.cur
Steel
HR3
StW22
SPHD
-
Mpa
Hot Rolled - Forming and
Drawing
$
hr2_steel.cur
Steel
HR2
StW23
-
A621
Mpa
Hot Rolled - Forming and
Drawing
$
hr1_steel.cur
Steel
HR1
StW24
SPHE
A622
Mpa
Hot Rolled - Forming and
Drawing
```

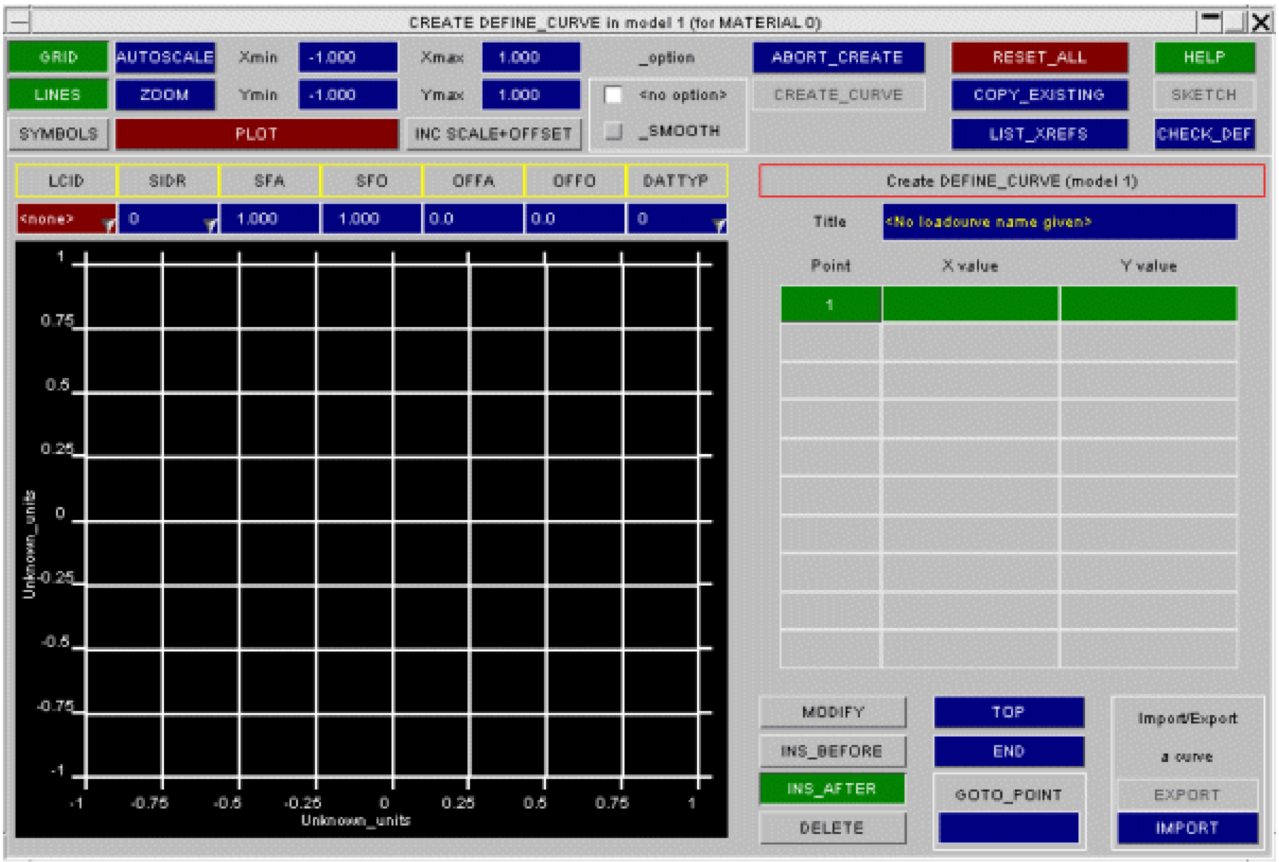
In the material creation window for PRIMER a material which refers to a loadcurve (**PIECEWISE_LINEAR_PLASTICITY**) is selected.

To import a loadcurve for the stress strain curve use the right mouse button to bring up the **LCSS** popup box. Select **CREATE** from the menu and this will start the Loadcurve creation box.

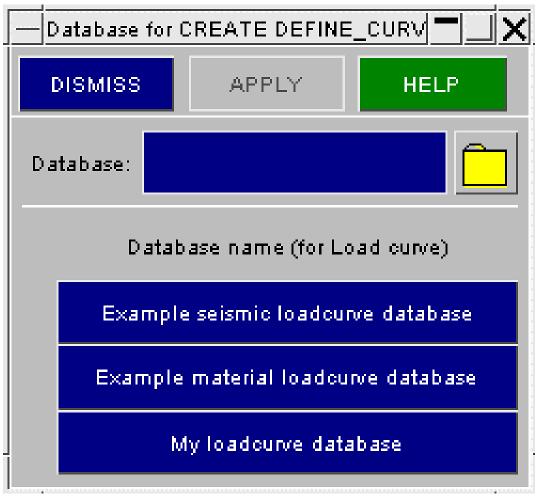


The loadcurve creation box allows a new curve to be created.

The **IMPORT** button on the bottom right of the window allows a curve to be read in from a file or database. Press this button.



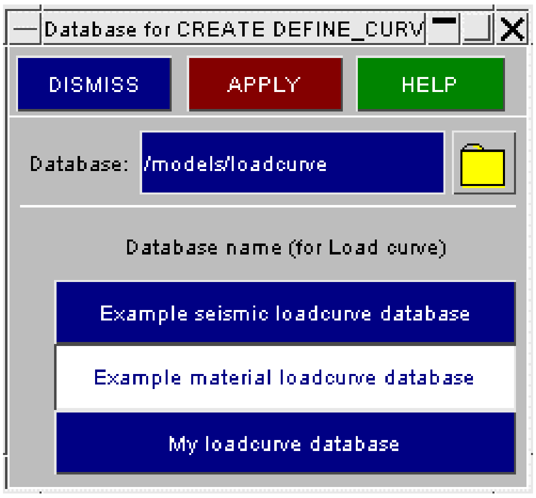
The Import part of the loadcurve creation box allows a curve to be read from a database by pressing the **DATABASE** button.



Pressing the DATABASE button in the loadcurve creation window starts the database selection window. Each button in the window corresponds to an entry in a .database file. The first two buttons are from the **.database** file in $OASYS. The last button is from the .database file in $HOME.

Only 3 databases are shown as these are the only ones which refer to **LCUR** databases.

Until a database is selected the **APPLY** button is inactive (greyed out).

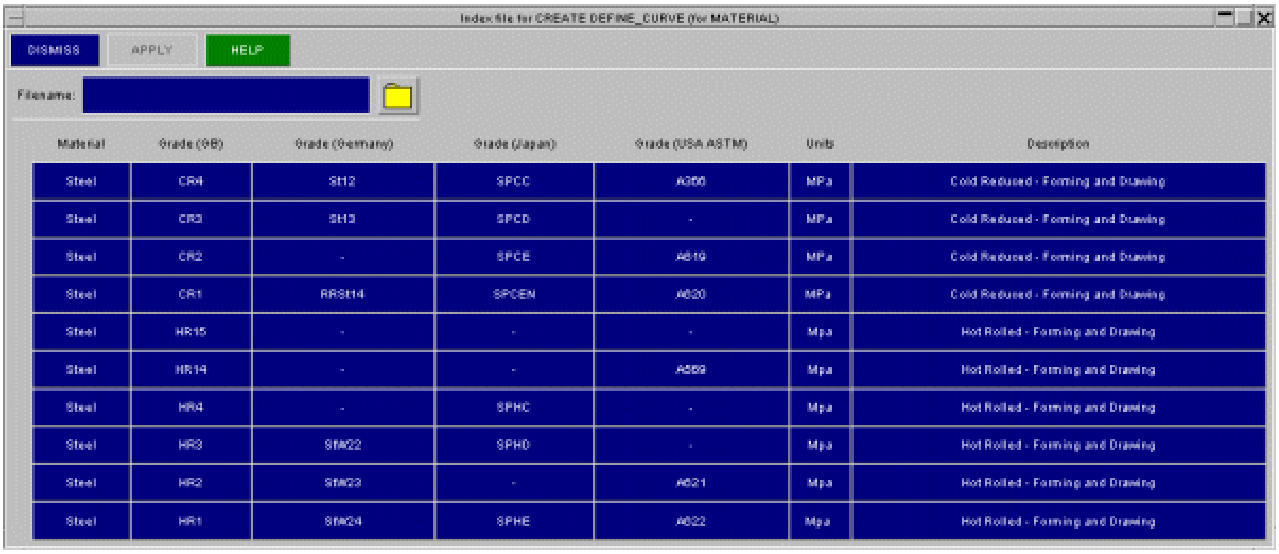


When a database file is selected it is highlighted and the **APPLY** button becomes RED allowing the user to select that database.

A different database can be selected as required. The one which is highlighted when **APPLY** is pressed is the one which will be read.

In our example we select a file from ’example material loadcurve database’.

When **APPLY** in the database selection window is pressed, PRIMER reads the .index file which is in that directory and creates a window with the entries from this file.

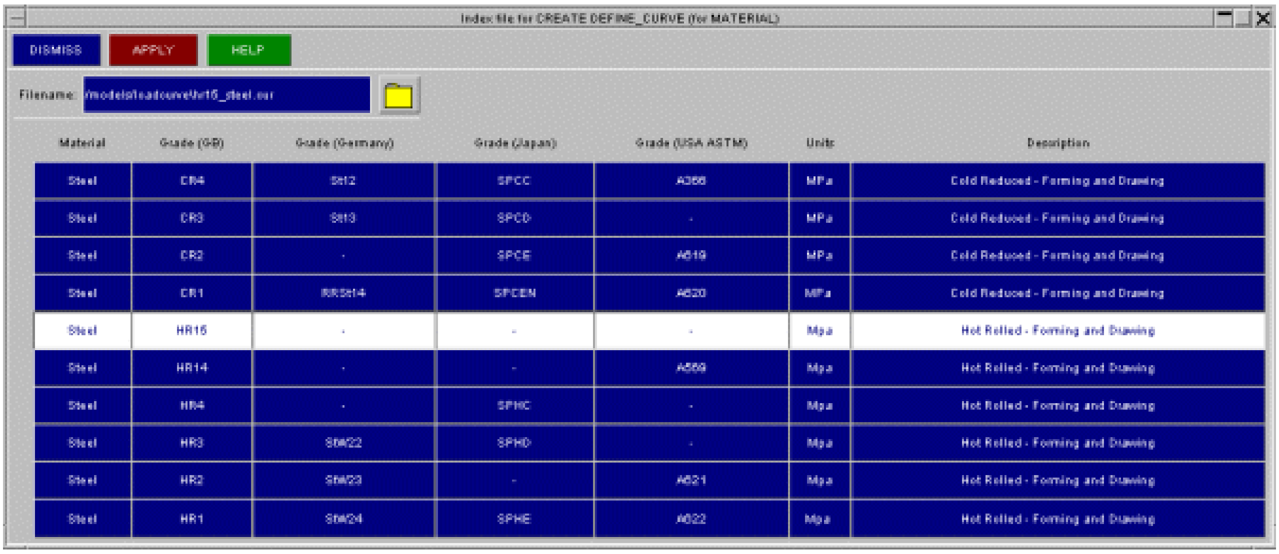


In this example you can see that the 10 entries which were in the .index file are all present in the window, each appearing on a row. If there were more then 10 entries in the .index file a scroll bar would allow you to scroll through the entries. Eight fields were defined for each entry. The first (the filename) has not appeared in the window but is stored internally. The remaining seven field headers appear above each column in yellow. If the number of fields does not fit on the window a scroll bar will be displayed.

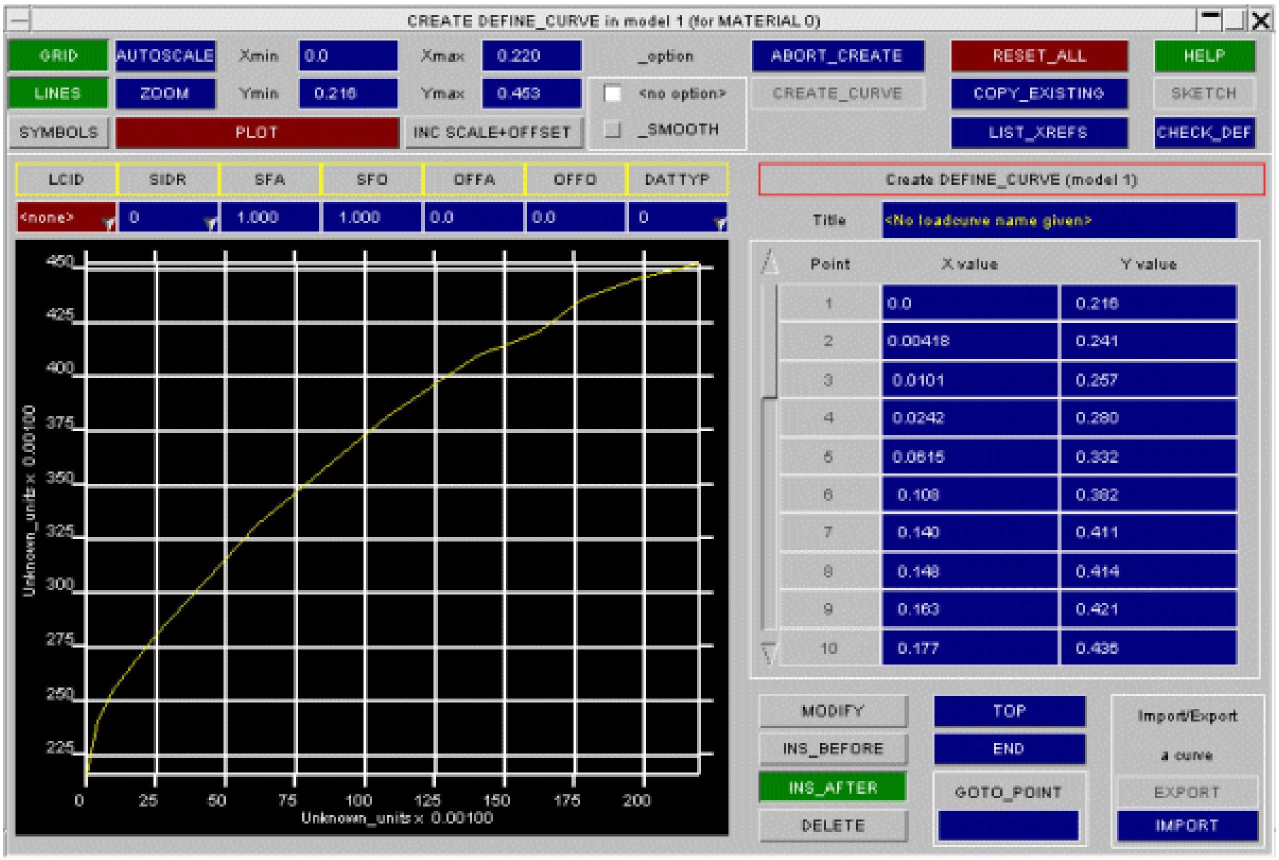
Until a file is selected the **APPLY** button is greyed out.

When a database entry is selected it is highlighted and the **APPLY** button becomes RED allowing the user to select that entry. A different entry can be selected as required. The one which is highlighted when **APPLY** is pressed is the one which will be read.

Here the HR15 steel is selected.



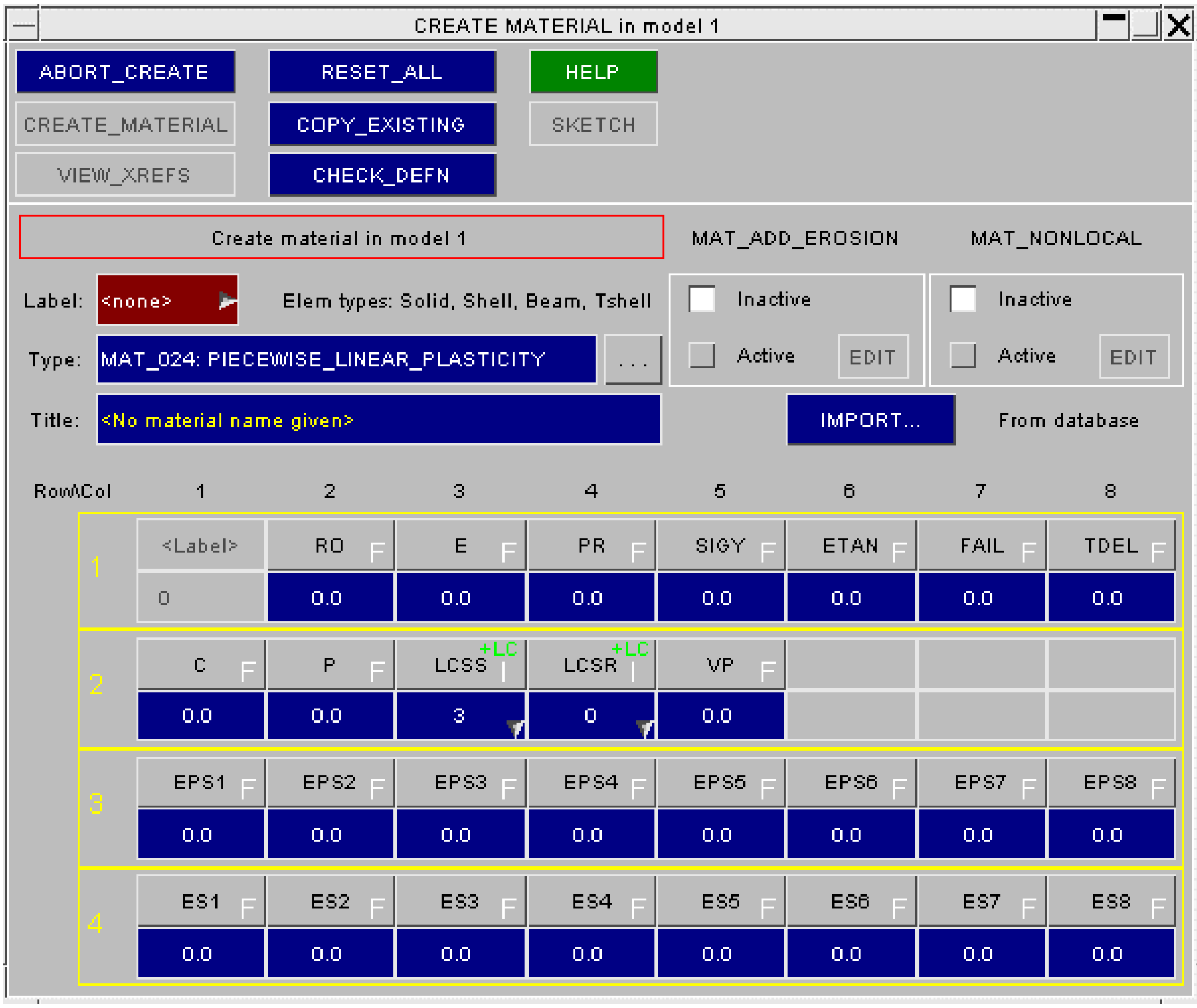
If **APPLY** is pressed this will be imported into the loadcurve editor and plotted.



The curve can be modified if required in the editor.

When a label has been given to the curve the **CREATE_CURVE** button will be ungreyed (made active) allowing the curve to be created.

When this is done the **LCSS** field in the material editor is updated with a new loadcurve ID referencing the imported data.


CREATE MATERIAL in model 1
ABORT_CREATE
RESET_ALL
HELP
CREATE_MATERIAL
COPY_EXISTING
SKETCH
VIEW_XREFS
CHECK_DEFN
Create material in model 1
MAT_ADD_EROSION
MAT_NONLOCAL
Label: <none>
Elem types: Solid, Shell, Beam, Tshell
Inactive
Active EDIT
Inactive
Active EDIT
Type: MAT_024: PIECEWISE_LINEAR_PLASTICITY
Title: <No material name given>
IMPORT...
From database
Row\Col 1 2 3 4 5 6 7 8
1 <Label> RO E PR SIGY ETAN FAIL TDEL
0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
2 C P LCSS LCSR VP
0.0 0.0 3 0 0.0
3 EPS1 EPS2 EPS3 EPS4 EPS5 EPS6 EPS7 EPS8
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
4 ES1 ES2 ES3 ES4 ES5 ES6 ES7 ES8
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0

5.3 Contact Penetration Checking

Primer can check for and display initial penetrations and crossed edges in contact surfaces. It can also be used to find which nodes will be tied (or not tied) by *CONTACT_TIED.

This capability can be invoked from three separate locations:

CHECK (from Tools) > RULES
CONTACT > PEN_CHECK
CONTACT > CREATE/EDIT > PEN_CHECK

However the routines called are common to all cases, and their detailed use is described here.

Fixing contact penetrations can be sound in section 5.4

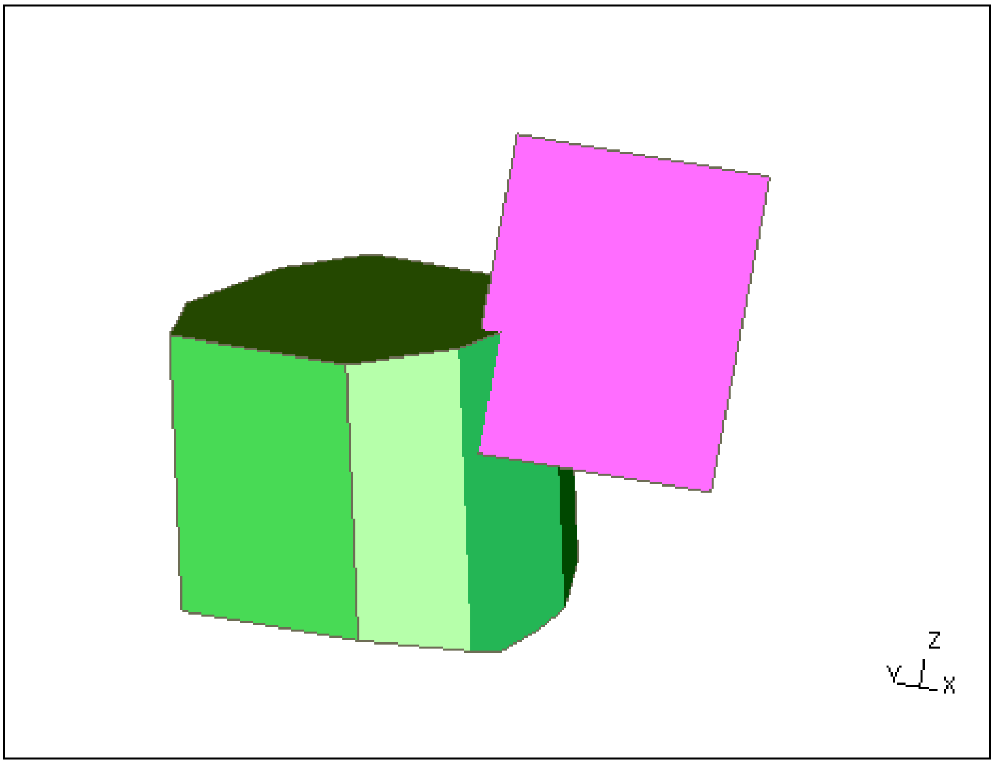
In this example a model contains two parts:

- Part 1 (pink) is a single shell.
- Part 2 (green) is a block of solids

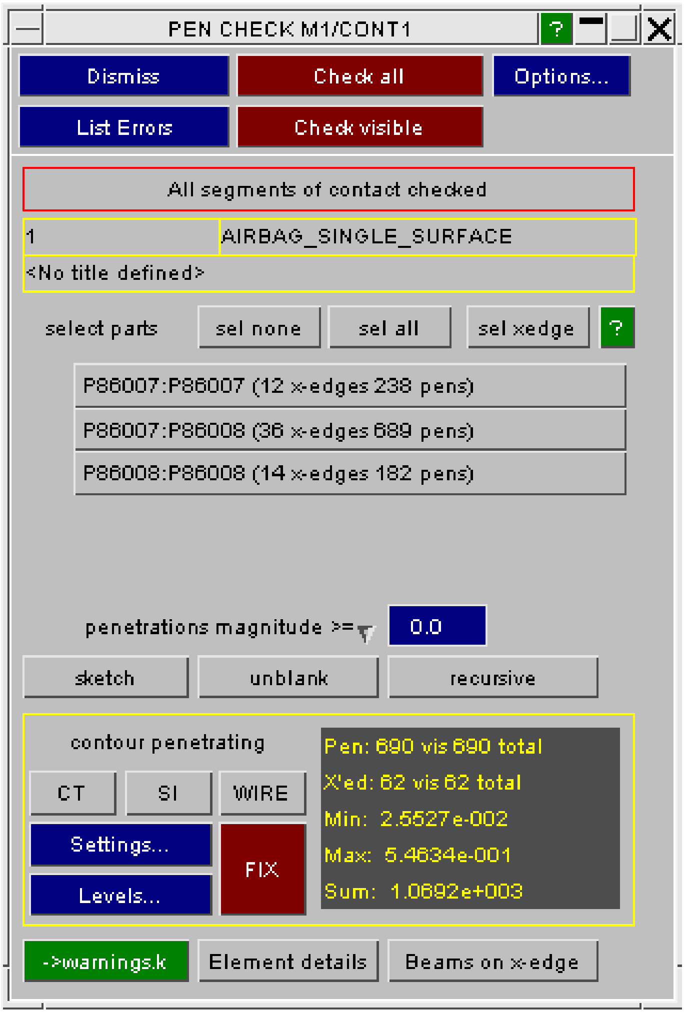
A contact surface (**AUTOMATIC SURFACE TO SURFACE**) between the two parts is defined:

- Part 2 is the slave side of a contact
- Part 1 is the master side

The two parts intersect by a small amount, as is clearly visible here.



5.3.1 Checking a Sliding Contact



Check all Checks the whole contact (recommended)

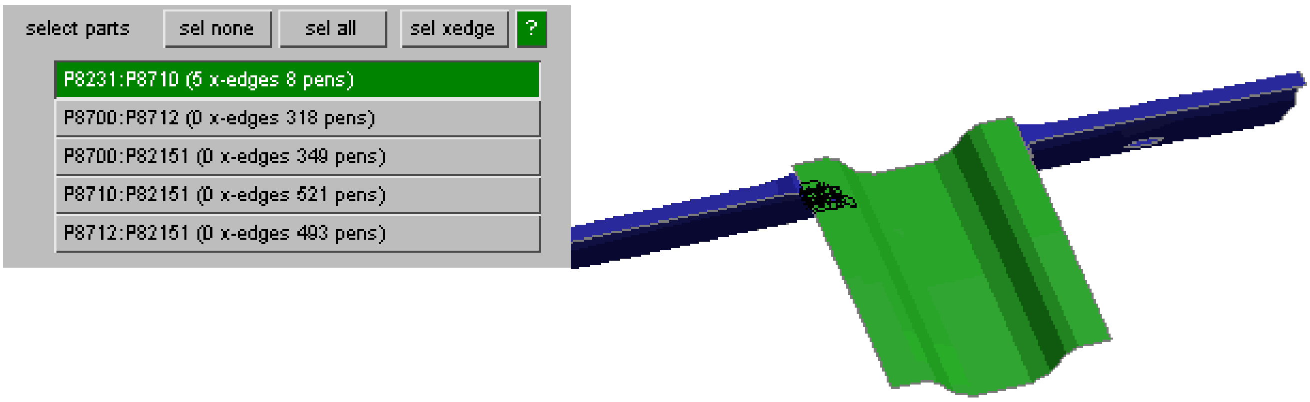
Check visible Only checks visible volume of the contact. If contact is large and substantial part of it is blanked, this option may save time particularly during iterative fixing procedures.

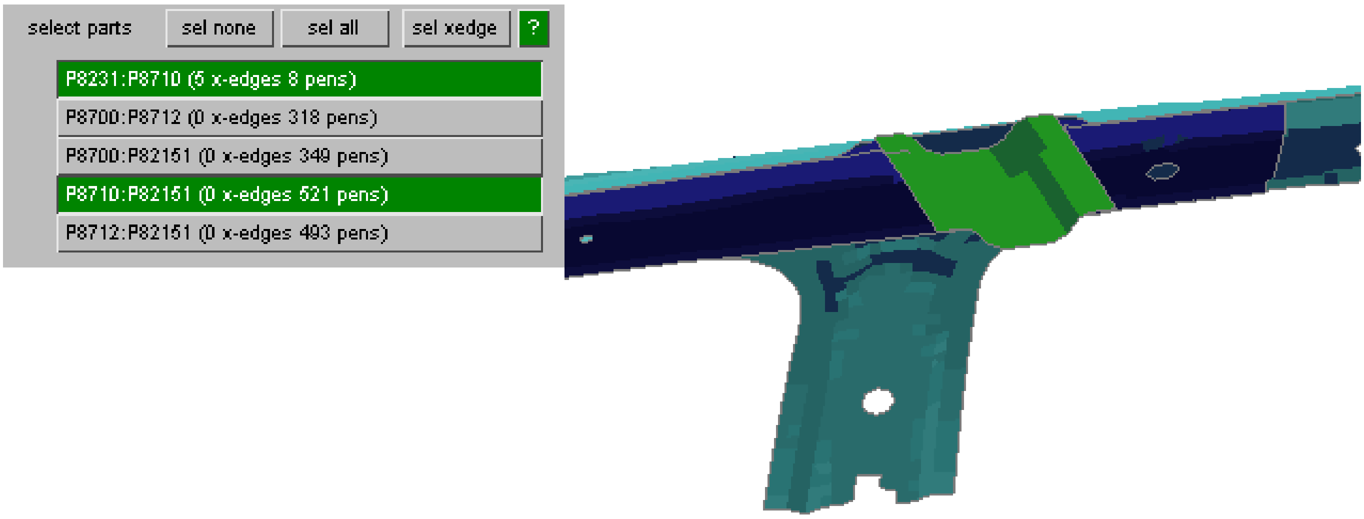
List Errors Lists all penetrating nodes, the element(s) they penetrate and the penetration distance. Also lists all crossed edges found.

Options... Maps the contact penetration check options panel in which checker settings may be changed.

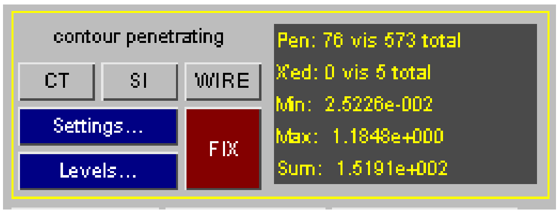
The front panel enables you to control visibility by unblanking interacting part pairs. It also pre-selects part pairs for fixing of crossed edges or penetrations in the default mode which is to observe blanking.

- Selection by a single click will exclusively select and "only" the display for a part pair.
- Ctrl-select will add another part pair to the selection
- Shift-select in this context will select all part pairs which use P1. e.g. Shift-sel on P8710:P82151 also unblanks P8231as in interacts with P8710.
- Ctrl-shift-select does the same for all part pairs which use P2.



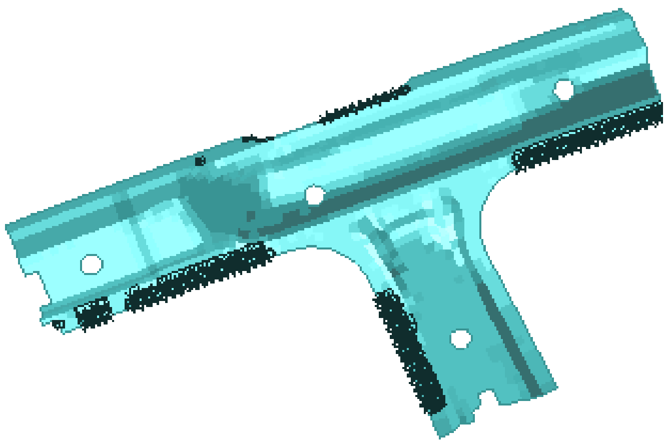


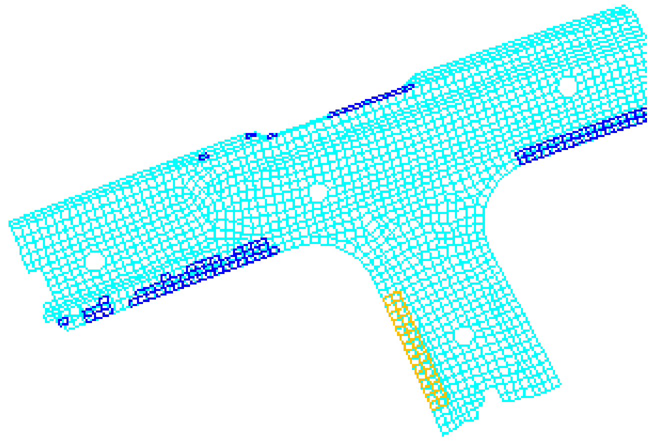
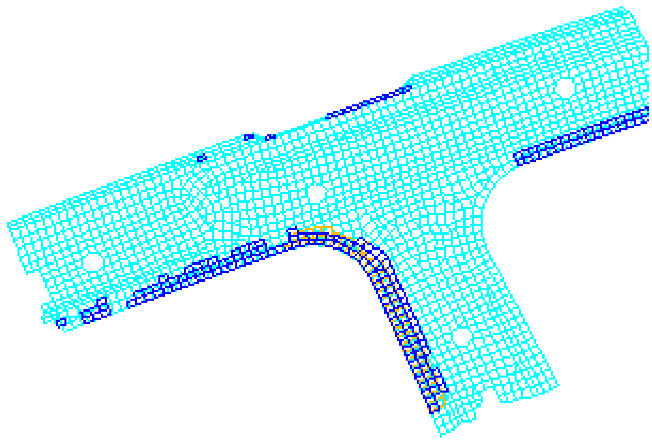
- **sel none** clears the selection and unblanks all parts in the contact
- **sel all** will "only" all the parts of the contact
- **sel xedge** will select all the interactions where crossed edges are found



The information panel gives the count of visible/total crossed edges and penetrations. Note - this is the (more useful) count of penetrating nodes not penetration events.

The following functions are especially useful when one part only is initially displayed.

	**sketch** will sketch all the visible segments involved in a penetration and all the segments that penetrate/cross them

	**unblank** will unblank underlying elements of all segments which penetrate/cross visible segments (i.e. blue and yellow shells)
	**recursive** will perform unblank until no more segments are found (note the extra blue shells which penetrate the yellow)

->warnings.k - this functions writes error sets to include file *warnings.k* which will be created if it does not exist. The same function is available for contacts from the model check error tree.

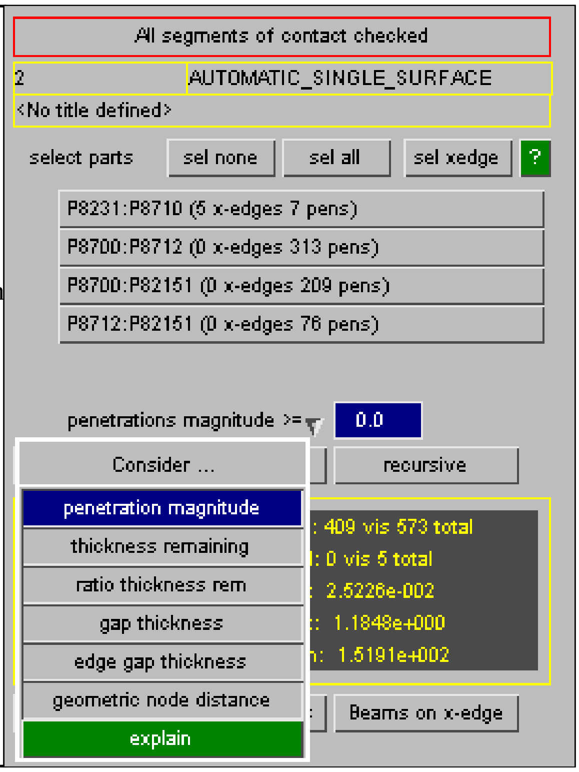
- If there are crossed edges in the contact, these will be written to a segment set named "*Contact <id>: Contact has crossed edges*".
- If there are penetrations, both a node set and a segment set will be created with matching name which also denotes the error, e.g. "*Contact <id>: penetration exceeds max allowable value @0.5*"

Filtering penetrations by magnitude

- Penetration magnitude - only consider penetrations greater than the given value
- thickness remaining - only consider penetrations where remaining unpenetrated thickness between segments is less than the given value. This is defined as 0.5*(t1+t2) - P, where t1,t2 are segment thicknesses and P is penetration magnitude
- thickness remaining ratio - only consider penetrations where remaining unpenetrated thickness expressed as ration of overall segment thickness is less than the given value

Changing this value will change the penetration count, the contour plot and the fixing procedure. Fixing will de-penetrate (not fully) but up to the specified limit.

This does not affect the handling of crossed edges.



CT, SI and **WIRE** provide Continuous Tone, Shaded Image and Wireframe plots respectively of the contact errors.

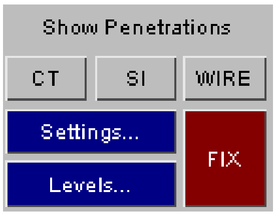
Settings... controls the parameters of these plots.
Levels... controls the contour bands used.

FIX accesses the de-penetration fixing function.

Element details permits more detailed examination of the errors in elements adjacent to a node, or in a particular part. The **CT, SI** and **WIRE** plotting modes are the same as above.

Beams on x-edge generates "null beams" on crossed edges. These can be used in external meshing programmes to identify where the problems occur, making remeshing easier.

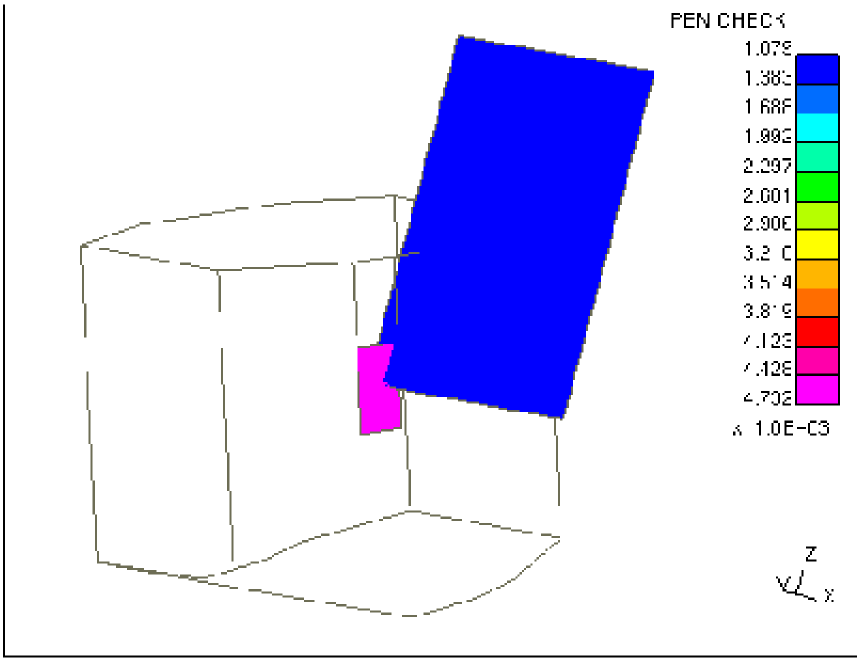
5.3.2 Plotting contact penetrations



This is a **CT** (Continuous Tone) plot of the contact penetrations.

Penetrated segments are drawn in a colour determined by the depth to which nodes penetrate them. The penetrating nodes and their "escape" vectors are drawn too.

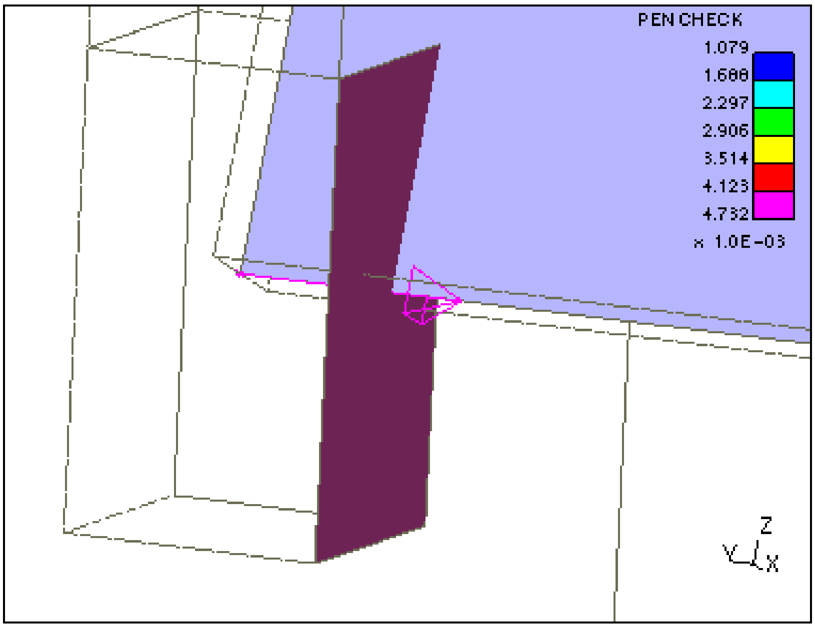
In this image the rest of the contact is drawn in "wireframe" mode: this, and other plotting parameters, are controlled in the **Settings...** panel.



This is a **SI** (Shaded Image) plot of the penetration region.

The escape vector of the shell node (to "escape" from the solid) is clearly visible.

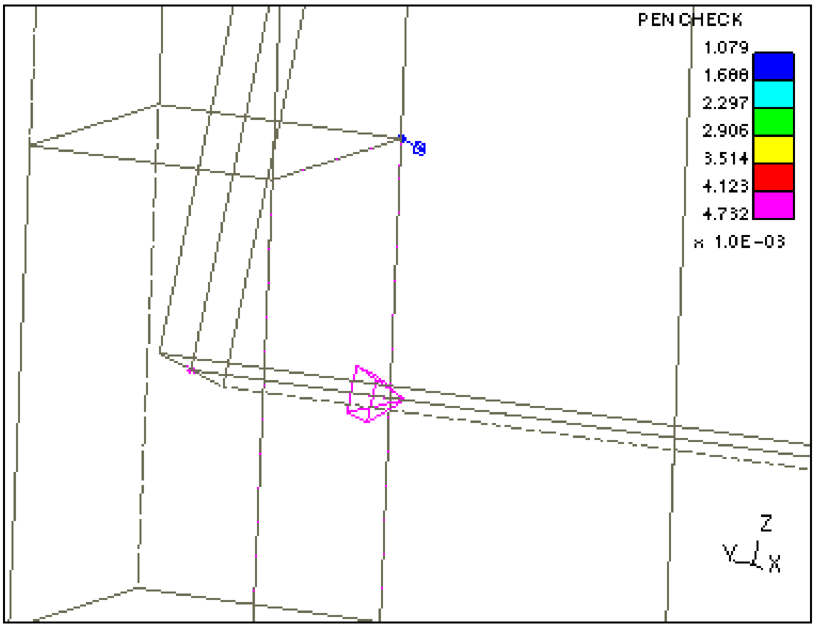
In addition the elements have been drawn "as thick" (controllable from the **Settings...** panel). This draws their thickness for contact purposes in grey lines.



This is a **WIRE** plot of the same region.

No shading or hidden surface removal takes place, and this makes it possible to see the other nodal penetration (of the solid node into the shell element). This was obscured in the previous plots.

The elements have still been drawn "as thick" here.



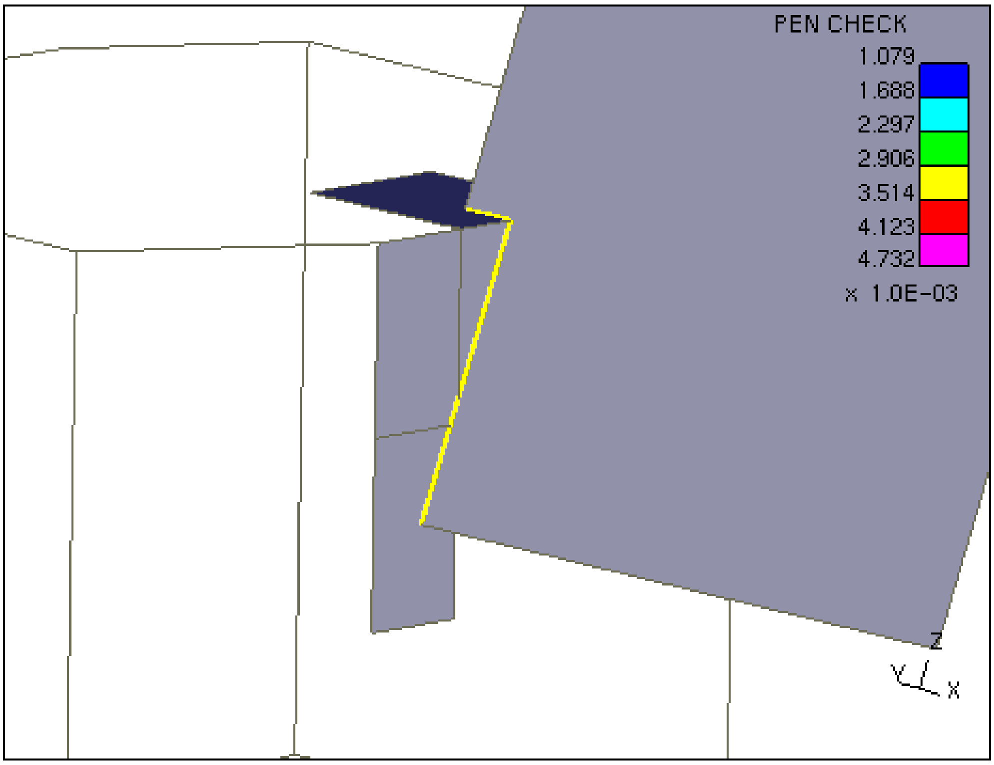
The plots above showed penetrations.

This is an `SI` plot of the crossed edges:

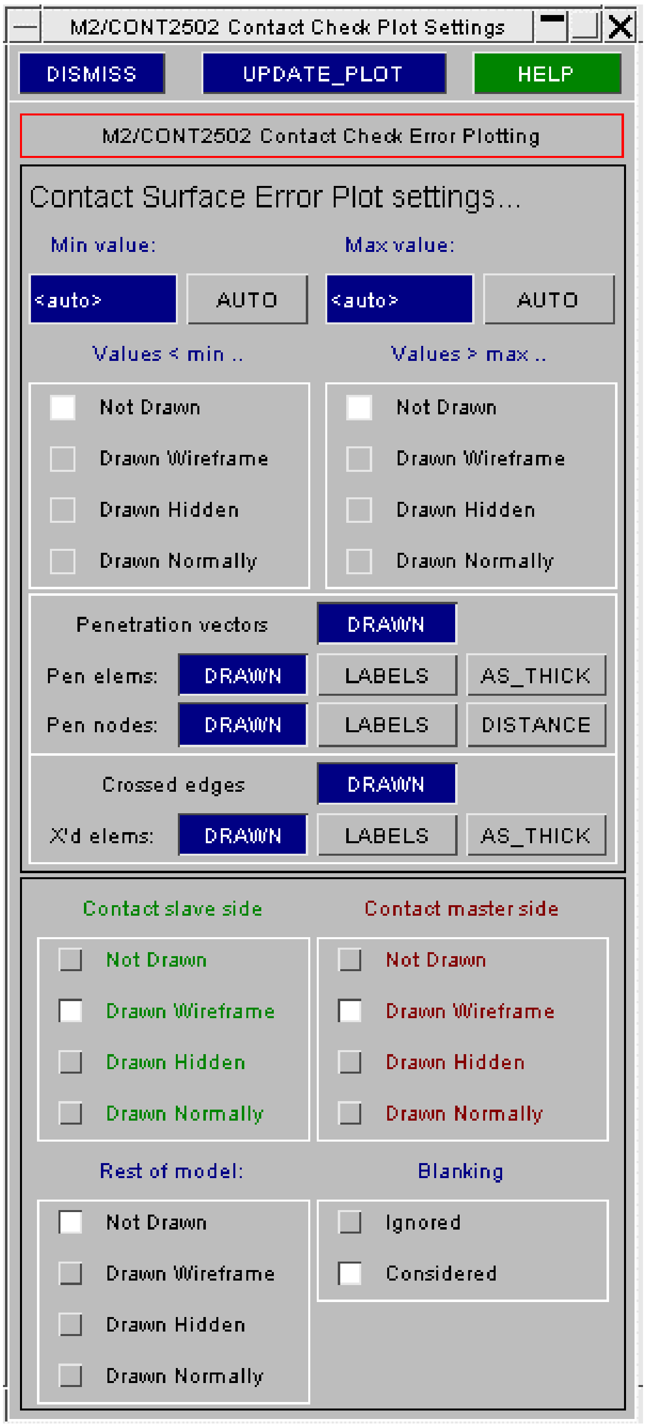
- Edges are drawn as thick yellow lines
- Penetrated elements are drawn in grey

It is possible to display penetrations and crossed edges on the same plot (the default) but this can lead to confusing images.

The display of each category of error is controllable separately in the **Settings...** panel.



5.3.3 Settings... Controlling plots



This section controls the min and max penetration distances that will be displayed (the contour band bounds).
By default the **Min value** and **Max value** are each automatic, displaying all penetrations, but each may be defined to a +ve value.

If either value is defined you can select how the segments which fall below/above this value are (or are not) displayed: [Not Drawn, Wireframe, ...].

This section controls what is actually drawn.

Penetration vectors: (which here are not drawn) are split into:

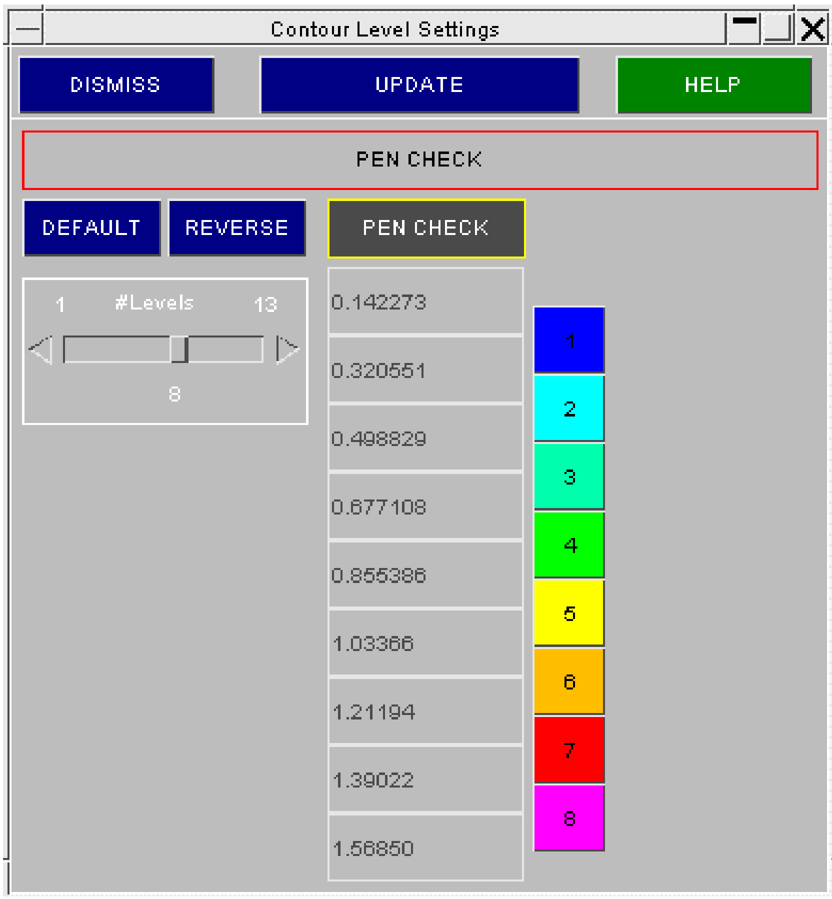
Penetrated elements	**DRAWN**	Whether or not they are drawn
	LABELS	Whether they are labelled (elem id)
	AS_THICK	Whether contact thickness is displayed
Penetrating nodes	**DRAWN**	Whether the nodes are drawn
	LABELS	Whether they are labelled (node id)
	DISTANCE	Whether penetration distance is added

Crossed edges:

Crossed elements	**DRAWN**	Whether or not they are drawn
	LABELS	Whether they are labelled (elem id)
	AS_THICK	Whether contact thickness is displayed

This section controls the display of each side of the contact [**Not Drawn**, **Wireframe**...] and also the rest of the model (same options).

5.3.4 **Levels...** Setting the contour bands

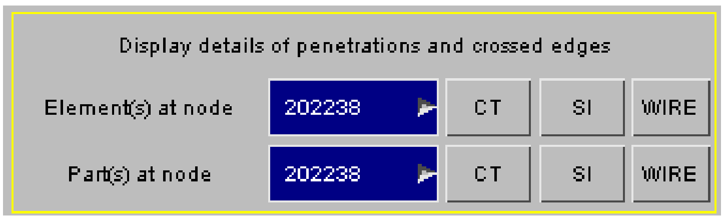


This panel (the standard panel in all contouring contexts) controls how many contour levels are displayed.

Note that it does not define the contour band values themselves:

> The upper and lower bounds are controlled in the **Settings...** panel, together with the display mode for items outside these bounds.

5.3.5 Details of errors local to elements and parts.

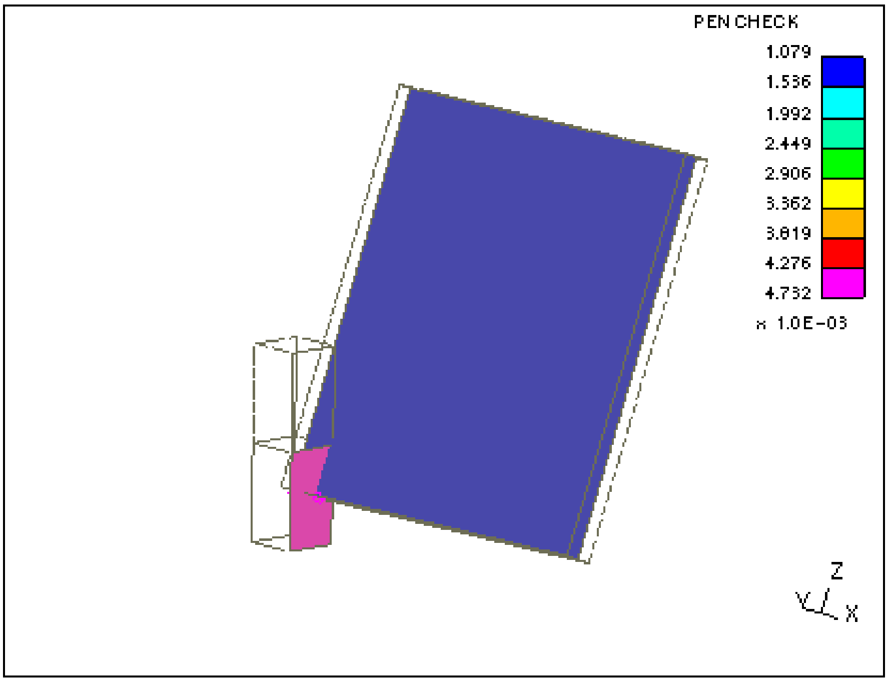


The main **CT**, **SI** and **WIRE** commands display errors for the whole model, as shown above.

The particular example illustrated is very simple, so display clutter is not a problem, but in a more complex model it is easy to imagine how confusing a plot of contact penetrations can get.

To make it easier to see what is going wrong you can select, by any means, a node (separate nodes may be selected for each case.)

Element(s) at the node



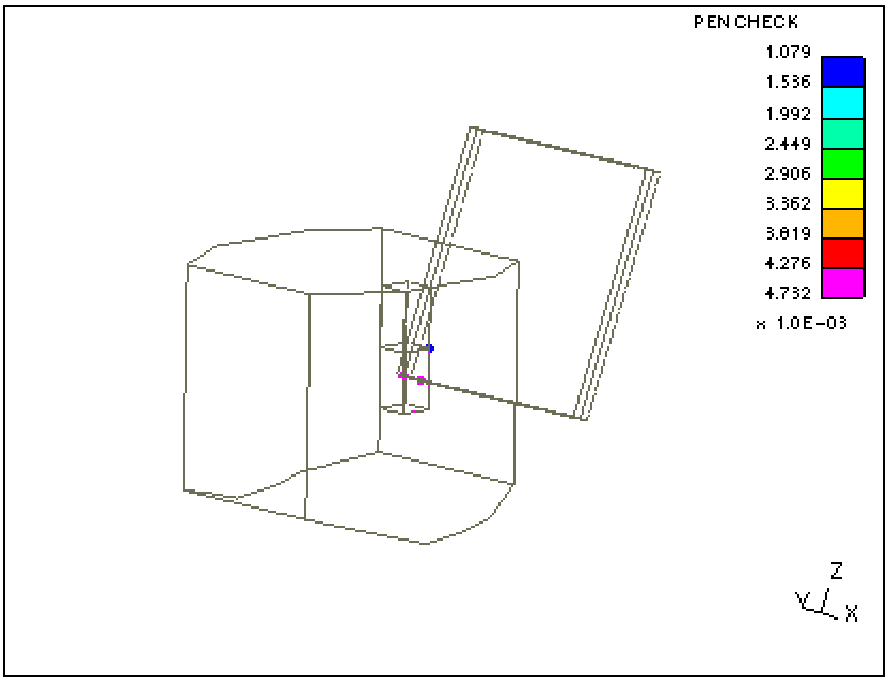
Only the elements related to this node are displayed, and the plot is autoscaled to these.

This means:
- All elements to which the node is attached
- All elements into which this node penetrates

This is an **SI** plot of the example.

All other plotting parameters (#levels, Settings... parameters, etc) are kept as before.

Parts(s) at the node

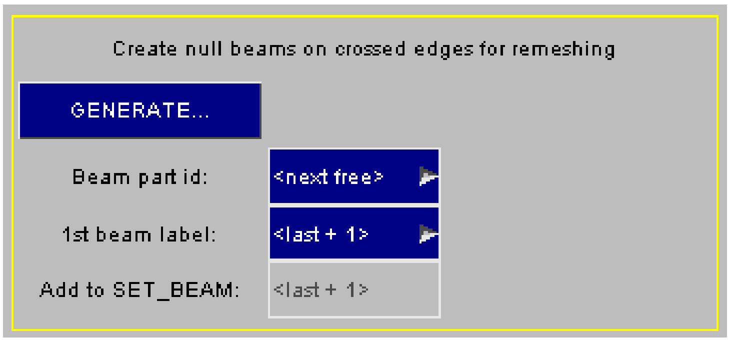


The elements related to this node are detected, as above; and all the elements of their respective parts are displayed in the mode selected in the **Settings...** panel.

This is a **WIRE** plot of the example.

Because this model is so simple the whole model, which only contains two parts, is drawn. However in a complex model this display mode allows you to determine how two parts interfere.

5.3.6 Generating "null beams" on crossed edges.



Generally the presence of crossed edges will require some remeshing, and this task will be performed outside Primer.

To make it easier to identify the edges externally you can generate "null beams" on these edges. A "null beam" normally (although you can change this) references **`*MAT_NULL`** and serves no structural purpose.

This procedure is entirely optional.
Primer allows you to control the following settings:

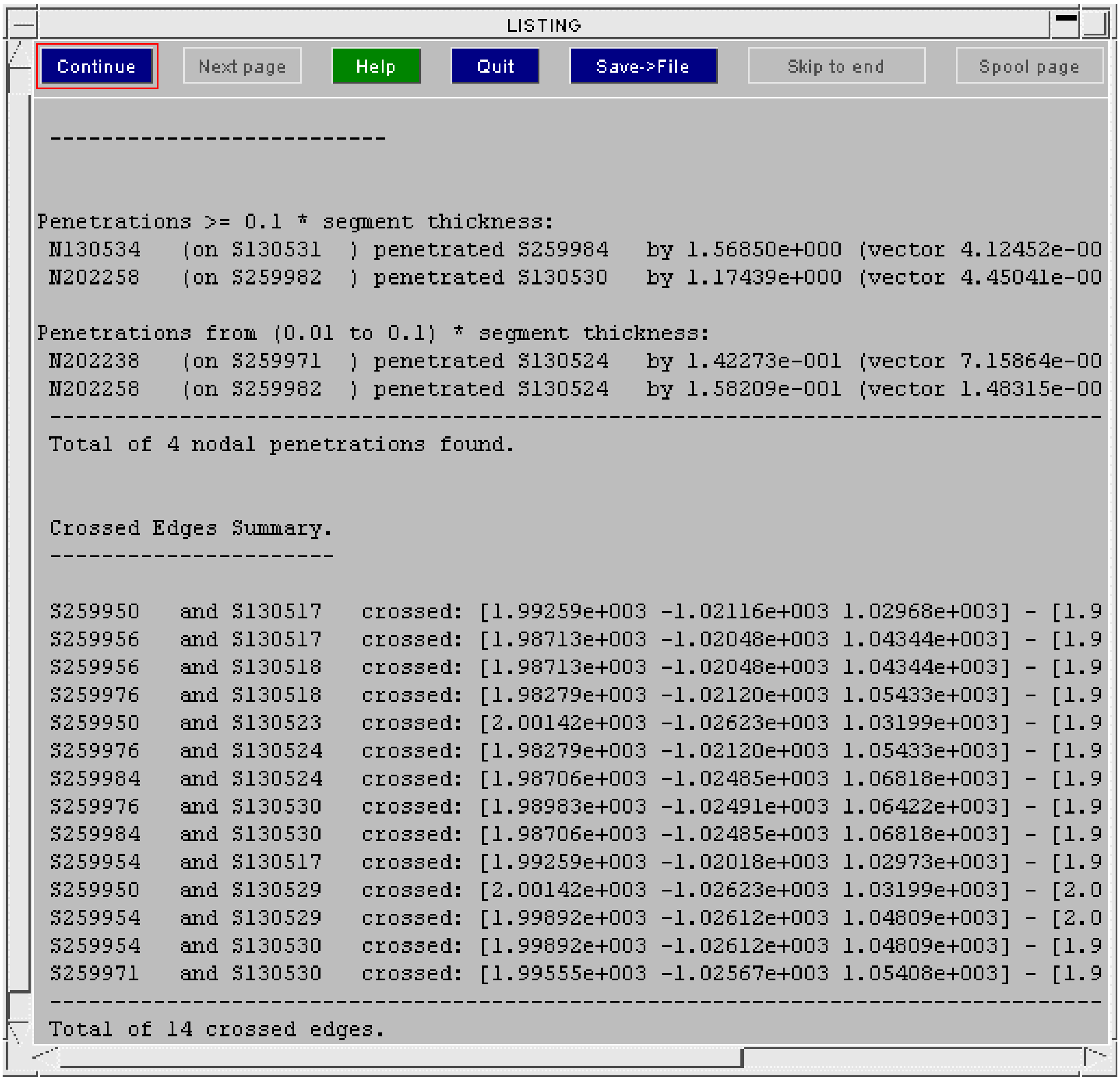
Beam Part id By default each set of null beams will be given new **`*PART, *SECTION_BEAM`** and **`*MAT_NULL`** definitions. Each will be given the "next free" label in its labelling sequence. If you want your beams to be generated in a specific part instead then define it here.

1st beam label By default beams will be generated using the "next free" beam label onwards. To generate them starting at a specific value (eg 100001) simply define the value: this may make them easier to identify.

Add to SET_BEAM By default the beams will not be placed in any sets. They may be easier to identify, and delete later when no longer required, if in a *__SET_BEAM__* definition. If you want to add them to one simply give its label: it will be created if it doesn't already exist.

Once any settings have been made **GENERATE** will create the beams. All nodes used for beams will be new nodes, starting at the "next free" node label.

5.3.7 LIST_ERRORS: Listing penetrations and edges to screen and file.



This example shows the listing of penetrations and crossed edges generated for the model above.

Nodal penetrations are grouped into the categories
- > 0.1 * Segment thickness
- From 0.01 to 0.1 * Segment thickness
- < 0.01 * Segment thickness

Crossed edges are simply listed in the order detected.

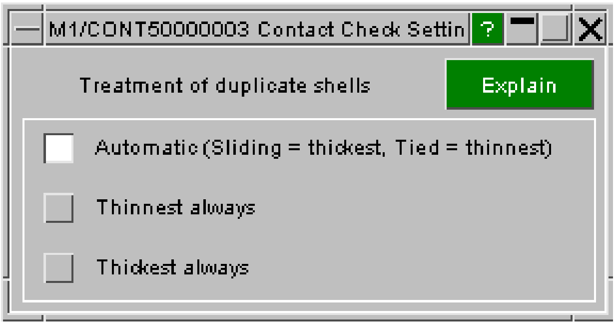
This listing is sent to the standard screen listing panel, and it may also be saved to file using the **SAVE->FILE** option.

5.3.8 Options... Controlling penetration checking

Treatment of duplicate shells.

This option controls which shell is used to determine contact properties when a contact segment lies on two or more coincident shells.

This requires a little explanation:



When LS-DYNA receives a list of elements, segments, parts or sets to define the geometry of a contact surface it uses them as follows:

- It builds a list of 3 or 4 noded segments from the shells or faces of 3D elements. (Incoming segments are used verbatim)
- It culls any duplicate segments (identical topology) from this list
- Then it searches through the whole model (not just the elements defined for contact) to find shells or 3D element faces under these segments.
- The chosen element under each segment is used to determine the its thickness and stiffness properties.

Clearly two ambiguous cases can arise when finding "the element under the segment":

1. A segment lies on a 3D element face that is also overlaid by a shell.

   In this case LS-DYNA prefers the shell element unless the I2D3D flag on *CONTACT optional control card B is set to 1, in which case the solid element is preferred. The contact checker in PRIMER examines the I2D3D flag and applies the same logic.

2. A segment lies on a shell that is one of two or more duplicate shells sharing the same topology.

   In this case the shell element used by LS-DYNA is not determinate and it is not currently possible to state which shell element will be used for contact properties. Tests also suggest that the choice in the SMP and MPP versions may be different, so that if any initial penetrations exist they may be computed differently.

In order to address the second of these two cases PRIMER uses this option to define behaviour explicitly when a segment lies on duplicate shells, using one of three settings:

Automatic (default)	In this mode the thinnest shell is preferred for "Tied" contacts, and the thickest for all other types. This approach is chosen to give conservative behaviour since "Sliding" types may over-report penetrations when compared with LS-DYNA, and "Tied" types may under-detect ties. Therefore the results of a contact check in PRIMER may be pessimistic when compared to initialisation in LS-DYNA, but it will err on the side of safety. This is the default setting.
Thinnest always	The thinnest shell is always used.
Thickest always	The thickest shell is always used.

The default value of this option may be varied by the preference:

`primer*contact_penchk_dup_shells:` with the possible values `automatic`, `thinnest` or `thickest`

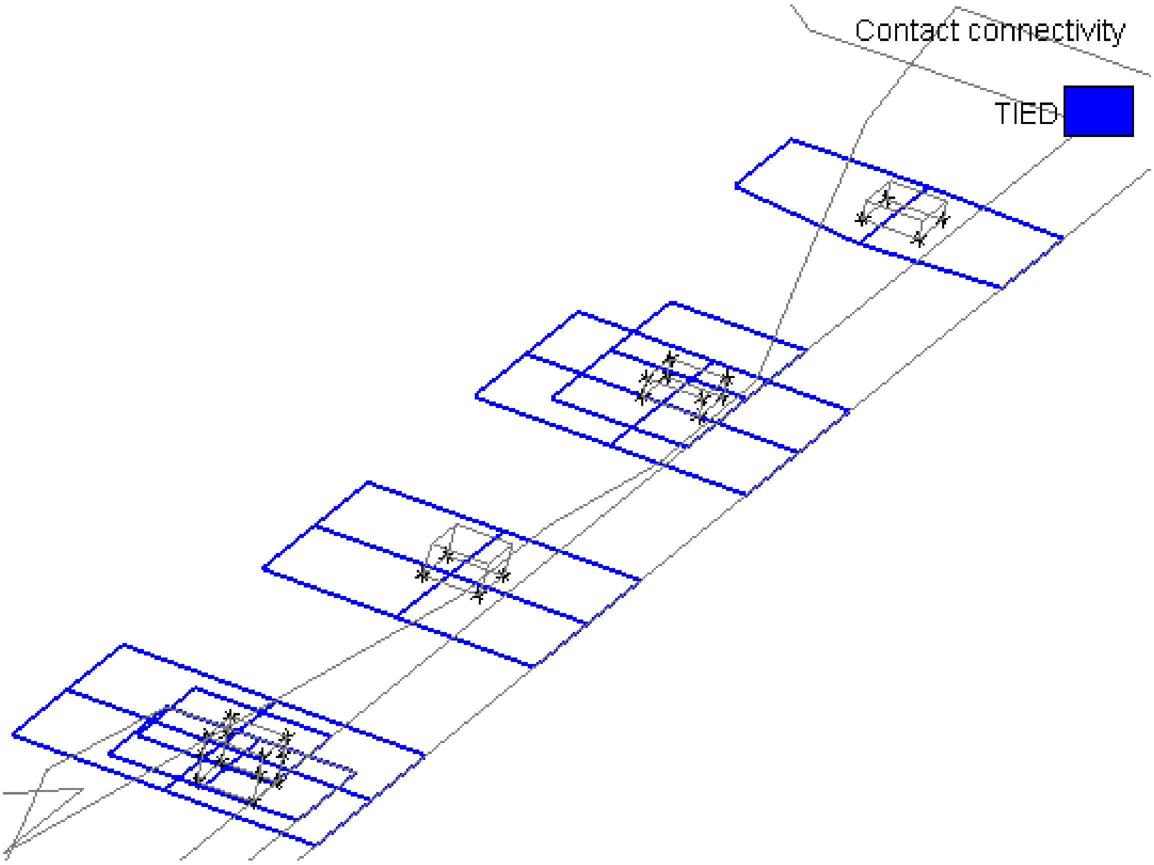
This is the status quo for Primer release 9.4, however it is likely that this setting - or at least the logic behind its default "automatic" option - will evolve as LS-DYNA develops to remove the ambiguity in the choice between duplicate shells.

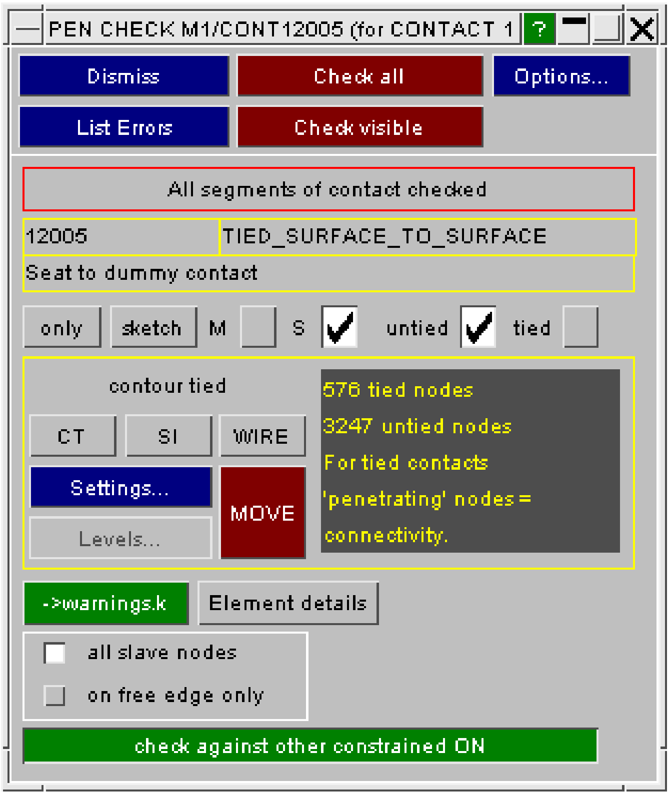
5.3.9 Checking a Tied Contact

The following contact types are treated as tied contacts in Primer:

```
*CONTACT_TIED_SURFACE_TO_SURFACE
*CONTACT_TIED_SURFACE_TO_SURFACE_FAILURE
*CONTACT_TIED_SHELL_EDGE_TO_SURFACE
*CONTACT_TIED_NODES_TO_SURFACE
*CONTACT_TIEBREAK_ SURFACE_TO_SURFACE
*CONTACT_TIEBREAK_ NODES_ONLY
*CONTACT_TIEBREAK_ NODES_TO_SURFACE
*CONTACT_SPOTWELD
*CONTACT_SPOTWELD_WITH_TORSION
```

For tied contacts, penetration of the slave node into the master segment means the node is tied.





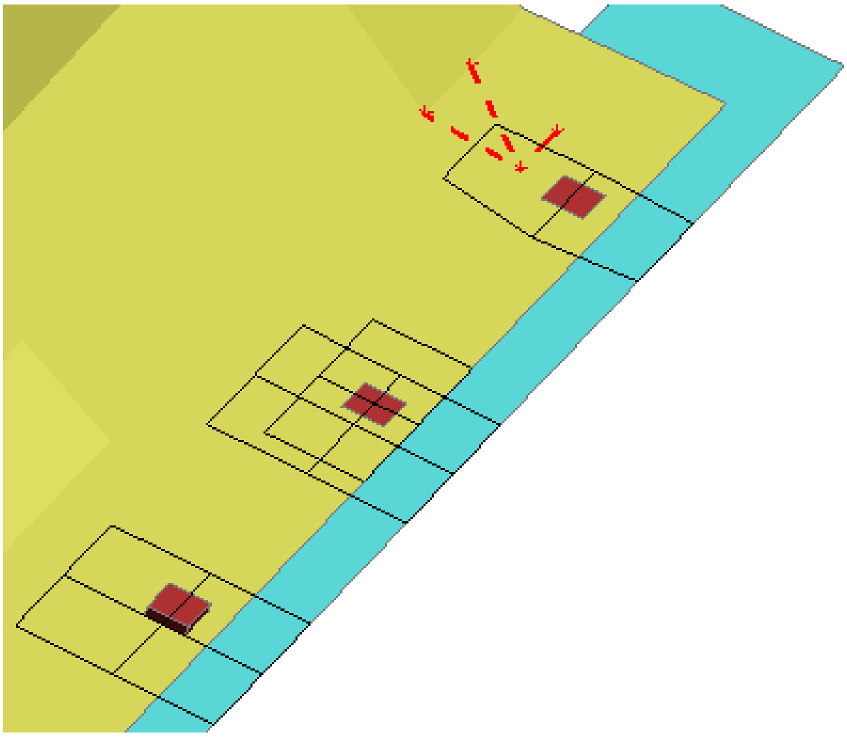
When contouring tied contacts, the tied node is sketched and blue is used to denote the segment to which it ties.

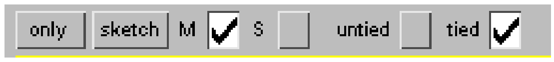
Primer will also the report the count of tied amd untied nodes. Some tied contacts may be expected to have zero untied nodes (e.g. those used for spotwelds) others may be using LS-Dyna's geometric tolerancing and intentionally contain untied nodes on slave side.

all slave node/free edge only option has no relevance for tied contact checking, it applies for MOVE option and only when slave nodes on shells

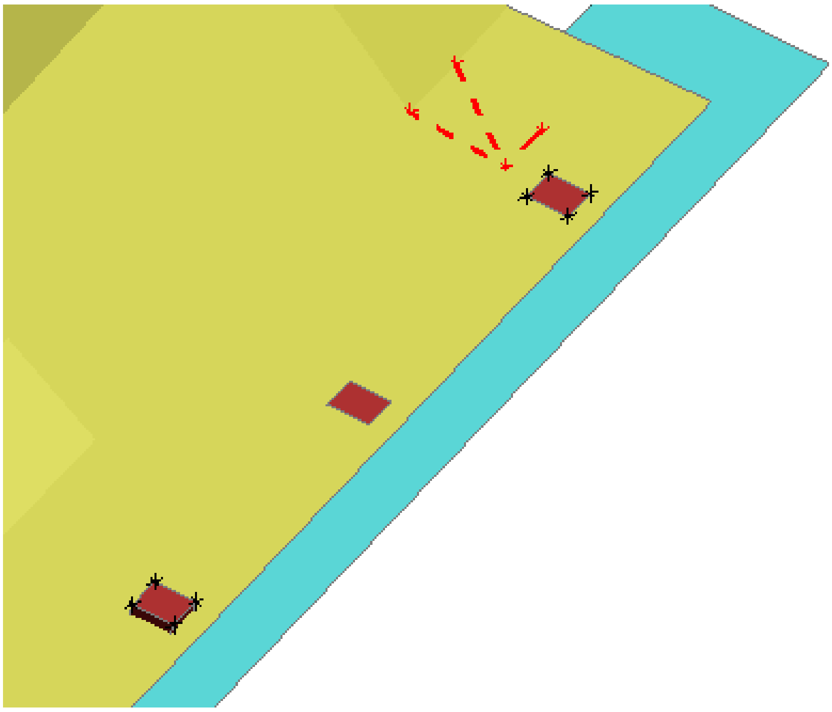
sketch & **only** - allow user to visualize what is **tied** and/or **untied** on master (**M**) and/or slave(**S**) side of the contact, according to which option is ticked. These settings do not affect the contour plot or node count.

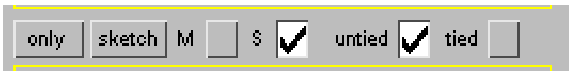
all slave nodes/on free edge only - this option only applies if the slave side consists of nodes on shells. If set to **on free edge only** the reported count of untied nodes and **sketch** & **only** functions will only consider slave nodes on shells on free edges.





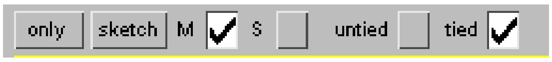
sketch applied to what is tied on the master side



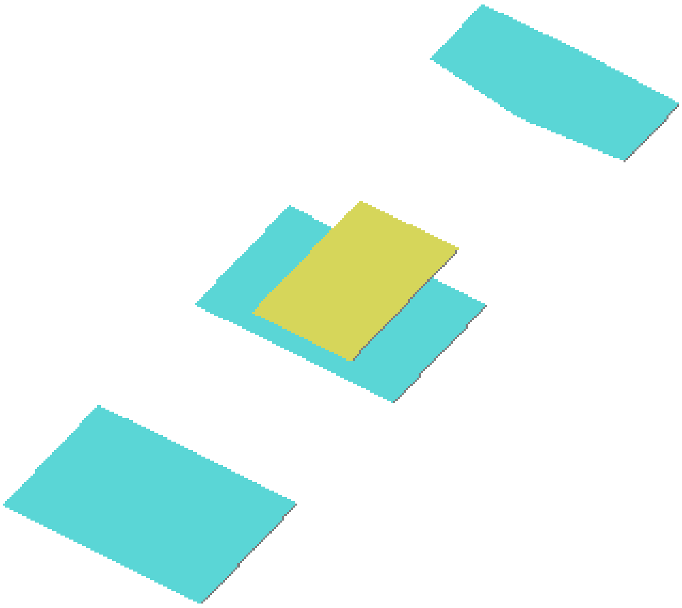


sketch applied to what is untied on the slave side

some nodes do not tie because this is a constrained contact and the nodal rigid body shown interfers with it, others because they are too far away from their segment



only applied to what is tied on the master side



->warnings.k - for tied contacts this function will write untied nodes to a node set, appropriately named, in include file *warnings.k.* Untied elements will also be written to a set.

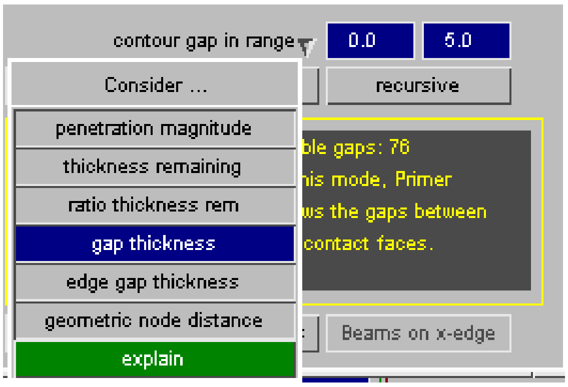
Check against other constrained - by default if this is a constrained (not penalty) contact and if other constrained contacts exist in the model, Primer will check for clashes with other constrained contacts. A slave node cannot be tied successfully to a segment, if another contact ties to this segment or a segment which shares a node with this one. You can turn this option off in CHECK > OPTIONS > CONTACT.

5.3.10 Notes on Contact Penetration Checking.

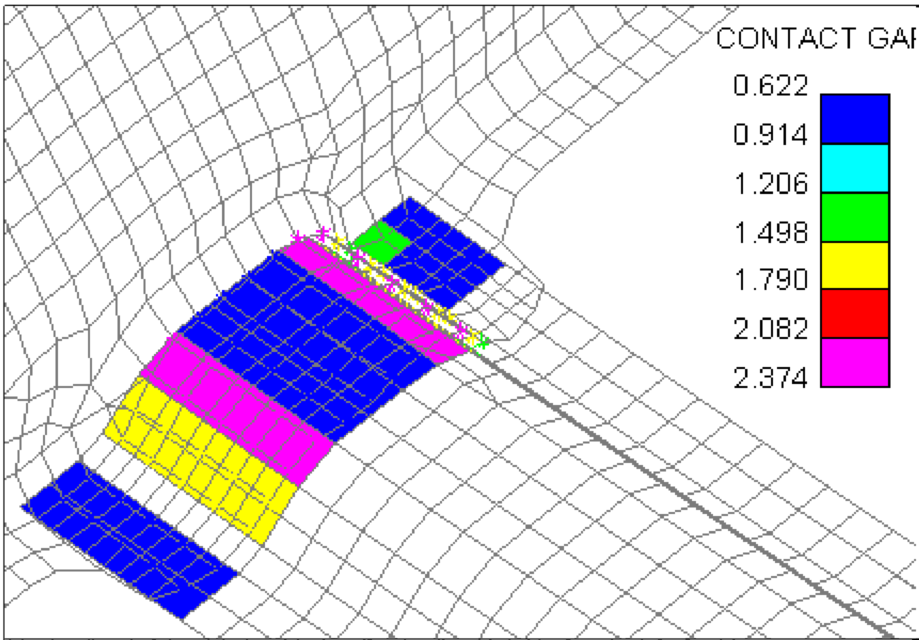
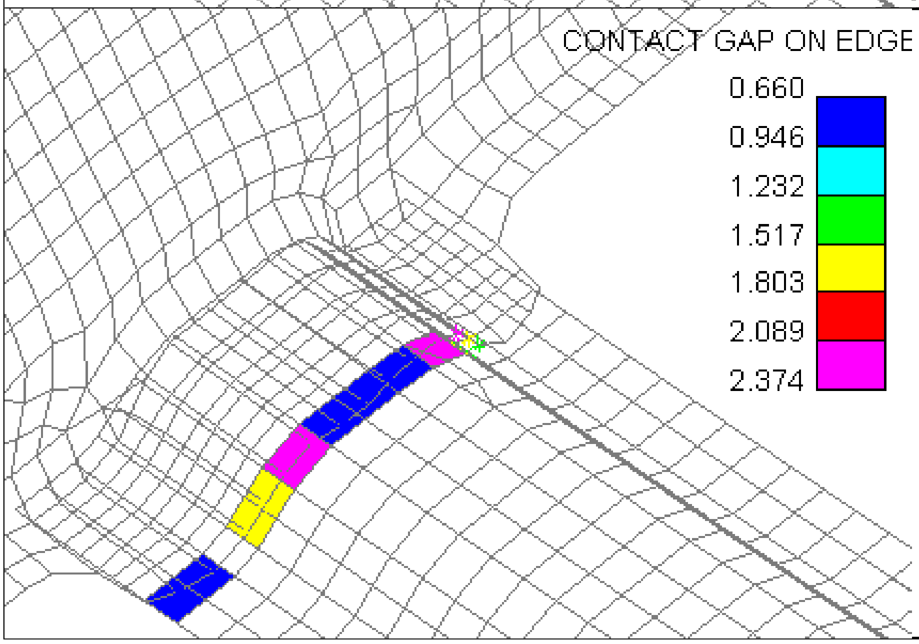
- The checking algorithms used in Primer aim to mimic those in LS-Dyna, but they are not identical. You must expect that the penetrations detected will differ slightly, although they should agree well in most cases. An exception is self-contact where LS-Dyna's initialization process of depenetrating nodes as it finds them will always yield appearance of less penetrating nodes (in otf file) than Primer's static geometric approach.
- When "old type" segment-based contacts (non-automatic surface to surface, nodes to surface and singe surface) are used, the default penetration depth "behind" a segment in LS-Dyna is "infinite" (= 1e20 or thereabouts). For efficiency in its bucket sort Primer limits the depth behind such a segment to be the longest distance from a node on it to an imaginary box containing the contact. Therefore "as_thick" plots, and reported penetration distances, will be limited to this value.
- Certain 2D geometries are not yet checked, these are:
    - "Edge to ..." contacts: Shell or segment edge contact is not considered
    - "Beam to ..." contacts: Contact along the length of a beam is not considered.

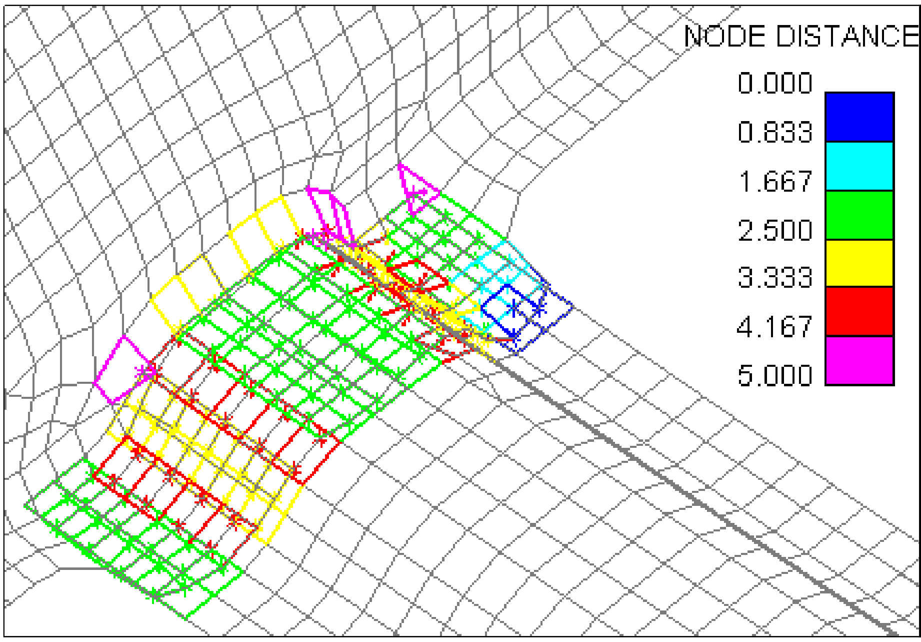
  However the (one way) penetration of nodes on edges and beams into segments is considered.

5.4 Contouring panel gaps for sliding contact



Penetration check module is now capable of contouring gaps in a user defined range between panels. Three modes are available. In gap thickness mode the **FIX** function becomes a **MOVE** function which removes gaps.

	gap thickness - will contour the gap between slave side nodes and master side shells for surface-surface contact. In single surface contact every node is treated as a slave node.
	edge gap thickness - as above but will only consider slave nodes on shells on a free edge



node distance - measures the distance of each slave node to mid-plane of its master segment and reports cases where the distance is less than the user specified value. This test is independent of contact thickness

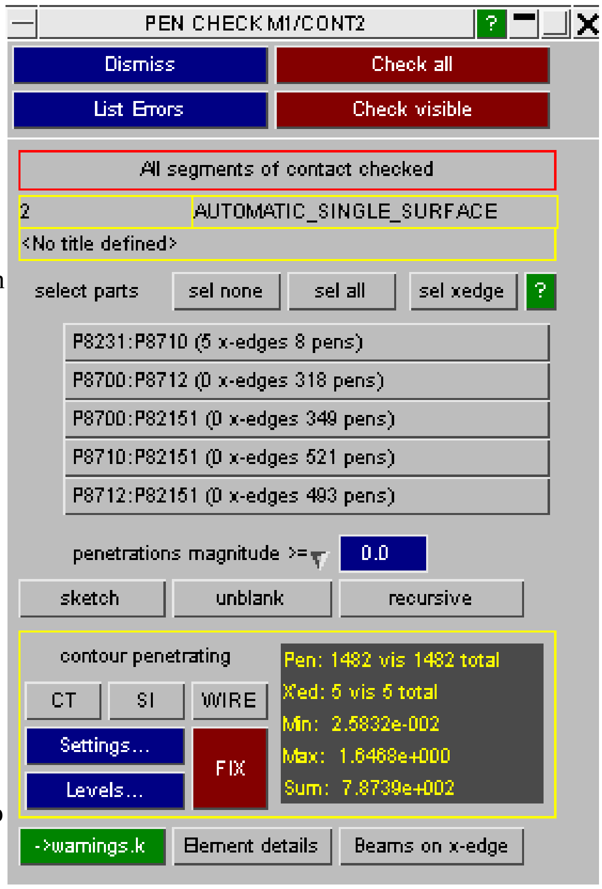
5.5 Contact Penetration Fixing

- Accessing the penetration fixing panel
- Correcting crossed edges
- Correcting penetrations

Primer can help fix initial penetrations and crossed edges in contact surfaces. This capability can be invoked via the penetration checker window accessed via:

CHECK (from Tools) > RULES
CONTACT > PEN_CHECK
CONTACT > CREATE/EDIT, PEN_CHECK

5.5.1 Penetration Fixing panel



The pen check panel shown will provide data on the number of crossed edges and penetrations in the model. For more information on penetration checking, see section 6.10 Contact Penetration Checking.

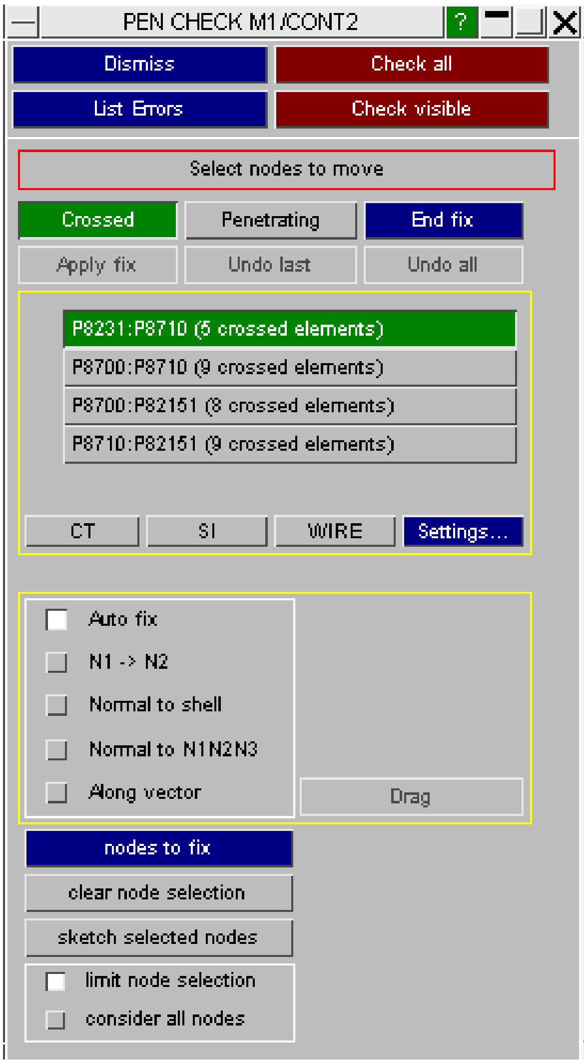
The penetration checker panel allows you to easily correct any crossed edges and initial penetrations in your model. This option can be accessed by pressing the fix button.

By default fixing is only applied to *visible entities* so you can use visibility control of the check panel to determine what is to be fixed. The crossed edge fixing panel then offers further visibility control As these need to be fixed one-on-one). If you pre-select *all the interacting parts or none*, all crossed edges will be up for consideration. Normally, it is recommended that you **remove all crossed edges** before attempting to remove initial penetrations.

Penetrations and Crossed edges can be removed using both automatic and manual methods. The Penetration checker can be used to check that all initial penetrations have been removed.

Penetration mode. Fixing will be applied using the *active check penetration mode*.The setting both limits to which nodes the fix is applied (e.g. ignore penetrations below a certain value) and controls the magnitude of the fix (e.g. setting **ratio thickness rem** < 0.7 will mean that *minimal depenetration* is performed to meet this target)

5.5.2 Correcting crossed edges



In the penetration fixing panel, select the CROSSED option. The panel will display all pre-selected crossed panels present in pairs as shown. Select the pair of crossed panels you wish to correct. Primer will highlight the 2 panels and show where the crossed edges exist and the fixing options will ungrey.

Primer allows you to correct crossed edges by moving selected nodes. **nodes to fix** will allow you to add nodes to the selection. By default the object menu will limit the selection to penetrating nodes, but you may switch this to **consider all nodes** which will enable you in manual modes to smooth the mesh.

The direction in which the selected nodes are moved can be specified by using one of the options

AUTO FIX

N1 -> N2

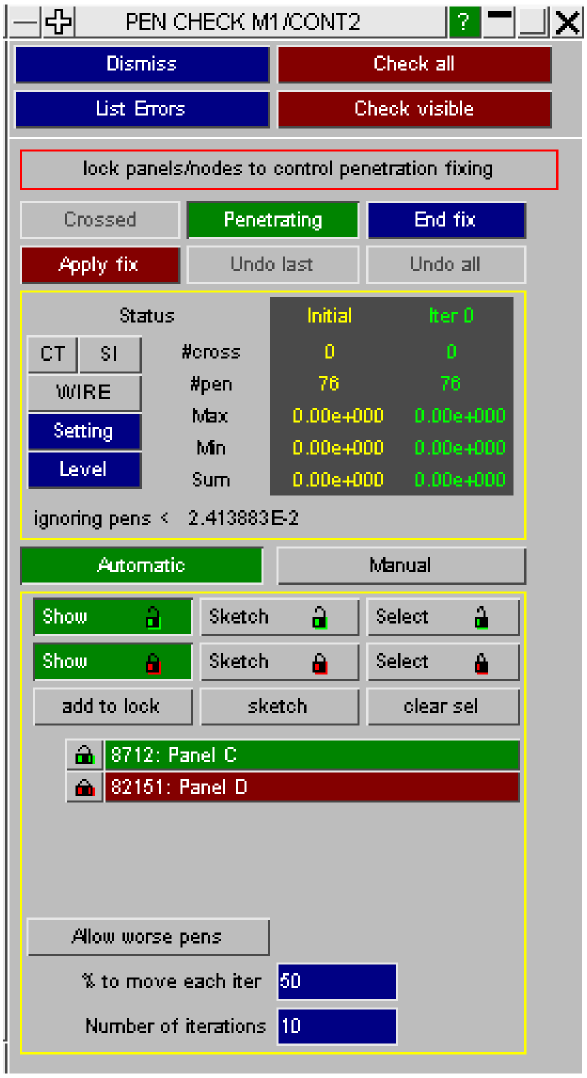
NORMAL TO SHELL

NORMAL TO N1N2N3

ALONG VECTOR

When the desired fixing method has been set, press the **Apply fix** or **End fix** (as appropriate) to lock the change in.

5.5.3 Correcting initial penetrations



In the penetration fixing panel, select the **PENETRATING** option. The panel will display information on the number of crossed edges, the number of initial penetrations which exceed the define penetration parameter, the maximum and minimum penetration and the sum of all penetrations. Primer offers both automatic methods and manual methods for fixing penetrations.

When the desired fixing method has been defined, press the **APPLY FIX** button.

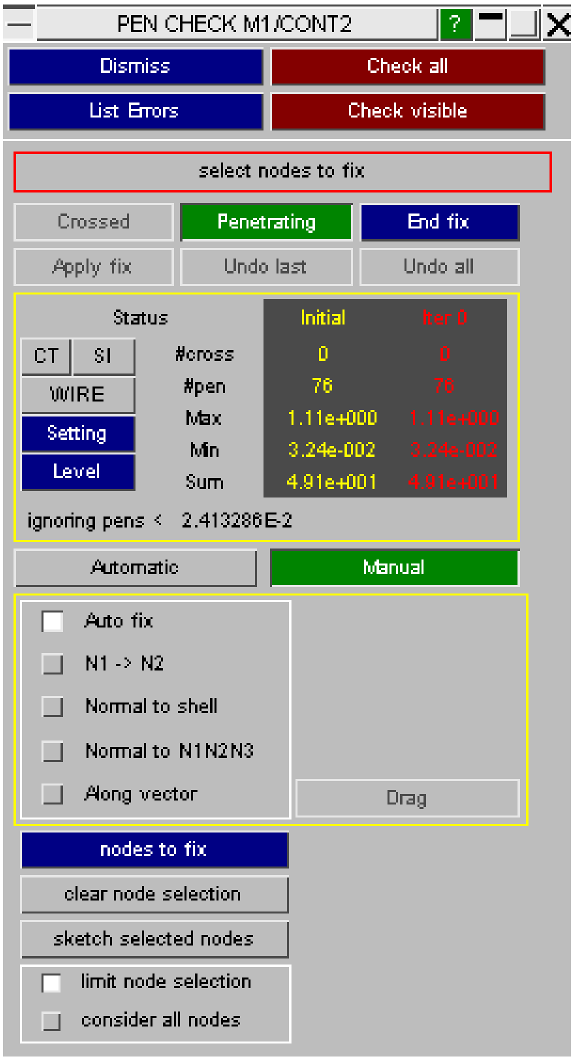
5.5.3.1 Automatic Fixing

This can be accessed by selecting the **AUTOMATIC FIXING** tab. Parts can be locked (indicated by a red, closed padlock) and unlocked (indicated by a green, open padlock) by clicking on the padlock tabs. Those parts with crossed edges will be automatically locked. Only unlocked parts will be moved in the fixing process. Only unblanked entities will be moved in the automatic fixing process. **add to lock** can be used to lock nodes additionally the the panel lock.

The Automatic Fixing method uses an iterative method. Nodes are moved only a percentage of the penetrating distance. To specify this percentage, enter the desired percentage in the % to move each iter box. The number of iterations to be carried out can be specified in the Number of iterations box. In some cases it may be necessary to allow worse penetrations initially in order to fix all penetrations in the model. If this could be the case, select the **Allow worse pens** tab.

Nodes will be fixed sufficiently to meet the current penetration criterion, this may be less than full depentration.

5.5.3.2 Manual Fixing



This option can be accessed by selecting the **MANUAL FIXING** tab.

Fixing is applied to selected nodes.

The direction in which the selected nodes are moved can be specified by using one of the options;

AUTO FIX

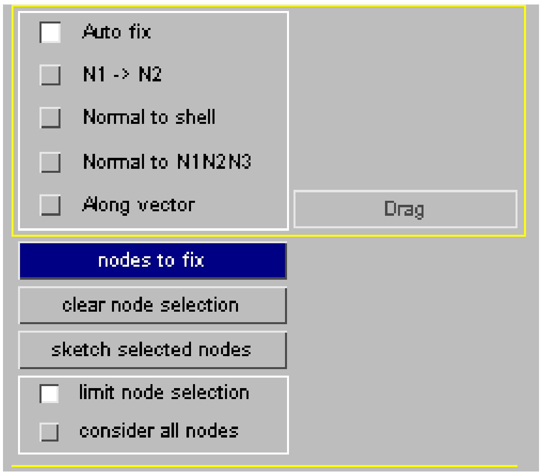
N1 -> N2

NORMAL TO SHELL

NORMAL TO N1N2N3

ALONG VECTOR

Selecting nodes to fix

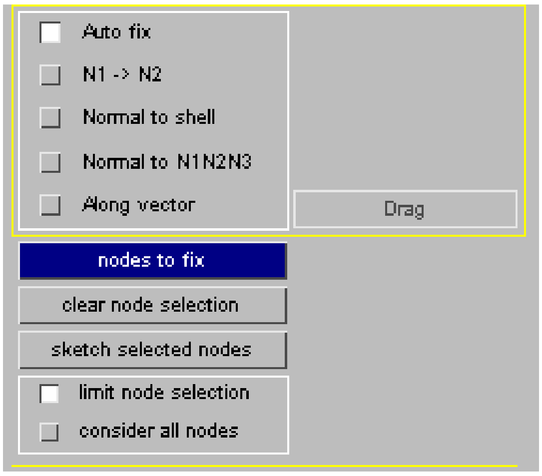


Primer allows you to correct initial penetrations by moving nodes selected by the **nodes to fix** function. By default the selection is limited to penetrating nodes, but you may select other nodes for manipulation to smooth the mesh changes by using **consider all nodes**.

The direction in which the nodes are to be moved can be specified by using the options; N1 -> N2, NORMAL TO SHELL, NORMAL TO N1N2N3, ALONG VECTOR, or alternatively the AUTO FIX option can be selected.

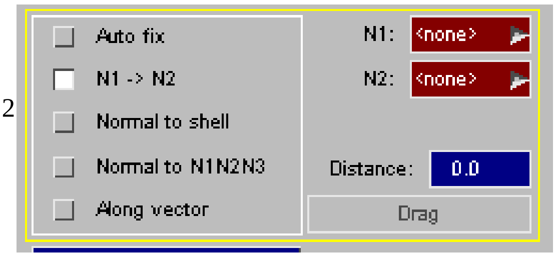
Auto fix

This option will move the selected node or nodes normal to the penetrating shell in order to remove the penetration.



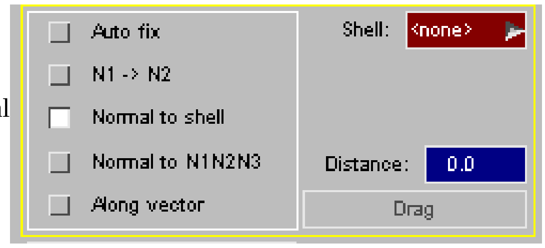
N1 -> N2

The selected node or nodes will be moved along a vector defined by 2 nodes. The nodes can be selected using any of the usual procedures. The extent to which the penetrating nodes are moved can be defined by distance, specified in the distance box, or by dragging the nodes, specified by selecting the **DRAG** button and dragging the nodes across the screen using the cursor.



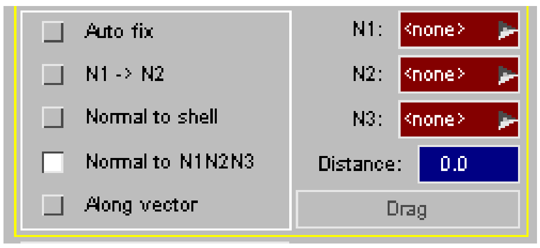
Normal to shell

Normal to shell allows you to move the selected node or nodes normal to a shell. The desired shell can be specified using any of the usual procedures. The extent to which the selected nodes are moved can be defined by distance, specified in the distance box, or by dragging the nodes, specified by selecting the **DRAG** button and dragging the nodes across the screen using the cursor.



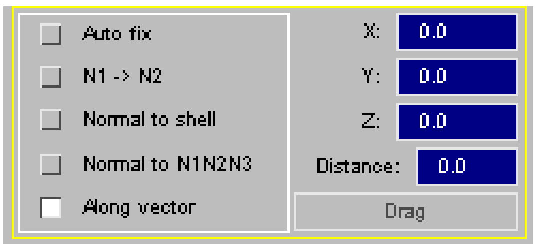
Normal to N1N2N3

Selected nodes will be moved normal to a plane defined by 3 nodes. These 3 nodes can be selected using any of the usual procedures. The extent to which the penetrating nodes are moved can be defined by distance, specified in the distance box, or by dragging the nodes, specified by selecting the **DRAG** button and dragging the nodes across the screen using the cursor.



Along Vector

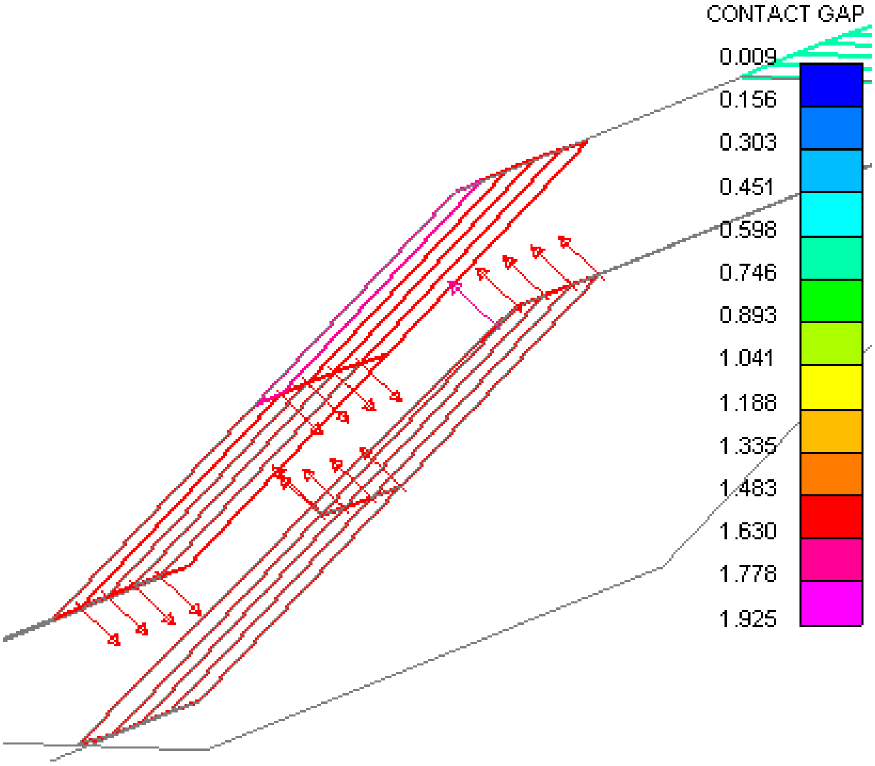
This option allows you to define a vector using co-ordinates along which the selected nodes are to be used. The extent to which the penetrating nodes are moved can be defined by distance, specified in the distance box, or by dragging the nodes, specified by selecting the **DRAG** button and dragging the nodes across the screen using the cursor.

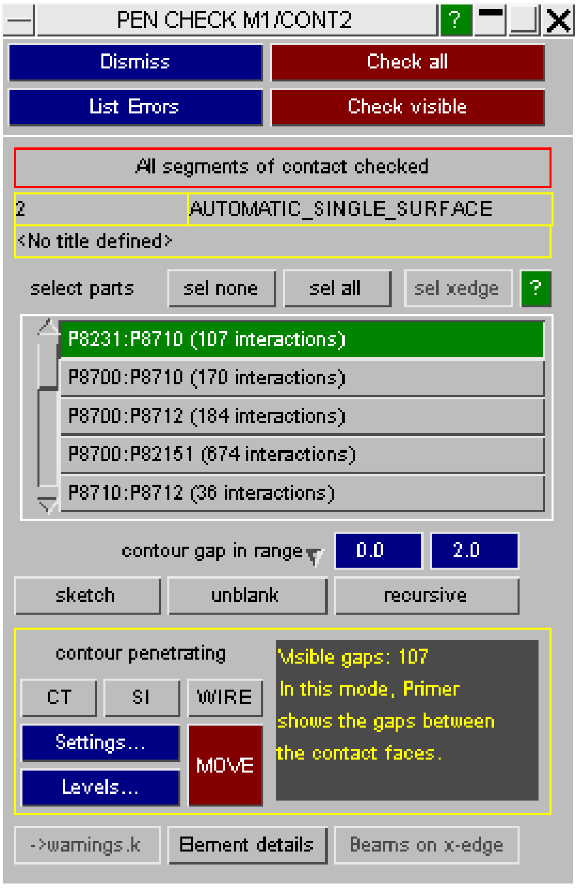


5.6 Contact gap fixing

Primer contours gaps in the user defined range.

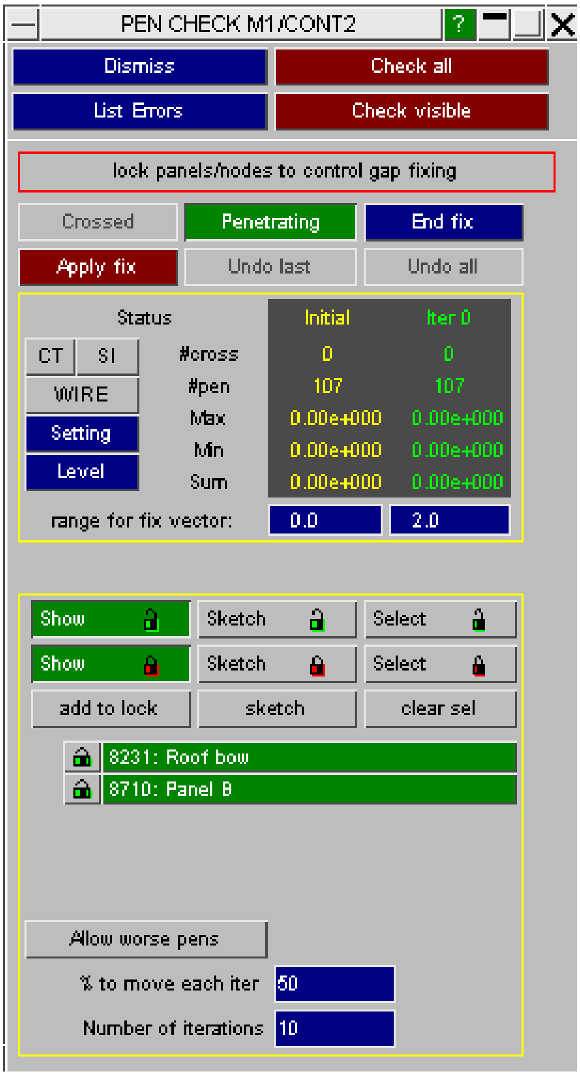
In gap mode the fixing function becomes **MOVE** - a function to remove gaps between visible segments.





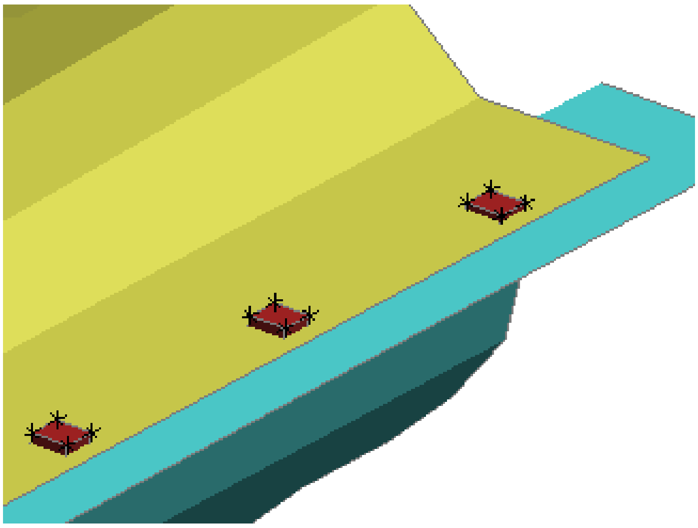
Gap fixing will observe panel and node locking in the same way as penetration fixing.

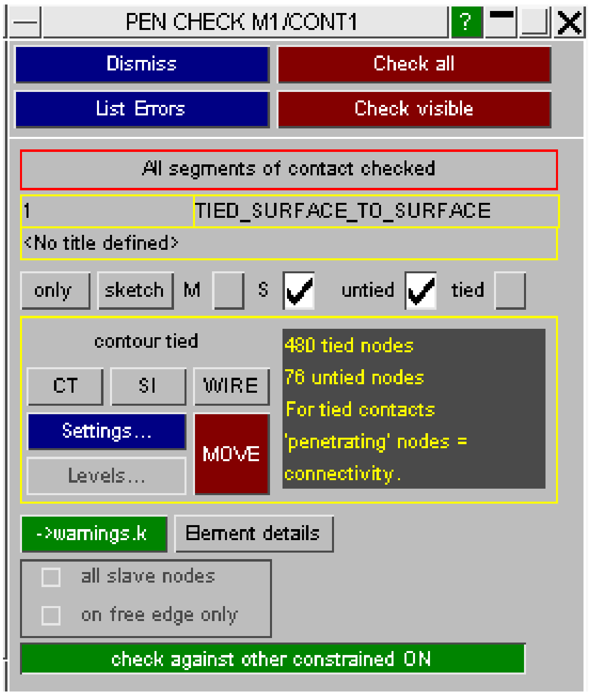
Any unlocked nodes which bear an approach vector in the CT plot will be moved along the vector iteratively, until the gaps are removed.



5.7 Tied contact fixing

When checking a tied contact the fix function becomes MOVE. This will move slave nodes to "best position".





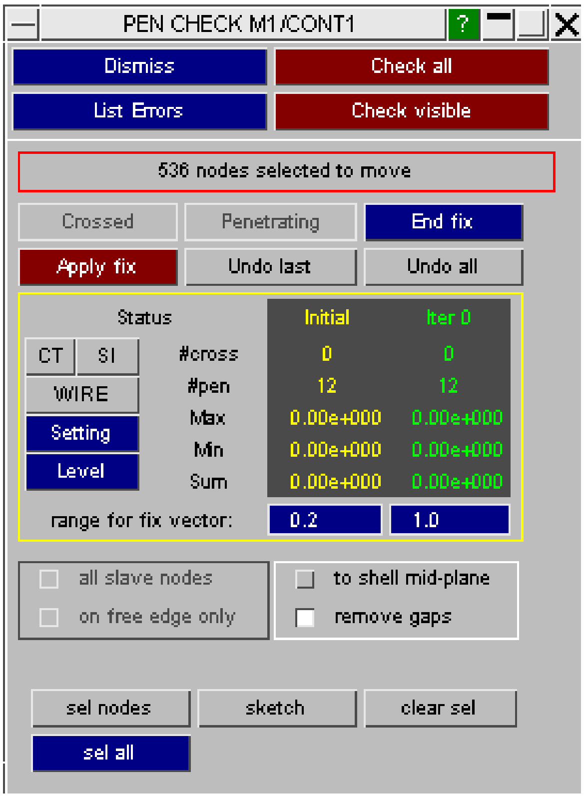
The range for fix vector needs to be set. CT plot will now show all the moveable nodes with a vector. Nodes to move must then be selected using **sel nodes** or **sel all** and **Apply fix** will become active.

to shell mid-plane - will configure the vector to move the node to the mid-plane of the shell to which it ties. This is recommended.

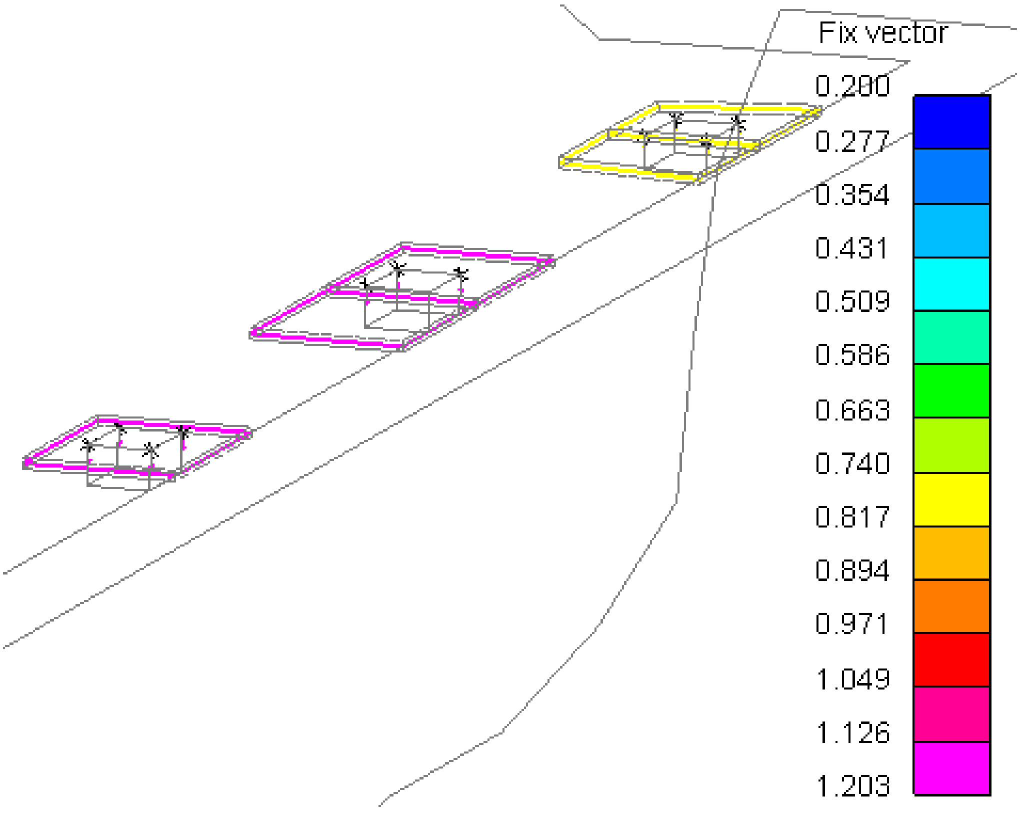
remove gaps - will configure the vector to move the node so there is no remaining gap. For offset contacts this fix may be more appropriate.

Either of the above methods will ensure that the node is in a position to tie geometrically (a node may not tie due to other reasons of course)

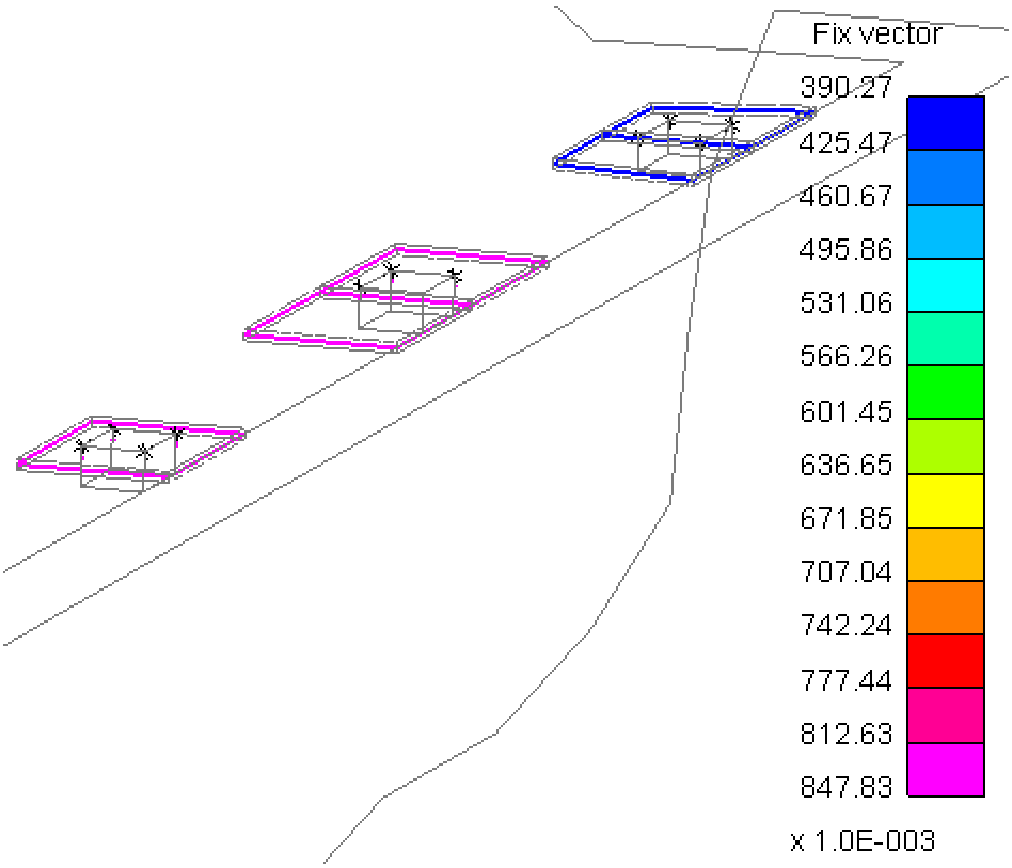
LS-Dyna has rules for determining whether or not a node will tie which depend on mesh size as well as distance off segment (see the manual entry under *CONTACT). It may be that some nodes with vectors drawn on them *are actually tied* and, therefore do not necessarily need to be moved.



Fixing to the shell mid-plane (shell thickness is shown)

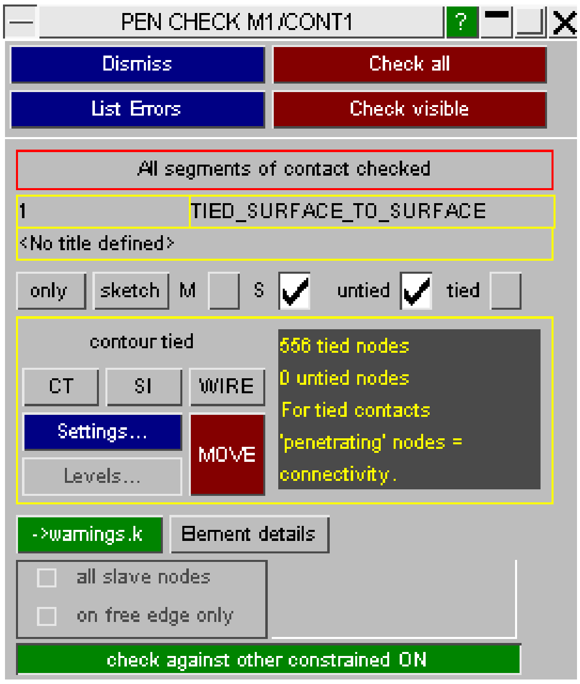


Fixing by removing the gap (shell thickness is shown)



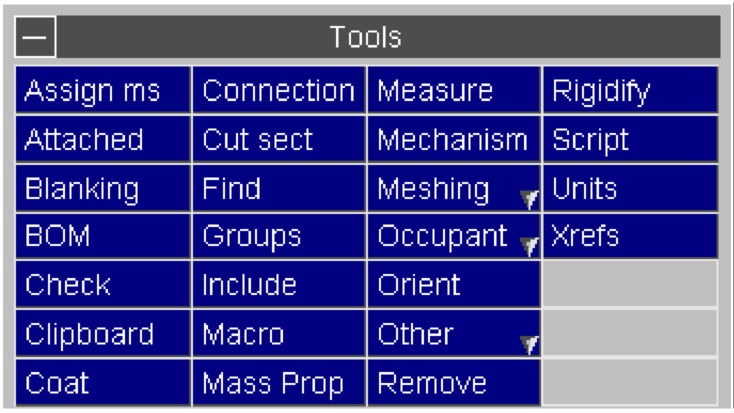
End fix will return you to the tied contact check panel, where you can re-check the count of tied nodes

all slave nodes/**on free edge only** - the option only applies if slave nodes are on shell elements. If set to **on free edge only** it will limit the MOVE option to apply only to nodes on free edges of shells



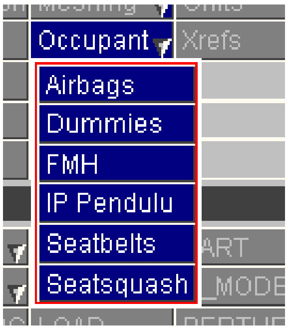
6 Tools

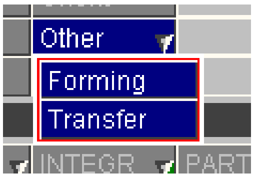
The main Tools panel provides top-level access to the following functions.



The following operations have been grouped under pull-down menus by category:

Meshing | **Occupant** related | **Other** functions





6.0 TOOLS

Operations and selecting entities

Select selection below for more details

Assign mass	Coat part	Macro	Orient	X-references
Attached	Connection	Mass Prop	*Other*	
Blanking	Cut Sect	Measure	Remove	
Bill of materials	Find	Mechanism	Rigidify	
Check	Groups	*Meshing*	Script	
Clipboard	Include	*Occupant*	Units	

Pull-down menu options

Meshing		*Occupant*		Other
Mesh		Airbags		Forming
Split		Dummies		Transfer
		FMH		
		IP Pendulum		
		Seatbelts		
		Seat squash		

6.0.1 Operational Hierarchy.

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

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

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

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

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

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

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

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

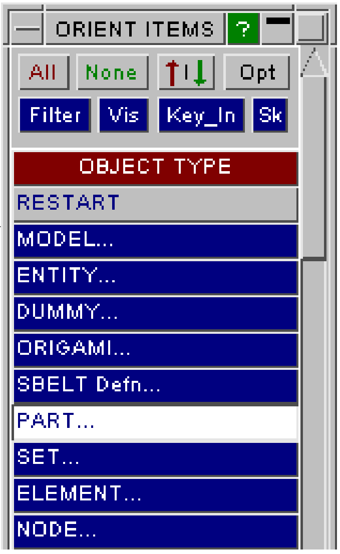
(The interaction of hierarchies and deletion is explained further under section 6.25**REMOVE**.

6.0.2 Selecting Entities for Operations.

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

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

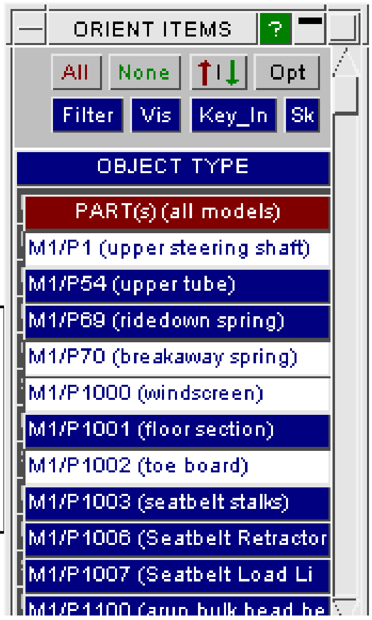
Primary selection of object type



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

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

Selection of objects from the menu list



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

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

You can also use

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

Of the other buttons at the top of this panel:

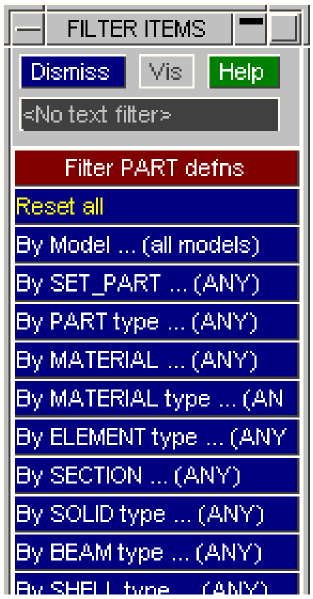
Opt(ions)	Further options (refresh, clipboard, blanking)
Filter	Applies "filtering" to what is shown
Vis	Maps a panel showing further "visible" picking options
Key_In	Maps a panel allowing you to key in label ranges directly
Sk(etch)	Sketches what is currently selected.

"Hover over": showing what will be selected.

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

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

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

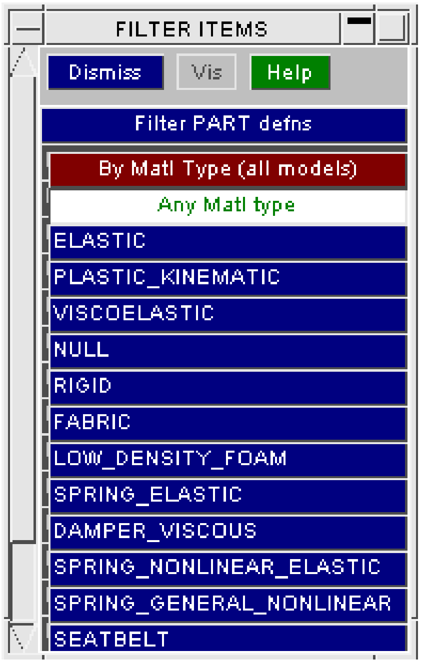


Where the list of objects is short this method of selection presents no problems. But in some cases the list may be hundreds or even thousands of items long, and a method of cutting down what is displayed is required. This is provided by the **FILTER** button at the top of this box.

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

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

Example of setting a **Filter**

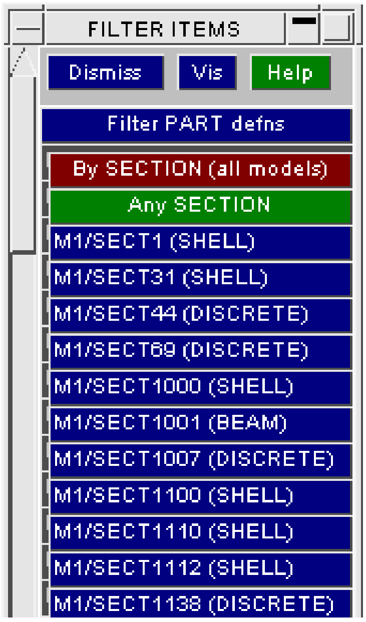


Here the **MATL TYPE ...** option for PARTS has been selected.

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

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

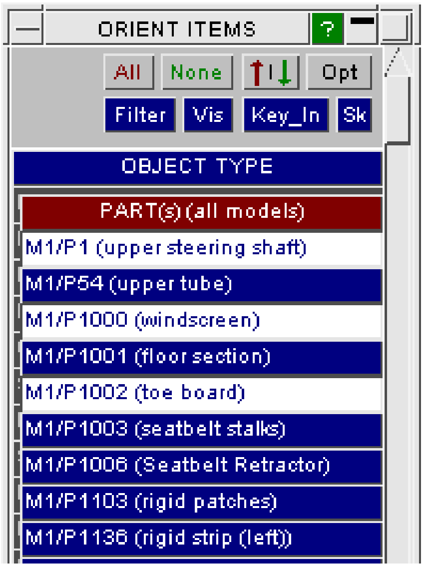
Another FILTER example, this time a pickable one.



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

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

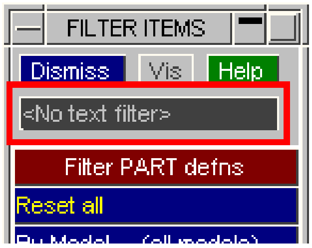
The influence of selecting a FILTER option



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

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

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



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

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

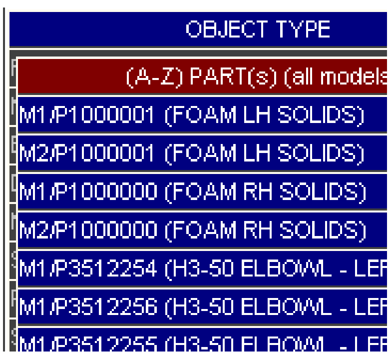
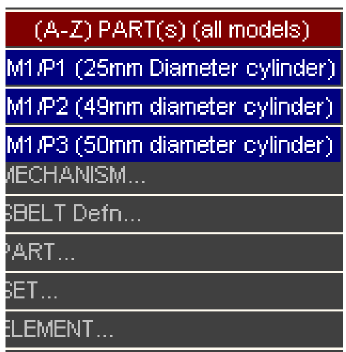
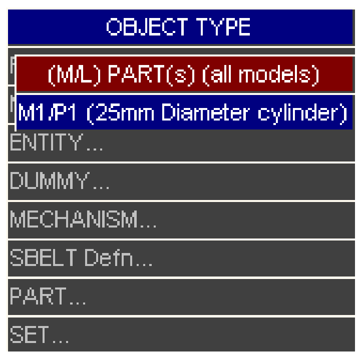
Text matching is not case sensitive, and leading and trailing white space characters are ignored. However embedded white space in a string is considered when pattern matching. In addition filtering supports the following "wildcard" characters:

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

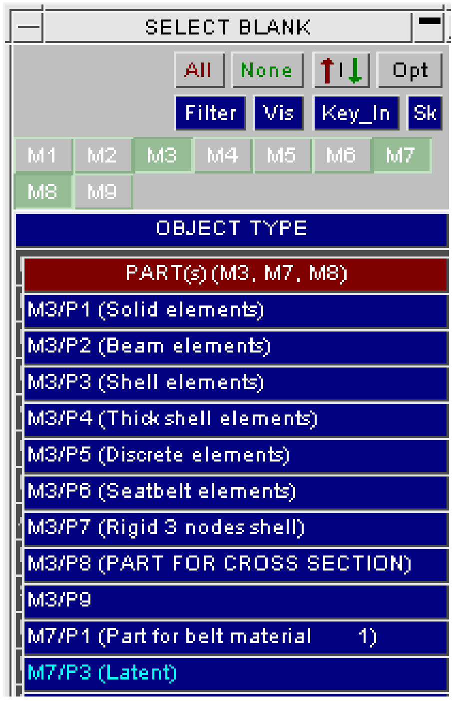
These may be used any number of times in a string, for example "**`*quick*fox*`**" will match the string "**`The quick brown fox jumped`**".

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

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

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

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

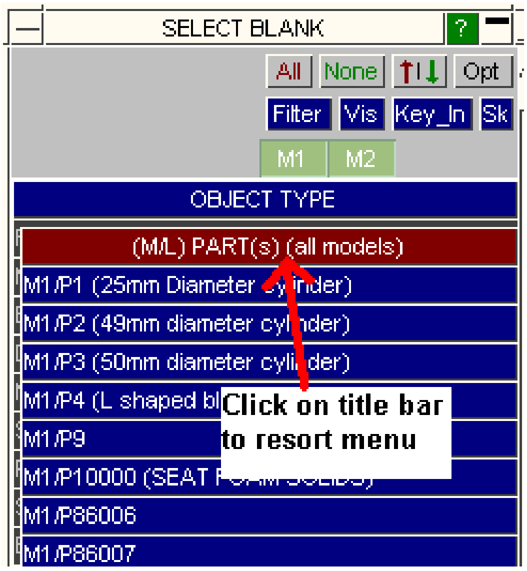


If your database contains more than one model then ***Mnn*** "tabs" will automatically be shown at the top of all menus where selection across multiple models would be legal: blanking in this example.

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

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

Sorting object menu contents



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

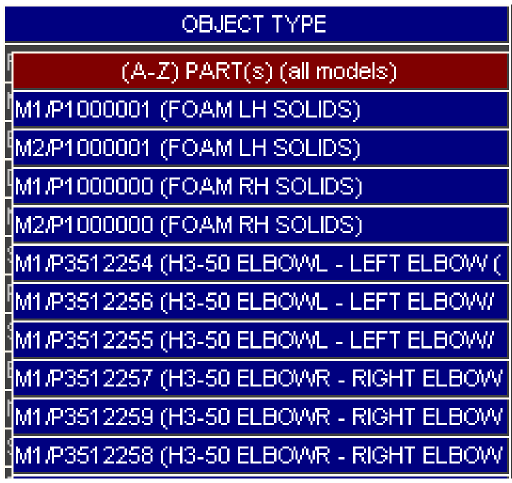
- All items in Model #1, sorted into ascending order by item label
- All items in Model #2, sorted by ditto

  etc

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

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

Further clicks will repeat the cycle above.



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

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

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

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

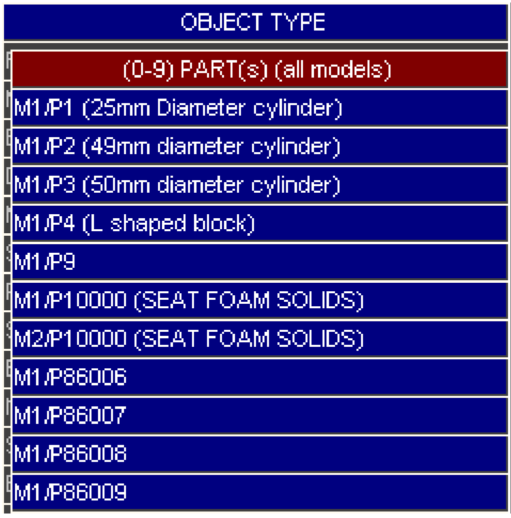
In this example two further clicks have given **0-9** sorting

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

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

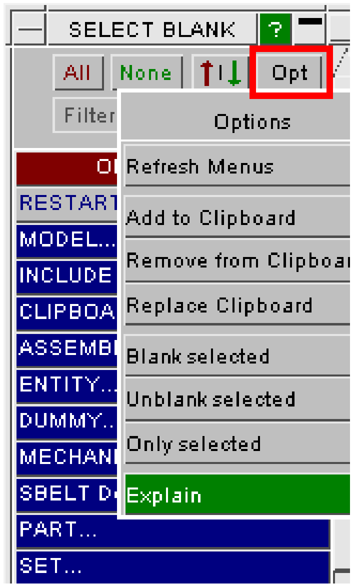
As before the top row of the menu shows the current sort method, now:

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



The **Options** popup menu

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



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

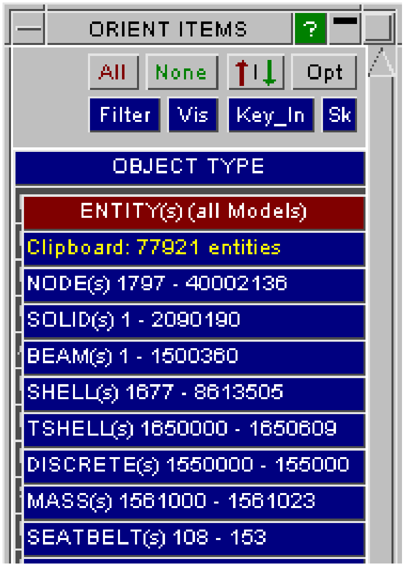
The clipboard may be used in a range of different ways in PRIMER, see section 6.8 **CLIPBOARD**, but in the context of object menus it may be used to save and reuse the current selection.

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

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

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

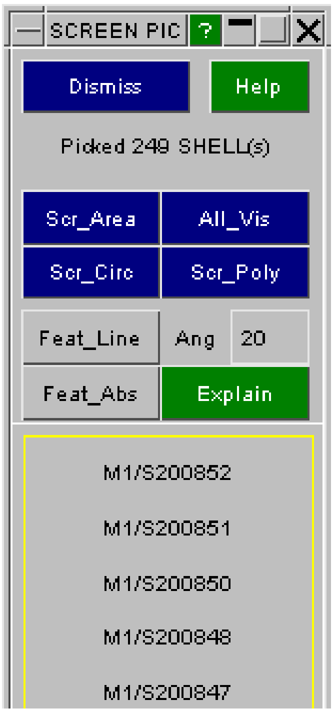
Selecting this row is a "one click" way of selecting all items on the clipboard which match the current type(s). This can mean multiple types, as in the example here using **`ENTITY`**, where the clipboard contains a mixture of entity types.



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

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

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



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

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

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

The mouse button during picking used is significant:

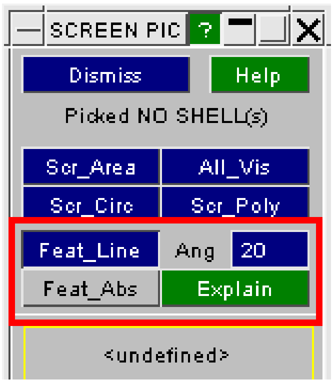
- LEFT mouse button selects
- MIDDLE button rejects the most recent selection
- RIGHT mouse button deselects (the picked items are removed from the list of currently selected items)

Mouse button usage is described in more detail below.

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

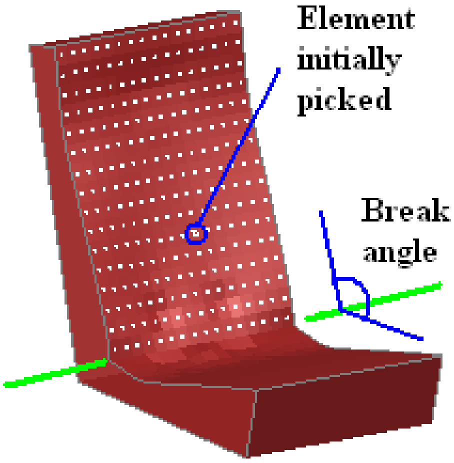
"Scalar" picking of single items	Just click on the approximate centre of the item to select it.
"Rectangular Area" picking of a range of items	Click and drag out a rectangular area. Everything within the area is selected.
Within the **Vis** panel there are the following further screen-picking options: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

Feature Line picking



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

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



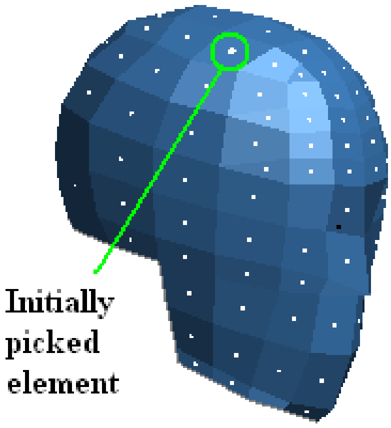
In this example using a crudely meshed seat, made from a single Part, the user has selected a single Shell element in the middle of the backrest with the feature angle set to 20 degrees.

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

The following rules apply to feature line picking:

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

Feat_Abs : Optional "absolute" feature angle



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

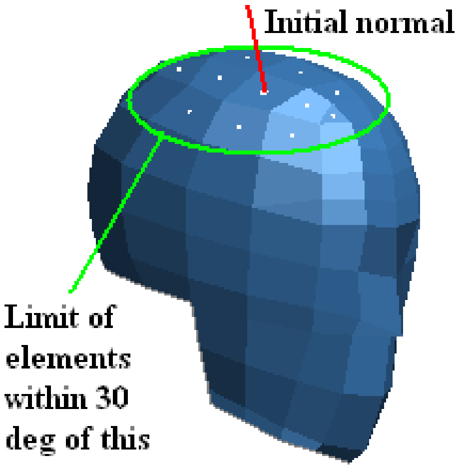
- the angular difference between the outward normal of the original node/element and this one must also not exceed the feature angle.

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

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

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

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



Rules applying to screen-picking

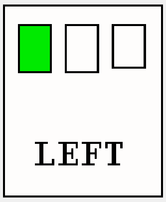
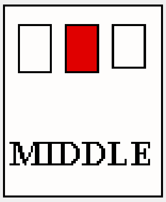
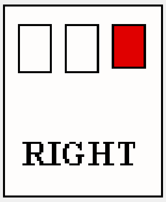
Screen-picked entries (by any method) go into the cursor list, of which the 10 most recent entries are shown in this box.

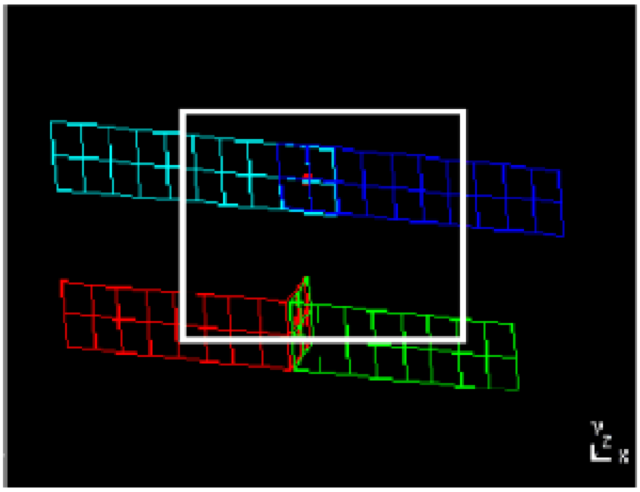
Note that:

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

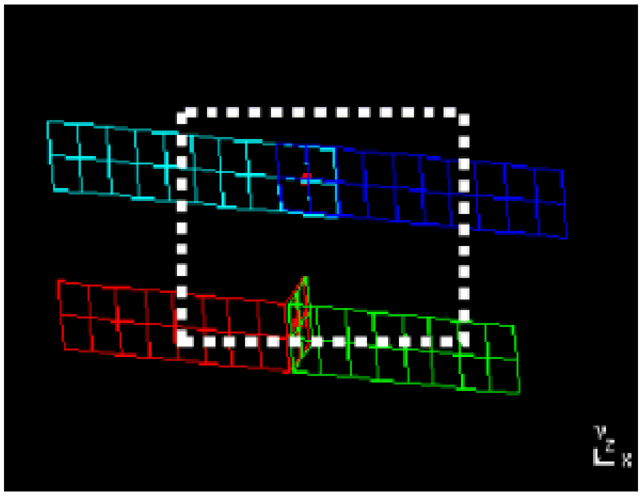
Rejecting items that have been screen-picked.

Picked items can be rejected (deselected) in exactly the same way that they were selected by using the middle and right mouse buttons:

Mouse Button	Function
	***Selects*** items: by singler pick, rectangular area, circular area or by arbitrary polygon as described above. A thin, solid white line is used to define areas and polygons.
	***Rejects*** the most recent selection: "last in, first out". The picking stack remembers all picks in the current operation, and repeated middle mouse clicks will back-track up it until it is empty. Area (of any type) picks are rejected en-bloc, ie items selected within a single area pick are also rejected via a single middle mouse click.
	***Rejects***: • What is explicitly selected (scalar pick) • All items in the area (multiple pick) A thick, broken white line is used to define areas and polygons.



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



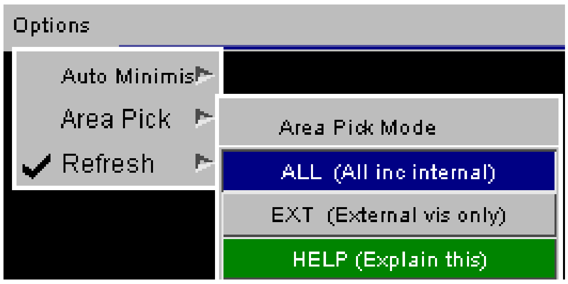
Right mouse button ***Rejects***:
Broken thick borders for areas and polygons.

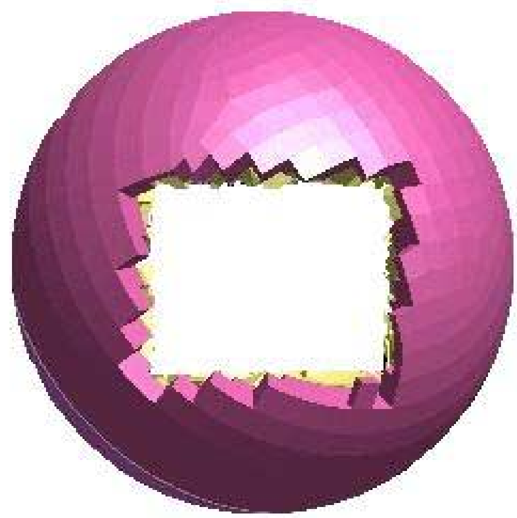
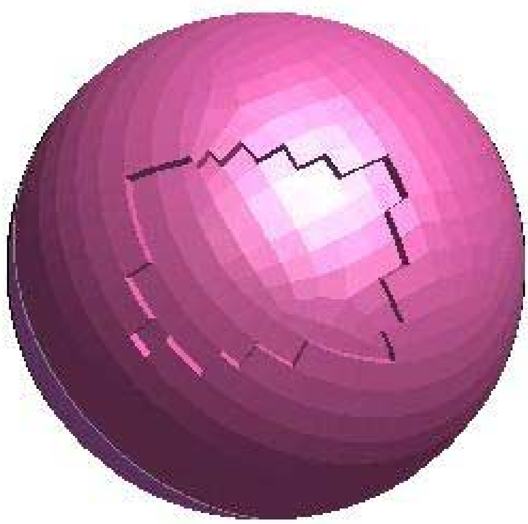
Area_Pick: What is "visible" when area or polygon picking

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

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

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



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

Using KEY_IN to type in selections

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

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

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

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

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

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

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

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

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

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

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

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

Selections are *not* cumulative across different item types

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

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

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

Selections persist following most operations (except BLANK and DELETE)

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

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

Exceptions are:

BLANKING once operated on the items chosen are deselected.

DELETE they will no longer exist!

To delete all current selections and start again

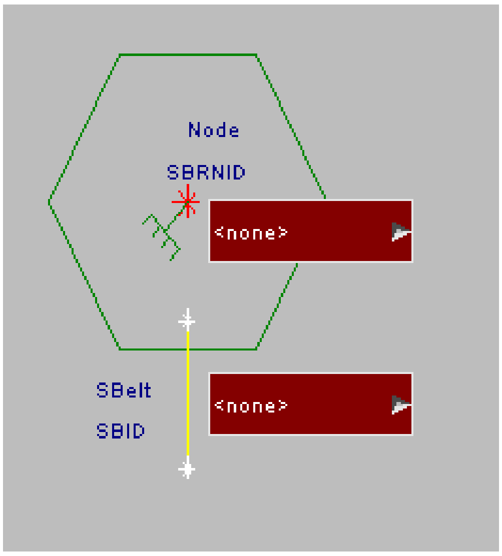
Using the **RESTART** row at the top of the object menu has the effect of canceling all current selections, unsetting the current object category, and resetting the selection process to its initial state.

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

6.0.3 Other selection methods

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

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



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

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

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

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

6.1 AIRBAGS

6.1.1 Folding Airbags

The airbag folder is designed to produce folded meshes from ones that are initially flat. Additionally, some facilities are provided to deal with 3-D initial configurations

The airbag folder can also generate an airbag mesh from scratch for some pre-defined geometries. At present this is limited to a star folded or a circular folded airbag. In future releases this functionality will be further enhanced.

During folding the airbag can be checked for distorted elements and initial penetrations. Once folded the airbag it can be positioned.

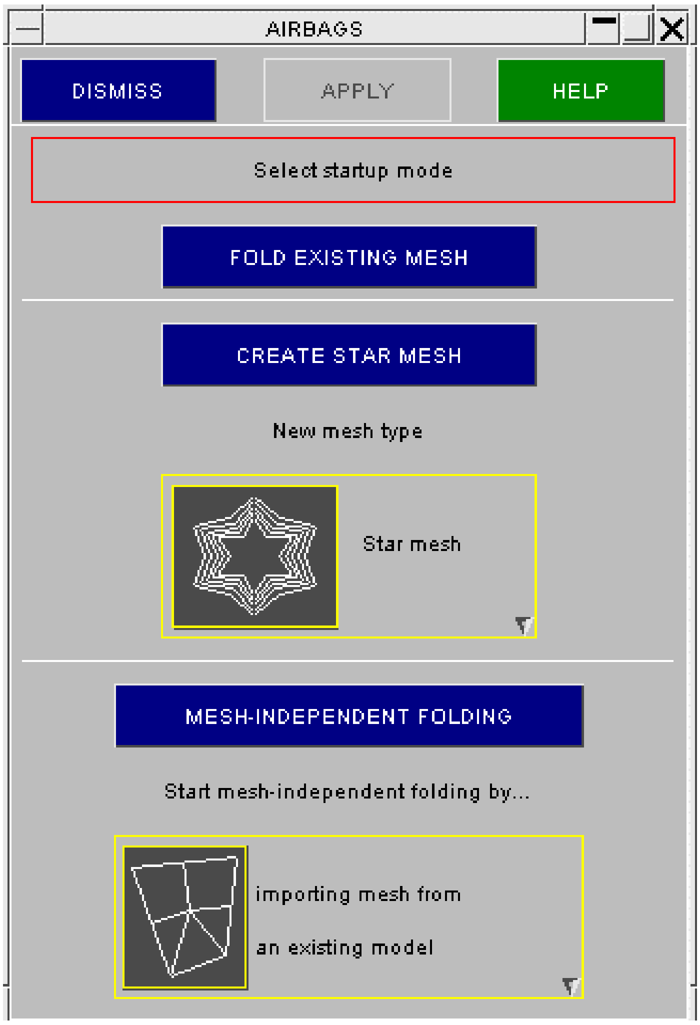
If you are starting from an existing mesh then each airbag must be a subset of only one model. Airbags usually consist of shell elements only and other element types should be avoided.

The airbag start up screen allows you to choose one of 3 possible modes.

1) Selecting **FOLD EXISTING MESH** will start the airbag folding process with an existing mesh.

2) **CREATE STAR MESH** can be used to create a new star or circular fold. To select whether to create a star fold or circular fold use the new mesh type popup box.

3) **MESH-INDEPENDENT FOLDING** allows you to create a circular airbag or import a mesh that is suitable for mesh independent folding.



Definitions: ORIGAMIs, FOLDs and ORIENTs

This section defines some of the terms which are used in the airbag folder.

The use of the term "Airbag", has the potential to cause some confusion. The actual **`*AIRBAG`** card in LS-DYNA consists of the surface of an airbag and the physical properties (gas thermodynamics) of the inflator. The tethers which might be included in a bag, for example, do not form part of the free surface of the airbag control volume, and thus they are not included in an LS-DYNA "airbag". The tethers, however, must be folded along with the remainder of the airbag, so they may need to be included in a geometrical definition.

Thus, to distinguish between the LS-DYNA **`*AIRBAG`** and the airbag as described here, the term ORIGAMI is used in

PRIMER. In fact, an ORIGAMI could potentially involve things which are completely unrelated to airbags: it is the umbrella definition containing everything required to define the geometrical extent of an airbag, and its associated folding operations:

- The origami label (which must be unique within a model) and title;
- A list of elements and nodes (as sets) which comprise the bag;
- A list of folds;
- A local coordinate system.
- A list of orientations

A **FOLD** is a generalised term for the many fold types available; eg Rolling a bag up is described as a "FOLD", as is a "Tuck" or a "Scrunch". Each FOLD definition contains:

- The fold number and type (thin, thick, roll, tuck, ...)
- An optional coordinate system;
- Geometrical data (location, direction, angle, thickness, ...)
- Optional subsets of nodes and elements for special cases.

An **ORIENT** is a transformation which is applied to the folded airbag to position it in the model. The different types available are translation, rotation and scaling. Each ORIENT contains:

- The orient number and type (translate, rotate or scale)
- Geometrical data (location, distance, angle ...)
- Optional nodes for special cases.

An ORIGAMI definition may contain any number of folds and orients, and a model may contain any number of ORIGAMIS. Elements and nodes may be referenced in more than one ORIGAMI, but this would not normally be sensible as the different fold operations might conflict: remember that a node can only have one current coordinate!

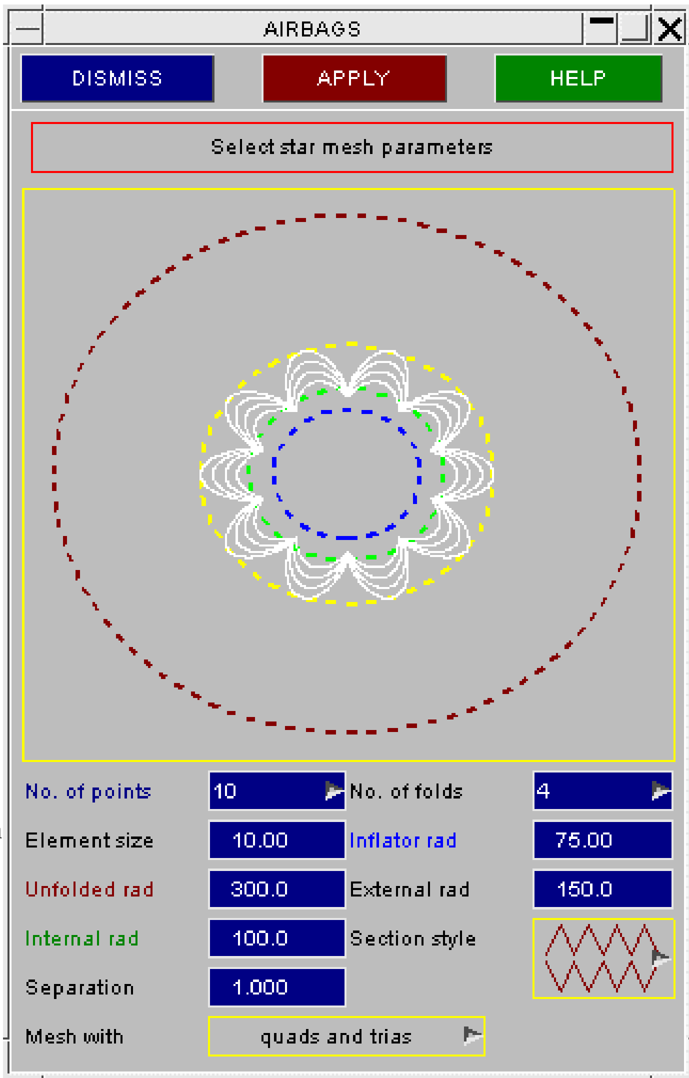
Creating and folding a new airbag from scratch

At present the following types of airbag mesh can be created:

A circular folded airbag. The airbag is compressed radially. To enable this the excess fabric forms folds.

A 'star' folded airbag. This is formed identically to the circular bag but an extra operation is performed. The airbag is pushed inwards at various points forming a star shape.

The star fold panel is shown in the figure to the right. This panel allows you to define the parameters which will be used to generate the mesh for the airbag. The parameters that can be changed are:



No. of points The number of points on the star fold.

No. of folds The number of out of plane folds which are done.

Element size The optimum size of elements which will be used when generating the airbag mesh.

Inflator rad The radius of the inflator at the centre of the airbag. This area will not be folded in any way. The inflator area on the bottom surface of the airbag will also be put into a different part. This allows the inflator part to be made rigid if necessary.

Unfolded rad The external radius of the airbag when unfolded flat.

External rad The maximum radius of each point on the star. i.e. the radius at the point tip.

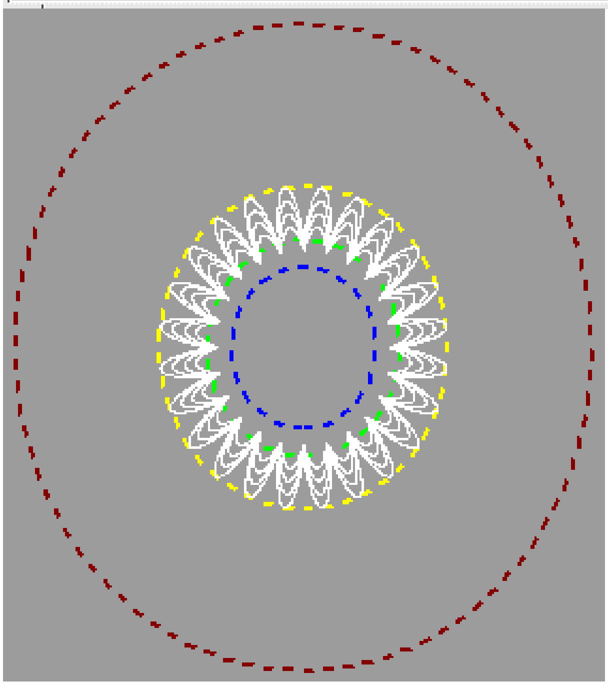
Internal rad The minimum radius of each point on the star.

Section style The cross section style of the folds used. The popup can be used to select two different styles.

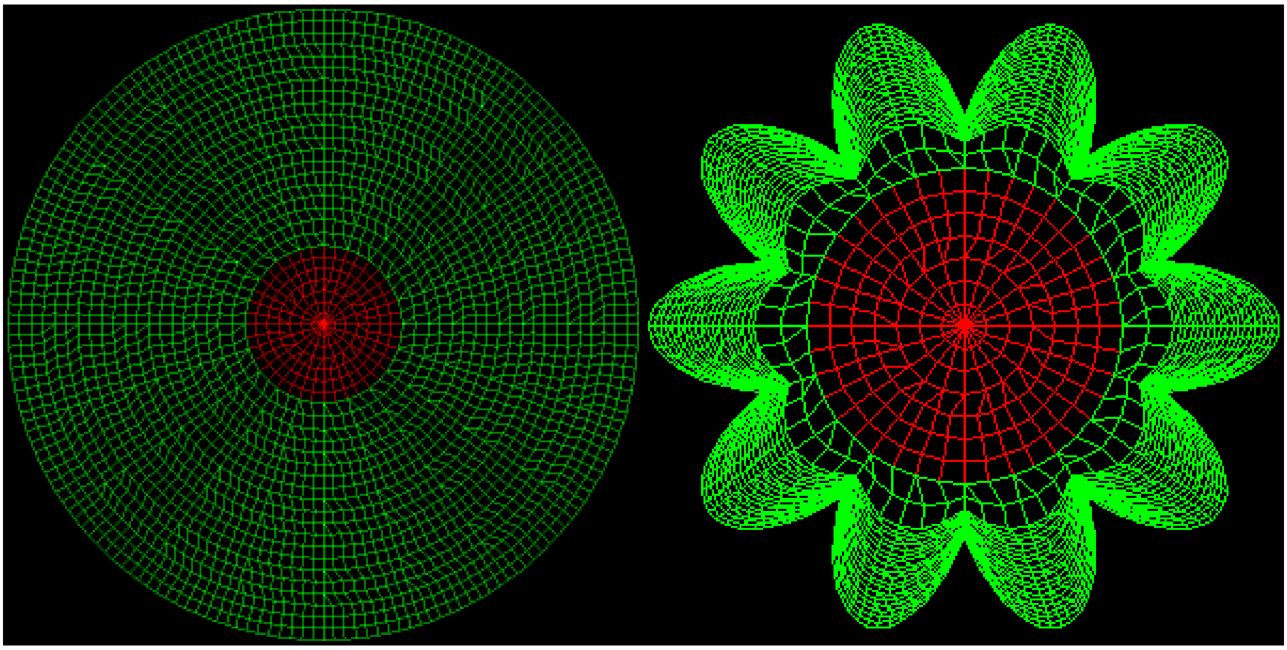
Separation The separation between the upper and lower surfaces of the airbag when unfolded flat.

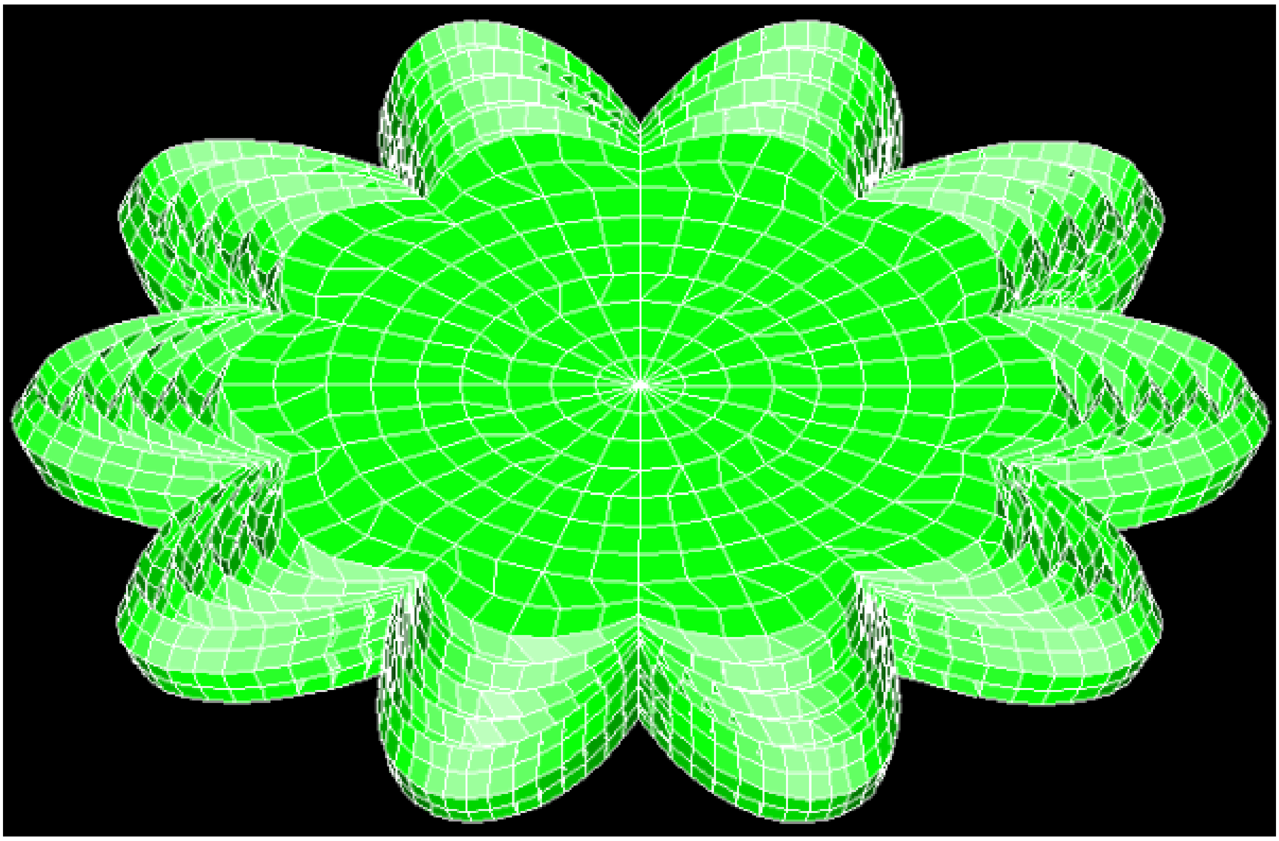
Mesh with The airbag can be meshed either with trias or with a mixture of (mainly) quads and trias. This popup can be used to select which you want

If you change any of the options the graphic also changes showing you the effect of the change. For example, the right-hand figure shows the effect of changing the number of points in the star fold to 24.



The following figures show a starfold with 10 points before and after folding.

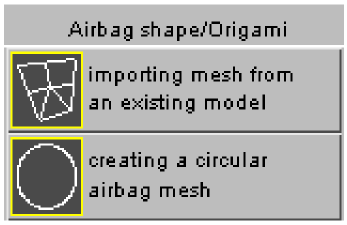




Once the star fold has been created you will be placed in the normal airbag folder. You can then position the bag, check for distorted elements etc.

Once the star fold has been meshed (created) you cannot change the number of points or other parameters. This is because the mesh is dependant on these parameters. If the airbag is not what you expected or required then the process needs to be repeated.

6.1.2 Mesh independent folding



Mesh independent folding allows you to create folds on the airbag at any position, regardless of the mesh on the airbag. It does this by remeshing the airbag after each fold to create a suitable mesh.
For this to work the airbag (or origami) has to be set up in a specific way.

Currently there are 2 methods for doing this.
1) Create a flat circular bag from scratch.
2) Import a mesh

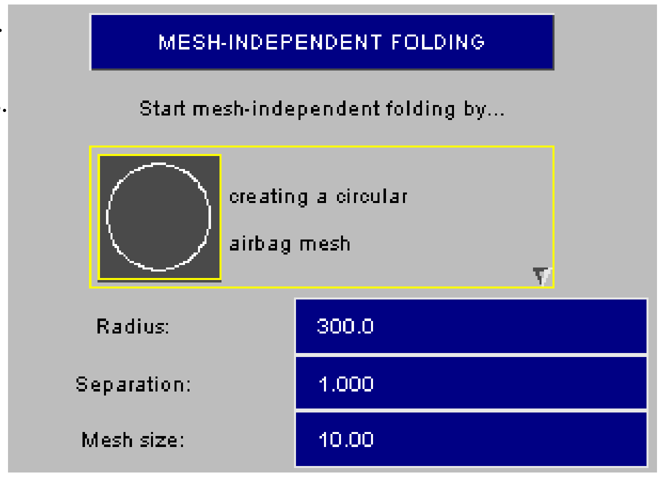
Alternatively, if an airbag has already been defined by one of these methods you can select the existing airbag.

Creating a circular airbag

This option is only available if you have not read an airbag into Primer.

If you have read a model then the option will be greyed out. To enable it **DELETE** the models or restart Primer.

1) From the airbag shape popup select the Circular option.
2) Give suitable values for:
a) the airbag radius.
b) the separation between the top and bottom fabric layers.
c) the element size for the initial mesh.
3) Press the **MESH-INDEPENDENT FOLDING** button.



The airbag will be created and you will be placed in the main folding screen. See section below for details of how to perform mesh-independent folding.

Importing mesh for mesh independent folding

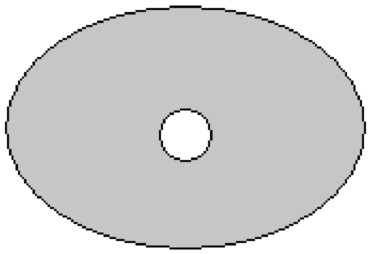
This option allows you to use a mesh that has previously been read into folder for mesh-independent folding.

1) Read an airbag file into Primer.
2) From the airbag shape popup select the Mesh option.
3) Press the MESH-INDEPENDENT FOLDING button.
4) Select the parts that make up the airbag.
5) Select fixed points on the airbag for remeshing.

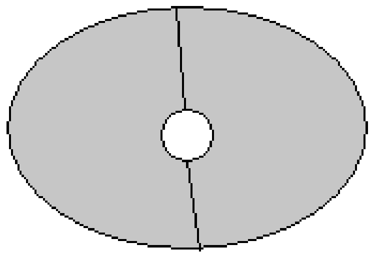
Selecting parts for mesh-independent folding

Folder needs to know which parts make up the airbag before mesh-independent folding can be done. There are limitations on parts that can be used:

1) Parts must be flat
2) Parts must not contain holes.
The algorithms that Folder uses when folding mesh-independent bags do not allow parts to have holes. If your airbag contains holes the parts need to be split.
For example

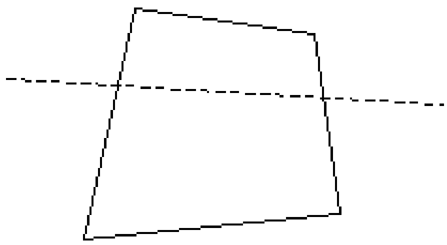


would become

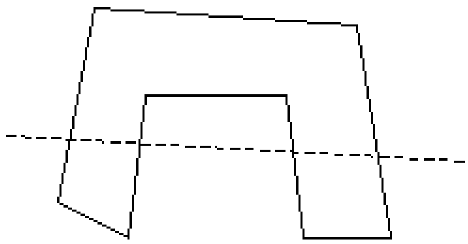


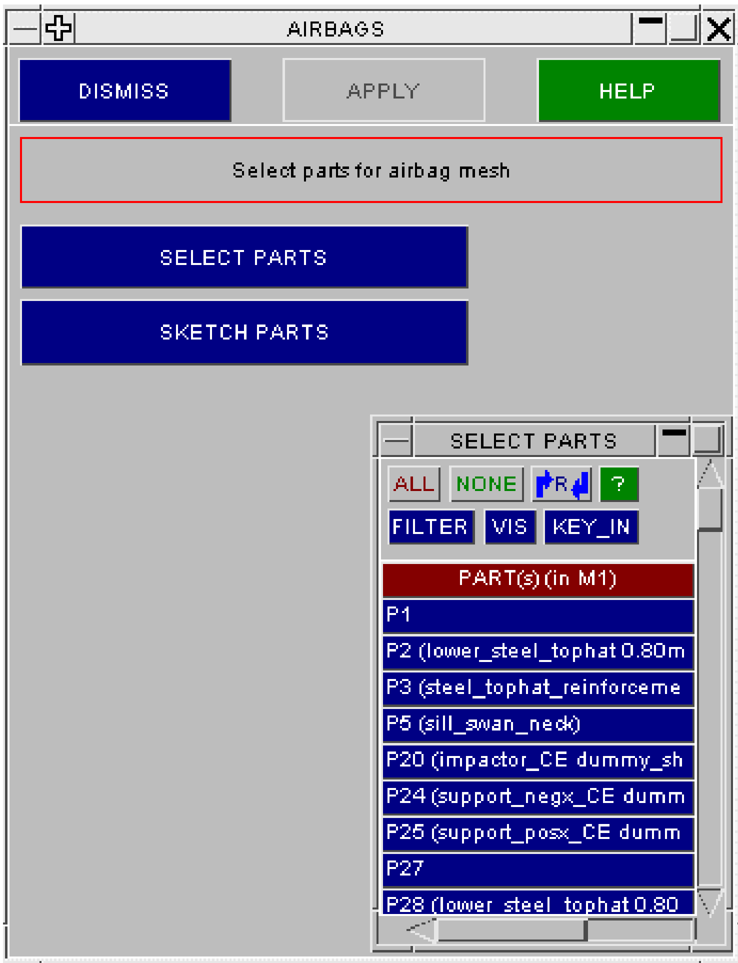
3) The top and bottom surfaces of the airbag must be different parts
4) If a fold will occur on a part, there must only a single fold line.

This part can be folded without problem.



This part cannot be folded as the fold line cuts the polygon in two places.





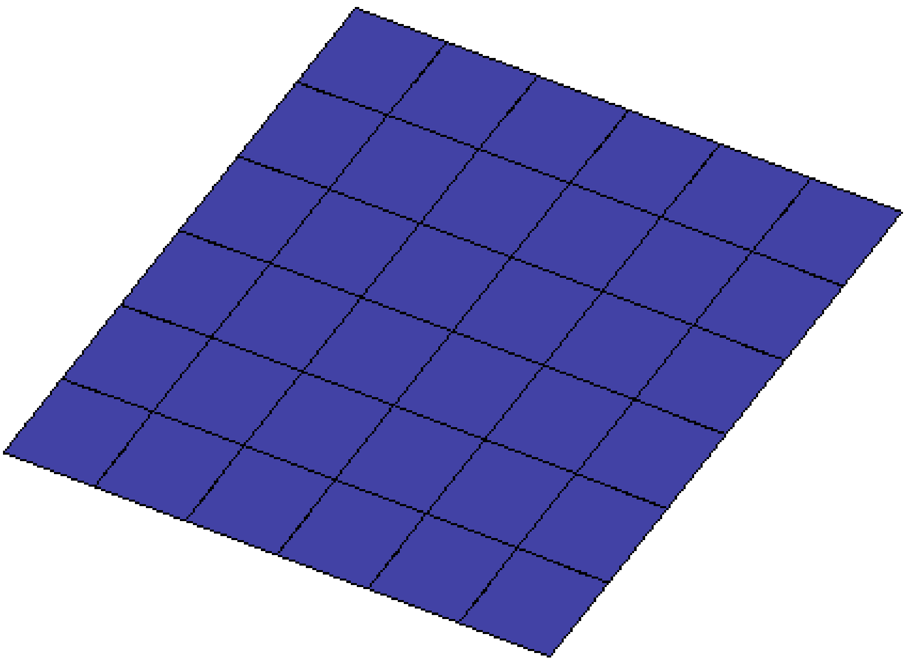
Select the parts that make up the airbag and press **SELECT PARTS**.

You can sketch the parts you have selected by pressing **SKETCH PARTS**.
In this figure 4 parts are selected.

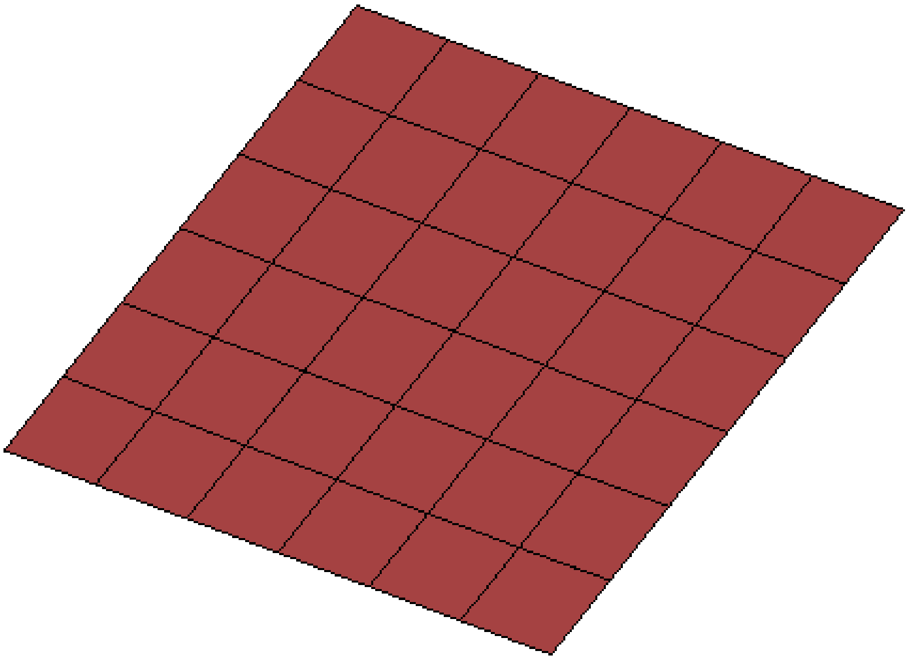
The following section shows some examples of meshes suitable for mesh-independent folding

EXAMPLE 1

A square airbag is needed for mesh-independent folding.

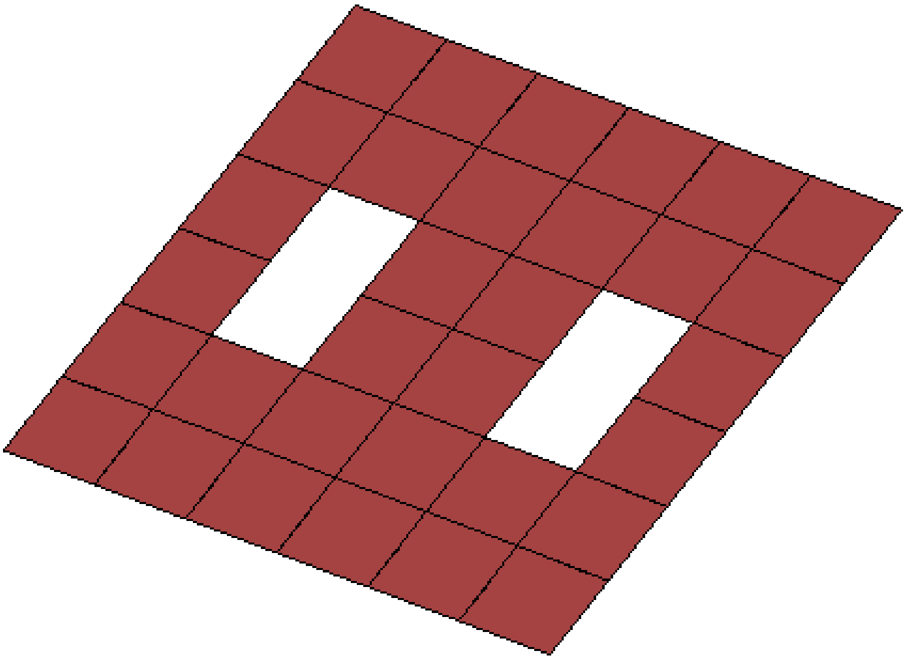


Top surface of airbag

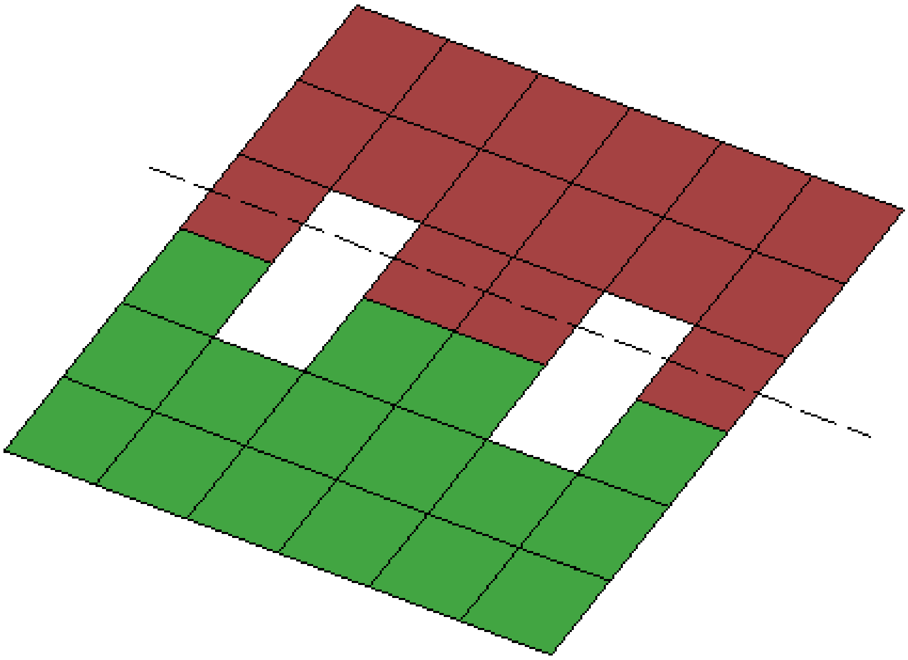


Bottom surface of airbag
This airbag can be used for mesh-independent folding. The top and bottom surfaces of the airbag are different parts, both are flat, and neither have holes.

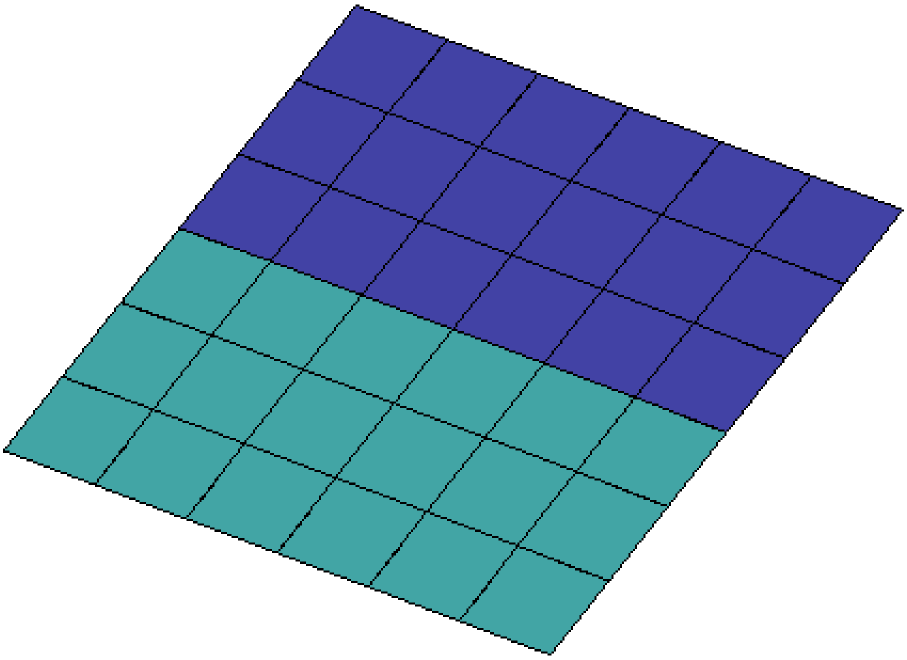
EXAMPLE 2



Bottom surface of airbag
This mesh cannot be used . The bottom surface of the airbag has holes in it.
To overcome this we can split the bottom surface into 2 parts. We also split the top surface to match



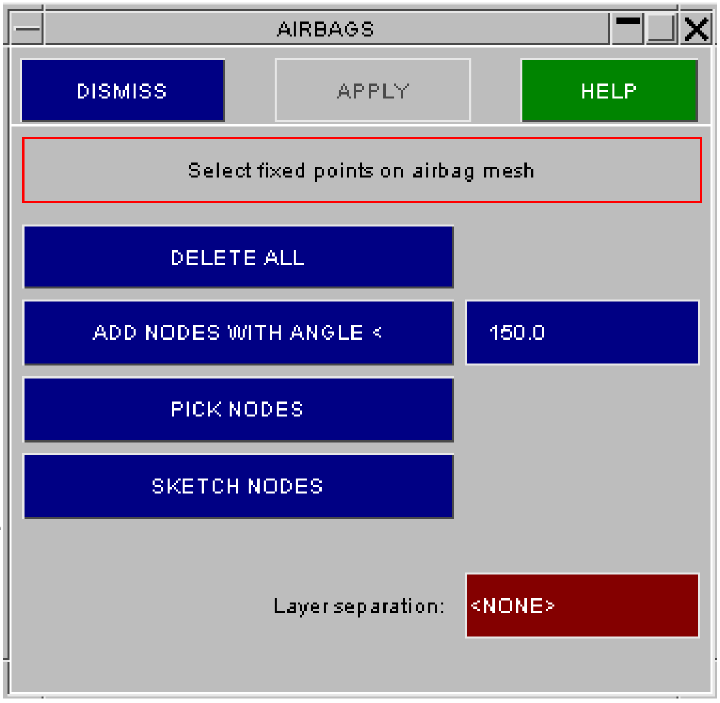
Bottom surface of airbag



Top surface of airbag

We have now eliminated the holes on the bottom surface so this airbag mesh can be used. Note that there would still be a problem if we were going to fold the airbag along the dashed line. In this case the fold line cuts the part in more than one place so will cause problems. If folds such as this are not going to be done the mesh is suitable.

Selecting fixed points for mesh-independent folding



Once the parts have been selected, folder needs to know about the fixed points on the mesh.

The fixed points are required so that folder can remesh the parts in the airbag as a folding is done.

For example, to remesh the simple airbag shown in example 1 above folder would need to know that the corners of the square are 'fixed' in space and so cannot be moved. All the other nodes on the boundary of the parts can be moved without changing the airbag shape.

The 4 corner nodes must be selected as fixed points. If no fixed points are selected folder does not know which nodes are essential and so cannot remesh the airbag as folding is done.

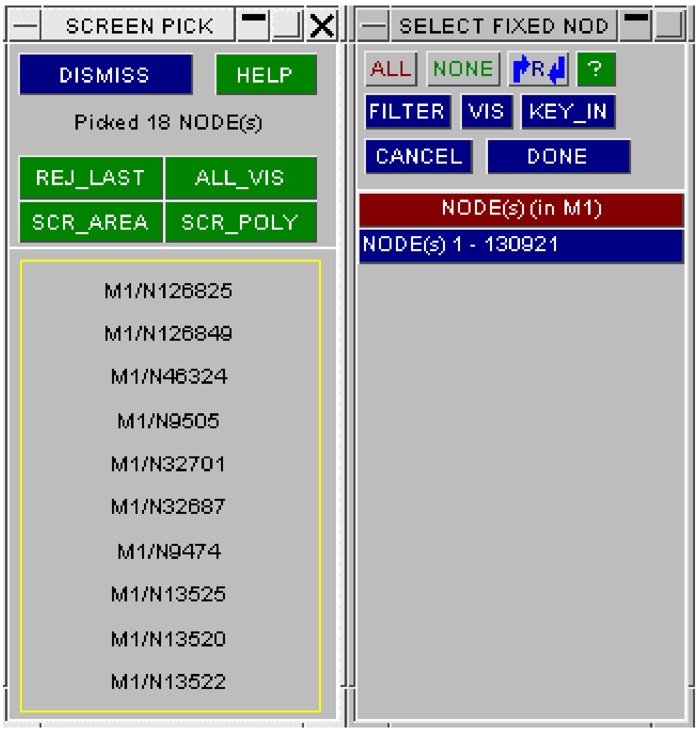
ADD NODES WITH ANGLE can be used to add nodes on the boundaries of parts which have an angle less than the specified value. For example the corner nodes on the square mesh in example 1 have edge angles of 90º so will be selected (< 135º). All the other boundary nodes have angles of 180º so will not be selected.

To enable folder to determine the different surfaces of the airbag you must enter the layer separation (the distance between the top and bottom surfaces of the fabric). In the screenshot above the value has not been entered yet so the **APPLY** button is not active.

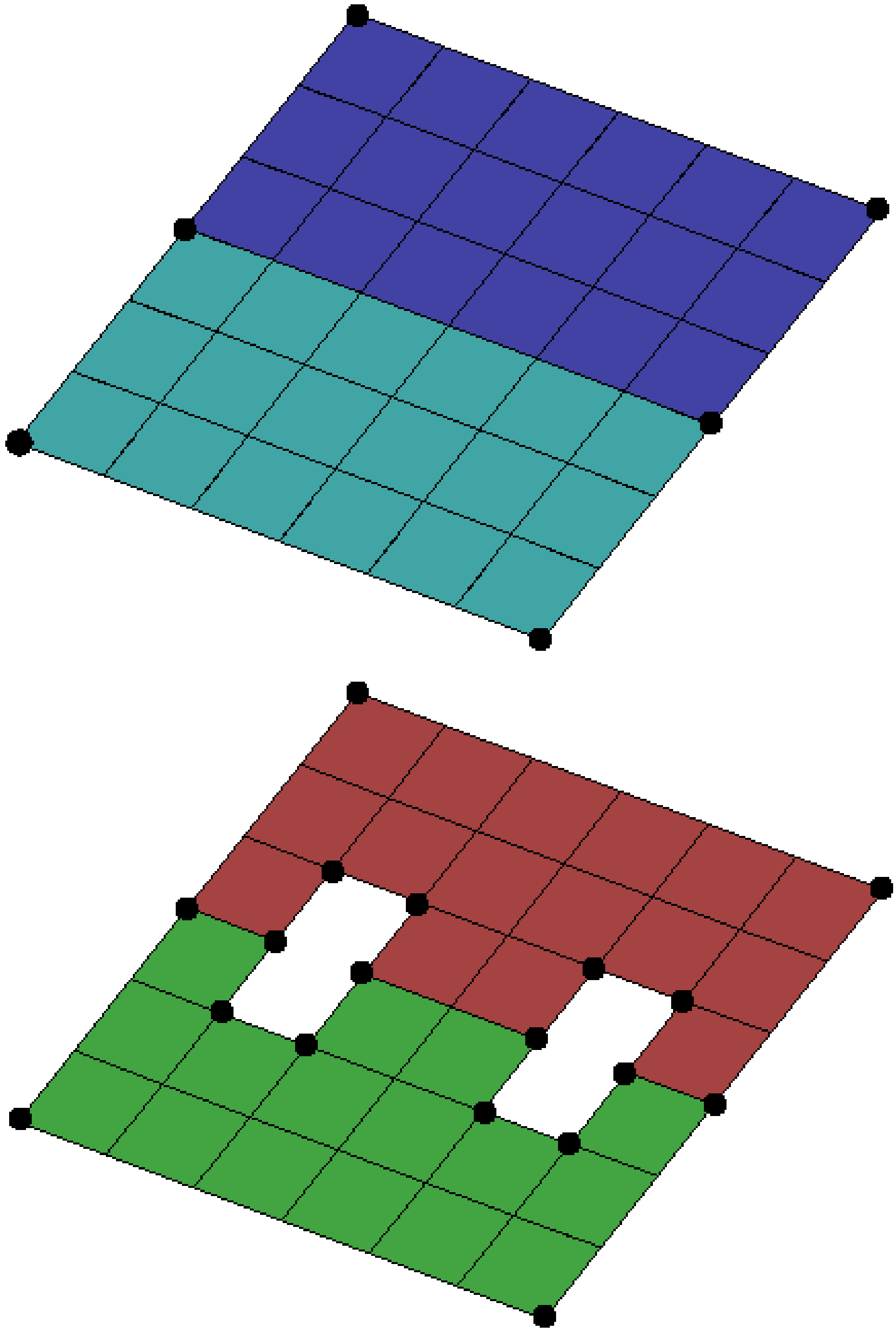
Once all the fixed nodes that you require are selected, **APPLY** will create the origami and take you to the main folding screen

PICK NODES can be used to manually pick nodes to become fixed nodes from the airbag.

When you have selected the nodes on the screen press **DONE** to add them to the fixed nodes



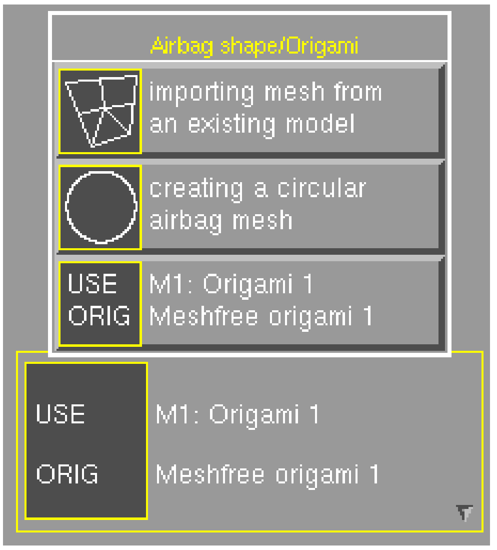
For another example, the mesh with holes in the bottom surface in example 2 earlier would need the following fixed nodes selecting (shown by the dots).



Selecting an existing mesh-independent origami

If you have previously created a mesh-independent origami you can return to the mesh-independent folder by selecting the origami. In the screenshot on the right another option **USE ORIG** is available. For each mesh-independent origami that exists an option will be shown.

An origami will only be shown if **ALL** the folds in it are mesh-independent folds. If it is not shown it is because one or more folds have been done in the normal airbag folder. In this case to be able to return to the mesh-independent folder these folds would need to be deleted.



Performing mesh independent folding

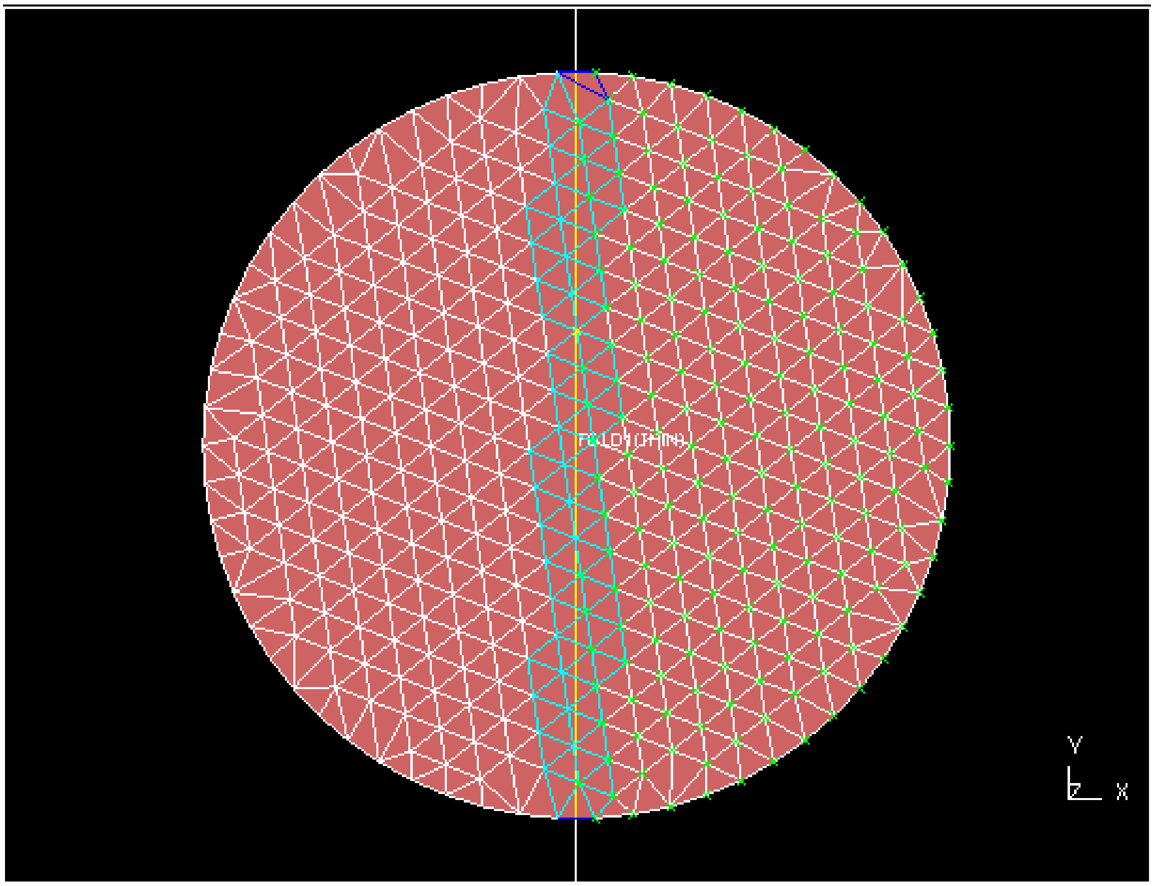
If the airbag has been created for mesh-independent folding or imported the **SPLIT MESH** and **REMESH** buttons will be available in the main folding window.
Mesh-independent folding is available for the following fold types:

- Thin folds
- Tuck folds
- Thick folds
- Spiral folds

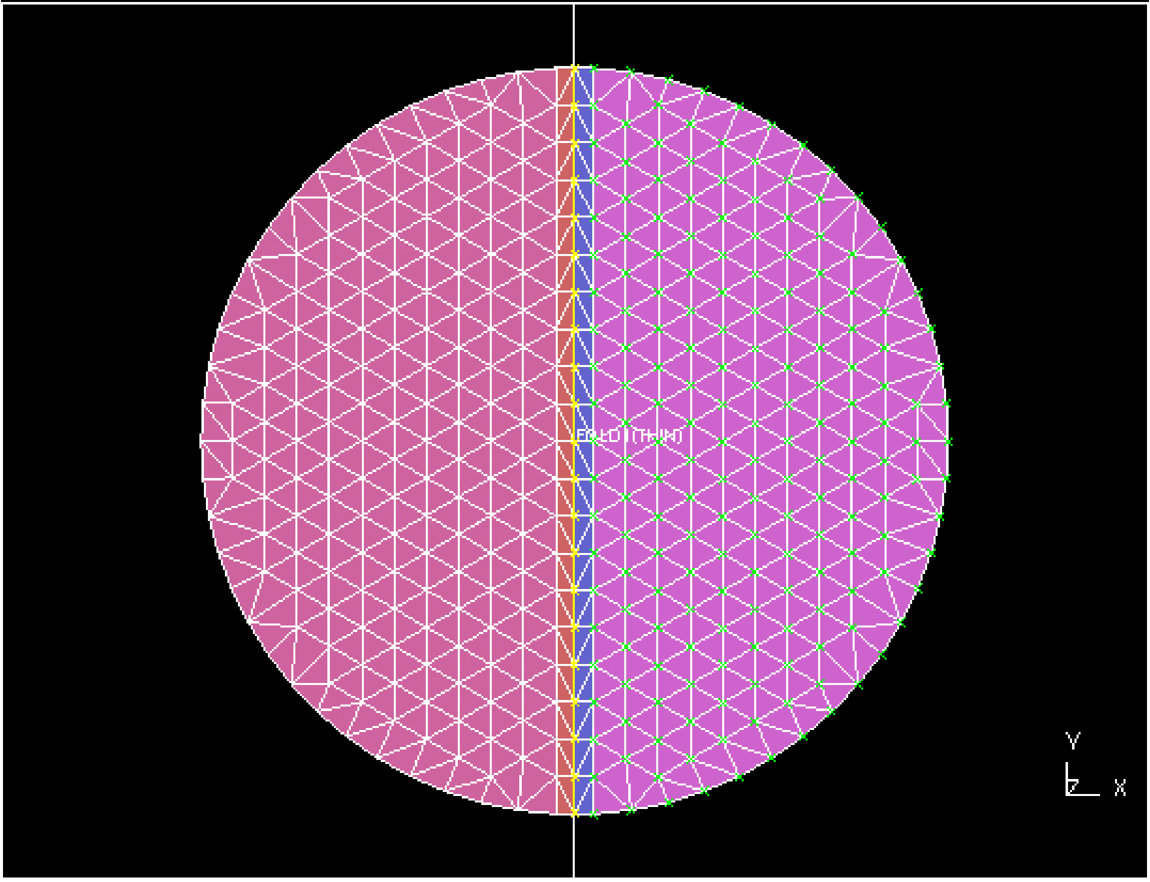
Other fold types are incompatible with mesh free folding and are not allowed (as the airbag is remeshed after each fold the fold definitions would change).

To perform a mesh-independent fold select the fold type, position angle etc as normal (see section 5.5.9). In the example a thin fold will be performed.
Set the tramline size and element size to the required values in the options panel (see section 5.5.14)

Press **SPLIT MESH**.

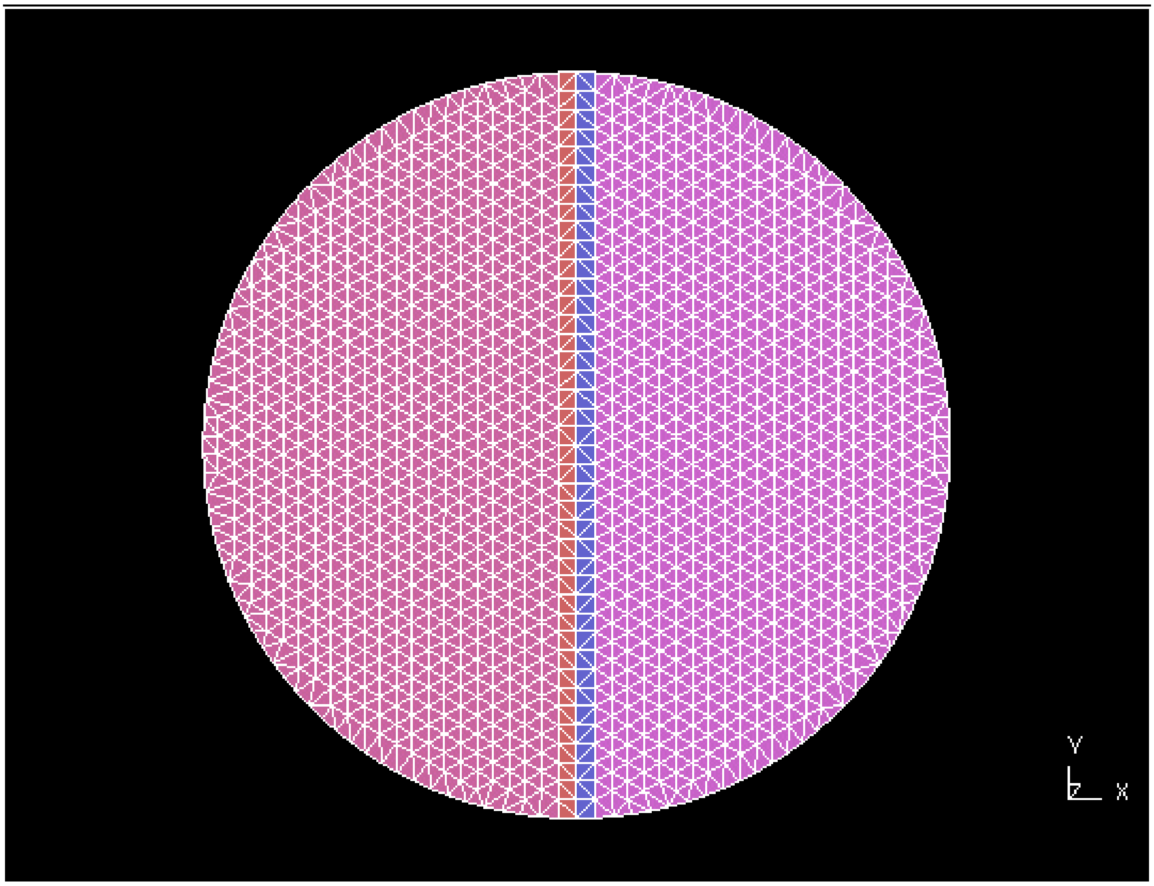


The mesh is split and remeshed. The example on the right shows the result of a thin fold.
The top and bottom surfaces of the airbag are split into new parts as required (as shown by the different colours) and then remeshed. Fixed points are automatically added to the airbag as required.
To undo a split **DELETE** the fold. The parts will be recombined and remeshed (as long as there are fixed points to enable folder to remesh)



At any time the element size in the options size can be changed and the airbag remeshed using the **REMESH** button. For example the airbag above has a tramline size of 5mm and an element size of 10mm. Changing the element size to 5mm and remeshing gives the mesh on the right.

Note that as the thin fold was made with a tramline size of 5mm, the airbag should not be meshed with elements smaller than 5mm as this could lead to more than one element across the tramline parts. This will cause problems with the thin fold.



6.1.3 Folding an existing mesh

The following sections deal with folding an existing mesh. There are several parts to this so it has been split up into several sections.

- Mesh orientation, airbag reference geometry and main orient command (this section).
- Airbag folding summary.
- Creating an origami definition.
- Creating a local coordinate system.
- Plotting modes.
- Using the main folding panel.
- Creating a new fold.
- Fold types.
- Subset folding.
- Options available in the airbag folder.
- Positioning the origami.
- Saving/reading origami and fold definitions.
- Tips and notes for successful folding.
- Folding example.

Suitable initial orientation of the mesh

The ideal way to fold an airbag is to define the unfolded airbag in the X-Y plane, fold it, and then orient it in space with respect to the vehicle space. If it is not defined topologically to be in the X-Y plane it would be sensible for the user to define a local co-ordinate system such that the local system is planar with the airbag (see Section 6.1.6). When folding then takes place it will visually appear to be in the global X-Y plane. On exiting the airbag folder the bag will return to the global space.

However, it is strongly recommended that folding should start with the initial geometry in the global X-Y plane, as this will reduce the chances of confusion about where the geometry actually is, and make life simpler!

How the ***AIRBAG_REFERENCE_GEOMETRY** interacts with folding

The ***AIRBAG_REFERENCE_GEOMETRY** definition is also crucial to folding, as it is used as the basis for all folding operations. If no such definition exists when folding starts, then PRIMER will automatically copy the nodal coordinates *in their current configuration* into the reference geometry. (Any existing reference geometry for a node will not be over-written.)

This is not a wholly satisfactory solution, as an initial reference geometry which gives a satisfactory element shape for stress calculation during analysis may not prove an ideal starting point for folding operations: typically where a 3D bag needs to be split into several 2D panels for folding, and then reassembled. In this situation it may be necessary to save the "true" reference geometry (for analysis) in a separate file, delete it from the input deck used for folding in PRIMER so that a more satisfactory geometry can be used for folding each panel, then re-introduce it prior to analysis.

This is not a wholly satisfactory state of affairs, and the geometrical basis for folding may in future be changed to be independent of the reference geometry.

Interaction between the **ORIENT** command and folding

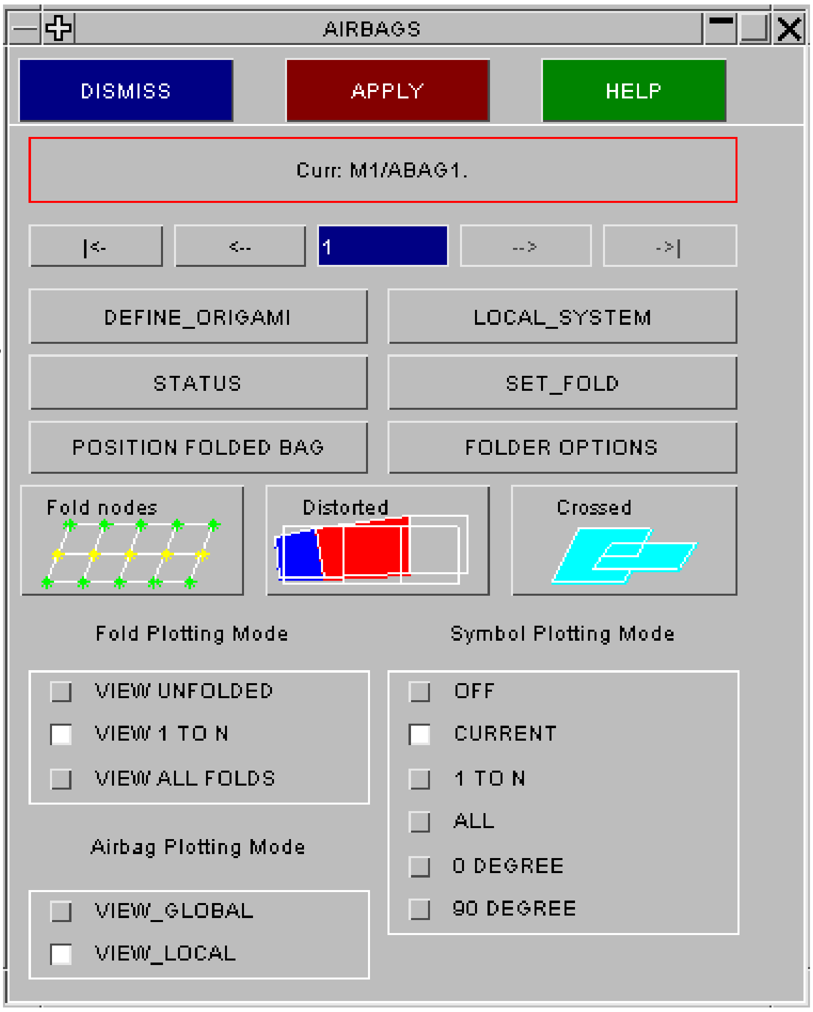
Both **ORIENT** and folding act to change the current coordinates of nodes, and the order in which they are applied is important since the most recently used will "win" in determining the final position of nodes. This is important since folding is always based on the *reference* geometry (which is unaffected by orientation commands), whereas **ORIENT** always operates on the *current* (ie as-folded) geometry. If you use **ORIENT** to position an airbag after folding and then return to the airbag folder at a later date to refold the airbag any orientations will be lost. To prevent this problem the airbag folder has built in orientation functions. These orientations are saved just like folds and so if a bag is refolded it can be repositioned using these stored orientations.

It is strongly recommended that you use the orientation functions build into the folder rather than the general orient functions of Primer so that any orientations you create are saved and can be modified or replayed in the future.

6.1.4 Airbag Folding Summary

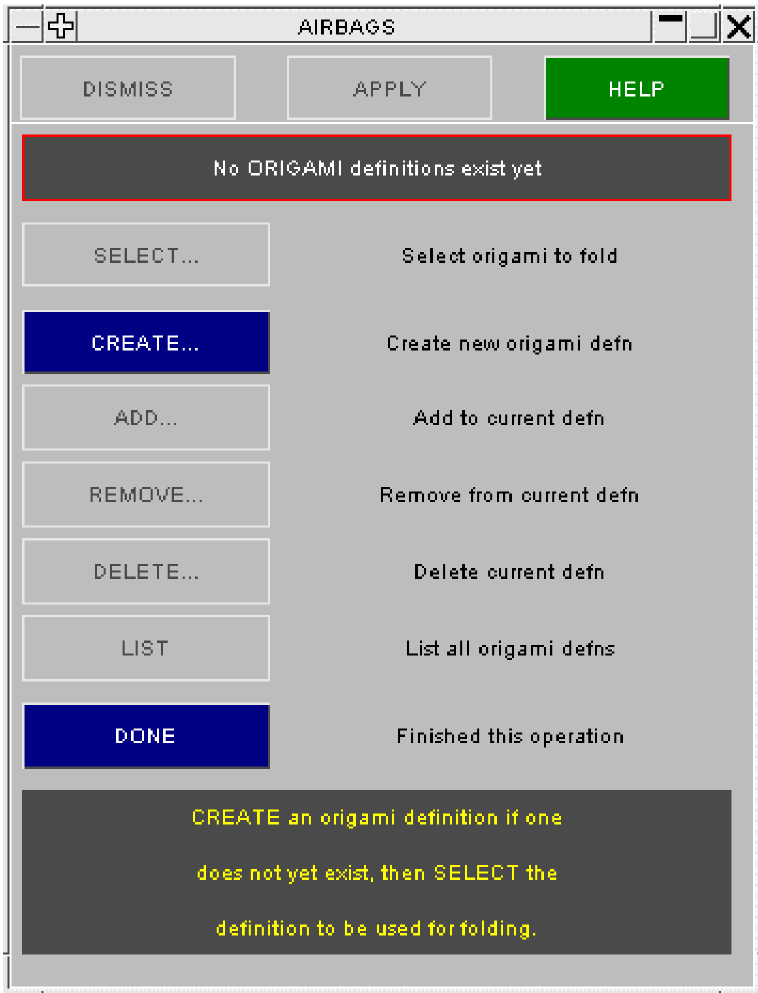
The following process should be followed when folding an existing airbag mesh:

- Read in a model containing airbag geometry (and possibly Origami definitions), and enter the PRIMER airbag module by using the **AIRBAGS** buttons.
- Define an ORIGAMI or select an existing one using **DEFINE_ORIGAMI**.
- Set the local coordinate system if your airbag is not in the global xy plane using **LOCAL_SYSTEM**
- Set the plotting modes that you require using the Fold Plotting Mode, Airbag Plotting Mode and Symbol Plotting Mode radio buttons.
- Set any options for the airbag folder using **FOLDER OPTIONS**
- Define or modify folds using **SET_FOLD** using subset folding, sets and layers as appropriate.
- If required position the folded bag using **POSITION FOLDED BAG**
- Exit airbag folder by pressing **APPLY**. Note that other methods of exiting from this model will lose current fold definitions.
- Save the model to disk.

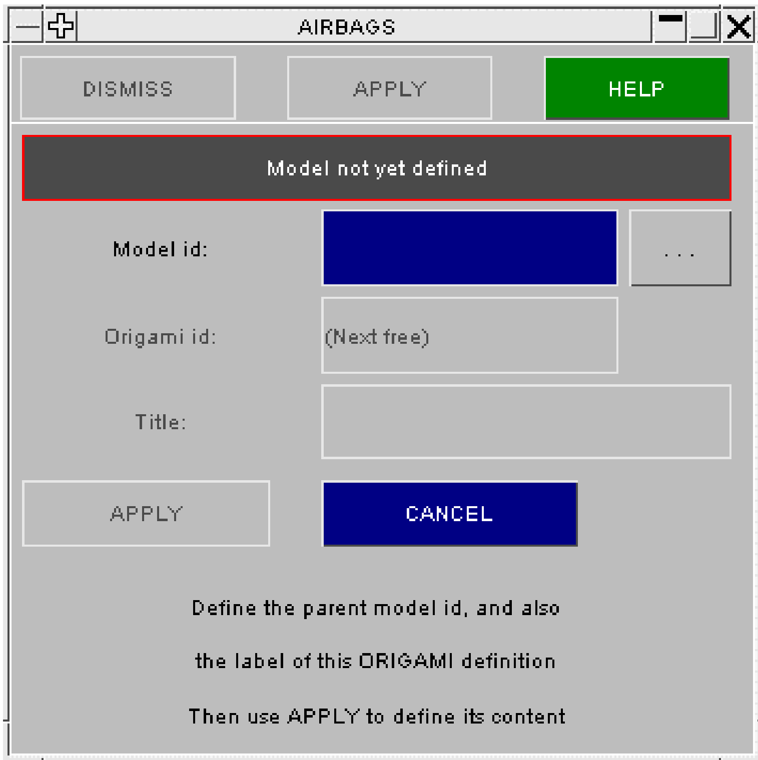


6.1.5 Define Origami: Selecting or creating an ORIGAMI Definition

Before you can fold anything you must have a current ORIGAMI definition. **DEFINE_ORIGAMI** in the main folding screen above gives the Origami definition menu (above left). In this example there are no existing definitions, so it is necessary to **CREATE...** one:.



An **ORIGAMI** definition can only exist in a single model, so the first phase of creation is to select a model, and then to define the Origami label and title. Any number of Origamis may exist within a model, but they must have unique labels.



Once the basic data has been defined you are presented with the standard selection menu which will allow you to define the SETs and/or PARTs and/or ELEMENTs which constitute this Origami definition. These define the nodes and elements which are to be folded.

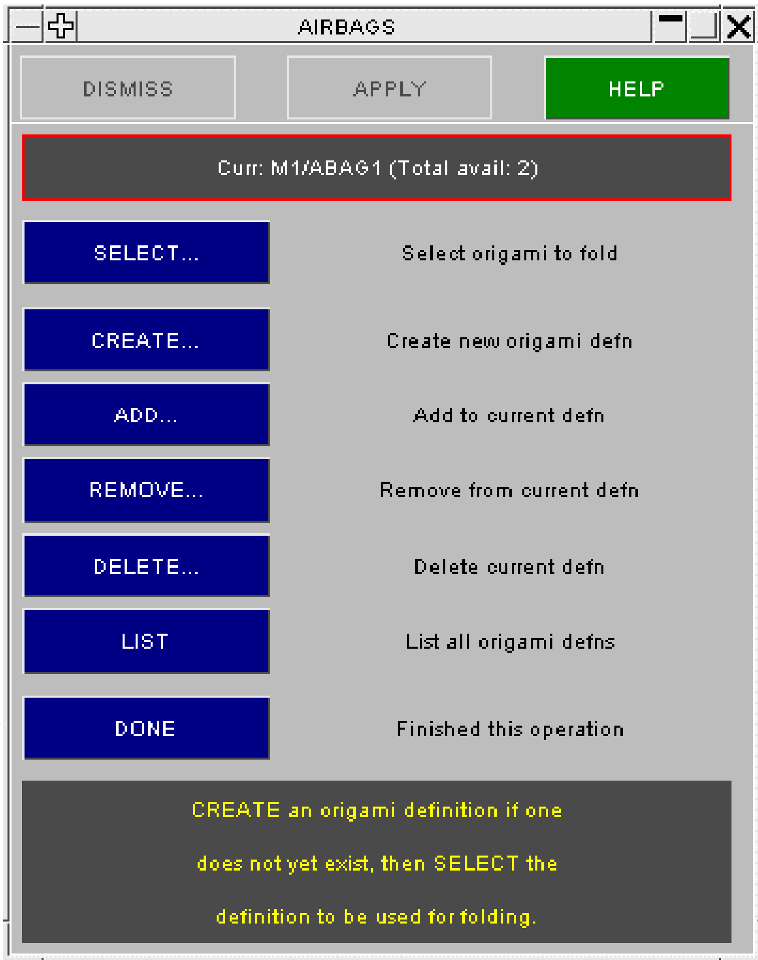
Finally the "AIRBAG REFERENCE GEOMETRY", which is used as the starting point for folding, is set up automatically by PRIMER. If this does not exist for any nodes to be folded, then it is created at this stage by copying the *current* geometry of those nodes.

The definition is now complete, and the Origami definition panel will now be fully populated as shown in the adjacent figure.

You can now **SELECT...** which definition you want to fold.

The other operations here, **ADD..**, **REMOVE..**, and so on are self-explanatory. Origami definitions can be edited at will by re-visiting this panel and manipulating them as required.

Use **DONE** to return to the main folding menu in order to proceed with folding.



The Origami definition is now a permanent part of your model, and will be written out after the ***END** card in a LS-DYNA deck so that it can be re-read in future PRIMER runs. (Note that ***ORIGAMI** is not a standard LS-DYNA keyword, and it is placed *after* the ***END** card so that it will be ignored by the LS-DYNA analysis code.)

It is possible to edit the ***ORIGAMI** data in a file by hand - Appendix III describes the format of this data - but it is ***strongly*** recommended that you do not attempt this as the data stored is quite complex: if you want to edit Origamis read them back into PRIMER and do the work there.

Also you should be very careful not separate ***ORIGAMI** definitions from their "parent" input files, because they contain references to coordinate systems, sets, elements and nodes that exist uniquely within those files. If you want to keep standard airbag files "on the shelf" make sure that they are complete with geometry and folding data kept together: PRIMER will merge airbags and structural models for you.

6.1.6 LOCAL_SYSTEM: Defining Airbag Local Axes

In general it is easiest to fold a bag if it is oriented initially in the global X-Y plane. However, if it is already in a vehicle this is unlikely to be the case. The user can therefore move the bag to a more convenient folding position.

This is done by pressing **LOCAL_SYSTEM** which will invoke the menu shown on the right. The user should then define the local X-Y plane (referred to as the origami local system) by selecting three nodes:

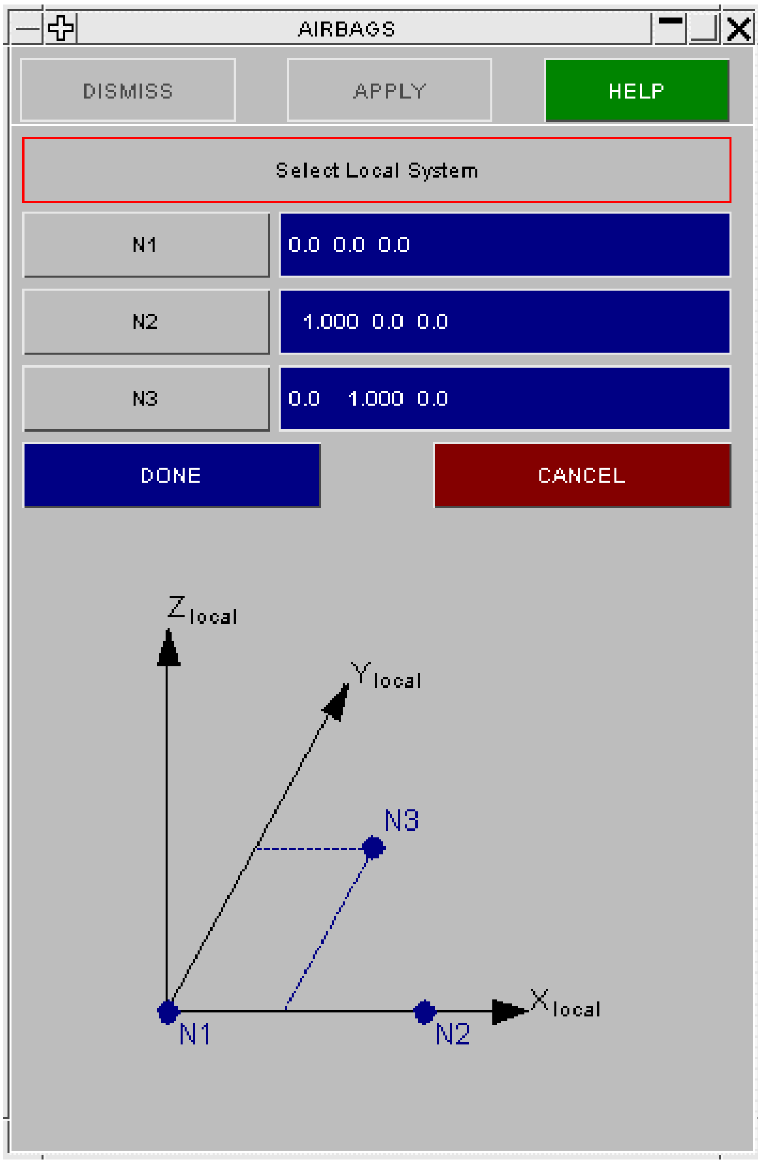
N1 defines the local axis origin;

N2 defines the local x-axis;

N3 defines another point in the X-Y plane.

Alternatively you can type in the local axis vectors, which PRIMER will normalise for you.

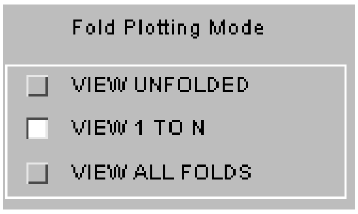
If this option is invoked when the current fold number is non-zero (see Section 6.1.8) the user can define a local axis system that affects the current fold only (referred to as the fold local system). Otherwise it applies to the Origami definition as a whole.



6.1.7 Plotting Modes

You can exercise control over how the Origami definition is drawn and annotated in a variety of ways. The following buttons are on the main folding window.

Fold Plotting Mode



VIEW UNFOLDED displays ORIGAMI without applying the folds to the displayed geometry;

VIEW 1 TO N displays the ORIGAMI folded up to the current fold (see Section 6.1.9) excluding any other folds;

VIEW ALL FOLDS displays all folds defined on the current ORIGAMI. This is not affected by the current fold number.

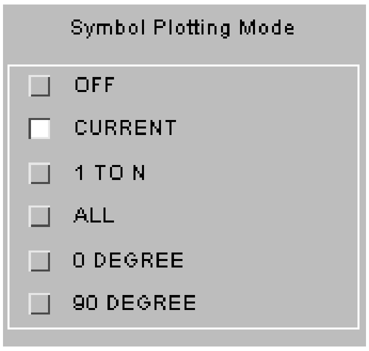
Airbag Plotting Mode (The coordinate system used for airbag display)

VIEW_GLOBAL Displays the airbag in the global cartesian system;

VIEW_LOCAL Displays it in its local system (if defined).

Symbol Plotting Mode (Plotting of fold line symbols)

Note that where folds slice through elements the actual fold line (ie along element edges) is shown as well as the defined fold line (ie across elements). The fold line symbol can be:



OFF No fold symbols are displayed;

CURRENT Only the current fold is displayed;

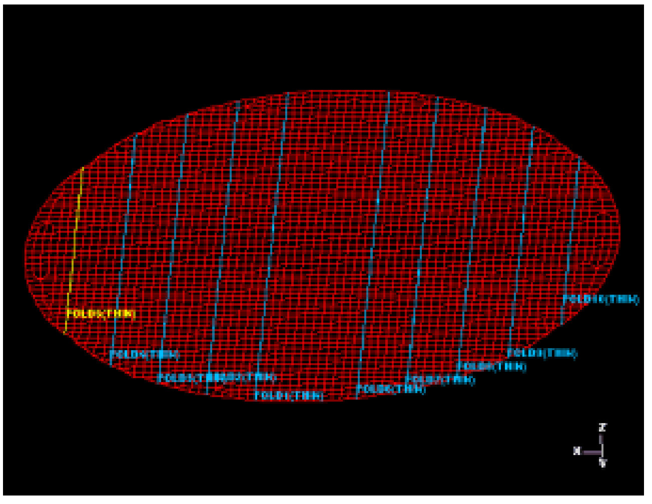
1 TO N Displays all folds up to the current fold;

ALL Displays all folds;

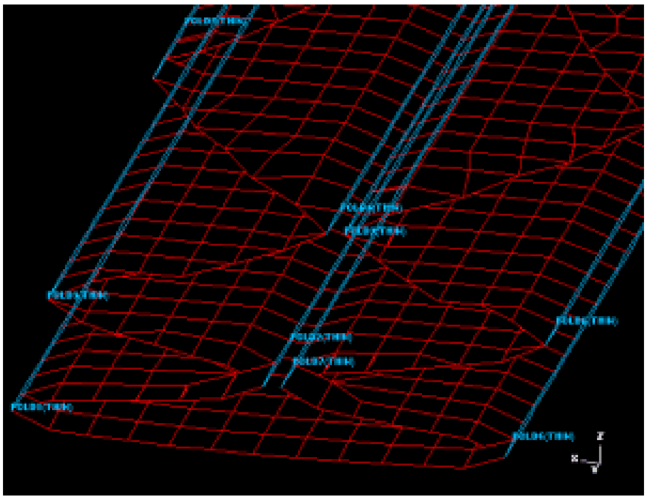
0 DEGREE Displays all folds perpendicular to the (local) X-axis;

90 DEGREE Displays all folds parallel to the local X-axis.

The following two examples show typical "unfolded" and "folded" displays of the same airbag, with fold line symbols superimposed in both cases.



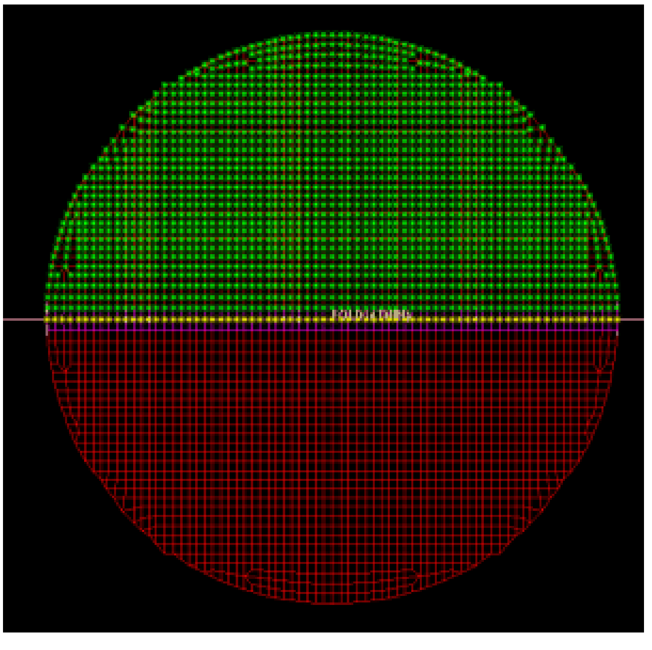
Unfolded mesh drawn in the global system showing all fold lines superimposed



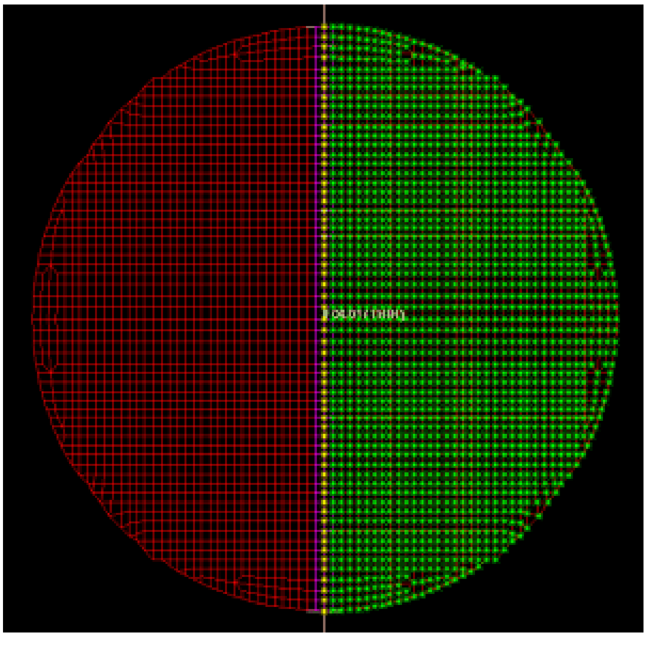
Folded mesh (all folds) showing fold lines superimposed.

Fold Node plotting (Visualisation of fold nodes)

By default when you make a new fold the nodes which will be moved in the fold are highlighted. This helps to show if you have the correct nodes selected for the fold. For example the next 2 figures show which nodes will be folded in a 0º or 90º fold.

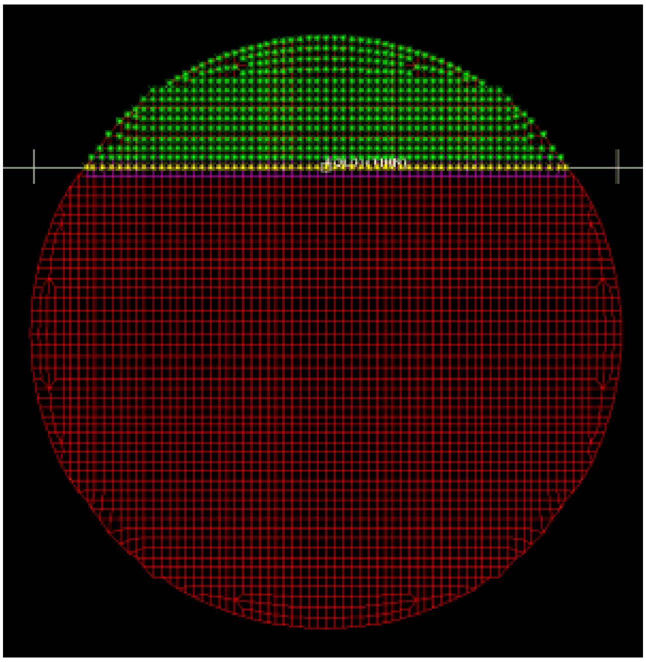


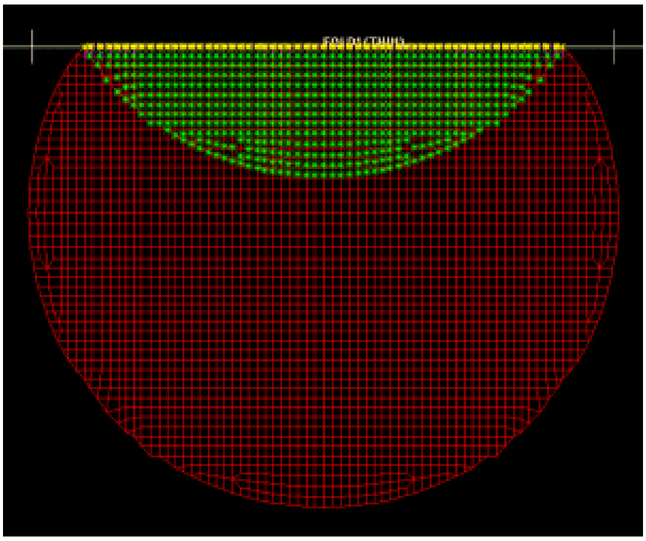
Fold nodes shown for a 0º fold from right to left.



Fold nodes shown for a 90º fold from top to bottom.

Changing any fold parameter such as the fold point, fold direction, fold angle automatically updates the display.



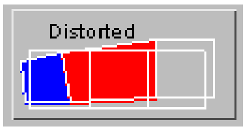


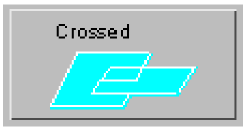
90º fold after changing fold point

90º fold after applying thin fold

Automatically drawing the fold nodes can be turned off by using the folder options (see section 6.1.11). At any time the fold nodes can be redrawn by using the **FOLD NODES** button.

Crossed and distorted element plotting





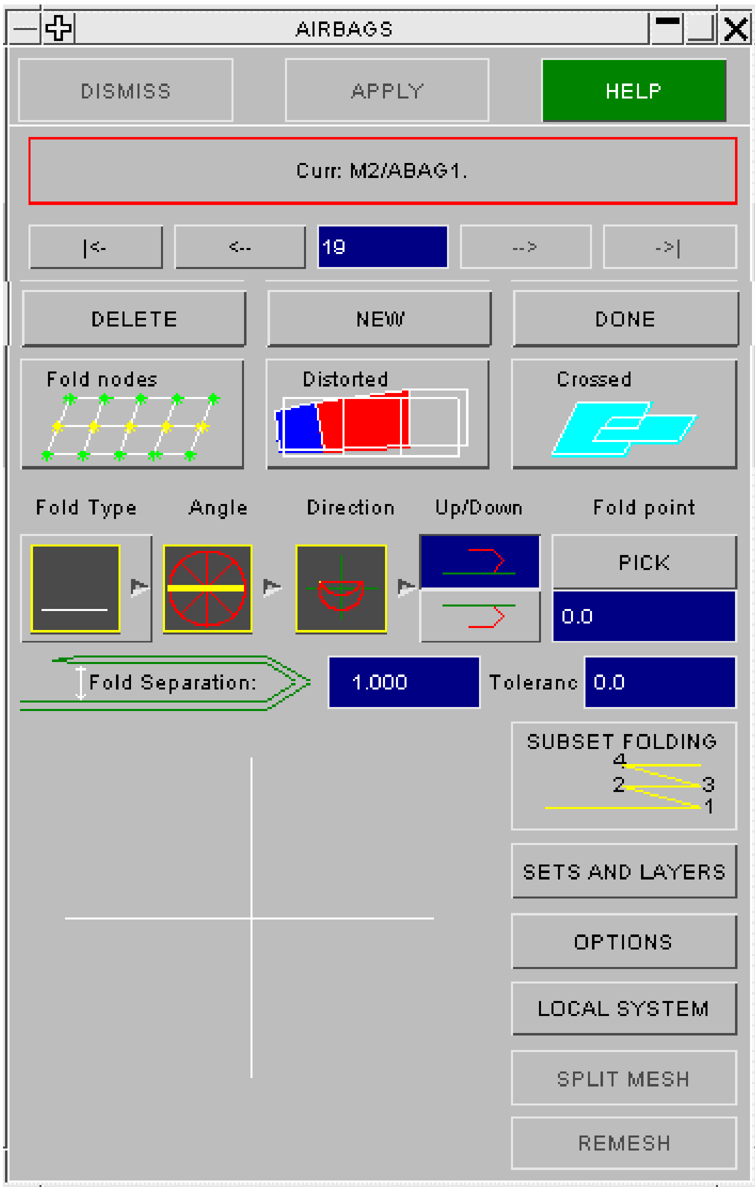
By default when doing a fold, elements which are distorted in the folding process and elements which have penetrations or are crossed are highlighted on the origami. Just as with the fold nodes in the previous section, when any fold parameters are changed the display is automatically updated.

Automatically drawing of crossed, penetrating and distorted elements can be turned off by using the folder options (see section 6.1.11). At any time the elements can be redrawn by using pressing the **DISTORTED** and **CROSSED** buttons.

Elements are crossed when a node from one element has passed through the mid plane of another element. Elements are defined as penetrating if a node from one element is within the thickness of another element but has not passed through the mid plane. These features are very useful when adjusting the tip scale factors for thin folds. As the fold tip is adjusted the display will show if there are any penetration problems. In this way any potential penetration problems can be visualised and fixed. The thickness which is used for the penetration check can be altered in the folder options (see section 6.1.11).

Element distortion is defined as the ratio of the current element side or diagonal length divided by the reference element side or diagonal length. Therefore if an element is stretched the number will be greater than 1. If the element is shrunk the number will be less than 1. Three different contour bands are available for plotting distorted elements. The ranges and the colours for each contour can be altered in the folder options (see section 6.1.11).

6.1.8 SET_FOLD Creating Fold Definitions



Pressing **SET_FOLD** in the main folding window invokes the folding menu in the adjacent figure.

The fold number is adjusted by the **|< ... >|** buttons.

Fold control and saving are controlled here.

Fold node, distorted and crossed element plotting buttons.

The current fold type (null, thin, etc.), fold angle, fold direction, fold orientation and fold location.

The default/fold thickness and tolerance are set here.

Subset folding button

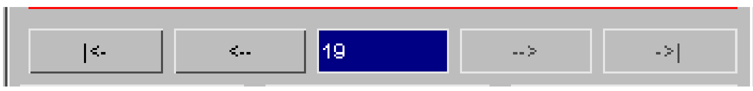
Sets and layers (section 6.1.10)

Airbag folder options

Local coordinate systems

Mesh independent folding

Selecting the current fold number



The currently active fold is set via the buttons or text box in this region. In the figure above fold zero is shown, which implies "the whole origami", and which allows you to set values for the whole airbag. Once finite fold ids (in this example #3) are shown, then operations apply to that fold only.

Controlling fold progress

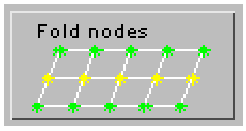
DELETE Deletes the current fold definition. All folds above this one have their number decremented by one.

DONE Finishes the folding process and returns the user to the main FOLDING menu.

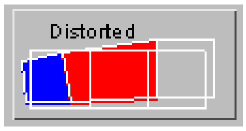
NEW Creates a new fold. If used in the middle of a sequence a new fold is inserted between the previous and next folds, and all folds above this point have their number incremented by one.

Viewing fold node and element information

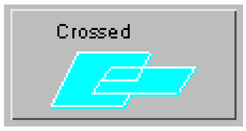
FOLD_NODES Redraws the fold nodes at any time. By default this is done automatically but if this is turned off with the folder options this button can be used



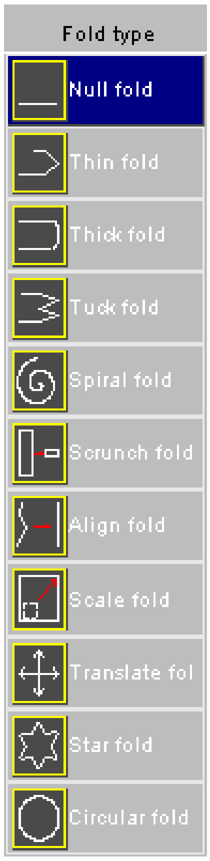
DISTORTED Redraws the distorted elements at any time. By default this is done automatically but if this is turned off with the folder options this button can be used to plot the distorted elements.



CROSSED Redraws any crossed elements. By default this is done automatically but if this is turned off with the folder options this button can be used.



Selecting the current fold type



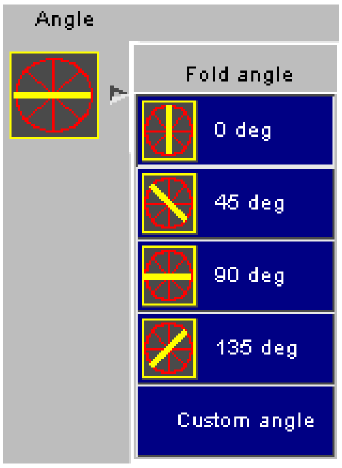
The **Fold Type** popup button controls the current fold type, which will be one of

- **Null fold**
- **Thin fold**
- **Thick fold**
- **Tuck fold**
- **Spiral fold**
- **Scrunch fold**
- **Align fold**
- **Scale fold**
- **Translate fold**
- Star fold
- Circular fold.

In this example a "thin" fold is current, but the type can be changed dynamically by simply selecting a new type with the popup menu. More information about each fold type is given below in section 6.1.9. Star and circular folds are unavailable from this menu. These are selected to be created from the initial airbag start up screen. For more information on these types see Creating and Folding a new airbag.

The fold type button will toggle between the currently selected fold type and a null fold.

Setting the Fold Angle



The Angle popup button can be used to select the angle of the fold. In this diagram the current fold angle is 0º . The popup contains default fold angles of 0, 45, 90 and 135º . Most folds will take place in either the X (0º) or Y (90º) directions, but if necessary, you can use Custom angle to set the fold to an arbitrary angle.

If a new fold angle is chosen the graphic on the Angle button will be updated so you can easily see the angle of the current fold.

Defining an arbitrary XY fold angle using Custom angle

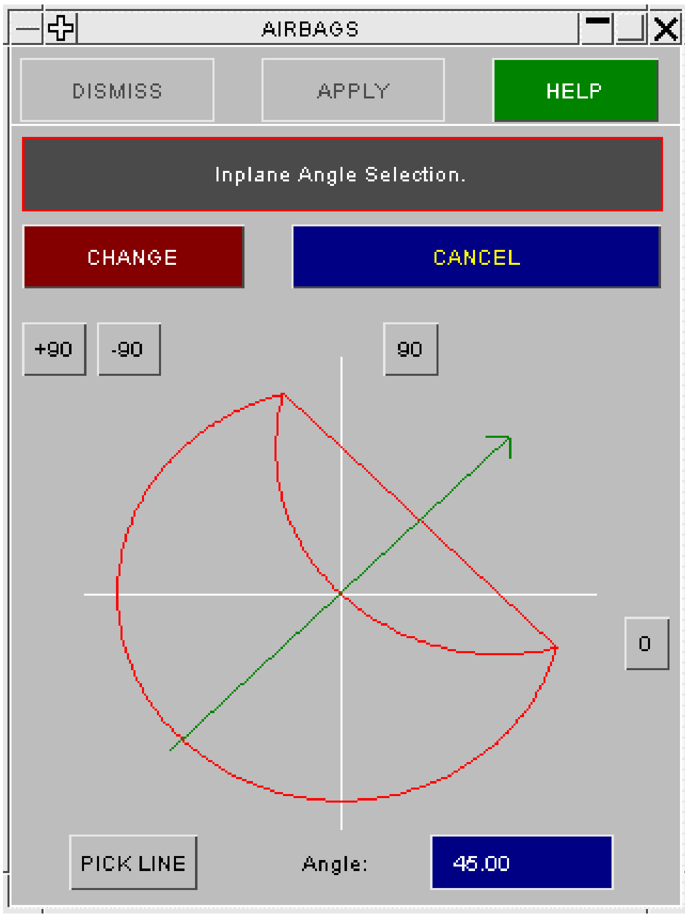
The adjacent figure shows the box after pressing Custom angle. This allows the user to define an arbitrary fold line.

The angle may be selected in either of the following ways:

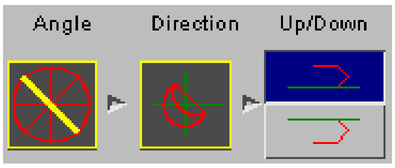
PICK LINE Calculates a line, and hence an angle from two screen-picked nodes.

Angle: Accepts a typed in angle, which may also be modified by **[0]**, **[90]**, **[+90]**, **[-90]**.

To accept the current definition press **CHANGE**, which will return to the fold definition menu.



Setting the Fold Direction



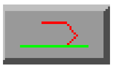
As well as setting the fold angle, the direction in which the fold is done needs to be set. In the adjacent figure the fold angle is set to 45º (you can tell this because of the angle of the yellow line). The graphic shown for the direction of the fold is also drawn at the 45º angle but there are two possible ways to do this fold

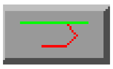
The upper right part of the airbag can be folded onto the lower left part of the airbag (**Forward**).
The lower left part of the airbag can be folded onto the upper right part of the airbag (**Reverse**).

You can select the fold direction by choosing either Forward or Reverse. The graphic on the **Direction** button will change to indicate your selection.

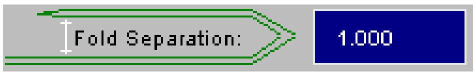
Setting the up/down direction

When a fold is done the folded material can either be folded on top of or underneath the unfolded material. The Up/Down buttons are used to set this.

 Folded material is folded on top of unfolded material.

 Folded material is folded underneath unfolded material.

Setting fold separation distance

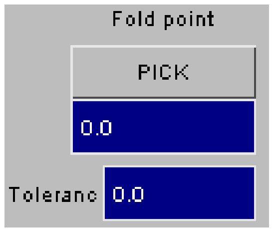


When a fold is current this sets the distance between layers, the thickness.

If fold #0 is current, this sets instead the Default Thickness to be used for all folds in the airbag, which may be overridden for individual folds.

When folding takes place the larger of the default and individual fold thicknesses is used, so a useful way to visualise an airbag is to set the individual fold thicknesses to their correct, relatively small, values; but to set the default thickness to a larger value, so making it easier to view folds. When folding is complete the default value can be reset to a more realistic value.

Setting the FOLD_POINT and Tolerance



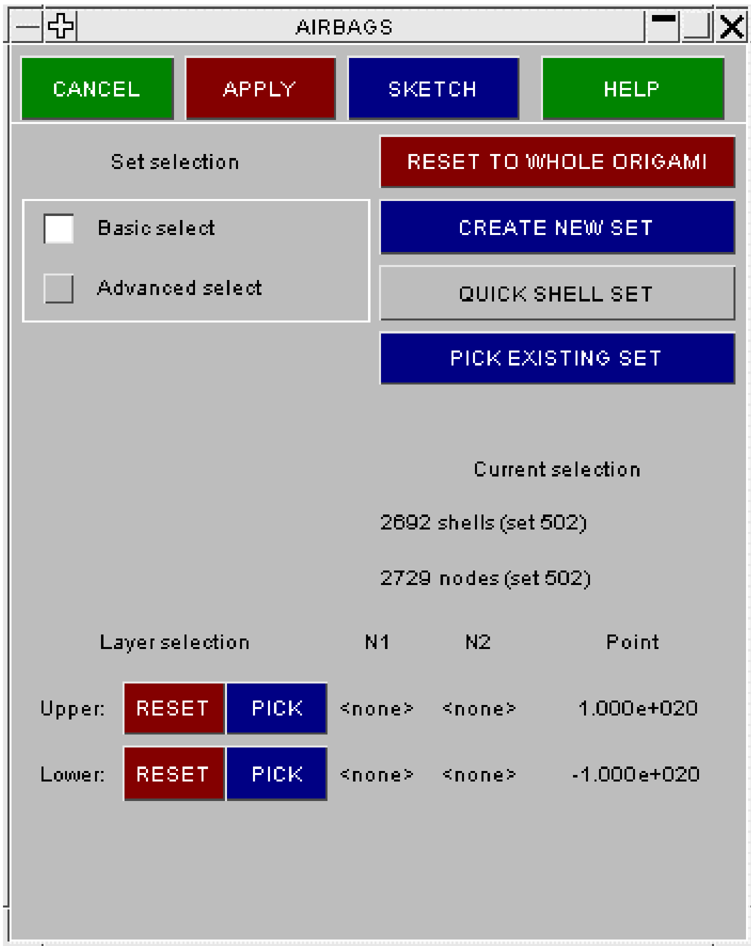
The fold point identifies an XY coordinate through which the fold line passes. This can be defined by using **PICK** to screen-pick a node, or by typing in an explicit value (which is a distance down the relevant axis). For thin and tuck folds this point should lie on, or be very close to, a node; for other fold types any reasonable location may be chosen.

Tolerance defines the tolerance margin either side of the fold line used when searching for nodes that lie on the line. You should aim to choose the smallest value that will include all nodes on the fold line.

6.1.9 SETS AND LAYERS Selecting a subset of the airbag for folding

By default the whole origami is considered for folding when a fold is defined. Sometimes you do not want to fold the entire airbag. For example, if you have done several folds already, on your next fold you may only want to fold the top layer of the airbag rather than the whole airbag. Sets and layers are used to select which bit of the airbag you want to fold. They work in different ways. Normally you will only need to use one or the other but both can be used together if needed.

Using Sets



A set is a collection of shells from the origami. If a set is defined then rather than folding the entire origami, just the shells in the set will be considered for folding. The set could contain a single shell from the origami or it could contain the whole origami.

There are two methods for selecting sets: **Basic select** and **Advanced select**. It is strongly recommended that you use **Basic select** as this is much simpler. If this cannot do what you need (which is unlikely) then use the **Advanced select** option.

The sets works by using sets of shells. These can be created inside the airbag folder, or if your model contains sets already they are available for use. To pick an existing set use the **PICK EXISTING SET** button. You can then choose from the list of available sets.

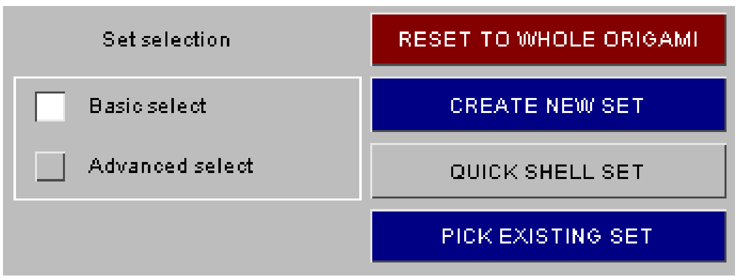
To see the current set selection you can use the **SKETCH** button. This will sketch which shells are in the set on the graphics window. You can reset the selection to the whole origami by pressing **RESET TO WHOLE ORIGAMI**.

To create a new set you can use **CREATE NEW SET**. This will start the standard set creation panel. Once the set has been created it will automatically be chosen for folding.

Alternately you may use **QUICK SHELL SET** to create the set at highest label + 1.

It is important to realise that just because a shell is in a set does not mean that it will definitely be folded. The shells that are in the set are the ones that will be considered for folding. As an example consider the first fold of an airbag where we have defined the set to be the entire origami. When the fold is actually performed the folder checks all the shells in the set. Some of these shells will be on the wrong side of the fold line and will be left in their original positions. Only the ones which are on the correct side of the foldline are actually folded. So even though all the shells were considered for folding, the folder actually only folded the shells on the correct side.

Quick set creation



The **QUICK SHELL SET** button allows you to create a shell set in a much quicker way than the normal creation method.

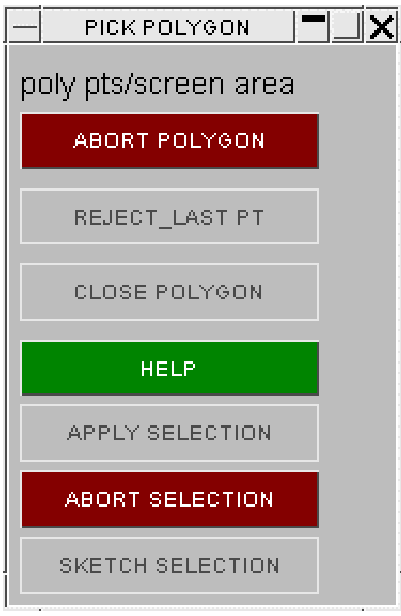
A shell set is automatically created for you with the next free label.

Shells can be added or removed from the set by the following methods

1. Clicking and dragging with the left mouse button will add shells to the set by box.
2. Multiple left mouse button clicks will add shells to the set by polygon.
3. Clicking and dragging with the right mouse button will remove shells from the set by box.
4. Multiple right mouse button clicks will remove shells from the set by polygon

To finish selecting the shells and create the set, press the **APPLY SELECTION** button.
When selecting by polygon, if you close the polygon by clicking back onto the first point, the shells will be added.

Alternatively you can press **CLOSE POLYGON** to select the shells. **REJECT LAST PT** will delete the last point in the polygon. **ABORT POLYGON** will restart the polygon.



Advanced set selection

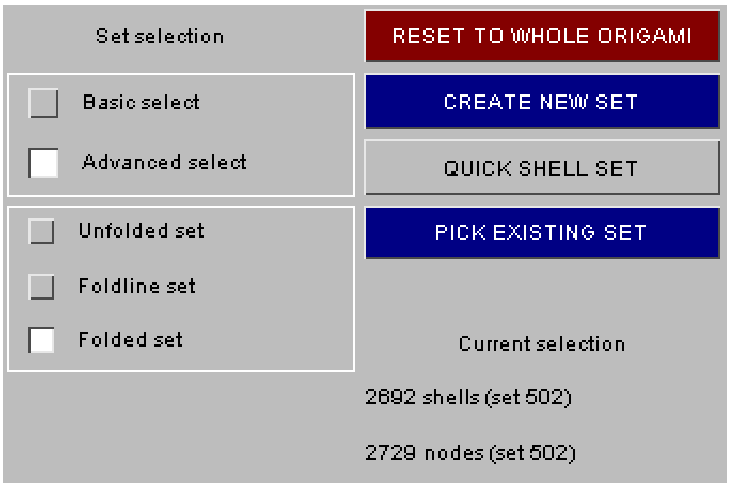
The advanced set selection works with sets just like the basic set selection. The difference is that instead of using a single set (basic select), 3 sets are used.

When a fold is performed there are three distinct regions of the fold.

The **Unfolded set**. The shells that will not move. i.e. they will be unaffected by the fold.

The **Foldline set**. The nodes which are actually on the fold line.

The **Folded set**. The shells that will be folded. i.e. the shells that will move during the fold.



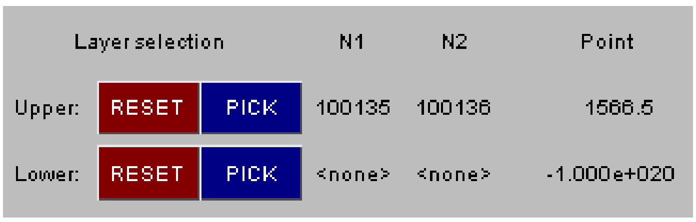
Each of these 3 sets can be selected individually. They can be completely different sets.

It is important to realise that the logic from the basic selection method still applies here. The folded set contains the shells that will be considered for moving during the fold. The foldline set contains the shells (actually the nodes from these shells are considered) that will be considered for the fold line etc.

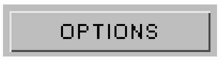
In actual fact the basic selection method works by setting all 3 sets to be exactly the same. It is then the folder which works out which shells should be folded, which should be left in place and which nodes are on the fold line.

Layers

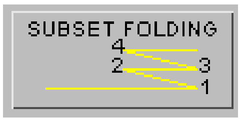
By default all layers (through the thickness of the airbag) will be folded, but sometimes it is convenient to restrict this to only layers within a given +/- Z coordinate.



Upper and Lower layer selection define respectively the upper (positive) and lower (negative) Z limits within which material will be folded. Selecting **PICK** from either prompts you for two nodes, and the Z limit used is set to the average of these. (The reason for having two nodes is that you need to define a point between the outermost layer to be folded, and the layer beyond that, and usually there will only be empty space there!). The default values that are used for the upper and lower limits are 1.0e+20 and 1.0e-20 respectively so by default the whole origami (or set) will be folded. **RESET** can be used to set the lower/upper layer back to this default. In this example the upper layer is 1.0e+20 and the lower layer is -0.3 so anything which is less than Z=-0.3 will not be considered for folding.

Airbag Folder options 

Various options can be set for the folder. See section 6.1.11

Subset Folding 

Subset folding can be used to quickly create folds. See section 6.1.10

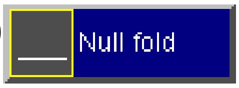
Fold Creation

The following process should be followed to create folds:

1. Create a **NEW** fold. By default a point point at 0 and left-right folding is done.
2. Fold angle: Set the fold angle by using the popup button to choose one of the preset angles or a custom angle. As you change the angle, the highlighted nodes on the screen change.
3. Fold direction: Decide on whether material is being folded in the forward or reverse direction. As you change the direction the highlighted nodes on the screen will change accordingly.
   Decide on whether folded material should finish above or below unfolded material. The top of the bag is defined as being in the direction of increasing Z.
4. Select material to be folded if necessary: ie define **Upper** or **Lower** layers and/or a set in the **SETS AND LAYERS** menu. As you change the layer or set selection the highlighted nodes on the screen will change accordingly.
5. Decide on the fold type (**Thin**, **Thick**, etc) - see Section 6.1.9.
6. On pressing the fold type required PRIMER folds the ORIGAMI. This may require adjustment of parameters (Separation, Tolerance) to achieve a good result.
7. Continue creating new folds until the complete ORIGAMI is folded.

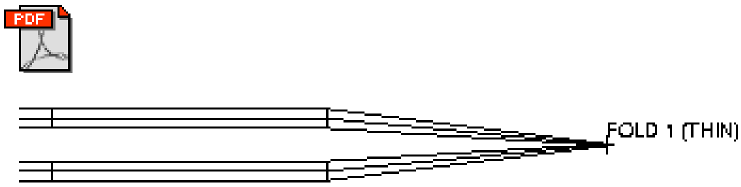
6.1.10 Fold types

Null Fold (Set attributes, but don’t fold mesh)



The **NULL** fold allows the user to define all the fold data without actually folding the ORIGAMI. This is useful when the current fold appears to be incorrect (eg too many layers have been folded). Pressing the NULL fold option effectively unfolds the current fold and allows the user to change the fold parameters (eg entity selection). The nodes which will be folded are still highlighted

Thin Fold (Perform a sharp "crease" fold)

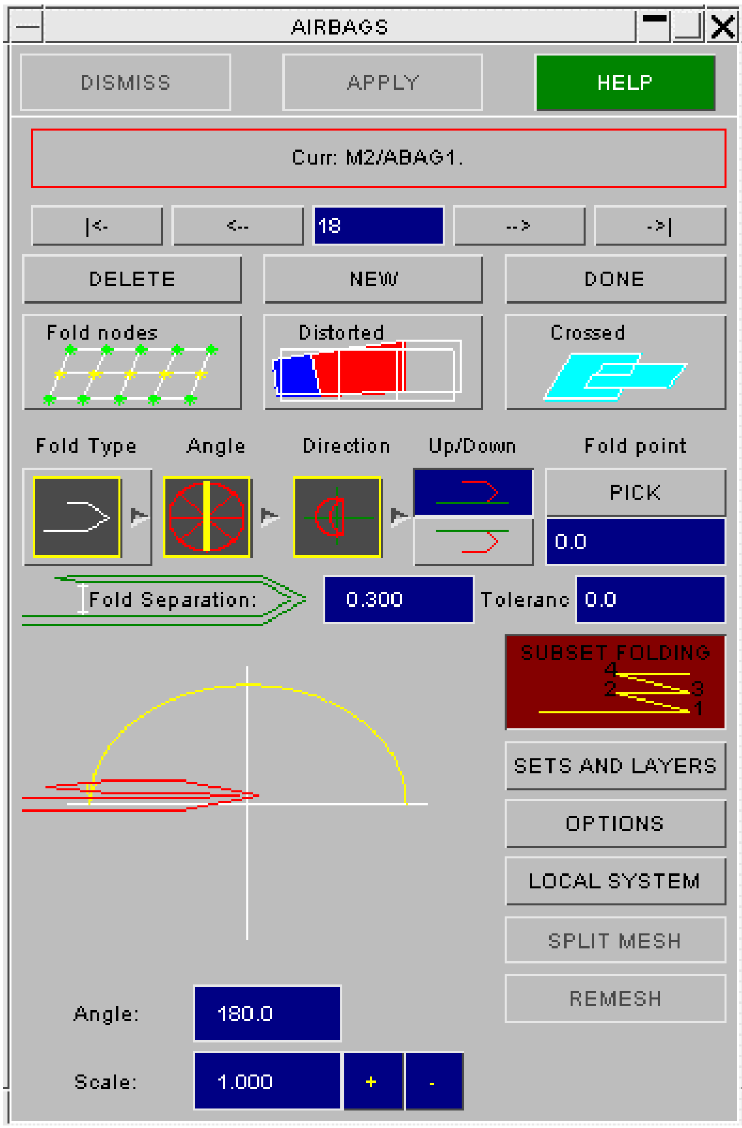


For airbags, the most common fold type is the thin fold. Material is creased sharply along a line of nodes to give a precisely defined shape.

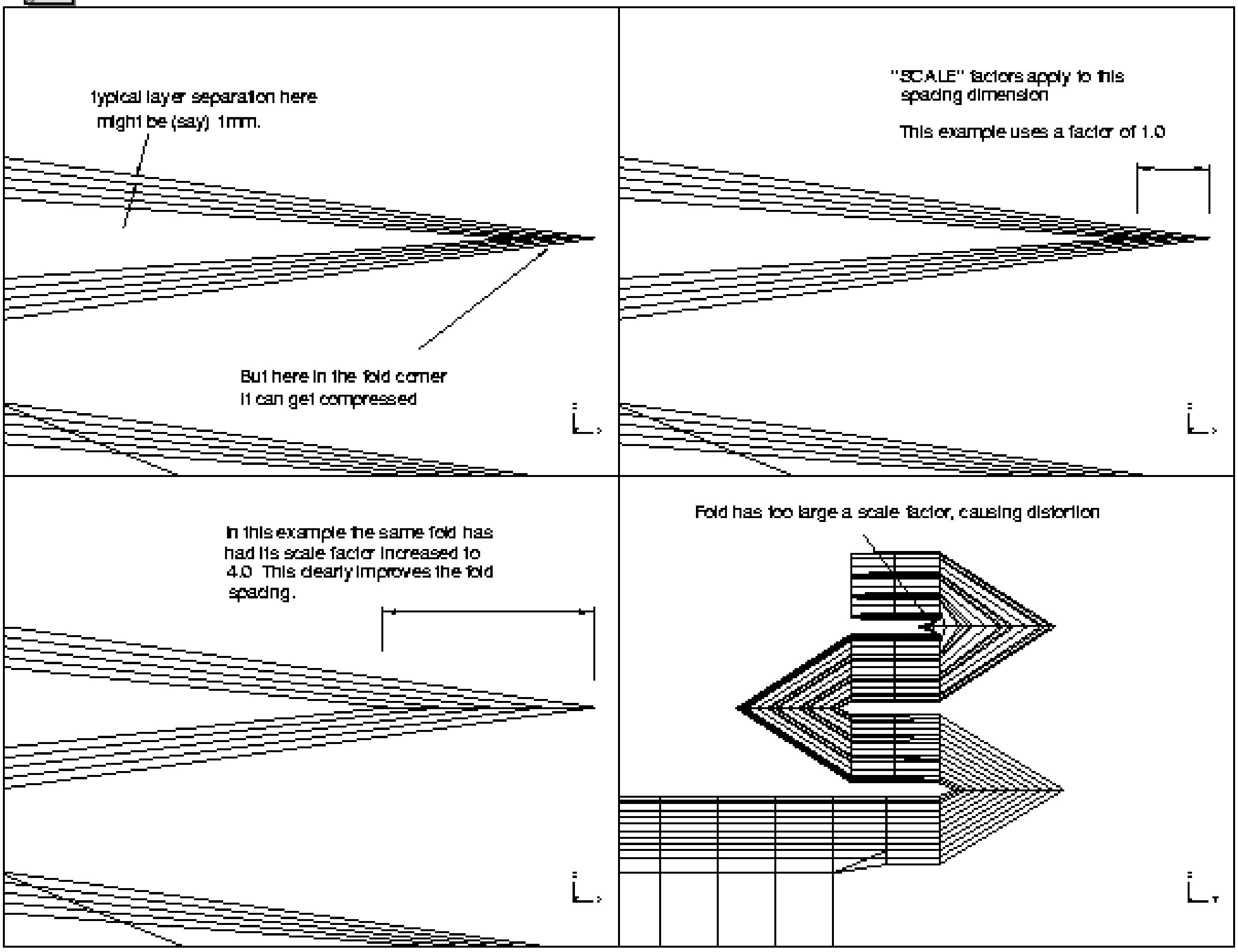
The definition and control panel for **thin** folds are shown in the figure below:

The panel shows the creation of a thin fold. the airbag has been folded from right to left - the right hand side ending up on top of the left hand side. The fold thickness is 0.3.

Typical thin folds, have a total fold angle of 180º with the centre portion rotated 90º though this is not a requirement. Angles larger than 180º are not permissible. These defaults are being used here. The thin fold graphic above shows that there is some pinching at the fold tip. The pinching can be reduced by increasing the **Scale** option to spread the layers at the fold tip.

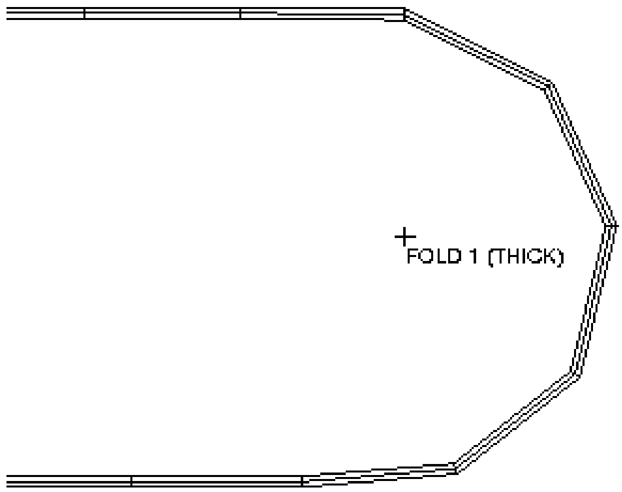


The figure below shows the effect of **Scale** on typical thin folds. It is generally desirable to try to select a value which causes the inner and outer material surfaces to be parallel, as this will lead to the contact algorithms being more reliable. However, the user should take care that this doesn't lead to gross local element deformations: as in the bottom right quadrant of the figure. This can also be visualised by using the **CROSSED** element button. If there are problems with element penetrations this will easily show it. Changing the scale factor will automatically update this plot so you can tell when any penetrations are eliminated.



By default all the elements and nodes selected for the Origami definition are included in the fold, but it is possible only to operate on a sub-set of these. The **SETS AND LAYERS** button can be used to select a subset to fold

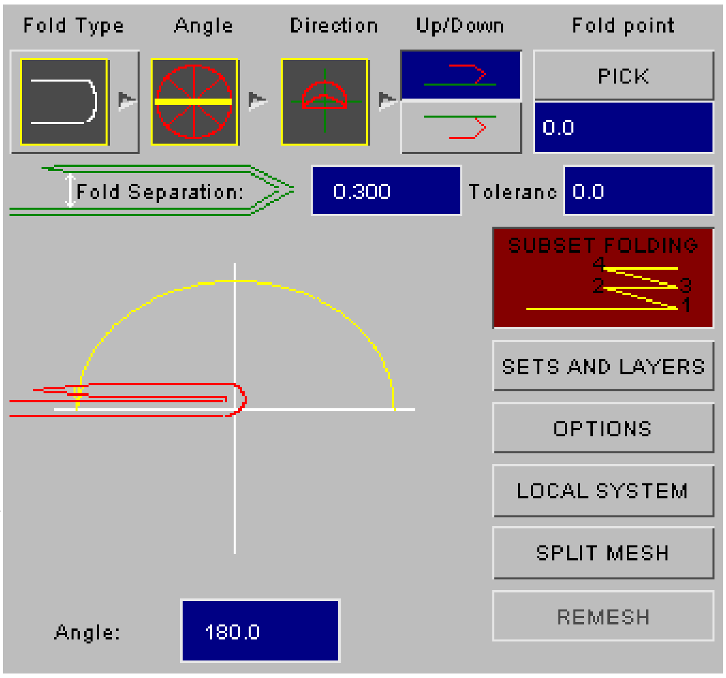
Thick Fold (A radiused fold spanning > 1 element) 



Thick folds are used when the mesh is extremely fine or when there are a large number of layers to be folded. In general the element size should be smaller than the fold radius or it is unlikely that a satisfactory fold will be created.

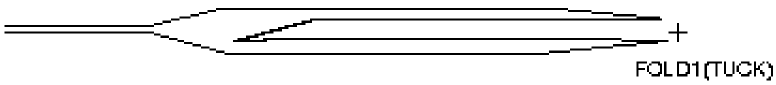
As this figure shows the effect is to create a radiused fold, using several elements, giving an effect similar to rolling the mesh round a circular former of the given radius.

The adjacent figure shows the fold creation menu - the material has been folded from right to left and the folded material is folded on top of the unfolded material. The only option which is available for a thick fold is the angle. In this figure the angle is set to 180º . The thick fold does not have a precise centre like the thin fold, but it does allow for arbitrary angles of orientation.



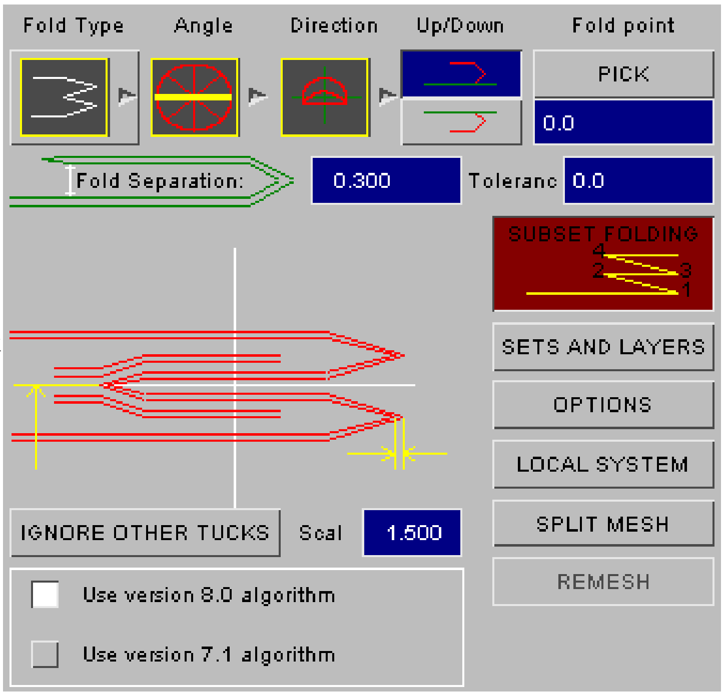
Usually, a fold point is located towards the packaging limits of a airbag. In this fold, the fold point is offset ahead of the centre point for the fold by the radius. The radius is equal to half the separation distance between layers. If more precise control over the selection of elements for this fold is needed, then the **advanced set selection** option in **SETS AND LAYERS** can be used to select the portion of the airbag to be folded. The **unfolded** set is used only to determine clearances in this fold type.

Tuck Fold (A thin fold tucked into the mesh centre)

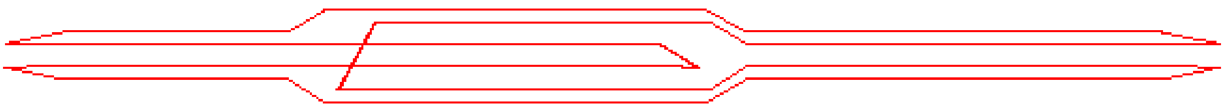


The tuck fold is also common in airbags. The material is folded inside the outer layers to form a "tuck". ("Up" and "Down" have no meaning here.)

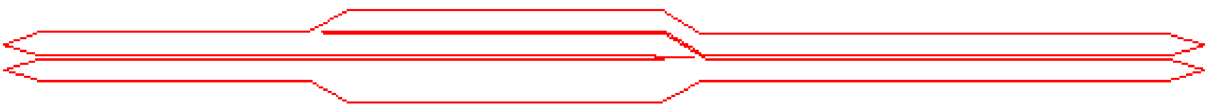
The adjacent figure shows the tuck fold creation panel.



In version 8.0 a second tuck fold algorithm has been added. This is not meant to replace the version 7.1 tuck fold as there will be situations when the version 7.1 fold will perform better than the version 8.0 tuck fold. However the new version 8.0 tuck fold will perform much better in situations where two tuck folds interfere with each other. To illustrate the point the next two figures show an cross section through an airbag with 2 interfering tuck folds (one from each side of the bag) folded with the version 7.1 tuck fold and the version 8.0 tuck folds.



Two interfering tuck folds using the version 7.1 tuck fold algorithm

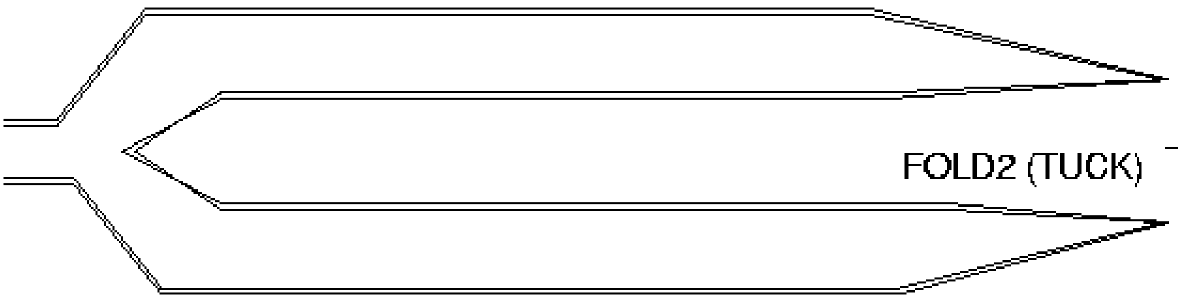


Two interfering tuck folds using the version 8.0 tuck fold algorithm

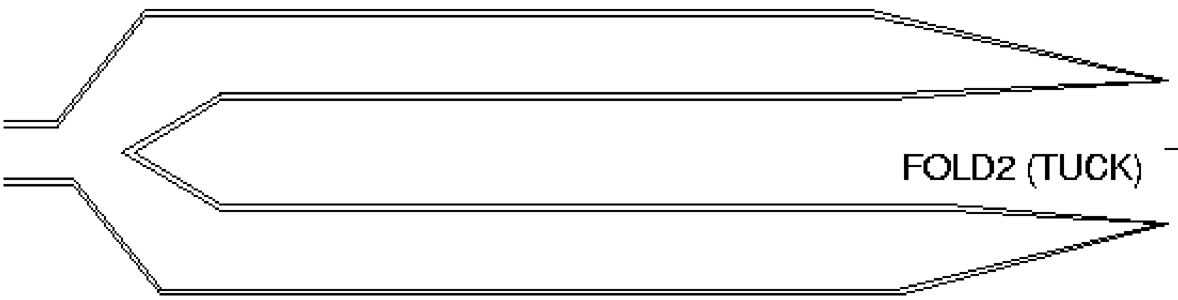
The tuck folds using the version 7.1 tuck folds penetrate through each other. If this airbag was deployed there would be problems with contacts and the airbag forming knots. Additionally the nodes on the fold tip are not in the correct place. The version 8.0 folds do not penetrate through each other and so this airbag will deploy correctly and the nodes at the tip of the fold are still in the correct position.

The default for tuck folds is to use the version 8.0 algorithm. If the fold cannot be performed with this algorithm you can still use the older 7.1 algorithm. This may be better for tuck folds which use multiple material thickness as there are some additional options which may help.

The following two figures illustrate the use of these options for the version 7.1 algorithm, the left hand figure shows that problems can occur with penetrations when using tuck folds for multiple layers. If problems occur then selecting **[>>]** (double layer mode) may help resolve the problem (right hand figure). But, the double layered mode is only valid if the fold tip lies along a line of nodes. If it does not then the single layered mode should be used.



Penetrations at tip

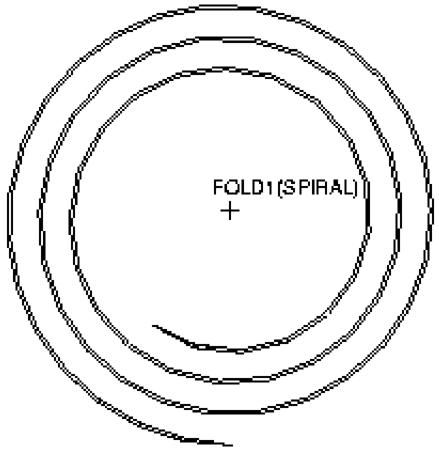


No Penetrations

By default, the folder attempts to locate the middle fibre of the unfolded material. Everything above the middle fibre is pushed up and everything below the middle fibre is pushed down so that the tip can be inserted and clearance maintained. This can be overridden if PRIMER selects the wrong location using **ZSPLIT** which prompts the user to pick two nodes. These define a plane whose normal vector starts mid-way between these nodes. The layers are then separated above and below this plane.

SCALE allows the user to reducing the pinching that occurs at the fold tip by increasing the node separation.

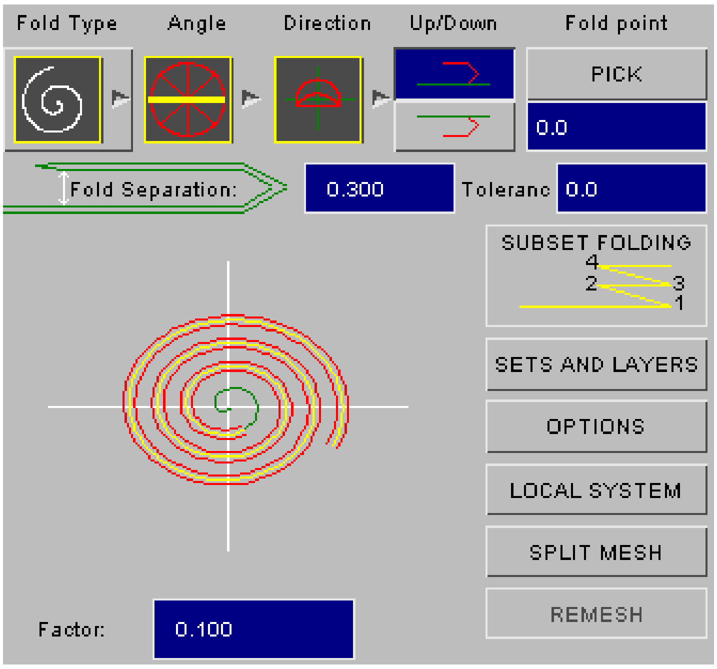
Spiral Fold (Rolling layers into a spiral)



The Spiral fold is used to roll up a flat bag.

An Archimedean spiral (radius is proportional to angle) is used, and PRIMER attempts to keep the characteristic element length constant at the middle fibre of the bag.

The adjacent figure shows the fold definition and options menu for the spiral fold.



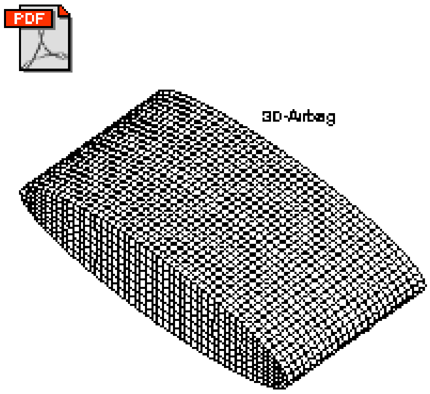
Controlling the spiral internal radius

PRIMER tries to preserve a constant arc length for the middle fibre of the bag as it is rolled, which presents problems at the spiral centre where the radius tends to zero and will always be smaller than the elements.

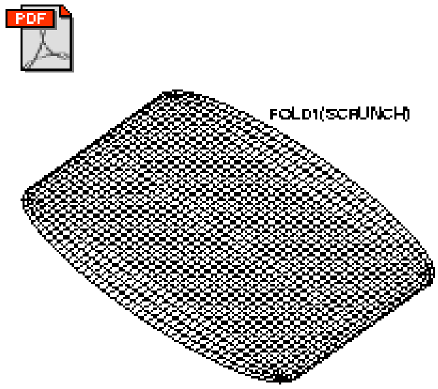
In practice a rolled airbag has a finite thickness and therefore will not use the early portion of this curve. PRIMER uses a **Factor**, of the original arc length of the airbag to specify that portion of the curve that is not to be used. For example, a Factor of 0.5 increases the total arc length to 1.5 times the airbags arc length and leaves the first 0.5 times this arc length unused. A factor of 0 would have no unused portion and the airbag would be rolled from the spiral centre. The options menu shows how much of the spiral is mapped and how much is unmapped. The default factor is 0.1.

The user can also specify a subset of the airbag to be folded using the **SETS AND LAYERS** options as for other types.

Scrunch Fold (Compressing a 3D bag to a flat shape) 



Before scrunch



After scrunch

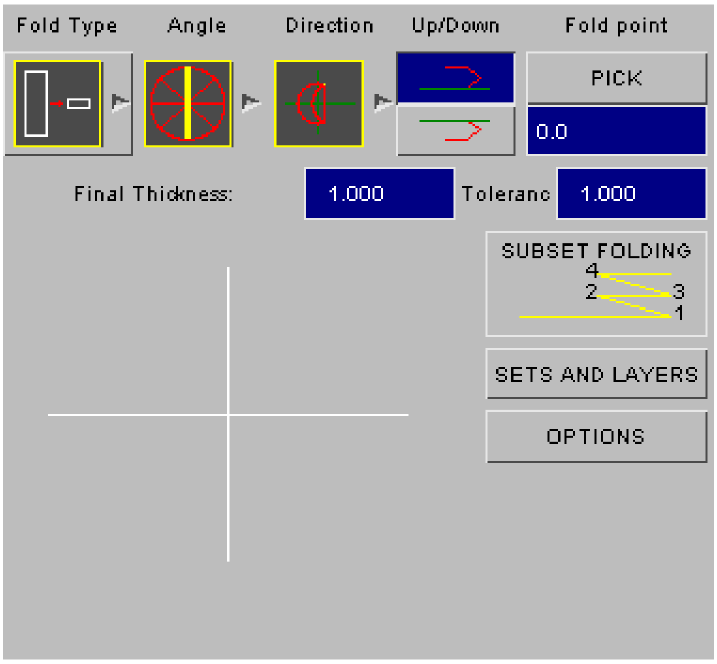
The adjacent figures show a 3D airbag being **scrunch**ed to a flat (2D) shape.

The option of splaying the sides out has been used.

This fold type can accomplish two separate functions:

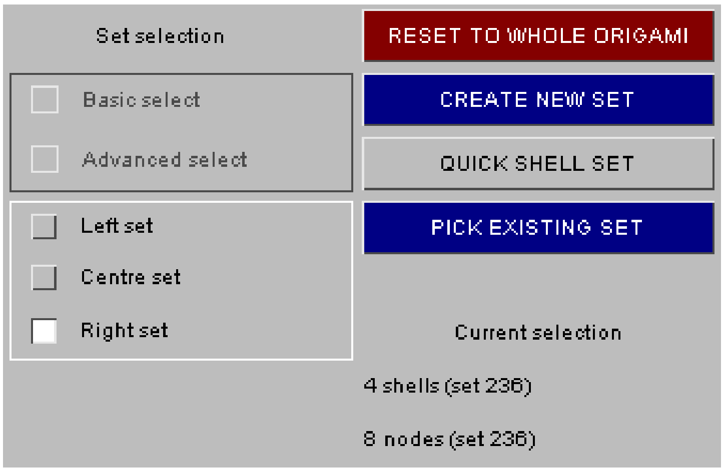
1. It can simply scale an existing bag in the local Z-direction so that it has a smaller final thickness.
2. It can flatten a 3D airbag so that the 2D folder (thin, thick etc) can be used. The bag is reduced in the Z-direction and the sides are pushed out.

The **scrunch** fold definition and options are shown in the adjacent figure.



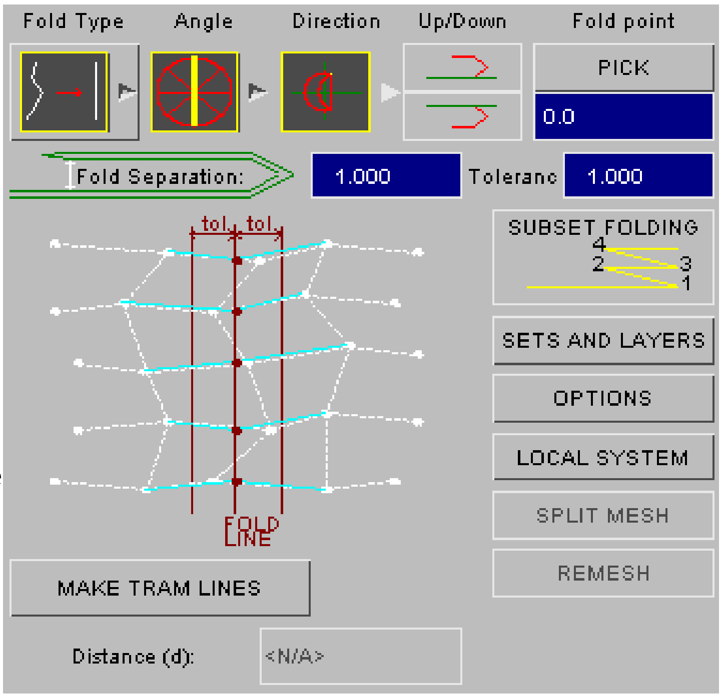
The user must tell PRIMER which elements form the side to be pulled out. This is done using the **Left** and **Right** sets from the **SETS AND LAYERS** menu. If neither of these sets is chosen, then a simple scaling is used. In the case shown above this could lead to the vertical elements having a zero side length (which may not be illegal if airbag reference geometry is used during the analysis).

When forcing the sides outwards, the top and bottom of the bag are located above and below a side node. The node is then pushed outwards based on the nearest distance to the top or bottom. When using this capability for pushing out side walls, it is important how the bag is oriented. The axis of the cylinder must be parallel with the local X-axis. The sides must be in the YZ plane.



FOLD_POINT has no effect here the ORIGAMI is scrunched about the local z=0 plane.

Align Fold (Aligning nodes on a fold line) 

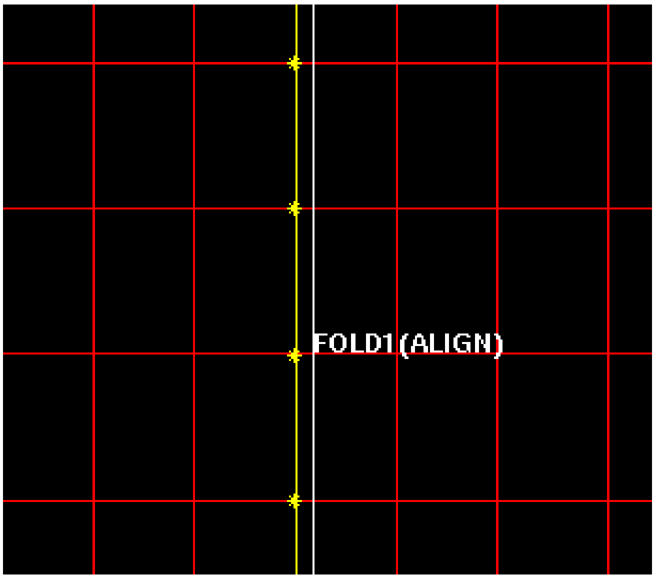


The align fold is used for projecting nodes onto a fold line. This is very useful if the nodes do not line up exactly on a fold line or if the fold line is in slightly the wrong position. The adjacent figure shows the align fold creation menu. In this example the **TRAM LINES** option has been set. This can be turned on or off by using the **MAKE TRAM LINES** button.

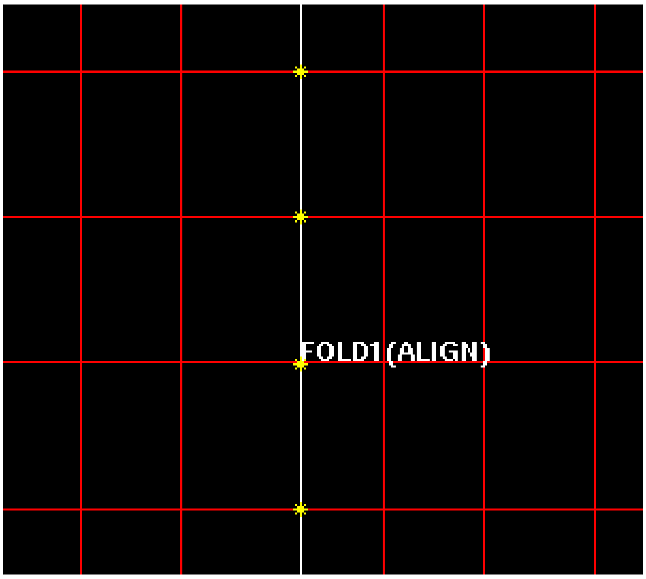
The normal options for selecting the nodes for the align fold can be used. You can select the nodes you want to fold using the **SETS AND LAYERS** options. Alternatively, you can use the nodes which were folded in the previous fold by using the **SUBSET FOLDING** option.

For both options you give a fold point and a fold angle to define the fold line. The nodes which will be aligned are highlighted on the screen. The **Tolerance** option can be used to increase the tolerance for selecting the nodes which will be moved onto the fold line.

The default options for the align fold just move the selected nodes onto the fold line. The following figures show the effect of this option with pictures before and after the align fold.



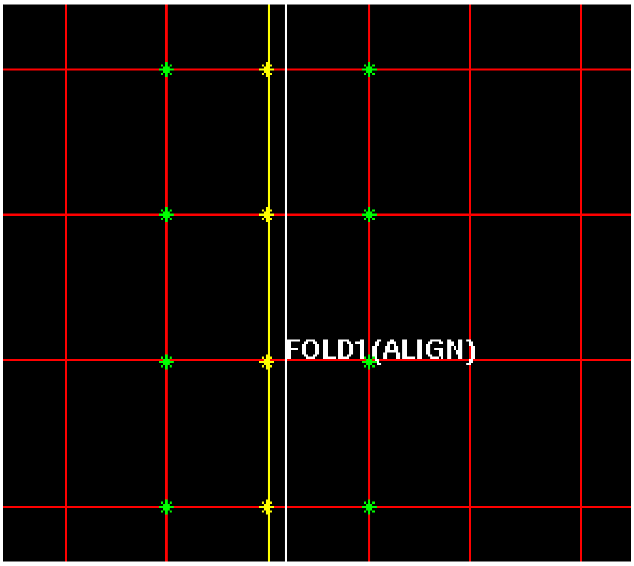
Before align fold



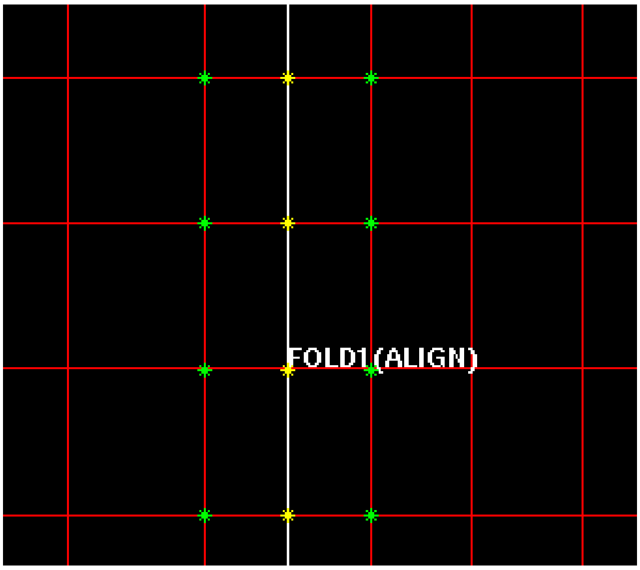
After align fold

Folds work best if the nodes adjacent to the fold line are a constant distance from the fold line. The **MAKE TRAM LINES** option does this. If the option is set then if a node from an element is moved onto a fold line the other nodes on the element which are not on the fold line will be moved to be a constant distance away. You can enter the distance to position the nodes with the **Distance** box. The diagram in the panel updates to show you that you are using the tramlines option.

The following figures show the effect of the tramlines option with pictures before and after the align fold.



Before align fold



After align fold

With the tramlines option set the adjacent nodes are a fixed distance from the fold line. Compare the above pictures to the ones with the tramlines option not set.

Scale Fold 

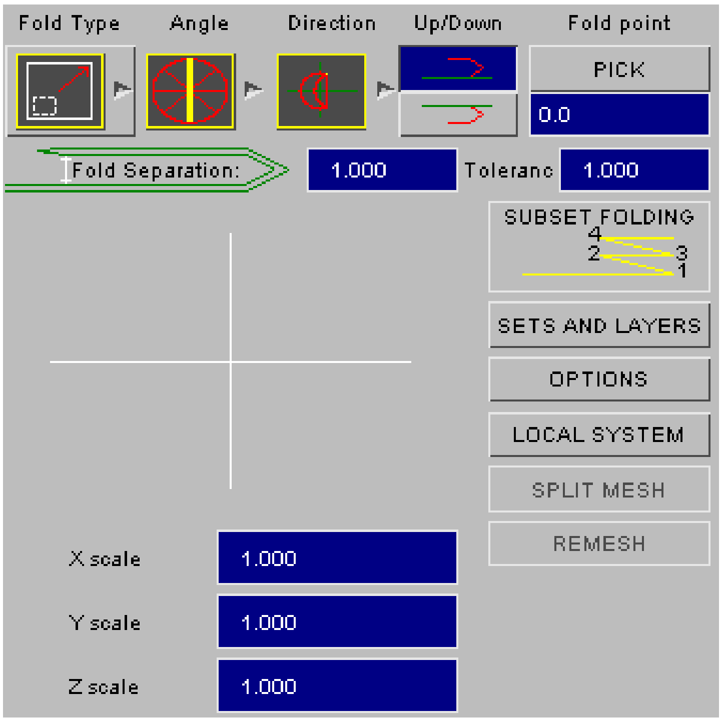
The scale fold is used for shrinking or enlarging parts of the origami.

The scaling is done in the local co-ordinate system of the fold, NOT the global co-ordinate system.

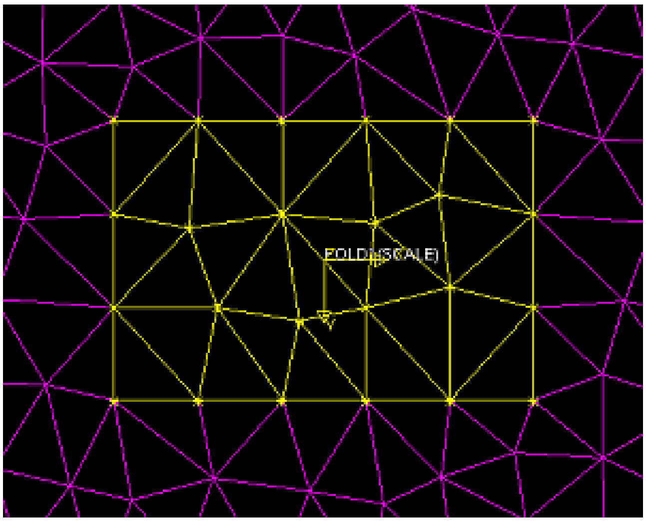
To help you visualise the local co-ordinate system the axis triad is drawn on the airbag.

The nodes/elements that you want to scale are selected in the normal way using sets and layers as required.

You can give separate scale factors for the X, Y and Z axes of the fold.



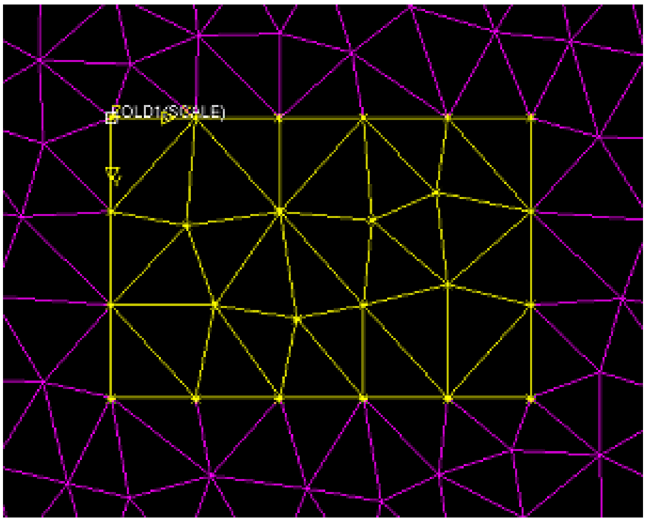
Examples of scale fold



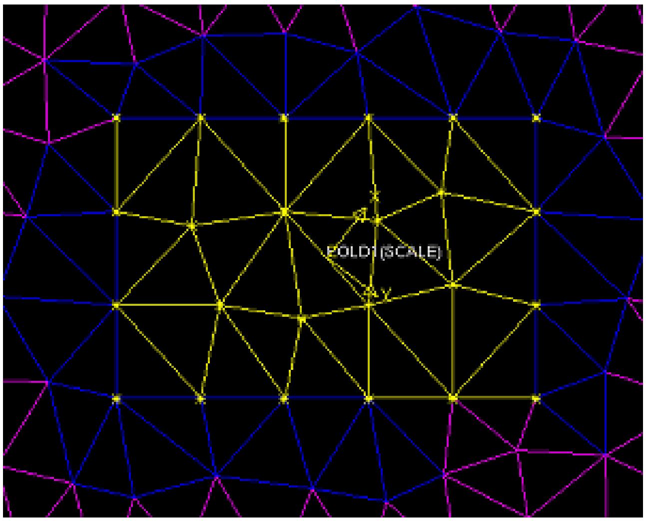
The yellow portion of the airbag has been selected using the sets and layers option (the fold nodes are shown highlighted by yellow stars).

The local co-ordinate system is shown on the diagram.

In this example the fold point is at the default (0) position.



In this example the fold point has been moved to the top left of the yellow portion by **PICK**ing a new fold point.



In this example the fold coordinate system has been changed by using the **LOCAL SYSTEM** function. Note that the directions of the local axes have changed but the origin is still at 0. To change the origin a new fold position would need to be picked.

Translate Fold

The translate fold is used for moving parts of the origami.

The translation is done in the local co-ordinate system of the fold, NOT the global co-ordinate system.

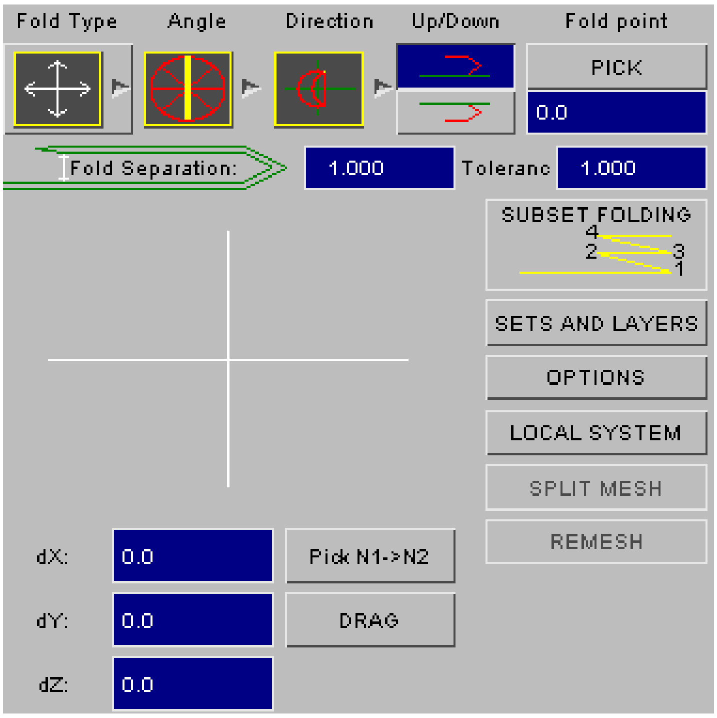
To help you visualise the local co-ordinate system the axis triad is drawn on the airbag.

The local axes can be changed using the **LOCAL SYSTEM**. The fold point (local axes origin) can be changed by selecting a new Fold point. See the examples for the scale fold as the method is identical.

The nodes/elements that you want to translate are selected in the normal way using sets and layers as required.

You can translate the airbag by one of 3 methods:

1. You can type in separate X, Y and Z distances in the text boxes.
2. You can pick 2 nodes using **Pick N1 -> N2** to define the distance.
3. You can drag the selection interactively using **DRAG**.



Press **DRAG**. The window will change as shown on the right. All other functions are unavailable while dragging.

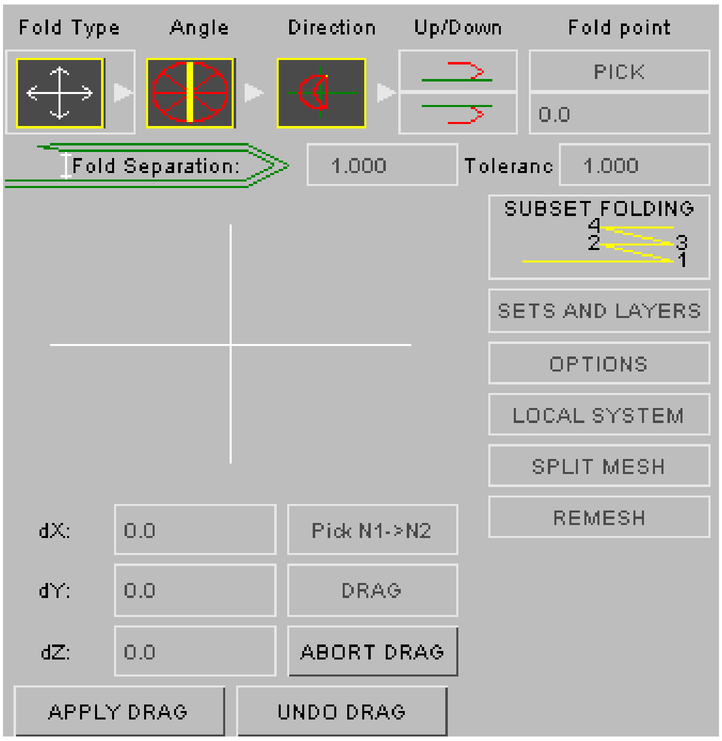
Click and drag with the mouse and the selection will be moved on the screen and the distance boxes updated. When the mouse button is released the screen will be redrawn. Make sure the Fold type button is toggled to show the translate fold type (not the null fold type) or you will not see the translation!

You can only drag in the XY plane of the fold. You can continue dragging the selection until you are satisfied.

You can cancel dragging at any time by pressing **ABORT DRAG**.

You can start dragging again (undoing the last drag you made) by pressing **UNDO DRAG**.

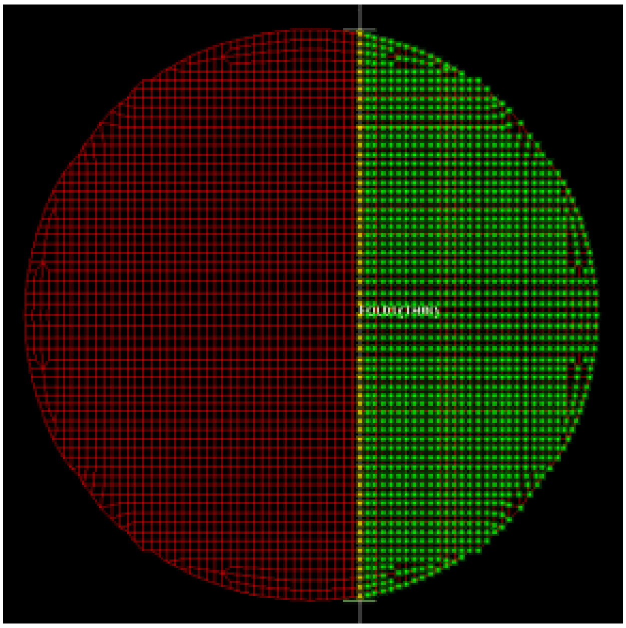
To finish the drag operation press **APPLY DRAG**. The distances will be saved.



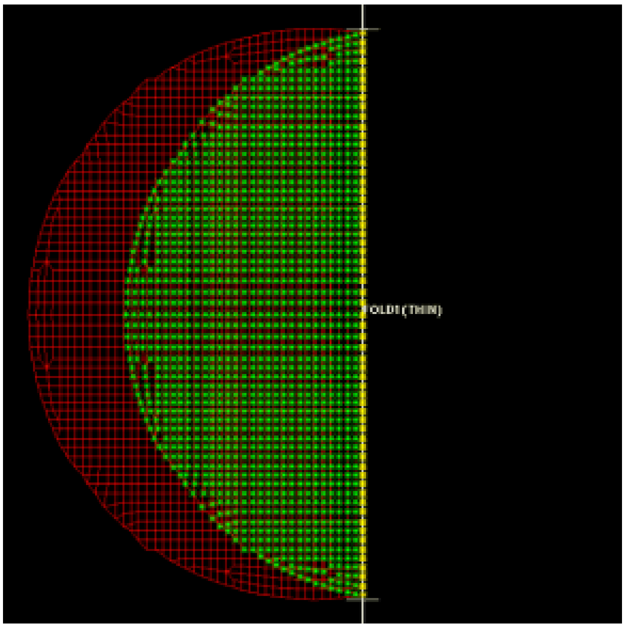
6.1.11 Subset Folding

Subset folding can be used with thin, thick, version 8.0 tuck folds and align folds. It can make the process of making multiple folds considerably easier. Every time you do a fold a list of the nodes which are folded is saved. If you now want to do a new fold in which the nodes you want to fold are a subset of the previous fold (i.e. all the nodes were folded in the previous fold) instead of defining a set or layers you can just press the **SUBSET FOLDING** button and the nodes from the previous fold will be used as the input to this fold. The following figures illustrate the use of subset folding when creating thin folds.

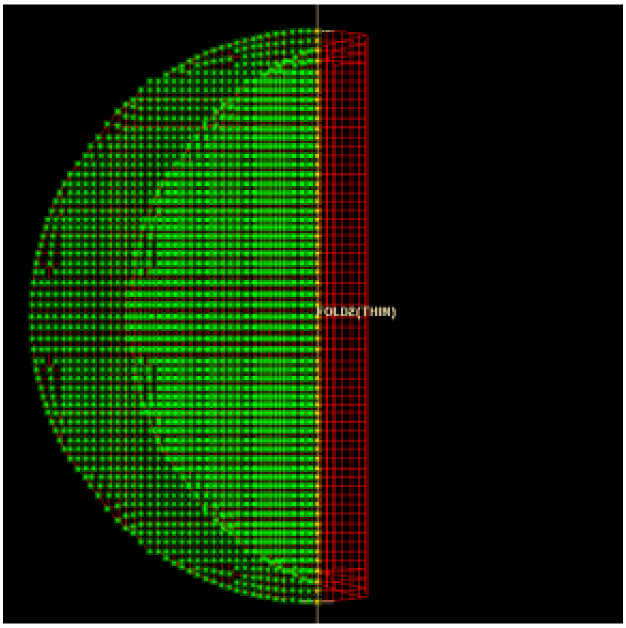
a)



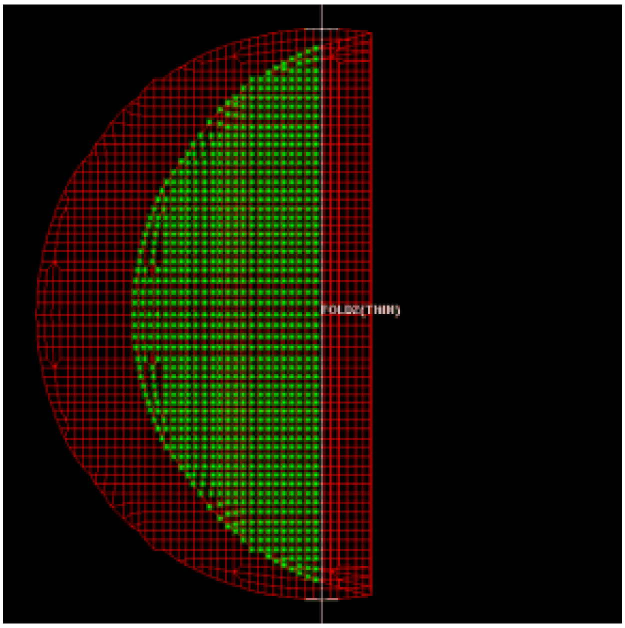
b)



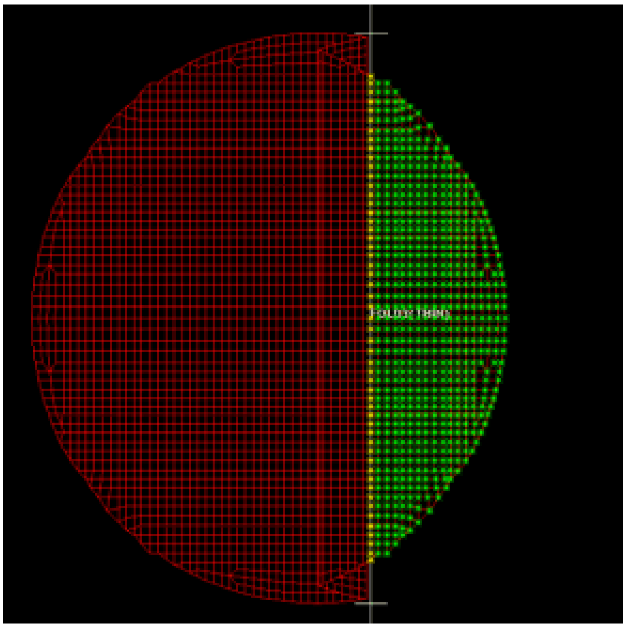
c)



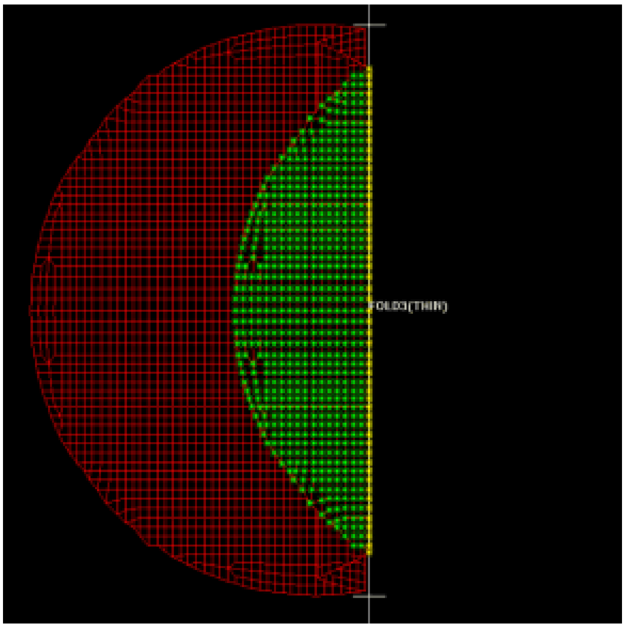
d)



e)



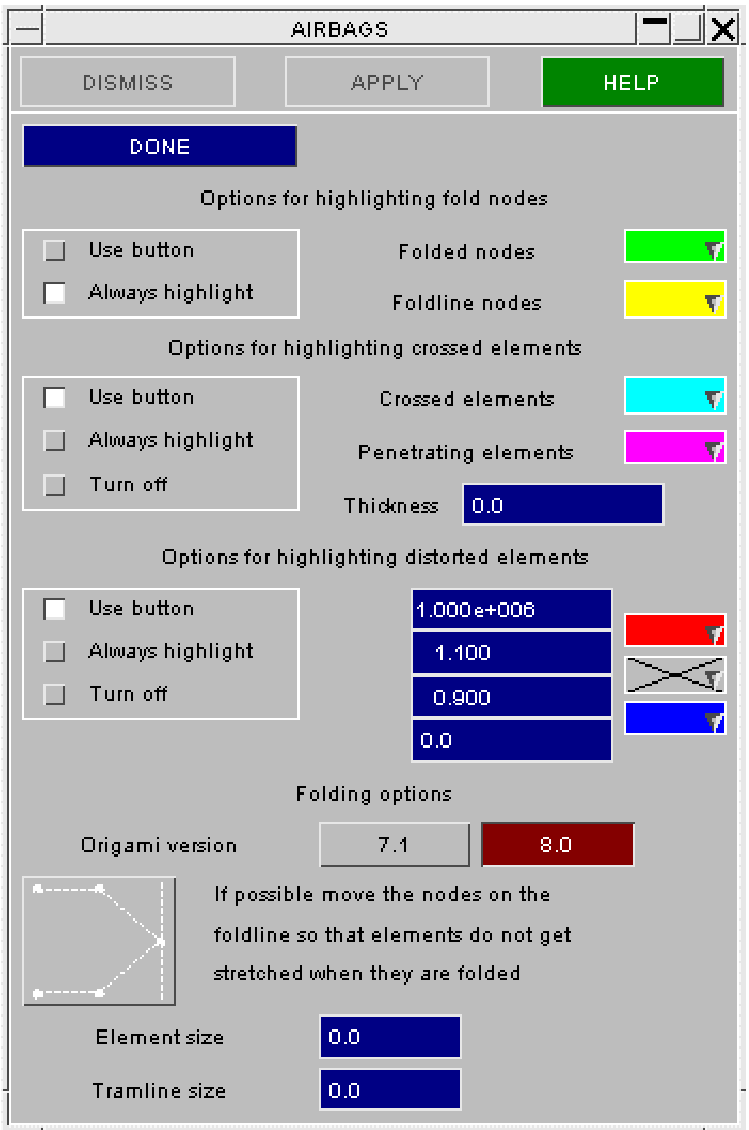
f)



A new fold (fold 1) is created on the airbag folding from right to left. a) shows the airbag before the fold, b) shows the airbag after the fold. If a new fold (fold 2) is now created folding from left to right by default everything to the left of the fold point will be folded (c). At this point you could define a node set or use layers to select the upper layer of the airbag to fold. Instead you can press the **SUBSET FOLDING** button. Now the nodes which will be used for fold 2 are the nodes which were folded in the first fold (d). If this fold is completed and the folding process continued with subset folding then when the third fold is defined (e) the nodes are automatically selected and the fold direction swapped from forward to reverse. (f) shows the airbag after the third fold. If the folding process was continued and a fourth fold created using subset folding the fold direction would automatically swap over to be forward. This process can be used to very quickly create >zig-zag= folds on an airbag. It is much quicker as no sets or layers need to be created when doing the folds. You can turn subset folding off at any time and continue folding by the normal methods.

6.1.12 Folder Options

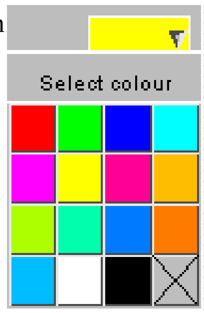
The **FOLDER OPTIONS** panel enables you to set various options which alter the way the folder works and what is drawn.



For fold nodes, crossed and penetrating elements and distorted elements you can individually choose to always plot them or to only plot them if the appropriate button is pressed. This option is set by using the radio buttons and setting the option **Use button** or **Always highlight**.

As the distorted and crossed element checking is very complicated it can take some time. If the delay when folding is unacceptably slow then these features can be turned off by using the **Turn off** option. If this is done then you will not be able to plot the distorted or crossed elements until one of the other options is chosen again.

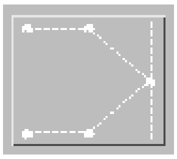
For each type of entity (e.g. Foldline nodes) you can also choose the colour that the entity is drawn in (or if it drawn at all) by using the popup menus. Each popup menu brings up a selection of colours to choose.



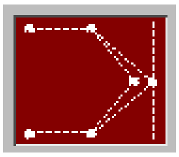
The colour popup allows you to choose from 15 basic colours. The cross at the bottom right of the panel (the X button) stops the entity being drawn completely. E.g. in the above figure, element distortion between 0.9 and 1.1 will not be highlighted.

The element thickness which is used for checking penetration can be altered by typing in a new thickness in the text box. The ranges for distorted elements can similarly be changed by typing new values in the text boxes.

Version 8.0 has some new fold options such as a new tuck fold algorithm and an element stretching option. As these options could change the way an existing origami is refolded there is an option to say if your origami will be folded using version 7.1 techniques or version 8.0 techniques. By default any new origami you define will be a version 8.0 origami and you will have access to these new functions. If you read in an old origami (version 7.1 and before) the origami version is set to be 7.1. A warning will be output when you first select the origami to say that you are using an old origami. The reason a warning is given is to bring to your attention that if you change the origami to be version 8.0 the folds may change. If you have a previously correlated airbag then leave the version at 7.1 and the folds will not be changed. You will not be able to use the new functions until the origami is changed to be version 8.0.



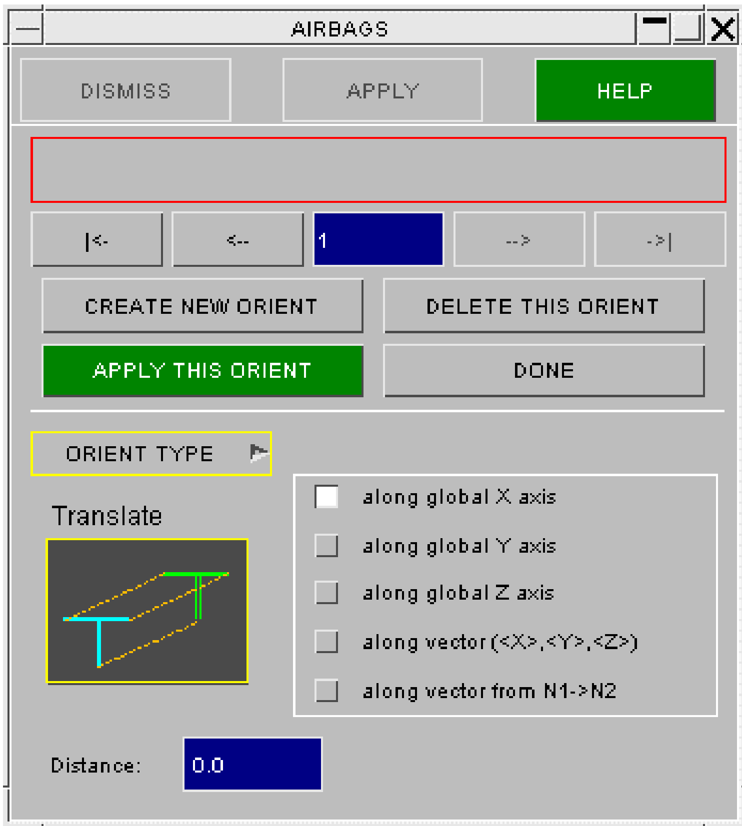
The final button on the panel is used to try to stop stretching of elements during folding. This only applies to thin folds. By default elements will be stretched as the centre plane of the elements which are being folded is placed at the fold point. Elements on one side of the plane will get smaller, elements on the other side of the plane will get larger.



If the button is pressed then the folder will attempt to stop any elements from stretching. To do this the fold point has to be moved slightly. For folds which are less than 180º this can generally be done. If the fold is a 180º fold then if you try to fold multiple layers which are thick a point is reached when the fold cannot be done without stretching any elements. If the thickness is less than this then the stretching will be eliminated. If the thickness is more than this then the stretching will be reduced as much as possible.

6.1.13 Positioning Folded Airbags

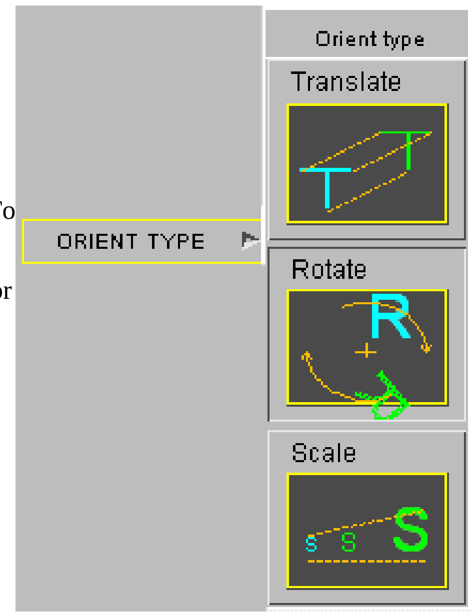
Once the airbag has been folded it can be positioned where you want it by using the **POSITION FOLDED BAG** option. The reason for using this rather than the normal **ORIENT** option in PRIMER is so that you do not lose the orientations if you refold the airbag. The orientations are created and viewed just like folds so you can create, delete or edit any orientation. They are cumulative transformations on the folded airbag. For example if you define a translation and then a rotation first the translation will be applied and then the rotation so the order of the orientations is important.



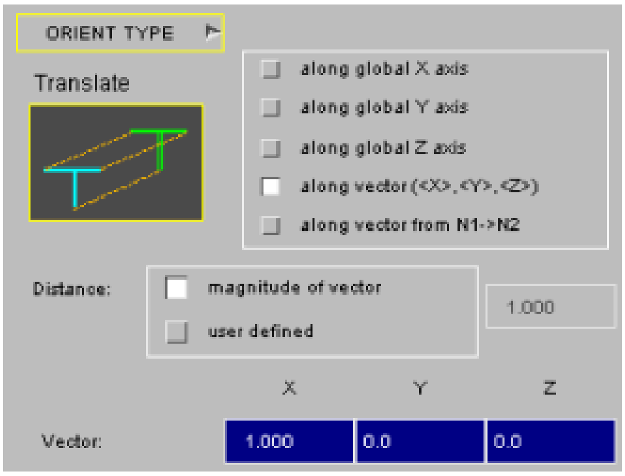
The left hand figure shows the initial window when you enter the positioning section. As there are no orientations the top row of buttons are greyed out. As orientations are defined these buttons can be used to move backwards and forwards through the orientations in exactly the same way as folds in the **SET FOLD** menu. The **DONE** button will return to the main folding window. To create a new orientation the **CREATE NEW ORIENT** button can be used. The default new orientation type is a translation (right hand figure).

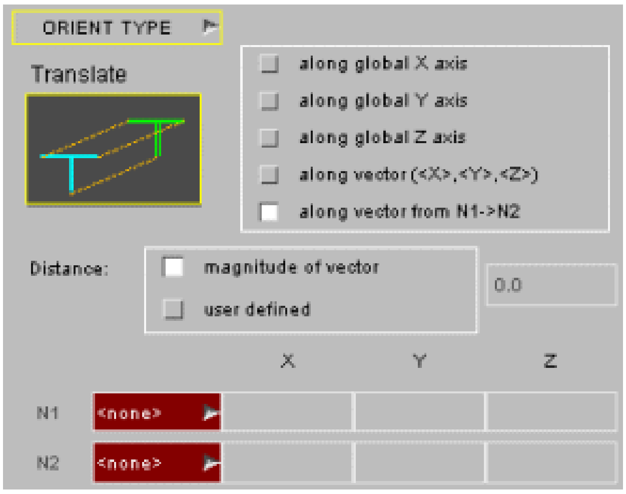
At any time an orientation can be deleted by using the **DELETE THIS ORIENT** button. Once the necessary values have been given for the orientation (for example the translation distance) the orientation can be applied by using **APPLY THIS ORIENT**. Once applied the button will change to **UNDO THIS ORIENT**. This button can be used to continually toggle between the unapplied and applied view of the orientation.

Three different orientations are available; translation, rotation and scaling. To change an orientation type use the top row of buttons (**|<** , **<** , **>** , **>|**) to select the orientation number to change and then use the **ORIENT TYPE** popup menu (figure on right) and choose either **TRANSLATE**, **ROTATE** or **SCALE**. The different options for translation, rotation and scaling are described in the following sections.



Translation





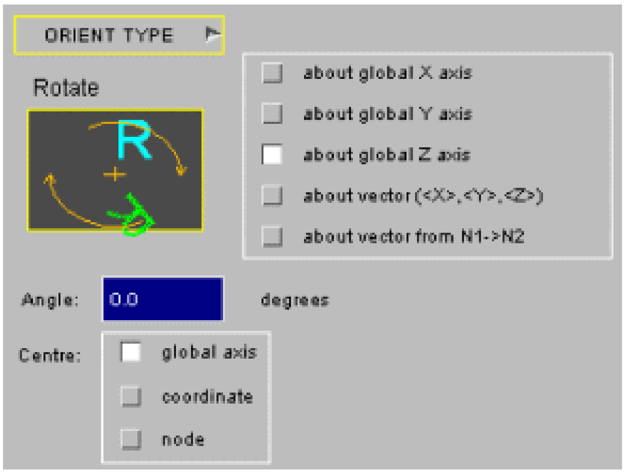
Translation along a vector <X>,<Y>,<Z> Translation along a vector from N1->N2

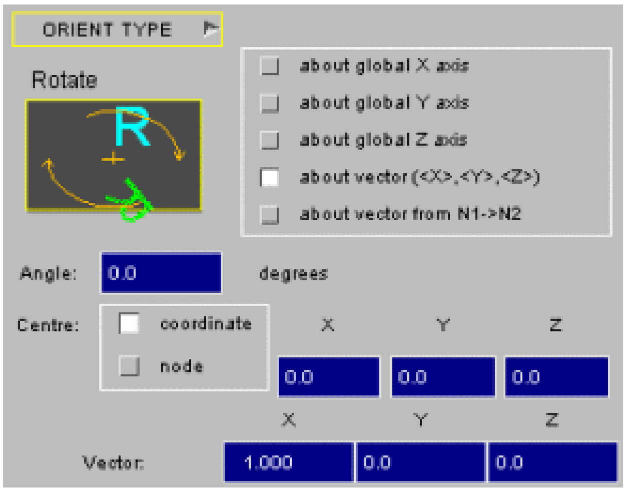
There are 5 different translation options. The first 3 are translating along the global X, Y or Z axis. For these options the translation distance must be given by typing the value in the text box.

The fourth option allows you to give a vector to translate along by typing in the X, Y and Z components of the vector (left hand figure). The distance that the airbag is translated along this vector can either be a user defined distance or the magnitude of the vector.

The fifth option translates the airbag along a vector defined from N1 to N2 (right hand figure). The 2 nodes can be typed in or picked using the popup menus. The translation distance can either be a user defined distance or the magnitude of the vector.

Rotation





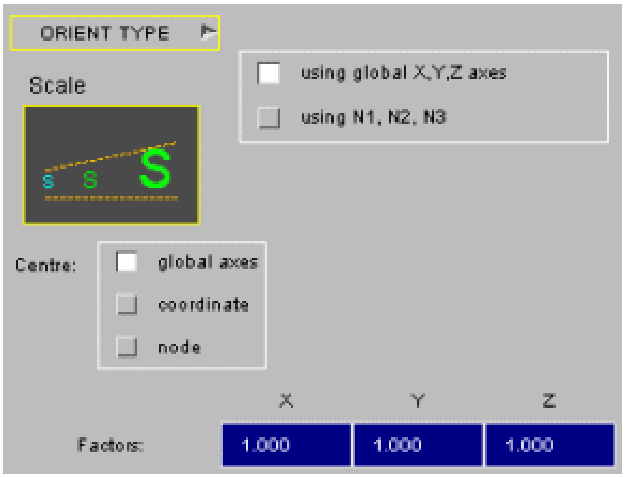
Rotation about a global axis Rotation about a vector

There are 5 different rotation options. The first 3 are rotating about the global X, Y or Z axis (left hand figure). For these options the rotation angle must be given by typing the value in the text box. There are 3 possible methods for specifying the centre of rotation. The centre can be the global axis, about a coordinate which you can type in, or about a node number which you can type in or select using the popup menu.

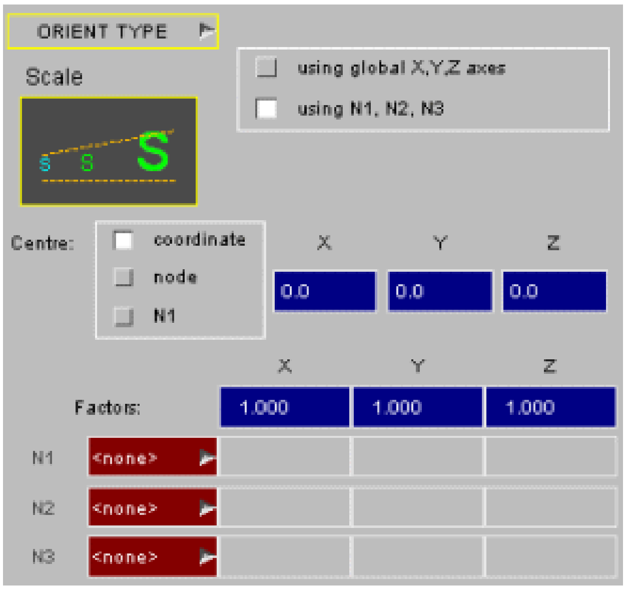
The fourth option allows you to give a vector to rotate about by typing in the X, Y and Z components of the vector (right hand figure). The centre of the rotation can either be a coordinate or a node.

The fifth option rotates the airbag about a vector defined from N1 to N2. The 2 nodes can be typed in or picked using the popup menus. The centre of rotation can be a coordinate, N1 on the vector or another node.

Scaling



Scaling using global axes



Scaling using local axes

There are 2 methods available for scaling the airbag. The first method allows you to scale the airbag in the global axes (left hand figure). Different scale factors can be used for the X, Y and Z directions if necessary. The centre for the scaling operation can be defined as either the global origin (0, 0, 0), a coordinate which you can specify by typing in the X, Y and Z values or a node number which you can pick or select by typing the number.

The second scaling method allows you to scale an airbag in directions other than the global axes by using three nodes. The three nodes are used to define a local coordinate system for the scaling. N1 is the origin for the local coordinate system. The vector from N1 to N2 is the local x axis. N3 defines another point which lies in the xy plane. This method is the same as ***DEFINE_COORDINATE_NODES** and is used a lot in LS-Dyna. For further information look at the user guide. The 3 nodes can be typed in or picked using the popup menus. As for the global scaling option the centre can be the origin, a coordinate or a node.

6.1.14 Saving and Reading ORGIAMI/Fold Definitions

There is no capability to directly read and write the ORIGAMI and FOLD definitions to a file. However, the information is stored in a section labelled **`*ORIGAMI`** at the end of the LS-DYNA (after **`*END`**) keyword file which includes all the fold information. This is automatically added when a LS-DYNA keyword file is written. The format of these is included in the comments (see also Appendix III). When read back into PRIMER, these definitions are available to the airbag folder.

To stop any **`*ORIGAMI`**, **`*FOLD`** and **`*ORIENT`** definitions from being output, the ORIGAMI definitions must be deleted.

Although the **`*ORIGAMI`**, **`*FOLD`** and **`*ORIENT`** definitions are available in ASCII form at the end of the LS-DYNA input, it is recommended that hand editing be avoided as it is error prone: to modify fold and orient definitions read them back into PRIMER.

Note also that **`*ORIGAMI`** definitions should not be separated from their "parent" input decks, since they make reference to nodes, sets and coordinate systems within those decks.

6.1.15 Additional Airbag Folding Notes

The following will help users to fold airbags successfully

- For **thin** and **tuck** folds make sure that mesh lines follow fold lines exactly, (or at least within **Tolerance**). To improve accuracy of these folds it is usually important that the mesh lines adjacent to a fold line are also straight and have a constant spacing (perpendicular to the fold) from the fold mesh line. This is not so critical for other fold types. If your mesh lines do not follow the fold lines exactly this can easily be fixed by using an **ALIGN** fold.
- Be sure that the airbag does not have any cuts: it should be a closed surface. Circular holes are not a problem, but there may be computational problems if there any internal free edges.
- Thick folds and spiral folds can result in penetrations between shells on different layers. These need to be done selectively and the radius may need to be increased to avoid penetrations.
- Plan ahead if possible. Frequently, there are many different orders in which folds can be done which will result in the same final folded configuration. One order is usually much easier to accomplish than the opposite order for complex bags. If difficulty results from trying to fold a bag in one order, then perhaps try the opposite order. Subset folding can make the folding process much easier but this can only be used when the nodes for a fold are a subset of the nodes from the last fold (i.e. if the fold order in the bag is from the centre towards the edge, not from the edge towards the centre).
- Be sure always to select **NEW** before selecting options for a new fold.
- If necessary a fold can be **DELETE**d and the user can start it again if something goes wrong.
- If nothing appears to happen when creating a fold ensure that sets, carried over from the previous fold, are not defined in this definition. This can happen as much of the previous fold's data is carried across when creating a new fold. Simply go to the **SETS AND LAYERS** feature and change the set or layer definition.

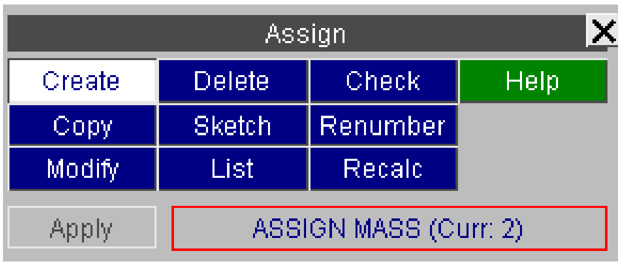
6.1.16 Folding Example

The above may sound somewhat complicated. In fact the easiest way to learn is to try the process and see what the various features do. To help this learning process the user should follow the example shown in Appendix IV.

6.2 ASSIGN MASS

The assign mass panel allows you to add mass and change the centre of gravity of a group, part-set or assembly. This is done by adding lumped masses on the nodes in the group (or a subset of the group if you use the subgroup option).

It is impossible to assign mass to nodes of a part which is defined with a **`*PART_INERTIA`** card as the lumped masses will be ignored by LS-DYNA. However, this does not mean that part inertias cannot be present in a massing up operation. They can be, but must be wholly contained in the group to be valid and included in the mass calculation. See the part inertias section in the problems below for more details



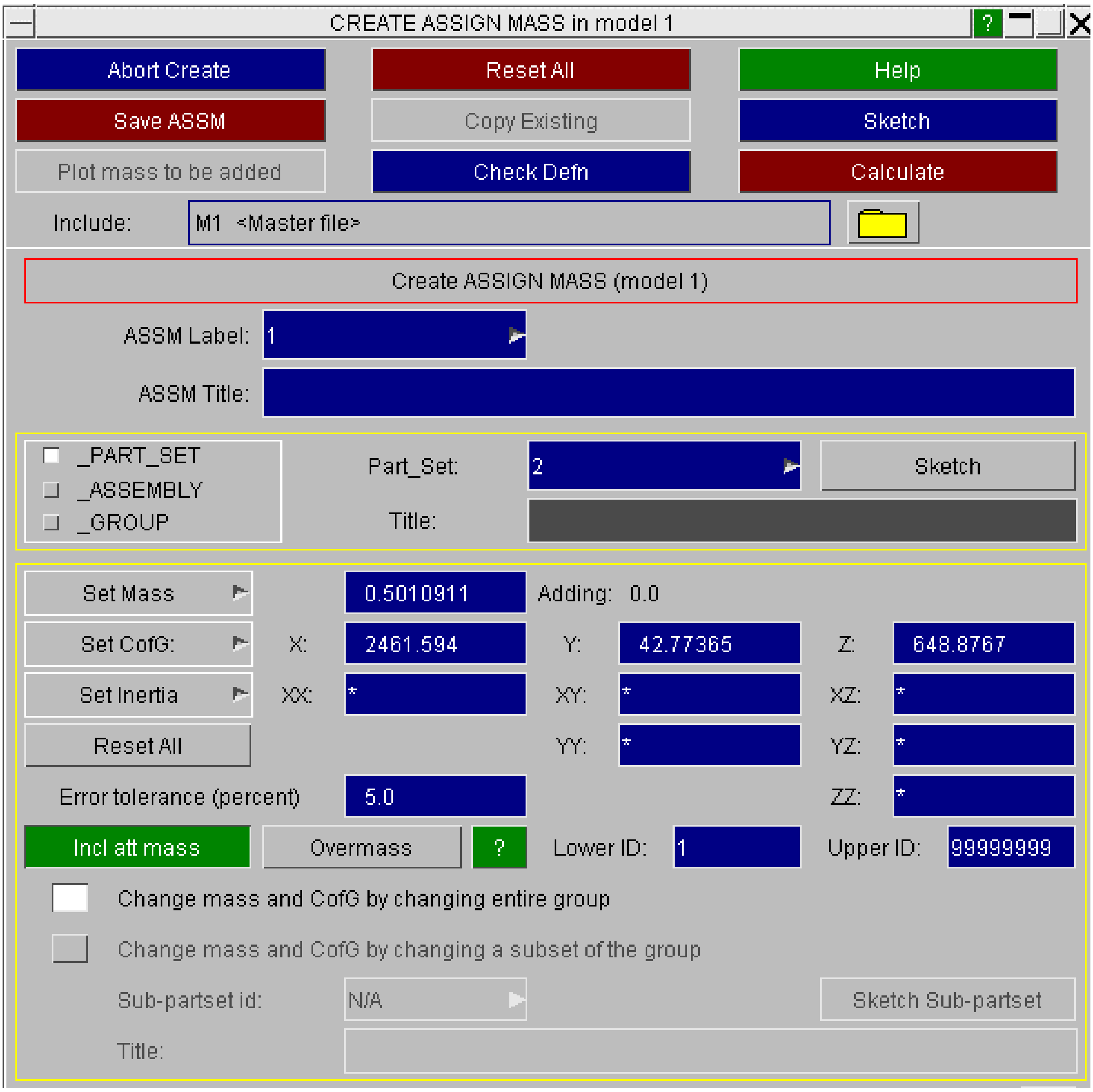
Basic assign mass operation

- Creating
- modifying
- recalculating

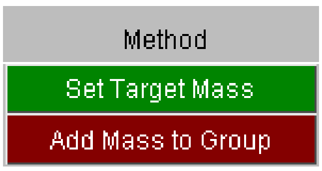
Advanced assign mass operation

- Including part inertias in the assign mass operation
- Hierarchy of assign mass
- Problems with assign mass
- Controlling suppression of text box warnings

Creating an Assign mass



Assign mass may be used to add mass to a group of entities (typically a selection of parts) or to achieve a target mass for them. The drop-down under **Set Mass** will set this mode.



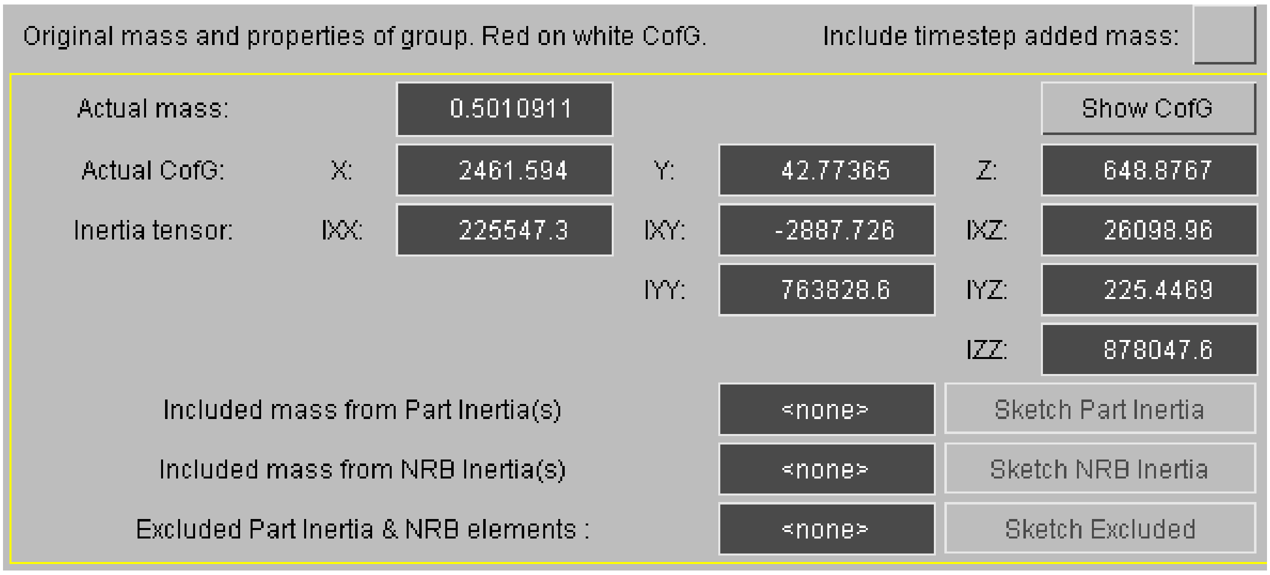
The user needs to select (a) the group which constitutes that mass of interest and (b) the group to which mass is to be added. (b) may be a sub-group of (a) or both groups may be the same(as in the above example).

Group (a) may consist of a Primer group, a part-set or an assembly. Group (b) may consist of a Primer group or a part-set.

The Primer group definition is versatile, but will require maintenance by the user, should the contents of the model change, therefore use of *ASSIGN_MASS_PART_SET (or *ASSIGN_MASS_ASSEMBLY) is often preferred as the contents are easier to maintain as the model updates (e.g. by using PART_SET_GENERATE).

*ASSIGN_MASS(_GROUP) is still available for backward compatibility.

Once the selection is made mass properties are displayed at the bottom of the panel the panel.



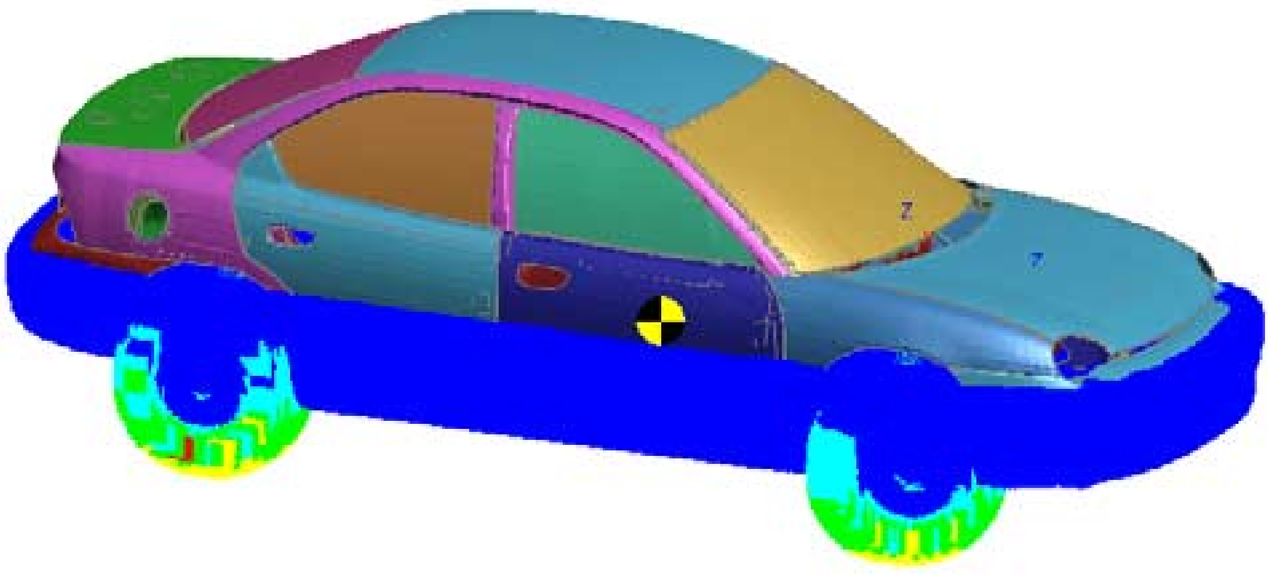
By default, **Incl attached mass** ia active, which means the mass of any attached lumped masses is implicitly included.

Timestep added mass is not included in the calculation by default, but may be by activation of the setting **Include timestep added mass**.

Reset All will set the target mass and CofG to the original properties of the selection. The drop-downs off **Set CofG** and **Set Inertia** may also be used to set the original values

By typing in to the text boxes, target values may be set for individual terms of centre of gravity and inertia or these may be left free (indicated by the wildcard symbol) to assume their own value.

Target mass must be increased above the original mass. Press **CALCULATE** to determine the mass distribution which will best meet the assigned properties.

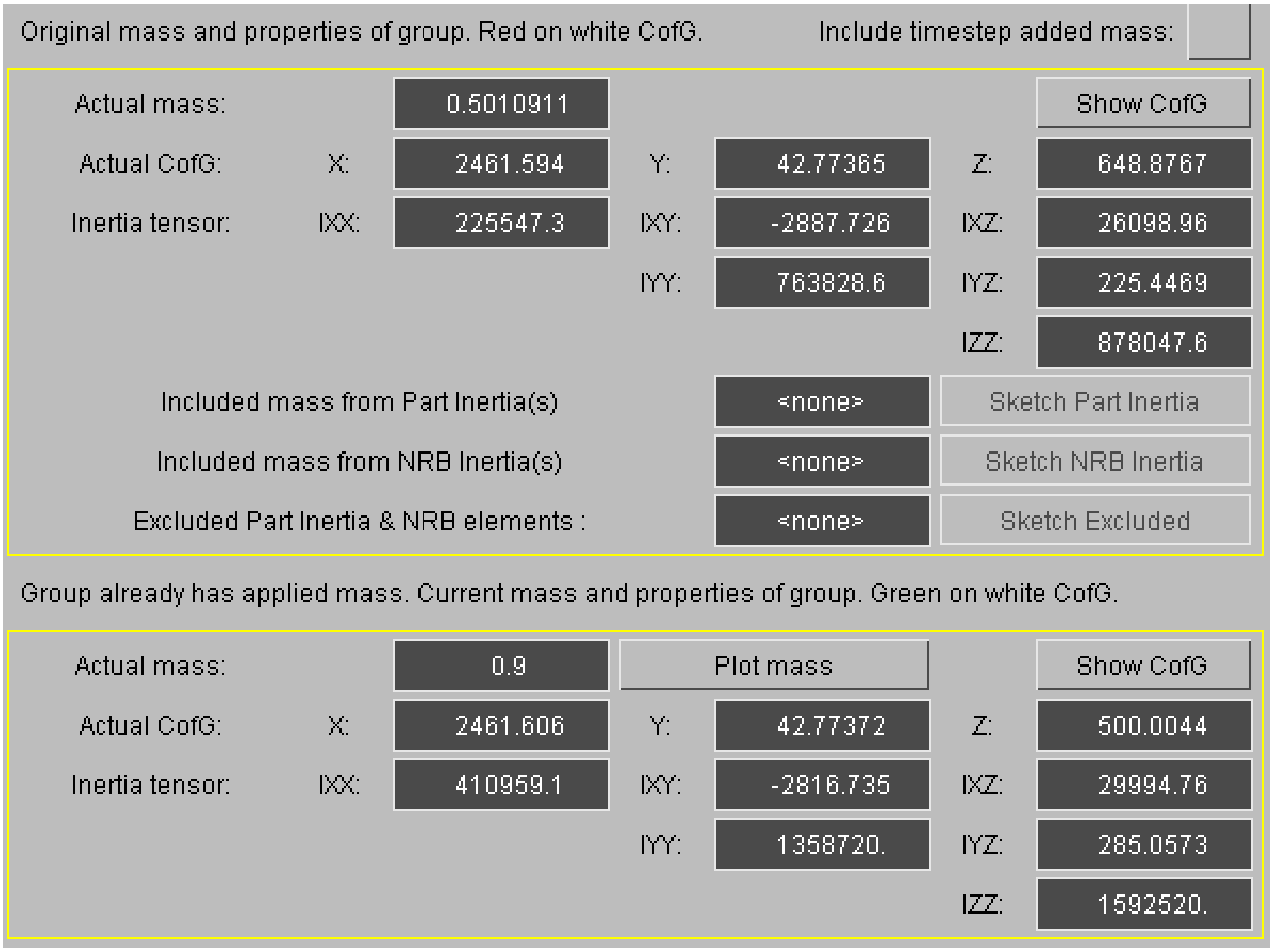


Plot mass to be added will show the proposed mass distribution. In this case, lowering the CofG has biased the mass toward the bottom

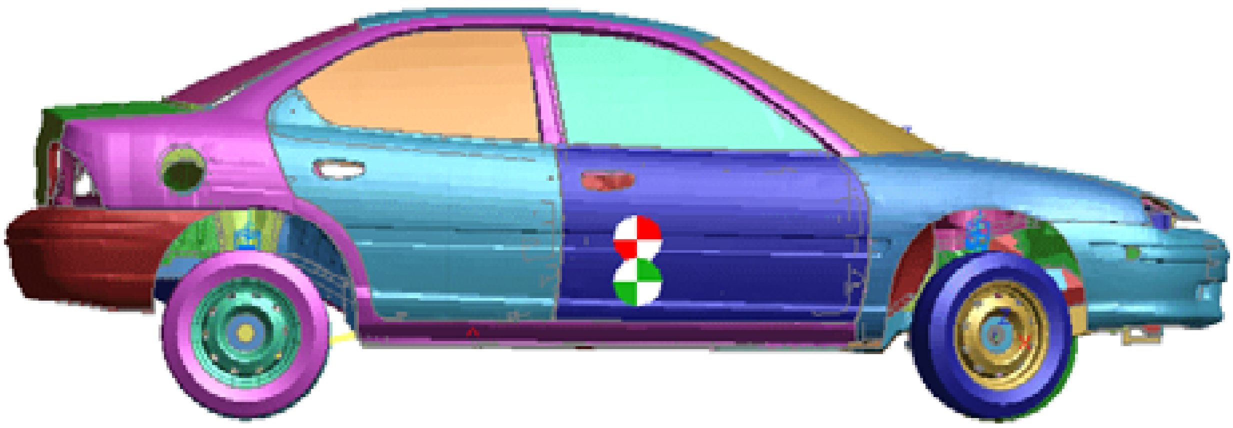
Create ASSM will then implement and save the solution.

Modifying an assign mass

If we modify the previously created definition the panel will display both the original and the current mass properties.



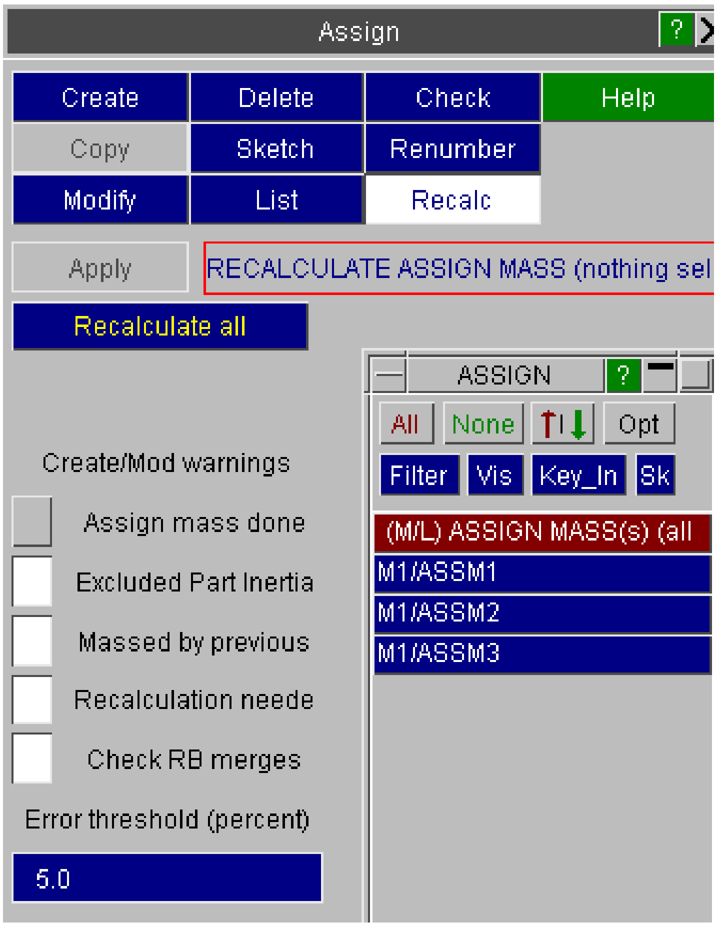
The modified CofG may be observed by using the **Show CofG** buttons.



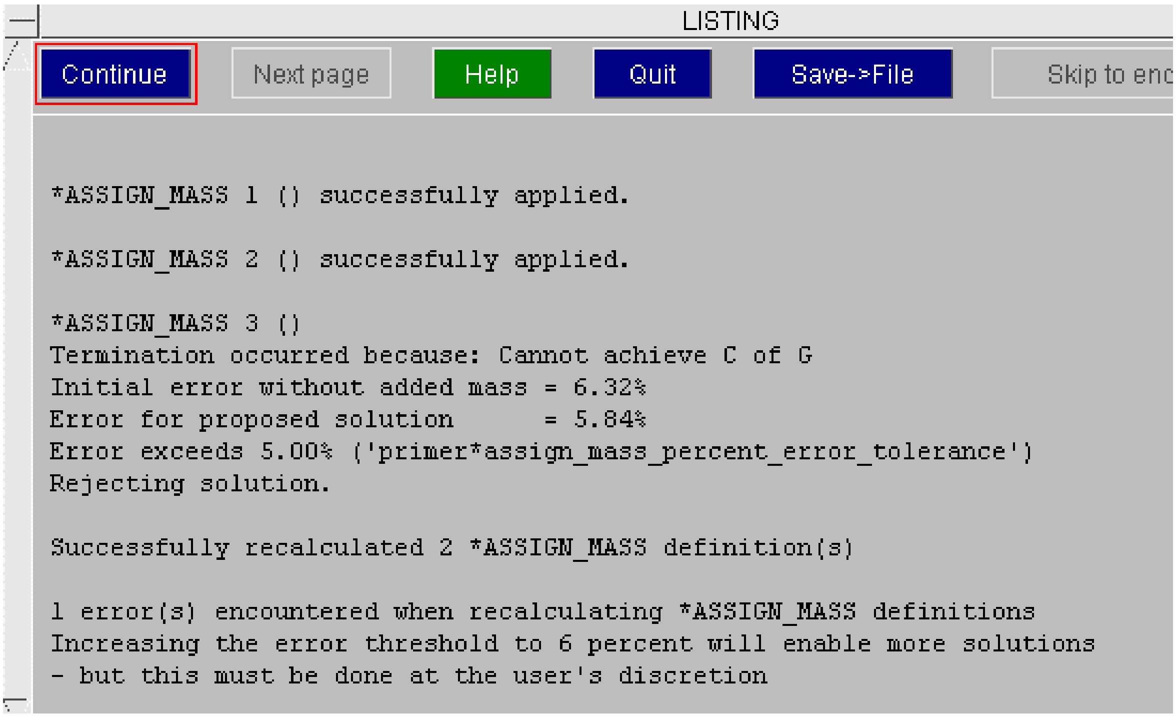
The definition may be modified and re-made with different target properties by using **Calculate** and **Update ASSM.**

Recalculating an assign mass

If you have a model which has been modified (remeshed, material properties changed, etc) with multiple assign mass, these can easily be remade by the **Recalculate All** function.



On completion, a listing panel will report any problem definitions.



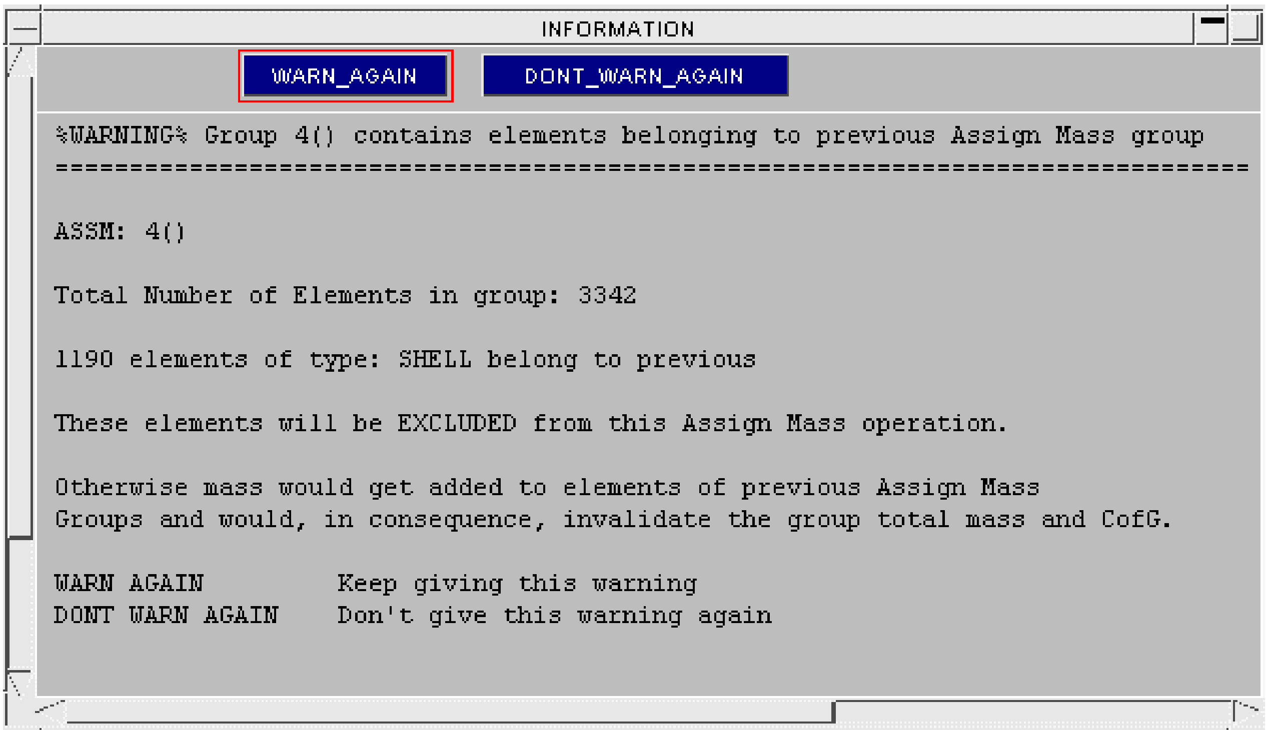
The same function can be accessed from the command line by the syntax ASSIGN APPLY_ALL.

Hierarchy of assign mass

If you are massing assemblies of components which have themselves been massed up, you must observe the hierarchy that the assign mass statements of the components precede (i.e. are at a lower label than) the assign mass statement of their corresponding assembly.

When you come to mass the assembly, you may either define a subgroup which contains all parts of the group except those which have already been massed, or, as this may be rather inconvenient, you can allow primer to **automatically exclude** those elements which have been massed previously.

When you **CALCULATE** the assign mass, you will get the following warning:



If we modify an Assign Mass group which contains elements which are used by a later (hierarchically higher) statement, a warning will be given and the user urged to apply the **RECALC** function. This will remake all the assign mass statement which have labels above the current one, thus accomodating the affect of modifying the mass of the lower group. In default mode, the elements will not be remassed. To maintain the integrity of the assign mass statements, it is recommended that the function be used in this way. However, some users have requested the ability to add mass to items already massed up. This may be done by setting the **OVERMASS** flag on both the overmassed and the overmassing assign mass statements.

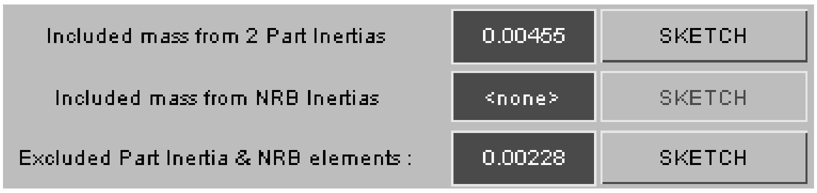
Including part inertias in the assign mass operation

If a group contains *PART_INERTIA or *CONSTRAINED_NODAL_RIGID_BODY_INERTIA cards they will be included in the assign mass calculation if they are completely contained in the group. For example if you mass up an entire car that contains an engine which is a part inertia that will be fine. If you try to mass up the rear 2/3 of the car so only half of the engine is in the group, the engine part inertia will not be included.

If your group does not contain any inertia definitions then the panel will be displayed as shown on the right.

If your group does contain some inertia definitions then the panel will be displayed as shown on the right. The mass will be shown for each type and the elements can be sketched.

The included mass from parts and NRB's is shown. These are inertias that are completely contained in the group and so are included in the calculation.

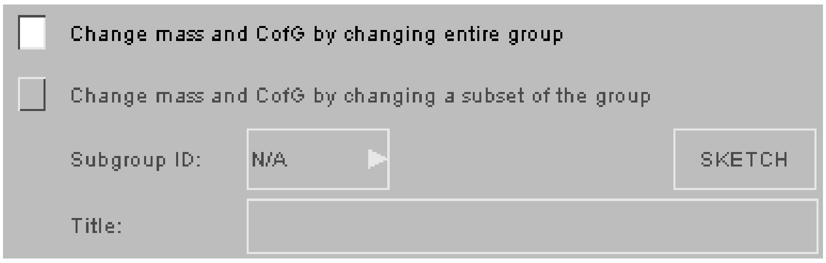


The Excluded part inertia and NRB elements are from inertias that are not completely contained. If this occurs Primer will give a warning and they will not be included in the mass calculation (see the problems section)

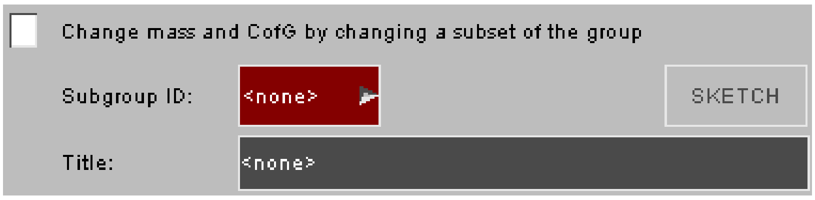
LS-DYNA will ignore any lumped masses that are on inertia definitions. They will be overwritten by the part inertia when DYNA initialises. Primer will not create any lumped masses on inertia definitions.

Changing the mass of a group by only adding mass to a subgroup

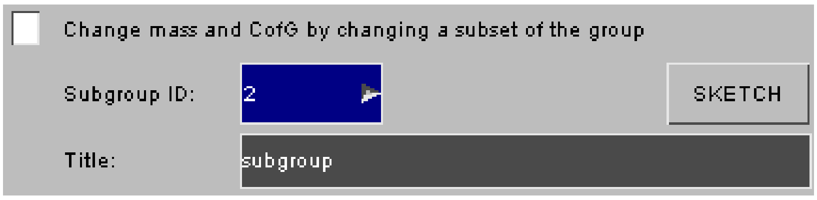
By default the assign mass function will try to change the mass and centre of gravity of the group by adding mass to all the nodes in the group (except the nodes on *PART_INERTIA and *CONSTRAINED_ NODAL_RIGID_BODY_INERTIA cards.



If you only want to change the mass on a certain part of the group instead then select **Change mass and CofG by changing a subset of the group**.



You can then select a subgroup which will be used by Primer for adding lumped masses to instead of the main group. This group **MUST** be a subgroup of the main group for this to work. If you try to use a group that is not a subgroup of the main group Primer will warn you.

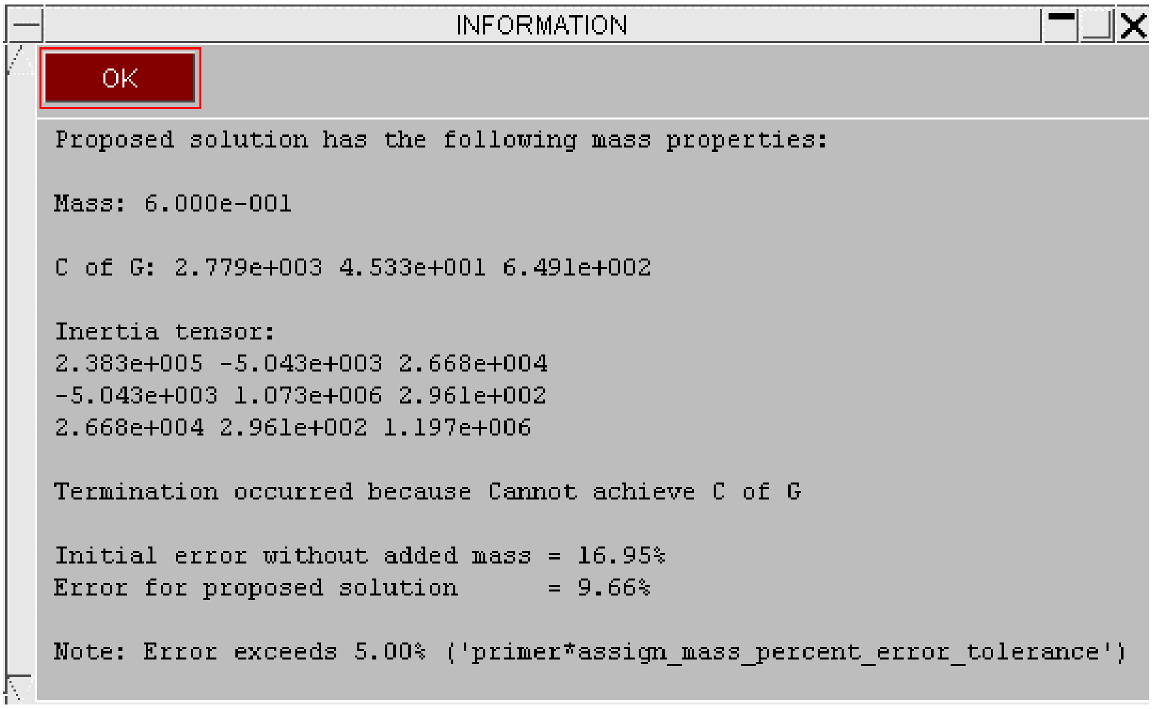


Problems with assign mass

Unable to achieve centre of gravity or inertia

The assign mass function in Primer works by adding lumped masses to (some but not necessary all of) the nodes of the group in an iterative process which attempts to meet the requirements of total mass, CofG and Inertia.

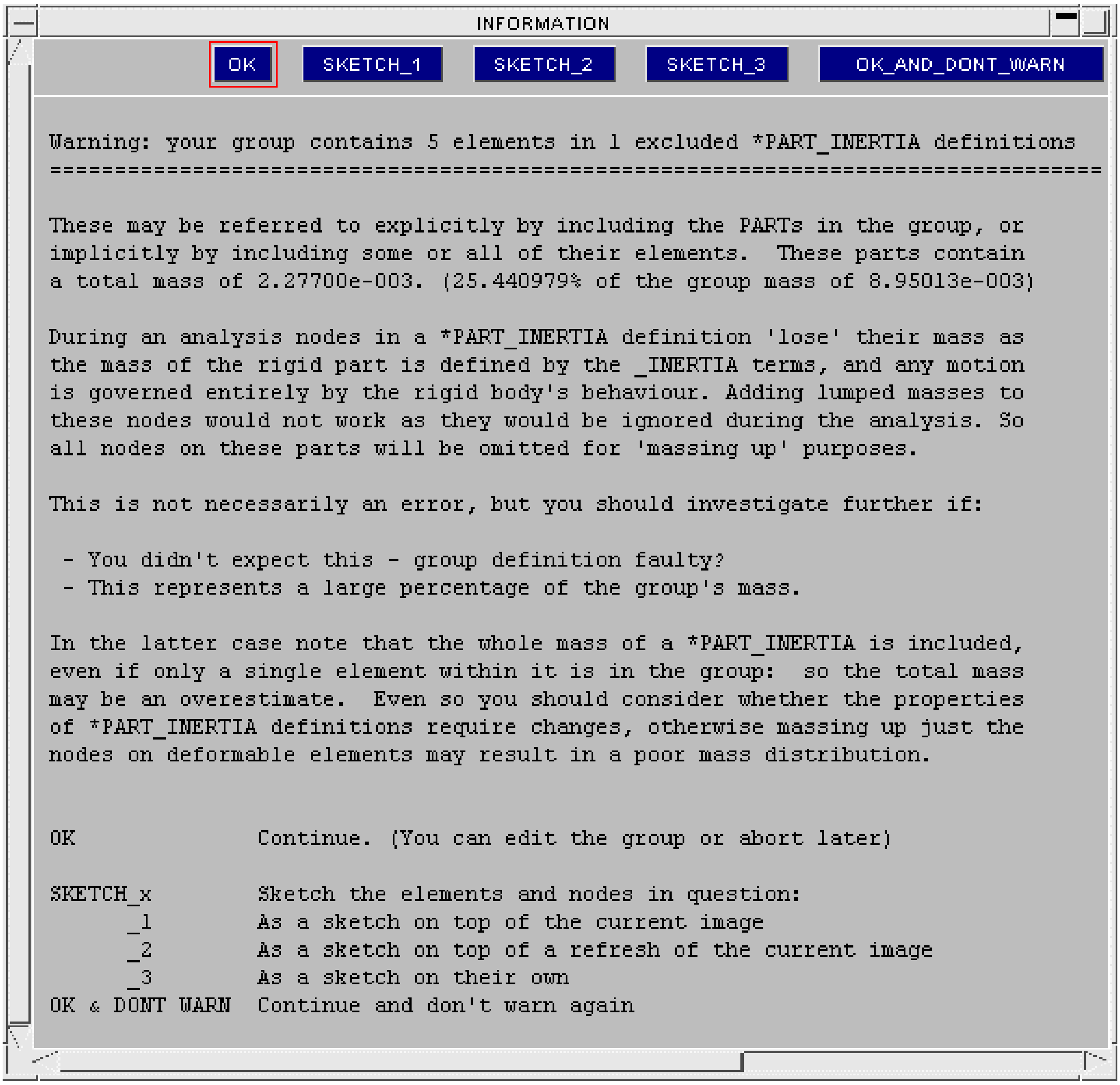
If the error of solution falls below the specified error tolerance (5% by default) the the mass distribution is simply applied as above. If a satisfactory solution cannot be reached an information panel will be offered giving the magnitude of the error.



This will usually occur because for the given amount of mass being added the CofG cannot be shifted by the required amount. Primer can add mass to nodes but it cannot remove mass.

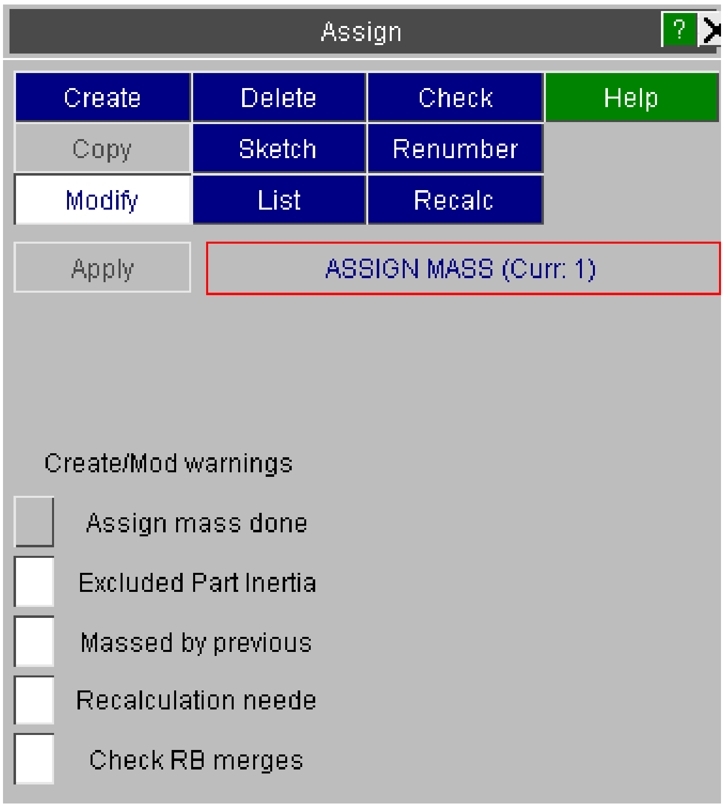
Part inertias

The assign mass function in PRIMER works by adding lumped masses to the group you have specified. This works perfectly OK (even on rigid materials). However, if you have a rigid material that has a ***PART_INERTIA** card, LS-DYNA ignores the mass of the elements and lumped masses and imposes the mass and inertia properties from the ***PART_INERTIA** card. This means that any lumped masses that are added in PRIMER during the assign mass function to nodes that are in a ***PART_INERTIA** will be ignored. An identical problem occurs if a node is part of a ***NODAL_RIGID_BODY_INERTIA**. Primer will include any inertia definitions that are completely contained in the group in the mass calculation but will not produce any lumped masses on the nodes in the inertia definition (see Including part inertias in the assign mass operation)
If an inertia definition is not completely contained it will not be included. To warn you of this, when you select the group in the assign mass panel, PRIMER checks to see if any of the nodes are on a ***PART_INERTIA** card or a ***NODAL_RIGID_BODY_INERTIA** card. If any of the nodes in the group are part of an **_INERTIA**, then Primer checks to see if the inertia is completely contained. If it is not, a warning screen is printed and the nodes will be excluded from the assign mass calculation.



Controlling suppression of text box warnings

The parent panel of assign mass has several warning options which the user can select or deselect as they wish:



The selection boxes allow the user to suppress warnings and errors that they feel are unnecessary.

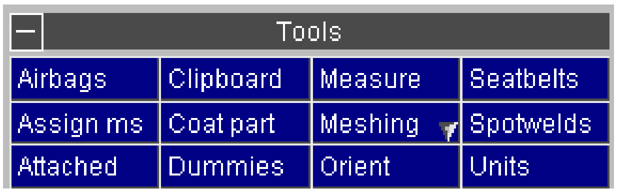
assign mass done - reports the completion of the assign mass and the number of lumped masses added

excluded part inertia - gives the warning described above that part/nrbc _inertia cannot be included

massed by previous/recalculation needed - warn that there are hierarchy conflicts in the definitions which require resolving

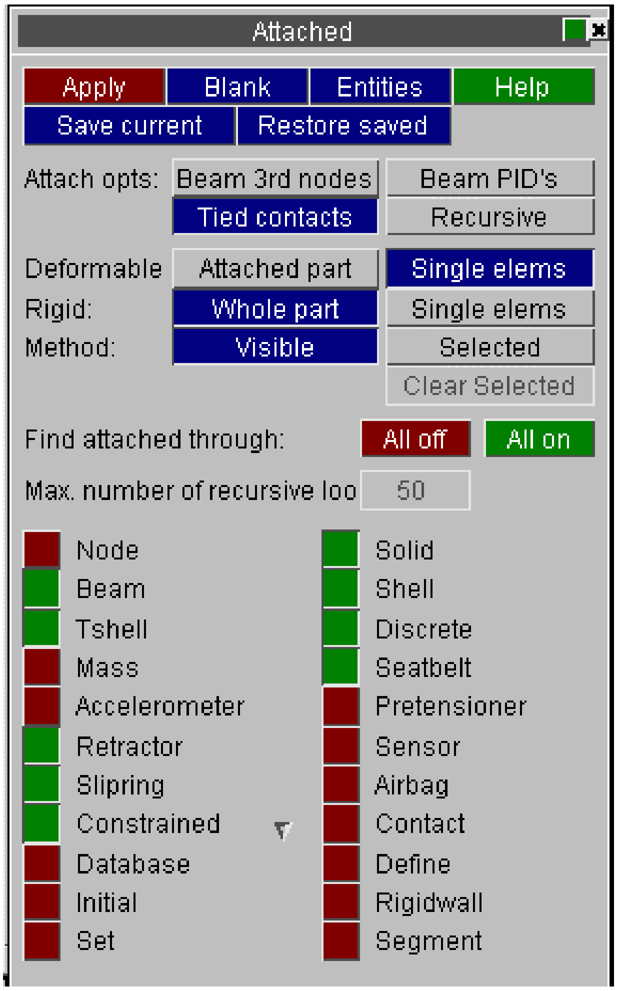
check RB merges - warns that a subset of slave/master rigid parts are included in the assign mass group. Whilst not an error this can cause confusion about the total mass (as Dyna assigns mass of slave parts to the master).

6.3 ATTACHED Displaying what is "attached to" things



The **ATTACHED** menu is invoked from the **Tools** panel.

6.3.1 Top level menu



This figure shows the top level "attached" menu.

When you enter the **ATTACHED** menu the following happens:

- Everything that is currently drawable (ie unblanked with its entity switch turned on) is unblanked.
- Everything else is blanked.
- This blanking status is "remembered".
- Sets the attached switches to find anything physically attached.

At this stage performing a drawing operation (**LI**, etc) will not result in any change to what is currently visible.

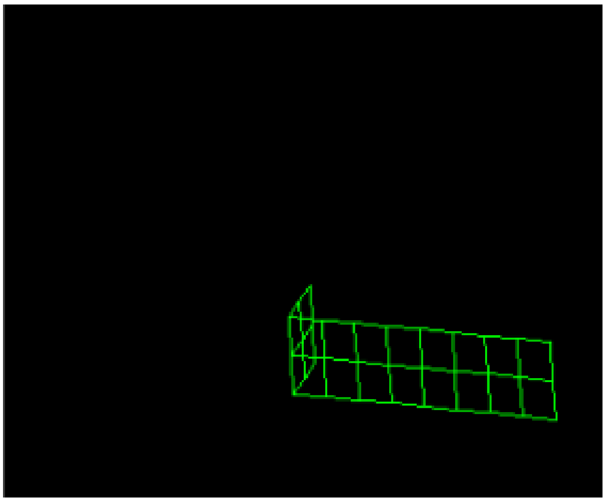
However each time you press **APPLY** Primer does the following:

- Looks at what you want to find attached (shells, constraints etc)
- Finds what is immediately "attached to" what is currently visible.
- Unblanks these newly found items.
- Redraws the image.

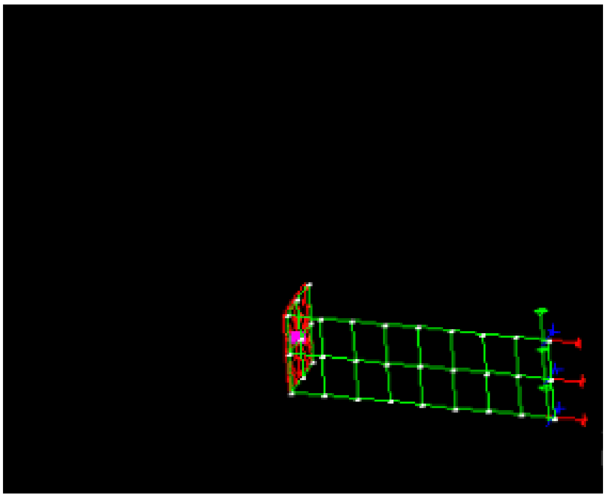
This results in progressively more and more of the model being drawn until nothing attached to what is currently visible (which is not necessarily the whole model) remains to be unblanked and drawn.

The following six images demonstrate how **ATTACHED** makes progressively more and more of a model visible:

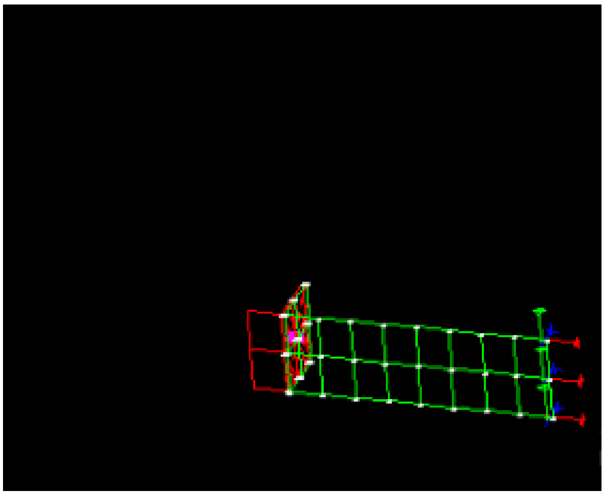
(1) Just one part visible



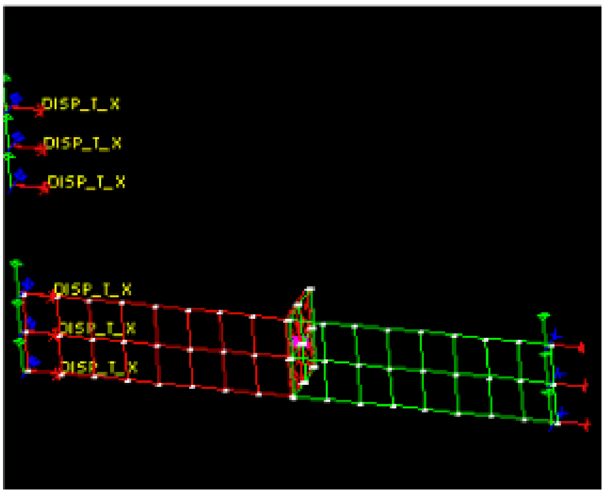
(2) Restraints, contact and spotweld to next part



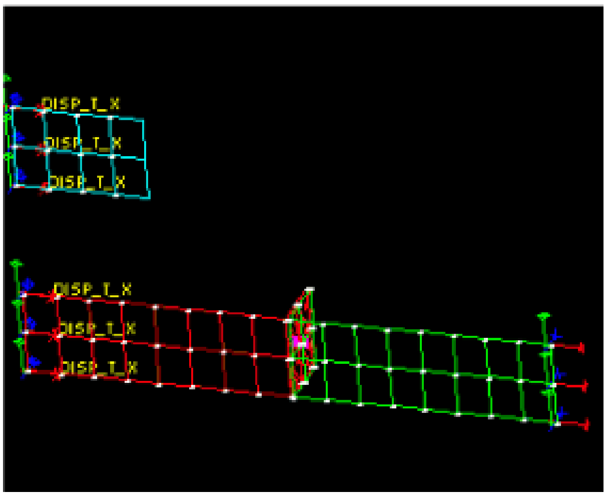
(3) Nearest elements on next (red) part attached to spotweld.



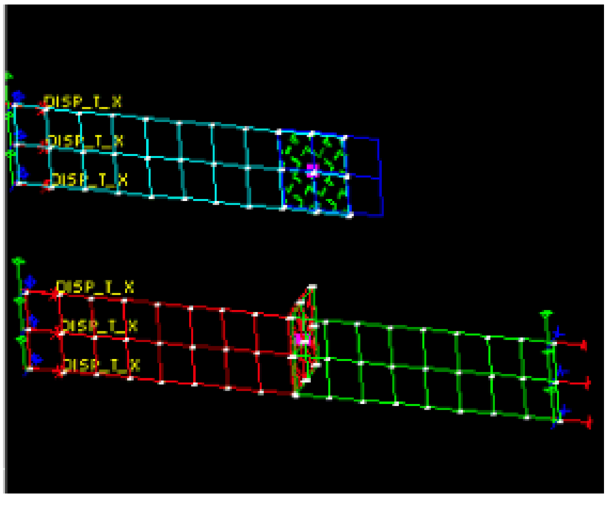
(4) (After a few **APPLY** operations) restraint set on red part



(5) (After more **APPLY**s) elements on the light blue part



(6) More **APPLY**s: spotweld to & elements of dark blue part.



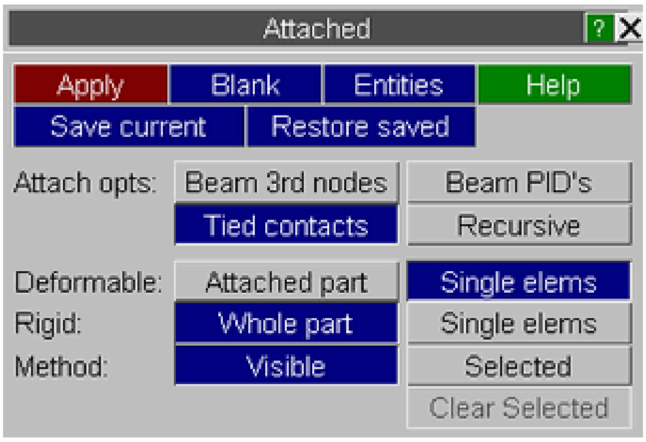
6.3.2 What does "attached to" actually mean?

In this situation attached generally means "***anything connected to, or referenced by, what is currently visible***".

This is a wider definition than the strictly structural "*any elements attached to visible nodes*" which, initially, might seem to be the more obvious method. The figures above show why this is so:

- Figure (4) shows the ***BOUNDARY_PRESCRIBED_MOTION_SET** (Labelled as DISP_T_X) applied to all the edge nodes on the left of the picture.
- Because the set used by this includes nodes on the end of the light blue part (visible in figure (5)) then elements attached to this part have become visible.
- Consequently the light and dark blue parts eventually become visible, even though they have no physical connection to the red and green ones.

6.3.3 Attached options



There are several options available to the user to increase the flexibility of the attached panel.

- Beam 3rd nodes
- Beam PID’s
- Tied Contacts
- Recursive

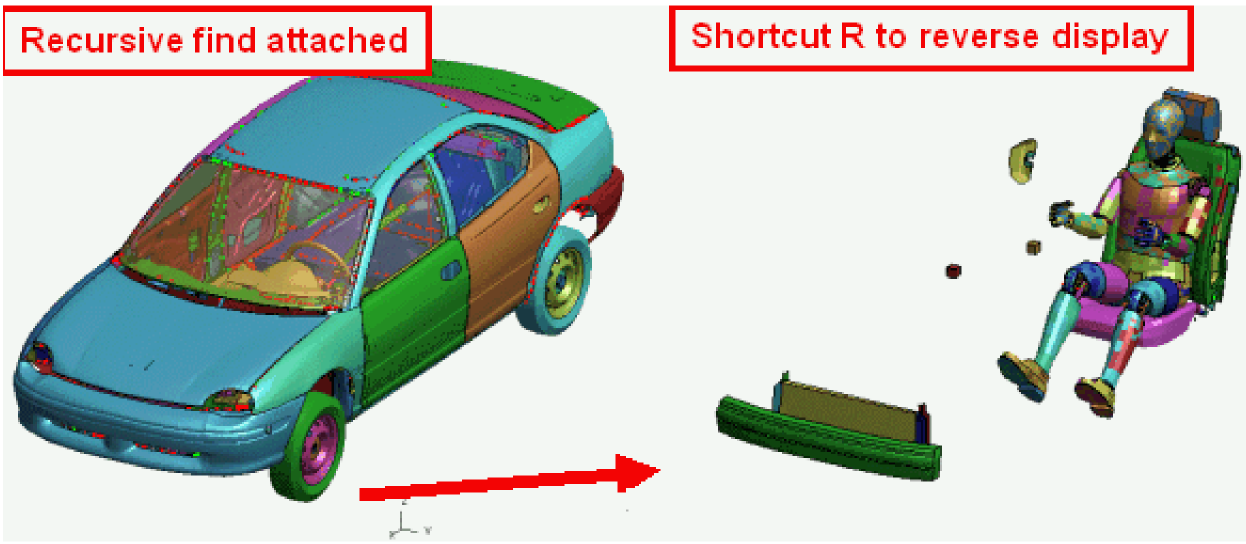
Beam 3rd nodes will find attached entities through a beam’s 3rd node (and vice versa).

Beam PID’s will find attached beams that refer to a part displayed through their PID1 and PID2 fields.

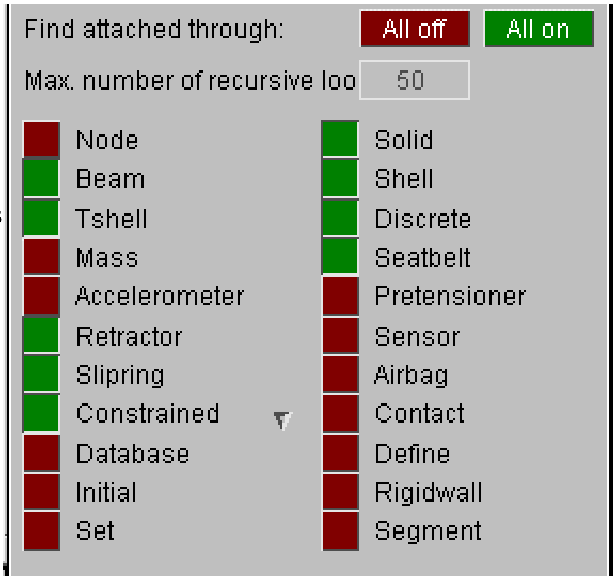
Tied contacts will find attached elements through DYNA tied contacts using PRIMER’s contact penetration checker.

Recursive will iteratively keep finding attached until no more can be found - note there is a failsafe value in PRIMER to allow for any anomalies that might cause this routine to go on forever (the STOP button also works here). Pressing reverse all (shortcut R) will then show any unattached parts.

Instead of finding attached to all the visible entities, the user can select the entity/entities they wish to find entities attached to. This is done by selecting **Selected** for the method instead of **Visible**. In selected mode, an object menu is used to select the "seed" items. Use **Clear Selected** to reset your selection.



6.3.4 Restricting the extent of "attached to" propagation

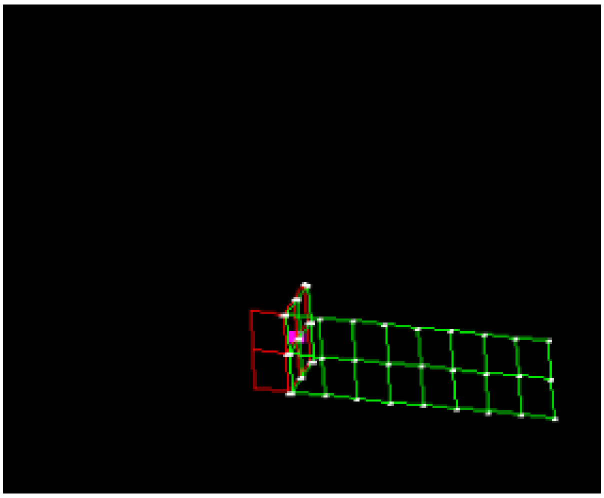


Because the definition in section 6.3.2 is all-embracing it may lead to too many things being made visible. Therefore it is possible to limit what is found attached through entity switches .

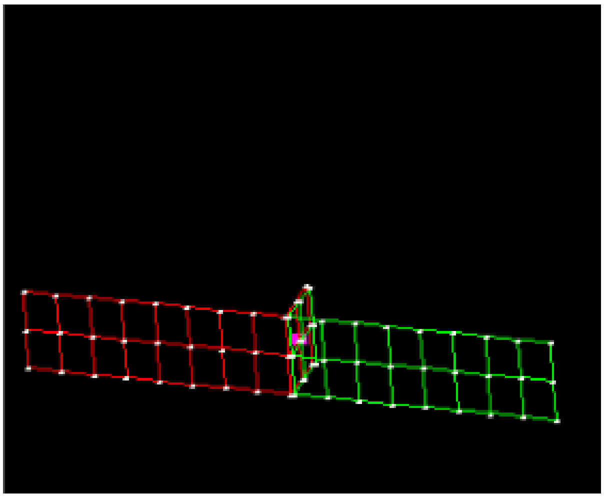
For example, you can still display shells, solids, beams ect but just find attached beams. Note that there is a triangle to the right of the **Constrained** entity switch. Right clicking here allows you to select to **Filter** the constrained entity types. A new panel will open up allowing you to turn on or off entity switches for the different constrained types. This allows you, for example, to find attached through nodal rigid bodies, but ignore constrained spotwelds.

In the example below the user has selected nodes and elements only, which results in the narrower "structural" definition of attachment referred to above. Starting from the same point as figure (1) above a series of **APPLY** operations gives rise to figures (7) and (8) below:

(7) The spotweld connects to the 2nd red part as before



(8) The final result: only nodes and elements are drawn



Now the blue parts are not diagnosed as being attached to the red and green ones, since the connection between them (the node set used by an initial velocity definition) has not been drawn.

6.3.5 Using and updating the "SAVED" status

When you enter **ATTACHED** the current visibility status is saved in a backup blanking table. All **APPLY** operations operate only on the current blanking table, leaving this backup unchanged.

The reason for this is simple: most usage of **ATTACHED** reveals too much information in the first pass, and it is necessary to go back and repeat the process with some **attached** categories switched off.

RESTORE SAVED	Copies the backup blanking tables to the "current" ones, effectively restoring the initial state. You can **RESTORE_SAVED** as many times as you like within a given usage of **ATTACHED**. Because the backup tables are always rebuilt from what is currently visible whenever you (re)enter **ATTACHED** they are effectively "lost" whenever you close this panel. To maintain a backup while performing other operations you can iconise this panel rather than closing it.
SAVE CURRENT	Copies the current blanking tables, ie what is currently visible, into the backup ones (overwriting them). This then becomes the new "saved" state. You can **SAVE CURRENT** at any time, but doing so loses your original saved state irretrievably.

6.3.6 How ATTACHED inter-reacts with Entity Viewing and Blanking

ATTACHED has to modify entity drawing and labelling settings (the province of **ENT**ity Viewing) and also blanking tables (**Blank**).

The entity switches you find attached through will inevitable effect the entity panel. If shells are searched for, and the entity panel doesn't have shells displayed, then the attached panel will turn the entitiy switch on too.

For convenience you can access the relevant panels directly from the **ATTACHED** panel, as shown here (they are no different when accessed here as opposed to from the main panel).

6.3.7 Some limitations of ATTACHED - mainly due to using `*SET_xxx`

Because the definition of "attached" is wider than the purely structural one of connected nodes and elements (see section 6.3.2 above), some problems can arise when definitions which use sets are diagnosed as being "attached".

Consider the following case:

- Initial velocities for the whole model are defined by **`*INITIAL_VELOCITY`** using a **`*SET_NODE`** that contains all nodes in the model.
- Once a single node in that set is detected as being "visible" then the set itself is also made visible. This has the consequence of drawing all nodes (with their initial velocities) in the model ....
- .... and the next thing you see, after a single **APPLY** operation, is the whole model being drawn.

This presents a dilemma: should **ATTACHED** track through **`*SET`** definitions or not?

If it does then the problem defined above occurs.

If it does not then, for example, extra nodes on a rigid part (defined by *__`CONSTRAINED_EXTRA_NODES_SET`__*) will not be drawn properly when the part is visible.

At present **ATTACHED** ***does*** track through **`*SET`** definitions, leading to the problem outlined above. (Although the case of set zero, where it means "all nodes in the model", is detected and trapped.)

You can stop specific cases happening by turning off their switch (for example to stop the initial velocity display turn off **`INITIAL`** switch). You can also **BLANK** specific items (although they will probably unblank themselves again after the next **APPLY** operation).

6.4 BILL OF MATERIALS

The Bill of Materials section in PRIMER enables you to check and if necessary update part names, the material title used for a part and the gauge of a part. This is done by reading a file containing the necessary information to check for each part.

6.4.1 File format

The bill of materials reader is designed to be able to read files produced by hand, spreadsheet programs and other programs. The files produced from spreadsheets are commonly known as CSV or comma separated files.

The format of the file has to follow the following rules:

- Blank lines in the file are skipped.
- A comment line can be included anywhere in the file by starting the line with a specific character (which you can define when reading the file into PRIMER).
- Lines that contain specific strings or characters can be skipped
- The Bill of materials file should contain one line for each part entry you want to check/modify.
- Each line should contain 'fields' that are separated with a specific character (which you can define when reading the file into PRIMER).
- The fields must be in the same order for every line.
- One of the fields MUST be the part ID.
- For auto-recognition of columns to work, columns must be given specific names.

Example

An example Bill of materials file is shown below.

```
$ Bill of Materials file:Example Bill of Materials
$ Date produced: April 2001
$ Produced by: Miles Thornton
Vehicle X,Bill of Materials version,8.6,Date,20/02/01

Part No,Title,Part ID,Material,Supplier,Gauge,Part mass
AA51201,sill_swan_neck,5,HP37  ,Company X,2.2,9.64E-03
AA51202,front_support_mem diagonal,101,HP37 ,Company X,2.2,4.74E-03
AA51203,Bumper_ft,104,HP37,Company X,1,3.71E-03
AA51204,A_pillar_lower_support_a,113,CR4  treatment C,Company Y,1.2,2.28E-03
AA51205,cowl.1,200,CR4 treatment C,Company  Y,1.2,6.40E-03
AA51206,A_pillar_lower_support_b,202,CR4 treatment C,Company Y,2,6.60E-03
AA51207,dash_x_member,203,CR4 plt3 grade,Company Y,1.2,2.25E-03
AA51208,dash_panel,204,CR4  plt3 grade,Company Y,1.2,4.48E-03
AA51209,front_floor_panel,304,CR4 plt3 grade,Company  Y,1.2,7.17E-03
AA51210,Tunnel_wall_FR,305,CR4 plt3 grade,Company Y,1.2,5.76E-03
```

We want to skip the first 3 lines and so have started them with a $.
The fourth line does not begin with a '$' but we can force the reader to skip the line if needed.
The fifth line is blank and so will skipped.
The next line describes the fields. This is not necessary but helps to see which fields are which. If you want to do automatic recognition of columns then a line that contains the column names is essential. This can also be skipped.
The following lines contain the information. Each line contains the information for one part. The fields are separated by a ',' (comma). The following sections show this file being read and used by PRIMER.

Automatic recognition of columns in Bill of materials file

If **Auto-detect columns** is switched on then when PRIMER reads a bill of materials file it looks for specific column headers on any of the lines which are not blank or comments. The first 50 lines in a bill of materials file are shown as a preview when reading the file. Primer will look at each of these lines in turn and look to see how many of the column names it can match to the standard column names. The line that matches the most headers is the one that will be used for the column titles. If you do not want PRIMER to recognise the columns this can be turned off

Rules for matching column titles

1. Each 'field' is read from the line in turn.
2. Any spaces are removed from the field.
3. The 'field' text is converted to lower case.
4. The text is compared to the headers below. If an exact match is found then that 'field' type is set to the type that matched

Standard Headers

Field type	**Allowed headers**	**Meaning**	**Field contains**
Part ID	pid, part, number, partid	Model Part ID (compulsory)	integer
Part title	title, parttitle, description	Part title	characters
Material title	material, mat, matname, materialname	Material name	characters
Material ID	materialid, matid, matnumber, materialnumber, mid	Material ID (for referencing standard database of materials)	integer
Cad part number/description	cad, cadpart, cadpartno, partno	Cad part number (not currently used by PRIMER)	characters
Gauge	gauge, thickness, thk	Thickness of part (on *SECTION_SHELL card)	real
Hourglass type	hourglasstype, hgtype	Hourglass type (on *HOURGLASS card)	integer
Hourglass coefficient	hourglasscoeff, hgcoeff	Hourglass coefficient (on *HOURGLASS card)	real
Lower ID	lowerlabel, lowerid,lower,low	start id for renum of nodes/elements/masses on part	integer
Upper ID	higherlabel,higherid,higher,high	end id for renum of node/elements/masses on part	integer
Element formulation	elform, formulation, element	ELFORM on *SECTION card	integer
Section ID	sectionid,secid,secnumber,sectionnumber,sid	Section ID	integer
Hourglass ID	hourglassid,hgid,hgnumber,hourglassnumber	Hourglass ID	integer

Examples

As spaces are removed and the fields are case insensitive all the following could be used for the Part ID field:

- Part ID (would match 'partid')
- P I D (would match 'pid')
- Part (would match 'part')
- PARTID (would match 'partid')
- Number (would match 'number')

The following could not be used:

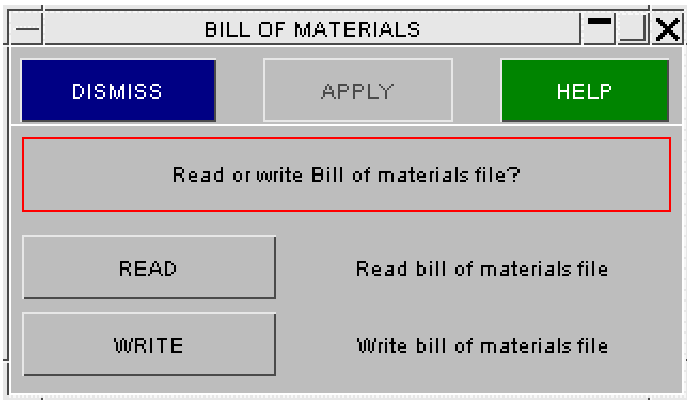
- Part number (As the string would not exactly match 'number')
- ID (as ID is not a valid title)

In the example file shown above all of the columns except for `Supplier` and `Part mass` would successfully matched.

6.4.2 Initial Screen

The initial screen allows you to choose whether to read or write a Bill of materials file.

The two options available are **READ** and **WRITE**.



6.4.3 Reading a Bill of Materials file

Selecting the file

The initial Bill of Materials screen is shown in the adjacent figure.To select the Bill of materials file either type the name into the blue **File:** box or press the **?** button to bring up the file selection panel.

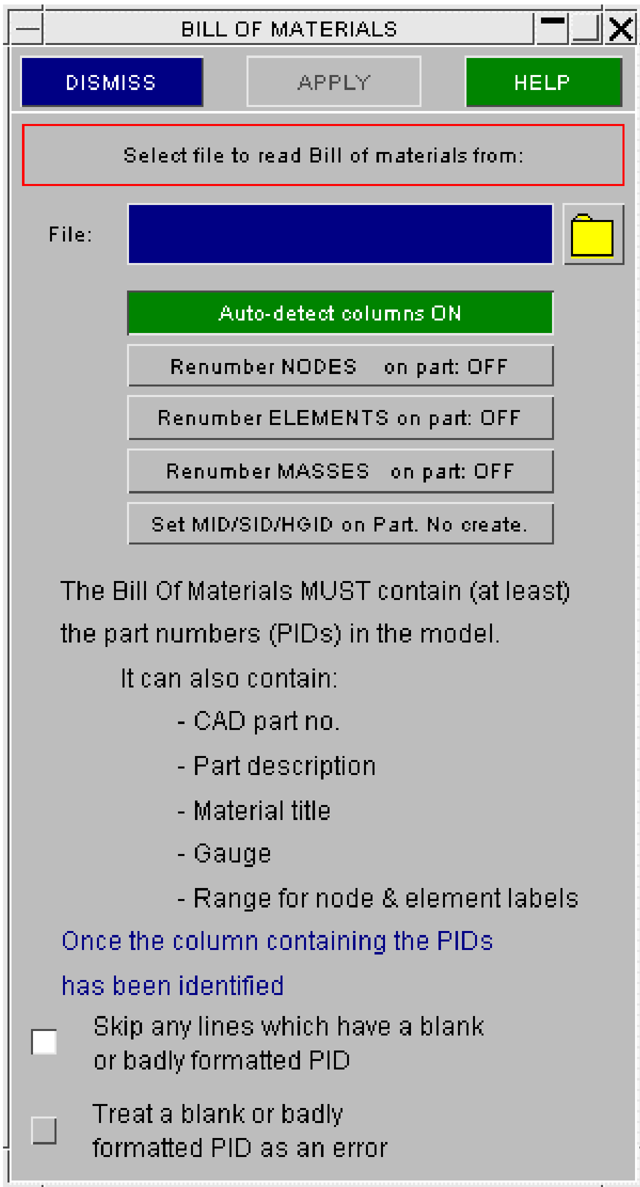
Until a file name is given the **APPLY** button will not be active. When the button is active, pressing it will scan the file and any comment lines can be selected.

In the image on the right **Auto-detect columns** is turned **ON**. PRIMER will try to recognise each field by using the column titles.

If **Renumber NODES/ELEMENTS/MASSES on Part** is active and label ranges are defined for this part, the renumbering function will be activated. On large models this may slow the read of the BOM considerably. Label ranges may overlap one another.

Two methods are available for handling materials, section or hourglass cards when these are shared by more than one part. The default option **set MID/SID/HGID** will set the part to reference the specified material, etc. It will the only adjust the material title, section properties, etc. if no other part references it. The option **Modify/Create Matl/Sect/Hgls** will always make a material/section/hgls card with the specified data, creating a new one and ignoring the specified ID, if this proves necessary, because another part refers to that card.

If the material, section and hourglass card for each part are kept unique , both methods will give the same result.

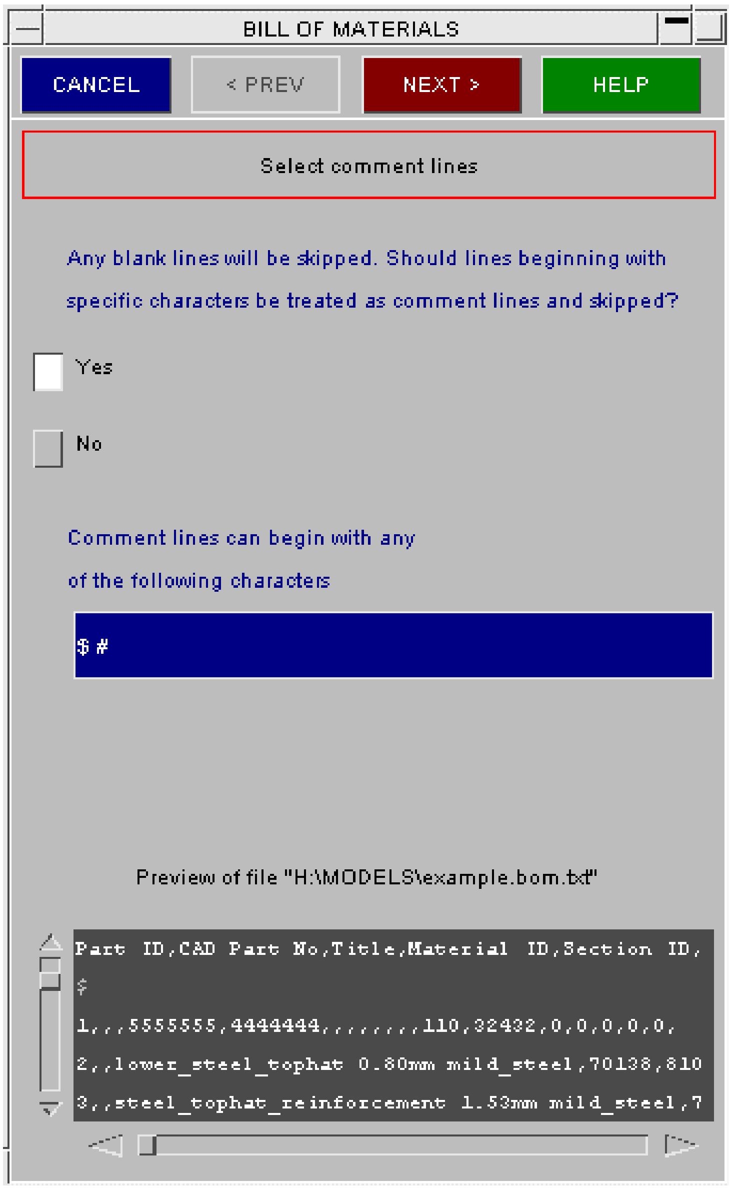


The radio buttons enable/disable error trapping when reading the part number field. In the example bill of materials file in the previous section the line

```
Part No,Title,Part ID,Material,Supplier,Gauge,Part mass
```

is not a comment line. If this line is read as an actual line of data an error could occur as instead of reading a number for the part ID, the string `'Part ID'` would be read instead.
If `'Skip any lines which have a blank or badly formatted PID'` is selected, the line would be skipped, a warning printed and the read will continue.
If 'Treat a blank or badly formatted PID as an error' is selected, this would be treated as an error and the read will stop.

Comment lines



Once the bill of materials file is selected it is scanned and a preview of the file is shown (the first 50 lines of the file are shown).

This preview can be used to help answer the questions which PRIMER asks. The scrollbars can be used to scroll the preview up and down and from left to right.

The default is not to skip any lines. To skip the comments the switch must be set to **Yes.** If the file does not contain any comments this step can be skipped.

To cancel reading and return to file selection press **CANCEL**.

To go on to the next step (skipping specific lines) press **NEXT >**

By default comment lines can begin with a $ or a #. Type the characters that you want comments to begin with into the blue box.

In the file preview any lines that will be treated as comments are shown in grey text instead of white text.

Skipping specific lines

In this example we want to skip the line that begins

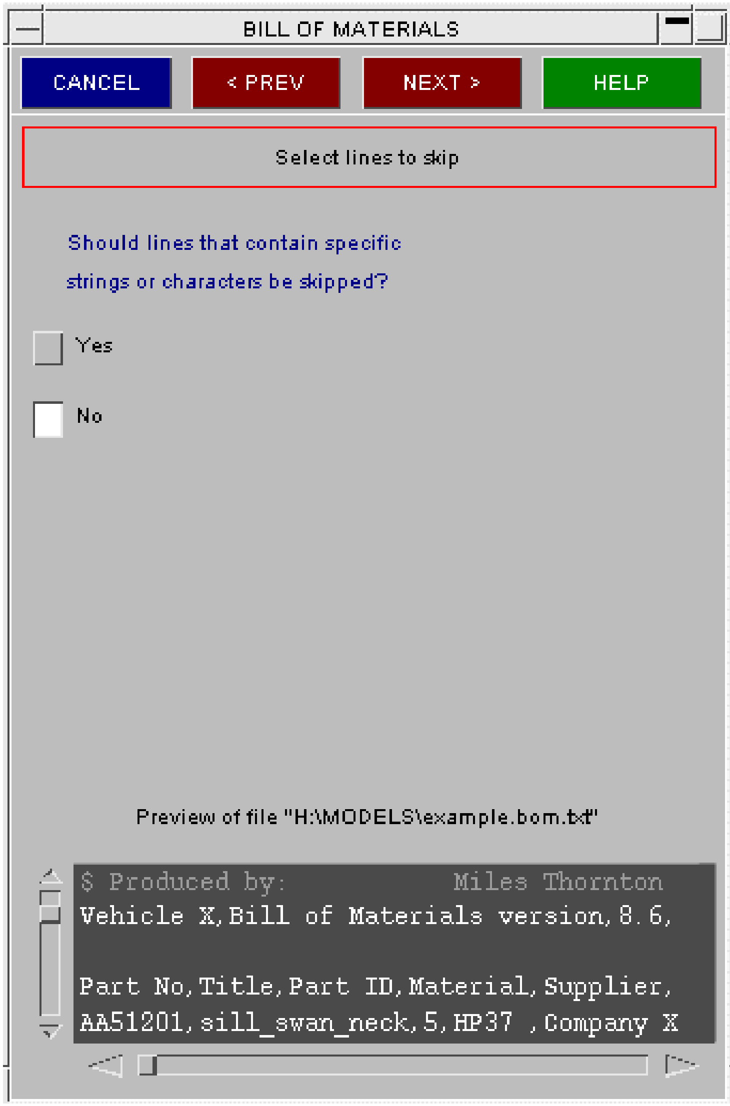
`Vehicle X, Bill of Materials`

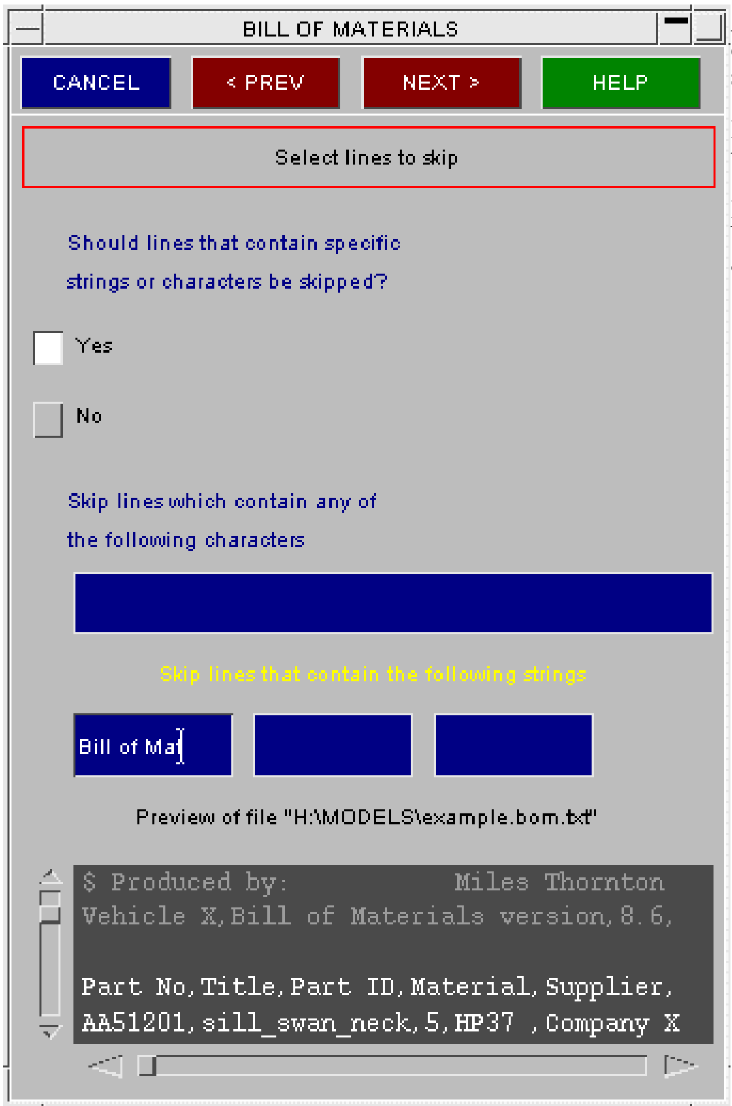
The default is not to skip any lines containing specific strings. To skip the line the switch must be set to **Yes.** If no lines need to be skipped this is not needed and can just be left at the default value.

To cancel reading and return to file selection press **CANCEL**.

To go back to the previous step (comment lines) press **PREV>**

To go on to the next step (selecting delimiters) press **NEXT >**





A line can be skipped that either contains a specific character or a specific string. Type the characters or strings into the blue boxes. Text is case sensitive.

In this example we have chosen to skip any lines that contain the string 'Bill of Mat'.

In the file preview any lines that will be skipped because they contain specific strings or characters are shown in grey text instead of white text.

Selecting delimiters

In this example the fields are separated by commas. e.g.

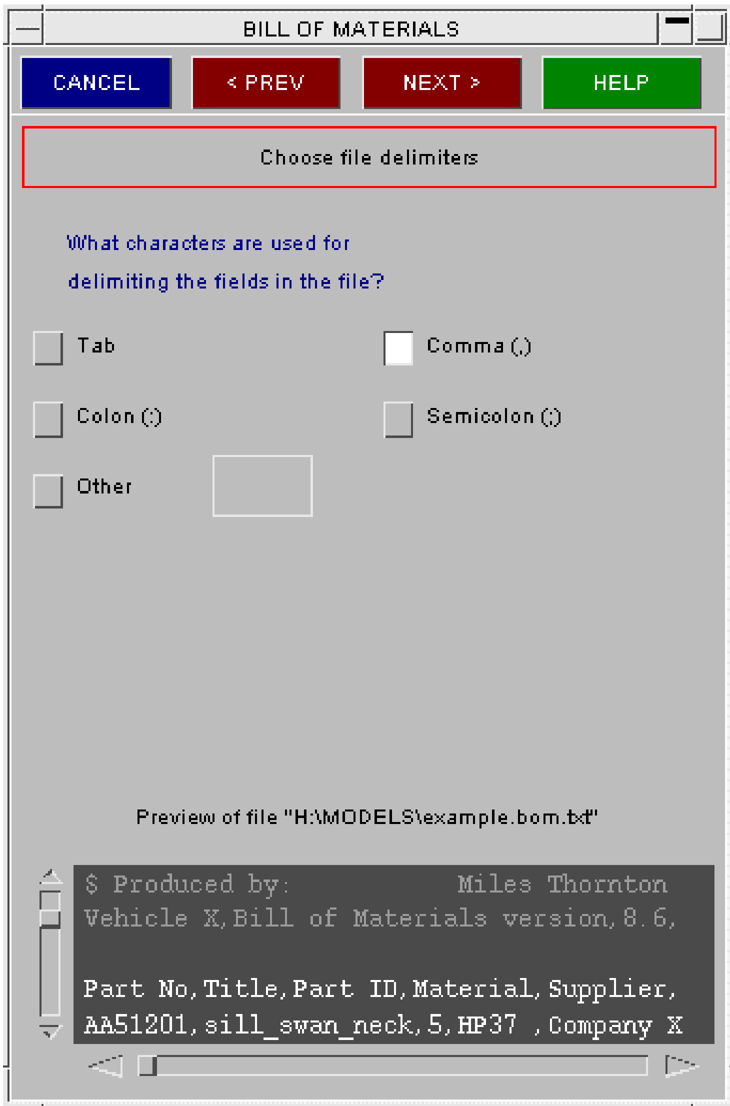
```
Part No,Title,Part ID,...
```

The default delimiter is a comma so this is OK for this example. If the data is separated by another character it can be chosen here. Other buttons are available for common delimiting characters. If your data is separated by a character that is not in the list press the **Other** button and type the character in the box. A space cannot be used to separate fields.

To cancel reading and return to file selection press **CANCEL**.

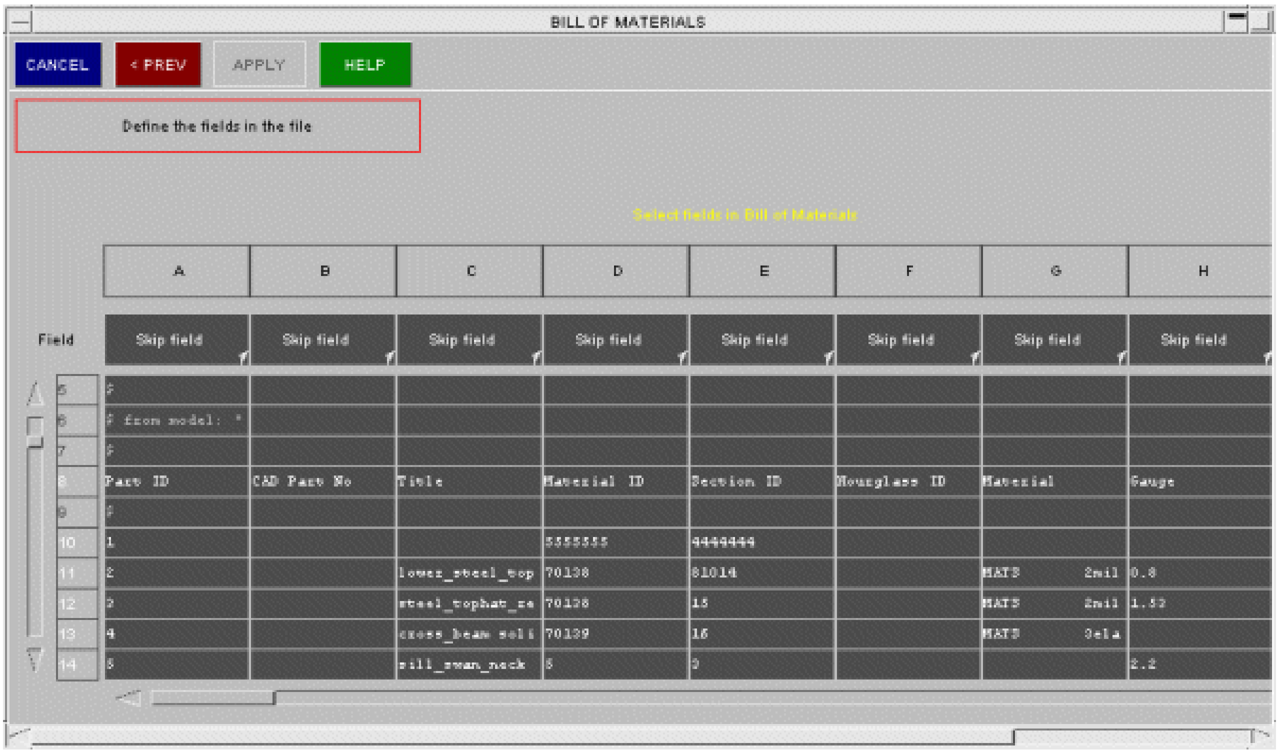
To go back to the previous step (skipping specific lines) press **PREV>**

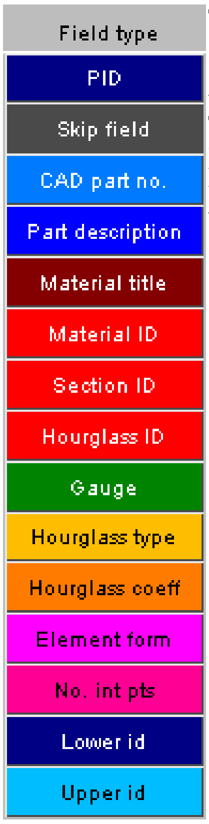
To go on to the next step (defining fields) press **NEXT >**



Defining fields

This panel enables you to choose which columns of the bill of materials to use and what the columns mean. A preview of the bill of materials is shown below.

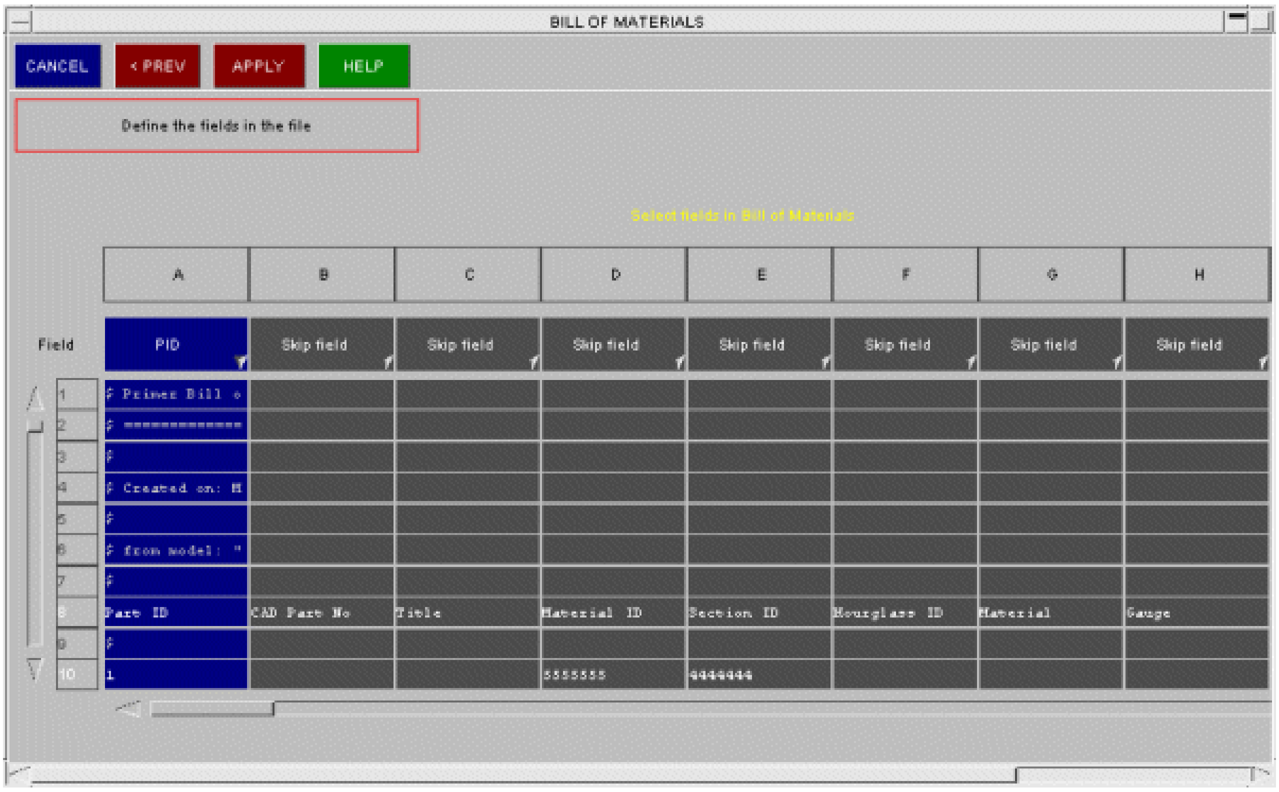




The lines that are going to be skipped are shown in grey rather than white. The data is shown in columns to make it easier to read. If there are more than 10 columns a scrollbar is used to view the other columns. To be able to do anything useful PRIMER needs to know which columns you want to use and what those columns mean. This is done by using the **Field** popup buttons in each column. The default action for each field is 'Skip field'. This can be changed by selecting any of the options from the popup. Once an action is selected the column will change colour and 'Skip field' will no longer be shown.

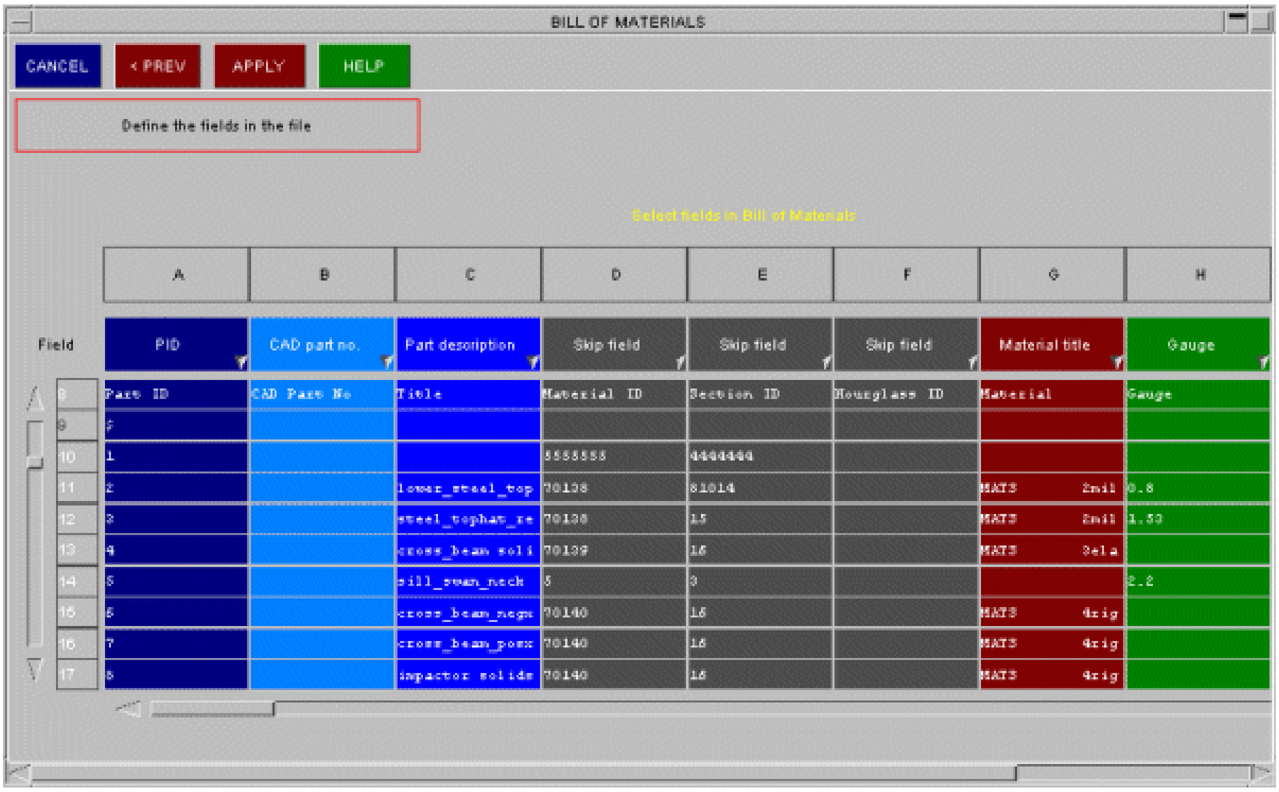
A field can be unset at any time.

For example, if the field for column A is set to be 'PID' it will be coloured dark blue as shown below.



At least the PID and one other field must be selected. The **APPLY** button will not be active until this is done. Once the button is active, **APPLY** will start reading the file and altering the selected fields.

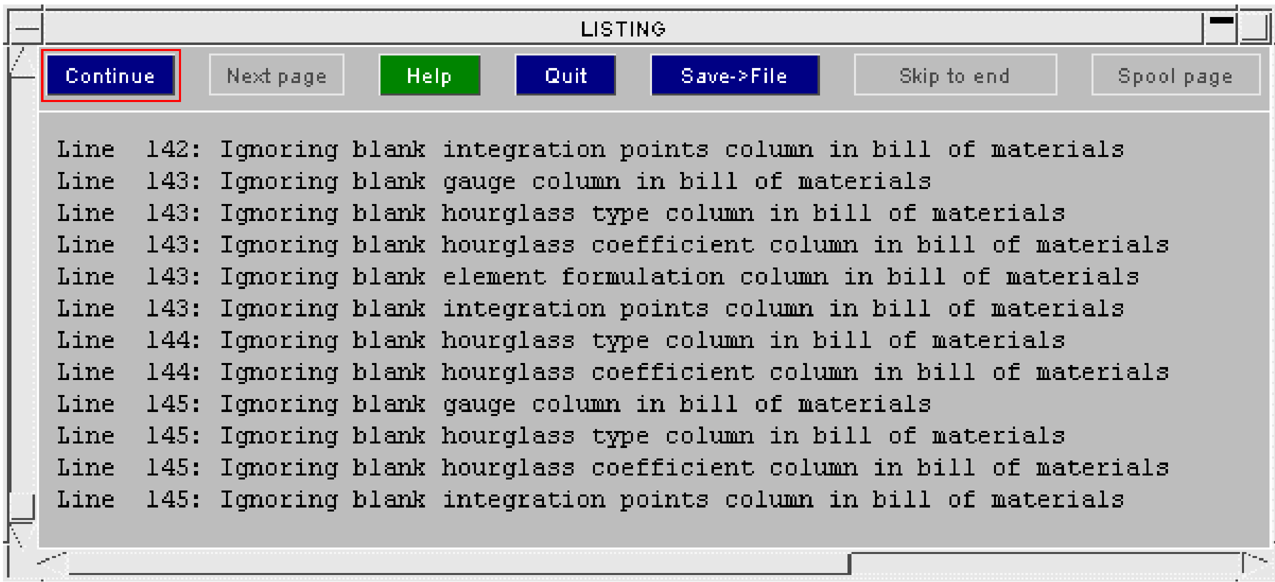
In the following example the CAD part no, Part description, PID, Material title and gauge have been selected.



Only the selected fields will be altered using the bill of materials. In the above example the CAD part no, Part description, Material title and gauge will all be altered as they have been selected. If only the PID and gauge were selected then only the gauge would change.

Listing output

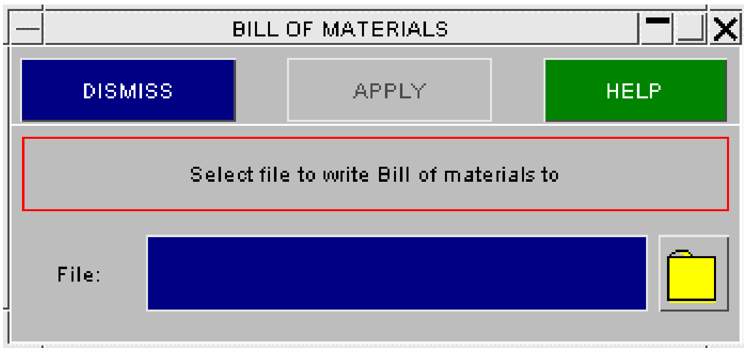
As the Bill of materials file is read messages are copied to a listing window.



This gives information about what the bill of materials is changing. If needed it can be saved to file by pressing the **SAVE -> FILE** button.

6.4.4 Writing a Bill of Materials file

To write a Bill of materials file type a filename into the text box or use the button to choose a file to overwrite. Once you have given a filename the **APPLY** button will become active and you can write the Bill of Materials.

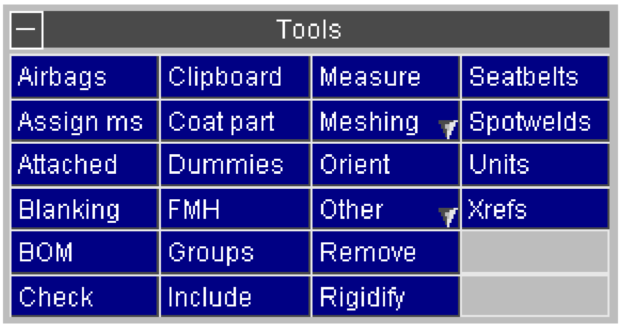


For each part in the model all the fields given in the standard headers above will be written. In addition 4 extra fields will be written:

- The part mass
- Mass added to the part from assign_mass structures
- Non structural mass added to shell parts (LS 960 and greater only)
- The total mass

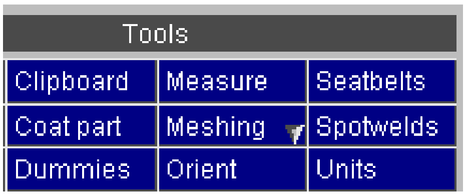
6.5 BLANKING Setting entity visibility.

The blanking menu in the **Tools** panel is covered in a separate section of the manual - see section 4.5 for details.



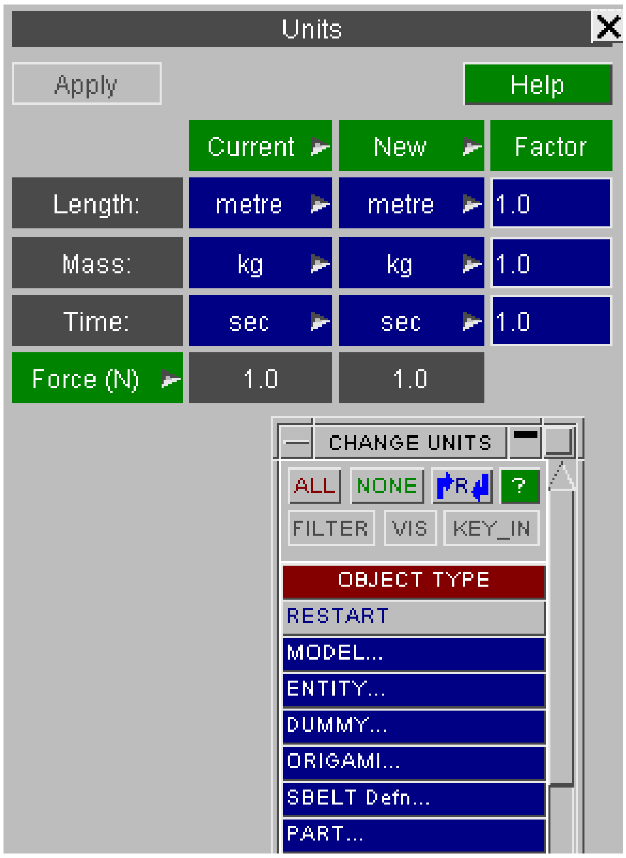
6.6 CHANGING UNITS

Length, Mass & Time



- Units of Length, Mass & Time may be scaled
- A whole model or a selection of items may be converted
- Curves belonging to multiple keywords will be copied automatically, if necessary

6.6.1 Selecting Units



To perform a unit conversion select the current units and the required new units. These are likely to be available under the Current unit and new unit popups, but if not may be selected from the individual unit popups.

The conversion factor will appear in the right hand column. If your units are not available from a popup, you may set the factor directly.

The force unit arising from the choice of units is displayed for your information.

Finally, select the items to which you want to apply the unit change. Primer will automatically propagate the change onto the appropriate items, e.g. selecting a part will get the section and material also.

Special treatment of loadcurves: if the same loadcurve is used by multiple keywords, primer will check the unit type of each and, if different unit types are found, offer the option of copying curves.

6.6.2 How Units change affects Parameters.

When a model uses parameters, and data fields defined by parameters are affected by a units change, then if no action were taken the value of the data fields would no longer be the same as the value of their parameters thus breaking the association between them. This would be unsatisfactory and parameters need to be updated during a units change, but this is not straightforward because parameters do not have intrinsic units.

PRIMER handles this problem as follows:

- During a units change operation any data fields which use parameters "tell" their parameter definition what factor has been applied to them.
- Each affected parameter "remembers" this factor temporarily.
- When the units change is complete all parameters are scanned to see whether factors should be applied, and for those affected:

  Scalar parameters are simply multiplied by this factor to give a new value.

Expression parameters are more difficult since they may be affected both by modifed scalar ones and by changes to each other. Therefore:

- Each expression parameter is re-evaluated, and the outcome compared with the <original value> * <any factor from units change>.
- If there is a mismatch then the outcome is factored to give the required value.
- This process is repeated iteratively, since expression parameters may reference one another in an arbitrary order, until no further changes are required.
  Once the correct factor has been determined for each expression then it is applied to the text string of that expression as a multiplier

This process effectively assigns an implicit unit type to parameters and scales them appropriately, so that their new values still match the scaled data fields to which they apply, which means that the association between parameters and data fields remains unbroken.

Situations in which units change of parameters may fail.

This process normally works well, but can fail for either or both of the following reasons:

If a parameter is used inconsistently:

For example if a parameter is used in a data field of length units, and also one of time units, then two different factors may be notified to the parameter definition. In this situation the most recently notified definition "wins" and will dictate the factor applied to the parameter, meaning that its association with some of the other data fields will be broken.

The solution is obvious: don’t use a parameter in incompatible contexts.

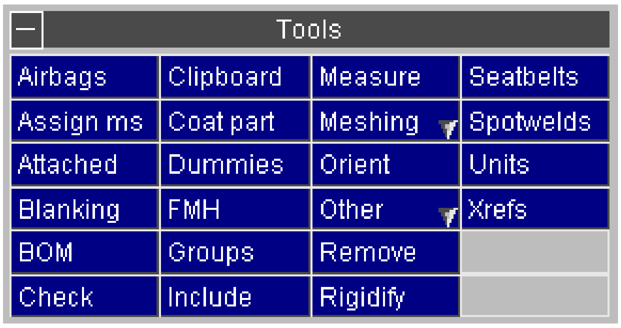
If only part of a model is subjected to units change, but parameters are used in all of it:

In this situation factors from changed fields will cause the parameters to be updated, which may break their association with unchanged data fields.

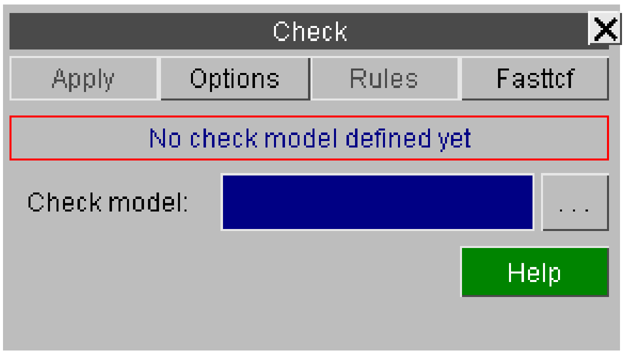
The solution is not to change units of a subset of a parameterised model. It is normally the case that a units change is required when models from different sources are merged together, and the units change should be applied to the model as a whole ***before*** it is merged.

6.7 CHECK Running the model checker, and setting its options.

The model checker is run from the **CHECK** option in the **Tools** panel - see section 3.9 for details.



6.7.1 The CHECK popup menu



APPLY Runs the model checker as described in section 3.9.

OPTIONS Maps the **CHECK** options panel as described in section 3.9.1

6.8 CLIPBOARD

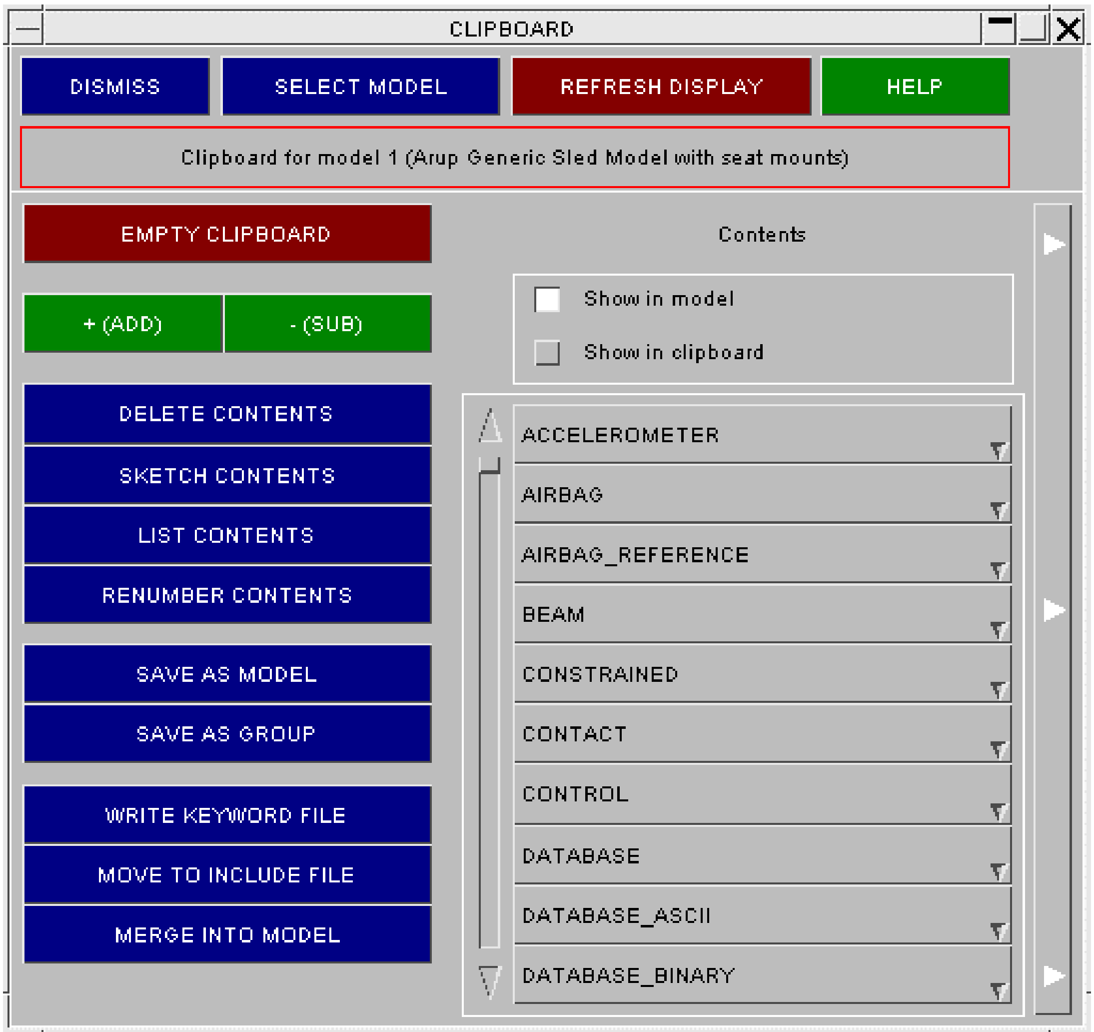
The clipboard function allows the user to work on a subsection of a model. Each model has its own clipboard and the clipboard contents remains part of the model. The clipboard contents can be;

- renumbered - this is useful for renumbering nodes and elements by part
- interrogated for their locations - useful when a model contains many include files
- saved as a group or new model - a quick way of obtaining a complete sub model
- written as a separate keyword file - useful when only specific parts need to be modified in a separate application
- moved to a new or existing include file - useful for model organisation
- merged into another model - a quick way of building new models
- reuse of selections in general object menus

The clipboard panel is displayed below

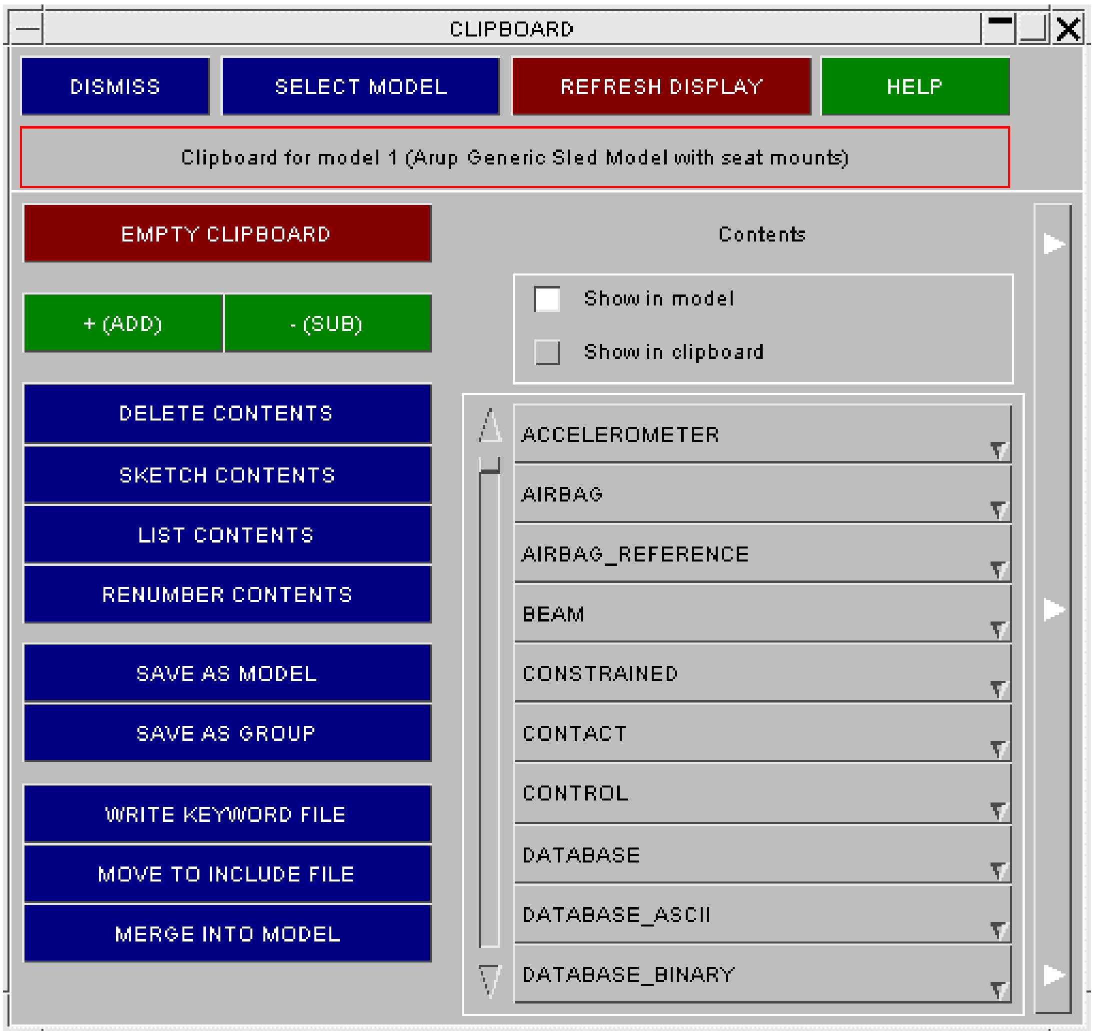
6.8.1 Adding and Removing items from the clipboard

Items can be added and removed from the clipboard by using **+(ADD)** and **-(SUB)**

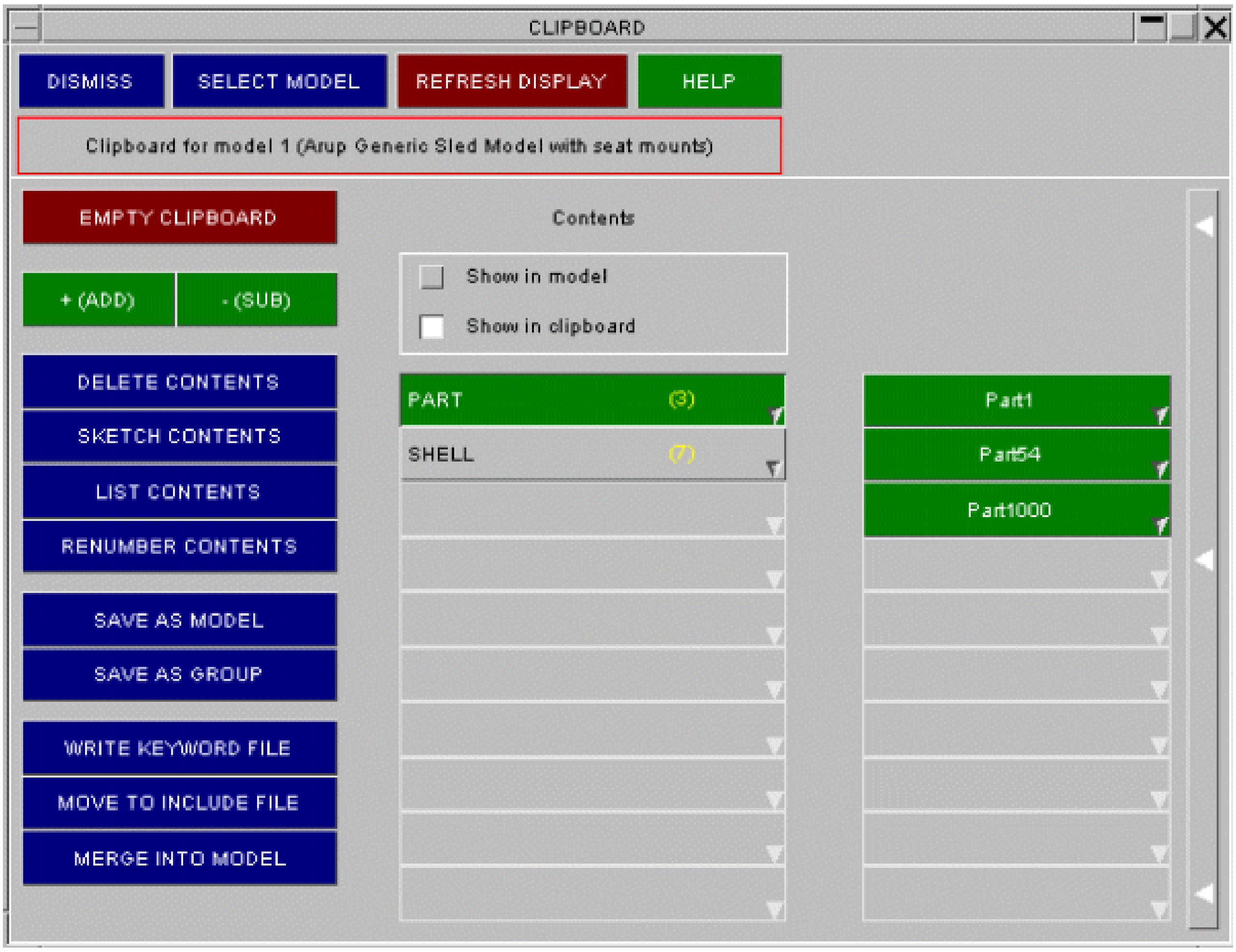


6.8.2 Display of Items on the Clipboard Panel

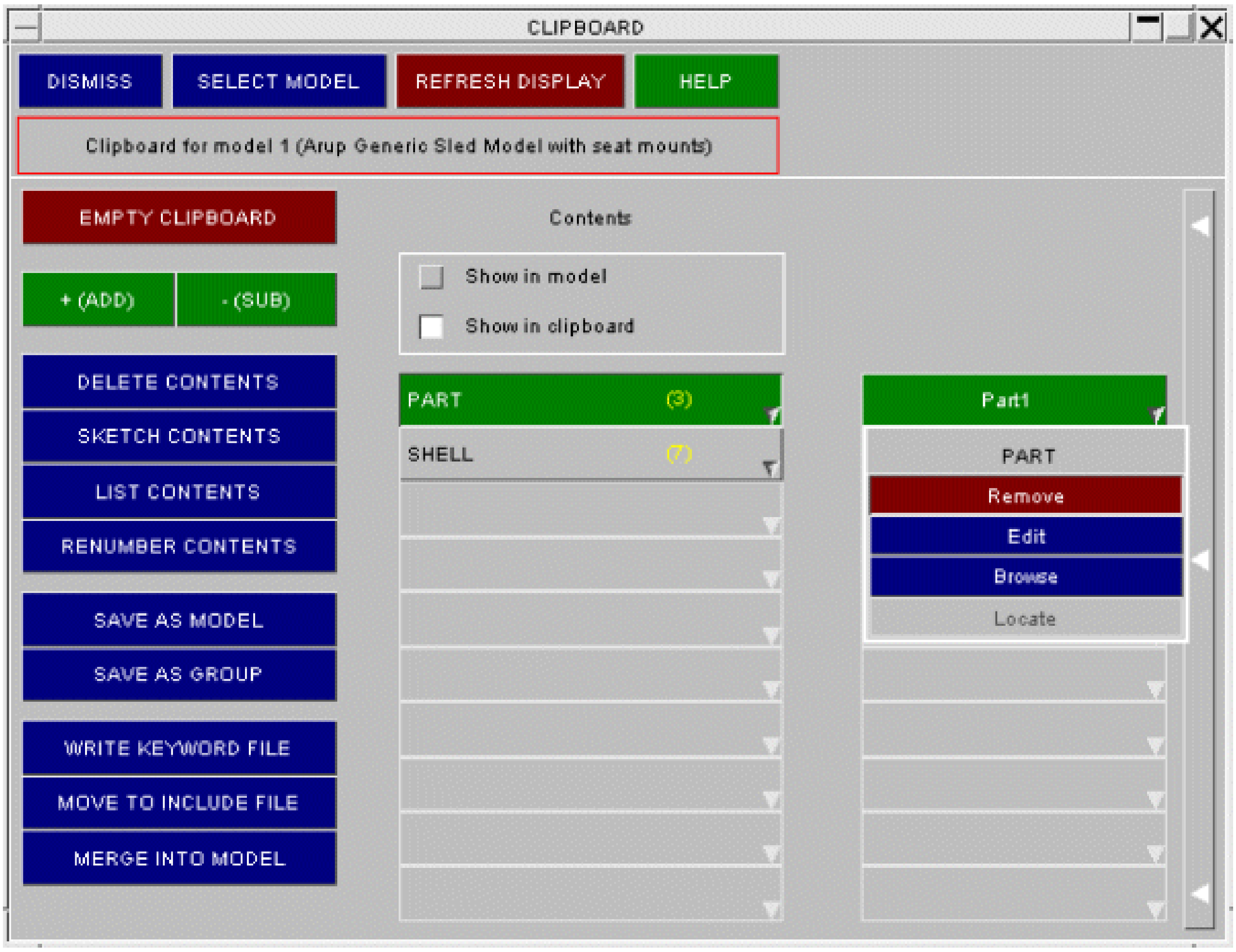
There are three modes of displaying items in the clipboard panel. The contents display can be set to show all types of entities in the model by activating **show in model**. If entities are in the clipboard the number present will be displayed in brackets. **Show in clipboard** will show only entities present in the clipboard with the number present again displayed in brackets.



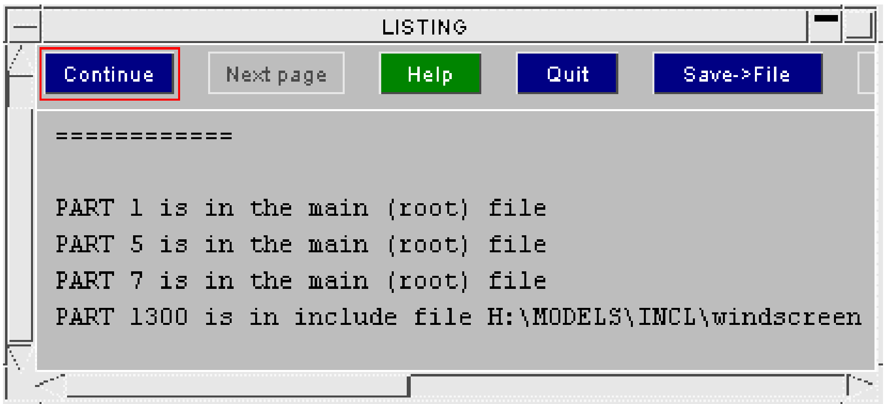
The bar at the right hand side expands the clipboard panel to show all members of an entity group present in the clipboard once that entity is selected by left-clicking the mouse over its button.



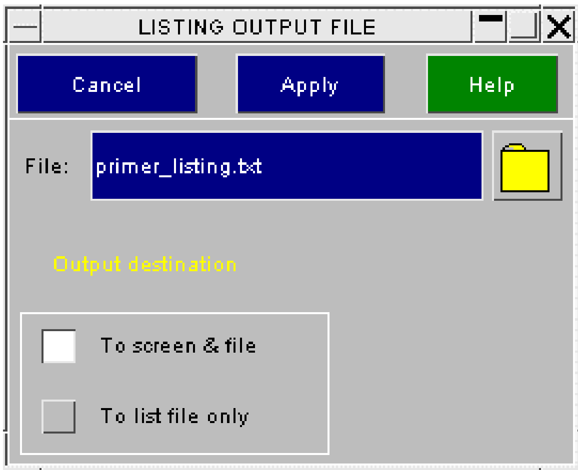
Right-clicking on the items in the right hand column will allow removal, editing or location of that item.



The location of all of a particular group of entities can similarly be found by right-clicking on the relevant entity button. Selection of **locate** will bring up a panel that will scroll through all entities present in the clipboard with their locations.



This information can also be output to a text file.



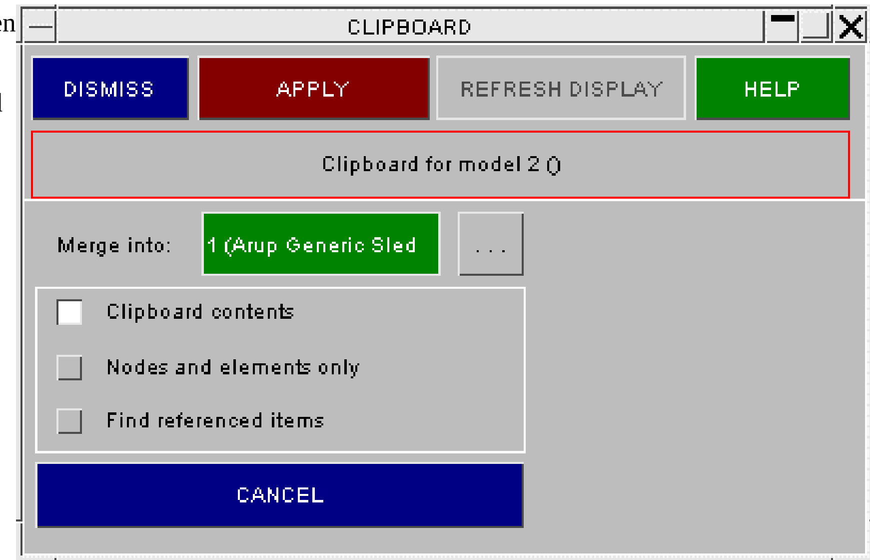
6.8.3 Referencing of Clipboard Items

When saving the clipboard contents to a new model or writing a keyword file there are three options to select the entities that are written.

1/ The clipboard contents can be written as they are.

2/ Nodes and elements in the clipboard and those belonging to parts in the clipboard can be written.

3/ Items referenced by entities in the clipboard can be included if desired.



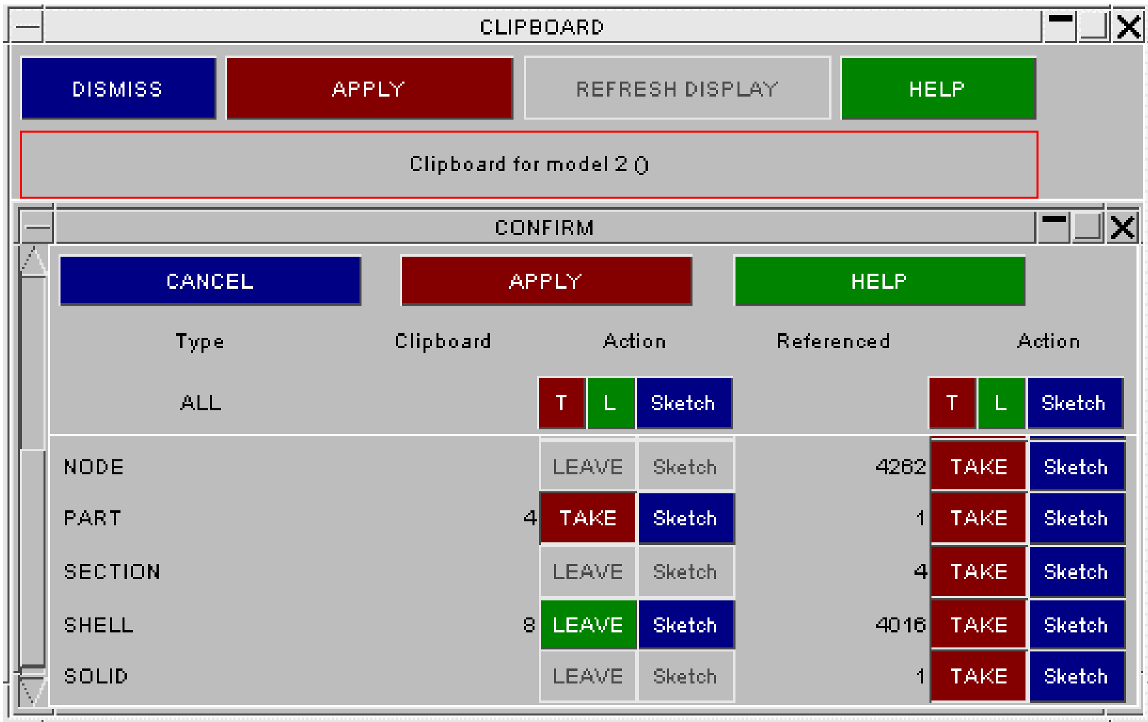
Option 3 is useful as PRIMER will find all entities necessary to produce a complete stand alone model or file.

PRIMER finds entities according to a hierarchy. Some examples of this follow;

- When finding referenced items, a clipboard containing an element alone would first find the nodes of that element and the part referenced by the element. However this is not a complete model as the material and section of the part also need to be included. PRIMER therefore carries on checking until all necessary entities are found. A clipboard containing only nodes would not find any elements or parts as these are not referenced and are 'downstream' of nodes in the PRIMER hierarchy.

- A clipboard contains a ***CONTACT_** entity. If the contact referenced a ***SET_PART** then this would be found together with the parts on that ***SET_PART**. Elements and nodes of the parts would also be found.

When PRIMER has finished finding all entities the following panel is displayed and at this stage the user has control over selection of items by clicking on the relevant button to take or leave each group of entities. Items originally present in the clipboard are displayed in the centre column and referenced items are shown in the right hand column of the panel.



6.8.4 Renumbering of Clipboard entities

This function makes node and element renumbering by part association a simple task.

The panel shows what appears when the clipboard renumber button is operated. The clipboard contains one part and referenced nodes and elements are found and automatically selected for renumbering.

Full details on use of this panel are at **RENUMBER**

6.8.5 Saving Clipboard entities as a new model/keyword file

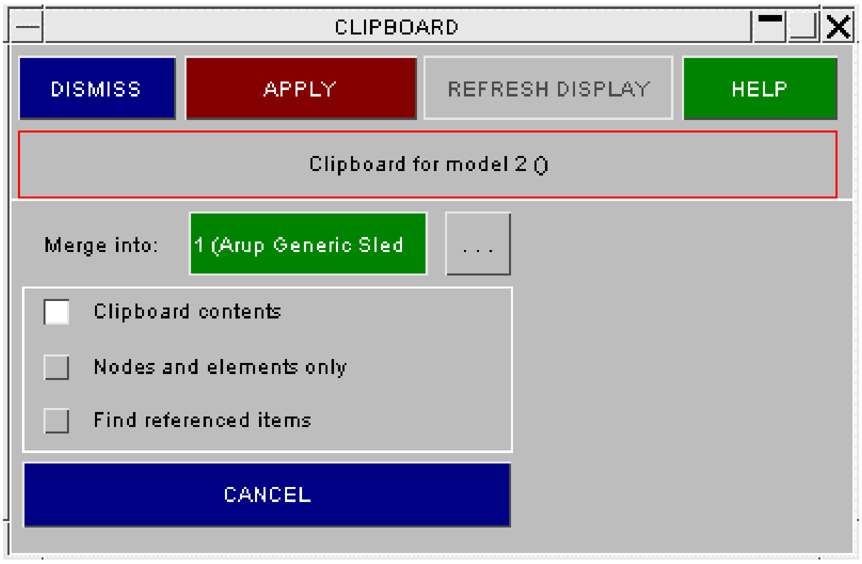
Sub sections of a model can be added onto the clipboard and then saved as a new model in PRIMER or written out as a new keyword file. The clipboard allows you three options:

1/ The clipboard contents can be written as they are.

2/ Nodes and elements in the clipboard and those belonging to parts in the clipboard can be written.

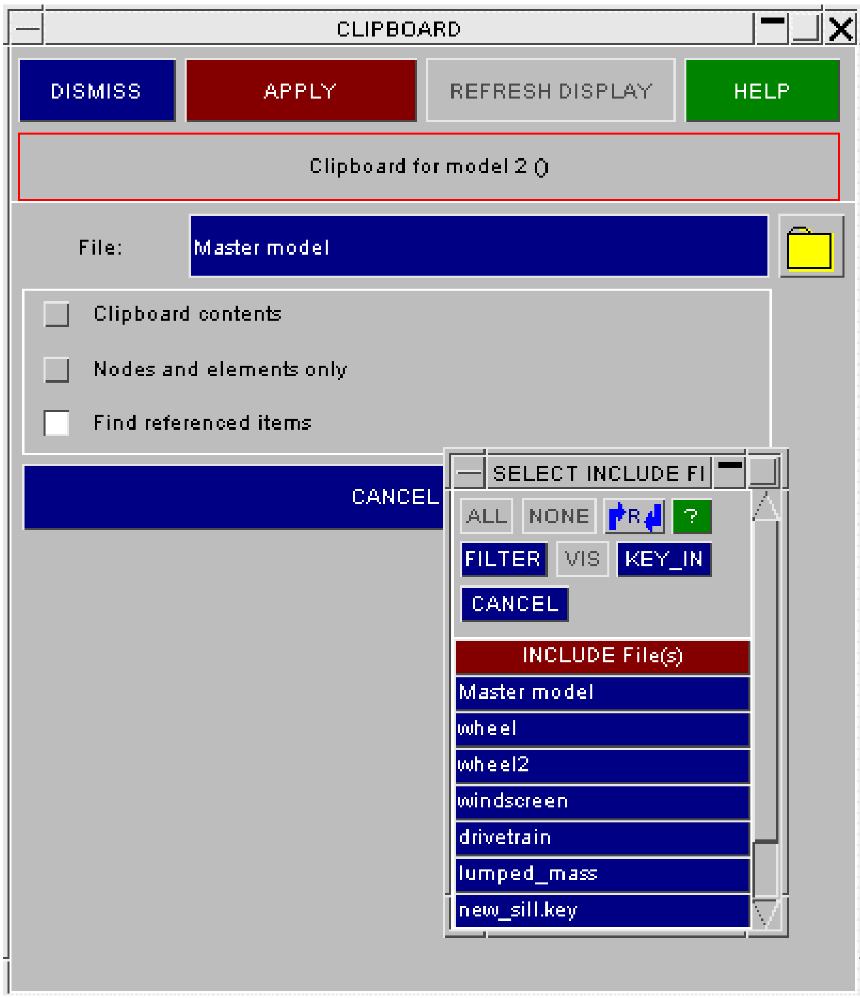
3/ Items referenced by entities in the clipboard can be included if desired. In this case a complete stand alone model will be produced.

When writing the clipboard contents to a new keyword file, you can access the writing options by clicking on the **>>> LS-Dyna output options** button.



6.8.6 Moving Clipboard entities into include files

The clipboard provides a powerful and controllable way of moving items into include files or from one include to another.



The panel above appears when the **move to include file** button on the main clipboard panel is operated. There are three possible options that allow the user control;

1/ The clipboard contents can be moved

2/ Nodes and elements in the clipboard and those belonging to parts in the clipboard can be moved

3/ Items referenced by entities in the clipboard can be moved

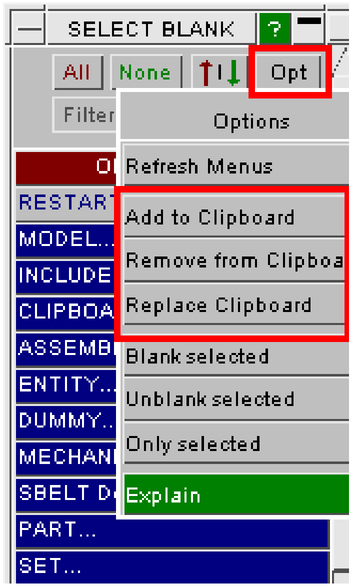
Depressing the [?] button will bring up current include files. These can be selected directly from the subpanel or a new filename typed in the box after **File:**

Selecting **Find referenced items** will bring up a panel allowing the user control of referenced items

6.8.8 Clipboard merge into model

This allows items in the clipboard to be merged into another existing model in the PRIMER session. This function works in exactly the same way as **MODEL > MERGE**

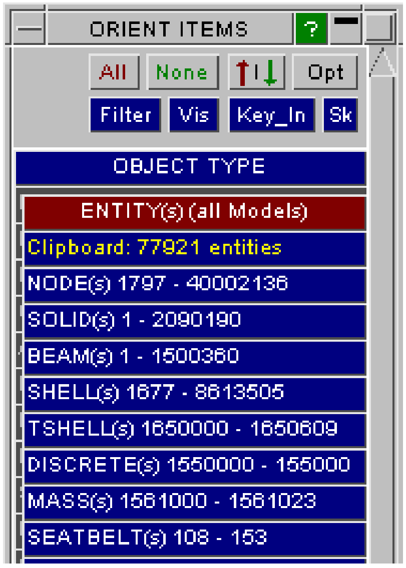
6.8.9 Clipboard usage in Object Menus



The Options button in all standard Object Menus includes options to:

- Add the current selection to the clipboard
- Remove it from the clipboard
- Replace the clipboard with the currently selected objects

This acts as an alternative way of manipulating the clipboard contents.



The clipboard contents may also be used for selection in object menus, as this example illustrates.

In all object menu contexts where:

- Multiple selection is legal

  and

- The clipboard contains 1 or more items of the specified type

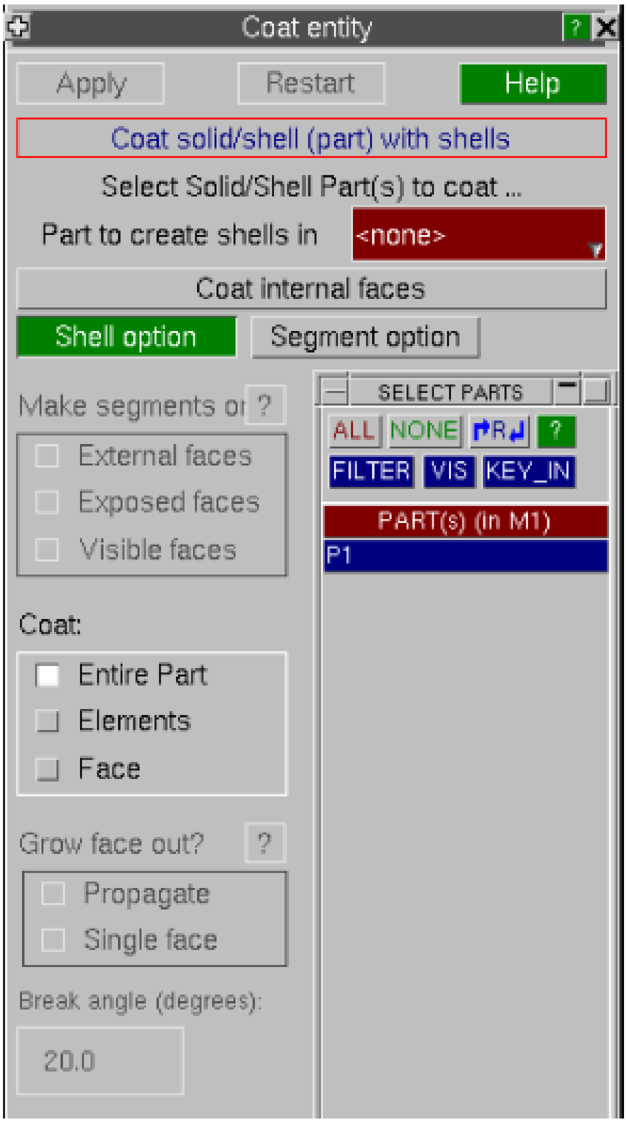
Then a "**`Clipboard: nnnn <item type>`**" row will appear at the top of the menu selection.

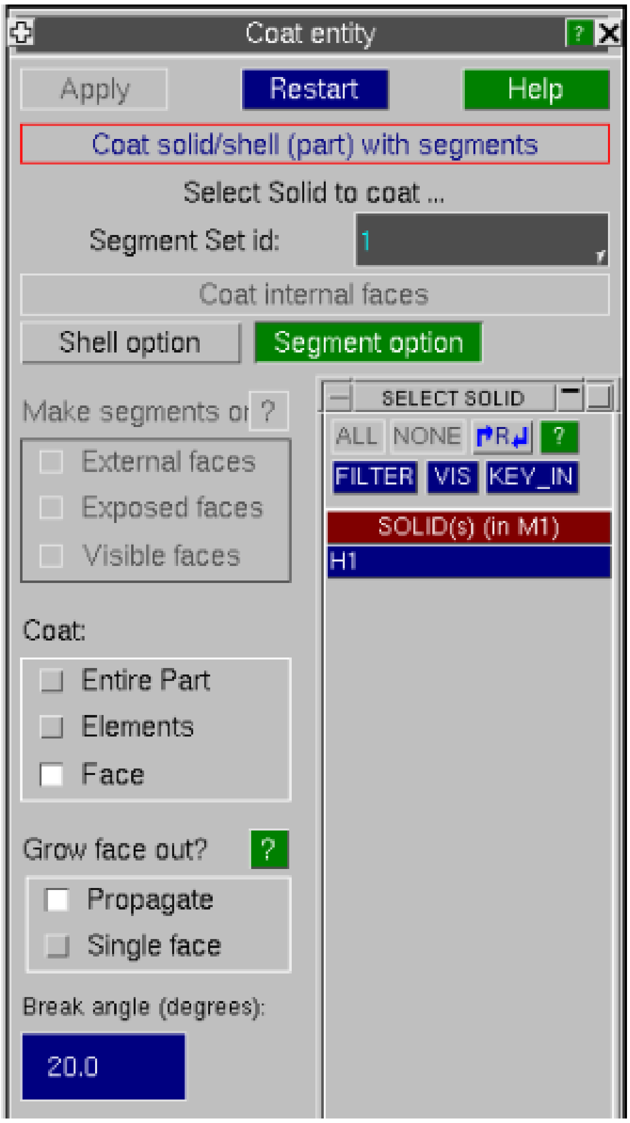
If this is selected then all items in the clipboard which are also legal in this context may be selected in a single click by choosing this "Clipboard" row. Items in the clipboard which do not match the currently specified type are not included.

This provides a means of reusing selection in a range of different contexts.

6.9 COAT ENTITY: Coating entities with shells or segments

It is often the case that a solid part needs its external surface coating with shell elements (usually null shells) for purposes such as defining contacts. This process is laborious when carried out manually, and the **COAT ENTITY** function is designed to do it automatically.





Coat Part

The function is generalised so that it will coat any solid or shell part(s) with either shells or segments. Furthermore, when coating solid parts with shells the option to coat internal faces is offered.

Any number of solid/shell parts may be selected through the respective object menus. For each coating operation, the new shells/segments created will always reside in a single part/set. Thus if two (or more) adjoining solid parts are *selected and coated together* (with the internal face option off for shell case) only the external faces will be coated.

Coat Elements

Selecting this option will permit users to coat specific solid or shell elements instead of a part.

Coat Face

This option allows users to coat specific solid faces with shells/segments. Two additional options are available for coating solid faces:

- **Propagate** - Users are directed to pick a face on any solid or shell element, and the associated break angle. All faces on that element and on adjacent elements that define an angle with the selected face that is less than the break angle are coated. This is the default mode for the **Coat Face** option.
- **Single Face** - The selected face on a solid element is coated. All other faces are ignored.

Coating with shells:

Part to create shells in may be an existing or new part in which the "coating" shells will be created. This can be any valid part, with "null" or ordinary structural materials.

Three options are available for coating element faces with shells:

- **External faces** - For 3D elements, shells are created on topologically external faces of the selected elements. These would be faces that would be visible if no elements of this type were blanked.
- **Exposed faces** - For 3D elements, shells would be created on the exposed faces of the selected elements. These would be the elements that would be visible if all but the selected elements were blanked.
- **All faces** - All selected faces including internal faces will be coated.

Coating with segments:

Set for segments. By default the highest+1 set id will be displayed. The user may select any existing set or type in the id of a new one. In the former case the new segments will be added (without duplication) to the set.

Three options are available for coating element faces with segments:

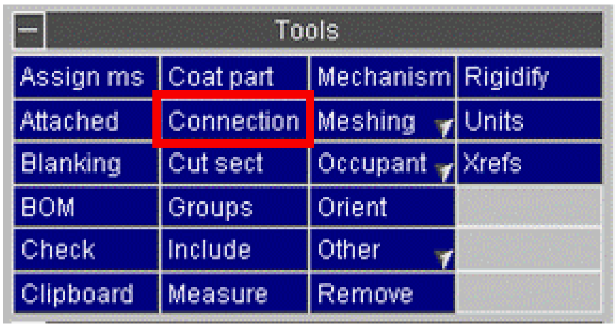
- **External faces** - For 3D elements, segments are created on topologically external faces of the selected elements. These would be faces that would be visible if no elements of this type were blanked.
- **Exposed faces** - For 3D elements, segments would be created on the exposed faces of the selected elements. These would be the elements that would be visible if all but the selected elements were blanked.
- **Visible faces** - For both 2D and 3D elements, segments are created on only those faces that are visible in the current view.

Once the parts have been defined **APPLY** creates shells/segments.

NOTE: Segment sets may also be created through **SET->SEGMENT->CREATE->COAT ELEMENTS** option.

Multiple (coincident) coatings may be applied to a part by calling this function repeatedly - any existing shells attached to solid faces are ignored.

6.10 CONNECTIONS



A connection is a new PRIMER entity introduced in version 9.3. It allows PRIMER to create/modify/delete **mesh independent** spotwelds, bolt connections and adhesive runs. Spotwelds consist of beams or hexahedral elements tied to the panels using a tied contact. Bolts are rigid connections between panels. Adhesives consist of runs of hexahedral elements tied to the panels using a tied contact.

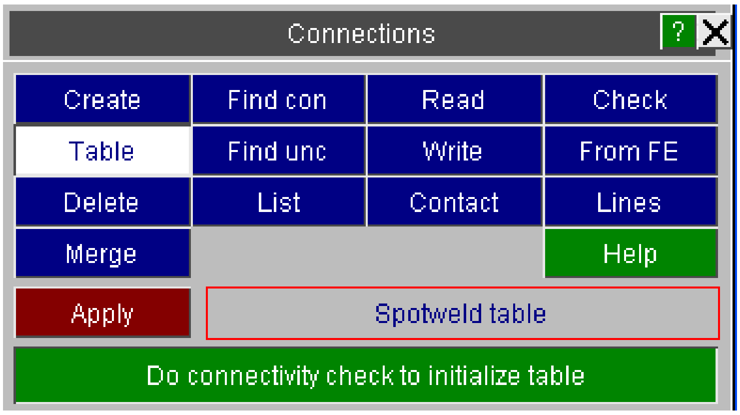
The **Connection** panel is used for all aspects of managing connection data.

The connection entity allows PRIMER to store all of the information that makes up the appropriate connection entity. That means that for example, it is possible at any time to change a beam spotweld into a solid spotweld or a bolt. As PRIMER knows what entities make up the connection it can delete the old entities and make new ones as required. The connection can be drawn (a 'blob' is drawn at the connection point, or a line indicating the path of an adhesive run) or labelled using the entities panel. The colour of the connection is drawn in depends on the state of the connection. The following colours and their meanings are used.

Colour	Meaning
Green	Realized. The connection is made and it does not have any errors
Blue	Provisionally realized, no contact check has been done
Red	Bad. The connection cannot be made because there is a problem
Orange	Invalid. The connection has been made but there is something wrong with it (e.g. the node is not tied correctly)
Yellow	Not checked. The connection has been made but PRIMER has not yet checked it to see if it is OK or not
Cyan	Latent. The connection point exists but it has not been made yet

The panel allows you to create, review, modify and delete connections. A 'connection file' can also be read by PRIMER to connect an entire structure very easily. Additionally, tools are available for checking and correcting bad connections as well as finding connected or unconnected panels. The initial spotweld panel is shown below. If your model does not contain any mesh independent spotwelds only the **Create**, **Read** and **From FE** options will be available.

The following options are available from the **Connection** panel.



Creating spotweld connections or bolt connections
Modifying/Reviewing an existing connection using the TABLE
Deleting spotwelds
Merging connections

Using find connected to find connected panels
Using Find unconnected to find unconnected panels
Listing connection data
Reading a connection file (PRIMER spotweld file format)
Writing a connection file (PRIMER spotweld file format)
Checking the spotweld contact
Checking spotwelds
Creating connection from existing FE entities
Modifying connections by creating lines

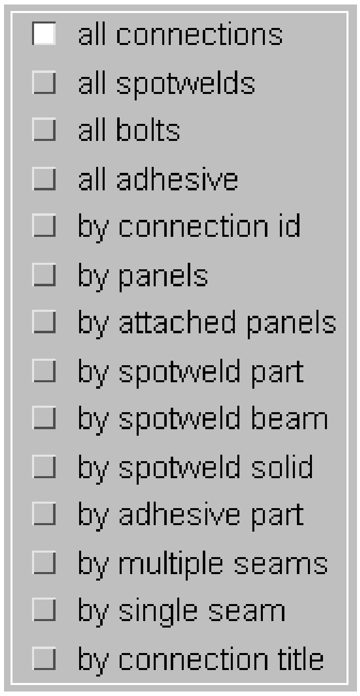
There are several options that control how connections in PRIMER work.

To achieve realized status a connection must be checked using the contact checker. Normally this is done automatically before the table is displayed. For very large models with multiple contact definitions this may take while, so the user may elect to postpone the connectivity check. In this case a simple geometric check is made, there is no guarantee that the weld will tie or even be present in a tied contact! Hence the connections will be displayed as blue - provisionally realized.

6.10.0 Methods of selecting connections

Several of the connections functions (e.g. **Table**, **Delete**, **List** etc.) allow you to select which connections you want to work on by several different methods.

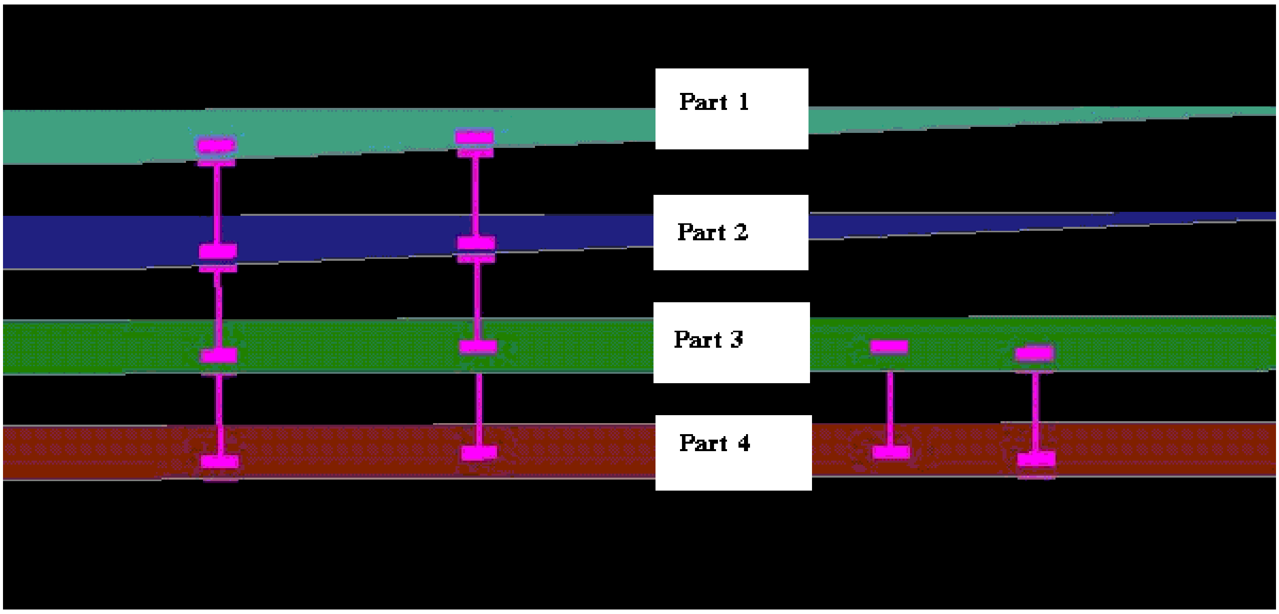
- **all connections**. All the connections in the model are selected.
- **all spotwelds**. All the spotwelds in the model are selected.
- **all bolts**. All the bolt connections in the model are selected.
- **all adhesive**. All the adhesive connections in the model are selected.
- **by connection id**. You can select which connections to modify by picking or using the object menus.
- **by panels**. Connections that use any of the selected parts in their layer definitions. Note that the connection does not have to be made for this.
- **by attached anels**. Any connections that are attached to any of the panels you select. Note this implies that the connection is ’realized’
- **by spotweld part**. Any spotweld beams or solids using the specified part(s).
- **by spotweld beam**. Choose connections by spotweld beam.
- **by spotweld solid**. Choose connections by spotweld solid.
- **by adhesive part**. Any adhesive runs using the specified part(s).
- **by multiple seams**. Any connections that only use some (or all) of the selected parts (see Multiple or single seam selection for a more detailed description).
- **by single seam**. Connections that use all of the selected parts (see Multiple or single seam selection for a more detailed description).
- **by connection title**. A box opens up to enter a title search string.



Multiple or single seam selection

When selecting by **multiple seam** the connections and/or their related entities that are attached to **ANY** of the selected parts **AND NOT** attached to **ANY** deselected parts will be selected. For example, in the figure below, if part 3 and part 4 are selected then the two beams on the right will be chosen. If parts 1, 2, 3 and 4 are selected then all 4 beams will be chosen.

When selecting by **single seam** the connections and/or their related entities that are attached to **ALL** of the selected parts **AND NOT** attached to **ANY** deselected parts will be selected. This will only ever be one seam. For example, in the figure below, if part 3 and part 4 are selected then the two beams on the right will be chosen. If parts 1, 2, 3 and 4 are selected then ONLY the two beams on the left will be chosen.


Part 1
Part 2
Part 3
Part 4

6.10.1 Creating connections

Automatic creation of connections from welds

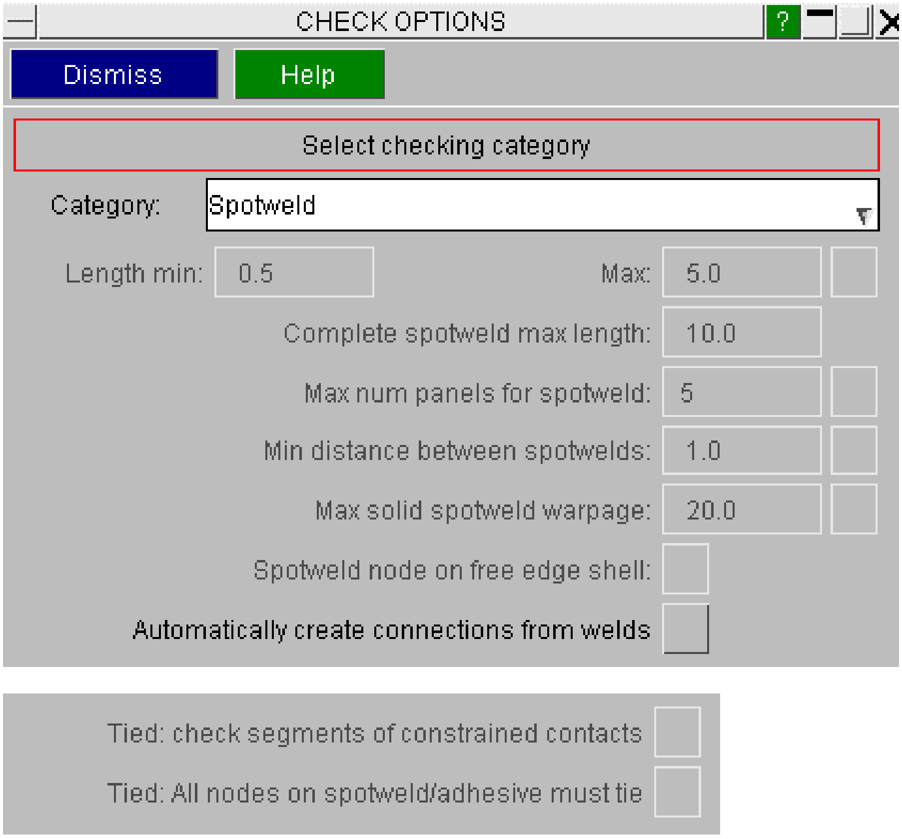
Management of spotwelds by connection entities is fundamental to Primer - weld creation, deletion of welded shells, weld checking, find attached, etc. All welds created in Primer will have a corresponding connection, maintained as post-end keyword.

As read models may, however, contain welds which do not have connections. By default, Primer will attempt to create connections from any existing MAT100 welds (beams or single solid) which do not already have them

- when the connections tool is activated
- when a model check is done
- when shells or shell parts are being deleted

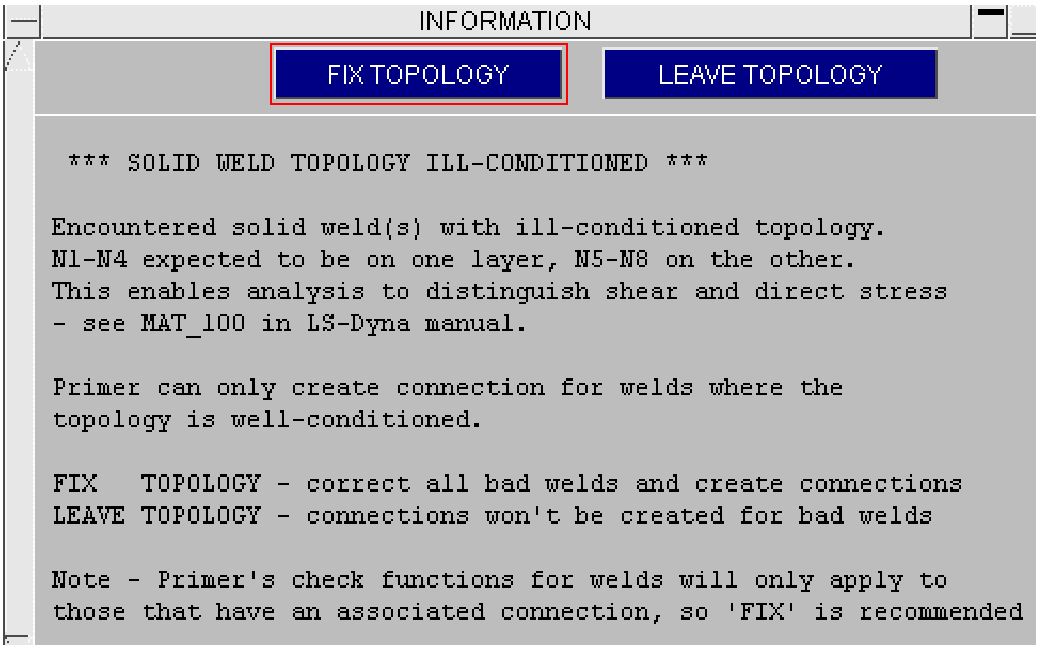
Primer will warn in the dialogue box when a model check or deletion operation has created connections. These connections are marked and will be ignored when the model modified function is applied.

Whilst this methodology is recommended, it is possible for the user to inhibit the automatic creation by the setting under **CHECK > OPTIONS > SPOTWELD**. This will also inhibit the checks which rely on connection logic **for all spotwelds**.



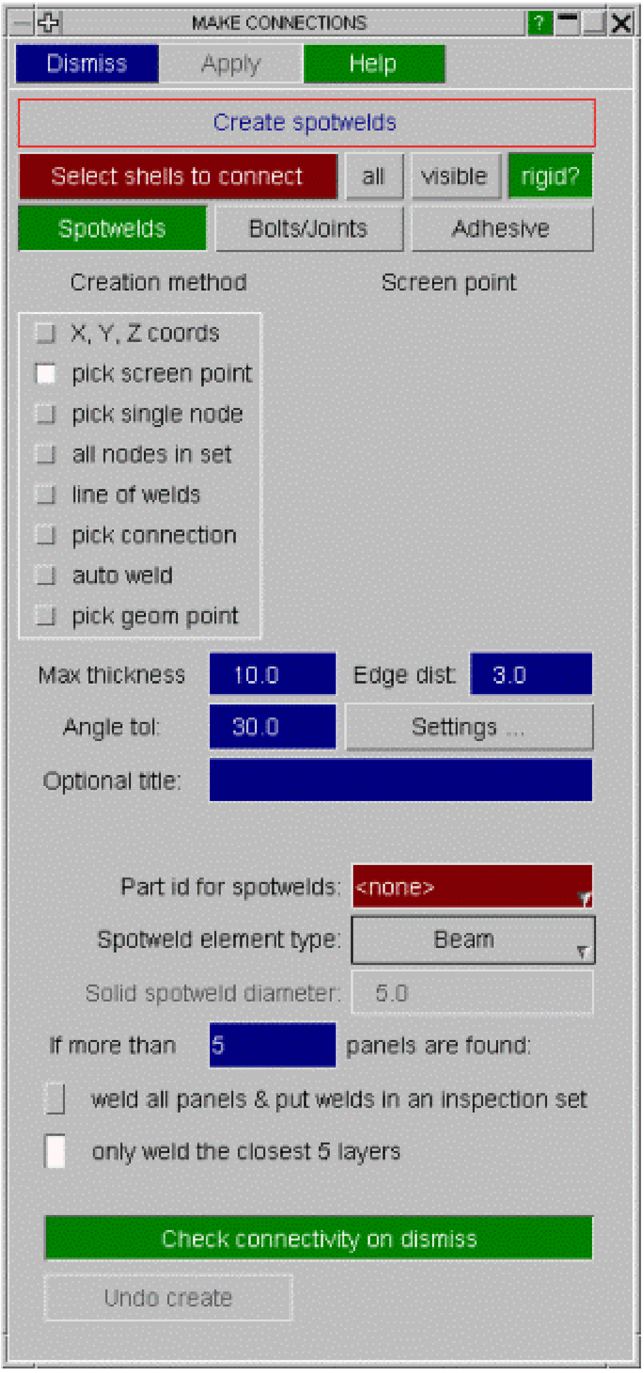
Fixing of solid topology

Creation of connections from solid welds (excluding nugget welds) requires that the solid topology is correctly configured. Nodes N1-N4 should be on one layer and nodes N5-N8 on the other. LS-Dyna actually requires this if direct and shear stresses are to be correctly calculated. An information panel will report the problem and give the user the option of fixing the topology.



Creating spotwelds

The connection creating panel allows you to easily create spotweld, bolt/joint and adhesive connections. The panel is shown on the below.



Before any connections can be created PRIMER needs to know 3 things:

1. The spotweld element type
2. A part to put the spotweld elements in.
3. Which shells to use during the spotwelding process.

When these steps are done you can start creating spotwelds. You can make a spotweld by either:

- Typing in the X, Y, Z coordinates. The spotweld will be created at this point.
- Picking an arbitrary screen point. This does not have to be a location of a node. The spotweld will be created at this point.
- Picking a connection from the model. The spotweld will be created at the connection location.
- Picking a single node from the model. The spotweld will be made at the node location.
- Picking a minimum of 2 screen points to create a line. Spotwelds will be created on this line. The number of spotwelds can be defined by number or pitch
- Using a node set. The default behaviour is now to replace each *Constrained weld which uses any nodes of the set. Previous method of making a spotweld at each node is still available.
- automatically detecting flanges and creating welds.
- Selecting geometry points that exist in any model read into PRIMER.

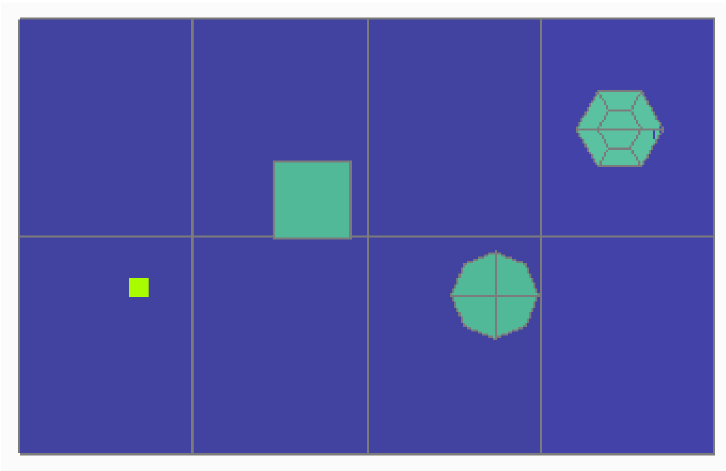
There are various options that can be set to control how spotwelds are made.

Choosing the spotweld element type

First, the type of spotweld connection must be defined. Spotwelds can either be beams or solid elements.

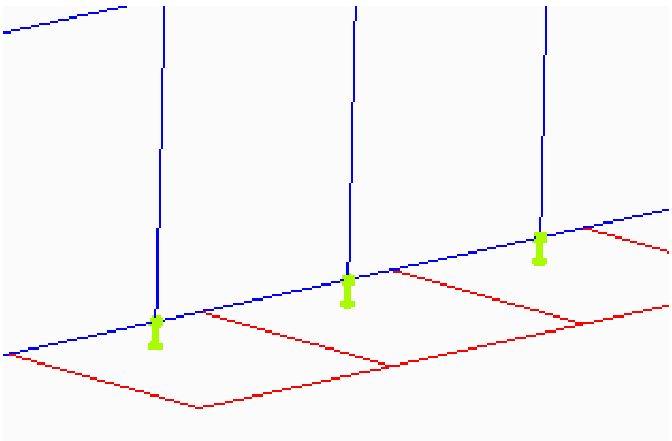
To create spotweld connections, PRIMER offers the following options:

The Beam and Hexa options allow you to create mesh independent spotwelds using a single beam, a single solid or multiple solids between panels. The image on the right shows examples of Beam, Hexa, 4 Hexa and 8 Hexa welds.
If one of the solid Hexa element options is selected, the spotweld nugget diameter can be modified.



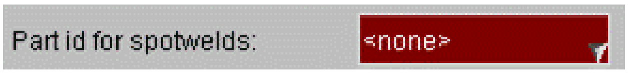
The MIG (beam) option allows you to create a beam to represent a portion of a MIG weld. Typically many of these connections would represent a MIG weld seam. The beam is meshed in (shares a node with the shell) at one end (the blue part in the figure on the right). The other end of the beam is projected onto the other panel and is mesh-independant (like the normal beam weld).

Also see converting MIG weld to beamless.



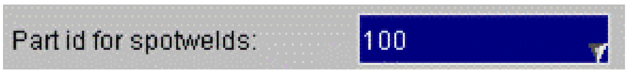
Choosing a part for the spotweld elements

PRIMER needs to know which part to put the spotweld elements into. If there is only one part in the model that is suitable (i.e. for beams if the part uses material *MAT_SPOTWELD and section type *SECTION_BEAM, or for solids if the part uses material *MAT_SPOTWELD) then PRIMER will automatically select it. Otherwise you will have to select it.



To select a part type in the part number, or you can use the standard popup functions (right click) to select or create the part. The part **must** use material type `*MAT_SPOTWELD` (material 100). If the spotwelds are defined as beam element, the part **must** use a section type `*SECTION_BEAM` type 9.

Once the part has been selected or created the part number will be displayed in the box:



Selecting which shells to use

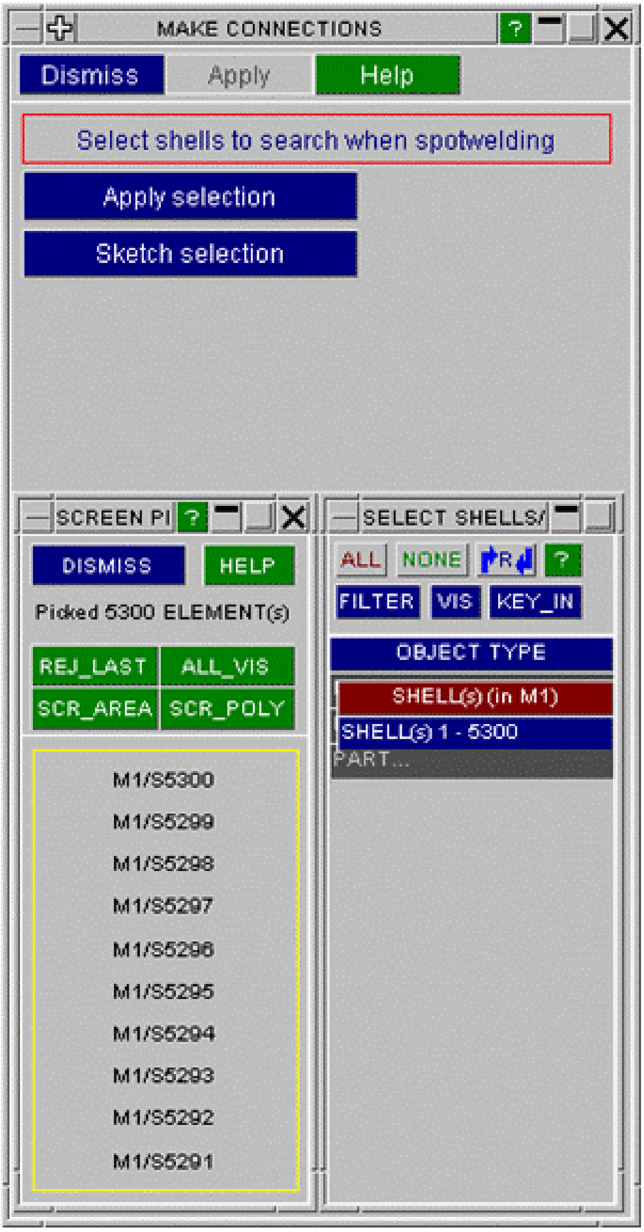
Before PRIMER can create any spotwelds it needs to know which shells it can use and which it cannot use. This may be as simple as just selecting all of the panels in the vehicle or you may just want to select 2 or 3 panels to spotweld.

all will select all deformable shells in the model

visible will select all visible deformable shells

If you are connecting to rigid parts you will need to select these using **select shells to connect**.

By default no shells are selected. Press the button to define them.



You define which shells to weld by using the standard object menus to select either parts or shell elements.

When PRIMER creates a spotweld from a point you give it, it looks to see what elements near the point are selected for welding and tries to create a spotweld between these elements. If you do not want a certain panel to be welded, do not include it in the set.

In the screen on the left 5300 shells have been selected. When a weld is created it will only be between these shells, even if there are elements from other panels near the weld point.

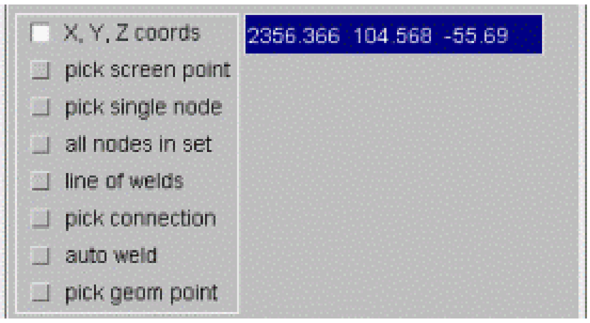
To finish selecting the elements to weld press the **APPLY SELECTION** button. You can sketch the elements/parts that you are selecting at any time by pressing **SKETCH SELECTION**.

Once the shells are selected the button updates:

Adding an optional title

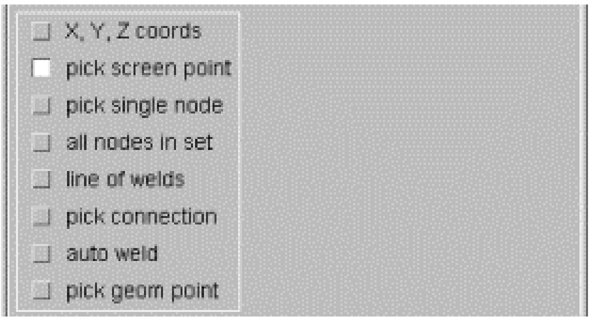
When creating any connection, a title can be added to the connection by typing the title in the optional title box.

Using coordinates



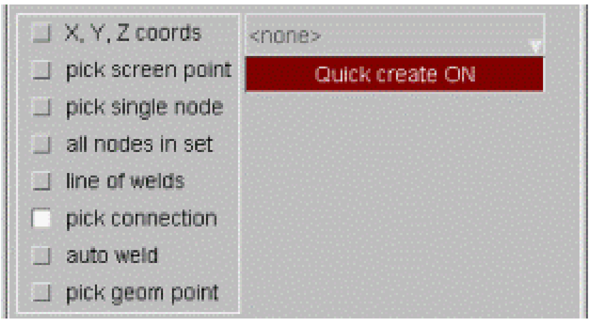
Type the X, Y, Z coordinates into the box and press the **APPLY** button. The spotweld will be created if it is possible. If the weld cannot be made an error message in the dialogue box will give the reason why. You can undo the spotweld if it is not what you want by pressing **UNDO CREATE**.

Using a screen point



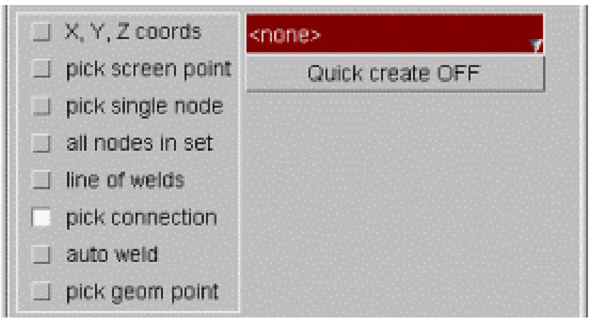
Using the cursor, select a point on the screen at which you wish the spotweld to be created. The spotweld will be automatically created at the point selected.

Using a connection

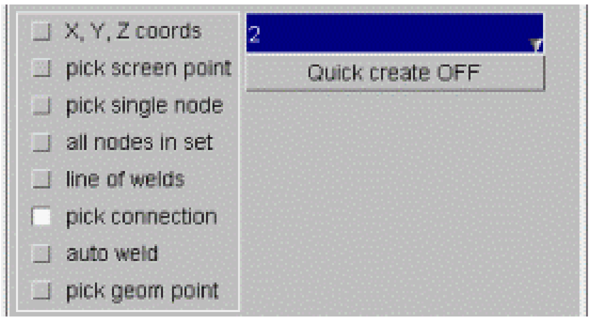


If quick create is turned on, you can just pick an existing connection from the screen. The spotweld will be created if it is possible. If the weld cannot be made an error message in the dialogue box will give the reason why. You can undo the spotweld if it is not what you want by pressing **UNDO CREATE**.

If a spotweld or bolt is already defined for the selected connection, you can substitute this existing element with the new spotweld by selecting the Delete old connection option

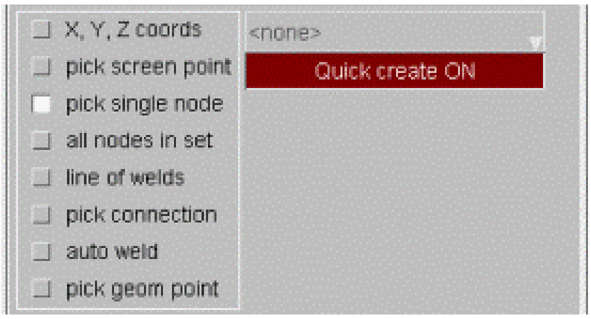


If quick create is turned off, you can type the connection number into the box or use the normal popup functions to select a connection.

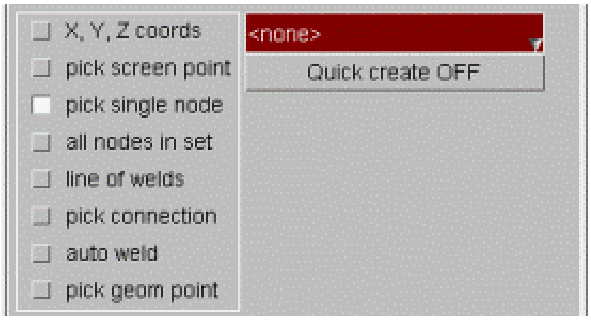


Once the connection has been selected the spotweld can be created by pressing the **APPLY** button. You can undo the spotweld if it is not what you want by pressing **UNDO CREATE**.

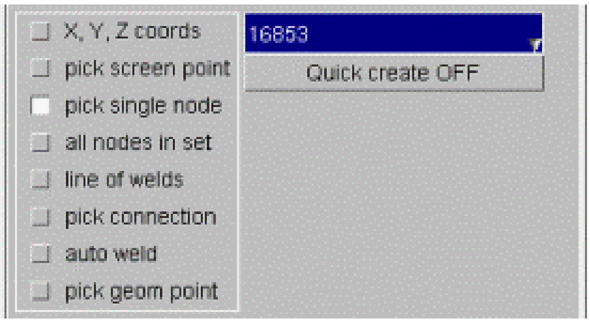
Using a node



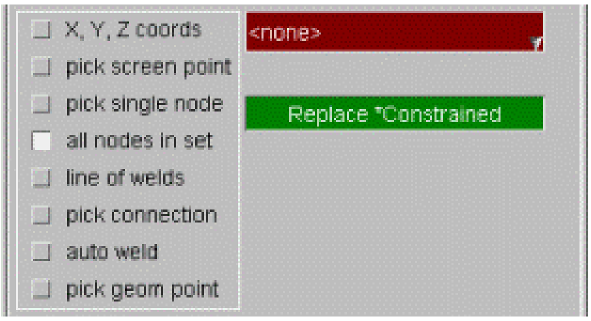
If quick create is turned on you can just pick a node from the screen. The spotweld will be created if it is possible. If the weld cannot be made an error message in the dialogue box will give the reason why. You can undo the spotweld if it is not what you want by pressing **UNDO CREATE**. If quick create is turned off you can type the node number into the box or use the normal popup functions to create or select a node.



Once the node has been selected or created the spotweld can be created by pressing the **APPLY** button. You can undo the spotweld if it is not what you want by pressing **UNDO CREATE**.



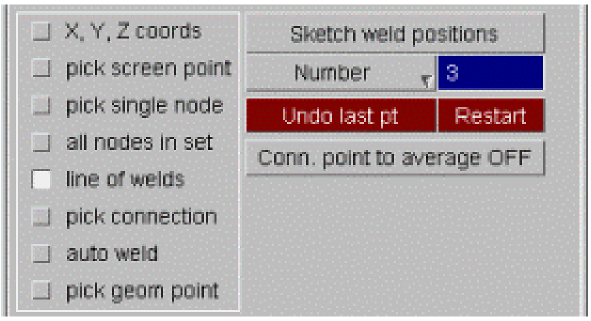
Using a node set



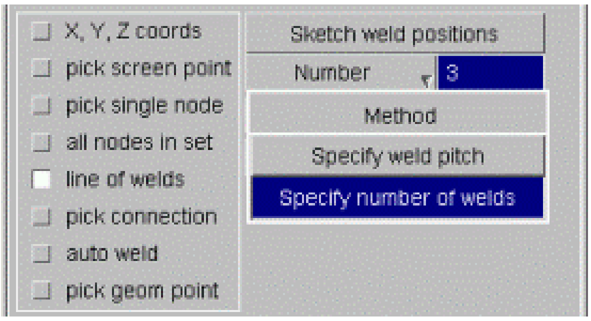
The "Replace *Constrained" option is designed to replace with a spotweld every *CONSTRAINED_SPOTWELD or *CONSTRAINED_GENERALIZED_WELD which has at least one node in the selected set. The spotweld will be created between panels selected for welding. NOTE - this may not be all the panels which the old weld joined if the original shell selection was incomplete. In this case the connection table will be invoked for these welds.

The old option "at every node in set" is still available. In this mode PRIMER will attempt to create a spotweld at every node in the set and dump to a node set any nodes where the weld could not be made.

Using a line of welds

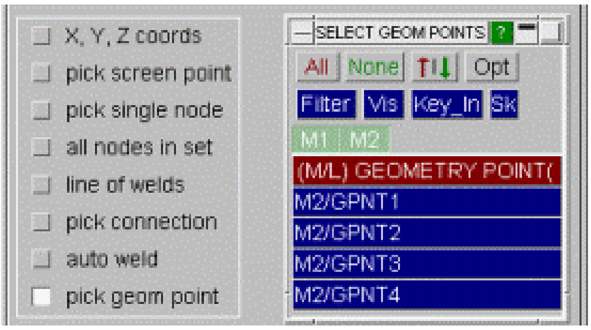


Using the cursor, select 2 or more points in order to create a line along which you wish to create spotwelds. For the MIG type spotwelds, this mode works in a different way. Here you select 2 nodes along a free edge/feature line. Primer will determine all the nodes along the free edge/feature line between the two nodes chosen. Clicking on **Apply** after this will create MIG spotwelds at all the nodes along the free edge/feature line between the 2 selected nodes. If you choose the same node twice for this operation, Primer will create MIG weld beams for every node around the free edge.



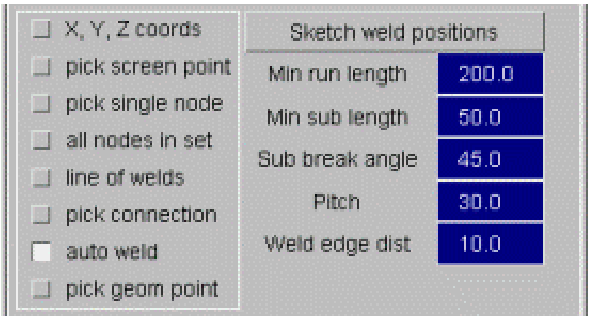
By either clicking on the tab or using the available popup function, specify whether the quantity of spotwelds you require is determined by **Number** or by **Pitch**. If using number, type the number of spotwelds required along the line in the box. If pitch is required, type the desired distance between spotwelds in the box. Once completed, press the **Apply** button to create the spotwelds.

Using geometry points

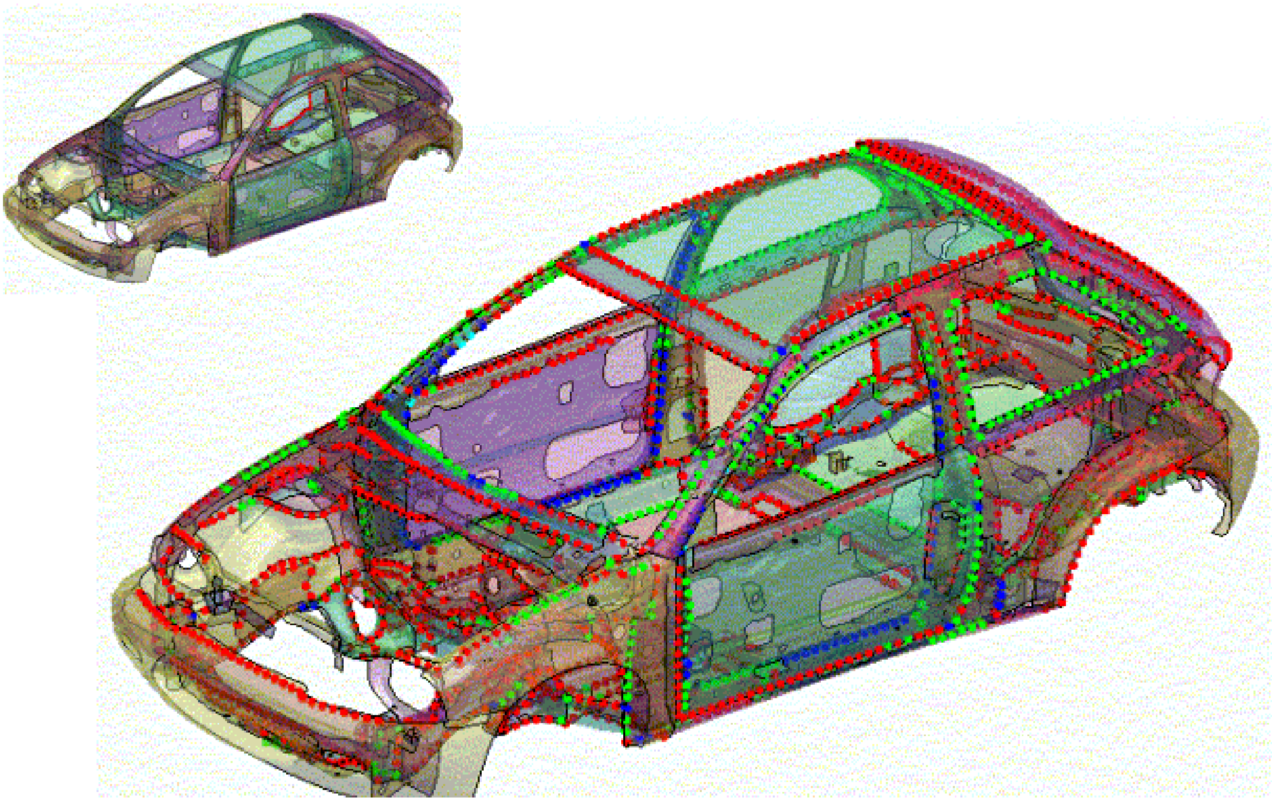


Geometry points can be selected from any model in Primer. The coordinates of these points are used as the coordinates of the connections to be created.

Auto Welding



PRIMER has the ability to automatically weld panels together with the only input being the shells to weld and a few user defined parameters, for example:



The selected shells are searched through, and any shells that are close together are flagged for the 2nd stage of the auto welding process. The second stage takes these shells and highlights any model free edges that belong to the shells - these are called **free edge runs**. Finally, each of these free edge runs are split into sub sections (at feature edges defined by a user defined angle **"sub break angle"**) and spotwelded at a user defined pitch and distance from the edge. The weld run is centred so there is an equal amount of space at the beginning and end of the weld.

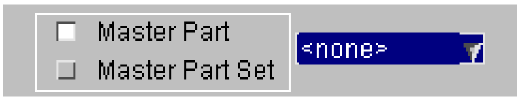
Any welds that are too close together are discarded - eliminating the chance of multiply welded parts.

You can sketch the possible weld points using the **Sketch weld positions** button; note that this shows all attempted weld points - weld runs that lie on top of one another will no doubt have many weld points unmade because they are too close.

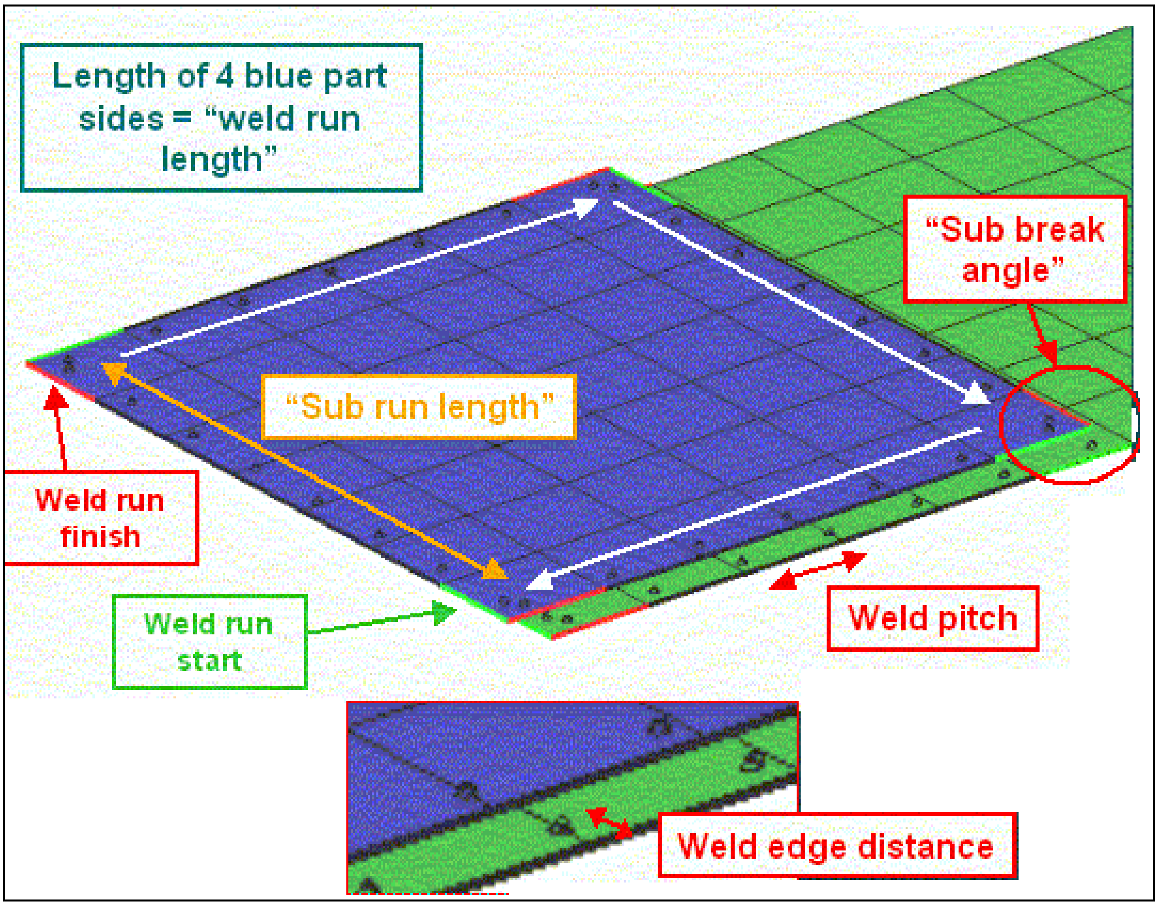
The user defined parameters are as follows:

- **Min run length**: any free edge runs that are less than this amount are discarded.
- **Min sub length**: any sub sections that are smaller than this amount are discarded.
- **Sub break angle**: the angle that determines how the free edge runs are split up.
- **Pitch**: pitch of the welds
- **Weld edge dist**: distance to weld in from the free edge run.

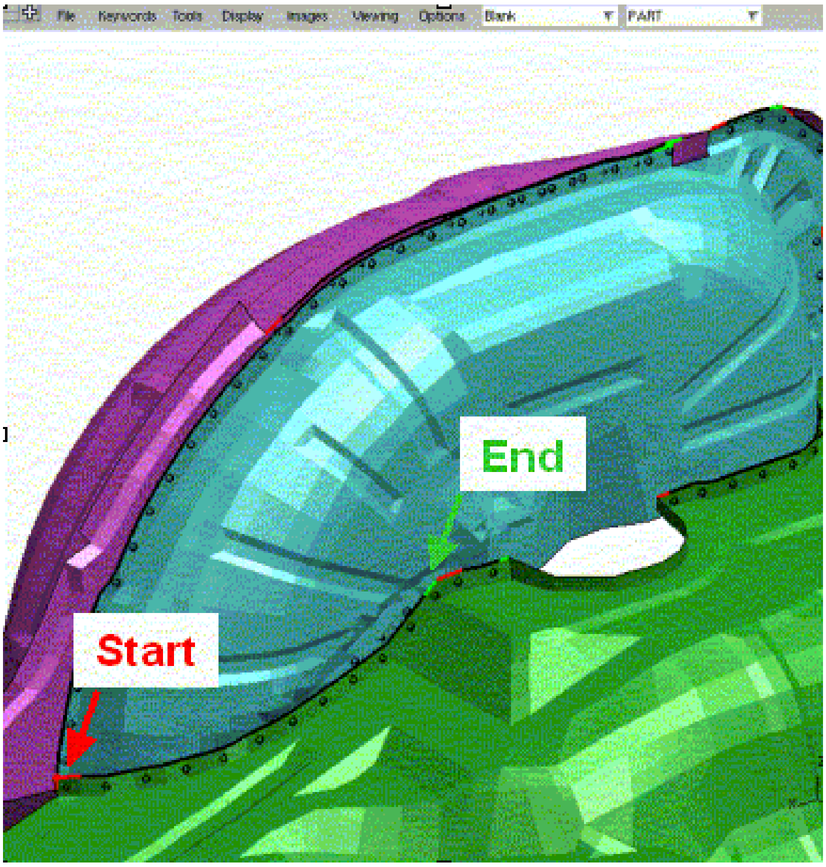
A master part or a master part set can be used to specify which panel(s) are used to determine the free edges. If specified, only free edges on the master part(s) are used to construct spotwelds. Without a master part/part set selected, all shells selected for connection are considered when determining free edges.



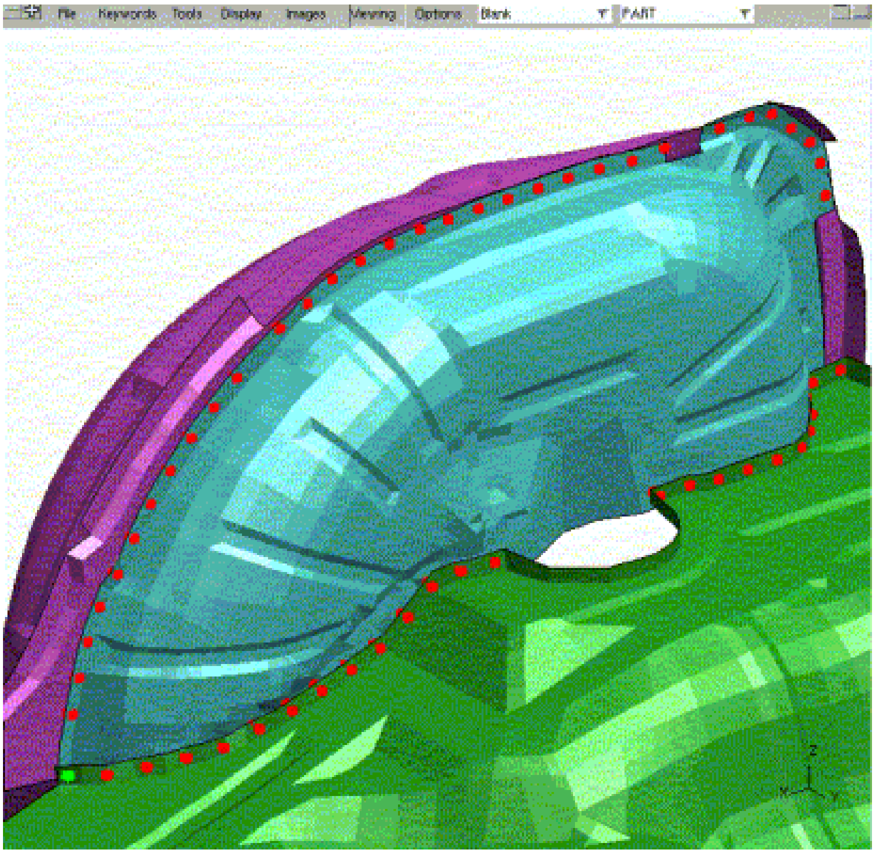
The following shows an example of sketching the positions, where the weld runs are shown as green lines (start position), red lines (end position) and black lines (min run lengths). Each possible position on each weld run is sketched as a square. Explanations of the parameters are also shown:



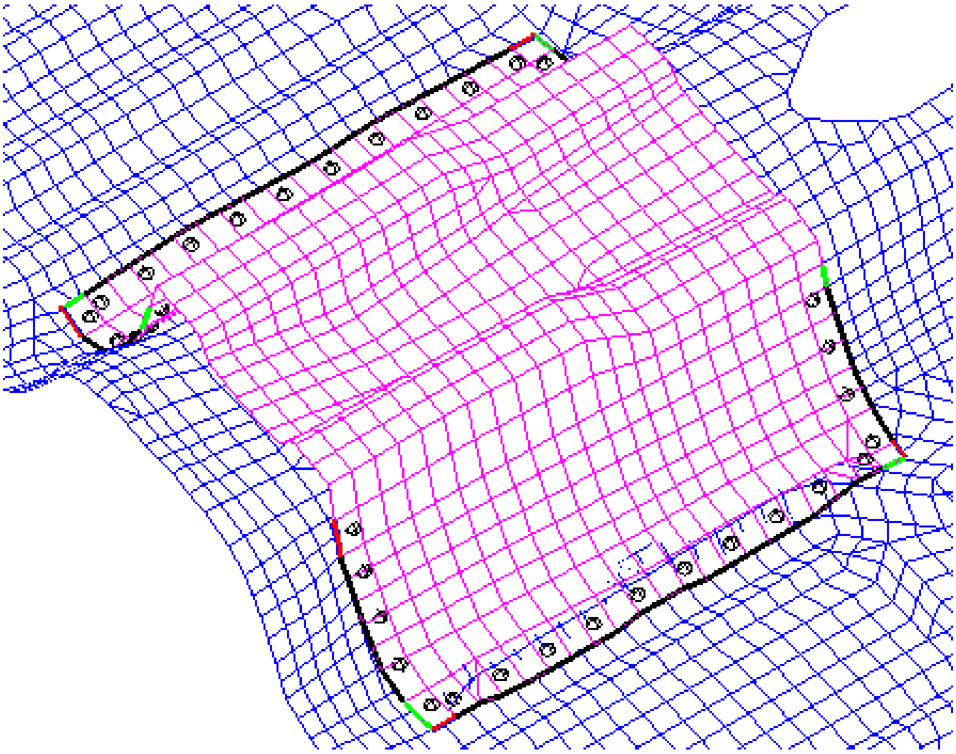
Sketch weld positions will sketch all the potential weld points PRIMER has calculated to attempt to weld. A green line marks the start of a run, and a red marks the end

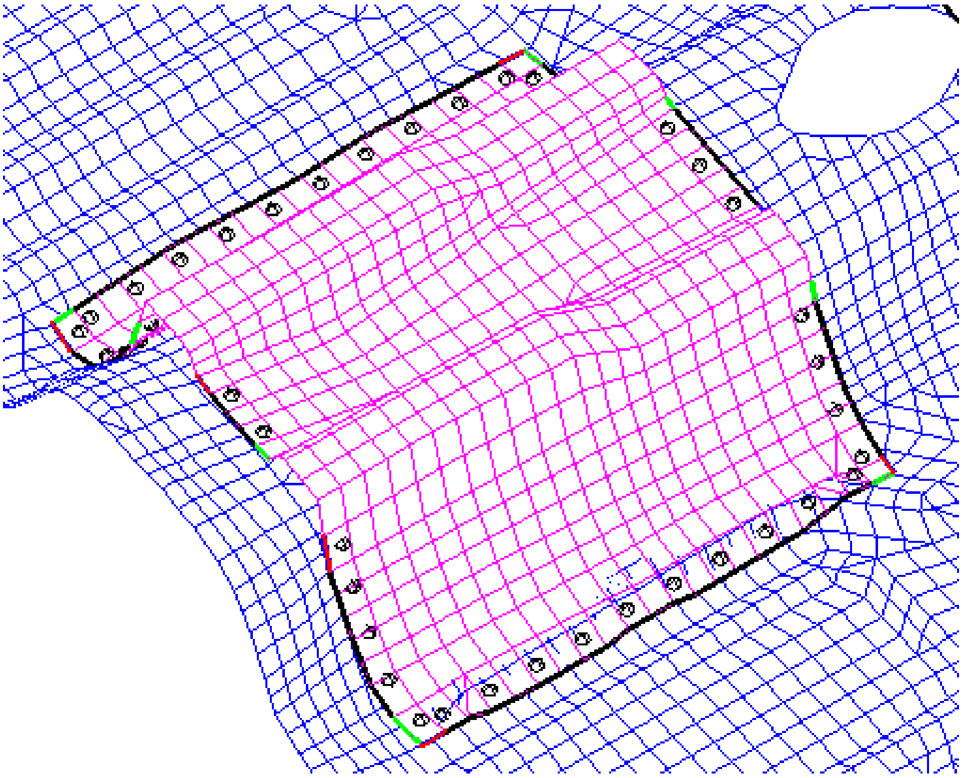


The following shows an example of the automatic welding, the red dots are the successful 2-panel welds, and the green dots are the 3-panel welds. Even though there were weld runs (and therefore weld points) that were next to each other, the auto weld routine checks for proximity and will not weld any points that are too close to each other.



The next image shows the result of changing the minimum sub length to create welds on the two short sections at either edge of the magenta panel. On the first image, the sub length isn't enough to allow the sections to be welded, on the second image, the sub length has been reduced so the auto spotwelder allows these welds:



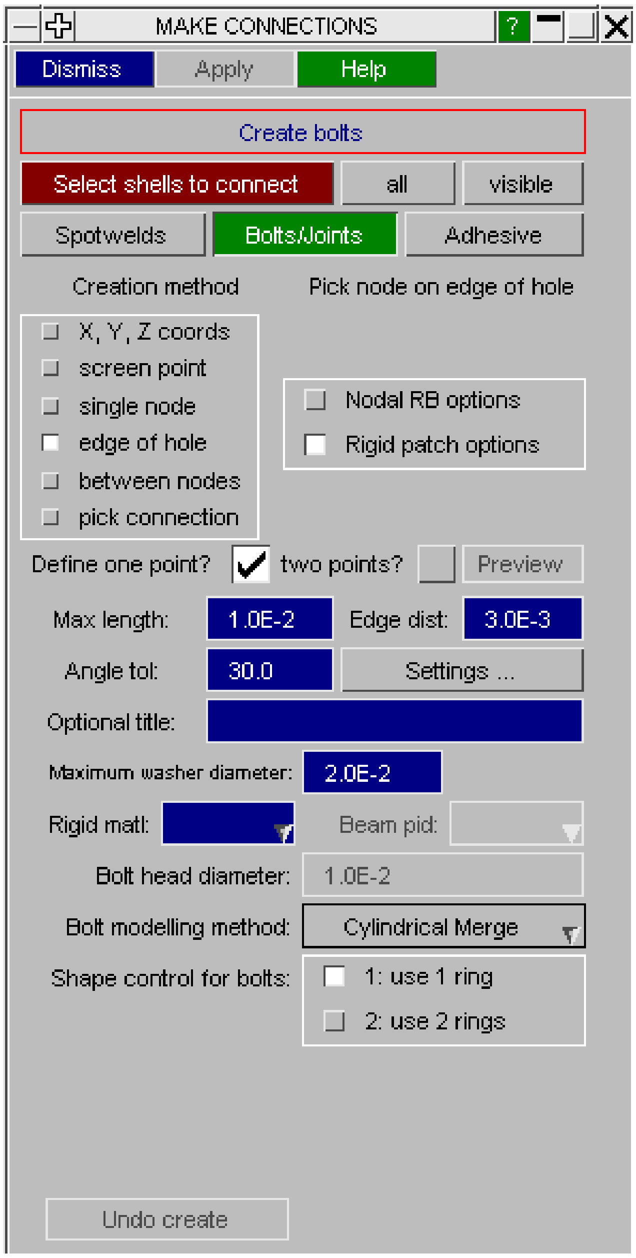


Note: There are 2 oa_pref options to control how tolerant the autoweld feature is when checking close welds. They are factors on the spotweld pitch and are used to calculate the distance to check for close welds on nearby seams.

The first is `autoweld_diff_seam_proximity` (0.5) which is a factor for checking welds on a different seam.
The second is `autoweld_same_seam_proximity` (0.8) which is a factor for checking welds on the same seam.

e.g. distance to check = Pitch * autoweld_same_seam_proximity

Creating bolts



The bolt/joint creation panel is shown on the left.

A bolt may be a single rigid entity (RB merge or NRB) joining multiple panels or when 2 panels are being joined, it may be 2 rigid entities (rigid patches or NRBs) connected by a beam, zero length discrete beam, revolute joint or ball joint.

The geometry of a **one point bolt** is described by a point, a diameter and a maximum length. The orientation of the bolt is calculated by Primer from the shells to be joined. This method is appropriate for short bolts.

The geometry of a **two point bolt** is described by two points and a diameter. In this case, "length" is max thickness of the connection at each end, the actual bolt length being unrestricted. The axis of the bolt is defined.

In the case of merge or rigid patch bolts, extra master parts (with no elements of their own) are created which allow the unlabelled rigid body merge to be identified.

These also provides parts which can be converted to PART_INERTIA should the mass properties of the bolt require adjustment.

Optional material id. The user may enter a rigid material id to be used for the rigid patches, otherwise PRIMER creates one with properties derived from the panels being attached.

If creating a bolt at a hole, the **maximum washer diameter** must be set to a sufficient value. The default of 20mm should be sufficient for most models.

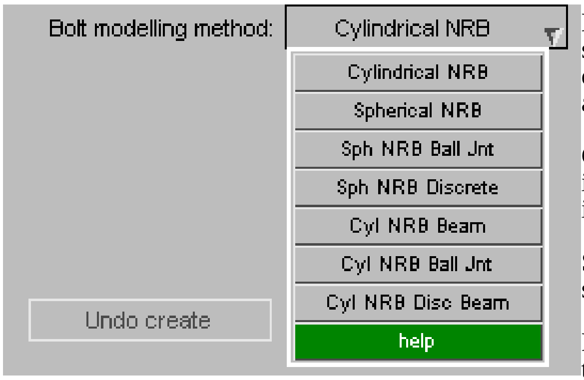
Bolts are now **drawn in preview** and will only be created when **APPLY** is pressed.

Creating single point bolts

A single point set by a variety of methods (see below) determines the position of the bolt head.

The **Max length** parameter sets the maximum possible length of the connection. Primer will search for shells along the bolt axis in both directions to this length from the connection point. So the connection point should be defined at the head/nut of the bolt not in the middle. The table function **update & remake (repos)** can be used to reposition the connection point.

For bolts which are created on panel mesh (i.e. excluding those at a hole) **Bolt head diameter** is specified. If a bolt is being created at a hole, the diameter is determined by the choice of shape control flag (see below). Bolts may be made at positions which combine meshes both with and without holes, in which case the shape control will be applied at the holes.

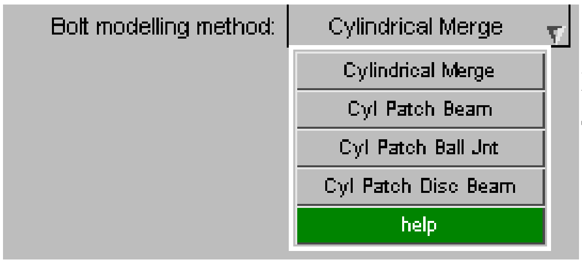


NRB methods use NODAL_RIGID_BODY to rigidify the panel shells. Spherical/cylindrical NRB form a single rigid body, the other methods will form 2 rigid bodies connected by the applicable FE.

Cylindrical methods determine a bolt axis and will only incorporate shells where the normal aligns (using Angle tol) with it.

Spherical method requires no alignment of shells and will simply sweep out a radius from the connection point.

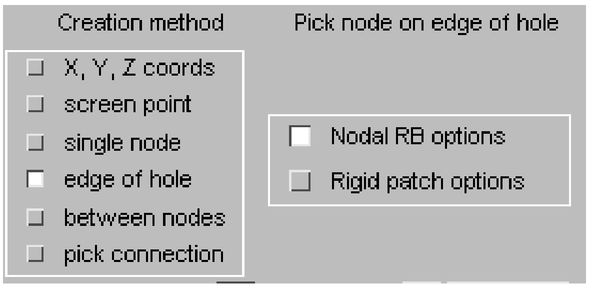
For bolt at hole, spherical methods are not applicable and therefore greyed.



Cylindrical merge forms rigid patches connecting layers which are slaved to single rigid master part. Other methods form 2 rigid patches connected by the applicable FE.

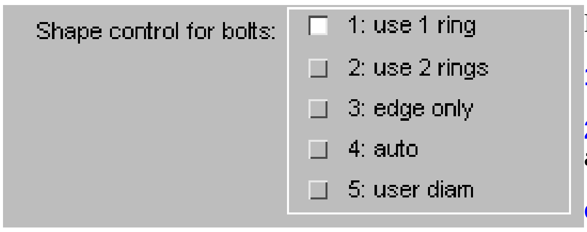
These methods will attach to rigid parts by making additional *Constrained_rigid_body definitions as necessary.

Single point bolt at hole



Picking a node on the edge of a hole in the mesh will set up the data for creating a bolt with its head at the centre of the hole. The prospective bolt will be sketched in preview.

The diameter of the bolt is determined by the shape control setting.



For NRB type bolts the options are

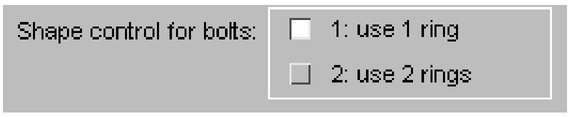
1 ring - rigidify nodes of all shells around the hole

2 rings - rigidify node of all shells around hole and all that attach to them

edge only - rigidify only the nodes around the hole

auto - calculate diameter from mesh density

user diam - use the input diameter

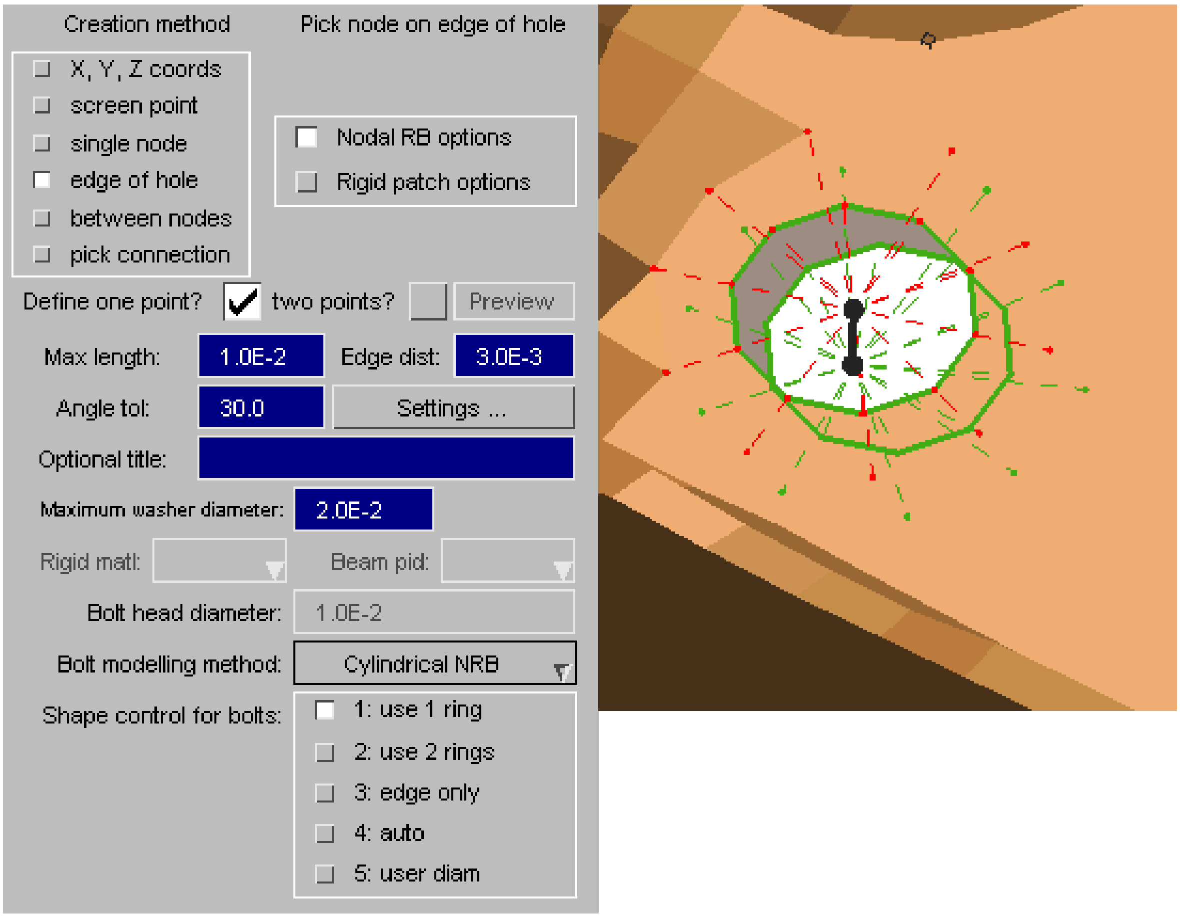


For patch type bolts the options are

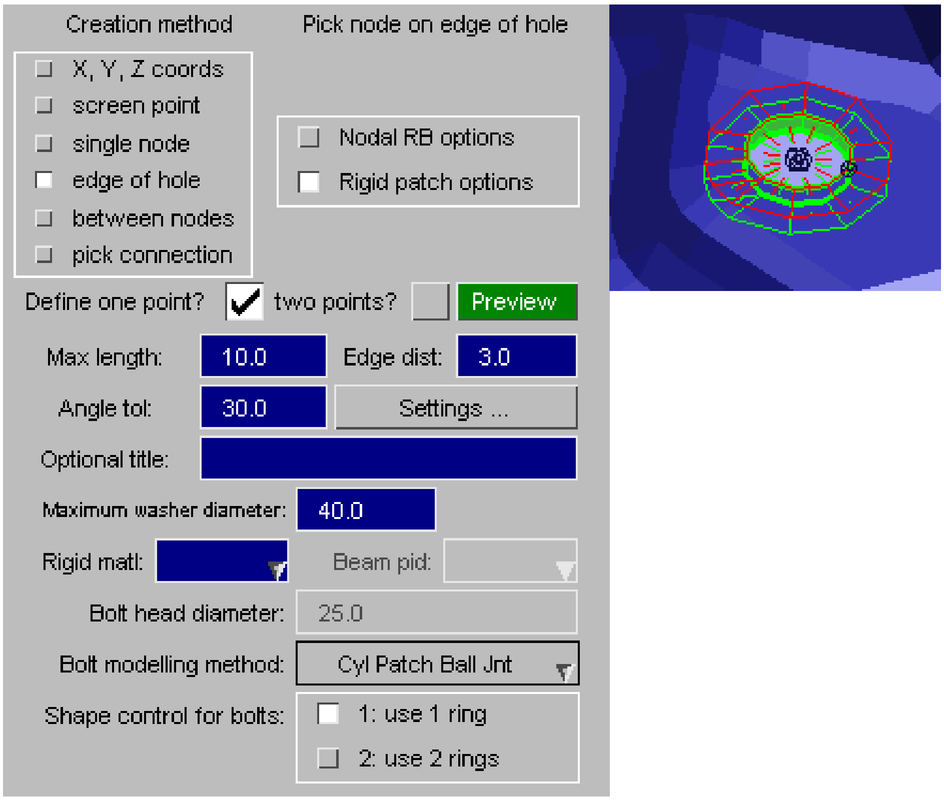
1 ring - rigidify all the shells around the hole

2 rings - rigidify those around the hole and those that attach to them

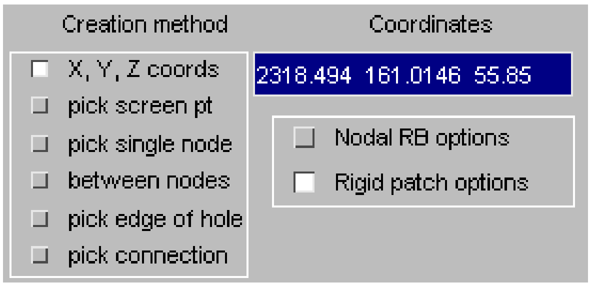
Preview of Nodal Rigid Body bolt with deformable beam formed by picking node on edge of hole. The free edge detected is drawn in green.



Preview of patch type "bolt" with ball joint connecting the rigid parts.



Single point bolt from coordinates



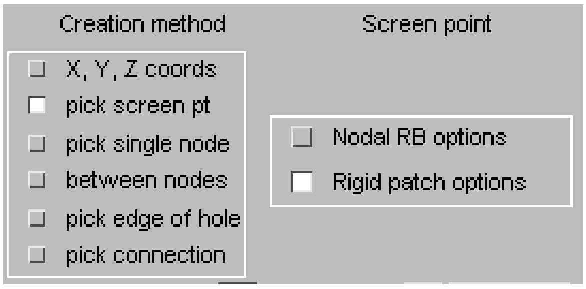
Type the X, Y, Z coordinates into the box and the bolt will be previewed if it can be made.

If the bolt cannot be made an error message in the dialogue box will give the reason why.

APPLY will create the bolt

UNDO CREATE will remove it

Single point bolt from screen point



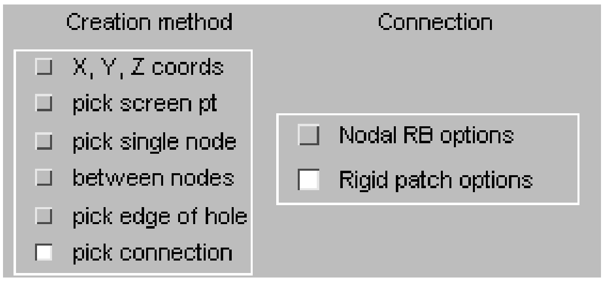
Using the cursor, select a point on the screen at which you wish the bolt to be created.

The bolt will be previewed if it can be made.

APPLY will create the bolt

UNDO CREATE will remove it

Single point bolt from existing (empty) connection

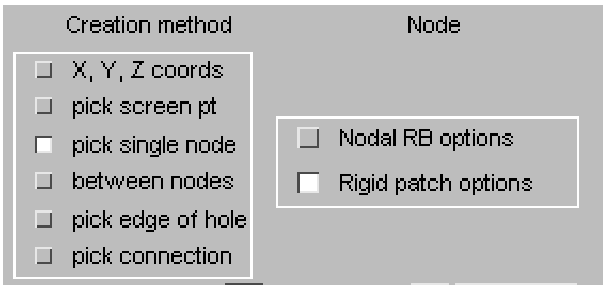


If you pick an existing empty connection, Primer will preview the bolt.

APPLY will create the bolt

UNDO CREATE will remove it

Single point bolt at a node

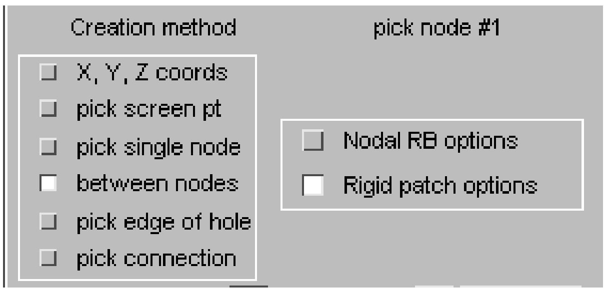


If you pick a node, Primer will preview the bolt that can be created.

APPLY will create the bolt

UNDO CREATE will remove it

Single point bolt between picked nodes

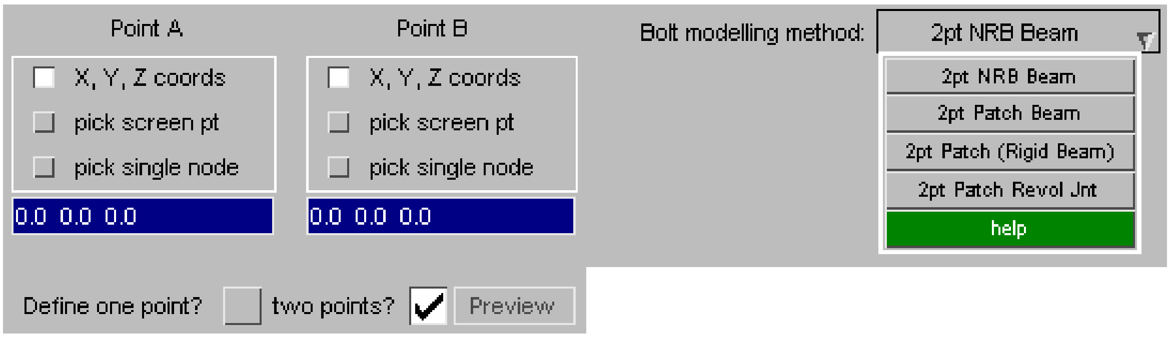


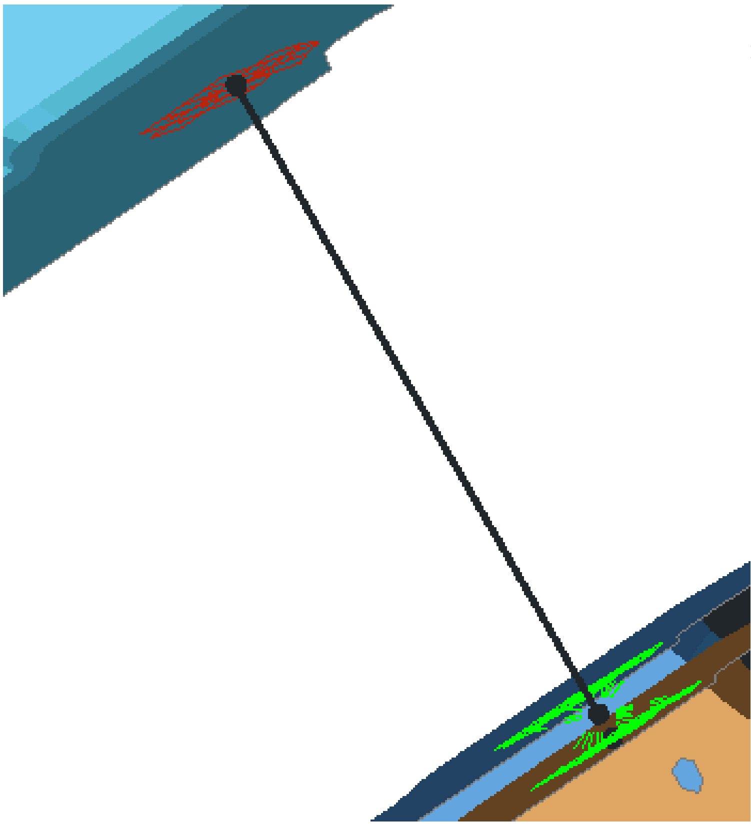
The "between nodes" function provides a way to create a bolt between N nodal points. Once you have selected a minimum of 2 nodes the connection can be previewed by pressing **PREVIEW**. Nodes are picked by a left click and may be deselected by a middle click on the mouse.

Creating two point bolts

For longer bolts the 2 point method is recommended. Each point may be defined by coordinate, screen point pick or node pick. If a node on the edge of a hole is picked, the centre of the hole will be taken as the connection point.

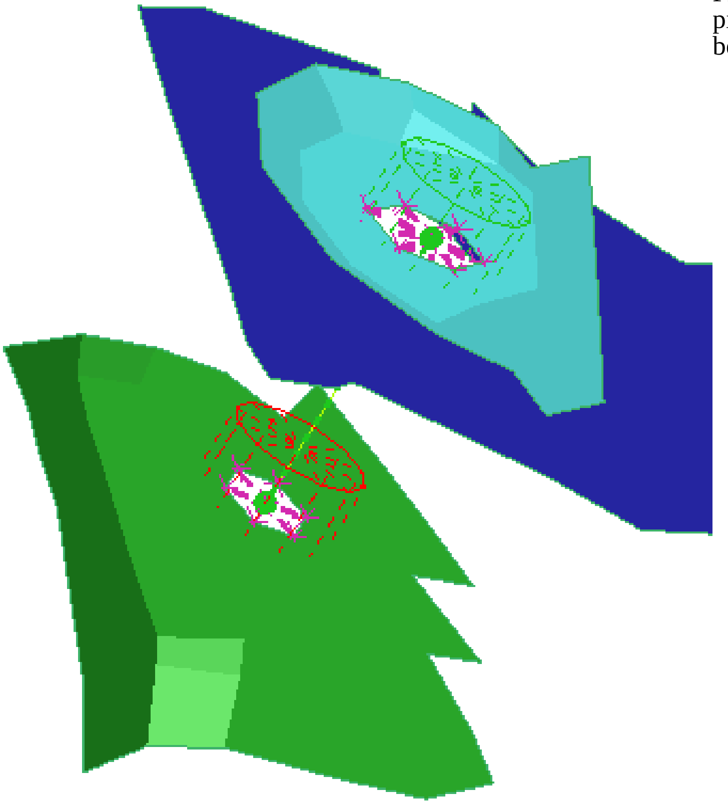
In this mode each of the two layers may contain multiple parts. Two separate rigid bodies are made (NRB or rigid parts) and may be joined by deformable beam, optional rigid beam or revolute joint.





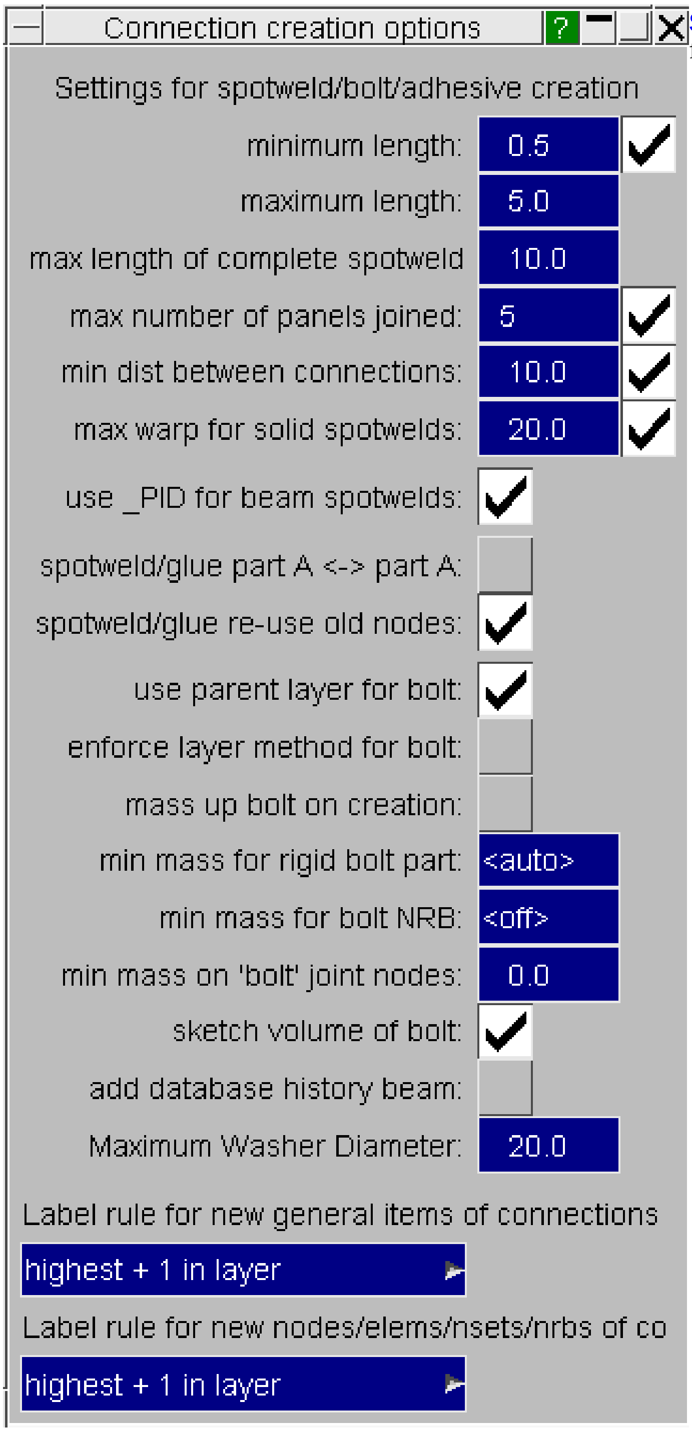
Preview of two point patch type bolt with beam made by pick screen point.

Note - multiple parts may included at each end



Preview of two point NRB type bolt made by picking a node on edge of hole with beam between rigid parts.

Bolt options



Settings ... on the create panel gives access to options for bolt creation

min mass on 'bolt' joint nodes - will add mass to the nodes of the joint to meet the target value (if necessary). Increasing mass of a joint increases its stiffness and stability.

minimum 'bolt' joint length - will increase the length of the revolute joint (if necessary) to meet the target value

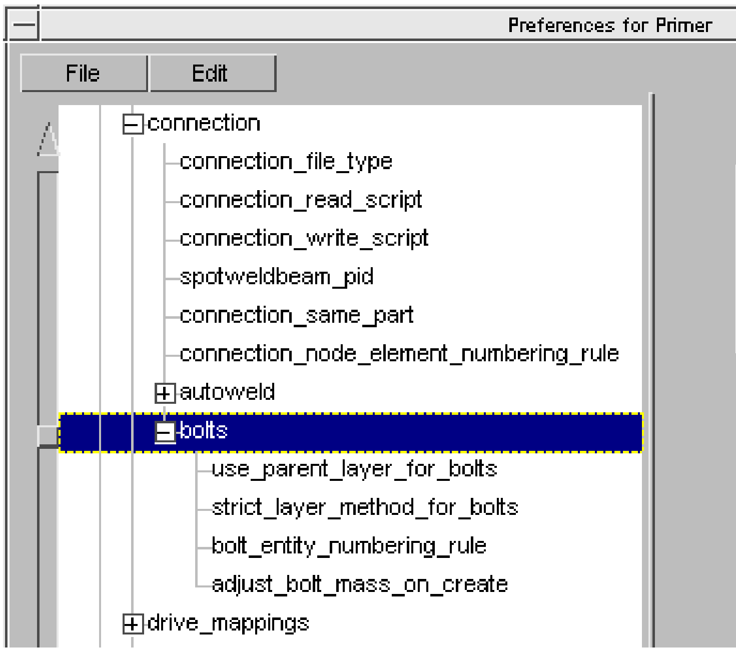
sketch volume of bolt - Primer will sketch the cylinder or cylinders (for 2 pt bolts) which show the nominal geometry of the bolt. This is useful for adjusting the diameter.

add database history beam - on creation of bolt, Primer will automatically make a Database History Beam id using the title of the bolt.

use parent layer for bolt FE - creation normally will put all FE into the current layer. However, for bolts if this option is set, bolts will try to use the parent layer. If parent panels are all in the same include, all FE will go into that include. If they are in different includes, shells/parts/ materials will be put with the overlaid parent part and any other items will go into the current include. On the connection table the layer of the connection itself and layer(s) of the FE data can be displayed using **view ...** conx include/FE include.

enforce layer method for bolt FE - if numbering rule is set to a layer type method and this option is set, it will block creation of the bolt if include file number ranges have not been set. Without this setting, Primer will create the bolt using highest+1 in model when number range is missing.

entity numbering rule - by default the same rule applies to welds/adhesive and bolts. The default is *model highest+1* but oa_pref **connection_node_element_numbering_rule** can modify this. For bolts only, oa_pref **bolt_entity_numbering_rule** will always over-rule any other setting.

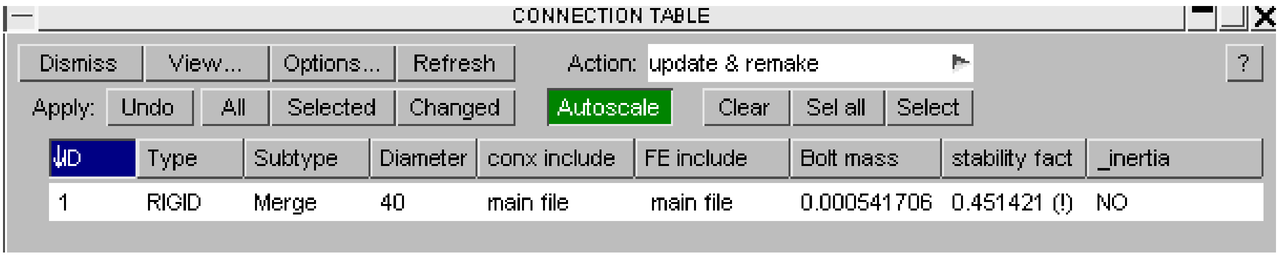


These may be configured using oa_pref settings

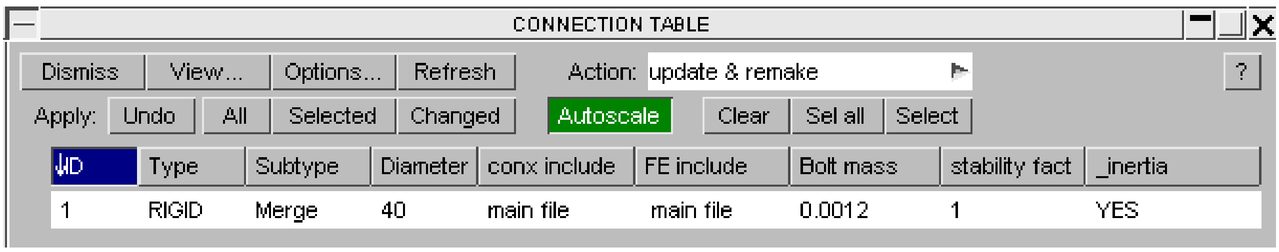
If **oa_pref bolt_entity_numbering_rule** is set. It will overrule the label rule set in the options panel which is therefore greyed.

mass up bolt on creation - small merge type bolts can lead to stability problems in mass scaled models. The problem arises because added mass on the deformable elements which attach to the rigid bolt gets lost. The mass of the bolt itself needs to be sufficient to compensate for this loss. By default Primer will automatically calculate the mass required, however user may set a target mass which will be used in preference (check > options > rigid). Nodal rigid bodies are implicitly stable in LS-Dyna so there is no stability criterion for these type of bolts, however user may set a target mass for these too.

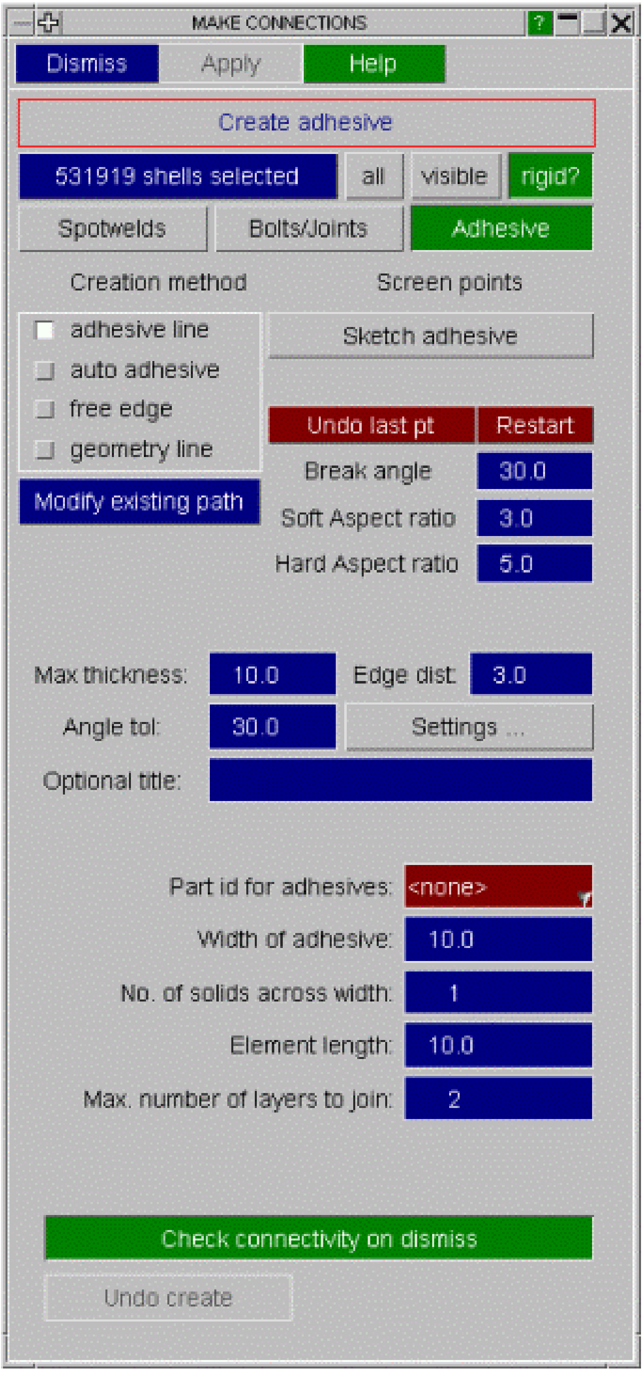
If this option is active and the requested mass exceeds the actual mass, Primer will convert the master part (merge type bolt) or nodal rigid body to an _INERTIA definition with mass properties appropriately scaled up. If it is not the mass of the bolt created may be too small for stability. The table will give the mass and the factor which should be >= 1.0 for stability.



The mass can be corrected by re-making the bolt with **mass up bolt on creation** active or by performing a model check and auto-fixing the master part/nodal rigid body. The table will then show that _INERTIA is being used.



Creating adhesive



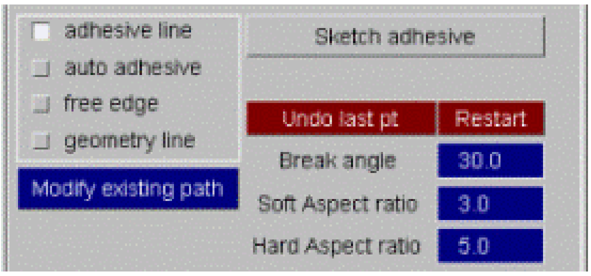
The adhesive creation panel is shown on the left. Adhesive is defined as a run of solid elements created between the panels you wish to connect.

Adhesives are created by defining a line along the panels you wish to join. The line can either be defined manually by clicking on the screen or through automatic methods, similar to automatic spotwelding.

Before any adhesive can be created, two things need to be defined. The part ID that new adhesive solid entities are put in needs to be specified. Also, the shells that will be connected needs to be specified (in the same way as spotwelds and bolts by using the **Select shells to connect** button).

Various inputs are defined to determine the final adhesive run. These include the width of adhesive, number of solid elements across the width, element length and various inputs to aid the adhesive definition when going round corners. These are explained in more detail below.

Generic adhesive buttons

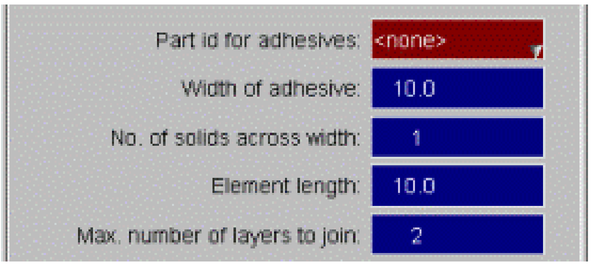


The available methods of creation are **adhesive line, auto adhesive, free edge** and **geometry line**. These can be selected on the radio buttons on the left. **Sketch adhesive** is used for sketching the proposed adhesive before creating it. This is particularly useful when creating long runs of adhesive. **Modify existing path** is used to modify the path points on an existing adhesive run.

Break angle controls how the defined path is split into sections. PRIMER will also try to meet the **soft aspect ratio** for the solid elements created, however this will not prevent creation. If a solid is found to fail the **hard aspect ratio** for the solids, then that solid will not be created. Note the aspect ratio check does not take into account the through thickness of the solid element, i.e. it is just in-plane aspect ratio.

When creating adhesive PRIMER will create all the solid elements it can when progressing along the adhesive run path. Some solid elements may not be possible (for example due to holes in the mesh). PRIMER will skip over these sections and create what it can. Because of this it is useful to use the **sketch adhesive** button as you are creating the adhesive to see what will be made and what will not.

When **adhesive line** is selected, the **undo last point** and **restart** buttons are available. These can be used as you are defining the adhesive path to undo points you have created and to start again.



The **part id for adhesives** is the part that the created adhesive solid elements will end up in. This must be defined before adhesive can be created.

The **width of adhesive** is the width across the run of adhesive.

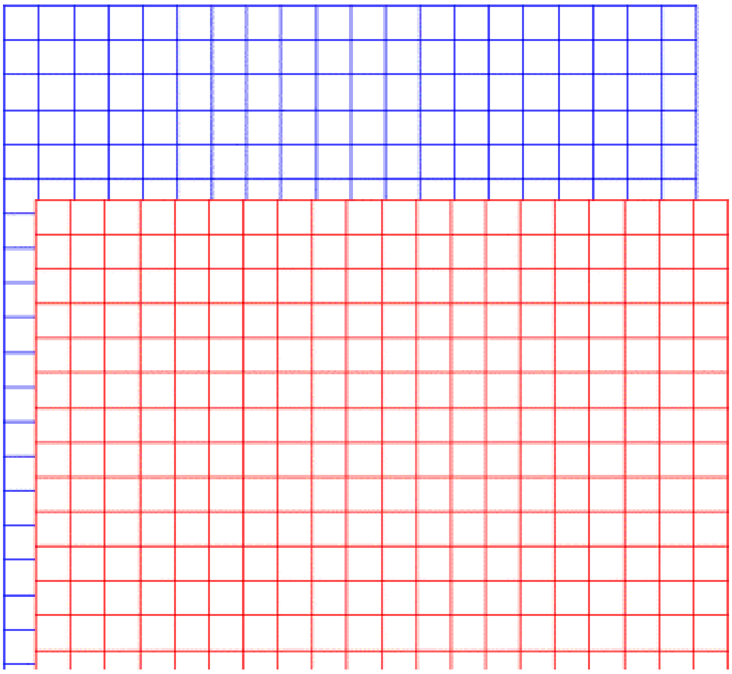
The **number of solids across the width** of the adhesive can also be specified on this panel.

The **element length** is the desired size of the solid elements along the length of the adhesive path

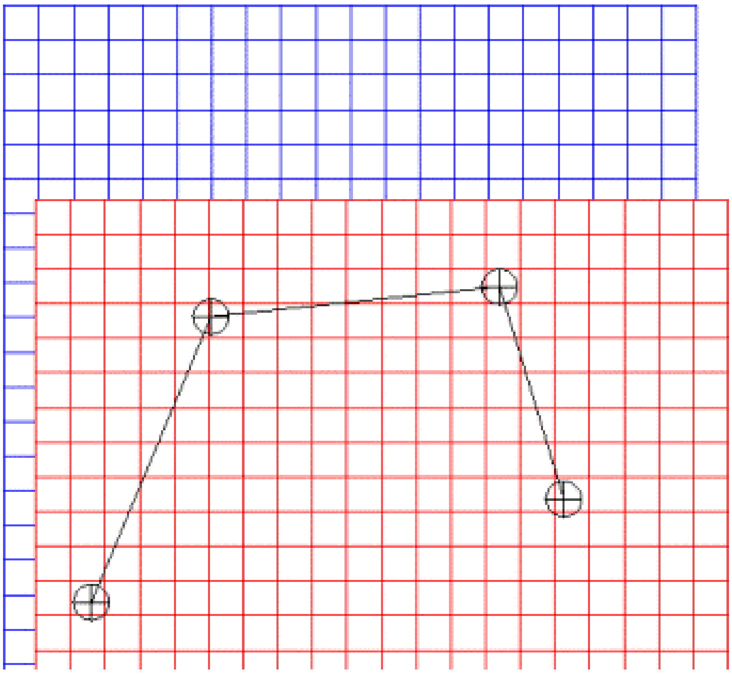
You can also increase the **maximum number of layers to join** from the default of 2.

Adhesive line creation method

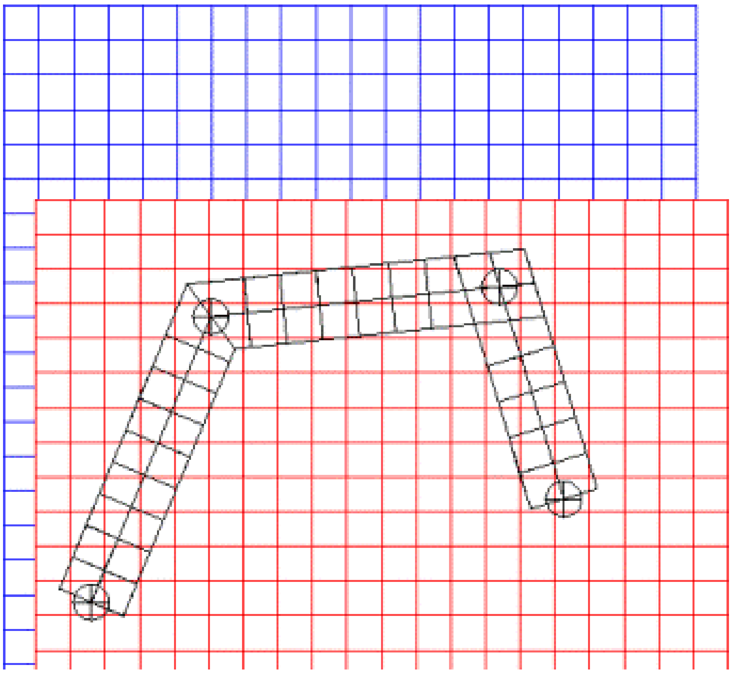
The following example shows how to create a simple run of adhesive using the adhesive line creation method.



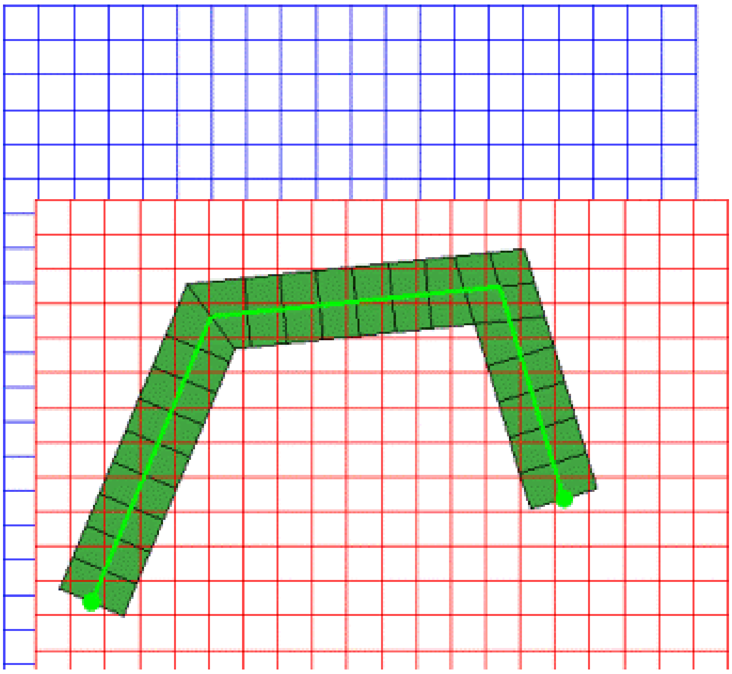
Say you want to join together two panels with an adhesive run using the **adhesive line** method. Remember, before you can create adhesive you must choose the part ID you wish the adhesive solids to end up in, and also the shells you wish to connect. You should also set your **adhesive width**, **number across width** and **element length** values.



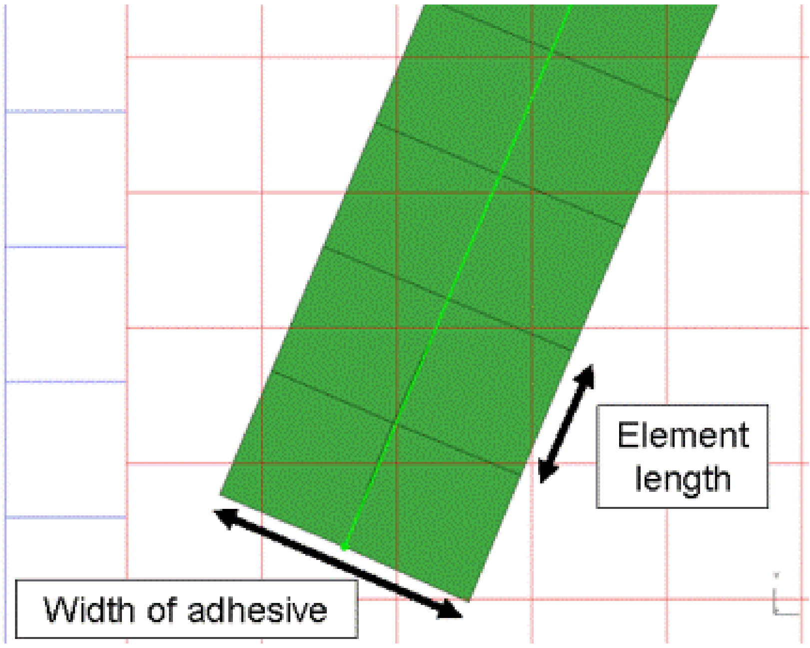
With **adhesive line** selected, click on the panels you have chosen to join to define points in the run. PRIMER will sketch the points and the line as you go along.



Clicking on the **sketch adhesive** button as you go along will allow you to preview the adhesive before actually creating it.



After you are happy with your defined path, clicking Apply will create the solid elements and create the connection entity. The connection entity is drawn as two blobs connected by a path line. The colouring of this connection entity is dependent on connection status and follows the same scheme as spotwelds.

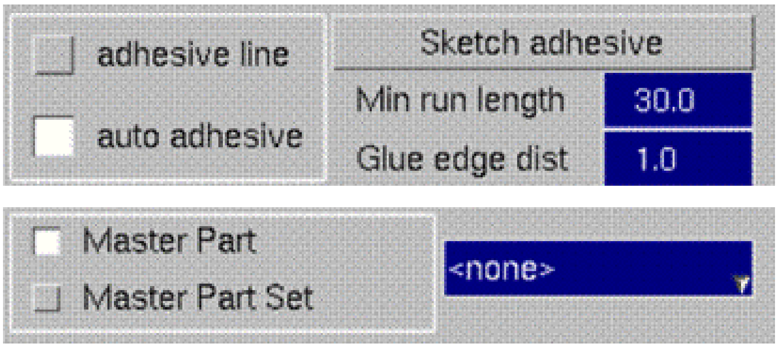


In the example shown, the **width of adhesive** has been defined as 20mm. The **number of solids across the width** has been defined as 2. The **element length** along the length of the adhesive has been defined as 10mm.

Auto adhesive creation method

Auto adhesive allows the user to automatically create adhesive runs between selected panels. The method works in a similar way to the automatic spotwelding feature described above.

The following additional buttons/inputs are available for auto adhesive creation:

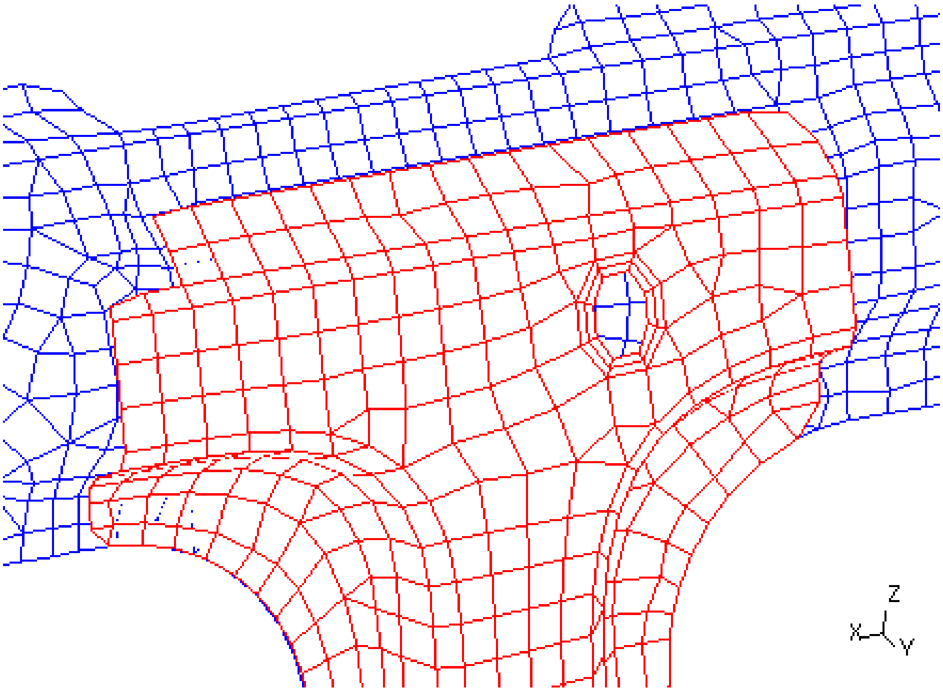


Min run length: any free edge runs that are less than this amount are discarded.

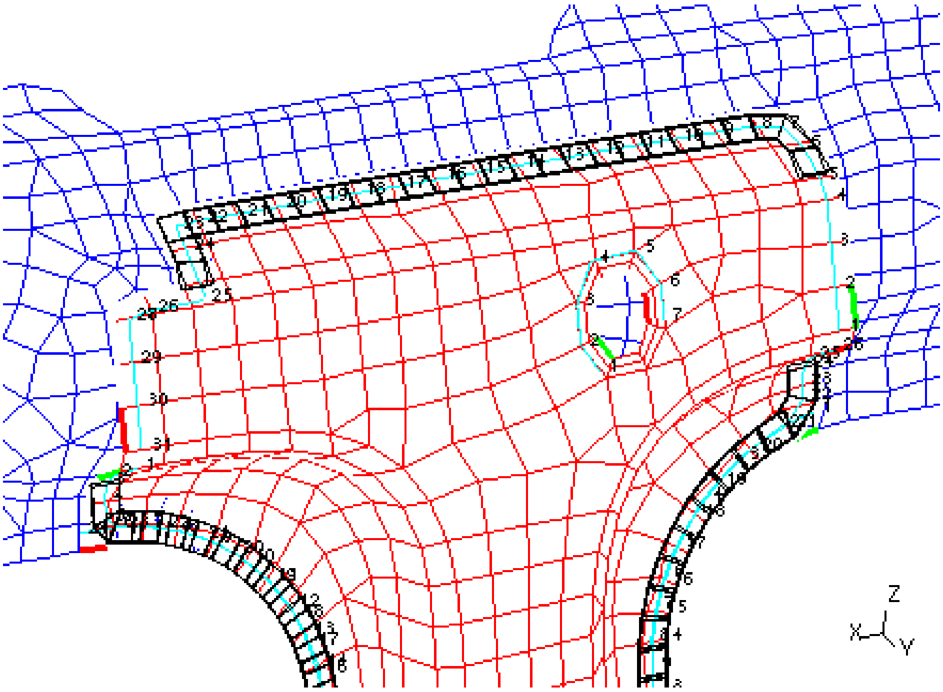
Glue edge dist: distance between the edge of the panel and the edge of the solid elements.

Master Part: a master part or a master part set can be used to specify which panel(s) are used to base the auto adhesive on. If specified, only free edges on the master part(s) are used to construct adhesive paths.

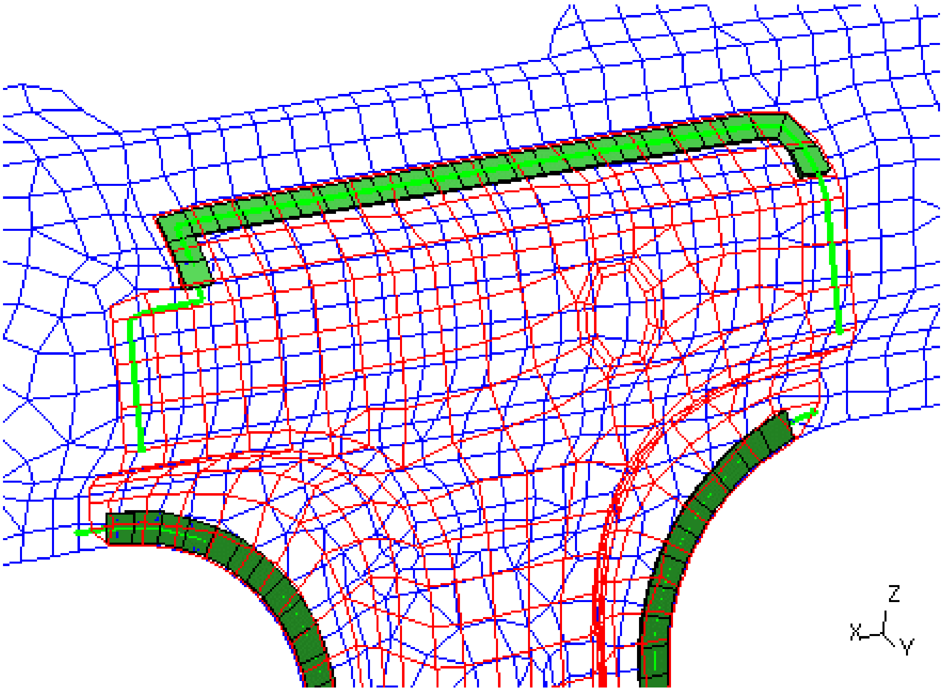
Without a master part selected, PRIMER will attempt to create adhesive from all free edges on the shells selected for connection.



To the left is an example of auto adhesive creation. In this case, the **adhesive width** is set to 10mm, the **number of solids across the width** is set to 1 and the **element length** is set to 10mm. The **glue edge distance** is set to 1mm, and the front part (red part) is set as the **master part**, meaning it is the part used to determine free edges and hence adhesive runs.



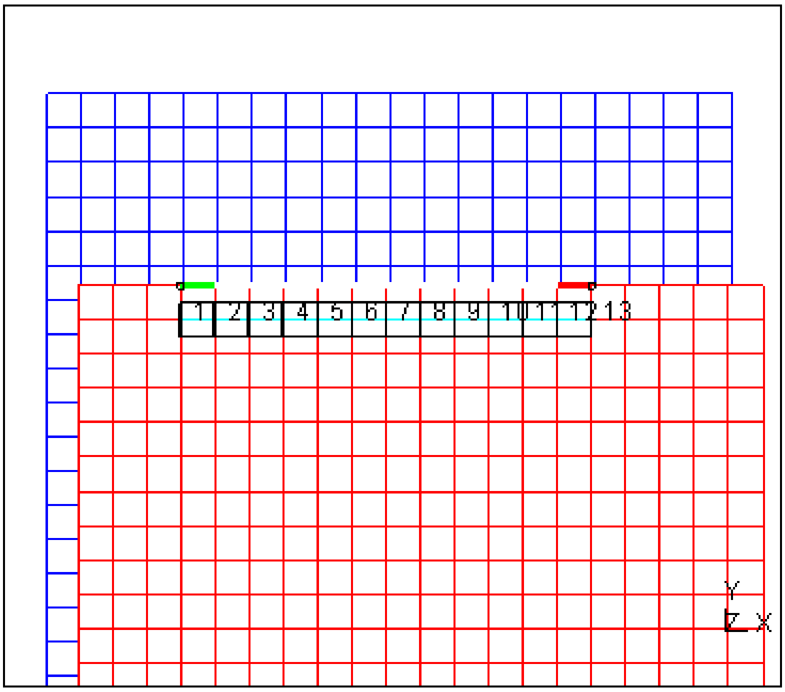
After setting the desired parameters, the **sketch adhesive** button can be used to preview the adhesive that PRIMER will create. The user can now change any of their inputs now and re-sketch the adhesive until they are happy with what will be created.



Clicking **Apply** will now create the adhesive. Note that **modify existing path** can be used to modify any runs created.

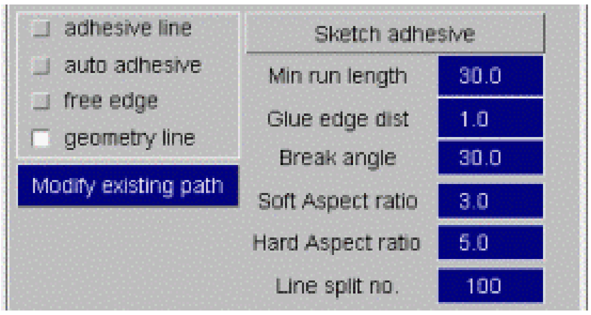
Free edge adhesive creation method

This is similar to the auto-create method in that it is based on free edges, but here the free edges are defined by the user rather than automatically determined by Primer.



The free edge length is defined by clicking on two nodes along the free edge. Primer will determine all the nodes along the free edge between the two selected nodes. Adhesive created using this method will follow the free edge. The distance between the edge of the adhesive solids and the free edge can be specified using **Glue edge dist**.

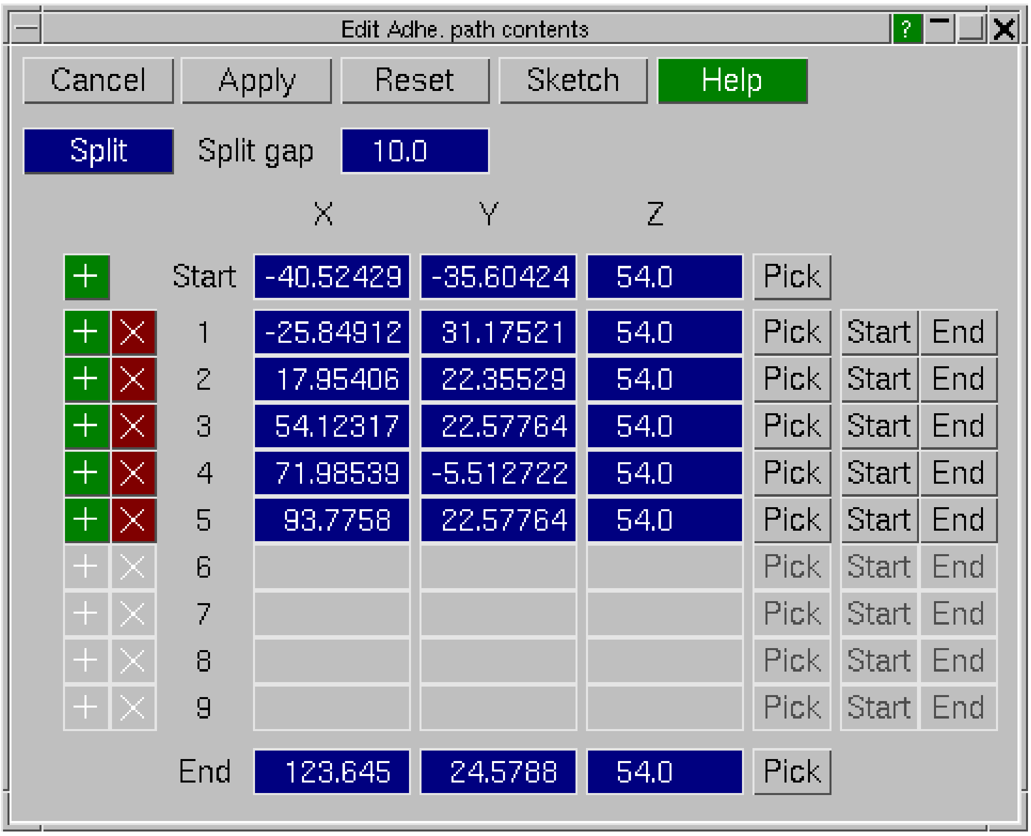
Geometry line creation method



The **geometry line** creation method can be used to create adhesive runs from geometry lines that exist in any model in PRIMER. Use **Line split no.** to specify how many increments the line is split into when creating the information for the adhesive path.

Modifying the adhesive path

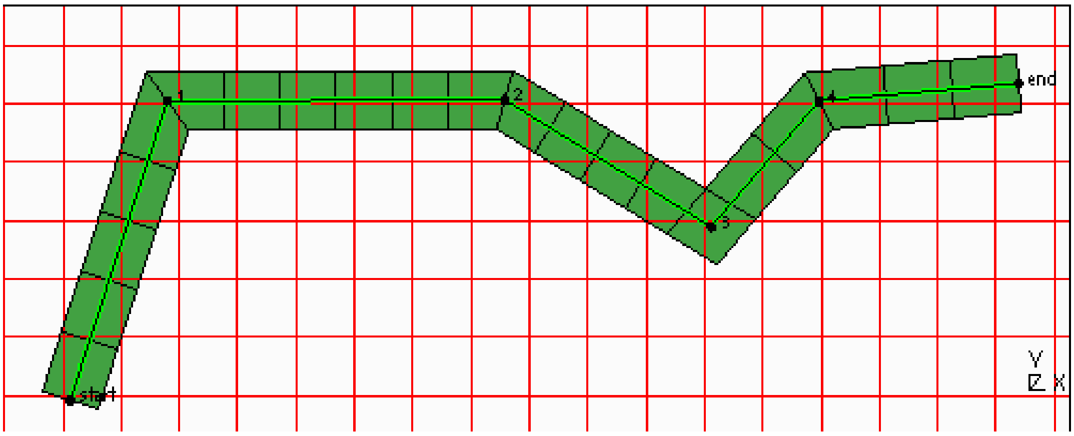
The path of an adhesive run can be modified by clicking on the **Modify existing path** button in the create panel. The path can also be modified in the same way through the connections table (see section 6.10.2). After clicking the button, you select the adhesive you wish to modify, and the following panel will appear.



The path modification panel displays the adhesive path information. The coordinates of the start and end points are displayed, as well as the in-between path points. Through this panel it is possible to carry out the following functions:

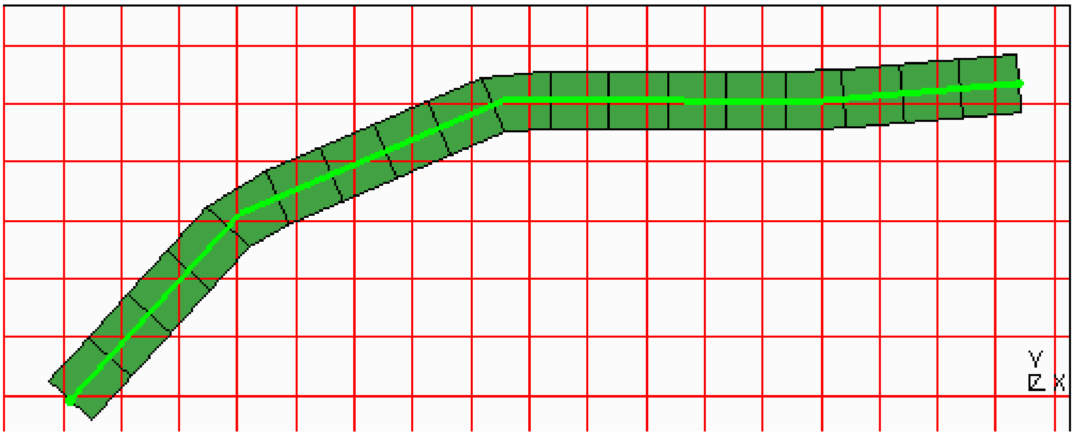
- Add or remove path points to the adhesive run.
- Modify the position of point on the adhesive path, either by typing in new coordinates or by clicking on the screen.
- Setting any of the existing path points to a new start or end point.
- Splitting the path at any point along the adhesive run with a defined gap.

To demonstrate some of these features, take the following example:

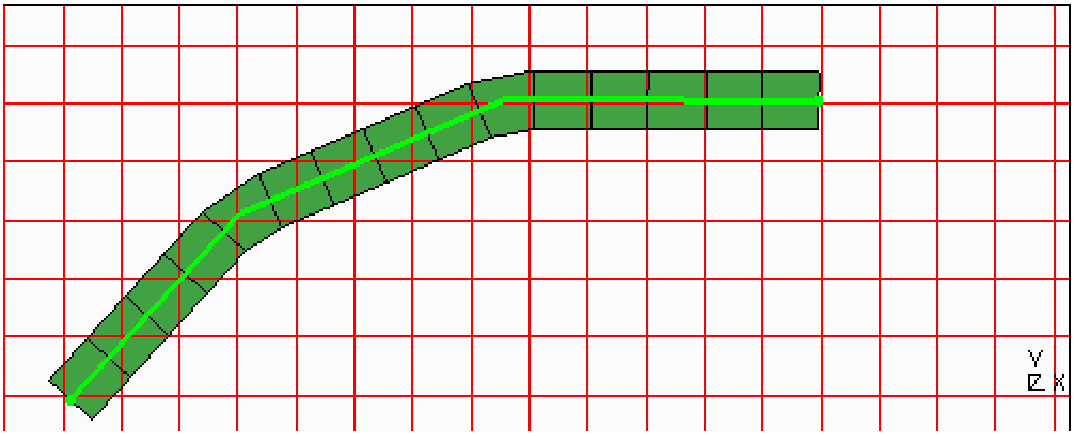


The above example shows a typical adhesive run. Clicking **Sketch** in the modify path panel will sketch the path points with numbers next to the points corresponding to the numbers shown on the panel. Note, you can return to the original path data by clicking on **Reset** (must be done before clicking on **Apply**).

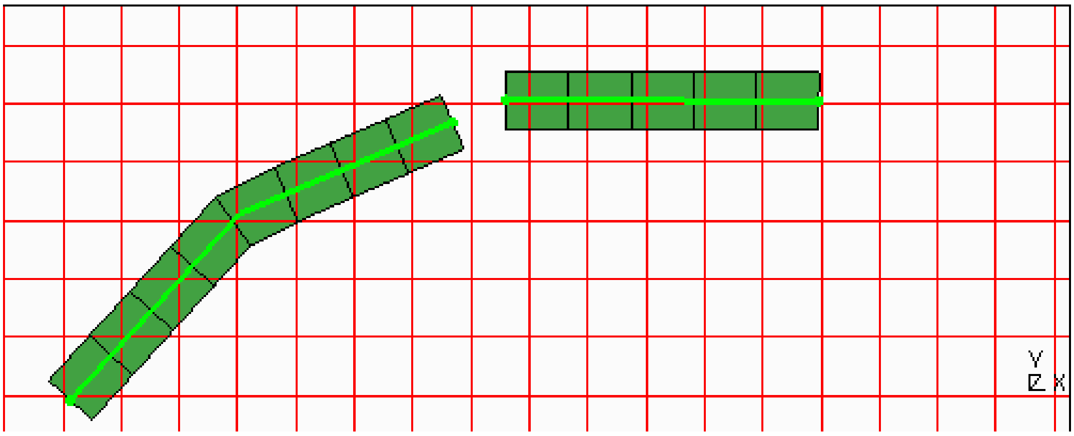
By clicking the red **X** button next to any of the points, that point will be removed. Clicking **Apply** will remake the adhesive run without that point. The image above shows the 3rd point removed. Similarly a point can be added by clicking on the green **+** button next to any of the points. This will create a new point after the point you clicked on. You can then select the position of the point by either typing in the coordinates or by clicking on **Pick** next to the new point, and then choosing a point on the mesh.



You can modify an existing point either typing in the coordinates or by clicking on the **Pick** button next to the point, and then choosing a point on the mesh.



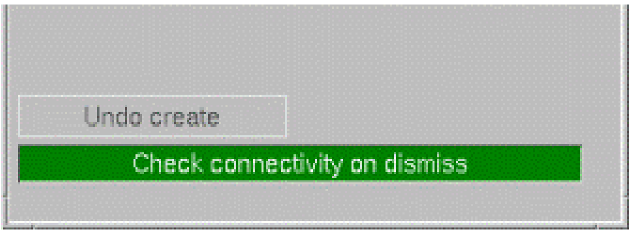
You can set any of the path points to the new start or end points by using the appropriate **Start** or **End** buttons next to the path point in the panel.



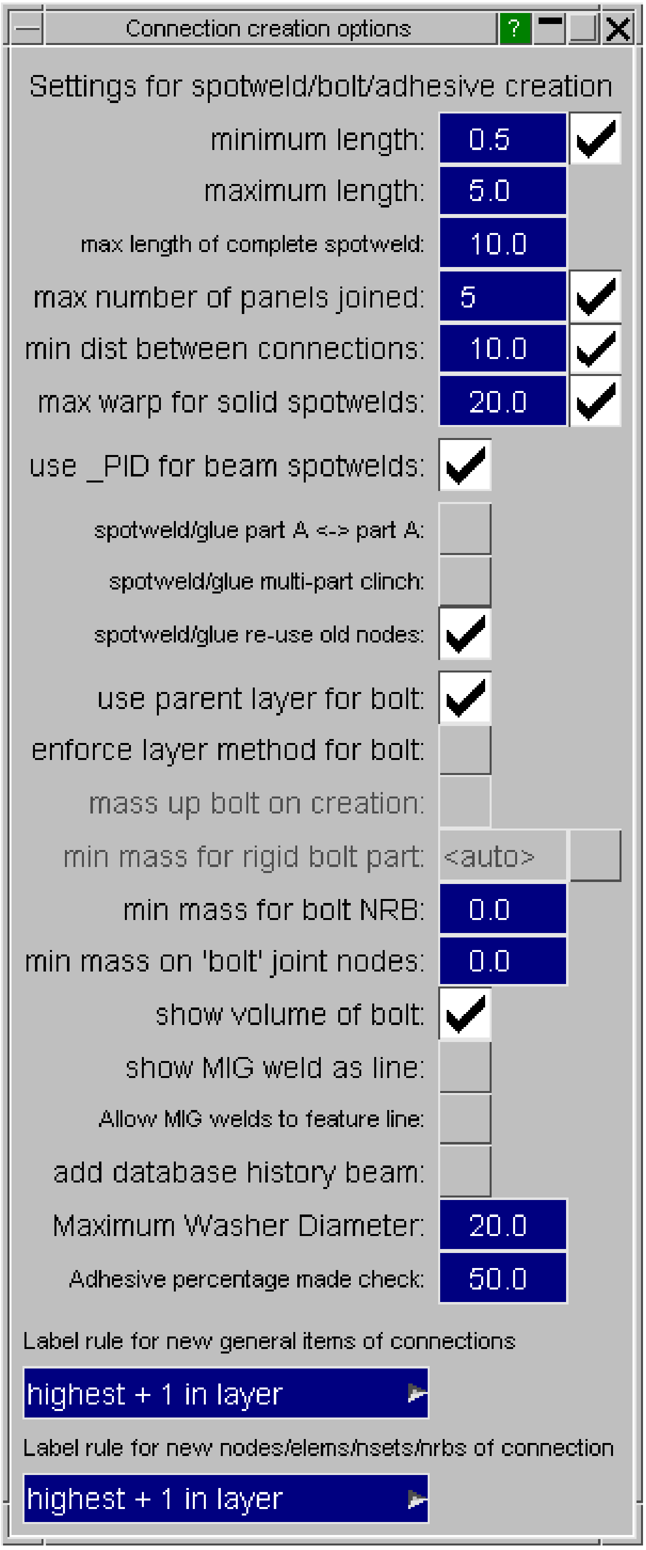
You can split the adhesive path by clicking on the **Split** button on the adhesive path panel, and then clicking on a node on the adhesive run. PRIMER will split the path at this point with the gap specified in the adhesive path panel. Note that this operation cannot be undone. After the split, the path information retained on the path panel is for the first of the two resulting adhesive runs.

Connectivity check

Upon exiting the connection create panel, PRIMER will automatically check the connectivity status of the connections. This check can be turned on or off at the bottom of the connection create panel (green is on).



Settings



The **Settings ...** option allows the user to select the label rule for any newly created FE items. This applies to spotwelds as well as bolts. One setting describes starter labels for nodes, elements, node sets and nrbs, another for all other items. The available rules are:

highest+1 in model

first free in model

highest+1 in include layer

first free in include layer

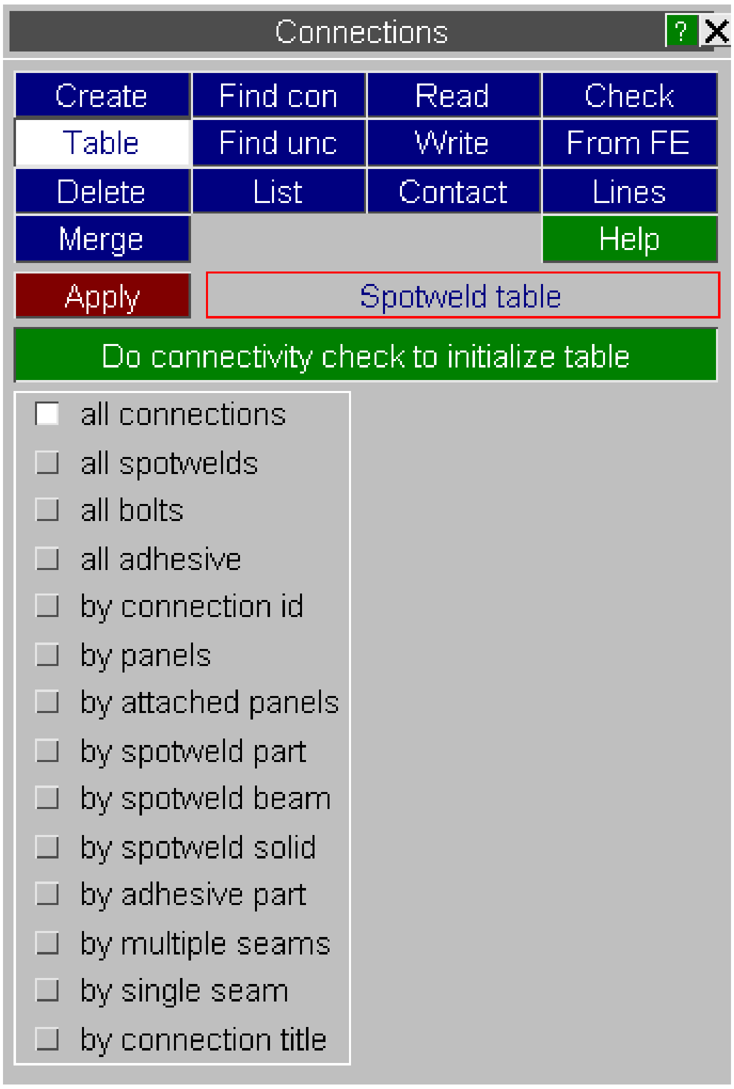
highest free in include layer

label above seed and **label below seed**.

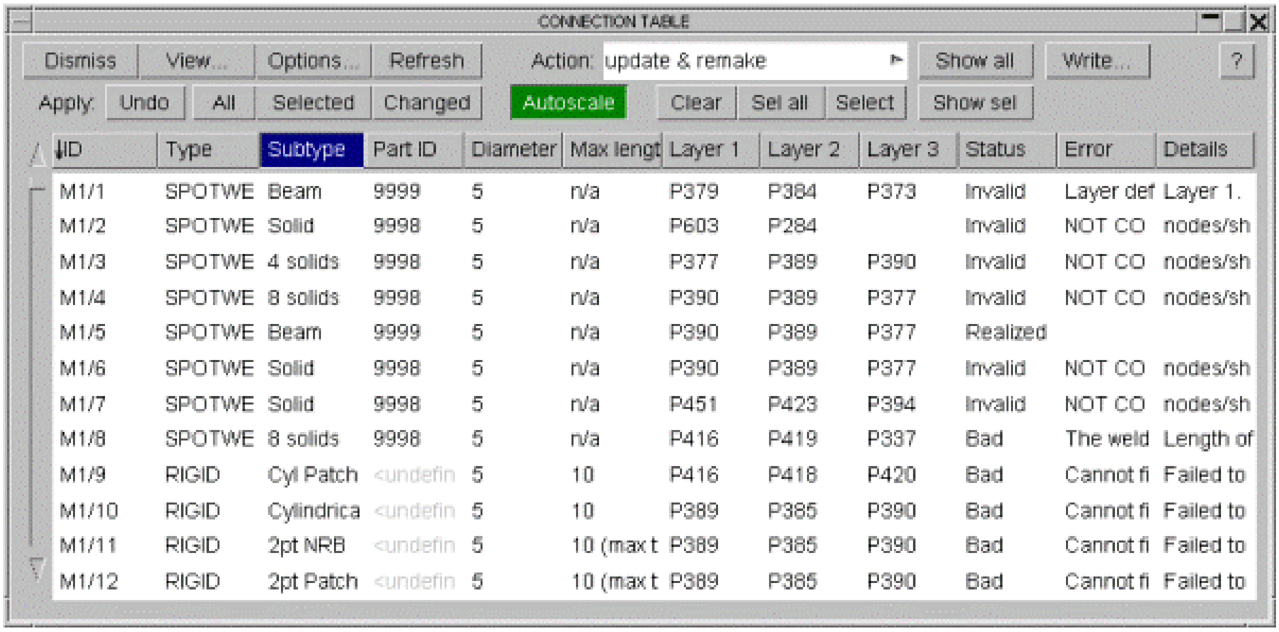
6.10.2 Connection table

The connection table allows you to review and modify connections data.

You can specify which connections to review a number of different methods. For more details of the different methods see section 6.10.0.



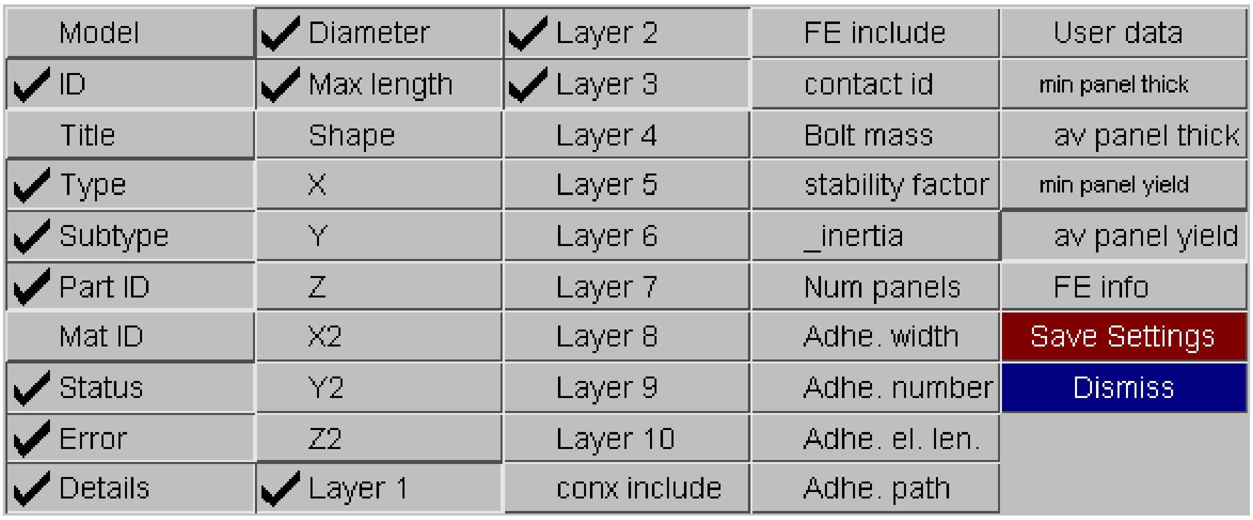
After specifying the connections to be reviewed, press the **Apply** button. The connection table window will appear.



Each row in the table represents a connection. At a glance, you can review connection Type (Spotweld, Rigid bolt or adhesive), element type (Beam or Solid), connection coordinates and the layers (Parts) connected together. The **Status** column tells you whether a connection has been defined and realized properly. If the connection has not been realized then the **Error** and **Details** columns give you information on the error. Often the error messages are too long to be shown in the column. In this case if you leave the mouse over the column the whole message will be shown in hover text.

Changing the table columns

The connection table can show various fields. To add or remove columns press the **View...** button which will bring the window shown below. The fields that are currently shown are marked with a tick symbol.



The columns can be made wider or smaller by dragging the sides of them in the header. The column order can be changed by dragging a column to a new position.

By default the rows are sorted by connection ID. You can sort by a different column by pressing on the column header. Pressing once will sort in ascending order. Clicking the column again will sort by descending order. The column that is currently used for sorting has an arrow drawn on it. This also shows if the sort is ascending or descending.

Available table columns

The following columns are available for display in the connections table under **View...**

Column	**Explanation**
Model	Model label
ID	Connection ID
Title	Title of the connection
Type	Type of connection (spotweld, rigid etc.)
Subtype	Subtype of the connection (beam, solid etc.)
Part ID	Part ID of the connection entities
Mat ID	Material ID for bolts (optional)
Status	Status of connection (realized, invalid, bad etc.)
Error	Error code for that connection
Details	More details on error
Diameter	Diameter of rigid bolt
X, Y, Z	Coordinates of connection (start point for adhesive)
X2, Y2, Z2	Coordinates of end point of adhesive
Layer 1,2,3,...	Layer information
conx include	Include file location of connection entity

FE include	Include file location of FE data within connection entity
contact id	Contact relating to connection entity
Bolt mass	Mass of bolt connection type
stability factor	Ratio of the required mass of the rigid bolt for stability to the total mass of the rigid bolt
_inertia	Part inertia flag
Num panels	Number of panels in connection (2T, 3T etc.)
Adh. width	Width of adhesive run
Adhe. number	Number of elements across adhesive width
Adhe. el. len.	Length of adhesive element along adhesive run
Adhe. path	Number of path points between start and end of adhesive (can also modify path through this column)
User data	Any typed in user data. This is not written out, so just remains for the current Primer session
min panel thick	minimum thickness of attached panels
av panel thick	average thickness of attached panels
min panel yield	minimum yield stress of attached panels
AV panel yield	average yield stress of attached panels
FE info	Information on FE that makes up the connection entity

Changing the default table columns

By default the connection table will show the Connection ID, type and subtype, Part, Diameter, 3 layers, status error and error details columns. If you want to change which columns are shown by default then change the columns shown to be the ones you want and press **Save Settings** in the **View...** popup. This will automatically add a preference `primer*conx_table_columns` to your home oa_pref file with the appropriate columns.

Selecting connections

Connection rows on the table can be selected in a number of ways. The easiest way is by left mouse clicking on the row. A selected row will be highlighted blue. Multiple lines can be selected using the shift or ctrl key combined with the mouse. There are also buttons at the top of the table to aid you in selecting and viewing connections in the table. The **Clear** button clears all current selections. The **Sel all** button selects all connections currently displayed on the table. **Select** will bring up an object menu and allow you to select connections using that method (i.e. being able to use various filters to select connections). **Show sel** will display in the table only those connections currently selected. **Show all** will bring back and display the original connections on the table.

Modifying connection data

The table window allows you to easily modify connection data. In the selected table row, right click on the field that you want to change. A popup menu allows you to change the option. Additionally, from all of the columns the following common options are available:

- **Update & remake**. This will remake any of the selected connection(s).
- **Sketch conx**. This will sketch the selected connection(s).
- **Sketch FE**. This will sketch the FE entities relating to the connection.
- **Show conx & panels**. This will blank everything apart from the selected connection(s) and associated panels.
- **Empty conx**. Empties the connection of it's FE entities, leaving the connection DORMANT.
- **Delete conx**. Delete the connection (and optionally the connection FE entities).
- **Upd & remake at (repos)**. This will remake any of the selected connection(s), and create the connection entity at the average of the nodal coordinates related to that connection (spotwelds and rigid bolts only).

The popup changes depending on the column. Some examples of connection modifications are given below.

Modifying connection Type

Right click a field of the **Type** column and the Change Type popup menu appears:

You can then choose between **Spotweld** or **Rigid** (Bolt) type connection.

Modifying connection Subtype

Right click a field of the **Subtype** column and the Change Subtype popup menu appears:

If the connection Type is **Spotweld**, the Change Subtype popup menu looks like this:

You can then choose the connection element type among the following options:

- Single beam element
- Single hexahedral solid element
- 4 hexahedral solid elements
- 8 hexahedral solid elements
- 12 hexahedral solid elements
- 16 hexahedral solid elements
- MIG beam elements

If the connection Type is **Rigid**, the Change Subtype popup menu looks like this:

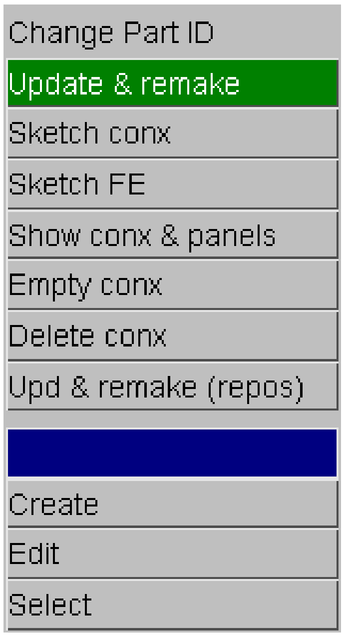
You can then choose the connection entity type among the following options:

- Various RIGID_BODY_MERGE types of rigid connection
- Various NODAL_RIGID_BODY types of rigid connection

For more information on different bolt types, see section 6.10.1.

Modifying connection element Part

Right click a field of the **Part ID** column and the Change Part popup menu appears:



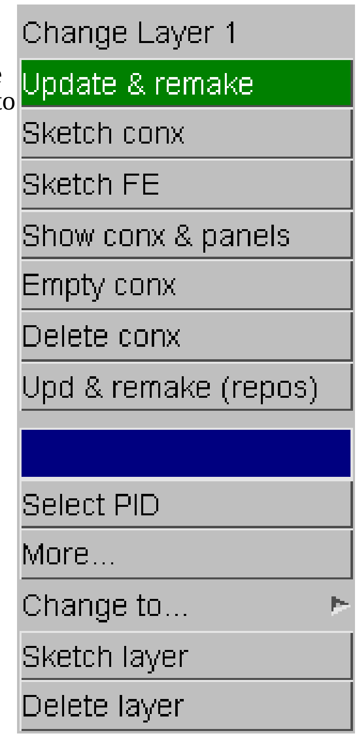
To modify the element Part data, you can type a new part ID in the text box. Alternatively, use **Select/Create/Edit** to choose a part from the part list, create a new part or edit a new part respectively.

Modifying connection coordinates

Right click a field of the **X / Y / Z** columns and the Change coordinate popup menu appears:

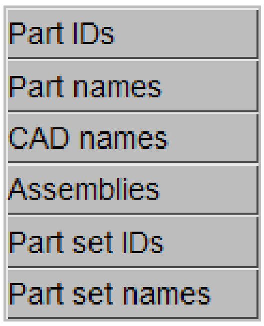
To modify the connection coordinates, you can type the new coordinates in the text box or pick a node. Alternatively, you can choose the Pick(from shell) option and using the cursor, select a point on a shell where the connection needs to be located.

Modifying connection layers



Right click a field of the **Layer** columns and the Change Layer popup menu appears:

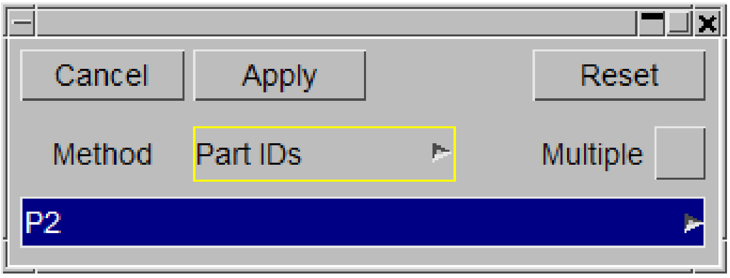
The vast majority of layer definitions will be a single part ID. In this case to modify the layer, you can type a new part ID in the text box or use **Select PID** which allows you to pick a part or select a part from an object menu.



However, layers do not have to be defined by part IDs. You can also define layers by:
- **Part IDs**
- **Part names**
- **CAD names**
- **Assemblies**
- **Part set IDs**
- **Part set names**

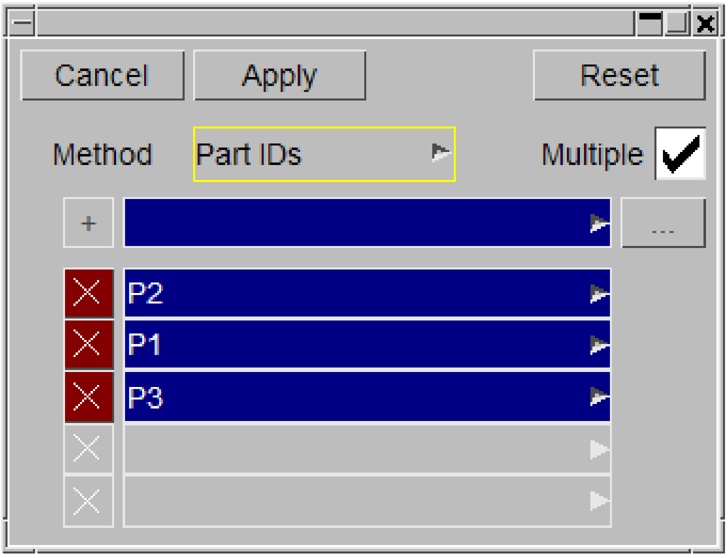
The **Change to...** option allows you to change the layer definition to be a different type. Click on the option of your choice and PRIMER will automatically update the connection layer type. For example you may want to use CAD names for the layer definitions instead of part IDs. If possible PRIMER will try to change any existing definition to the new type (e.g. if you change the layer definition from Part ID to Part name PRIMER will change the definition if the existing part has a name)

Alternatively, if you press **More...** a more detailed panel allows you change the layer definition.



Layers can also be defined with multiple parts, part sets etc.. If you press **More...** to access the detailed layer panel then you can select multiple parts by selecting the **Multiple** checkbox. In this case you can add/remove multiple items from the layer definition.

This is useful in some circumstances. e.g. if you are spotwelding a tailor welded blank then there could be multiple parts that represent the entire panel (as there are different thickness' for each part). If you wanted to make spotwelds involving this then it is much easier to include all of the parts for the layer definition for the blank. If you didn't then the part could vary depending on the position of the weld.



Modifying connection include

Right click a field of the **conx inc** columns and the Change conx include popup menu appears:

To modify the connection coordinates, click on **Change include**. The standard include select panel will open. See section 5.1.6 for more information on the include selection panel.

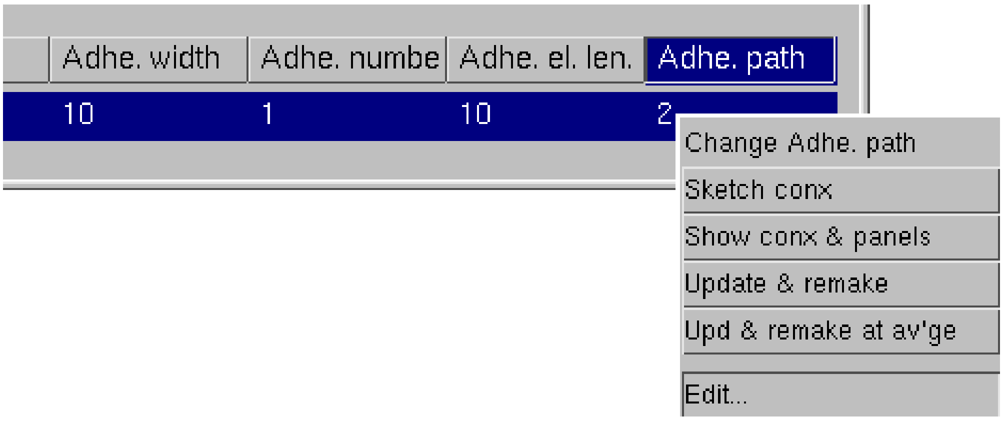
Listing the connection FE includes

Right click a field of the **FE inc** columns and the FE include popup menu appears:

This will print a listing to the screen with information regarding which include the various FE entities within the connection are in (nodes, beams etc.).

Modifying adhesive data

Adhesive data can be modified on the connection data. Adhesive width, number of elements across the width and adhesive element length can call be modified by right clicking on the field and typing in a new value. The path data can also be modified on the connection table. Under the **adhe. Path** column, the number shown is the number of path points between the start and end points of the adhesive run. Right clicking on the field and choosing **Edit** opens up the adhesive path modification panel allowing the user to modify the path data. For more information on the adhesive path modification panel see section 6.10.1.

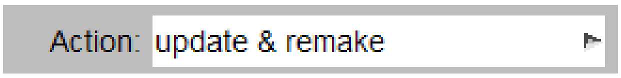


Deleting a layer from a connection

Sometimes you might want to delete a layer from a connection. For example in the connection window example at the top of this section, connection 1 has 3 layers. Layer 1 contains part ID 1, layer 2 contains part 3, and layer 3 contains part 3. If you wanted to delete layer 2 from this connection thus making a spotweld between parts 1 and 3 then right click on the **Layer 2** definition for the connection and select **Delete layer**. The entry will become blank. If you **Update & remake** the connection, the layer will be deleted.

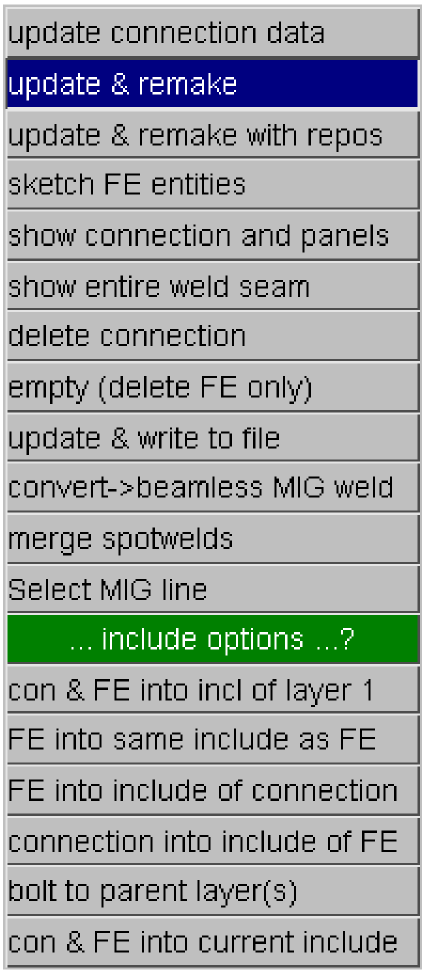
Changing the action for connections

The current action for the connections table is shown in the **Action** field.



Right clicking on the button will show the possible actions (shown on the right). The available options are:

- **update connection data**. The connection is updated with the current values in the table.
- **update & remake**. The connection is updated with the current values in the table and then remade.
- **Update and remake with repos**. The same as above, but the connection entity is created at the average position of the nodes associated with the connection.
- **sketch (with FE entities)**. The connection is sketched.
- **show connection and panels**. Everything apart from the selected connection(s) and associated panels will be blanked.
- **Show entire weld seam**. All connections that use the same layers are shown.
- **delete connection**. The connections are deleted. This gives the option to delete both the connection itself and the FE entities, or just the connection itself, leaving the FE entities unchanged.
- **empty (delete FE entities only)**. The FE entities that make the connection are deleted but the connection definition is left 'latent'
- **update & write to file**. The connection is updated with the current values in the table and then written to file
- **convert->beamless MIG weld.** Converts the connection to a beamless MIG weld.
- **merge spotwelds**. Merges spotwelds that are close to each other. For example two 2T welds can be converted to a 3T
- **select MIG line**. Select all MIG welds in a line with currently selected MIG welds.

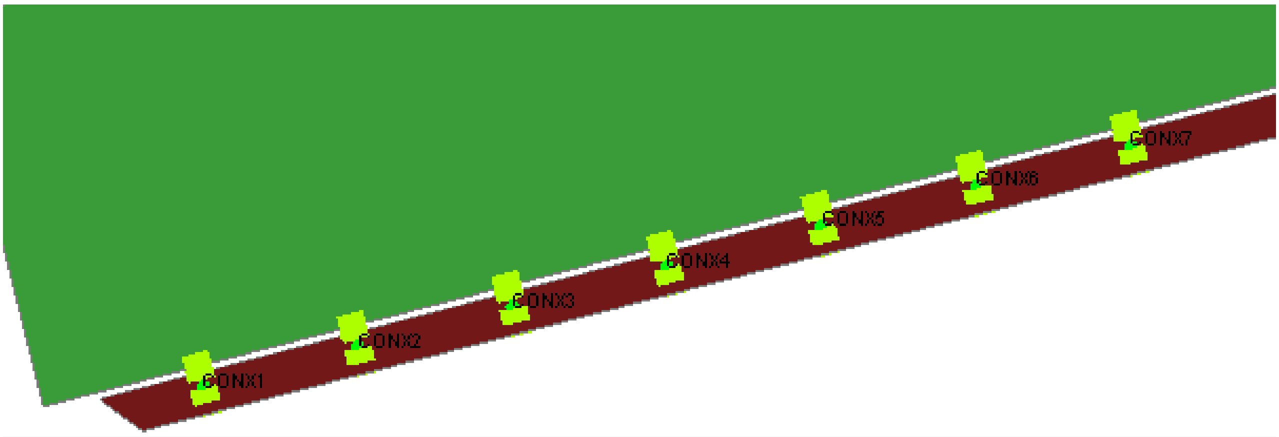


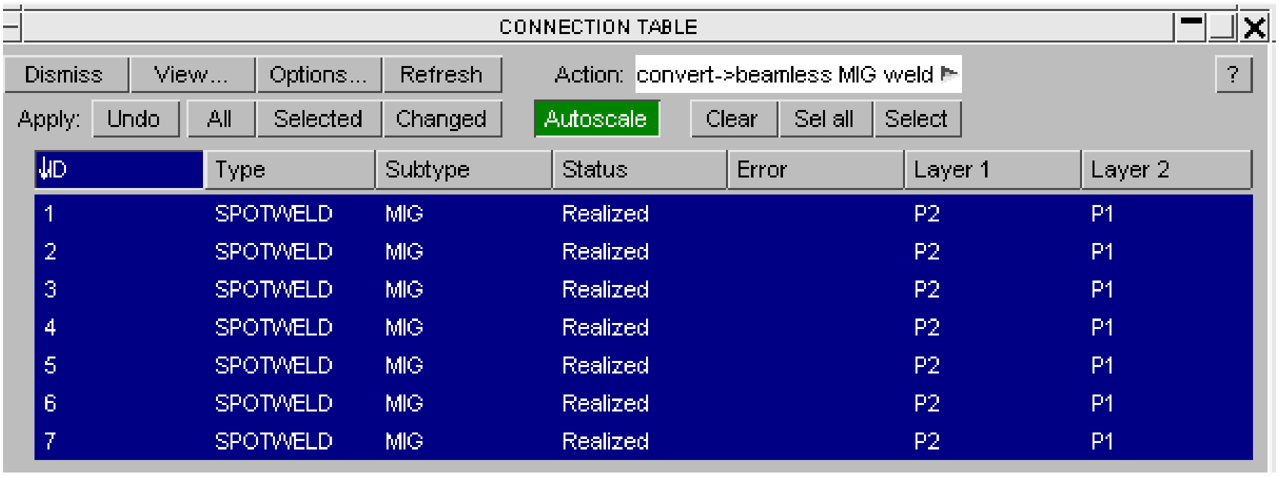
Additionally there are options for controlling which include file the connection entity and the FE entities are in.

Converting MIG weld to beamless

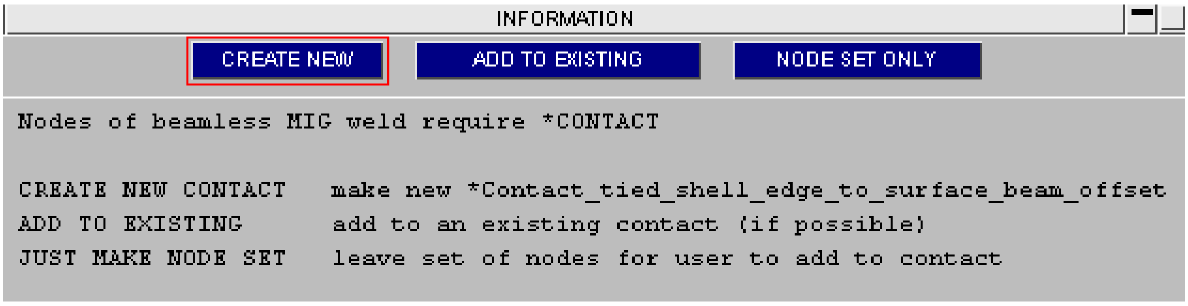
Primer supports a MIG beam weld which is meshed to a shell on one side and tied using spotweld contact on the other. If weld failure is not an issue, users may prefer to model this weld simply using *CONTACT_TIED_SHELL_EDGE_TO_SURFACE_BEAM_OFFSET with a node set on the slave side.

The function **convert -> beamless MIG weld** can be applied to a selection of conventional beam MIG welds (their status may be REALIZED or INVALID).

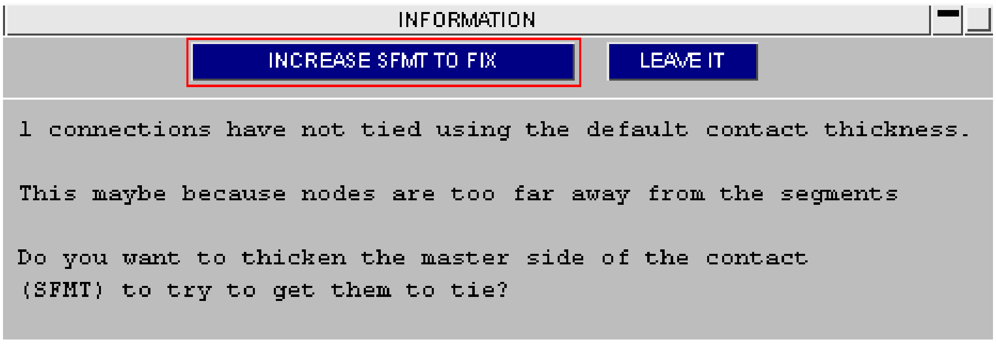




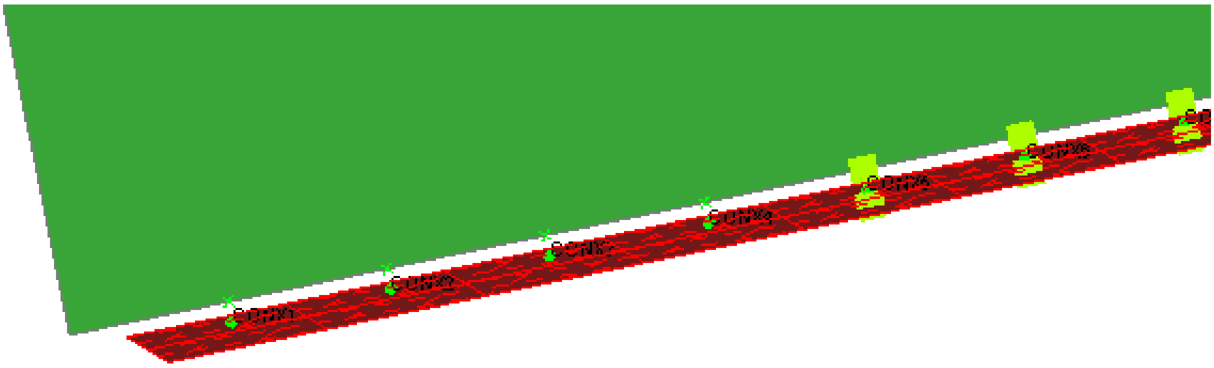
You may create a new _OFFSET contact, add nodes to node set of an existing one which is suitable (if any is found) or just dump the nodes to a set for sorting out later.



If the nodes are found not to tie because they are too far away you can run **INCREASE SFMT TO FIX** which will thicken the master side of the contact iteratively until all nodes are tied.



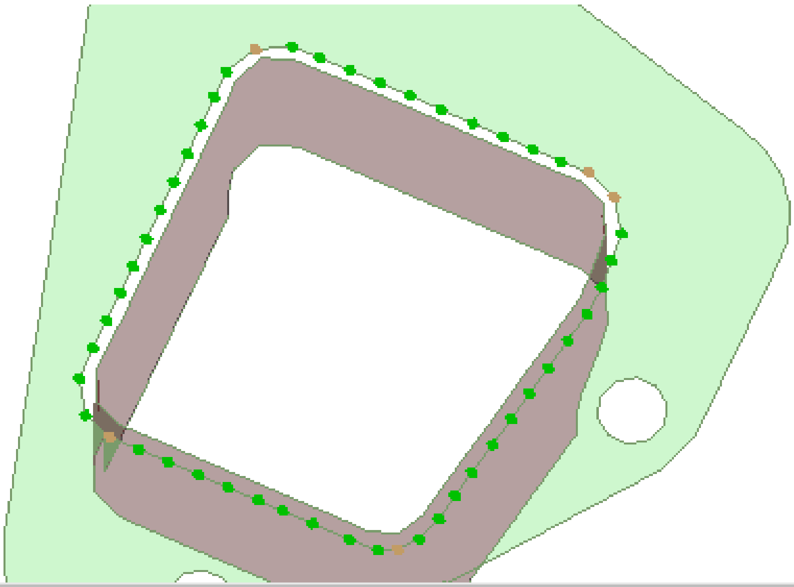
The contact alone then provides connectivity between the panels.

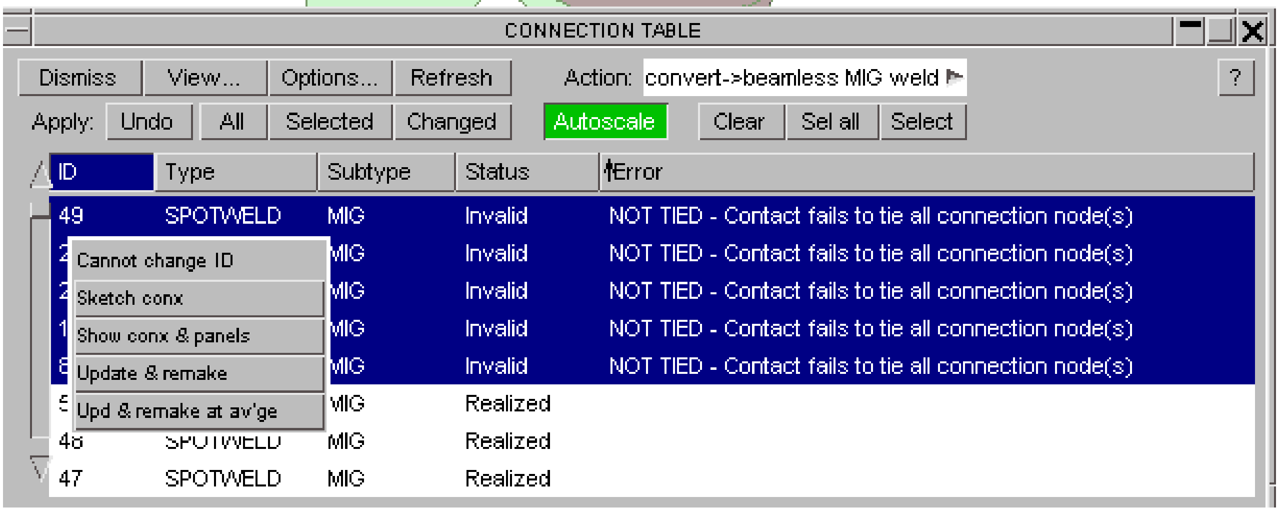


If Primer fails to tie all nodes, you will get the following error message.

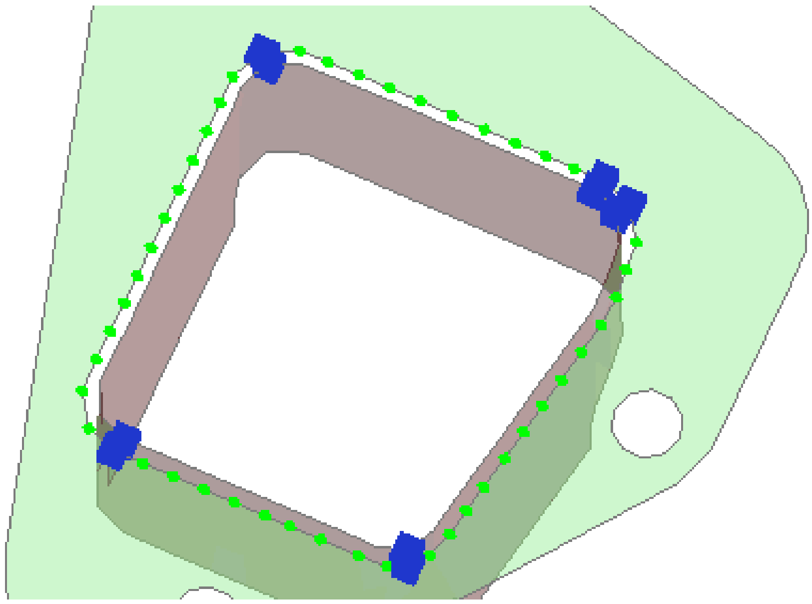
```
OK

Contact failed to tie all connections

Primer increased factor on master side contact thickness up to 4.0 but
5 connection nodes remain untied.

SFMT is currently set at 1.900000. Increasing it will not help.

Nodes may not be tying for another reason. Please check your contact
using Primer's 'Contact Pen Check' function. You may find it useful to turn on
segment thickness - 'Settings ...' -> 'As thick'
```

The connections that failed to convert are left with **NOT TIED** error and **invalid** status (denoted by orange colour).





You can use **update & remake** to reform these as conventional beam MIG welds. Alternately, you may be able to get them tie by adjusting parameters on the tied contact which control the search depth, such as MAXPAR. Such tuning is beyond the scope of this function.

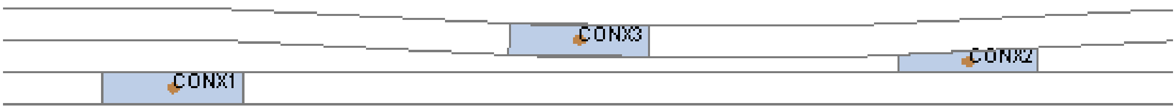


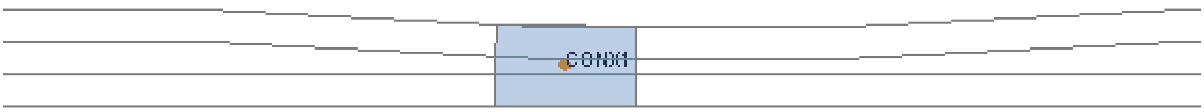
Merging spotwelds

The action **merge spotwelds** provides an alternate method to deletion for dealing with conflicting welds. The function uses parameter **min dist between connections** if it is non-zero (see settings). Two or more spotweld connections may be selected on the table and the action applied. Primer will then calculate the average position of the selected welds. To proceed the function requires that

- all welds are within **min dist between connections** of the average position (if set to zero this restriction is ignored)
- all welds must share at least one layer with another selected weld
- all welds must have the same sub-type, PID and diameter

Primer will then attempt to make a weld at the average position which connects all the layers involved. If this is successful the old welds will be deleted, if it fails they should be left unchanged.





Modifying the include (layer) of connections and their FE

- **Con & FE to include of layer 1** - move connections and FE into layer of first found part in layer definition 1
- **FE into same include as FE** - move all FE of connection into layer of primer FE element
- **FE into include of connection** - move all FE into same layer as connection itself
- **Connection into include of FE** - move connection (and FE) into same layer as primary FE element
- **bolt to parent layer** - if all connected shells in same layer, move connection and all FE into that layer. If in different layers, move rigid shells/parts/materials to same layer as overlayed parent shells, nut connection/master part/C_RBOD/NRBC remain un-moved
- **Con & FE to current layer** - move connection and all FE into current layer

Primary element in this context means first found beam/solid for spotweld, NRBC or master part for bolt.

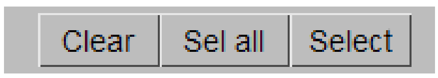
Applying/undoing connection action

Once you have selected the action you require pressing the **All**, **Selected** or **Changed** buttons will apply the action to all connections, the connections that you selected, or only the connections that are changed respectively.

To Undo any connection modification that you made (shown in red in the table), press **Undo**.

Selecting connections in the table

If you want to select connections in the table you can clear the entire selection or select all connections using **Clear** or **Sel all** respectively. Alternatively you can use **Select** to pick connections from the screen or select them using an object menu.



Display only selected on the connections table

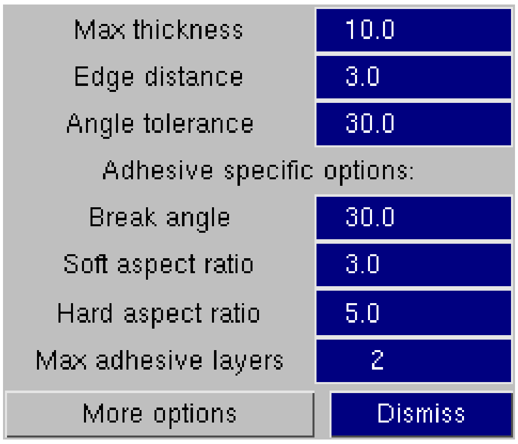
The connections displayed on the table can be changed to show a subset of the original connections placed on the table. This is done by selecting on the table the connections you wish to show (either by clicking on them or using the Select button). Then click on the button **Show Sel**. This will update the table to show just the connections selected. Any **Apply** operation (**Undo, All, Selected, Changed**) will now just apply to those connections displayed on the table. To display on the table all the connections that were originally on the table click on **Show all**.

Write the connection table data as a CSV file

The **Write...** button on the connections table allows you to write the current table contents as a CSV file.

Options in the connections table

Pressing the **Options...** button shows a popup allowing you to change options for how connections are treated in PRIMER. For more details see the section on connection options.

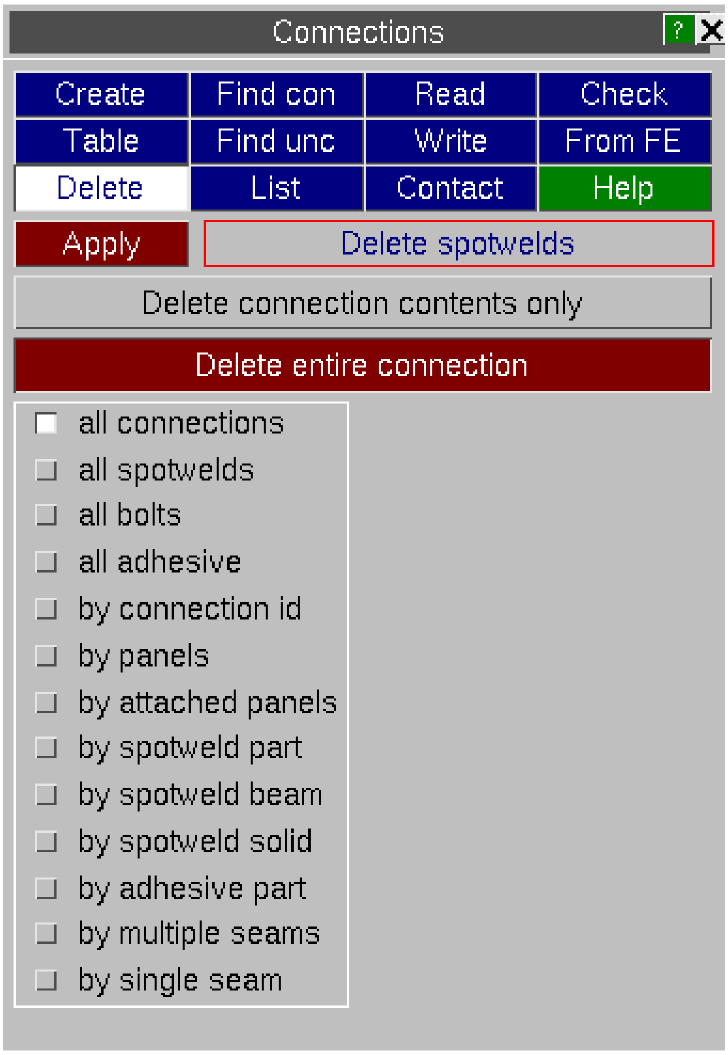


6.10.3 Deleting connections

This panel allows you to delete connections and their related entities.

You can specify which connections to delete by a number of different methods. For more details of the different methods see section 6.10.0.

Once you have selected which connections you want to delete and the method for deleting them pressing **Apply** will delete the selected connections.



Selecting the deletion method

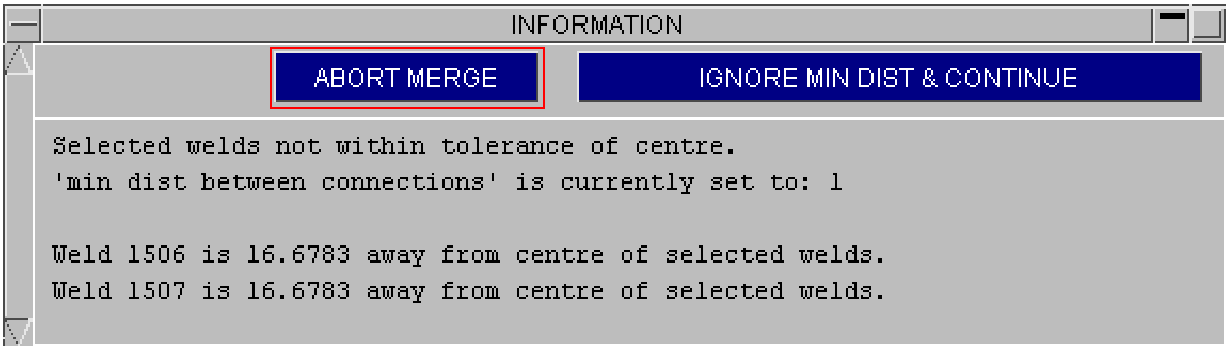
If **Delete entire connection** is set , connections and related FE entities (spotweld beams, NRBs, etc.) will be deleted all together. The connection point is lost.

If **Delete connection contents only** is set, the connection related FE entities will be deleted but the connection data (coordinates, layers, etc.) will not be deleted and therefore the connection can be remade later.

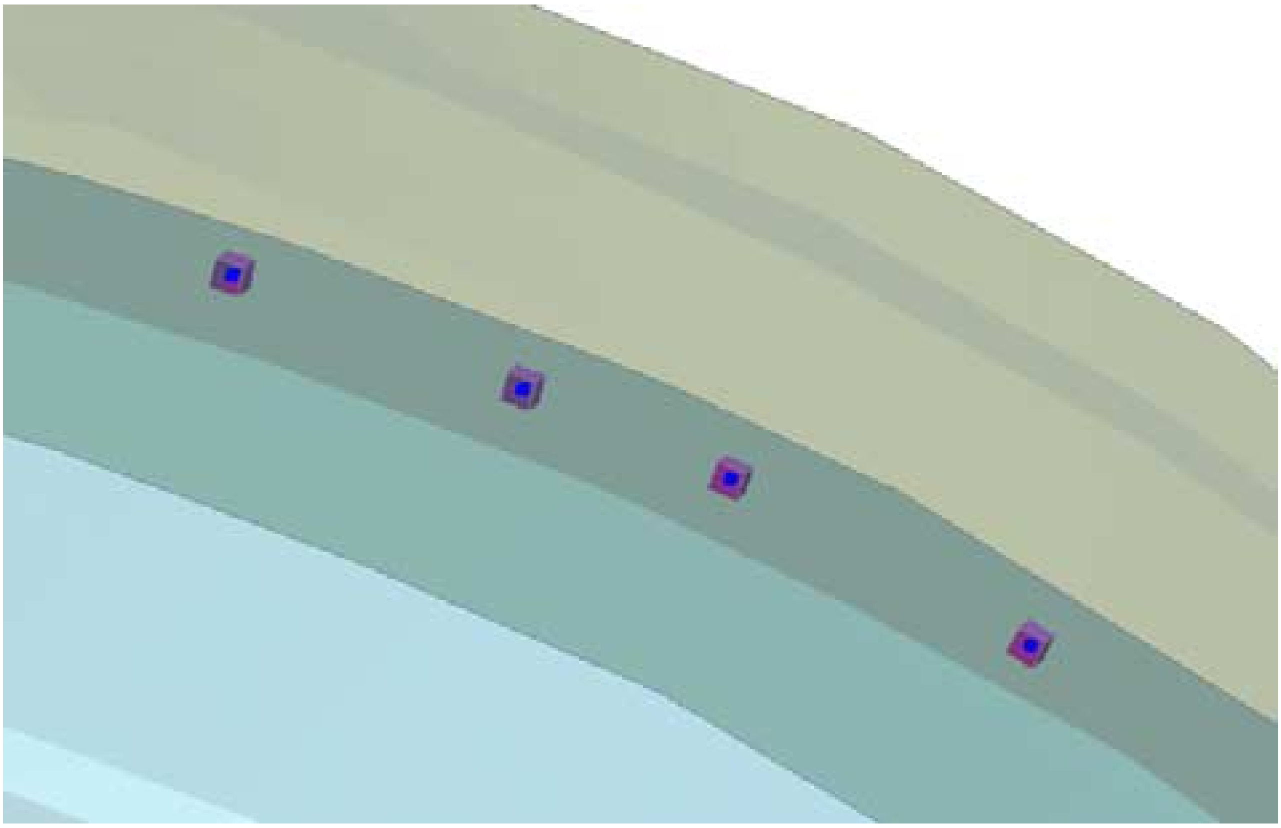
Merging spotweld connections

Two or more welds can be merged together by selecting them and applying the merge. A weld will be made at the average coordinate of the selected welds, re-making the layer definition as necessary.

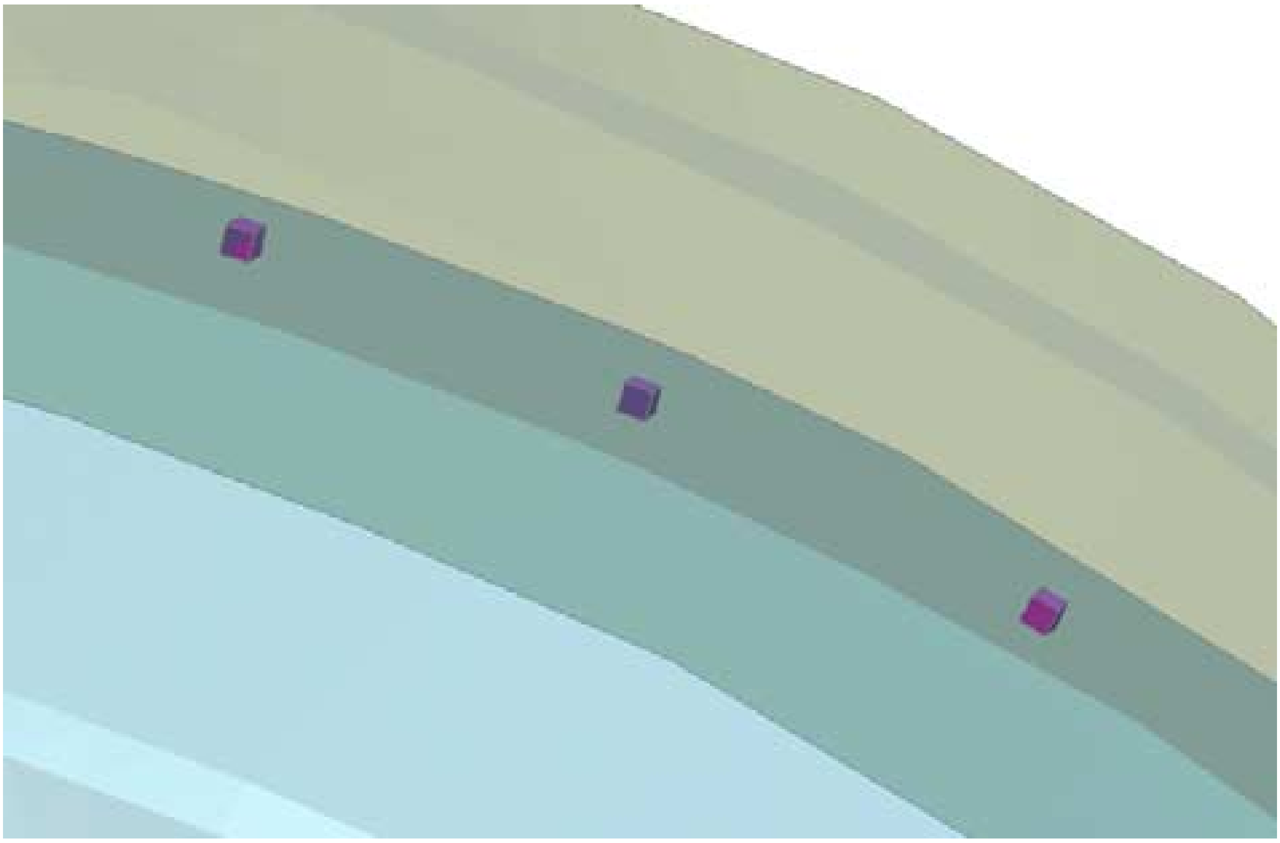
If the option **min dist between connections** is set, Primer will expect all selected welds to be within that range of the averaged position. If not a warning will be isssued.



The selected welds must share at least one part, have the same pid, configuration and diameter. Then the merge will proceed.



The two cetral welds have been merged to form a new weld.



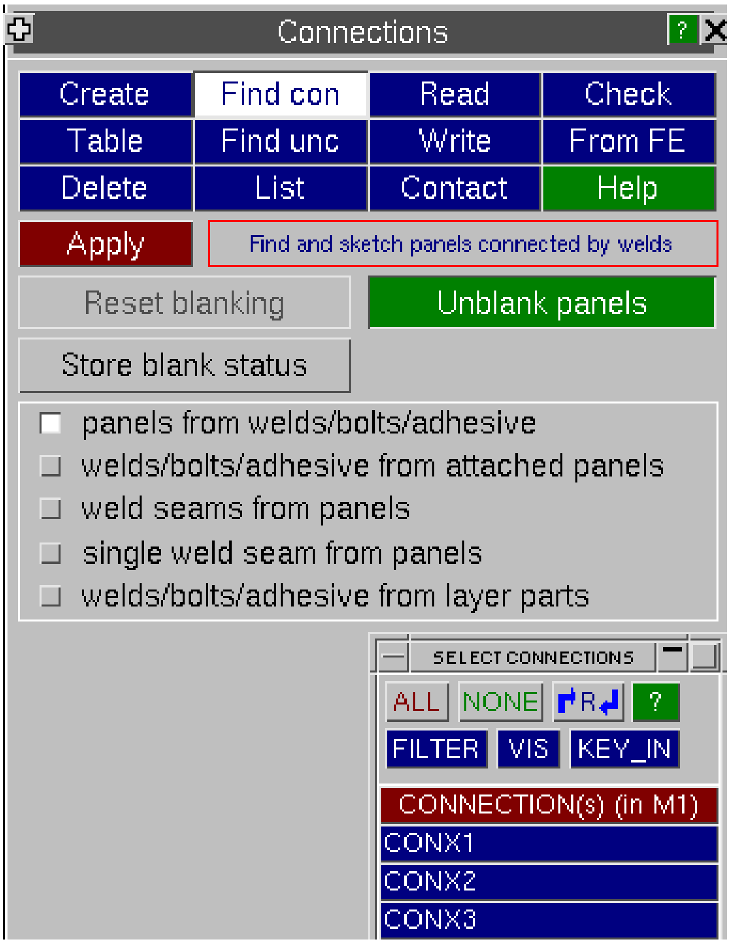
6.10.4 Finding connected panels

This panel allows you to find connections that are tied to panels. It also allows you to find panels tied to connections.

There is also a switch that alters which panels are displayed.

When finding panels tied to connections (**panels from welds/bolts/adhesive**), select the connection and press **Apply**. PRIMER will blank the whole model then unblank the connection and all panels attached to it. In order to undo the blanking that PRIMER has just done, press the **Reset blanking** button.

When finding connections tied to panels (**welds/bolts/adhesive from panels**), select the panel you wish to find the connections attached to by any of the usual methods and press **Apply**. This option will only find attached if the connection has been made, i.e. contains FE entities. To find connections associated to a panel by connection layer (i.e. the connection may not be realized and may not contain FE entities) use **welds/bolts/adhesive from layer parts**. If the **Unblank panels** button is set, PRIMER will blank the whole model then unblank the selected panel, the attached connections and any other panels tied to these connections. If the **Unblank panels** button is not set, PRIMER will blank the model then unblank the selected panel and attached connections. In order to undo the blanking that PRIMER has just done, press the **Reset blanking** button. You can store the blanking status by hitting **Store blank status**.



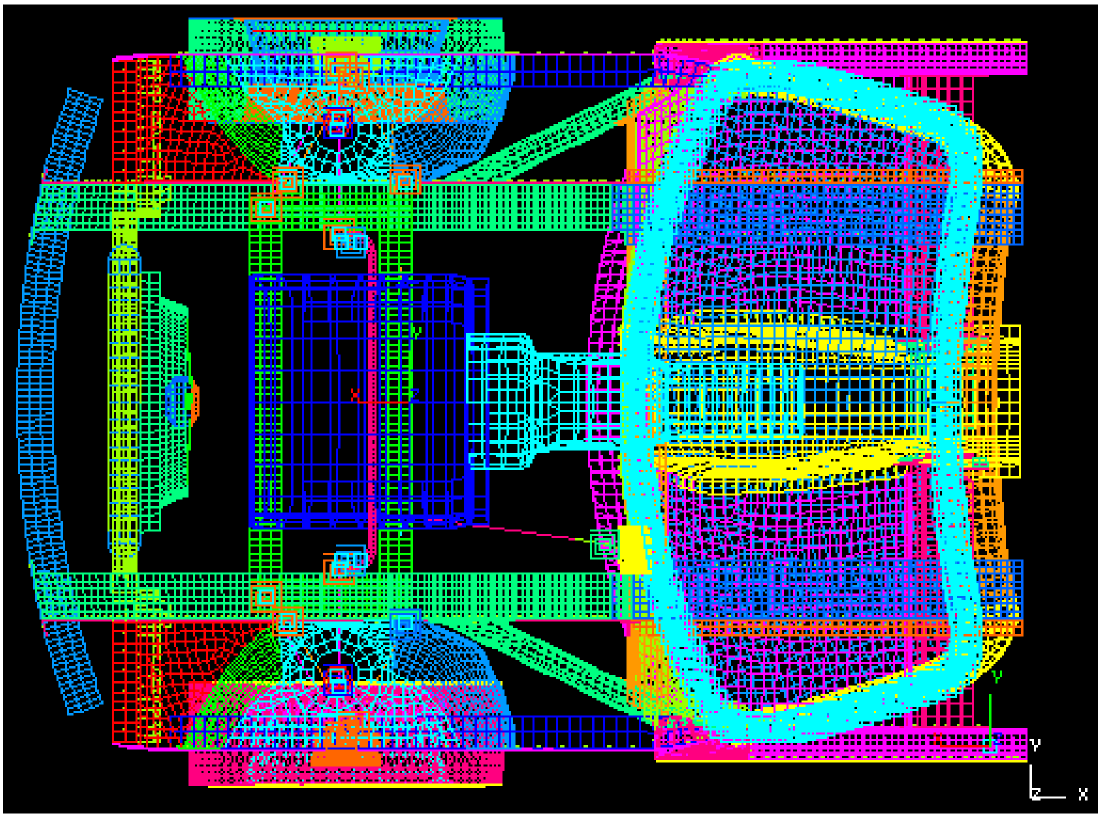
When finding seams connected to panels - select the panels you want the seams to join and press **Apply**. If you select parts 1, 2, 3 and 4 then **ANY** connection which ties **ANY** combination of these parts together will be shown.

When finding a single seam from a panel, select the panels you want the seam to join and press **Apply**. If you select parts 1, 2 and 3 then **ONLY** the connections joining **ALL** of these panels will be displayed.

For more on how the function works look at the following example.

Example

The following figure shows the front of a vehicle. It has been welded together using the PRIMER spotwelding ability. We want to find which panels are attached to the floorpan by spotwelds or bolt connections.

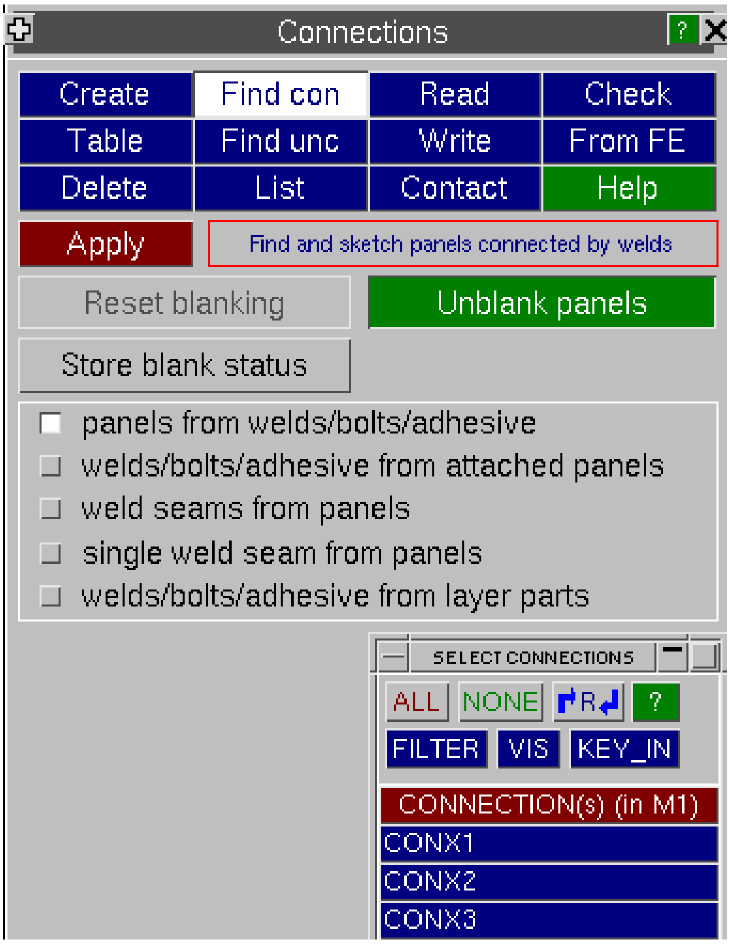


First, we select the panels we want to find connections attached to by either picking the panel from the screen or selecting the panel from the list.

Secondly, set/unset the **Unblank panels** switch.

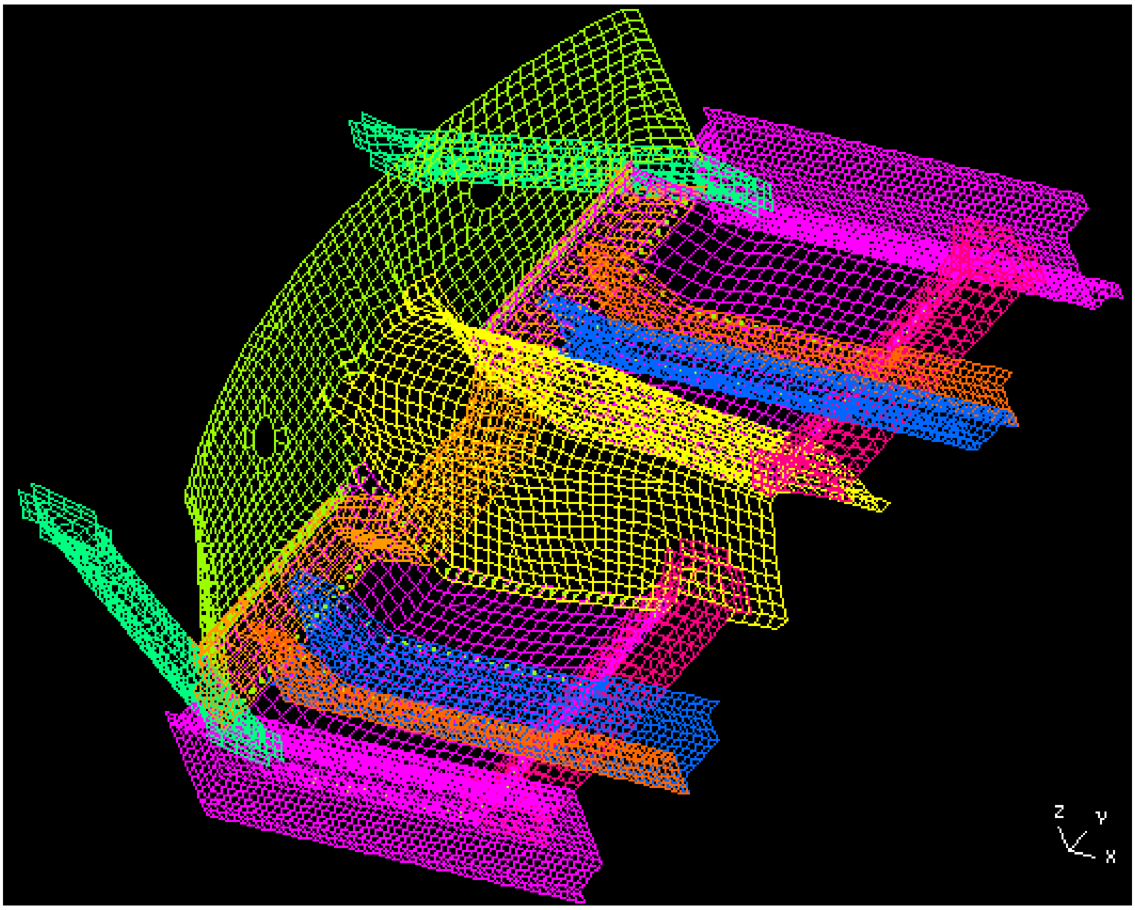
Press the **Apply** button.

Result if **Unblank panels** is set.
Result if **Unblank panels** is unset.



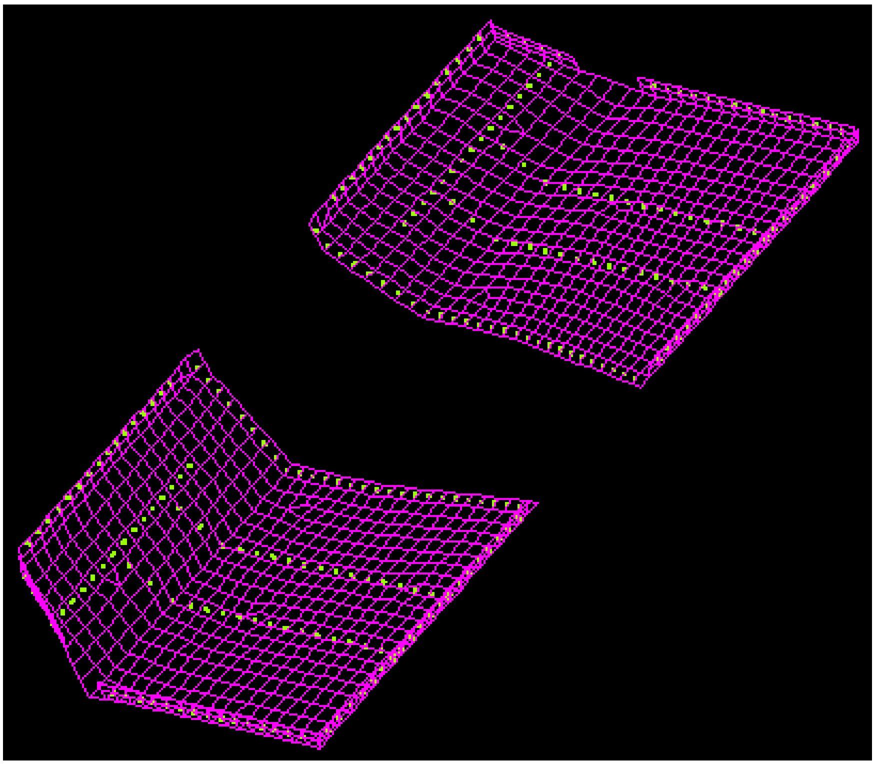
Result if **Unblank panels** set

If the switch is set PRIMER will blank the model, unblank the part you selected, find and unblank the connections attached to that part, and also find and unblank the panels that are attached by those connections.



Result if **Unblank panels** unset

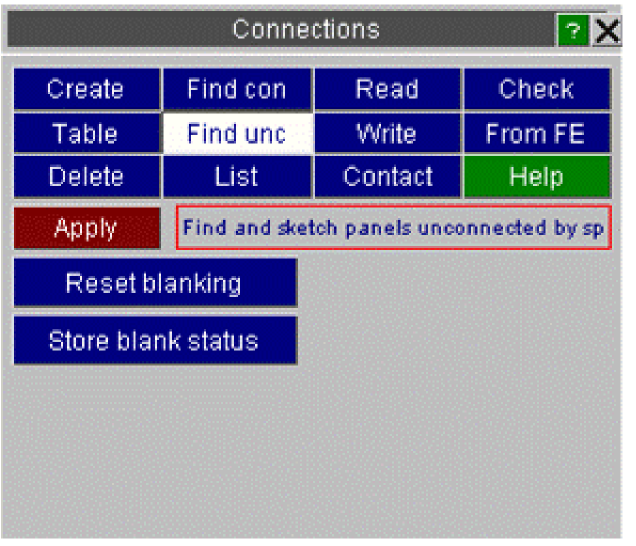
If the switch is unset PRIMER will blank the model, unblank the part you selected and find and unblank the connections attached to that part.



6.10.5 Finding unconnected panels

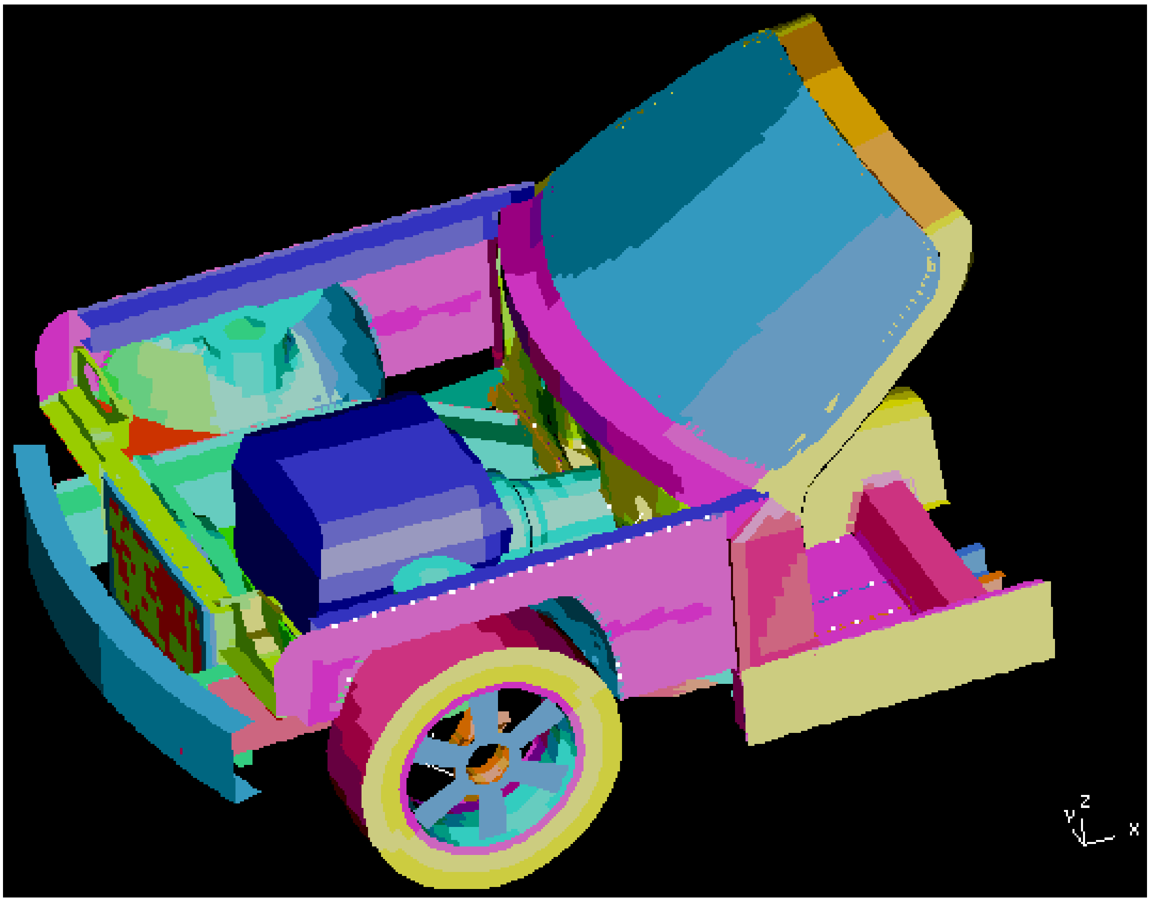
This tool allows you to find panels that are not attached to any connection. This facility enables you to quickly check that the panels you expect to be connected together,either by bolts or spotwelds, actually are welded together!

To see how the function works look at the following example.



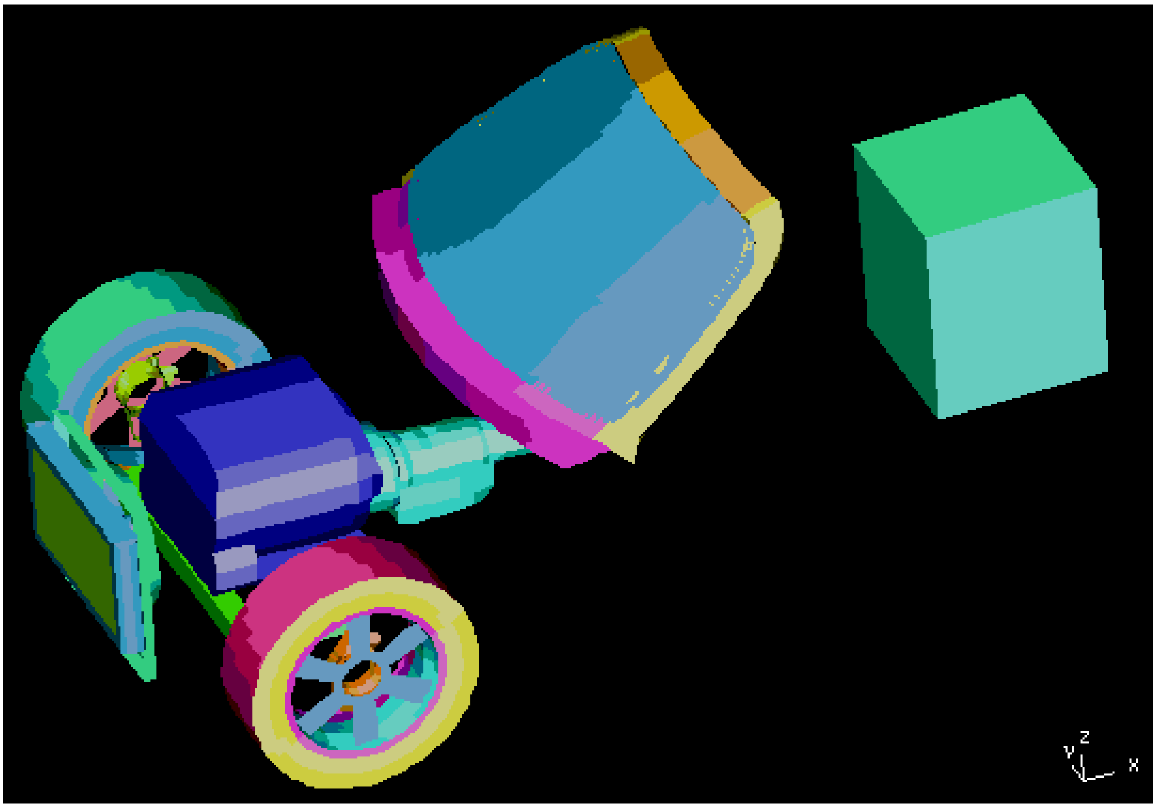
Example

The following figure shows the front of a vehicle. It has been welded/bolted together using the PRIMER connections ability. We want to find which panels are not attached by any spotweld nor beam connection.



Press the **Apply** button.

PRIMER blanks the model, and unblank any parts in the model that are not attached together by any connection.



This shows that the radiator, wheels, engine, gearbox, windscreen and screenrail are not connected to the rest of the vehicle. In this case the screenrail should be connected by spotwelds! We can now go back and fix this before submitting the job. Doing this quick check can help find problems which may be missed otherwise.

Reset blanking and **Store blank status**

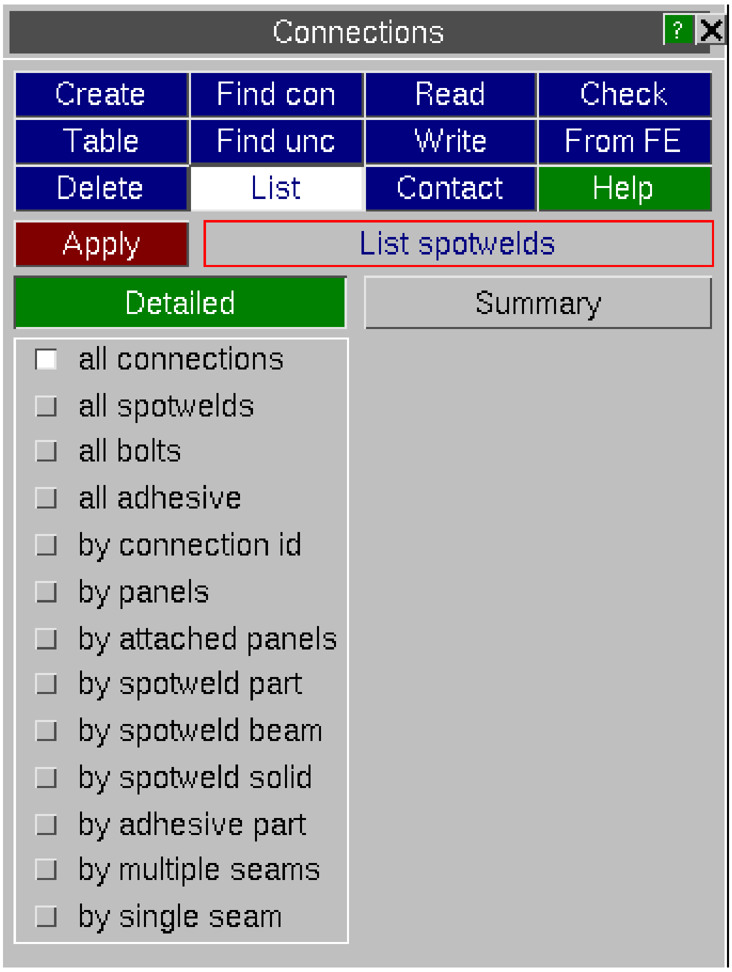
If you want to undo the blanking that PRIMER has just done you can press **Reset blanking**. PRIMER will reset the blanking to the previous state. Using **Store blank status** will store the current blanking status - this is useful because when you exit the panel it goes back to the stored status.

6.10.6 Listing connections

This panel allows you to list connections.

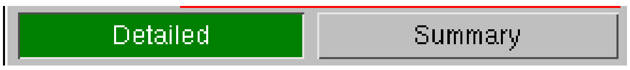
You can specify which connections to review by a number of different methods. For more details of the different methods see section 6.10.0.

There is also an option that effects how the spotwelds are listed :
Summary or **Detailed**.



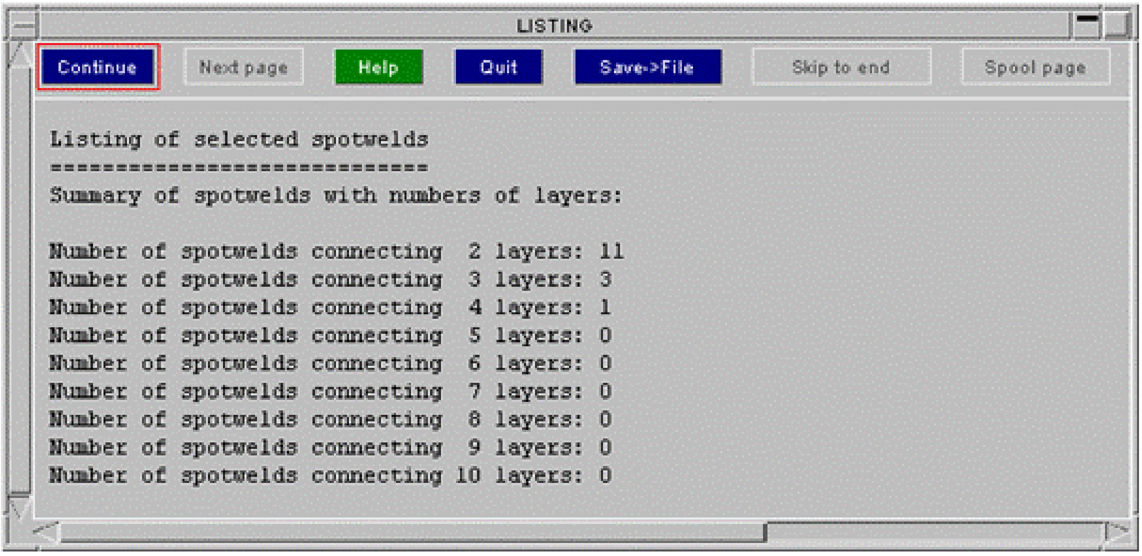
Summary or detailed listing

When you select connections to list you can choose how connections are listed. The options are either **Detailed** or **Summary**.



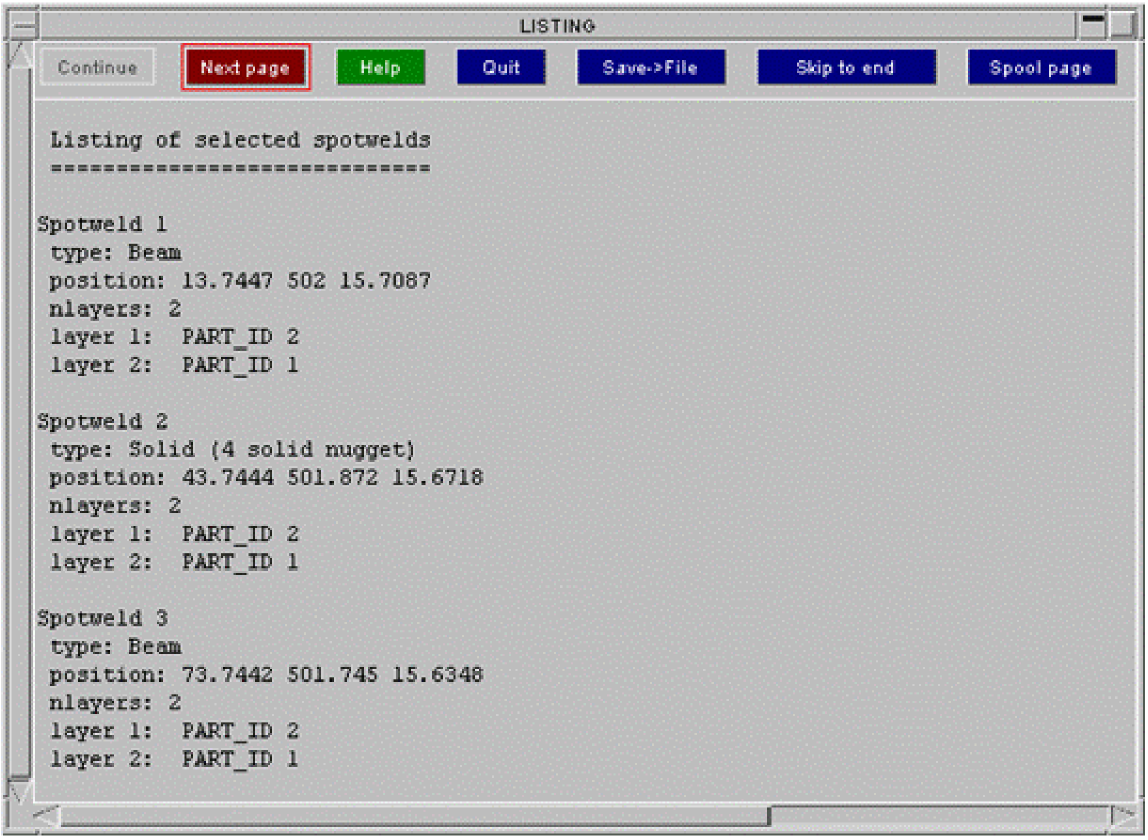
Summary listing

In summary listing PRIMER shows how many connections tie 2, 3, 4... panels together. This is useful as a quick check. If you know that the maximum number of panels that you weld for any spotweld is 3, and there is a 4 noded weld then there is an incorrect weld.



Detailed listing

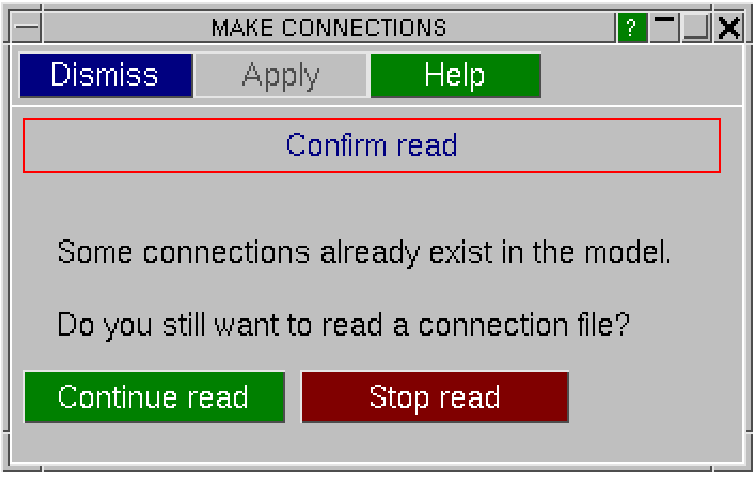
In detailed listing PRIMER shows the coordinates and panels of each connection. The format of the listing is similar to a PRIMER connection file.



6.10.7 Reading connections from a file

When you go to read a connections file, if your model already contains connectionsPRIMER will ask you if you want to continue with the read or not (see figure on right). Press either **Continue read** to continue or **Stop read** to finish.

If your model does not contain any connectionsthe main read panel will be displayed.



The main read panel allows you to read mesh independent connections from a PRIMER spotweld file, an xml connection file, a Catia spotweld file, a UG file or other type of file. PRIMER can also store a user defined script using PRIMER's javascript functionality. The script can be written to read a particular file format and then is used through this panel to read connection data into PRIMER. See section 10 for information on PRIMER's scripting ability.

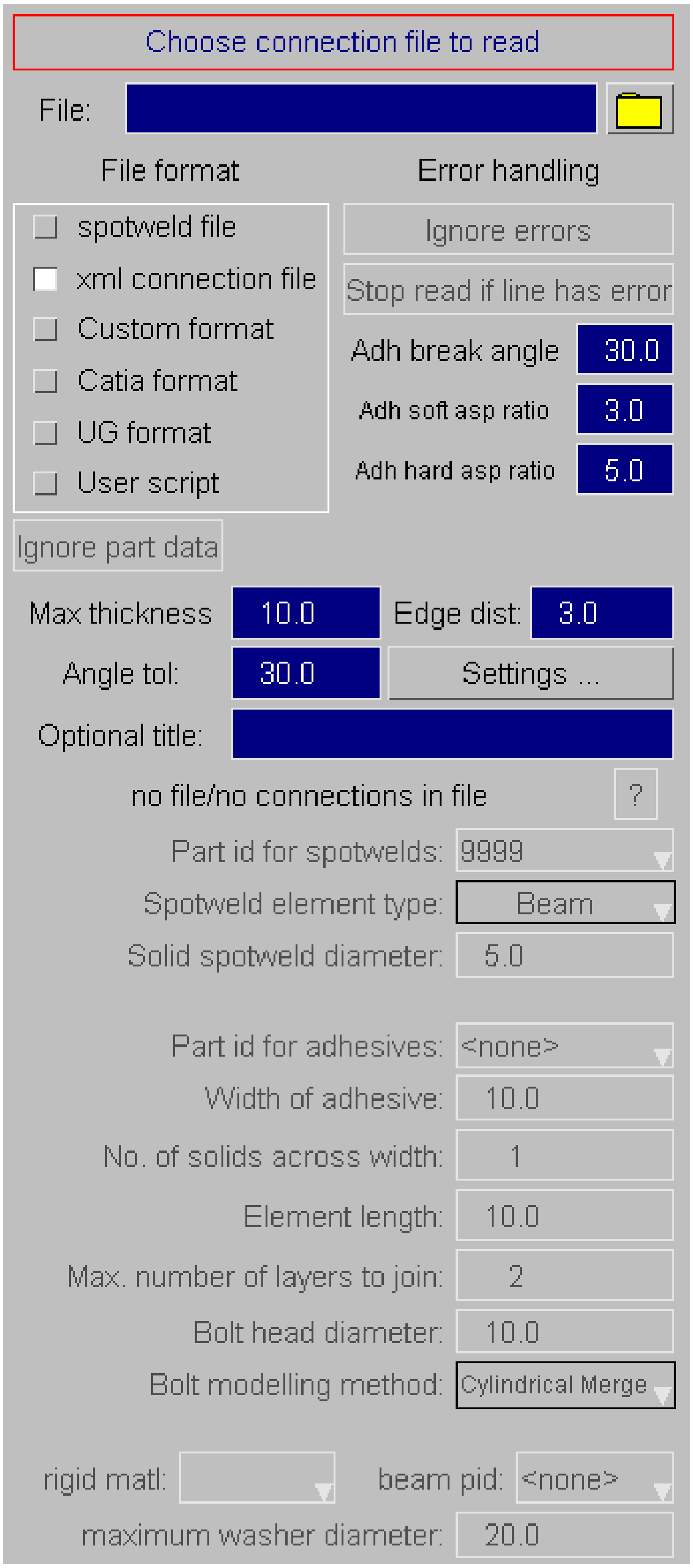
Choosing part for beams/solids and filename

Choosing file format

Error handling

Options

Actually reading the file



Specicifying a title

An optional title can be specified when reading in connections. This title will be applied to every connection read in. This is useful for keeping track of where connections came from in your model. For example, you can set the title to be the same as the name of the file read in. Connection titles can be displayed in the connection table.

Choosing part for beams/solids and filename

PRIMER needs to know what type of spotwelds to create and which part to put the spotwelds into. These can be set with **Part id for spotwelds** and **spotweld element type**. Additionally if you are making spotweld solids PRIMER needs to know what size to make the solid spotwelds. This is set with **solid spotweld diameter**.

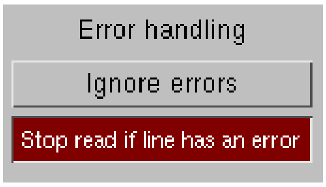
To select a part type in the part number, or you can use the standard popup functions (right click) to select or create the part. The part **must** use material type **`*MAT_SPOTWELD`** (and **`*SECTION_BEAM`** type 9 for beams). Once the part has been selected or created the part number will be displayed in the box:
If there is only one part in the model that is suitable (i.e. for beams if the part uses material `*MAT_SPOTWELD` and section type `*SECTION_BEAM`, or for solids if the part uses material `*MAT_SPOTWELD`) then PRIMER will automatically select it. Otherwise you will have to select it. Other inputs are avialable and will become ungreyed once the file to be read in has been selected.

Choosing file format

PRIMER is designed to be able to read a wide variety of files. PRIMER has it's own spotweld file format. If this format is used then PRIMER knows which fields are which in the file and so it can be read in one step. Select **Primer spotweld file**. PRIMER also supports Catia and UG spotweld files and the format is set automatically in PRIMER so it can also be read in one step. Select **Catia format** or **UG format**.

If the file you want to read is not a PRIMER or Catia spotweld file then you need to tell PRIMER how it should be read. Select **Custom format**.

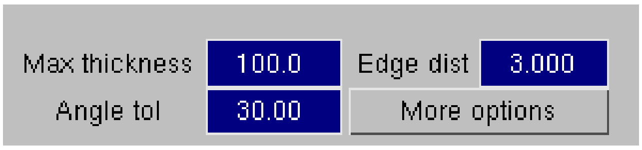
Error handling



When PRIMER is reading a **PRIMER spotweld file** there are 2 possible actions that can be taken if an error occurs when reading the file: Either stopping the read completely when an error is found, or ignoring an error.

To show the two options, consider the following example. PRIMER is reading a line from the spotweld file and has already read the X, Y and Z coordinates and panels 1 and 2. Now, when trying to read the third panel, an error occurs as PRIMER reads the string 'aaaa' from the file and it is expecting to read a panel number. If **Stop read if line has an error** is selected PRIMER will stop and give the line number of the error. If **ignore errors** is selected, PRIMER will just ignore the third panel and make the weld (if possible) between panels 1 and 2 at (X, Y, Z).

Options



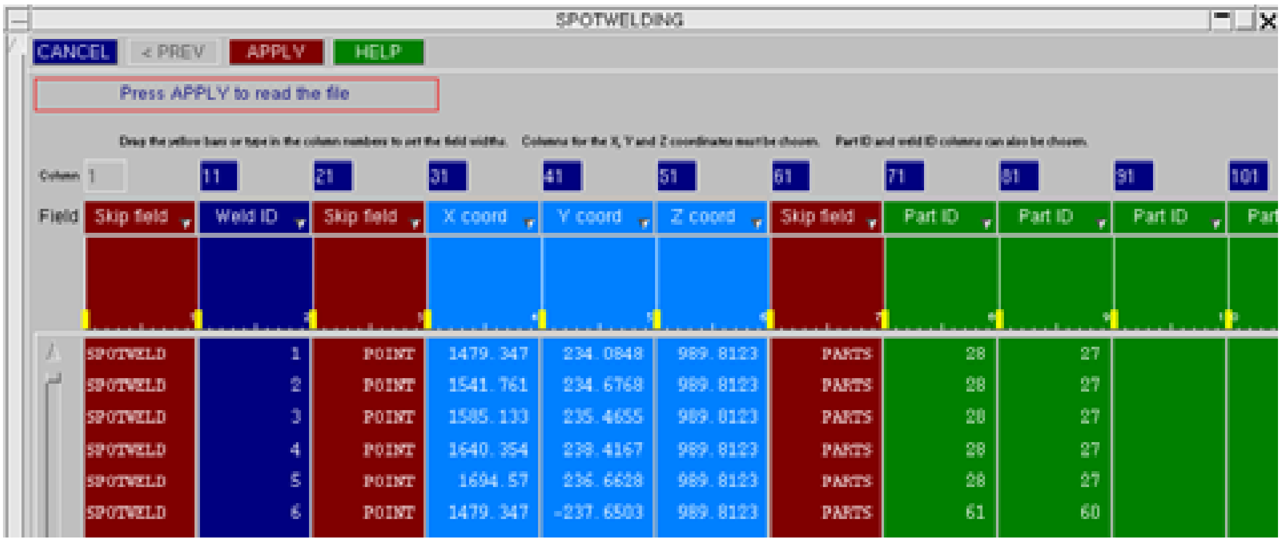
These options control how the spotwelder works. They are identical to the options that are used when creating spotwelds. For more information see the section on spotweld options

Reading the file

Pressing **APPLY** will start reading the file. How the file is read is dependant on the file format chosen: PRIMER spotweld format, Catia spotweld format or Custom format.

PRIMER spotweld file

PRIMER will read the first 50 lines of the file and put a preview on the screen.



The preview should be similar to the image above. PRIMER knows what each field is so everything is set for you automatically. Pressing **APPLY** will now actually read the file. If the file is not a PRIMER spotweld file or you want to stop the read press **CANCEL** to return to the main reading panel.

XML connection file

From version 9.3 PRIMER can use a new xml format for connections. This contains more information than the PRIMER spotweld file and can be used to describe bolts and adhesive lines as well as spotwelds. Each connection entry consists of connection type, point (or line info) and a list of the layers to be connected. A layer is typically a part, but it may consist of multiple parts, see modifying connection layers.

Layers provide a powerful way of ensuring that correct connectivity in a model is achieved, as a connection will only be passed as valid (or realized) if it successfully joins all the layers in its definition.

For spotwelds, the file includes the spotweld part ID, the FE type (beam, solid or multiple solids) and the diameter (for solid welds). For bolts, FE type is defined as NRB or merge (for these an optional rigid material id may be given) and the diameter. For adhesives, adhesive width, number of elements across the width and element length are stored. Also for adhesive, the part ID of the solids in the connection are stored, along with additional information for the path of the adhesive.

Prior to read, PRIMER will scan the file to check if any required information is missing (e.g. part ID or diameter), and you will be able to supply this to the edit panel. Such information will **only** be used for connections which have parameters missing in the xml data. See Choosing part for beams/solids and filename for more details.

On completion of read, if any connections have not been made they will be put on the connection table for you to investigate and fix. For more details see section 6.10.2.

For models with existing connections, the xml connection file may be written out from the connection table using the "Update & write to file" function.

For more information on the format see the Spotweld file formats section.

Catia spotweld file

PRIMER will read the catia spotweld file automatically and go straight to step 7 at the end of the reading process. The format of the file is set within Primer and therefore it is important to check that the Catia weld file matches the catia format set by PRIMER.

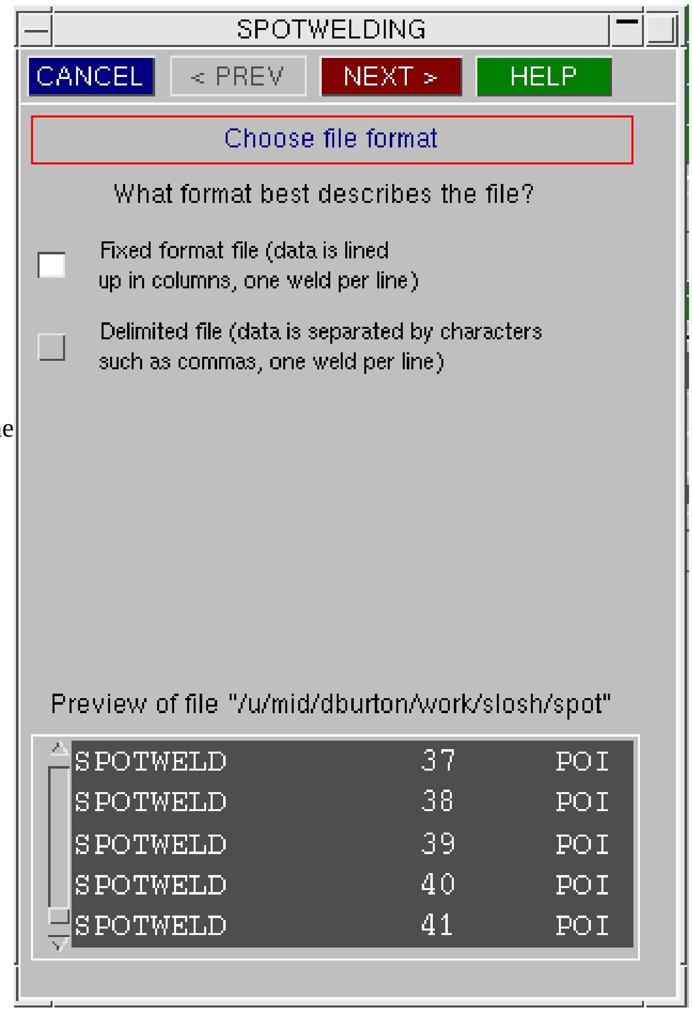
UG spotweld file

PRIMER will read the UG spotweld file automatically and go straight to step 7 at the end of the reading process. The format of the file is set within Primer and therefore it is important to check that the UG weld file matches the UG format set by PRIMER.

Custom format

If the file is a custom format, PRIMER will ask you a series of questions to determine the format of the file. Once the format has been determined you will be able to read the file.

Step 1: Fixed/Delimited



The first step is to determine the format of the file. PRIMER will try to read 2 types of files:

Files that have fields of fixed widths (these are like the fields in LS-DYNA keyword files that are generally 10 characters wide).

Files that have fields that are separated by a specific character such as a comma. An example of a file like this would be a CSV file produced by a spreadsheet program.

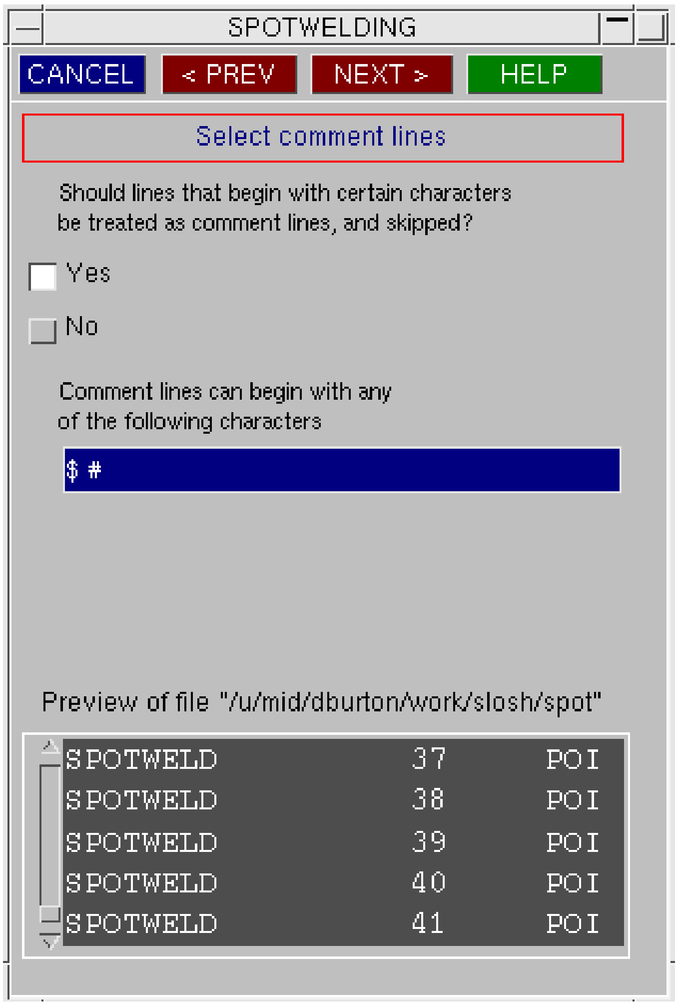
PRIMER shows a preview of the file at the bottom of the panel. You can use this to view the file and determine which of the 2 formats best describes the file.

Once you have chosen the format that best describes your file press **NEXT >** to go onto the next step. **CANCEL** will return you to the main screen.

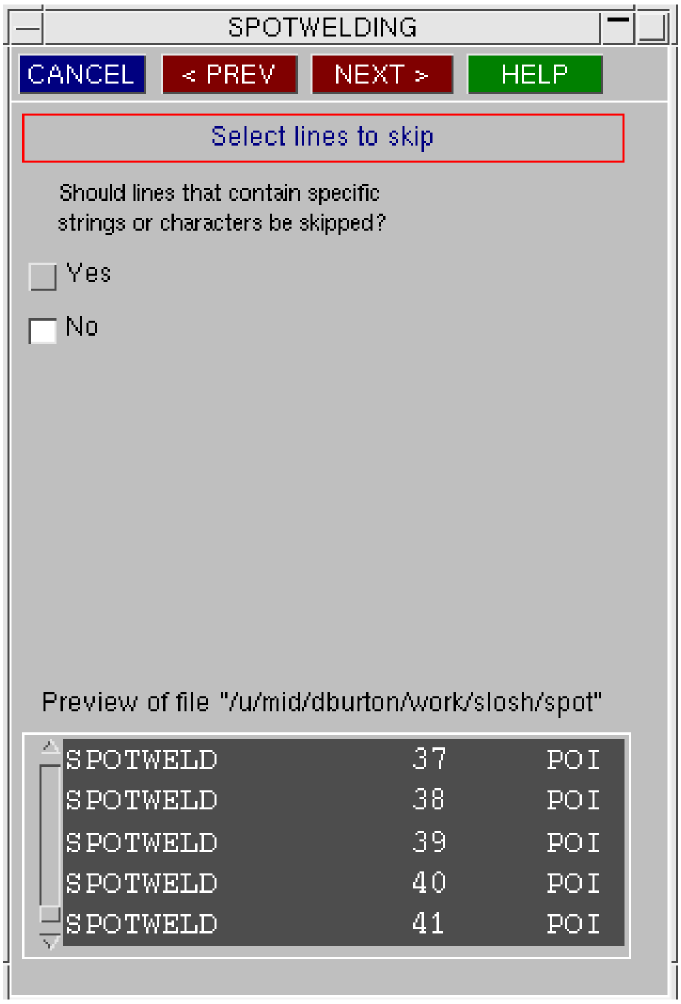
Step 2: Comment lines

The second step is to determine if any lines in the file should be treated as comment lines and skipped. This is like comment lines in a LS-DYNA keyword file that can begin with a '$' character.

Once you have chosen the comment setting press **NEXT >** to go onto the next step. To go back to the previous step press **< PREV**. **CANCEL** will return you to the main screen.



Step 3: Skip strings and characters



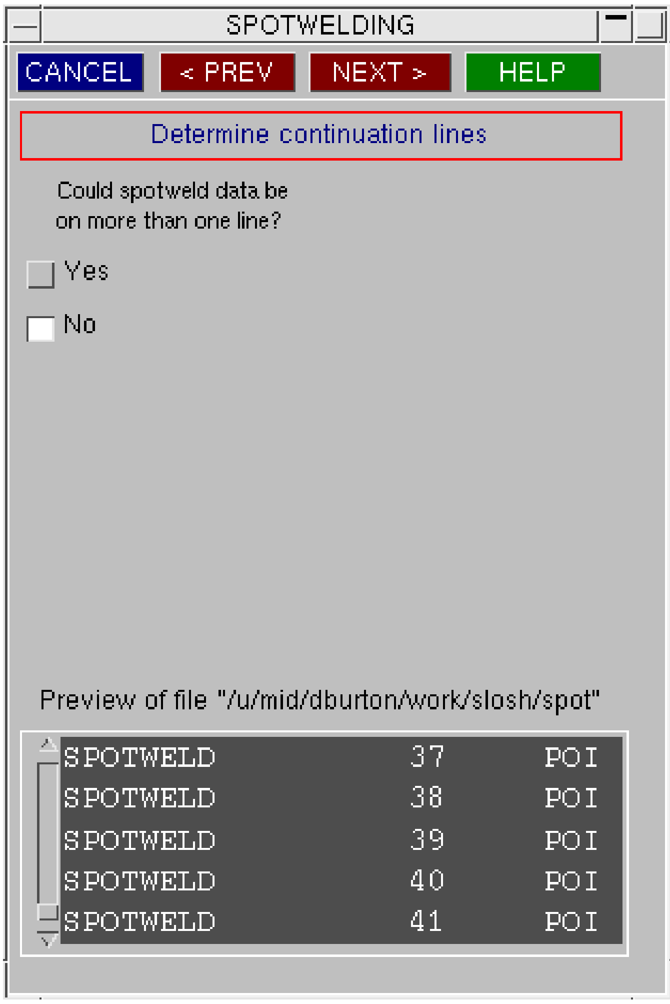
The third step is to determine if any lines in the file that contain specific strings or characters should be skipped.

Once you have chosen the string and character settings press **NEXT >** to go onto the next step. To go back to the previous step press **< PREV**. **CANCEL** will return you to the main screen.

Step 4: Continuation lines

The fourth step is to determine if spotweld data can continue onto a second line. It is strongly recommended that you have one line per spotweld. However, if spotweld data can continue on to a second line PRIMER will try to read it with these settings.

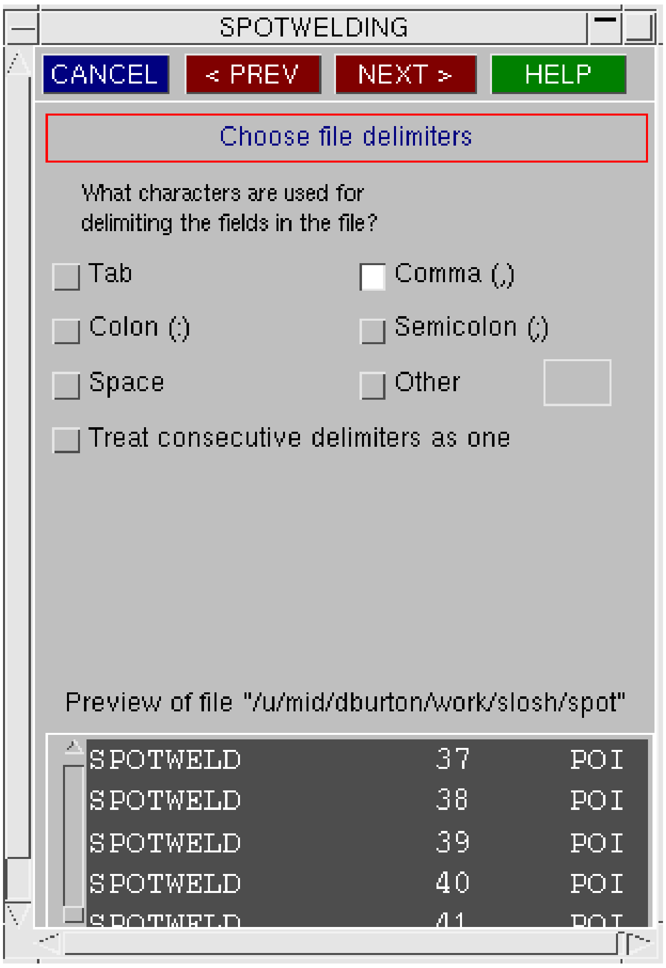
Once you have chosen the continuation setting press **NEXT >** to go onto the next step. To go back to the previous step press **< PREV**. **CANCEL** will return you to the main screen.



Step 5: Choosing delimiters

The fifth step is only done if you are reading a file in delimited format. You need to tell PRIMER what character(s) to use as field delimiters. Additionally there is a switch to **treat consecutive delimiters as one** delimiter. This is most commonly used when the 'space' character is used as the field delimiter. If some of the fields are separated by more than one 'space' then PRIMER will treat it as a single 'space'.

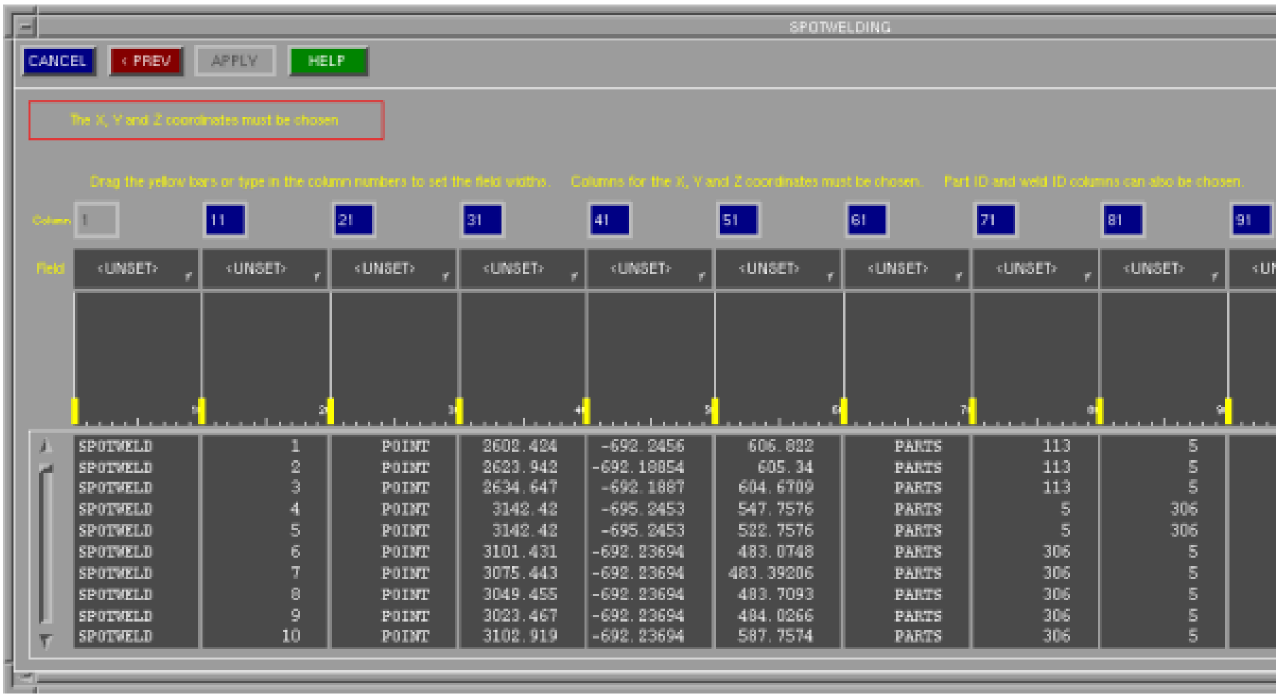
Once you have chosen the delimiter setting press **NEXT >** to go onto the next step. To go back to the previous step press **< PREV**. **CANCEL** will return you to the main screen.



Step 6: Choosing fields

The sixth step allows you to choose which fields are which. PRIMER shows a preview (the first 50 lines) of the file showing how it will decode the fields from the settings you have chosen in the previous steps.

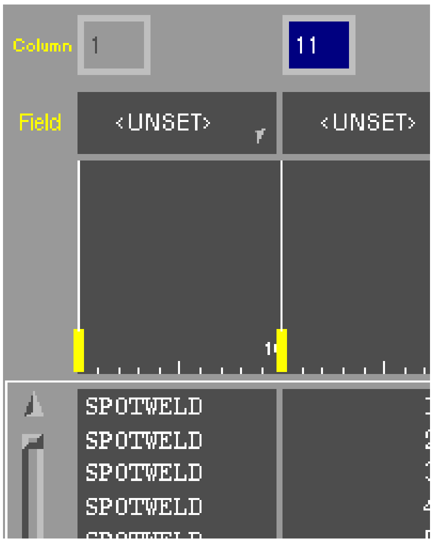
Choosing field types



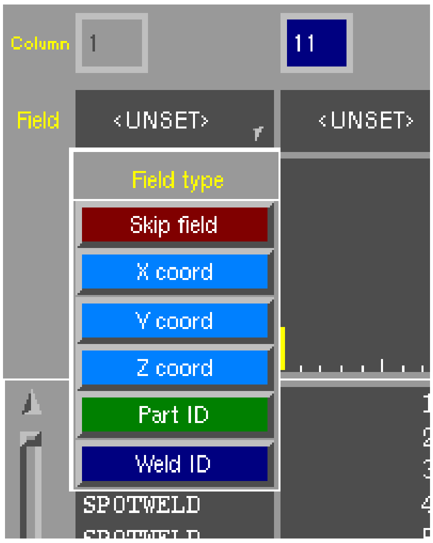
The image above shows the fields that PRIMER has read. If it is incorrect you can go back and change the settings as necessary. In this example the fields are:

Field	columns	description
1	1-10	Skip this text
2	11-20	Weld ID
3	21-30	Skip this text
4	31-40	X coordinate
5	41-50	Y coordinate
6	51-60	Z coordinate
7	61-70	Skip this text
8	71-80	Panel ID 1
9	81-90	Panel ID 2

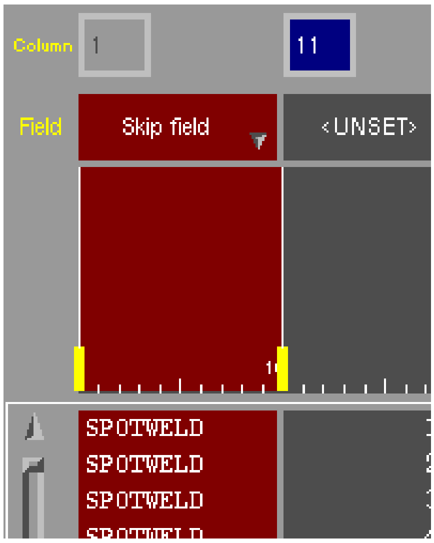
Initially all the fields are **<UNSET>**. Use the popup to change the field to the required type. For example to change field 1 to ’skip this field’:



Field is initially unset

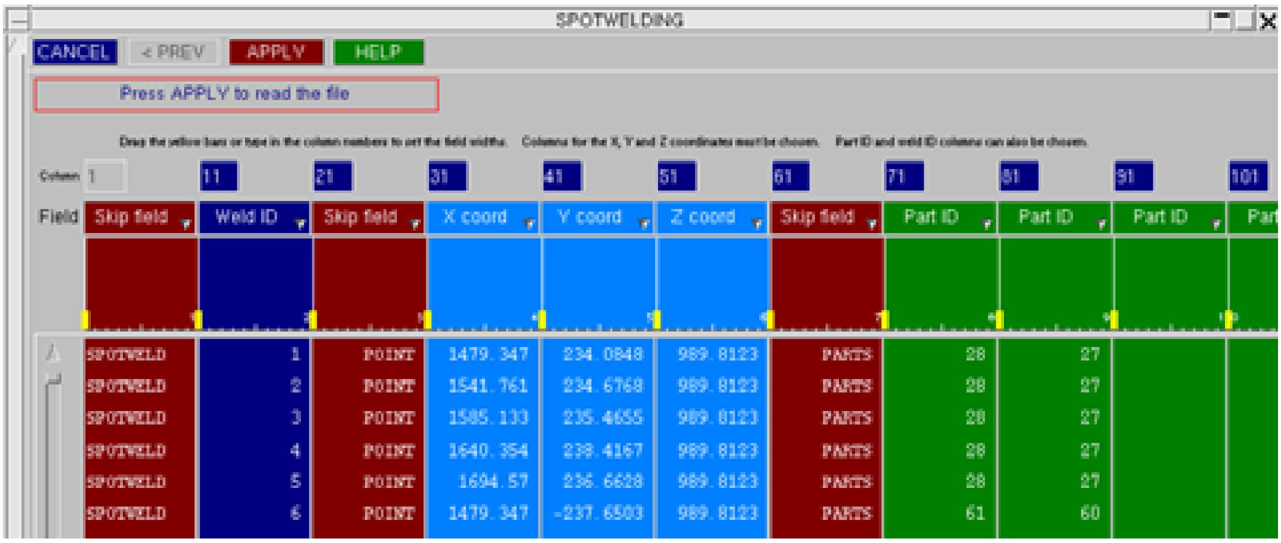


Use the popup and select **Skip field**



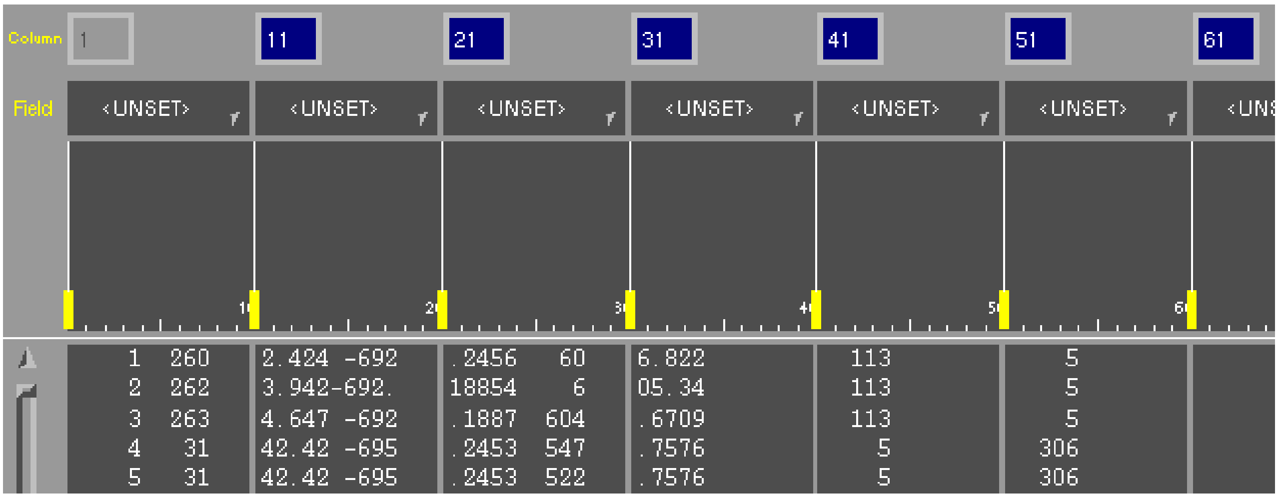
Field is now set to **Skip field** and coloured to show it is set

Repeat this until all the fields have been set to the required values. You **MUST** define the **X coord**, **Y coord**, **Z coord** and **Part ID**s.

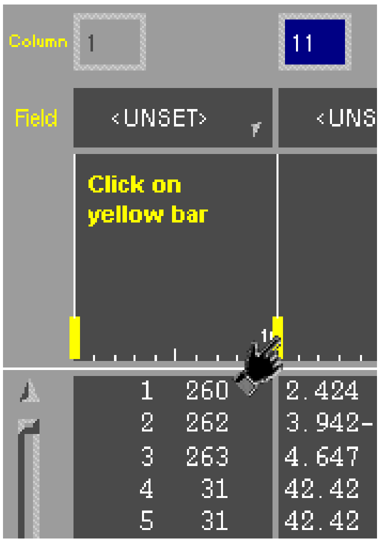


Choosing field widths

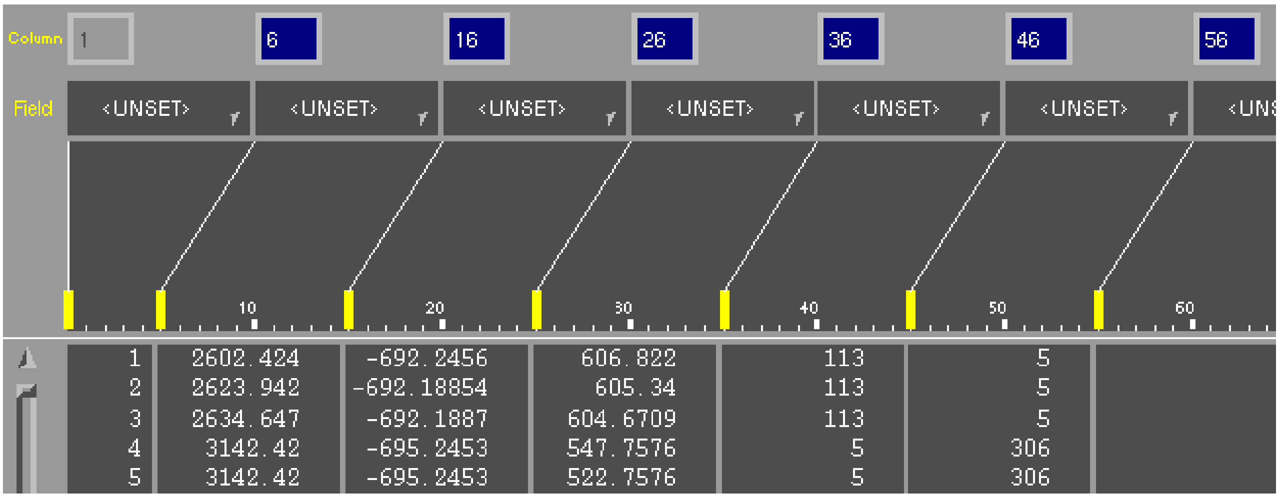
In the example image below the fields are not just 10 columns wide. We need to change the column widths.



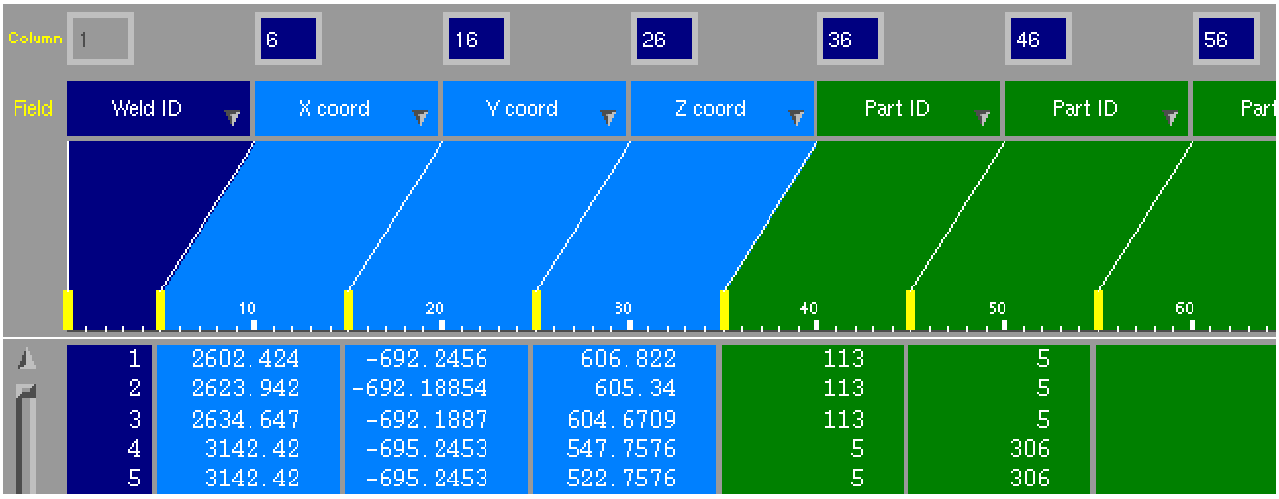
Field 1 should be columns 1-5, not 1-10. To change this you can either type in the new column numbers in the blue boxes or you can drag the columns to the correct sizes. The yellow bars enable you to drag the columns by clicking on one of them with the mouse and dragging it to the left or right until it is in the correct place.



Repeat this process until all the fields are the correct width



The field types can then be chosen as necessary.



Step 7: Warnings and errors after reading the file

After PRIMER has finished reading the spotweld file it will display a listing panel giving information on the welds it has not been able to create.

PRIMER will also do a check of all the welds that it has created to see if any are too close together (the pitch between the welds is too small).

Step 8: Fixing bad welds

If any of the welds in the file could not be created PRIMER will put them onto the connection table and you can use it to visualise and fix the welds. For more details see section 6.10.2.

6.10.8 Writing spotwelds to file

This panel allows you to write mesh independent spotwelds to a PRIMER spotweld file, an XML connection file, a UG weld file or an IGES file. Select which file type using the lower radio button selection.

The XML file also supports the export of bolts and adhesive connections.

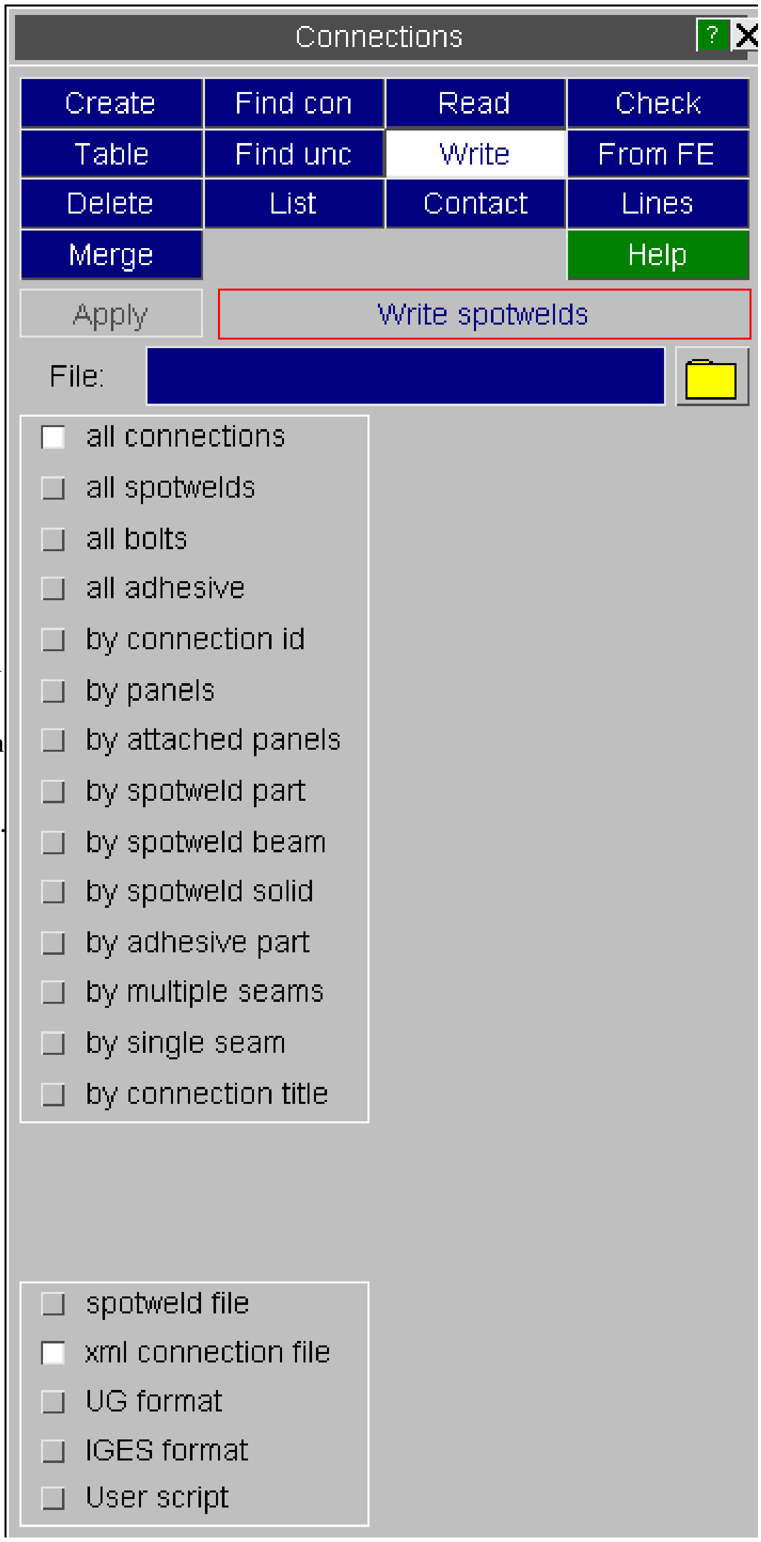
When choosing the IGES option, adhesive connections are written as IGES lines. All other connection types are written as IGES points.

You can specify which connections to write by a number of different methods. For more details of the different methods see section 6.10.0.

Type in the name of the file you want to write or use the file selector button.

Once you have specified the filename, which connections you want to write and the file format press the **Apply** button to write the file.

A script can also be used to write connection data out in a user defined format. The user would create a javascript that would write connection data in the format desired, and then this script can be used through the connection write panel to write the connections in that format. See section 10 for more on PRIMER's scripting functionality.



6.10.9 Connection contact

Mesh independent spotweld beams/solids and adhesive solids (connections in Primer) are tied to their respective panel shells using tied contacts in LS-DYNA. For solids the preferred contact is *CONTACT_TIED_NODES_TO_SURFACE, for beams it is *CONTACT_SPOTWELD. These constrained contacts give the correct shear stiffness for a weld connection. Penalty (_OFFSET) contacts will generally not give adequate stiffness - vehicle models can show 10% underestimate of torsional stiffness when penalty contact is used for all spotwelds.

Modelling constrained contacts: Constrained contacts require rigorously correct modelling as they are incompatible with other forms of constraint and may interfer with one another. For example, if a *CONSTRAINED_NODAL_RIGID_BODY attaches to a node of a shell to which a spotweld beam is attached by *CONTACT_SPOTWELD, the spotweld will be released. Similarly, if a piece of foam is tied to a panel by a constrained tied contact, spotwelds cannot be attached to the same shells by a *CONTACT_SPOTWELD. The best solution to this problem is to transfer the nodes which will not tie out of the constrained contact to an _OFFSET contact.

Primer's **CONNECTION > CONTACT** function has been designed to create spotweld/adhesive contact(s) with automated deployment of _OFFSET contact where necessary.

6.10.9.1 Checking the connection contact

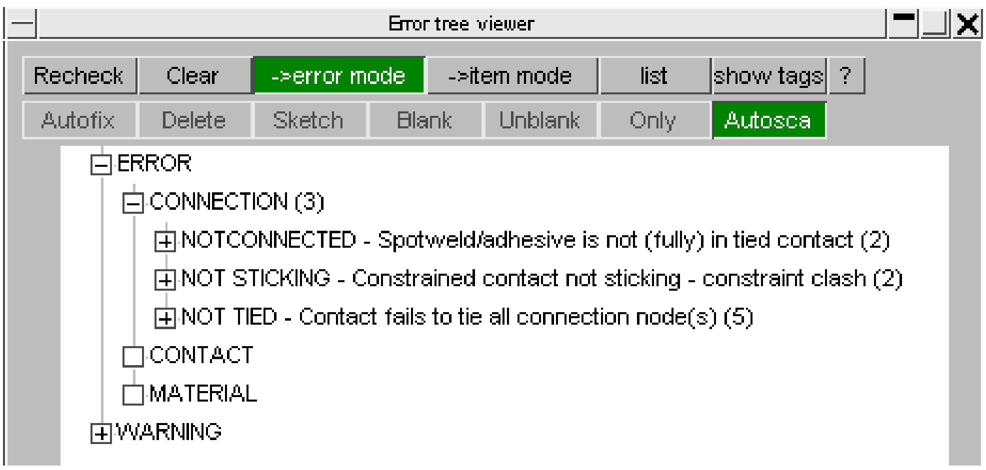
Connection contact is checked in a variety of contexts.

- when you run a model check - to report connections in error
- whenever connections are put onto the table - so their status is reported correctly
- using **CONNECTION > CHECK > CONNECTIVITY**
- using **CONNECTION > CONTACT**

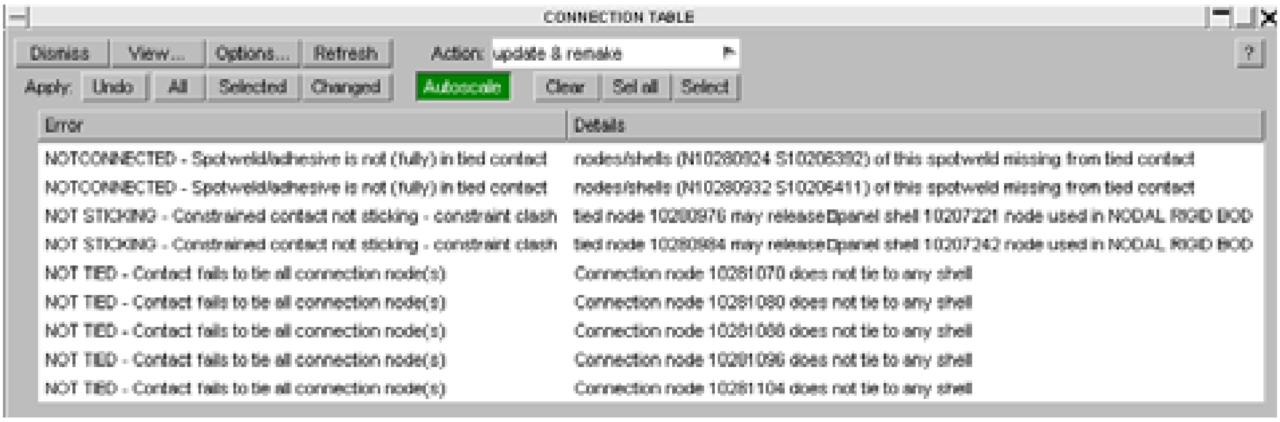
All use the same checking function and will report error code for each connection such as

- NOTCONNECTED - nodes or shells are missing from the contact definition
- NOT TIED - node and shell are in contact but fail to tie, usually because they are too far away
- NOT STICKING - constraint clash prevents contact from working
- BAD CONTACT - contact type is invalid

The results of model check are displayed in a tree view. The drop-down allows easy transfer of connections to the table, e.g. for re-making.



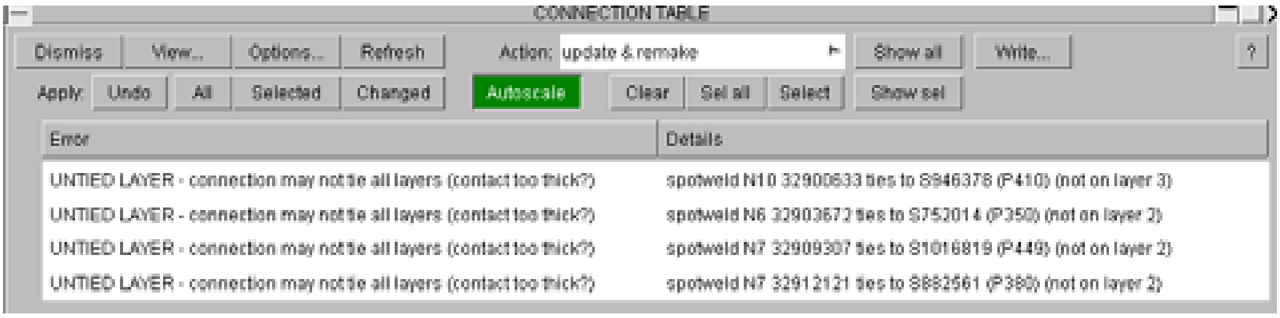
Details on the table provides useful information about the error.



UNTIED LAYER Connections tie but to wrong layer

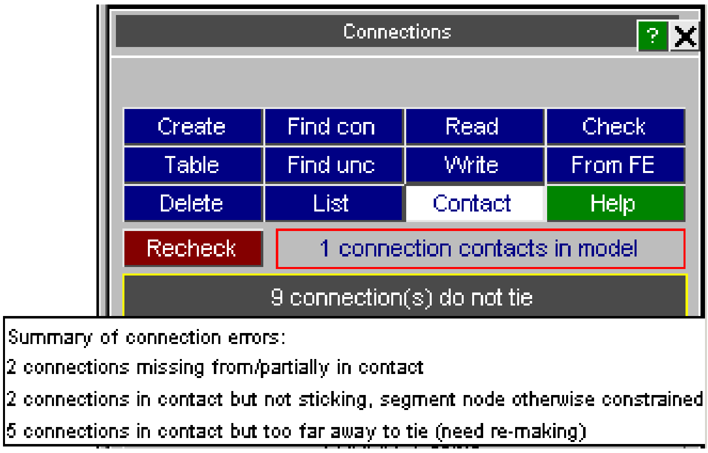
This error appears to be unfixable. Spotwelds are correctly projected, nothing is missing from the contact and all the nodes and their corresponding shells are tied.

It has actually resulted from the fact that the tied contact is defined with an excessive thickness wrt the shells being tied. This means that it is ambiguous to which shell a node is tied and Primer suspects that the actual shell tied will not be on the part in the layer definition. This error will not arise if the contact thickness is consistent with the shell thickness.

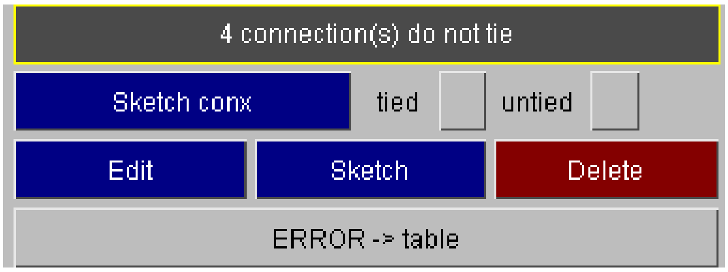


6.10.9.2 Using Connection > Contact

This function will report a summary of errors in hover text activated when the mouse is over the dark grey button.

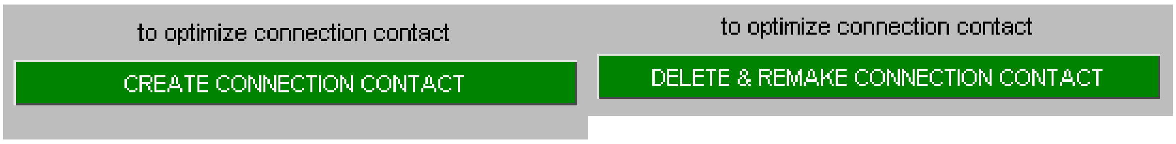


Tied and untied connections may be sketched with **Sketch Conx** and contacts used for connections may be accessed directly by **Edit Contact**. Problematic connections may easily be put on the table using **Error->table**, where the **show conx & panels** feature will enable you to display the area of interest. **Delete** may be used to remove contacts selected on the object menu.



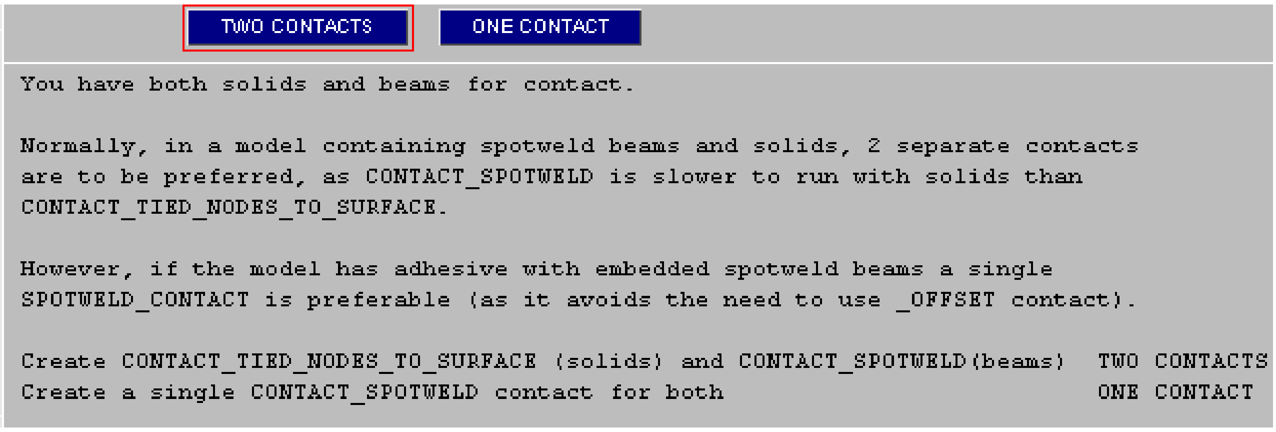
6.10.9.3 Connection > Contact: fully automatic fix

If there is no connection contact **CREATE CONNECTION CONTACT** will make one. If contacts already exists, the sister function **DELETE & REMAKE CONNECTION CONTACT** will delete the existing connection contacts and re-make them. These functions work on all connections and will yield an optimum solution with the minimum number of connection contacts. In a model with multiple contacts you may prefer to fix a sub-set of connections. To this end the functions **NOT-CONNECTED->contact** and **NOT-STICKING->offset** contact are available.



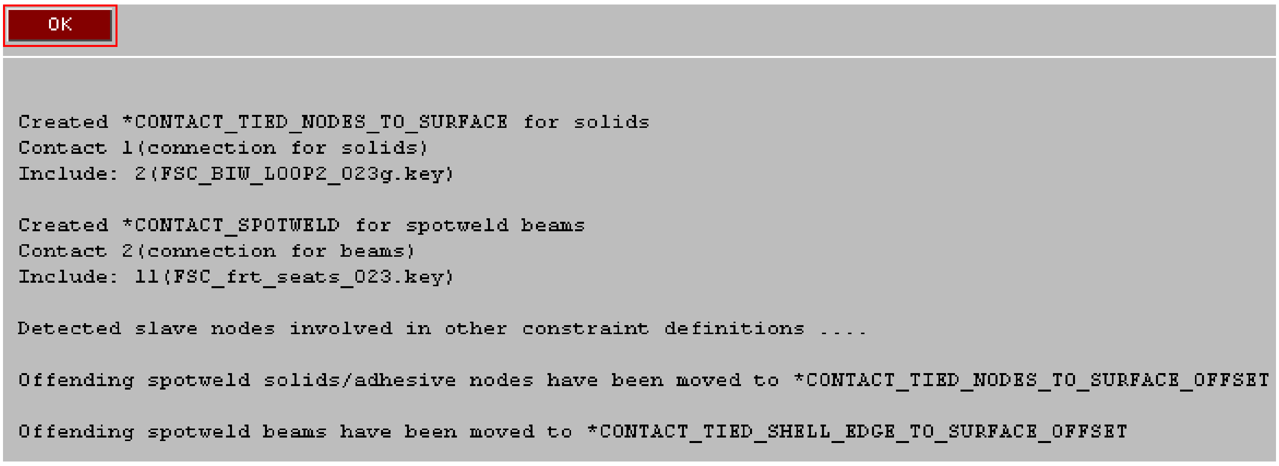
Models with beams and solids: A model with both spotweld beams and spotweld/adhesive solids will normally require separate contacts as CONTACT_TIED_XXX_TO_SURFACE cannot be used for beams (as it has no rotational constraint) and CONTACT_SPOTWELD is inefficient for tying solid elements.

However, if spotweld beams are embedded in a solid adhesive one may prefer to use a single CONTACT_SPOTWELD rather than have to create an _OFFSET contact (to avoid constraint clash). Therefore Primer will offer the choice.

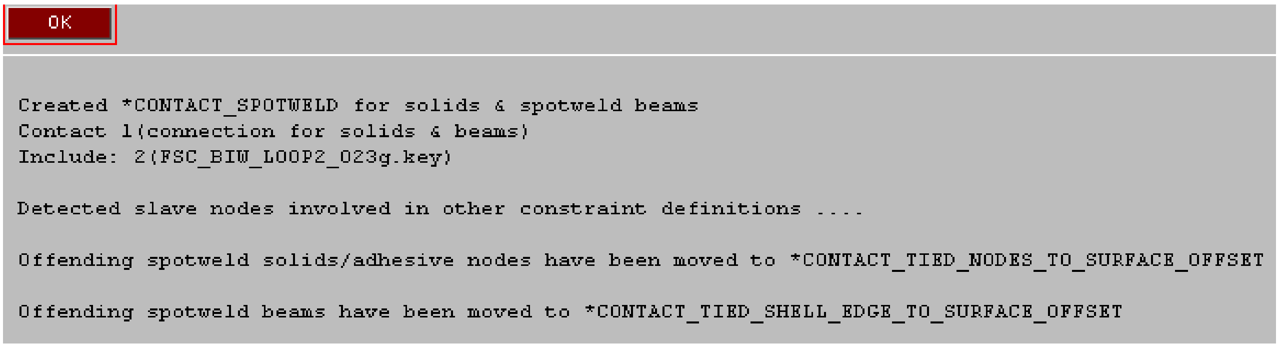


If we take the **TWO CONTACTS** option Primer will create both contacts with the slave side initially defined by PART or PART-SET. Primer will then automatically check for any constraint clashes. If detected, the slave side(s) will be converted to NODE-SET and the clashing nodes moved to appropriate _OFFSET contacts, defined using slave side node-set tied to master side shell-set.

This achieves full connectivity, with minimal use of penalty method which can be detrimental to contact stiffness. The model now contains 4 connection contacts.

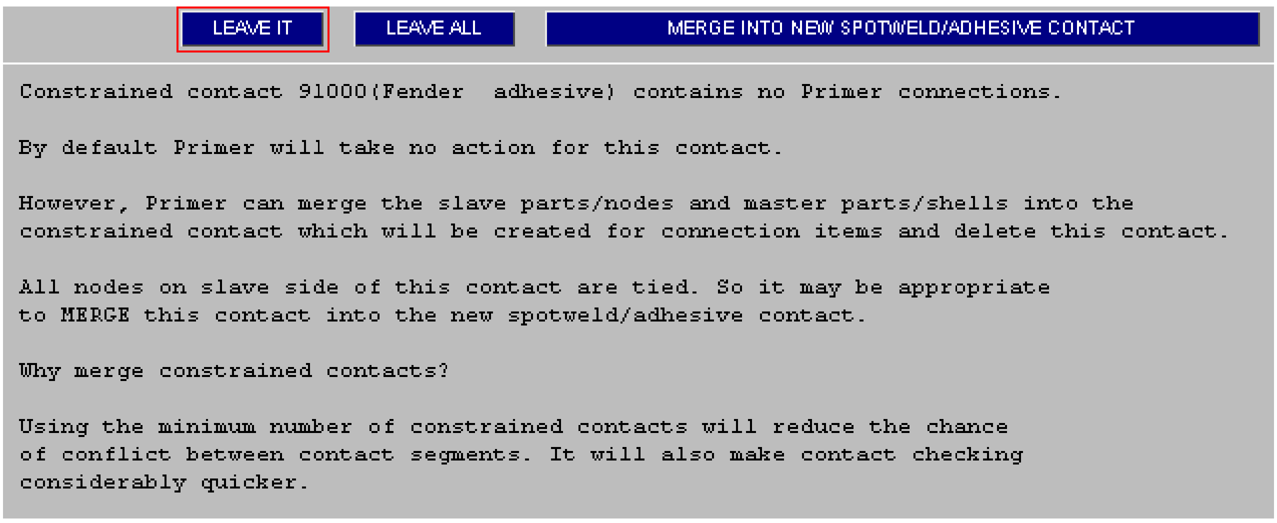


The **ONE CONTACT** option results in 3 connection contacts, as we still require 2 _OFFSET contacts.



Handling mixed contacts on re-make: Contacts may already exist which contain on the slave side both connection and non-connection items (e.g. solid spotwelds and solid foam). In this case, Primer will create connection contact as before, but in addition will add the non-connection nodes to the slave side, and their corresponding parts to the master side. Thus connectivity of the non-connection part should not be lost.

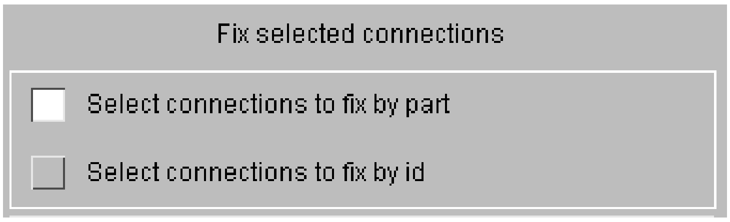
Handling constrained contacts with no connections: To avoid clashes it it advantageous to have the minimum number of constrained contacts in a model. If Primer finds, when creating or re-making connection contacts, that constrained contacts already exist in the model which have no connections on the slave side, you will be given the option to merge these into the connection contact to be created. This action is recommended if you intend all the slave nodes to tie. If you choose to **LEAVE** you may find that Primer has to convert more of the connection contact to _OFFSET.

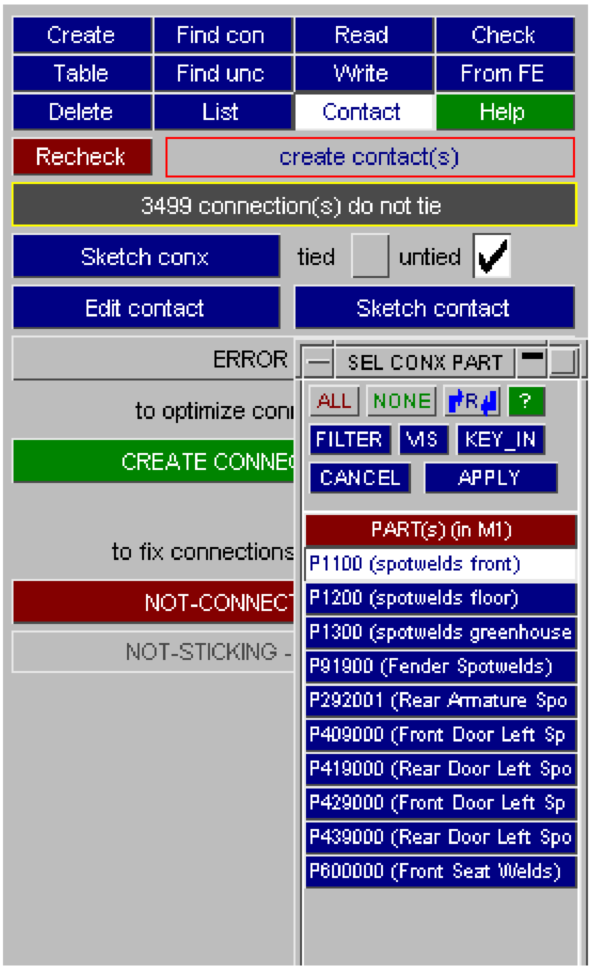


6.10.9.4 Connection > Contact: semi-automatic fix

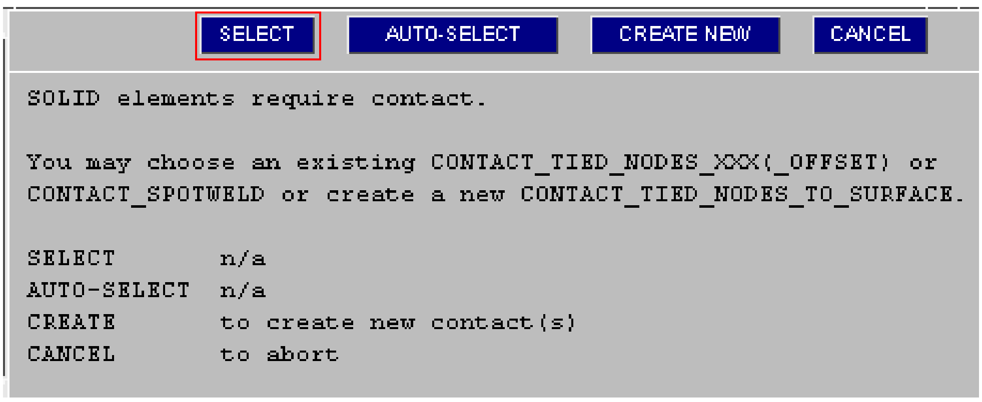
NOT-CONNECTED->contact. This function is designed to handle cases where the user does not want to re-make *all connection contacts* but prefers to fettle the individual contacts.

Connections to be fixed may be selected by part, which gives a chance of preserving definition by part on the slave side of the contact or by connections which will always convert the slave side to a node set.

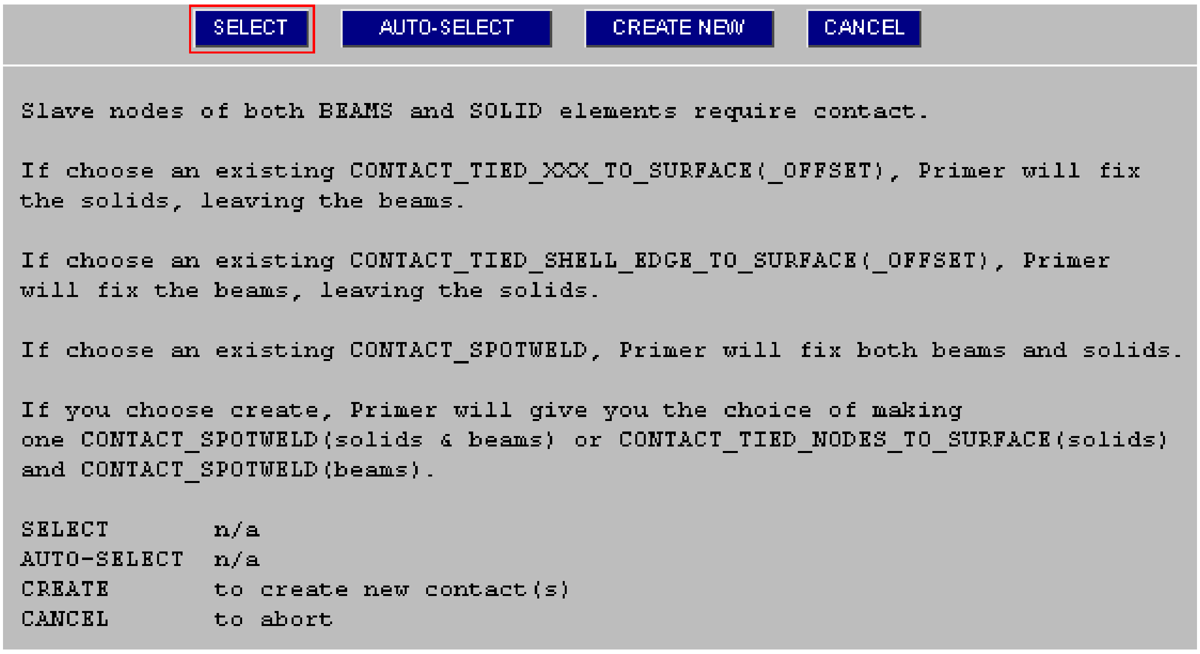




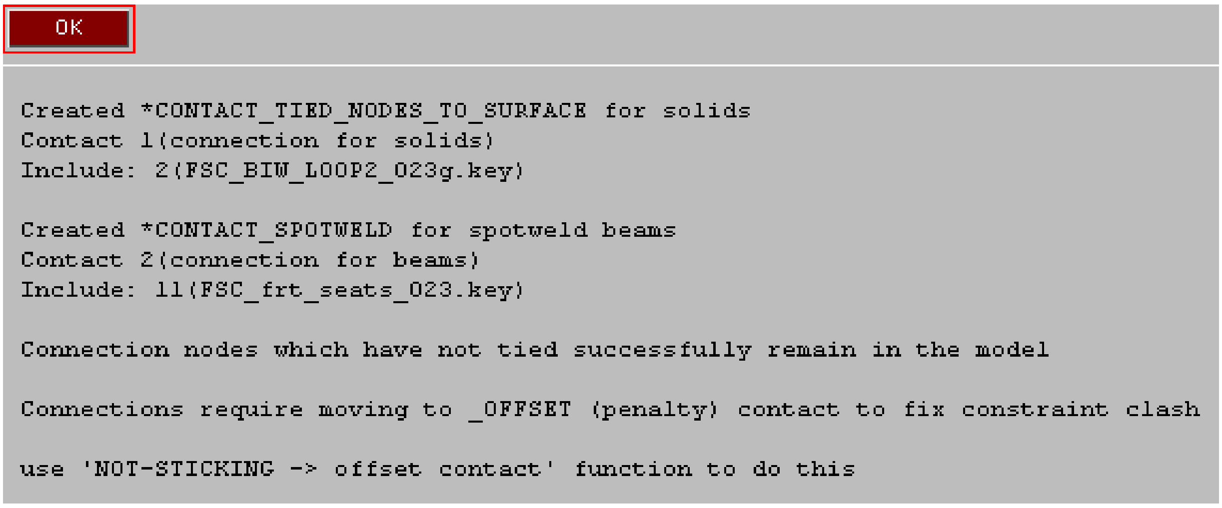
CREATE NEW will create a new constrained connection contact (by PART/PART-SET) of the appropriate type. You may also **SELECT** an existing contact and add the connections to it. Primer will refuse so to do, if the contact type is incompatible.

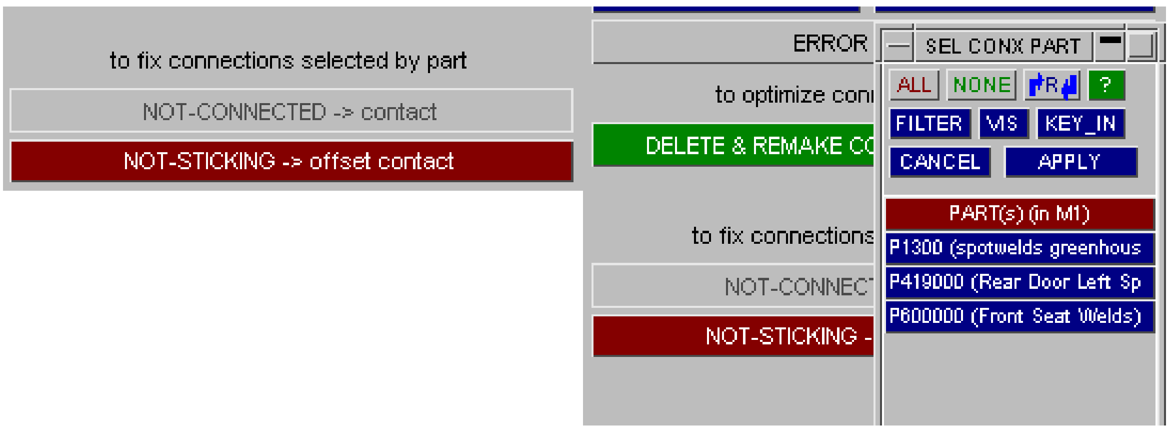


If both beam and solid parts have been selected for contact Primer's action will depend on the contact you select. The information on the panel will guide you.

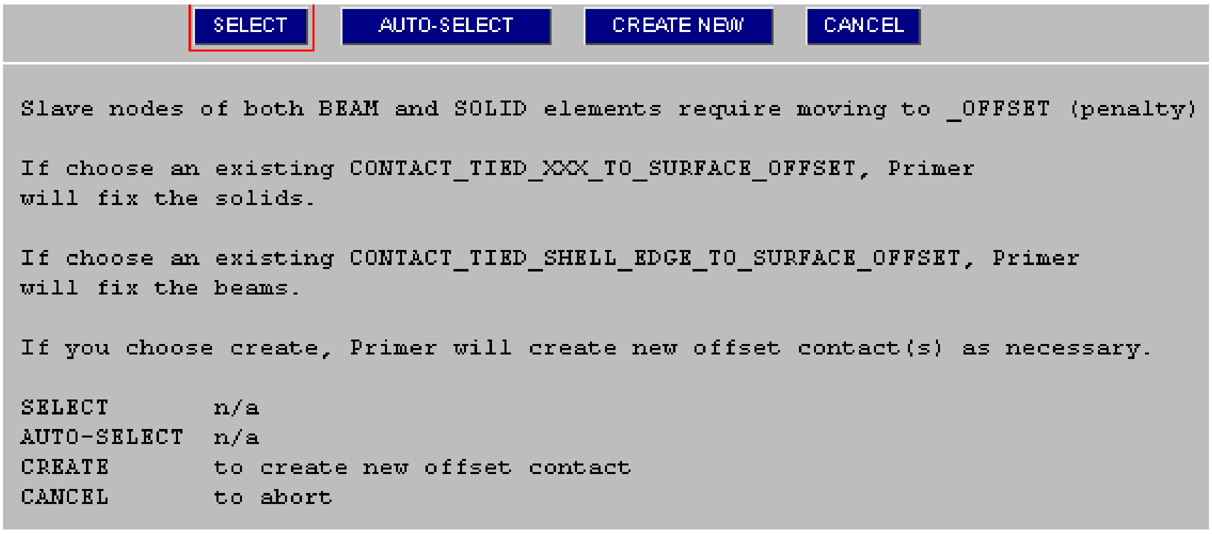


On creation of the contact, Primer will check for constraint clashes and invite you to fix them using **NOT-STICKING -> offset contact** function which will have become active.



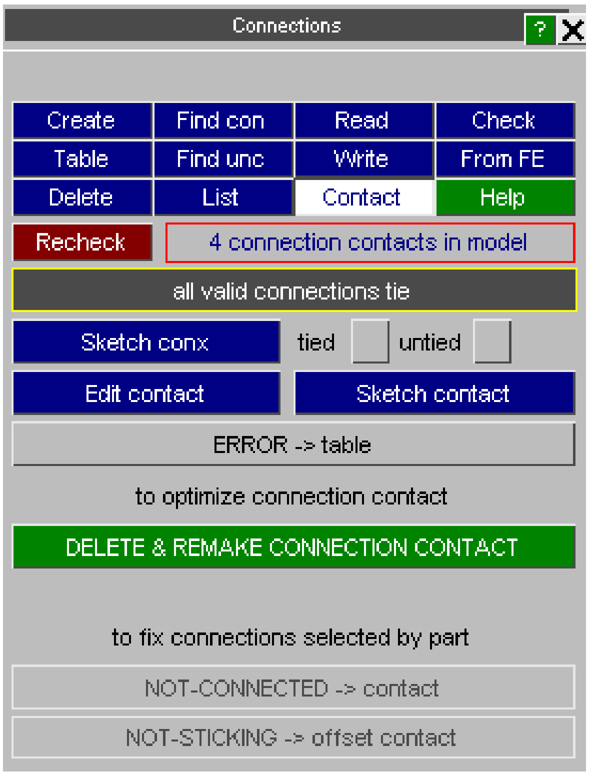


Connections may again be selected by part or by connection. Primer will move them from the constrained contact to the appropriate _OFFSET contact which can be selected or created.



On completion of the process, you should get the report that **all valid connections tie** and the semi-automatic connection fixing options will be greyed.

On the table all connections should have REALIZED status and there should be no error messages.



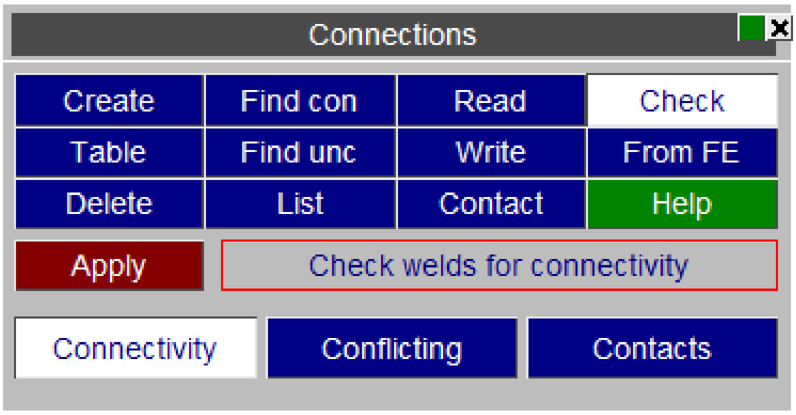
6.10.10 Checking connections

As well as checks under the normal model checking, there are three PRIMER functions that allow you to check properties of spotwelds.

Connectivity checks the quality of spotweld connectivity.
Conflicting, where PRIMER checks for the distances between welds and offers to delete welds that are too close together.
Contacts where PRIMER looks to see if the spotwelds are part of a tied contact in the model.

Checking connectivity

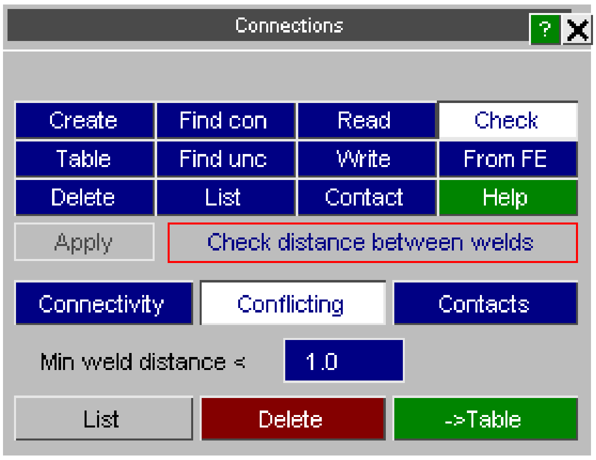
PRIMER will check all of the spotweld connections in the model to see if they are tied correctly. If any of the weldsare not PRIMER will put them onto the connection table and you can use it to visualise and fix the welds. For more details see section 6.10.2.



Checking conflicting welds

PRIMER will check to see if any spotweld connections in the model are too close to each other. Enter the **Min weld distance** and then you can either

- **List** - list all conflicting the welds
- **Delete** - delete a subset of welds to remove the conflict (same as model autofix)
- **->Table** - send all conflicting welds to table where you can delete them or merge them

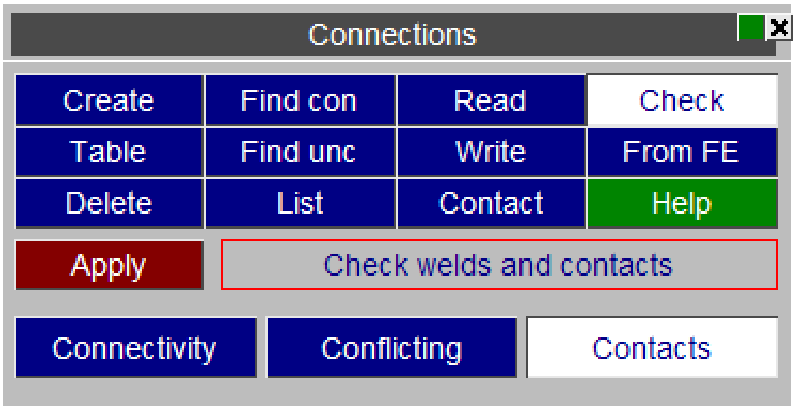


Checking weld contacts

PRIMER will check to see which spotweld connections are not in a tied contact.

Any beams, solids or parts that are not in a tied contact will be put into sets so that you can visualise them.

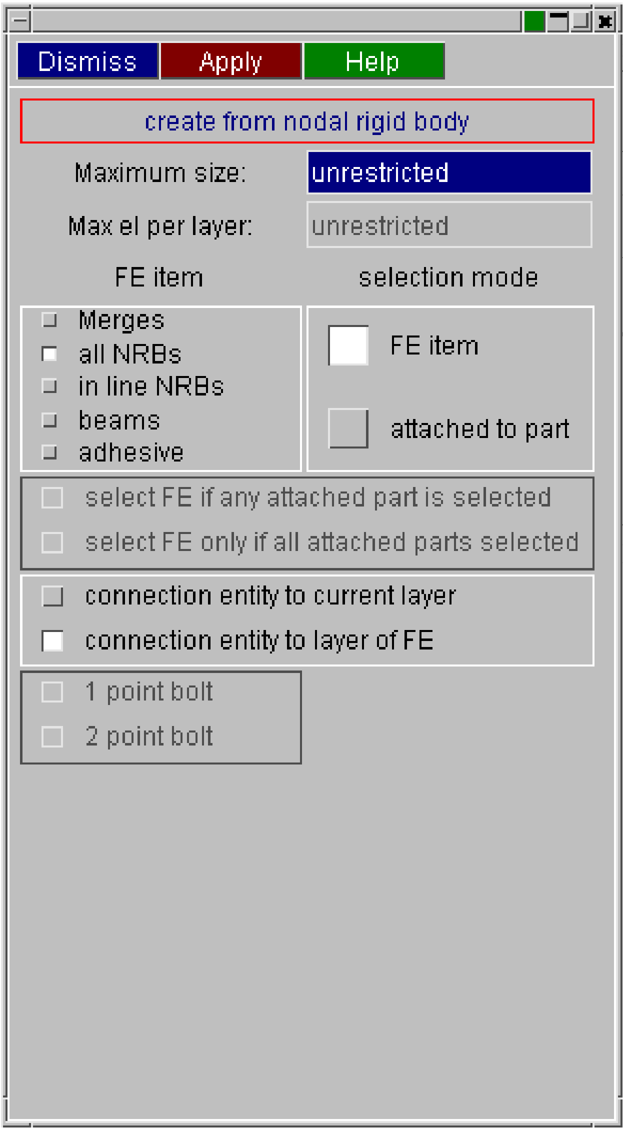
Note - this function will only work when you have a **single connection contact**. For a more generic treatment you should use the function under **CONNECTIONS > CONTACT**.



6.10.11 Creating bolt connections from FE data

PRIMER will automatically create connections for any existing beam spotwelds and solid spotwelds in your model. At present PRIMER will not make connections for ’nugget’ solid spotwelds (i.e. spotwelds that use multiple solids and a *DEFINE_HEX_SPOTWELD_ASSEMBLY card. Primer can also make connection entities for existing solid adhesive runs in the model which conform to "Primer like" adhesive connection entities.

For ’bolt’ type connections PRIMER cannot make the connections automatically. The **FROM FE** panel enables you to create them from selected Nodal Rigid Body or Constrained Rigid Body definitions.



Nodal rigid body or one (of a possible chain) of rigid body merges may be selected from the object menu in the usual way.

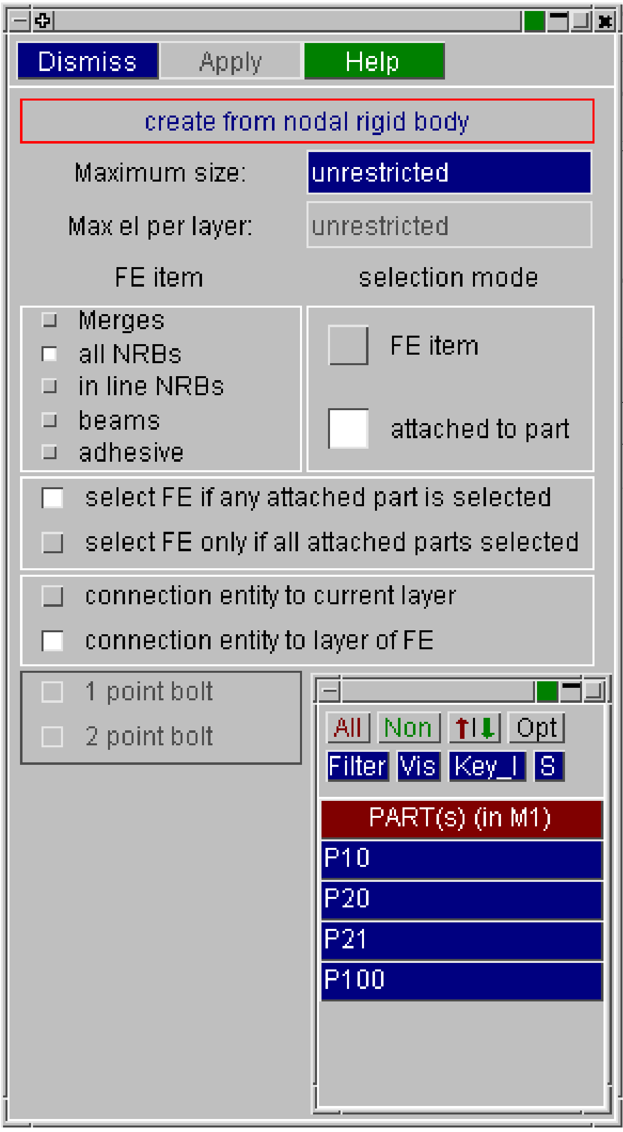
Beams may be selected to create connection beam bolts. In this mode, you may set Primer to create one point or two point connections.

For each item selected a calculation of dimension is made and user has the option to exclude those that exceed the maximum size (if specified).

The dimension is an approximate measure of the connection’s diameter.

line nrbs option will limit the object menu to offer only nodal rigid bodies which form a straight line

Where is the connection put? If the model contains include the default behaviour is to put the connection entity into the layer of the NRB or the C_RBOD from which it is formed.



The FE items may also be found by selecting by attached part.

The options are to select the item if it attaches to any or the parts selected or, more restrictively, to select only those items where all the attached parts are selected.

On **APPLY** connection with correct layer data will be created. The FE data itself *will not be changed.*

The user is recommended to invoke the newly made connections on the TABLE and check their diameter, as this may require adjustment.

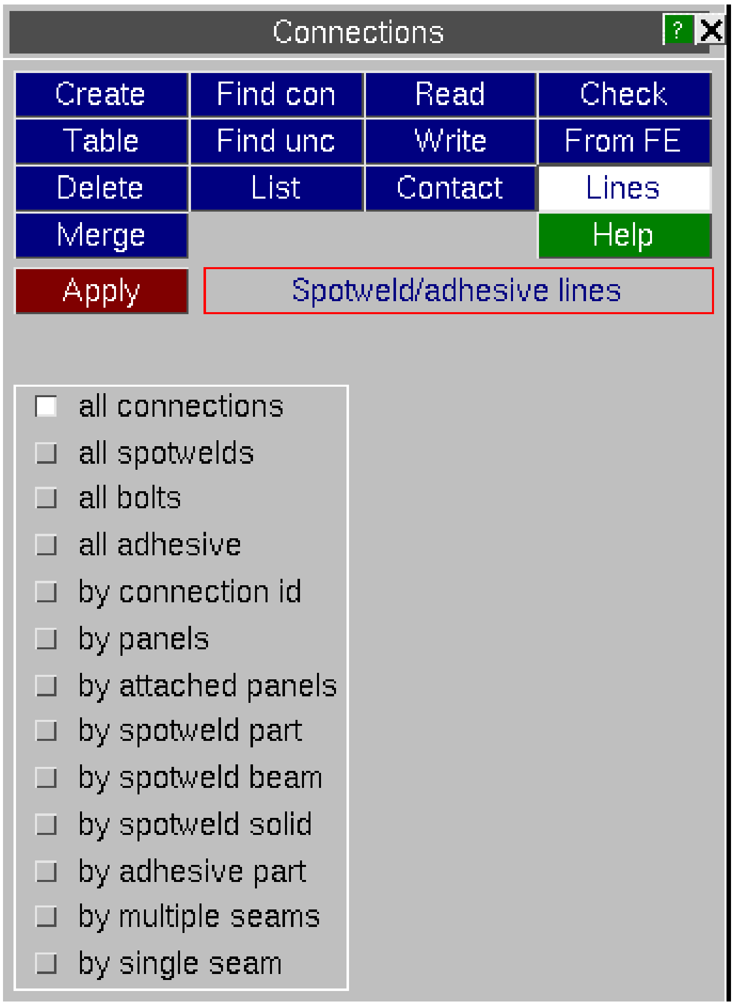
Note on remake of connections - the FE items may be changed slightly after remake. Merge type bolts will remake with a master part (itself containing no elements) and slave parts overlaying the panel, so an extra part has been created. NRB type bolts may remake at a slightly different size. In all cases, connectivity of the layer panels will be maintained.

For adhesive connections, you select the solid element runs you wish to create connection entities for. On **APPLY** Primer will create connection entities. The FE data itself *will not be changed.*

6.10.12 Modifying spotwelds and adhesives through lines

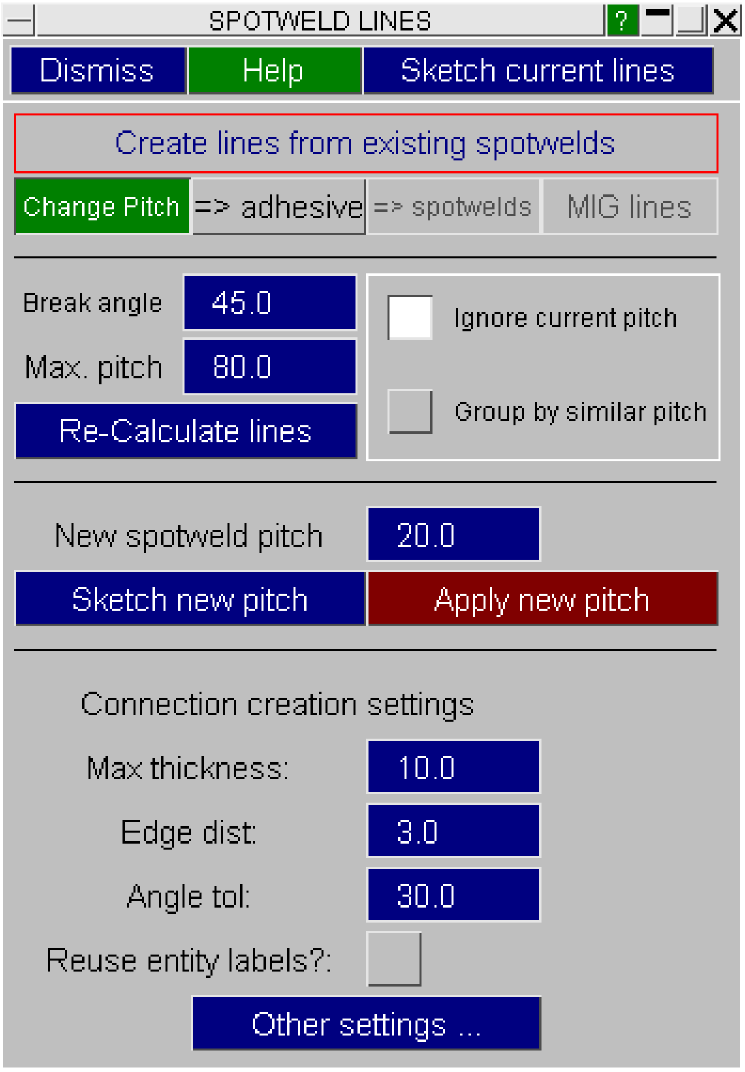
Spotweld connections exist as individual entities within PRIMER, but the **LINES** tool can be used to modify groups of spotwelds that form lines. A common use of this feature is to modify the pitch of a run of spotwelds. This tool can also be used to convert a line of spotwelds to a run of adhesive, or vice-versa

The spotweld lines feature can be accessed by selecting LINES in the connections panel. You can then use the standard methods for selection of connections. For more details of the different methods see section 6.10.0. Note that you can only modify spotwelds or adhesive, not spotwelds or adhesive together.

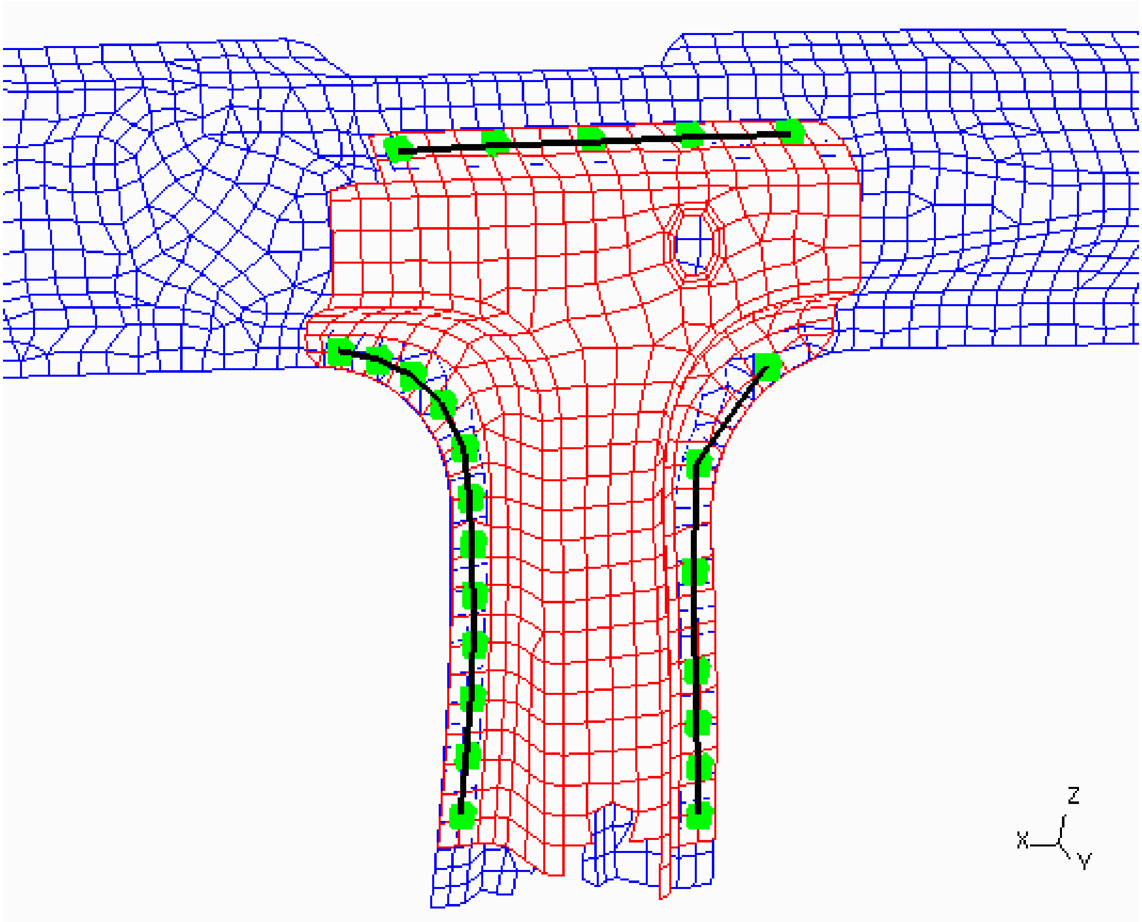


After selecting the connections you wish to modify, PRIMER will group the the spotweld lines panel will open up. The panel will look different depending whether you select spotwelds or adhesives.

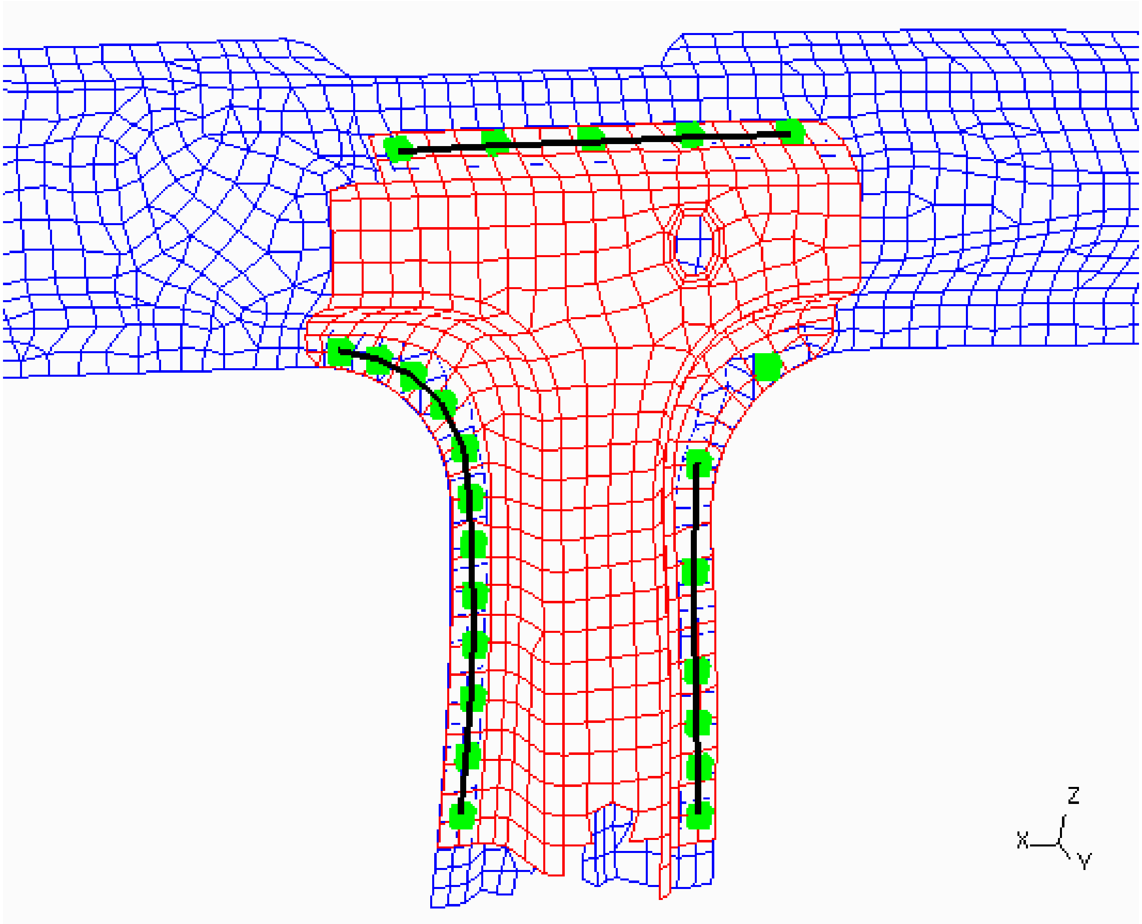
Modifying spotwelds



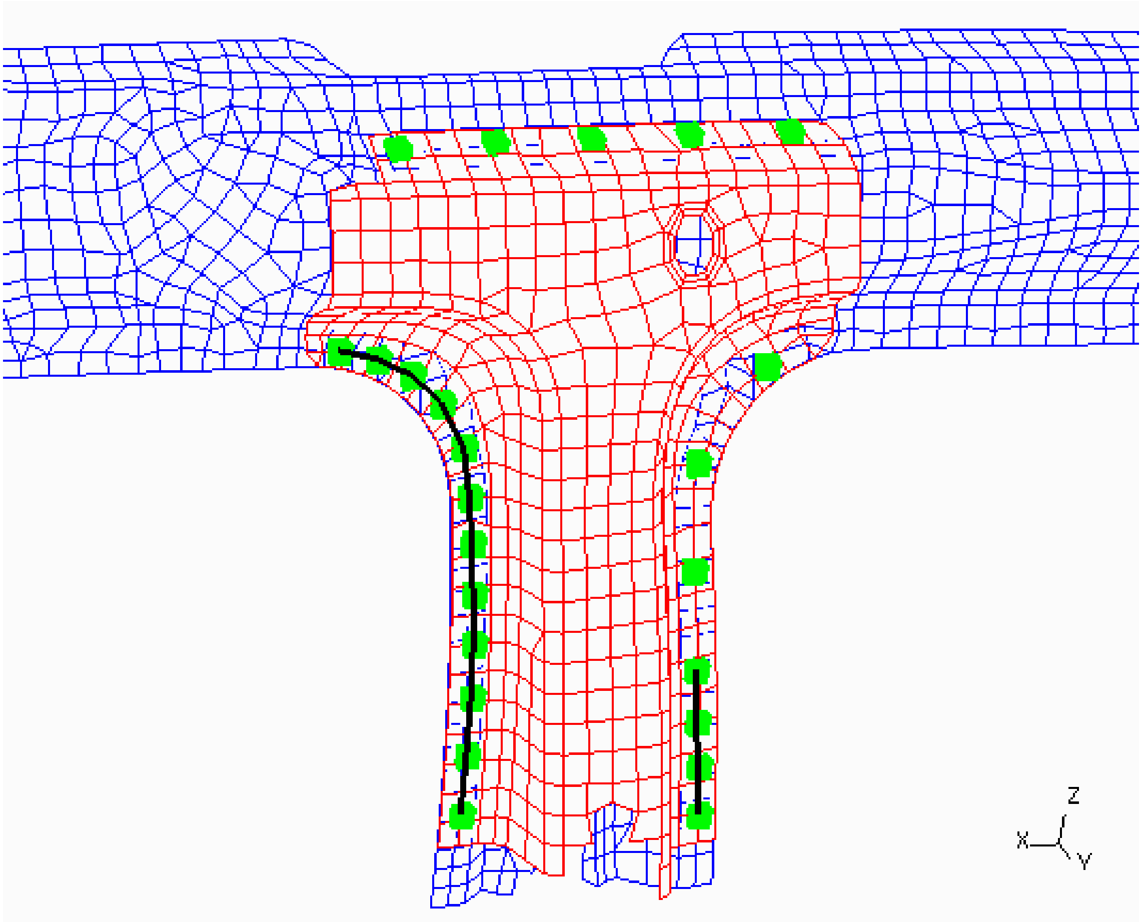
The above panel opens up when modifying spotwelds. When opening up the panel, PRIMER will group the spotweld connections in the model together into line groups, and sketch the lines on the screen.



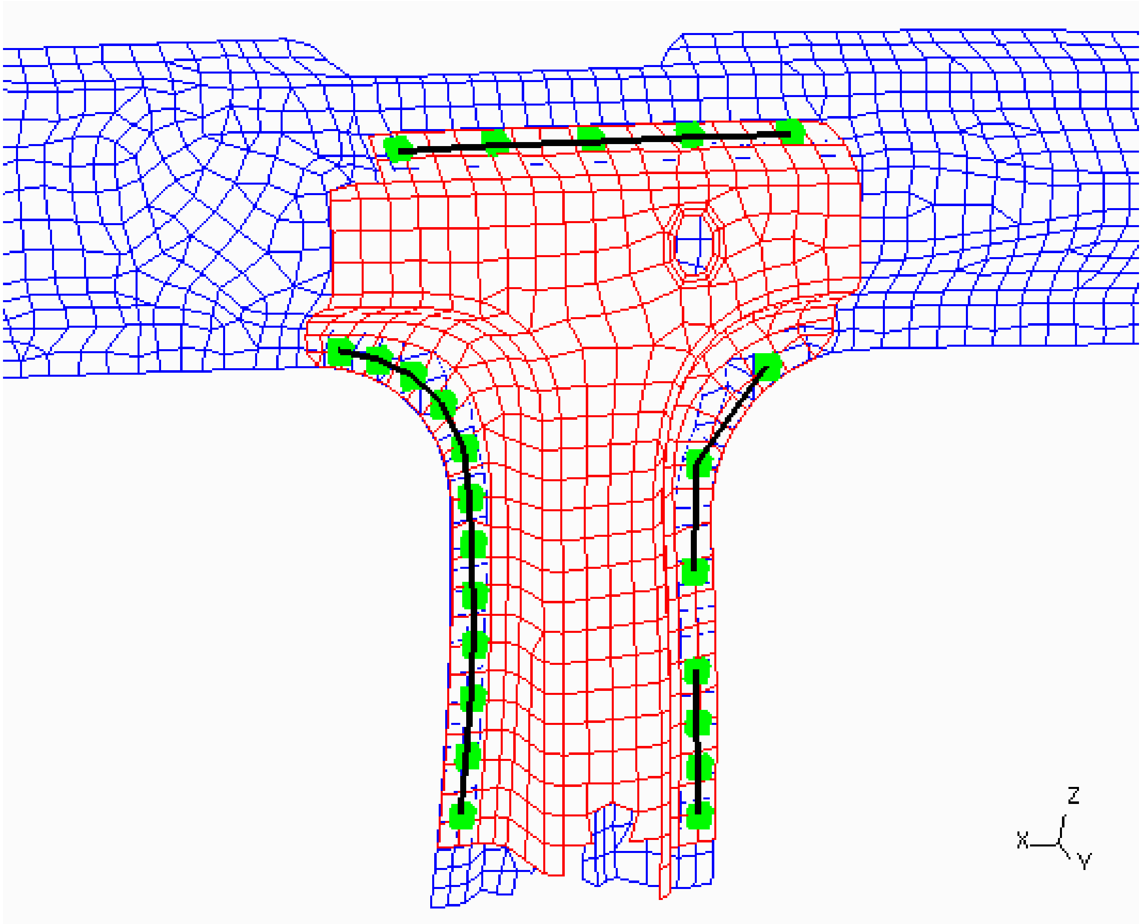
The default when opening the panel with spotwelds is **Change Pitch** mode, which can be used to modify the pitch of the selected spotwelds. At the top of the panel, there is a **Sketch current lines** button. This can be used to sketch the lines currently being used by the panel should you lose the original sketching (can occur if you redraw for example). The top part of the panel can be used to modify the inputs PRIMER uses to calculate the groupings of spotwelds to form lines. The **Break angle** value is used to separate the lines should the angle between two sections of the line by greater than this value. The following image shows the affect of changing the break angle from the default of 40.0 to 20.0.



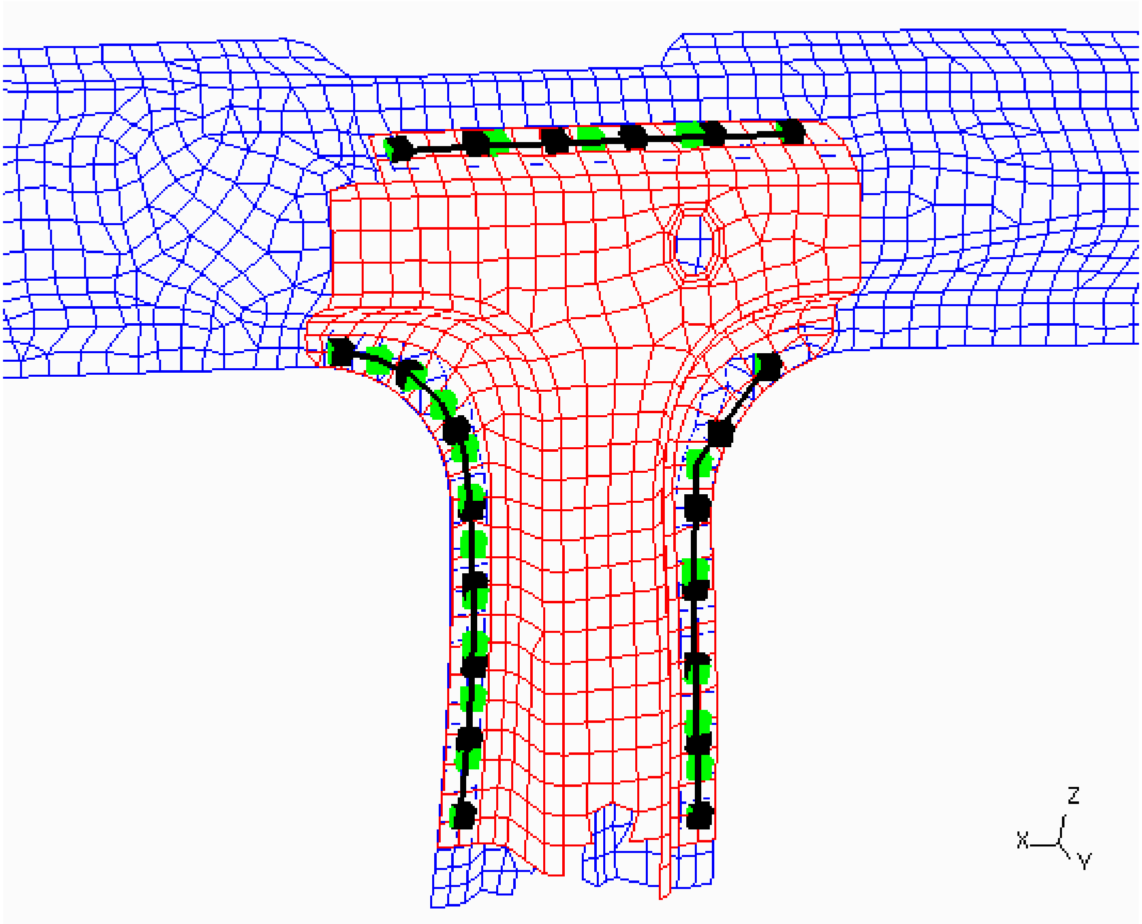
The **Max. pitch** value is the maximum allowed distance that two spotwelds can be apart and still considered to be part of the same line. The following image shows the affect of changing the maximum pitch from the default of 80.0 to 40.0.

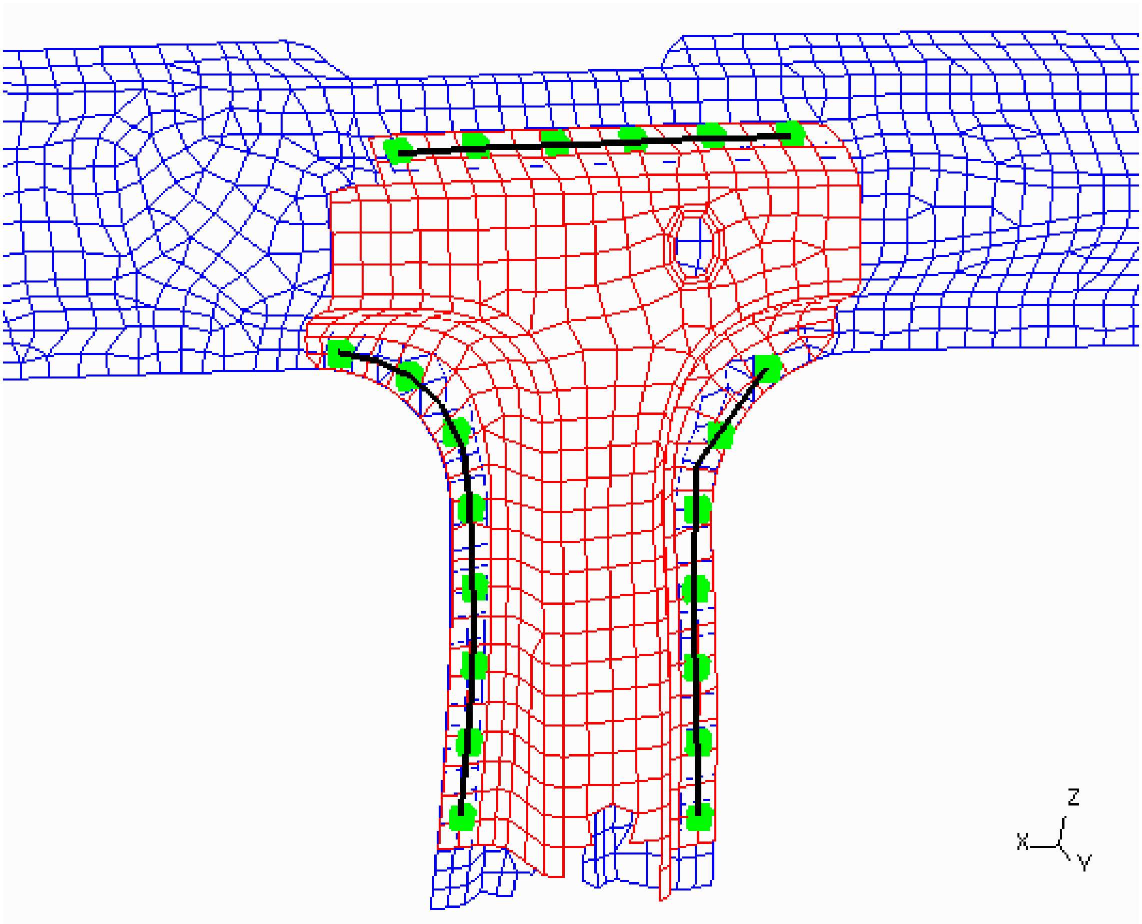


If **Group by similar pitch** is toggled, PRIMER will group together spotwelds that have a similar pitch (as, for example, spotwelds may be irregular along a flange and you may want these to be considered separate lines). The default of **Ignore current pitch** will not group spotwelds according to this rule. The following image shows the affect on the example of toggling **Group by similar pitch**. Note the line on the lower right flange is now split into two, and the spotweld pitch is inconsistent along the flange.



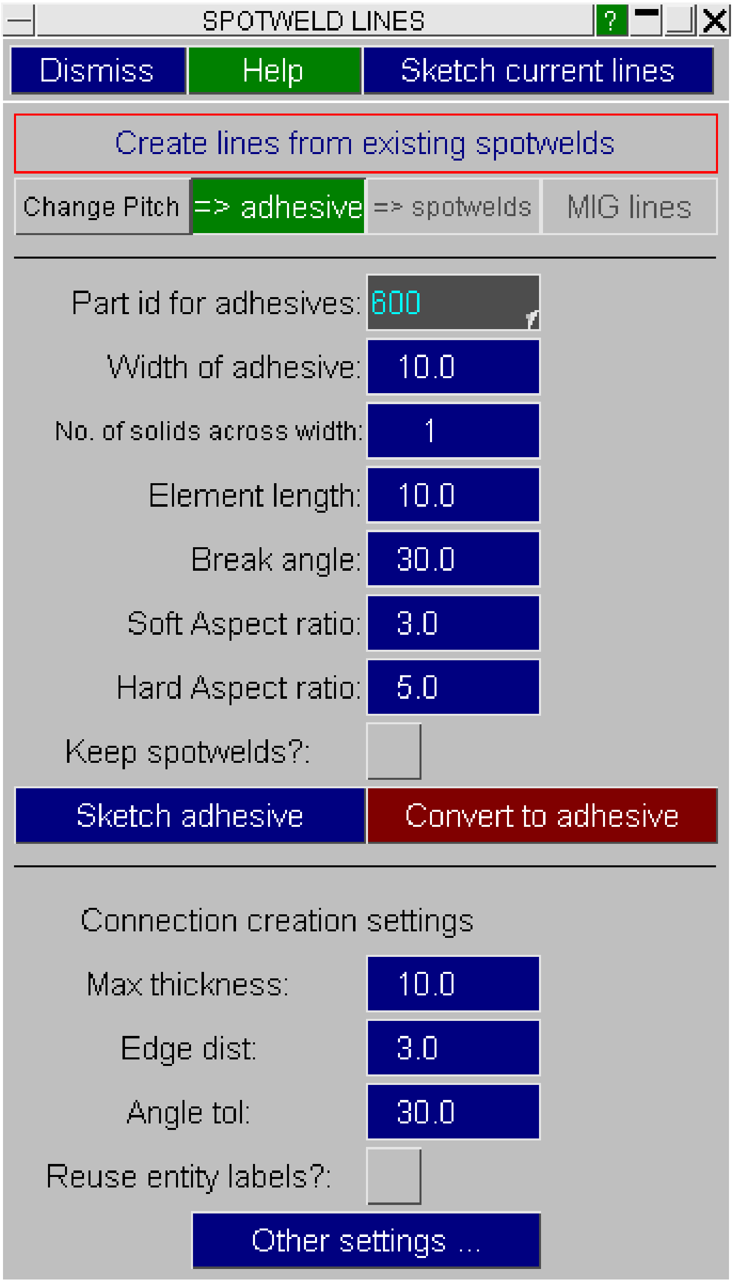
Clicking on **Re-calculate lines** will re-calculate the line groupings based on the current settings. The new pitch you want to apply to the spotweld line groupings in typed into **New spotweld pitch**. The proposed new spotweld positions can be sketched by clicking on **Sketch new pitch**, and the new spotweld pitch can be applied using **Apply new pitch**. The following images show the sketch case and the apply case for the above example when implementing a new pitch of 40.0mm.


Z
X
Y

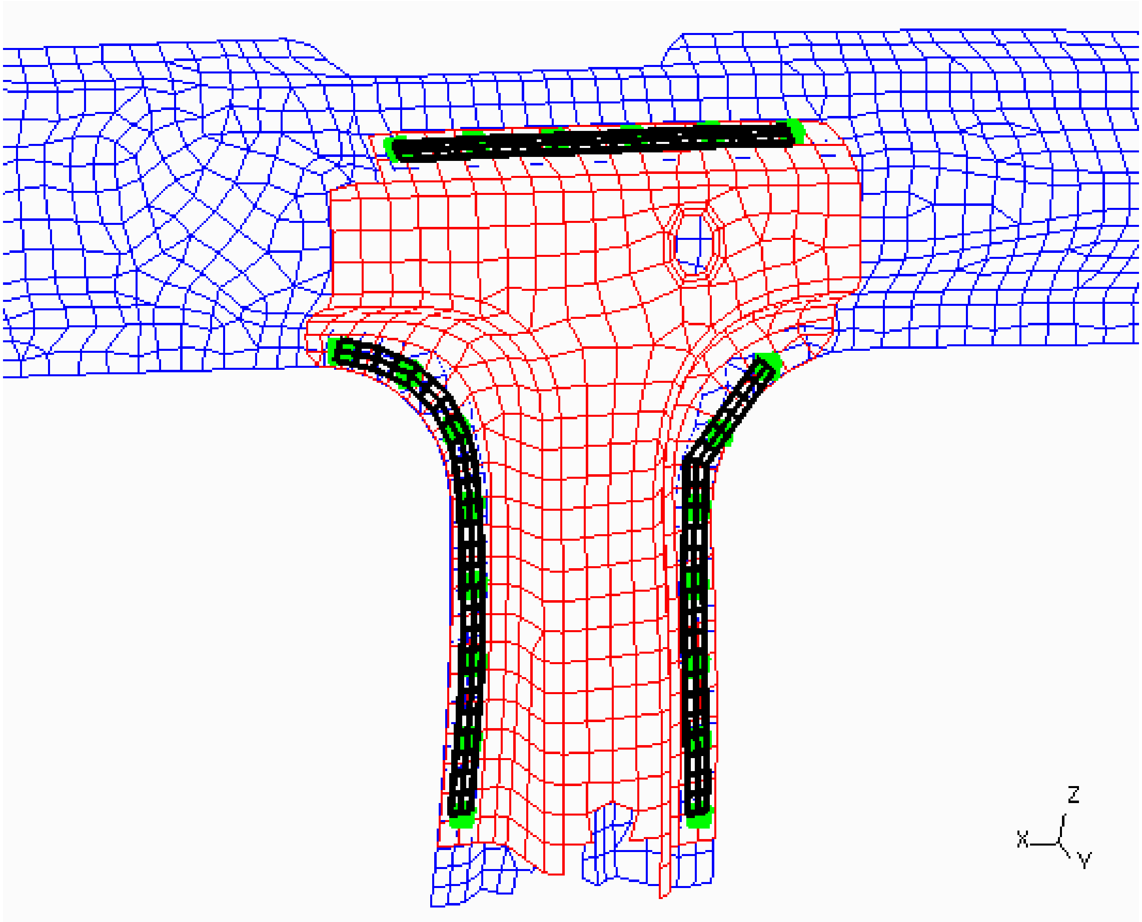

Z
X
Y

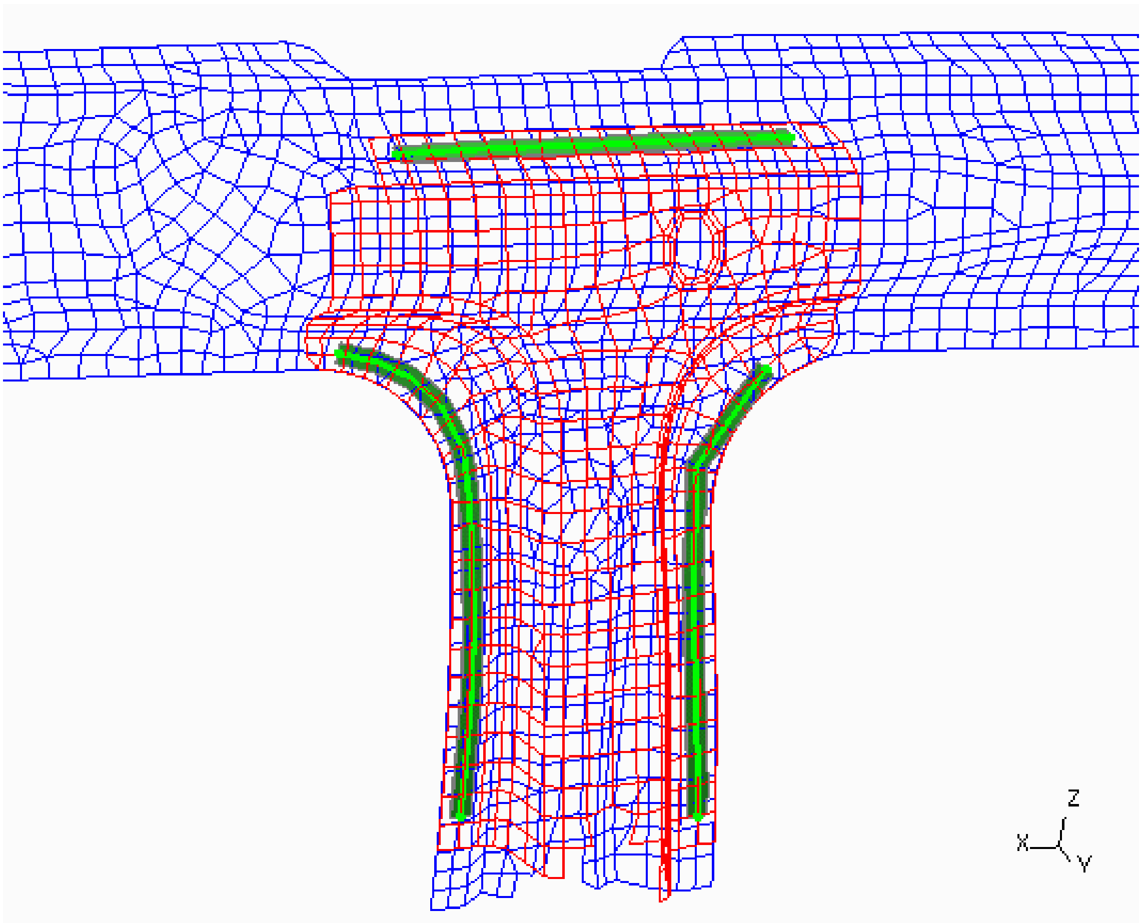
Note that before applying spotwelds, PRIMER will check that the information held for all the spotwelds in the line is consistent. For example, the diameter could vary between spotwelds, or the include file the spotwelds are in. If PRIMER finds inconsistent data, a message will appear asking the user if they wish to proceed. If the do proceed, PRIMER will use information from the first spotweld in the line to create all new spotwelds along the line length.

As mentioned previously, this panel can also be used to convert lines of spotwelds into adhesive runs. Click on **=> adhesive** to see the adhesive creation options.



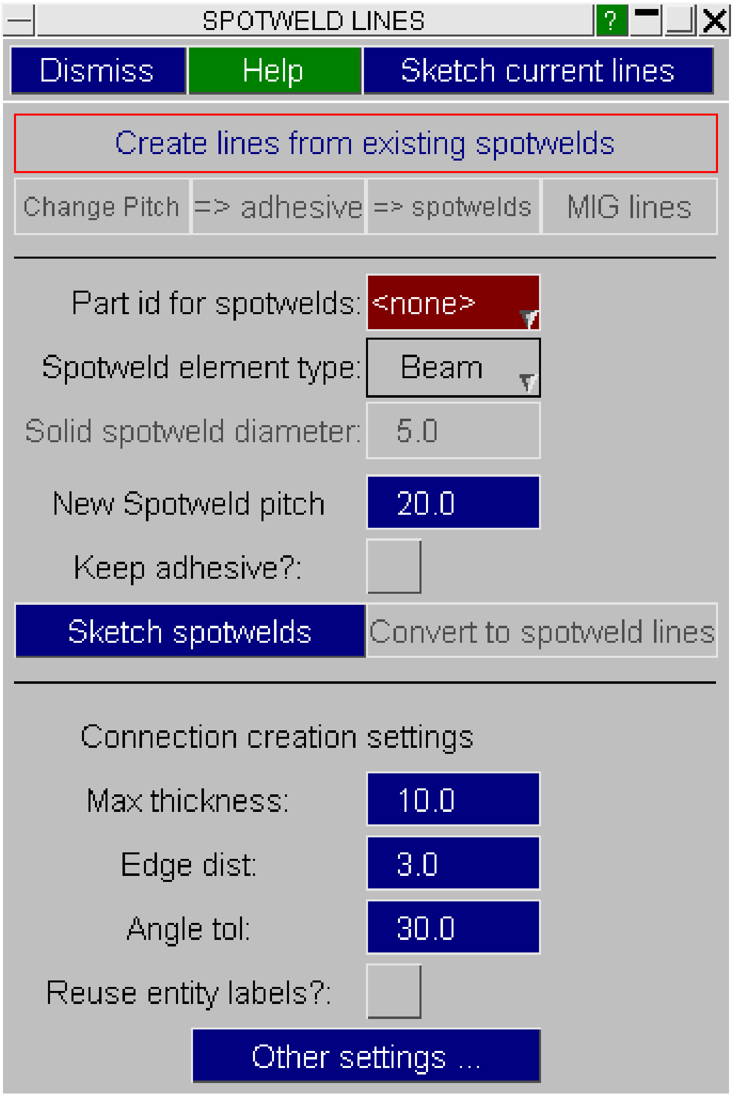
See the create adhesive section for details on the inputs for adhesive creation. The part ID of the solids that the adhesive solids will be created in is the only required parameter. Check the **Keep spotwelds?** tick box should you wish to create adhesive runs but retain the original spotwelds. Again, the user can sketch the proposed adhesive before creation (**Sketch adhesive**). Clicking on **Convert to adhesive** will apply the creation of the adhesive.


Z
X
Y


Z
X
Y

Modifying adhesive

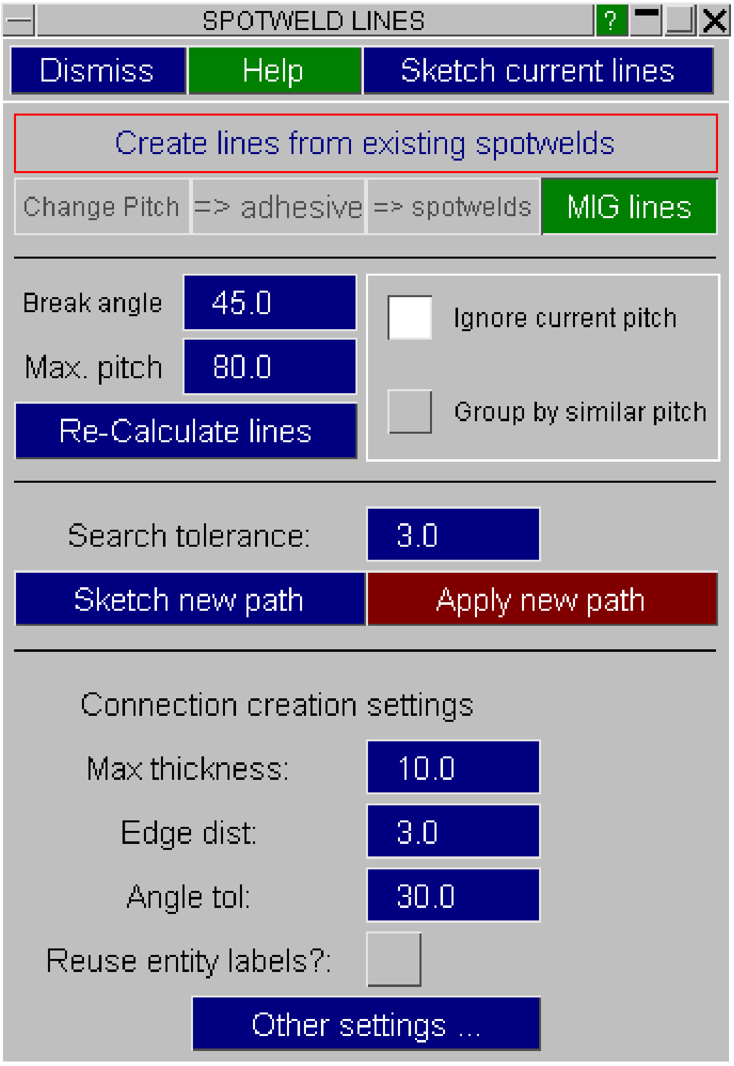
Converting adhesive runs to spotweld lines is also possible. This option is available to you when selecting adhesives when opening up this panel.



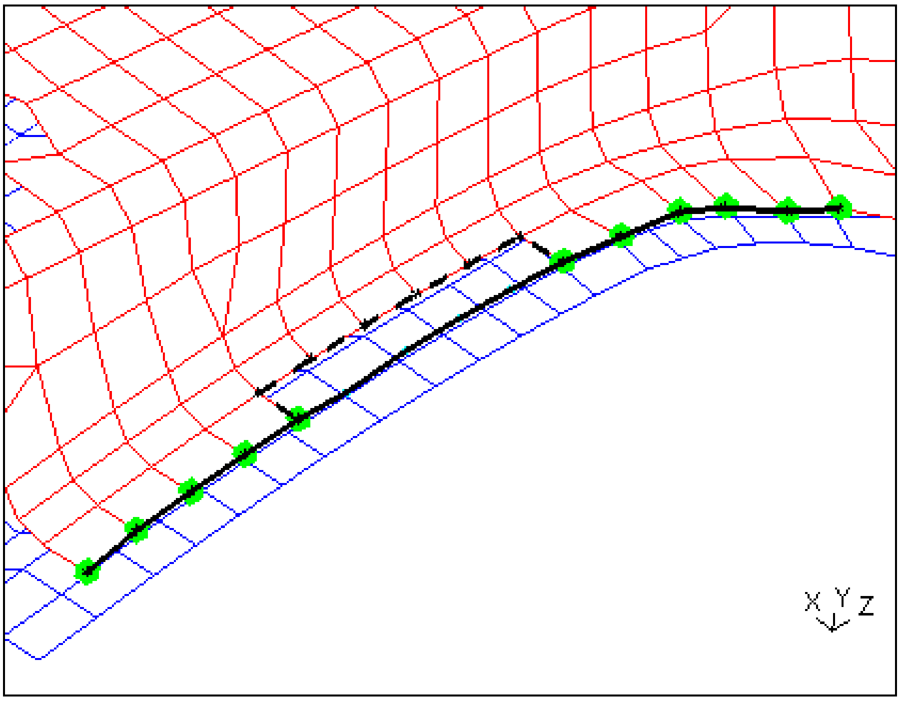
See the create spotweld section for information on the inputs. The part ID of the spotweld solids or beams will be created in is the only required parameter. Check the **Keep adhesive?** tick box should you wish to create lines of spotwelds but retain the original adhesive. Again, the user can sketch the proposed spotwelds before creation (**Sketch spotwelds**). Clicking on **Convert to spotweld lines** will apply the creation of the spotwelds.

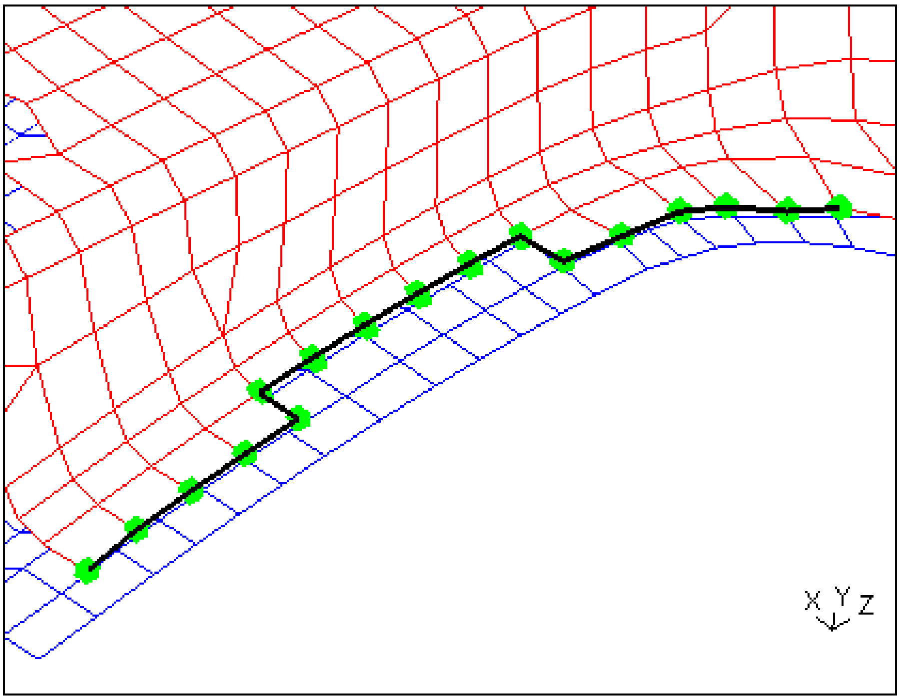
Modifying MIG spotweld types

When openning the spotweld lines panel with only MIG type spotwelds selected, the following panel is presented.



This panel is used to reapply MIG type spotwelds should the mesh of the free edge the MIG welds are attached to change. The panel will again automatically group the MIG welds into lines. The search tolerance is used by the line end point to look for a new start node on the updated free edge mesh. Clicking on **Sketch new path** will display the new path the MIG welds will be applied to. Clicking **Apply new path** will reapply the MIG spotwelds to the new mesh.





Connection creation settings

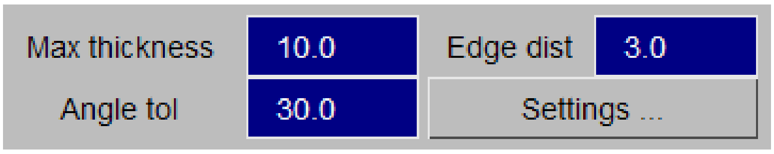
At the bottom of the spotweld lines panel are connection creation settings. These settings are used when creating new spotweld and adhesive entities. During creation, some spotwelds may not be made correctly due to the settings. A common example of this is if the user sets the new pitch to be smaller than the value set for minimum distance between welds (default 10.0mm). Should PRIMER note be able to make new connections for whatever reason, the user is given the opportunity to open these on the connections table for investigation.

For more information on the connection settings see connection settings.

Check the Reuse entity labels? if you wish PRIMER to reuse the entity labels of the previous spotwelds/adhesive when creating new connections.

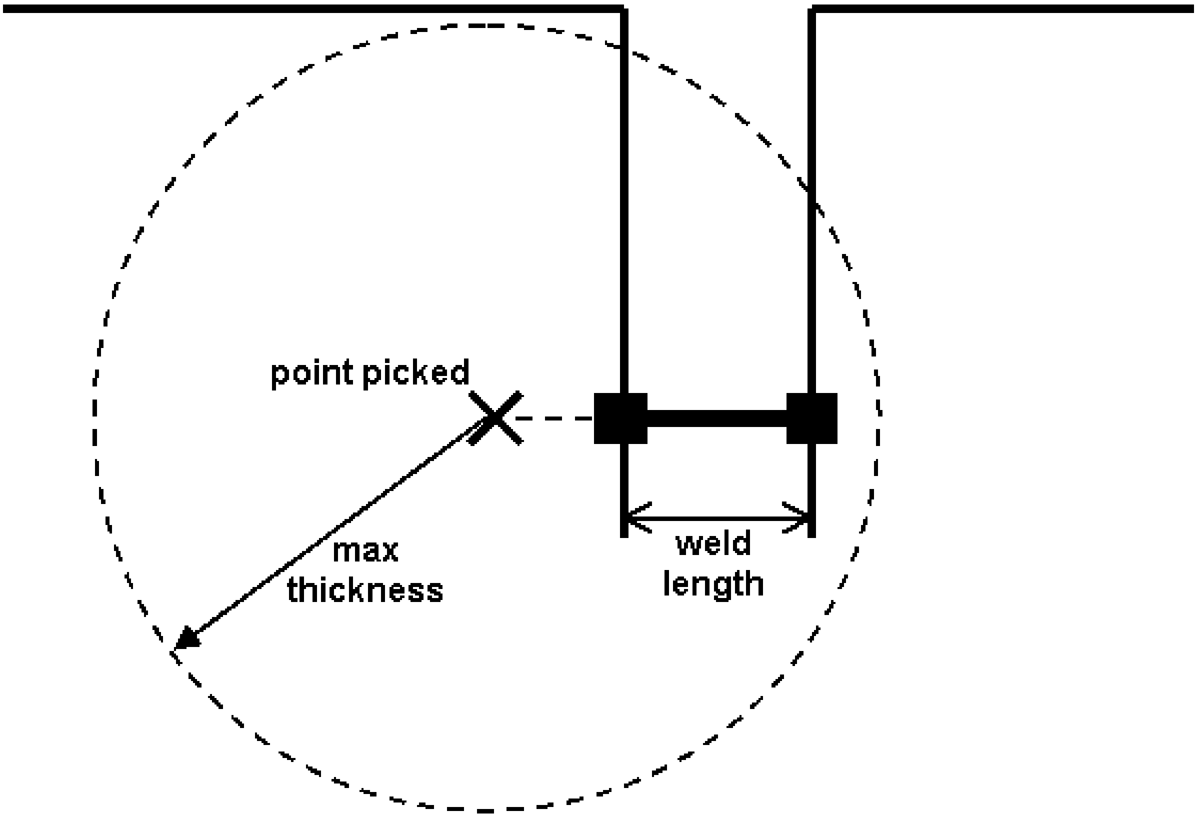
6.10.13 Connection options

There are several options that alter the way that connections are created. Some are found in the connection creating panel



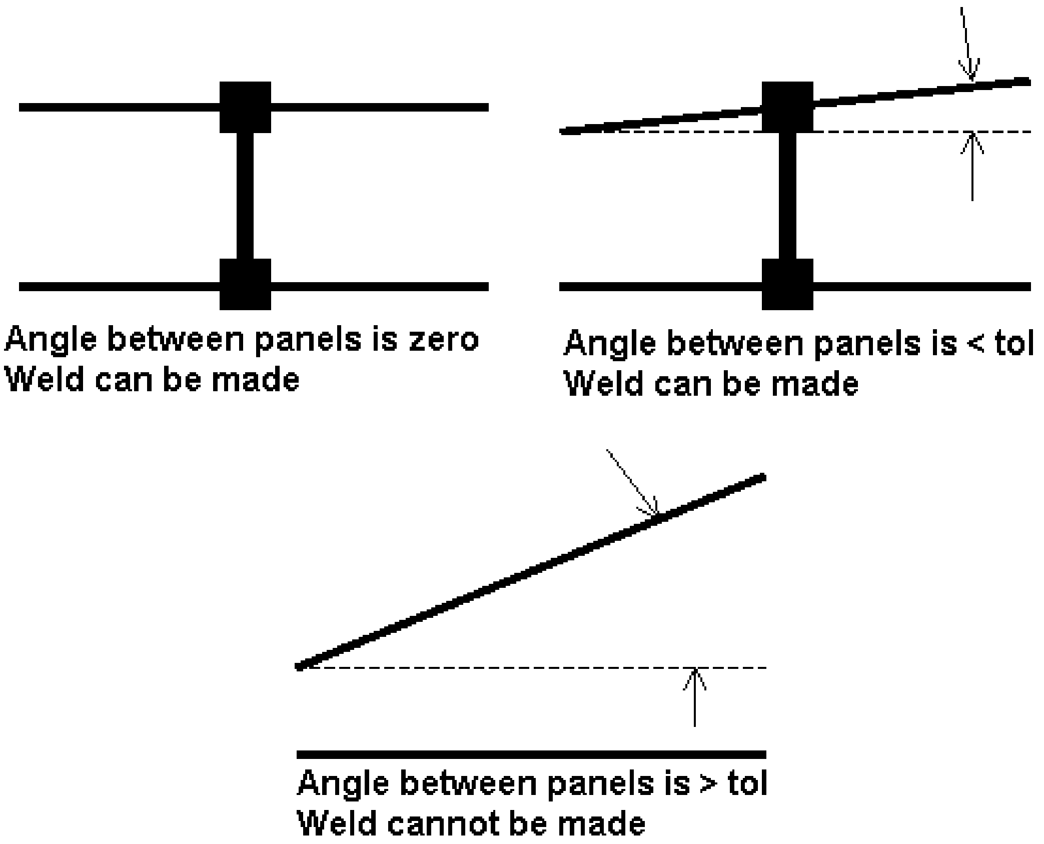
Max thickness

The **Max thickness** dimension is the radius from the point you pick that PRIMER will search for panels to try and weld. In the figure below both panels are inside the circle and so can be welded. If one of the panels was outside the circle the weld would not be able to be made.



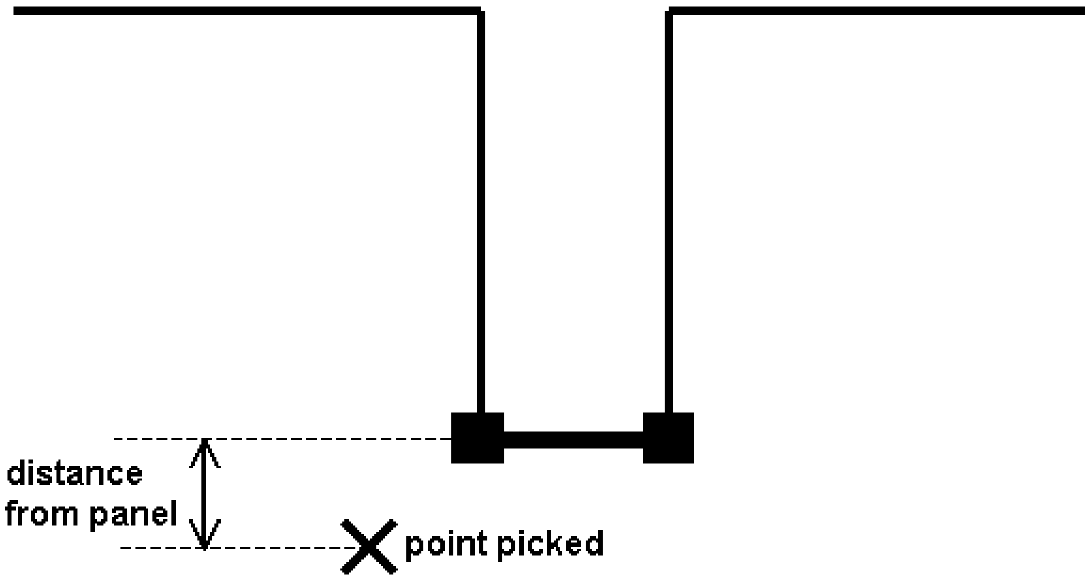
Angle tolerance

PRIMER uses the **angle tolerance** to check elements that it can weld. If the angle between the elements is less than the tolerance, the weld can be made. The figure below shows examples of welds that can and cannot be made.

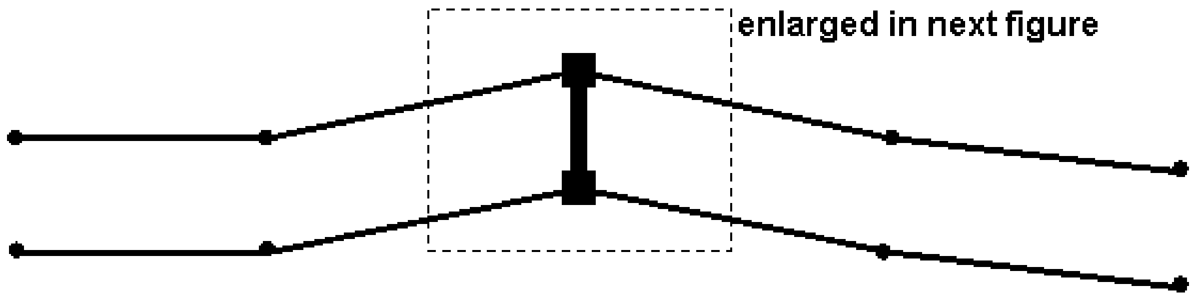


Edge tolerance

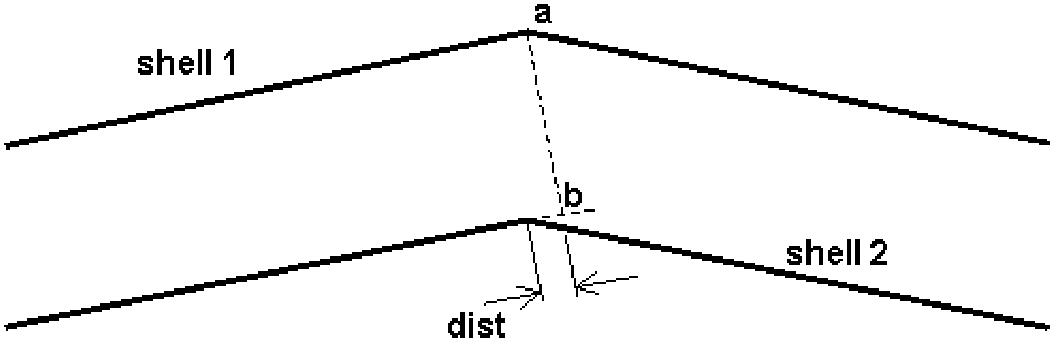
The **edge tolerance** is used to try to find elements if the point you pick is not on a flange. The figure below shows 2 panels with flanges. We want to weld the 2 panels together. The point that is picked is not actually on (or near) the flange. PRIMER checks to find the distance from the panel flanges. If this is less than the edge tolerance then the weld will be made at the end of the flanges. This may not be ideal and in reality you do not want welds on the edges of flanges. To avoid this try to ensure that the weld points are on flanges.



The edge tolerance is also important when welding curved panels. The spotwelder in PRIMER works by locating a point on a panel/element and then creating a beam that is perpendicular to that element. For a curved panel this sometimes does not work. The weld that we actually want to make is shown in the top figure.



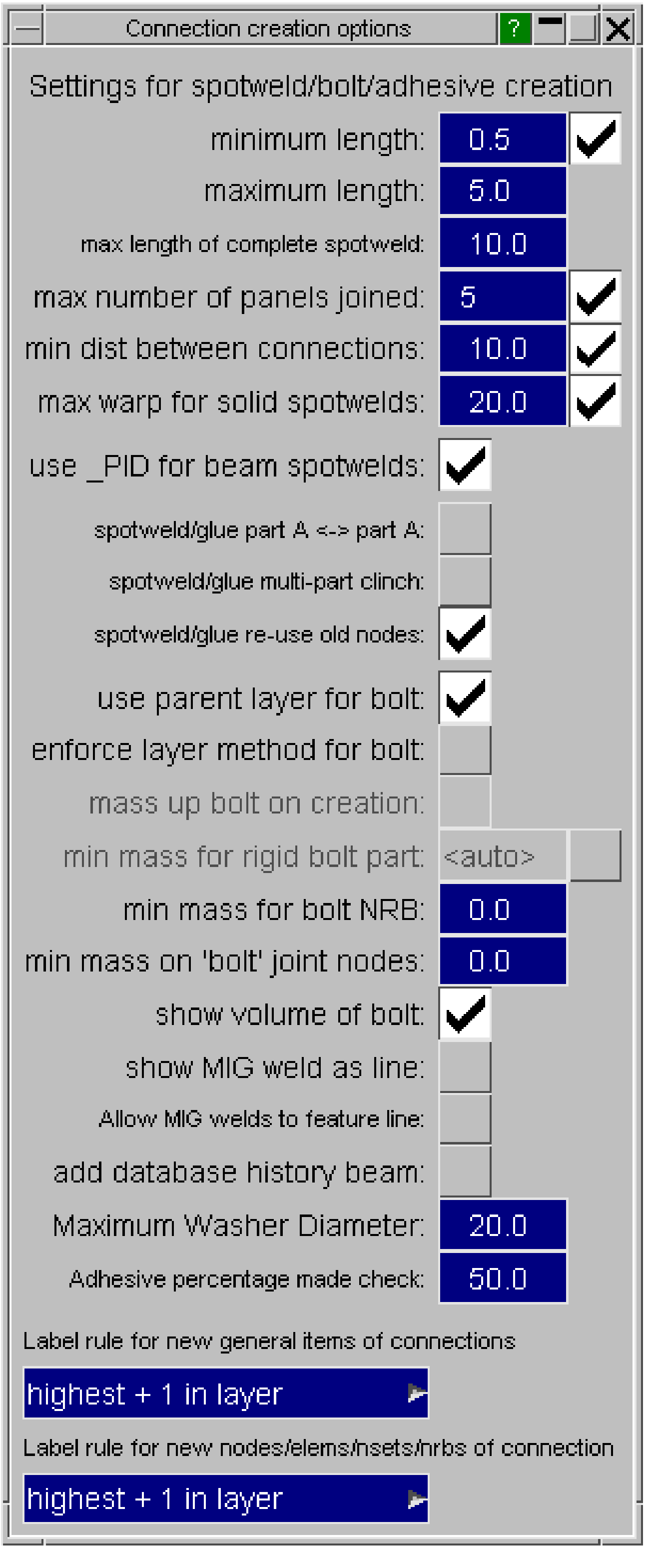
Point **a** is selected for the weld and PRIMER chooses **shell 1** for one end of the weld. PRIMER then projects perpendicular to **shell 1** and looks for elements to weld. PRIMER finds **shell 2**. If PRIMER used this point the weld may not be able to be made as the angle between the panels may be greater than the angle tolerance. Instead, PRIMER checks to see if the distance, **dist**, is less than the edge distance. If it is then the weld (as shown above) can be made.



Other options used when checking/creating spotwelds

Some other options are used in the spotwelder. These are found in the **Settings...** panel.

The figure on the right shows these options.



The **minimum length** and **maximum length** set the minimum and maximum allowed length for a single spotweld beam/solid. This is different to the **max length of complete spotweld** which is the total length of all the spotweld beams/solids in the weld. The figure below shows the difference between these.

max number of panels joined sets the maximum number of panels that PRIMER will allow to be connected together.

The **min distance between spot** allows PRIMER to check the pitch between connections. If a panel has 2 connections that are closer than the minimum distance a warning will be printed. This is very useful to checking for bad weld positions or possible manufacturing problems.

max warp for solid spotweld sets how distorted solid spotwelds can be before PRIMER refuses to create them.

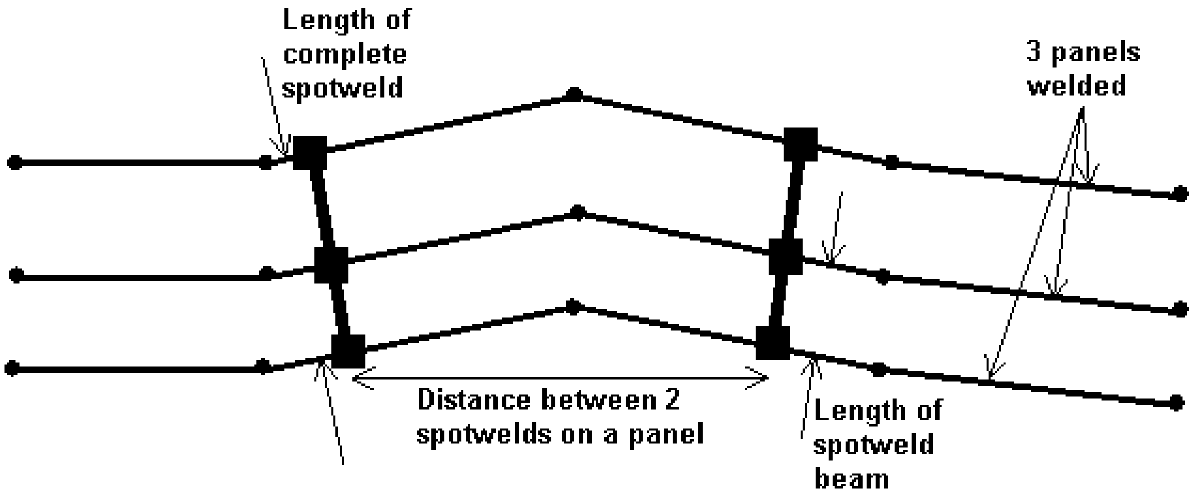
The **use _PID for beam spotwelds** option sets the _PID option on a beam element when creating a spotweld, and supplies the appropriate part ID's.

When **spotweld/glue part A <-> part A** is active, PRIMER is able to connection the same part together (useful for parts folded on themselves or clinches).

When **spotweld/glue multi part clinch** is active, PRIMER is will allow multiple part clinches to be created.

The **label rule for new general items** and **label rule for new nodes/elems/nsets/nrbs** allow you to set what labels are chosen for new entities that are created.

See also bolt options.


Length of complete spotweld
3 panels welded
Distance between 2 spotwelds on a panel
Length of spotweld beam

6.10.14 Spotweld file formats

PRIMER spotweld file format

The PRIMER spotweld file format is designed to be able to be easily read by PRIMER and people. The file should contain either comment lines or spotweld data lines

Comment lines

Any line that begins with a $ (dollar symbol) is treated as a comment line and skipped. This is the same as LS-DYNA.

Spotweld data lines

Any line that is not a comment line is treated as a line containing spotweld data.
Each line contains data for one spotweld.
Each line consists of up to 15 fields.
Each field is 10 characters wide (like most LS-DYNA keyword data) giving a maximum line length of 150 characters.

The fields are defined as follows:

Field number(s)	**Column numbers**	**Description**
1	1-10	Field skipped. Usually contains string 'SPOTWELD' for readability
2	11-20	Spotweld ID number
3	21-30	Field skipped. Usually contains string 'POINT' for readability
4	31-40	Spotweld point X coordinate
5	41-50	Spotweld point Y coordinate
6	51-60	Spotweld point Z coordinate
7	61-70	Field skipped. Usually contains string 'PART' for readability
8-15	71-150	Part ID number. Up to 8 panels to weld together.

Example file

```
$ Primer spotweld file
$ ====================
$
$ Created on: Thu Feb 15 15:04:34 2001
$
$ from model: "ARUP CRASH MODEL - Training (workshop6)"
$
$         < weld ID>          < X coord>< Y coord>< Z coord>          < Part 1 >< Part 2 >< Part 3 >
$
SPOTWELD           1     POINT  2602.424 -692.2456   606.822     PARTS       113         5
SPOTWELD           2     POINT  2623.942-692.18854    605.34     PARTS       113         5
SPOTWELD           3     POINT  2634.647 -692.1887  604.6709     PARTS       113         5
SPOTWELD           4     POINT   3142.42 -695.2453  547.7576     PARTS         5       306
SPOTWELD           5     POINT   3142.42 -695.2453  522.7576     PARTS         5       306
SPOTWELD           6     POINT  3101.431-692.23694  483.0748     PARTS       306         5
SPOTWELD           7     POINT  3075.443-692.23694 483.39206     PARTS       306         5
SPOTWELD           8     POINT  3049.455-692.23694  483.7093     PARTS       306         5
SPOTWELD           9     POINT  3023.467-692.23694  484.0266     PARTS       306         5
SPOTWELD          10     POINT  3102.919-692.23694  587.7574     PARTS       306         5
```

Catia spotweld file format

The Catia spotweld file has a set format and is read into PRIMER in the following way:

The lines are split into words and parsed until one matches the following format:
<string> <string> <string> <floating point number>,<floating point number>,<floating point number> <strings>

The first string is stripped of non-digits to obtain a weld ID. e.g. 4118-5o converts to 41185.
The second string identifies how many parts the weld should attach and the reader should look for.
The three floating point numbers are stored as X Y and Z coordinates for the weld.

The reader then takes the amount of parts to look for (from the second string) and looks in the following lines for a part ID. The part ID is taken from characters 28 to 34 inclusive and should result in a 7 digit number. If it's identified as a 2 thickness weld, then Primer will expect a part ID in each of the next 2 lines.

An example file is as follows:

```
=========================================================================
| BIW Assembly Features :                                               |
=========================================================================

SPOT WELDS
==========

ID      Feature Location (X,Y,Z)
--      ------- ----------------
4118-5o 2T Spot 4074.49,-295.82,1680.00 Ordinary class 3 Jul 29,2002 No
   - NEED TO BE FIXED - 1: 55354119AB A (MO-6000 44A, 0.79 mm)
   - NEED TO BE FIXED - 2: 55396175AA A (M3-67, 2.03 mm)
4118-7o 2T Spot 2041.68, -301.24, 1680.00 Ordinary class 3 Jul 29,2002 No
   - NEED TO BE FIXED - 1: 55354119AB A (MO-6000 44A, 0.79 mm)
   - NEED TO BE FIXED - 2: 55396175AA A (M4-67, 2.03 mm)
4118-9o 2T Spot 2038.04, -356.46, 1680.00 Ordinary class 3 Jul 29,2002 No
   - NEED TO BE FIXED - 1: 55354119AB A (MO-6000 44A, 0.79 mm)
   - NEED TO BE FIXED - 2: 55396175AA A (MY-67, 2.03 mm)
```

UG spotweld file format

The UG spotweld file has a set format and is read into PRIMER in the following way:

The first line is skipped, and after that the only lines PRIMER reads are lines containing "spot". PRIMER expects the following comma separated order:

<string>,<weld id>,<number of panels to weld>,<X coord>,<Y coord>,<Z coord>,<part strings>

The part strings should match exactly what PRIMER contains in the *PART title fields.

An example file is as follows:

```
WELD_TYPE,ID,NUMBER OF SHEETS WELDED,X_POS,Y_POS,Z_POS,CONNECTED PART
1,CONNECTED PART 2,CONNECTED PART 3,CONNECTED PART 4
SPOT_WELD_TYPE_UNKNOWN
resistance
spot,2,2,2623.941895,-693.717041,605.340027,A_pillar_lower_support_a,sill_swan_neck,
resistance
spot,3,2,2634.646973,-693.717102,604.670898,A_pillar_lower_support_a,sill_swan_neck,
resistance
spot,4,2,3142.419922,-693.741089,547.757629,sill_swan_neck,seat_xmember_outer,
resistance
spot,5,2,3142.419922,-693.741089,522.757629,sill_swan_neck,seat_xmember_outer,
resistance
spot,6,2,3101.430908,-693.741089,483.074829,seat_xmember_outer,sill_swan_neck,
resistance
spot,7,2,3075.443115,-693.741089,483.392059,seat_xmember_outer,sill_swan_neck,
resistance
spot,8,2,3049.455078,-693.741150,483.709290,seat_xmember_outer,sill_swan_neck,
resistance
spot,9,2,3023.467041,-693.741150,484.026611,seat_xmember_outer,sill_swan_neck
```

PRIMER XML connection file

A small example of a PRIMER XML connection file is given below.

```
<?xml version="1.0"?>
<!-- Primer connection file -->
<!-- ===================== -->
<primer_connections version="9.3">
  <connection type="spotweld">
    <title></title>
    <id>1</id>
    <coord x="11.292786" y="70.951309" z="21.500000" />
    <diameter>5.000000</diameter>
```

```
    <method>hexa</method>
    <spotweld_part_id>101</spotweld_part_id>
    <layer type="PART_ID">
      <part id="1" />
      <part id="3" />
    </layer>
    <layer type="PART_ID">
      <part id="2" />
    </layer>
  </connection>
</primer_connections>
```

The main tag of the file must be `primer_connections`. Currently the only version supported is `9.3`.
Each connection is then given inside a `connection` tag. The currently available types are `spotweld, rigid` and `adhesive`. Several tags are then used inside the `connection` tag to define the connection property. Most tags should be obvious.

For connection type `spotweld`, the available `methods` are `beam`, `hexa`, `4_hexa`, `8_hexa`, `12_hexa` and `16_hexa`.
For connection type `rigid`, the available `methods` are `rigid_body_merge` or `nodal_rigid_body`.
For connection type `adhesive`, the only available `method` is `solid`.
The method tag is optional. If it is omitted then PRIMER will use the default option when reading the connection file.

Spotweld connections can contain the `spotweld_part_id` tag which tells PRIMER what part to create the spotweld in. It is optional. If it is omitted then PRIMER will use the default option when reading the connection file. For rigid connections there is an equivalent optional tag to specify the material called `rigid_material_id`, and for adhesive connections there is an equivalent optional tag to specify the material called `adhesive_material_id`.

Adhesives contain more information than spotwelds and bolts (an example is shown below). The adhesive information contains the end point (coord2) and the path data (points between the start and end points). It also contains adhesive width, number of elements across the width and element size.

```
<?Xml version="1.0"?>
<!-- Primer connection file -->
<!-- ===================== -->
<primer_connections version="9.3">
  <connection type="adhesive">
    <title></title>
    <id>2</id>
    <coord x="-13.702406" y="-39.206017" z="54.000000"    />
    <coord2 x="100.094444" y="-33.088951" z="54.000000"    />
    <method>solid</method>
    <adhesive_part_id>600</adhesive_part_id>
    <adhesive width="10.000000" number="1" size="10.000000"    />
    <path x="18.048088" y="-27.360266" z="53.999996"    />
    <path x="42.419262" y="-18.718697" z="54.000000"    />
    <path x="82.520035" y="-17.456446" z="54.000000"    />
    <layer type="PART_ID">
      <part id="10" />
    </layer>
    <layer type="PART_ID">
      <part id="21" />
    </layer>
  </connection>
/primer_connections>
```

A `layer` tag is then given for each layer of the connection. The available layer types are `PART_ID`, `PART_NAME`, `CAD_NAME`, `ASSEMBLY`, `SETPART_ID`, and `SETPART_NAME`. The layer tag then contains the item(s) that are used for the layer. The tag and attribute change depending on the layer type. The following table shows the values.

Layer type	**Tag**	**Attribute**	**Example**
PART_ID	part	id	`<part id="1" />`
PART_NAME	part	name	`<part name="panel12345" />`
CAD_NAME	part	CADname	`<part CADname="CATIA_12345_xyz" />`
ASSEMBLY	assembly	name	`<assembly name="biw" />`
SETPART_ID	partset	id	`<partset id="1" />`
SETPART_NAME	partset	name	`<partset name="body_side_parts" />`

6.11 CUT SECTIONS

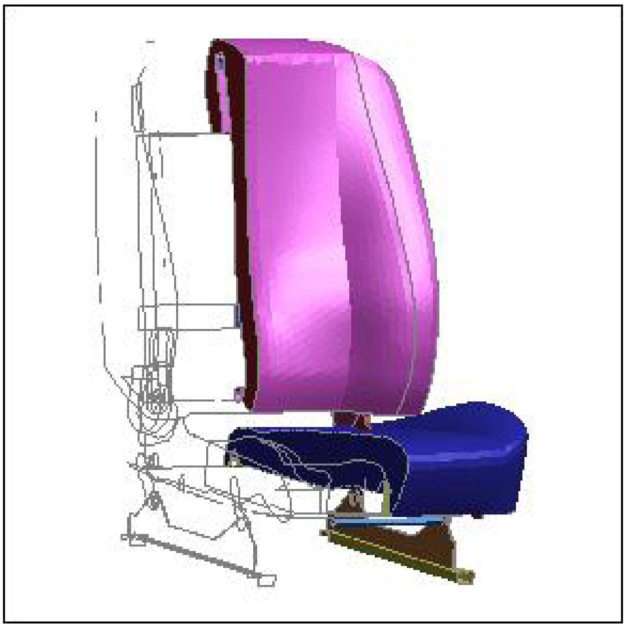
The **Cut Section** menu is invoked from the Tools menu or from keyboard shortcut X.

A cut-section, sometimes referred to as a "cutting plane", is a flat plane that cuts through the model. It may be located anywhere in space and oriented at any angle.

When the **Cutting switch** is turned on the intersection of the plane with the model is calculated and the interpolated cut plane is drawn.

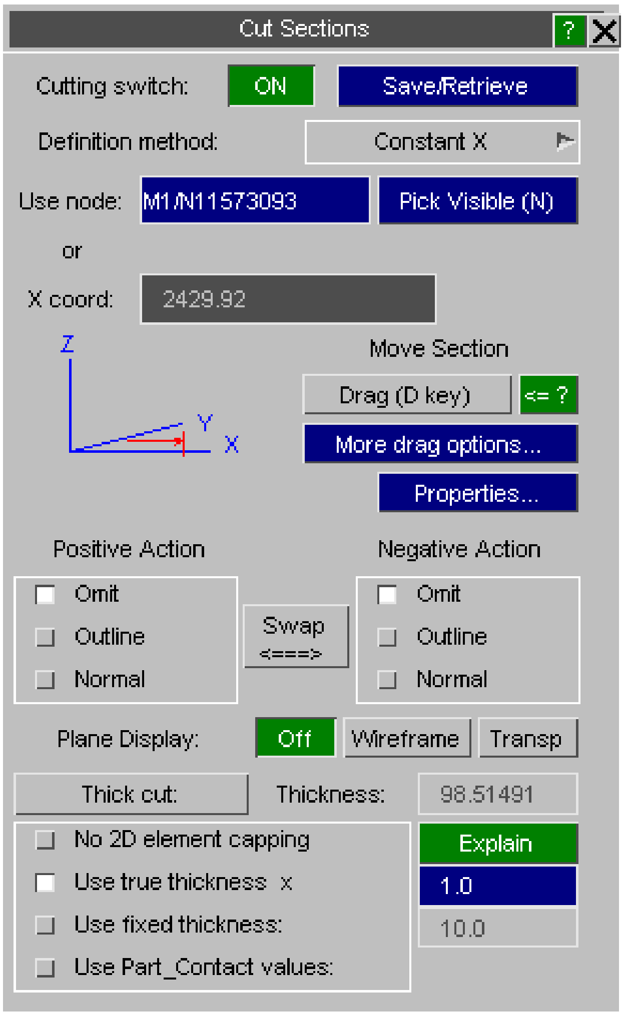
Various options, described below, define if and/or how the model either side of the plane is drawn.

This image shows a conventional plot of a seat model, with no cut sections active



In this image the a cut section has been turned on, located roughly half-way across the seat. The +ve side (on the right) is drawn normally, the -ve side (on the left) is drawn in outline.

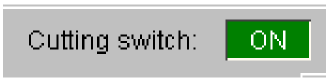
6.11.1 The Cut Sections panel



The parts of this panel are summarised below. Click on one to jump to the more detailed description.

Cutting switch	Normally OFF, in which case cut-sections are inactive. May be toggled on/off at any time.
Save/Retrieve	Saves cut-sections, and restores previously saved ones
Definition method	Toggles between the six different ways of creating a cut section
Properties	Extracting engineering properties (Area, I, etc) of the cut elements
Move Section	Various options for dragging the section with the mouse
Positive & Negative action	How the +ve and -ve sides of the plane are displayed
Plane display	Whether, and how, the actual plane itself is displayed
Thick cut	Optional "thick" (finite thickness) plane display
2D element capping	How the cuts through 2D elements are displayed

6.11.2 Cutting Switch



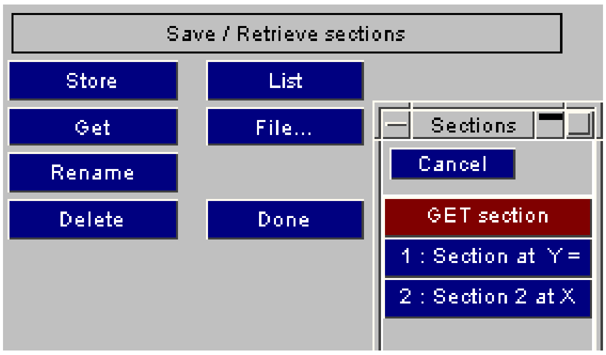
Controls whether or not the cutting plane is active

Initially cut sections are turned off and no plane is active. When a plane's properties have been specified turn this on to see the effects. It can be toggled on/off at any time.

6.11.3 Save/Retrieve

Controls the saving and retrieving of cut-section definitions.

Only one cut section can be active at any time, but any number of cut section definitions can be saved to disk for subsequent retrieval.



To save a section:

- Open a file with **File...** (default "section.cut")
- **Store** the file, giving the section a name.

To retrieve a section:

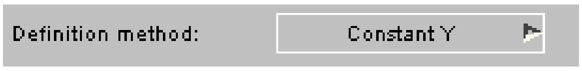
- Open a file if not already open
- **Get** the section from the list. It will be applied immediately

Sections in the file can also be **Delete**d and **Rename**d at will. To return to the main cut section panel use **Done**.

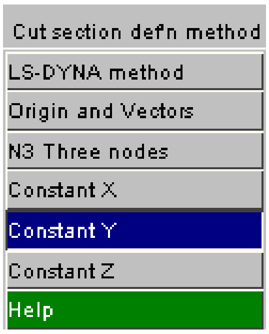
You can open a different cut section file at any time, and have any number of such files on disk. The default name is "section.cut", but any name may be used.

The file format is common with D3PLOT, so the same section definitions can be used in both programmes.

6.11.4 Definition method



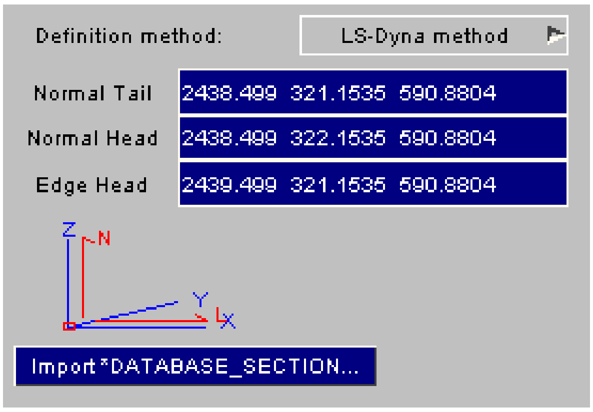
There are six different ways of defining a cut section, chosen from the pull-down menu. The data entry panel changes for each mode. Click on a method below for details.



LS-DYNA method	Tail, Head and Edge head coordinates are defined.
Origin and vectors	Origin coordinate is defined, then vectors for local X axis and XY plane
N3 Three nodes	Three nodes: N1 at origin, N2 giving X axis, N3 the XY plane
Constant X	Cut at constant X value
Constant Y	Cut at constant Y value
Constant Z	Cut at constant Z value

Regardless of how the plane is defined its actual characteristics and geometry will be the same.

LS-DYNA method

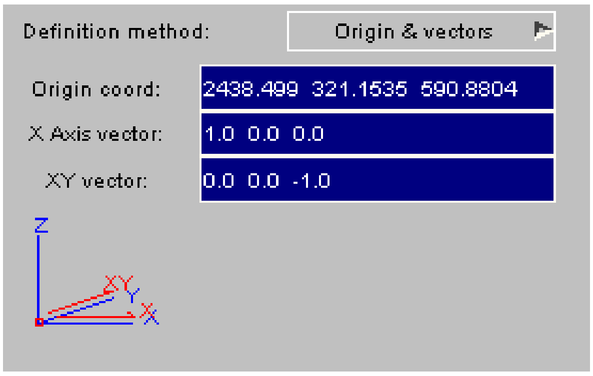


This entry method mimics the data format of the *DATABASE_CROSS_SECTION card in the LS-DYNA input deck. You define:

- The Tail coordinate of the normal vector (origin)
- The Head coordinate of the normal vector (local Z axis)
- A Head coordinate of a vector on the XY plane

If there are any *DATABASE_CROSS_SECTION cards in any models the definition can be built from those using **Import...**

Origin and Vectors

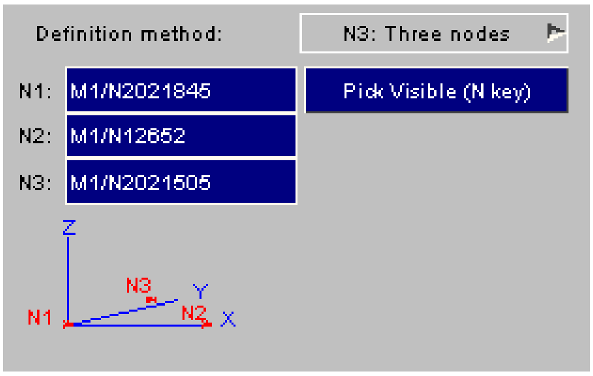


Here you give:

- An origin coordinate
- A vector defining the local X axis
- A vector on the local XY plane

The normal (local Z) vector is obtained from the vector cross product of these.

N3 Three nodes

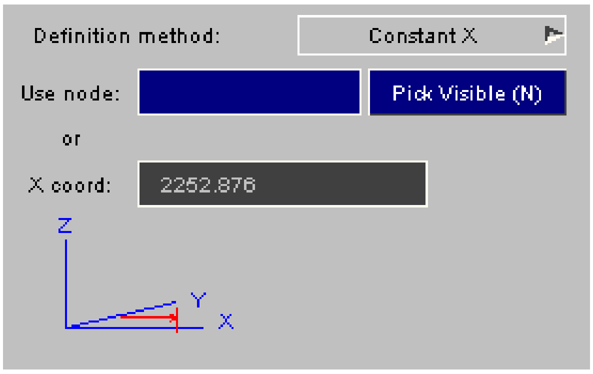


Here three nodes are defined:

- N1 is the plane origin
- N2 is on the local X axis (vector N1N2)
- N3 is on the local XY plane

The normal (local Z) vector is obtained from the vector cross product of these.

Constant X
Constant Y
Constant Z



In these cases define either:

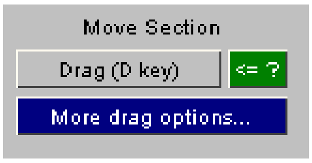
- A coordinate on the relevant axis

  or

- A node, from which the relevant coordinate will be extracted. (The **N** keyboard shortcut will jump straight to this mode.)

A plane will be defined at a constant value of the relevant axis at that point.

6.11.5 Dragging the cut-section



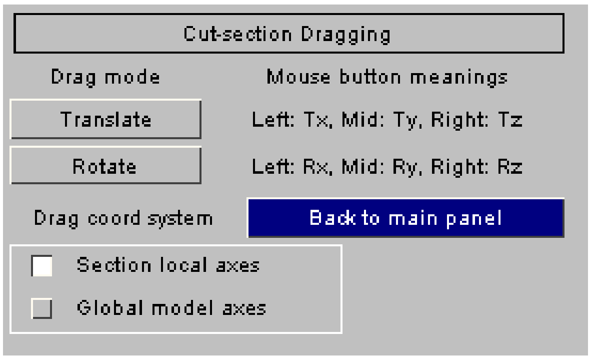
Once the cut-section has been defined it can be moved to a new position and orientation by dragging with the mouse.

Drag (D key) Either clicking on the button, or using the **D** keyboard short-cut invokes this mode.

The Cut-section panel acquires control of the mouse (the cursor symbol changes to "sect drag" to signify this) and the mouse buttons work as follows:

Mouse button	Cursor Symbol	Action
Left	Tz	Translates the plane in the normal (local Z) direction
Middle	Rx	Rotates the plane about its local XX axis
Right	Ry	Rotates the plane about its local YY axis

More drag options...



This gives access to a more complex set of options for dragging the section. You need to choose:

- Drag mode: either translate or rotate
- Drag coordinate system: section local or global

Mouse buttons then translate/rotate in/about axes:

Left button : Tx / Rx
Mid button: Ty / Ry
Right button : Tz / Rz

How mouse motion is interpreted when dragging

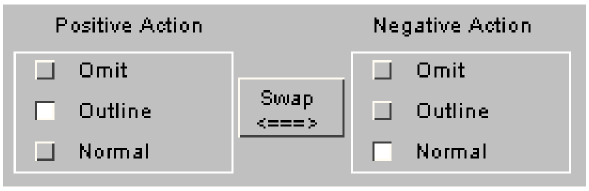
In most cases the mouse motion is projected onto the section axis to be dragged, as shown on the screen, giving an intuitive result as if you had grabbed the section with the mouse and dragged it.

However this method fails when the section axis to be dragged points directly in or out of the screen since the dot product of its vector (screen Z) with mouse motion (screen XY) is zero. Therefore when the axis to be dragged lies within approximately 1 degree of screen +/-Z then an alternative method is used:

- +ve mouse motion in screen X or Y equates to +ve motion down the section drag axis.
- -ve mouse motion gives the opposite effect.

Put more simply: in these cases mouse motion to the right (+X) or up (+Y) results in +ve motion down the section axis, and left (-X) or down (-Y) gives -ve motion.

6.11.6 Positive & Negative action



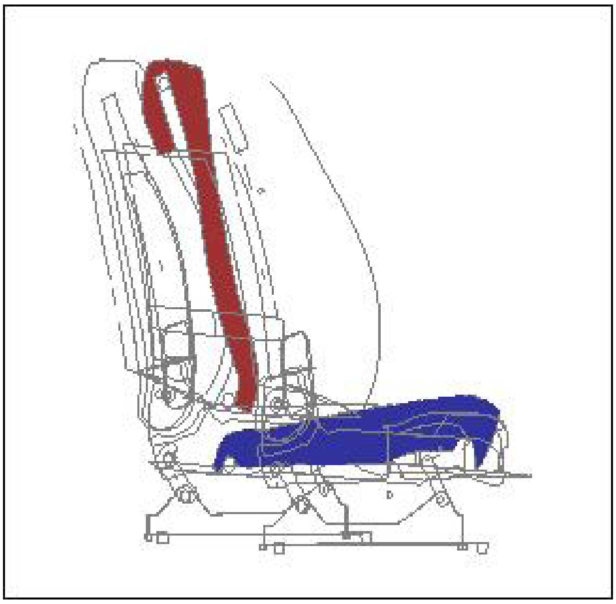
Controls how the image on either side of the plane is rendered.

The cutting plane itself is always rendered in the current display mode, but for each side of the cutting plane you must choose how the image is to be rendered:

- **Omit** means that it will not be drawn at all
- **Outline** means that it will be drawn in wireframe outline, in the edging mode of the current display mode.
- **Normal** means that it will be drawn in the current display mode.

Swap <===> simply swaps the +ve and -ve display modes around and redraws.

Any permutation of modes can be drawn on either side, here are some examples for the model above:



In this example both +ve and -ve sides have been set to **Outline**.

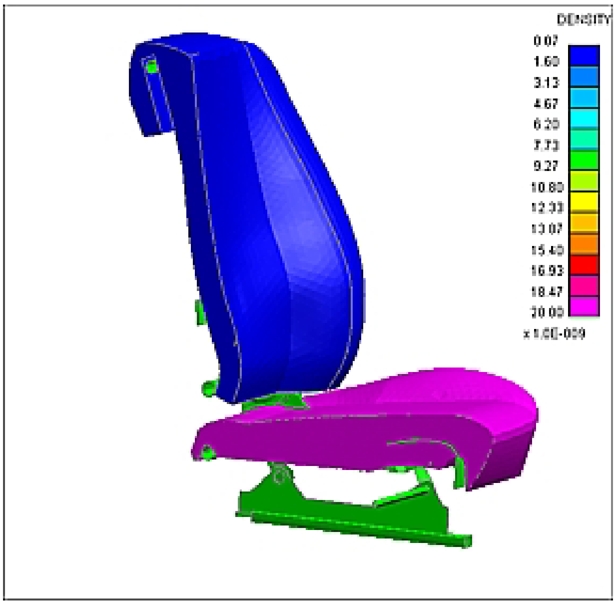
Because the current display mode is **Shaded**, with free edge outlines, this means that they are rendered in free edge wireframe mode.

The cutting plane itself is clearly visible in shaded mode, and because this model contains solid elements these are capped on the cut plane and therefore easy to see.

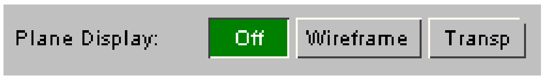
Here the +ve (far) side is displayed in **Normal** mode, and -ve side has been **Omit**ted.

The display mode is SI this time, showing element density.

This demonstrates the cut-sections can be used with any plotting mode, includingdata-bearing ones.



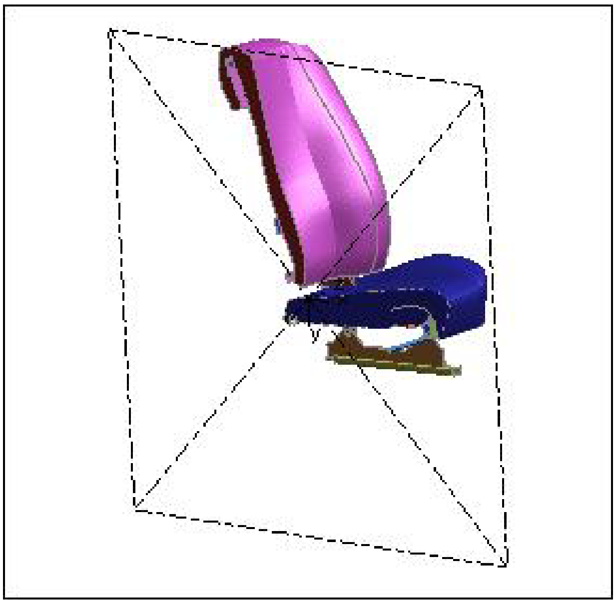
6.11.7 Plane Display



Whether, and how, the actual plane definition itself is displayed. By default plane display if Off and it is not shown. However it is possible to draw the plane in one of two modes:

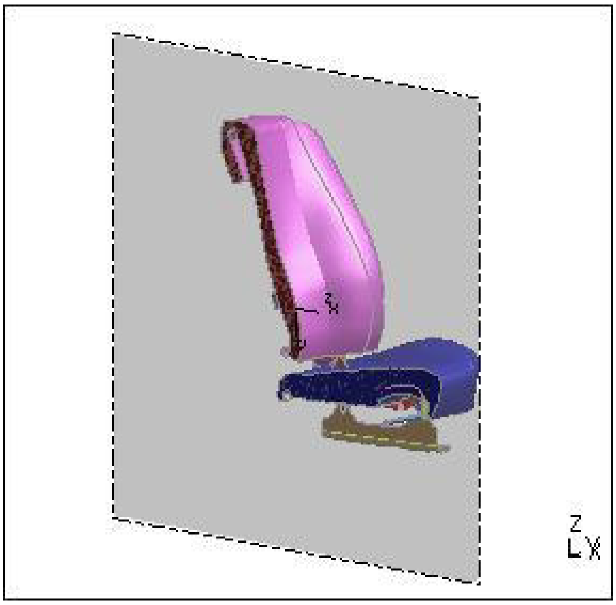
Wireframe

Draws the plane boundaries and a diagonal as a wireframe overlay on the plot



Transparent

Draws the plane as a partially transparent square, occupying model space and intersecting the structure.



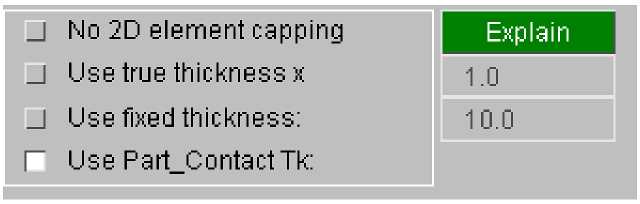
6.11.8 Thick cut

Alternative "thick" plane display mode. In this mode the plane is extruded in the local Z axis by +/- Thickness/2.0 either side of its "thin" position, effectively forming two planes with solid structure in between.

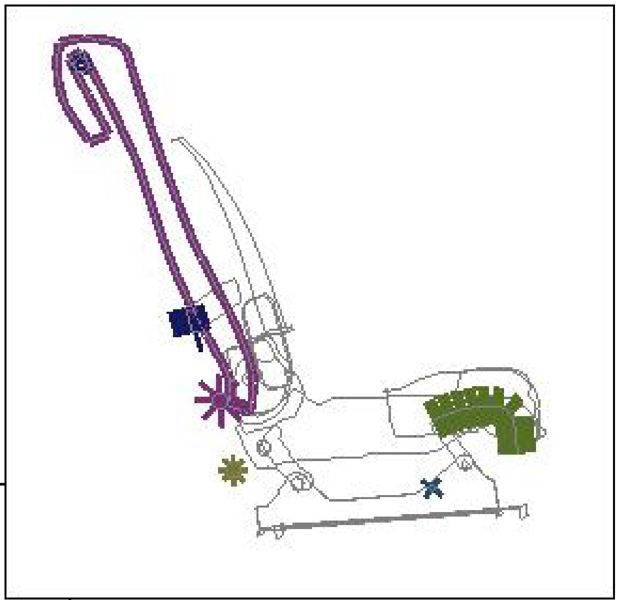
This example shows the model above rendered in this way.

Note that nothing is displayed outside the extruded +ve and -ve planes, (and the Positive abd Negative action options are inoperative).

6.11.9 2D element capping



Controls how the cut edges of 2D elements (shells) are displayed.



When shell elements are cut it is possible to draw their cut edges in three modes:

- **No 2D capping**. The cut is simply a line with colour but no thickness.
- **True thickness x factor**. Extracts the true shell thickness, multiplies it by <factor> and uses that value. This is probably the most useful since it shows actual model dimensions, although a factor > 1.0 is often necessary to visualise thicknesses.
- **Fixed thickness**. Uses a constant value in model space units for all shells.
- **Use Part_Contact values**. (See also notes on plotting contact thickness below)

For shells on a `*PART_CONTACT` card: If explicit thickness `OPTT` is defined this is used, or If scale factor `SFT` is defined then the truethickness x `SFT` is used or The unscaled true thickness is used.
For other shells the unscaled true thickness is used.

In this example solids have been turned off leaving only shells, which have been rendered using True thickness x 15.0 to make them stand out at this scale.

Notes on using cut sections to plot contact thickness

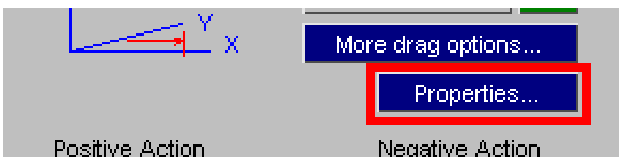
There have been requests for cut sections in PRIMER to show contact thickness generally, but this is not really practical for two reasons:

- A given element may be in more than one contact, and the thickness used can be influenced by the contact type and the settings on the contact card itself, so there may not be a unique value.
- Inside LS-DYNA the relationship between elements specified for contact and those actually used is rather weaker than it may at first appear. When a contact surface is created the following process is used to determine the geometry of the contact:
  - Segments are built from all shells or 3D element faces in the contact definition, or explicit segments are used directly.
  - Duplicate segments are eliminated.
  - The element under each segment is then used, whether or not it was specified in the original contact definition.
  - If shells overlay solids, or coincident shells are present, the choice of element is complicated further.

Therefore in a model with more than one contact surface it is nearly impossible to determine a general "thickness used for contact" for every shell, and the only real solution is to limit to display to elements in a given contact.

The contact penetration checker performs all these calculations for the specified contact surface, and if Settings, As Thick is chosen when displaying penetrations then the thickness of each segment will be shown. If cut sections are then turned on they will apply to these penetration plots, and in this way it will be possible to visualise penetration thicknesses.

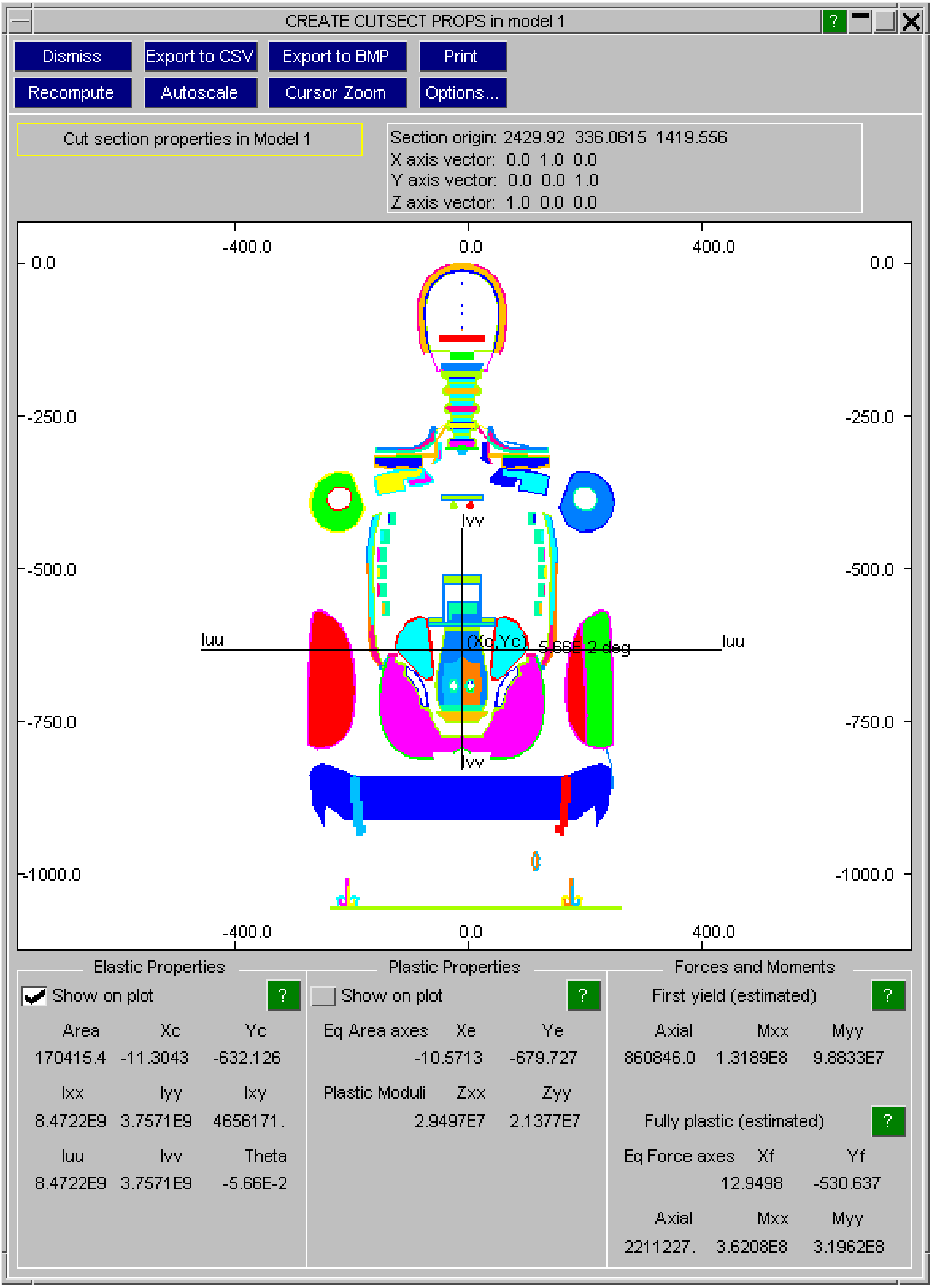
6.11.10 **Properties**: Computing cut section properties



PRIMER is able to derive section properties (Area, 2nd moments of area, plastic moduli and section capacity) from the elements cut by the plane. Properties:

- Are derived from 3D (solid and thick shell), 2D (thin shell) and 1D (beam) elements only. All other types cut by the plane are ignored.
- Are calculated only from what is currently visible, so blanking and entity switches may be used to limit what is used in the calculation
- Use the local XY plane of the cut-section as their frame of reference.

An example of a cut through a dummy positionined on a seat is shown below



The method used to calculate properties.

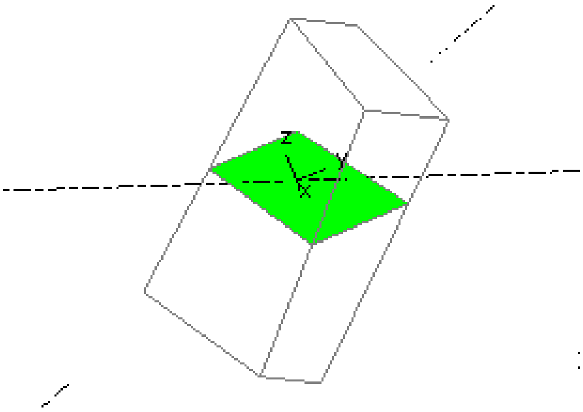
In general terms the cut area through each element is computed as a flat 2D polygon, or series of polygons in the case of complicated beam sections, and transformed into 2D cut section XY space. The origin of the cut section is implicitly the origin of this 2D space system, its X axis becomes left/right and its Y axis becomes up/down. Properties are then calculated in that XY space system using standard mathematical formulae.

The way that PRIMER calculates cut sections through the various element types is described below.

Cutting through 3D (solid and thick shell) elements

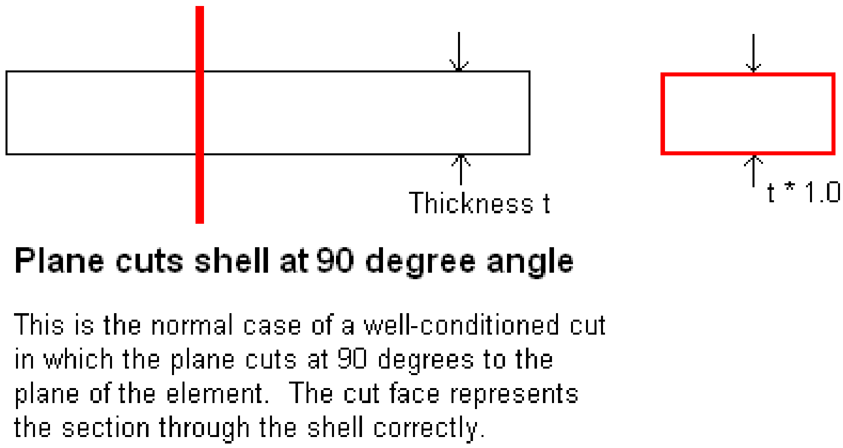
This is simple: the polygon generated where the plane intersects the element is used unconditionally. It will have between 3 and 6 sides depending on the location and orientation of the cut.

This example shows a plane cutting an 8 noded "brick" element at some oblique angle.



Cutting through 2D (thin shell) elements

Where the cutting plane intersects the shell cleanly at 90 degrees to its plane, the well-conditioned case, the cut plane through the element gives a good representation of the true element thickness.



However when the plane cuts the plane at an oblique angle as shown in the images below the calculation of the cut place depends on the "2D and 1D section cut"setting in the **Options** panel.

This has two possible settings

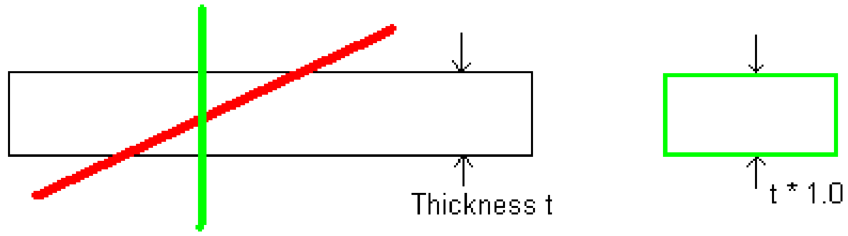
- **90 degree cut**, in which the cut surface calculated is always that of the well-conditioned orthogonal cut, which is the default because it is a "safe" option when computing section properties.
- **True cut**, in which the actual surface generated by the intersection with an oblique plane is computed. This is useful for visualisation of actual geometry, but will tend to over-estimate section properties.

These two options are illustrated below:

90 degree cut
(default)

In this case the effective cut plane through the element is always orthogonal, ie at 90 degrees to its plane, regardless of the actual angle at which the section cuts the element.

This is a "safe" method when calculating section properties, since otherwise a very oblique cut through a shell can result in a an area much deeper that its actual thickness which, in turn, could lead to the section properties being over-estimated.



Oblique case, 90 degree cut option

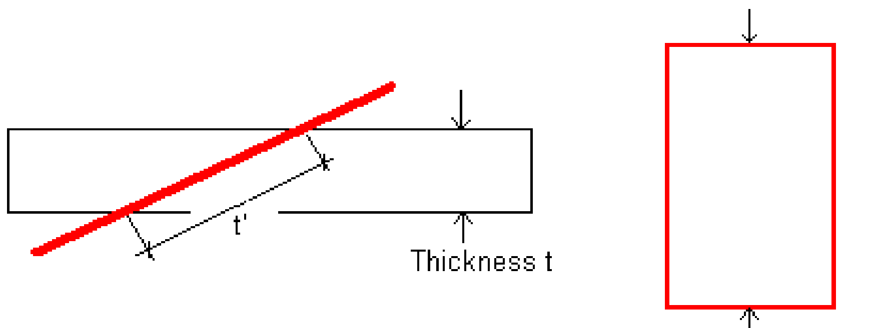
Even though the cut section (red line) cuts the element at a shallow angle the cut surface is computed at 90 degrees to the plane of the shell (green line).

This is a "safe" solution since it will not over-estimate section capacity.

"True" cut

In this case the true intersection between plane and element is computed, and for oblique cuts this can give apparent sections much deeper (t') than the element thickness (t) as shown here.

This can be useful when you wish to visualise the genuine cut geometry, but it is strongly deprecating if you are making use of the section properties calculated by PRIMER as it is likely to lead to their being an over-estimate of the true capacity.



Oblique case, "true cut" option

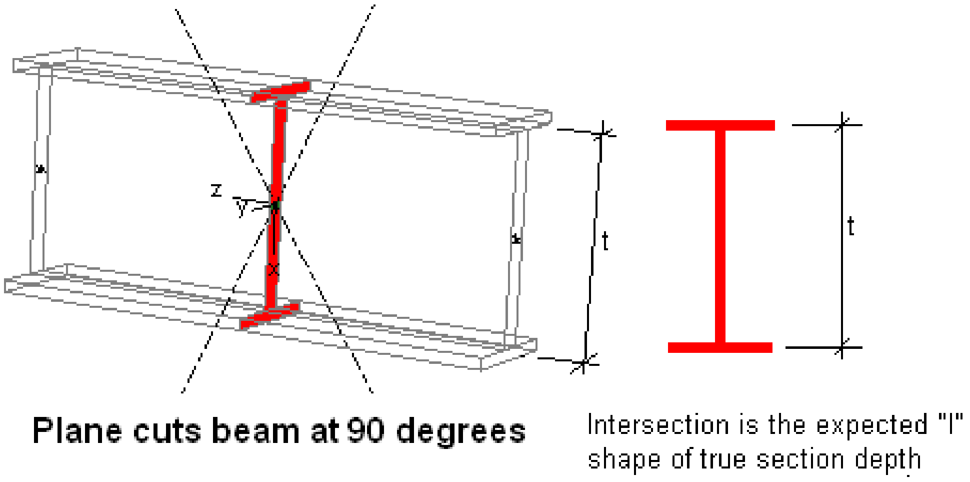
The actual cut surface is used, giving a "deeper" section as the cut becomes more and more shallow.

This can over-estimate section capacity, so it should not be used for calculating properties, but it is sometimes useful to be able to visualise the true intersection geometry.

Cutting through 1D (beam) elements

In order to calculate the intersection PRIMER expands beams to their "true" 3-dimensional section shape, and then cuts through that using the same "90 degree" or "true" cut rules that are used for shells.

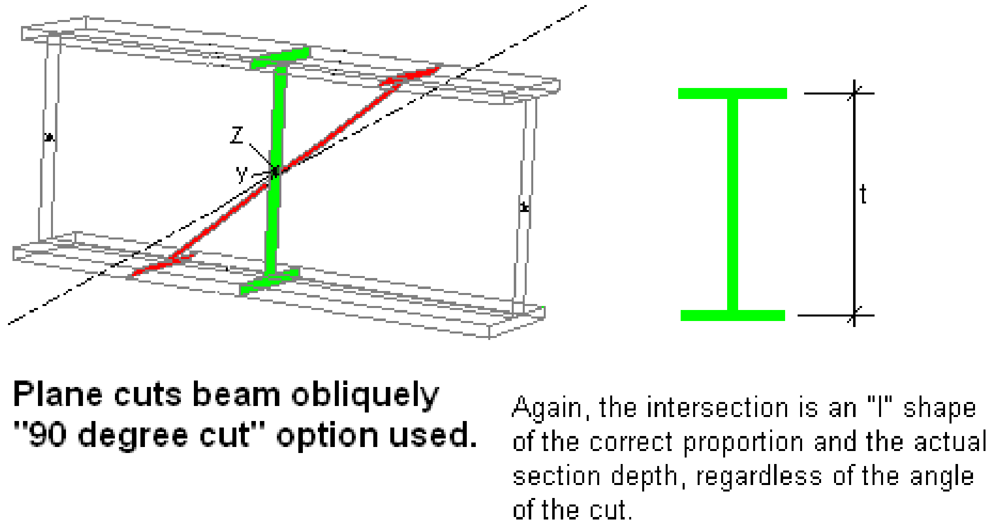
(Note that by default PRIMER does not show true beam sections in the main graphics window, but this can be turned on using the **Display Options** panel. However the cut-section properties display window will always show the true beam shape, as this is required in order to compute properties correctly.)



90 degree cut option

Calculates the section at exactly 90 degrees through the element, regardless of the actual angle of intersection between beam and plane.

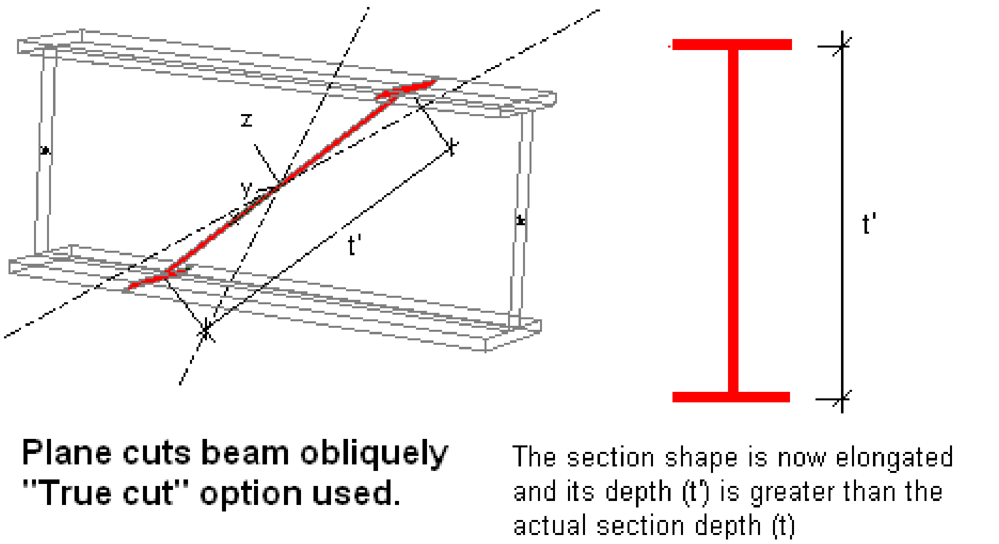
This is a "safe" option when calculating section properties, and hence the default.



True cut option

Used the actual intersection between beam and plane, resulting in the section becoming taller as the cut angle gets shallower, as in this example.

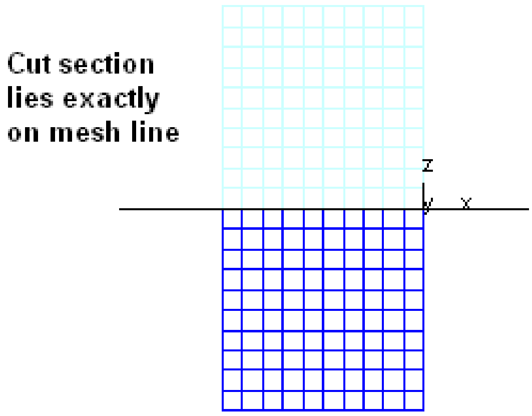
This is useful if you need to visualise the true geometry of the cut plane, but it should not be used when calculating section properties as it will over-estimate the section capacity.



How ill-conditioned cuts are handled

It is normally the case that the cutting plane intersects elements cleanly, leaving no doubt about which elements are being cut. However where a mesh is rectilinear, and the cut plane is positioned exactly on a line of nodes, then a problem can arise as it is not clear whether the plane:

1. Lies in the gap between adjacent rows of elements, and doesn't cut anything
   or
2. Cuts elements both above and below the plane
   or
3. Only cuts elements on one side of the plane



Clearly (1) is not satisfactory, and (2) would be dangerous since it would tend to double the true properties, therefore (3) is adopted as follows:

- For each cut element a check for ill-conditioning, as in cut exactly at node (or nodes) location is made.
- Where this is found to be the case the cut plane is temporarily shifted by a very small amount in its +ve Z (outward normal) direction, and the cut calculation is repeated.

The effect of this is always to consider the element on the +ve Z side (in plane local axes) of such a cut, and to ignore the element on the -ve Z side. In the image here this means that the light blue elements would be considered, and the dark blue ones ignored.

This solution is safe, but it may not always give the result that you want. The best solution is to avoid this problem by moving the plane so that it cuts near to the centre of elements, rather than at mesh lines, since in this way you control explicitly which elements are cut.

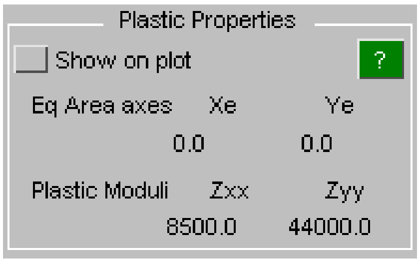
Calculation of Elastic Section Properties

Note that X and Y axes here are the cut-section local (X,Y) plane, and the centroid position is given relative to the origin of the plane.

The following engineering properties are calculated:

Total **Area**	The sum of all cut-section polygon areas.
Geometric centroid (**Xc,Yc**)	The result of the 1st moment of area about X and Y axes, divided by the **area**
2nd moments of area **Ixx, Iyy, Ixy**	In each case the sum of the local I value for each polygon, + its area * distance squared from the relevant axis.
Principal 2nd moments of area **Iuu** (max) and **Ivv** (min), and the angle **theta** between **Iuu** and **Ixx**	The result of transforming the tensor [Ixx, Iyy, Ixy] using Mohr's circle to give principal values and the angle between **Iuu** and **Ixx**.

Calculation of Plastic Section Properties

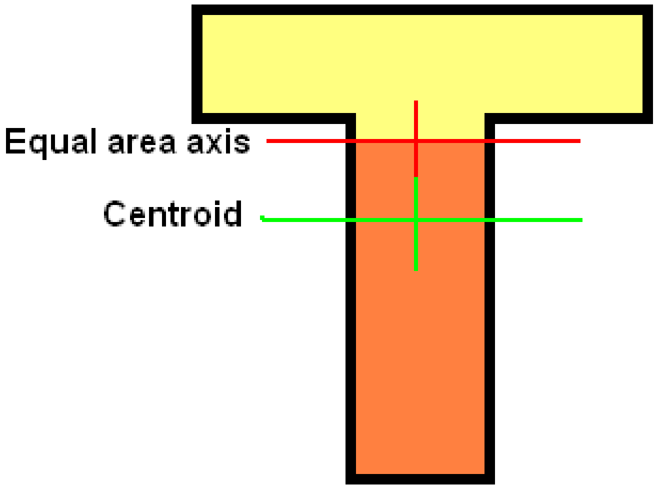


The following plastic properties, commonly used by structural engineers, are calculated:

Equal area axes (**Xe,Ye**).

These are the axes which give equal areas about X and Y respectively.

For an unsymmetrical section such as the Tee shape here the equal areas axis will not lie on the centroid in the depth axis. It is located where the areas above and below it, drawn here in yellow and orange, are equal.



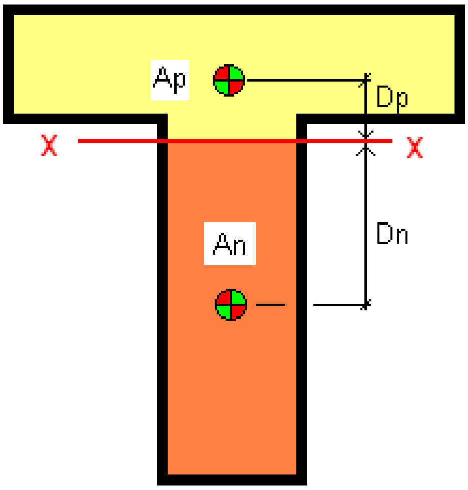
Plastic moduli (**Zxx,Zyy**) [sometimes **Sxx,Syy** in North America]

These are the sum of "area * distance from centroid of area to equal areas axis" about X and Y respectively.

This example shows the computation of the **Zxx** value (**XX** is the horizontal equal area axis here) for the Tee section above.

The plastic modulus **Zxx** about equal area axis **XX** is given by:

Ap * Dp + An * Dn



Where:

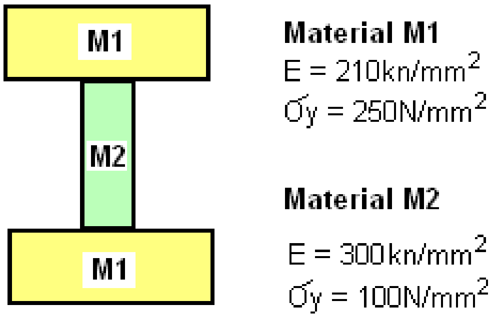
Ap = Area on +ve side of XX axis (yellow) } these areas are
An = Area on -ve side of XX axis (orange) } equal in value
Dp = Distance from centroid of area **Ap** to **XX**
Dn = Distance from centroid of area **An** to **XX**

If all elements in the section have the same yield stress then multiplying the modulus by the yield stress gives a crude approximation of the fully plastic bending capacity of the section. See also the Force and Moment calculations below which may give more useful results for a typical section made up of multiple material types.

Calculation of "First Yield" capacity of the section

This gives the axial force and bending moments at which the first part of the section to reach yield stress is exactly at that stress, taking into account all the different material and element types that comprise the section.

To illustrate this consider the following imaginary section with two different material properties, M1(yellow) and M2 (green), that have different Young's Modulus (E) values and yield stresses.



In all cases we use the elastic properties (area, Ixx, Iyy) and make the assumption that plane sections remain plane, ie that the distribution of strain through the section depth is linear. In the following images the distribution of strain through the section is shown, annotated with the corresponding stresses.

Please read the warnings below before using these values.

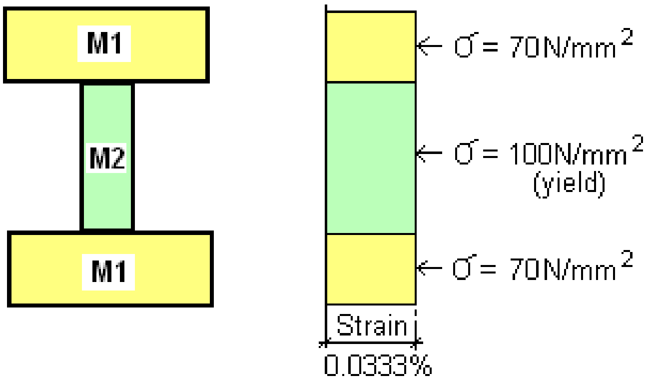
Axial case.

Here the strain in the section is constant through its depth, and first yield occurs when the the weaker material M2 reaches yield at 0.0333% strain.

M1 has an E value of 210kN/mm2, so at 0.0333% strain its stress is 70N/mm2

The axial force capacity of the section is then simply the sum of stress * area:

```
axial force = (area of M1 * 70)
            + (area of M2 * 100)
```



Axial Case

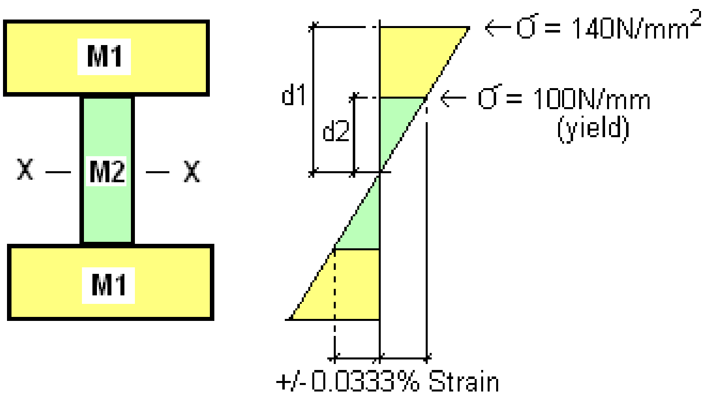
Mxx Bending case

Here the limiting strain of 0.0333% is still controlled by the outer fibre of material M2 at depth d2 from the XX axis, which reaches yield first.

However the linear strain distribution through the section means that the outer fibre of material M1 reaches about 0.0666% strain (assuming d1 = 2 x d2), so the peak stress in material M1 is about 140N/mm2 .

Therefore the bending moment capacity of the section about its XX axis, **`Mxx`**, is, obtained from the standard formula **`M/I = stress/y`** giving:

```
Mxx = (Ixx for M1 * 140 / d1)
    + (Ixx for M2 * 100 / d2)
```

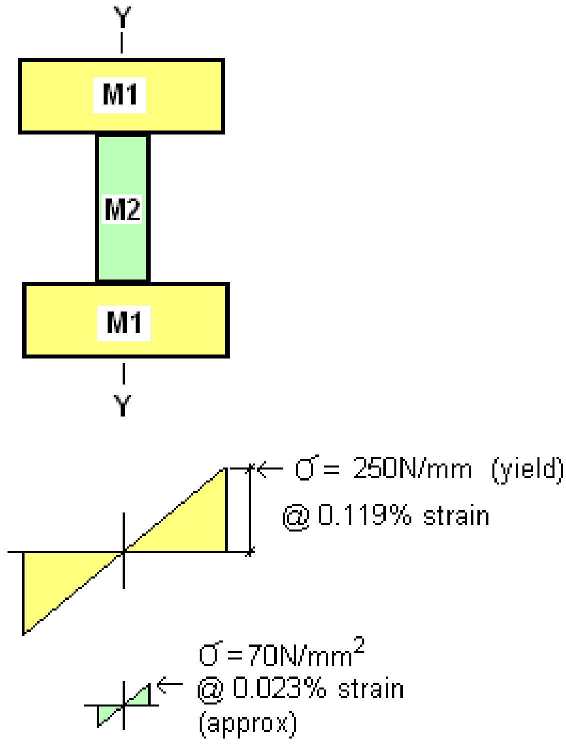


Mxx Bending Case

Myy Bending case

This is calculated using the same method as **`Mxx`**, but now the narrowness of the web means that first yield is reached at the outer fibres of material M1 at a strain of 0.119%.

The strain in the outer fibre of the web, material M2, will be about 20% of this, approximately 0.023%, giving an outer fibre stress in M2 of about 70N/mm2.



Myy Bending Case

Warnings about the "First Yield" calculation

1. It will be clear from the calculations above that both Young's Modulus (E) and yield stress must be available for every material in the section if this calculation is to be valid. These values are obtained from the material (*MAT) cards, but for some material types - especially brittle and crushable ones - either or both may not be well defined. Also it is possible for that the relevant material cards may not be present in the deck.

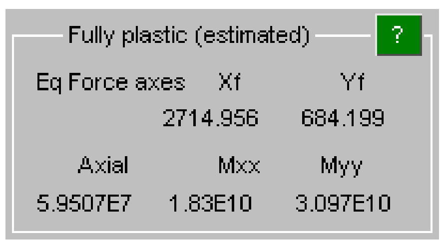
   Please see the **Options** panel below to see the alternatives that PRIMER uses when either of these properties are missing. Regardless of the choices you make there if any element in the section cannot extract either E or yield stress from the material card then the results will be marked as "Estimated".

2. This is a purely elastic calculation and, in the case of Mxx and Myy, assumes that all materials in the section behave symmetrically in tension and compression in the elastic regime, and have the same yield stress in both tension and compression. For ductile materials, eg steel, this is likely to be true; however for brittle materials (eg concrete) yield in tension may be very different to yield in compression.

3. It is assumed that each cut element is homogeneous with a single E value and yield stress. This will not be the case for composites, made up of layers of different materials; nor may it be a valid assumption for orthotropic materials.

Please consider the sections being cut through when you use this feature, and satisfy yourself that the calculation is valid for your model.

Calculation of Fully Plastic capacity of the section



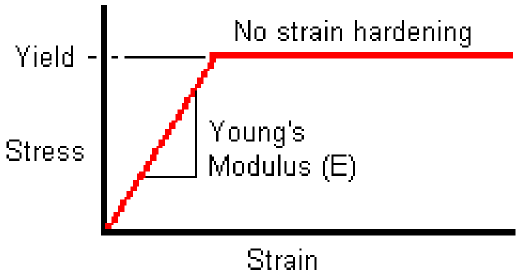
These values calculate the axial force and bending moment capacity of the section assuming that all materials are at their yield stress.

This is very similar to the calculation of plastic moduli, except that the actual yield stresses of all materials in the section are used, making it possible to estimate the plastic capacity of a section comprising multiple different material types.

Instead of calculating "equal area" axes PRIMER now calculates "equal force" axes in which the force (the sum of area * yield stress for all cut elements) is equal on both sides of the axis.

All materials are assumed to behave in an "elastic / perfectly plastic" stress strain curve, symmetrical in tension and compression. Since plane sections must remain plane it is necessary that each material be able to maintain a constant yield stress over a wide range of strain values, hence the requirement for no strain hardening.

The illustrations of the plastic capacity below use the same section as used in the "First yield" examples above, but note that the images now show the distribution of ***stress*** through the section depth (rather than strain above).

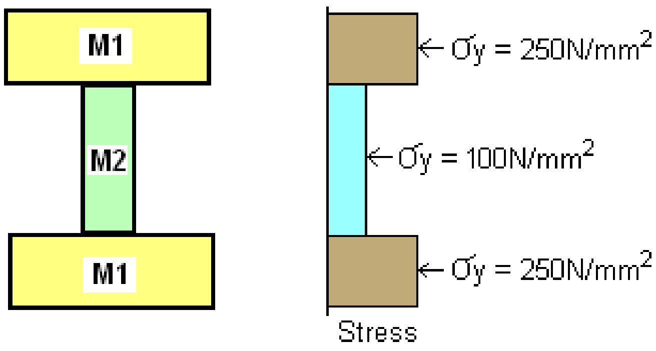


Elastic / perfectly plastic Stress / strain curve

Fully Plastic axial capacity

Each material is at yield, so the axial force capacity of the section is then simply the sum of yield stress * area:

```
axial force = (area of M1 * 250)
            + (area of M2 * 100)
```

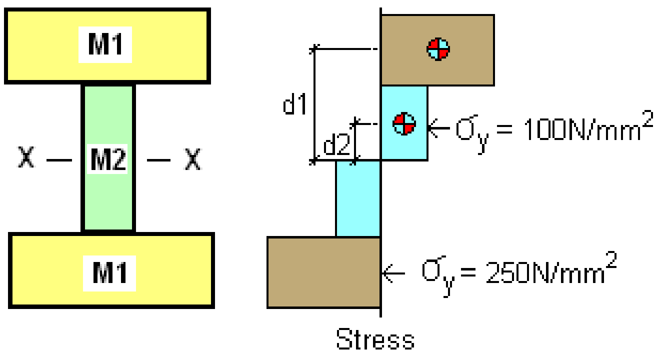


Axial case: Fully plastic

Fully plastic Mxx bending capacity

Again each material is at +/- yield stress, so the total bending capacity ois given by the sum of (area * yield stress * distance from centroid to XX axis) for all cut sections. So in this example

```
Mxx = (Area M1 * 250 * d1)
    + (Area M2 * 100 * d2)
```

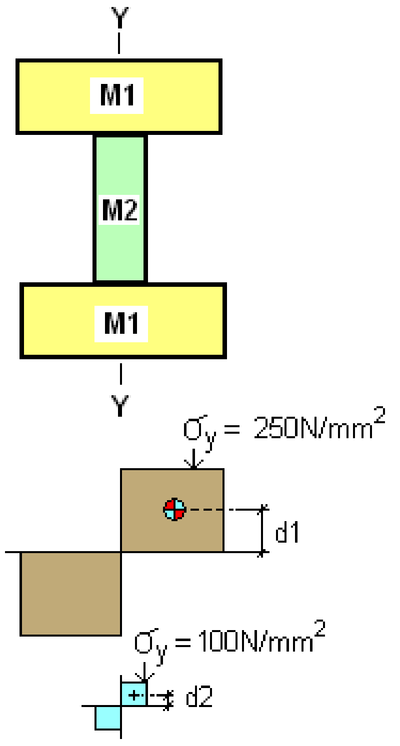


Mxx Bending Case: fully plastic

Fully plastic Myy bending capacity

Essentially the same calculation as Mxx. Each material is at +/- yield stress, so

```
Myy = (Area M1 * 250 * d1)
    + (Area M2 * 100 * d2)
```



Myy Bending Case: fully plastic

Warnings about the "Fully Plastic" capacity calculation

1. These calculations require the yield stress of every material in the cross section. These values are obtained from the material (*MAT) cards but for some materials, for example crushable or brittle ones, a yield stress may not be well defined. In this situation PRIMER has various options for determining a yield stress, see the **Options** panel below.

2. Fully plastic calculations like this are inherently unrealistic since the vast majority of real materials do not exhibit "elastic / perfectly plastic" behaviour. Moreover LS-DYNA material models tend to define quite complex post-yield stress/strain characteristics, all of which are ignored here, so these values should not be considered to be anything more than a crude estimate of plastic capacity.

3. In the case of bending (Mxx and Myy) these calculations assume symmetrical yield behaviour, with the same yield stress in tension and compression. This may be reasonable for ductile materials (eg steel) but can be hopelessly wrong for brittle ones (eg concrete).

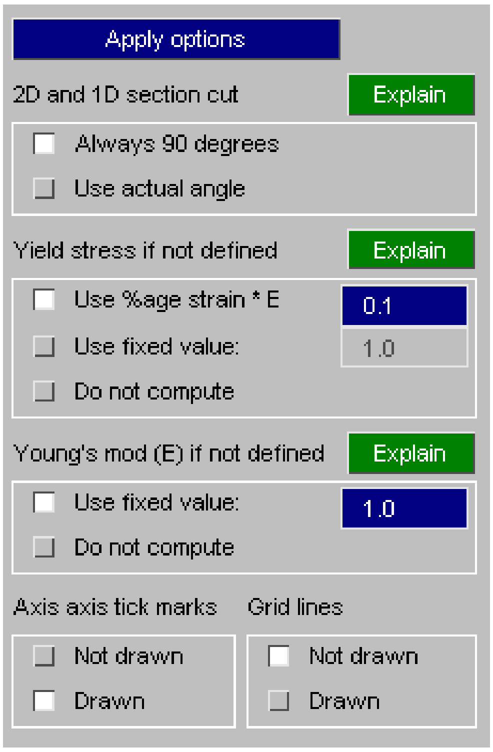
4. It is assumed that each cut element is homogeneous with a single yield stress. This will not be the case for composites, made up of layers of different materials; nor may it be a valid assumption for orthotropic materials.

Please consider the sections being cut through when you use this feature, and satisfy yourself that the calculation is valid for your model.

Options

Controlling calculation and plotting.

2D and 1D section cut



Controls how 2D (shell) and 1D (beam) sections are generated when the angle between element and cutting plane is not orthogonal.

- **Always 90 degrees** gives a cross-section that cuts the element at right angles to its in-plane axis, giving a "safe" shape for calculating section properties.

- **Use actual angle** calculates the actual area cut through the element, which can be useful if you need to visualise this correctly. However this is not recommended for section property calculations as it can result in over-estimates of area and hence capacity.

The effects of these two options on shells and beams is illustrated above for shells and beams.

Yield stress if not defined

When calculating both First Yield and Fully Plastic section capacities PRIMER needs to know the yield stress of all cut element materials, and it will normally extract this from their material (*MAT) cards.

However that card might not be present, or for some more exotic material types it may not be possible to calculate a yield stress, in which case some other value must be used, and the following three options are provided:

Use %age strain * E	Calculates a yield stress by multiplying the Young's Modulus (E) by a %age strain value.
Use fixed value	You define a yield stress to be used if a value cannot be found.
Do not compute	Elements for which a yield stress cannot be found are omitted from the calculation.

Young's Modulus (E) if not defined

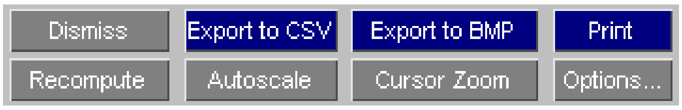
When calculating First Yield section capacity PRIMER needs to know the Young's Modulus (E) of all materials so that it can compute the stress at a given strain. As with yield stress this value is normally extracted from the material card of the cut element, but this may be missing or the material may not have a well-defined E value. When this value cannot be found you have the following options:

Use fixed value	You define a Young's Modulus value to be used
Do not compute	Elements for which an E value cannot be found are omitted from the calculation.

Axis tick marks and grid lines

This control whether or not tick marks showing the dimensions of the cut-plane coordinate system are shown on the image, and also whether or not grid lines are drawn between these tick marks.

Image Capture options



The information on the cut-section properties panel can be captured both numerically and graphically as follows:

Export to CSV

Sends the numerical information at the bottom of the panel to a Comma Separated Variable (.csv) file in a format suitable for import into a spreadsheet. This is an ASCII text file so it is also suitable for import into any external programme, and can also be read by humans.

Its format should be self-explanatory from this example:

```
Cut section properties for model 2
Section origin,           0.000000e+000, 0.000000e+000, 0.000000e+000
X axis vector,            0.000000e+000, 1.000000e+000, 0.000000e+000
Y axis vector,            0.000000e+000, 0.000000e+000, 1.000000e+000
Z axis vector,            1.000000e+000, 0.000000e+000, 0.000000e+000
Cut area,                 2.968603e+005
Cut centroid Xc Yc,       2.498571e+003, 5.909979e+002
2nd moms Ixx Iyy Ixy,     4.275801e+010, 1.890375e+011, 2.316741e+010
2nd moms Iuu Ivv Angle,   1.926190e+011, 3.917650e+010, 8.121208e+001
Equal area axes Xe Ye,    2.714534e+003, 6.853174e+002
Plastic moduli Zxx Zyy,   9.119786e+007, 1.544448e+008
1st yield Axial Mxx Myy,  5.950707e+007, 9.546364e+009, 1.685284e+010
Equal force axes Xf Yf,   2.714956e+003, 6.841990e+002
Eq force Axial Mxx Myy,   5.950706e+007, 1.830389e+010, 3.096836e+010
```

Export to BMP

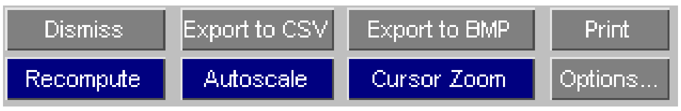
Create a bitmap (.bmp) file of the panel contents. This format has been chosen since it is readable by any 3rd party graphical software, and because it does not use data compression the quality of the lines and text on the image is not degraded.

Note that this panel, as with any sub-window in PRIMER may also be copied to the system clipboard by using the "Copy->Clipboard" option in the drop-down menu under the `[-]` button in the top left corner of the window.

Print

Where supported this will send a copy of the panel to the printer.

Image manipulation options



Recompute

Recalculates and redraws the current image, taking into account any changes tothe model that may have occurred since the results were last updated.

Moving the cut-section, blanking elements or changing entity visibility does not result in an automatic update of this panel, and Recompute should be used to see the results of any such changes.

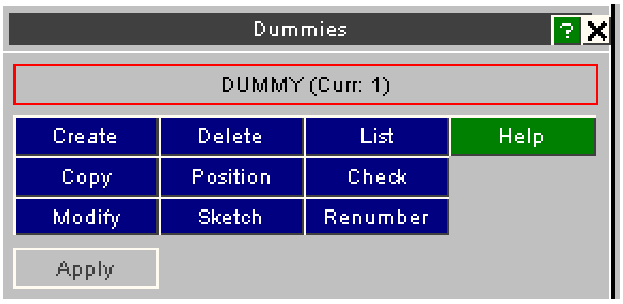
Autoscale

Redraws the image autoscaled to the current menu panel size. Use this to return to the original image size following a zoom operation, or if you have resized the menu panel and the image no longer fits properly.

Cursor zoom

Click on this to enable cursor zooming in this panel. When selected you can drag out a rectangular area with the cursor, which will be enlarged to fill the window.

6.12 DUMMIES Positioning Occupants



Dummies positioning capabilities have been enhanced In PRIMER release 9.3. The main changes are:

- Dummy definitions may be created and edited interactively
- Positioning has been enhanced to include a "free dragging" mode.
- "Points" may be added to dummy assemblies to give further restraint and positioning options.
- Dummies may be "child" definitions of general mechanisms, and positioned in conjunction with those mechanisms.
- Dummy positions may be saved and restored, and also interchanged between models via an independent positions file.
- Dummies may be positioned from the command-line and thus also in batch mode.

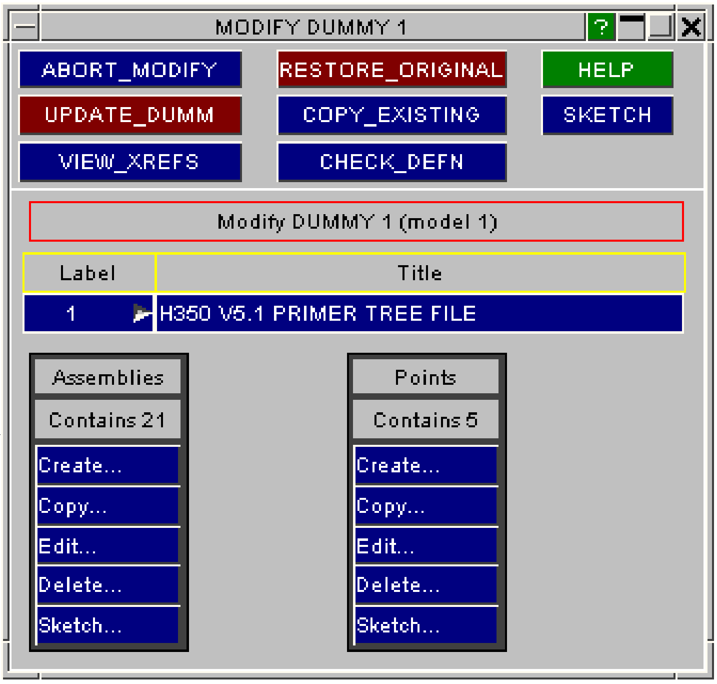
What is a "dummy"?

A "dummy" is simply a collection of ordinary nodes, elements, materials, etc. but with the extra property that a *DUMMY definition has been created for it. This is a special set of additional data which defines how the parts of the dummy are connected together, and in what order. It is described fully in Appendix II, but the key points are:

- A dummy is made up of a series of "assemblies", each of which contains one or more parts and node sets. For the purposes of positioning each assembly is assumed to move as a rigid body, although its definition may include both rigid and deformable components for the purposes of analysis. There is a limit of 100 assemblies per dummy.
- Assemblies are connected together in a hierarchy, in which "parent" and "child" relationships are strictly defined. For example the torso is "parent" to the upper leg, which in turn is "parent" to the lower leg, which is "parent" to the foot; thus the foot is a "child" of the lower leg, and great-grandchild or the torso.
- Generalised Stiffness definitions between assemblies define the (local) axes a child may rotate about with respect to its parent, and a node must also be specified which gives the position on the parent about which the child rotates.
- The "root" part, generally the lower torso or pelvis, is assumed to rotate about the dummy "H-Point".
- Dummies may have a coordinate system defined, in which the angles of the "root" part are expressed, if none is defined then the dummy is assigned a system which is initially aligned with the global cartesian system. This system will be rotated as the dummy as a whole is rotated.

Dummies may constitute separate models, or may be part of a complete structural model, it makes no difference. **`*DUMMY`** definitions are appended to the file after the **`*END`** keyword, and so are ignored by the analysis code.

6.12.1 Creating and Editing dummies



Dummy definitions contain Assemblies and Points.

Assemblies are collections of one or more parts, which may be any permutation of rigid or deformable, that make up body components (torso, head, limbs, etc.).

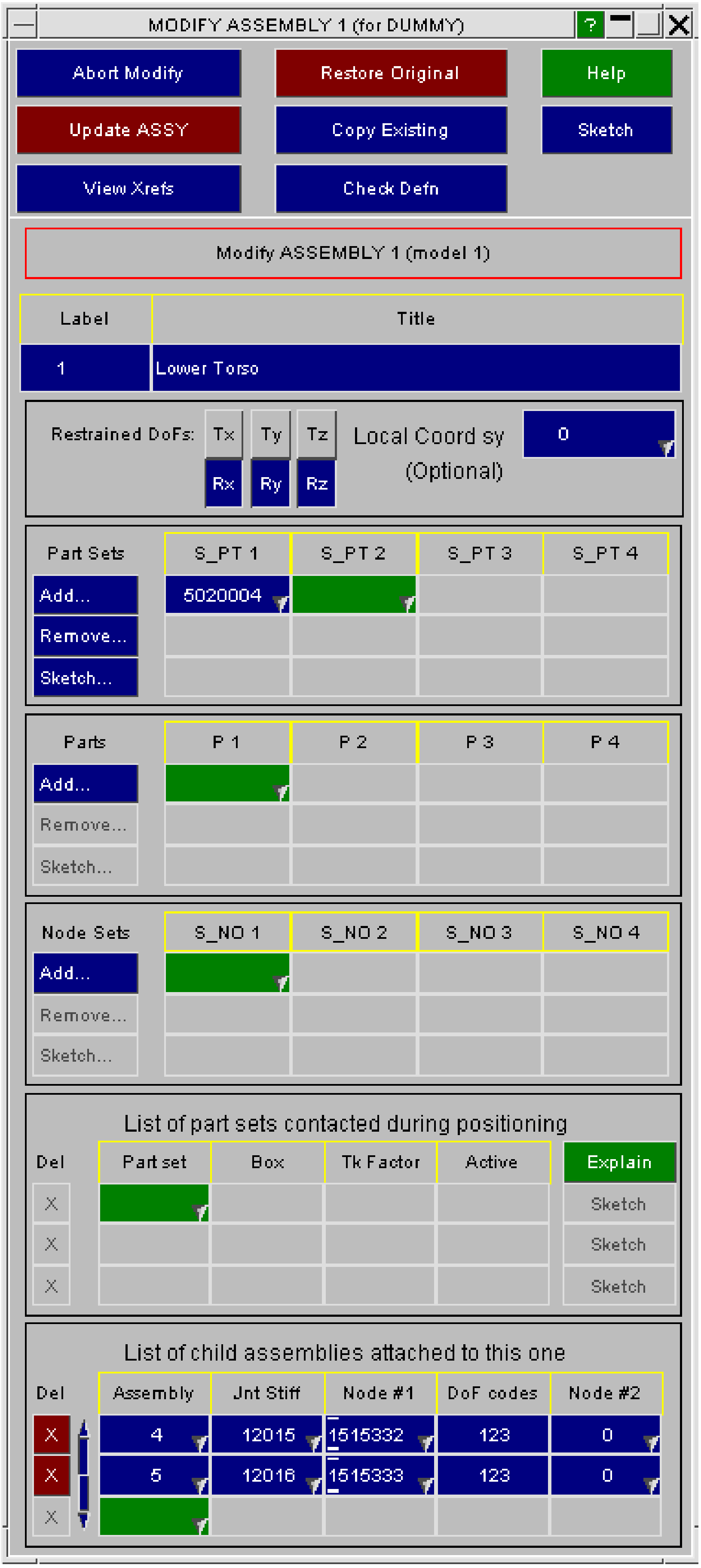
Points are optional, and any number may be defined. They are coordinates in space, "tied to" and a property of their parent assembly, that may have restraints in any combination degrees of freedom. If a local coordinate system is defined for a point then any restraints act in that system.

Points may also be used for positioning: if a new coordinate if specified for a point then the "free drag" positioning algorithm will move the dummy accordingly.

Use **Create...**, **Copy..., Edit...**, etc. to select and operate on the relevant items.

When the definition is correct use **UPDATE_DUMM** to save it.

Assembly Creation and Editing



Label and title

An assembly must have a label. Labels are "private" to this dummy, thus in a model with 2 dummies each may have assembly label 1 and they will not clash.

A title is optional but is recommended.

Restraints and coord system

Assemblies may have optional restraints. These are used during "free drag" positioning to constrain movement.

If a coordinate system is defined restraints are in that system, otherwise they are in the global system.

Part sets and parts

An assembly must contain at least one part, and any number may be used in any permutation of explicit parts and part sets. It is legal to define a part more than once (e.g. in a set and explicitly): it will only be used once.

See "Good modelling Practice" below for some further rules and suggestions.

Node sets

Assemblies may also contain node sets. These can be a useful way of "associating" nodes with an assembly since all nodes in the set will be moved with it.

Contact

Contact between dummy assembly and structure.

Any number of part sets of "structure" may be specified, with an optional box to limit the volume of space in contact.

Tk factor is a factor on true thickness used for contact, and Active enables contact to be turned on/off at will. (It can also be switched on/off on the positioning panel.)

This is not a true *CONTACT definition, but rather a pseudo contact used during positioning only.

Child assemblies

Dummies assume an explicit parent -> child hierarchy, and an assembly may have any number of children (or none).

Child assemblies are connected at a point (node #1) on the parent. Rotation axes are defined by one of:

- A joint stiffness (*Constrained Joint Stiffness) giving local axes and stop angles.
- A second node #2, limiting rotation to about the axis N1N2.
- Neither stiffness nor N2, leaving rotation axes implicitly global.

The <DoF code> is also used at joints which only permit rotation about a single axis to limit rotation to that axis: 1 = about X, 2 = about Y, 3 = about Z

Good Assembly modelling practice

When defining assemblies the following rules should be borne in mind:

- **It is illegal for a part to be defined in more than one assembly.**

  The reason is obvious: since moving an assembly will move a part then having the part in two assemblies would cause conflicting movements at the assembly nodes. This will be detected and flagged as an error during the pre-positioning check.

- **It is illegal for a node in assembly A also to be defined in assembly B - with three exceptions.**

  The reason is the same as for parts above: a node may not have more than one possible source of motion, otherwise its position will be updated wrongly. This will be detected during pre-positioning checks and flagged as an error.

  However there are three important exceptions to this rule:

  1. **It is legal to have "extra nodes" on rigid parts in assembly A when their parent part is in assembly B.**

     This is a special case that is detected during the pre-positioning checks, and the normal PRIMER logic that moving a rigid part also moves any "extra" nodes is not applied in this case. This permits joints to articulate during positioning, but then to be locked during analysis in their "as positioned" configuration. (However where extra nodes on a part are in the same assembly as the part they will be moved in the normal way.)

  2. **It is also legal to have a slaved rigid part (from a rigid body merge) in assembly A when its master part is in assembly B.**

     This is another special case, like "extra nodes" above, that is detected and handled differently during positioning. This is again to permit articulation during positioning but locking during analysis.

  3. **Parts made of null shells (Material type 9) may legally cover multiple assemblies, but must not be defined in more than one.**

     Many dummies are covered by a "skin" of null shells that allow contacts to track across the gaps between the various assemblies. Inevitably these will include nodes common to assemblies A and B at such a gap, and may in fact extend further over the dummy model so that a single null shell part covers several assemblies.

     The pre-positioning check detects these parts, which must be shells referencing material type 9, and excludes them from the positioning algorithms - effectively making them non-structural in this context. But see (4) below for an exception.

     However if any nodes on these null shell parts are not also "structural" nodes on assemblies their positions will not be updated during positioning. This would be bad modelling practice anyway since all nodes on null shell parts need to be attached to real structure in order to give them stability during the analysis and stiffness for the contact.

  4. **An exception to an exception (3) is made for "isolated" null shell parts, which *are* moved with their assembly**

     Experience has shown that some modellers include target markers on their dummies made up of pairs of null shell parts, and they attach these using tied contacts. Exception (3) above resulted in these target markers being left behind when an assembly was moved because they did not share any structural nodes with the assembly.

     Therefore the "null shell" test has been extended and these parts ***are*** moved with the assembly if they do not share any structural nodes with it.

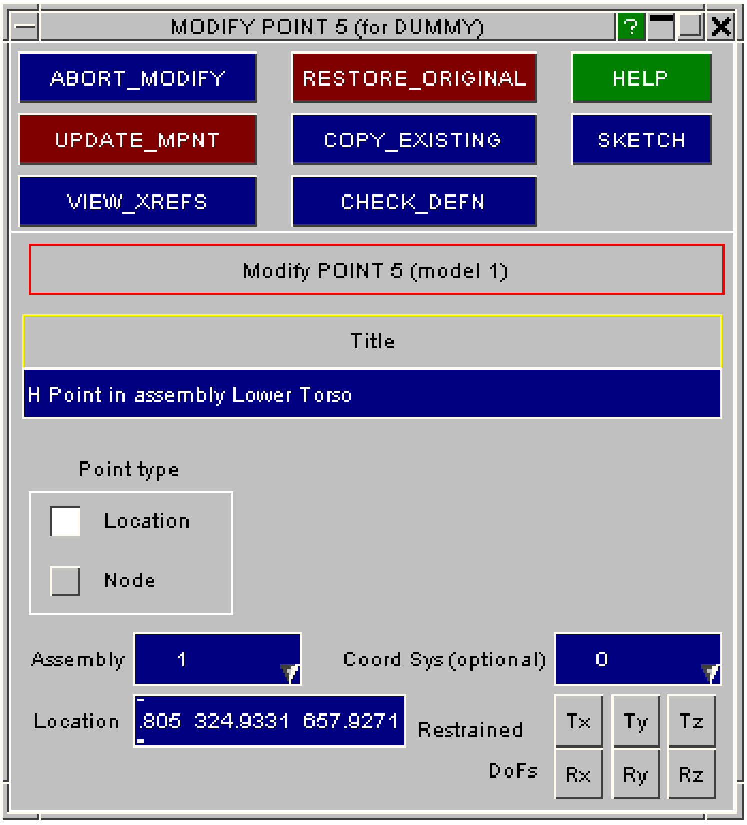
The "is it isolated?" test needs to be quick as it is invoked during dragging operations, so it is not fool-proof and sufficiently devious modelling practice may defeat it. Users taking advantage of this should ensure that the null parts used for "isolated" parts are unique to their assembly.

- **Coordinate systems used for Joint Stiffness definitions must be defined by nodes in the relevant assembly.**

  When defining coordinate systems for Joint Stiffnesses it is important that these use nodes (*Define Coordinate _**NODES**) on the assembly to define their axes, otherwise they will not be updated as the assembly moves and the joint axes will not rotate correctly. PRIMER can visualise coordinate systems and joint stiffnesses, but some modellers add a triad of rigid beams at these locations to visualise coordinate systems during post-processing.

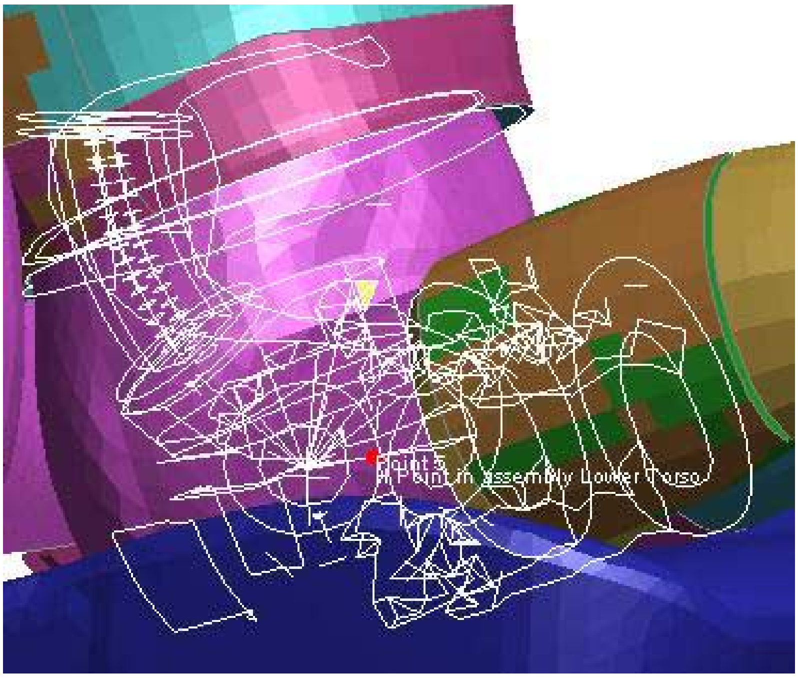
Further hints on good modelling practice may be found in the section on Rules for "tree" files in Appendix II.

Point creation and editing



Title	A title will be generated automatically, but you can supersede this with your own. Points do not have labels
Point type	A point defined by Location is a coordinate in space that is attached to, and moves with, its parent assembly. A point defined by node is essentially the same: it obtains it current coordinate from the node. The node should normally be part of the parent assembly, but this is not mandatory.
Restraints and coordinate systems	A point's movement may be restrained in any combination of degrees of freedom (or none). If a local coordinate system is defined restraints act in that system, otherwise they are global.

Visualising Points



Points may be visualised by using the **Sketch** options both on the parent dummy panel and on their edit/create panels.

Here is a picture of the point in the example above: it is the dummy's H point, located in the centre of the pelvis. (This point is created automatically if the dummy definition includes an H_POINT card.)

It is shown as a circular red symbol, labelled with its title, with the free edges of the parent part also displayed.

Automatic creation of a point at the H-Point.

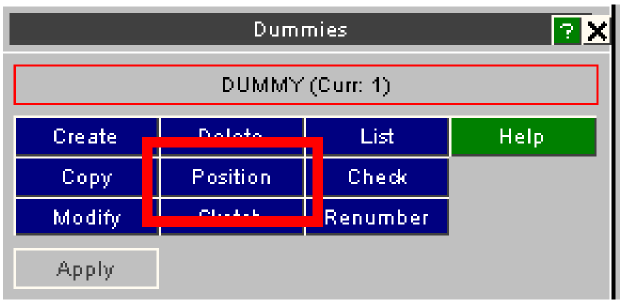
If your dummy definition includes an H-Point then a "point" as described above will automatically be created at that location, attached to the root assembly of the dummy.

Dummy "Tree" files.

The information describing the dummy is saved in special keywords following the *END card.

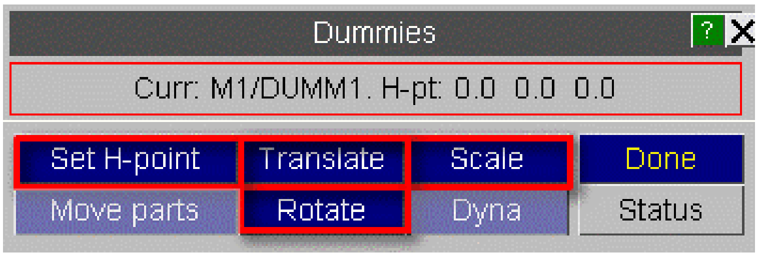
Collectively these data cards are know as a "Dummy Tree file" and their format is described in Appendix II. This appendix also gives further rules that apply to dummy models, and advice about good and bad modelling practice. It is recommended reading if you are creating a dummy model as there are some tried and tested techniques that are known to work ... and equally some common pitfalls!

6.12.2 **Position**: Positioning dummies



Once a dummy has been defined, or read in from file, it can be positioned in a variety of ways.

Operations that apply to the whole dummy as a rigid unit:

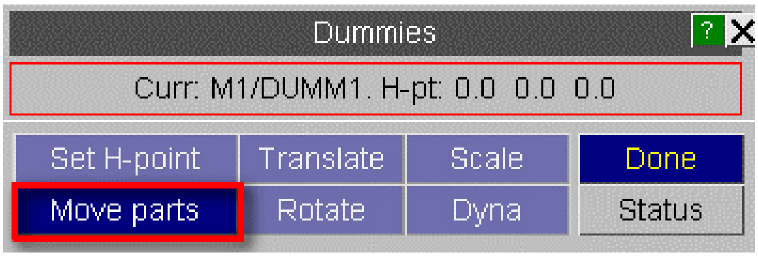


Translate, Rotate, Scale act in exactly the same way as under the (main programme) **Orient** command. The whole dummy is moved to its new position either by explicit commands or by dragging with the mouse.

Set H-Point is simply **Translate** by another method: the dummy is translated by the difference between the current and the required H-point position. However this is often a convenient method when a specific H-Point is required.

It is important to appreciate that these commands move the dummy as a rigid whole, with no articulation of its limbs.

6.12.3 Move Parts: Positioning Dummy Assemblies

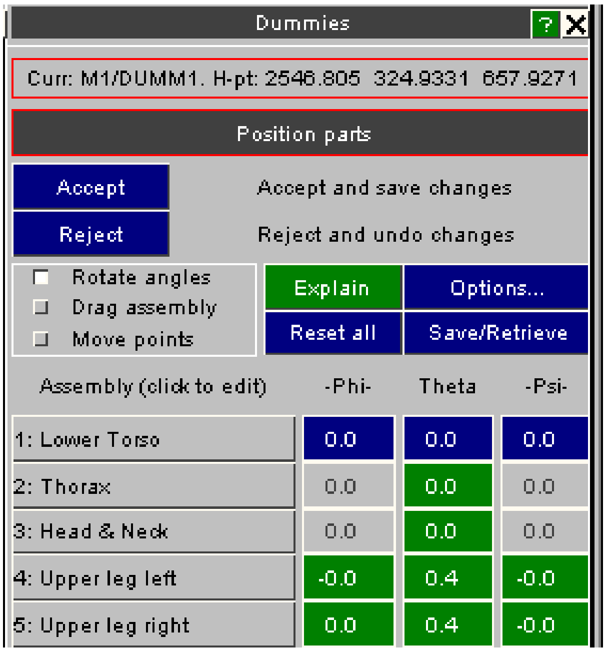


The rest of this section describes the process of positioning the dummy assemblies, i.e.***with*** articulation of its limbs.

When you enter the dummy positioner with the **Move Parts** command several operations are performed:

- Correctness of the dummy definition is checked. Parts and nodes should not appear in more than one assembly, and you are warned if they do and given some options for diagnosing and correcting these errors.
- You cannot have both **Dummy** and **Mechanism** positioning active at the same time in the same model. (This is because of the way positioning data is stored: the two processes would conflict.) If you attempt this you will be forced to shut down one operation before you can start the other.
- The current dummy position is saved as an "initial position". If things go wrong in the positioner you can return to this as any time by using **Reset all**, and if you abort positioning using **Reject** the dummy will automatically be restored to this position.

The main positioning panel



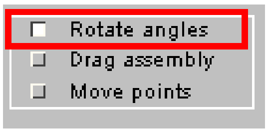
Assuming that these checks pass you then drop into the positioning panel. For dummies this operates in one of three modes:

Rotate angles	In this mode explicit rotation of assemblies about their parent connection node takes place.
Drag assembly	In this mode "free" dragging of the dummy takes place, combining translation and rotation.
Move points	In this mode points can be defined and edited, and "free" movement performed by giving updated coordinates for them.

A dummy is positioned by any combination of these modes, and when it is satisfactory the user must **Accept** it to make the geometrical changes permanent, or **Reject** it to abandon positioning and restore the original geometry.

Further positioning commands below describes these and other options in more detail.

Rotate Angles: Explicit rotation of assemblies about their connection nodes.



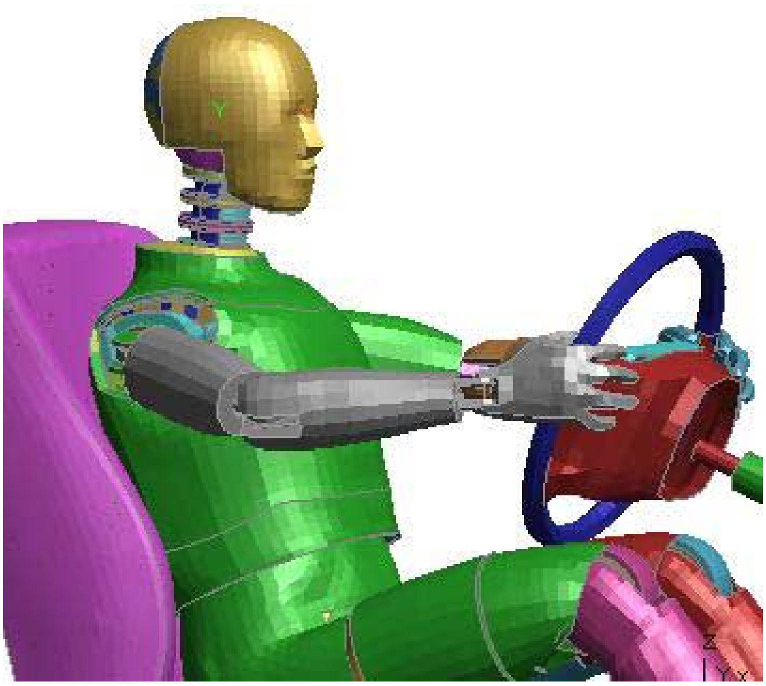
For each assembly a row showing the current joint angles is shown. Angles on a blue background are in the main dummy axes system, those on a green background are in the local system of the joint stiffness connecting this assembly to its parent. Greyed out angles are locked against rotation by the <DoF code> of the assembly.

Assembly (click to edit)	-Phi-	Theta	-Psi-
1: Lower Torso	0.0	0.0	0.0
2: Thorax	0.0	0.0	0.0
3: Head & Neck	0.0	0.0	0.0
4: Upper leg left	-0.2	13.8	-0.2
5: Upper leg right	0.2	14.1	0.2
6: Lower leg left	0.0	0.1	0.0
7: Lower leg right	0.0	0.1	0.0

In this mode an assembly is selected and rotated about one of the three axes. Both the selected assembly ***and its children*** are rotated as a rigid unit, and rotation only takes place about a single axis at a time.

"Euler" angles are used to specify and compute angles in this mode. These are described in more detail under "Euler angles in PRIMER" below.

In the example here the user has selected the upper right arm of the dummy (coloured grey to denote selection), and it can be seen that the elbow, lower arm, wrist and hand have all been selected too.

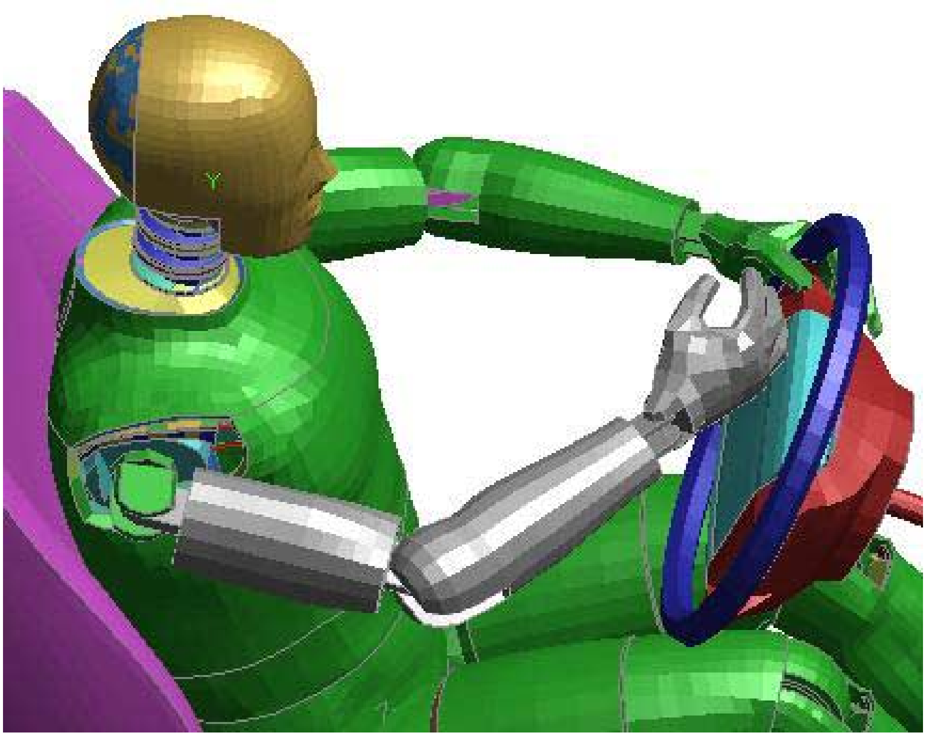


Dragging with the mouse is the easiest method:

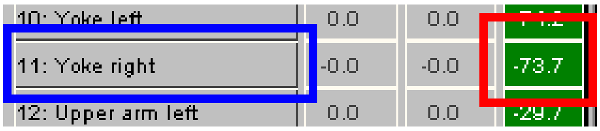
Mouse button	Drags axis
Left	Local X (Phi)
Middle	Local Y (Theta)
Right	Local Z (Psi)

Dragging is only permitted about the axes implied by the DoF Code specified on the parent assembly. In addition rotation will be limited to the stop angles specified on any Joint Stiffness definition for the joint.

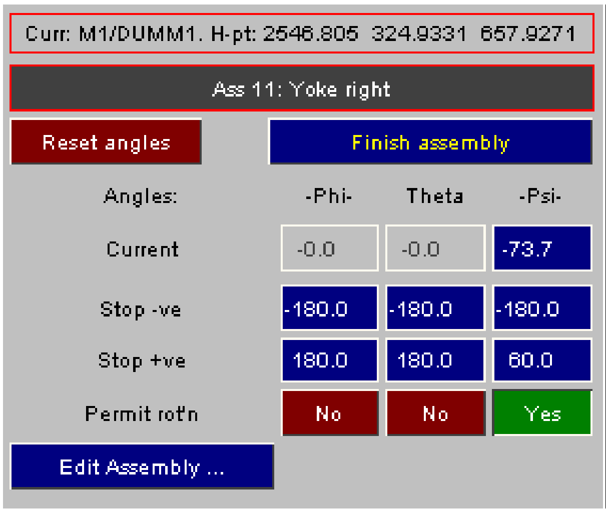
This image shows the arm after some movement, demonstrating how the assemblies below the upper arm in the hierarchy all move as a rigid combination, rotating about the (nearly) vertical axis at the right hand shoulder yoke joint.



Assembly angles can also be set explicitly by typing angles into the appropriate row text entry box (red outline):



Or the full editing panel for an assembly can be mapped by clicking on the "name" button (blue outline). The editing panel allows all the angle attributes of the assembly to be adjusted.

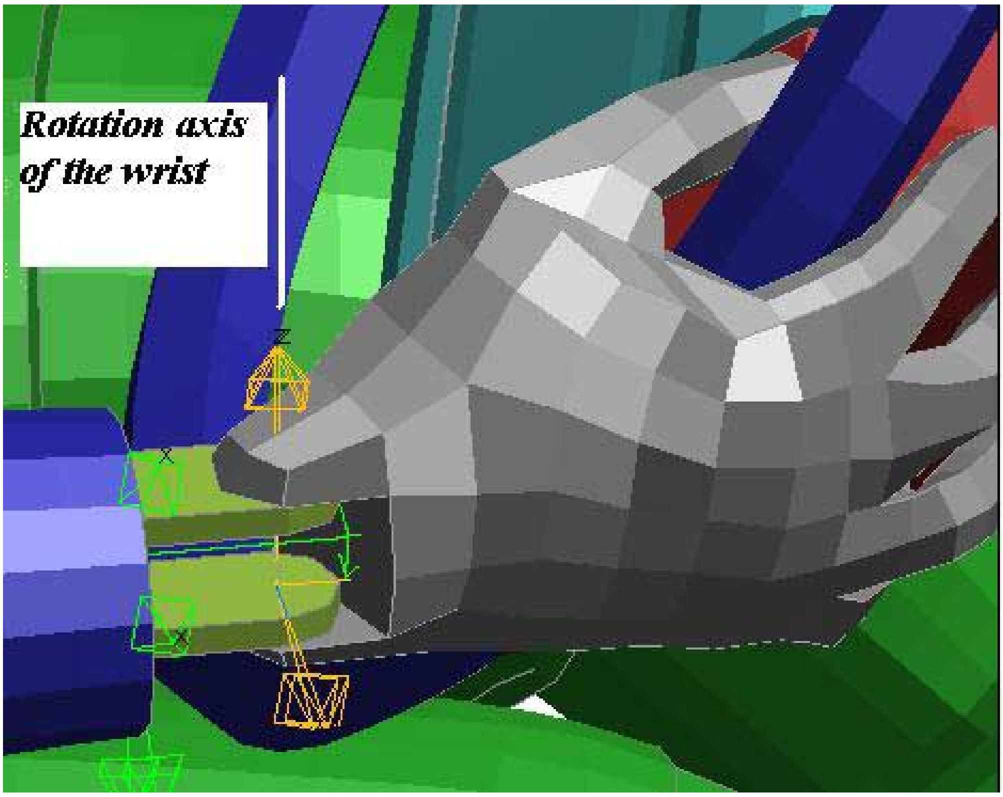


This image shows the axes of explicit dragging more clearly.

Here only the right hand is being dragged, and the display of *Constrained Joint Stiffnesses has been turned on to show the local axis systems at the wrist to hand joint.

Rotation is only permitted about the joint's local Z axis, shown here with a white line superimposed on the image. There are two local axis systems at the joint: one on the wrist (parent) and one on the hand (child). Initially they were coincident.

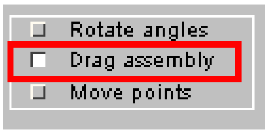
It can be seen from the divergence of the local X axes (coming out of the joint towards the observer) that the hand has been rotated a little.

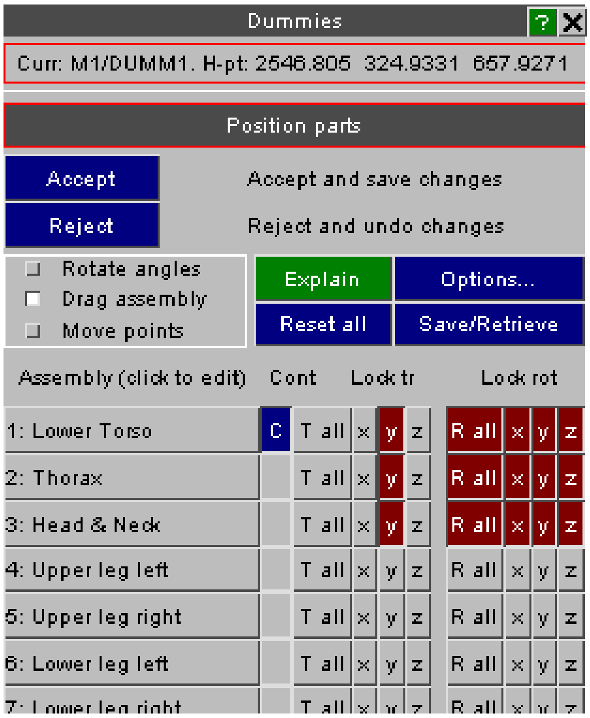


The characteristics of **Rotate Angles** mode:

- **It is precise**: movement is calculated using trigonometry about the rotation point on the parent assembly. This preserves the accuracy of the dummy model and, in particular, preserves the coincidence of nodes at joints.
- **It takes no notice of** restraints on assemblies, or restrained points within them: these are considered in "dragging" modes only.
- **It can be frustrating** to use since positioning of a limb requires a set of rotations about various joints back up its "tree", making precise positioning awkward.

Drag Assembly: Free dragging of limbs using mechanism analysis





In **Drag Assembly** mode the positioning panel changes.

Each assembly is still shown as a row, but now:

- Clicking on the "name" button brings up the assembly editing panel as above.
- You can select the degrees of freedom to be restrained (locked) during positioning for each assembly. Restraint acts in the coordinate system of the assembly (if defined), otherwise in the global system.

In this example the lower torso, thorax and head are restrained in Ty, and against all rotations. Contact [C] is switched on between Lower Torso and structure.

As before you click on an assembly to drag it, but now the dummy is treated as a mechanism, and it will follow the mouse movement in a natural way, subject to any restraints placed upon it, and also the properties of the joints between assemblies.

Joint rotation axes and stop angles are honoured as in **Rotate Angles** mode above, but otherwise the dummy is treated as a pin-jointed set of rigid assemblies, and will respond to dragging using rigid body mechanics.

Mouse motion following picking on an assembly works as follows:

Mouse button	Resulting action
Left	All limbs attached to this limb, in both "parent" and "child" directions, that are not fully restrained become draggable, and will follow to where the motion of this limb drags them.
Middle	Only this assembly and its children will move.
Right	Only this assembly, its immediate parent and its children will move.

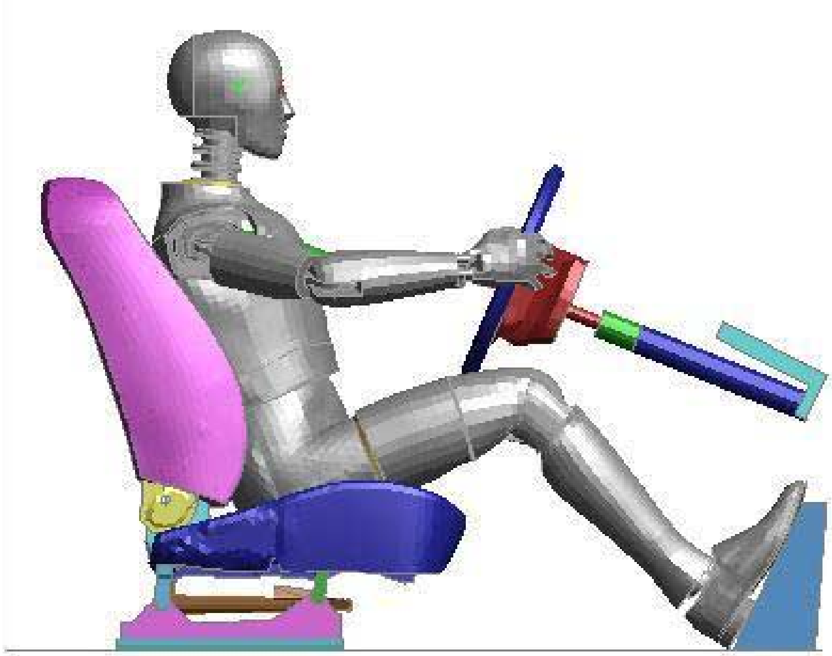
An example of Drag Assembly free dragging.

The following sequence of images shows how this might be used in practice. In this example the dummy has been positioned in the seat, with hands attached to the steering wheel and feet to the pedals. Both hands and feet are fully restrained in all degrees of freedom, the torso, thorax and head are restrained against all rotations and also Y (out of plane) translation.

The user has clicked on the lower torso with the left mouse button, so the whole dummy is selected for movement, and drags it progressively further forwards. This sequence would be carried out in a single operation, and for this dummy the drag occurs in near real-time on a modern desktop computer.

Initial condition.

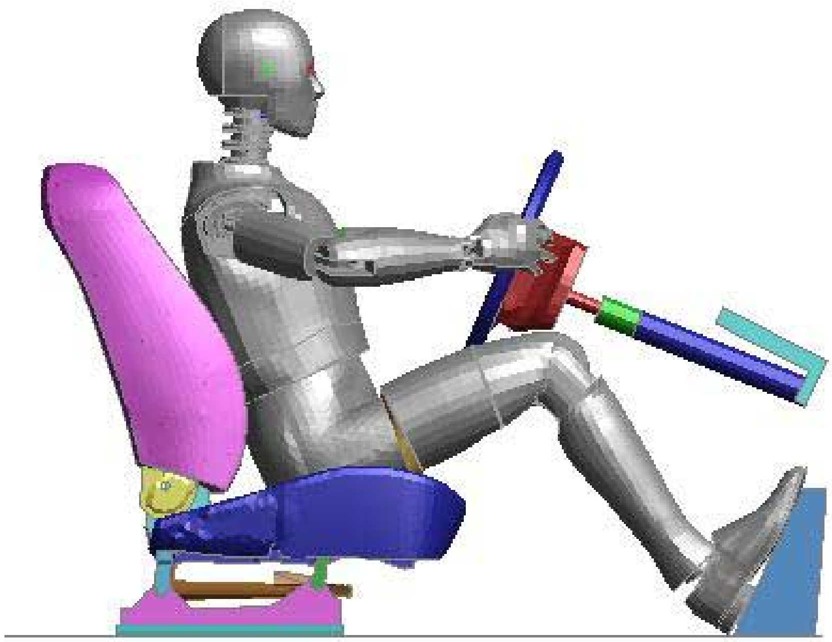
The user has clicked on the lower torso, which selects the whole dummy, and is about to drag from left to right



After about 100mm movement to the right.

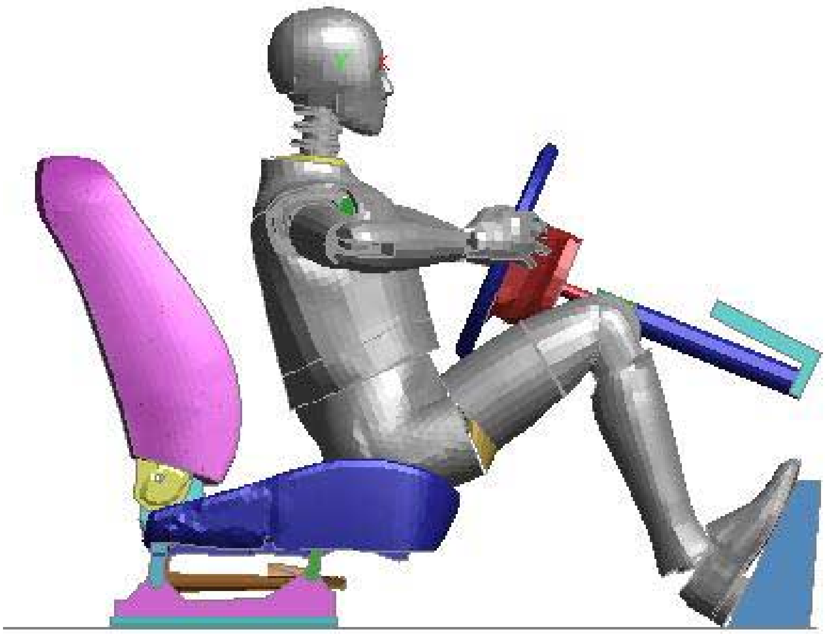
Notice that the hands and feet have remained fixed, the knees have moved up and the elbows have moved out.

Because of their rotational restraints the head, torso and pelvis regions have remained upright.



Final position.

The elbows have moved up and outwards, and the knees have moved up.



Here is the final position in an isometric view.

Arm and leg movement is very obvious!

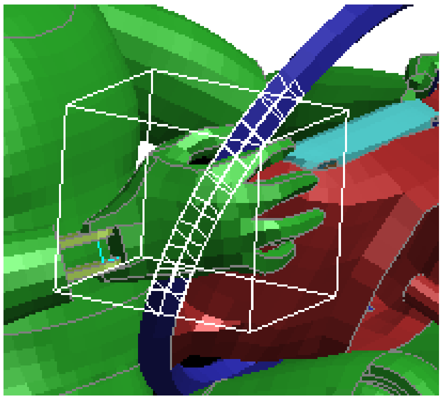
The characteristics of **Drag Assembly** mode:

- **It is approximate**: movement is calculated using rigid body mechanics in an iterative scheme, and some small errors are inevitably generated. Using the default **Options** errors will be of the order of 1 part in 10,000, or around 0.2mm for a typical dummy model which, in engineering terms, is not significant.

  However ls-dyna requires node pairs at joints to be coincident to a very tight tolerance, and the actions taken to achieve this when you **Accept** the positioned dummy are described below.
- **It considers** restraints on assemblies, or restrained points within them. You can switch restraints on/off at will during the positioning process, and indeed the "move to position, then clamp in place" process is the obvious way to work.
- **It is intuitive**: movement is a reasonably natural mixture of translation and rotation, more or less what one would get in real life from grabbing a limb and pulling it.

Using assembly to structure contact

In the example above the hands are fixed rigidly to the steering wheel, which prevents them from rotating and therefore forces the elbows out at an unrealistic angle.



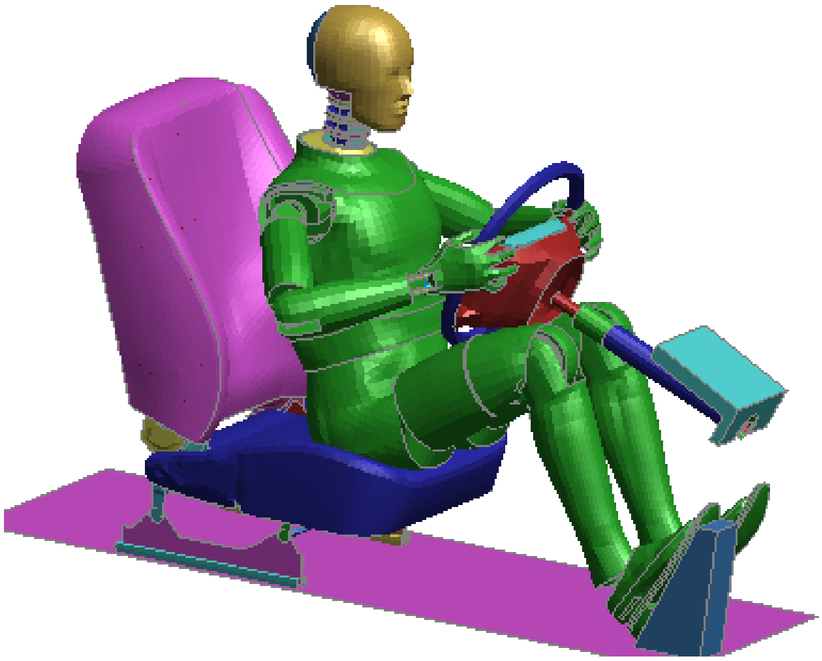
An alternative way of modelling the connection of the hands to the wheel is to define a contact between them and to turn off the fixity. This allows the hands to rotate on the wheel in a more realistic fashion and gives an altogether better final shape.

Contact for dummy positioning is not a "true" contact using the *CONTACT card, but rather a simplified version defined on the assembly editing panel as a "list of part sets contacted during positioning". Here the part set includes the steering wheel, and a box has been used to limit contact to just the section of the wheel near the right hand. A similar contact has also been set up for the right hand.

List of part sets contacted during positioning

Del	Part set	Box	Tk Factor	Active	Explain
X	5020023	2	1.0	1	Sketch
X					Sketch
X					Sketch

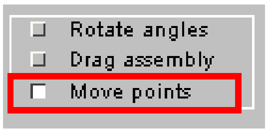
Compare the result with the final image from the example above. The positions of the arms and hands are more natural as they have been able to rotate on the wheel.



The disadvantage is that movement is much slower because of the need to compute contact, making it much harder to drag the dummy interactively when contact is used since response is so slow. For this reason contacts can be turned on/off via their [C] buttons in the "cont" column of the positioning panel.

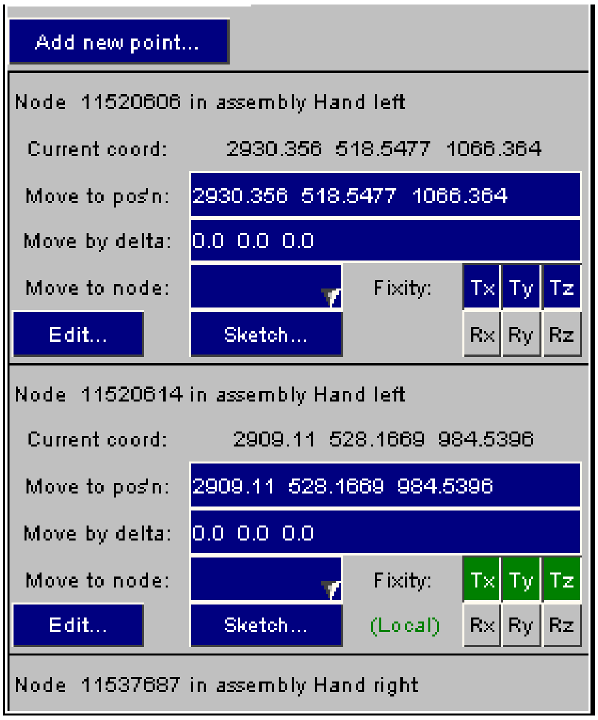
However when positioning a dummy by specifying displacement at a point motion is driven by PRIMER itself and the result is acceptable.

Move Points: "free" movement driven by updated point positions.



An alternative way of using the "free" dragging mode is to set new target positions for points. As described above any number of points can be defined in an assembly, and used both to apply localised restraint and to drive movement.

In this example two points have been created on the left hand, providing an alternative way of fixing it to the steering wheel which permits rotation about the axis through the two points. (Note that the 2nd points has a local coordinate system, so the fixity buttons are in green and "local" appears as a reminder that they are acting in the local system.)



Move to pos'n	Will move the point to the new coordinate specified
Move by delta	Will move the point by the specified [dx.dy,dz]
Move to node	Will move the point to the position of the chosen node.

In all cases the effect is similar to dragging with a mouse, with the difference that PRIMER will drive the iterative scheme for you to try to achieve the new position.

Iteration will continue either until the target point is reached, or the changes between successive iterations become insignificantly small. The latter is necessary since, obviously, it is possible to set a target position for a point that cannot be achieved because of restraints.

Using wild-card coordinates for positions.

It is sometimes the case that you want to move a point a certain distance along one axis, but not to constrain its movement along other axes. To allow for this PRIMER permits the following "wild-card" as opposed to"explicit" coordinate entry syntax.

Values entered explicitly as zero mean exactly that. Therefore

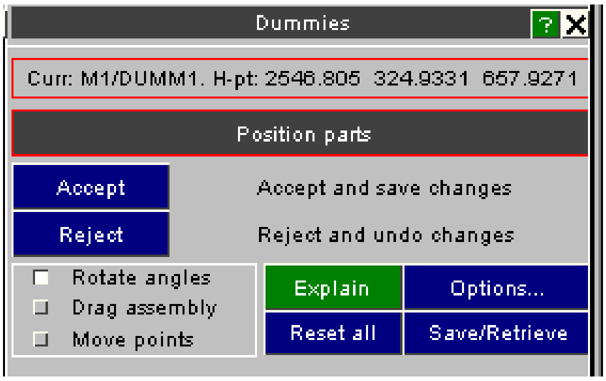
[*Move by delta*] 100.0 0.0 0.0	Means "move by 100 in X, but try not to have any movement in Y or Z."

Omitted trailing values, or values entered as an asterisk "*" are taken to mean "not constrained". Therefore

[*Move by delta*] 100.0	Means "move by 100 in X, but don't care about movement in Y or Z"
	Means "move by 100 in Z, but don't care about movement in X or Y"

The same syntax may be used for absolute [*Move to pos'n*] coordinate entry.

Further positioning commands



The following commands are common to all three positioning methods described above.

Accept	Accept position and save changes.
Reject	Abandon positioning, and restore initial position
Reset all	Restores the dummy to its initial position
Options...	Further positioning options
Save/Retrieve	Save and retrieve positions

Accept: accepts the current position and saves the changes you have made.

Once you are happy with the current position use **Accept** to save it and finish positioning.

Before it saves the position PRIMER examines all the nodal pairs at joints in the dummy to check that positioning has not pulled them apart. It applies a twin tolerance:

- An absolute value of 1.0e-3. This is the value hard-wired into the LS-DYNA keyword reader.
- A distance of 1.0e-6 times the model longest diagonal

Any joints at which separation of nodal pairs exceeds this figure will be listed and you will be given the option of **Autofixing** them.

This is performed by moving each pair of nodes to their average position and, so long as the errors are small, this is an acceptable distortion of the model. This is an iterative process since if a node is on more than one joint then correcting for joint A may move it out of position for joint B. If there are still errors after 5 passes the operation is abandoned and it is left to the user to sort out.

WARNING: You should avoid repeated [Position, Accept, Autofix] cycles on a model.

This is because each **Autofix** operation changes the geometry of your dummy slightly, and while a single such change may be insignificant repeated use of this feature will build up cumulative errors.

It is better to achieve a position in a single operation from an unmodified dummy model. If you are planning to generate a series of positions in succession you should use **Model, Copy** to create a new model from an original one each time, and create the position in the copy.

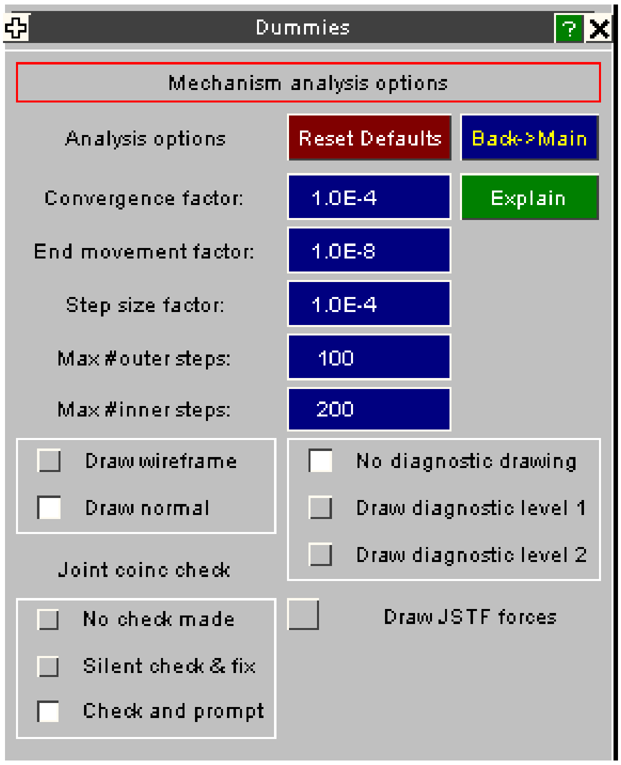
Reject: rejects the current position, restores the initial one and exits the positioner.

Use **Reject** if you want to abandon positioning and restore the initial position. All changes made during positioning will be lost, and you will return to the main **Position** menu with the model unchanged.

Reset all: restores the intial position.

Sometimes positioning goes horribly wrong and the best thing is to start again. **Reset all** restores the initial position that was saved when you entered the positioner, cancelling all changes made since then. You can use this at any time.

Options... Setting positioning options.

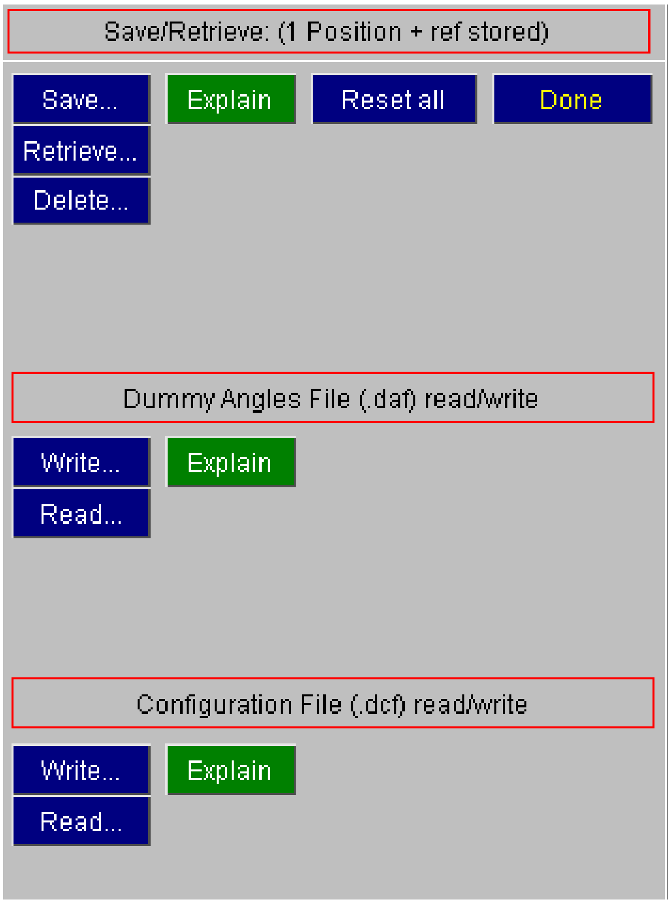


The following options affect the positioner:

Convergence factor End movement factor Step size factor Max #outer steps Max #inner steps	These options all affect the free dragging positioner only. They should not normally need to be changed, but if the dummy moves very slowly, or "gets stuck", then increasing the Convergence and Step size factors may help. Avoid much larger values as the solution will become inaccurate.
Draw wireframe Draw normal	Sets the graphics mode to be used when dragging assemblies. "Draw normal" shows the assembly in grey using normal graphics, but this demands quite a lot of cpu time and only slower computers "Draw wireframe" may be necessary to get acceptable dragging speed.
Joint coinc check	Is the post Accept joint node coincidence check. By default it will check and report any results, asking you what action to take. You can it work silently, which will fix any errors without further input, or turn it off altogether.

Diagnostic drawing and JSTF force display are for programmer debugging purposes and - hopefully - can be ignored!

Save/Retrieve: Saving and restoring dummy positions.



You can store any number of dummy positions, and retrieve them at any time into the positioner.

A stored position contains a [centre of gravity] plus [3x3 direction cosines] for each assembly in the dummy, making it possible to return to a given configuration without any further calculation. Position data is stored in the dummy tree file and is saved when the keyword file is written out.

Save...	Saves the current position. You only need to give a unique name.
Retrieve..	Restores a saved position. This becomes the current position and the dummy geometry is updated immediately causing it to "jump" to the new position.
Delete...	Deletes the selected positions. Deletion is permanent!

A more detailed explanation of saved positions, including card formats, is given in Appendix IIc.

Dummy angle files: saving dummy positions in a model independent file

While saved positions are a powerful feature they have the disadvantage that they are specific to the current model, and also (because they contain explicit assembly centres of gravity) to the current dummy's geometry.

To make it possible to transfer positioning information between similar, but not identical dummies - for example a new version of an existing dummy - PRIMER also supports a model-independent "dummy angles file" (.daf extension).

This file contains:

- The dummy H-Point position.
- The rotation angles of the dummy.
- The rotation angles of each assembly.

On reading this file back in:

- Each assembly angle is reset to the specified angles, applied in the order [Rx, Ry, Rz].
- If the user chooses to move the H-Point then:
    - The dummy as a whole is translated to the specified H-Point position
    - If "whole dummy" rotation angles are present these are applied as an absolute orientation

This file is likely to work well when transferring positions between successive versions of similar dummies but, obviously, it will not be suitable where the target dummy geometry is significantly different to that of the source. However used intelligently it should be a useful tool.

See The Dummy Angles File in Appendix IId for a full description of the .daf file format, and the section on Euler angles in PRIMER below for a precise description of what "rotation angles" actually mean and how they are applied.

Dummy Configuration file: saving the current positioning settings only

In some situations you may wish to save and retrieve only the restraints (and any local coordinate systems) applied to the dummy during mechanism-style positioning, and a "configuration file" (.dcf) performs this function.

It contains the fully set of cards normally written between **`*DUMMY_START`** and **`*DUMMY_END`** in the keyword file, but when read back in ***only*** the following information is processed and applied to the current dummy:

Assemblies	Any restraints, and any local coordinate systems used for these.	(Assemblies are matched by label)
Points	Any restraints, and any local coordinate systems used for these.	(Points are matched by name)

Note that geometry, coordinates, connectivity and the like are ignored when this file is reread.

6.12.4 Batch (command line) positioning

A subset of the interactive positioning commands described above are also available in command-line line form. While these can be used interactively the main purpose of them is to enable positioning to be performed in batch mode. These commands will provide visual feedback if the graphical user interface is running, but if it is not (PRIMER started with "-d=batch" command line option) they will still function. A full listing of command-line commands is given in Appendix X11.

The positioning commands are invoked by the [Primer >] **`DUMMY`** command, and occupy a hierarchy as follows:

At **DUMMY >** level		
ASSEMBLY	Select an assembly by name or number, then perform one of the following operations upon it:	**FIX** *dof code* — Restrain the assembly in degrees of freedom *dof code* **TRANSLATE** *dx, dy, dz* — Translate assembly ***by*** amount *dx,dy,dz* **RX** or **RY** or **RZ** *theta* — Rotate assembly ***to*** angle *theta* degrees about x/y/z **RESET** — Undo all dummy transformations and return to initial state **DONE** — Finish with assembly and return to **DUMMY >** prompt
POINT	Select a point by name or number, then perform one of the following operations upon it: (Note: moving the point implicitly moves its "owner" assembly.)	**FIX** *dof code* — Restrain the point in degrees of freedom *dof code* **TRANSLATE** *dx, dy, dz* — Translate point assembly ***by*** amount *dx,dy,dz* **POSITION** *x, y, z* — Translate point assembly ***to*** coord *x, y, z* **RESET** — Undo all dummy transformations and return to initial state **DONE** — Finish with point and return to **DUMMY >** prompt
CONNECTION	Select a connection by name or number	**SLIDE** *distance* — Applies to **LINE** connections only, and will slide the joint by *distance* down its AB axis. **ANGLE** *theta* — Applies to **LINE** and **HINGE** connections only, and rotations the assemblies to achieve angle *theta* (in degrees) about the AB axis.
POSITION	Specify a position *name*	Retrieves and applies the stored position *name*
H_POINT	Specify coordinate *x, y, z*	Will move the Dummy H-Point ***to*** coord *x, y, z*
READ_CONFIG	Specify a *filename*	Retrieves a free-standing dummy configuration file (the keywords and data between ***DUMMY_START** and ***DUMMY_END**). *Filename* will usually have the extension .dcf
READ_DUMMY_ANGLE	Specify a *filename*	Retrieves and applies the overall orientation, H-Point and joint angles stored in a Dummy Angles File (usually extension .daf).
ACCEPT	Accept the current dummy position, save its updated geometry and return to the main [Primer >] prompt.	
RESET	Undo all transformations and restore the initial geometry of the dummy, remaining at this prompt level.	
QUIT	Undo all transformations and restore the initial geometry of the dummy, then return to the main [Primer >] prompt.	

Meanings of terms in the table above

dof code Is a numeric Degree of Freedom code made up of any permutation of 123456, where

1 = Tx, 2 = Ty, 3 = Tz, 4 = Rx, 5 = Ry, 6 = Rz

For example code **136** means restraint in TX, Tz, Rz

Code **0** may also be used, meaning "free all restraints" *dx, dy, dz* Is a translation vector, ie a relative movement from the current position, made up of three numbers.

For example `10.0 20.0 30.0` means translate 10.0 in X, 20.0 in Y, 30.0 in Z.

"Wildcard" syntax is permitted: any number entered as an asterisk ("*"), and omitted trailing digits, are treated as "free" values. For example:

`10.0`	means translate 10.0 in X, but permit Y and Z to adopt any value.
`* * 20.0`	means translate 20.0 in Z, but permit X and Y to adopt any value

x, y, z Is an absolute coordinate.

For example `10.0 20.0 30.0` means coordinate X=10, Y=20, Z=30.

Wildcards as for translations above are permitted

theta Is an angle in degrees for the given degree of freedom.

In a dummy model angles are absolute values expressed in the coordinate system of the connection between this assembly and its parent. In most cases this will mean the system implied by the local axes of the joint stiffness definition at the joint.

6.12.5 Using dummies as "children" of mechanisms

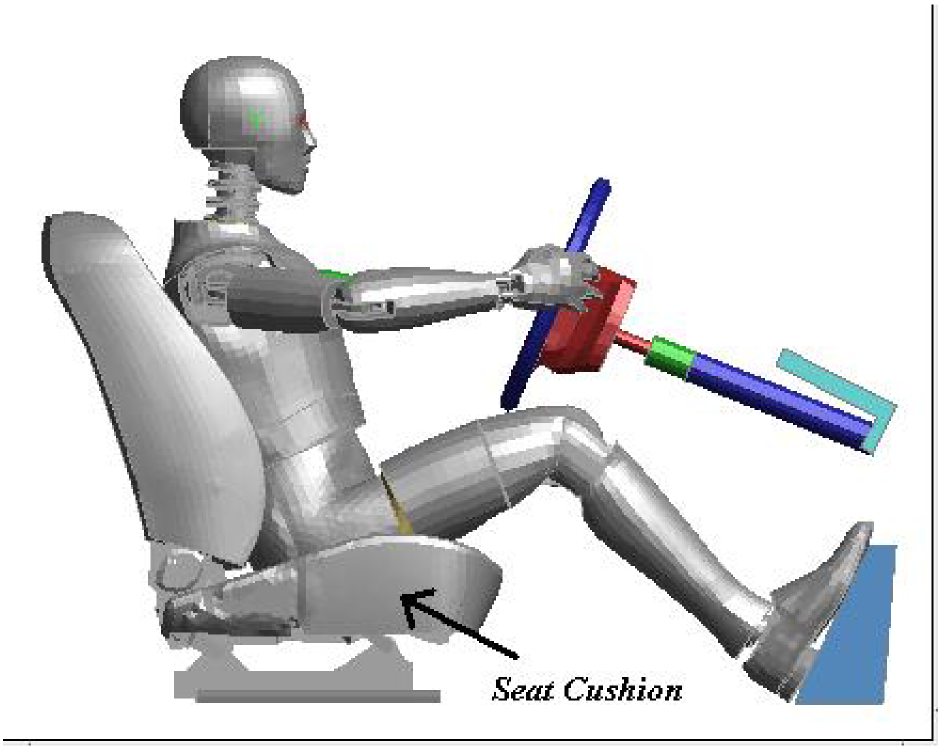
Dummies work as components of larger models, and it is usually the case that they are positioned on a seat with their feet in the floor or pedals, and their hands may be placed on a steering wheel. These "cockpit" components may themselves be capable of articulation, and the Mechanism capabilities of PRIMER may, for example, have been used to set the position of the seat.

Clearly when part of the cockpit moves it is likely that the dummy will need to be repositioned, and to make this easier PRIMER permits a dummy model to be made a "child" of a mechanism, and to move with it. When operating as a child a dummy is moved in the "free dragging" mode described above, with all the positioning capabilities and settings still active. The main difference is that the motion of the dummy is driven by the controlling assembly of the parent mechanism.

In the example below the seat assembly has been defined as a mechanism, and the lower torso of the dummy has been slaved to the motion of the seat cushion in degrees of freedom TX, Ty, Tz.

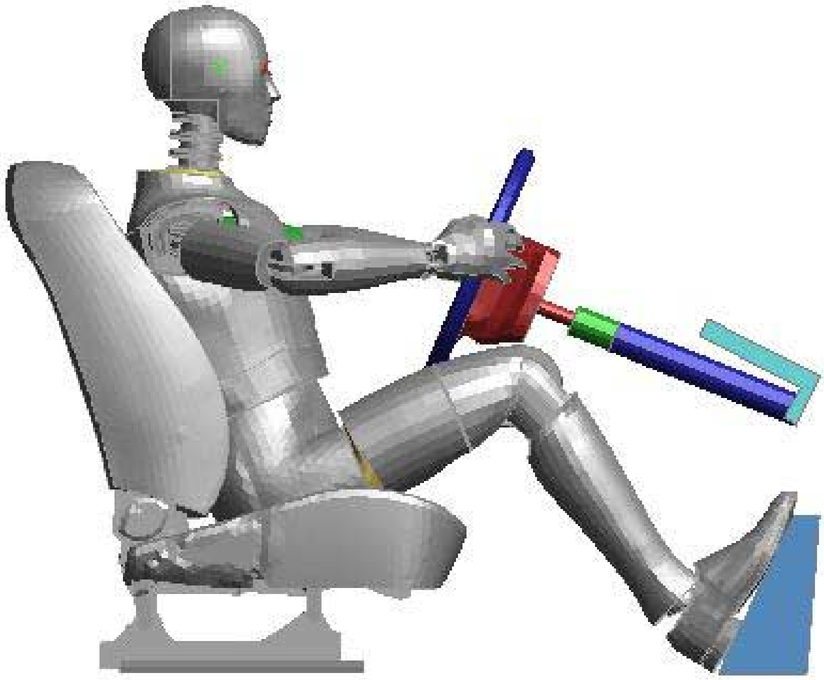
Initial state.

User has clicked on the seat cushion and the whole mechanism (seat) plus slaved dummy turn grey to denote that they are being dragged.



Intermediate state.

The seat has moved forward and risen up on its links, taking the dummy with it.



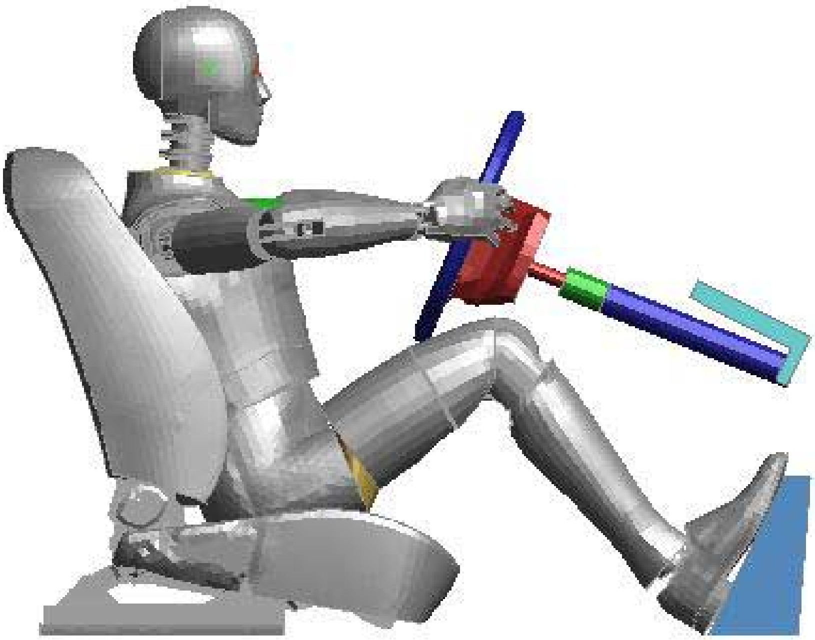
Final (very uncomfortable!) position.

In this example the seat has been moved forward and down to a ridiculous degree, but this demonstrates two things clearly:

(1) The dummy motion has remained linked to that of the seat.

(2) Connection between seat and dummy is in translation (TX, Ty, Tz) only with no rotational connection.

This is made clear by the way that the seat cushion has tilted down but the pelvis, torso and head of the dummy have not rotated.



The use of a dummy as a child of a mechanism is controlled entirely on the Mechanism panel, described in section 6.21.1.

6.12.6 Notes on using dummy angles.

There are some problems with the way GENERALIZED_STIFFNESSES have been programmed into LS-DYNA. These all arise from the method of defining joint rotations as angles about the three Cartesian axes on the "parent" side of the joint, often referred to in literature as "Euler angles".

- Where rotations take place only about one axis then there is no problem: the current angle as reported by PRIMER will be the simple cumulative sum of all rotations to date about that axis.
- However where rotations are permitted about more than one axis then life becomes more difficult since the order in which Euler angles are applied matters in two related ways:

1. A rotation about Phi, followed by one about Theta and then one about Psi will not give the same result as applying the same rotations in a different order

2. Once the current rotation about any one angle is non-zero then rotations about any other axis will result in some compound set of rotations that may not be the expected numerical sum of rotations about the individual axes.

   (To demonstrate this try the following: set the initial view in PRIMER to a plan on XY [**SXY**], then compare rotations of 90 degrees about screen X, then Y, then Z [RS 90 0 0], [RS 0 90 0], [RS 0 0 90] against the same rotations in the order Z,Y,X.)

- Therefore the reported angles for assemblies free to rotate about all "parent" axes may not be the simple cumulative sum of the incremental rotations about each axis.

To understand this requires some explanation of how "Euler angles" are used inside PRIMER, and also of how it computes and maintains the current orientation of dummy assemblies.

Euler angles in PRIMER

In the following discussion the [Phi, Theta, Psi] angle notation used by the *CONSTRAINED_JOINT_STIFFNESS card in LS-DYNA will be referred to as [X,Y,Z]. This easier to write and also to understand!

The order in which PRIMER applies Euler angles.

PRIMER treats a [X,Y,Z] compound angle definition as a set of rotations that it applies in the order X, Y, Z. To be more specific:

If the rotation matrix about X is written [Rx], and those about Y and Z as [Ry] and [Rz]

then a single compound matrix [Rc] is assembled from [Rz] . [Ry] . [Rx]

This looks counter-intuitive, but in fact when concatenating rotation matrices the effective order of rotations is right to left, ie the most recently applied rotation matrix (here [Rx]) is effectively the first rotation; thus the matrix above does indeed give rotations in the order X, Y, Z.

If you read up about Euler angles and robotics you will find that there are other possible application orders, but this is in many ways the simplest and most intuitive, so it is what PRIMER uses!

The current orientation, and computing updated Euler angles from this

Internally PRIMER keeps track of each assembly’s orientation using "direction cosines", which are effectively a local coordinate systems expressed by three vectors at right angles. When a Dummy is first read into Primer these direction cosines are initialised for each joint, taking into account any initial angular differences implicit in the *CONSTRAINED_JOINT_STIFFNESS definition.

When an assembly is rotated the compound rotation matrix [Rc] described above is applied about the "parent" node, resulting in some new orientation, and these direction cosines are updated accordingly, so that they always maintain an accurate description of the assembly’s orientation with respect to its parent.

The angles reported in the Dummy assembly rotations panel, and those used in the Dummy Angles File, are calculated from these direction cosines and ***not*** from some cumulative sum of applied angular rotations. There are two reasons for this:

1. Using a "cumulative sum" only works for rotations about a single axis; once rotations about all three axes are permitted then rotations quickly get jumbled up together. Therefore such an approach would not work for those assemblies (typically head and leg components).

2. Dummy assemblies may be moved arbitrarily during the "drag" mode positioning process, resulting in a large number of small incremental displacements and rotations. Not only would it be expensive to keep track of these, but it would also lead to a considerable cumulative error due to adding small increments to a (relatively) large running total.

In cases where rotation is about only one axis then the angles derived from the direction cosines will match those that would be computed from a "cumulative sum", but once rotation becomes significant about 2 or more axes then the values will differ. To see why this should be consider the rotation matrices required to build up the full matrix of direction cosines.

[Rx]: Rotation about the X axis: [Ry]: Rotation about the Y axis:[Rz]: Rotation about the Z axis:

[Rx]	[Ry]	[Rz]
Sx = Sin(theta X)	**Sy** = Sin(theta Y)	**Sz** = Sin(theta Z)
Cx = Cos(theta X)	**Cy** = COs(theta Y)	**Cz** = COs(theta Z)

```
[ 1  0   0   ]      [ Cy   0  Sy ]      [ CZ  -Sz  0 ]
[ 0  Cx  -Sx ]      [ 0    1  0  ]      [ Sz  CZ   0 ]
[ 0  Sx  Cx  ]      [ -Sy  0  Cy ]      [ 0   0    1 ]
```

Concatenating these together in the order [X, Y, Z], ie [Rz] . [Ry] . [Rx] gives the compound matrix [Rc]:

```
[   Cy.Cz    Sx.Sy.Cz - Cx.Sz    Cx.Sy.Cz + Sz.Sz   ]

[   Cy.Sz    Sx.Sy.Sz + Cx.Cz    Cx.Sy.Sz - Sx.Cz   ]

[   -Sy      Sx.Cy               Cx.Cy              ]
```

From which it can be seen that a set of Euler angles can be extracted as follows (using the notation <**ij**> is row <**i**>, column <**j**>)

Theta X = arctan(32/33) Since (Sx.Cy / Cx.Cy) = (Sx / Cx)
Theta Y = arcsin(-31)
Theta Z = arctan(21/11) Since (Cy.Sz / Cy.Cz) = (Sz / CZ)

However there are four well known problems with this calculation method:

1. The rotation about the Y axis, theta Y, can only be obtained in the range +/-90 degrees from the arcsin() operation.

2. At the special case of Theta Y very close to +/-90 degrees, ie Cy = 0, the calculation of the rotations about the other two axes is ill-conditioned. To see why, here is the [Rc] matrix above with Cy = 0:

```
[   0      Sx.Sy.Cz - Cx.Sz     Cx.Sy.Cz + Sz.Sz   ]

[   0      Sx.Sy.Sz + Cx.Cz     Cx.Sy.Sz - Sx.Cz   ]

[   -Sy    0                    0                  ]
```

   Clearly the arctan() operations will be upon (0/0) for both theta X and theta Z, ie undefined.

3. If rotation has taken place about more than one axis then the angles returned from this calculation will not necessarily be the same as those input, although the result of multiplying through by them to achieve a new orientation will be correct.

4. As mentioned above the rotation order [X, Y, Z] is implicit in this calculation, and combined rotations about 2 or more axes in a different order will give a different result. (Although, again, it will give a consistent result when used to orient a dummy or limb.)

PRIMER deals with these problems as follows:

1. If rotation takes place about the Y axis only, or nearly so, (ie theta X and theta Z both less than +/- 10 degrees) then special exception logic is used to calculate theta Y in the full range of +/-180 degrees. Therefore dummy limbs which are locked in X and Z rotation may be rotated safely in the full Y range.

2. The ill-conditioning problem is treated by using double-precision arithmetic, and relying on the (sensible) behaviour of the standard **atan2()** function near these singularities. In practice this solves the problem in virtually all cases, although rotations of exactly +/-90 degrees about the Y axis should still be avoided if at all possible if stop angles are to be used.

3. Compound rotations about multiple angles are calculated as described above, and a rational set of angles that matches these cosines is returned, even if it is not what you input in order to create them. This does not normally matter unless "stop angles" have been defined, in which case these will become increasingly unreliable as the angular movement of the assembly departs from its initial orientation.

4. The "order of combined rotation" problem is really the same as (3) above, and is treated in the same way.

In all cases the angles reported are "correct", in that if they were applied to the initial reference position of the assembly they would give the current orientation, however they may not be what you expect. Perhaps a good way of thinking about this is to consider a journey across the earth's surface defined by increments of both latitude and longitude: your input would be a series of "rhumb line" increments, whereas your position would in fact be reported as the "great circle" angles required to get there. In addition if you travelled around the world, and approached your start position again, the angles returned would be those of the "shorter" distance, possibly negative, rather than the "longer" ones travelling around the globe.

Realistically "stop angles" will only work properly for joints that are only permitted to rotate about a single axis, in which case they can be computed unambiguously in the full +/-180 degree range.

Achieving explicit Euler angles in the positioning panel.

Once again, if rotation is only to be about a single axis then what you type in will be what you get reported back. However if you want to type in an explicit set of angles about multiple axes you may find that you don’t get what you expect. The best solution to this "composite angle" case is to proceed as follows:

1. "Undo" any current angles in the order Z, Y, X. In other words set Theta Z to zero, then Theta Y, then ThetaX; in this way you will get back to angles (0.0, 0.0, 0.0).

2. Set the new angles in the order X, Y, Z. In this way they will not "interfere" with each other.

It follows from this that if you want to modify just one angle of an existing set you will be able to do so directly if:

- It is theta Z (ie the last cumulative one to be applied).

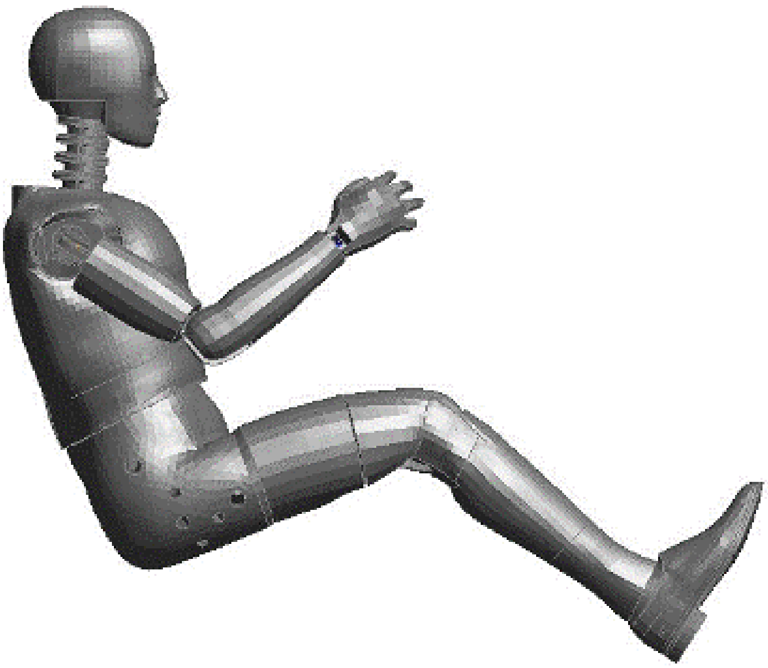
- Or the rotations applied "after" it are zero.

  So if you want to rotate about Y you will be able to get away with simple typing if theta Z is zero. If it isn’t then it will be necessary to reset theta Z to zero, apply the new Y angle, then restore the original theta Z value.

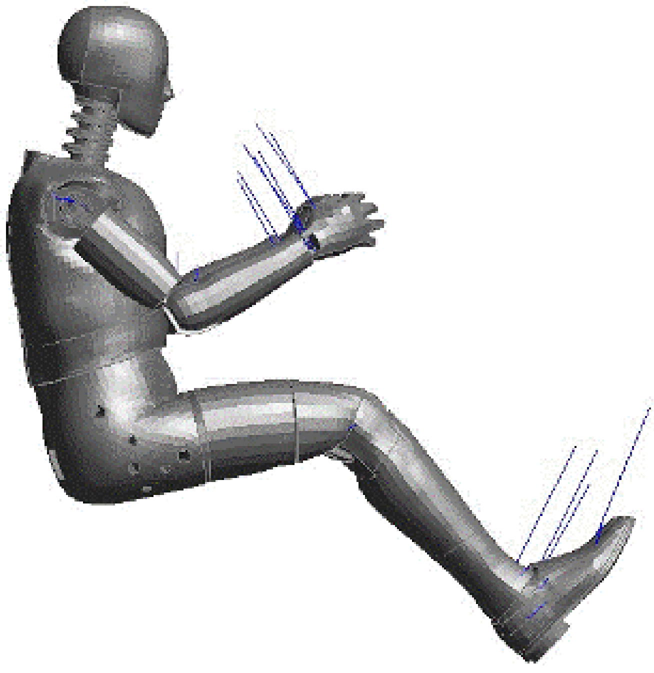
6.12.7 Dummy positioning using LS-DYNA

The LS-DYNA dummy positioning tool in PRIMER allows the user to create an LS-DYNA analysis automatically to position a dummy. This works in a similar way to the PRIMER seatsquash LS-DYNA method. This method of positioning is appropriate for those who wish to capture deformations in the foam/rubber parts of the dummy that occur due to the positioning, that are not captured by PRIMER’s traditional dummy positioning methods. The LS-DYNA positioning method is illustrated below:

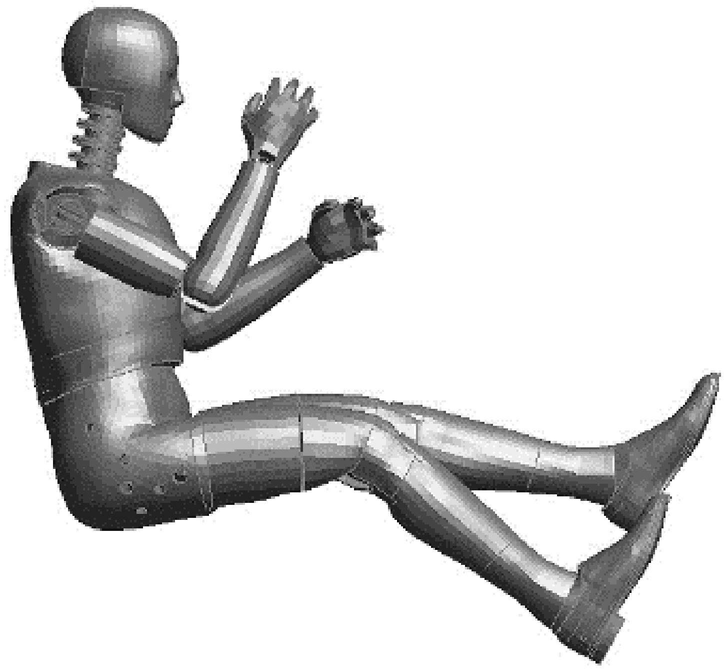
Starting point - dummy in original position.



Through using the PRIMER LS-DYNA positioning tool, an LS-DYNA analysis is created that will "pull" the dummy into the desired position.



A DYNAIN file is produced by the LS-DYNA analysis. PRIMER can now import the DYNAIN data back into the original model. This will mean updated coordinates capturing the deformation of the dummy foam/rubber parts will be updated in the original model.

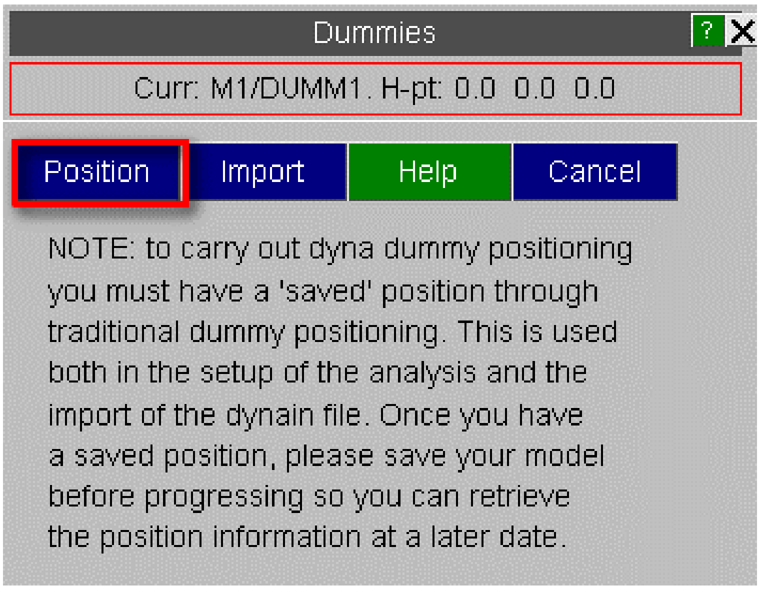


The tool works through a series of steps to create the LS-DYNA analysis. Before using this tool, the user must save the target position using Primer's traditional dummy positioning methods, as the saved position is used to create the entities required to "pull" the dummy into position. The initial model should contain just the dummy.

Step 1

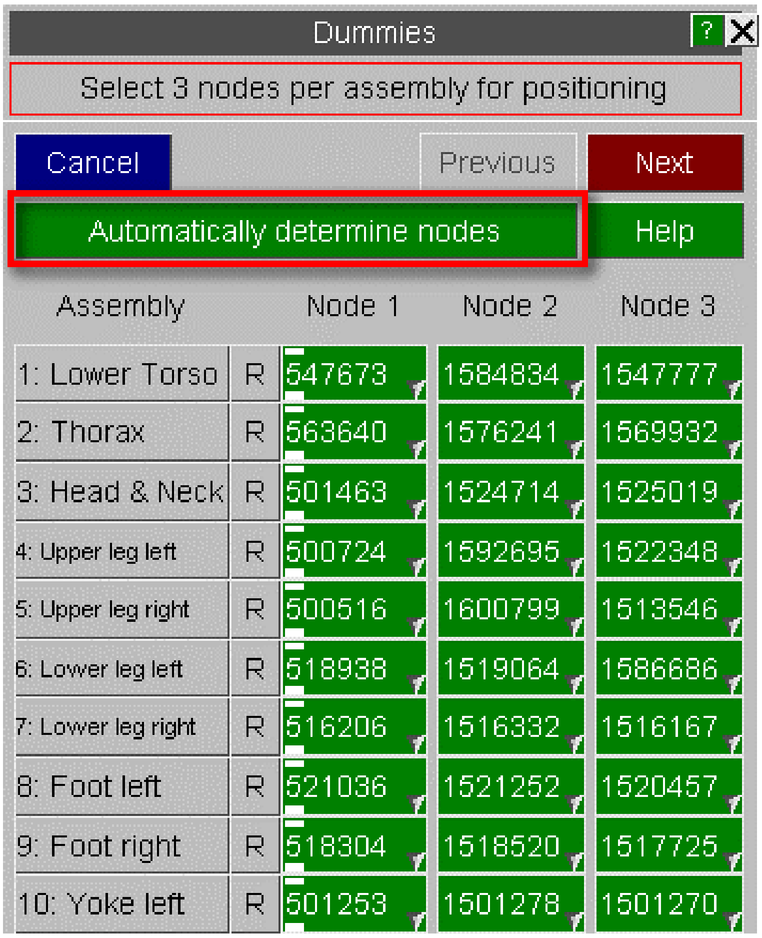
Step 1 is a reminder to ensure you have a saved position using the traditional PRIMER positioning methods. See section 6.12.2 for information on saving positions.

Once you have a saved position, press **Position**.



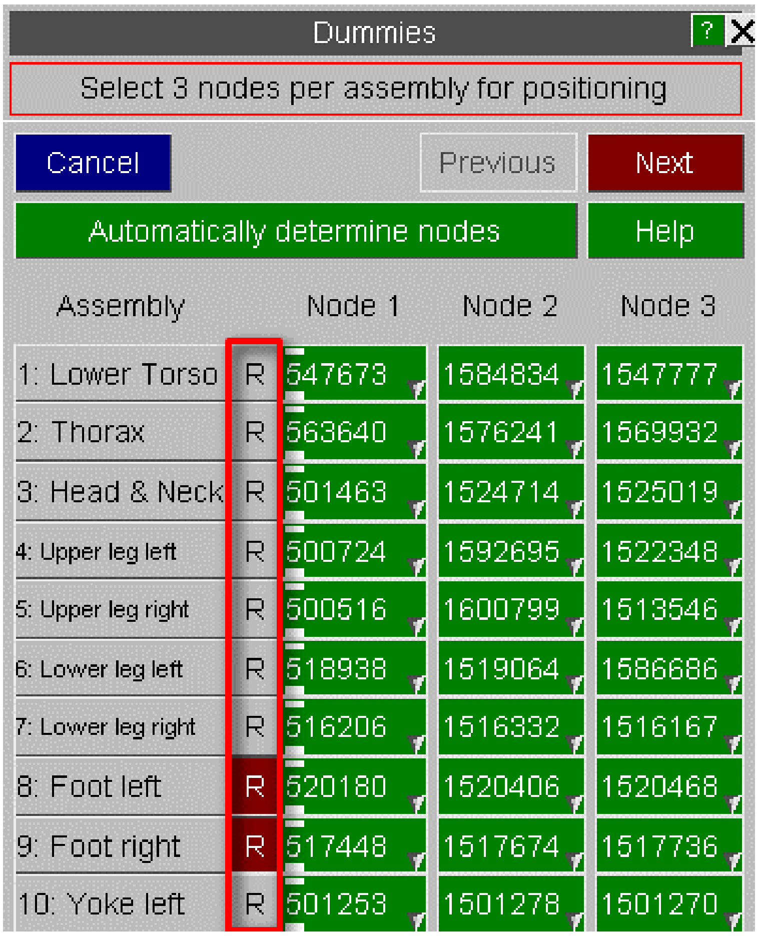
Step 2

To position the dummy, PRIMER requires three nodes per dummy assembly. These nodes are used to "pull" each assembly into their final position in a LS-DYNA analysis. The nodes selected should be on a rigid or stiff part of the assembly. The nodes can be set manually, however it is recommended to use **Automatically determine nodes**. This will find rigid nodes in each assembly to use during positioning.



You also have the option to rigidify any assembly for the LS-DYNA analysis. This may be beneficial for soft extremities, such as hands and feet, which may become excessively deformed during the LS-DYNA positioning. If the **R** switch next to the assembly is toggled on, PRIMER will rigidify the assembly when setting up the analysis in the final step. Note that when toggling the **R** switch, Primer will automatically recalculate the 3 nodes for that assembly, as there will now be more/less "rigid" nodes in the assembly.

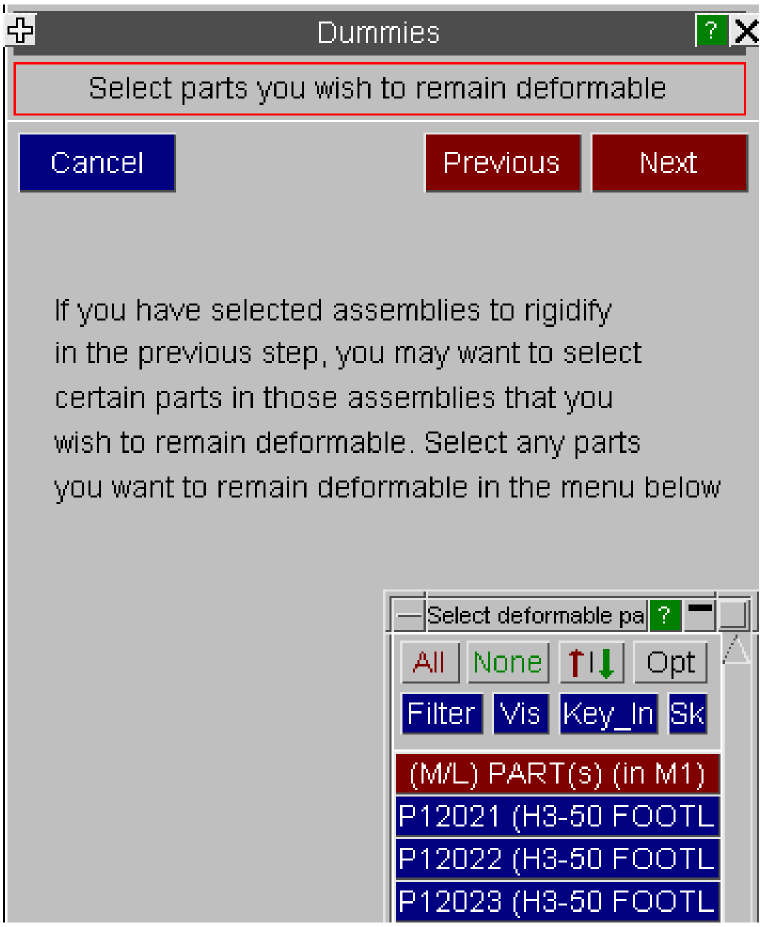
Once you have selected the nodes for each assembly, and chosen which assemblies you wish to rigidify, click **Next**.



Step 3

The next step is optional and allows you to select any part from the assemblies you have chosen to rigidify that you may wish to remain deformable.

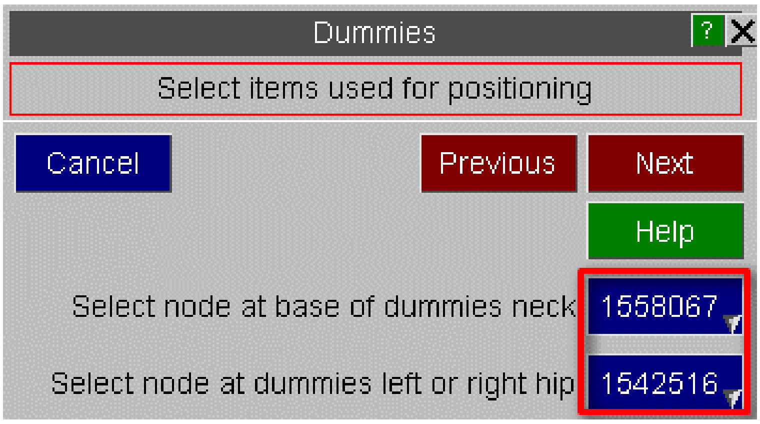
Once you have made any selection, or to skip this step, click **Next**.

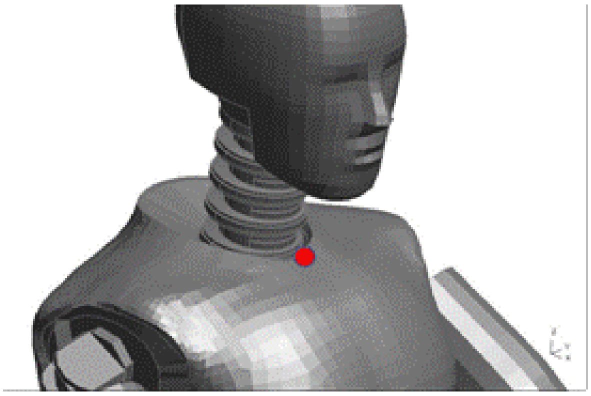


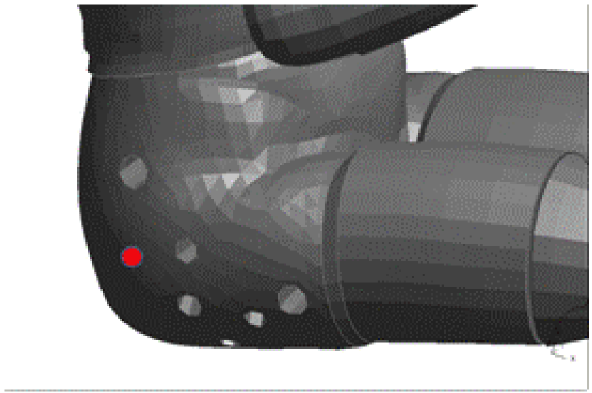
Step 4

When the analysis model is created, PRIMER will move the dummy to the desired H-Point. PRIMER will then use the neck base node to rotate the dummy around it's local Y axis to the approximate final position. This helps to reduce the overall movement of the assemblies during the analysis and hence can reduce run times. It is recommended that the neck node be set to a node at the top of the torso/base of the neck. Primer will also use the hip node to rotate the dummy around it's local Z axis to the approximate final position. It is recommended that this be set to a node on the right or left hip (outer surface). See images below for examples of this.

Once you have selected the two nodes, click **Next**.



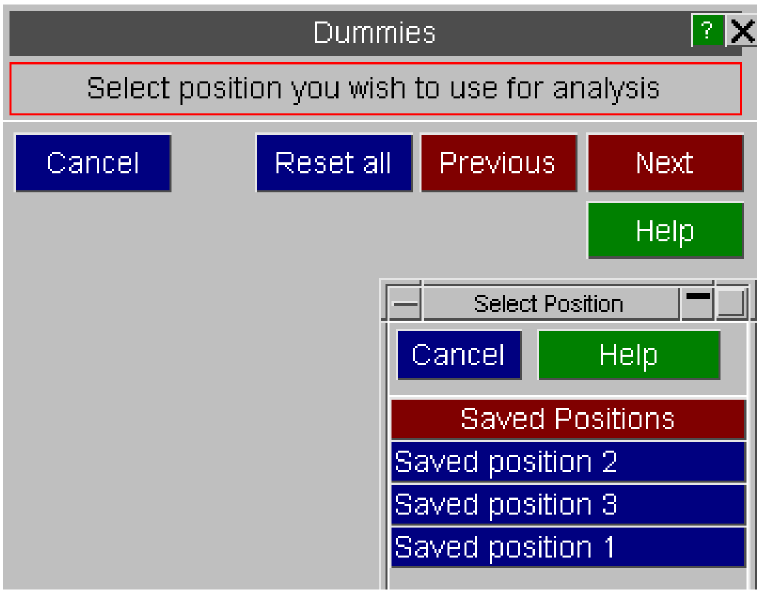




Step 5

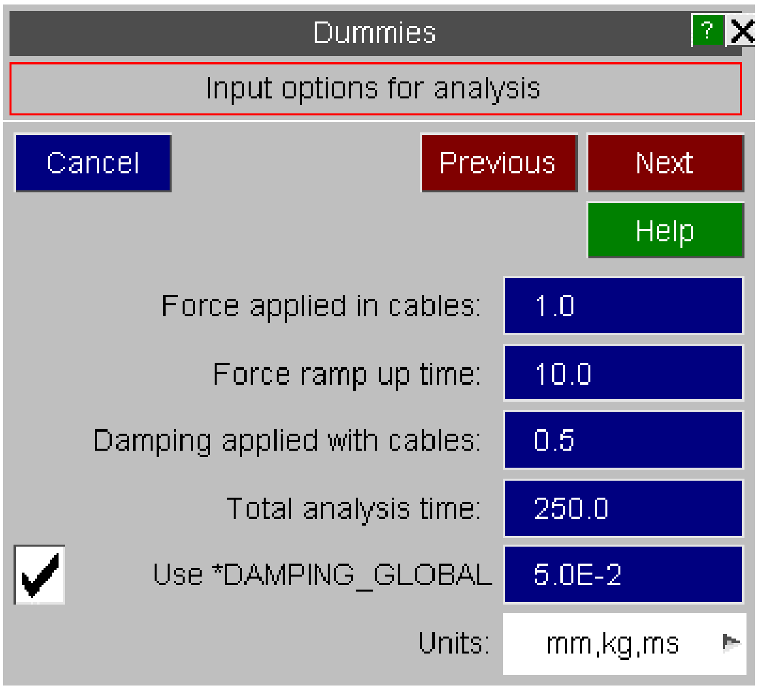
The next step is to choose the position saved earlier. This is the position you want to achieve through the LS-DYNA analysis.

Once you have chosen the position, click **Next**.



Step 6

At this point you may wish to save the model. The nodes chosen in the previous steps can be saved to the dummy tree in the keyword file. Leave the **Dummies** tab open, and click on **Model->Write** to save the model.



The LS-DYNA method works by creating cables and dampers that pull the assemblies into the final position. A number of inputs relating to the analysis can be set on the final panel:

Force applied in cables - The force that is applied to the cables to pull the assemblies into position.
Force ramp up time - The time taken to ramp the force up from zero.
Damping applied with cables - This is the damping applied in-line with the cables used to pull the assemblies into position.
Total analysis time - Termination time for the analysis.
Global damping - Global damping applied in the analysis (can also be turned off).
Units - Specify the units system you are working in to update the above values accordingly.

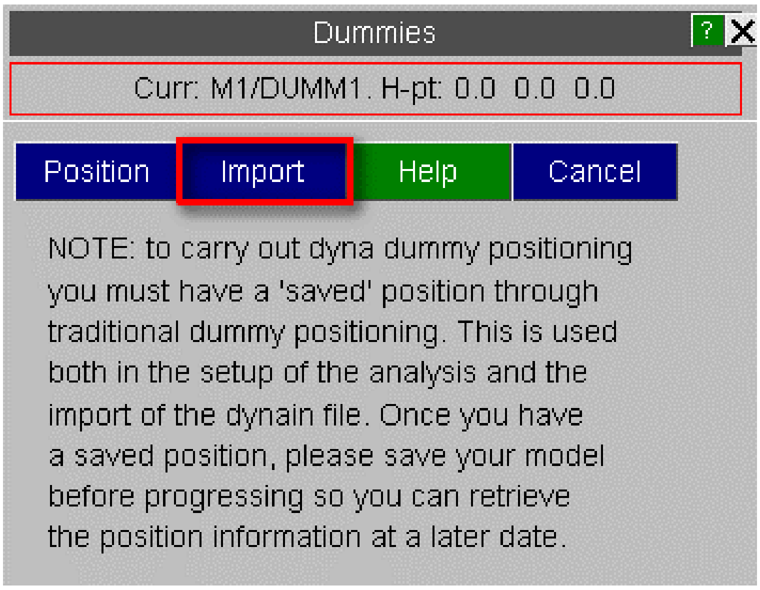
It is recommended to use the default settings to start with. These can then be modified with subsequent analysis is required.

Once you have set the values you want, click **Next**, the **Apply**.

The analysis is now ready to run. Once complete, the analysis will produce a DYNAIN file, which will have a name similar to "end_stage001_dynain". This contains coordinate and initial stress information from the analysis which can be imported back into the original model using PRIMER. To import the DYNAIN file back into the original model, use the following steps:

Step 1

Step 1 is a reminder to ensure you have a saved position using the traditional PRIMER positioning methods. You need the original saved position for the import tool should you wish to further modify the position of the dummy using Primer’s traditional rigid body dummy positioning methods. See section 6.12.2 for information on saving positions.

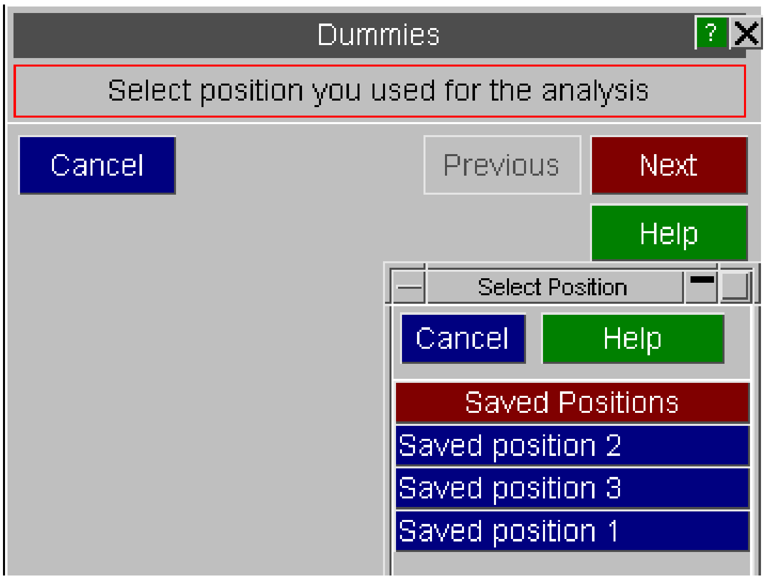


To continue with the import procedure, press **Import**.

Step 2

Step 2 is to select the position you used when creating the LS-DYNA analysis. This is the position saved using Primer's traditional rigid body positioning methods. The position needs to be chosen here to ensure any further positioning is possible.

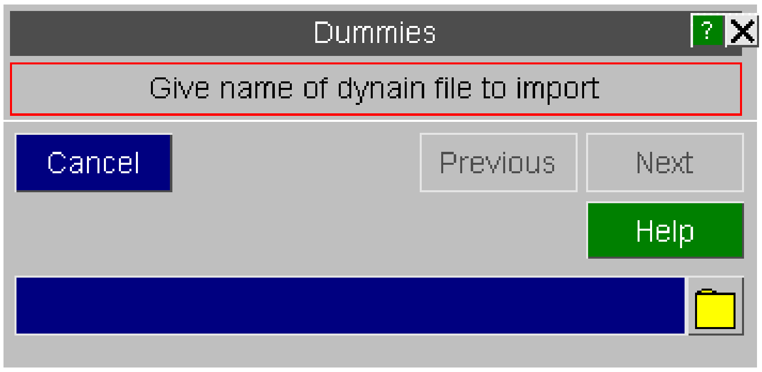
Once you have chosen the position, click **Next**.



Step 3

Step 3 is where you select the DYNAIN file produced by the LS-DYNA analysis.

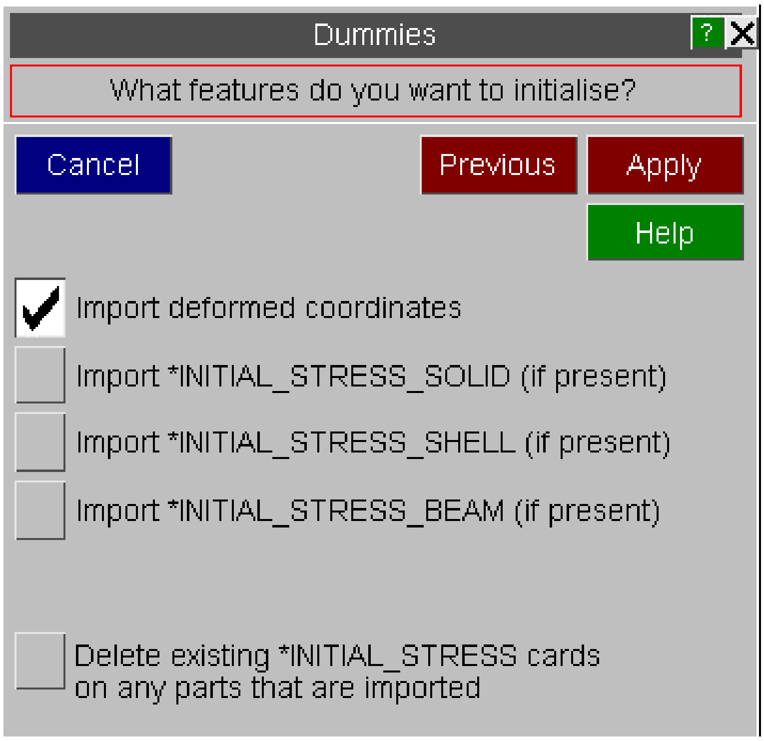
Once you have chosen the DYNAIN file, click **Next**.



Step 4

Step 4 is where you select what data you wish to import from the DYNAIN file.

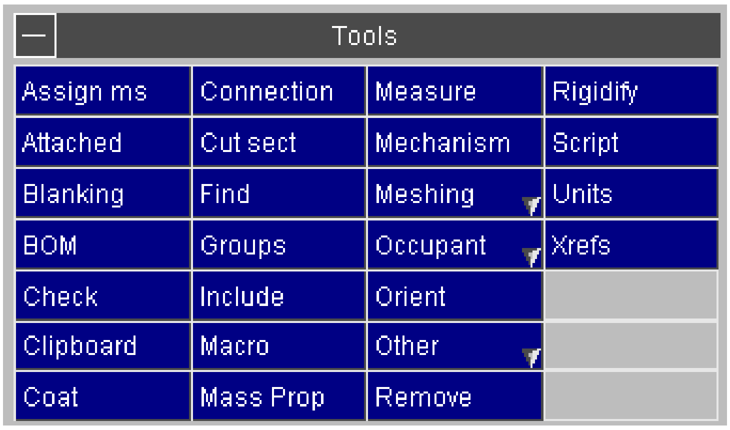
Once you have made your selection of what you want to import, click **Apply** to import the data into your model.



6.13 FIND AND SKETCH

The **Find** tool allows you to locate an item in your model.

The features developed for this function have been generically incorporated into Primer's sketch function when it is applied to a single item

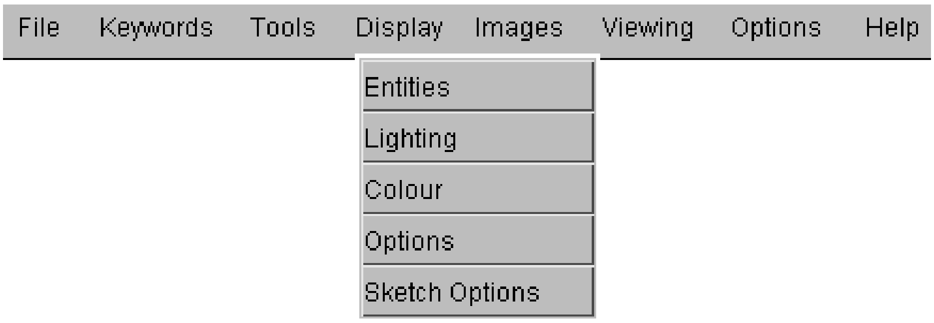


6.13.1 Sketching a single item

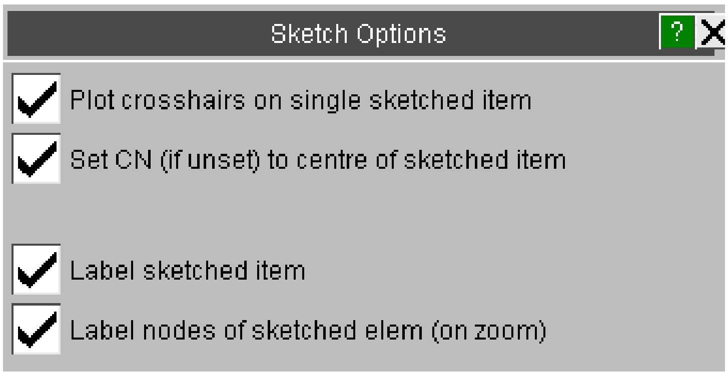
There are many ways in Primer to sketch items. For example

- sketch function available to each keyword
- sketch off object menu
- sketch on drop-down available in many contexts
- quick-pick sketch using id key in

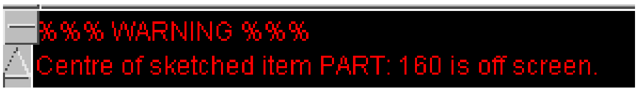
If a **single item** is being sketched this will be done according to the sketch options under Display.



By default the following actions will be taken in addition to the previous sketch function. These options may be switched off.



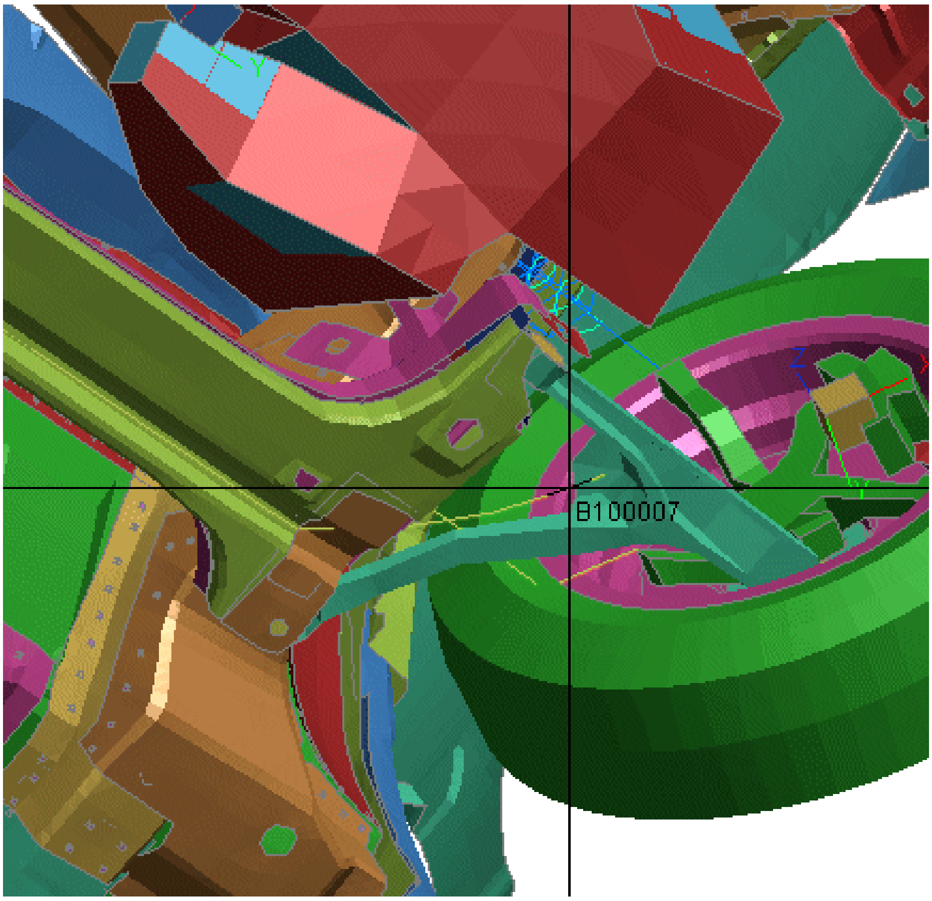
Plot crosshairs Draws crosshairs centred on the sketched item. This is particularly useful if the item is very small and the normal sketch cannot easily be seen. If the item is off screen a warning is given in the dialogue box.

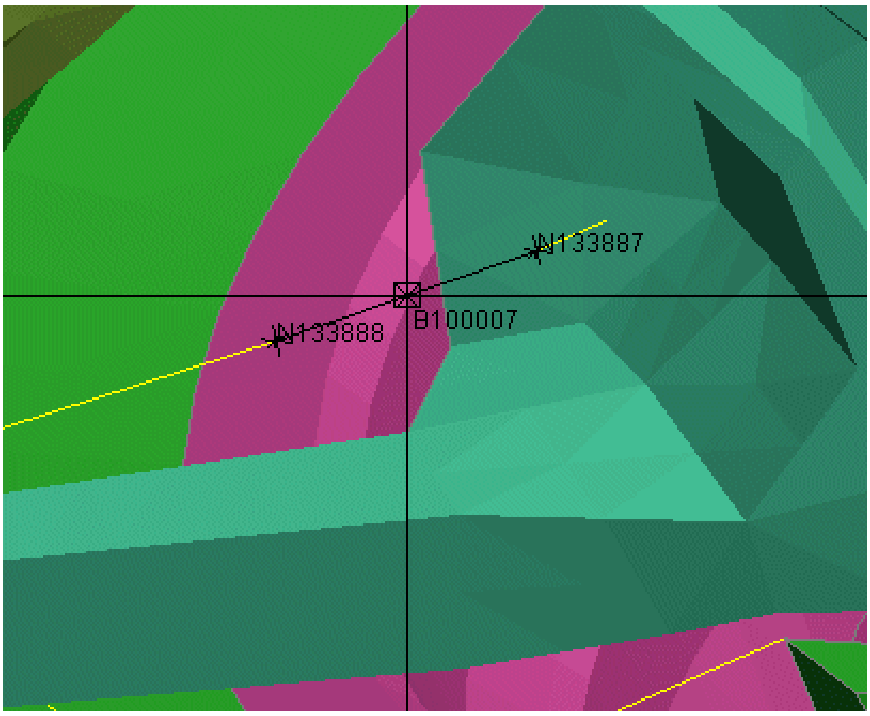


Set CN This option will not apply if the user has pressed CN in the view control panel. Otherwise, the centre of rotation/zoom will be set to match the cross-hairs if the sketched item is on screen. The option will automatically activate if an off screen item is brought into view and similarly de-activate if an on screen item is moved out of view. Pressing **CN** will unset the centre.

Label sketched This will label the sketched item in the form Mn/Sn.

Label nodes of sketched elem This will label the nodes for some items if the zoom is deemed sufficient to show them.



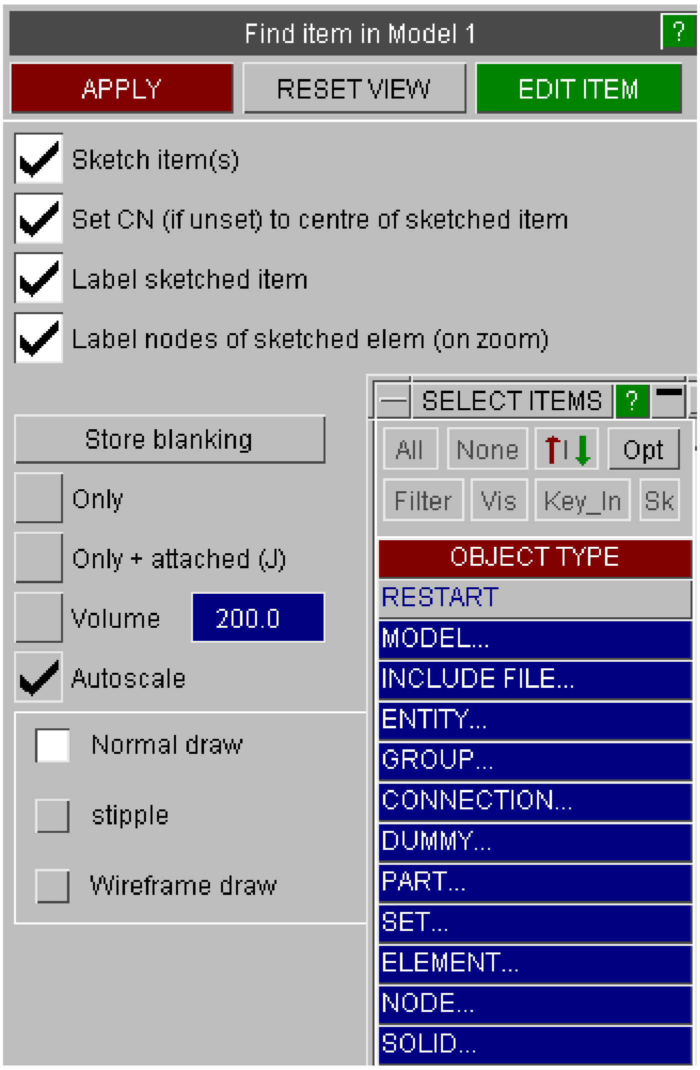


Press of Delete key or redraw by using Li/Hi/Sh will remove the cross-hair, clear the sketching and unset the CN.

If you sketch multiple items simultaneously, Primer will work as before and only apply a sketch.

If a cross-hair unexpectedly does not appear when sketching from object menu, it is most likely that more than one item is selected. Try clearing the selection with **None** and then reselecting.

6.13.2 Finding an item

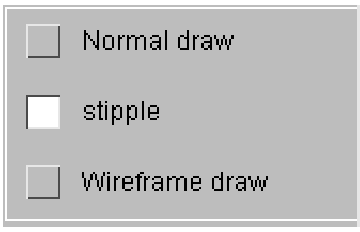


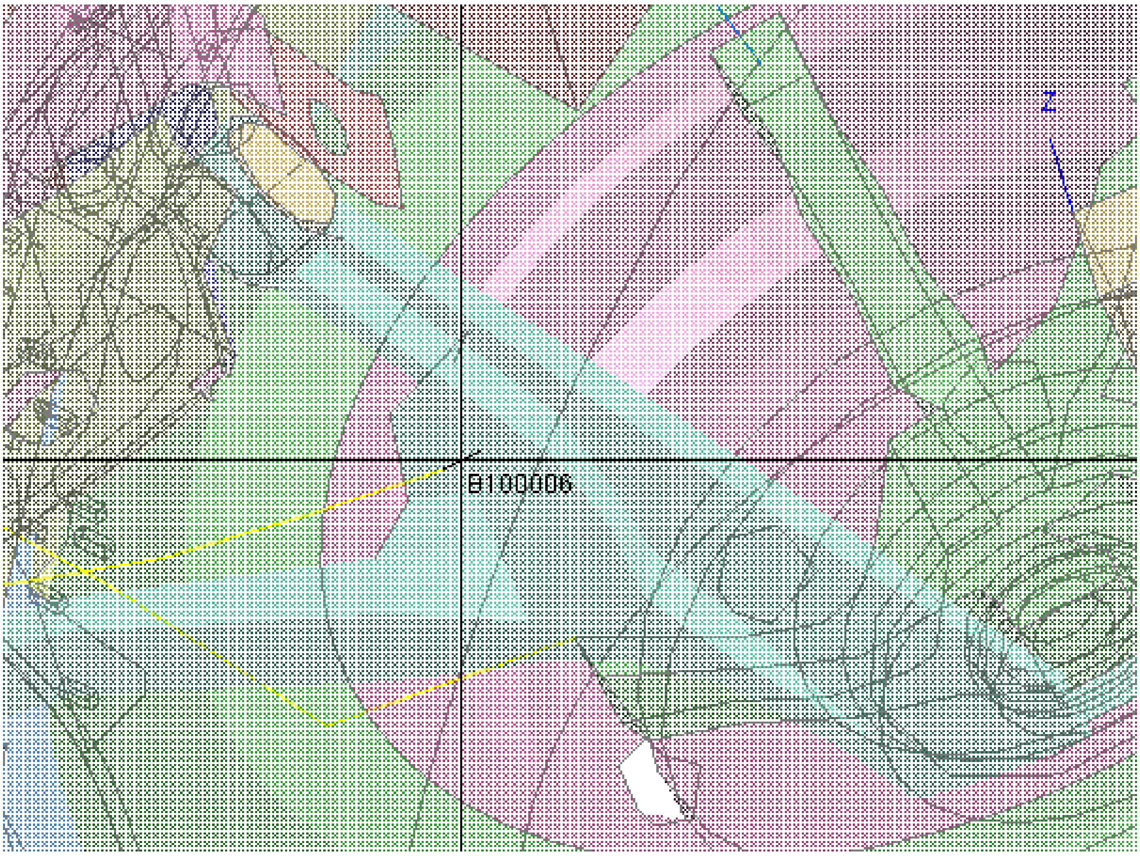
In its default mode, **FIND** is a sketch function for a single item with a generic object menu. The sketch options may be set exactly as described above.

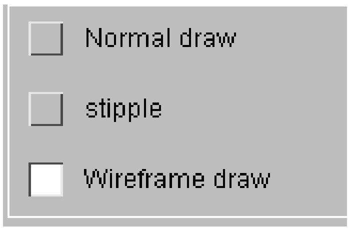
FIND also offers a number of additional features.

Stipple and wireframe draw

The drawing mode, normal draw by default, can be set to use stippled draw (a form of transparency) or wireframe draw to enable the sketched item to be seen. This is useful if the item is enclosed.






Normal draw
stipple
Wireframe draw

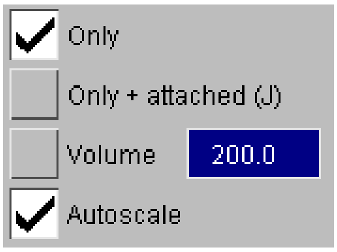

B100006

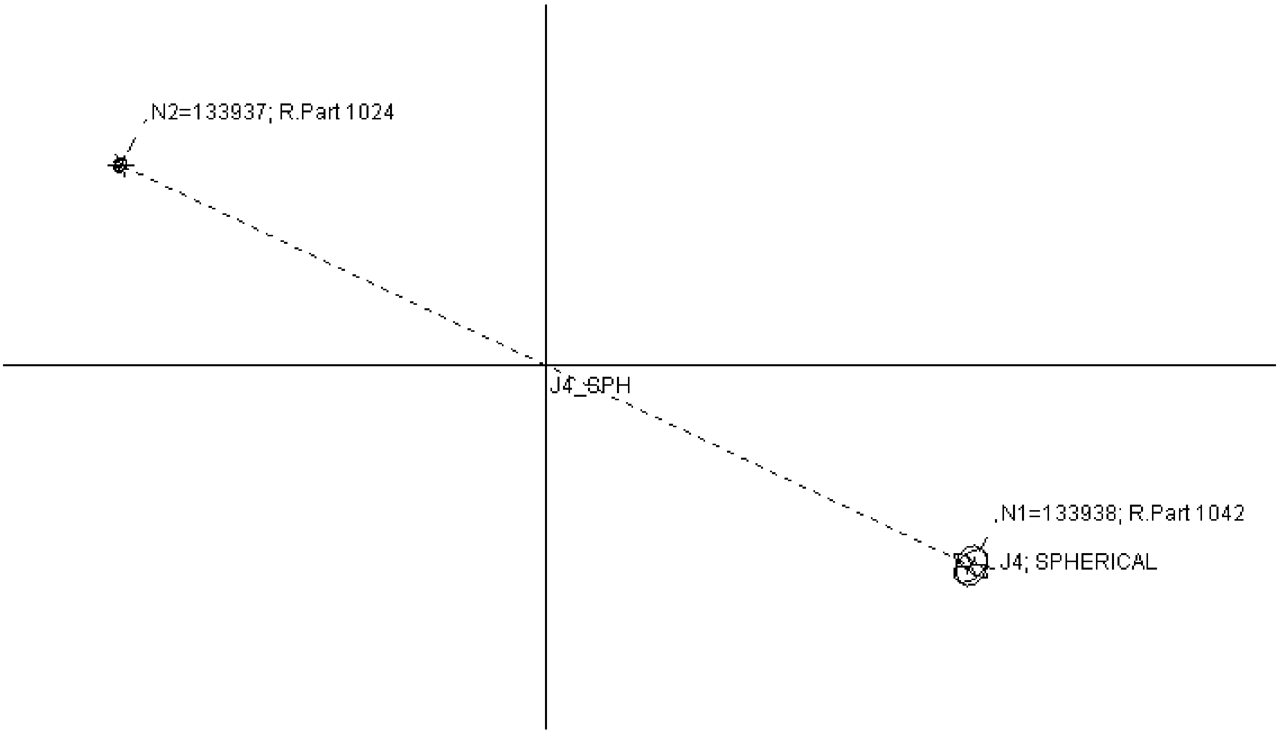
Find with Only

Before applying automatic blanking you may wish to store the current blanking status of the model by a press on **Store blanking.** On completion of the Find operation you can use **RESET VIEW** to restore the image.

3 methods are available Only, Only with attached and a Volume clipped view. By default autoscale will be applied but this can be de-activated.

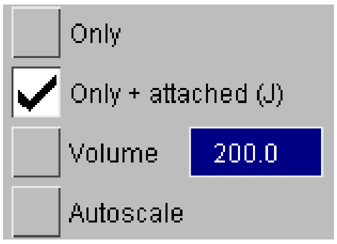
Only on an ill-conditioned spherical joint gives the following image.

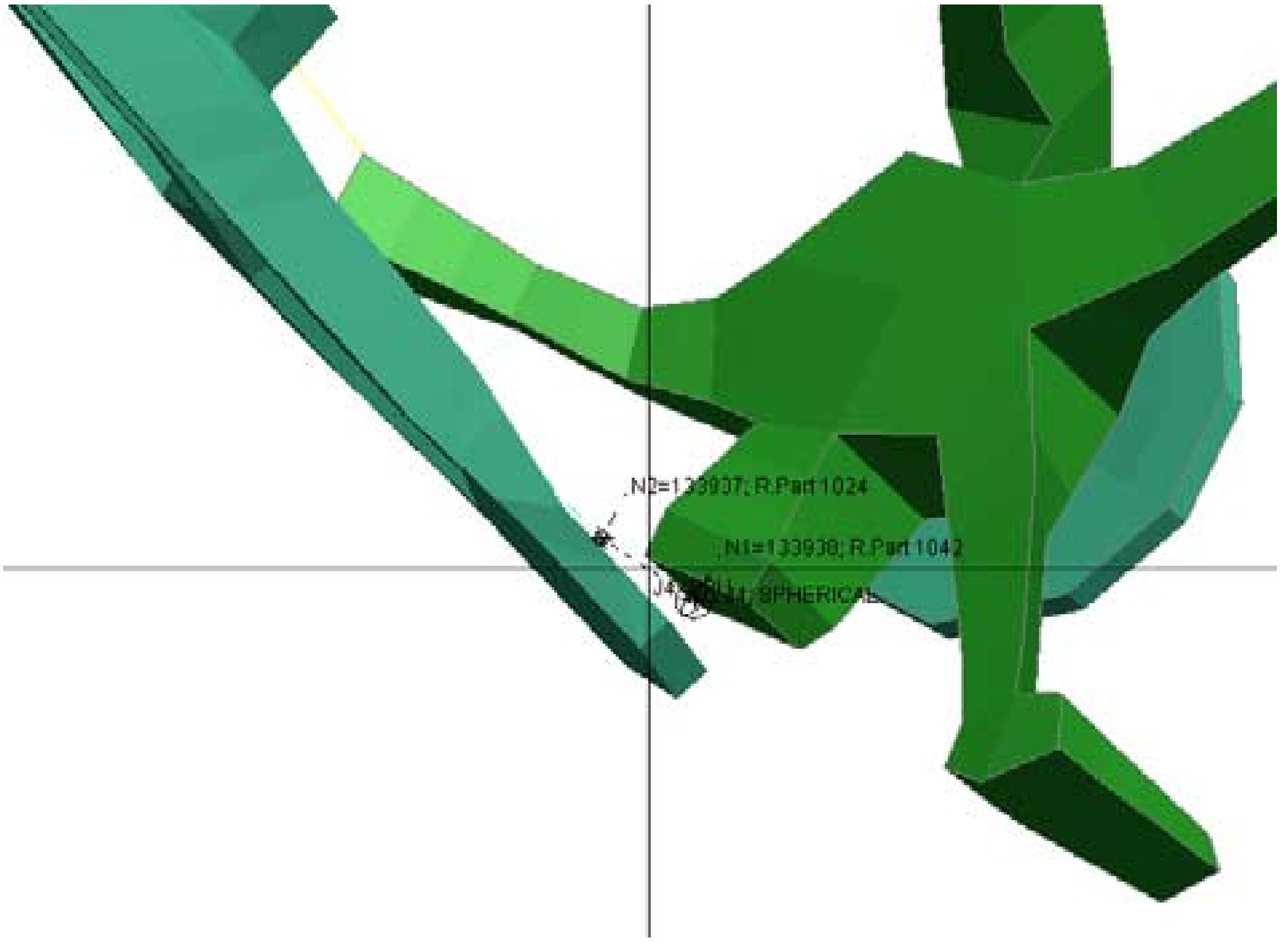




Find with Only & Attached

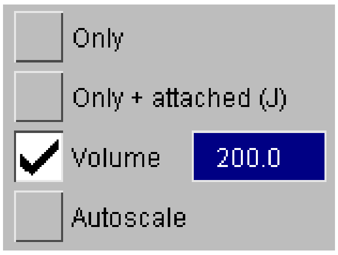
Switch to Only + attached to find the rigid bodies on the joint

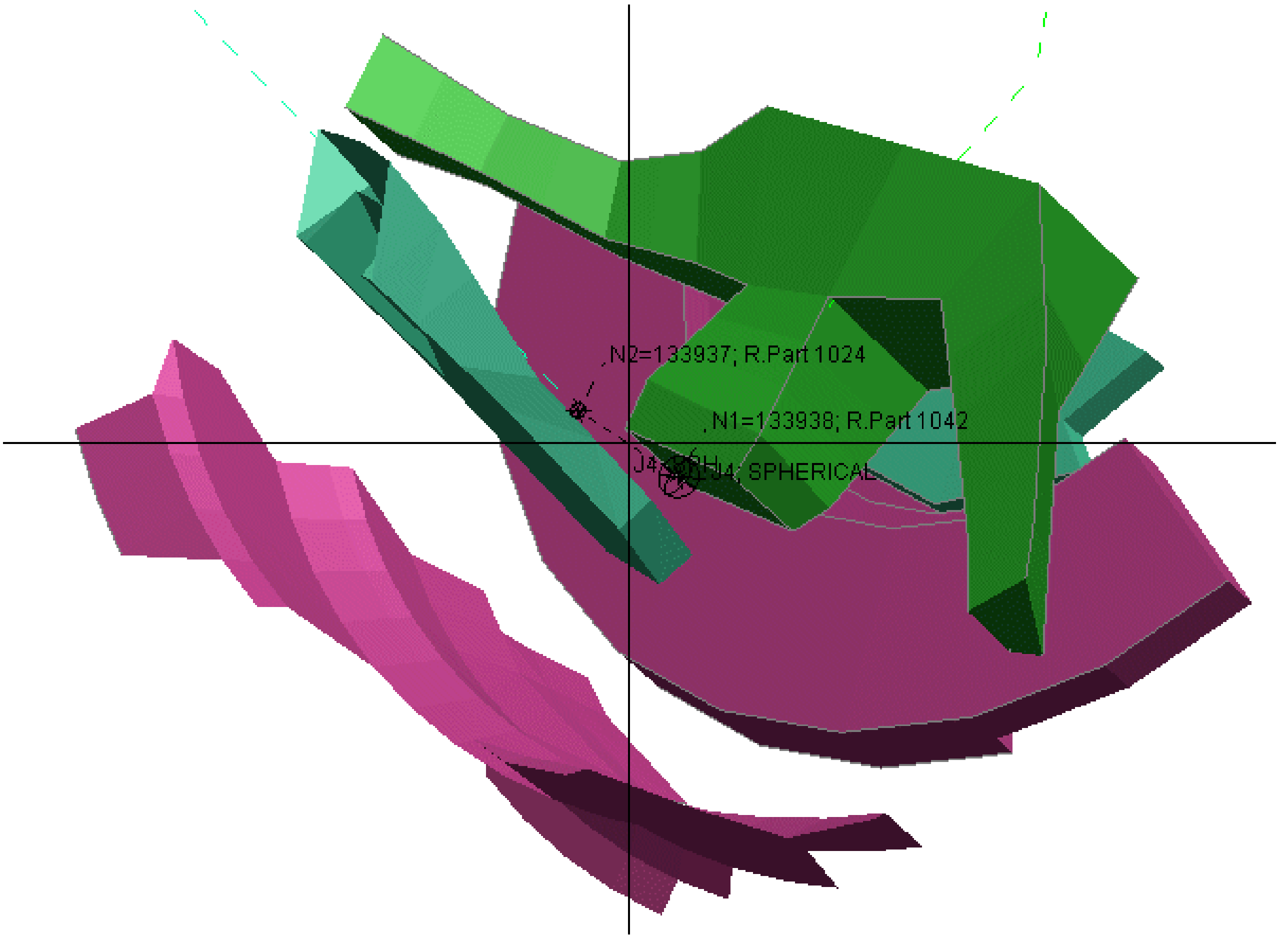




Find with Volume Clipping

Volume clip view will show everything that is attached to nodes that lie in the spatial volume NxNxN centred on the selected object, where the value of N is controlled by the user.





Editing with Find

If there is an active item as displayed on the find header, EDIT ITEM may be used to open an edit panel or keyword editor.

6.14 FMH Free Motion Headform

This feature has been written in order to position the freemotion headform according to FMVSS 201. Firstly at least one **`*HEADFORM`** definition must have been read in from file. The **`*HEADFORM`** card is similar to the **`*DUMMY`** definition and contains a number of keywords (described below) with information required by the positioner. These appear after the **`*END`** card and are ignored by LS-Dyna but used by PRIMER. An example of a headform tree file is given in appendix X

`*HEADFORM_START`

The headform label and title.

`*REF_POINT`

This is a node label, already exiting in the model, about which the headform will be rotated.

`*UNITS`

The mass, length and time units used in the model (same options available as for a *DUMMY definition).

`*COMPONENTS`

The part set which makes up the headform definition, the part on the headform to be used in the contact definition and the label of this contact definition.

`*TARGET`

The target definition at which the headform is currently positioned. Blank if no target definitions exist in the model or the headform is not currently in position.

`*AXES`

The label of a **`*DEFINE_COORDINATE_NODES`** definition already existing in the model to define the headform local co-ordinate system.

`*HEADFORM_END`

The end of the headform definition.

Along with the **`*HEADFORM`** definition another keyword has been included to store information regarding the target points in the model.

`*TARGET_POINT_START`

The first line contains a label, an acronym (as defined in FMVSS201) and an optional title. The second line contains the co-ordinates of the target. The third line contains the minimum and maximum horizontal angles. The fourth line contains the impact velocity for this target point, the part set to be used as the slave side in the contact definition and the current headform position number (see below).

`*HEAD_POSITION`

At a given target point a number of different angles are normally investigated. Any number of unique positions can be stored with each target point to facilitate moving the headform about in the model. This keyword contains a label and a title, the co-ordinates of the headform reference point, the horizontal and vertical angles, a flag to indicate where the horizontal angle is in the allowable range and a positional node ID.

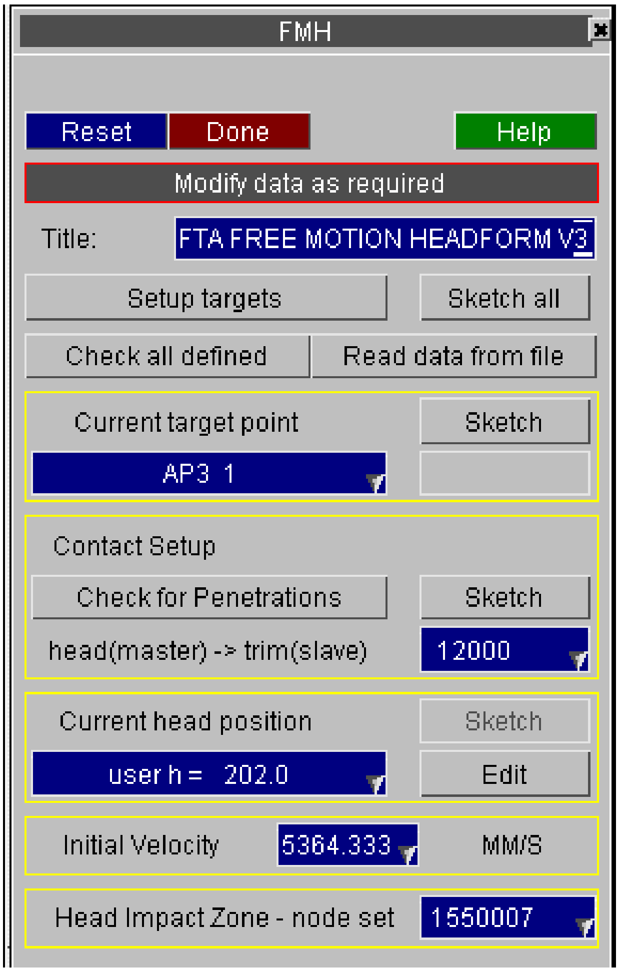
`*TARGET_POINT_END`

Ends the target point definition.

6.14.1 Positioning the headform

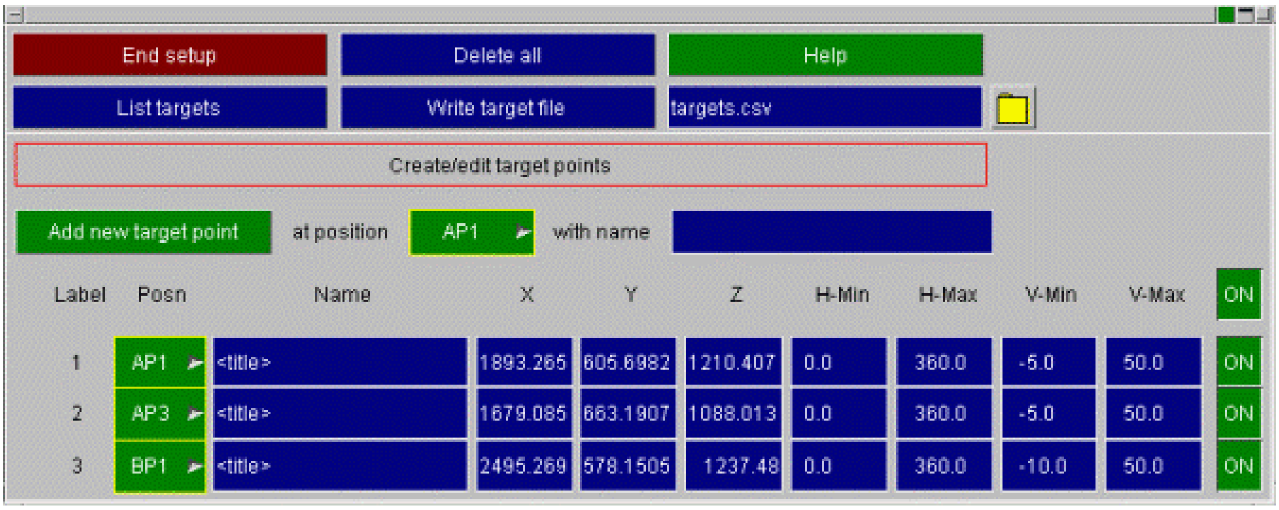
The figure below shows the main headform-positioning panel. This has been designed for use from top to bottom to create a run-ready input deck. A target definition must exist in the model in order to be able to position the headform. This can be created with the **SETUP_TARGETS** button.

If multiple headform definitions exist in the model the definition to be positioned must first be selected.



6.14.2 Setting up Targets

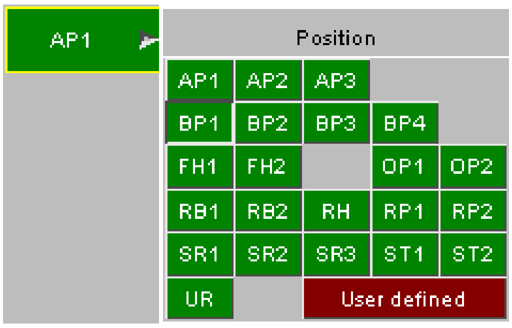
The **SETUP_TARGETS** button accesses the model target database.



You can have as many target points in your model as you want. A scrolling list shows all of the points. Additionally you can have more than one point at the same position.

When a target point has been assigned a head position it’s target file button (on rhs) will become active. You can then write a csv targetting file which can be used by the command line build models from csv file function (see Appendix).

Adding a new target point



To add a new target point:

1. select the position name you want the point at using the popup menu shown on the right
2. enter a description/name for this point
3. Press **ADD new target point**.

The point will be added to the list of available target points. You can then modify the point coordinates and min and max angles.

Removing a target point

To remove a target point use the popup on the required target point and press **Remove target point**. If the target point is currently in use by a headform you will be asked to confirm removal of the point

Sketching a target point

To temporarily sketch/draw a target point on the screen use the popup on the required target point and press **Sketch target point**. If you want to see the target points at all times then you want to turn target point drawing on instead of sketching them. See drawing and labelling target points below.

Changing the position of a target point

To change the location/position of a target point either:

- type the new X, Y and Z coordinates into the test boxes.
- use the popup on the required target point and press **Pick new location**. You can then select a node from the screen. The coordinates will be taken from that node

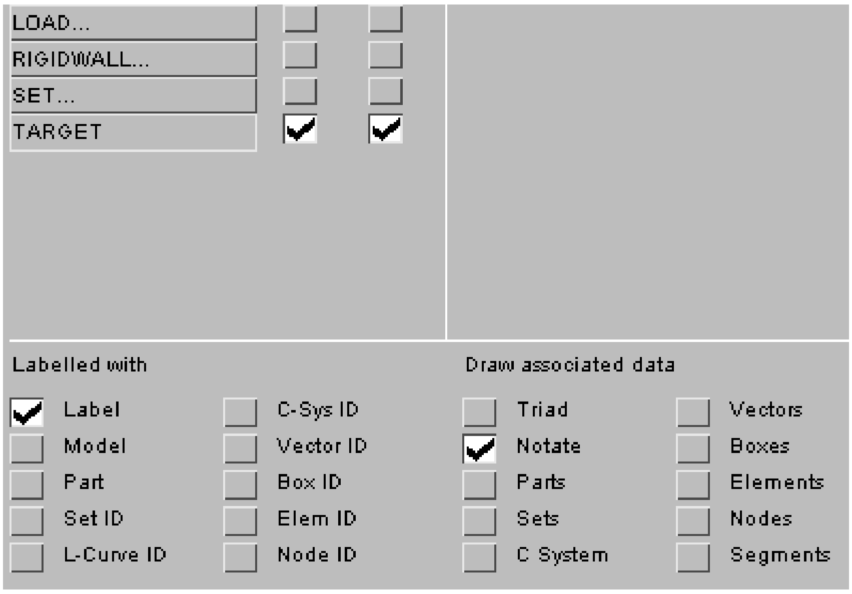
Changing the size of target points

The size of target points can be changed in the main **OPTIONS** panel.

Drawing and labelling target points

Target points can be drawn and labelled just like nodes and elements. They are turned on in the **ENT**ity Viewing .

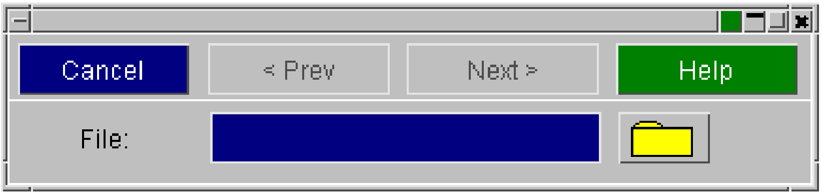
In addition to drawing and labelling target points the **NOTATE** function can be used. If this is turned on then the name and position of each target point is written on the screen as well as the target point number.



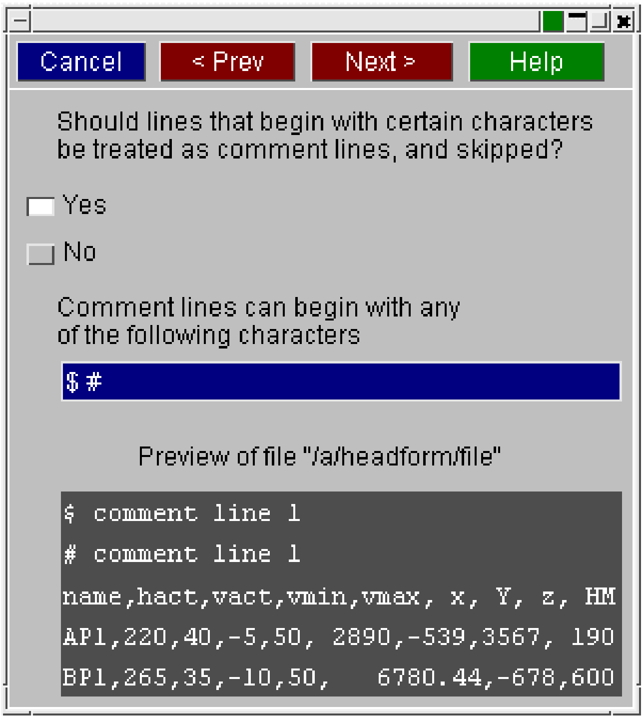
6.14.3 Reading headform position data from a file

You can read in headform position data from an external delimited file by clicking on the **Read data from file** button. This will guide you through a series of panels where you can specify the file type and what data you wish to read from the file. The file would generally be of a CSV format with each row containing information for a target point/head position. The sequence of panels is:

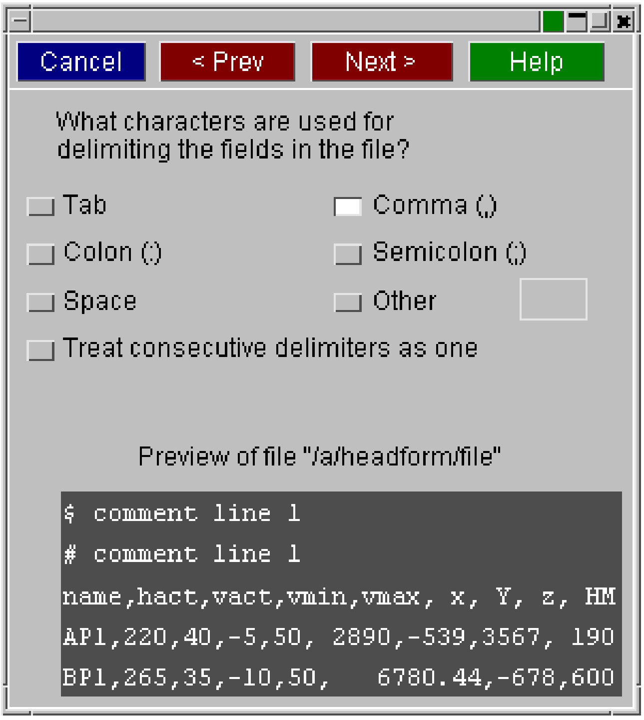
Select the file to read.



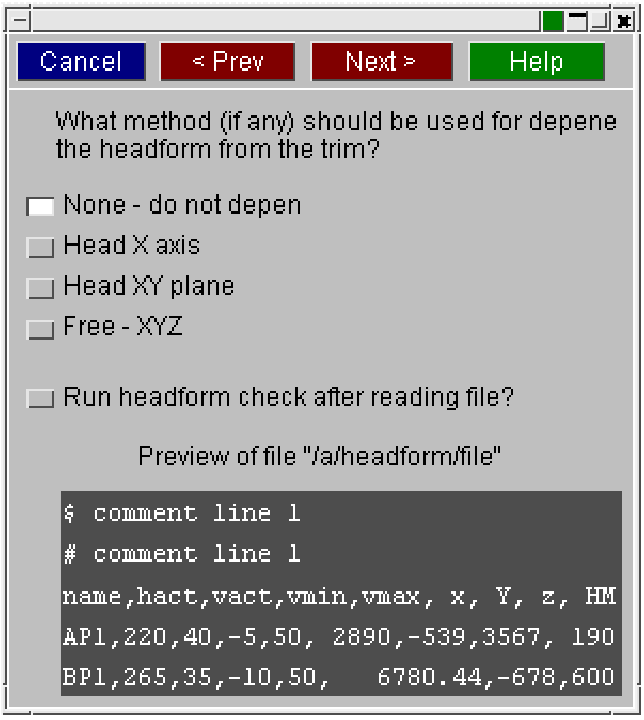
Specify any lines to ignore by defining characters at the start of the line that signify a comment. Note that a preview of the file is shown at the bottom of the panel.



Specify the delimiter for the data in the file.



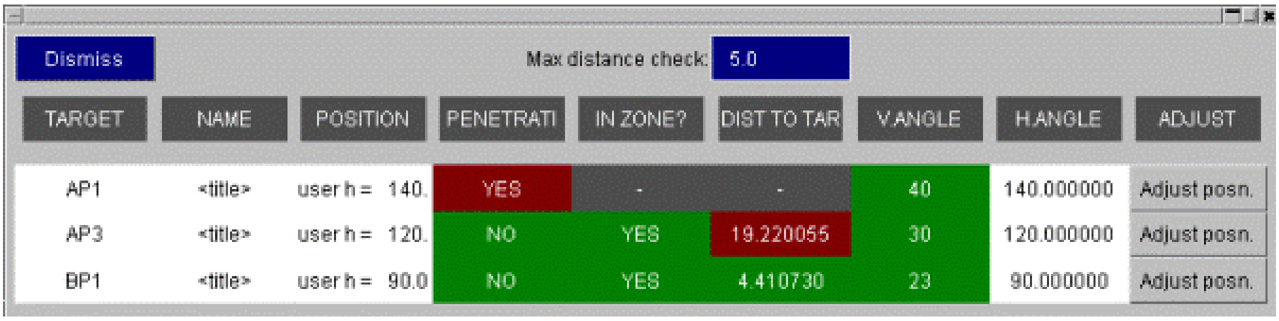
Choose depenetation options for when the data is read in. You can choose to automatically depenetrate the headform from it's starting position in different dergrees of freedom. You can also choose to run the headform checks after reading in the data (see section 6.13.4).



Finally the data is presented to you in a table format. If there where suitable titles in the input file, Primer will have attempted to guess the type of data in each column. If not, you can specify this on the panel by right clicking on the column headers and choosing the type of data from the resulting popup. After the columns have been assigned, click on **Apply** to read in the data and setup headform position information from the data. Note that the minimum that has to be contained in the file is the x, y, z coordinates of the target point.

6.14.4 Checking defined targets

You can check the status of all currently defined target points/positions by clicking on the **Check all defined** button. This will check each position in turn and then report the results to the screen in a table.



The check panel reports a number of things:

TARGET - The target point.

NAME - The target point name.

POSITION - The position within the target point definition.

PENETRATIONS? - Reports whether the current position of the headform means there are penetrations between the headform and the trim.

IN ZONE? - Reports whether the initial contact point of the headform to the trim is within a user defined head impact zone (see below).

DIST TO TARG - Reports the distance from the initial contact point to the defined target. Note the **Max distance check** value above this column determines whether the value in the table is shown in red or green.

V-ANGLE - Current vertical angle of the headform. This is checked against defaults or user defined values on the target setup panel.

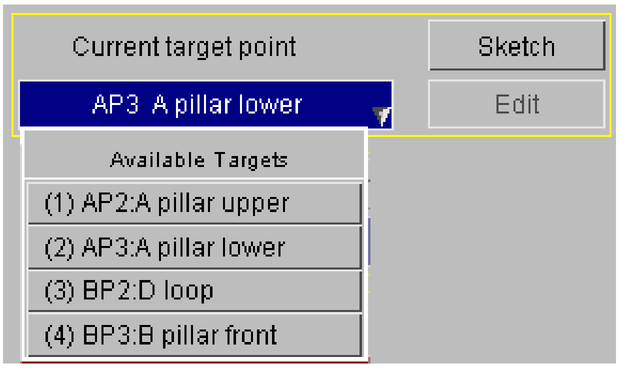
H-ANGLE - Current horizontal angle of the headform.

ADJUST - Opens up the position editing panel specific for this position on the table. This allows you to modify the position. After clicking **Done** on this panel you are returned to the check table and the details for this row are updated.

6.14.5 Selecting a target

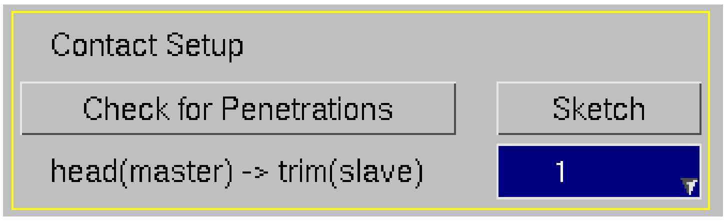
Clicking on the red button on the left-hand side activates a target point. Clicking on the button when it is green will remove the target point. Once active the required information can then be input and the target selected for positioning the headform via the popup on the main positioning panel.

The **NODE** buttons on the target database panel allow the user to select a node in the model from which the co-ordinates are taken for use by the target.



When a target point is selected for use the headform is moved to that target point with the headform reference node located at the target co-ordinates. If a position definition (see below) exists and has been previously selected, the headform angles will be determined by that otherwise it will default to 0º vertical angle and the target minimum horizontal angle.

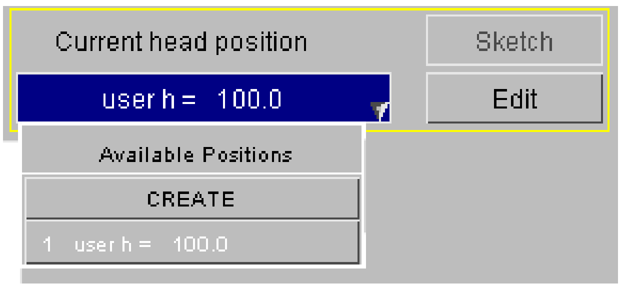
Next the contact needs to be set-up. The contact definition exists in the headform tree file, under HEADFORM_START -> COMPONENTS. This contact must be created or edited to suit the headform. It is best to use an automatic surface-surface contact defined by part set, with the master side defined on the headform.



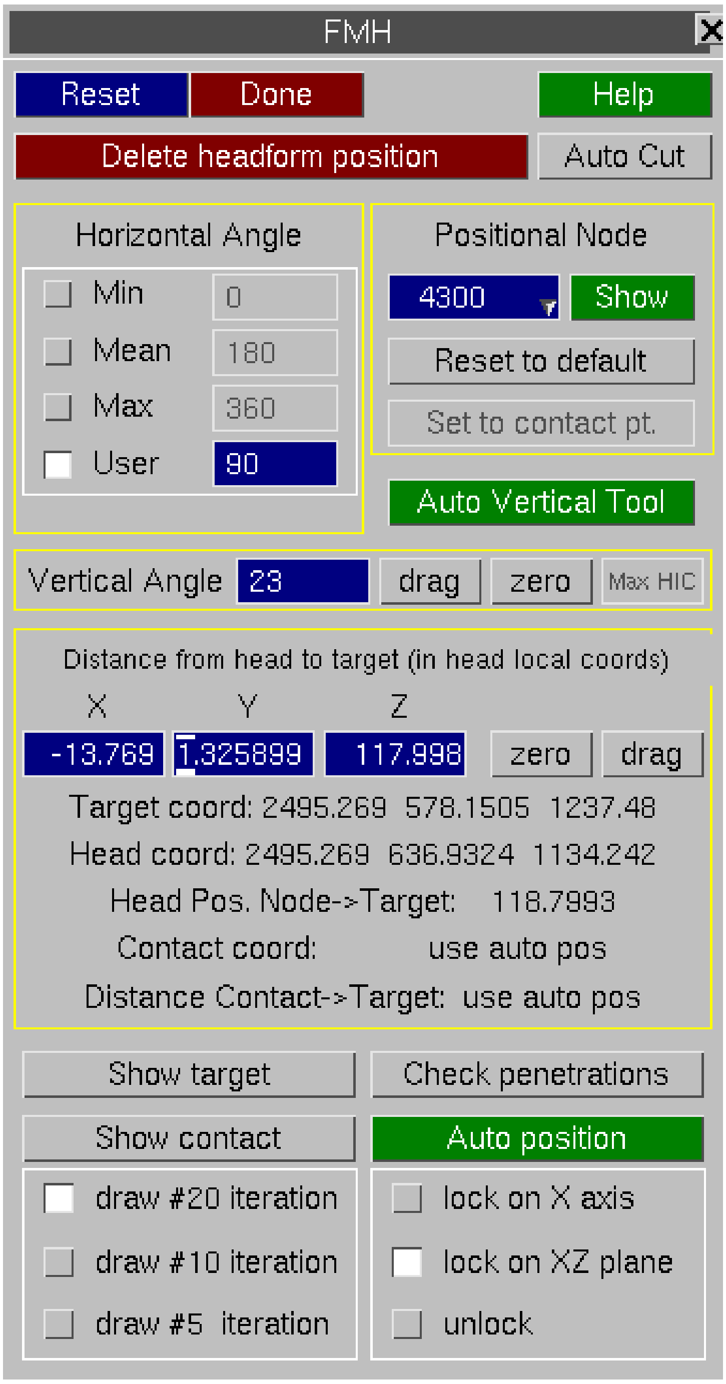
The slave side of the contact may require editing so that it contains the correct parts for each position. Alternatively a set can be defined with all the relevant parts and a box added to the contact. As the head is moved to different positions the box will be translated with it. In FMH mode the function which enlarges boxes as they are rotated is suppressed as this was found to enlarge the contact box excessively.

The contact can be edited and sketched in the usual way and also checked for initial penetrations. Those elements that are within a shell thickness apart or actually crossing over are sketched in white.

The headform is now positioned relative to the selected target point. This requires a position definition to be created. If position definitions already exist within the model for the selected target point then they can be selected via the popup.



Creating or editing a position definition will bring up the panel shown in the panel below. This will create a unique position for the head form that is stored in the keyword deck. This can then be re-used at a later stage without having to re-create it.



Delete headform position will remove the current position and return to the main panel. **Auto cut** will turn on PRIMER's cutsection feature and automatically orient the section through the centreline of the headform. This can aid in headform positioning.

The horizontal angle is the angle between the global X axis and the headform local X axis in the global XY plane. This can be set to the minimum, mean or maximum angle as defined on the selected target. If none of these angles are required a user defined angle can be entered which is required to be between the minimum and maximum.

The positional node is used for positioning and rotating. By default, when setting the vertical angle, Primer will rotate the headform around the head reference node. This can now be changed to any node on the headform. The default can be reset by clicking on **Reset to default**. After positioning the headform automatically, the initial contact point is known. The positional node can then be set to the initial contact point by clciking on **Set to contact pt**. With an alternative to the headform reference node chosen as the positional node, the X,Y,Z didtance from the target point shown on this panel will now use the new node. Also, clicking **zero** will zero the headorm to the target point at this new node.

The Auto vertical tool can be used to automatically position the headform to its maximum vertical angle by simulating the rolling of the headform on the trim. More information on this can be found below.

The vertical angle is the angle between the headform local X axis and the global X axis in the local XZ plane. This can either be typed in or dragged into position using the **DRAG** button and clicking and dragging the headform in the graphics area.

Max HIC This will rotate the head from +25 to -25 of its current position and leave it at the angle which minimizes the spread of penetrations in the XZ cutting plane of the head. This is theangle at which the head is least able to roll on impact and consequently should give the highest HIC value. It is recommended that the head be positioned at <0 0 0> first (note the positional node is reset to the head reference node for this operation).

AUTO POSITION This function will attempt to position the head to minimize the distance between the initial contact point and the target point. This distance (contact->target) is measured in the head local YZ plane, i.e as viewed along the X axis (line of flight).

There are 3 modes for controlling the corrective motion of the head.

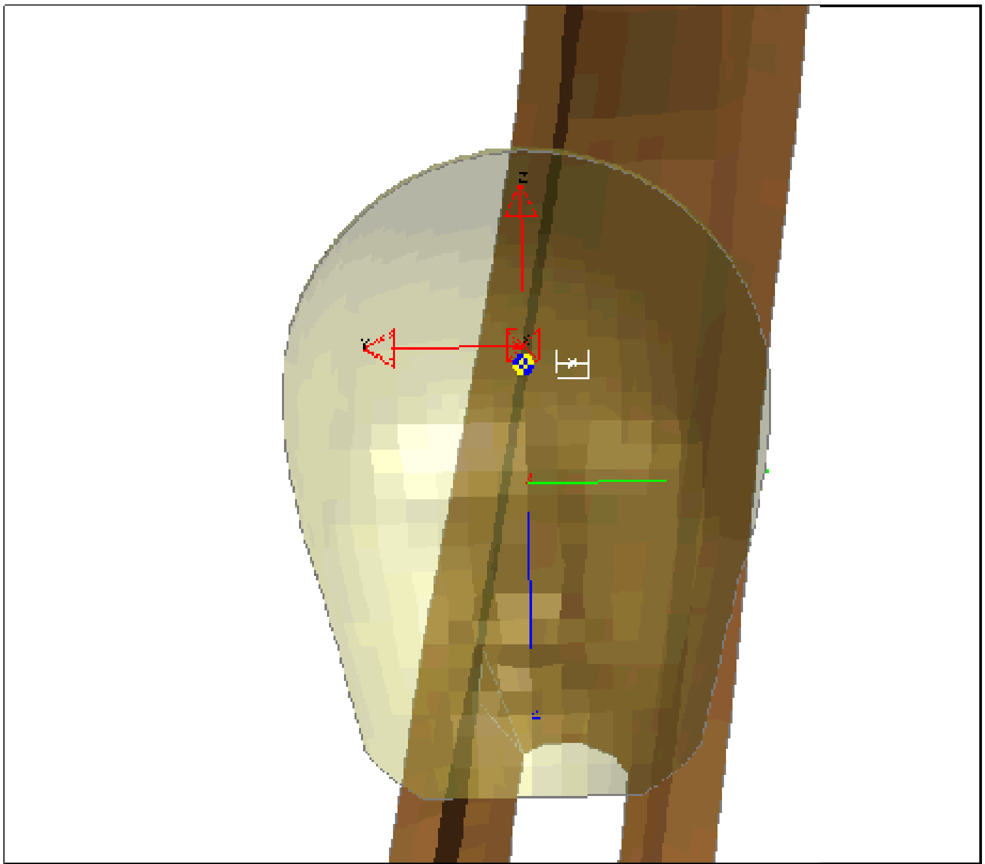
Lock on axis - in this mode the head will only be moved along its line of flight (local X), i.e.without varying local Y or Z coordinates

Lock on XZ plane - the head can move axially (local X) and up or down (local Z) but the local Y coordinate does not vary

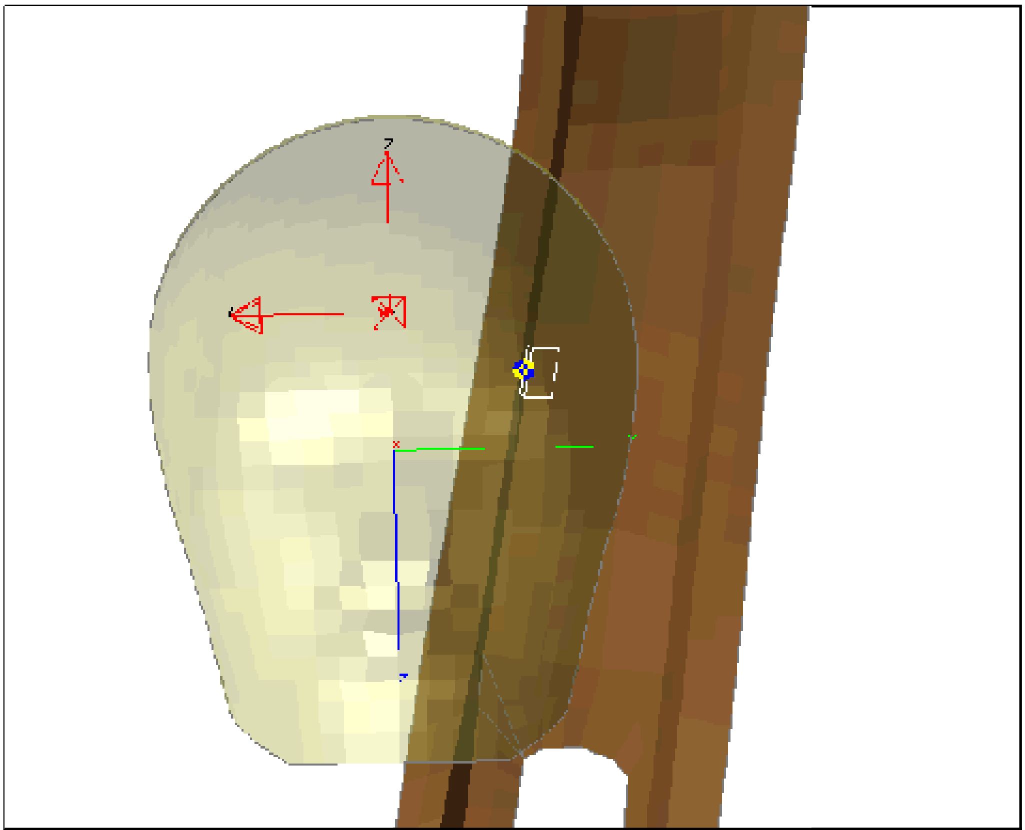
Unlock - the head can move freely

The example below shows how the different modes will position the head when faced with a "difficult" target point, which has been positioned in a re-entrant corner of the trim. In such cases, keeping the head locked to the XZ plane limits how close the initial contact point and target point can become. However, freeing the motion may result in excessive sideways movement and even an initial contact point outside the defined perimeter.

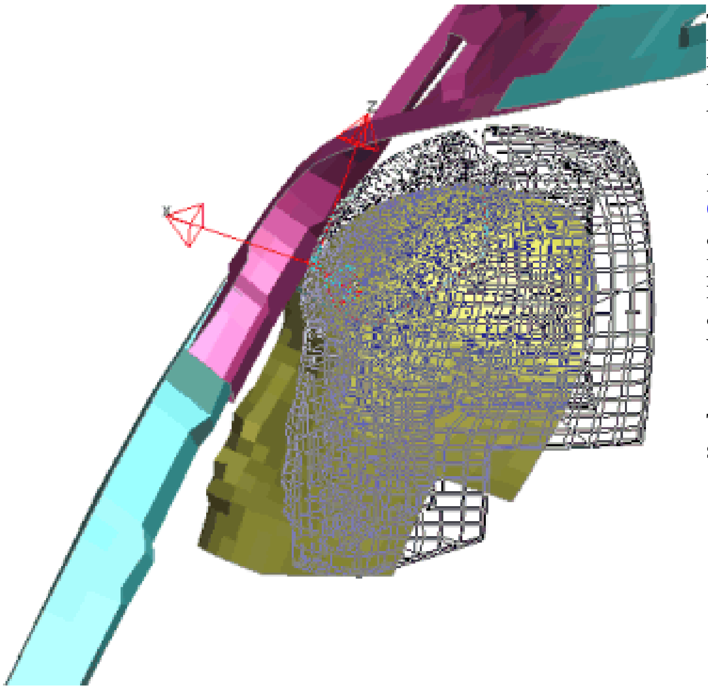
The automated positioning system is designed to assist in headform positioning, however, the user must user his own judgement about whether the iterative process has actually achieved the position most suited for test.



Head positioned with lock on XZ plane cannot initial contact get very close to target point.



Head positioned freely gets initial contact closely aligned but has moved sideways excessively.

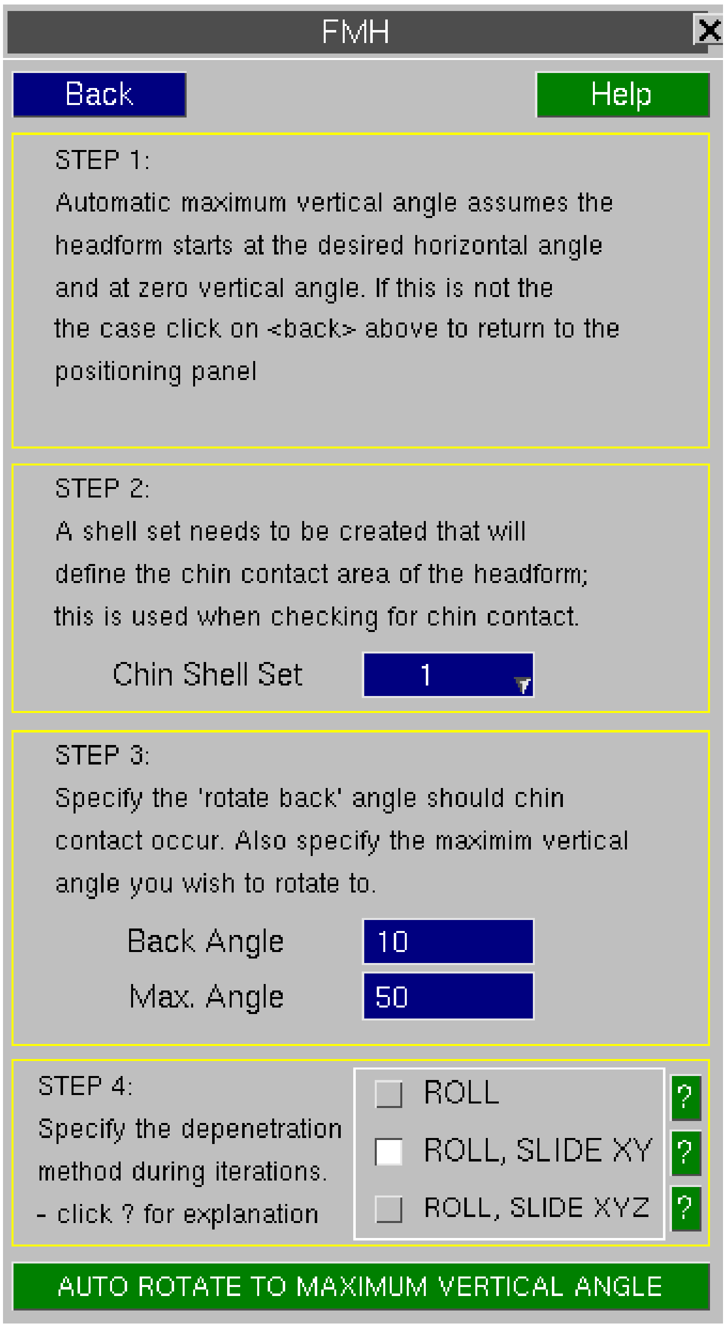


The headform can also be dragged in the headform local X, Y and Z directions using the left, middle and right mouse buttons respectively. Also its vertical angle can be dragged.

During the drag operations if the **CHECK_PENETRATIONS** button is made active then the contact is checked for penetrations each time a mouse button is released. The penetrations are sketched in white and the **CHECK_PENETRATIONS** is updated with a feedback message.

The contact and the target point can also be sketched at any time during these operations.

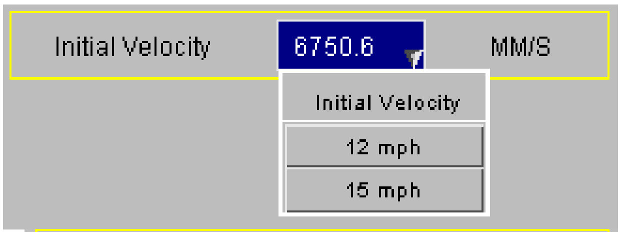
Auto vertical tool



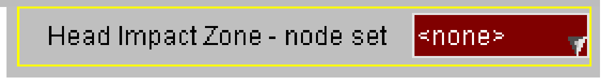
The auto vertical tool can be used to automatically position the headform to its maximum vertical angle by simulating the rolling of the headform on the trim. The user can set a maximum angle and a ’back angle’ which is used when contact occurs between the chin and the trim. This process works by starting off with the headorm in contact with the target point. It will then slowly rotate the headform (depenetrating along the way) until it reaches its maximum vertical angle or the chin touches the trim (whichever comes first). The headform will then rotate back by the desired back angle (usually 5 or 10 degrees), again depenetating the headform along the way.
Ensure the headform is positioned at the desired horizontal angle before using this tool. A shell set to define the chin area also needs to be created. The headform should also be positioned at the point of first contact as well, although Primer will run the automatic position process before entering this panel to find this point. As the headform rotates the contact point may change, and therefore the rotation centre point will change automatically. Also, the user can choose the method of head depenetration when rotating. With the depenetration method set to ’ROLL’, Primer will depenetrate the headform from the trim along the x-axis of the headform coordinate system. This simulates the rolling of the headform off the target point during rotation. With the ’ROLL, SLIDE XY’ setting, Primer will depenetrate the headform from the trim on the XY plane of the headform coordinate system. This simulates the headform rolling off the target point during rotation and then sliding back towards the target along the XY plane. With the ’ROLL, SLIDE XYZ setting, Primer will depenetrate the headform freely in all directions of the headform coordinate system. This simulates

the headform rolling off the target point during rotation and then being free to slide back towards the target point in X, Y and Z directions. **Auto rotate to maximum vertcial angle** will apply the process.

The headform initial velocity is automatically created in the headform local X axis direction. The popup gives the option of selecting the two common impact velocities (in miles per hour) or the velocity can be typed in (in model units).



You can specify a node set to define the headform impact zone. This is used in the headform checking panel described above to ensure that the point of first contact is within the impact zone



6.15 GROUPS

Groups can be used in PRIMER to collect things together. Anything that has a label can be put into a group, for example `*PART`, `*NODE`. Things that do not have labels such as `*CONSTRAINED` and `*BOUNDARY` cannot be grouped.
At present the only use for groups is for assigning mass. In future releases of PRIMER groups may have other uses. When a group is used (for example in assigning mass all we are really interested in is structural items to add mass to) PRIMER will automatically calculate the contents of the group. If the group contained part 1, PRIMER will automatically find the elements that are in part 1 and then the nodes that are on those elements.

6.15.1 Group I/O

All Groups in the model may be written to an Ascii file using the **EXPORT** function. Similarly an ascii groups file may be read in using **IMPORT**. The **PART_LIST** function writes an explicit list of all parts contained in groups to ascii file: group_parts.asc.

You can make groups that you create in PRIMER available in D3Plot by writing a groups file.

6.15.2 Group format

Groups are written to the keyword deck after the `*END` keyword. As with any keyword lines beginning with a `$` are treated as comments and skipped.
It is strongly recommended that you do not edit the groups by hand. Use the group creating/editing capabilities of PRIMER.
Entities can be added to a group by one of three methods;

- Selecting all entities (optionally in a box)
- Selecting a list of entities (optionally in a box)
- Selecting a range of entities (optionally in a box)

The box is used for selecting a subset of the entities that are in the box. For example if the group contained '*all elements inside box 1*' then only the elements that are in box 1 will be used, not all the elements.

Entities can be added or removed from a group. For example you may want a group to contain '*all elements except those in box 1*'. You could do this by first '*adding all elements*' and then '*removing elements in box 1*'.

Example

```
*GROUP
$<title>
This is the example group title
$  <label> < visual attributes of the group>
         1 R255G000B000 50 CURRENT * * * * UNBLANKED
$ The following lines give examples of a group
$
$ Adding all parts to a group
PART ALL
$ Adding all parts to a group in box 1
PART ALL BOX 1
$ Removing all parts from a group in box 2. Note -PART
-PART ALL BOX 2
$ Adding a list of shells to a group
SHELL 1 20 23 100 200
$ Adding a list of shells to a group in box 2
SHELL 1 20 23 100 200 BOX 2
$ Adding a range of nodes (100 to 1000) to a group
NODE 100:1000
$ Adding a range of nodes (100 to 1000) to a group in box 2
NODE 100:1000 BOX 2
$ Removing a range of beams from a group. Note -BEAM
-BEAM 50:60
```

Order of calculation of Groups

The order in which things are added to/removed from groups is important. To stop any ambiguity the following order is used.

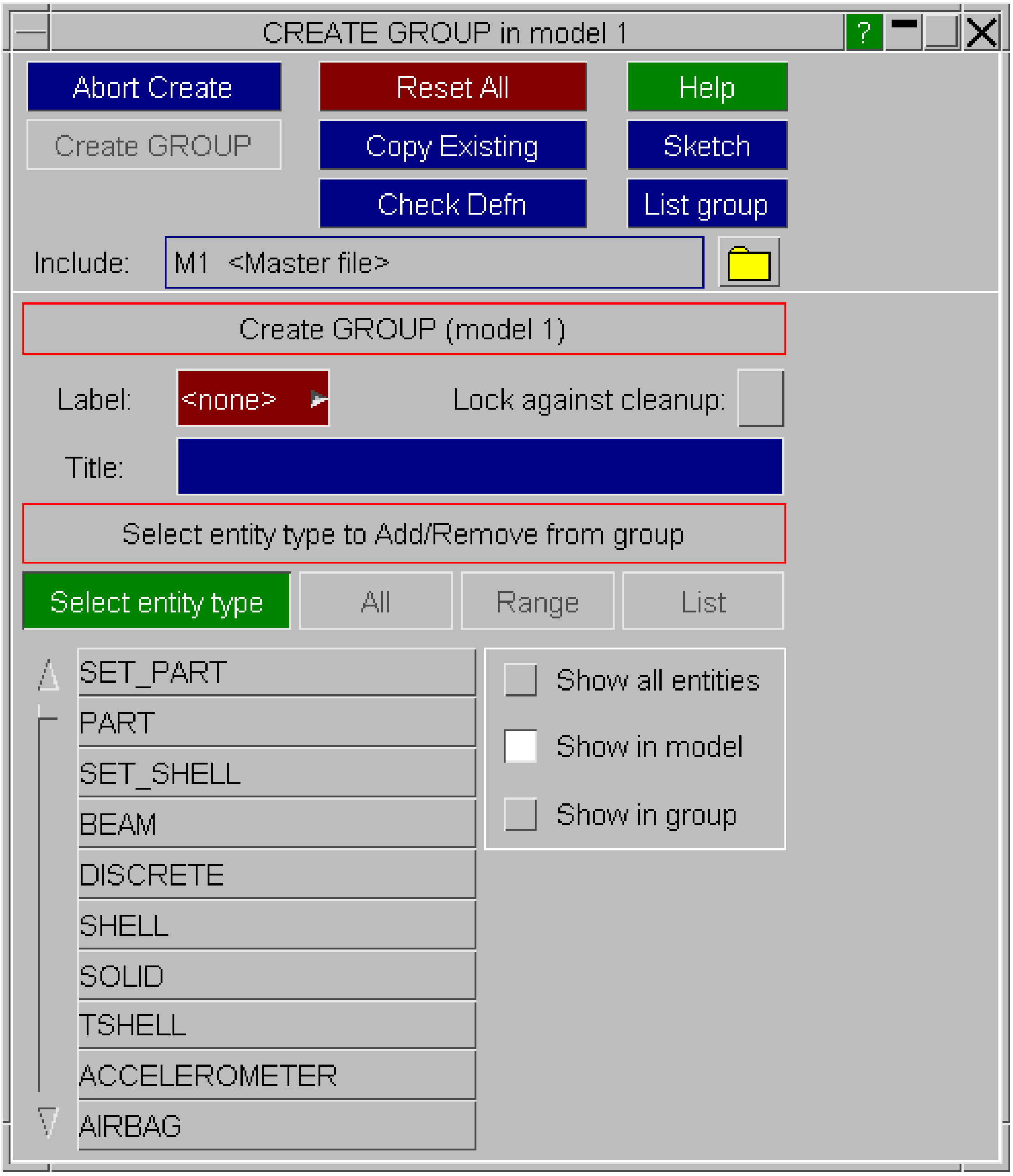
First, entities are added to groups in the following order:

- **SET_PART**
- **PART**
- other set types
- elements
- other entities alphabetically

Then, entities are removed from groups (if needed) in the same order.

6.15.3 Creating/Editing Groups

The initial screen for creating/editing groups is shown below. Before a group can be created a **Label** needs to be given. A **Title** can also be given if needed.



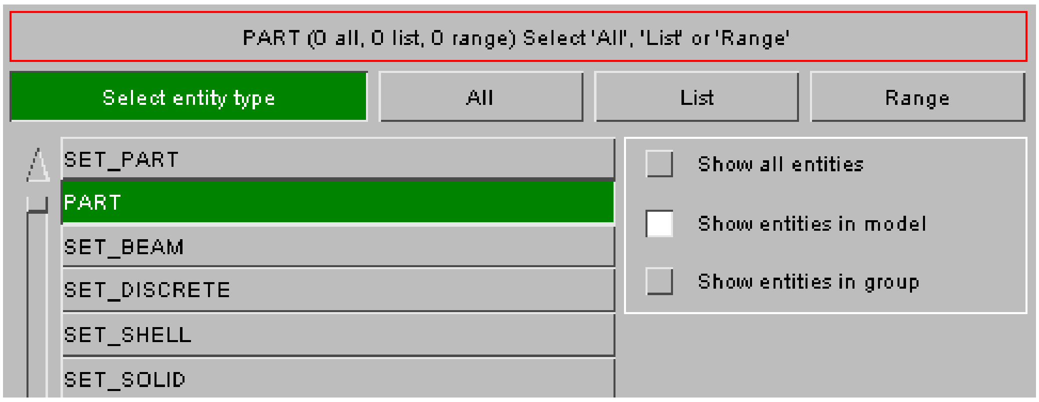
Once the label for the group is given the **CREATE_GROUP** button becomes active to create the group.

Locking the contents of a group against clean up

By default an entity that is not used in a model will be removed from a group in a model 'clean up unused' operation. The **Lock against cleanup** checkbox will prevent the contents of a group from being cleaned up. This is saved in the *GROUP keyword written after *END by PRIMER. For example, this could be useful if you want to make some sets in your model which you know will be needed at some time in the future but are currently not being used. If the sets are added to a group they will not be deleted by PRIMER.

Selecting entity type

Any entity which has a label can be added to a group. Before anything can be added to the group you have to choose the entity type you want to add. This is done with the list on the left hand side of the menu. By default all the entity types that are present in the model that you are editing are shown. This can be changed by using the radio buttons on the right. You can see all entity types (even if they are not present in your model), the entities in your model, or just the entities that are present in the group. Once the entity type is chosen the type is highlighted and the **All**, **List** and **Range** buttons become active to enable you to edit that type. For example if **PART** is selected:



The feedback box (shown at the top of the figure) changes to show what is defined in the group by **PART**. In this example above there are no entries by all, list or range so all are zero. As PARTs are added this will change.

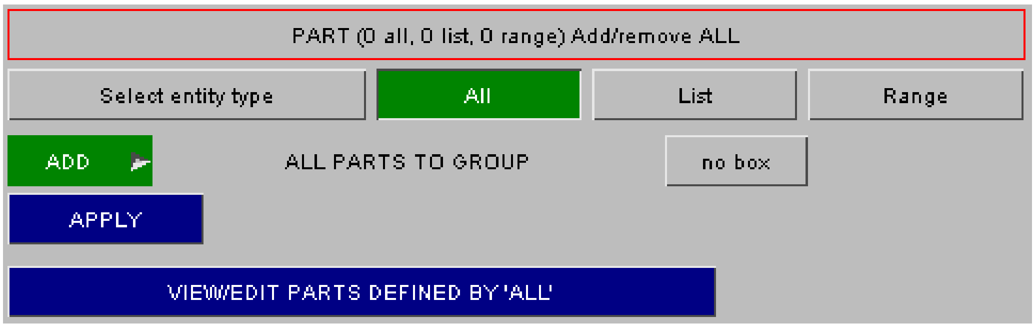
6.15.4 Adding, Editing and Deleting Entities in Groups

The following sections explain various ways of adding and manipulating entities within groups.

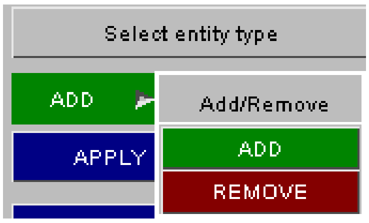
Adding entities by **ALL**
Adding entities by **LIST**
Adding entities by **RANGE**

6.15.5 Adding entities by **All**

Press the **All** button.
The screen changes to:

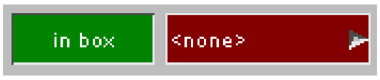


Selecting **ADD** or **REMOVE**

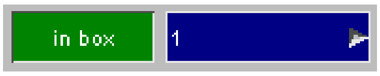


The green **ADD** button indicates that PARTs will be added to the group. If you want to remove PARTs instead of adding PARTs to the group, then the **Add/Remove** popup button can be used to change the action. If **REMOVE** is selected the button will change to red.

Using a box



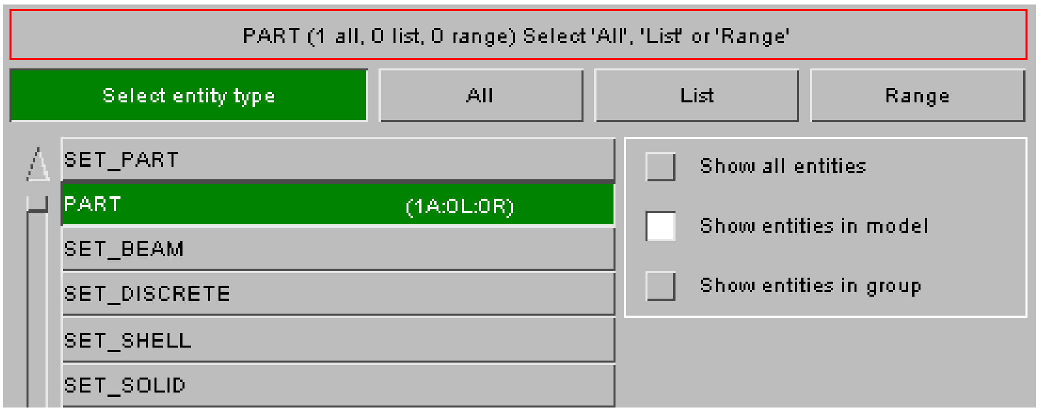
In the above example we are adding all PARTs to the group. We are not using a box to limit the selection as **no box** is selected, If instead we wanted to add the elements from all PARTs that are in box 1 this can be done by clicking on the **no box** button. The display will change as shown on the left.



Until a box is selected the **APPLY** button will be inactive. You can turn off the box selection again by pressing the **in box** button. Once a box is selected by either typing in the box number or using the popup menu it will turn blue and the **APPLY** button will be activated.

Saving into the group

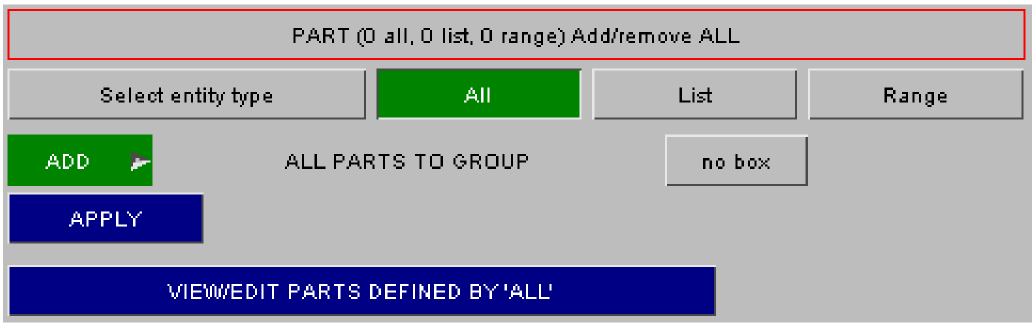
Once you have chosen to Add/Remove and if you want to use a box or not the selection can be saved into the group by pressing the **APPLY** button.



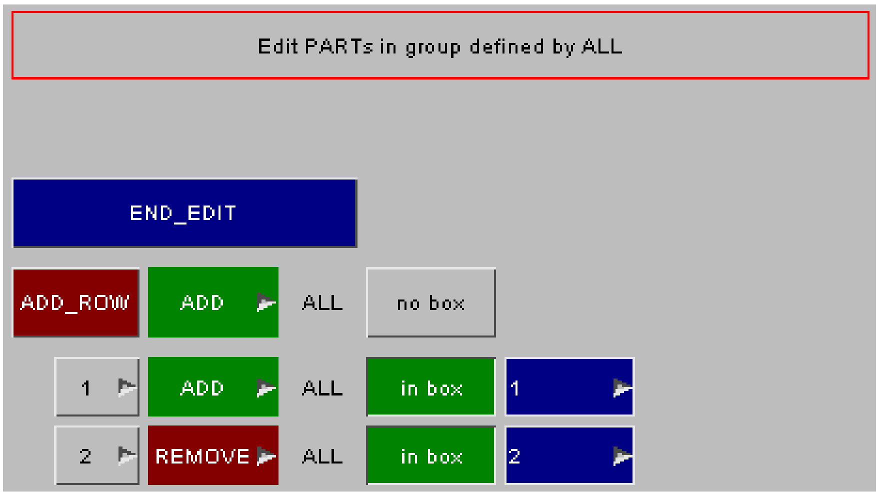
After the selection is saved to the group the screen refreshes back to the main screen (shown above). The feedback button (at the top of the image) changes to show that 1 selection of PARTs by all has been added. Additionally this is shown on the **PART** entity button. In this way you can easily see what entity types are present in the group.

6.15.6 Editing/deleting entities by **All**

Press the **All** button.
The screen changes to:

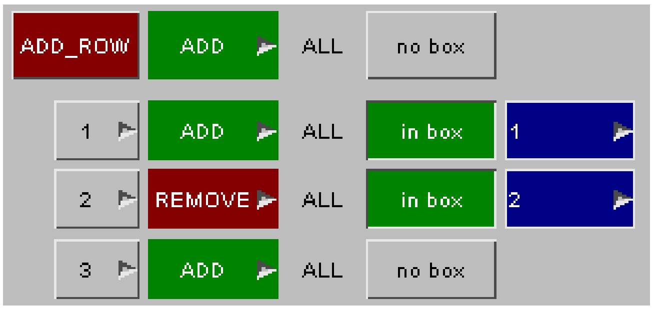


Press the **VIEW/EDIT PARTS DEFINED BY 'ALL'** button. The screen changes to the panel on the right. In this example there are 2 entries. Firstly ADD all PARTs in box 1. Secondly REMOVE all PARTs in box 2.



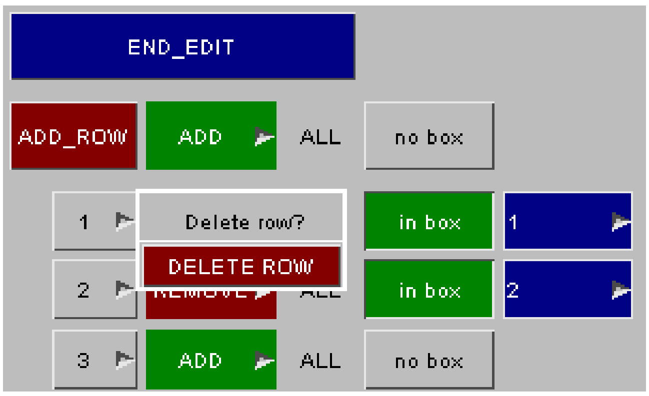
Adding an 'all' row using the group editor

A new entry can be added to the group by using the **ADD_ROW** button. The **ADD/REMOVE** and **box** buttons work in the same way as the panel to add entities by all. In the example above a new row 'ADD all parts' would be added to the group. A box would not be used as it is turned off. Adding the row would result in:

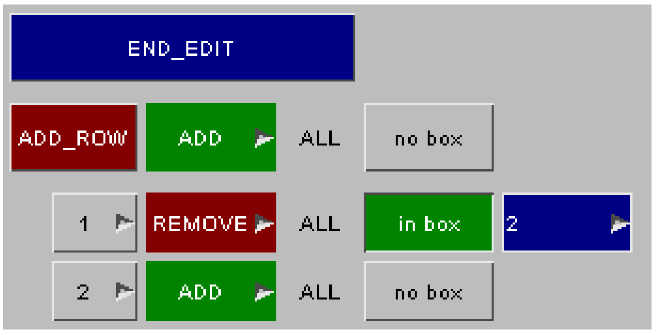


Deleting an 'all' row using the group editor

To delete an entry from a group use the **Delete row?** popup. For example, to remove entry/row 1 right click on the **1** button and select **DELETE ROW** from the popup menu.



The row is deleted and the remaining 2 entries are moved up the list.

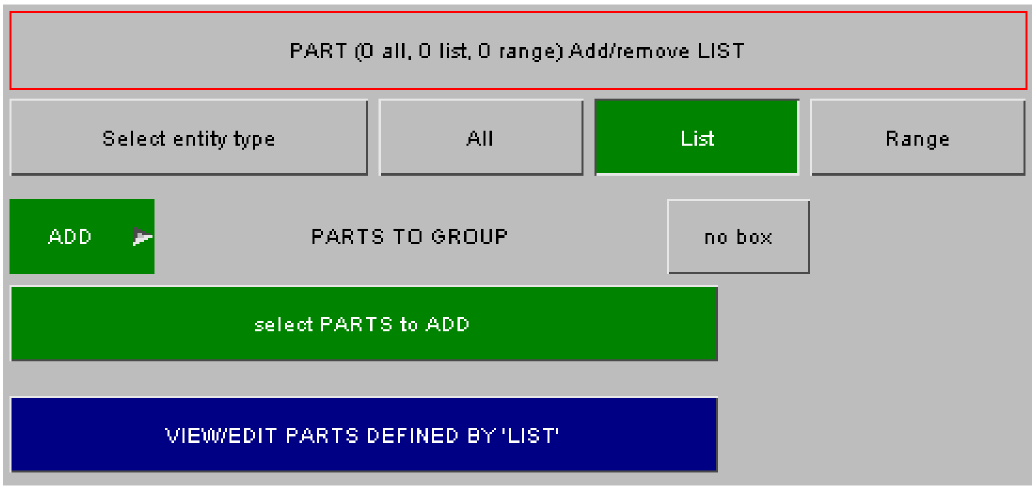


Editing an 'all' row using the group editor

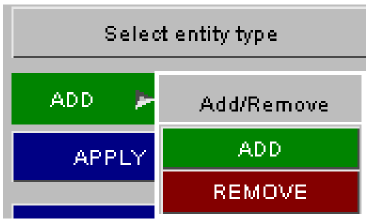
Each of the existing rows can be modified if needed. For each row the **ADD/REMOVE** and **box** buttons work in the same way as the panel to add entities by all.

6.15.7 Adding entities by **List**

Press the **List** button.
The screen changes to:

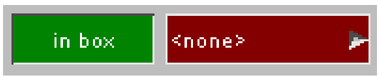


Selecting **ADD** or **REMOVE**

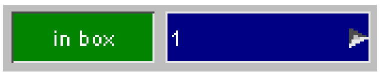


The green **ADD** button indicates that PARTs will be added to the group. If you want to remove PARTs instead of adding PARTs to the group, then the **Add/Remove** popup button can be used to change the action. If **REMOVE** is selected the button will change to red.

Using a box



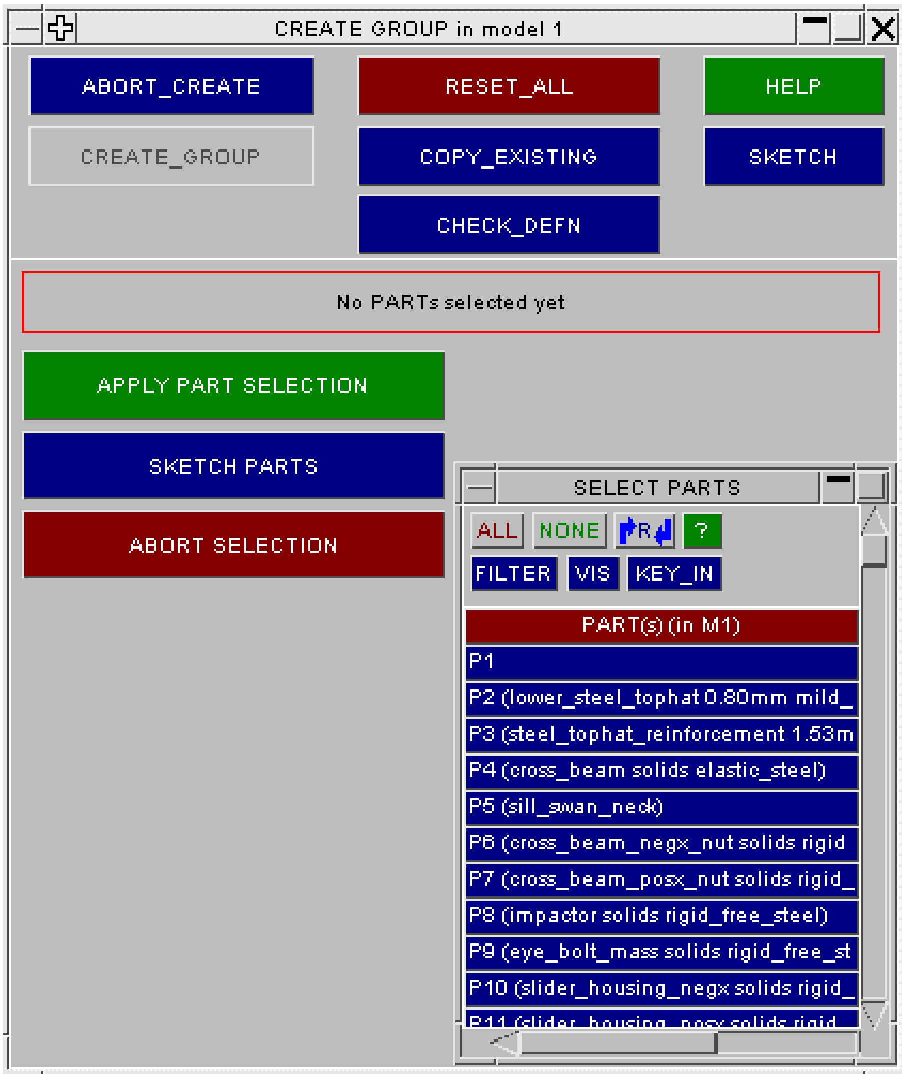
In the above example we are adding PARTs to the group. We are not using a box to limit the selection as **no box** is selected, If instead we wanted to add the elements from PARTs that are in box 1 this can be done by clicking on the **no box** button. The display will change as shown on the left.



Until a box is selected the **APPLY** button will be inactive. You can turn off the box selection again by pressing the **in box** button. Once a box is selected by either typing in the box number or using the popup menu it will turn blue and the APPLY button will be activated.

Selecting entities to add

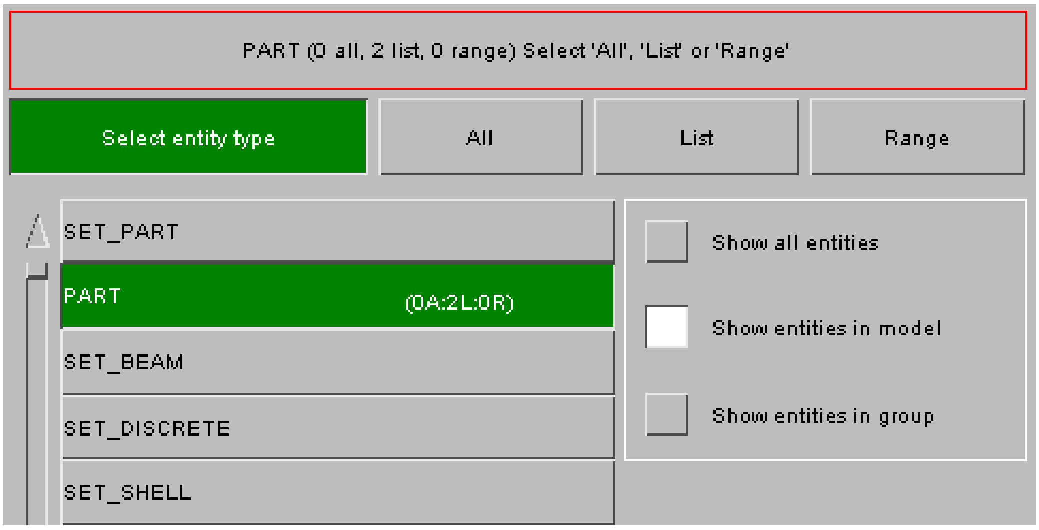
To start selecting the entities you want to add by list press the **Select PARTS to ADD** (eg if entity type is `PART`). The screen changes to:



The standard PRIMER object menus appear which allow you to select PARTs. Select the PARTs that you want to add by either selecting them from the list, picking visible parts etc. You can abort adding the entities at any time by pressing **ABORT SELECTION**. You can sketch the entities that you currently have selected to add by pressing the **SKETCH PARTS** button.

Saving into the group

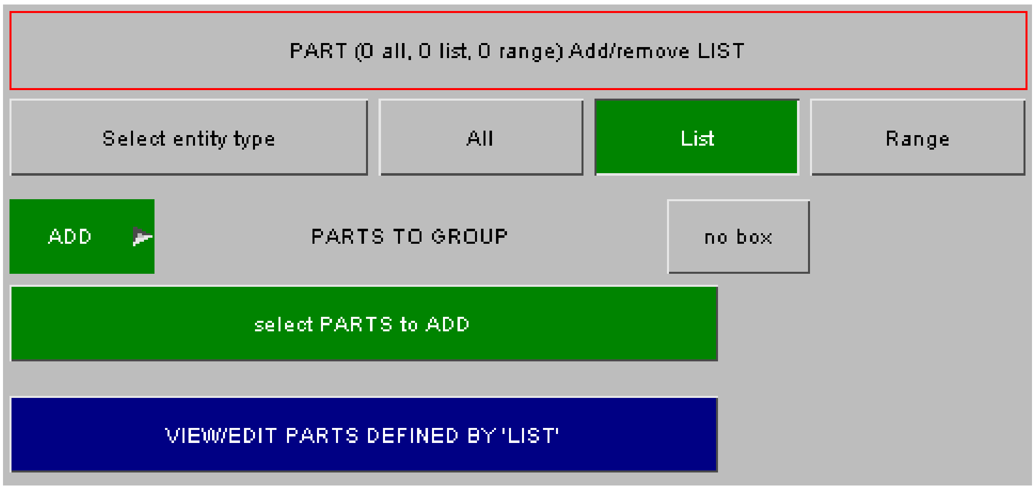
Once you have chosen the parts that you want to add using the object menus you can save them into the group by pressing the **APPLY PART SELECTION** button.



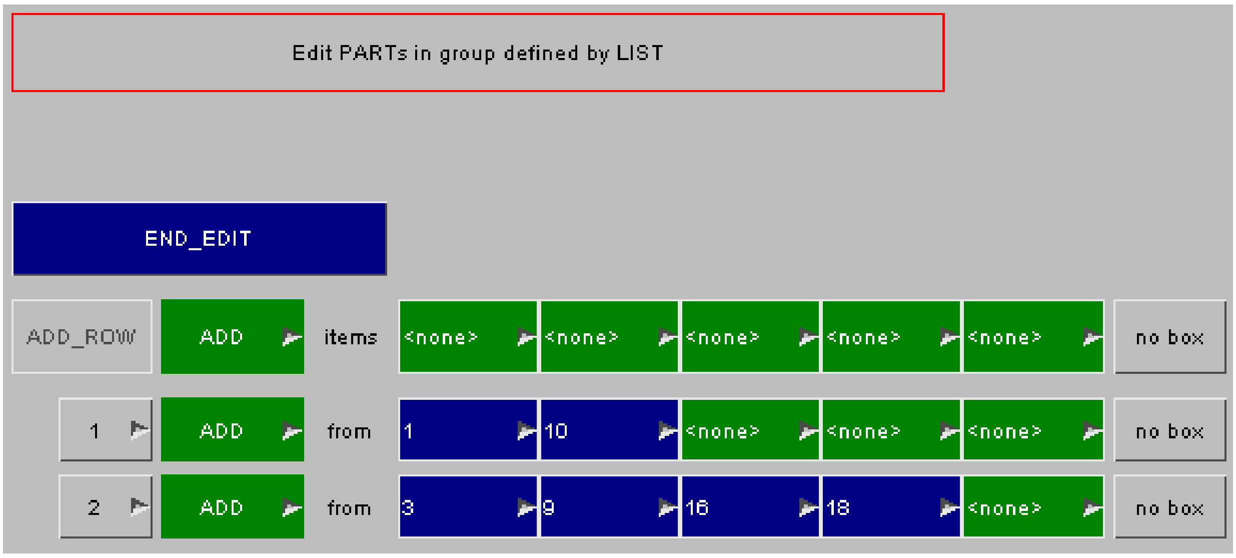
After the selection is saved to the group the screen refreshes back to the main screen (shown above). The feedback button (at the top of the image) changes to show that 2 selections of PARTs by list have been added in this example. Additionally this is shown on the **PART** entity button. In this way you can easily see what entity types are present in the group.

6.15.8 Editing/deleting entities by **List**

Press the **List** button.
The screen changes to:

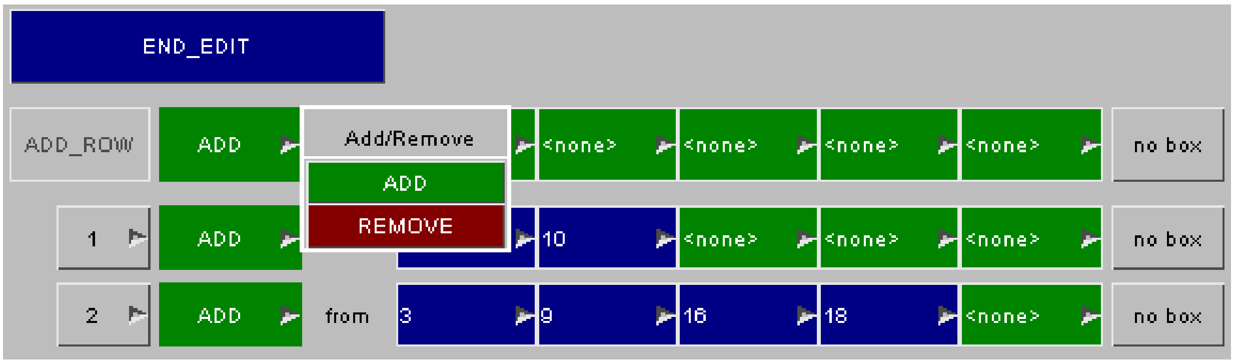


Press the **VIEW/EDIT PARTS DEFINED BY 'LIST'** button. The screen changes to the panel below. In this example there are 2 rows which have a total of 6 entries.

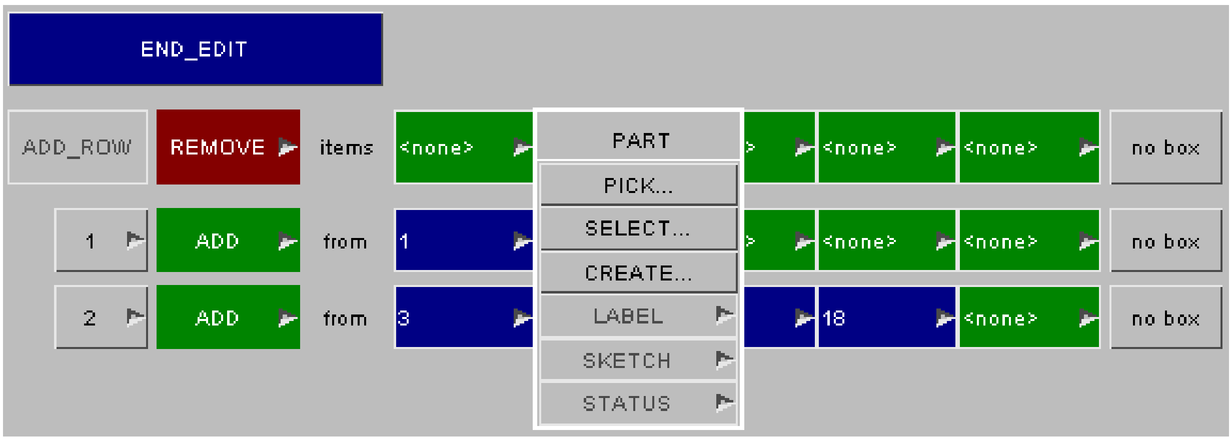


Adding a 'list' row using the group editor

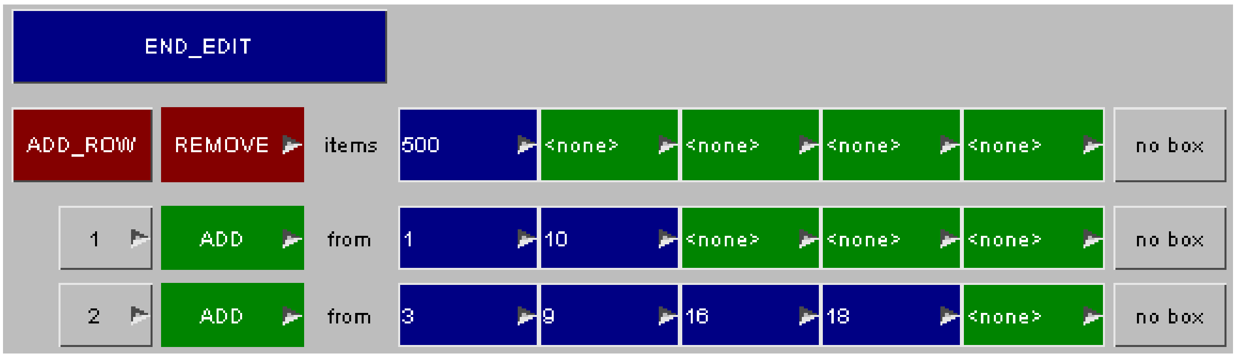
A new entry can be added to the group by using the **ADD_ROW** button. The **ADD/REMOVE** and **box** buttons work in the same way as the panel to add entities by list. As an example we will add the row to the group 'Remove **PART** 500'. At present the **ADD_ROW** button is inactive as there is nothing to add. Firstly, the Add/Remove popup must be used to change the **ADD** button to **REMOVE**.



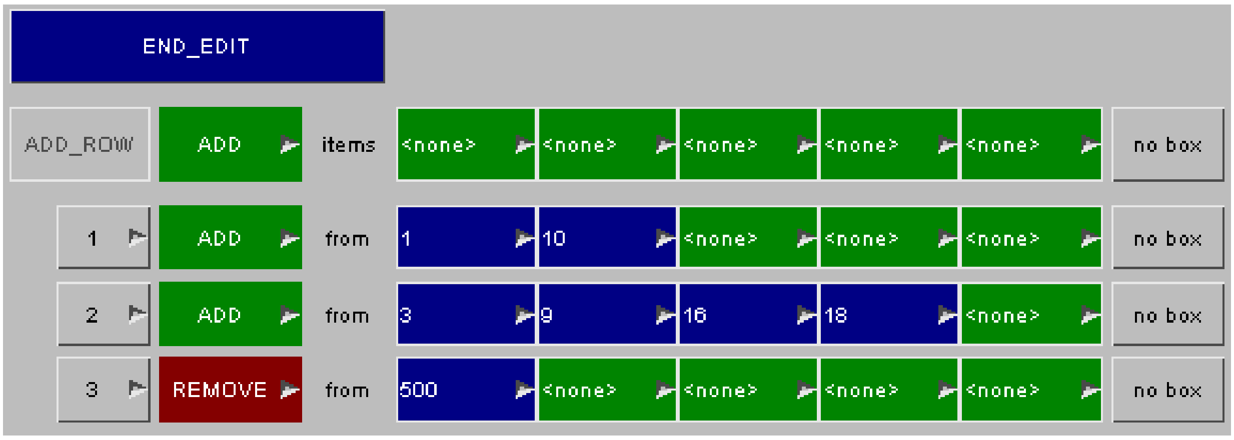
Next the part needs to be selected. You can either type the part number into the box or (as shown) use the popup menu to select the part from a list or pick it from the screen.



Now a part has been selected the **ADD_ROW** button becomes live.

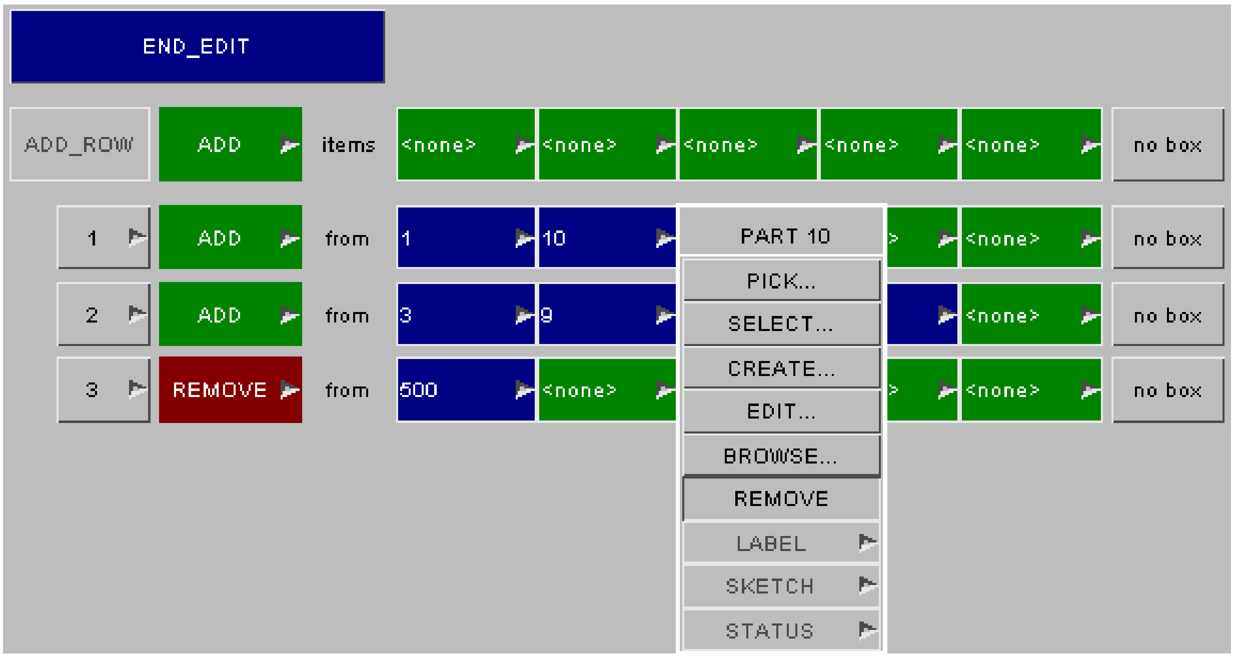


Now to add the row to the group simply press the **ADD_ROW** button. A new row '3' will be added to the list. The add row will be reset back to the default values.

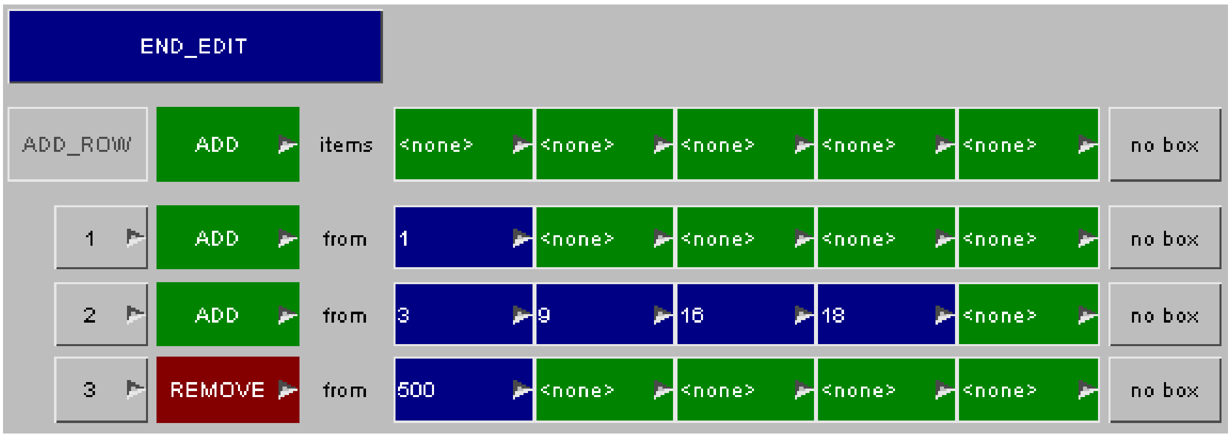


Editing a 'list' row using the group editor

Each of the existing rows can be modified if needed. For each row the **ADD/REMOVE** and **box** buttons work in the same way as the panel to add entities by list. An entity can be removed from a row by either deleting the number in the text box or by using the **REMOVE** option on the popup. For example to remove part 10, right click on 10 to bring up the popup menu and select **REMOVE**.

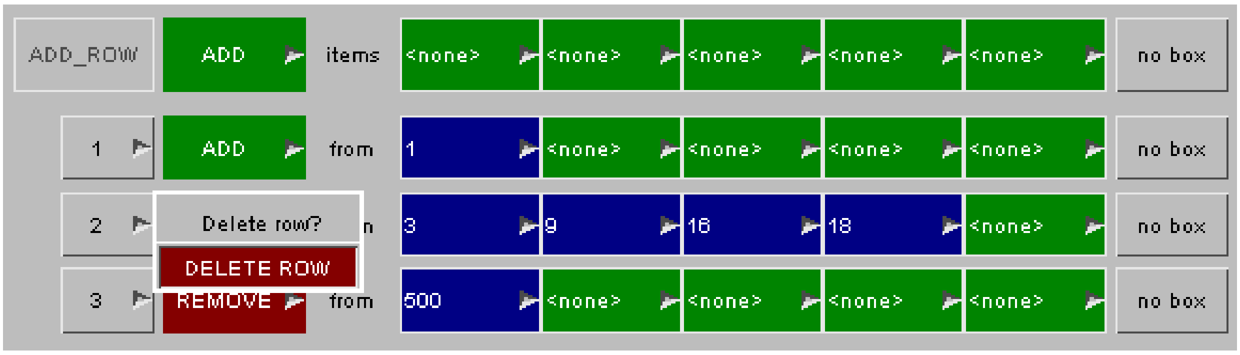


The part will be removed from the row (see image below). To save the change into the group press **END_EDIT**.

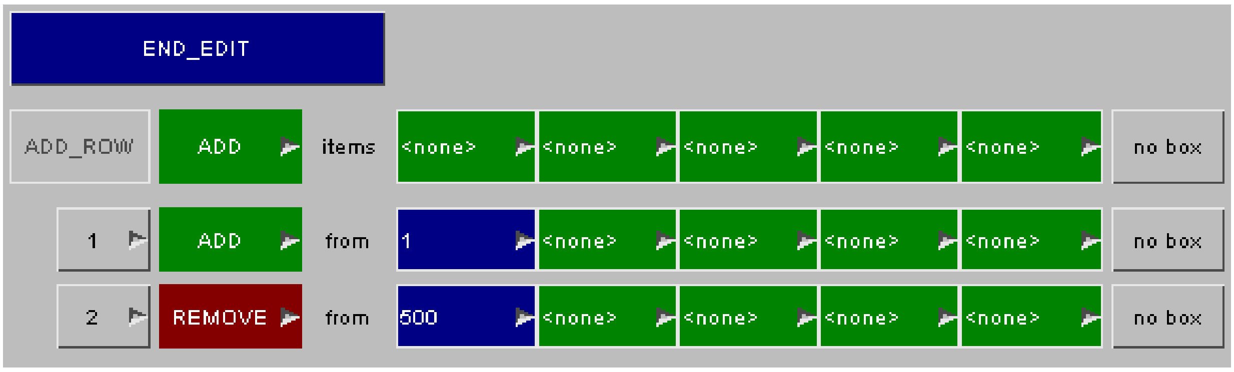


Deleting a 'list' row using the group editor

To delete an entry from a group use the **Delete row?** popup. For example, to remove entry/row 2right click on the **2** button and select **DELETE ROW** from the popup menu.

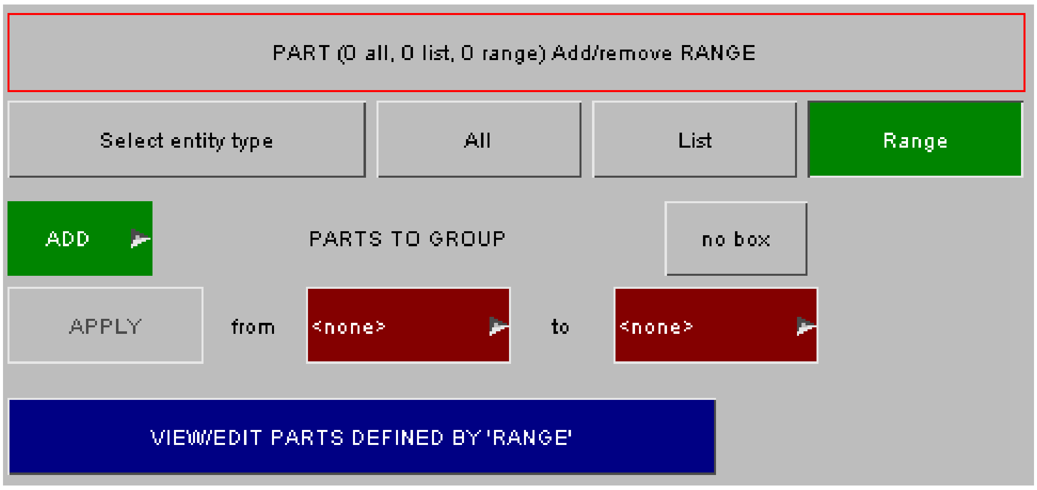


The row is deleted and the remaining entries are moved up the list.

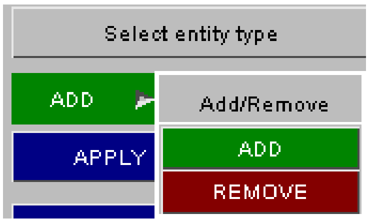


6.15.9 Adding entities by **Range**

Press the **Range** button. The screen changes to:

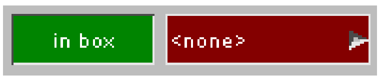


Selecting **ADD** or **REMOVE**

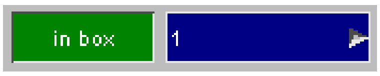


The green **ADD** button indicates that PARTs will be added to the group. If you want to remove PARTs instead of adding PARTs to the group, then the **Add/Remove** popup button can be used to change the action. If **REMOVE** is selected the button will change to red.

Using a box



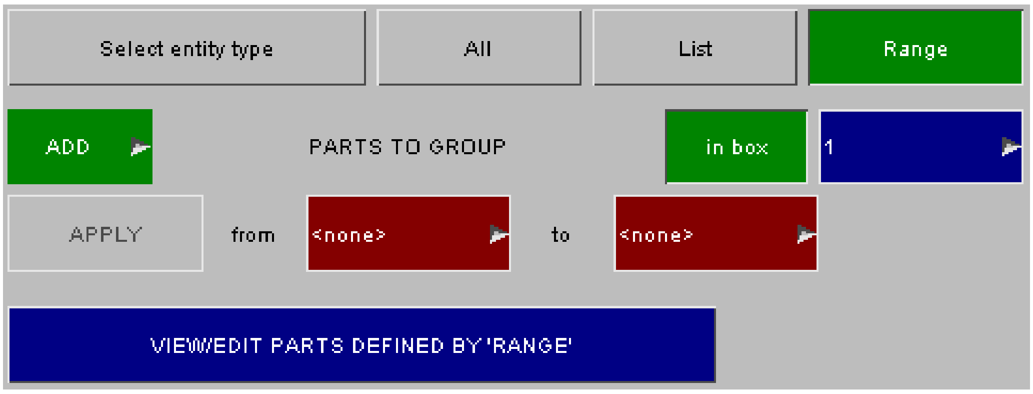
In the above example we are adding a range of PARTs to the group. We are not using a box to limit the selection as **no box** is selected, If instead we wanted to add the elements from a range of PARTs that are in box 1 this can be done by clicking on the **no box** button. The display will change as shown on the left.



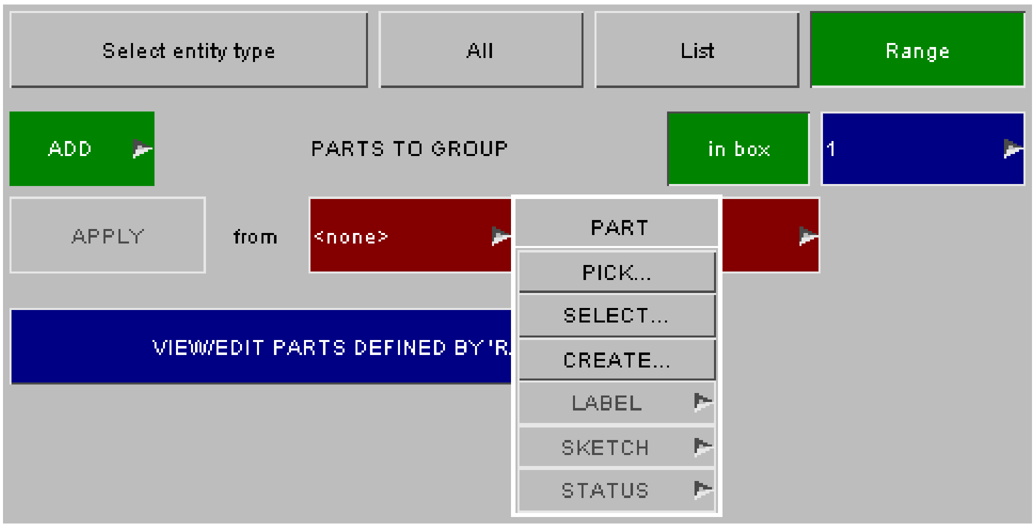
Until a box is selected the **APPLY** button will be inactive. You can turn off the box selection again by pressing the **in box** button. Once a box is selected by either typing in the box number or using the popup menu it will turn blue and the **APPLY** button will be activated.

Selecting entities to add

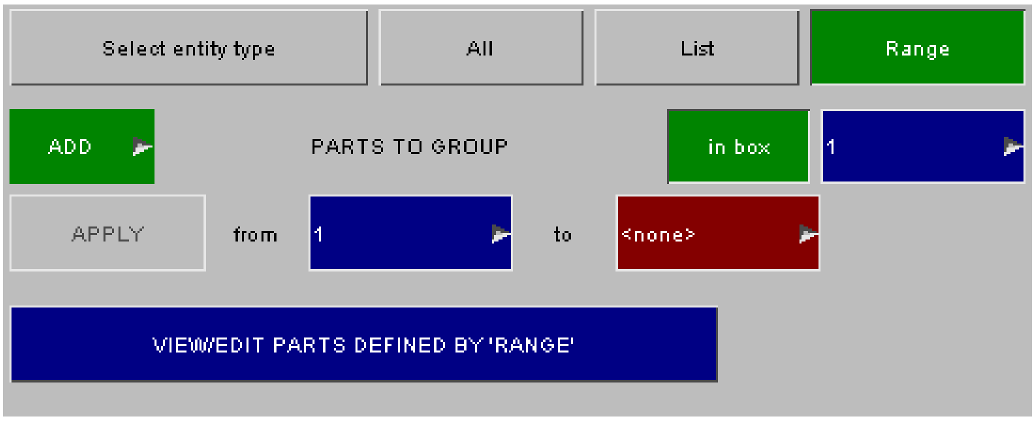
The following example shows how to add a range of entities to a group. We want to add 'Parts between 1 and 45 that are in box 1'.
First select the box as described above and choose **ADD or REMOVE** (in this example we want to add parts).



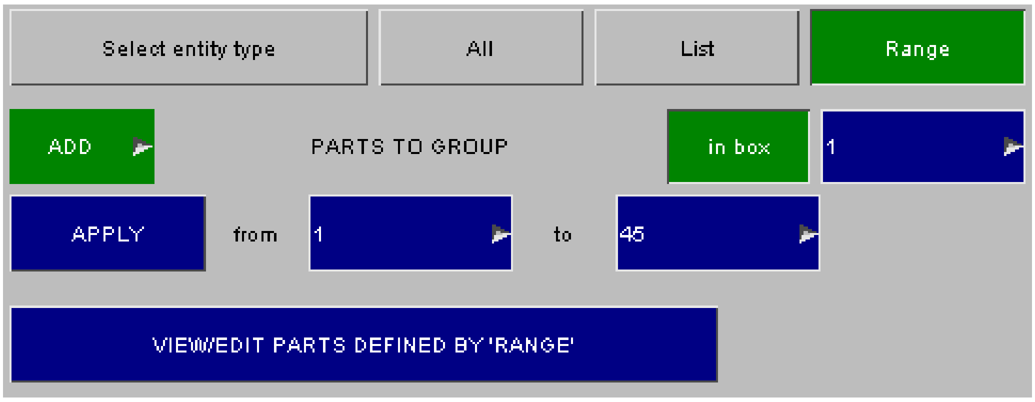
A start and end range must be given. You can either type the numbers into the boxes or use the standard PRIMER object menus to select the part or pick it from the screen. For example right clicking on the **From** field:



When the part is selected/picked (in this example part 1 is picked) the **from** field is filled in.

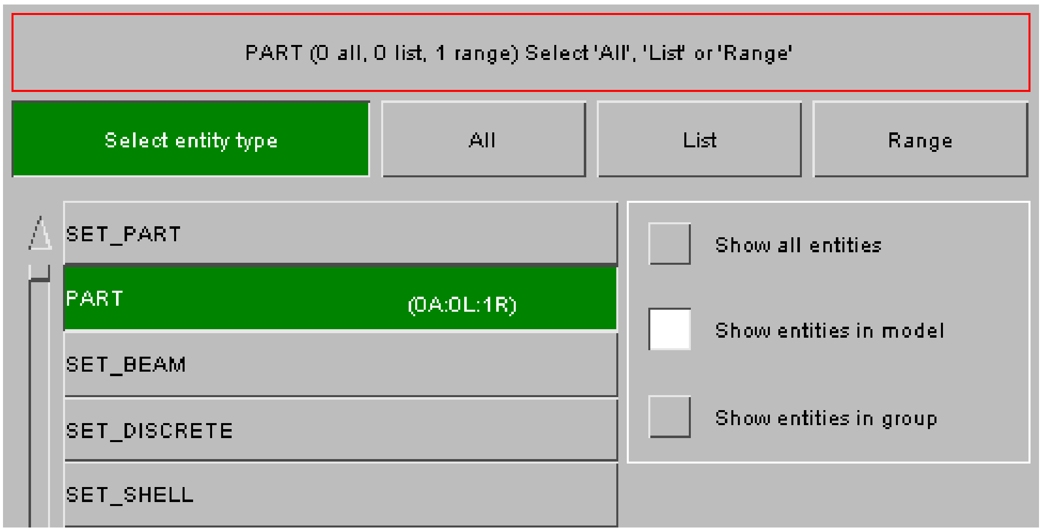


Similarly, the **to** field can be selected.



Saving into the group

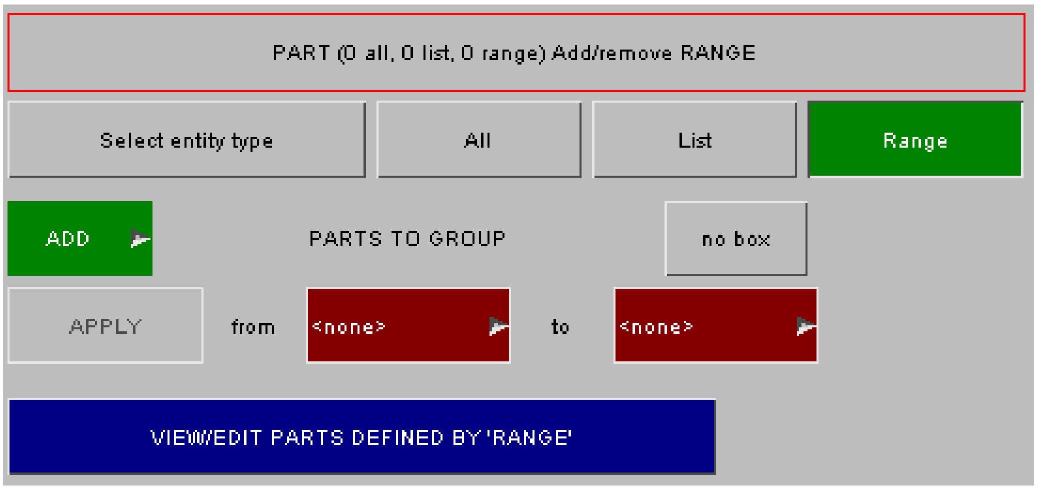
Once you have chosen the **to** and **from** fields for the parts that you want to add the **APPLY** button becomes active. You can save them into the group by pressing the **APPLY** button.



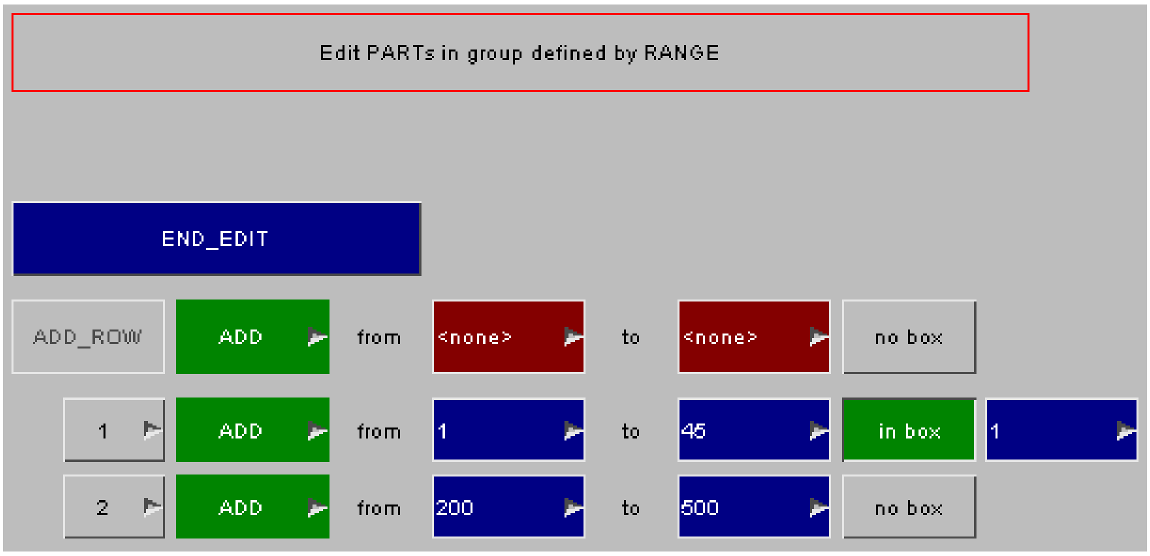
After the selection is saved to the group the screen refreshes back to the main screen (shown above). The feedback button (at the top of the image) changes to show that 1selection of PARTs by range has been added in this example. Additionally this is shown on the **PART** entity button. In this way you can easily see what entity types are present in the group.

6.15.10 Editing/deleting entities by **Range**

Press the **Range** button. The screen changes to:

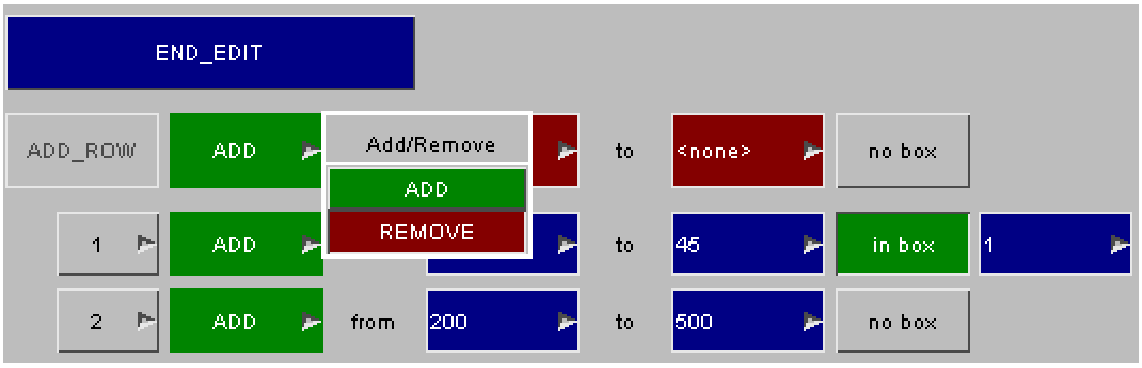


Press the **VIEW/EDIT PARTS DEFINED BY 'RANGE'** button. The screen changes to the panel below. In this example there are 2 rows. Row 1 adds parts 1 to 45 in box 1. Row 2 adds parts 200 to 500.

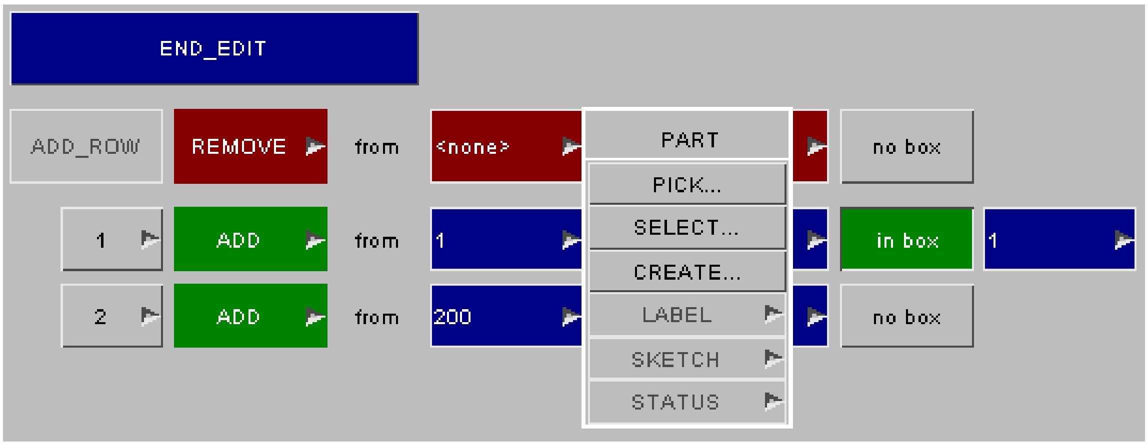


Adding a 'range' row using the group editor

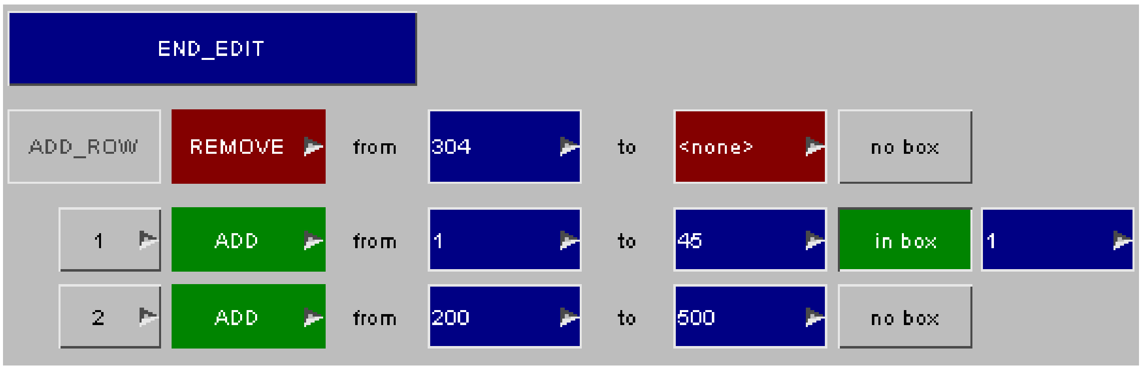
A new entry can be added to the group by using the **ADD_ROW** button. The **ADD/REMOVE** and **box** buttons work in the same way as the panel to add entities by list. As an example we will add the row to the group 'Remove **PARTs** 304 to 306 in box 2'. At present the **ADD_ROW** button is inactive as there is nothing to add. Firstly, the Add/Remove popup must be used to change the **ADD** button to **REMOVE**.



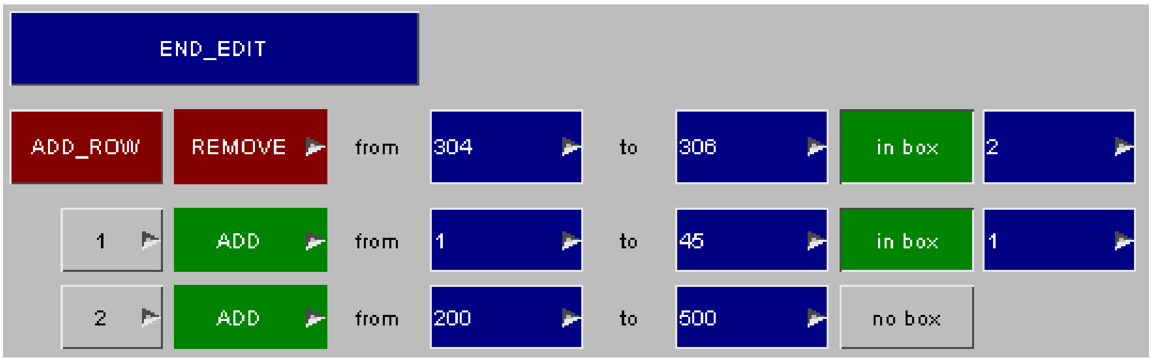
Next the to and from parts needs to be selected. You can either type the part number into the box or (as shown) use the popup menu to select the part from a list or pick it from the screen.



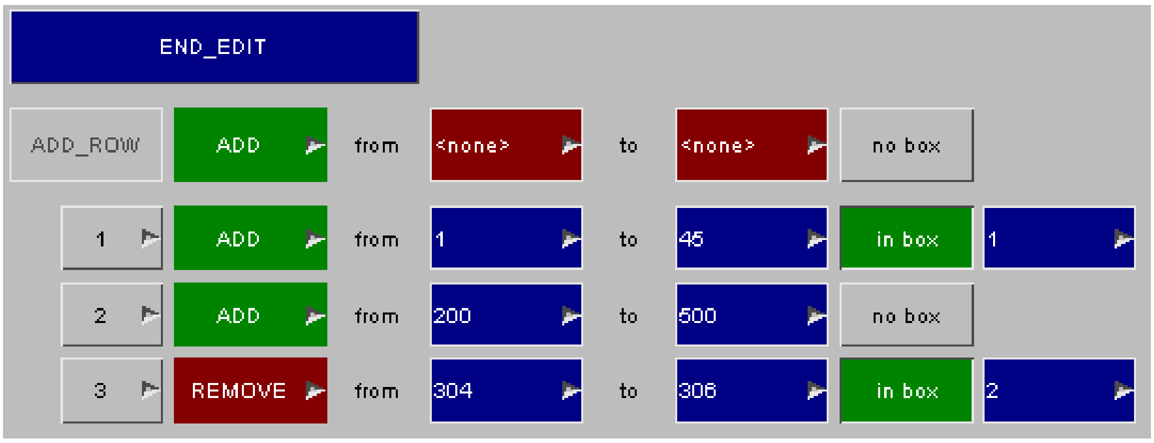
Once the part has been selected the **from** field will be filled in:



A similar process can be done to select the **to** and **box** fields. When the **to** and **from** fields are selected the **ADD_ROW** button becomes live.



Now to add the row to the group simply press the **ADD_ROW** button. A new row '3' will be added to the list. The add row will be reset back to the default values.



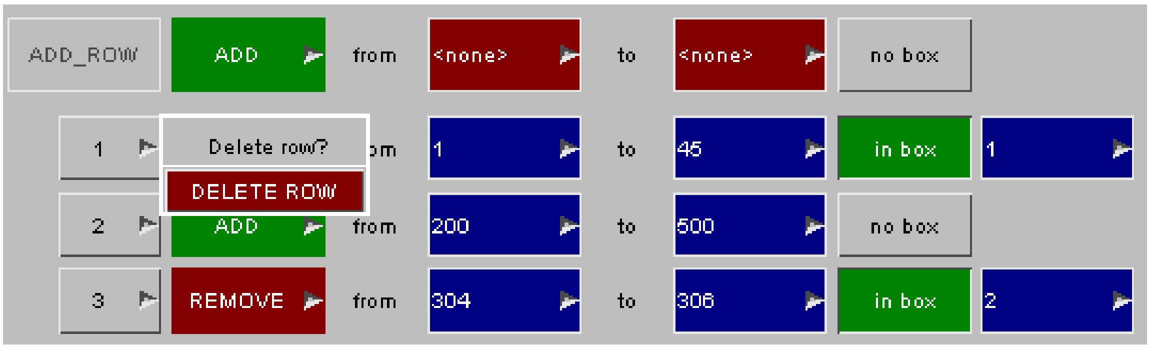
Editing a 'range' row using the group editor

Each of the existing rows can be modified if needed. For each row the **ADD**/**REMOVE** and **box** buttons work in the same way as the panel to add entities by list. The **from** and **to** entities can be modified by either changing the number in the text box or by using the popup menu.

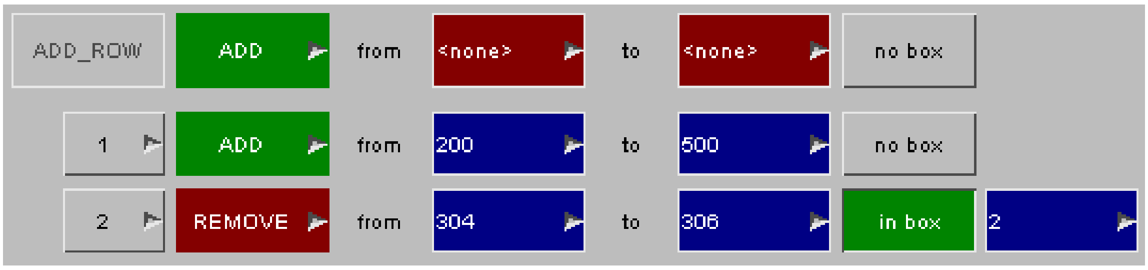
To save the change into the group press **END_EDIT**.

Deleting a 'range' row using the group editor

To delete an entry from a group use the **Delete row?** popup. For example, to remove entry/row 1right click on the **1** button and select **DELETE ROW** from the popup menu.

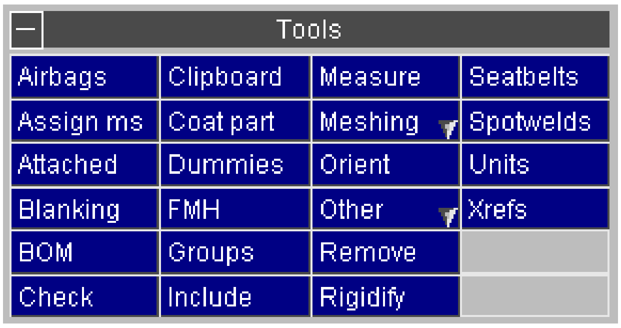


The row is deleted and the remaining entries are moved up the list.

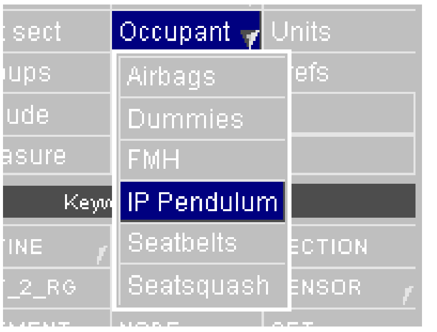


6.16 **INCLUDE** Controlling *INCLUDE files.

The **INCLUDE** menu in the **Tools** panel is covered in a separate section of the manual - see section 3.13 for details.



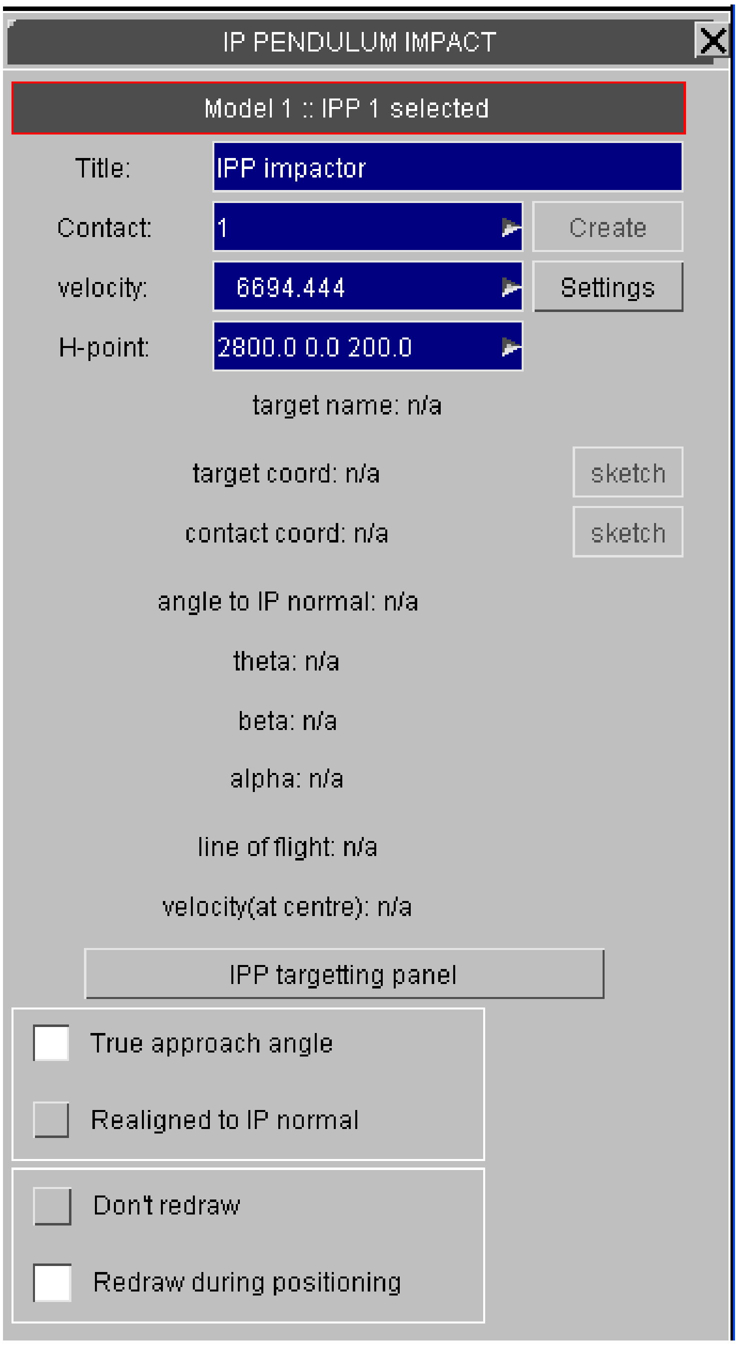
6.17 INSTRUMENT PANEL PENDULUM



The IP Pendulum function can be used to specify multiple Instrument Panel Pendulum impact models for ECER21. This function supports interactive and batch model processing. Automated positioning and depenetration is available.

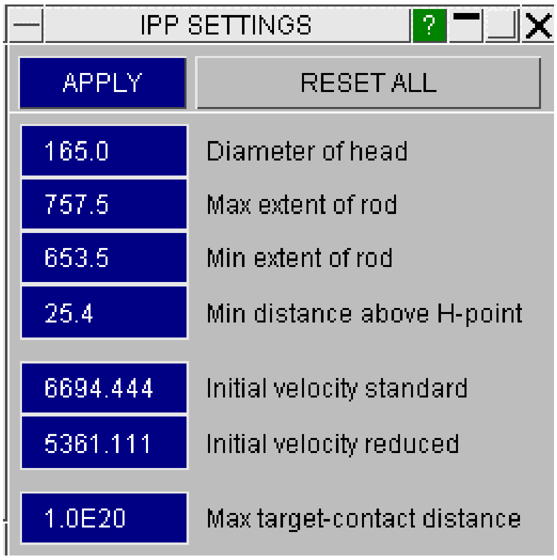
The initial screen for setting up IP Pendulum parameters is shown below. Post *END data, if available, is read in by default. Pendulum to IP contact can be created by clicking on the Create button.

Standard and reduced impact airbag velocities can be specified using the appropriate text box and popup. Base and forward H-point coordinates can also be specified in this panel. The **IPP targetting panel** can be reached by clicking on the appropriate button.

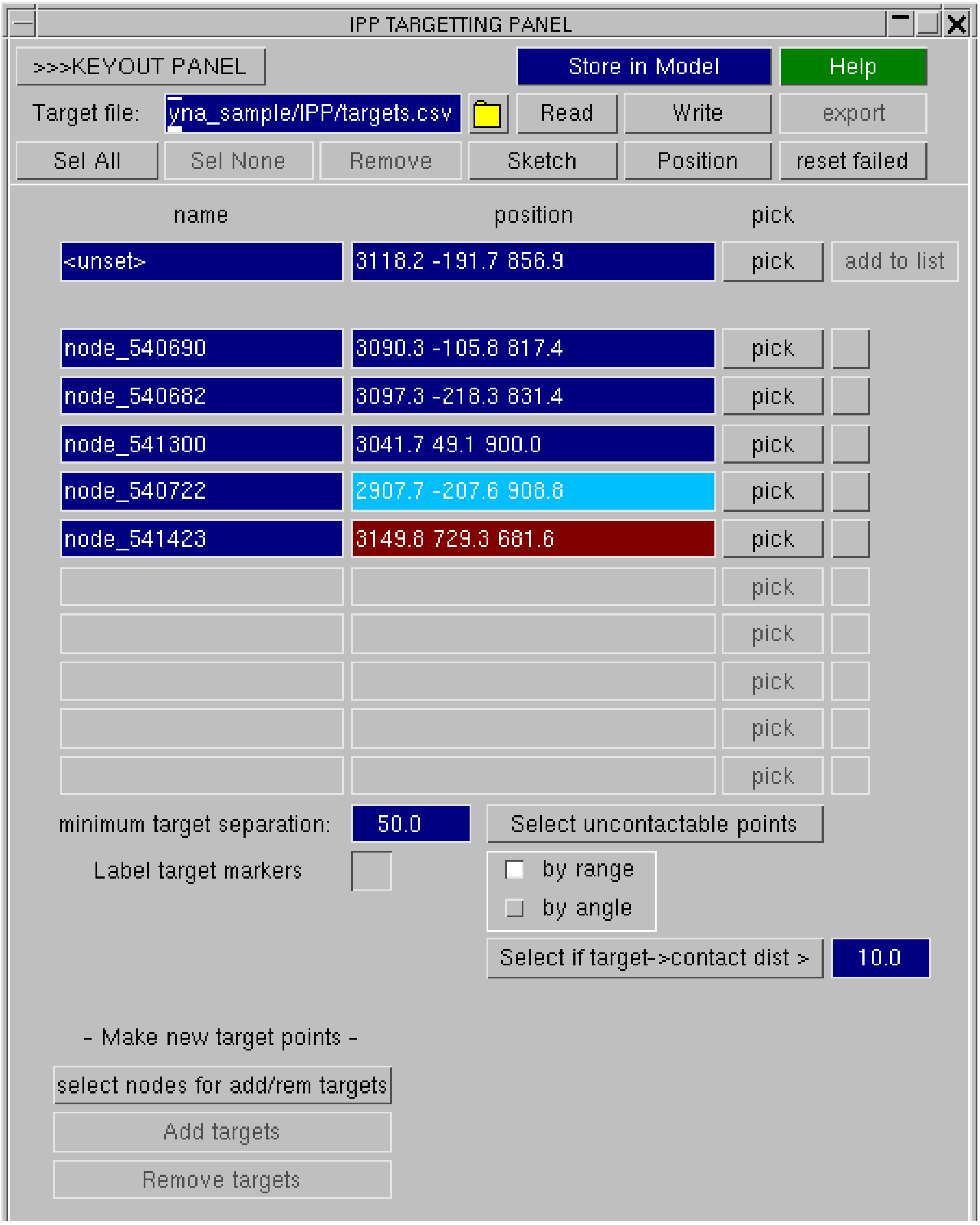


The setting panel enables the user to change the default settings which are defined as per regulation. These values are used by the positioner to determine whether or not a target point is legitimate. The head diameter should be consistent with the model used.

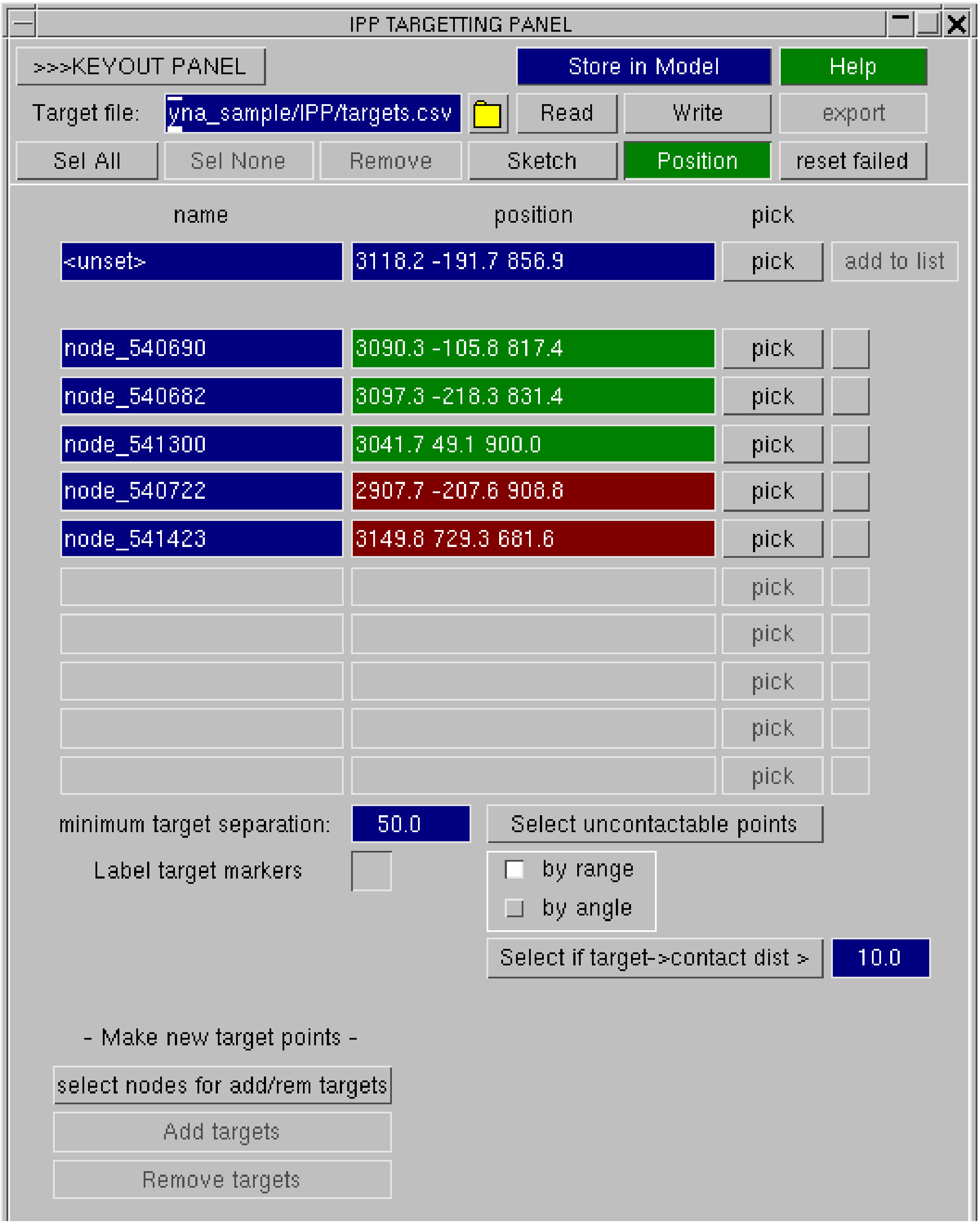
The user setting **Max target-contact distance** will exclude from the list of succesfully positioned points, any which fall outside this limit.



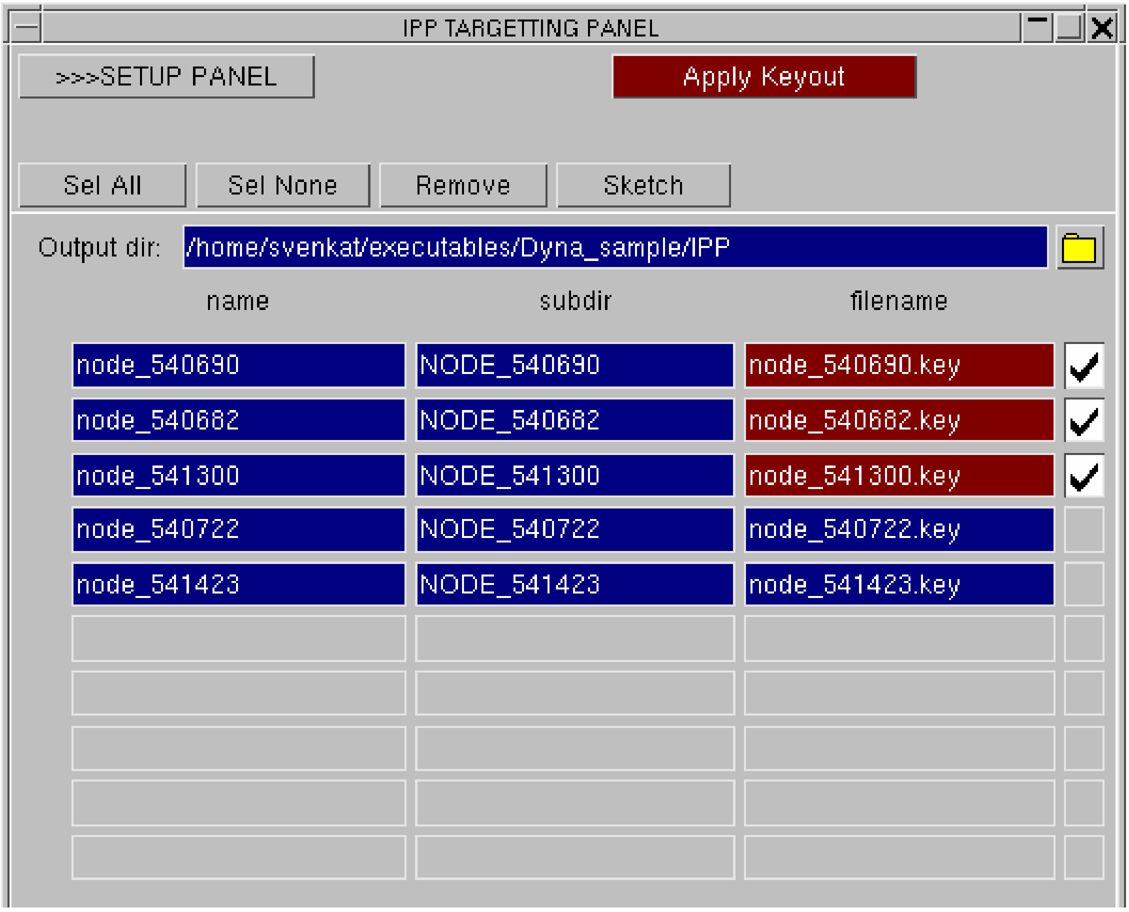
The IPP targetting panel, as shown below, displays existing targets in the model. A csv file can be read in using the **Read** button to load new points. New target points can be also added using the **sel nodes for add/rem targets** button. Prior to positioning, target points appear on a light blue background if they are misaligned with the trim normal. This indicates that they probably cannot be contacted by the pendulum. A red background indicates that they definitely cannot be reached by the pendulum. A dark blue background suggests that points are not yet positioned. When the **Position** button is clicked, selected targets are positioned and de-penetrated.



Target points that have been successfully positioned are shaded green. Failed points are shaded red.



The list of target points can be stored in a model or written to a csv targetting file for batch processing. This can be done by switching to the **KEYOUT PANEL**. File path, sub-directory names, and file names can be modified by users prior to keyout.



IPP parameters can be specified in the input csv file in the following manner when the IPP build is run interactively:

```
IPP
target_point_start, px, py, pz, nx, ny, nz, h_flag, r_flag, v_flag
node_540690, 3090.3, -105.8, 817.4, 0.0, 0.0, 0.0, 0, 0, 0
node_540682, 3097.3, -218.3, 831.4, 0.0, 0.0, 0.0, 0, 0, 0
node_541300, 3041.7, 49.1, 900.0, 0.0, 0.0, 0.0, 0, 0, 0
node_540722, 3157.7, -247.6, 908.8, 0.0, 0.0, 0.0, 0, 0, 0
```

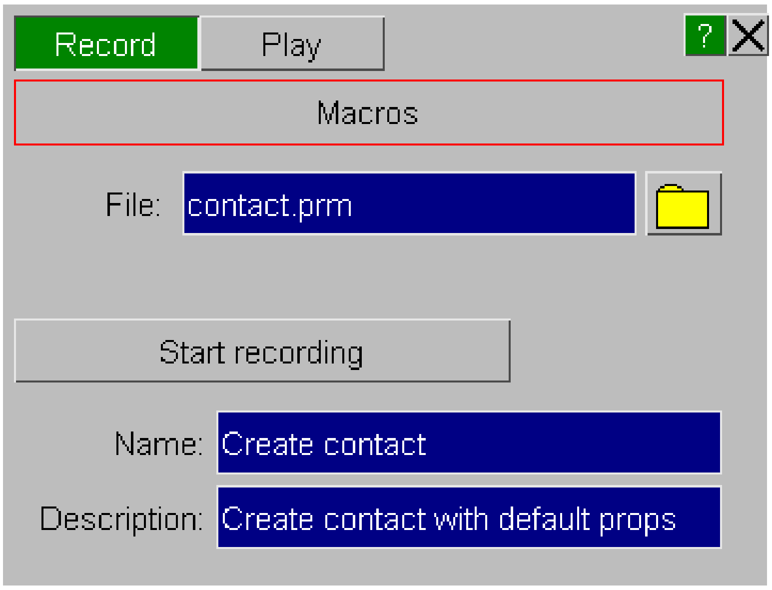
Additional information would be required if the IPP build is run in batch mode. In this case, the csv file can be specified as follows:

```
IPP
INCLUDE, C:\test\include.key
IMPACTOR, C:\test\pendulum.key
CONTACT, pendulum to trim
HPOINT,     3400.0, 394.0, 220.0
HPOINT_F,   3445.0, 394.0, 239.0
VELOCITY,   6694.1
VELOCITY_R, 5361.1
OUTPUTDIR, C:\test
target_point_start, px, py, pz, nx, ny, nz, h_flag, r_flag, v_flag
node_540690, 3090.3, -105.8, 817.4, 0.0, 0.0, 0.0, 0, 0, 0
node_540682, 3097.3, -218.3, 831.4, 0.0, 0.0, 0.0, 0, 0, 0
node_541300, 3041.7, 49.1, 900.0, 0.0, 0.0, 0.0, 0, 0, 0
node_540722, 3157.7, -247.6, 908.8, 0.0, 0.0, 0.0, 0, 0, 0
```

6.18 MACROS

The **Macro** panel allows you to record and playback a sequence of commands in PRIMER similar to using visual basic macros in Excel etc. They are deliberately made to be human readable so that a macro can be edited by hand. The easiest way to see the format of a macro file is to record one and see what commands that PRIMER uses.

6.18.1 Recording a macro



To record a macro press the **Record** button at the top of the panel. Give a filename for the macro in the textbox or press the folder icon to select a new file location. Macro files in PRIMER should have the extension prm (**PR**imer **M**acro).

Once a filename has been given the **Start recording** button will become active.

A name and a description for the macro can also be given for the macro. These will be saved in the macro as `MacroName()` and `MacroDescription()` commands and are then used as the name and hover text for a button in the macro panel.

Press **Start recording**. Any commands that you now do in PRIMER will be written to the macro file.

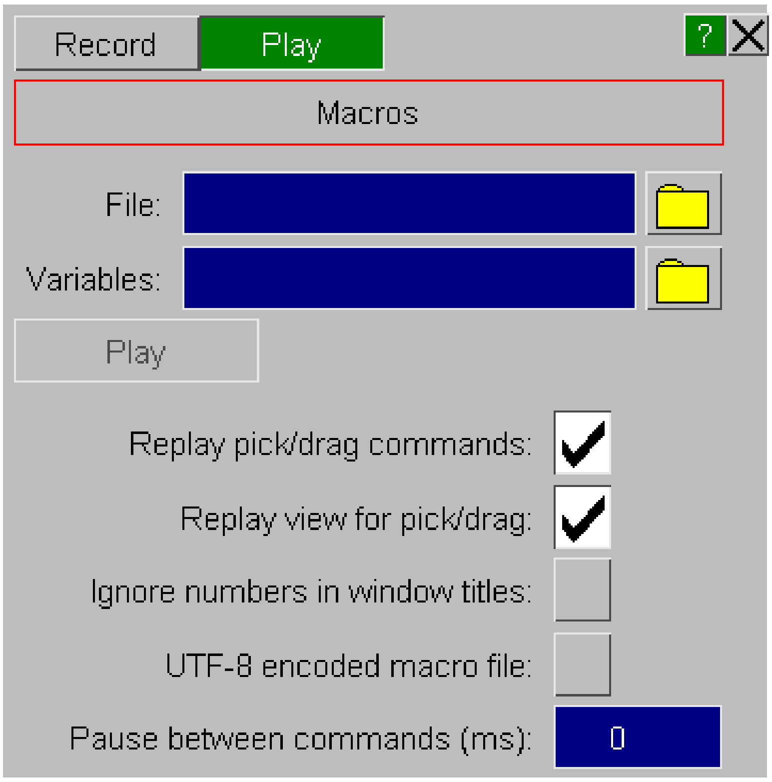
The button will turn red and read **Stop recording**.

When you have finished recording the commands you want press the **Stop recording** button. The file will be closed and PRIMER will stop recording commands.

6.18.2 Playing a macro

To play a macro press the **Play** button at the top of the panel. Give a filename for the macro in the **File:** textbox or press the folder icon to select a macro file (macro files in PRIMER should have the extension prm (**PR**imer **M**acro)).

Once a filename has been given the **Play** button will become active.



Variables

A macro can be edited so that instead of using fixed values, it can use variables for certain key values and/or numbers. The user can then change these variables as required. For an example consider the following simple macro example which selects some shells to orient and applies the translation 0.000 0.000 100.000.

```
MacroName("Orient")
MacroDescription("Test macro variables work in orient")
Window("Tools/Keywords").Button("Orient")
Window("Orient").Menu("ORIENT ITEMS").Select1("SHELL...")
Pause("Select shells to translate")
In Window("Orient")
    .Textbox("Translation distance") = "0.000 0.000 100.000"
    .Button("Apply")
End In
Window("Orient").Window("CONFIRM ORIENT").Button("Accept")
```

We want to change the translation to use variables so the user can change the values. We need to do 2 things.
1. Define the variables in the macro
2. Change the numbers in the translate to be variables.

To define the variables in the macro we have to add `MacroVariable()` commands at the top of the file. each command defines one variable. Variable names must only contain letters, numbers and the underscore character. They cannot contain spaces. To refer to a variable we use the variable name preceeded by a dollar. e.g. for variable X_TRANS we use $X_TRANS. Alternatively the syntax ${X_TRANS} can be used (i.e. dollar followed by variable name in curly brackets). Each MacroVariable() command defines a name for the variable (e.g. "X_TRANS") a description (e.g. "X translation distance") and the default value for the variable (e.g. "0.0").

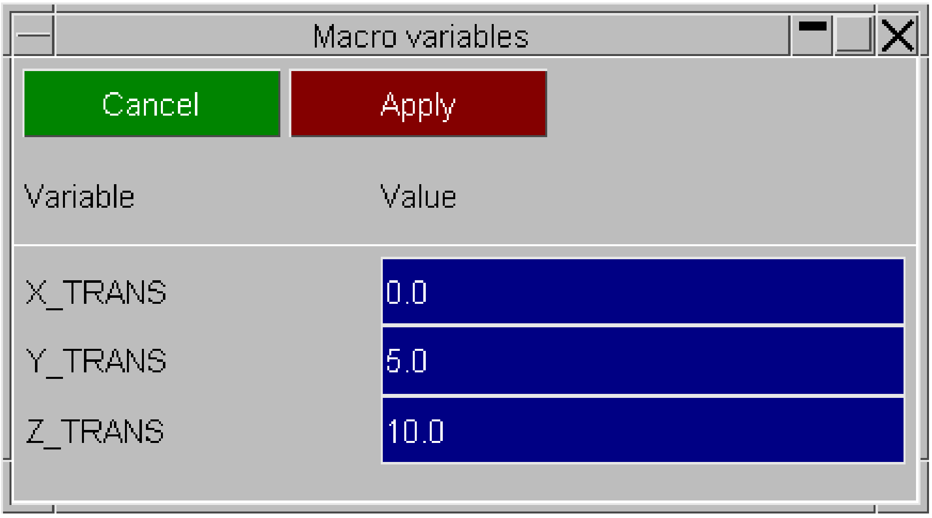
Adding these lines to the macro gives (added/changed lines shown in bold):

```
MacroName("Orient")
MacroDescription("Test macro variables work in orient")
MacroVariable("X_TRANS", "X translation distance", "0.0")
MacroVariable("Y_TRANS", "Y translation distance", "5.0")
MacroVariable("Z_TRANS", "Z translation distance", "10.0")
Window("Tools/Keywords").Button("Orient")
Window("Orient").Menu("ORIENT ITEMS").Select1("SHELL...")
Pause("Select shells to translate")
```

```
In Window("Orient")
   .Textbox("Translation distance") = "$X_TRANS  ${Y_TRANS}  $Z_TRANS"
   .Button("Apply")
End In
Window("Orient").Window("CONFIRM ORIENT").Button("Accept")
```

Using variables interactively

When you play a macro PRIMER scans the top of the macro to see if there are any `MacroVariable()` commands. If any are found then PRIMER shows a window with all of the variables. This allows you to change the variable values. Hovering over a variable name shows the description for each variable as hover text.



When the correct variable values are chosen the macro can be run by pressing **Apply**.

Using variables with CSV files

Instead of having to type the values for all variables interactively the values can be read from a CSV file. Give the name of the CSV file in the **Variables** textbox. The file should contain one variable and its value per line. Lines beginning with $ are treated as comments. e.g. the variables from the above example would look like.

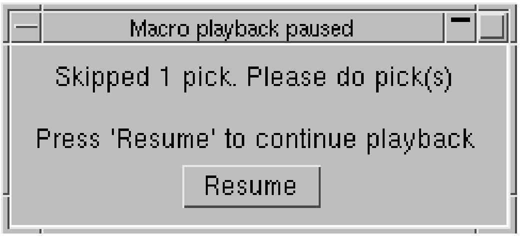
```
$ Example macro variables CSV file.
X_TRANS,0.0
Y_TRANS,5.0
Z_TRANS,10.0
```

Playback options

There are some options that alter the way that macros are played back

Replay pick/drag commands:

If a macro file contains any pick or drag commands then by default when the macro is played back the picks or drags will be played back exactly as they were recorded (i.e. the same position of the pick/drag on the screen will be replayed). If the **Replay pick/drag commands** option is selected this is what will happen.
If the option is unselected then the pick/drag command will be skipped and the macro playback will pause to allow you to replace the pick with whatever you want.A window will be mapped on the screen.



Once you have replaced the pick(s) or drag(s) then press **Resume** and the macro playback will restart.

Replay view for pick/drag

Whenever a pick or drag command is recorded PRIMER saves the current view in the graphics window to the macro with a `ViewMatrix` command. If this option is selected then on playback the view will be restored before picking. If it is not selected then the command will be skipped and the view will not be updated.

Ignore numbers in window titles

Ignore any numbers in window titles. See section on making macros work with different models for more details.

UTF-8 encoded macro file

Indicates that this macro contains Unicode text for Pause or MacroVariable commands and is UTF-8 encoded. If the macro contains a `MacroUTF8Encoded()` command this option will automatically be selected.

Pause between commands

Sometimes it is useful to have a pause between commands when playing the macro (e.g. if you are debugging a macro). This textbox allows you to give a pause (in milliseconds) between commands on playback.

Unicode

The text shown for `Pause()` commands and the descriptions in `MacroVariable()` commands can contain unicode text (e.g. Japanese or Chinese Kanji). If you want to use unicode then the macro file **MUST** be UTF-8 encoded and the **UTF-8 encoded macro file** playback option must be selected. If the `MacroUTF8Encoded()` command is added to the top of the macro then the option will automatically be selected.

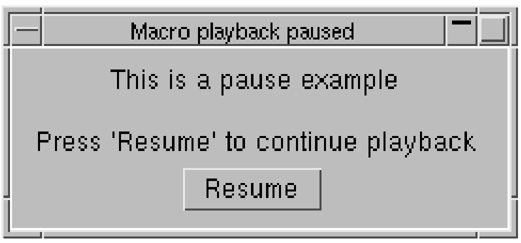
To show the unicode text the appropriate font must be used. This can be set using the preferences `primer*cjk_unix_font` and `primer*cjk_windows_font`.

Temporarily suspending macro files

It may be useful to pause playback of a macro so that the user can do certain operations. This can be done by manually adding a `Pause()` command to the macro. e.g. adding the command

```
Pause("This is a pause example")
```

will map the window shown on the right and then pause playback of the macro. The user can then do whatever operations are necessary and then press **Resume** to continue playback of the macro



This method could also be used to replace a sequence of picks in a macro. e.g. if a macro was recorded to create a contact and some parts were picked for the slave side of the contact there would be commands like
`In GraphicsWindow("GRAPHICS 1")`

```
   .ViewMatrix(1, 0, 0, 0, 1, 0, 0, 0, 1, 50, 72.5, 11.5, 34.8075, 316.838)
   .Pick1(1709, 1905)
   .ViewMatrix(1, 0, 0, 0, 1, 0, 0, 0, 1, 50, 72.5, 11.5, 34.8075, 316.838)
   .Pick1(2016, 1888)
   .ViewMatrix(1, 0, 0, 0, 1, 0, 0, 0, 1, 50, 72.5, 11.5, 34.8075, 316.838)
   .Pick1(2024, 1461)
   .ViewMatrix(1, 0, 0, 0, 1, 0, 0, 0, 1, 50, 72.5, 11.5, 34.8075, 316.838)
   .Pick1(1700, 1440)
   End In
```

recorded in the macro. These could all be deleted and replaced with a command

```
Pause("Select parts for slave side of contact")
```

which would prompt the user to pick the appropriate parts.

Making macros work with different models

When PRIMER writes a button press to a macro it uses the title of the window you clicked in and the text on the button to create a human readable command in the macro. e.g. in the following macro:

```
Window("Keywords").Button("PART")
Window("Part").Menu("SELECT PART").Select1("M1/P152    (MC-A-ARM-BUSH1-L)")
In Window("MODIFY PART M1/P152")
    .Textbox("SECID") = "10"
    .Textbox("MID") = "10"
End In
```

the user has:

1. Pressed button **PART** in the **Keywords** window *[which maps the part modify panel]*
2. In the menu **SELECT PART** in the **Part** window selected the entry **M1/P152 (MC-A-ARM-BUSH1-L)** *[which modifies part 152]*
3. In window **MODIFY PART M1/P152** changes the **SECID** and the **MID** values to 10.

This is fine if the macro is going to be replayed on exactly the same model but it will fail if the macro is played back on a similar (but not identical) model if the title of part 152 is different (or if the ID of the part with title "MC-A-ARM-BUSH1-L" is changed. This is because in step 2 the text on the menu button that is recorded includes the part ID and the part name.

When PRIMER replays a macro and it finds a window or menu title or a menu entry it first tries to find a unique match for the text in the macro command. If it finds one then that window is used. If that fails then it looks for a window or menu title or a menu entry which contains the text somewhere in the title. You can use this fact to make macros more portable across models.
For example if the above macro is always going to be used to edit part 152 you can modify the second line to

```
Window("Part").Menu("SELECT PART").Select1("M1/P152")
```

and the macro would still work if the title of the part is different. However you must be careful when doing this. If part 152 exists in the model then the above will work. If part 152 does not exist but part 152000 does exist then this would still match and so may edit the wrong part by mistake.

Alternatively, if the part number may change but the part title will always be the same you could make the macro more portable by modifying lines 2 and 3 to:

```
Window("Part").Menu("SELECT PART").Select1("MC-A-ARM-BUSH1-L")
In Window("MODIFY PART")
```

This will work as long as the part name is unique (and is the first match).

Occasionally it may be useful to play a macro in a completely different window to the one it was recorded in. For example you could record a macro which sets the necessary defaults for a contact that you modify. In this case if you were modifying contact 1000 in model 1 when you were recording the macro the macro could look like:

```
In Window("MODIFY CONTACT M1/CONT1000")
     .Textbox("sfs") = "0.1"
     .Textbox("sst") = "0.5"
     lots more commands
 End In
```

Now you may want to set the defaults for contact 1001. You could not use the above macro to modify contact 1001 as the title for this would be `"MODIFY CONTACT M1/CONT1000"`. If you ignored any numbers in the title then you could use the macro as the title for both windows would reduce to `"MODIFY CONTACT M/CONT"`. The **Ignore numbers in window titles** option does this.

Grouping commands in the same window together

To try to make macros easier to read and to make hand editing them easier PRIMER will group commands that are in the same window together using `In` and `End In` commands. e.g. the following macro:

```
Window("MODIFY PART M1/P152").Textbox("SECID") = "10"
Window("MODIFY PART M1/P152").Textbox("MID") = "10"
Window("MODIFY PART M1/P152").Textbox("EOSID") = "10"
Window("MODIFY PART M1/P152").Textbox("HGID") = "10"
```

is identical to

```
In Window("MODIFY PART M1/P152")
    .Textbox("SECID") = "10"
    .Textbox("MID") = "10"
    .Textbox("EOSID") = "10"
    .Textbox("HGID") = "10"
End In
```

but the second form is more concise and easier to read and edit (e.g. if you wanted to change the ID of the part being modified you would only have to do it on one line).

6.18.3 List of macro commands

Below is a full list of all of the commands available in macros. Comments can be put anywhere in a macro file. Any line that begins with $ or # is treated as a comment and will be skipped.

Commands in windows

A window is identified in PRIMER with the commands. PRIMER will try to find a window exactly matching *name*. If that fails a partial match will be tried. See the section on making macros work with different models for more details.

Command	**Description**
DialogWindow(*"name"*)	A window that is shown asking the user to confirm an action or answer a question.
GraphicsWindow(*"name"*)	The main graphics window
Menu(*"name"*)	An object menu
PopupWindow<*number*>()	A popup window. <number> identifies the popup. i.e. PopupWindow1 is a normal popup. PopupWindow2 is a popup mapped from a popup etc.
Window(*"name"*)	A 'normal' window

The following commands are used in conjuction with a 'window' command. Some commands have various alternatives. Where this is possible the alternatives are separated by `|` and are in square brackets `[]`. e.g. the command

```
.[|Ctrl|Shift]Click( "name" )
```

can be

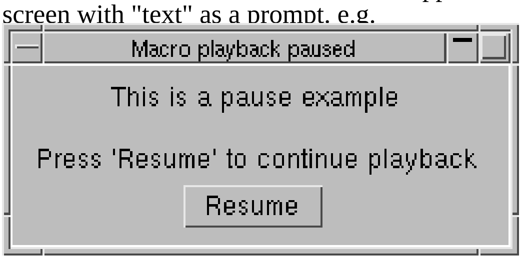
```
Click( "name" )
CtrlClick( "name" )
ShiftClick( "name" )
```

Command	**Description**
.Actions()	Equivalent to pressing the top left 'actions' button  for a window
.[Left|Right|Up|Down]Arrow()	Equivalent to using arrow keys to scroll a list

<table>
<tr><td>.Bitmap()</td><td>Equivalent to pressing the 'SAVE->BITMAP' button in the top left actions popup for a window</td></tr>
<tr><td>.[|Ctrl|Shift]Button("name")</td><td>Press, Ctrl Press or Shift Press on button name.</td></tr>
<tr><td>.[|Ctrl|Shift]Button("name") = [on|off]</td><td>Press, Ctrl Press or Shift Press on toggle button name. The state is set by using = on or = off.</td></tr>
<tr><td>.[|Ctrl|Shift]Click("name")</td><td>Click, Ctrl click or Shift click on item name in a table or tree.</td></tr>
<tr><td>.Collapse("name")</td><td>Collapse branch name in a tree.</td></tr>
<tr><td>.Dismiss()</td><td>Dismiss a window. Equivalent to pressing the top right button for a window</td></tr>
<tr><td>.[|End|Start]Drag<number>(<x>, <y>)</td><td>Drag at location x, y using mouse button number.</td></tr>
<tr><td>.End()</td><td>Equivalent to pressing End key to go to the bottom of a list</td></tr>
<tr><td>.Expand("name")</td><td>Expand branch name in a tree.</td></tr>
<tr><td>.Feedback("name") = "value"</td><td>Perform feedback function on button name with value (e.g. when typing in parameter, show the list of matching parameters)</td></tr>
<tr><td>.Help()</td><td>Show help for a window. Equivalent to pressing the top right button for a window</td></tr>
<tr><td>.Home()</td><td>Equivalent to pressing Home key to go to the top of a list</td></tr>
<tr><td>.Hover("name")</td><td>Perform hover function on button name (e.g. when hovering over a button with a parameter, the parameter data is shown)</td></tr>
<tr><td>.Lower()</td><td>Lower a window in the stacking order.</td></tr>
<tr><td>.Maximise()</td><td>Maximise a window. Equivalent to pressing the top right button for a window</td></tr>
<tr><td>.Minimise()</td><td>Minimise a window. Equivalent to pressing the top right button for a window</td></tr>
<tr><td>.Page[Down|Up]()</td><td>Equivalent to pressing PageUp or PageDown to move up/down a page in a list</td></tr>
<tr><td>.Pick<number>(<x>, <y>)</td><td>Pick at location x, y using mouse button number.</td></tr>
<tr><td>.Picking()</td><td>Restart picking in a window. Equivalent to pressing the top left button for a window</td></tr>
<tr><td>.Popup("name")</td><td>Map popup window for button name (equivalent to right clicking on button)</td></tr>
<tr><td>.Radio("name") = "value"</td><td>Set radio button name to value</td></tr>
<tr><td>.Resize(<data>)</td><td>Resize/move window</td></tr>
<tr><td>.Restore()</td><td>Restore a minimised window.</td></tr>
<tr><td>.Select<number>("name")</td><td>Select name from object menu at depth number. PRIMER will try to find a menu entry exactly matching name. If that fails a partial match will be tried. See the section on making macros work with different models for more details.</td></tr>
<tr><td>.Slider("name").Move[Up|Down|Right|Left]()</td><td>Move slider name up, down, right or left.</td></tr>
</table>

.Tab(*"name"*)	Press tab *name* in a window.
.Textbox(*"name"*) = *"value"*	Set textbox *name* to *value*
.ViewMatrix(*<view data>*)	Records the current view data to the command file so that when replaying the command file the view can be restored before doing a pick/drag

Other commands

Command	**Description**
$ *comment*	Any line beginning with $ is a comment
# *comment*	Any line beginning with # is a comment
Dialog(*"command"*)	Iss *command* in the dialogue window
Dynamic[Rotate|RotateZ|Scale|Translate](*<data>*)	Dynamic viewing command
End In	End a group of commands in the same window.
FunctionKey(<key>)	Equivalent of pressing function key
In *<window>*	Start a group of commands in the same window.
Justify("[Top|Bottom|Left|Right|Centre| TopRight|TopLeft|BottomRight|BottomLeft]")	By default Pause commands are shown at the top right of the PRIMER window. The Justify command changes the position that they are shown in. For example if you want them to be shown in the centre of the screen add the command Justify("Centre") to the macro **before** the Pause command.
MacroDescription(*"description"*)	Hover text that will be shown for the macro on the button in the macro panel. This must be in the first 10 lines of the macro.
MacroName(*"name"*)	Name that will be shown for the macro on the button in the macro panel. This must be in the first 10 lines of the macro.
MacroUTF8Encoded()	Indicates that the macro contains Unicode text for Pause or MacroVariable descriptions and is UTF-8 encoded. See the Unicode section for more details. This must be in the first 10 lines of the macro.
MacroVariable(*"name"*, *"description"*, *"value"*)	Adds a variable definition to the macro. See the Variables section for more details. These must be defined near the top of the file before the variable(s) are used.
Pause(*"text"*)	Temporarily suspend command file playback to allow user interaction. A window is mapped on the screen with "text" as a prompt. e.g.  Press Resume to resume playback.

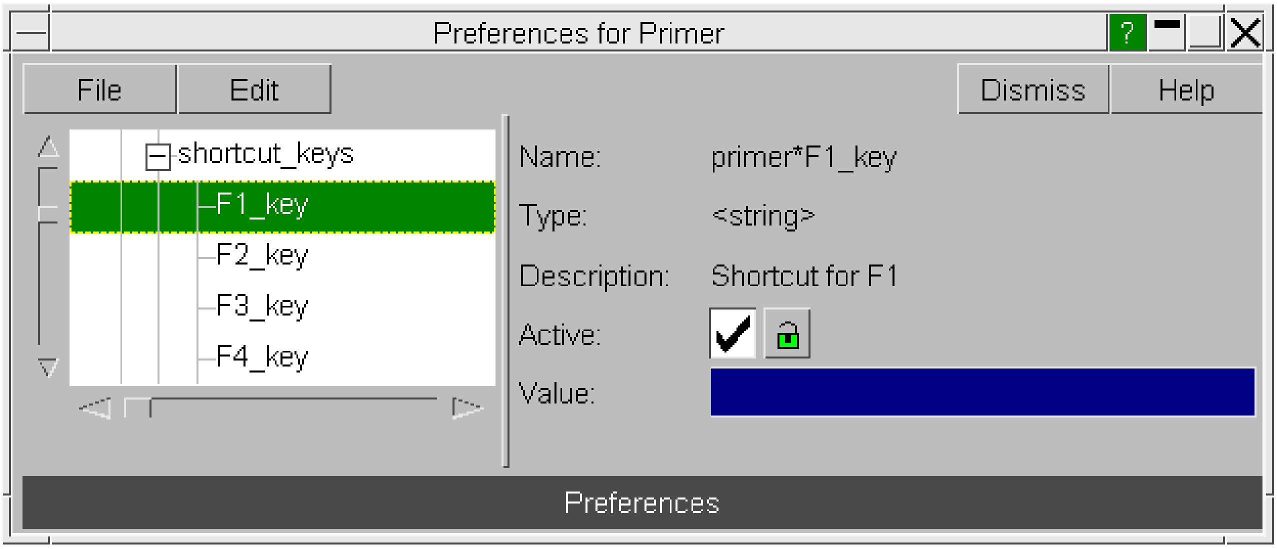
`Promptln( "message" )`	Write message to the dialogue box.
`Promptln( "message" )`	Write message to the dialogue box, adding a new line
`SelectFile( "filter" ) = "name"`	Select file *name* from the File selection window.
`ShortcutKey( <key> )`	Equivalent of doing shortcut *key*
`Version( "version", <build> )`	Records the version and build of PRIMER into the macro. this may be used in the future to help playback of macros recorded in earlier versions of PRIMER.
`WindowSize( <x>, <y> )`	Records the size of the graphics window to the macro so that when replaying the window can be resized to the correct proportions (if different) so that picks and/or drags work correctly.

6.18.4 Assigning macros to shortcut keys

Macros can be assigned to shortcut keys to make them quick and easy to run. To do this use the **Assign Macros to shortcut key** button (or press the shortcut key '**?**')

Using the above button will only assign the macro to the shortcut key for this session of PRIMER. If you want to always assign the macro to a shortcut key use the **`primer*F1_key`** preferences etc in the **`shortcut_keys`** branch for PRIMER in the oa_pref preference file. Note that the preference can be used to run either a shortcut, a script or a macro. For PRIMER to know that the preference should run a macro, the macro **MUST** have the extension **prm**.

The image below shows the preferences editor with the **`primer*F1_key`** open.



6.18.5 Assigning macros to buttons

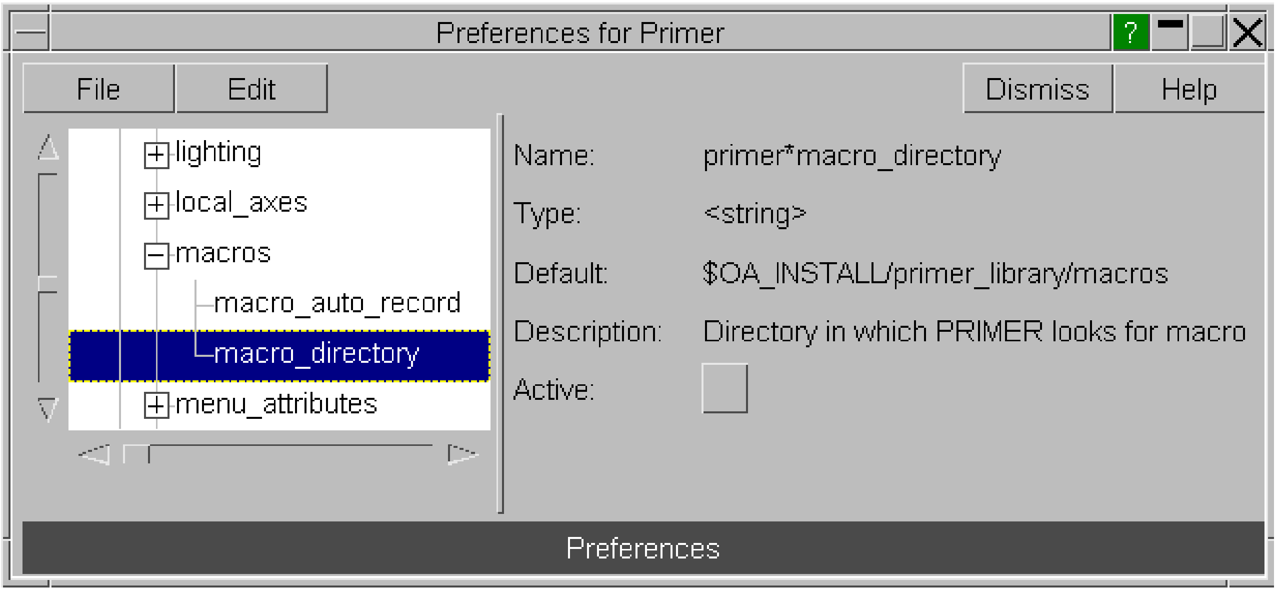
When PRIMER starts it automatically looks for macros in the directories:

- $OA_ADMIN/primer_library/macros (if $OA_ADMIN is defined)
- $OA_INSTALL/primer_library/macros
- $OA_HOME/primer_library/macros

Each macro that is found is assigned to a button in the macros panel. The text that is shown on the button is read from the `MacroName()` command at the top of the macro. This is automatically added by Primer when you record a macro if you enter some text in the **Name** textbox. Additionally hover text for the button is read from the `MacroDescription()` command at the top of the macro. This is automatically added by Primer when you record a macro if you enter some text in the **Description** textbox.

The directory that PRIMER looks in for macro files can be changed in the oa_pref files in $OA_ADMIN,

$OA_INSTALL and $OA_HOME by using the `macro_directory` preference.



For example if you change the **`macro_directory`** preference in the oa_pref file in the $OA_INSTALL directory to /test/primer_macros then PRIMER will look for macro files in the directories:

- $OA_ADMIN/primer_library/macros (if $OA_ADMIN is defined)
- /test/primer_macros
- $OA_HOME/primer_library/macros

6.18.6 Limitations

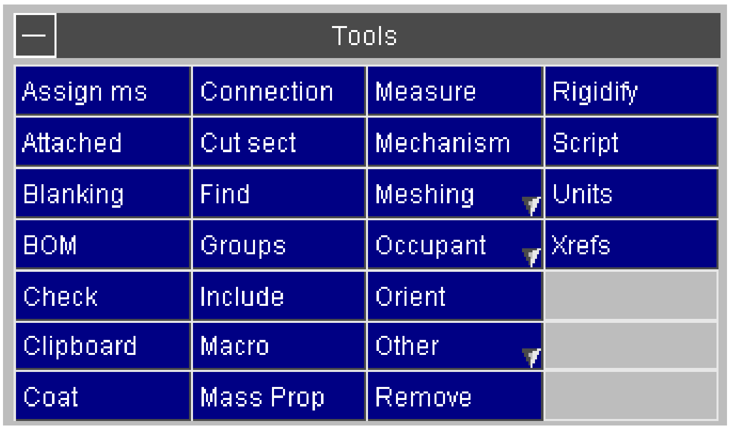
The biggest limitation with macros is that windows are identified by their title. This is fine if the titles are unique but causes problems if there are 2 windows with the same title. e.g. If you record a macro that modifies 2 different parts the windows will have different titles and the macro will be replay correctly. However, if you create 2 parts at the same time then both windows will have the same title and so on playback PRIMER will not be able to identify which of the 2 parts the button presses etc should be replayed in and the playback will fail.

In reality this is not a big limitation as the above situation is rare. You just need to be careful to only have one creation window for a particular type open at one time.

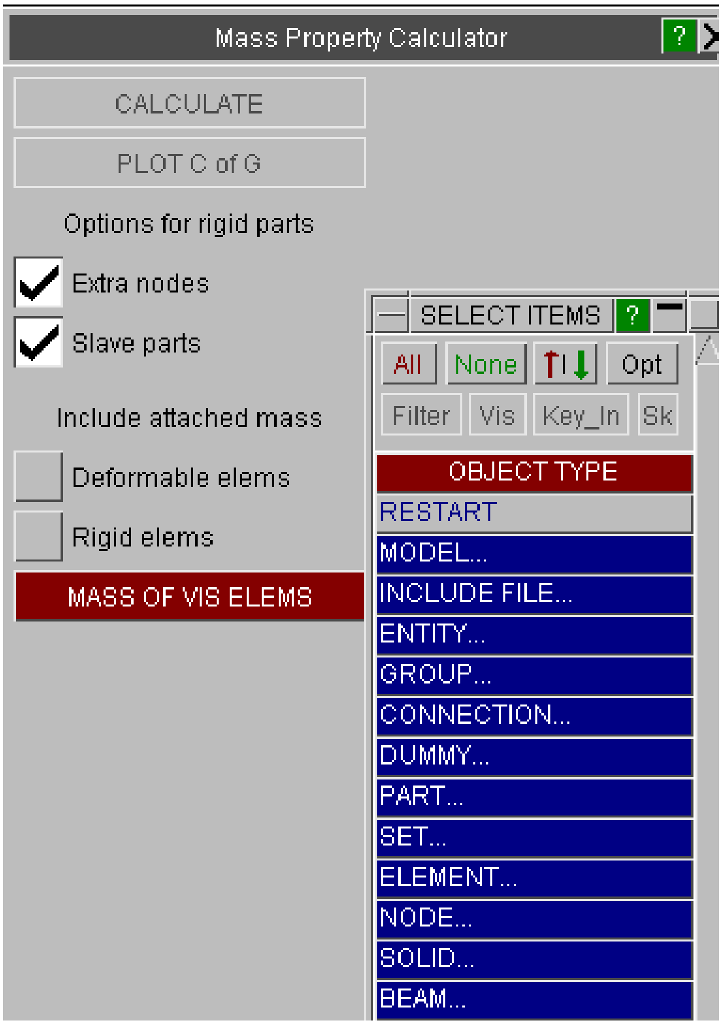
Dragging items in the part tree is not supported by macros. As a workaround you should right click on the selected parts and choose ’Cut’ and then right click on the destination and choose ’Paste’. These actions will be recorded correctly in the macro.

6.19 MASS PROPERTY CALCULATOR

The **Mass Prop** tool allows you to calculate the mass properties of a selection of items from the model. It is accessed from the main Tools menu

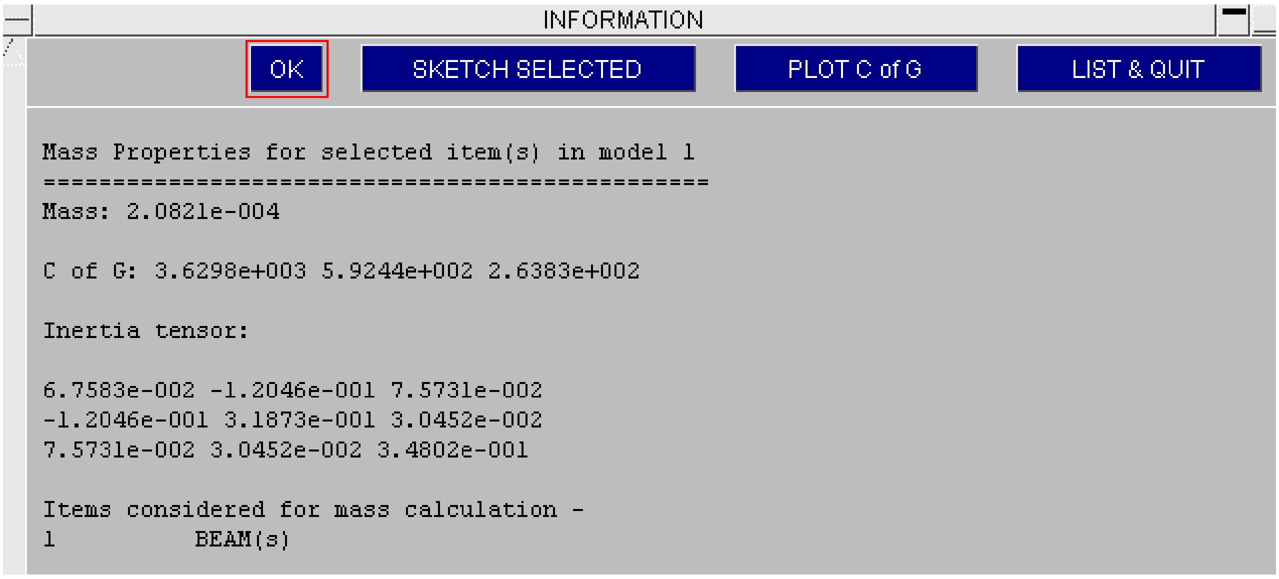


6.19.1 Selecting an item



Select the chosen item from the object menu and press **CALCULATE.** The mass properties will be reported in an information panel.

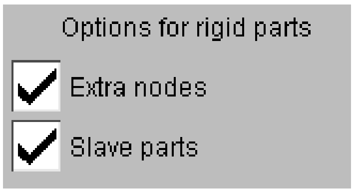
6.19.2 Calculating properties



In most cases, Primer will devolve the selection to the element level and sum the nodal masses derived from these elements as appropriate.

For a rigid element this will include the mass share of any deformable element attached.

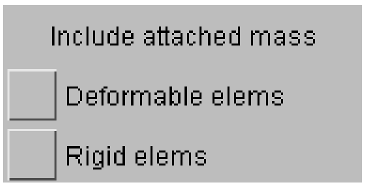
If a rigid part is selected, by default the mass of *Constrained_extra_nodes and elements of any parts slaved to this one by *Constrained_rigid_bodies will be included in the calculation. These options can be switched off.



For deformable elements mass at nodes attached to rigid parts/nrbs will not be subtracted. Consequently for a part you may get a slightly higher mass from this function than the part table gives.

If you make a selection which does not devolve to elements (such as a constrained joint) Primer will sum the masses of the nodes involved (devolved from elements that may not be selected) and report that.

Options may be set to include lumped mass attached to the nodes of selected elements. These are off by default.

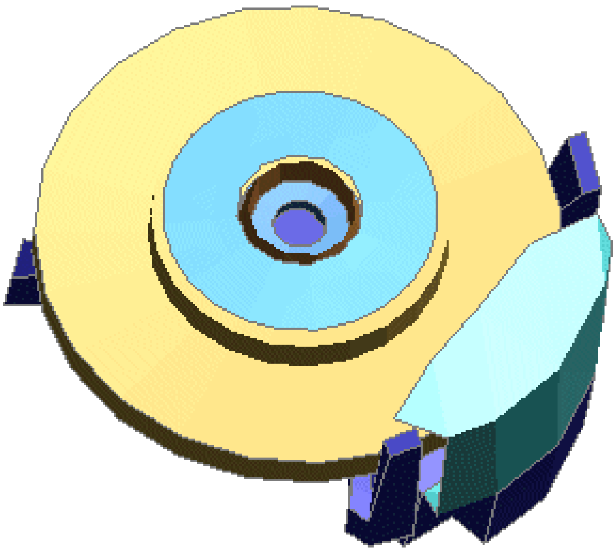


For a deformable part the value of lumped mass is shared equally amongst all the parts that attach to the node, so the calculation will only include that share that applies to selected elements.

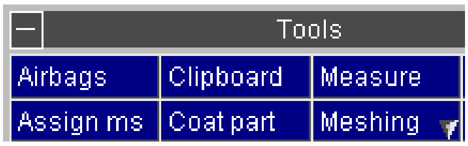
6.19.2 Mass of what is visible

MASS OF VIS ELEMS is useful to determine mass properties for a set of displayed components when blanking has been applied.

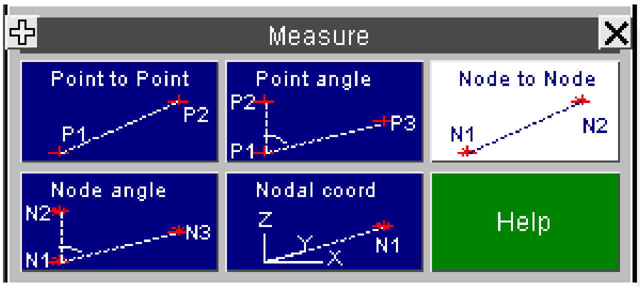
```
OK    SKETCH SELECTED    PLOT

Mass Properties for visible elements in model 1
================================================
Mass: 1.5865e-003

C of G: 3.7306e+003 -7.0049e+002 3.1578e+002

Inertia tensor:

1.0355e+001 -2.7300e+000 -9.0069e-002
-2.7300e+000 1.0015e+001 -5.7336e-001
-9.0069e-002 -5.7336e-001 9.3889e+000

Items considered for mass calculation -
222        SOLID(s) (sum of solid volumes = 1.76255e+006)
194        SHELL(s) (sum of shell areas = 36317.3)
7          PART(s)
6          CONSTRAINED_RIGID_BODIES(s)
1          CONSTRAINED_EXTRA_NODES(s)
```



6.20 MEASURE Measuring the distance and angles between nodes and points on the screen.



The **MEASURE** command is invoked from the **Tools** panel at the top of the screen or from the shortcut key M. There are five options, 3 using nodes and 2 using screen points, which measure:

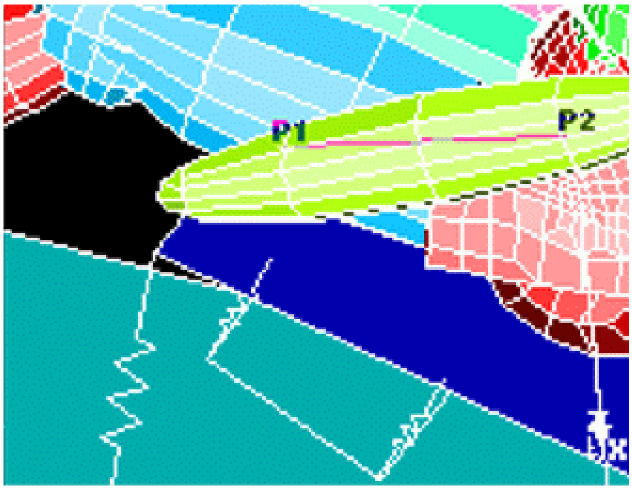


Point to Point	The (x,y) distance between two screen points P1 and P2.
Point angle	The angle between vectors P1P2 and P1P3
Node to Node	The (x,y,z) distance between two nodes N1 and N2.
Node angle	The angle between vectors N1N2 and N1N3.
Nodal coord	The coordinate of node N1.

Screen "points" are simply transient 2d locations on the display picked with the cursor. They do not have any structural significance and are not part of any model. Distances and angles computed from them are in the 2d screen (x,y) space system.
In this example the user has selected two points, labelled `P1` and `P2` on the screen, and the 2D projected vector between them is to be computed.

The reported distance, and its orientation with respect to the model, will be a function of the current transformation matrix.

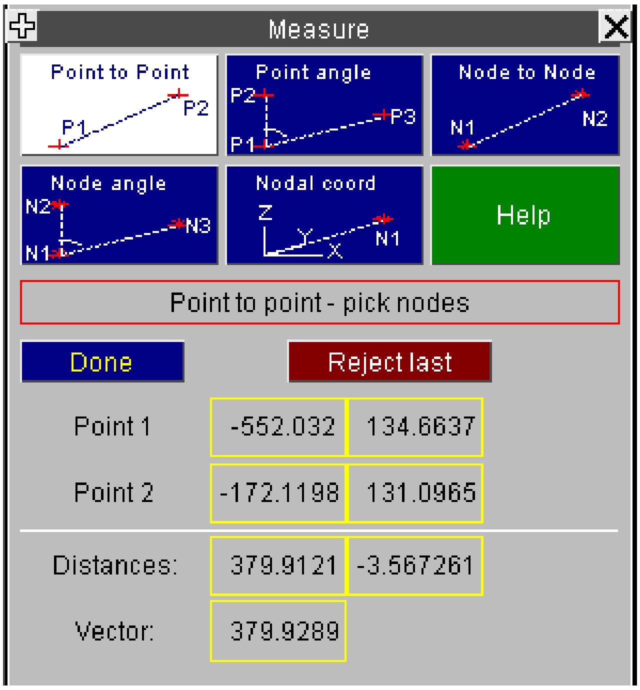
Therefore point-based measurement should be used when projected distances are required. For true 3D model space measurement it is better to use nodes.



6.20.1 Point to Point

Measures the projected 2D distance between two screen points.

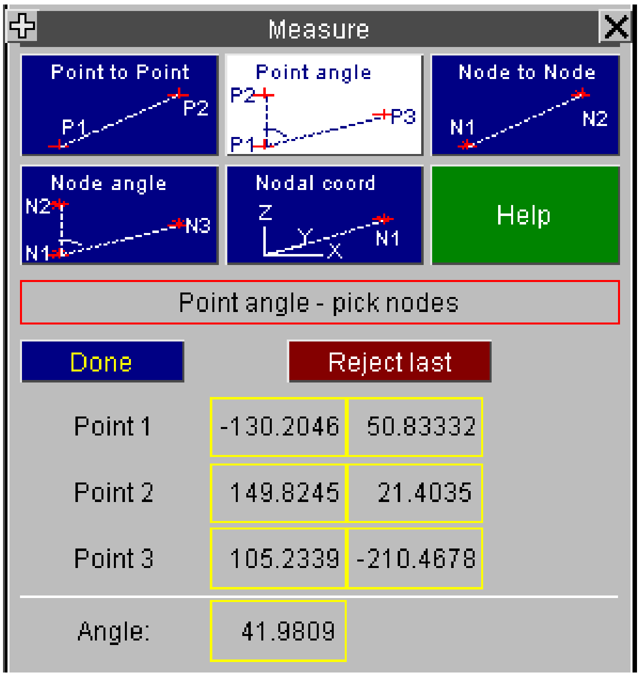
Select two points with the mouse, and the vector and (x,y) components of the distance between them is reported in screen space units.



6.20.2 Point angle

Point angle measurement computes the angle between the vectors P1P2 and P1P3.

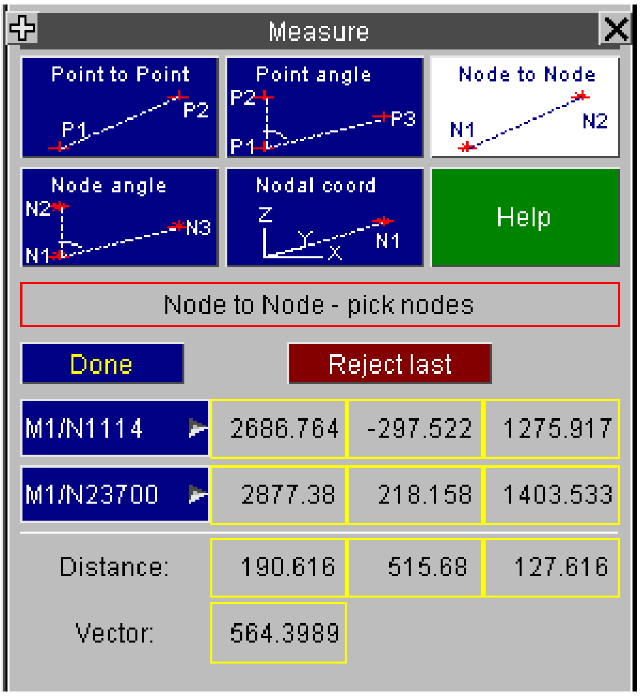
The angle is computed in the 2D screen plane and reported in degrees.



6.20.3 Node to Node

Node to node measurement computes the vector between nodes N1 and N2, and reports it as model space (x,y,z) and magnitude components.

Nodes may either be screen-picked, or have their label typed in, or use the standard popup options. As in this example nodes need not be in the same model.

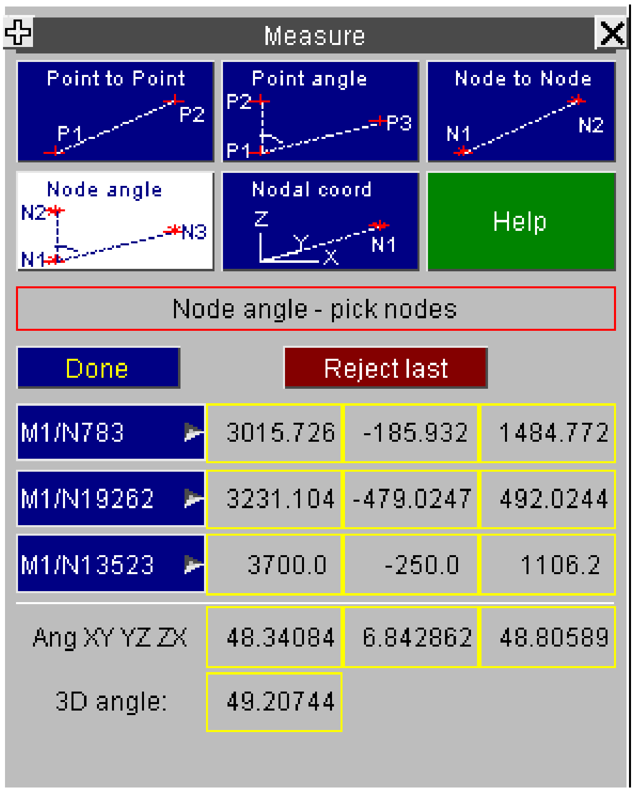


6.20.4 Node angle

Node angle measure computes the angle between vectors N1N2 and N1N3.

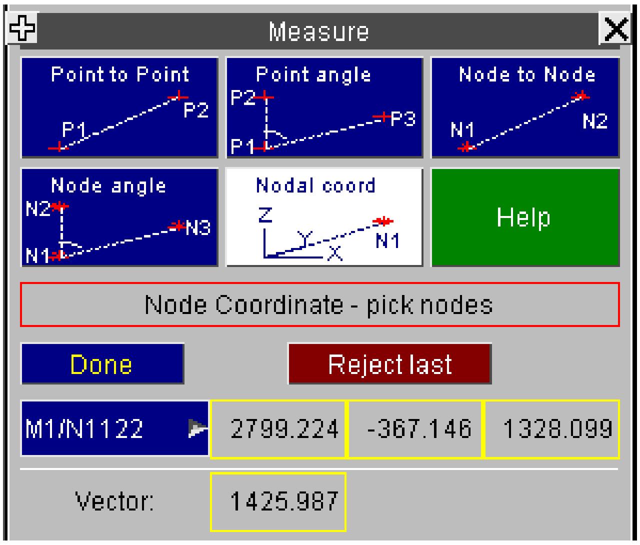
These are reported in model space (x,y), (y,z) and (z,x) planes as well as in 3D (x,y,z) space. Units are degrees.

Nodes are screen-picked or otherwise selected as in 6.5.3 above, and may be in different models.

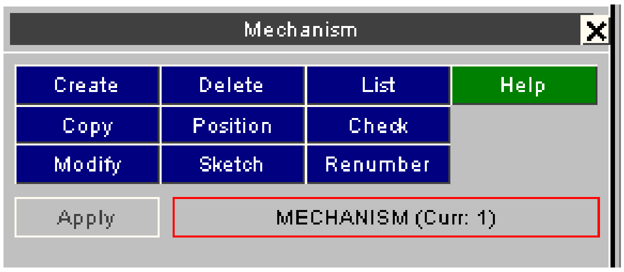


6.20.5 Nodal coord

Nodal coordinate gives the coordinates of the selected node in model space units, and also its distance from the origin.



6.21 MECHANISM Creating and analysing mechanisms



Mechanism analysis is a new capability in PRIMER release 9.3

It enables you to define a mechanism from components of your model and to analyse how they move as a collection of rigid bodies attached by joints of various types.

What is a "mechanism"?

A mechanism is a collection of two or more "assemblies" joined together by "connections".

- **Assemblies** are made up of any number of parts (deformable and/or rigid) and/or node sets, and for the purposes of mechanism analysis they are treated as totally rigid entities. There is currently a limit of 100 assemblies per mechanism.

- **Connections** may be one of a range of simple joints: currently pin, hinge and sliding. (These are not the same as the *Constrained Joint type in ls-dyna, although they have similar characteristics.)

The mechanism definition is stored in ***MECHANISM** cards following ***END**, in a format described in Appendix IIb. A model may contain any number of mechanism definitions.

Mechanism analysis is performed using an iterative scheme, so the results are approximate and will result in small distortions of the model. The default error tolerance is 1.0e-4 times the bounding box dimension around the mechanism, or ~0.1mm for a typical 1 metre sized mechanism. This amount of error is usually insignificant, but the convergence tolerance can be tightened, at the expense of longer calculation times, to achieve greater accuracy where required. A special check is included for failure to preserve coincidence of pairs of nodes defining ls-dyna joints, and automatic fixing of this can be carried out.

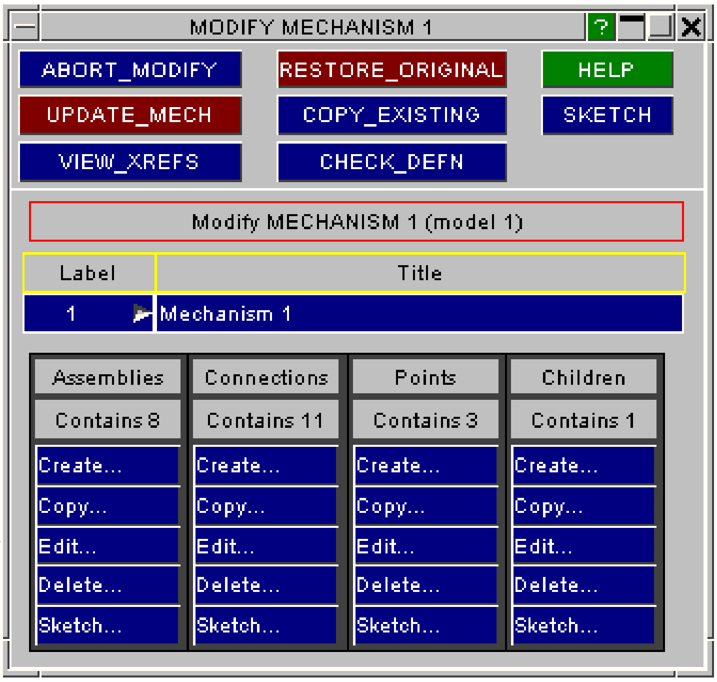
Once a mechanism has been created its motion can be analysed using rigid body kinematics: an assembly can be dragged using the mouse and the motion of the whole mechanism is calculated and displayed interactively. The new position can be saved and the coordinates of the assemblies updated to provide a new structural configuration for subsequent analysis.

Mechanisms may also have "points" defined within them. These provide both a means of creating restraints and also of driving motion.

Mechanisms may also be organised in a hierarchy where a "parent" mechanism drives the motion of a "child" via connected degrees of freedom. The "child" may also be a Dummy definition, permitting the dummy’s position to be driven by the motion of its parent mechanism.

> (There are many similarities between Mechanisms and Dummies in PRIMER , and indeed when positioning a Dummy in "free dragging" mode the dummy is in fact turned into a mechanism during the drag operation.)

6.21.1 Creating and Editing mechanisms



Mechanism definitions contain Assemblies, Connections, Points and Children.

Assemblies are collections of one or more parts and/or node sets. The parts may be any permutation of rigid or deformable.

Connections join assemblies together. At present PRIMER contains three connection types: pin, hinge and line.

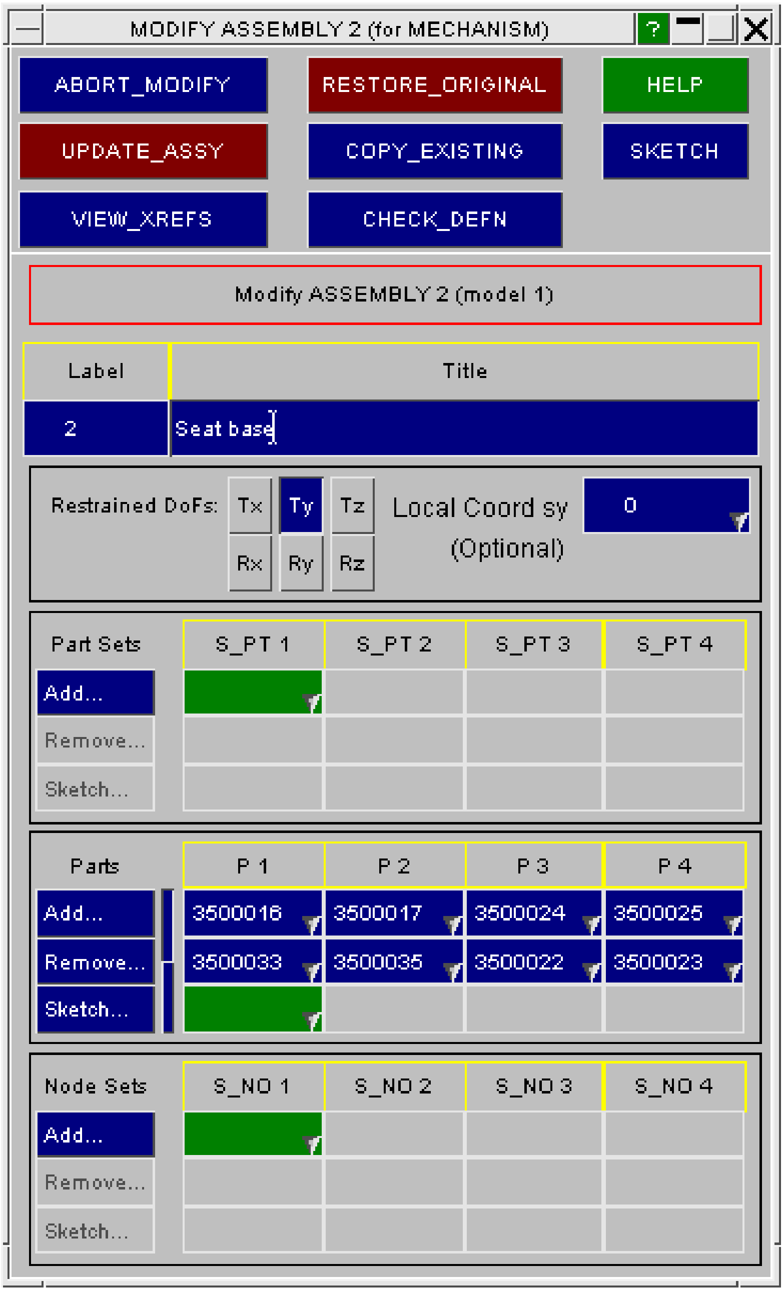
Points are optional, and any number may be defined. They are coordinates in space, "tied to" and a property of their parent assembly, that may have restraints in any combination degrees of freedom. If a local coordinate system is defined for a point then any restraints act in that system. Points may also be used to drive movement.

Children are optional. They may be other assemblies or Dummy definitions, and their motion is driven by their parent mechanism. Motion is transmitted in selected degrees of freedom from parent to child.

Use **Create...**, **Copy..., Edit...**, etc to select and operate on the relevant items.

When the definition is correct use **UPDATE_MECH** to save it.

Assembly Creation and Editing



Label and title	An assembly must have a label. Labels are "private" to this dummy, thus in a model with 2 dummies each may have assembly label 1 and they will not clash. A title is optional but is recommended.
Restraints and coord system	Assemblies may have optional restraints. These are used during "free drag" positioning to constrain movement. If a coordinate system is defined restraints are in that system, otherwise they are in the global system.
Part sets and parts	Any number of parts may be used in any permutation of explicit parts and part sets. It is legal to define a part more than once (eg in a set and explicitly): it will only be used once.
Node sets	Node sets may be added to the assembly. All nodes in these sets will move with the assembly.

(A significant difference between Dummy and Mechanism assemblies is that mechanism assemblies do not have any sense of hierarchy: there is no concept of a "root assembly" in a mechanism, nor is there a "tree" of connectivity. All assemblies have the same seniority, and the connections between assemblies can be completely arbitrary.)

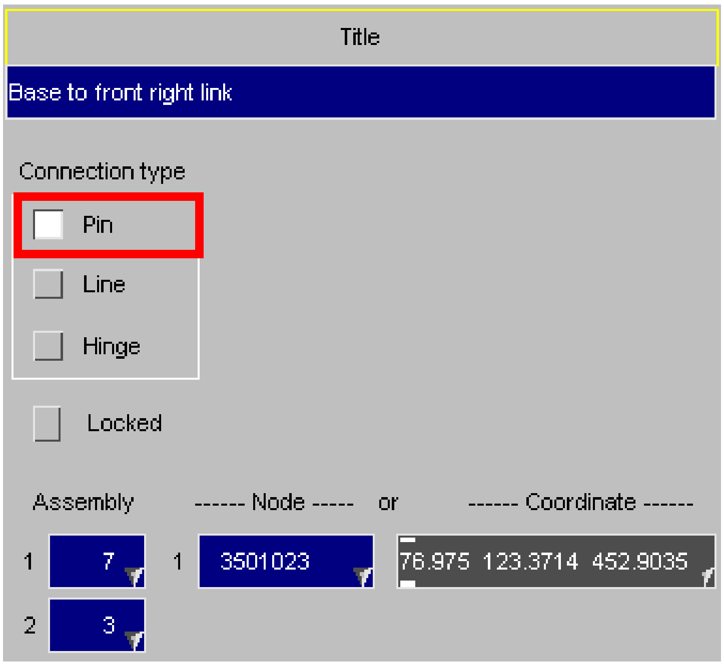
Connection creation and editing

There are three types of connection available, demonstrated here by their usage in a seat mechanism.

The examples below show how the various assemblies of a seat mechanism have been joined together with connections.

This seat has runners attached to the floor in which sliders travel fore and aft. The base is attached to the sliders by links, and the seat back to the base by a hinge at its base.

PIN joint



Two assemblies are specified (the order doesn't matter).

Either a node or a coordinate is also specified at the pin location point. The coordinates of the node, or the specified coordinate, are used to define the connection point (a parametric coordinate) on both assemblies.

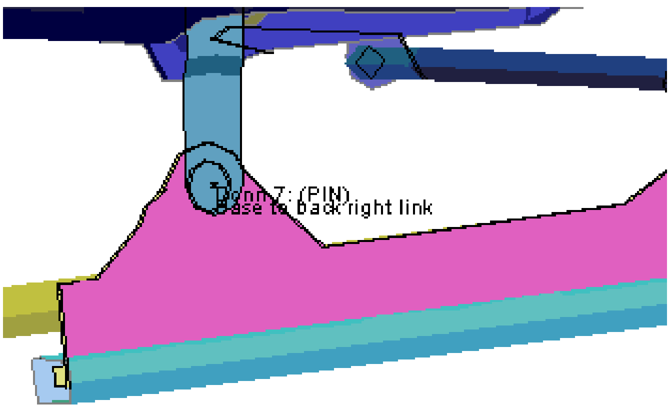
- If a node is specified it does not have to be on either assembly, but it is recommended that it is on one of them since, if its motion is not updated as the assemblies move, the connection point will move relative to the assemblies.

- If a coordinate is specified it is implicitly tied to assembly #1, and its motion will be updated with that assembly.

This joint, like all types, may be locked: see below for an explanation of this.

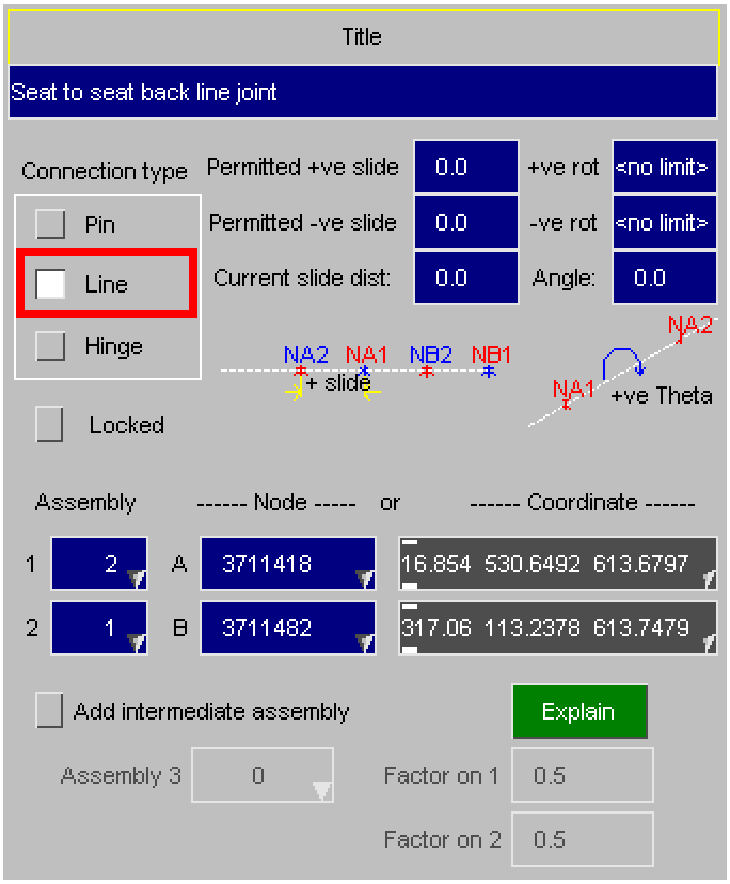
The pin acts like a spherical joint, providing connectivity in Tx, Ty, Tz; but no rotational constraint in its default unlocked condition.

In this example a pin joint has been used to connect the link between the sliding base of the a seat and the cushion frame.



A PIN connection joins two assemblies at a point, acting like a spherical joint.

LINE joint



A LINE joint connects two assemblies 1 and 2 along the line between points A and B. Either or boths points may be defined by nodes (NA and NB), or by explicit coordinates.

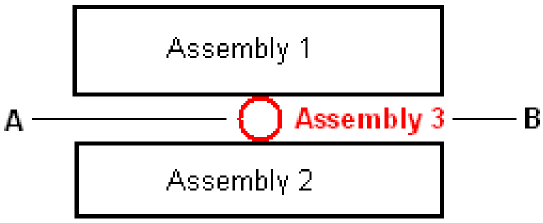
- If nodes are used neither node has to be on either assembly, but it is best to put both on one or the other so that they move as the assemblies move.

- If coordinates are used then their position is "tied" to that of assembly A only. A single assembly is used to prevent build-up of rounding errors during positioning.

In this example a LINE joint has been used to model the sliding of the runners fore and aft in the guide rails attached to the floor.

Optional assembly 3

There is also the option of adding a 3rd "intermediate" assembly between the first two, with its motion specified as some proportion P of assembly #1, and implicitly (1-P) of assembly 2.



The most common usage of this would be when defining a roller between two assemblies (here assembly 3 in red), where the roller motion is geared to that of assemblies 1 and 2.

There is no "master/slave" relationship: all assemblies are equal, and any assembly can drive the motion of the other two.Motion is constrained to axis A-B, the separation shown here is artificial.

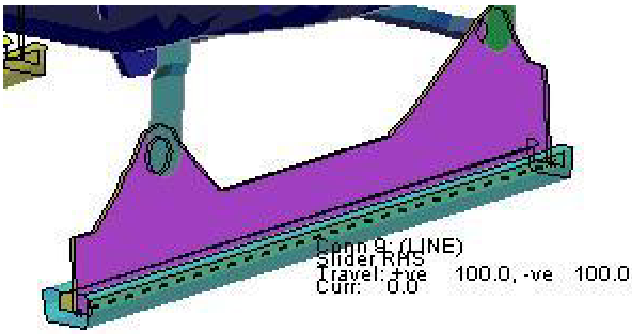
Motion is constrained to:

Sliding along the axis A - B

Travel is limited in both +ve and -ve directions to the permitted distances given, with the sign convention being:

+ve travel is assembly 1 moving along the vector A -> B

The "current slide distance" is set to zero when the connection is first defined, and thereafter is updated as the mechanism is analysed. The permitted limits and current value can be reset manually at any time in this panel.



A LINE joint allows assemblies 1 and 2 to slide along and rotate about axis NA-NB within the limits specified.

Rotation about axis A - B

By default there are no limits to this rotation, as shown in this example.

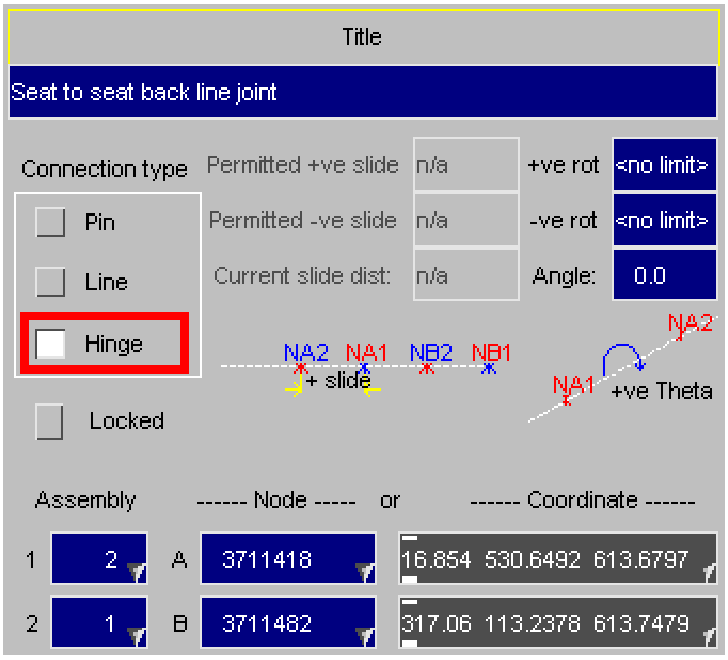
If limits are defined they should be expressed in degrees, the +ve rotation in the range 0 to +180 and the -ve rotation in the range 0 to -180. Rotation will be clamped to these limits.

+ve rotation is clockwise about vector A -> B

The "current rotation angle" is set to zero when the connection is first defined, and is updated and can be reset manually in exactly the same way as the translation distance - its actual value is notional.

If the "current rotation angle" value is changed it is important that any limiting rotation angles are also changed if necessary so that the -ve limit is less than or equal to the current value, and the +ve limit is greater than or equal to it.

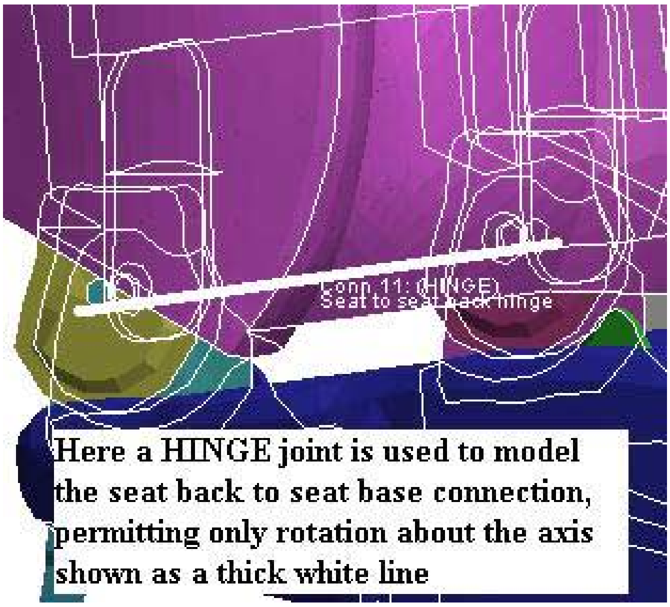
HINGE joint



A HINGE joint also connects two assemblies 1 and 2 along the line between points A and B, each point being defined either by a node or an explicit coordinate. It is exactly like the LINE joint above except that translation along the axis A-B is not permitted.

- If nodes are used neither node has to be on either assembly, but it is best to put both on one or the other so that they move as the assemblies move.
- If coordinates are used then their position is "tied" to that of assembly A only. A single assembly is used to prevent build-up of rounding errors during positioning.

In this example a HINGE joint has been used to model the seat back to seat base connection, permitting only tilting backwards and forwards about the transverse axis.



Only rotation about the axis A-B is permitted, within the limits specified, translation along that axis being restrained.

(A LINE joint with its permitted translation distances set to zero is exactly the same as a HINGE joint.)

Locking connections.

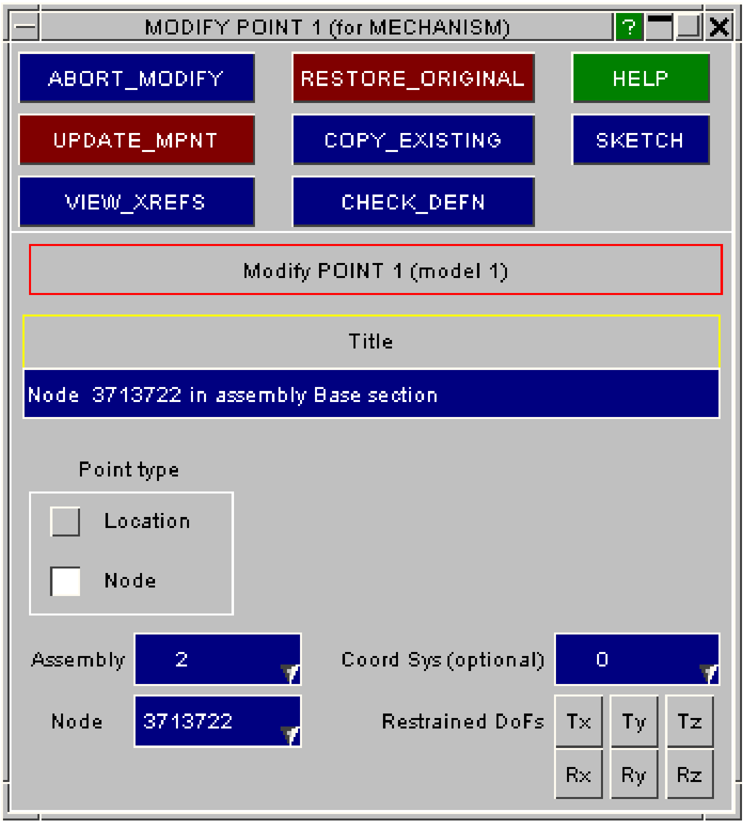
It is normally the case that connections will be required to articulate in the expected degrees of freedom while analysing a mechanism, but there are also times when it is useful to be able to lock a connection completely. For example in the seat example above you might move the fore/aft slider in its base runner to some position, and then lock it there.

Another example might be a headrest which must be slid to a particular height and then locked into position.

When a connection is locked all six degrees of freedom are constrained, and the effect is as if the two assemblies on either side have been merged into a single rigid assembly.

Connections may be locked and unlocked at any time, either on the connection editing panel as shown above or on the main positioning panel when it is in "Connection list" mode.

Point creation and editing



Title — A title will be generated automatically, but you can supersede this with your own.

Points do not have labels

Point type — A point defined by **Location** is a coordinate in space that is attached to, and moves with, its parent assembly.

A point defined by **Node** is essentially the same: it obtains it current coordinate from the node. (However if the node is not on a part or node set of the assembly it will not move with the assembly.)

The node should normally be part of the parent assembly, but this is not mandatory.

Restraints and coordinate systems — A point's movement may be restrained in any combination of degrees of freedom (or none).

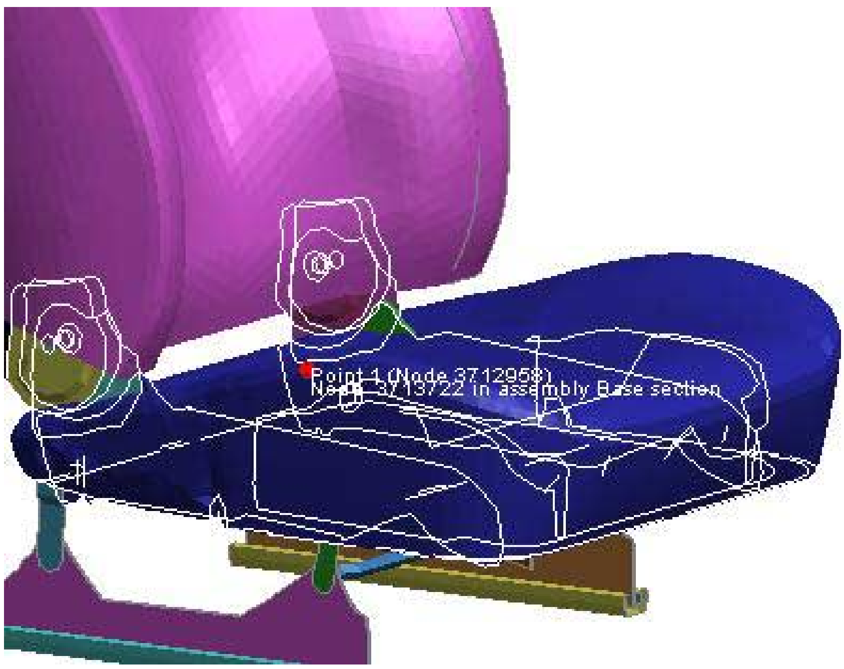
If a local coordinate system is defined restraints act in that system, otherwise they are global.

Visualising Points

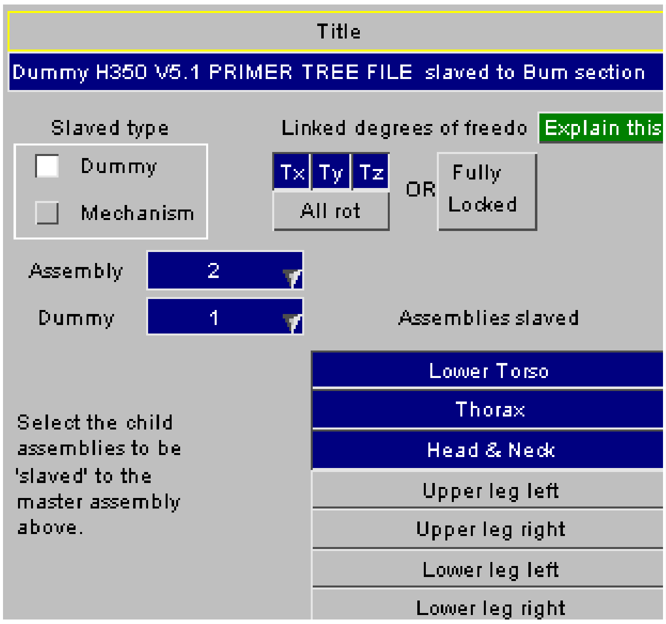
Points may be visualised by using the **Sketch** options both on the parent dummy panel and on their edit/create panels.

Here is a picture of the point in the example above: it is a reference point in the seat base.

It is shown as a circular red symbol, labelled with its title, with the free edges of the parent part also displayed.



CHILD mechanisms



It is possible to define a mechanism or dummy that is a "child" to this mechanism. You define the following:

The child type (here a Dummy) and the mechanism or dummy label.

The master assembly on this mechanism.

The degrees of freedom to be linked.

The assemblies on the "child" to be linked via these degrees of freedom to the master assembly. Here the Lower Torso, Thorax and Head & Neck have been linked.

Child mechanisms may be nested to any level (child has child has child ...). Dummies may not have children.

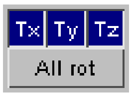
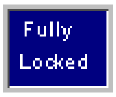
Warning: Mechanisms may not be recursive.

This means that a mechanism may not refer to itself as a child either directly (mechanism A has child mechanism A) or indirectly (mechanism A has child B which itself has child A). A moment's thought will reveal why this should be so: a mechanism cannot "drive" its own motion!

Primer will detect any attempts to create recursive mechanisms and report this an error; the positioner will also reject recursive mechanisms.

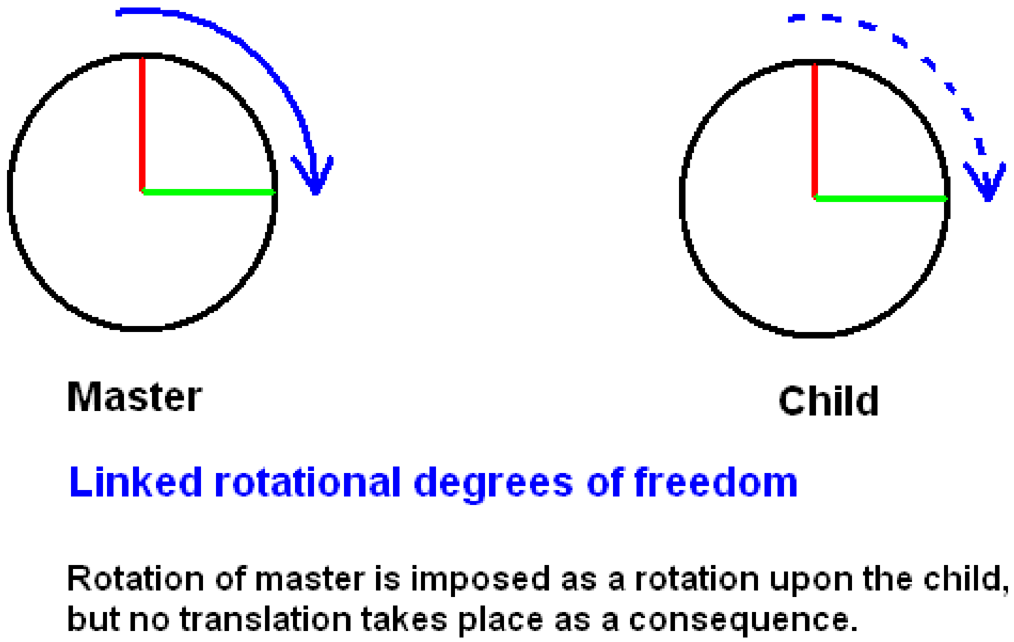
Linked degrees of freedom of children

PRIMER offers two related but different ways of slaving the motion of the child to its master.

	Linked degrees of freedom: **Tx, Ty, Tz** and **All rot**. The effect of these is similar to **`*CONSTRAINED_NODE_SET`** in that the chosen degrees of freedom of the master assembly are imposed on the child. Any permutation of the translational DoFs (Tx/y/z) and/or all rotational DoFs may be chosen. (Linking of individual rotational DoFs is not supported.)
	**Fully locked** The effect of this is like **`*CONSTRAINED_RIGID_BODY`**: the slave assemblies are merged into the master one to form a single rigid body.

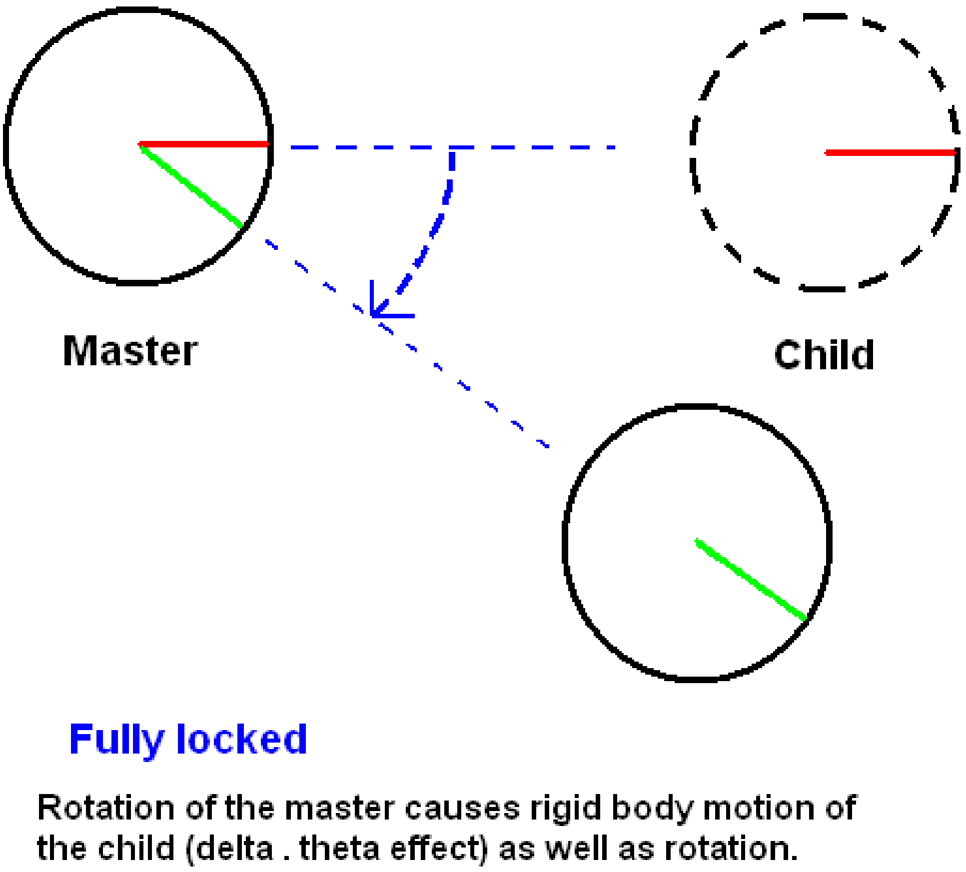
The effect of translation is easy enough to understand, and the two methods have the same effect in pure translation, but there are important differences between these two methods where rotations are concerned. In particular:

Selecting all linked degrees of freedom (**Tx**, **Ty**, **Tz** and **All rot**) is ***not the same as*** using **Fully locked**. The following figures explain why.



When rotational degrees of freedom are linked the rotation of the master is imposed on the child, but no translation arises from this.

A good way of thinking about this is to consider the master and child assemblies to be connected by a chain, like the pedals and back wheel of a bicycle. Rotating the pedals causes the back wheel to rotate, but has no tendency to try to lift it into the air.



When the master and child are fully locked then the child is both rotated and translated by the motion of the master since they are effectively a single rigid body.

Why have the two alternative linking methods?

Although **Fully locked** might at first sight appear to be the logical choice, experience has shown that when slaving dummies to seats the most natural behaviour is obtained if only the translational degrees of freedom (**Tx**, **Ty**, **Tz**) are linked. This is because any rotation of the seat cushion is not transferred to the dummy, which can remain in its upright position looking straight ahead even if the seat tilts underneath it.

How Child Mechanisms & Dummies work

When a child is slaved to a master mechanism the motion of the master assembly is imposed on the child assemblies in the degrees of freedom specified as described above.

During analysis the motion of the master assembly is computed and then applied to the child assemblies. There is feedback of force from the child to the master, so movement of the master will be constrained if it tries to push the child against a restraint. However in other respects it is a one-way treatment: moving child assemblies will not cause the master mechanism to move.

This is best demonstrated by example. Here a Dummy has been positioned in a cockpit, on a seat, and the dummy is a child of the seat linked in Tx, Ty, Tz.

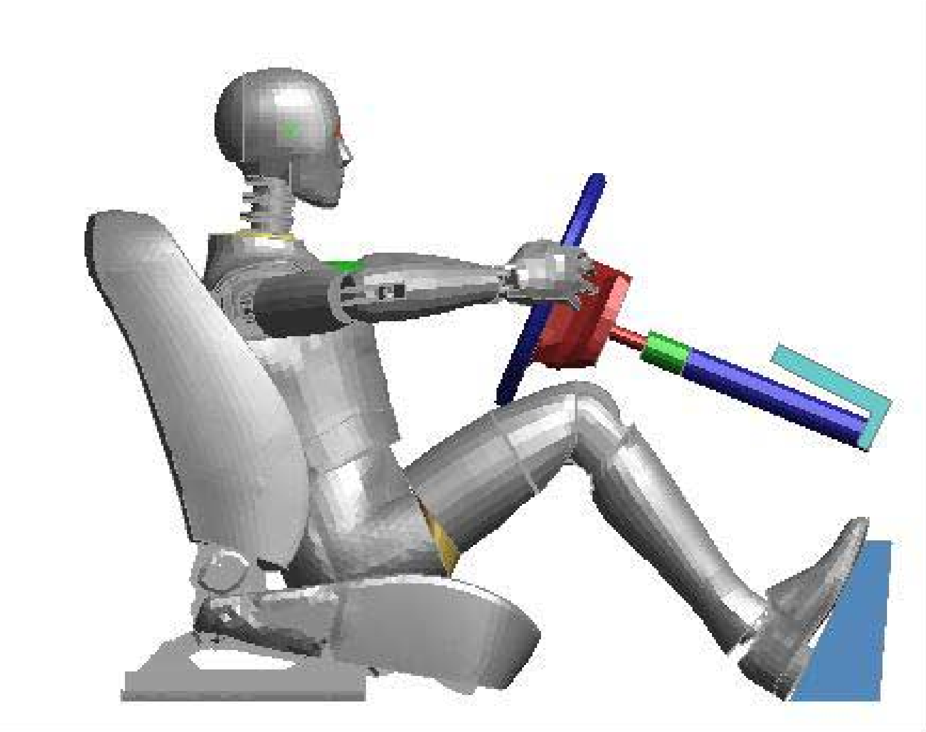
In this example the seat has been moved forward and down to a ridiculous degree, but this demonstrates two things clearly:

(1) The dummy motion has remained linked to that of the seat.

(2) Connection between seat and dummy is in translation (Tx, Ty, Tz) only.

This is made clear by the way that the seat cushion has tilted down but the pelvis, torso and head of the dummy have not rotated.

A more detailed exposition of the use of a Dummy as a Child of a mechanism is given in section 6.12.4 Using dummies as "children" of mechanisms, from which this image is taken.

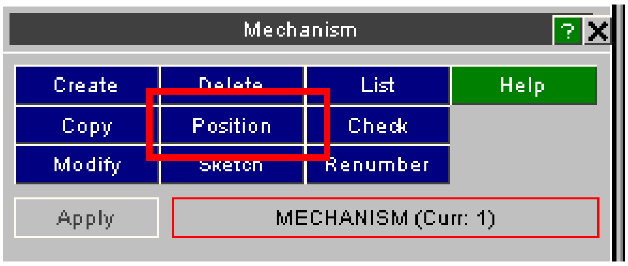


Mechanism Keywords in the input deck.

The information describing the mechanism is saved in special keywords following the *END card.

The formats of these cards (which are similar to those of a Dummy tree file) are given in Appendix IIb

6.21.2 **Position**: Analysing mechanisms



Once a mechanism has been defined, or read in from file, it can be analysed (positioned) in a variety of ways.

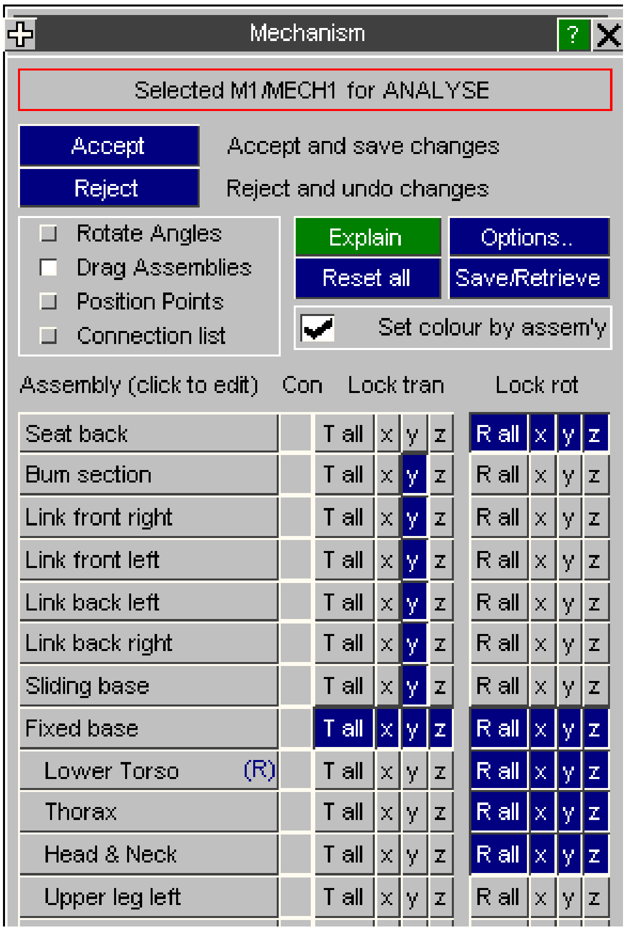
When you enter the mechanism positioner several operations are performed:

- Correctness of the mechanism definition is checked. Parts and nodes should not appear in more than one assembly, and you are warned if they do and given some options for diagnosing and correcting these errors. A more detailed treatment of potential errors and good mechanism modelling practice is given under Modelling Rules in Appendix II. (Although these rules refer to Dummies they apply equally to mechanisms.)

- You cannot have both **Dummy** and **Mechanism** positioning active at the same time in the same model. (This is because of the way positioning data is stored: the two processes would conflict.) If you attempt this you will be forced to shut down one operation before you can start the other.

- The current mechanism position is saved as an "initial position". If things go wrong in the positioner you can return to this as any time by using **Reset all**, and if you abort positioning using **Reject** the mechanism will automatically be restored to this position.

The main positioning panel



Assuming that these checks pass you then drop into the positioning panel. For mechanisms this operates in one of four modes:

Rotate angles	In this mode explicit rotation of assemblies about their parent connection node takes place.
Drag assembly	In this mode "free" dragging of the mechanism takes place, combining translation and rotation.
Position points	In this mode points can be defined and edited, and "free" movement performed by giving updated coordinates for them.
Connection list	Lists all connections (in mechanisms only) allowing you to lock them. Also to edit and sketch them.

A mechanism is positioned by any combination of these modes, and when it is satisfactory the user must **Accept** it to make the geometrical changes permanent, or **Reject** it to abandon positioning and restore the original geometry.

Set colour by assembly temporarily changes the colours used in the graphics so that this mechanism's assemblies are drawn in the standard PRIMER colour sequence, and the rest of the model in light grey. This is automatically switched off when you leave mechanism positioning.

Note that this example contains a Dummy model slaved to the parent seat mechanism. The Assembly names are indented to the right a little to emphasise that they are "children". The "(R)" against "Lower Torso" denotes that it is the Dummy's root assembly.

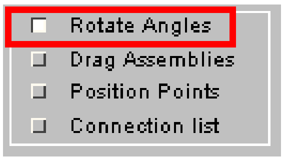
Further positioning commands below describes these and other options in more detail.

Dragging the mechanism with the mouse

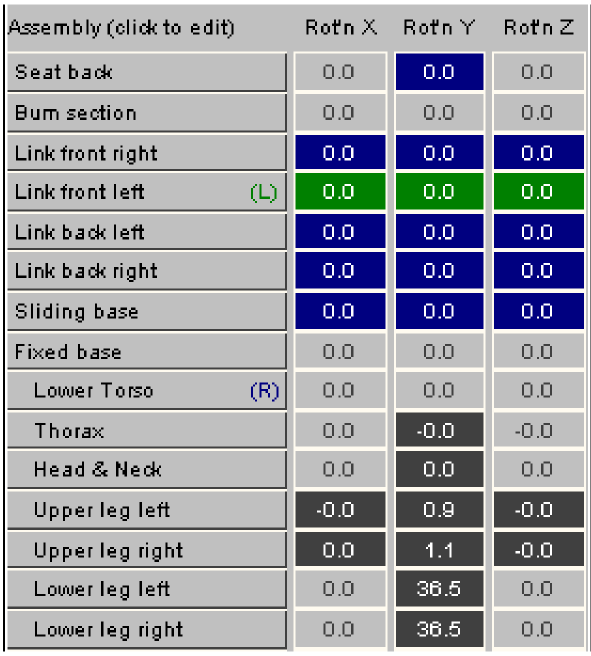
During Mechanism positioning the cursor is always active for picking and dragging assemblies, and operates in the same way (translational drag) in all four positioning modes above.

Mouse button	Action		Mode
Left	Normal select and drag		Motion is always translational drag in the direction implied by the current mouse motion in screen space as projected onto the model. (Note that this cursor behaviour is different to that in Dummy positioning: mouse-driven rotation is not available for mechanisms.)
Middle	As "left", but drag speed is x 2.5		
Right	As "left", but drag speed is x 5.0		

Rotate Angles: Rotation of assemblies



Rotating *Mechanism* assemblies:



Assembly (click to edit)	Rot'n X	Rot'n Y	Rot'n Z
Seat back	0.0	0.0	0.0
Bum section	0.0	0.0	0.0
Link front right	0.0	0.0	0.0
Link front left (L)	0.0	0.0	0.0
Link back left	0.0	0.0	0.0
Link back right	0.0	0.0	0.0
Sliding base	0.0	0.0	0.0
Fixed base	0.0	0.0	0.0
Lower Torso (R)	0.0	0.0	0.0
Thorax	0.0	-0.0	-0.0
Head & Neck	0.0	0.0	0.0
Upper leg left	-0.0	0.9	-0.0
Upper leg right	0.0	1.1	-0.0
Lower leg left	0.0	36.5	0.0
Lower leg right	0.0	36.5	0.0

For each assembly a row showing the current joint angles is shown. Angles on a blue background are in the mechanism's native system, those on a green background are in the local axes of the assembly's coordinate system where this has been defined.

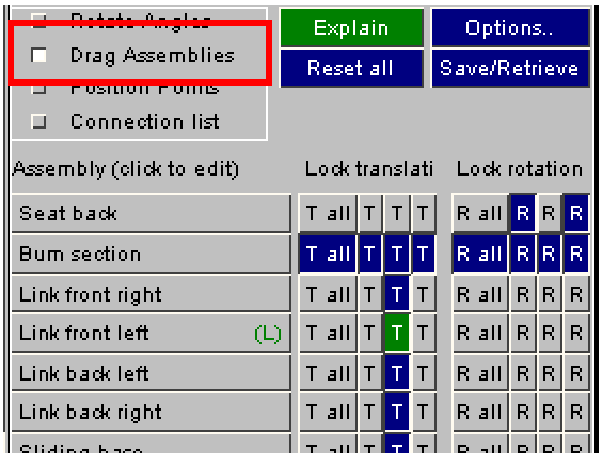
Typing in a new angle will cause the assembly to rotate to the new value using the iterative analysis (free drag) calculation method.

Mechanism assemblies are initially set to angle zero degrees (as shown here) when first defined, and thereafter their rotation angles are computed from their rotation relative to that initial orientation.

Angles are expressed in the mechanism's native system

In order to give consistent calculation and reporting of angles for assemblies, those with no local coordinate system are reported in the Mechanism's native system (on a blue background).

This is assigned automatically and is initially aligned with the global cartesian system, however if the mechanism as a whole is rotated with the **Orient** command then this system also rotates, thus reported angles will not change.



Rotation of mechanism assemblies is only permitted about axes that are not restrained.

You can view the status of these restraints by swapping to the "Drag Assemblies" mode of the positioning panel.

You can see here that the seat base ("Bum section" - a technical term!) is fully restrained in all degrees of freedom, while the seat back is restrained only against rotations Rx and Rz in anticipation of the example rotation below.

Rotating *Dummy* assemblies

Rotation works somewhat differently. For compatibility with the Dummy angle positioning process:

- Angles are computed relative to the coordinate system of the joint stiffness coordinate system at the parent side of the assembly's attachment node.
- Angular changes are applied explicitly via trigonometrical calculation, not via the iterative dragging scheme, and take no notice of assembly or point restraints.
- Both the selected assembly ***and its children*** are rotated as a rigid unit, and rotation only takes place about a single axis at a time.

This behaviour is very different to that of rotations of mechanism assemblies, and the angle entry buttons have a dark grey background to reinforce this point. It is usually best to position a Dummy in the occupant positioner, and then let its behaviour as a "child" of a mechanism be driven by the motion of that parent mechanism.

Please see **Rotate Angles** in the dummy documentation for more details.

Example of positioning a mechanism assembly by rotation.

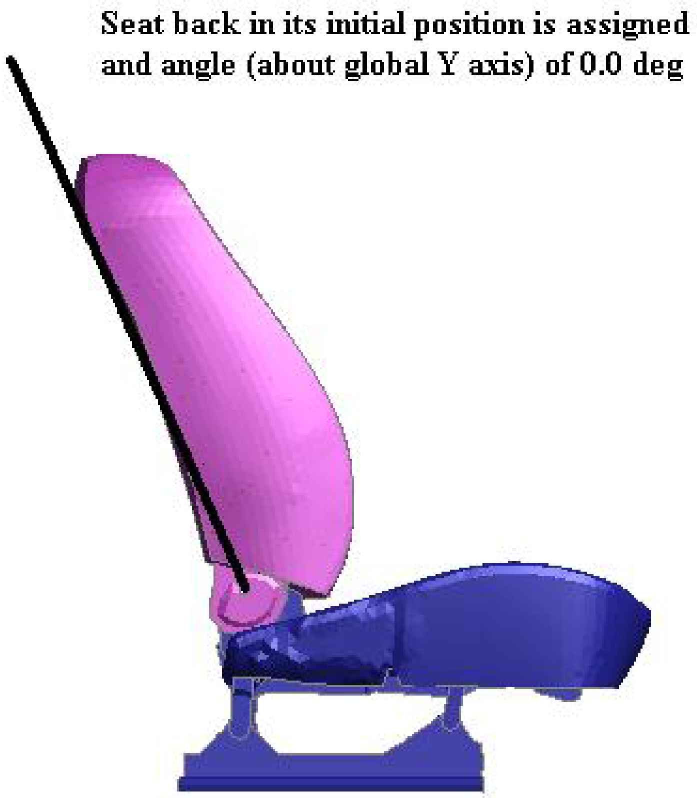
Remember that the seat back in this demonstration model is connected to the seat base by a HINGE connection permitting rotation only.

This image shows the seat back in its initial position.

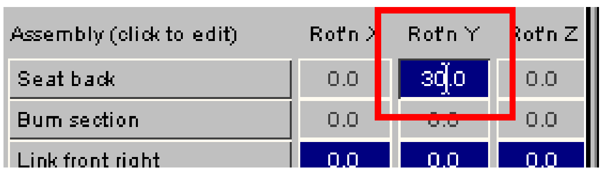
This becomes the initial orientation of 0.0 degrees about all axes.

We propose to rotate by 30 degrees clockwise about the global Y axis, which is coming towards the observer out of the plane of the page.

(*Because the seat has not been rotated as a whole using* **Orient** *the mechanism axes are aligned with the global axes. If the mechanism as a whole is rotated then rotations will take place about the rotated axes - see the notes on the mechanism's native coordinate system above.*)



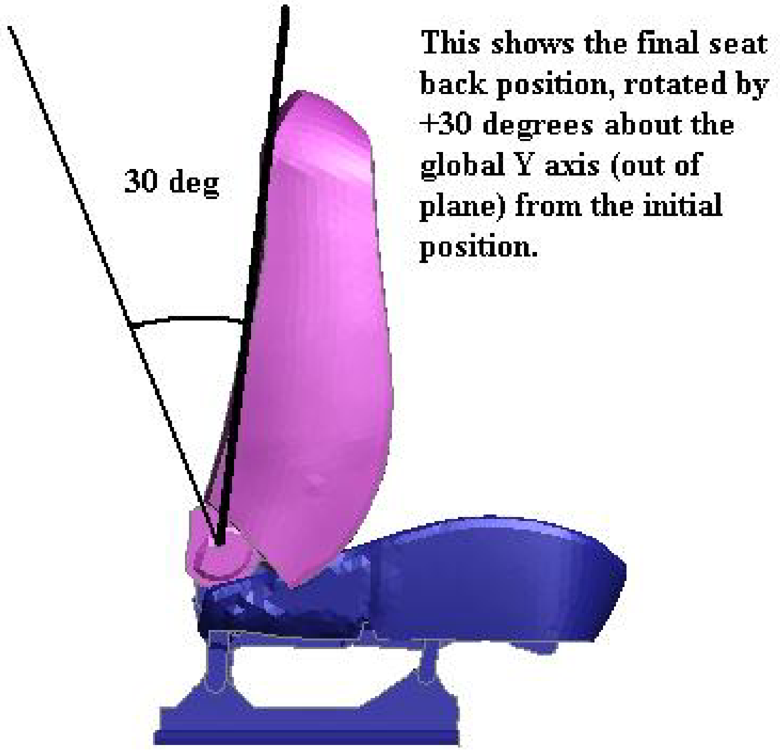
The user types the new angle into the Y rotation column



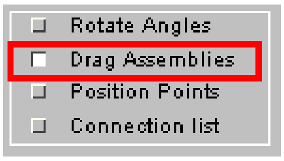
PRIMER drives the rotation automatically, using the iterative calculation method.

Feedback on progress achieved is given every 10 iterations.

Here is the final position, achieved after about 40 iterations.



Drag Assemblies: Free dragging of assemblies



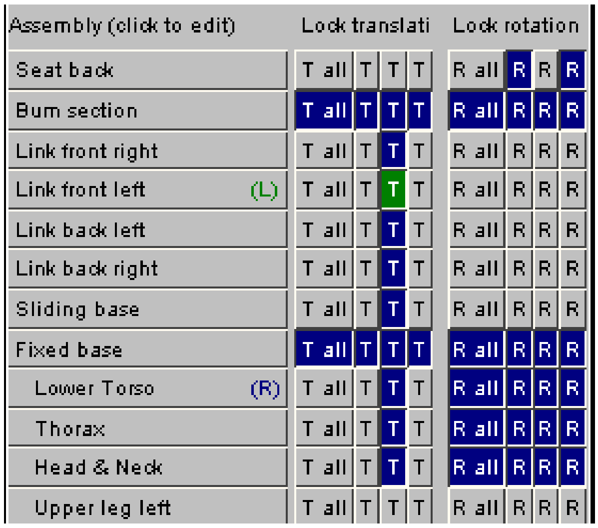
In **Drag Assemblies** mode the positioning panel changes.

Each assembly is still shown as a row, but now:

- Clicking on the "name" button brings up the assembly editing panel as above.
- You can select the degrees of freedom to be restrained (locked) during positioning for each assembly. Restraint acts in the coordinate system of the assembly (if defined), otherwise in the global system.

Restraints shown in blue are in the global system, those in green (here "Link front left") are in the local system of the assembly.

Restraints can be set and unset at any time during positioning.

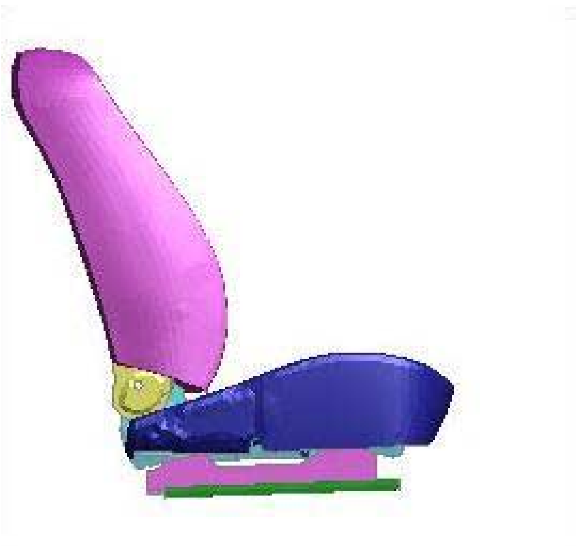


Assembly (click to edit)	Lock translati				Lock rotation			
Seat back	T all	T	T	T	R all	R	R	R
Bum section	T all	T	T	T	R all	R	R	R
Link front right	T all	T	T	T	R all	R	R	R
Link front left (L)	T all	T	T	T	R all	R	R	R
Link back left	T all	T	T	T	R all	R	R	R
Link back right	T all	T	T	T	R all	R	R	R
Sliding base	T all	T	T	T	R all	R	R	R
Fixed base	T all	T	T	T	R all	R	R	R
Lower Torso (R)	T all	T	T	T	R all	R	R	R
Thorax	T all	T	T	T	R all	R	R	R
Head & Neck	T all	T	T	T	R all	R	R	R
Upper leg left	T all	T	T	T	R all	R	R	R

An example of **Drag Assemblies** free dragging.

The following sequence of images shows how this might be used in practice. In this example the dummy has been positioned in the seat, with hands attached to the steering wheel and feet to the pedals. Both hands and feet are fully restrained in all degrees of freedom, the torso, thorax and head are restrained against all rotations and also Y (out of plane) translation.

The user has clicked on the lower torso with the left mouse button, so the whole dummy is selected for movement, and drags it progressively further forwards. This sequence would be carried out in a single operation, and for this dummy the drag occurs in near real-time on a modern desktop computer.



Initial condition.

The user is about to click on the seat cushion (blue) and move it up and forward

In progress.

The seat has to rise upwards on its links in order to move forwards

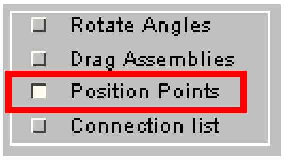
Further progress.

The seat has made progress forwards, rotating on its fore and aft links.

Final position.

The seat has come back down again to achieve its final forwards position.

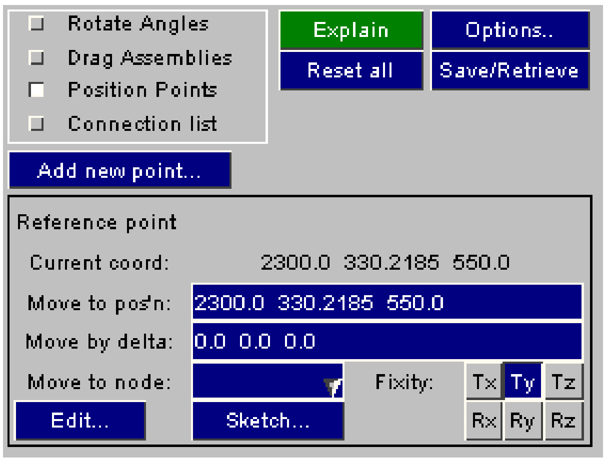
Move Points: movement driven by updated point positions.



An alternative to dragging with the mouse is to set new target positions for points. As described above any number of points can be defined in an assembly, and used both to apply localised restraint and to drive movement.

In this example a point has been created in the seat base, acting as a reference point for movement.

It can be moved by any combination of the following three methods:



Move to pos'n Will move the point ***to*** the new coordinate specified

Move by delta Will move the point ***by*** the specified distance [dx.dy,dz]

Move to node Will move the point ***to*** the position of the chosen node.

In all cases the effect is similar to dragging with a mouse, with the difference that PRIMER will drive the iterative scheme for you to try to achieve the new position.

Iteration will continue either until the target point is reached, or the changes between successive iterations become insignificantly small. The latter is necessary since, obviously, it is possible to set a target position for a point that cannot be achieved because of restraints.

Using wild-card coordinates for movement of positions.

It is sometimes the case that you want to move a point a certain distance along one axis, but not to constrain its movement along other axes. To allow for this PRIMER permits the following "wild-card" as opposed to"explicit" coordinate entry syntax.

Values entered explicitly as zero mean exactly that. Therefore

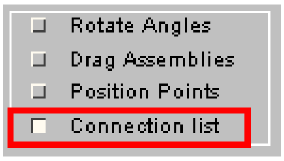
[*Move by delta*] 100.0 0.0 0.0	Means "move by 100 in X, but try not to have any movement in Y or Z."

Omitted trailing values, or values entered as an asterisk "*" are taken to mean "not constrained". Therefore

[*Move by delta*] 100.0	Means "move by 100 in X, but don't care about movement in Y or Z"
	Means "move by 100 in Z, but don't care about movement in X or Y"

The same syntax may be used for absolute [*Move to pos'n*] coordinate entry.

Connection list: editing and locking connections

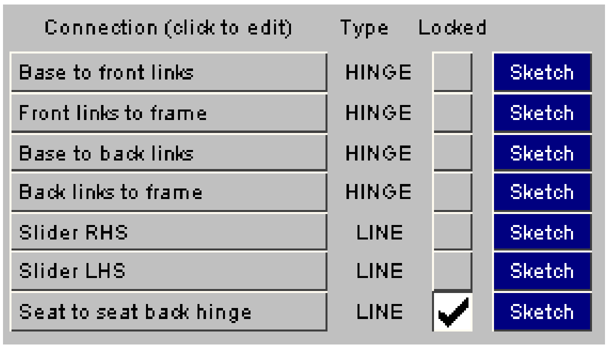


This panel lists all the connections in the mechanism (but not in any child dummies),

Connections may be edited by clicking on their name button, and also sketched.

More usefully they can also be locked and unlocked: in this example the connection between seat and seat back has been locked. (Un)locking can take place at any time so, for example, two assemblies can be moved relative to one another and then locked in that configuration for subsequent positioning.

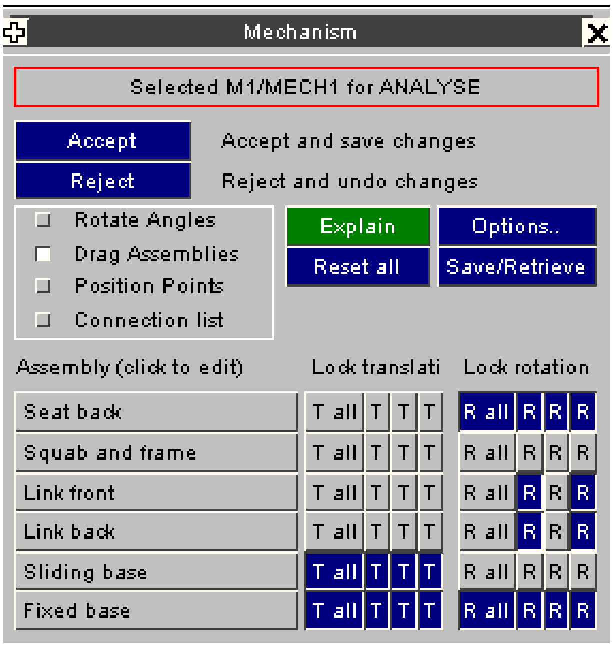
"Locking" makes the connection totally rigid in all 6 degrees of freedom, with the effect that the two assemblies on either side become merged into a single rigid combination.



Further positioning commands

The following commands are common to all four positioning methods described above.

Accept	Accept position and save changes.
Reject	Abandon positioning, and restore initial position
Reset all	Restores the mechanism to its initial position
Options...	Further positioning options
Save/Retrieve	Save and retrieve positions



Accept: accepts the current position and saves the changes you have made.

Once you are happy with the current position use **Accept** to save it and finish positioning.

Before it saves the position PRIMER checks all the nodes at (ls-dyna) joints in the mechanism to check that positioning has not pulled them apart. It applies a twin tolerance:

- An absolute value of 1.0e-3. This is the value hard-wired into the LS-DYNA keyword reader.
- A distance of 1.0e-6 times the model longest diagonal

Any joints at which separation of nodal pairs exceeds this figure will be listed and you will be given the option of **Autofixing** them.

This is performed by moving each pair of nodes to their average position and, so long as the errors are small, this is an acceptable distortion of the model. This is an iterative process since if a node is on more than one joint then correcting for joint A may move it out of position for joint B. If there are still errors after 5 passes the operation is abandoned and it is left to the user to sort out.

Warning: Avoid repeated [Position, Accept, Autofix] cycles on a model.

This is because each **Autofix** operation changes the geometry of your mechanism slightly, and while a single such change may be insignificant repeated use of this feature will build up cumulative errors. In particular repeated **Autofixing** of joints, which moves pairs of nodes to their average coordinate, may introduce alignment errors.

It is better to achieve a position in a single operation from an unmodified mechanism model. If you are planning to generate a series of positions in succession you should use **Model, Copy** to create a new model from an original one each time, and create the position in the copy.

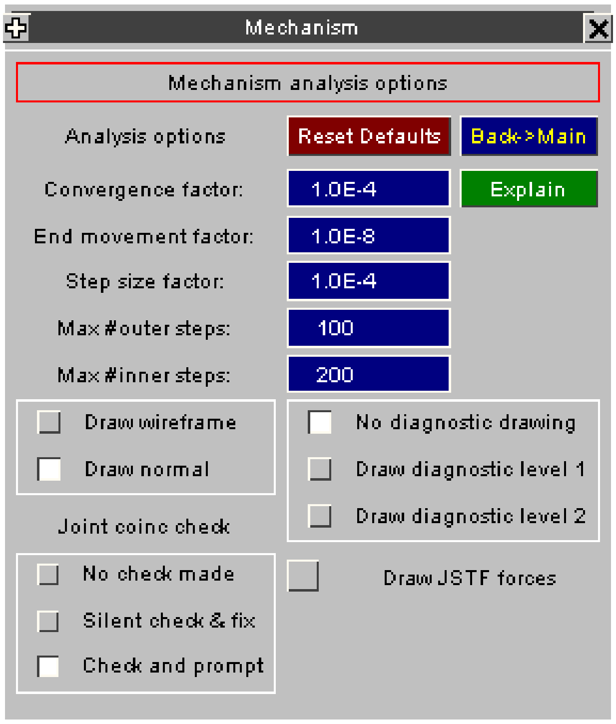
Reject: rejects the current position, restores the initial one and exits the positioner.

Use **Reject** if you want to abandon positioning and restore the initial position. All changes made during positioning will be lost, and you will return to the main **Position** menu with the model unchanged.

Reset all: restores the intial position.

Sometimes positioning goes horribly wrong and the best thing is to start again. **Reset all** restores the initial position that was saved when you entered the positioner, cancelling all changes made since then. You can use this at any time.

Options... Setting positioning options.

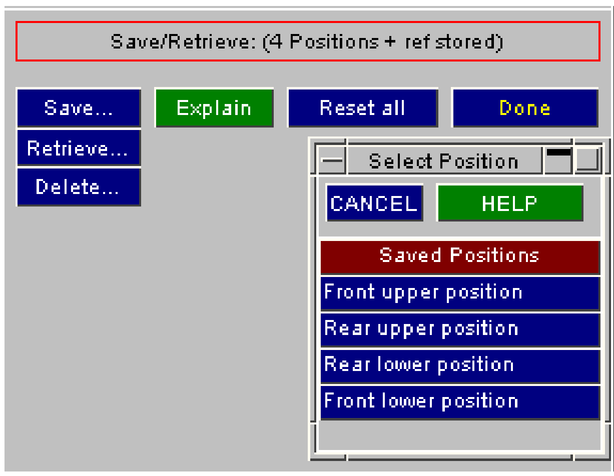


The following options affect the positioner:

Convergence factor End movement factor Step size factor Max #outer steps Max #inner steps	These options all affect the free dragging positioner only. They should not normally need to be changed, but if the mechanism moves very slowly, or "gets stuck", then increasing the Convergence and Step size factors may help. Avoid much larger values as the solution will become inaccurate.
Draw wireframe Draw normal	Sets the graphics mode to be used when dragging assemblies. "Draw normal" shows the assembly in grey using normal graphics, but this demands quite a lot of cpu time and only slower computers "Draw wireframe" may be necessary to get acceptable dragging speed.
Joint coinc check	Is the post Accept joint node coincidence check. By default it will check and report any results, asking you what action to take. You can it work silently, which will fix any errors without further input, or turn it off altogether.

Diagnostic drawing and JSTF force display are for programmer debugging purposes and - hopefully - can be ignored!

Save/Retrieve: Saving and restoring mechanism positions.



You can store any number of mechanism positions, and retrieve them at any time into the positioner.

A stored position contains a [centre of gravity] plus [3x3 direction cosines] for each assembly in the mechanism, making it possible to return to a given configuration without any further calculation.

Position data is stored with the mechanism definition and is saved when the keyword file is written out.

Save... Saves the current position. You only need to give a unique name.

Retrieve.. Restores a saved position. This becomes the current position and the dummy geometry is updated immediately causing it to "jump" to the new position.

Delete... Deletes the selected positions. Deletion is permanent!

Where a mechanism has "child" mechanisms or dummies positions are saved and retrieved for these as well. Since position selection is by name retrieving a position which does not exist in a child will leave the child unmoved.

A more detailed explanation of saved positions, including card formats, is given in Appendix IIc.

6.21.3 Batch (command line) positioning

A subset of the interactive positioning commands described above are also available in command-line line form. While these can be used interactively the main purpose of them is to enable positioning to be performed in batch mode. These commands will provide visual feedback if the graphical user interface is running, but if it is not (PRIMER started with "-d=batch" command line option) they will still function. A full listing of command-line commands is given in Appendix X11.

The positioning commands invoked by the [Primer >] **MECHANISM** command, and occupy a hierarchy as follows:

At **MECHANISM>** level			
ASSEMBLY	Select an assembly by name or number, then perform one of the following operations upon it:	**FIX** *dof code*	Restrain the assembly in degrees of freedom *dof code*
		TRANSLATE *dx, dy, dz*	Translate assembly ***by*** amount *dx,dy,dz*
		RX or **RY** or **RZ** *theta*	Rotate assembly ***to*** angle *theta* degrees about x/y/z
		RESET	Undo all dummy transformations and return to initial state
		DONE	Finish with assembly and return to **MECHANISM >** prompt
POINT	Select a point by name or number, then perform one of the following operations upon it: (Note: moving the point implicitly moves its "owner" assembly.)	**FIX** *dof code*	Restrain the point in degrees of freedom *dof code*
		TRANSLATE *dx, dy, dz*	Translate point assembly ***by*** amount *dx,dy,dz*
		POSITION *x, y, z*	Translate point assembly ***to*** coord *x, y, z*
		RESET	Undo all dummy transformations and return to initial state
		DONE	Finish with point and return to **MECHANISM >** prompt
CONNECTION	Select a connection by name or number	**SLIDE** *distance*	Applies to **LINE** connections only, and will slide the joint by *distance* down its AB axis.
		ANGLE *theta*	Applies to **LINE** and **HINGE** connections only, and rotations the assemblies to achieve angle *theta* (in degrees) about the AB axis.
POSITION	Specify a position *name*	Retrieves and applies the stored position *name*	
READ_CONFIG	Specify a *filename*	Retrieves a free-standing dummy configuration file (the keywords and data between ***MECHANISM_START** and ***MECHANISM_END**). *Filename* will usually have the extension .mcf	
ACCEPT	Accept the current mechanism position, save its updated geometry and return to the main [Primer >] prompt.		
RESET	Undo all transformations and restore the initial geometry of the mechanism, remaining at this prompt level.		
QUIT	Undo all transformations and restore the initial geometry of the mechanism, then return to the main [Primer >] prompt.		

Meanings of terms in the table above

dof code Is a numeric Degree of Freedom code made up of any permutation of 123456, where

1 = Tx, 2 = Ty, 3 = Tz, 4 = Rx, 5 = Ry, 6 = Rz

For example code **136** means restraint in Tx, Tz, Rz

Code **0** may also be used, meaning "free all restraints" *dx, dy, dz* Is a translation vector, ie a relative movement from the current position, made up of three numbers.

For example **`10.0 20.0 30.0`** means translate 10.0 in X, 20.0 in Y, 30.0 in Z.

"Wildcard" syntax is permitted: any number entered as an asterisk ("*"), and omitted trailing digits, are treated as "free" values. For example:

`10.0`	means translate 10.0 in X, but permit Y and Z to adopt any value.
`* * 20.0`	means translate 20.0 in Z, but permit X and Y to adopt any value

x, y, z Is an absolute coordinate.

For example **`10.0 20.0 30.0`** means coordinate X=10, Y=20, Z=30.

Wildcards as for translations above are permitted

theta Is an angle in degrees for the given degree of freedom.

In a dummy model angles are absolute values expressed in the coordinate system of the connection between this assembly and its parent. In most cases this will mean the system implied by the local axes of the joint stiffness definition at the joint.

6.21.4 Using mechanisms and dummies as "children" of mechanisms

Both mechanisms and dummies may be defined as "children" of a mechanism.

A "child" is a separate Dummy or Mechanism in which one or more assemblies are linked to an assembly in the master in any combination of Tx, Ty and Tz degrees of freedom. When the master mechanism is positioned then its motion also drives the motion of the linked assemblies in the child causing it to be positioned too.

Force feedback from child to master takes place, so the master will feel resistance if it tries to push the child where it doesn't want to go. However this is a one-way treatment: while child assemblies can be moved within their own child mechanism this will not transmit force to their master, so dragging child assemblies will not move the master mechanism.

Child ***mechanisms*** may be nested to any depth: a child mechanism may itself have children.

Child ***dummies*** may not be nested: dummies cannot have children.

The principal use of this capability is to position a dummy on a seat mechanism, and this is described in section 6.12.4: Using dummies as "children" of mechanisms.

The process of defining and linking together master and child is described in CHILD mechanisms above.

6.21.5 Mechanisms and *INCLUDE_TRANSFORM

It is sometimes useful to place mechanism assemblies inside include files which use **`*INCLUDE_TRANSFORM`** plus **`*DEFINE_TRANSFORMATION`** to define their orientation and position.

PRIMER detects this automatically, and when such mechanisms are positioned an extra 4 lines are added to the relevant **`*DEFINE_TRANSFORMATION`** keywords which are equivalent to the change in each assembly's position.

Lines added by PRIMER to **`*DEFINE_TRANSFORMATION`**
`TRANSL dx dy dz` **`ROTATE 1 0 0 cx cy cz tx`** **`ROTATE 0 1 0 cx cy cz ty`** **`ROTATE 0 0 1 cx cy cz tz`**

<cx cy cz> is the centre of rotation, tx/y/z the angle in degrees.

Note that all four lines are always written, even when a given transformation is in fact zero.

This is to enable PRIMER to "know" what is there and overwrite it if the assembly is re-positioned during a session, thus avoiding a build-up of many lines.

A consequence of this is that you must not mix mechanism positioning and manual editing of these **`*DEFINE_TRANSFORMATION`** definitions within a single session, otherwise you may confuse the logic which updates these lines.

The effect on output from PRIMER is that such assemblies will revert to their untransformed position, and on subsequent reading into LS-DYNA (or back into PRIMER) will revert to their "as positioned" state when the transforms are applied.

Within a single PRIMER session these transformations are overwritten if the assembly is repositioned, so that a session will only ever add an extra 4 lines. However if you exit and restart PRIMER (or reread or copy the model) then any subsequent positioning will create a new block of 4 lines.

Therefore if a mechanism is to be generated in several different positions these should be created within a single PRIMER session or, which would be better because it would also reduce the build-up of small residual errors, by starting each time from the original (untransformed) file.

Rules when using *INCLUDE_TRANSFORM with mechanisms:

- PRIMER only peforms these modifications if an assembly is in one or more **`*INCLUDE_TRANSFORM`** files ***and*** these each refer to an existing **`*DEFINE_TRANSFORMATION`** definition.
- PRIMER assumes that a given **`*DEFINE_TRANSFORMATION`** definition is used ***only*** by the **`*INCLUDE_TRANSFORM`** file(s) that contains this assembly. (But see the change for 9.3.1 described below.)
- PRIMER permits an assembly to be spread over several **`*INCLUDE_TRANSFORM`** files (although this is not recommended) and each such file must either refer to the same **`*DEFINE_TRANSFORMATION`** definition, or to separate definitions that are used only by these files.
- PRIMER will create a new block of the four **`*DEFINE_TRANSFORMATION`** lines described above when an assembly in such an include file is first positioned, and it assumes that it "owns" these last four definitions for the duration of the session. If you edit these cards manually in a session in which you have performed mechanism positioning the results may go wrong.

Recommended modelling practice when using *INCLUDE_TRANSFORM with mechanisms:

It is ***strongly*** recommended that if **`*INCLUDE_TRANSFORM`** files positioned by **`*DEFINE_TRANSFORMATION`**s are to be used with mechanisms (and dummies) then the following practice be adopted:

1. **Use a "one include file per assembly" policy.**

   Put the nodes, parts, elements, sets etc that make up an assembly into a single file, do not split them over multiple include files.

2. **Also use a "one assembly per include file" policy.**

   Do not put multiple assemblies in an include file, as a single **`*DEFINE_TRANSFORMATION`** cannot then be applied to them.

3. **Do not mix "assembly" and "non-assembly" data in include files.**

   This will avoid transformations being applied to "non-assembly" structure that should not be moved.

4. **Use a unique transformation for each such include file.**

Create a single ***DEFINE_TRANSFORMATION** for each such include file, and use it only for that file. This will avoid possible conflicts.

If this practice is adopted then it is highly unlikely that things will get confused.

Rule (1) above is not strictly necessary but it is good practice, and it should avoid problems arising from small dimensional errors where part of an assembly is moved en-bloc by a ***DEFINE_TRANSFORMATION** and part by explicit updates of nodal coordinates. Remember that joints in particular are sensitive to even very small initial misalignments!

Change of behaviour in release 9.3.1 to handle multiple assemblies in *INCLUDE_TRANSFORM file

In the original release 9.3 software it was assumed that users would adhere to the rule that "***INCLUDE_TRANSFORM** could only apply to a single assembly" as described above.

However experience showed that some users would import a complete mechanism or dummy (containing multiple assemblies) in a single include file subject to a single transform. This caused problems because every assembly added its own 4 lines of transforms, leading to a ridiculous result.

Therefore from release 9.3.1 onwards a check has been added as follows:

Each assembly is checked for membership of ***INCLUDE_TRANSFORM**s and the extra positioning lines are only added to the associated ***DEFINE_TRANSFORMATION** card if the following criteria are satisfied:

1. The ***INCLUDE_TRANSFORM** file ***must*** contain ***only*** items in that assembly. It may contain the whole assembly, or only a subset of it, but it may not refer to anything else in a different assembly or to unrelated structure not in the mechanism/dummy definition.

2. An assembly may be made up of components from multiple include files (although this would be unusual), and transformations will only be applied to those include files which satisfy rule #1 above.

This means that - for example - a dummy may be imported in an ***INCLUDE_TRANSFORM** file and located in the model using a ***DEFINE_TRANSFORMATION** associated with this. Any subsequent mechanism or dummy positioning will work normally, but extra transformation lines will no longer be added to the ***DEFINE_TRANSFORMATION**, meaning that the nodal coordinates will be updated when that include file is written out.

Extension of the above logic in release 10.0

It turned out that the check added above, which was based only on which nodes were in which include files, was defeated by a case where a modeller placed nodes for all assemblies in include file A, then parts and elements for all assemblies in include file B, and subjected both files to the same transform. Since the test above assumed implicitly that nodes and elements would be in the same include file PRIMER wrongly created multiple transformations for the second file B.

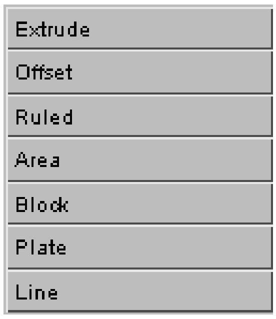
Therefore the check for multiple assemblies in an include file has been extended to look at parts, elements and nodes. This is still bad practice, and the modelling practice described above is recommended for all cases where mechanism and dummy assemblies are placed in include files subject to transformations.

6.22 MESHING

PRIMER has some simple meshing facilities. The options available under the Meshing tool are:

Mesh	Simple meshing facilities such as extrude, offset, ruled surfaces, plates etc
Split	Split shells
Create hole	Create a hole in an existing mesh
Remove hole	Remove a hole from an existing mesh
Remesh area	Remesh an area of existing mesh

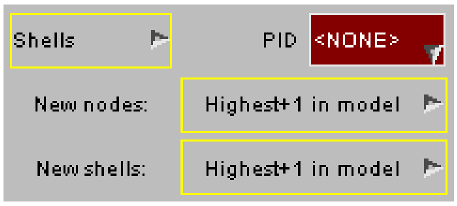
6.22.1 Simple meshing operations



Primer has simple meshing capabilities. Currently you can:

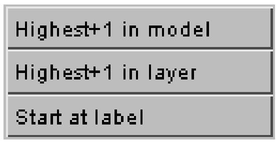
- Extrude nodes to create beams or shells, or extrude shells to create solids.
- Offset shells to create shells or solids.
- Create a ruled mesh between two lines of nodes.
- Mesh a quadrilateral or triangular area by giving the corner nodes.
- Create a cuboidal solid block.
- Create a rectangular shell plate.
- Create a line of beams between 2 nodes.

To change the mode use the popup at the top left of the meshing panel.



For all of the meshing modes you must give a part ID for the new beams/shells/solids to be created in.

Additionally you have control of the numbering that is used for new nodes and new beams/shells/solids.



Use the popup to select which option you require.

If you choose **Start at label** then give a label number to start from. Primer will try to use that number. If a node or beam/shell/solid already exists with that label it will revert to **Highest+1 in model**.

Extrude

Extrude allows you to:

- Extrude nodes to create beams.
- Extrude nodes to create shells.
- Extrude shells to create solids.

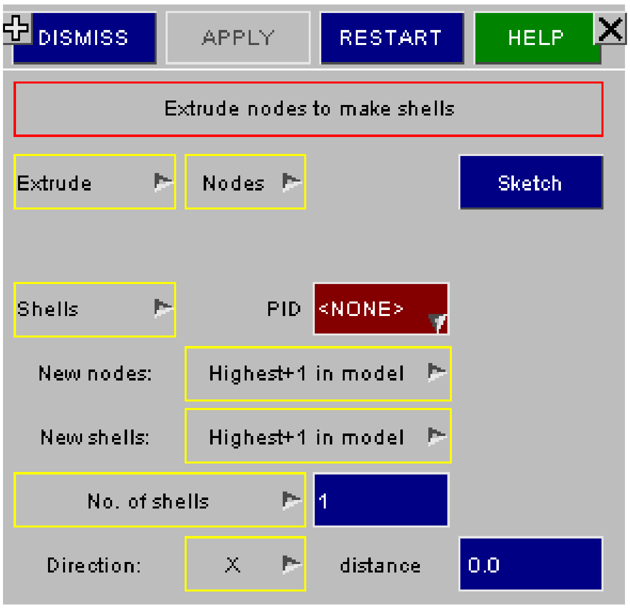
The popups allow to choose the mode.

The number of beams, shells or solids to create in the extrude direction can either be given or you can give the element size, in which case Primer will determine the number required.

The extrude direction can be given by:

- The global X, Y, or Z axes. Give the distance
- A vector given by 2 nodes. Either give a distance or use the length N2-N1.
- A vector given by X, Y and Z components. Either give the distance or use the length of the vector.

When extruding nodes to create shells, if the last node chosen is the same as the first node then the shells created will wrap round.



Offset

Offset allows you to:

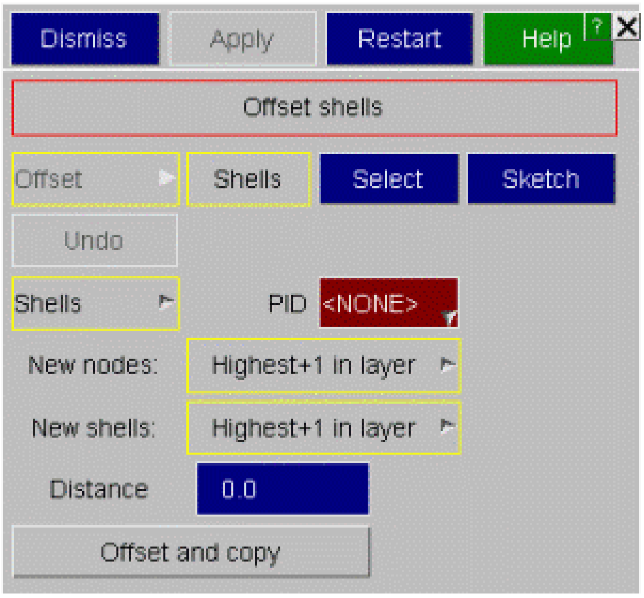
- Move shells or create shells offset a specific distance from existing shells.
- Create solids by offsetting existing shells.

For both options ensure that the normals of the shells you want to offset are consistent, otherwise different directions could be used.

For shells give the distance that the new shell should be offset from the existing shells.

For solids give how many solids should be created while offsetting and the distance to offset.

For shells, to move the original shells rather than create new ones toggle the **Offset and Copy** button to **Offset - no copy**.

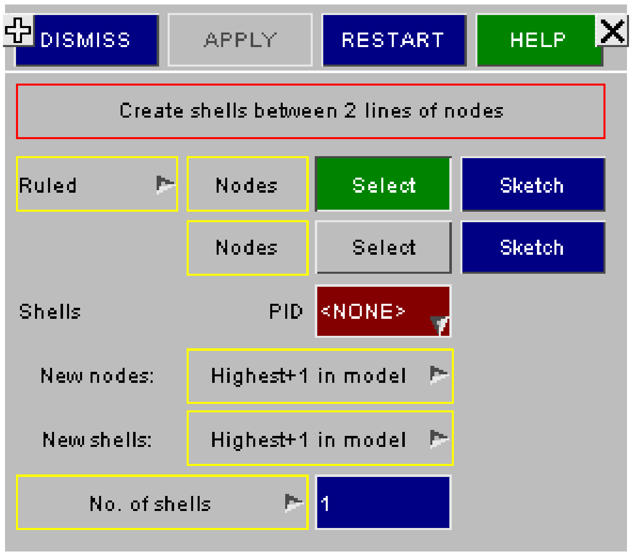


Ruled

Ruled allows you to create a mesh of shells between 2 lines of nodes.

Currently there has to be the same number of nodes in each set of nodes. You can toggle the set that you are picking nodes for by pressing the **Select** button.

Give either the number of shells to create or the size of shells to create, in which case Primer will determine the number required.

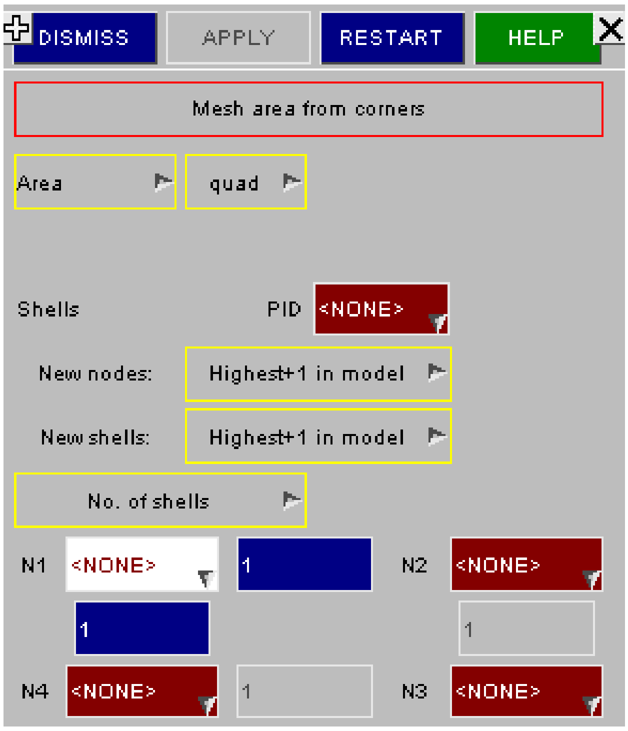


Area

Area allows you to mesh a quadrilateral or triangular area. Use the popup to choose which type to mesh.

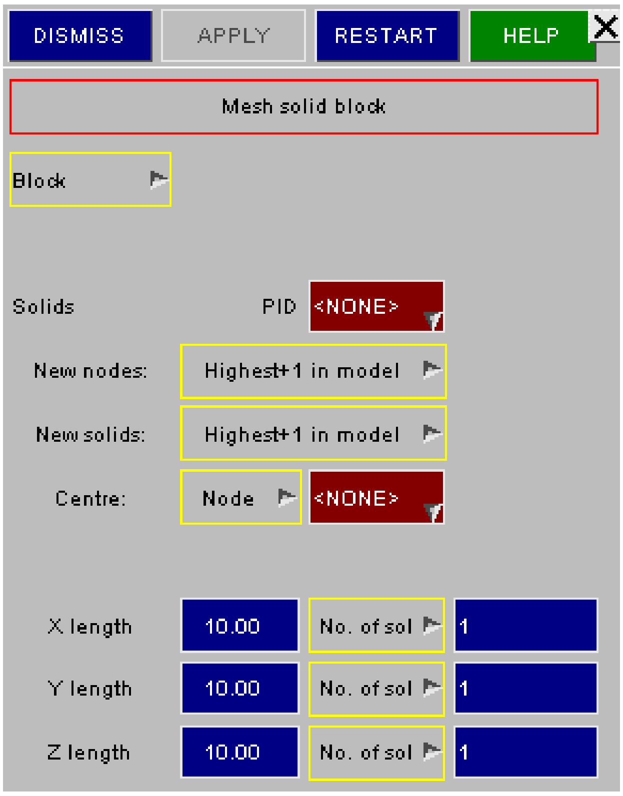
Pick the 3/4 corner nodes for the mesh. Currently, for a quadrilateral mesh the number of shells between N1and N2 must be the same as between N3 and N4, and the number of shells between N1and N4 must be the same as between N2 and N3. For a triangular mesh the same number must be used for all sides.

Give either the number of shells to create or the size of shells to create, in which case Primer will determine the number required.



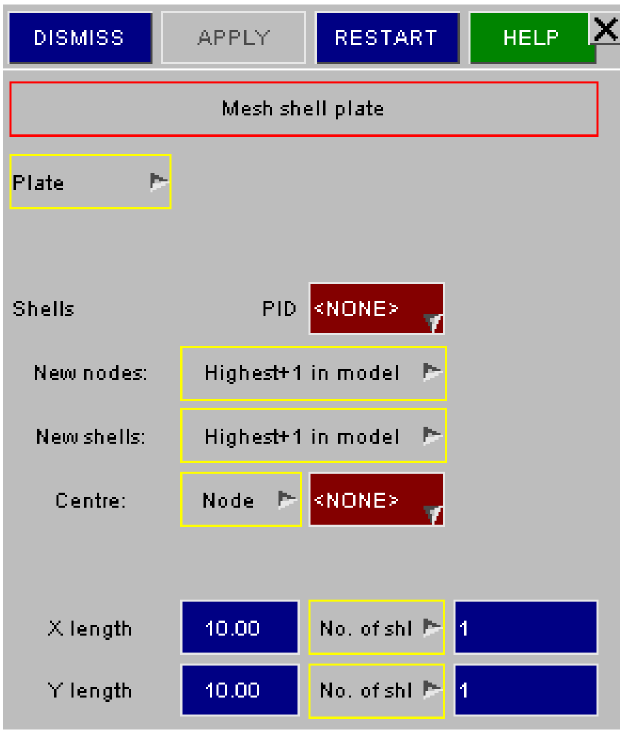
Block

Block allows you to create a simple block of solid elements. Give a node or coordinate for the centre of the block, then for the X, Y and Z directions give the length of the block and either the number of solids or the element size.



Plate

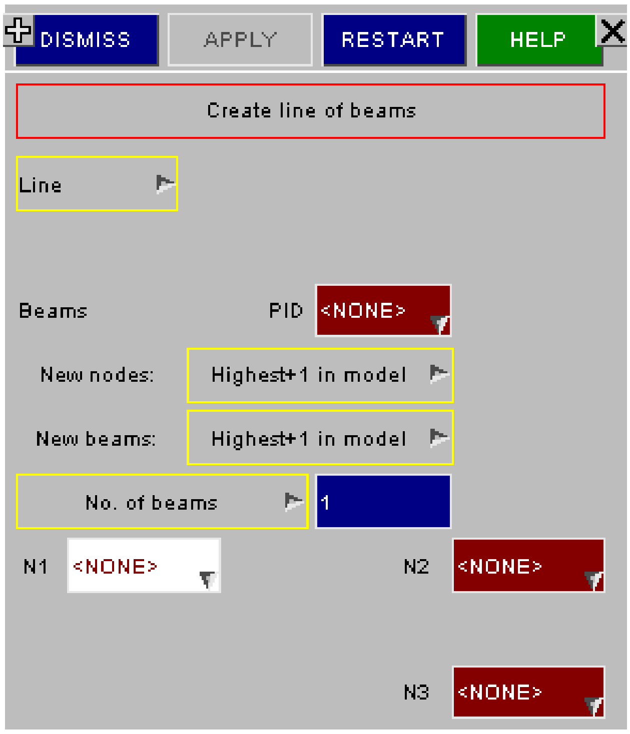
Plate allows you to create a simple plate of shell elements in the xy plane. Give a node or coordinate for the centre of the plate, then for the X and Y directions give the length of the plate and either the number of shells or the shell size.



Line

Line allows you to create a line of beams between N1 and N2. N3 must also be given and will be used as the 3rd node for all of the beams that are created.

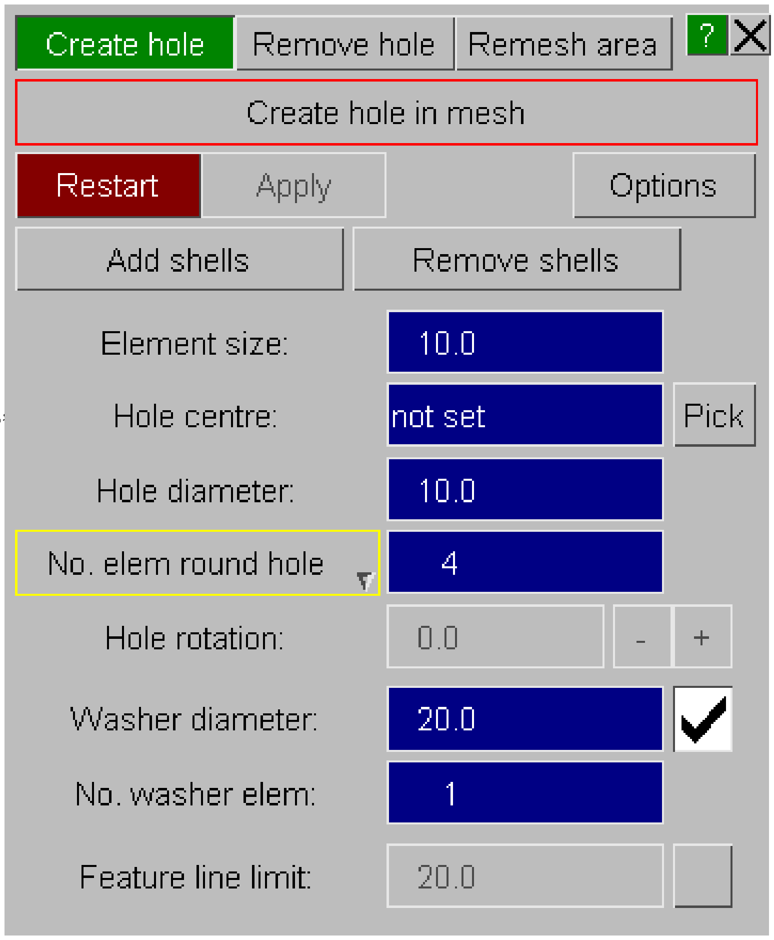
Either give the number of beams to create or the size of a beam, in which case Primer will determine how many to create.



6.22.2 Create hole in mesh

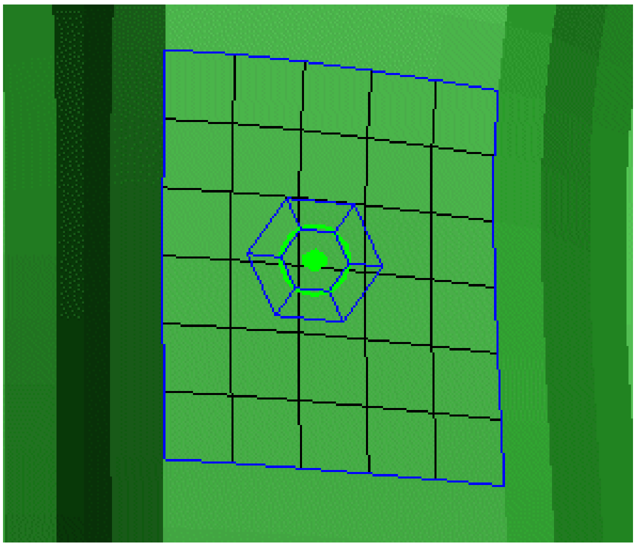
Create hole allows you to make a hole in an existing mesh without any geometry. There are various options to control the mesh and hole created.

Element size allows you to change the size of elements that will be created.
Hole diameter changes the diameter of the hole created.
The number of elements that will be created around the hole can either be specified by using **No. elem round hole** or the size of the elements can be given instead by using the popup and changing to **Elem size round hole**.
The hole can be created with or without ’washer’ elements around the hole. This can be turned on or off by using the checkbox. If it is on then the **Washer diameter** and **No. washer elem** can be used to control how the washer is created.
You can make a hole at a specific coordinate by using the **Hole centre** textbox or you can pick a point on a shell by using the **Pick** button.



Once you have selected the hole location PRIMER will draw a preview of the hole that will be created (here shown with 1 washer element and 6 elements around the hole). The elements which PRIMER proposes to remesh are also sketched. You can now update the hole properties if required and the preview will update. If the hole is made bigger then PRIMER will automatically select more elements to remesh. If you want to manually add or remove elements to the selection you can use **Add shells** and **Remove shells**.

The **Hole rotation** can be used to change the orientation of the elements for the hole. Either use the **+** and **-** buttons or type in a new angle.
Feature line limit can be used to stop automatic selection of shells which are beyond a certain angle. e.g. in this example the shell selection on the left and right of the hole stops at the change in angle.



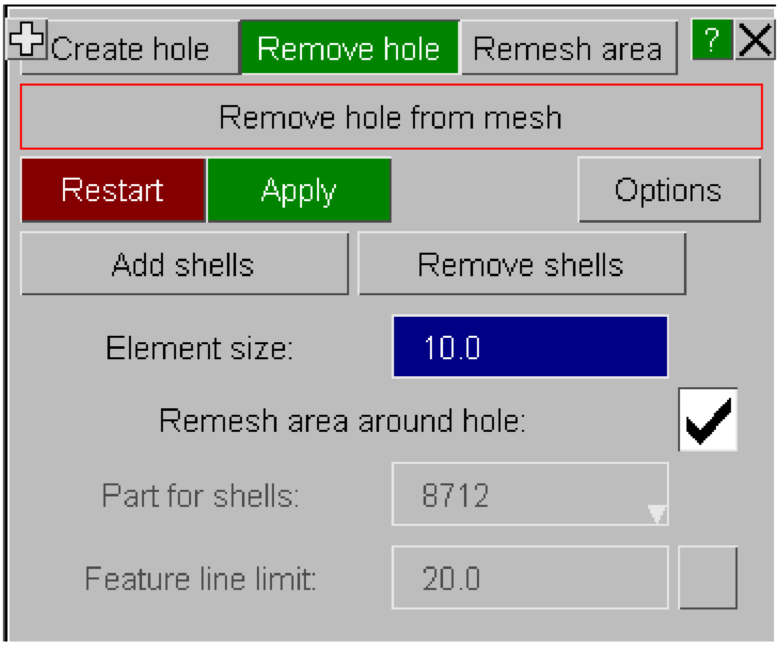
Its meaning is the same as selection in object menus with feature lines.

When you are happy with the hole properties pressing **Apply** will create a preview of the mesh that PRIMER will create. You can still change the hole properties and PRIMER will update the mesh interactively. To actually create the mesh press **Confirm**.

6.22.3 Remove hole from mesh

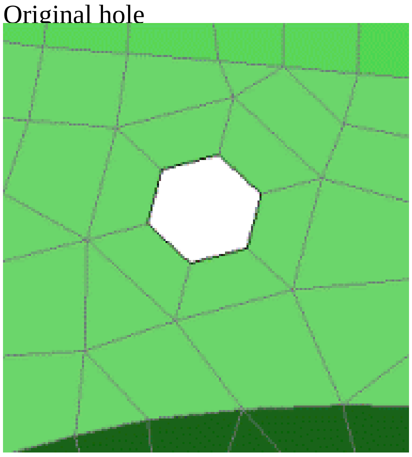
To remove a hole from a mesh pick a shell element next to the hole. PRIMER will then automatically select shells around the hole to remesh similarly to creating a hole. **Add shells** and **Remove shells** can be used to alter the selection.

There are two ways that PRIMERcan remove the hole. It can either just fill in the hole with new elements or it can completely remesh the area around the hole to remove the hole completely. This is controlled by the **Remesh area around hole** checkbox. If selected then the result is shown of the right hand image below. If not selected then by default the shells created in the hole are in the same part as the shells around the hole, but this can be altered with the **Part for shells** textbox.

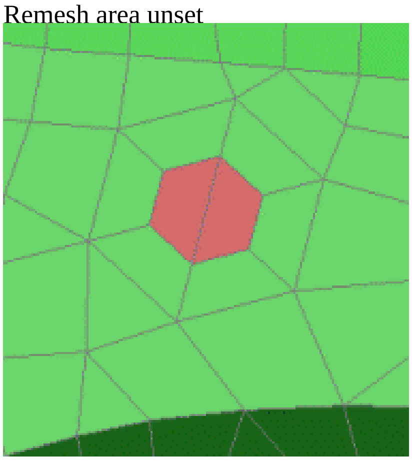


Feature line limit can be used to stop automatic selection of shells which are beyond a certain angle. Its meaning is the same as selection in object menus with feature lines.

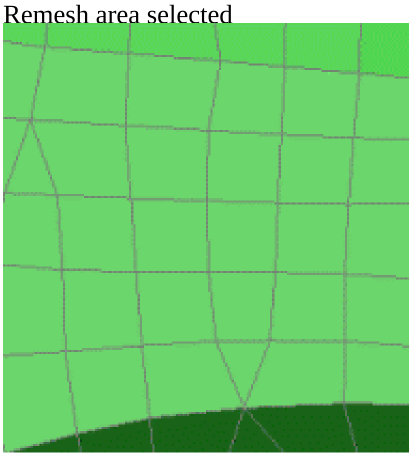
Original hole



Remesh area unset



Remesh area selected

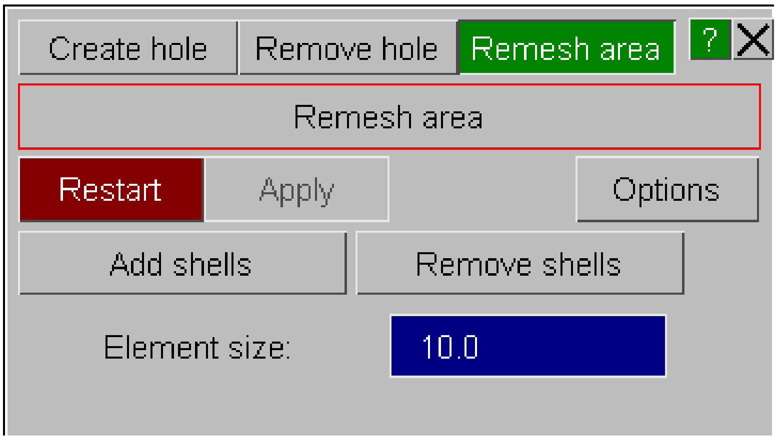


When you are happy with the properties pressing **Apply** will create a preview of the mesh that PRIMER will create. You can still change the properties and PRIMER will update the mesh interactively. To actually create the mesh press **Confirm**.

6.22.4 Remesh area

Remeshing an existing area is possible. The size for new shells is given with the Element size textbox. The shells to remesh are selected with the **Add shells** and **Remove shells** button. They must form a single area without any holes (i.e. there must be a single continuous boundary).

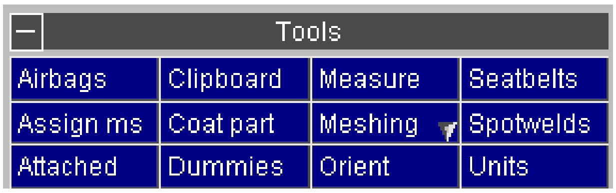
When you are happy with the properties pressing **Apply** will create a preview of the mesh that PRIMER will create. To actually create the mesh press **Confirm**.

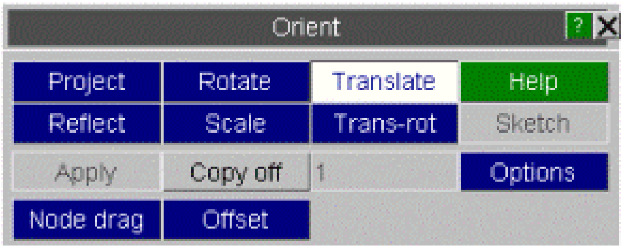


6.22.5 Meshing limitations

The hole creation, removal and area mesh features in PRIMER are new for version 10.0. They are an initial attempt to add some meshing capabilities to PRIMER and are quite limited. The mesh quality may not be optimal. It is expected that the mesh quality and meshing functionality will improve over time with subsequent releases.

6.23 ORIENT Translating, rotating, scaling, reflecting, projecting





The **ORIENT** command is invoked from the MAIN top box, to give the master panel shown in this figure.
There are currently six types of orientation available:

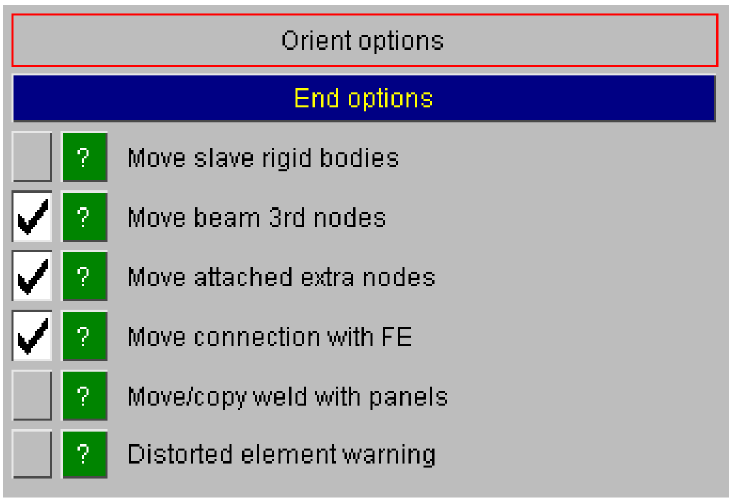
TRANSLATE	shift by global vector, n1->n2, or normal to plane
ROTATE	rotate about global or local axes
REFLECT	reflect about a distance [d] down a given axis.
SCALE	factor nodal and other coordinates by [Sx,Sy,Sz]
PROJECT	project nodes to line or plane or mesh surface
TRANS-ROT	translate and rotate in one operation

There are also links to two other orient related features in PRIMER:

NODE DRAG	drag node interactively and mesh optimisation
OFFSET	offset shells

Normally just the selected items are oriented by the amount specified. It is also possible to **INTERPOLATE** movement to achieve other effects.

Options for Orient



Slave rigid bodies may be implicitly oriented when their master part is. The default is not to orient.

Beam 3rd nodes (which define the beam section) by default will be oriented when the beam itself is.

When a rigid part is oriented, by default the constrained extra nodes will also be oriented. This may distort deformable elements.

When all FE entities of a connection are oriented, by default the connection itself will also be oriented.

There is an option to move (or copy) welds that attach to panels which are being oriented.

There is an option to report that nodes of unselected elements have been moved as a result of the orientation as this may have caused distorted elements.

6.23.1 TRANSLATE Shifting by [dx,dy,dz]

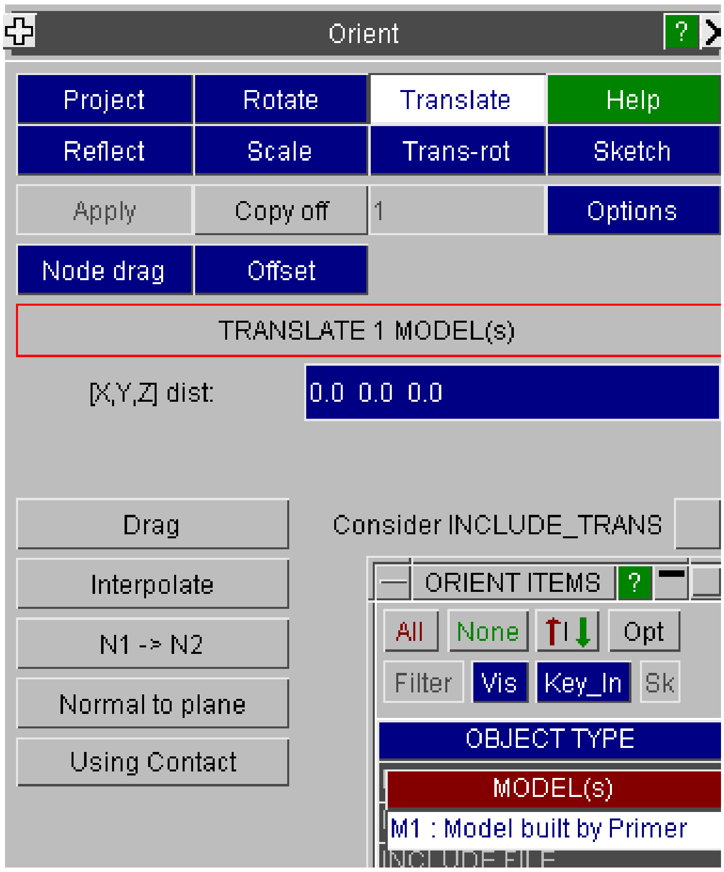
On entering TRANSLATE you must first select the objects to be moved, then

- enter a global translation vector [X, Y, Z]
- define a vector n1->n2 and a distance
- define a plane with 3 nodes (or pick a shell) and a distance for normal projection.

APPLY will update the nodal coordinates.

When a transformation is applied the image is redrawn so that you can see what the result looks like, and you are given the options of accepting, rejecting or repeating the transformation before it becomes permanent.

If you reject an orient, the nodal coordinates are restored from a backup cache, so no rounding error is incurred.



INTERPOLATE is described in section below.

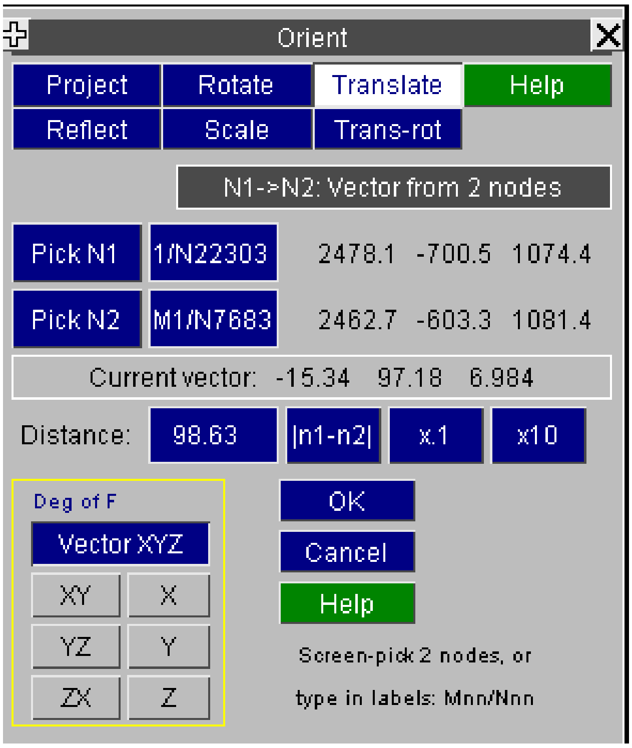
Alternative ways of defining a translation distance.

N1 -> N2 Using the vector between two nodes.

In this method you select two nodes: either by screen-picking them or by typing their labels into the relevant boxes in format M<model number>/N<node label>. The vector is computed from the coordinates of N2 - N1 and the distance set.

You can choose the degrees of freedom of this vector to use. By default **VECTOR_XYZ** is in force, meaning all of the [x,y,z] components, but you can reduce this to two or one component only using XY, ... Z.

When you have obtained the desired vector use OK to return to the main TRANSLATE panel, where you can then **APPLY** it.

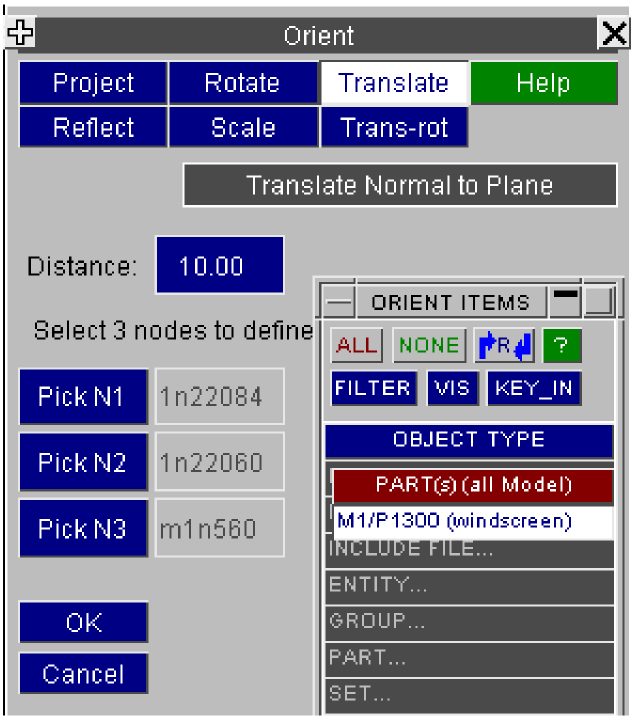


NORMAL TO PLANE

Select the items to translate, and click on the **NORMAL_TO_PLANE** option.

Define the plane by picking on 3 nodes and set the translation distance.

Use **OK** to return to the main TRANSLATE panel, where you can then **APPLY** the orient.



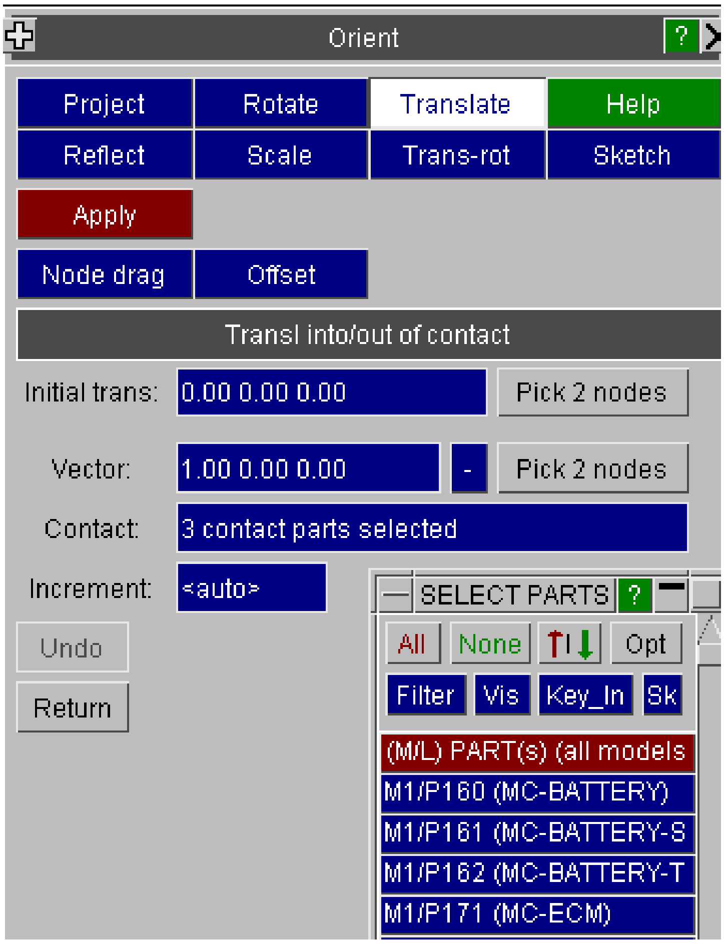
Translate Using Contact

With this function the selected items (typically an impactor or barrier model) may be translated along a defined slide vector until they are brought into position or depenetrated (if initially penetrating).

Contact part(s) on the main model (typically the vehicle) must be defined. The orient items and the part do **not** need to be in the same model as this function has special logic to create contacts across models.

If initial penetration is detected, items will be moved *against the direction of slide vector* until depenetrated. If not, they will be moved *in the direction of the slide vector*.

The increment for each iteration may be set by the user, although the automatic method should work for most models.

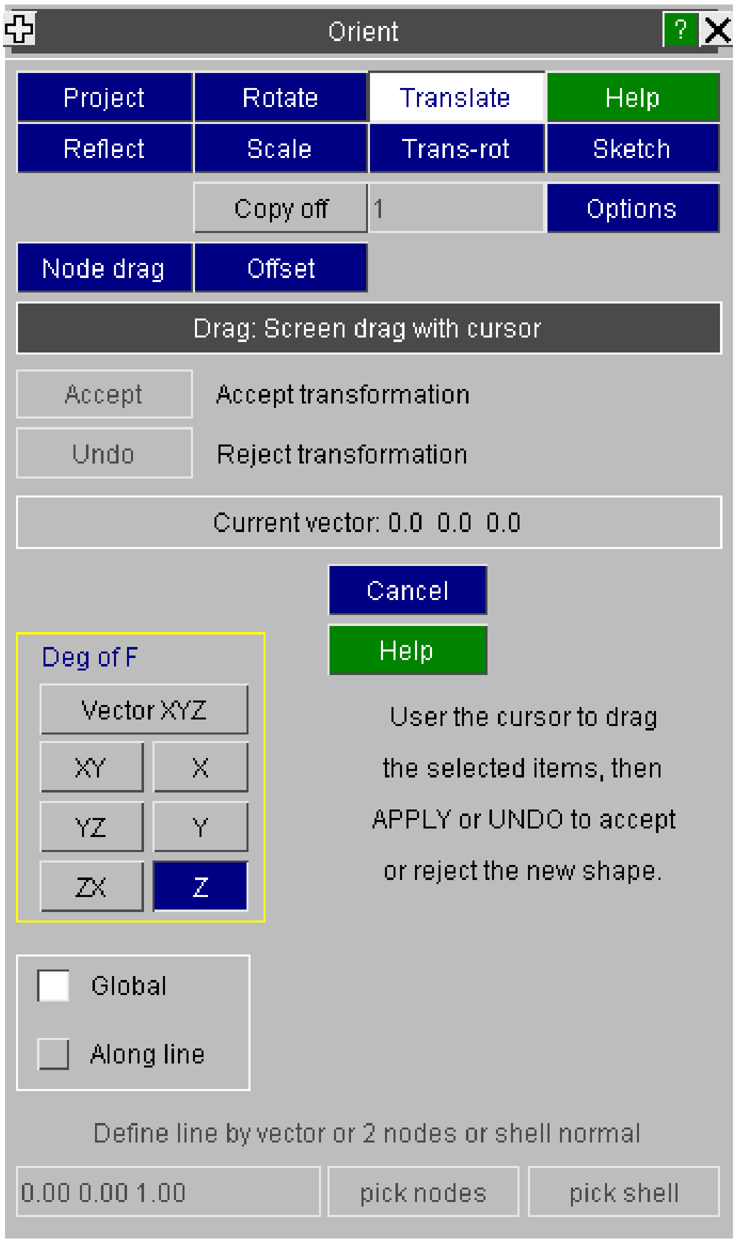


DRAG TRANSLATE Using the cursor to "drag" objects.

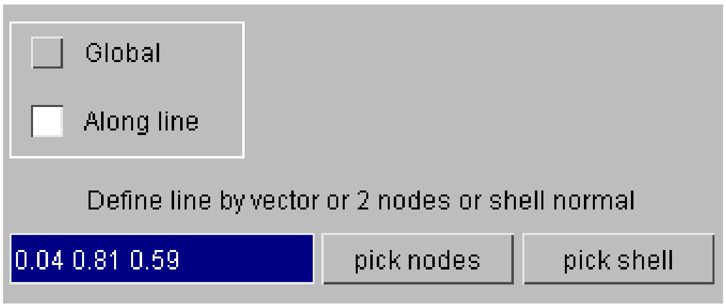
Click down the left mouse button at any point on the screen (it's not related to the object) and drag it in the desired direction. The object, as a reduced set of vectors if it is large, will follow the mouse across the screen, stopping when you release the mouse button.

Then use **APPLY** to accept the transformation, or **UNDO** to reject it and restore the status quo ante.

Drags take place in the plane of the screen, so the actual [x,y,z] vector will depend on the current view. It is strongly recommended that you use one of the XY ... Z options to limit object motion to either a plane or a single vector.



The motion may be limited to an arbitrary vector by switching to **Along line** mode. The vector may be typed in, defined by 2 node picks or the normal of a picked shell.



On completion of drag, the new position may be accepted or rejected.

6.23.2 ROTATE Rotating by x,y,z

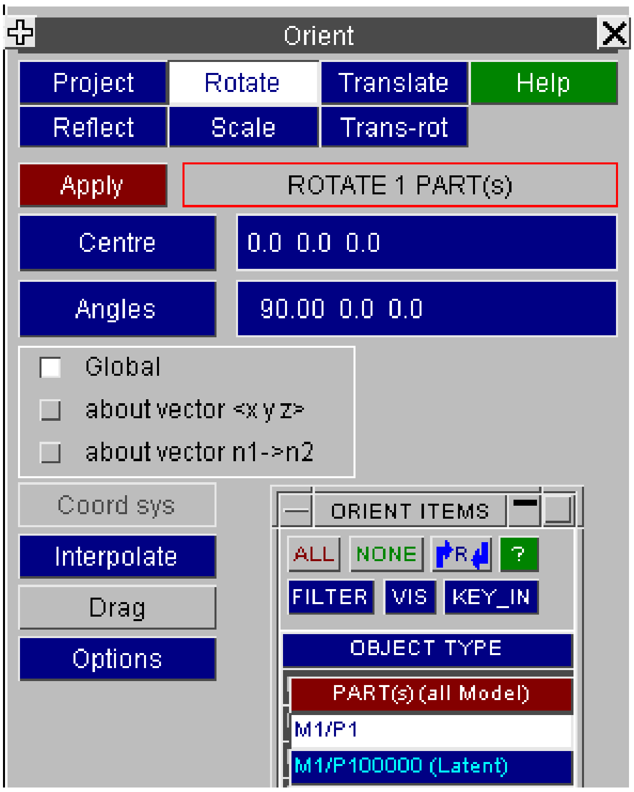
On entering ROTATE you must define:

- The objects to be rotated;
- The centre about which to rotate;
- either the rotation angles (**GLOBAL**_option) or a single angle and a local axis (**about_vector**_option)

Local axis of rotation may be defined by a vector <x y z> or two nodes.

When these have been entered press **APPLY** to make them take effect.

As with TRANSLATE when the rotation is applied the image is redrawn so that you can see what the result looks like, and you are given the options of accepting, rejecting or repeating the transformation before it becomes permanent.



CENTRE: Defining the rotation centre

Instead of typing in an [x,y,z] coordinate you can use **CENTRE** to pick a node to be used as the rotation centre. The node’s coordinate will be placed in the "centre" box for you.

For rotate and for scale, if a single part is selected the useful option **centre @ part CofG** will become active.

ANGLES: Defining rotation angles

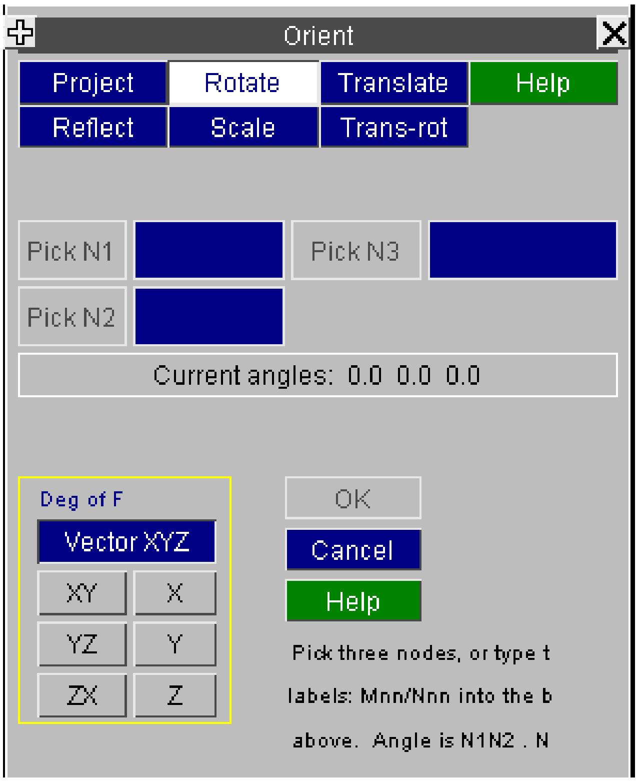
As an alternative to typing in angles about the [x,y,z] axes you can use the **ANGLES** command to calculate the angle between 3 nodes, as shown in this figure.

Pick, or type in the labels of, 3 nodes.

The vectors N1N2 and N1N3 are computed, and then the angle between them.

You can choose the 3D angle, or the projected value about any axes using **XY, ...Z** as before.

When the angle is satisfactory use **OK** to return to the main ROTATE box where you can then apply it.



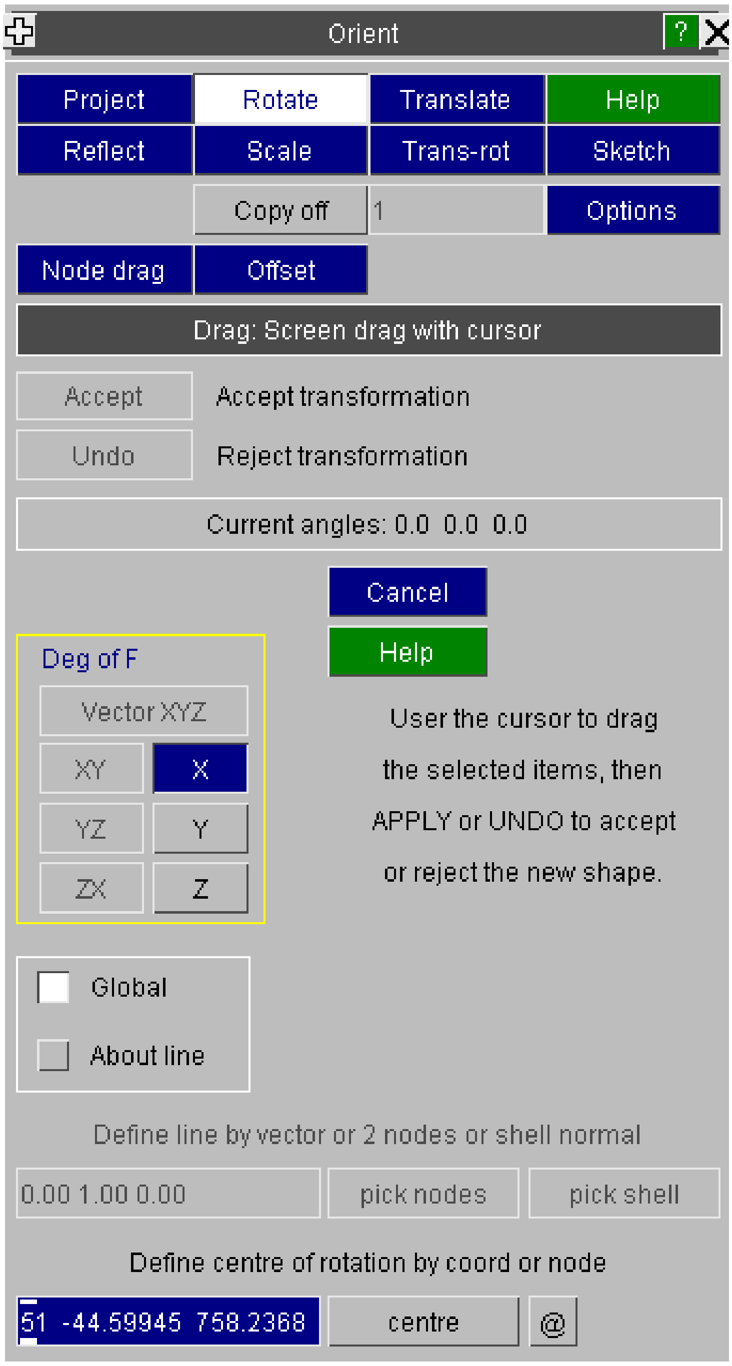
DRAG ROTATE Using the cursor

As an alternative to specifying rotation angles you can "drag" the objects, about the defined centre, using the **DRAG** option shown in this figure.

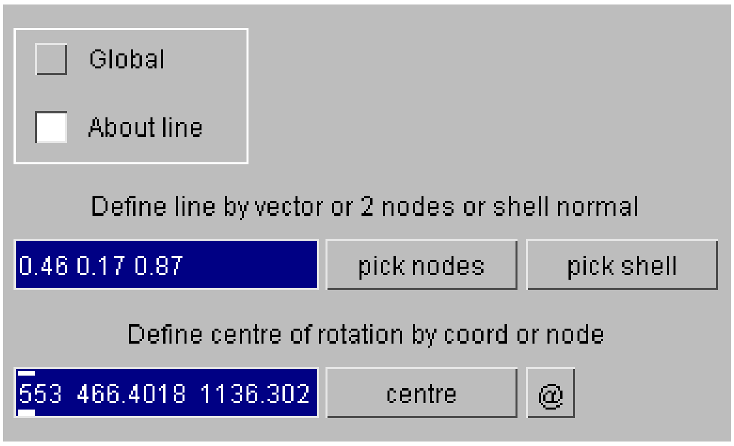
Place the cursor anywhere on the graphics screen, press the left mouse button and, keeping it depressed, move it until the desired position is reached, then release it. The image (or a subset of it if it is large) will move across the screen, and then be redrawn at the new position.

Use **APPLY** to make the change permanent, or **UNDO** to reject it and return it to how it was before. (Using **CANCEL** also implicitly rejects any dragged rotations.)

Dragging can only take place about one axis at a time, the default being global **X** as shown here. The centre of rotation may be typed in or set by a node pick. The **@** button will sketch the current centre.



The **About line** option will allow a rotational drag about an arbitrary line.



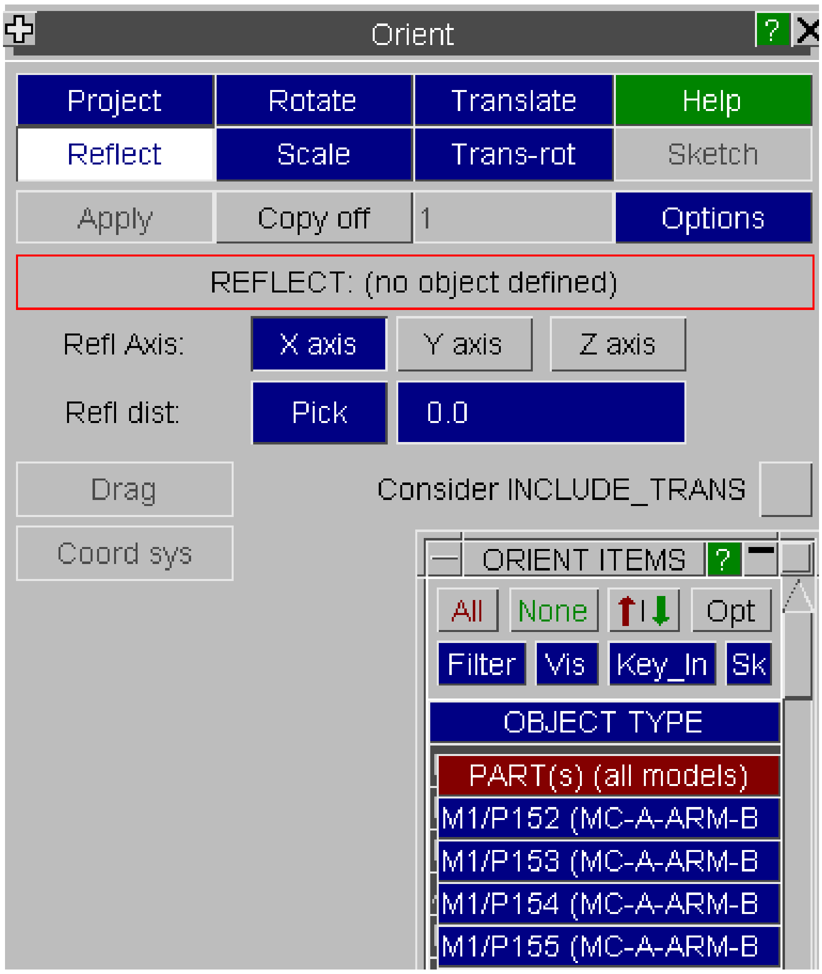
6.23.3 REFLECT: Reflect about an axis.

Reflections take place about one of the global X, Y or Z axes, about a plane at a specified distance down that axis.

To use it:

- Select the objects to reflect.
- Pick an axis: X Y or Z
- Define a distance, or use **PICK** to use a nodal coordinate to define the reflection plane distance.
- Use **APPLY** to make the reflection happen.

As before the image is drawn showing the new configuration, and you can accept, reject or repeat the transformation.



Notes on REFLECT:

- At present reflection may only take place about global axes.
- ***IMPORTANT***: A reflection is ***NOT*** the same as a rotation by 180 degrees!

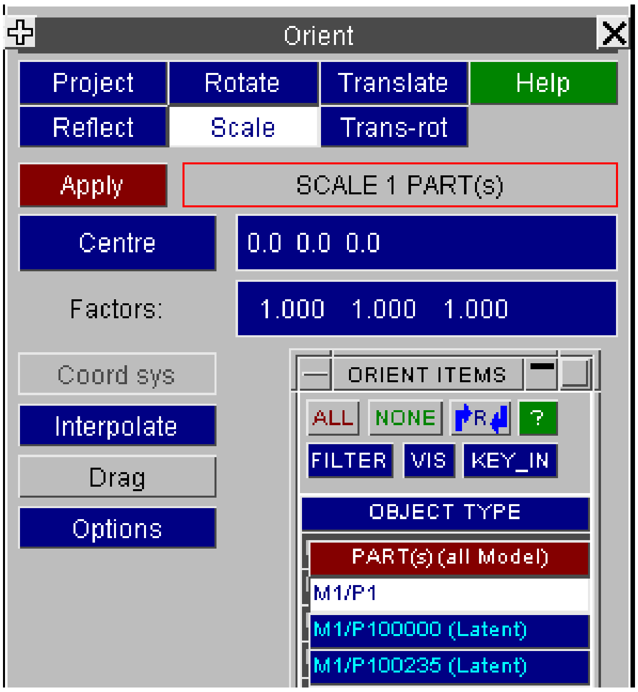
  Reflection not only moves coordinates, but also reverses the topology ordering of elements with 3 or more nodes so as to preserve their local axis systems (and +ve volume in the case of 3D elements). Whereas rotation by 180 degrees just moves the coordinates. So although the results may look similar they can have different properties.
- ***NEGATIVE SCALE****:* Primer treats negative scale (SCX * SCY * SCZ < 0) as reflection i.e. the topology of elements **is reversed**. This change was necessitated by the fact that *INCLUDE_TRANSFORM can only encode a reflection as a -ve scale and LS-Dyna does reverse the topology in this case (otherwise it would not work with 3d elements).

6.23.4 SCALE Scale by [Sx,Sy,Sz]

You can scale objects by independent factors about each of the [x,y,z] axes. To do this:
- Select objects as before;
- Define the coordinate to scale about.
- Define the factors for each of the [x,y,z] axes.
- Use **APPLY** to make it happen.

As before the image is drawn showing the new configuration, and you can accept, reject or repeat the transformation.

Negative factors are allowed, see the notes above on **REFLECT**.



CENTRE Defining a central coordinate to scale about.

Instead of typing in a coordinate you can use **CENTRE** to define a node whose coordinate will be used as the centre of scaling.

For rotate and for scale, if a single part is selected the useful option **centre @ part CofG** will become active.

What is and is not scaled

Scale factors can be different in each of [X,Y,Z] directions, which implicitly ties scaling to the global coordinate system and restricts it to spatial and directional quantities. There is no ambiguity where coordinates and vectors are to be scaled, but the issue of scalar length values is more difficult because they do not usually have an orientation.

For example scaling the thickness values T1 - T4 on a *SECTION_SHELL card would not be appropriate.

Therefore the general rule in PRIMER is that:

- Coordinates and vectors expressed in the global system ***are*** scaled
- Single ("scalar") length values are ***not*** scaled, however there are some exceptions:
    - The "finite" lengths on *RIGIDWALL cards (lenl, lenm, etc) ***are*** scaled. See Section 5, Rigidwalls for details.

DRAG: Dragging with the cursor.

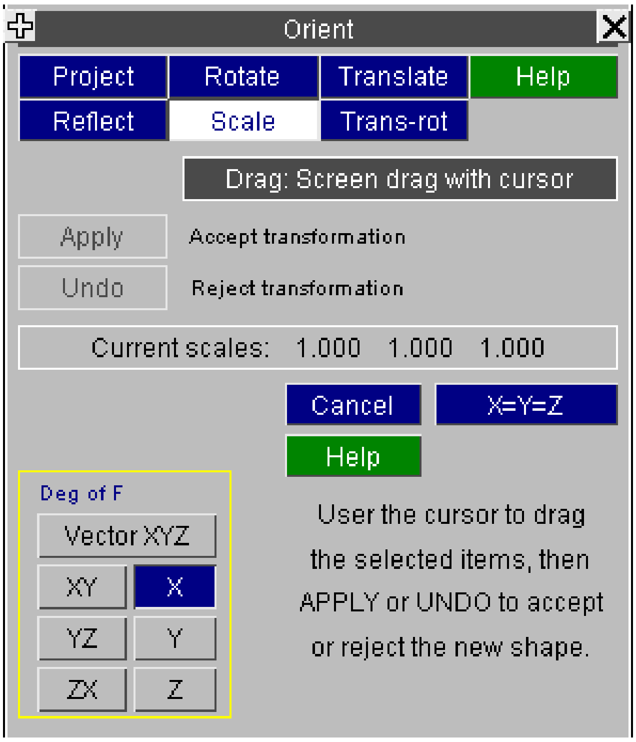
As an alternative to specifying scale factors you can "drag" the objects' scales, with respect to the defined centre, using the **DRAG** option shown in this figure.

Place the cursor anywhere on the graphics screen, press the left mouse button and, keeping it depressed, move it until the desired size is reached, then release it. The image (or a subset of it if it is large) will expand or contract on the screen, and then be redrawn at the new position.

Use **APPLY** to make the change permanent, or **UNDO** to reject it and return it to how it was before. (Using **CANCEL** also implicitly rejects any dragged rotations.)

Dragged scaling can take place using any combination of **XYZ ... Z** axis factors, and by default the factors about each axis are computed independently from the cursor movement in that axis as projected from the current view.

This tends to give unsymmetrical scaling if more than one axis is in use, so you can "clamp" all the factors to have the same values using the **X=Y=Z** button: particularly useful when combined with **VECTOR_XYZ** to expand or contract by a uniform amount in all three directions.

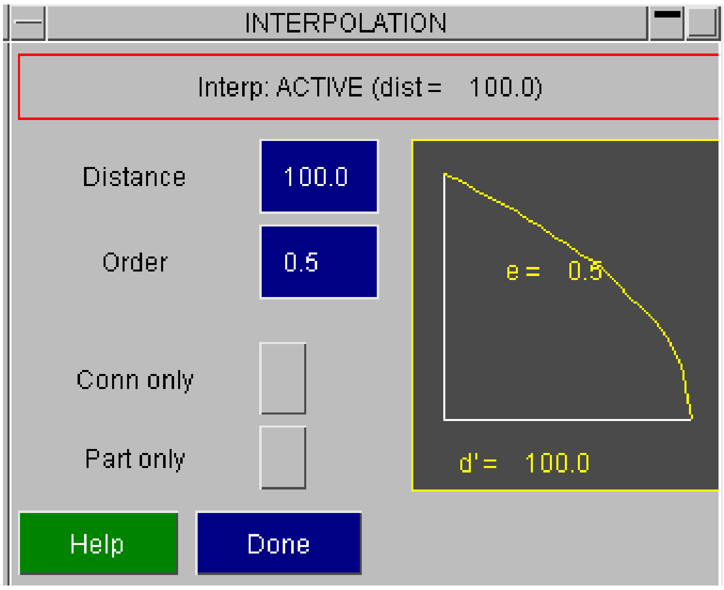


6.23.5 The **Interpolate** command.

Applying interpolation to **ORIENT** functions.

The **TRANSLATE, ROTATE** and **SCALE** functions all act by default only on the chosen items. However if **INTERPOLATE** is used the effect can be spread over a wider area by applying "interpolated" values to adjacent nodes

This is made active by setting a **Distance** value (in this example 100) in the interpolate box. This value remains current until changed, setting this value to 0 turns interpolation off again.



The **Interpolate** button is shown in red when it is active.

Once a **Distance** value has been set the coordinates of unselected nodes within a radius <**Distance**> of any explicitly selected nodes will have their coordinates updated as follows:

$$C_{new} = C_{old} + \delta * \left(\frac{d}{D}\right)^{Order}$$

$$\text{Where } \left(\frac{d}{D}\right) > 0.0$$

Where: **C** = Coordinate of this node

δ = Coordinate change due to Translation / Rotation / Scale

d = Distance from this node to nearest explicitly selected node

D = The specified Distance value

Order = The specified Order value

The **Order** value defaults to 1.0, giving a linear interpolation, but any positive value > 0.001 is permissible, and a sketch of the factor vs. distance is shown in the box. (In the figure above 0.5 has been used.)

Conn only Restricting movement to "connected" nodes only

The **Conn only** switch limits nodes that are eligible for transformation by **INTERPOLATE** still further: if it is switched on only those nodes which are connected via element mesh to explicitly chosen nodes, (as well as being within Distance), are eligible for movement.

"Connected" in this context means that it is possible to get from the node in question to any explicitly selected node via a continuous mesh of structural elements. The connection path does not have to be direct, PRIMER will follow mesh of any complexity, but there must not be any breaks to cross.

This is intended for use within very crowded areas of mesh where a purely geometrical selection of nodes for movement could lead to undesirable results by including unrelated items.

Part only Restricting movement to nodes of parts

In this mode, Primer will identify parts of which nodes have been selected for orient and only allow interpolation on nodes of these parts.

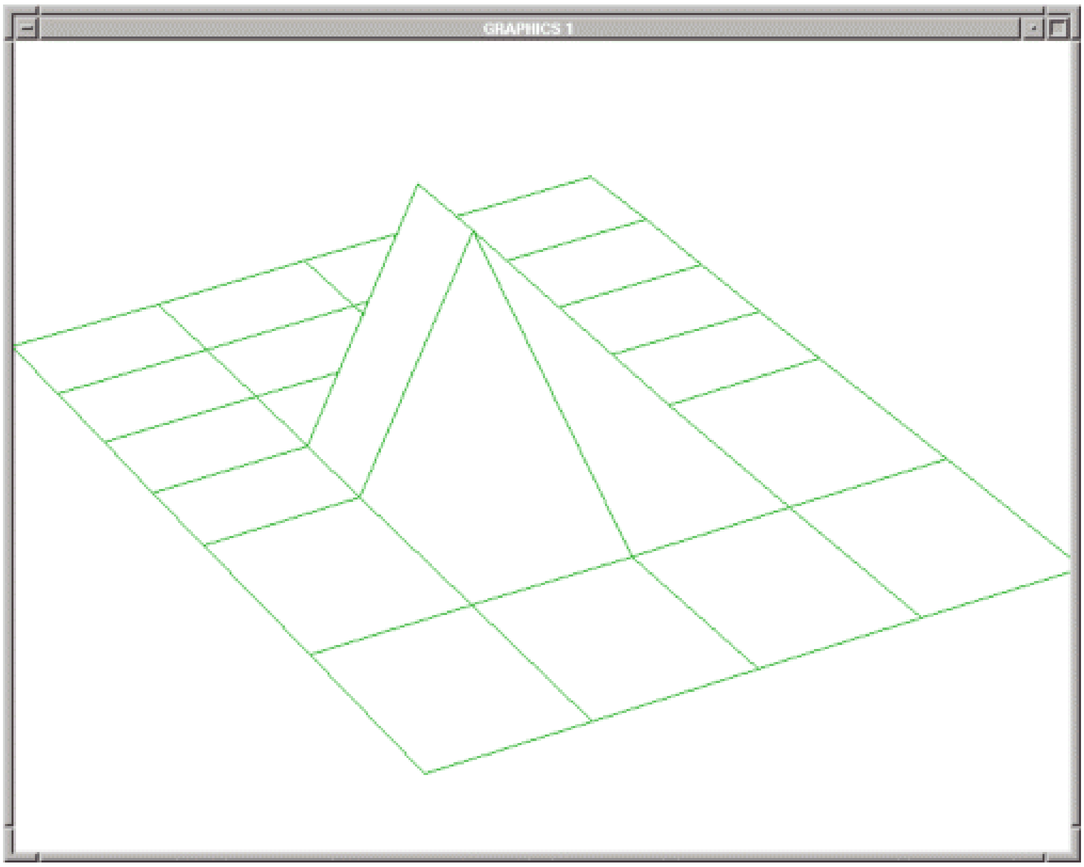
Warning about speed penalties

Using **INTERPOLATE** can slow down transformations significantly, particularly if large **Distance** values are used. This is because a bucket sort for the "nearest explicitly selected node" is required for every candidate node within **Distance** of selected nodes, and the number of nodes in the sort increases as the cube of **Distance**.

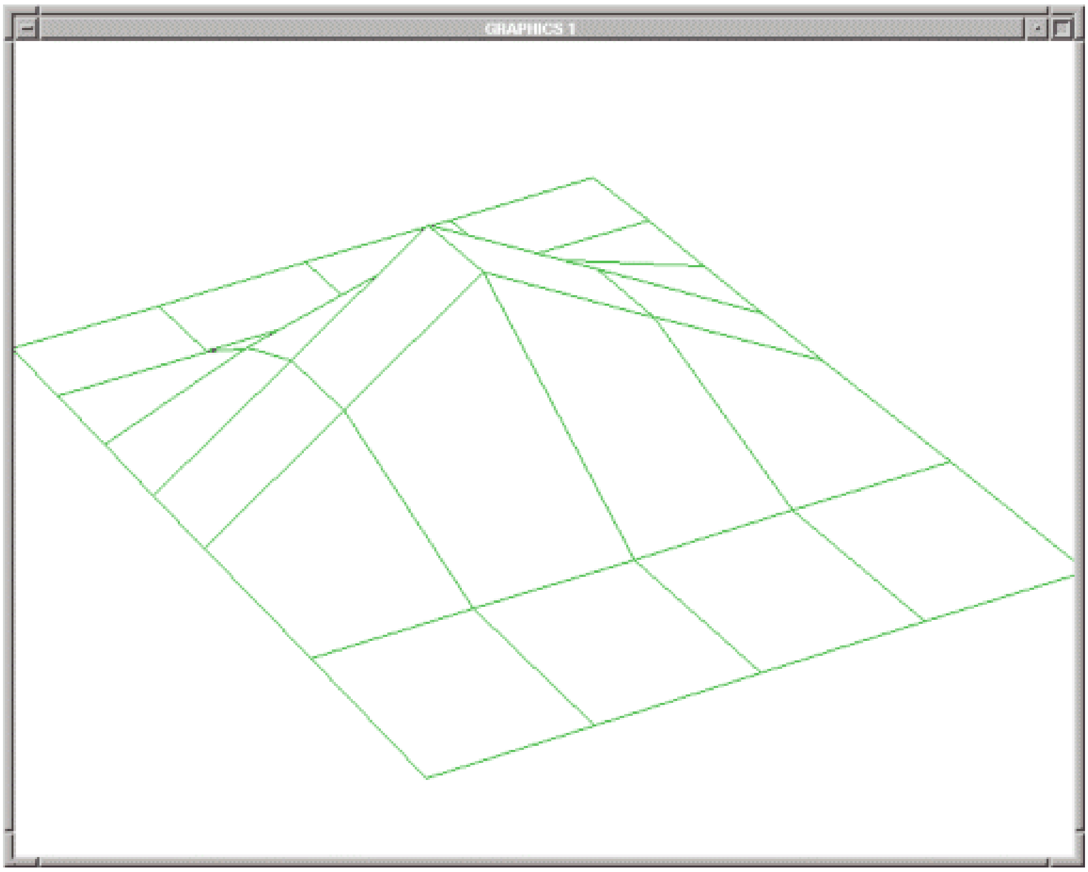
So don't be surprised if there is a significant delay when interpolation is used with large **Distance** values in big models. Using **Conn only** can speed matters up as it usually reduces the number of candidate nodes for movement.

Example use of INTERPOLATE:

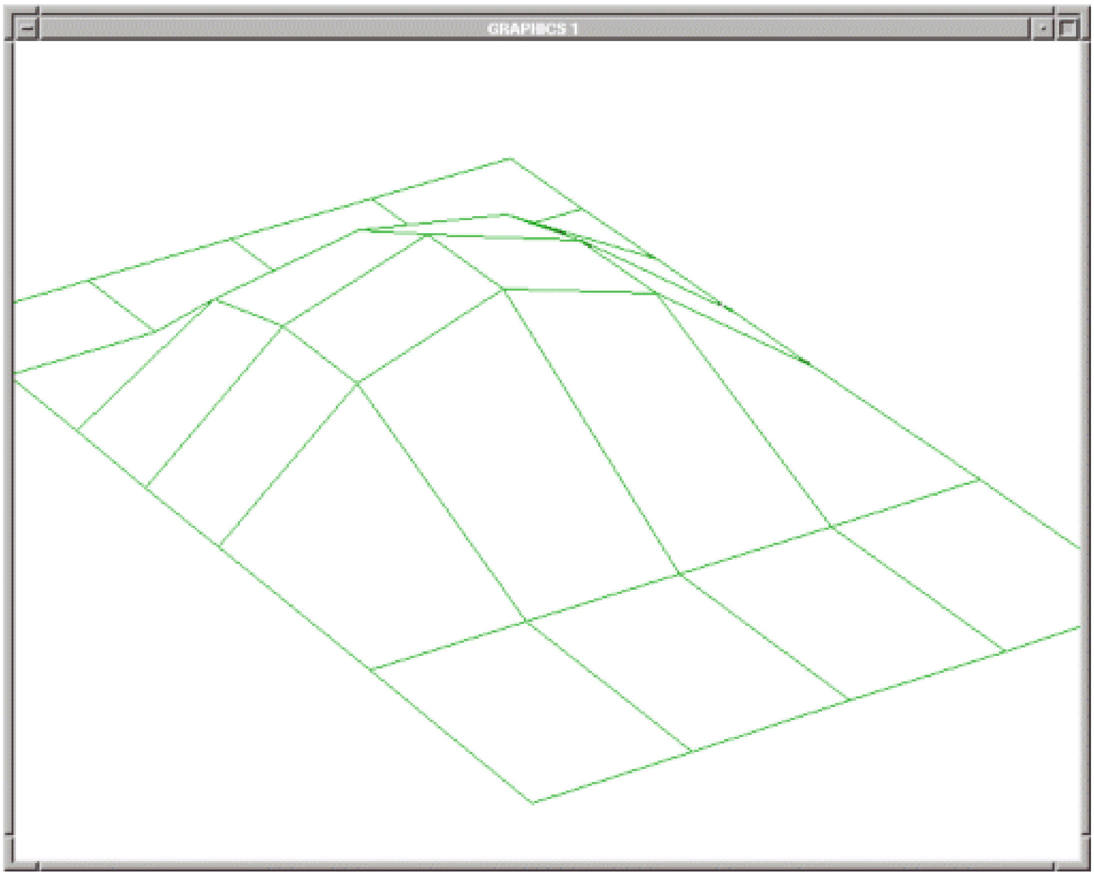
Distance = 0
In this figure two nodes in the centre of a flat plate have been raised, with no **INTERPOLATE** value set.
It is clear that only they have moved, and adjacent nodes are unaffected.



Distance = 20, **Order** = 1.0
In this figure the same nodes have been moved, but now **INTERPOLATE** has been switched on. The **Distance** chosen is equal to half the smaller mesh dimension.
Here **Order** = 1.0, so there is a linear interpolation between the selected nodes and the edge of the mesh.



Distance = 20, **Order** = 0.5
In this final figure the **Order** value has been reset to 0.5, giving a curved variation from centre to edge of the mesh. This shows how a non-linear effect can be achieved.



6.23.6. PROJECT: project to a line, plane, or surface

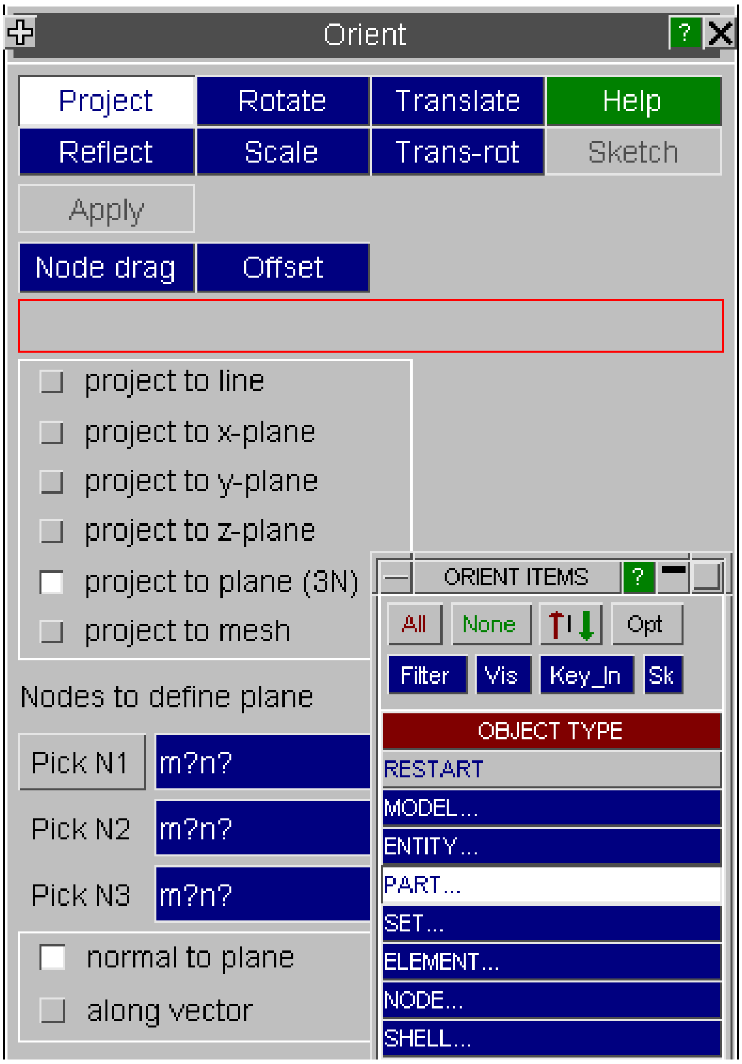
The projection option operates on the nodes of the items selected through the object menu.

The project-to-line operation will move them to the nearest point on the defined line.

Projection-to-plane can use either a global plane, defined by a single coordinate or a node pick, or an arbitrary plane defined by 3 nodes. Projection may be done normal to the plane or along a vector.

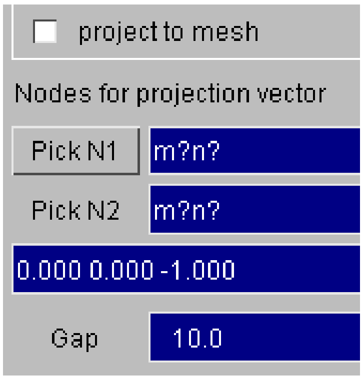
Projection-to-mesh requires a direction vector and the mesh can be defined using shells, shell sets, or shell parts.

To apply the orient press **APPLY**. You will then have the option to **UNDO_ALL**, if the orient is not as you wanted it.

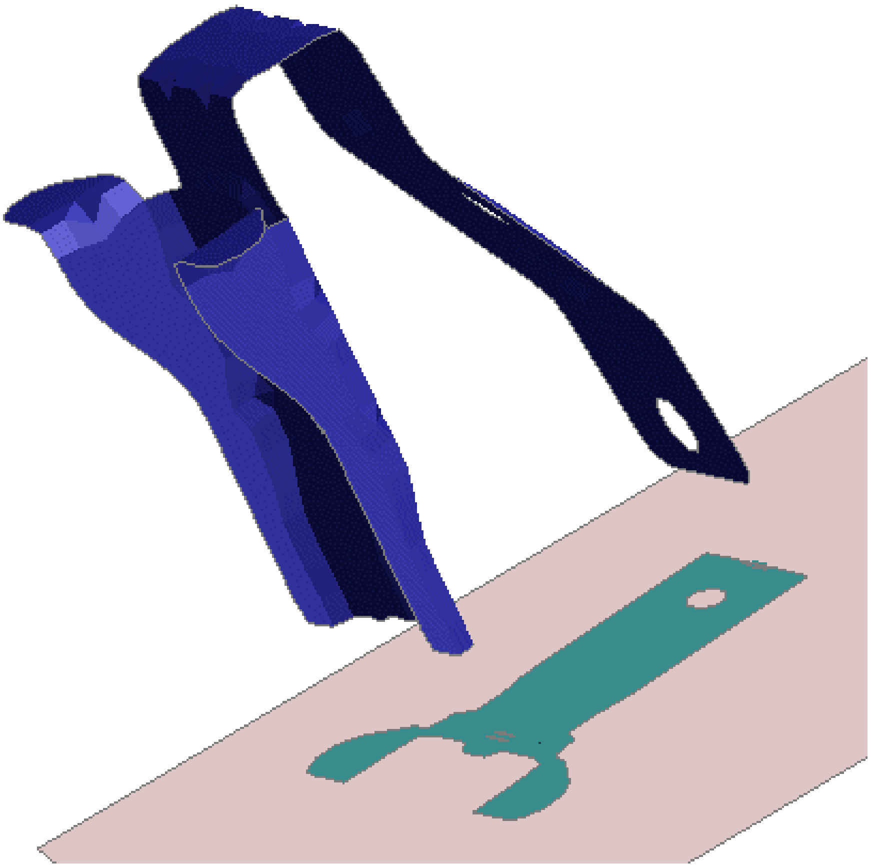


Projecting across models

Project to mesh can be used to project across models, i.e. the projected items and the mesh to which they project do not need to be in the same model.



- a gap value may be set to offset the projection
- COPY may be used with project



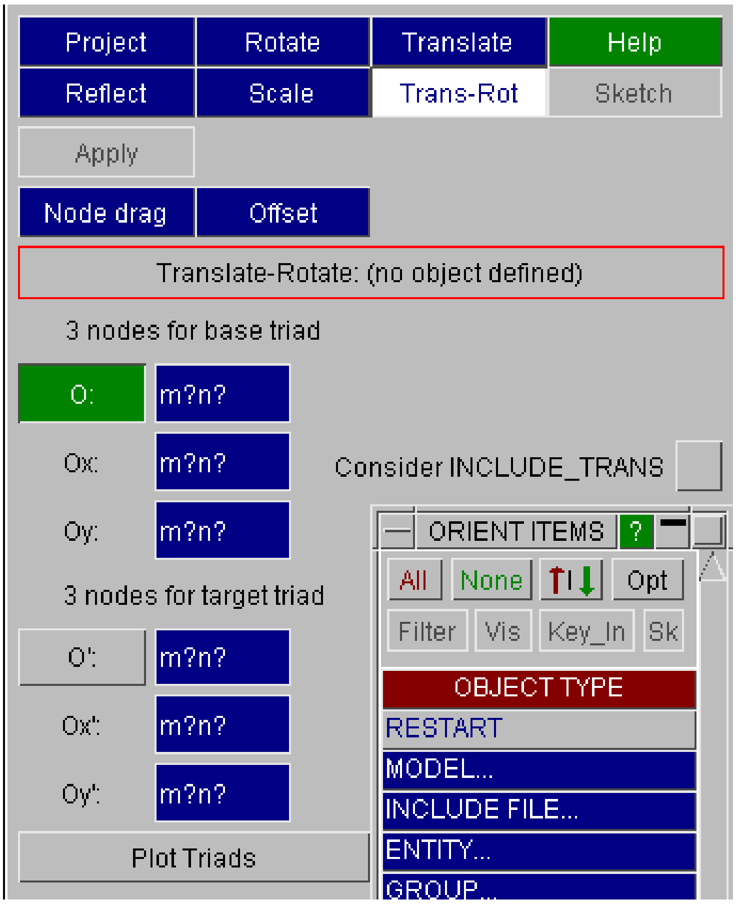
6.23.7. TRANS-ROT: translate and rotate

The Translate-rotate function requires the user to define a base triad and a target triad.

Each is defined by an origin node (O) a second node prescribing the X vector (Ox) and a third node lying in the XY plane (Oy). The appropriate Y and Z vectors are then found.

A translation vector of base origin node to target origin node and a set of rotations (base triad to target triad) have now been established.

This translation and rotation will then be applied to the select items.

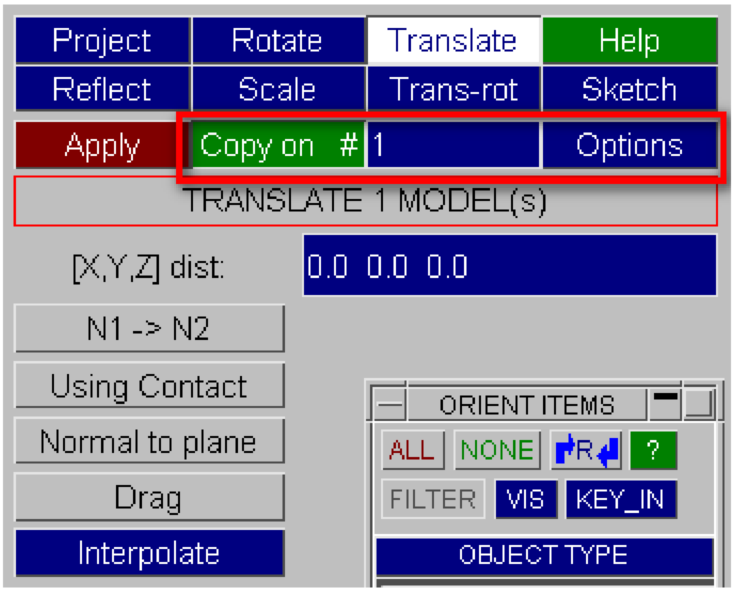


6.23.8. Copy and Orient

The **COPY ON** option is available for REFLECT, ROTATE, TRANSLATE, SCALE and PROJECT.

The copy function is **not** available for translate by **Contact Orient** or for **Trans-Rot** (triad to triad orientation).

The copy can be applied once or multiple times, the orient being incremented each time. The initial orient may be defined by a DRAG operation or explicitly. The copy function can be turned on at the top of the orient panel. The copy options can be opened by clicking on **Options**.

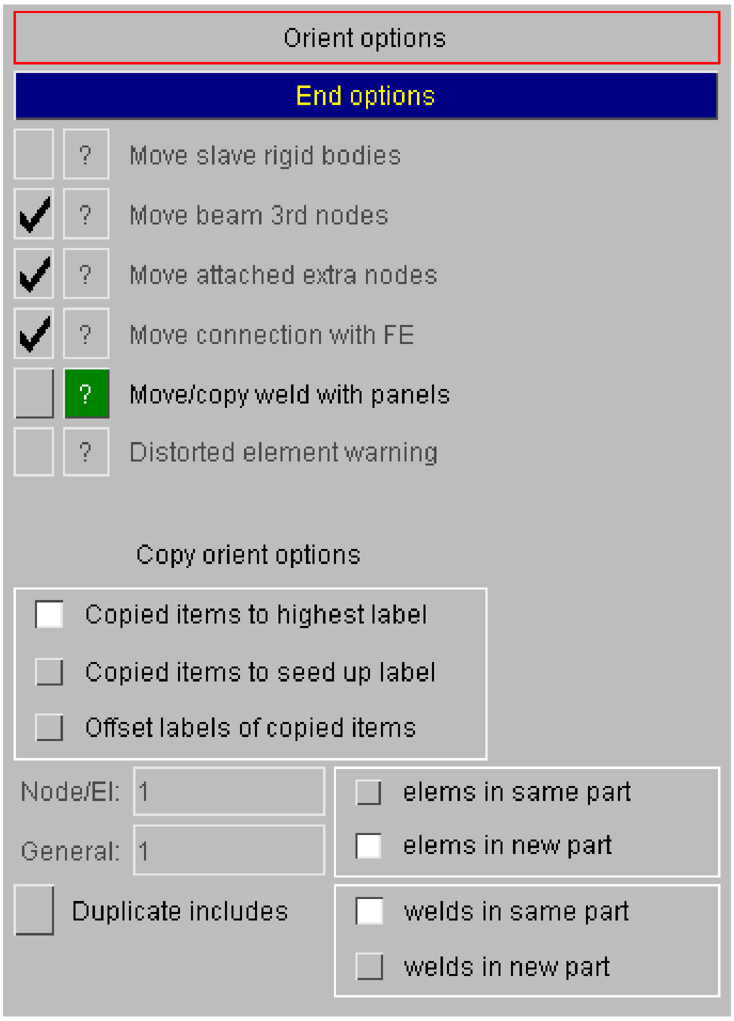


When you select **COPY ON** the orient option are temporarily pre-configured as follows:

- Move slave rigid body OFF
- Move beam 3rd node O
- Move attached extra nodes ON
- Move connection entities ON

Move/copy welds with panel is available for user to set.

When copy is deselected (or the orient panel dismissed) Primer will restore the settings to their previous value.



Labels for new items By default the new items will be labelled starting with the highest current label + 1 for each item type. Alternately, the user may specify a pair of seed labels (this or the next available label will be used) or a pair of offsets. One label is for the more populous type of item (nodes, elements, node sets, nrbs), the other for all other types. In the offset case, Primer will check that all the offset labels are available.

If include label ranges are defined for the model, Primer will always try to correct any out of range labels once the orient operation is completed.

Part for new items For copied element the user nay choose to
- **elems in same part** put copied elements in the original part
- **elems in new part** put copied elements into a newly created part

If the user selects **elems in new part**, items referenced by the original part (e.g. material, section) will also be copied also. If the part is referenced directly by something (e.g. Boundary Prescribed Motion Rigid or contact by part) this card will also be copied.

Part for welds If the options **copy welds with panels** and **elems in new part** are set, by default the option is to keep welds in their original part. If you want a new part for the welds you may switch from **welds in same part** to **welds in new part.**

Duplicate includes If this is switch is active, when items in include files are copied a duplicate include is created (named to xxx_1, etc) for them.

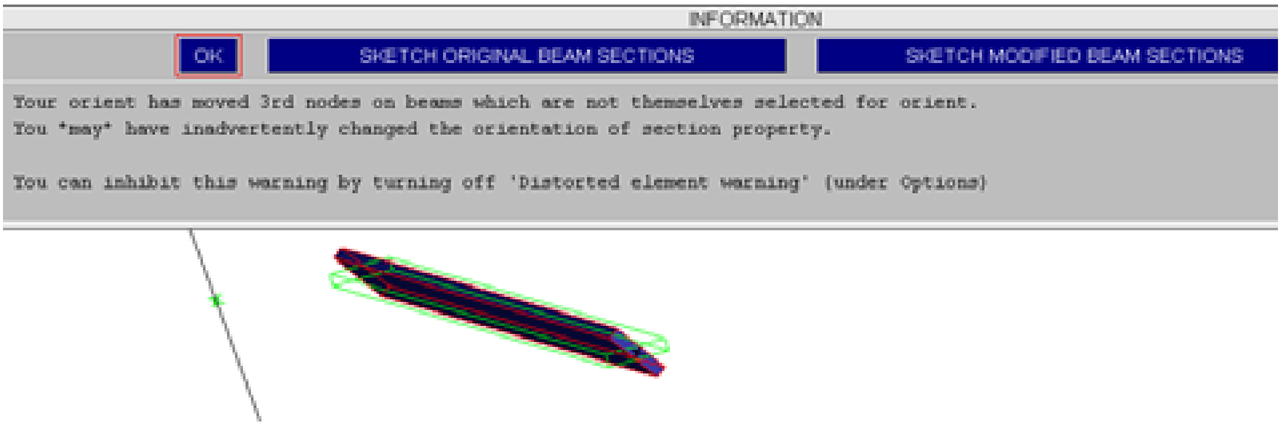
6.23.9. Check for element distortion

With the option **Distorted Element Warning** active, Primer94 will detect when an orient operation moves some but not all the nodes of an element.

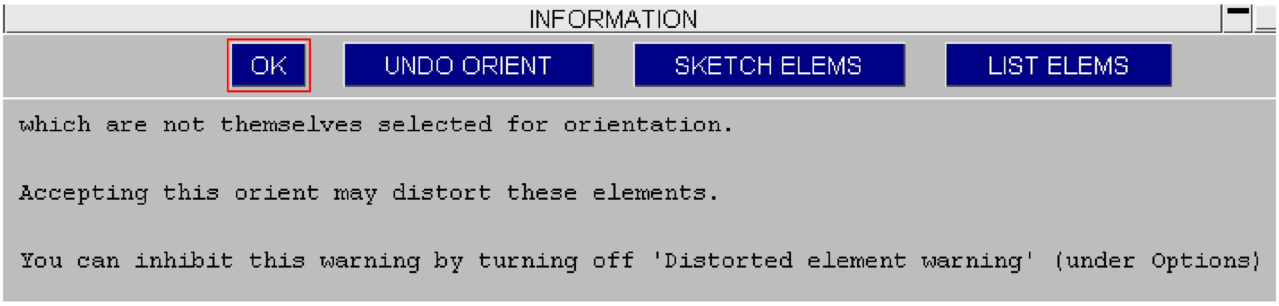
This typically occurs when orienting a rigid part with the option to **Move attached extra nodes** active. These nodes attach to deformable elements which will get distorted.

The function will allow you to sketch the problematic elements and undo the entire orient or just undo the orient of extra nodes if applicable.

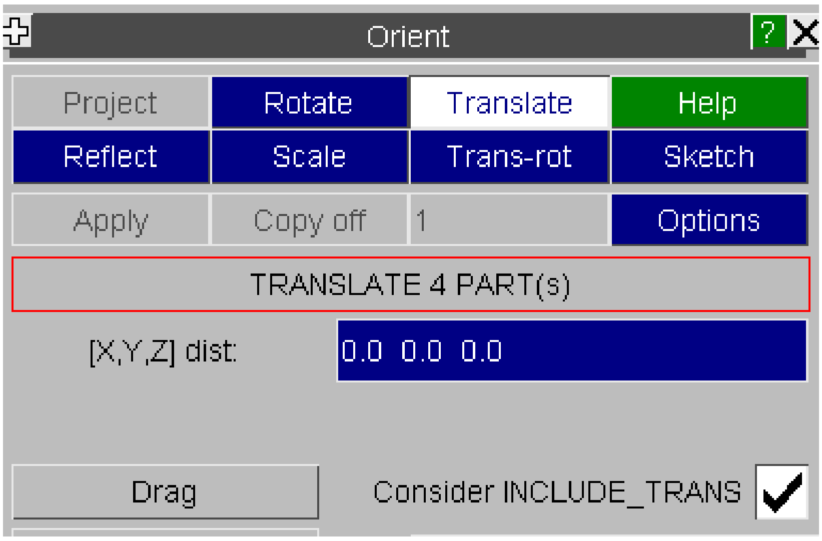
Additionally the 3rd nodes of non-circular beam sections are checked. If the orientation of the section changes, you will get a warning message which allows you to sketch the original and the current section.



Subsequently you will be offered the option to undo the orient.

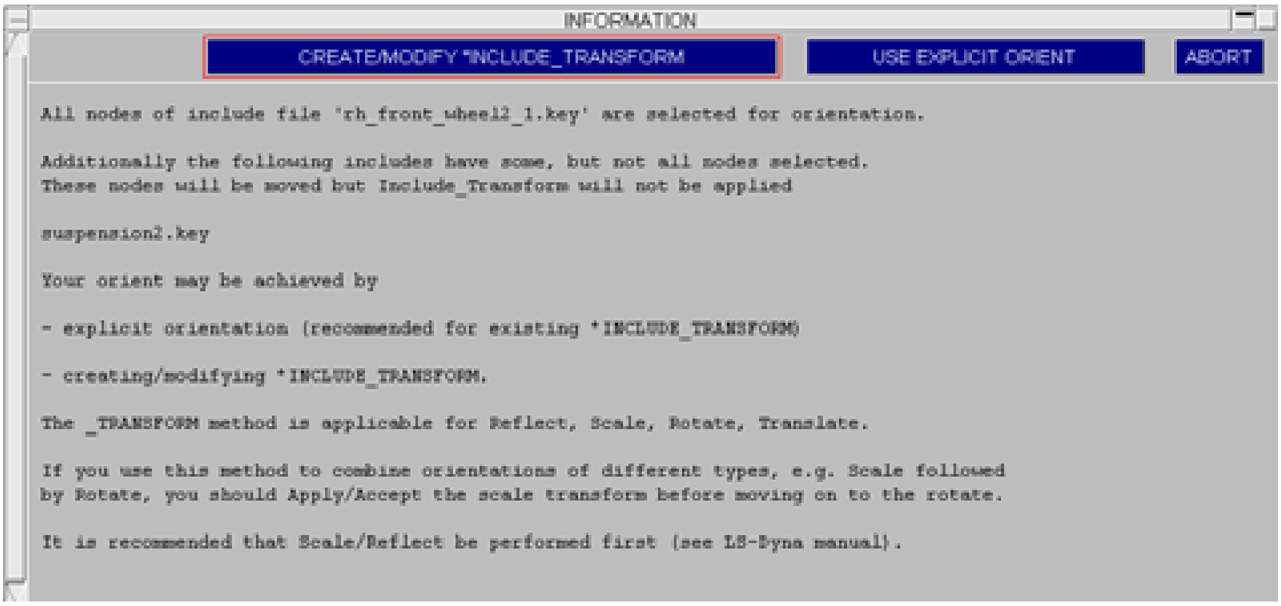


6.23.10. Orient and Include Transform

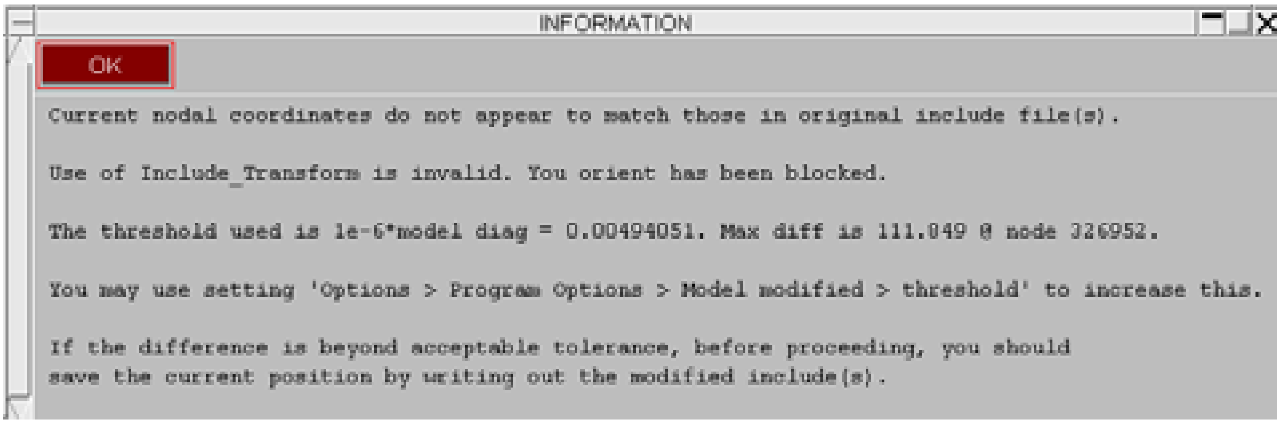


Consider INCLUDE_TRANSFORM option is available for Scale, Reflect, Rotate, Translate, Trans-Rot orients.

In this case, before orient is performed, Primer will inspect what is to be oriented looking for include files where all nodes are selected. If found, you will be offered the opportunity to create/modify Include_transform (as an alternative to explicit orient). You will also be warned of any includes where a subset of nodes have been selected - these obviously cannot use the transform method and will be oriented explicitly.



If you select the transform method Primer will then check that the nodes are in their original as read position, i.e .that an explicit orient has not been already applied. If the nodes fall outside the given tolerance the orient will be blocked. If the max diff reported seems small enough you may wish to increase the tolerance under Options > Program Options > Model Modified > Threshold. Otherwise, you need to save the include in its current position, before you can implement the include tranform method.



6.24 OTHER

Other lesser-used options are grouped into this popup. These are:

6.24.1 **Forming**

6.24.2 **Transfer**

6.24.1 METAL-FORMING

The **METAL-FORMING** specialist function allows you to include the effects of metal forming on parts in crash models by giving the part in the crash model the initial thicknesses and plastic strains from the metal-forming analysis. The models can have different meshes and different orientations. PRIMER will map the results from the forming model mesh onto the crash model mesh.

Select the **Other** pop-up menu from **Tools** and then press the **FORMING** button to start the process.

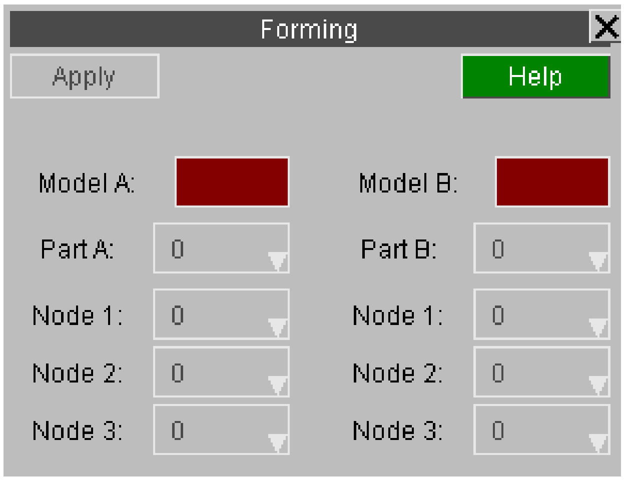
Main panel

The main metal-forming panel is shown in the adjacent figure.

The panel allows you to map the results from **Part B** in the forming model (**Model B**) onto **Part A** in the crash model (**Model A**).

Model A must be the crash model.
Model B must be the forming model.

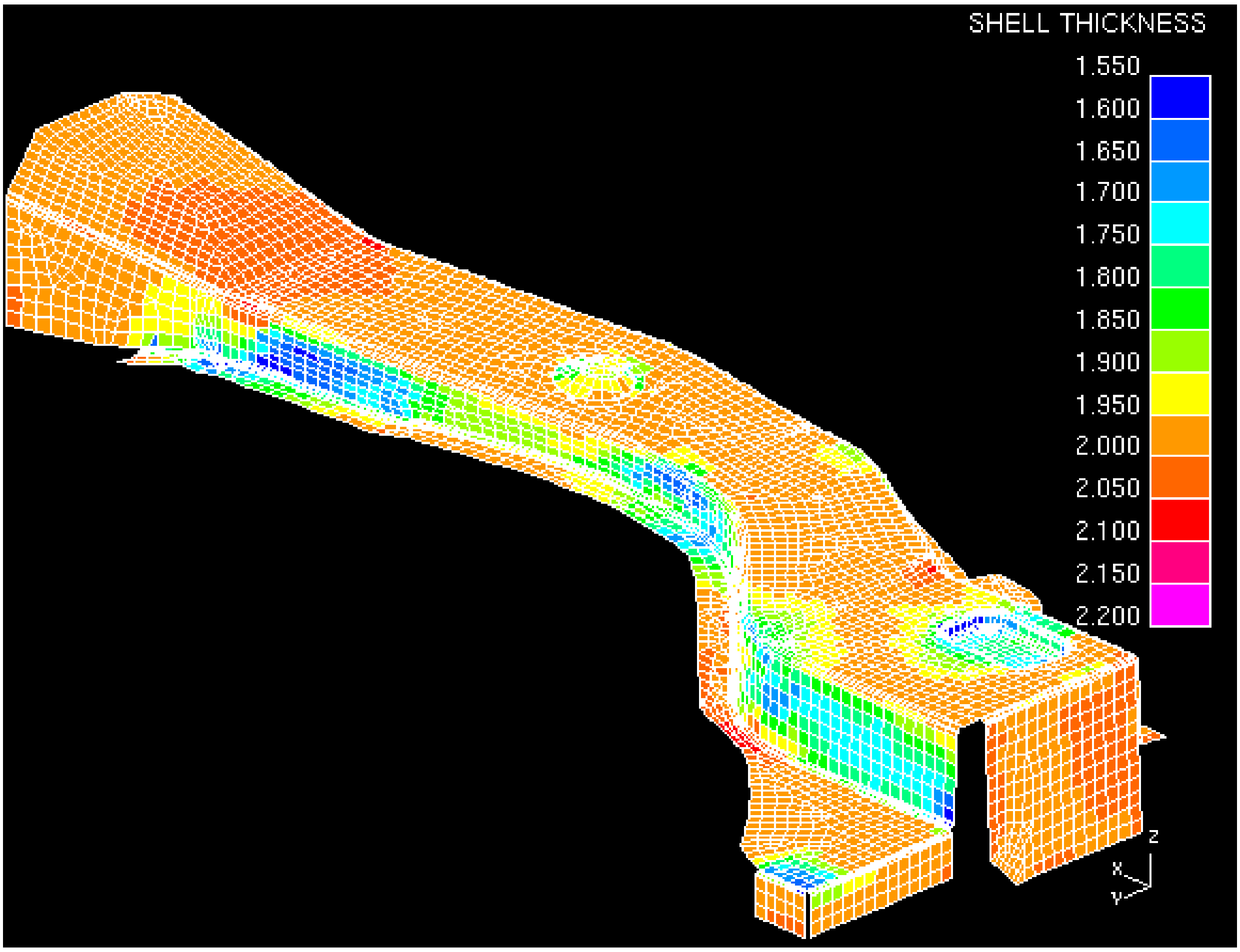
The process to map the results from the forming model onto the crash model is:



- Read the crash model and the forming model into PRIMER.
- Type in the model numbers of the crash model into **Model A** and the forming model into **Model B** (e.g. 1 and 2).
- Pick, select or type the part in the crash model you want to modify into **Part A**.
- Pick, select or type the equivalent part in the forming model into **Part B**.
- Give 3 pairs of equivalent nodes in **Model A** and **Model B** for orientation - i.e. **Node 1** in **Model A** is equivalent to **Node 1** in **Model B**.
- Press **APPLY** to map the results from the forming model to the crash model.

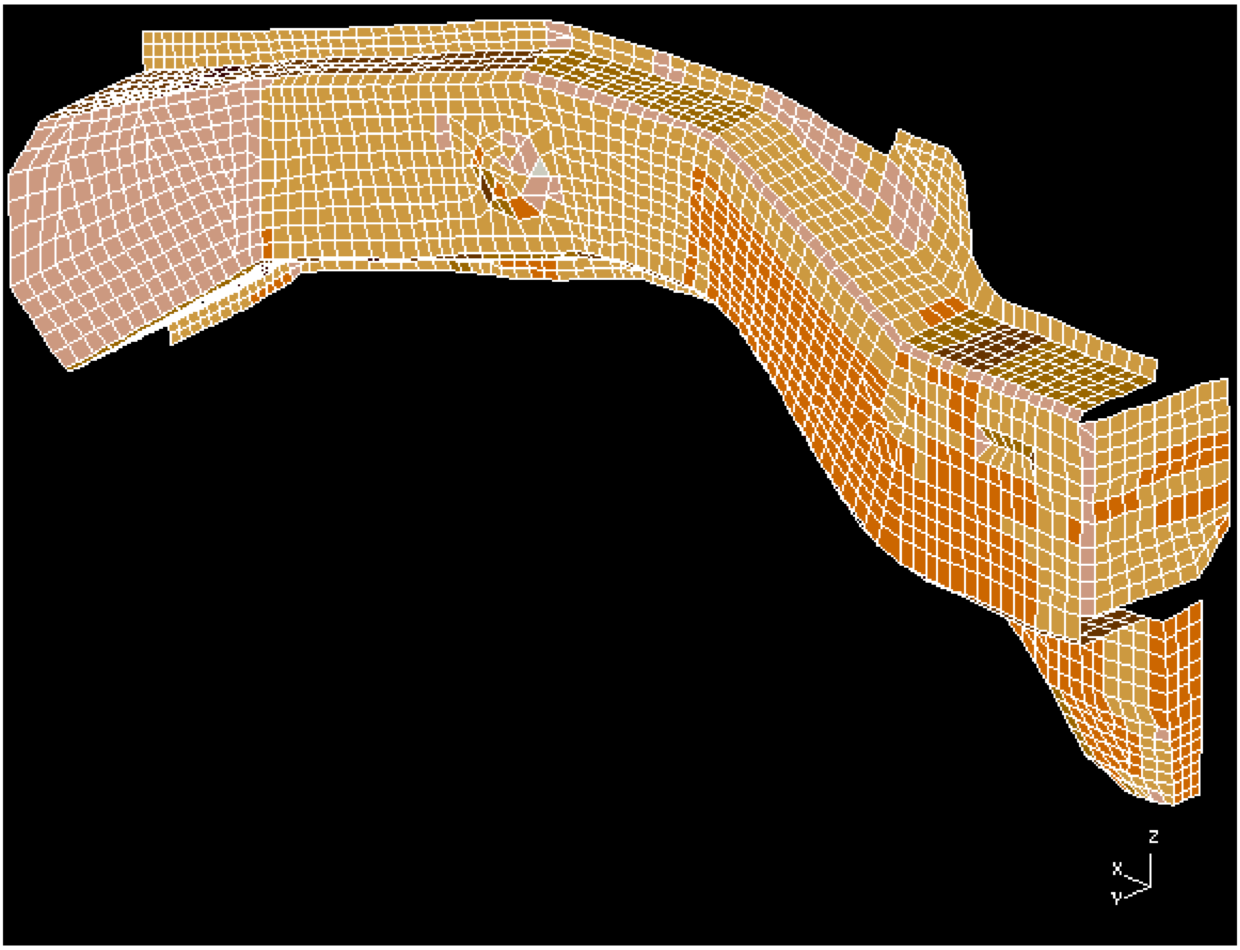
Example

The image below shows the thickness distribution from the forming analysis.



Thickness distribution in forming model

The crash model has a uniform initial thickness and a different mesh to the forming model

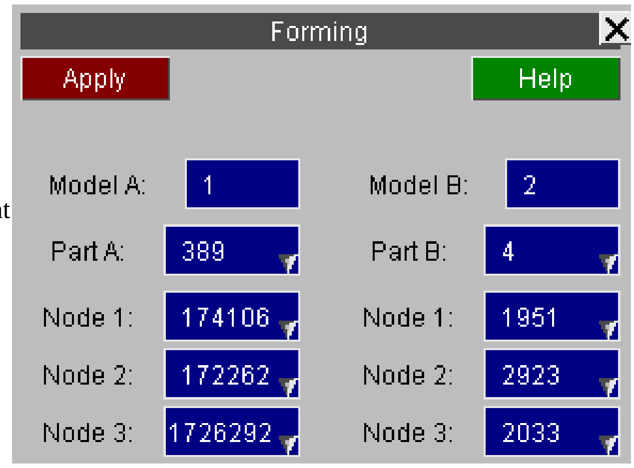


Mesh of crash model panel

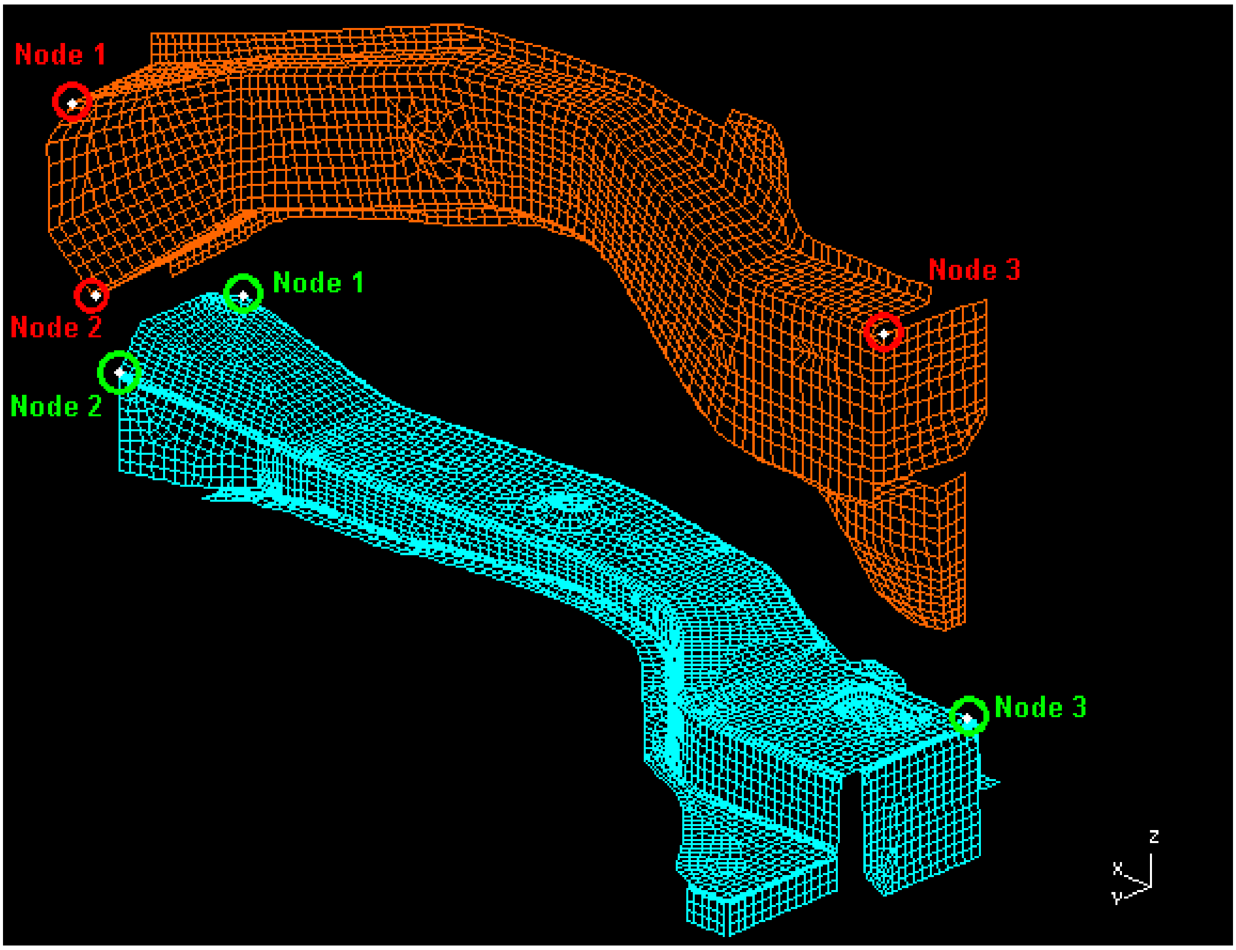
To map the results from the forming model panel onto the crash model panel we give the model number part and 3 nodes for each model.

The Crash model details are given in **model A** on the left side of the panel.

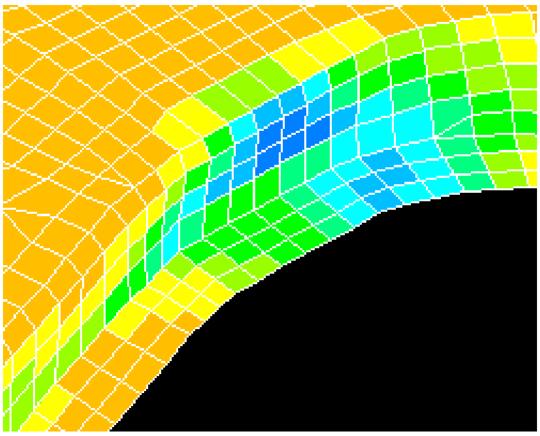
The Forming model details are given in **model B** on the right side of the panel.

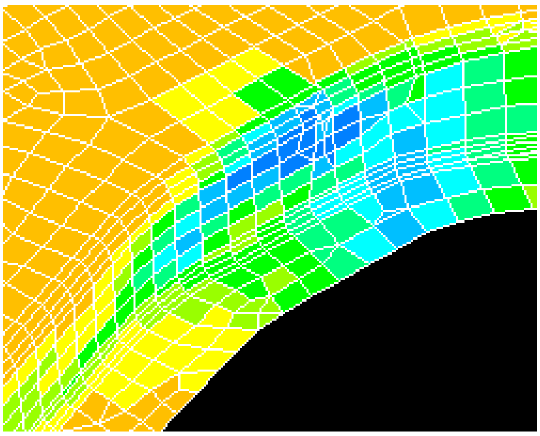


The figure below shows the locations of nodes 1, 2 and 3 in both models. These nodes **must** be at equivalent points on the panel. It is essential to make the 3 nodes as far apart as possible and not colinear so that PRIMER can map the results as accurately as possible.



When the **APPLY** button is pressed Primer takes the results from the forming model and maps them onto the crash model. For example plotting shell thickness gives:


Crash model

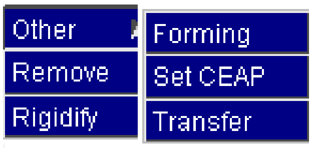

Forming model

Notes on metalforming

When PRIMER does the mapping process from the forming model onto the crash model it looks in the forming model for elements that are near the equivalent element in the crash model. There is a built in tolerance of 25mm in this process. If PRIMER cannot find any elements in the forming model within this tolerance the mapping will not be done for that element.

6.24.2 TRANSFER DATA

Transferring data between "source" (reference) and "target" models.



The **TRANSFER_DATA** function is designed to copy properties into model A (the "target" model) from model B (the "source"). The intention is that a model which has had some of its content stripped by a journey through an external piece of software, for example during remeshing, can be re-united with its original properties in a simple operation. Alternatively models from diverse sources can be "married up" with standard definitions held in read-only files.

- Most of the parameters in this panel can be preset in the "oa_pref" file - see below.
- In addition this operation can be performed in dialogue-only mode, and hence in batch - see Appendix X11.

The function is invoked from the pop-up menu **Other** in the **Tools** panel.

This figure shows the main **TRANSFER_DATA** panel in its initial state.

It is mandatory that you define the following parameters:

- **Target model** (into which it is transferred)
- **Source model** (from which data is extracted)

The source and target models must be different.

You may then define your **Data type to transfer** from the options below. Any permutation of these can be selected.

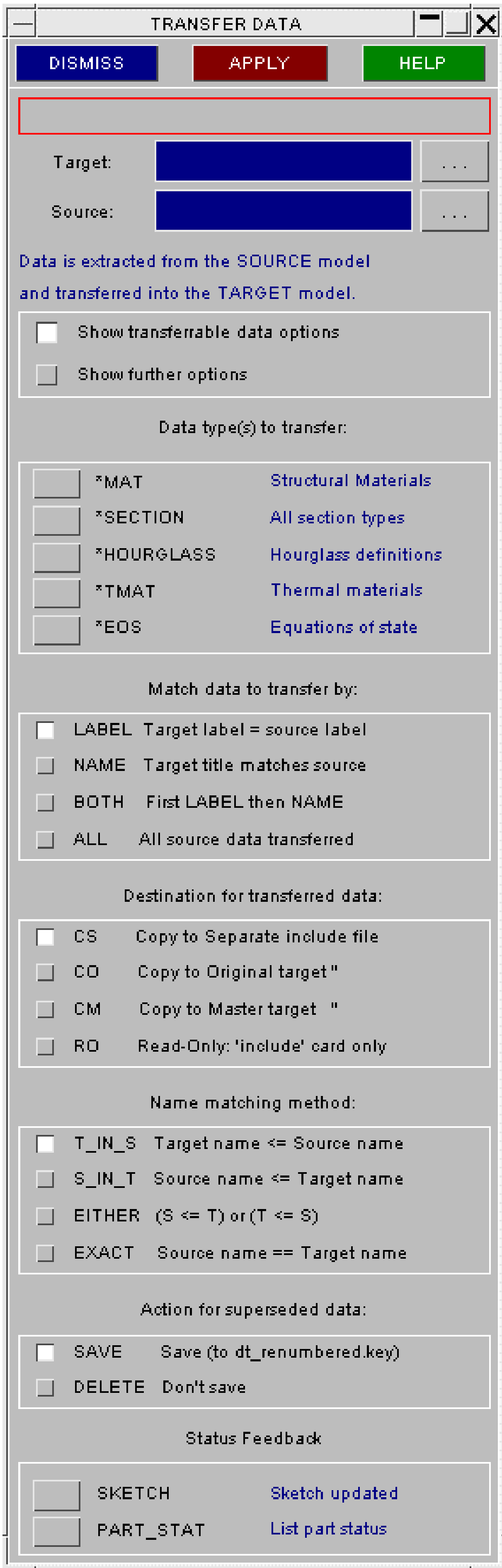
***MAT**	Structural materials
***SECTION**	Element sections
***HOURGLASS**	Hourglass definitions
***TMAT**	Thermal materials
***EOS**	Equations of state

Under further options, you may define one or more types for re-population of missing data from the following list.

DISCRETE	Spring/damper
JOINT	Joint props
WELD/RIVET	weld/rivet data
NODE_SET	node set data
NODAL_RB	nrb data

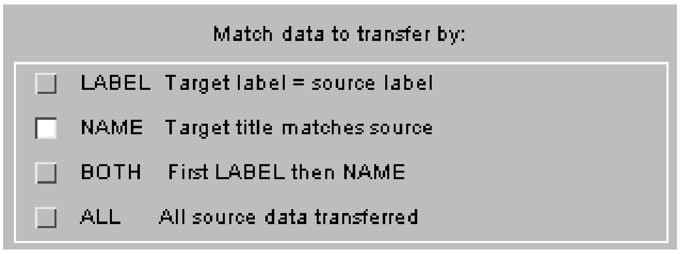
Once you have defined any one of these the **APPLY** button will become "live" and you will be able to proceed. However you should also consider the following settings:

Match data	Determines how data in the source model is selected for transfer
Destination	Determines how the transferred data is treated and stored in the target model
Name Matching	If you are matching data by **NAME** (or **BOTH**) then you should also consider which Name Matching Method to use. Name matching rules Explains the rules for case sensitivity, white space handling, etc when objects are matched by **NAME**.
Superseded data	Controls whether superseded (overwritten) objects in the target model are **SAVE**d or **DELETE**d.
Status Feedback	Controls how much visual and listing feedback about the transfer operation is given to the user.



Match Data to transfer by...

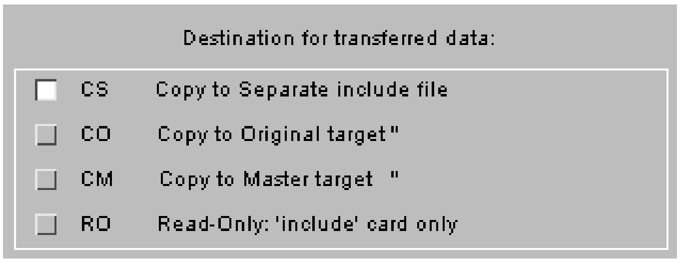
There is a range of ways in which data can be matched for transfer:



LABEL	If the labels in source and target models match
NAME	If the name in the target model is equal to, or a subset of, the name in the source model. This requires that the objects to be matched have titles, which is possible using: • The _TITLE keyword suffix in LS960 and above • The $PR_TITLE comment line in earlier versions
BOTH	Objects are matched first by LABEL, then by NAME
ALL	All data in the source model is transferred into the target.

Destination for transferred data...

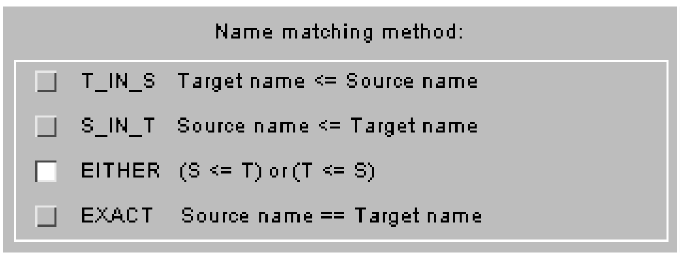
When data is transferred from source to target you can control how it is treated and stored.



CS (Copy to Separate include file)	Moves all transferred data to a new include file called `dt_transfer_from_<source>.key`
CO (Copy to Original include file)	All transferred data remains in the same include (or master) file as the original target data that has been overwritten.
CM (Copy to Master file)	All transferred data is moved into the master target file, regardless of its original location.
RO (Copy as read-only)	This assumes that the source file will be included verbatim in the ultimate keyword file that is analysed, so the following actions are taken: • Transferred data is placed into a special include file in the target model, marked as "read only". • When the target model is eventually written out this file is not included. • However the original <source> filename is referred to in an *INCLUDE statement via its full pathname. This means that the <source> file must be available on the computer that ultimately runs the analysis, using the same pathname.

Name Matching Method...

When objects are matched by **NAME** (or **BOTH**) the methods used to determine whether or not names match must be set. This option only becomes "live" when that is the case.



T_IN_S (Target IN Source)	The ***target*** name must equal the ***source*** name, or be a subset of it. For example: • Source name = "**MAT_123**" • Target name = "**123**" Would be matched, since "**123**" is a subset of "**MAT_123**"
S_IN_T (Source IN Target)	The ***source*** name must equal the ***target*** name, or be a subset of it. Therefore the example above would not be matched since "**MAT_123**" is not equal to or a subset of "**123**"
EITHER (T_IN_S or S_IN_T)	***Source*** may be a subset of ***target***, or ***target*** of ***source***, or they may be equal. Therefore the example above would be matched on the **T_IN_S** basis.
EXACT (Source == Target)	The ***source*** and ***target*** names must be identical. (No subset matching).

More rules for name matching...

When names are matched, by whatever method, the following rules also apply:

Case sensitivity	The matching process is always case-***in***sensitive (case is ignored). Therefore "**ABC**" matches "**abc**" matches "**AbC**" matches "**aBc**" etc
Leading and trailing spaces	Leading and trailing "white space" is always removed before matching takes place. Therefore " **ABC** " and "**ABC**" will match. (But "**ABC DEF**" and "**ABCDEF**" will not - the spaces are embedded.)
Empty names	Names that are empty are always ignored. So unnamed objects will never be matched.

Superseded Data: what happens when target objects are to be overwritten.

Transferring data from source to target models means implicitly that the target data will be overwritten. You can choose to:

SAVE The original target data is copied to a new, unused label and is transferred to include file "dt_renumbered.key". This new copy is not referenced by anything and can be removed by a **CLEANUP_UNUSED** operation.

DELETE The original target data is overwritten and lost permanently.

Status Feedback: controlling visual and tabular feedback

A summary of what has been overwritten with what (by label) is always printed, together with totals, but you can choose to have further feedback:

SKETCH Sketches on the current image all objects that have been modified by the data transfer operation

PART_STATUS Produces a table (by label) of all parts that have been modified (or have had some attribute, eg material, modified); and also a table of those that are unchanged.

How data transferred by **NAME** gets special treatment

When data is transferred by **NAME** it is possible that more than one object in the target model will match an object in the source model, in which case multiple copies of the source data will be transferred into the target model. For example consider the following case for materials where **EITHER** name matching is used:

Original target model contents	Source model contents	Resulting output in target model
`MAT #1 "Steel 316s12"`	`MAT #1 "Aluminium"`	`MAT #2 "Generic steel"`
`MAT #2 "Steel to BS4360"`	`MAT #2 "Generic steel"`	`MAT #2 "Generic steel"`
`MAT #3 "Special steel for supports"`	`MAT #3 "Concrete"`	`MAT #2 "Generic steel"`

In this case all three target materials contain the word "`steel`", so they will all be overwritten by material #2 (`Generic steel`) from the source model. This would leave three materials in the target model with the same label, so how is this resolved? What happens is this:

Every existing target object that is about to be overwritten has its properties copied to the highest object label + 1.

In this case since superseded data is to be **SAVE**d, before overwriting the "old" source material properties:
- Original target material #1 is copied to #4.
- Material #2 is copied to #5
- Material #3 is copied to #6

These new (#4 .. #6) definitions are not referenced by anything, and are placed into a separate include file. If they are not required a **CLEANUP_UNUSED** operation will delete them.

Source object data is copied in verbatim

- Source material #2 is copied verbatim into target materials #1, #2 and #3
- At this stage there will be 3 materials in the target model all labelled #2 - clearly illegal

A post-transfer check and sorting out takes place

Once all transfers by **NAME** have taken place then any clashes or illegalities in the target model are sorted out.
- Where any incoming label from a source object clashes with a pre-existing (but not overwritten) target object of the same label, then that original target object is re-labelled.
- Then a check for multiple instances of transferred in objects is made (which will all share the same label), and these are "collapsed" into a single definition.

In the example above the three target materials, all labelled #2, would be collapsed into a single material definition and all references to them (eg from *PART cards) are implicitly also rationalised to reference this single material definition.

How are objects such as loadcurves which might be referenced by transferred data handled?

For example you might transfer a material which contains references to loadcurves in the source model.

In this situation these "supporting" objects are also transferred, although the logic for handling label clashes in the target model differs from the data above:
- If there is no label clash they are copied in "as is".
- If there is a label clash the original definition in the target model is renumbered to a free slot, and the new data is copied in.

This is different because although existing objects in the target model might be relabelled, they are not superseded.

Setting **TRANSFER_DATA** defaults in the "oa_pref" file.

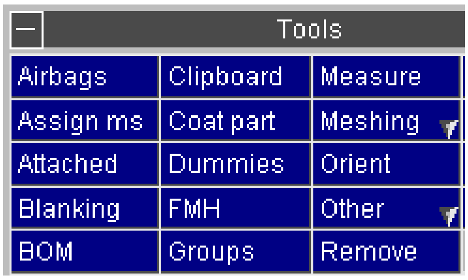
Most of the settings in the **TRANSFER_DATA** panel can be pre-defined in the "oa_pref" file. The following table shows the default settings, and the ways in which they can be modified in that file:

Setting	Default value	"oa_pref" file keyword	"oa_pref" options
Target model	*Undefined*. But if only one model is present then this is assumed	n/a	n/a
Source model	*Undefined*	`primer*transfer_source_file:`	*<filename>*
Data Type	*Undefined*	`primer*transfer_data_type:`	• MAT • SECTION • EOS • HOURGLASS • TMAT
Match By method	**NAME**	`primer*transfer_match_by:`	• ID • NAME • BOTH • ALL
Destination Action	**CS** (Copy to Separate)	`primer*transfer_action:`	• CS • CO • CM • RO
Name Matching method	**EITHER**	`primer*transfer_name_match:`	• T_IN_S • S_IN_T • EITHER • EXACT
Superseded data action	**SAVE**	`primer*transfer_superseded:`	• SAVE • DELETE
Feedback options	**SKETCH** (only in graphical mode) **PART_STATUS**	n/a	n/a

Performing **TRANSFER_DATA** operations in command-line mode.

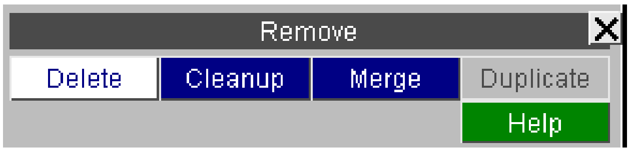
All the operations above can also be performed in command-line mode, and therefore also in batch. See Appendix X11 for more information about this.

6.25 REMOVE Delete unwanted, model clean-up, node merging and duplicate elimination.



The **Remove** command is invoked from the **Tools** panel at the top of the screen.
It has three options:

Delete	Delete unwanted entities from any model;
Cleanup	Remove unused non-structural model entities;
Merge	Combine together nodes within a certain tolerance

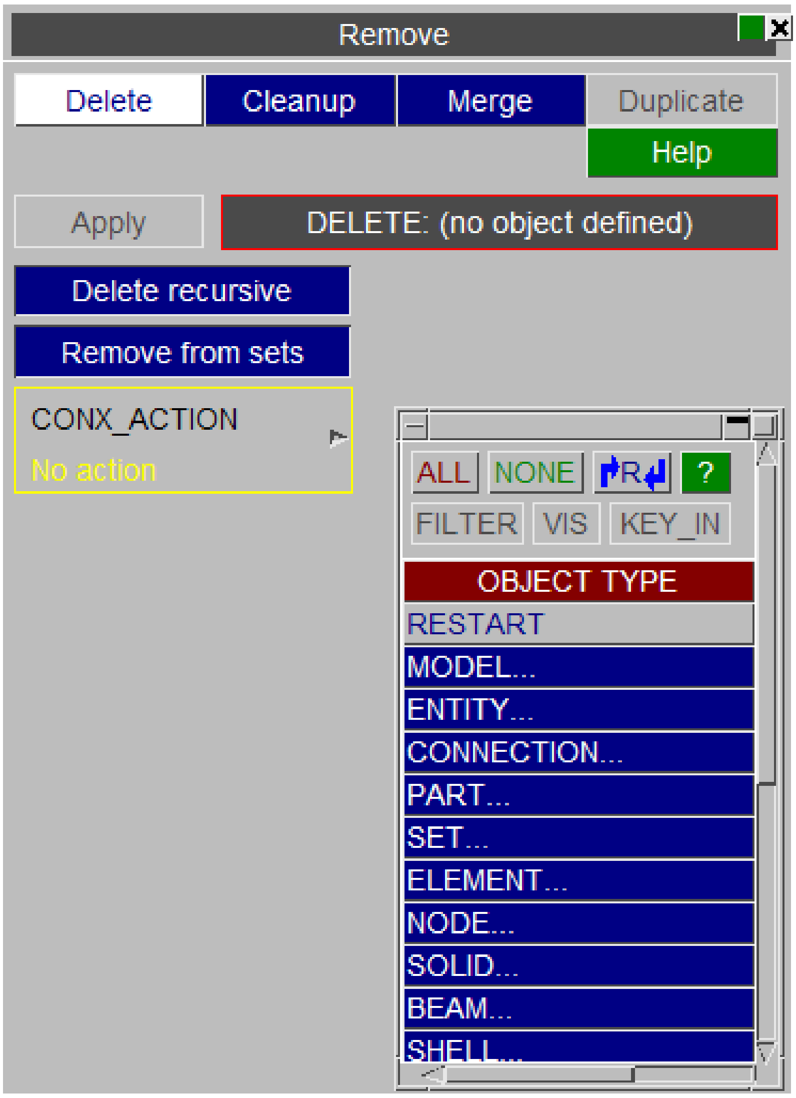


6.25.1 DELETE UNWANTED

To delete model entities use **Delete**.

This option can be used to delete anything from entire models to individual items, using the standard PRIMER hierarchy.

(Note, however, that deleting a complete model is much faster if the main **MODEL> DELETE** command is used because no cross-checking for dependencies is required.)



To delete entities:
- Select items to be deleted as described in Section 6.0.2.
- On pressing **Apply** PRIMER searches through the model hierarchy (subject to the recursion settings, see rule 2 below).

- You are presented with a list of everything marked for deletion, and given the opportunity to modify it.
- When you confirm the deletion list those items are removed.

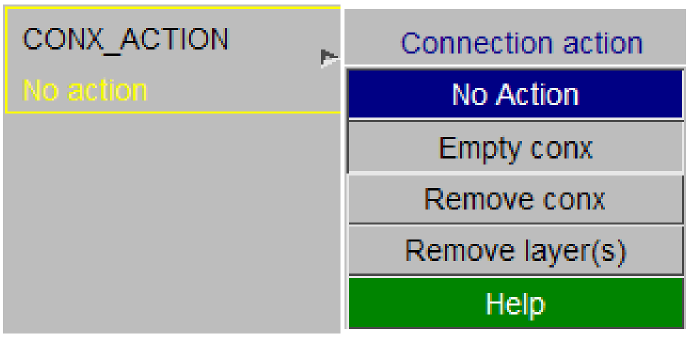
6.25.1.1 Deletion Rules

There are some strict rules governing how deletion is applied which are intended to prevent you accidentally leaving something "hanging" by deleting an item upon which it depends. These rules are:

- No entity may be deleted if it is "locked" because it is referenced by another entity which is not itself being deleted. For example a part definition cannot be deleted if it is referenced by an element which is to remain. The entity hierarchy is described in section 6.0.1.

  However if "Forced deletion", added in Primer 10.0, is used then items which are "locked" can be deleted. However they will be replaced with Latent definitions, so this is not a good general solution to the "why can't I delete this?" problem. Forced deletion is described in more detail below.
- Switching **Delete recursive** on or off (see figure above) before pressing **Apply** affects the selection of entities to be deleted. For example if parts are selected for deletion and recursive deletion is switched on then all entities (constraints, elements and nodes etc.) associated with that part will also be deleted. Without recursion only the chosen parts are selected for deletion, and only those which are not referenced by any entities will in fact be deleted. (see rule 1).
- If the **REMOVE from sets** button is selected then PRIMER removes deleted entities from sets. With this option any deleted part (or node etc) will also have its reference removed from any part sets (or node sets etc.).Without this option a part (or node etc.) referenced by a part set (or node set etc.) will not be deleted when it is referenced by another entity (ie the set) which is to remain.

Connections may need some special treatment during deletion. The **CONX_ACTION** popup allows you to set what action to do when PRIMER needs to look at connections that are attached/tied to parts or shells that you select for deletion. The options are:



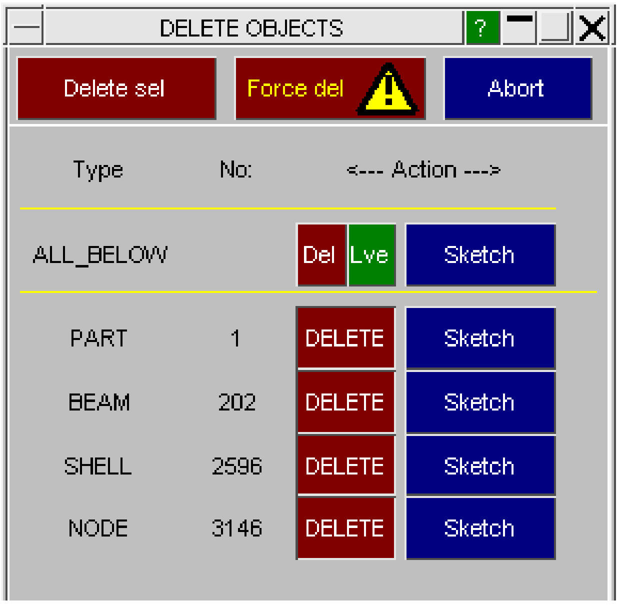
- **No Action**. PRIMER will not take any special action. This could cause problems if the connection uses `*ELEMENT_BEAM_PID` as the element refers to the part which will stop the part being deleted.
- **Empty conx**. PRIMER will delete the entities (e.g. nodes and solids) that make up the connection and leave the connection latent.
- **Remove conx**. PRIMER will delete the connection and it's entities.
- **Remove layer(s)**. PRIMER will delete the entities (e.g. nodes and solids) that make up the connection and then remove the layer containing the part from the connection. If the connection then only has one layer the connection will be deleted.

6.25.1.2 Example

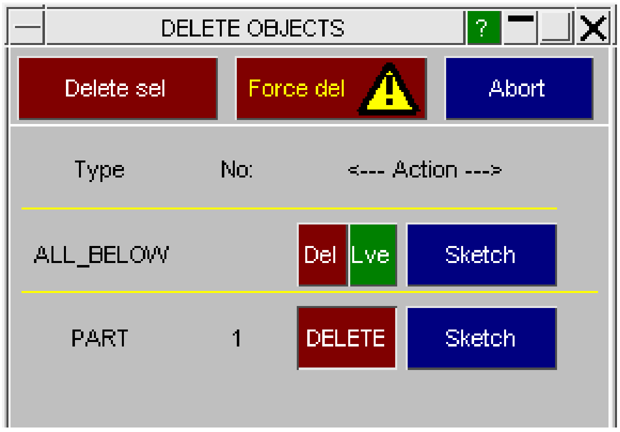
The figures below show the effect of recursive deletion. The entities selected by PRIMER with (left column) and without (right column) recursion are shown. In this example the user selected one part to be deleted.

- **SKETCH** will sketch each of the entities that are to be deleted. (**SKETCH** may not be available for all entity types.)
- **ABORT_SEL** will quit the delete operation making no changes to the model.
- The user can toggle each of the entities using the **DELETE/LEAVE** and **DEL/LVE** toggles so that they will be deleted or left as required.
- **DELETE_SEL** executes the deletion using the above rules and reports to the user the number of entities that have been deleted - see the lower pair of figures.

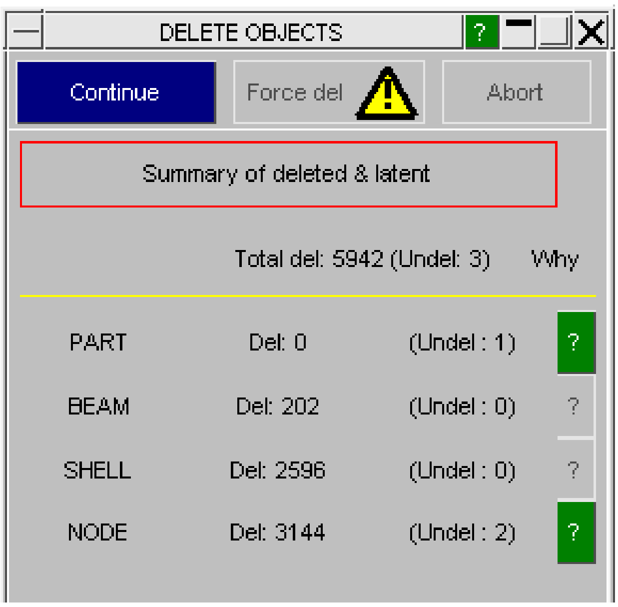
DELETE_RECURSIVE ON **DELETE_RECURSIVE OFF**



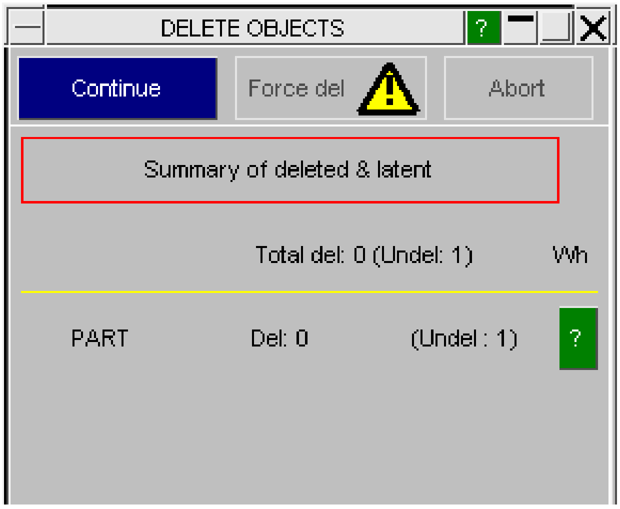
The left figure shows that recursive deletion has found a number of elements, nodes etc.



Without recursion PRIMER finds the entities shown on the right(ie the part only).



Following **DELETE_SEL** the part and its associated data have been deleted on the left, since recursive deletion has picked up all the subordinate items.



On the right nothing has been deleted since the part is"locked" by its elements.

6.25.1.3 Finding out why an item is locked against deletion

You can find out why items have not been deleted by clicking on the relevant **[?]** button in the "Why" column.

For example the PART definition above was not deleted, and the "Why" button produces this panel which shows that it is referenced by:

- Two *SET_PART definitions. (These would not in themselves lock the part if **Remove from Sets** is active)
- Two connections. It is these which are truly locking the part.

You can delve deeper into exactly what is locking the item by using the **X-Refs** button to map the standard Cross-references panel.

6.25.1.4 **Force Del** Using forcible deletion

Deleting items whether they are "locked" or not.

At a first glance forcible deletion, which deletes items even when they are "locked" by a reference from elsewhere, appears to be a simple solution to the problem of removing things which refuse to be deleted. However before using it you should understand what it does and how this can affect your model.

If you select **Force del** rather than the normal **Delete sel** the following happens:

- Items which can be deleted using the normal rules, ie which are not locked, are removed as above.
- Items which remain, but which are marked for deletion are then also forcibly removed ...

  ... but they are replaced with Latent (empty) definitions of the same type.

If you run a **Model > Check** on a model that has undergone forcible deletion you will find that all these Latent definitions generate "Referenced but undefined" errors, and the deck is very unlikely to initialise in LS-DYNA in that state.

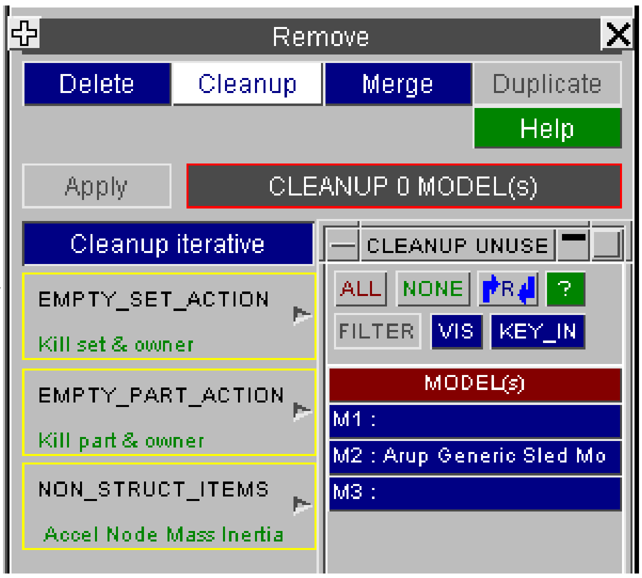
So if it leaves all these errors what is **Force Del** for? Two special cases:

1. When replacing include files it is useful to be able to delete ***all*** existing file contents, regardless of references from elsewhere in the model, since we know that importing the new include file will re-populate the latent items with new definitions. PRIMER uses this during the [Tools] **Include, Replace contents** process.

2. A very careful user can also perform a similar operation manually. If you know that you are going to replace what you have removed, or deal with the latent definitions in some way, you can use this feature manually. However it will be your responsibility to sort out all the Latent definitions.

What you should ***not*** do is use **Force Del** as a quick fix to the problem that normal deletion refuses to remove locked items. Instead you should use the **[?]** "why" buttons to find out why items are locked, and deal with the problems at source.

6.25.2 Clean-up Unused



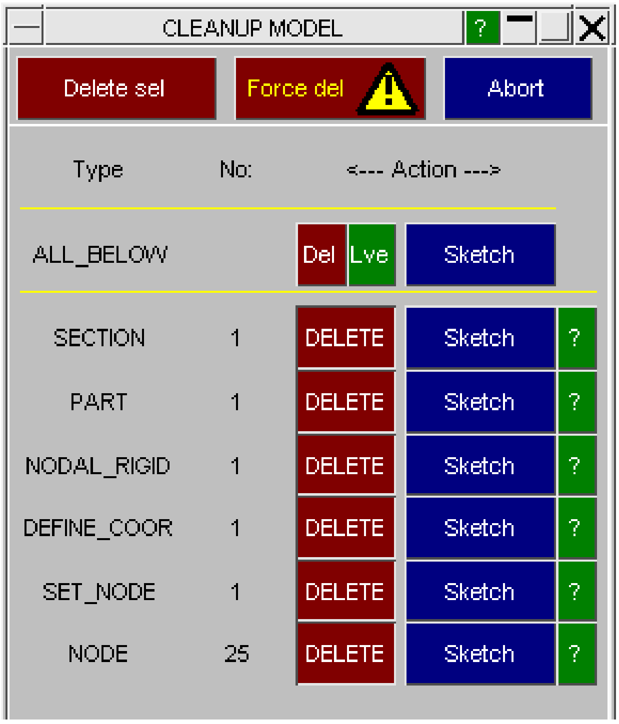
CLEAN-UP UNUSED removes redundant entities from models. For example loadcurves that are not referenced, non-structural nodes, etc.

On entering **CLEAN-UP UNUSED** from the **REMOVE** menu the user is prompted to define which model(s) to cleanup. More than one model can be cleaned up at once but the user may find it safer to clean up a single model at a time.

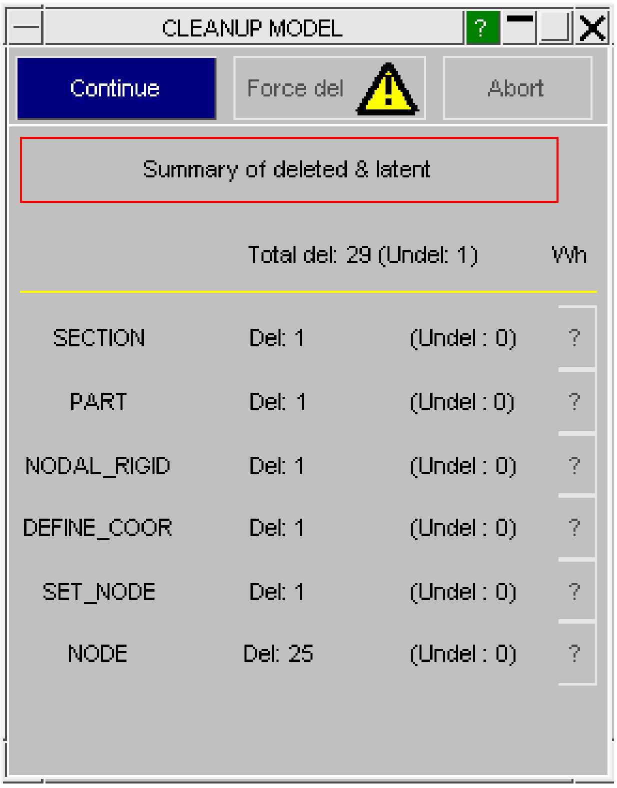
Once model(s) have been chosen **APPLY** starts the clean-up process.

6.25.2.1 CLEANUP UNUSED example

In the same way as for **Delete** you are presented with a list of items identified for removal, and you must confirm these. In the example above pressing **Apply** gives:



This figure shows the confirmation panel of items identified as unwanted.



This figure shows the resulting deletion echo.

Normally everything marked as unwanted should indeed be deleted, but if anything remains you can use the **[?]** button in the Why? column as above to find out what is locking it.

6.25.2.2 Switches controlling CLEANUP UNUSED

Clearly PRIMER must search through the model(s) chosen to identify things that are no longer needed, and there are several switches which may be used to control this process.

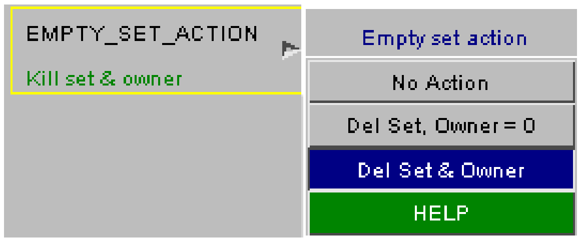
CLEANUP_ITERATIVE Whether to use iterative searching for items.

Sometimes when an item is found to be redundant removing it can lead to other items becoming redundant. It may require multiple passes through the model to identify all these consequential deletions.

For example, if a model contains a part with no elements then in the first iteration the part will be flagged for removal. Iteration 2 will find that the section and material properties etc that this part referenced are also no longer required and will flag them for removal (unless other parts reference these). Iteration 3 will find any loadcurves etc used by the materials that have been flagged for removal (if these aren't used by other materials). And so on until nothing remains to be found.

By default this iterative process will be used, but you can turn it off in order to limit the extent of a clean-up operation to a single pass. This can give more control over what is removed in each **CLEANUP** operation.

EMPTY_SET_ACTION Dealing with empty SETS.



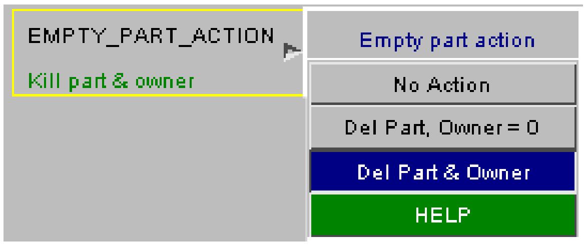
When all the contents of a SET have been removed (following a **REMOVE** operation) the empty SET definition itself may remain.

This is not strictly illegal, but it can cause problems in the analysis code at run-time since LS-DYNA may crash if sets with no contents are found.
Therefore PRIMER treats it as an error, and provides the following options for dealing with it:

No action	The set is not removed, and references to it remain
Del Set, Owner = 0	The set is marked for deletion, and any references to it are replaced with a zero. This can cause unexpected outcomes when <`set id = 0`> implies "use the whole model", as is the case in some contexts - use with care!
Del Set & Owner	Both the set ***and the item referring to it*** are marked for deletion. This is the default setting, and generally the most useful.

.There is an exception in the last (**Del Set & Owner**) case in that where a reference to a set is optional, for example "set of nodes exempted from ..." where replacing the reference with a zero would be harmless, that solution is adopted instead and the "owner" definition is not marked for deletion.

EMPTY_PART_ACTION Dealing with empty Parts

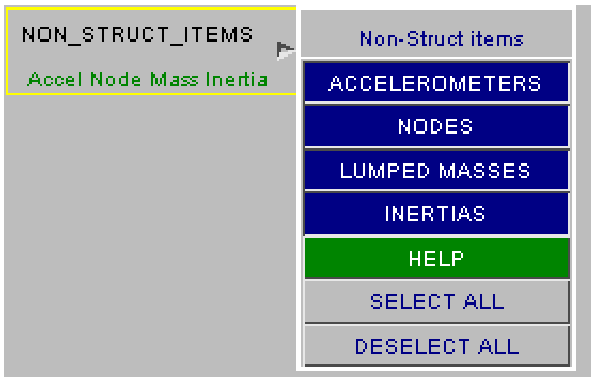


If all the elements have been deleted from (or transferred out of) a part then it will be empty.

As with empty sets this is not strictly illegal, but it can cause problems in LS-DYNA.
Therefore PRIMER treats it as an error and provides the following options:

No action	The part is not removed, and references to it remain.
Del Part, Owner = 0	The part is marked for deletion, and any references to it are replaced with a zero. This can cause unexpected outcomes when <`part id = 0`> implies "use the whole model", as is the case in some contexts - use with care!
Del Part & Owner	Both the part ***and the item referring to it*** are marked for deletion. This is the default setting, and generally the most useful.

NON_STRUCT_ITEMS Dealing with items that have no structural purpose



Following the removal of other things you can be left with valid and legal objects which are nevertheless "non-structural", meaning that they will not play any part in an analysis.

PRIMER can detect and mark for deletion the following:

ACCELEROMETERS	Accelerometers which exist in isolation
NODES	Nodes which are not attached to elements, not extra nodes on a rigid part or some other constraint, and not useful in any other context.
LUMPED MASSES	Lumped mass elements attached to non-structural nodes.
INERTIAS	Inertia elements attached to non-structural nodes.

Some of the seatbelt-related elements (sliprings, pretensioners, etc) can also be non-structural by the definitions above. However they may often be imported as part of pre-meshed dummy models, and will become structural when attached to a vehicle, thus it would be unfortunate if they were accidentally deleted. Therefore they are not included in these checks and will need to be deleted manually if required, but the overhead of leaving them is minimal.

Other things marked for deletion during a cleanup operation.

As well as items which are unused, plus those which meet the criteria above, the following things are also automatically checked and marked for deletion as required:

CONSTRAINED definitions which have become redundant or invalid:

- Generalized welds referencing sets containing fewer than 2 nodes;
- Linear constraints ditto
- Node sets ditto
- Shell to solid defns ditto
- Tied nodes ditto
- Joints for which attached parts are absent, no longer rigid, or non-structural;

SEGMENTS that are no longer valid:

- Because their parent set, load or other definition has been removed;
- Because they no longer lie on a shell element, or the face of a 3D element.

"Latent" definitions serving no useful purpose plus their referees:

- Items referenced in sets, boundary conditions, database cards, initial definitions, etc that have never been explicitly defined or referenced in other contexts. These can be deleted along with the references to them since they do not serve any independent purpose in isolation. (For example a restraint on a node not referred to anywhere else is redundant.)

6.25.2.3 When multiple calls to CLEANUP UNUSED may be required

Usually turning the "iterative" switch on will identify all items that are to be removed in a single operation, but if a large number of different item types are to be removed it may require multiple clean-ups to remove absolutely everything unwanted from a model.

The reason is that in iterative cleanup mode PRIMER has to ask the question "if I remove this then can I remove that?" to many levels of complexity, and if a lot of different entity types are removed there comes a stage at which this gets too difficult to resolve in a single operation.

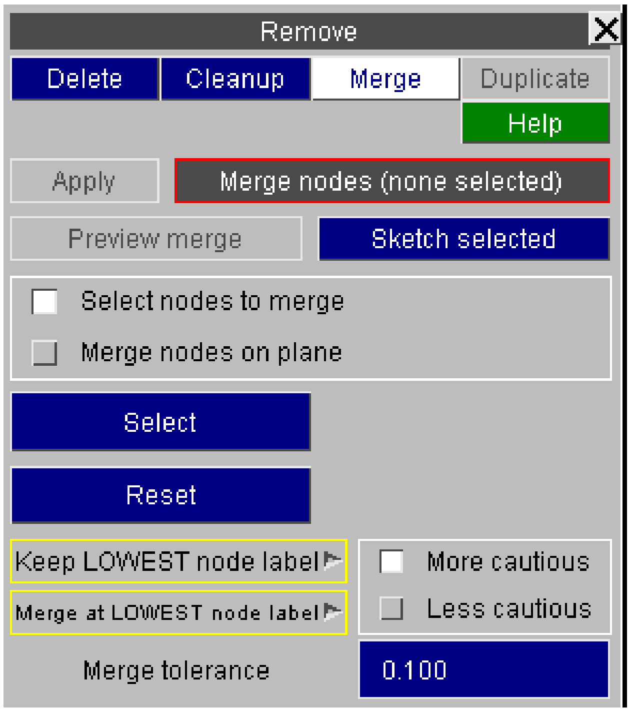
Therefore if a long list of different item types is identified for removal it is usually worth repeating the **CLEANUP UNUSED** operation until the message `"No items found to delete"` is returned. This guarantees that no further unwanted items remain.

6.25.3 MERGE_NODES

This operation will permit coincident (and/or close) nodes to be merged.

The MERGE NODES function is accessed either through the **REMOVE** button on the top panel or through KEYWORDS->**NODE**->**MERGE**.

6.25.3.1 Using MERGE NODES



Merge applied to a selection of nodes or on a global plane, using user defined tolerance.

- retain lowest/highest node label
- merge at lowest/highest node or at average position

Merge nodes will never collapse a shell element.

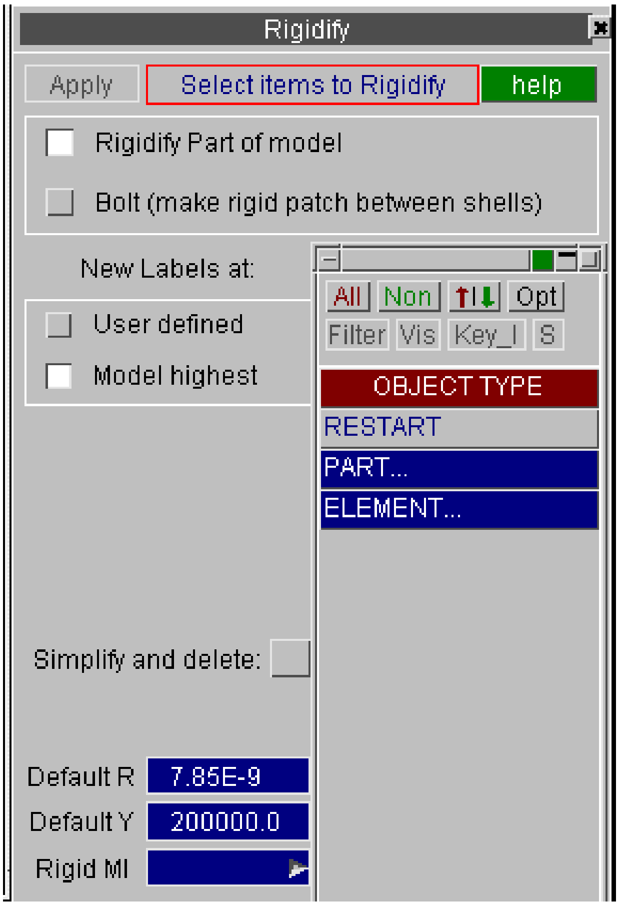
In "*Less cautious*" mode it may change the length of beam elements and modify nodal rigid bodies (possibly reducing them to a single node).

PREVIEW MERGE will highlight the nodes to be merged.

6.26 RIGIDIFY

The **RIGIDIFY** function has 2 modes.

6.26.1 Rigidify Part of Model



Rigidify Part of Model is designed to make rigid a selection of Parts or Elements in a model. If some elements of a part are selected, it does this by creating new rigid parts and moving the selected elements into them. If a part is selected or all elements of a part, a rigid material card is substituted. All rigidified parts are slaved to a dummy master part.

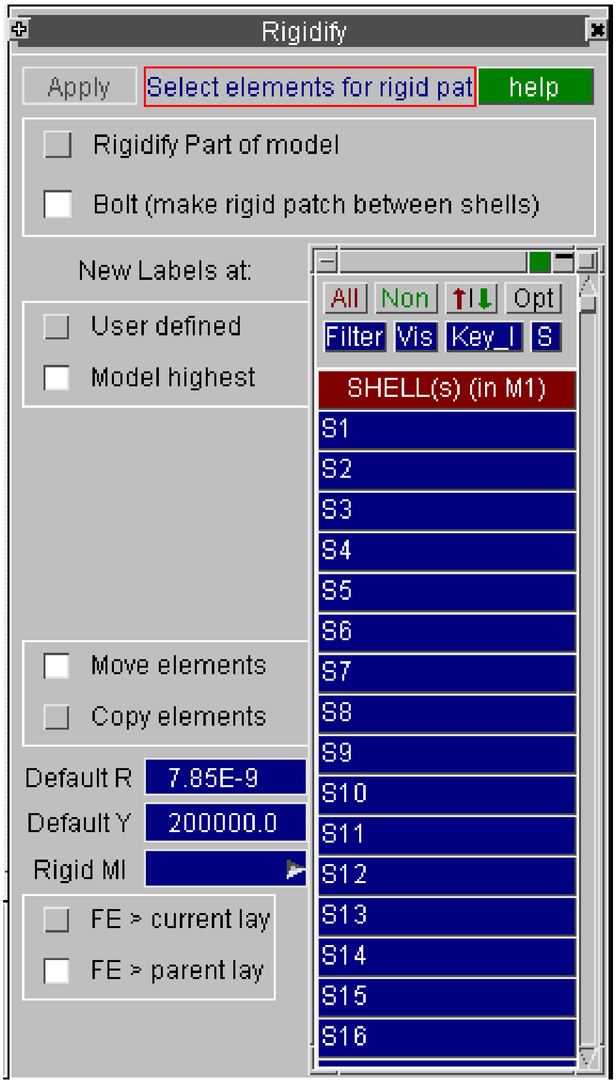
The density and modulus of newly created materials will normally match those of the deformable ones they replace, to preserve the part mass. If these values cannot be determined, the default values will be used.

The user simply selects items in the usual way with the object menu and presses **APPLY**.

However, the consequences of applying **RIGIDIFY** to the model are **considerable and irreversible**. It is therefore recommended that the user always saves the model before he/she presses **APPLY**.

The user can choose to delete the original elements chosen to rigidify if they wish. In this case, the mass and inertia properties of the original elements are applied to the dummy master part in the for of a *PART_INERTIA. This option is activated by checking the **Simplify and delete** check box.

6.26.2 Create Rigid Patch



Create Rigid Patch will rigidify a selection of elements and (if they belong to different parts) create a merge onto the first part selected. The option is available to copy the selected elements rather than just move them into the new rigid part (default).

New Part Labels: These may be created starting with a user defined label (which will be automatically incremented to first free) or at the highest in layer + 1 for the part or at model highest + 1. If highest in layer + 1 is not available, model highest + 1 is used.

6.26.3 How RIGIDIFY will change the model

All the structural elements (shells, thick shells, solids and beams) that have been selected will end up in a rigid part slaved to the dummy master rigid part.

Selection of a deformable Part: if a part is selected directly or all the elements of a part are selected, a new material will be generated (keeping the same values of E and rho) and the part->material reference updated. The part will be merged to the dummy master part.

Selection of deformable elements: if some elements of a deformable part are selected, a new part and material will be created and the elements will be moved into this part. The part will be merged to the dummy master part.

Selection of a rigid part or an element of a rigid part: in this case only the merge to the dummy master is made.

The rigidified parts will all be written to a group, so the user can easily determine the mass properties.

6.26.4 How RIGIDIFY will flag items for deletion

To ensure that the rigidification of the model does not incur errors Primer will flag selected items for deletion.

Certain nodal constraints within the rigidified area will require removal, these include joints, constrained welds, boundary prescribed motions, boundary spcs, constrained extra nodes and nodal rigid bodies. In these cases connection to nodes located outside the rigidified area will be retained by the creation of additional constrained extra nodes.

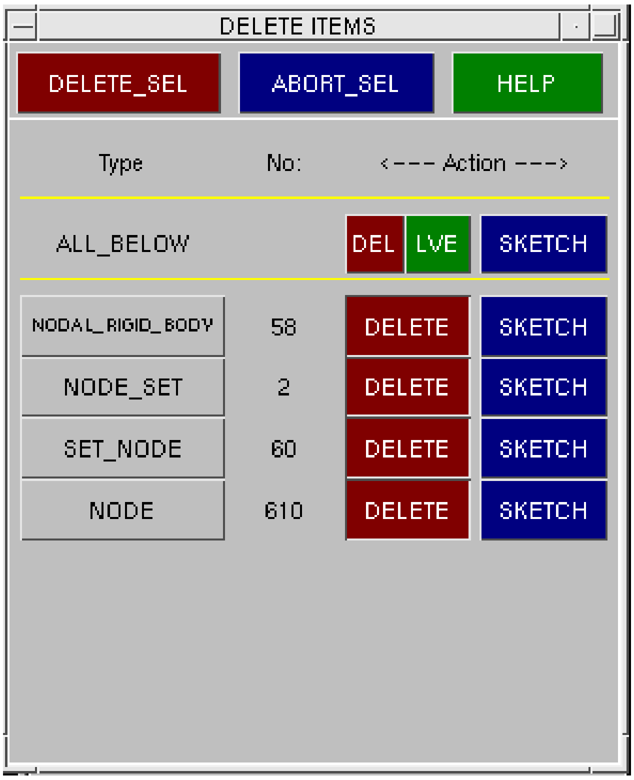
Rigid body merges, where the part to be rigidified is the slave, will be removed to avoid clashing with the merge onto the dummy master.

Spotweld beams will be flagged for removal where both sides of the weld are rigidified. Where one side of the weld remains deformable, the spotwelds will be retained, but a warning will be issued if the contact is not suitable for rigid parts.

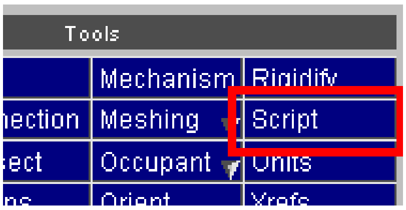
Contacts where all parts have been rigidified will be flagged for removal as they are superfluous.

The user has control over the deletion panel and may choose to leave the offending items. If deletion is aborted, a Primer model check may contain errors.

If **Simplify and delete** is selected, the elements will also be flagged for deletion.



6.27 SCRIPT Using Javascript in PRIMER



Scripting is covered in section 10 of this manual, see:

Using Javascript in PRIMER

A Brief Tutorial

The Javascript API Manual

6.28 SEAT-BELTS Fitting seatbelts and related elements.

The seatbelt fitting routines are designed to fit one or more belts to a structure, using **`shell`** and **`1d and/or 2d seatbelt`** elements. It is also possible to define the belt-related items such as **`retractors`**, **`sensors`**, etc.

While intended primarily for Occupant analyses, the "structure" used for belt fitting does not have to be a dummy: it can be any combination of solid, shell and thick shell elements, and belt fitting may be used in any context where a line of elements needs to be fitted to a mostly convex shape.

Quick links to seatbelt fitting commands:

CREATE	Creating a "structure" definition onto which to fit the belt
Fitting #1	Creating an initial belt "path"
Fitting #2	Fitting the belt "path" to the structure
MESHING	Creating a belt &/or shell element mesh on the fitted belt path
CONTACT	Creating a contact between belt and dummy (optional)
Related items	Creating retractors, sliprings, etc
Auto Refit	Automatically refitting the belt when the dummy moves
Saved belt data	What is saved in a "Belt" definition.

Four noded seatbelt shell elements

LS-DYNA release 971 introduces the concept of 4 noded "seatbelt shell" elements which have the special property that they can pass through slip-rings and retractors in their "shell" form.

These are implemented in LS-DYNA as follows:

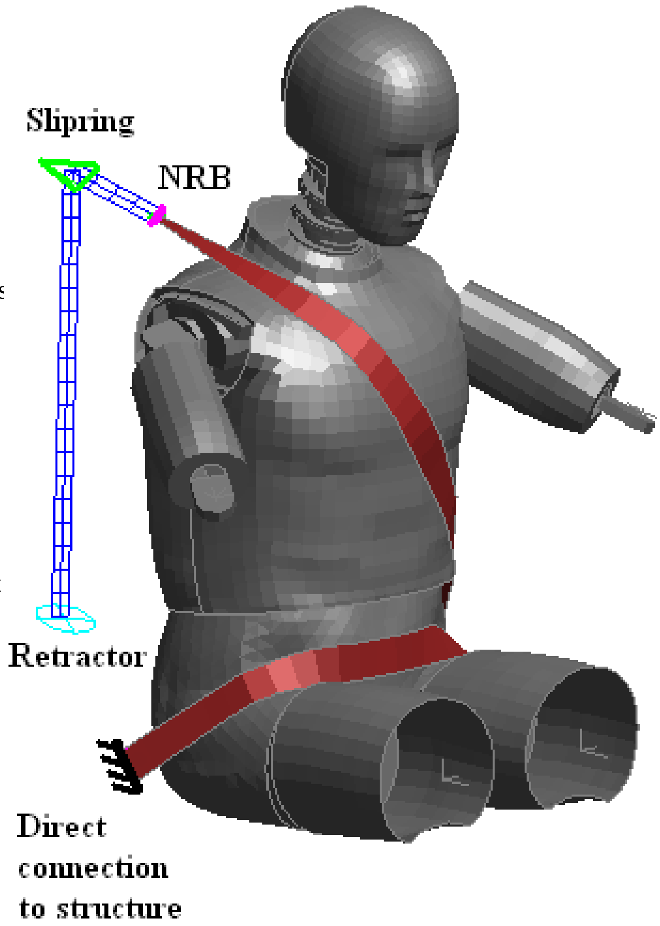
- Despite being entered on a *ELEMENT_SEATBELT card these are in fact SHELL elements, sharing the same label range as conventional shells with whose labels they must not clash. They should use *SECTION_SHELL but *MAT_SEATBELT cards.
- Sliprings and retractors have to be given information in an alternate form, with careful numbering of node and element sets.
- On input LS-DYNA automatically creates parallel rows of conventional (1D) seatbelt elements, sliprings and retractors which act in the normal way, thus carrying the seatbelt shell elements through themselves as "passengers" on the relevant nodes. The shells are used purely for contact.

PRIMER release 9.4 will create and fit belts using these 2D seatbelt shell elements, and is capable of building a belt definition from them alone or in combination with conventional shells. Mesh continuity through sliprings is preserved, and all the various sets that need to be defined in retractors, sliprings and section definitions are also created.

It is our observation that LS-DYNA 971 R4 or later is required for these 2D Seatbelt Shell elements to work correctly.

6.28.0 Description of Seatbelt Fitting

Traditional 1D Seatbelts + shells method.



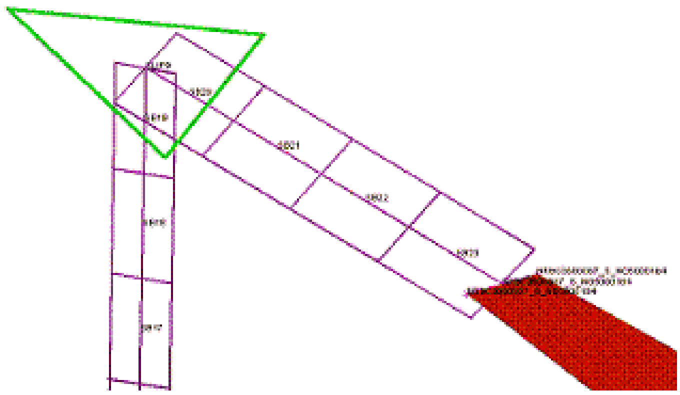
A "belt definition" in PRIMER has traditionally been made up out of a combination of **SHELL** and **SEATBELT** element types. Typically **SHELL** elements will be used where the belt touches the dummy, so as to give a realistic area of contact, and **SEATBELT** elements will be attached at the ends of each section so that they can interact with **RETRACTOR** and **SLIPRING** elements.

This figure shows a typical seatbelt definition using a traditional mixture of 1D seatbelt elements and shells

- **SHELL** elements (red) cover the regions where the belt crosses the dummy.
- A **SLIPRING** has been created behind the right shoulder. (There is another one at the left pelvis, not visible here.)
- A **RETRACTOR** has been created at the base of the B-Pillar behind the right pelvis.
- **1D SEATBELT** elements (blue) are used to connect to the slipring at the shoulder, (and at the left pelvis), and also for the whole section from slipring to retractor.
- Nodal rigid bodies (**NRB**) are required to connect the 1D seatbelt elements to the shells.
- At the belt end a direct connection to structure via a NRB if deformable, or extra nodes on rigid part if rigid.

PRIMER would permit the whole belt to be made of **1D SEATBELT** elements, but this would only be suitable for rigid dummies as these elements only give a "line" (zero width) contact with the dummy structure elements.

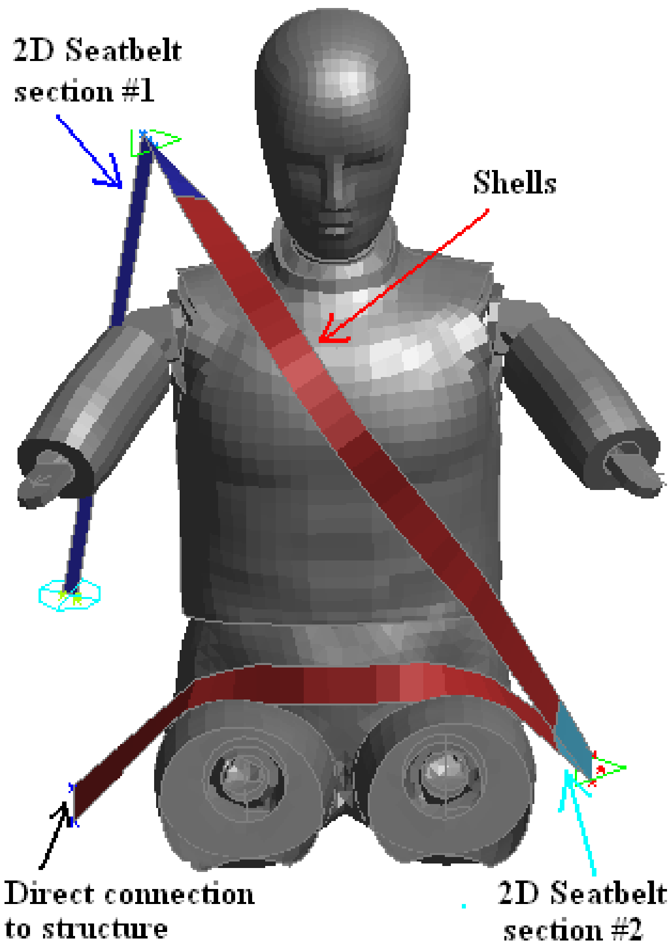
All stretches of **SHELL** elements are terminated with nodal rigid bodies or, where a shell connects directly to a rigid part, by making the shell end nodes "extra" on that rigid part.



This is to achieve **SHELL** to **SEATBELT** connections, as shown in this figure, and also to stabilise the end element if it connected directly to the structure.

This leads to the end shell being artificially stiff in a transverse direction, but this is acceptable in this context.

Alternative Shells + 2D Seatbelt shells method.



From PRIMER 9.4 onwards it is also possible to mesh belts using the 2D Seatbelt Shell elements available in LS-DYNA 971.

This image shows the same example as above with the 1D seatbelt elements replaced with 2D Seatbelt shells, allowing existing shell element properties to be preserved but better geometry to be achieved at slipring and retractor locations.

- Shell elements across the dummy are still used, in exactly the same way as above.
- The seatbelt shell mesh is continuous through sliprings and into retractors.
- The belt mesh is realistic in shape and orientation where it passes through sliprings, especially at the shoulder, making sensible contact between belt, dummy and structure at these points achievable.
- Nodal rigid bodies between Shell and Seatbelt elements are no longer required since the mesh is continuous.
- Mesh ends, whether seatbelt or ordinary shells, which connect directly to structure are still given Nodal Rigid Bodies or, if the structure is rigid, made "extra" nodes on that part.

Note that PRIMER generates new Part, Section and Material properties for each isolated section of 2D Seatbelt shell mesh. (Here blue from retractor to shoulder, light blue at left hand side of pelvis through slipring.)

This is because LS-DYNA requires 2D Seatbelt shell meshes to be continuous and it does not "track across" any intervening stretches of ordinary shells, so two separate definitions are required in this example.

Alternative fully 2D Seatbelt Shell method.

PRIMER is also able to generate a mesh that is wholly made up of 2D Seatbelt shells.

Since 2D seatbelt shells are actually genuine shell elements the result is visually indistiguishable from the mixed shell / 2D seatbelt shell result above, except that only a single part / section / material definition is used because the stretch of elements is continuous.

The full choice of meshing methods available

The choice of meshing method is up to the user. We anticipate that users with existing 1D seatbelt element + shell belt definitions wishing to achieve better contact behaviour at slipring and retractor locations will convert the 1D belt sections to 2D seatbelt shells, leaving the intervening shell sections unchanged in order to retain consistent behaviour. Users creating new belt definitions may wish to make them entirely from 2D belt elements.

PRIMER 9.4 permits existing *BELT definitions previously created in PRIMER to be refitted and/or remeshed in any of the following:

1. 1D seatbelt elements
2. 2D seatbelt elements
3. Conventional shells
4. Mixed 1D seatbelt elements + shells
5. Mixed 2D seatbelt elements + shells

Options (1) and (3) are unlikely to be used in practice, but they are provided for completeness.

Contact between belt and dummy.

In modern modelling practice it is likely that the belt will be included in a larger contact surface that includes dummy, vehicle, seat, steering wheel and - possibly - airbag.

However PRIMER can generate a separate contact between belt and dummy if required.

Two possible contact surfaces are offered

`AUTOMATIC_SURFACE_TO_SURFACE` For contact between belt shell elements, 2d seatbelt shells and the underlying structure.

`AUTOMATIC_NODES_TO_SURFACE` For contact between 1D seatbelt element nodes and the underlying structure.

Either or both may be omitted if unnecessary, and all parameters may be altered as required.

In PRIMER 9.4 the various fields on the contact all have their default values, with the exception of the "soft" option which is set to 1. Experience has shown that the relative stiffnesses of belt and dummy may be very different, and the use of this "soft constraint" option can improve contact behaviour.

Refitting a belt when the dummy moves.

If the dummy on which the belt has been fitted is subsequently moved, perhaps by Dummy or Mechanism positioning, then the belt can be automatically refitted to the new dummy position using the **AUTO_REFIT** function.

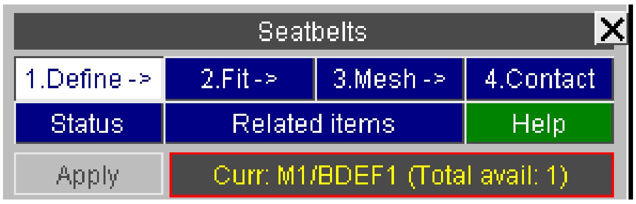
In most cases no user intervention is required and the belt, in its current form, is remeshed to fit the new dummy position.

Saving belt definitions: the *BELT keyword (after *END)

PRIMER permits any number of belt definitions to be defined in a model, (cf: Dummies and Airbag Folding definitions), and these are written out as a **`*BELT`** block of keywords after the **`*END`** card of an LS-DYNA deck so that they are not lost between PRIMER runs. This also means that if a dummy is moved it is possible to go back to the seatbelt definition and to refit the belt to the new dummy position.

(Go to next section.)

6.28.1 **Define:** Defining "Structure" for seatbelt fitting

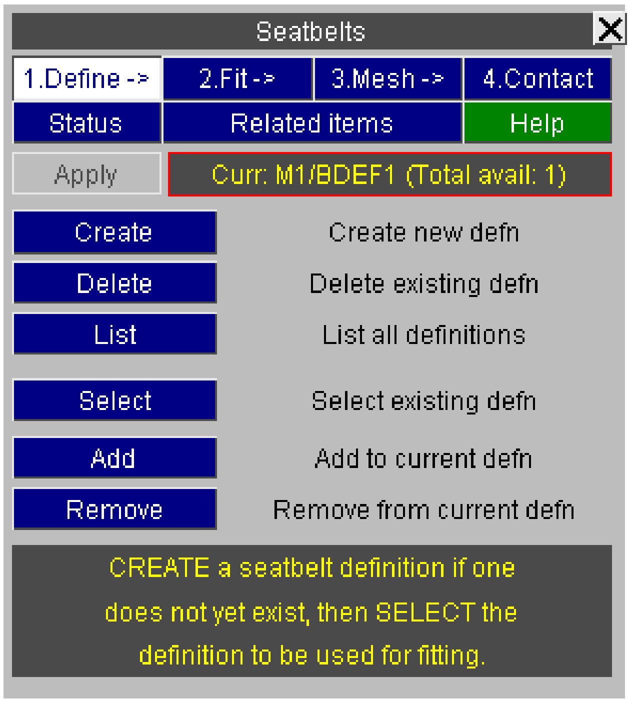


Belt elements are fitted directly onto "structure" elements, (it is not necessary to have pre-defined contact surfaces), so clearly you must have some structure to which to fit your belt definition!

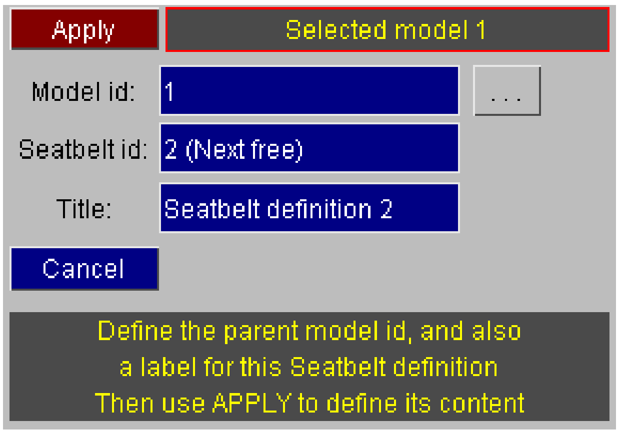
In PRIMER this may be any combination of `SHELL`, `SOLID` and `THICK SHELL` elements., the only limitation being that they must be in a single model. (If you have separate models, for instance #1 is a car and #2 a dummy, you will have to merge them before fitting a seatbelt.)

Before fitting can take place you must **SELECT** an existing belt definition (implicitly containing structure) or **CREATE** a new one. Selecting **Define** from the main seatbelt menu as shown on the right brings up the menu shown below.

CREATE	Manages the creation of a new seatbelt definition.
DELETE	Deletes existing definitions.
LIST	Lists existing definitions and their contents.
SELECT	Selects an existing definition and makes it the current one.
ADD	Edits the current definition by adding new elements.
REMOVE	Edits it by removing elements.
DONE	Exits this panel to return to the main seatbelt menu.



CREATE Creating a new belt definition.



You must first select the model in which the new definition will reside.

Then you must give a label and title for this new definition, and press **APPLY**.

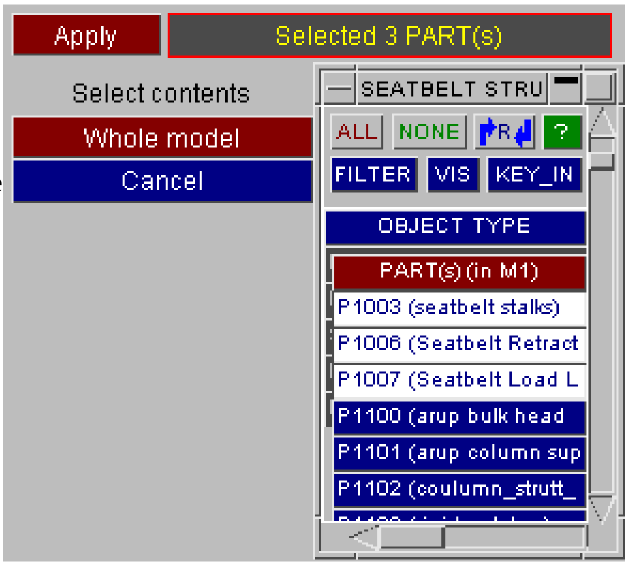
Labels are arbitrary, but must be unique within a model.

Then select the **SHELL**, **SOLID** and **THICK SHELL** elements that will constitute the structure for this definition.

PRIMER will create sets of these element types (**SET_SHELL**, etc) when you press **APPLY**.

Then **SELECT** this new definition to make it current, and use **DONE** to return to the top seatbelt menu.

(It doesn't matter if you select items which are outside these element categories, and indeed you can select the **WHOLE_MODEL** if you wish, since PRIMER will extract the relevant element types and ignore the rest.)



Efficient selection of structure elements

It is tempting, and quite legal, to select the whole model here; but you should consider the following:

1. These sets are used during the iterative belt form-finding process to create the pseudo-contacts against which the belt elements are fitted. The time taken in this process rises a linear function of the number of elements, so it is wasteful to select elements which will never be needed: for instance internal structure and elements on the rear facing sides.
2. These sets are also used to define the master side of the belt-to-dummy contacts for the actual analysis, so the same efficiency and speed considerations apply there.

The best structure definition is one which includes only those elements actually required for the fitting process and contact during subsequent analysis, and it is worth taking some trouble to achieve this - especially with large and complex dummies.

Once a structure has been defined the next step is to fit it (see section 6.28.2 for details).

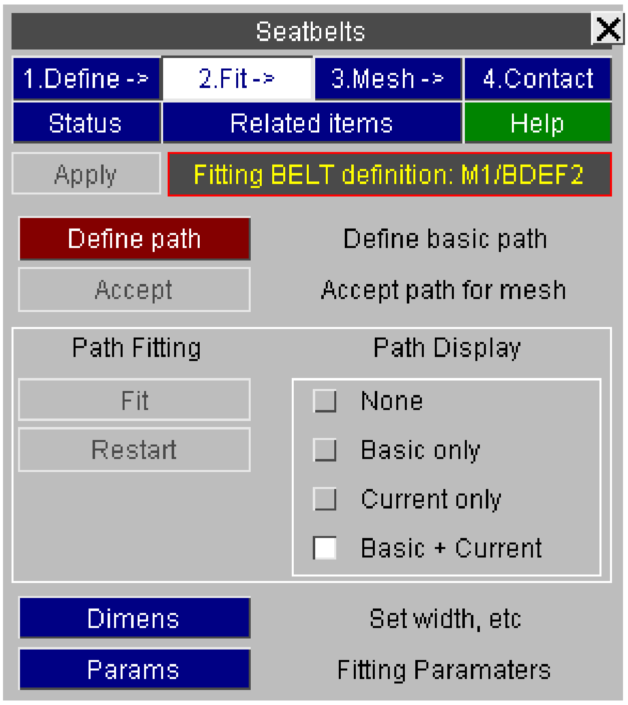
6.28.2 **Fit** #1: Creating a belt "path"

The second stage of seatbelt definition requires you to "fit" the belt by:

- defining a crude "path" for the belt;
- assigning geometric properties to that path (width, thickness, etc)
- assisting PRIMER in fitting the path to the structure using a form-finding process.

And hence obtaining an accurate geometry for the actual belt which can then be meshed.

The main **FITTING** panel is shown here in its initial state with no path defined. To define one the **Define Path** command is used.



What is a belt "path"?

A "path" is a connected set of 2 or more points which define an initial, crude line for the belt.

Each point is simply an [x,y,z] location in space, and has no structural significance: it is used solely to define an initial shape. Paths have the following rules:

- A path may contain any number of points, and by default all points except the two ends are free to move. Any intermediate point may be fixed, meaning form-finding won't move it.
- A path consists of one or more segments. A segment is a path section of at least 2 points, and is demarcated by its end points being fixed. Each segment is treated separately for both form-finding and subsequent meshing.
- If two adjacent points are fixed the path segment between them is assumed to be straight and will not move during fitting.
- At each unfixed point the adjacent lines must form an angle of > 90 degrees. If an acute angle is found the point will be fixed automatically and form a break between the segments on either side.

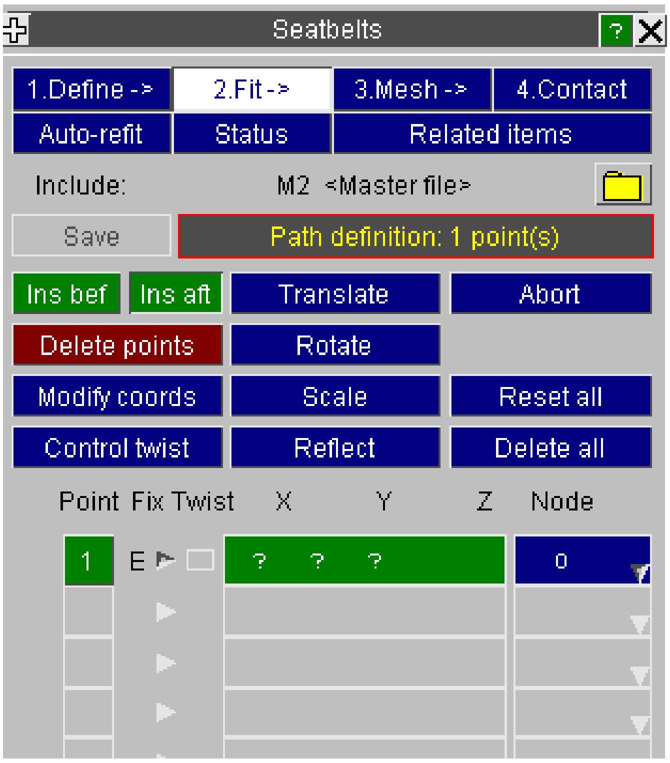
Path points may be have one of the following attributes:

The following attributes may be defined by the user	
`Known`	The belt is "known" to pass through this point, and this acts as a constraint upon form-finding which will make the centreline of the pass go through the point.
`Fixed`	A designated "fixed" point that demarcates the ends of any attached segments.
`Acute`	If an otherwise free intermediate path point has an acute angle (< 90 degrees) between adjacent segments then PRIMER automatically defines the point as "Acute". This implicitly fixes it and creates a break between the adjacent segments.
`End`	Both ends of the path unless a Retractor has been defined. "End" points are implicitly fixed.
`Retractor`	A retractor will be created at this point. This is only legal at path end points, and the presence of a retractor supersedes the automatic "End" definition.
`Slipring`	A slipring will be created at this point. This is only legal at intermediate path points, and a slipring implicitly "fixes" the point and supersedes any "Fixed" or "Acute" definition.

The following attributes are assigned automatically by PRIMER unless a user-defined attribute from the list above supersedes them.	
`Unfixed`	This is the default for intermediate path points with obtuse angles, which are free to move during form-finding.
`Acute`	If an otherwise free intermediate path point has an acute angle (< 90 degrees) between adjacent segments then PRIMER automatically defines the point as "Acute". This implicitly fixes it and creates a break between the adjacent segments.
`End`	Both ends of the path unless a Retractor has been defined. "End" points are implicitly fixed.

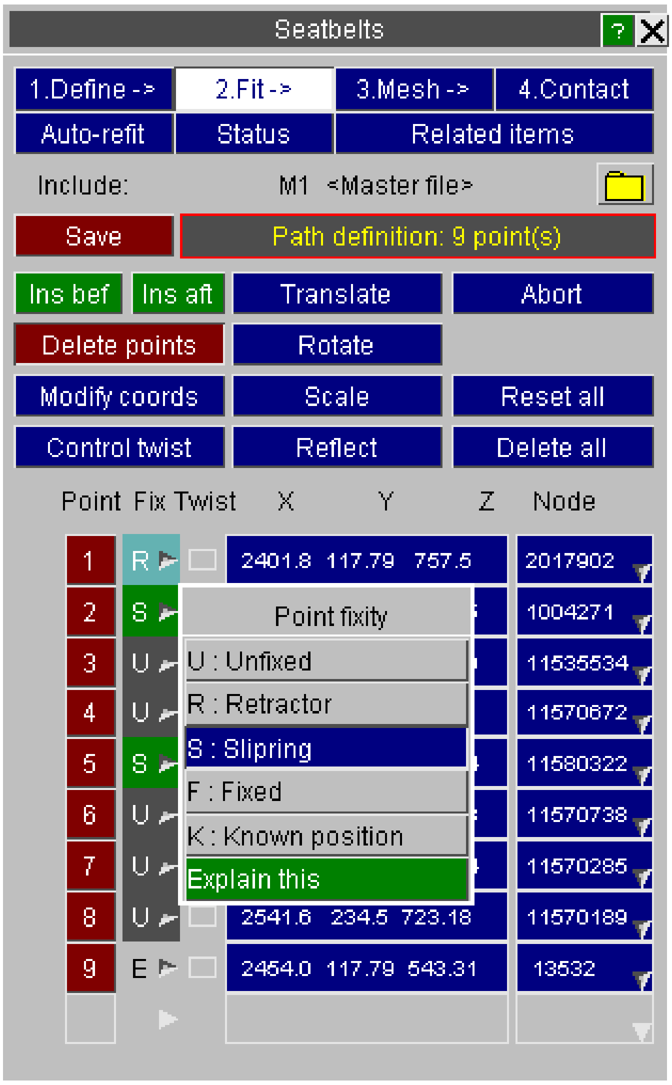
Creating a belt path

Belt paths are created in a simple text and visual editor, invoked by the **Define path** command in the **FITTING** menu.



The panel above shows the path editor in its initial, empty state before any path has been defined.

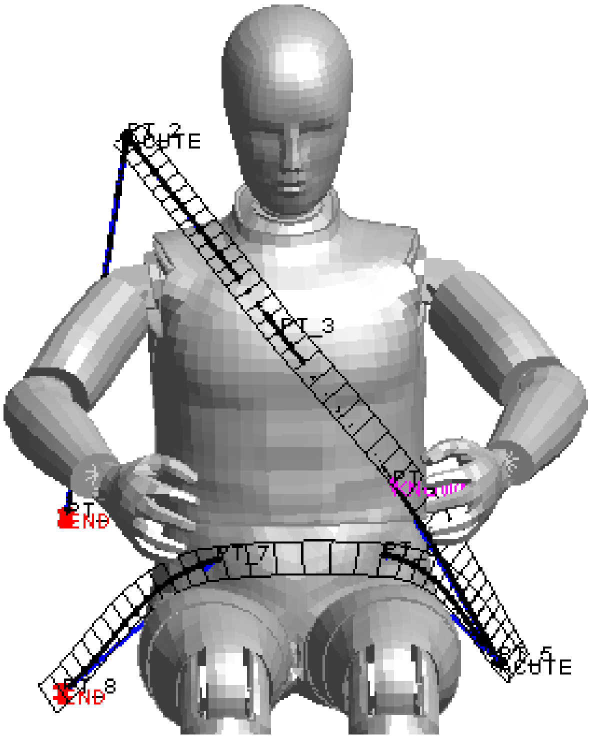
By far the easiest way to create a path is to click on nodes on the dummy and structure. The coordinates of the nodes are abstracted and become the path points.



Here some points have been defined, and the user has mapped the Point Fixity popup menu, which allows points to be set to one of:

U : Unfixed
R : Retractor
S : Slipring
F : Fixed
K : Known position

Defining the initial path



This figure shows a typical belt path round a dummy as initially defined. It has three segments:

1. Retractor to upper right shoulder slip-ring.
2. Upper right shoulder to left pelvis slipring.
3. Left pelvis slipring to anchor point behind right pelvis.

Note that:

- Only 8 points have been defined. As a general rule the best paths are those with the fewest points, and point 4 in this definition could be omitted . However the user has designated it as a "known" point.

- PRIMER has marked the "End" and "Acute" points automatically, and has inserted breaks into the path at the two acute points where sliprings will be defined later.

From PRIMER 9.4 onwards the "as projected" belt path is always shown at the basic path creation stage. Path projection takes place for any path segment with 3 or more points, and is in the outward normal direction determined from nearby structure or intrinsic path curvature, whichever is more relevant at that point.

The projected belt path also shows the approximate mesh that will be created. In this example only a single line of elements has been selected, but this can be altered at any stage.

Editor modes

The editor allows you to type in coordinates for points at any time: just click on the relevant box and (over-)type the coordinate value. But this is seldom convenient, and it is usually much easier to define coordinates by picking nodes, and to adjust them by dragging them with the mouse. This is accomplished as follows:

INSERT mode

This mode is used to create new points. You must define where in the path the new points are to be inserted, and you do this by clicking on the label button of the relevant point. The new point is inserted either before or after this button according to whether you are in **INSERT_BEFORE** or **INSERT_AFTER** mode. (If there are no existing points you will skip this stage and go straight to insertion at point #1.)

Once you have selected the insertion position that row will be filled with empty, green coordinate boxes and you can type in values. Alternatively you can use the cursor to pick nodes, and the coordinates of these will be inserted.

Note: The coordinates of the picked node are used, and the node itself is "remembered" for future reference during meshing or refitting.

It doesn't matter that nodal coordinates lie on the neutral axis locations of shells: the path is projected outwards prior to fitting, then pulled back to the shell surface, making the use of **AUTOMATIC_xxx** contact types possible.

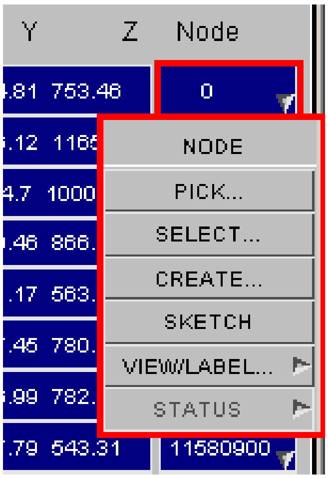
MODIFY mode

This mode permits point coordinates to be adjusted. You can do this in any permutation of the following three ways:

1. Use the popup menu in the "Node" column in the path editor to screen-pick or select a new node.

   This is the preferred method since the point will still have a node associated with it, which is valuable if a retractor or slipring are defined at that point since it determines clearly which (structural) node the element should be associated with. (Field SBRNID). It is also valuable at path end points where the belt is attached directly to structure since, again, it defines which node should be used.

   This is still true in the case of 2D belt meshes, as although sliprings, retractors and belt ends will all have at least two nodes nevertheless they can still be associated with the specific nodal location via a nodal rigid body or extra nodes on the parent rigid part.



1. Over-type the point coordinates with new values. The new coordinates will be applied immediately, but note that the associativity between the point and the node originally used to pick it will be lost.

2. Drag points visually. Click on a path point with the relevant button (see below) and, holding that button down, drag the point to its new location.

Left button	:	Drags in global	X
Middle button	:	Drags in global	Y
Right button	:	Drags in global	Z

   As with method 2 the association between point and its original definition node will be lost once its coordinate is updated.

In all cases both the visible path and the relevant coordinate in the editor will be updated dynamically so that you can see what is happening.

Control Twist mode

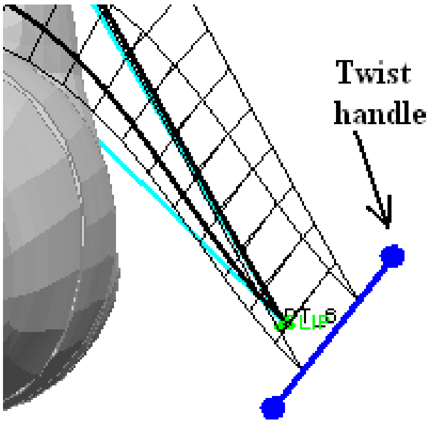
The orientation of the belt path is chosen initially by PRIMER to give what it believes to be the best fit. Broadly speaking:

- The outward normal of structure elements is used at intermediate points, and at end points the most recent normal is extrapolated to give a consistent end section.
- If no nearby structure elements are found then the path is treated as lying approximately on the surface of a sphere with its centre at the average (central) coordinate of the "structure" nodes.
- Special rules apply to straight sections between two fixed points, where a "best guess" at the wanted path is adopted.

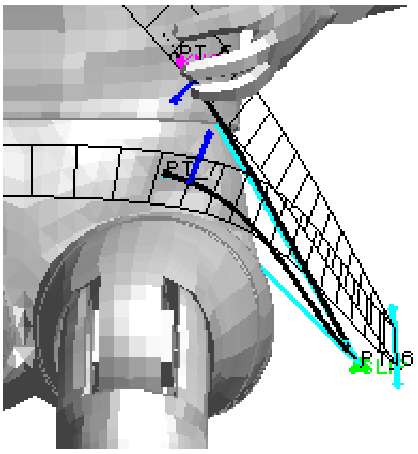
However in many cases these defaults are inadequate and it is necessary to adjust the twist of the belt path.

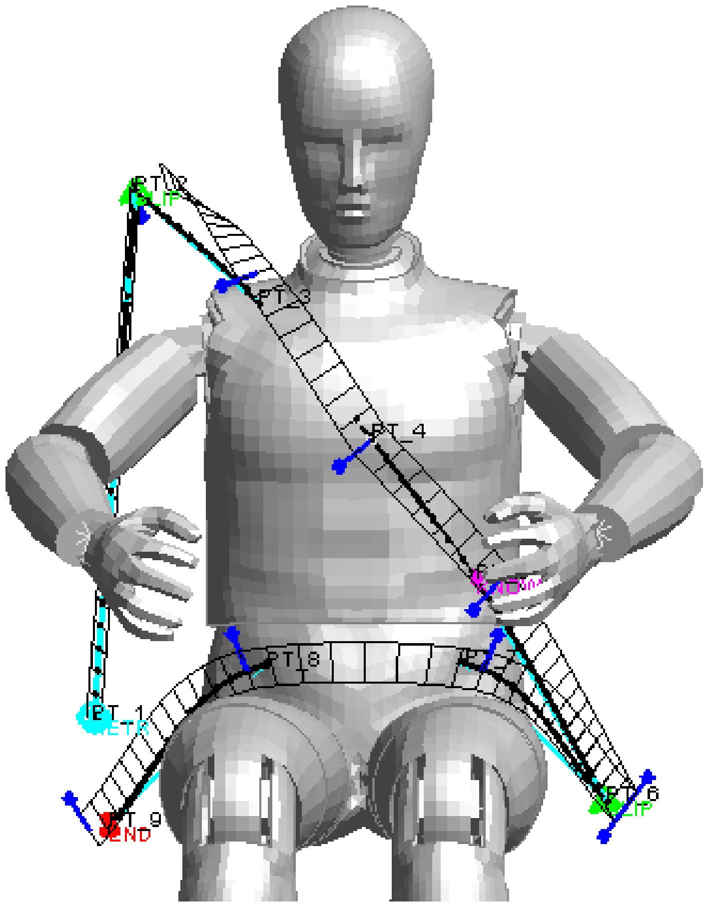
In "Control twist" mode each path point has a "twist handle" attached to it, and by clicking and dragging on that handle you can adjust the twist of the path at that point. The axis that is twisted depends on the mouse button used:

- The **Left** mouse button alters the **radial** (outwards) vector
- The **Middle** mouse button alters the **transverse** (rotation about radial) vector.

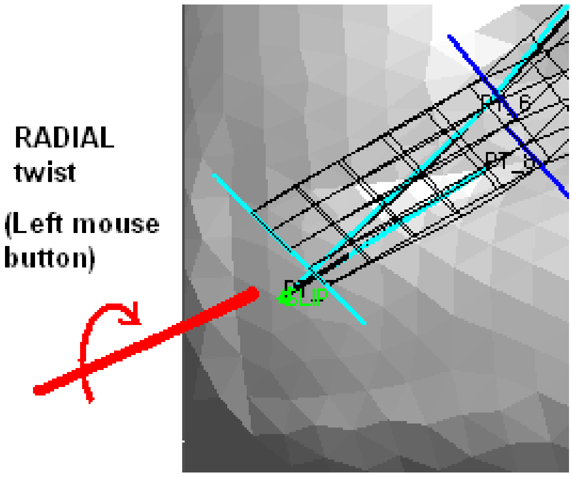


Here the path at the lower (pelvis) slipring has been rotated to
a more realistic angle by rotating its radial vector:



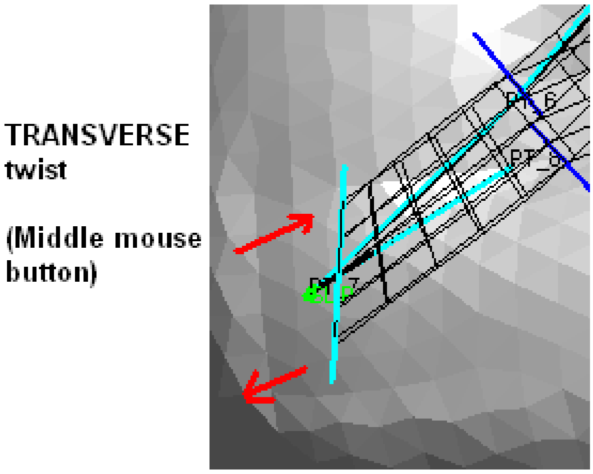


RADIAL twist, imposed by the left mouse button, changes the direction of the radial (outward) vector, effectively twisting it about the long axis of the belt at that point:



Radial twist can be applied at any path point.

TRANSVERSE twist, imposed by the middle mouse button, rotates the transverse mesh lines about the radial (outward) vector at that point.



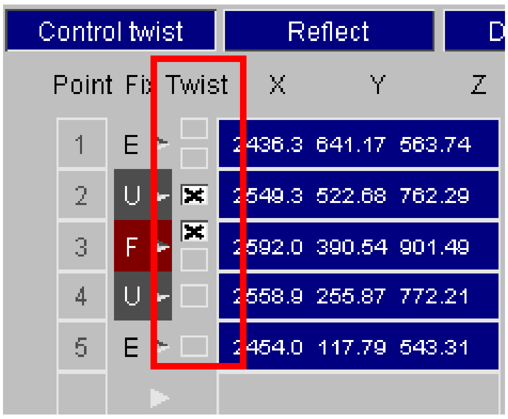
Transverse twist can only be applied at end points, retractor and slipring locations, and fixed points.

Once a point has been twisted away from its "natural" orientation PRIMER remembers this and shows it in two ways:

1. The colour of the twist handle on the plot changes from dark blue to light blue (compare the lower and upper images above).

2. In the path editor a cross [X] is shown in the "Twist" column, see points 2 and 3 in the image on the right.

The twist can be adjusted further at any time by dragging the handle to a new position, and it can be reset to its default by clicking on the [X] symbol in the path editor.



Fixed points and acute points will have two [X] buttons, see point #3 in this figure, and also two twist handles on the plot. This is because the path twist on either side is independently controllable at these points.

DELETE mode

This mode lets you delete points. Either press the label button of the point to be deleted, or click on the point with the mouse (left button). With either method it is deleted immediately.

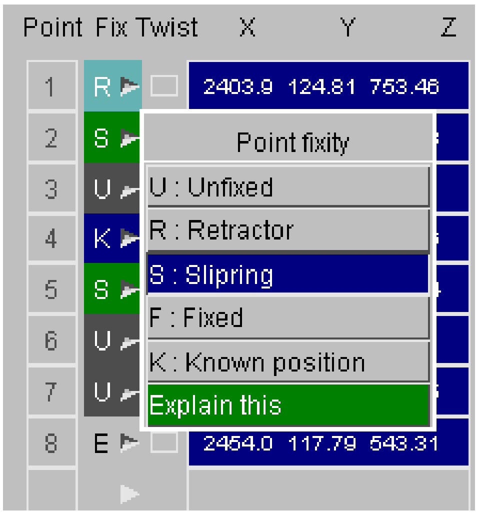
The following table lists the colours used both in the path editor and for the graphics in the various path editing modes.

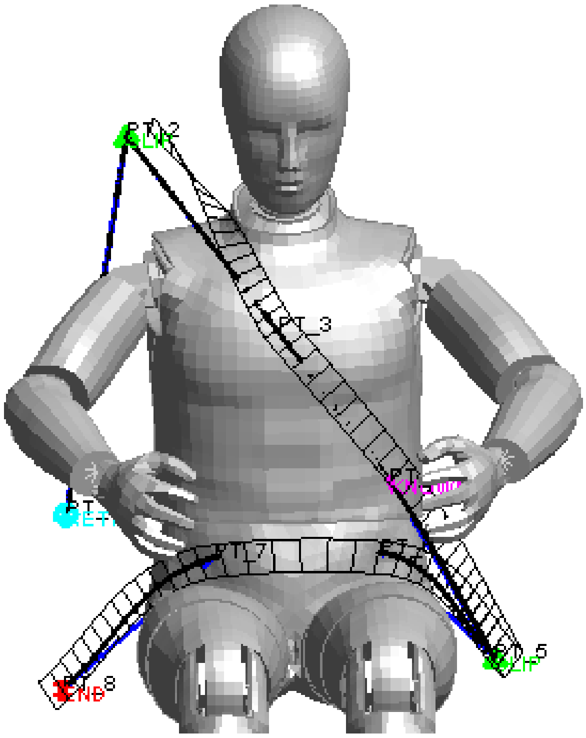
Mode	Path Colour	Cursor	Label buttons
INSERT	Green	Picks nodes for point coords	Choose point to insert before / after
MODIFY	Blue	Left: Modifies X coord Middle: " Y " Right: " Z "	<Not active>
TWIST	Cyan	Left on point pick handle twists path	<Not active>
DELETE	Red	Picks points to delete	Choose point to delete

Adding detail to the basic path

Adding sliprings and retractors

Using the popup menus against the "Fix" column in the path editor:





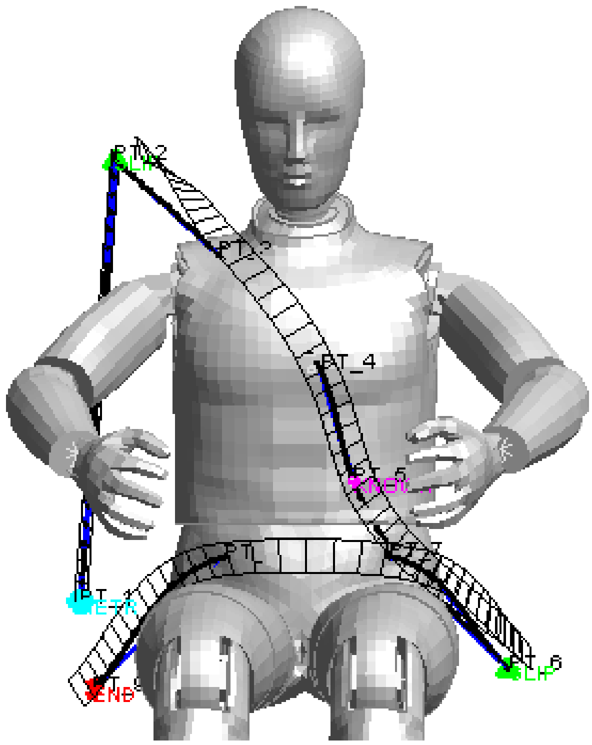
The user has now added:

- A Retractor at point one
- A Slipring at point 2
- A Slipring at point 5

Adding "Known" points

In this example the user wants to skew the path of the belt across the abdomen, so an extra point #5 has been added and marked as "Known" using the popup menu above.

During fitting this will force the path to pass through this point, even though it is not on the "natural" shape of the belt.

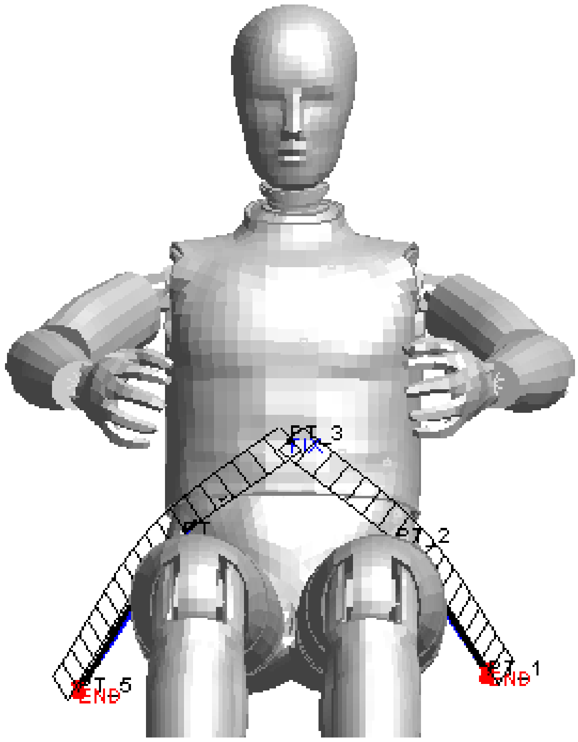


Adding "Fixed" points.

Fixed intermediate points effectively form a fixed end to the adjacent sections, and this would be unusual in the case of a normal lap and diagonal 3 point belt.

However, as this example shows, were you to create a full four or five point harness you would have to do this in two or three separate belt fitting operations, each meeting at a common point at the centre of the abdomen.

This picture shows how you might create the lap section with the fixed point #3 forming the buckle. The two shoulder belts would be added by a similar process, also meeting at the node used for point #3.

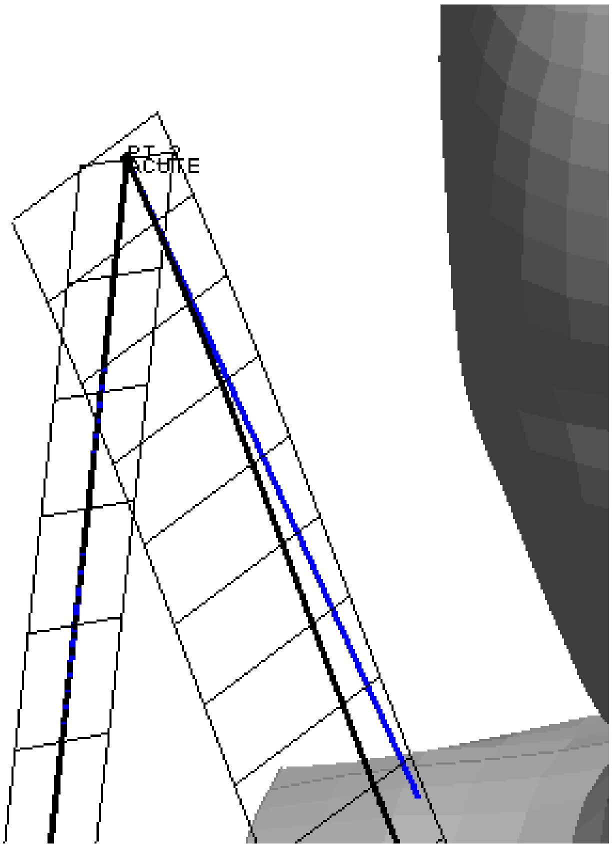


The effect of Sliprings on the belt path.

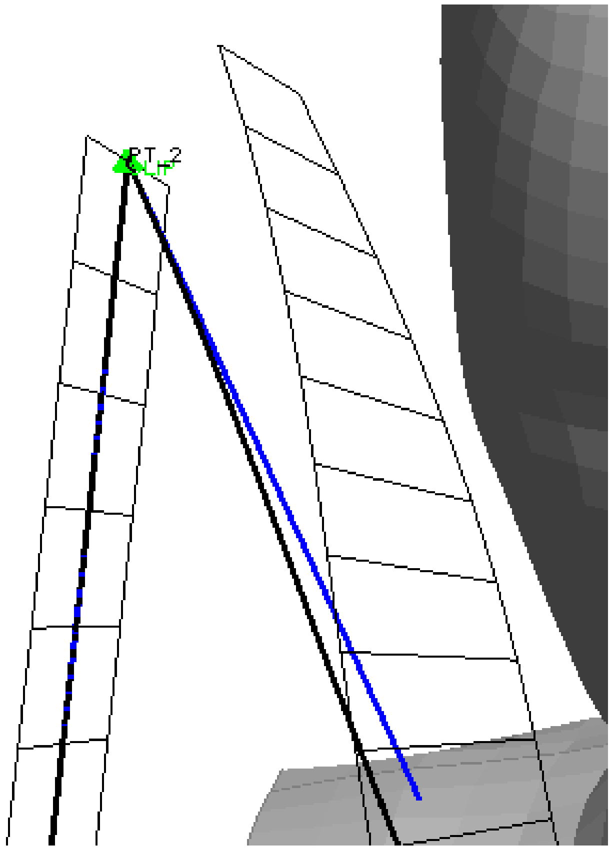
Now that PRIMER (from release 9.4) can fit 2D belts through sliprings it has become necessary to consider the orientation of

Slipring Case 1 : Shoulder location.

Note the difference between paths at the shoulder in the images above before and after a slipring was added, shown enlarged below.



Before slipring, simply an "acute" point.



After slipring defined, twist of free section has changed.

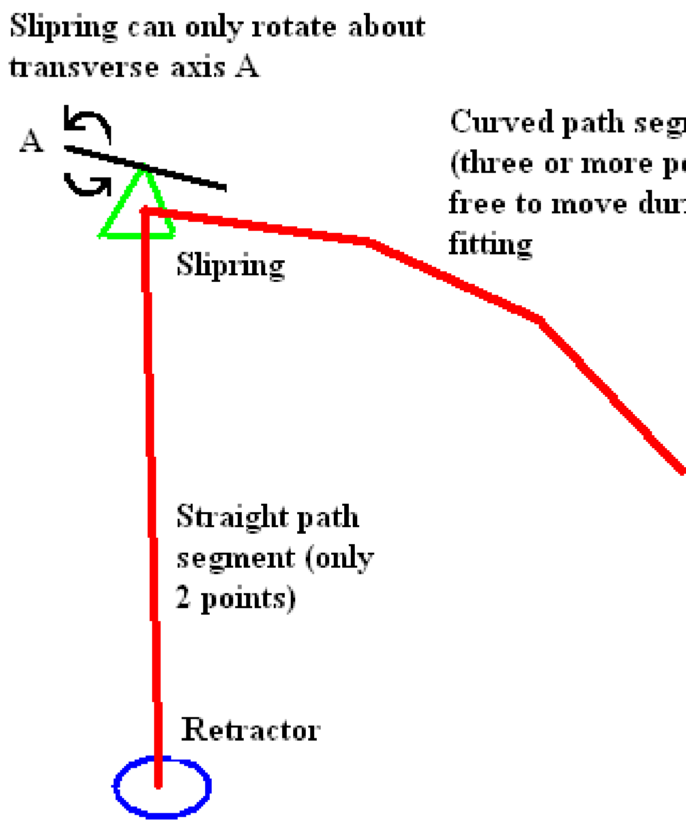
This example shows how the special case of a slipring at the shoulder location is treated. PRIMER assumes that:

- If the section on one side of a slipring has only two points, ie is straight and will not be moved during fitting.
- And the section on the other side has 3 or more points, ie is free to move.

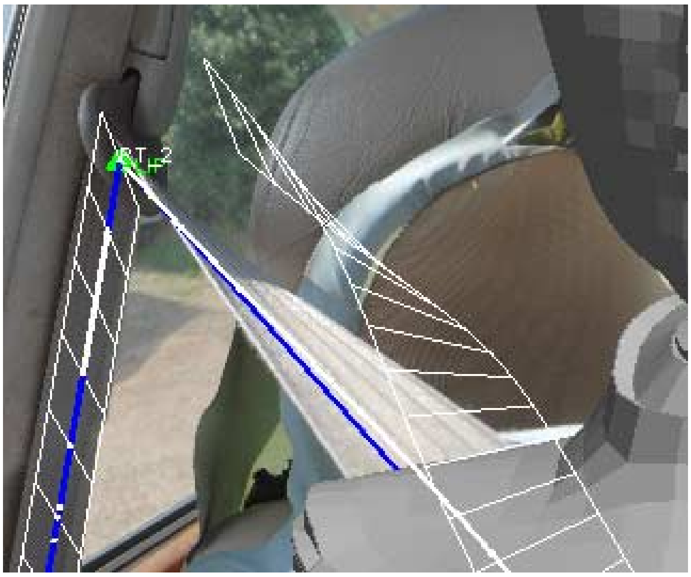
Then it is dealing with a shoulder slipring that is only free to rotate about (approximately) the horizontal axis across the vehicle, axis "A" in the diagram here.

This requires that the section of the belt going back over the shoulder towards the slipring must adopt a reverse twist in order to give the correct belt path through the slipring.

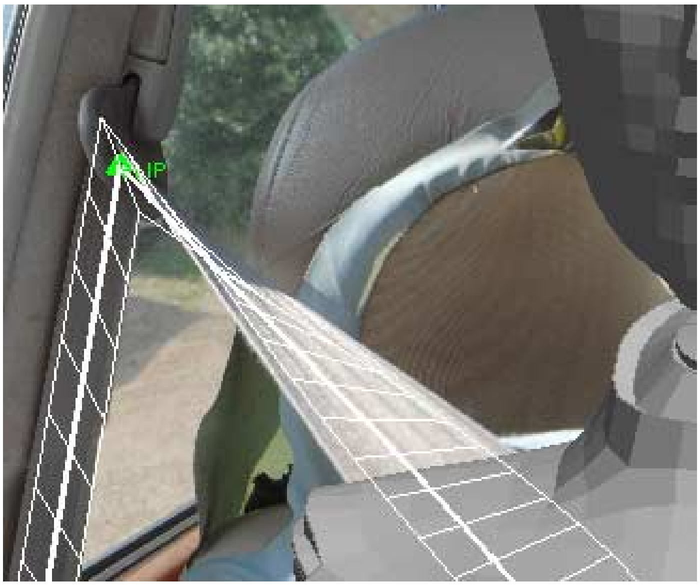
PRIMER treats axis "A" as being the outward normal of the straight section of belt between retractor and slipring, which poses a problem since it is just a line and lines have no orientation. Therefore PRIMER estimates what it believes will be a credible orientation for that section of the belt path, but it may require adjustment to its twist angle to achieve the correct angle at the slipring.



The following two images show seatbelt path fitting at the shoulder slipring in PRIMER superimposed on top of photographs of the s belt detail taken in a real vehicle, and they demonstrate how the twist of the belt passing over the shoulder has to rotate backwards to the angle at the slipring imposed by its limited rotation axis.



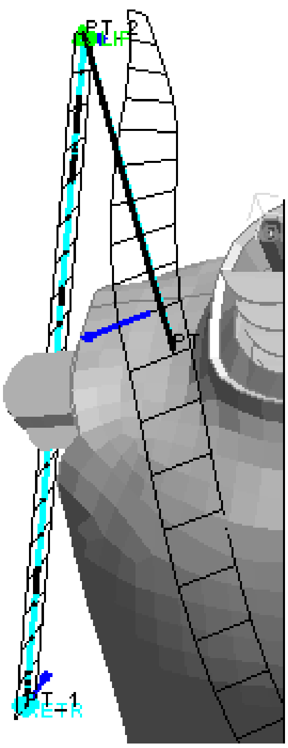
Before fitting, with free section of path projected above its final position



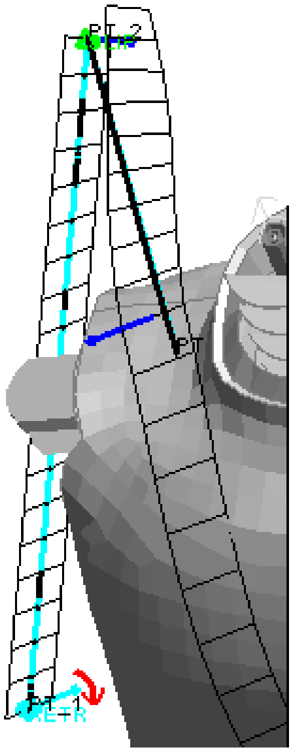
After fitting, showing the final shape of the belt path.

The orientation of the shoulder slipring is based on the outward normal of the straight vertical path from retractor to slipring, and since this is a straight line there is no "correct" value for this. PRIMER attempts to choose a default orientation that makes sense, but it may not always get it right making it necessary to adjust the twist of this straight section to achieve the correct geometry.

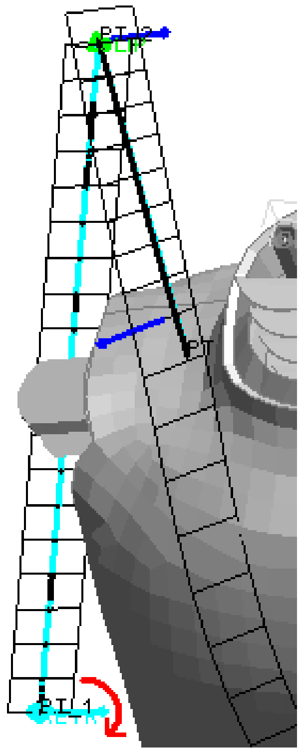
The following figure shows how altering the twist of the vertical section influences the shape of the belt at the shoulder slipring.



Default orientation, considerable twist at shoulder.



Belt rotated by about 30 degrees



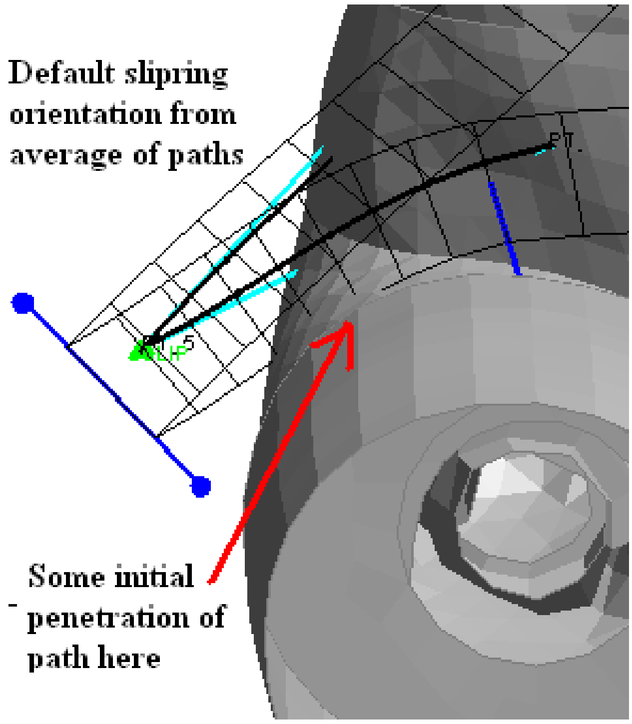
Belt rotated by about 80 degrees

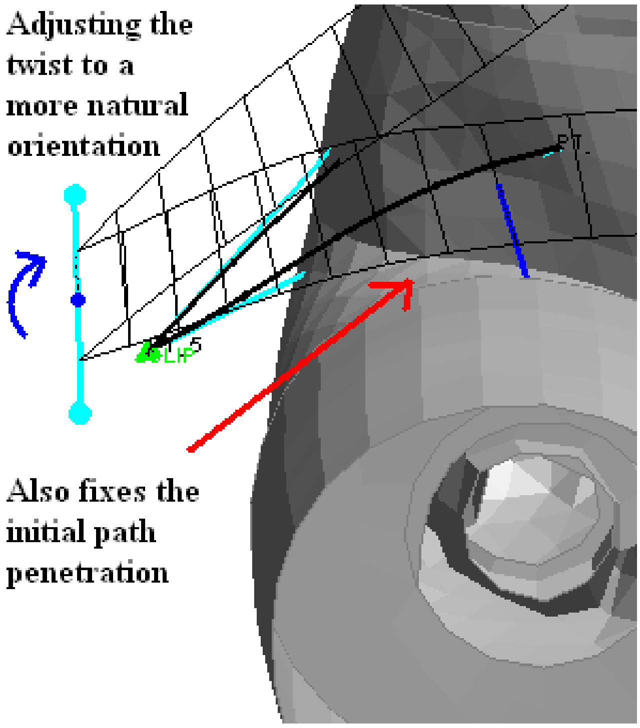
In this example the belt has been rotated at the base (retractor) location, but it could equally well have been rotated at the top (slipring) location. If an explicit twist is applied only to one point on a straight section of path then the whole section will rotate as shown here, however separate twists may be applied at each end in which case the path will twist between them.

Slipring Case 2 : Pelvis location

At the pelvis the slipring lies between two sections of belt that are both curved and thus free to move during fitting. Its orientation is unknown, and cannot be deduced from its location or the belt geometry, so instead of attempting to control the twist of the belt path it simply adopts the average twist angle and orientation of the two sections attached to it.

The following example illustrates how this slipring may affect the belt path, and how it can be adjusted to correct this.

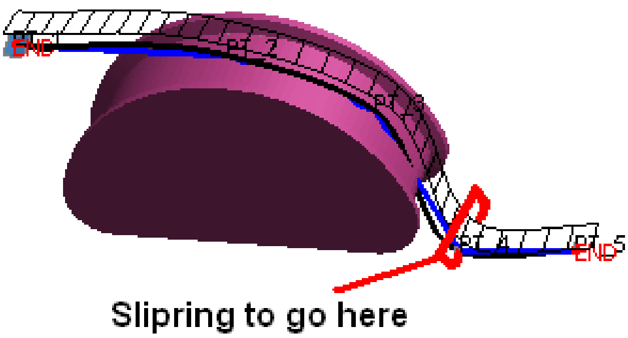


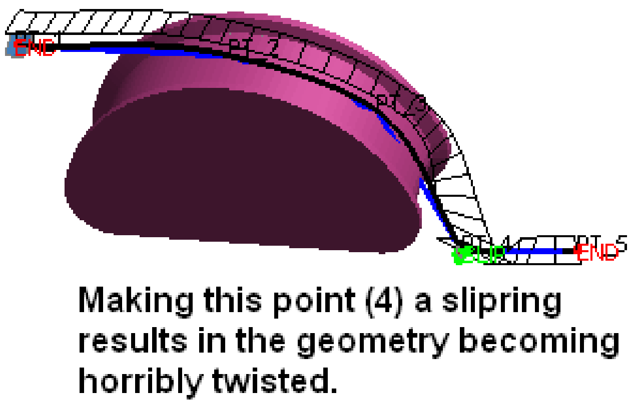


Slipring Case 3: Dealing with less common geometries.

In case 1 above PRIMER makes the assumption that a straight section of belt between fixed end (retractor) and slipring with no intermediate points must be a shoulder slipring, and it applies special rules to slipring geometry and location. This can cause problems if the same [fixed point ... slipring point] topology is encountered in other geometries, as the example below shows.

This is a simplified seatbelt pull-out test in which a belt passes over a platen and then through a slipring to a fixed anchorage point.

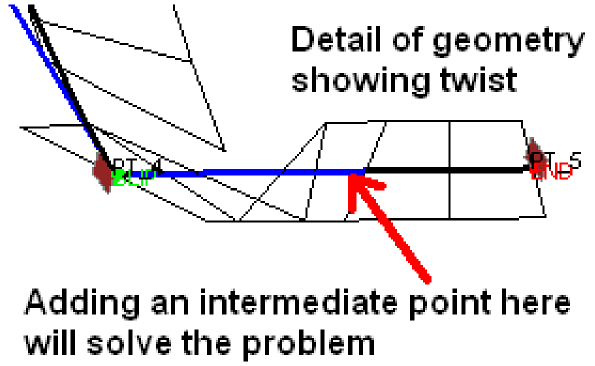


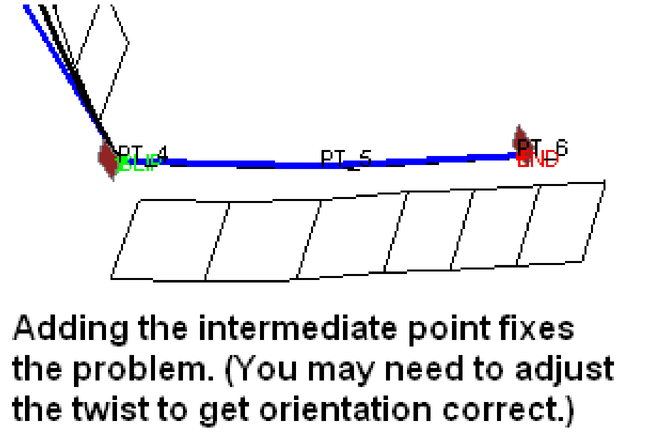


This image shows the initial path before the definition of any sliprings.

Addition of a slipring at point 4 results in the path becoming twisted because PRIMER has detected the straight path between point 4 and the fixed end at point 5, and treats this as a special "shoulder" slipring.

This logic fails because of the reversal of curvature in the belt path.



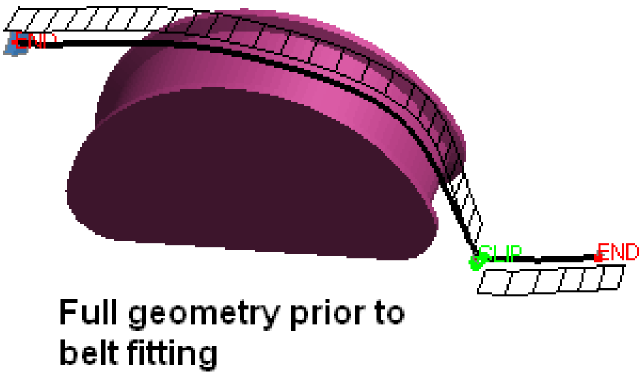


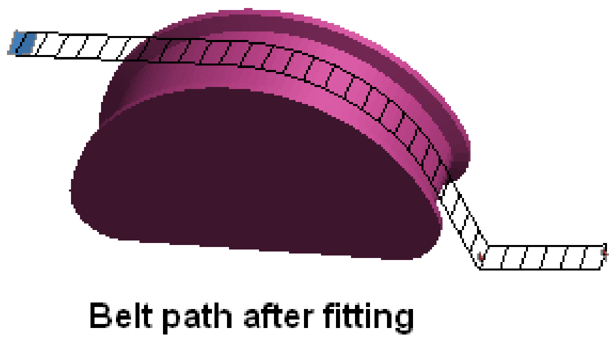
This image shows a detail of the slipring location, and also shows where an extra point needs to be added between existing points 4 and 5 so that PRIMER will not treat this as a "shoulder" slipring.

Here the extra point has been added resulting in the slipring reverting to a conventional one (as used in Case 2, Pelvis above).

Because of the reversal of curvature between the two belt sections the path, which is always on the outside radius of a curve, is below the line. This doesn't matter as the fitting process will pull both path sections together at the slipring location.

Also the twist of the path between points 4 and 6 had to be adjusted manually to line it up correctly since the curvature in this nearly straight section is ill-defined.

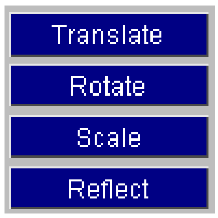




Here is the full model prior to fitting.

And here is the full model with fitting complete. Note how the two belt sections have come together at the slipring location.

Orienting the whole path

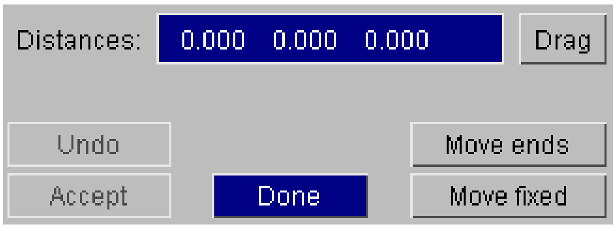


It can be convenient to move the whole path as a single entity, for example when re-fitting an existing belt path to a dummy definition that has moved.

Therefore it is possible to apply the standard PRIMER orientation functions to the points that make up the path.

The transformations are self-explanatory, and usage is similar to that of the main **ORIENT** commands, with the exception of the treatment of fixed and end points. The **TRANSLATE** variant is used in this example, but it applies equally to the other modes.

By default only unfixed points are moved, but you can choose to move end and/or fixed points as well.



MOVE_ENDS Allows end points to move

MOVE_FIXED Allows all fixed points to move.

Paths can be oriented by typing in transformations, or **DRAG**ged with the mouse. Dragging uses the same <*mouse button*> vs <*orientation axis*> arrangement as modifying single points: left button is X axis, middle Y and right Z.

Aborting path operations

CANCEL Cancels the current editing operation leaving the external (saved) path unchanged.

DELETE_ALL Deletes the scratch path inside the editor.

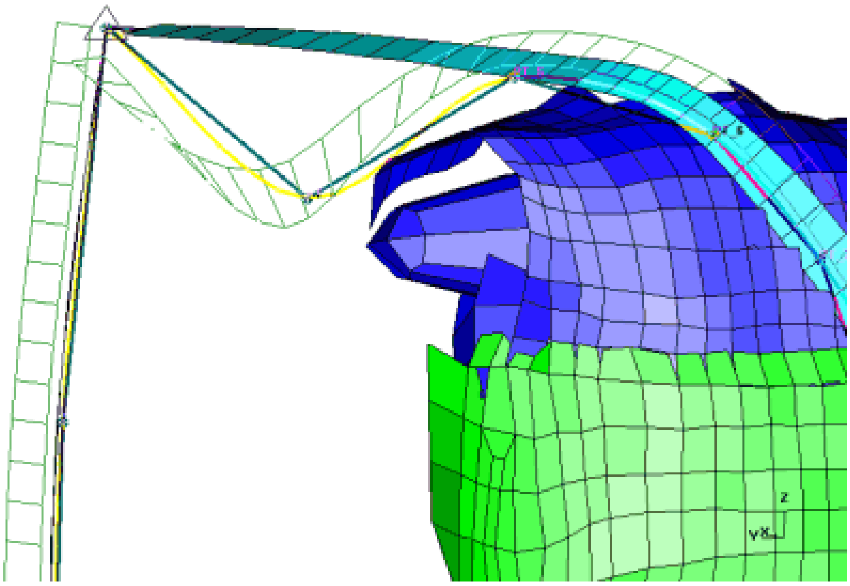
RESET_ALL Deletes both saved and scratch paths.

Hints on basic path definition.

Avoid reverse curvature

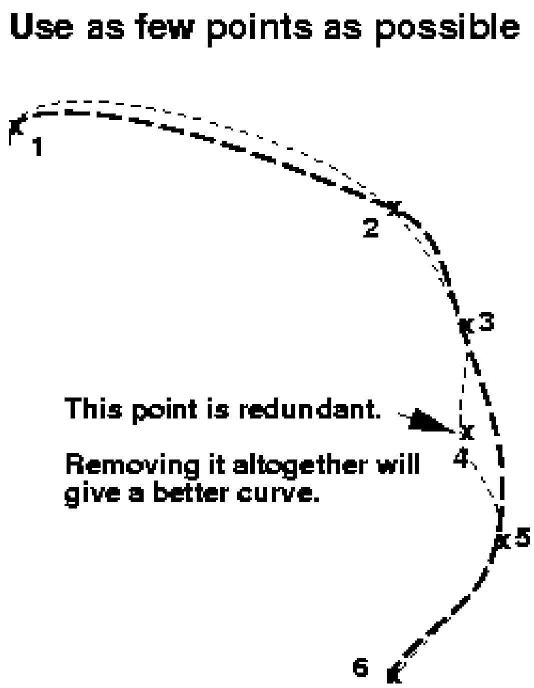
In this example an extra, redundant point has been added between the shoulder and the slip-ring, and it has been located such that it causes reverse curvature. (A correctly meshed belt is also shown for comparison.)

This shows that the chassis mesh gets twisted - almost through 180deg - which is clearly wrong, and while it could be corrected by adjusting the belt twist it is better to avoid it in the first place by sensible positioning of path points.

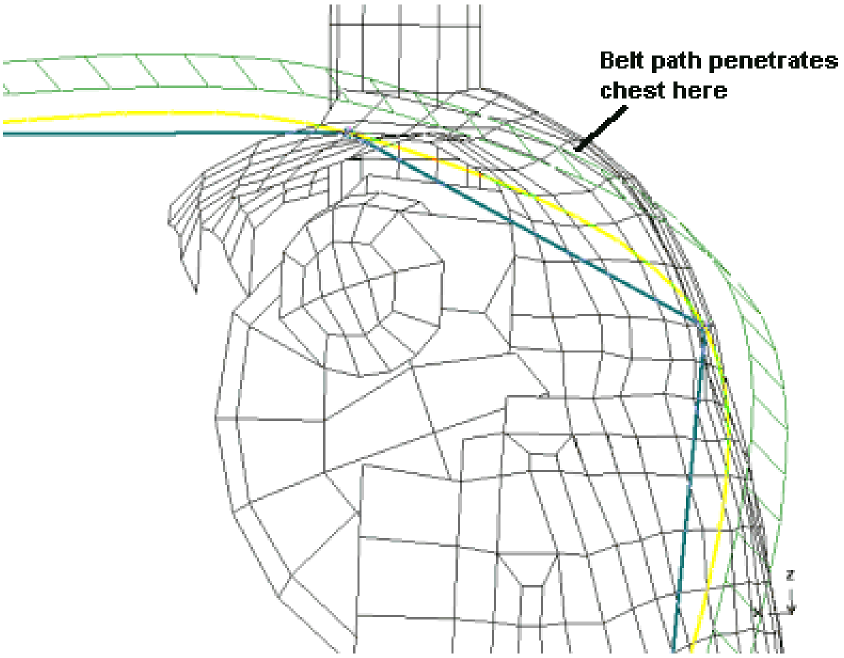


A common cause of this happening is the use of too many points when defining a path. In the example here point 4 is unnecessary: the dotted line shows how it causes reversal, whereas omitting it (long dashes) works OK.

Remember that an initial path does not have to follow dummy contours slavishly: it needs only to make a suitable initial shape for subsequent form-finding, thus a very simple shape will be adequate.



Choose as few points as possible for each path segment

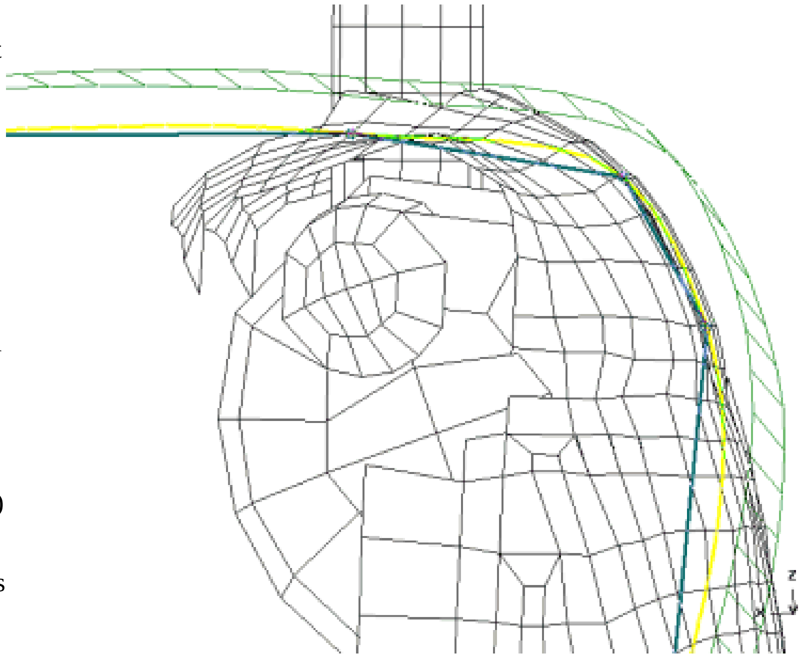


In this example the advice has been taken too literally!

A point has been chosen on the shoulder, and another on the abdomen, but the path between then fails to clear the bulge of the upper chest.

If this was fitted part of the belt would get stuck "inside" the chest shells.

The problem could be solved in two ways: either by increasing the initial distance by which the belt is projected outwards prior to fitting (which would be an inefficient solution), or by adding an extra point.



Here the problem has been solved by adding an extra point on the upper chest to force the belt path clear of the dummy.

(This is more efficient than increasing the projection distance since fewer form-finding iterations will be required to pull the path back into contact with the dummy.)

However the principle of using as few points as possible to define the belt path is correct as it will give a simpler and smoother path, as well as reducing the likelihood of reverse curvature. (As with approximating functions with polynomials, a lower order equation will give a smoother, if less precise, fit.)

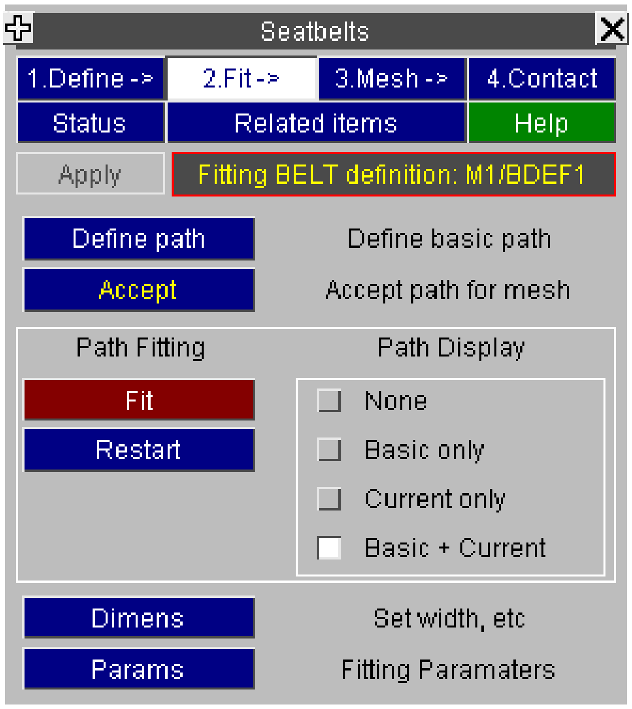
Remember that the initial path does not have to adhere slavishly to the dummy's contours: a simpler and looser shape will work just as well as it will be pulled back onto the dummy surface by the form-finding process.

Good basic path creation practice

Use as few points as possible.	Extra points are usually unnecessary, and can cause curvature problems: 4 or 5 per segment is plenty.
Basic path shape is not critical	The basic path does not need to follow the dummy shape slavishly: a "loose" initial fit giving good curvature will be quite satisfactory.

Once a basic path has been defined the next step is to fit it to the structure, see the next section.

6.28.3 Fit #2: Fitting the belt "path" to the structure

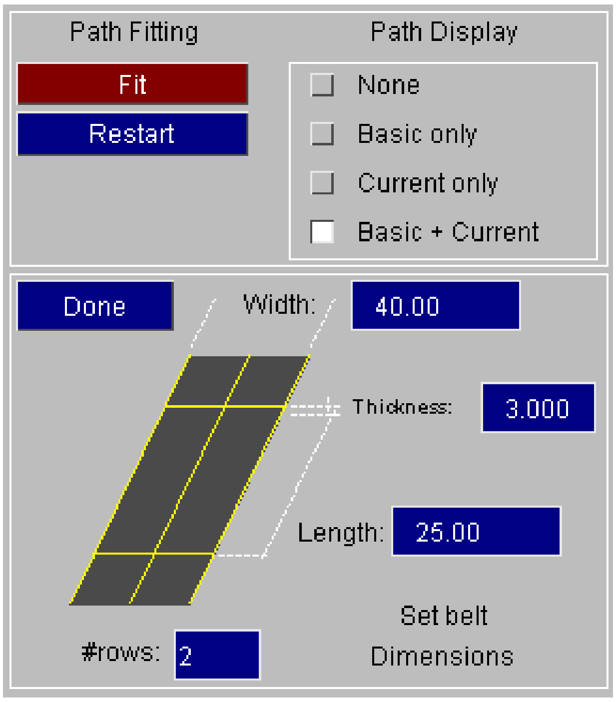


Once you have defined a basic "path", and hence an initial "chassis" mesh, you can proceed to fit it to the structure.

You may need to adjust both its **DIMENSIONS** (width, length, #rows, etc), and also the fitting **PARAMETERS** (convergence value, contact attributes, projection distance, etc) before you proceed.

Finally, once the fitting process is complete, you can **ACCEPT** the result and return to the main seatbelt definition menu to mesh it.

DIMENSIONS Setting width, length, etc



The dimensions of a belt are:

#rows: The number of (shell) elements across the width.

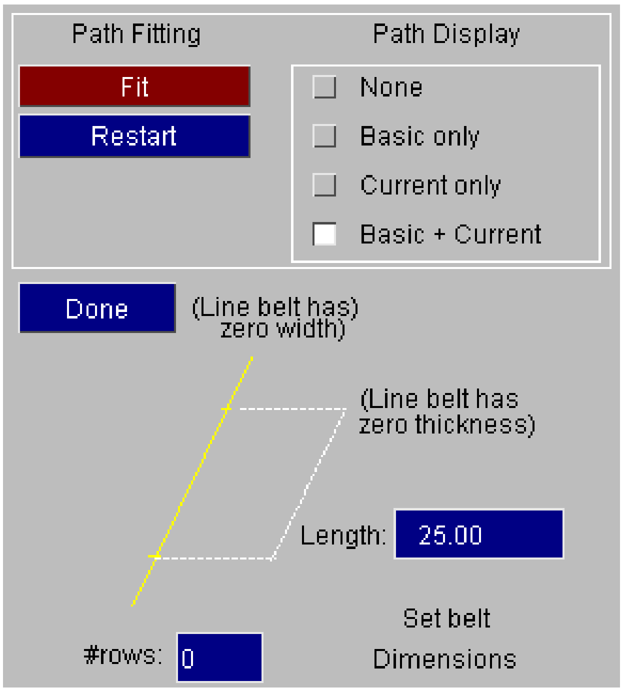
Width: The total width of the belt. (For **#rows**>0 each shell is **Width**/**#rows** wide.)

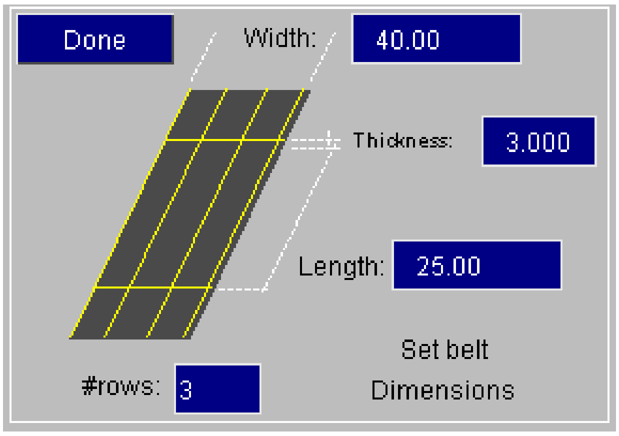
Thickness: Shell element thickness.

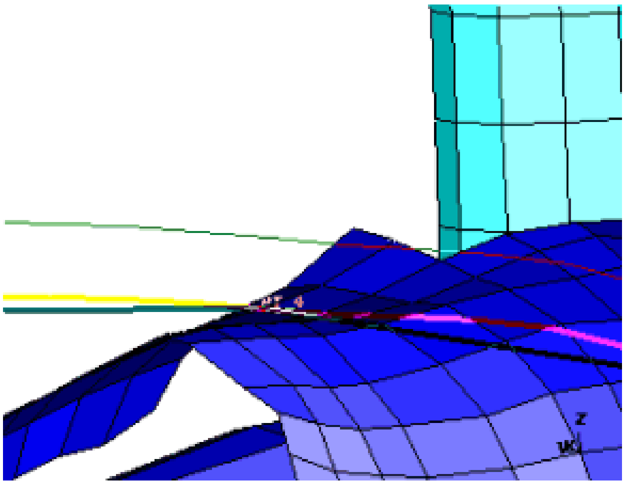
Length: Characteristic element length (both seatbelt and shell elements).

Clearly the first three refer only to belts that include shell elements. If **#rows**=0 then the belt becomes seatbelt elements only.

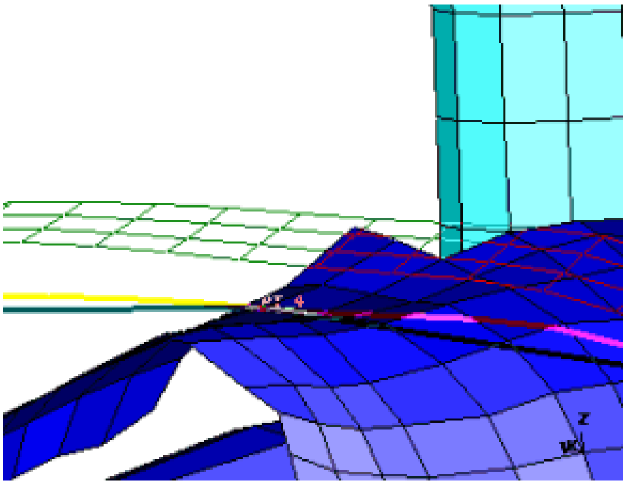
The following pairs of figures show alternative cases of no (seatbelt only) and 3 shell rows:







#rows=0: The seatbelt element only case



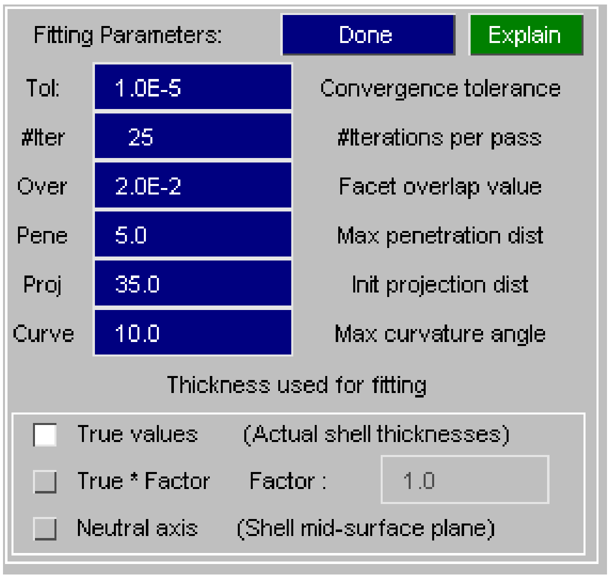
#rows=3: Three shells across width

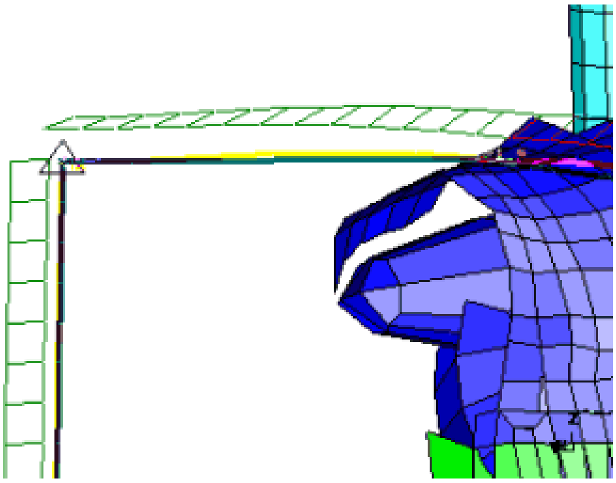
PARAMETERS #1 Basic control of the form-finding process.

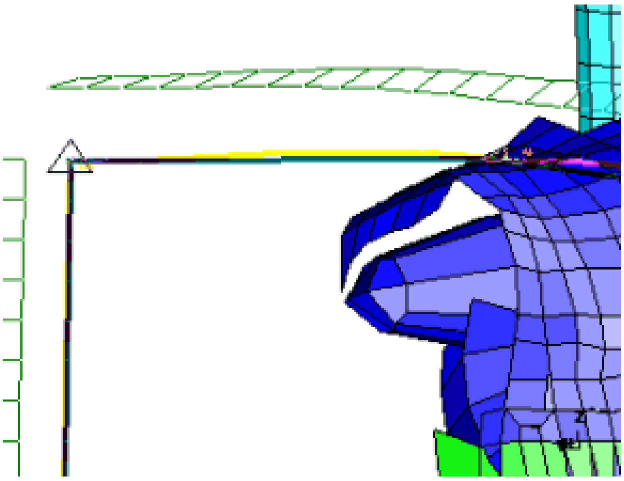
This panel contains controls for the form-finding process, and also the parameters for belt to structure contact that takes place during fitting. At this stage we will only consider:

Proj The "projection distance" which defines how far the chassis mesh is projected outwards prior to fitting.

The remaining controls will be discussed once the fitting process itself has been explained.







The **Proj**(ection) value is the distance by which the chassis mesh is lifted "outwards" from its defined position prior to fitting. The two examples above show the results of changing from the default of 25mm to 50mm.

There is no "correct" value to use, the criteria are:

- The distance must be sufficient to lift ***all*** chassis mesh segments clear of the underlying structure so that, especially in the case of shells, they start fitting on the correct side of the elements. (Remember that the basic path has almost certainly been defined using nodes on the element mid-planes.)
- The distance must not be so great that it causes the chassis mesh to interact with unrelated bits of structure, or distorts the initial shape so much that it doesn't get pulled back onto the structure correctly.
- It is uneconomical to use excessively large values since it will require many form-finding iterations just to close the gap with the structure.

Special treatment of fixed (segment end) points

Note that projection is applied to all points, *including* the fixed (segment end) ones and, as in the examples above, intermediate fixed points will have separate projections for each of the two adjacent segments.

Fixed points get exceptional treatment during form-finding: at these points the chassis mesh is only permitted to move along the "radius" vector back to the point so, ultimately, an end point should arrive back at its defined position - subject to any interaction with the structure.

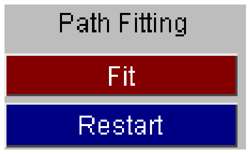
If structure intervenes at an intermediate segment end point causing the two segment ends not to be coincident following form-finding, the first segment will "win" at the meshing stage and the common end node will be at its location causing potential distortion of the second segment's end element. You should correct this situation by altering the segment end point to avoid such distortions.

Special treatment of "known" intermediate points

Intermediate "known" points are also treated specially. In effect there is a strong force pulling them back to their "known" position during the form-finding operation, and this is stronger than the force which tries to straighten the belt path at that point.

Movement of these points is not as constrained as that of "fixed" ones, but the effect is very similar.

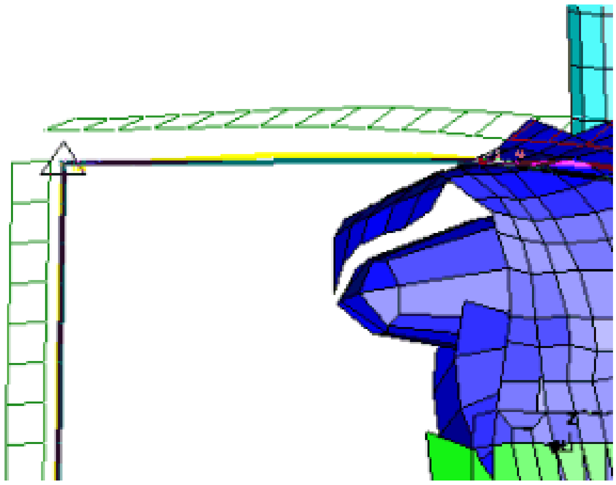
FIT: Commencing the form-finding operation

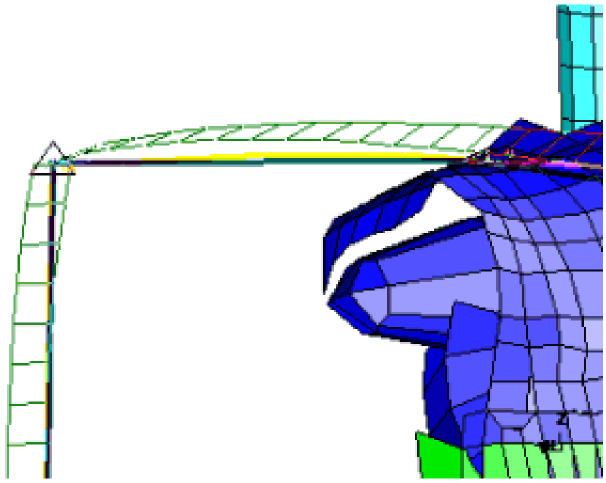


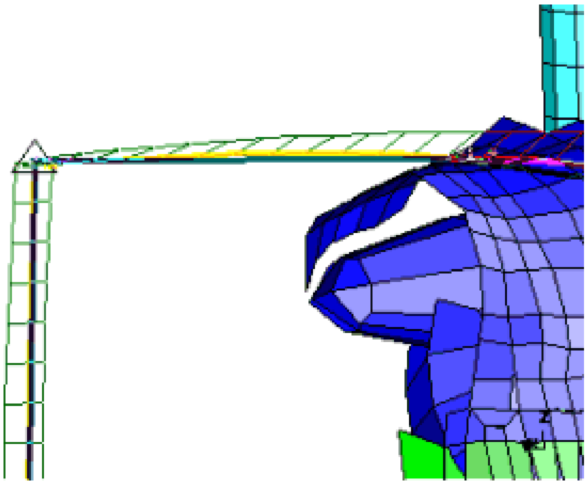
Once the chassis mesh, dimensions and basic parameters are defined you can initiate the form-finding process with the **FIT** command. This loops through steps (1) to (4) as follows:

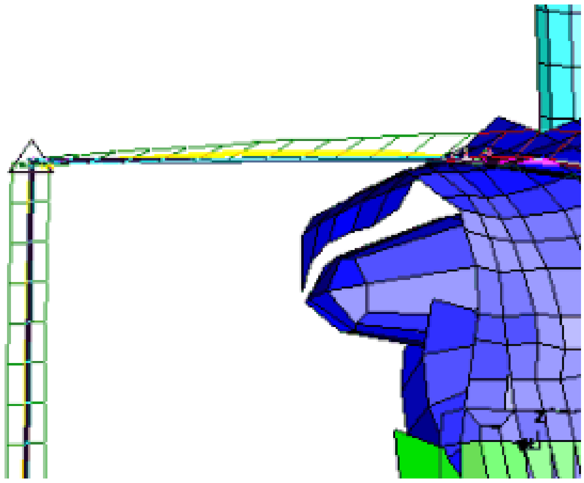
1. For each point on the chassis mesh a new position is computed that is closer to the dummy structure, and the point is moved to that location.
2. Contact between each point and dummy is checked at this new location, and the point is pushed out again if necessary to prevent any penetrations.
3. Convergence is checked by comparing the current chassis mesh with that at the previous iteration, and form-finding halts if the change is less than the specified convergence tolerance.
4. Form-finding also halts after the user-defined number of iterations so that you can check progress periodically.

The following is a typical sequence showing fitting from initial position to final convergence: between 100 and 200 iterations is typical for most belts.


Fitting #0: At initial position


Fitting #1: After 50 iterations


Fitting #2: After 100 iterations


Fitting #1: Converged at 131 iterations

When fitting halts, due either to (3) or (4) above, you can do any of:

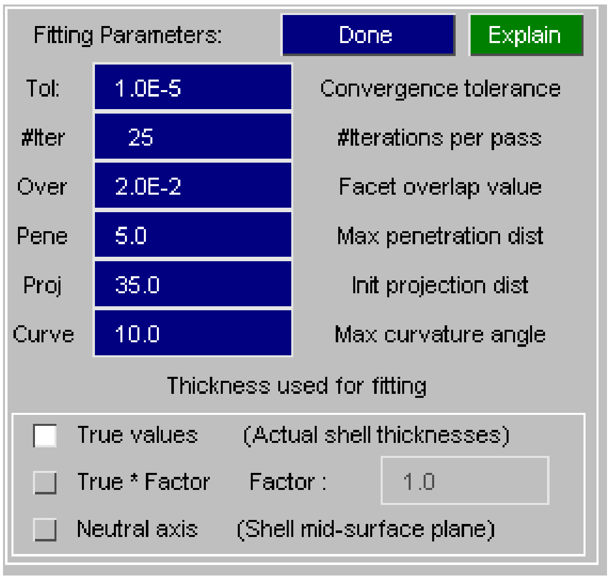
- Continue the process from where it stopped with no changes by pressing **FIT** again. It will be necessary to do this several times to get the belt to converge when the number of iterations required is larger than the periodic halt (`#iter`) value.
- Adjust some **PARAMETERS**, then continue from where it stopped by pressing **FIT** again. But make sure that the changed parameters are valid when applied to the current, partially fitted position. If they aren't start again

using **RESTART**.

- Adjust **PARAMETERS** and/or **DIMENSIONS** and restart from the initial projected-out position, using the revised values, by pressing **RESTART**.

- Or if you are happy with the converged shape press **ACCEPT** which will:
  - Save the current path, dimensions and fitting parameters permanently in the current seatbelt definition. Prior to this point you were working with a scratch copy.
  - From the main seatbelt menu you can go ahead and mesh the result with actual Finite Elements.

PARAMETERS #2 More about controlling the form-finding process.



Now that the form-finding process itself has been described we can revisit the **PARAMETERS** panel to consider some of its more obscure options.

`Tol` Is the tolerance used to decide when form-finding has converged. The smaller this value the more precise the final shape will be.

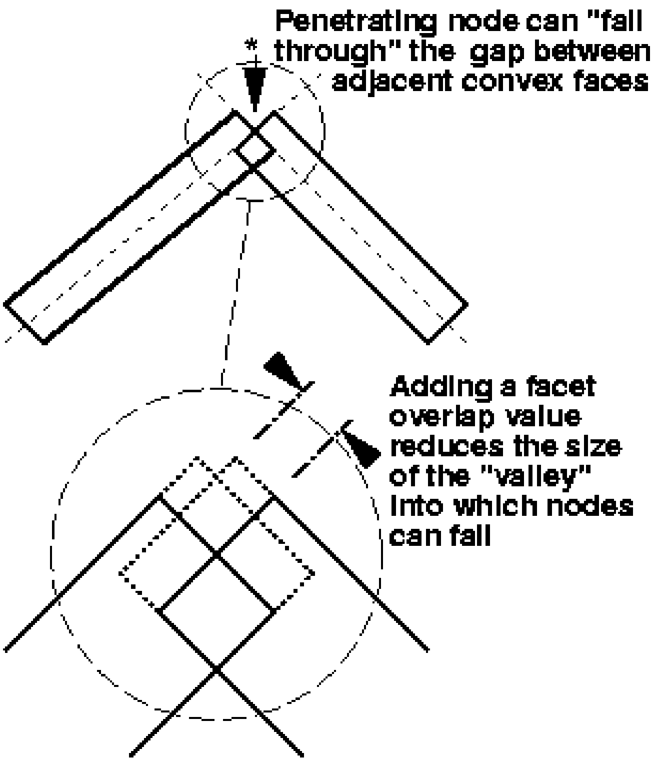
#iter Is the number of iterations that will take place before form-finding pauses to resort the "buckets" used to determine belt to structure contact. The default of 25 iterations is normally adequate, but if you have an extremely fine structure mesh and unwanted penetrations are occurring reducing this value may help.

`Over` Is the facet overlap value used during contact. It increases the size of structure element facets by (`1.0 +` **Over**) to try to stop nodes falling through the gaps of curved surfaces.

As this figure shows there is a "valley" between adjacent facets on a convex surface into which penetrating nodes can fall.

Artificially increasing the facet size for contact purposes, dotted area, reduces the size of this valley and therefore helps to prevent incorrect penetration.

The default value of 2e-3, (ie the facet dimension is factored by 1.002), works well for most cases; but larger values may be needed if penetration occurs.



Pene Is the maximum distance behind a 3D (solid or thick shell) facet at which penetration is considered.

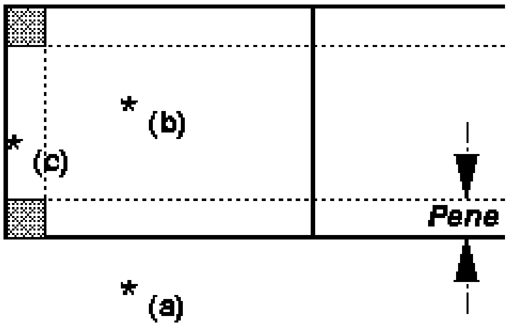
In this figure the **Pene** distance is shown by the dotted line, and point (b) is therefore too far inside the solid elements to be considered for contact.

Problems can also occur in the shaded areas inside the element corners since a penetrating point can only be ejected from one face, and this may not be the one through which it penetrated.

There is no easy solution to this problem save adjusting the belt path to move the offending point away from the element corner - and this may prove to be a problem during the analysis too. Good (structure) meshing practice would avoid sharp mesh corners where contact occurs by chamfering the edge.

Usually the **Pene** value only needs adjusting when a belt is fitted to a region containing layers of thin 3D elements, when it may need setting to half the thinnest 3D element dimension.

Contact with facets on 3D elements



Point (a) is outside and will not be considered.

Point (b) is inside, and its distance from an external face is greater than *Pene*, so it too is not considered.

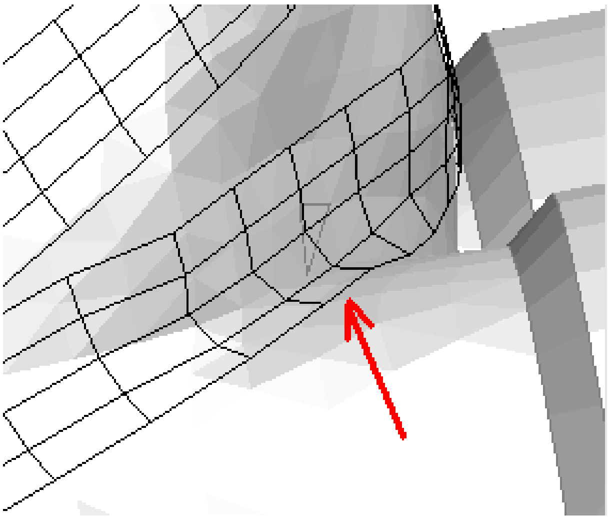
Point (c) is inside the *Pene* distance and will be moved to the external surface.

Curve is the maximum permitted transverse curvature of the belt

By default this is zero, meaning unconstrained, but if a value is set then this sets the limiting angle (in degrees) between transversely adjacent belt facets. This can be useful to stop the belt "creasing" down into sharply concave areas of the dummy.

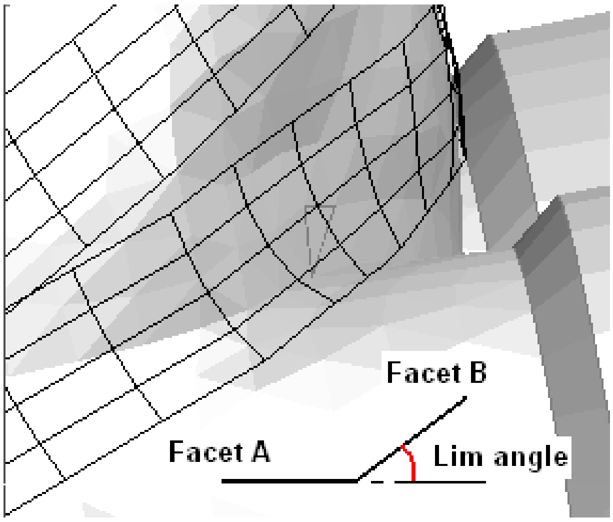
The examples below both show the belt mesh in the region where the lower torso meets the upper leg, in which typical dummies have a concave angle of approximately 90 degrees.

In this example no **Curve** value is set, and the belt creases sharply in the concave region (arrowed). This can sometimes give rise to poor belt element behaviour during the analysis.



Curve = 0 degrees (default)

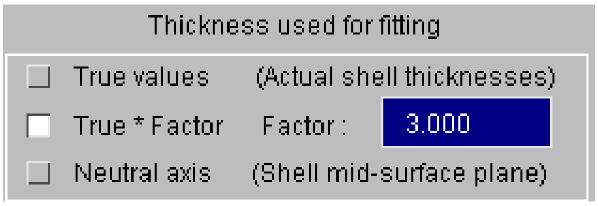
This example is the same in all respects, except that a **Curve** value of 10 degrees has been used. It can be seen that the belt is now "stiff" in the transverse direction and has not creased down into the concave region of the dummy.



Curve = 10 degrees

Thickness used for fitting

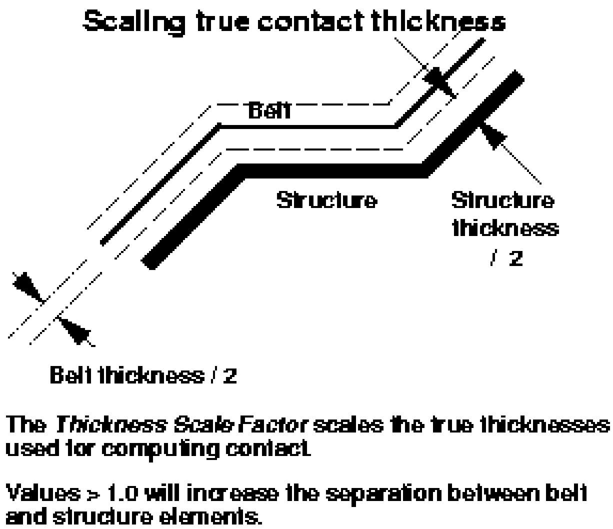
By default the chassis mesh is pulled back onto the structure such that the true external surfaces of belt and structure elements just meet.

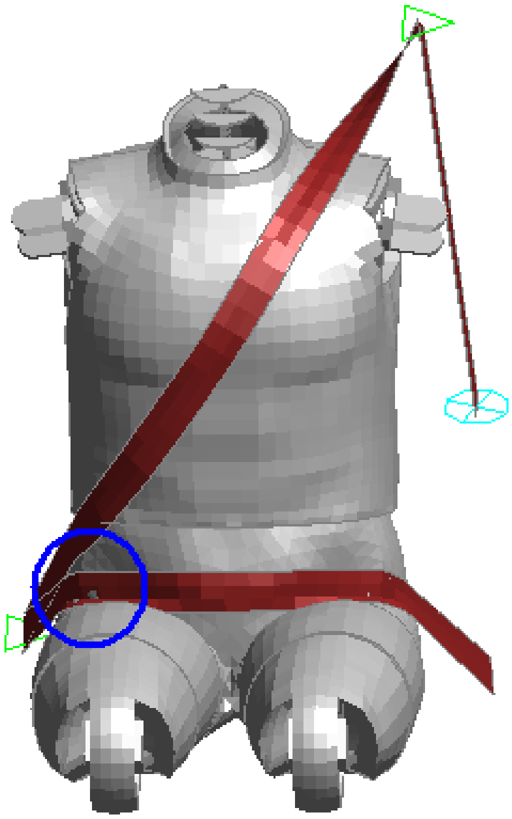


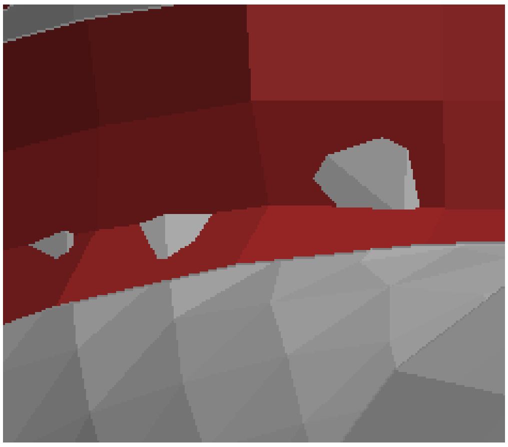
The **Thickness** factor scales the true element thickness values allowing you to increase the separation between belt and structure.

This can be useful both for visual purposes and for optimising contact during the actual analysis.

The images below (contrived artificially) show how scaling contact thickness might be used to prevent initial penetrations or - more likely - crossed edges during initialisation.

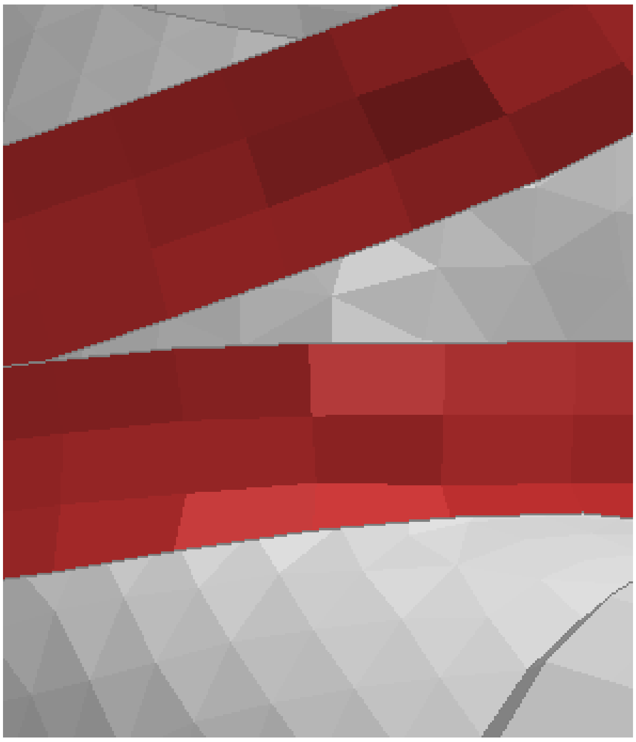






This image shows the regions of crossed edges in detail.

In this example the belt path has been fitted very tightly, using an unrealistically small contact thickness, leading to some crossed edges in regions of very "bumpy" mesh.



Here the belt has been refitted using a factor of 1.5x the true contact thickness, and the crossed edges in that region have been eliminated.

Detecting and solving problems during fitting

The path gets stuck on the "wrong" side of some structure.

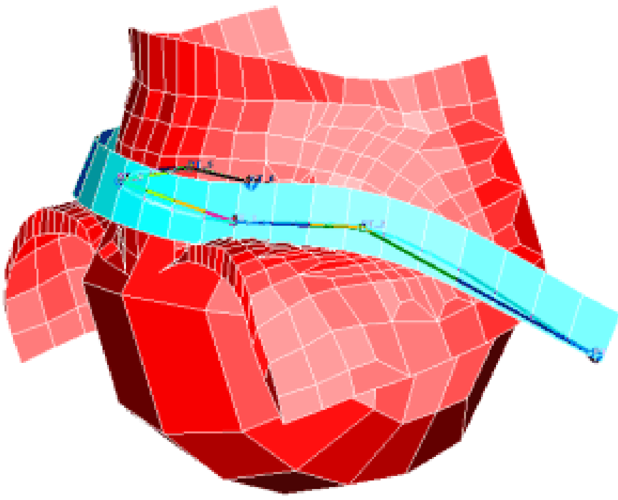
This is usually caused by the basic path not being defined correctly and having initial penetrations. It can be solved by moving path points or possibly adding new ones - see the section on "Defining the initial path" for more advice.

It is also sometimes caused because the route the belt takes when being pulled in is not the same as the outwards projection, and this leads to it getting snagged on intervening structure. Also it may be that not enough "structure" has been defined leaving a gap for the belt to "fall through".

In this (contrived) example of fitting to a pelvis the belt above the left leg penetrates the upper leg structure.

This is caused by the 3rd path point being too low down causing the lower edge of the belt path to be on the wrong side of the upper leg structure.

Moving this point up (**PATH...**, **MODIFY**), then re-fitting the belt will fix the problem.



I want to mesh chest and lap belts using two different element properties

If you define the whole belt, chest and lap, in one operation then it will only be possible to mesh it using a single property. Also the nodes at fixed point locations, eg sliprings, will be common to their adjacent segments giving continuity across sliprings.

To get separate properties it will be necessary to create two separate belt definitions: one for the chest section and one for the lap. These will then be able to have different properties defined during their meshing operations.

By default this will give separate nodes at end points, thus losing the continuity of the belt through sliprings. However the **MESHING** operation permits you to stipulate the end nodes to be used, so if continuity is required the second section to be meshed can still be made to join up with the end of the first one.

There is no limit to the number of belt definitions that can be defined in a model, and they may be joined up as required.

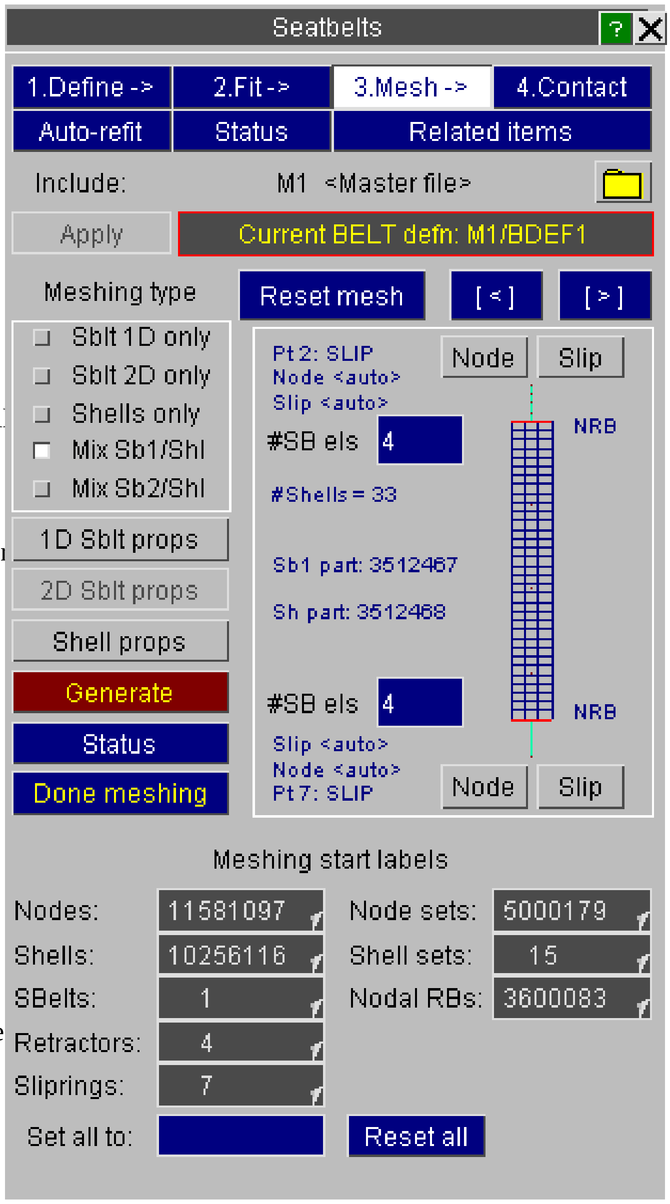
This example shows how a simple four-point harness could be generated using four separate belt definitions: left and right sides, upper and lower cross-sections. Two different section properties are used.

Once this section has been completed it is possible to mesh the seatbelt definition (see section 6.28.4: Meshing)

6.28.4 **Mesh:** Meshing the fitted "chassis" mesh with structural Finite Elements

This is the final stage of seatbelt creation in which true Finite Element (FE) mesh elements are applied to the "chassis" mesh you have created and fitted.

This figure shows the standard meshing panel.



Each chassis segment is meshed separately in turn so that the mix of **SEATBELT** and **SHELL** elements in each segment can be controlled independently: each segment may be all belts, all shells, or a specified mixture of the two.

You move between belt segments using the [>] (forwards) and [<] (backwards) buttons. A diagram of the current section is shown, and will also be sketched on the graphics image.

Meshing type determines how this section of belt will be meshed. There are five options described in more detail below.

1D Seatbelt, 2D Seatbelt and **Shell properties** define the "chassis" element properties for each type of element. This means the Part, Section and Material definitions for each type plus any further optional data on the relevant Element cards.

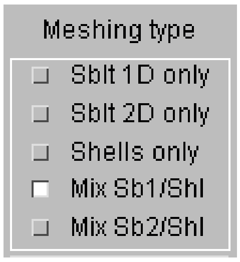
It is also possible to select explicit nodes and, if relevant, sliprings or retractors at each end of the belt segment. However this can usually be left in "automatic" mode.

In the case of a belt segment which mixes 1D or 2D seatbelt elements and shells it is also possible to define how many seatbelt elements should be used at each end.

Once the mesh definition is complete and all required properties have been given **Generate** may be used to create the mesh.

Meshing start labels allow you to control the labels from which each of the various belt components are created. These default to the "next free in layer", but may be set to any value so long as there is enough contiguous free space above them to hold all the required items.

Meshing type: Defining the mix of SEATBELT and SHELL elements.



Each segment may be meshed with one of:

`Sblt 1D only`	**1D *ELEMENT_SEATBELT** elements only
Sblt 2D only	**2D *ELEMENT_SEATBELT** elements only
`Shells only`	Conventional **SHELL** elements only
`Mixed Sb1/Sh`	A mixture of **1D SEATBELT** and **SHELL** elements
Mixed SB2/Sh	A mixture of **2D SEATBELT** and **SHELL** elements.

Historically 1D Seatbelt elements have been used to attach to retractors and in stretches through sliprings, and Shell elements have been used where contact with the dummy is required. This is **Mixed Sb1/Sh** mode.

Now that 2D Seatbelt (shell) elements may be used then such meshes may be (re-)created in two possible ways:

1. Replacing the stretches of 1D seatbelt elements with 2D seatbelt shells, but preserving the stretches of conventional shells in between. This is **Mixed SB2/Sh** mode, and may be preferred by users who wish to retain belts meshed with shells and simply want to improve the contact between belt and surrounding structure in slipring and retractor regions.

2. Replacing the whole belt with 2D seatbelt shell elements, which is **Sblt 2D** only mode.

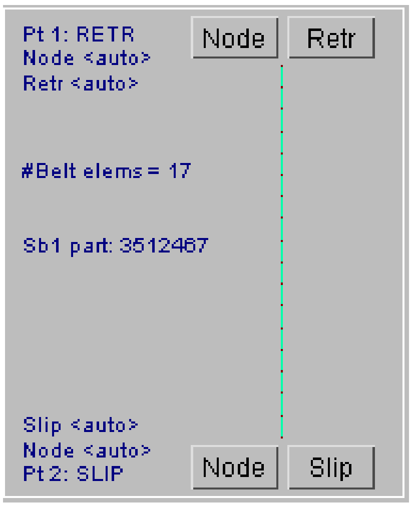
PRIMER will remesh an existing belt definition in any of the new types above, changing element, slipring and retractor properties as required. However certain permutations are not geometrically possible:

- Where a belt passes through a slipring then the sections on both sides must be either 1D Seatbelt or 2D seatbelt elements as sliprings cannot mix 1D and 2D belt types. In addition retractors may only attach to 1D or 2D seatbelt types. Therefore at a Slipring or Retractor only the following permutations are available:

Adjacent belt type	Permitted adjacent meshing types
1D Seatbelts	`Sblt 1D only` or `Mixed Sb1/Sh`
2D Seatbelts	**Sblt 2D only** or **Mixed SB2/Sh**

- A single stretch of seatbelt may be of a single type (1D Sbelt, 2D Sbelt, Shells) or a single mixture of types (1D Sbelt + Shells, 2D Sbelt + Shells). It is not possible to combine all three element types, or to mix 1D and 2D seatbelt elements in a single stretch.

Sblt 1D only: Meshing with 1D Seatbelt elements only



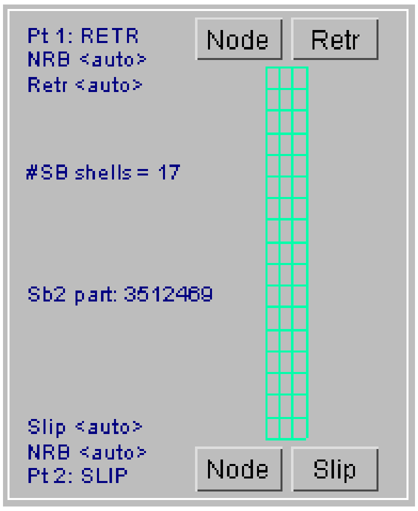
This mode is typically used in a mixed 1D Seatbelt + Shell mesh for the initial stretch between retractor and slipring at shoulder.

There is nothing further to define since the stretch will contain only a single type of element.

End details are handled as followed:

At a retractor or slipring	The belt connects directly to the retractor or slipring element
At structure	The last belt node uses the relevant node on the structure
At another 1D belt element	The belt elements share a common node at the point.

Sblt 2D only: Meshing with 2D Seatbelt elements only

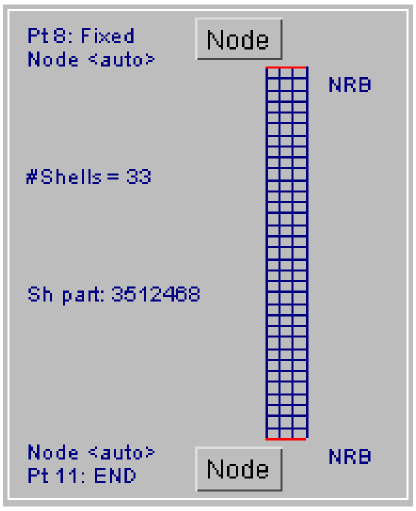


This mode may be used for any stretch of belt.

End details are handled as followed:

At a retractor or slipring	The belt connects directly to the retractor or slipring element
At deformable structure	A nodal rigid body is created from the line of nodes at the end of the belt, plus the relevant node on the structure.
At rigid structure	The nodes on the end of the belt are placed in a node set which is used to make an "extra nodes on rigid part" definition assigned to the rigid structure part.
At another 2D belt element	A nodal rigid body of the nodes at the ends of each section would be created. (This would he highly unusual in practice.)

Shells only: Meshing with conventional shells only

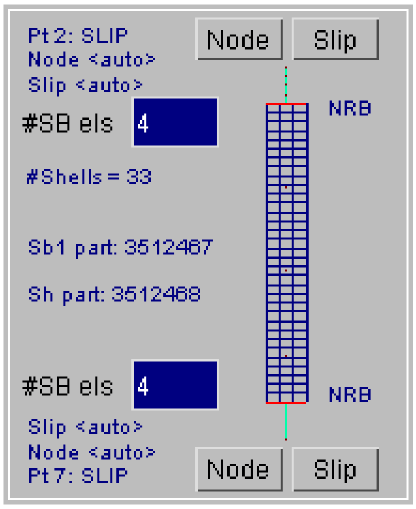


This would be unusual since "shell only" stretches of mesh cannot attach to sliprings or retractors, so this would only be applicable to an isolated stretch of belt with fixed connections at each end.

The end details are handled as follows:

At deformable structure	A nodal rigid body is created from the line of nodes at the end of the belt, plus the relevant node on the structure.
At rigid structure	The nodes on the end of the belt are placed in a node set which is used to make an "extra nodes on rigid part" definition assigned to the rigid structure part.
At another shell (belt) element	A nodal rigid body of the nodes at the ends of each section would be created. (This would he highly unusual in practice.)

Mix Sb1/Sh: Meshing with 1D belt elements + shells



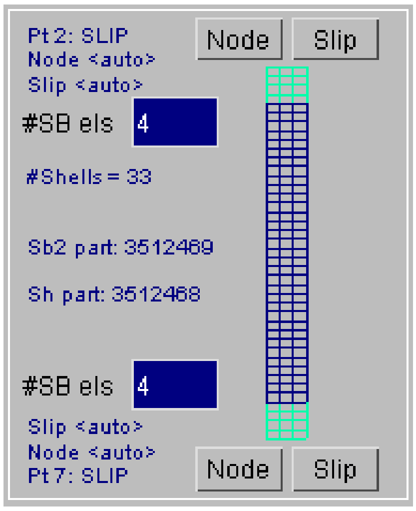
This is the traditional method of combining connectivity through sliprings and retractors (using 1D seatbelt elements) with good contact across the dummy (using shells). A typical stretch of belt will be mainly shells with a few seatbelt elements at each end.

#SB els defines how many 1D seatbelt elements are to be used at each end, which must be at least one if there is a slipring or retractor at that end.

The end details are handled as follows:

1D belt element at structure	The end node of the belt element is the specified node on the structure.
1D belt element at slipring / retractor	The belt connects directly to the slipring/retractor.
Shell to 1D belt connection	A nodal rigid body is formed from the end row of shell nodes, and the 1D belt element also has an end node in this node set.
Shell directly to deformable structure	A nodal rigid body is created from the line of nodes at the end of the shell, plus the relevant node on the structure.
Shell directly to rigid structure	The nodes on the end of the shell are placed in a node set which is used to make an "extra nodes on rigid part" definition assigned to the rigid structure part.
At another shell (belt) element	A nodal rigid body of the nodes at the ends of each section would be created. (This would he highly unusual in practice.)

Mix Sb2/Sh: Meshing with 2D belt elements + shells



This method could be used as a direct replacement for the mixed 1d belts + shells above. In this case **#SB els** defines how many 2D belt elements are used at each end. Since 2D seatbelt elements are really shells no special measures need to be taken at the transition between the two element types.

The end details are handled as follows:

2D belt element at slipring / retractor	The belt connects directly to the slipring/retractor.
2D belt or Shell element directly to deformable structure	A nodal rigid body is created from the line of nodes at the end of the belt/shell, plus the relevant node on the structure.
2D belt or Shell element directly to rigid structure	The nodes on the end of the belt/shell are placed in a node set which is used to make an "extra nodes on rigid part" definition assigned to the rigid structure part.
2D belt or Shell to another belt/shell (belt) element	A nodal rigid body of the nodes at the ends of each section would be created. (This would he highly unusual in practice.)

How the ends of stretches of shells or 2D seatbelts are handled

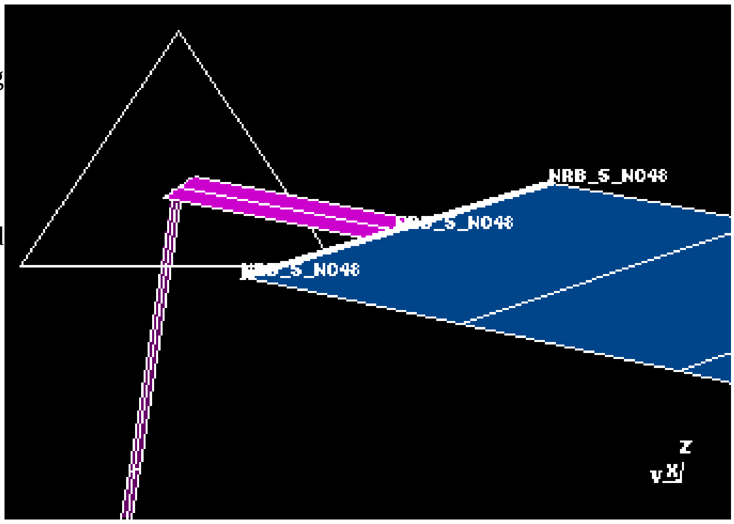
The ends of stretches of (seatbelt) shell elements need special treatment since they need to be attached at a single point either to a structural node or to the adjoining seatbelt element. This presents two problems:

- For odd numbers of shells across the belt (**#rows** = 1, 3) there is no central node to attach to.
- In all cases fixing at a single point will lead to excessive warping, hourglassing or other distortion of the end shells because of the lack of end constraint.

Therefore PRIMER usually creates a "Nodal Rigid Body" (NRB) at the ends of stretches of shell elements: both to constrain the ends against warping and to provide an attachment point. (Exceptions are given below.)

This example shows a shell (blue) to seatbelt (purple) connection next to a slipring. A Nodal Rigid Body has been created using the two shell end nodes, and an extra node has been created at the shell centreline to which the seatbelt element is attached.

There are some special cases that arise because of the limitation that nodes can only be constrained by one rigid body at a time.

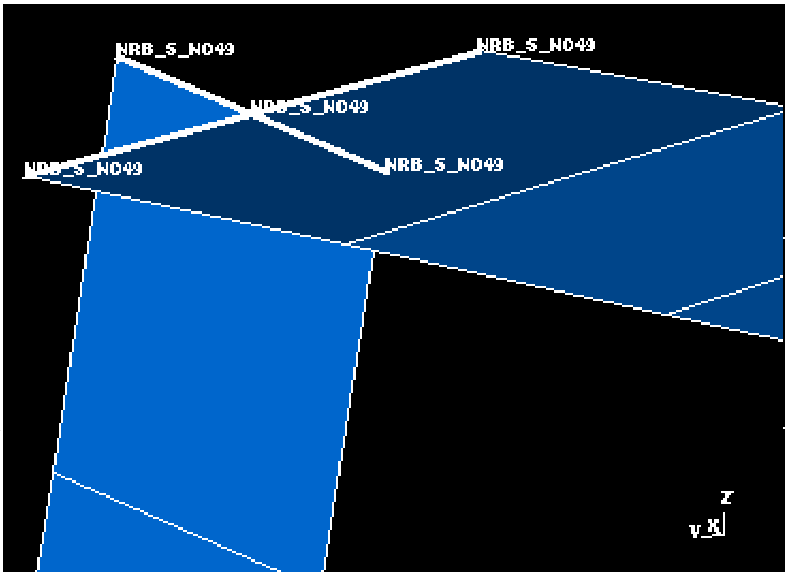


Shell mesh meets shell mesh at fixed point (no intervening seatbelt elements)

In this situation Nodal Rigid Bodies are required at each shell end, but because they share a common centre node they cannot be separate.

Therefore PRIMER creates a single NRB and places all the nodes into it, as shown in this example.

This has the effect of fixing the belt end geometries together, ie they can't twist relative to one another. If this is not acceptable it will be necessary to mesh the two segments as separate belt definitions, then to connect the central end nodes (which will be separate) manually with a joint, which may need lumped masses at the nodes for stability.

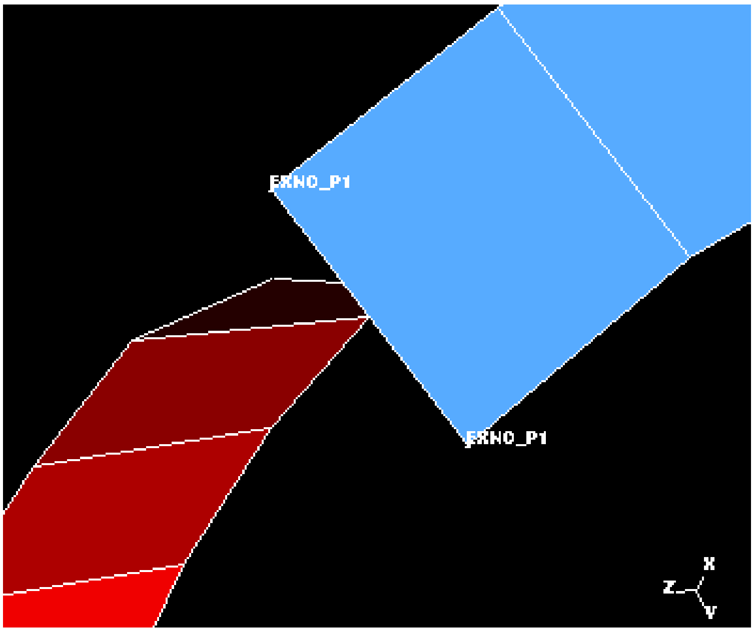


Shell mesh meets rigid structural node. (also no intervening seatbelt elements)

Where a shell mesh attaches directly to a node on rigid structure, PRIMER does not create a nodal rigid body: instead it makes the end nodes of the shell "Extra Nodes" on the rigid part of the structure.

In this example the (blue) belt shells are attached to (red) rigid part #1, and the two end nodes have been made Extra Nodes on part 1.

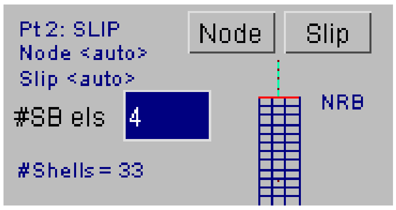
Again, this fixes the end shell relative to the rigid structure, inhibiting relative rotations. If this is unsatisfactory then let PRIMER create the shell centre node and NRB, then attach the structure manually to the central NRB node via a joint (maybe with a mass) as above.



Other illegal constraints on end nodes.

At present PRIMER only checks for the cases above when meshing belts. It would be possible to imagine some other very obscure illegal constraints, but these are so unlikely that they are ignored. You will have to fix such cases manually if they arise.

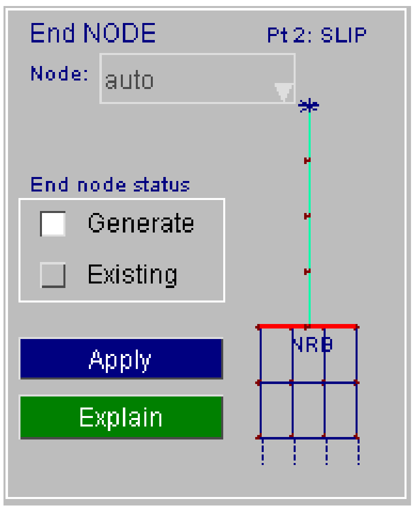
Controlling the end connectivity of each segment



By default PRIMER generates any nodes required at the end of each segment, ensuring that connectivity between stretches of belt is maintained. Also if a slipring or retractor are specified at the end of a segment this will also be created.

However it is possible to override this behaviour and to specify existing nodes, sliprings and retractors at the ends of segments, and this might be necessary when connecting up a belt meshed in PRIMER to one meshed elsewhere that the belt fitter does not "know" about.

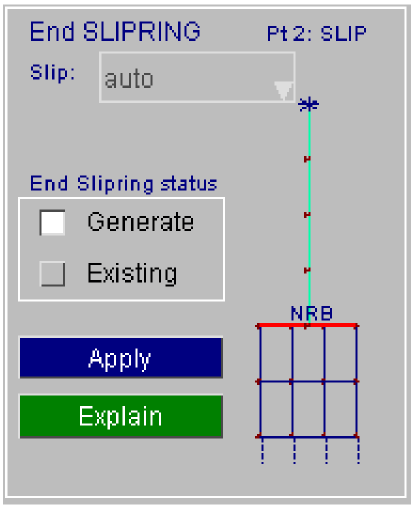
NODE: defining the end node for 1D seatbelt element connectivity



When a segment ends in a 1D seatbelt element the end node is normally created or selected automatically, but you can override this by choosing **Existing** and specifying an existing node.

In this case it is your responsibility to ensure that connectivity between adjacent stretches of belt is correct.

SLIP or **RETR**: defining a slipring or retractor (1D and 2D cases)



In a similar way if a slipring or retractor have been defined at the end of a stretch of belt PRIMER will normally create them automatically, however you can override this and select an existing element.

Note that:

- In the 1D case any existing seatbelt elements will be replaced as required with new elements created as part of this meshing operation, and will be left "floating" in the model. You will need to delete these manually.

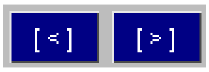
  The slipring or retractor node (SBRNID) will be replaced with the correct node for this mesh. This may result in the slipring / retractor moving in space in order to be at the correct position.

- In the 2D case any existing element sets will be emptied of their contents, and new elements from this meshing operation inserted. Again you will need to tidy up any elements left "floating" in the model.

  Any existing node set (-SBRNID) will be emptied and repopulated with new nodes at the correct positions. As in the 1D case this may result in the slipring/retractor moving to a new location.

Navigating the meshing panel between segments.

When a belt definition has more than one segment the meshing panel only displays one segment at a time: by default the first. In addition the segment highlighted on the display will be the current one.



To move between segments use **[<]** to go backwards, and **[>]** to go forwards. You can revisit and modify segments as often as you like, they only get meshed when the **GENERATE** command is used.

Setting element properties.

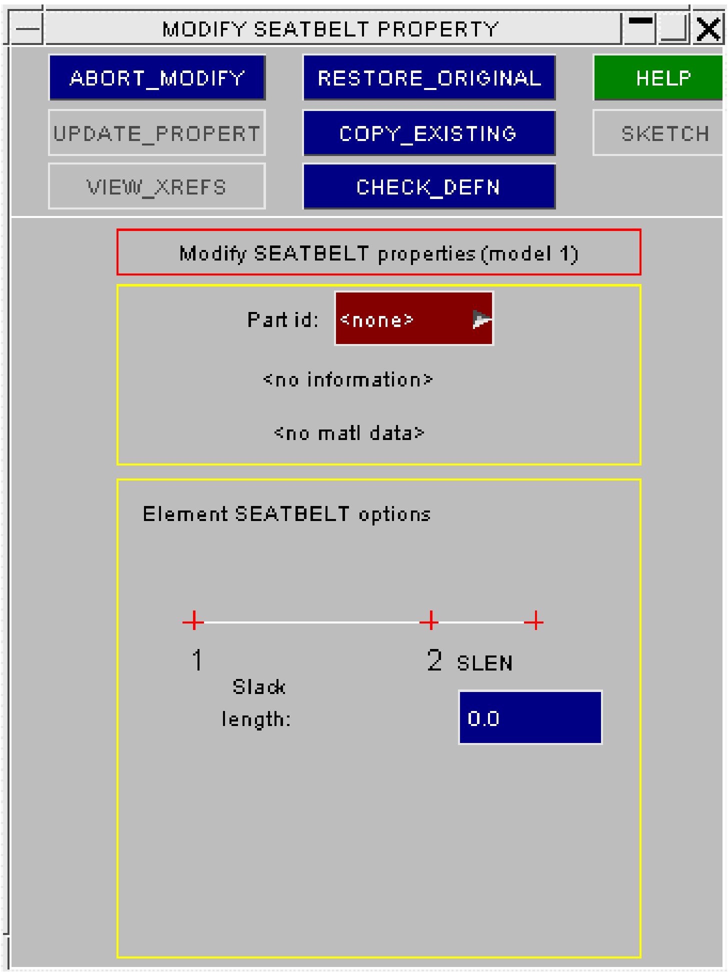
Before elements can be generated PRIMER must know what properties (Part ids, etc) to assign to the elements.

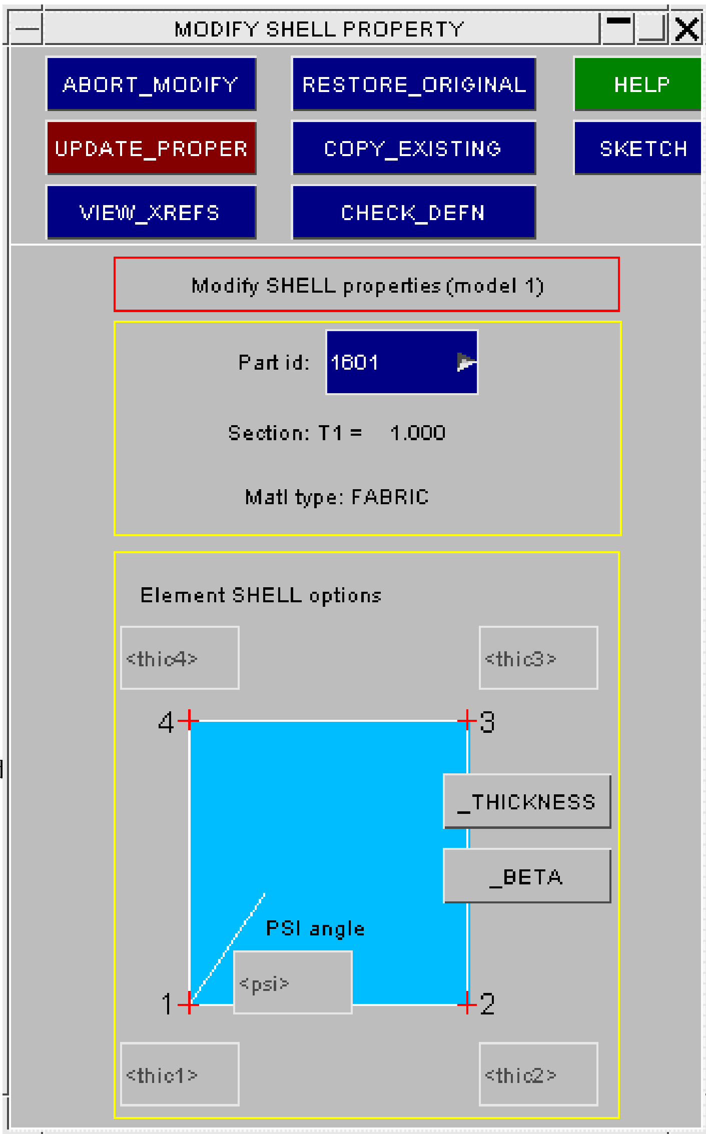
Therefore it is necessary to select an existing property, or create a new one for the relevant class(es) of element. If a property is required but has yet to be defined that button will have a red background, and the **GENERATE** button will be greyed out until it has been selected or created. Properties not required will have their button greyed out.

This panel shows the 1D seatbelt element create/ edit box.

The user is required to give a part ID, this can either be typed in directly or selected/created via the part pop-up box. Information about the section and material properties from the part to be used are shown below the part ID.

There is also the option of setting the element-specific "slack length" value.





This panel shows the shell property creation/editing box.

As above, a part ID is required to be set. Here one has been selected and the properties are displayed below the ID. The **update_property** button has now been made active.

The shell element specific data, (variable thickness at nodes and beta angle) is not used here.

This panel is also used for 2D Seatbelt Shell elements which, despite being defined as ***ELEMENT_SEATBELT** are in fact shells and must have the following attributes:

- A ***PART** card unique to this seatbelt shell definition.
- A ***SECTION_SHELL** (***not _SEATBELT***) definition that is unique to the part. Since the shells are used only for contact purposes the properties on this card are not structurally significant, however a sensible thickness should be used and a membrane formulation (type 5) is recommended.
- A ***MAT_SEATBELT** card. It is recommended that this too is unique to the ***PART** definition above.

Unique definitions are required because of the extra work performed in the LS-DYNA keyword reader to "track down" the belt and generate 1D belt, slipring and retractor elements. It gets confused if multiple belts used the same basic definitions.

Actually creating the Finite Element mesh

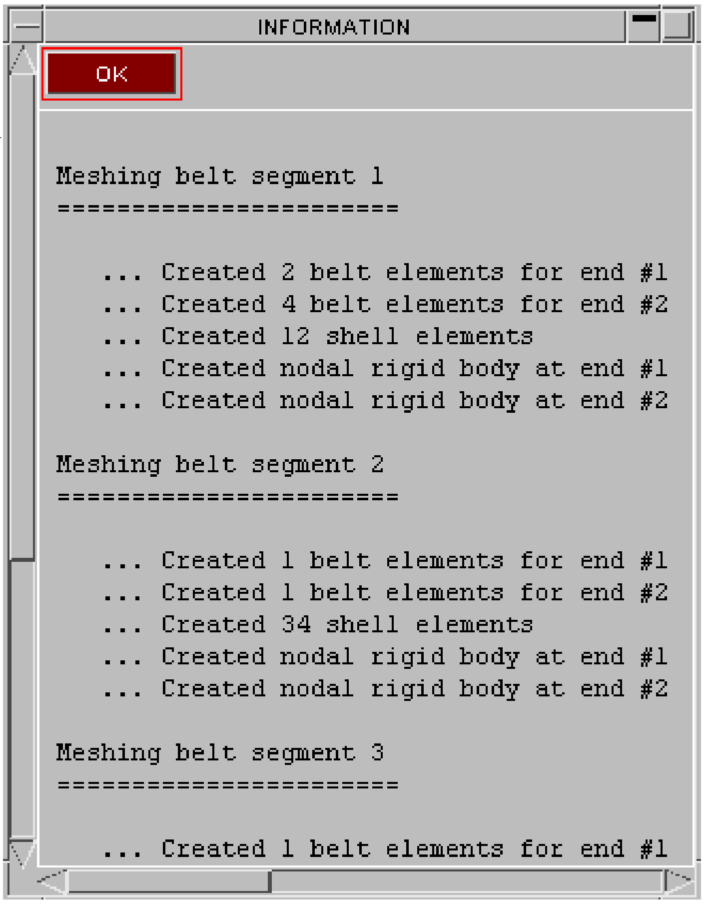
Once you have defined all the necessary information use **GENERATE** to go ahead and generate the finite elements themselves.
If you are modifying an existing seatbelt definition you will be asked first whether you want to delete the existing nodes, elements, etc. You have the choice of deleting them or leaving them - it doesn't matter which - as the new definition will supersede any previous one.

However if you have chosen start labels for the new mesh that you know will overlap with the existing ones then you must delete them!

PRIMER will generate a confirmation message listing what has been created.

Here a typical mixed seatbelt and shell belt, with nodal rigid bodies connected the two element types.

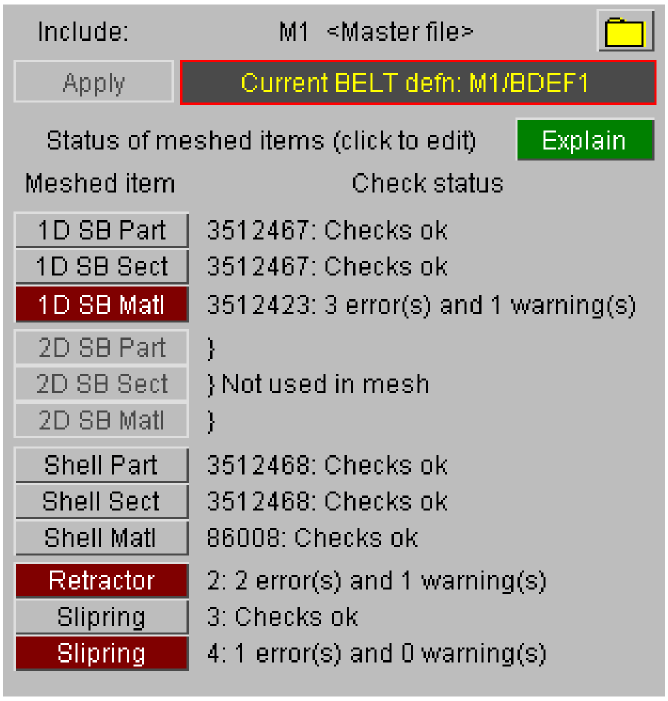
And that's it: you've created a seatbelt!



Post-meshing review

Following a meshing operation PRIMER runs a check on the new definition and shows a summary of the results. It is not uncommon for the definition of retractors to be incomplete (PRIMER cannot know how they are to be activated) and for other errors to occur.

Items with errors are highlighted in red, as shown in this example, and you can click on them to open their editin panels and correct the problems.



What is stored following meshing

Internally PRIMER saves the following information about Seatbelt Definitions:

The sets of structure definitions. (See section 6.28.1).

The basic path. (See 6.28.2).

The "chassis" mesh and associated fitting data. (See 6.28.3).

The belt FE data: sets of shells, first and last seatbelt, slipring and retractor elements, and the sets of nodal rigid bodies used.

Contact data if it exists (See the next section).

A **MESHING** operation operates on the information saved in (1) to (3) above without changing it, therefore if you are happy with the belt shape but don't like the mix of element types you have created, you can go back and change it at will.

What you *can't* do in **MESHING** is change the "chassis" path, and in particular you can't change the number of shells across the width: to do that you must go back to **FITTING** (section 6.28.3), fit a revised chassis mesh with the new number of rows, then **ACCEPT** and remesh it.

Changing and remeshing an existing belt definition.

Once a seatbelt has been created and meshed in PRIMER, (ie that there is a set of *BELT cards after *END) then you can do the following:

Remesh the existing path geometry using a different mixture of elements. For example change an existing Mixed 1D/Shell belt to a Mixed 2D/Shell belt.	Simply use **Mesh** to define a new arrangment of elements or element types. Note that simply remeshing does not, in itself, permit the following: • Changing the basic belt path • Changing the number of rows of elements across the belt, or any other aspect of the belt element geometry. The existing belt definition will be deleted and the new one will replace it. (If you have not previously "Fitted" the belt during this PRIMER session it will be necessary to do that first, since the detailed meshing path is not stored in the keyword file. Performing a "Fit" will repeat the form-finding operation using the saved basic path, which regenerates a detailed path suitable for remehsing.)
Modify the basic belt path, or change the belt geometry, and then remesh.	Firstly use **Fit**, where you can: • Define a new belt path by adding, subtracting or moving path points. • Change the belt dimensions or number of row Once the revised path geometry is correct **Save** it, re-**Fit** the belt to the dummy, **Accept** the result, and then re-**Mesh** it as above. The existing definition is deleted and a new one created as before.
If the dummy has moved, and you want to refit the existing belt definition to it in its new location.	The **Auto-refit** capability will perform this task if no other changes are required to belt organisation or topology. **Auto-refit** attempts to re-use all existing node, element and set labels so that the belt just appears to move to the new location. Otherwise the process may be performed manually as above.

More details about meshing 2D seatbelt elements.

2D seatbelt elements, introduced in LS-DYNA 971R4, can be meshed by PRIMER but they are quite intricate and the following explanation has been added at the request of users to try to explain in more details how this is done.

What is a 2D seatbelt element, and how does it work?

LS-DYNA imports these elements using the ***ELEMENT_SEATBELT** card which traditionally has been used to define 1D seatbelt elements, however if nodes 3 and 4 are defined on this card it becomes a 2D seatbelt element. Confusingly 2D seatbelt elements are actually shells, albeit special ones, and their label range forms part of that used by ***ELEMENT_SHELL**. Therefore PRIMER is also forced to treat them as shells to avoid conflict. The following summary may help:

- 2D seatbelt elements are defined on the ***ELEMENT_SEATBELT** card by defining nodes N3 and N4.
- Internally they are actually **SHELL** elements, and their labels must not clash with existing shells.
- They should refer to a conventional ***PART** card.
- The ***PART** card should refer to a ***SECTION_SHELL** card (***not *SECTION_SEATBELT***) which should be unique to each stretch of 2D belt elements.
- The ***PART** card should refer to a ***MAT_SEATBELT** definition.

- ***ELEMENT_SEATBELT_RETRACTOR** and ***ELEMENT_SEATBELT_SLIPRING** definitions must also obey special 2D syntax, using sets of nodes and elements.
- A node set (field <**edgset**>) on the ***SECTION_SHELL** card is used to tell LS-DYNA where the end of the belt is, and the ordering of nodes across it.

- The ordering of nodes, seatbelt elements and the sets of these used in retractors and sliprings must be topologically consistent throughout the belt.

Inside the LS-DYNA keyword reader the following happens when these elements are read in:

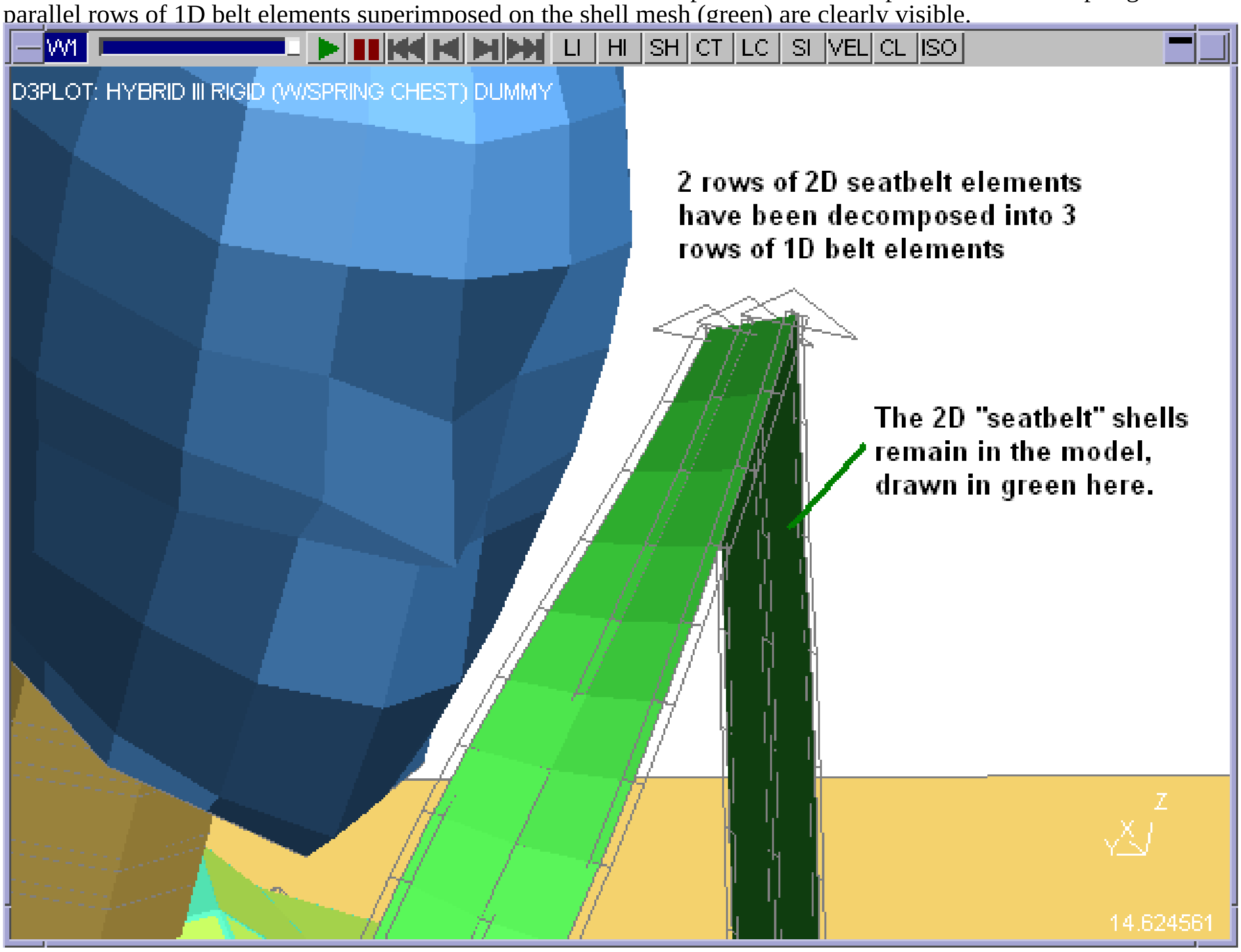
- The **N** parallel lines of 2D ***ELEMENT_SEATBELT** elements making up a belt are used to generate **N+1** parallel lines of conventional 1D seatbelt elements.

- The 2D seatbelt elements, which are really shells, are retained as shells and are given notional non-structural properties. The properties used are not known, but their job is to hold the parallel lines of 1D belt elements together during the analysis and also to provide a 2D contact between belt and dummy.

- 2D retractors and sliprings are also decomposed into **N+1** parallel rows of their 1D equivalents, and the rows of 1D seatbelt elements pass through these.

- During the analysis these rows of 1D belt elements carry the force in the seatbelt, although the details of how they work with the 2D shells are not known.
- Examining the output during post-processing reveals how this decomposition from 2D belt items to their 1D equivalents has taken place.

The following image, showing (during post-processing) the shoulder slipring detail of a model containing a 2D seatbelt with 2 rows of "belt" shells demonstrates how this decomposition has taken place. The three sliprings and parallel rows of 1D belt elements superimposed on the shell mesh (green) are clearly visible.



How 2D belts are meshed in PRIMER

During the fitting operation PRIMER maintains a "scratch" belt definition showing the number of parallel rows of elements that the user has defined. During 2D belt meshing these simply become 2D **`*ELEMENT_SEATBELT`** definitions, but the following extra work has to be performed:

- At any retractor and slipring positions a **`*SET_NODE`** is required to define the structure to which the retractor/slipring is attached. This is defined as field **`sbrnid`** on the retractor and slipring cards using -ve label to designate "set for 2D element".
- Sets of shell elements (**`*SET_SHELL`**) have to be created (data field **`sbid`** on the retractor card, **`sbid1`** and **`sbid2`** on the slipring card, all using -ve labels) to define the row of 2D seatbelt elements, which are really shells, adjacent the the retractor/slipring.
- At any free end of the belt PRIMER will create a nodal rigid body to act as a rigid fixing using the set of nodes at the free end, unless the end node location is already on a rigid part in which case these nodes become "extra nodes" on that rigid part instead.
- PRIMER will also place a node set at a free end of the belt into field **`edgset`** on the **`*SECTION_SHELL`** card of the belt elements. This is required to tell LS_DYNA both where the end of the belt is, and what its topology and orientation are. Since **`edgset`** can only refer to a single belt this explains why a separate **`*SECTION_SHELL`** card is required for every 2D belt in a model. If a retractor has been used PRIMER will use its node set **`sbrnid`**, otherwise it will use the node set of the free end.

Special care has to be taken to ensure that the topology of all elements and the ordering nodes in the belt, and in all sets, is consistent so that LS-DYNA can rely on it to determine the belt geometry.

Special geometrical rules also have to be applied at slipring locations to make sure that the 2D belt mesh is continuous through them and aligned sensibly. This has been described previously in "The effect of sliprings on the belt path".

Output from 2D Seatbelt elements

Time-history output of 2D seatbelt data appears in the ascii file **sbtout**, generated by ***DATABASE_SBTOUT**, and has changed slightly over time:

- In LS971R4.2.1 and older output reveals how the internal representation of these belts is split up into parallel 1D rows as above.
- In later LS-DYNA versions output is merged back into single sliprings and retractors, as it appears in the input.
- There is an undocumented data field **1p1dblt** in row 2, column 7 of the ***CONTROL_OUTPUT** card which will cause output to revert back to the raw 1D output if set to 1.

Output of data for plotting is possible because LS-DYNA will generate beams on top of "discrete" elements, which includes 1D seatbelt elements, if the **beam** field in column 3 of the ***DATABASE_D3PLOT** card is set correctly. Therefore beam plotting can be used to visualise the forces in the belt elements.

6.28.5 **Contact:** Creating a contact between belt and dummy

Once you have created and meshed your seatbelt the final, stage is to create a contact between it and the dummy. In most models this will be done outside the seatbelt fitter since the contact will almost certainly have to include structure not explicitly included in the dummy and belt definition (seat, dashboard, airbag, steering wheel, etc).

However if a simple contact between belt and dummy will suffice PRIMER provides the option of generating these contact definitions automatically for you.

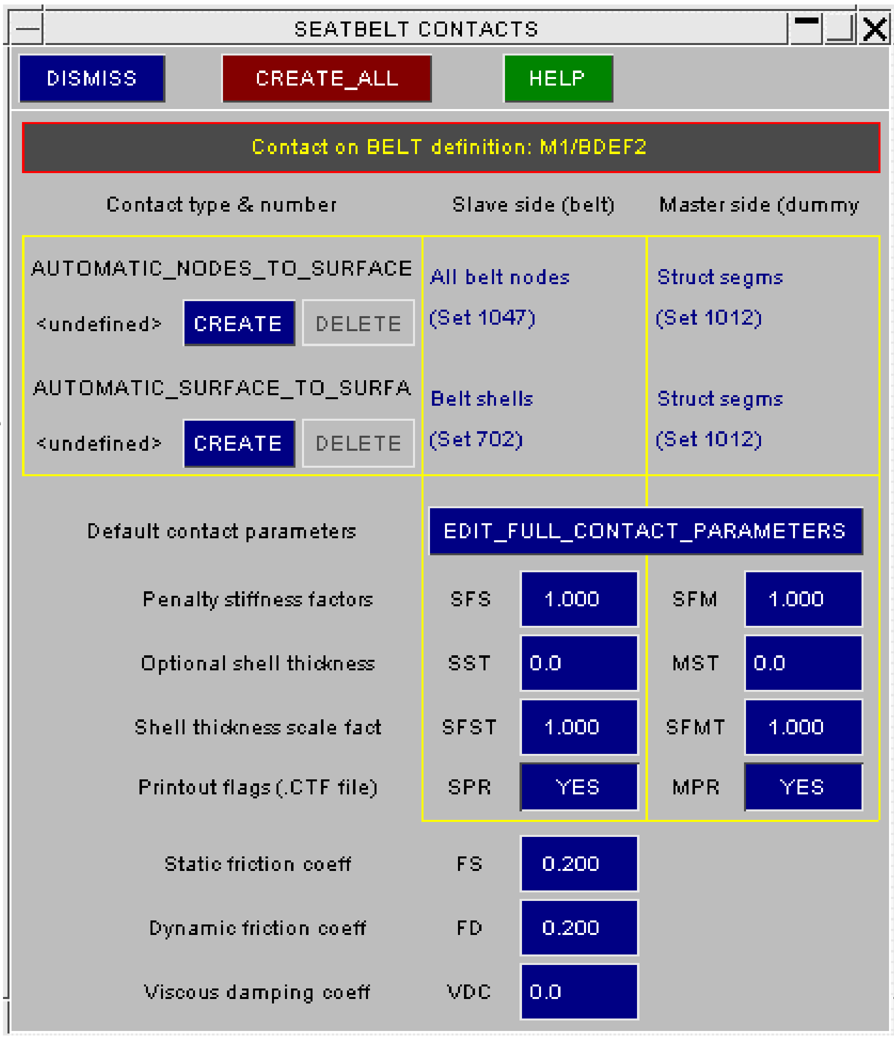
PRIMER works on the assumption that either or both of the following contact surfaces will be required:

`*CONTACT_AUTOMATIC_SURFACE_TO_SURFACE` Between belt shells and dummy.

`*CONTACT_AUTOMATIC_NODES_TO_SURFACE` Between belt nodes and dummy.

The need for the second, node-based contact arises when **`*ELEMENT_SEATBELT`** elements are used, since these are "line" elements with no effective surface.

PRIMER generates "chassis" contact definitions as shown in the adjacent figure.



Prototypes of the two contact types defined above are created by assuming that the dummy structure (as segments) form the master side of both, and creating two further sets:

A **`*SET_NODE`** of all nodes on the belt as the slave side of the **`NODES_TO_SURFACE`** contact.

A **`*SET_SHELL`** of all shells in the belt as the slave side of the **`SURFACE_TO_SURFACE`** contact.

In both cases the "structure"side is defined by a **`*SET_SEGMENT`**, because this allows it to include an arbitrary mixture of element types.

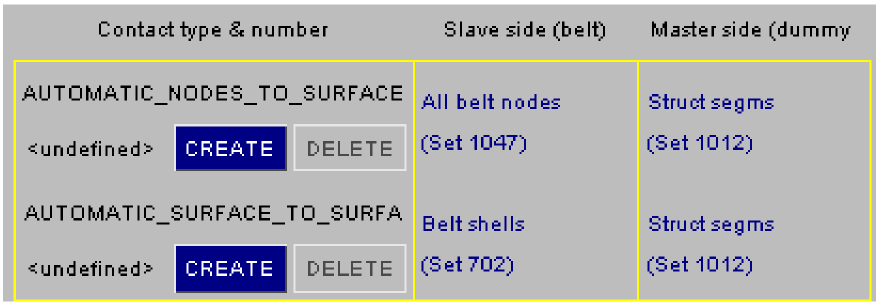
Default parameters are set up for the complete contact, and the main ones are shown here.

To create both these contacts with the default settings use **CREATE_ALL**. This will turn these "chassis" definitions into actual contact surfaces which, while part of a belt definition, are normal contact definitions which may be viewed and edited just like any other.

The default settings will create two contacts, which implies some duplication since all the nodes on shell elements will be included in the **`*NODE_TO_SURFACE`** contact, which is a bit wasteful. In addition default parameters may not be suitable for all models.

To create/edit contacts selectively

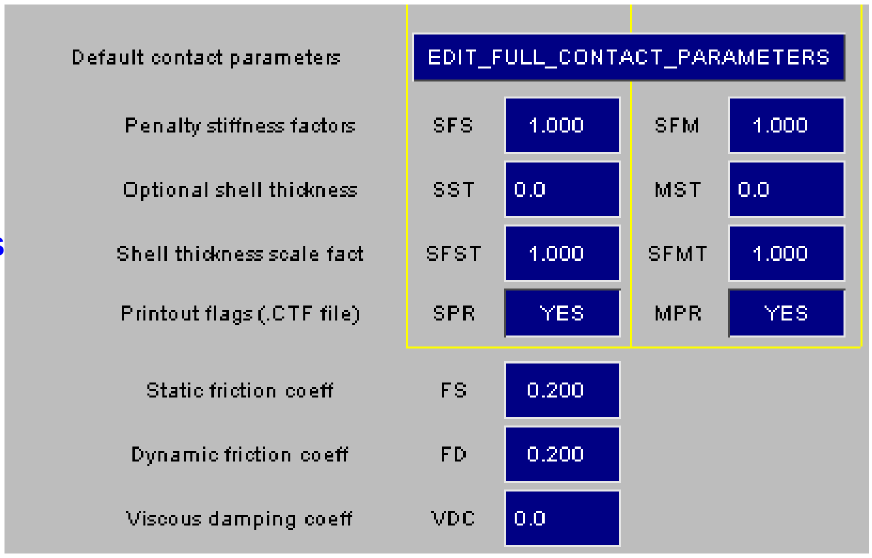
Instead of using **CREATE_ALL** you can create the surfaces selectively using the appropriate **CREATE** buttons. Once created they may be **EDIT**ed or **DELETE**d at will.



To change default settings

When contacts are created automatically the default parameters shown here will be used.

Only the most common ones are given in this box, and to gain access to the full set use **EDIT_FULL_CONTACT_PARAMETERS**



Editing seatbelt contacts once they have been defined

There is nothing special about seatbelt contacts and the sets used to define them: they can be changed, deleted and re-created just like any others, either from this panel or (at any time) from the main **CONTACT** keyword editing command.

For example to remove the nodes on shells from the **NODES_TO_SURFACE** contact in this example edit **SET_NODE** #52 to remove the nodes in **SET_SHELL** #6. And to define a contact between (say) and airbag and the belt it would make sense to reuse **SET_NODE** #52.

Saving belt contact information to file

Any contact surfaces made here will be saved in the seatbelt "tree" file structure, together with their sets. These are references to the surface and set definitions, not the definitions themselves, and are useful for reviewing and editing belt to dummy contact.

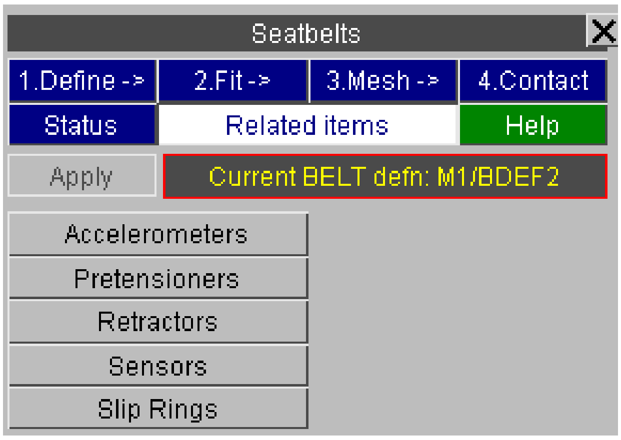
However if a seatbelt has been remeshed the "old" sets are invalid, so both they and the contact definitions are discarded and new ones must be created for the revised geometry.

6.28.6 **Related Items:** Creating Retractors, Sliprings, and other seatbelt-related elements

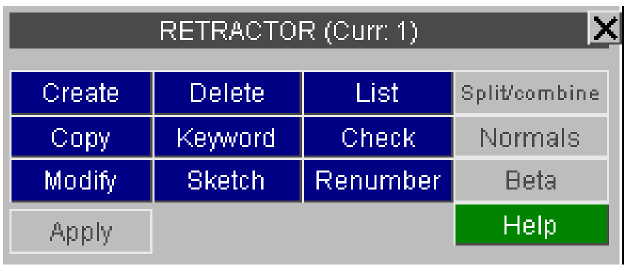
The main seatbelt control panel allows you to create and manipulate the other "seatbelt-related" items:

RETRACTORS	Spool in seatbelt elements, are triggered by sensors.
SLIPRINGS	Feed seatbelt elements through themselves to model material passing from one side to the other.
PRETENSIONERS	Pulls in material to tighten a belt, having been triggered by various means.
SENSORS	Provide a "trigger" for items above by detecting acceleration thresholds, relative movement, etc.
ACCELEROMETERS	Attach to a rigid body and provide accelerations in the frame of reference of that body for post-processing.

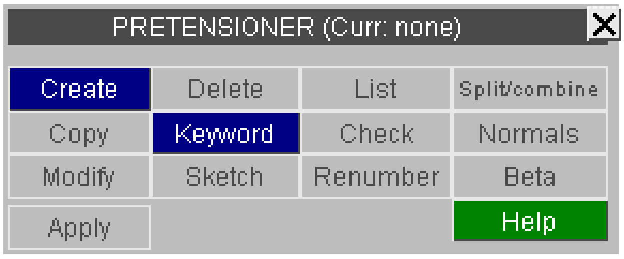
(There is nothing special about creating these elements from inside the seatbelt fitting panel, they may equally well be created from the normal **ELEMENT** keyword.)



Generic top level panel for all types

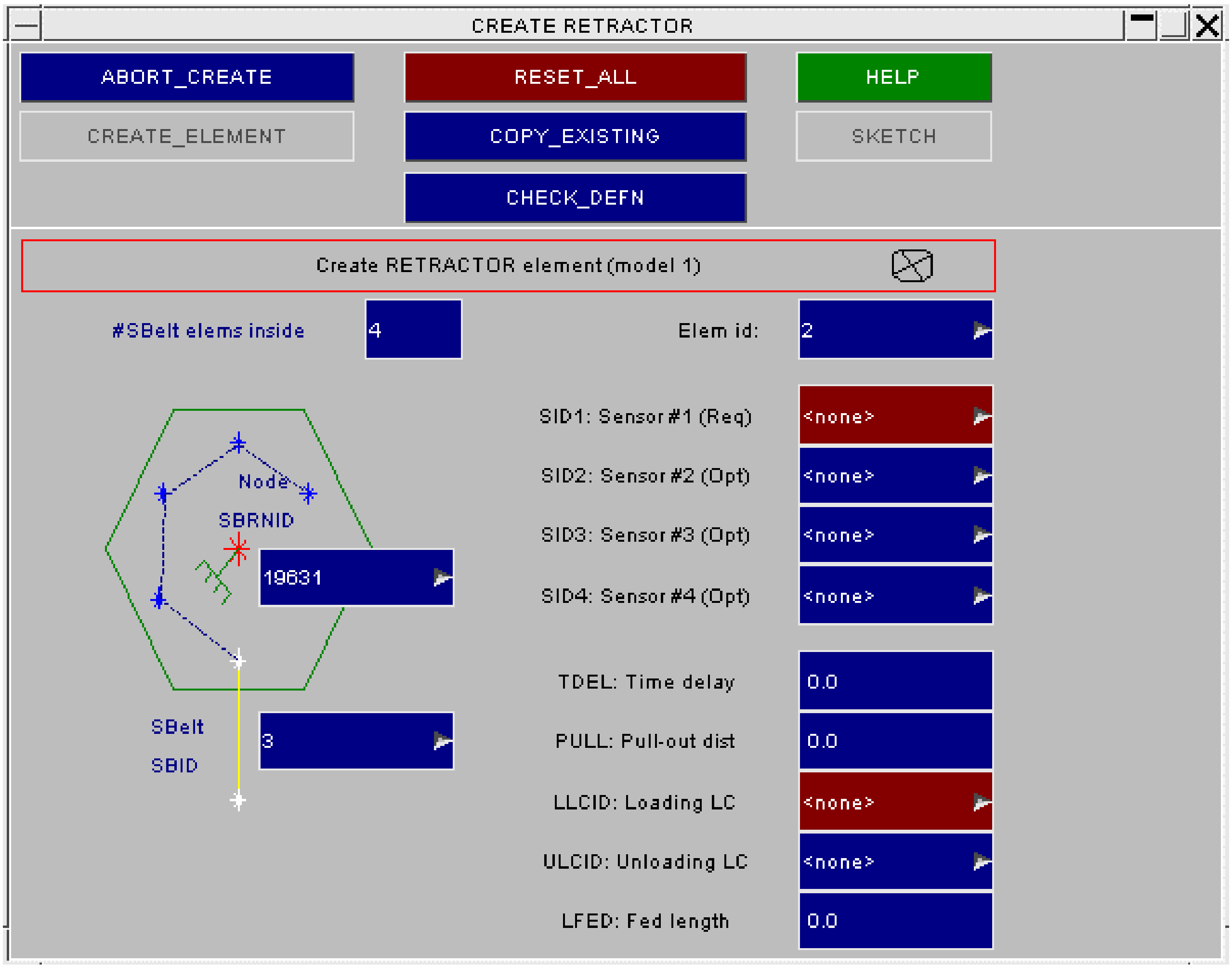


Top **RETRACTOR** panel



Top **Pretensioner** panel

All types have the same options and layout in their top panel, so only two examples are shown. In this example a retractor already exists, so the **MODIFY** and **DELETE** options are available, but no pretensioners have been defined yet, so only the **CREATE** and **KEYWORD** options are available.



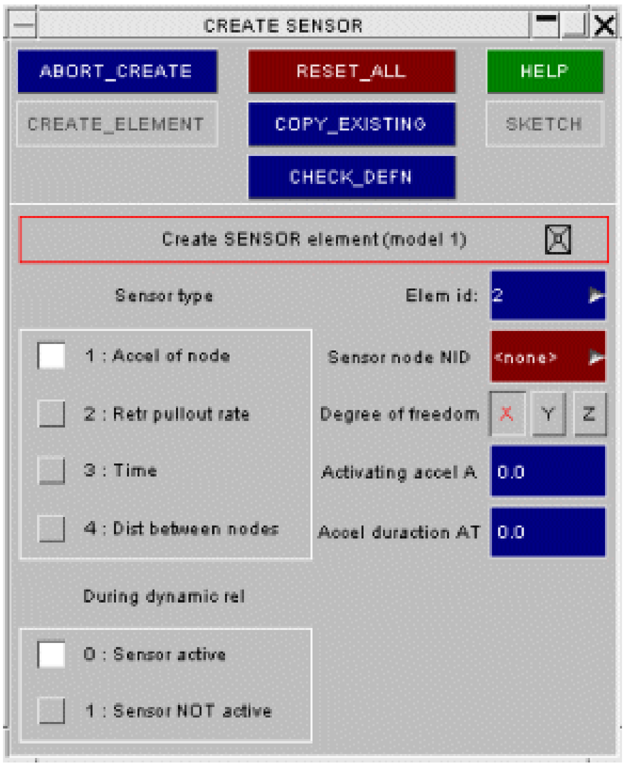
Creating a **RETRACTOR**

This panel shows the process of creating a retractor, with some items still to be defined. The layout and controls are standard for all seatbelt-related types:

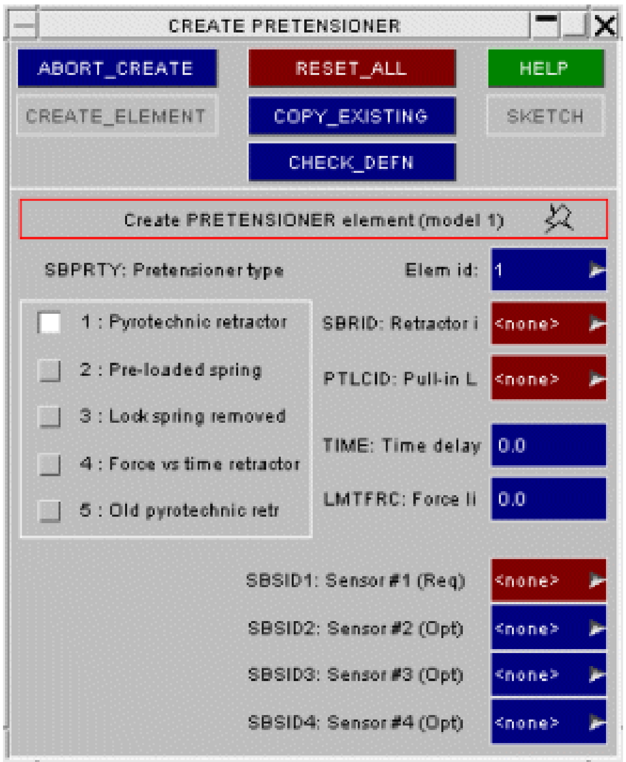
- Boxes with a red background are mandatory data that is missing (here the first sensor and a loadcurve id). The **CREATE**/**UPDATE** button will be "live" only when these missing items have been filled in.
- Boxes with a blue background have already been filled in or are optional data.

The top options

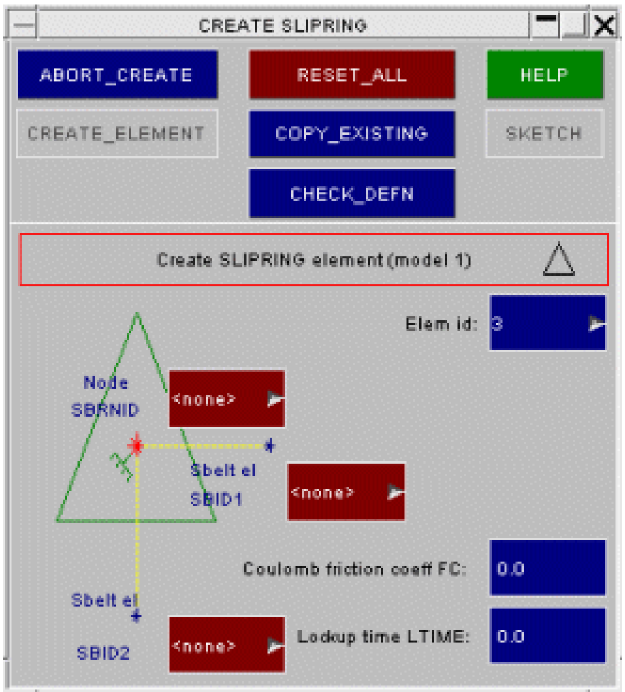
RESTORE/**RESET**	Resets the definition to its original state (modify) or zero (create).
COPY_EXISTING	Copies an existing definition into this one.
LIST_XREFS	Lists what (if anything) references this element.
CHECK_DEFN	Checks the definition so far, listing any errors found.
CREATE/**UPDATE**	Creates a new (create) or overwrites the existing (modify) definition.
ABORT	Abandons this operation leaving any original definition unchanged.



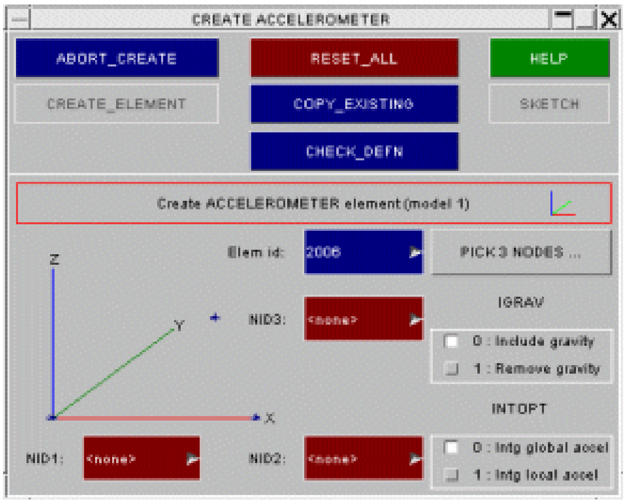
Create **SENSORS**



Create **PRETENSIONERS**



Create **SLIPRINGS**



Create **ACCELEROMETERS**

These figures show the create/modify panels for the remaining types. All follow the same standard layout, and use the same box colour and top options. With reference to the analysis code user manual the input required is self-explanatory.

The model in which elements are created

When only one model exists there is no ambiguity, but if more than one model is present in the database you will need to define the model in which the element(s) are to be created.

6.28.7 **Auto-Refit**: Refitting a belt automatically when the dummy moves

When a dummy is moved, for example by Occupant or Mechanism positioning, the old seatbelt definition will tend to get left behind, so it will become invalid and require repositioning and remeshing.

Primer can perform this task automatically so long as the belt was previously created in Primer, and the keyword input deck contains **`*BELT_xxx`** cards after **`*END`** as described in section 6.28.8.

How **Auto-Refit** works

When Primer reads in a keyword file containing **`*BELT_xxx`** data it automatically finds the nodes on the "structure" that are nearest to each belt path point, and stores their initial positions. During the refit operation the [dx,dy,dz] movement vectors at these "nearest nodes" are applied to the corresponding belt base path points, thus the path is updated to the new dummy position and the belt-fitting process can be repeated to find a new path around the dummy.

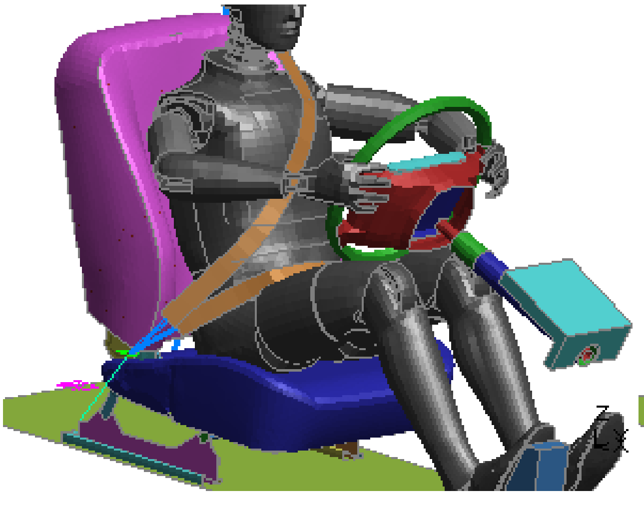
Using the information stored about the belt mesh Primer can then delete the old mesh and, using the same attributes, create a new mesh on the refitted path. This process is not limited to belt elements: it also tracks down each original belt section to find any retractors and sliprings, or direct connections to the structure. If found these are re-used and connected up to the revised belt mesh.

At present contact between the belt and the dummy is not recreated.

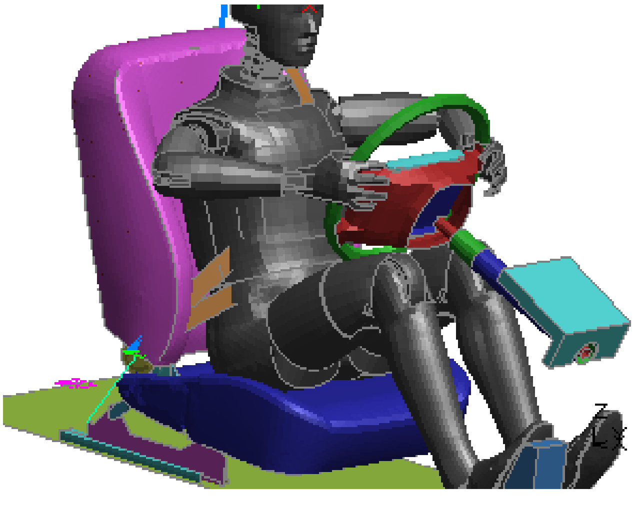
The whole process is automatic: provided the dummy shape is not dramatically different, and its movement from its original position is not excessive, a simple **APPLY** should refit and remesh the belt in a single operation.

Auto-Refit example

The following model shows a belted dummy both before and after it has been moved.

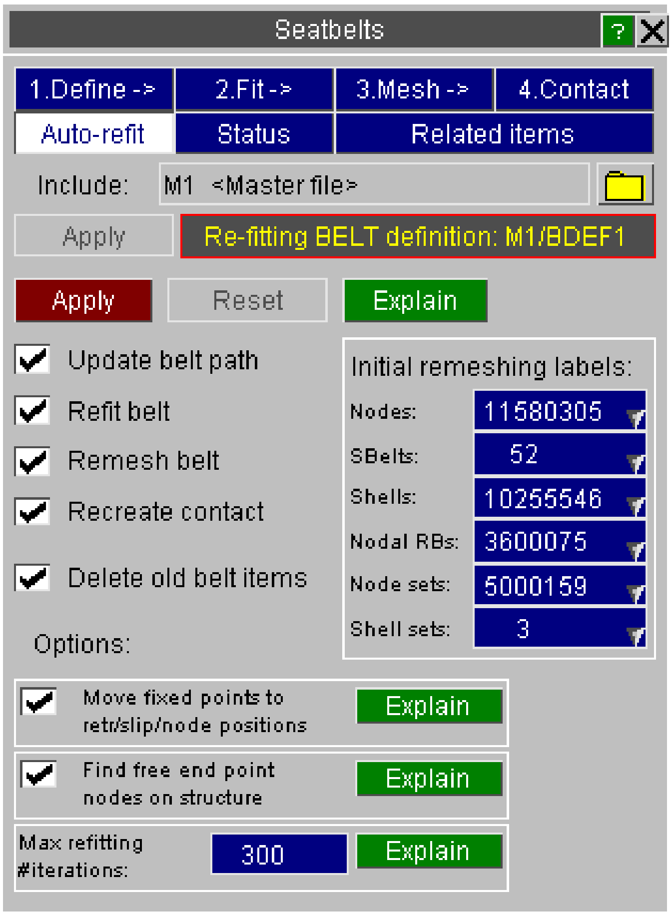


This image shows the occupant belted up in its mid position.



Here Mechanism positioning has been used to move the occupant and seat forward (by about 50mm) and down. However the belt has been "left behind" and needs to be repositioned.

The Auto-Refit panel:



This panel effectively automates a repeat of the normal belt fitting process using the following stages:

- **`Update belt path`** moves the initial path points to their new locations, computing offsets from the motion of their "nearest nodes".
- **`Refit belt`** performs a new form-finding operation using the revised path.
- **`Remesh belt`** creates a new mesh on the fitted path, using the attributes, element types and properties of the original belt. Any existing retractors and sliprings are re-used, and direct connections of belt ends to structural nodes are retained.
- **`Recreate contact`** at present does nothing, and can be ignored. (Experience has shown that users prefer to manage their own belt to dummy contacts.)
- **`Delete old belt items`** will, if selected, automatically delete the old belt elements, nodal rigid bodies, etc once the new definition has been created.

As a general rule you should leave all these options selected. It is only when the automatic refitting goes wrong that it may be necessary to intervene and perform some stages manually.

The following options require a little explanation:

`Move fixed points to retr/slip/node positions.`

Sometimes basic belt path fixed end points do not lie exactly at the "structural" position of the (old) as-meshed belt. Typically such points should be at slipring, retractor or structure node positions, but sometimes the end sections of belt meshes are adjusted by hand after fitting, leading to a small difference between "basic path" and "as-meshed" belt element end positions.

For auto-refitting to produce good shapes it is usually necessary in these cases to move the end point of the basic path onto the revised "as-meshed" position, and it is recommended that this option is left selected.

`Find free end point nodes on structure.`

When the original belt mesh is checked a search is made for any retractors or sliprings at the end of sections so that they can be reused. However it is normally the case that one end of the belt is bolted directly to the vehicle or seat structure, typically at the lower outside end, and this point will not have a retractor or slipring.

If this option is selected then a search will be made for a structural node near to the path end point, with preference being given to a node used in the topology of the end belt element, and this node will be used to terminate the new belt mesh. Again it is is recommended that this option is left selected.

`Max refitting #iterations.`

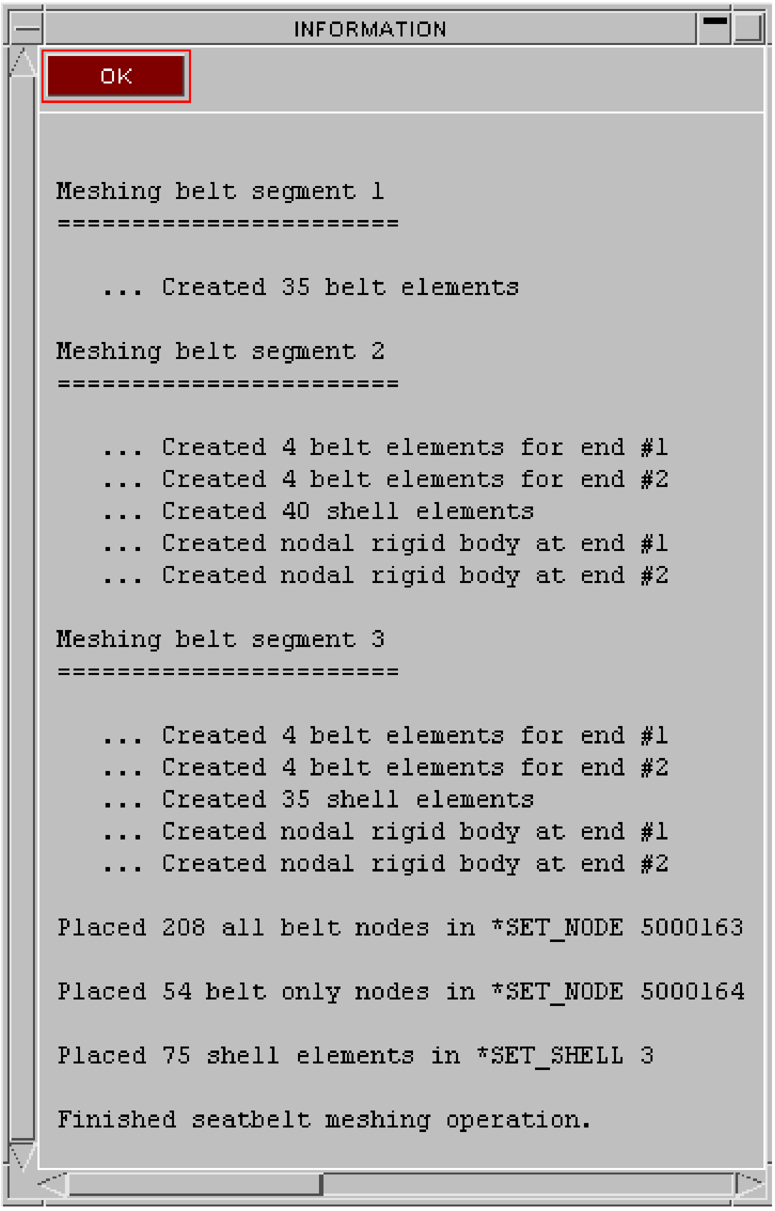
This simply acts as an upper-bound on the number of iterations the form-finding algorithm will go through when attempting to find a new shape. If the differences between old and new dummy positions are very great it is sometimes the case that the automatic update of the basic path will produce an initial shape that cannot give a solution, and this limit will stop it iterating for ever! The default value is 300, but it can be changed to any sensible value.

APPLY - executing the auto-refit.

Once you have selected the appropriate options press **APPLY** and the refitter will do the rest.

Assuming that all phases work correctly it will fit and create a new mesh, delete the old one (if this option is selected), and the refit will be complete. The only task remaining is to sort out contact between the new belt and the dummy.

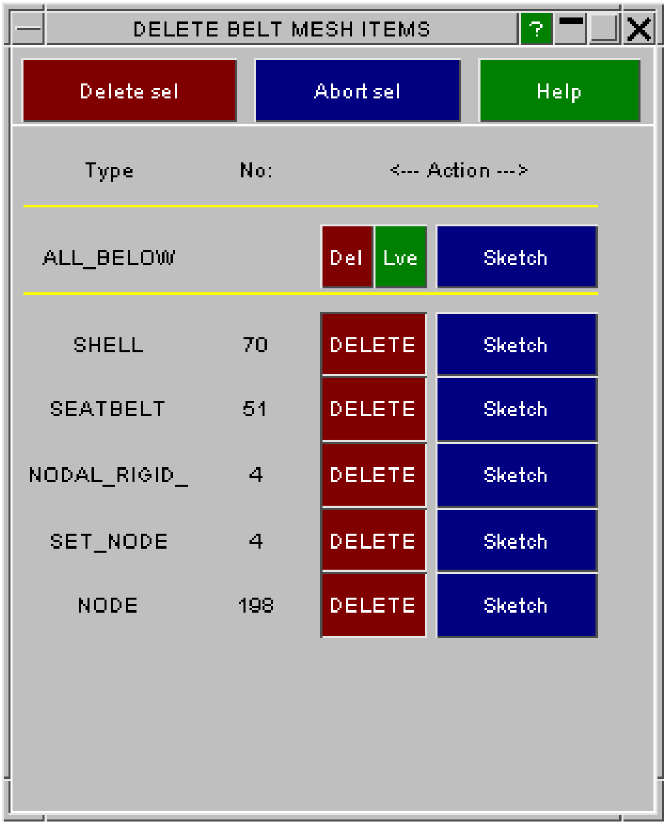
In the example above the following was produced.



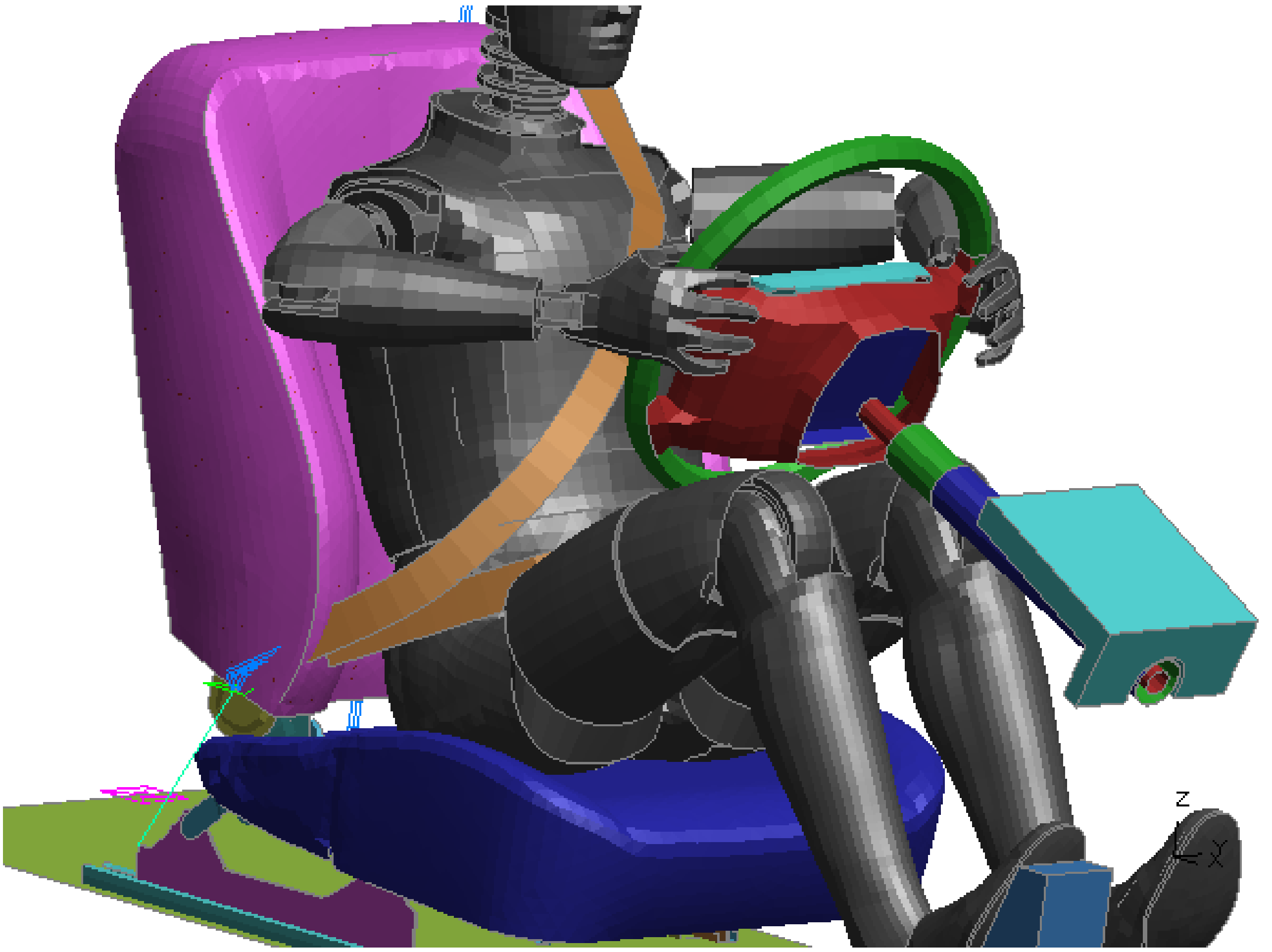
"Delete old belt items" was selected, so the original mesh was also marked for deletion.

Note that this goes through the normal PRIMER deletion logic, so you must confirm it and you can cancel deletion of the old mesh even at this stage if you wish to.

Note also that the 2 sliprings and 1 retractor used in the original belt have not been marked for deletion, since they have been reused.



Finally the image is updated to show the new mesh:



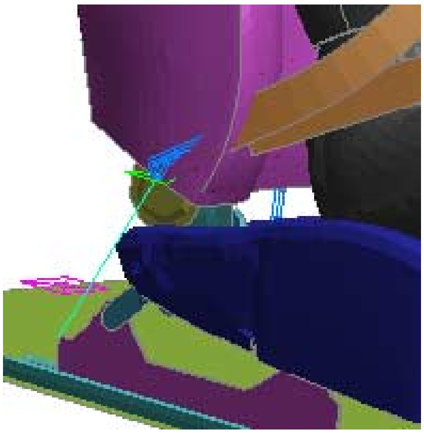
If it goes wrong ...

Hopefully it won’t, but common problems are:

Attachment points are no longer in valid positions

This (artificial) example shows a common problem with belt remeshing following dummy movement: the stalk for the belt buckle has been moved with the seat assembly below it, but this has been insufficient to make the remeshed belt clear the seat back cushion and the belt penetrates it.
It has happened here because the seat base has moved down by a significant amount but the base slider, to which the belt stalk is attached, has only moved forwards and the stalk has not rotated forward to account for this.
The vertical movement here is a bit excessive (the dummy looks very uncomfortable in the picture above) but it illustrates the point.

It is a good idea to try to visualise where the end points of a newly refitted belt will end up before refitting, in order to avoid this sort of problem, and some manual movement of belt end positions may be required to prevent it.

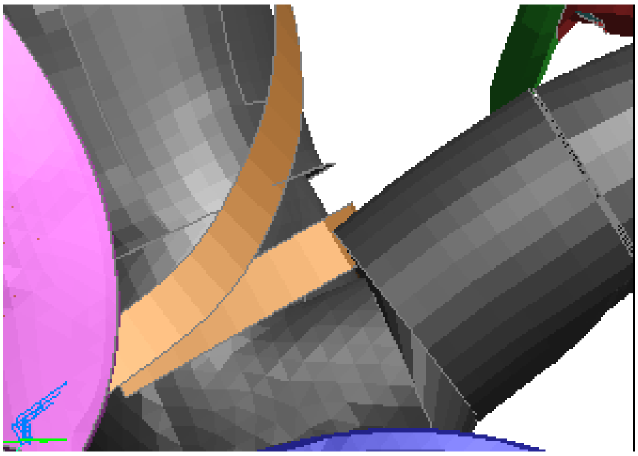


The dummy has moved so far from its original position that the "structure" definition needs updating.

A common problem when dummies are moved forwards is that their upper legs move upwards, and belt fitting may need to consider contact between base path and legs where before, in the original position, this was not necessary. This example shows the problem: the upper legs are not in the "structure" definition, so were not considered for contact during refitting and are penetrated by the belt.

Adding the parts of the upper legs to the belt "structure"and repeating the refit may solve this problem.

However very large movements from the initial position may simply not work, in which case it will be necessary to abandon automatic fitting and revert to manual refitting of the belt from scratch.



The belt was originally fitted, then the whole dummy and belt was moved to a new position leaving its "base path" definition behind.

In this situation attempting to reproject from the base path almost certainly will not work because its relationship to the dummy’s new position is invalid. It might be possible to "Orient" the dummy base path to re-align it with the dummy’s new position, but it will probably be quicker just to refit the belt manually.

Trying repeated **Auto-Refit** operations

You can go through the cycle:
- Try a refit
- Adjust some part of the geometry or belt definition
- Try another refit

As many times as you like. Each cycle will delete the previous belt and repeat the fitting procedure with the revised definition, making it easy to adjust the solution to get the desired result.

6.28.8 Saving Seatbelt Definition data to file, and its use for re-meshing

Seatbelt definitions are automatically saved in LS-DYNA format output files by appending extra keywords after the LS-DYNA *END card. The keywords are:

`*BELT_START`	Giving label, title, structure sets and other key information.
`*BELT_MESH`	Giving the dimensions and parameters used during meshing.
`*BELT_PATH`	Giving the coordinates and attributes of all basic path points.
`*BELT_END`	Terminates the definition.

Details of the format are given in Appendix V, but users should avoid editing these sections since errors may cause internal inconsistencies.

Deleting these sections from the end of a file is legal: the analysis will still run, but PRIMER will not "know" about the belt definitions when the file is reread.

Remeshing existing belt definitions

When a file containing this extra **`*BELT_...`** data is read back into PRIMER the belt definitions will be created automatically. This makes it possible to re-mesh a seatbelt either automatically or by hand.

Auto-Refit: Remeshing a belt automatically

This is the preferred method, and will recreate both belt and any sliprings and retractors in the new dummy position. It is covered in the section 6.28.7 .

Manual refit: Remeshing the belt by hand.

It is possible to repeat the refitting and remeshing process by hand as follows:

1. Only the basic path is stored in the file, so it will be necessary to repeat the **FITTING** operation to obtain a "chassis" mesh before remeshing can take place, see "Changing and remeshing an existing belt definition" in section 6.26.4 for more details.

2. It is not possible to reposition a belt by "adjusting existing nodal coordinates", even if the amount they need to move is small (although it might be feasible to use **ORIENT**, **TRANSLATE** instead with suitable interpolation). You must generate a new mesh each time, usually deleting the old one.
   The reason is that since a characteristic element length is used, moving the chassis mesh by even a small amount may change the number of elements in a segment, and then mapping the new shape onto the old mesh would be impractical.

3. Manual intervention during remeshing will usually be required at retractor and slipring locations, since these element types will "lock" connected seatbelt elements preventing their deletion. The simplest solution is to create the new mesh anyway, then to edit the retractor locations replacing the "old" seatbelt elements with the "new" ones, and then to delete the redundant old ones.

Now that automatic refitting has been added to Primer it is recommended that this be used instead.

6.28.9 Command-line (batch) commands for seatbelt fitting

A limited subset of the operations described above can be performed in command-line mode. These are intended to permit batch mode refitting of an existing belt definition, and they also include the ability to modify some aspects of belt geometry and fitting.

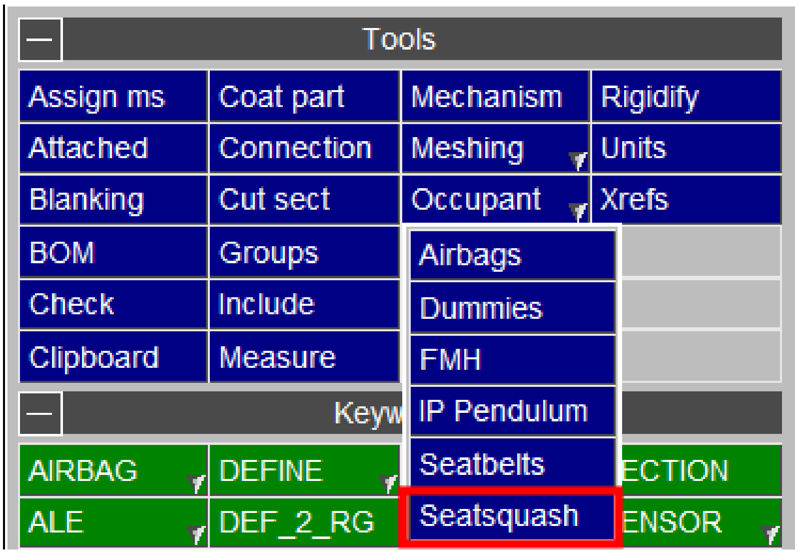
BELT	Refit an existing seatbelt to a dummy. Primer will select a seatbelt definition in the model automatically. Available options are: **SELECT** — Select a different belt definition for refitting **REFIT** — Refits the current belt definition to its dummy **PR_BASIC** — Retrieve and update basic belt dimensions and form-finding parameters **PR_REFIT** — Retrieve and update belt refit parameters **DONE** — To return to the main menu prompt

6.29 SEAT FOAM COMPRESSION

Positioning a dummy to its H-point position will generally lead to penetration with the seat foam components.

In PRIMER release 9.3, the user has the capability to deform the foam under the dummy and remove seat foam penetrations.

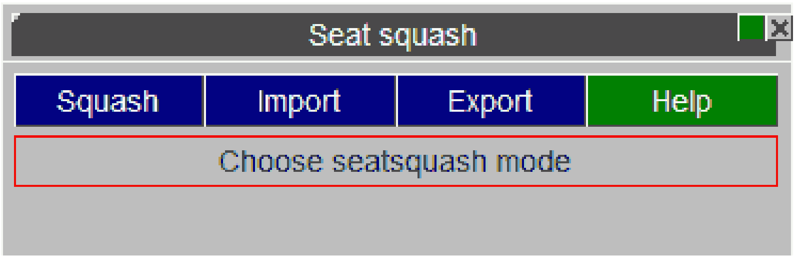
To access the Seat squash menu, click on the **Occupant** button and, in the drop-down menu, select the **Seatsquash** tool.



Three options are available in the seatsquash panel.

- **Squash**
- **Import**
- **Export**

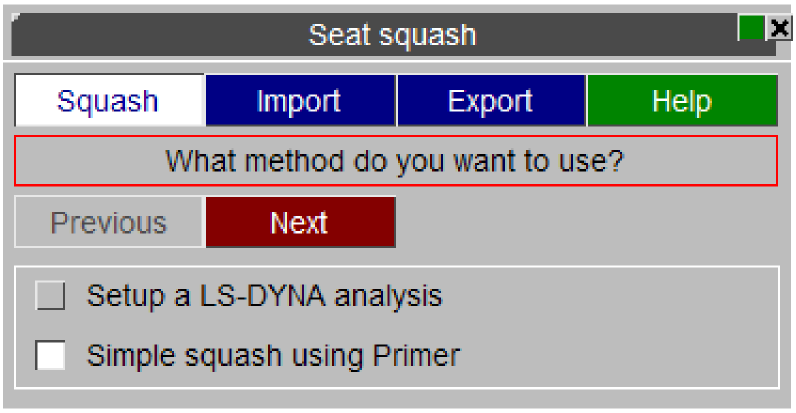
They are described in the following sections.



6.29.1 Undoing a seatquash operation

There is no ’undo’ button in the seatsquash. However, if before you start doing a seatsquash you export the coordinates of the seatfoam (and other parts if required) you can undo the seatsquash by importing these coordinates back into PRIMER.

6.29.2 Squash options



Press the **Squash** button to start a seatsquash.
Two methods are available to deform the seat foam:

- Setup a LS-DYNA analysis
- Simple squash using Primer

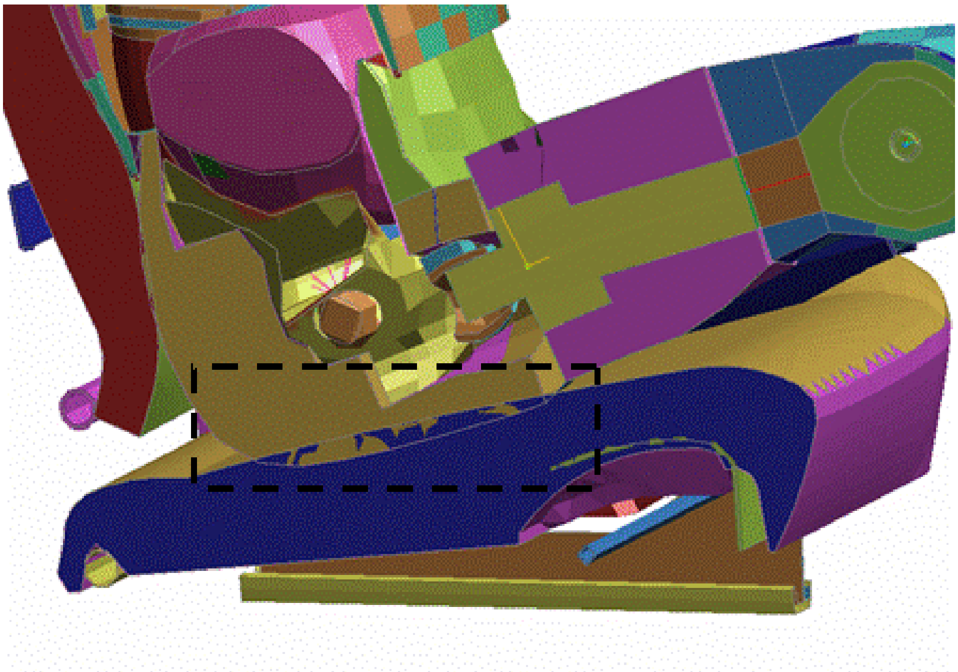
The PRIMER method uses the contact depenetrator in PRIMER to push the dummy into seat. The seat is deformed uniformly through its thickness. This obviously will not have the correct material response but it is meant as a quick method. If the seat deformation is critical then you should use the LS-DYNA method.

The LS-DYNA method will create an LS-DYNA import deck which will push the dummy into the seat. This should be run using LS-DYNA and the dynain file which it creates can be imported back into PRIMER to deform the seat.

Select the option you want and press **Next** to start the process.

Simple squash using PRIMER

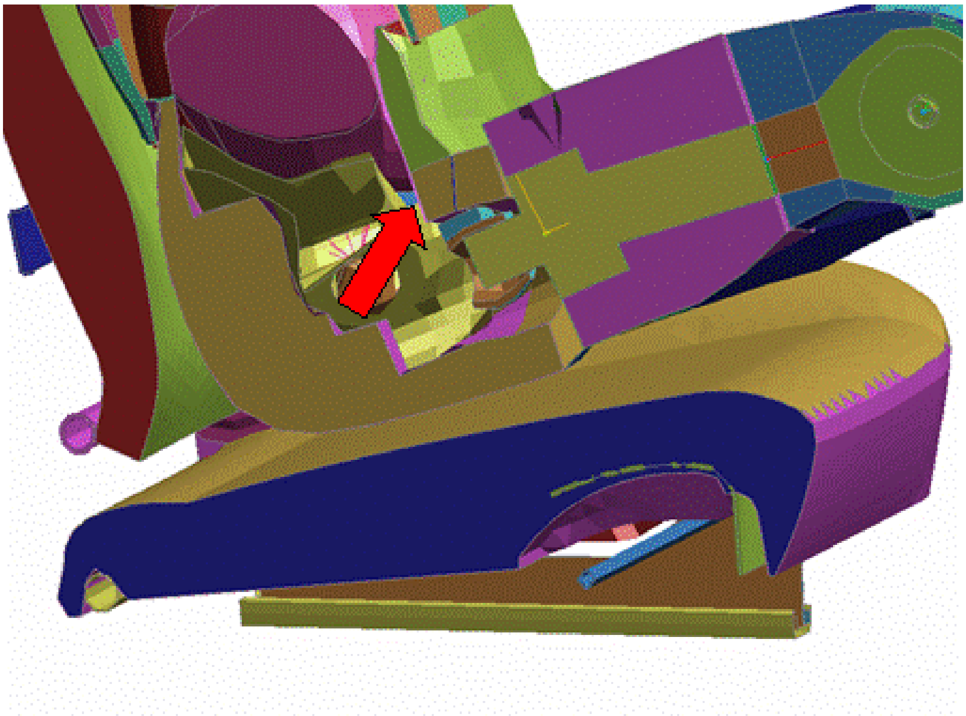
Process description



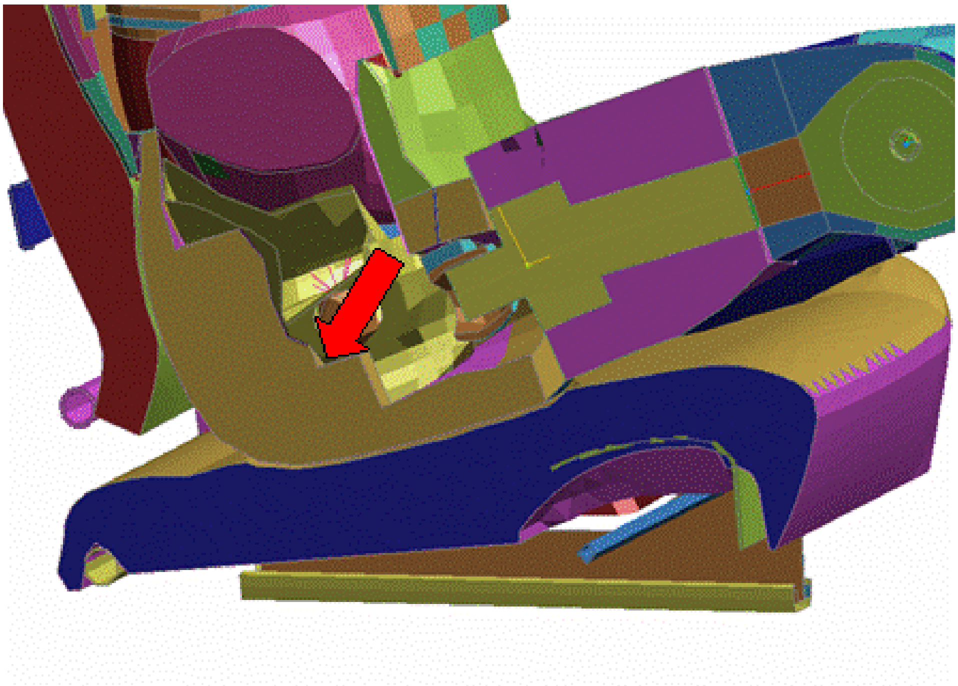
Initial state: model with penetration

Cut section through dummy and seat showing initial penetration between dummy components and seat foam.

The dummy is at the correct H-point but penetrates the seat.



The dummy is moved from its initial H-point position, following the direction prescribed by the user until the contact with the seat top shell part has no penetration.



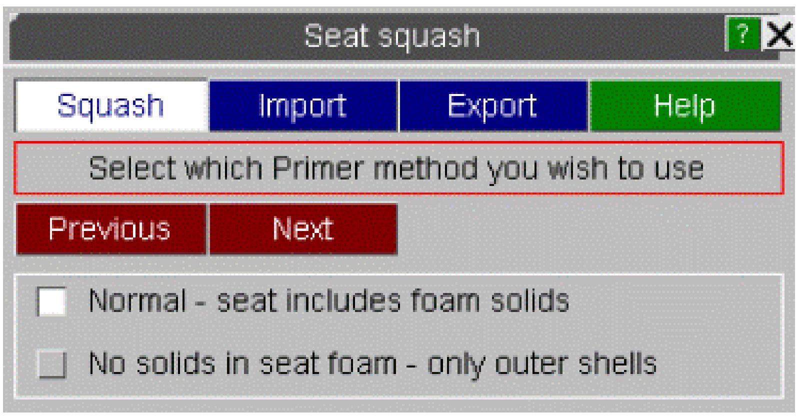
Final state: Seat foam compressed

The dummy is moved back to its initial H-point position by increments while using Primer contact depenetration option. The seat foam is progressively squashed under the dummy. Interior nodes within the foam components are also displaced to uniformly distribute strain.

Step 1

There are two types of simple seat squash. The first type is where you specify the solid elements in the seat, and the solids are deformed during the compression. The second type does not consider the seat solid elements. Use the second type if your model does not contain seat foam solids, and you just wish to deform the outer shells of the seat and mesh the solids after the deformation.

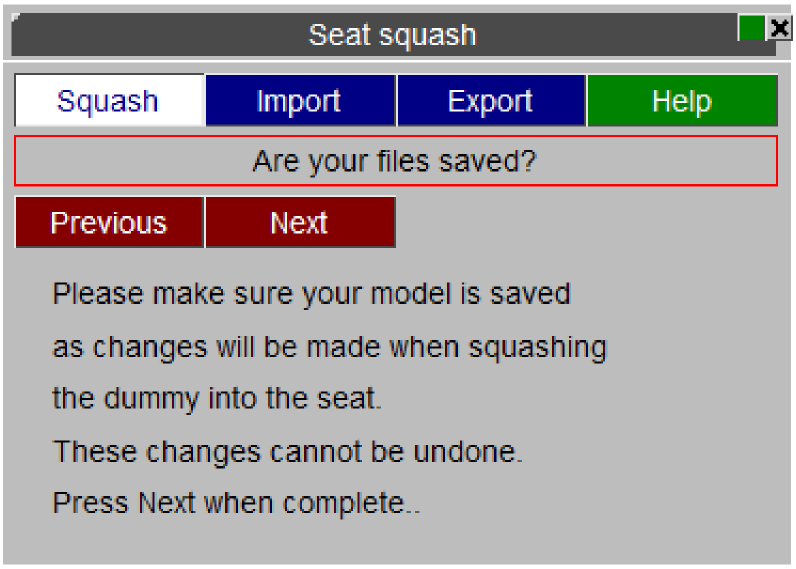
Once you have chosen the method you wish to use press **Next**



Step 2

Before you go any further you should save your model. PRIMER will prompt you to save your model as the seat squash changes are irreversible (although you can import coordinates from a dynain file to effectively do an ’undo’ See section 6.29.1 for more details).

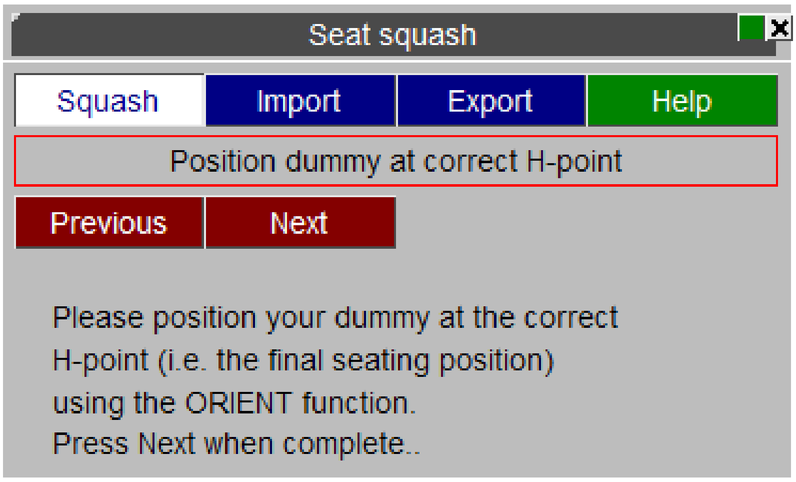
Once you have saved your model press **Next**



Step 3

You will be asked to move your dummy to the correct H-point location. Position the dummy by either using the orient menu or dummy positioning menu.

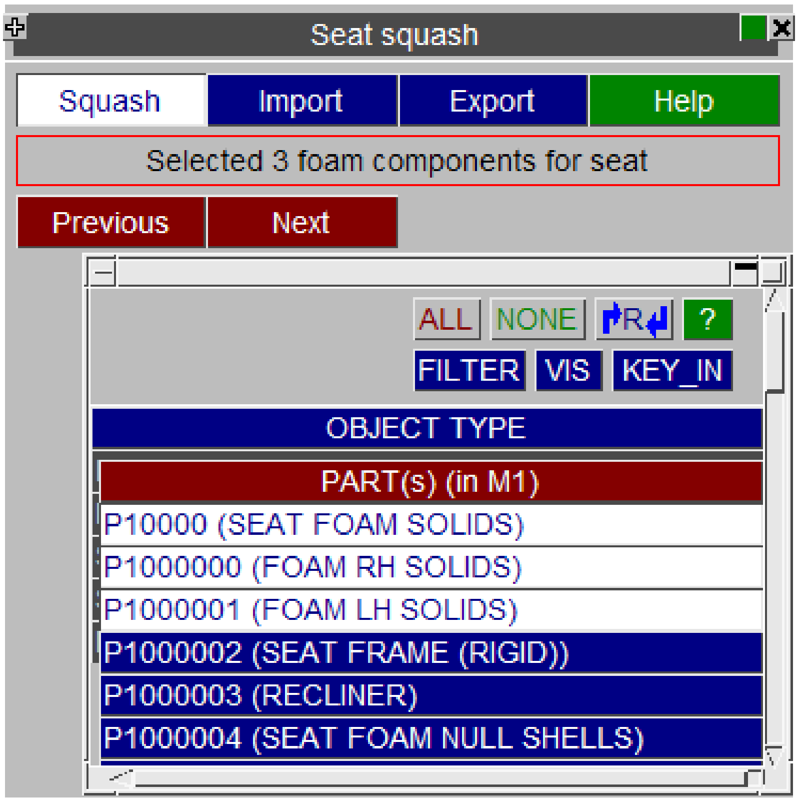
Once this is done press **Next**



Step 4

Select the foam parts of the seat using the standard object menu. These are the parts that PRIMER will squash the dummy into. This step is only available if you have chosen the "normal" method in step 1.

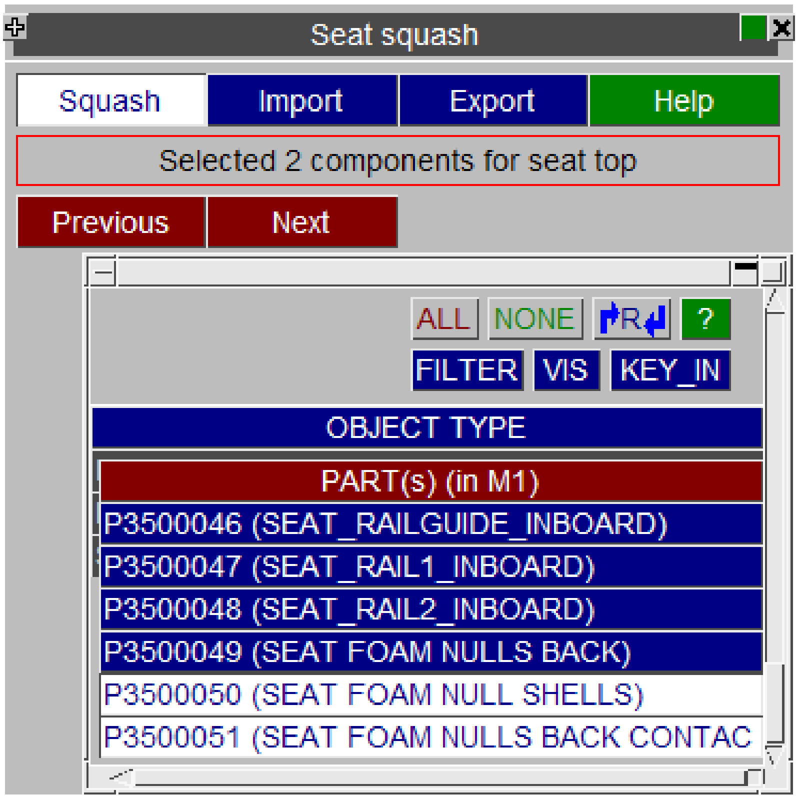
To continue press **Next**



Step 5

Select the coating shell parts on the top surface of the seats. These can either be defined by parts or by sets of parts.

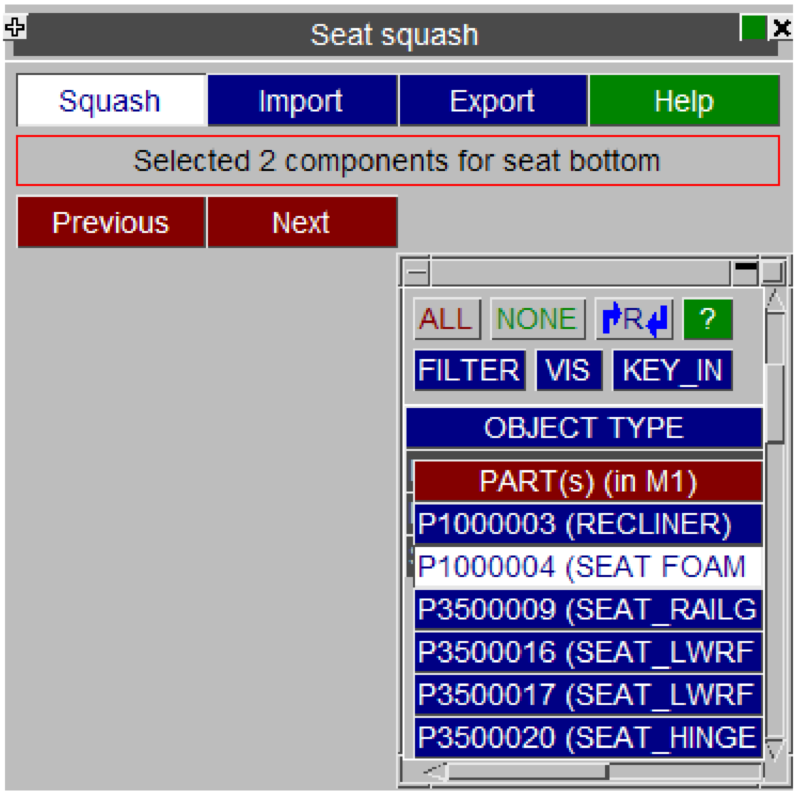
To continue press **Next**



Step 6

Select the coating shell parts on the bottom surface of the seats. These can either be defined by parts or by sets of parts. These parts will be fixed. PRIMER Will deform the seat evenly as required between the top and bottom surfaces of the seat.

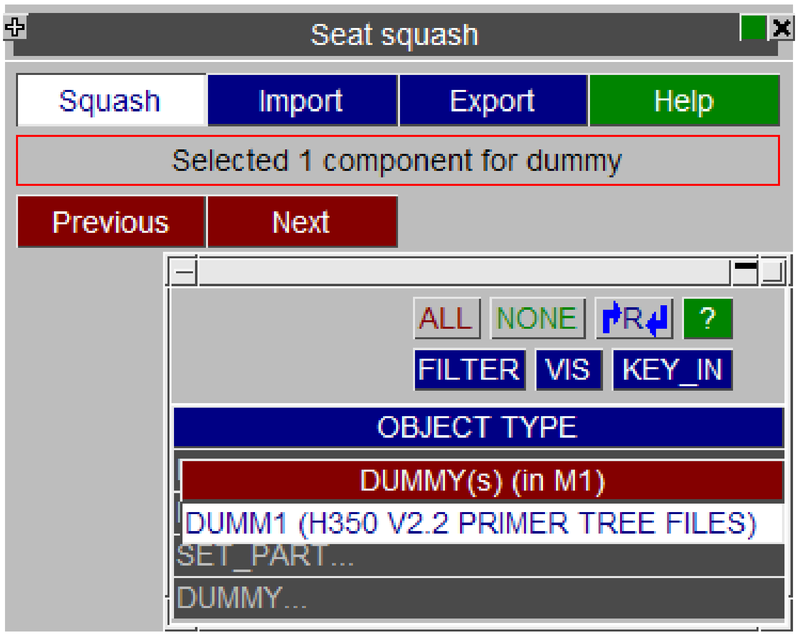
To continue press **Next**



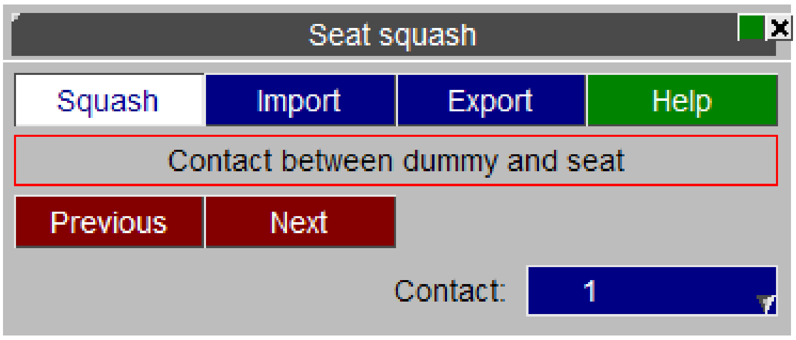
Step 7

Select the dummy components. You can use the **DUMMY...** option in the standard object window if required..

To continue press **Next**



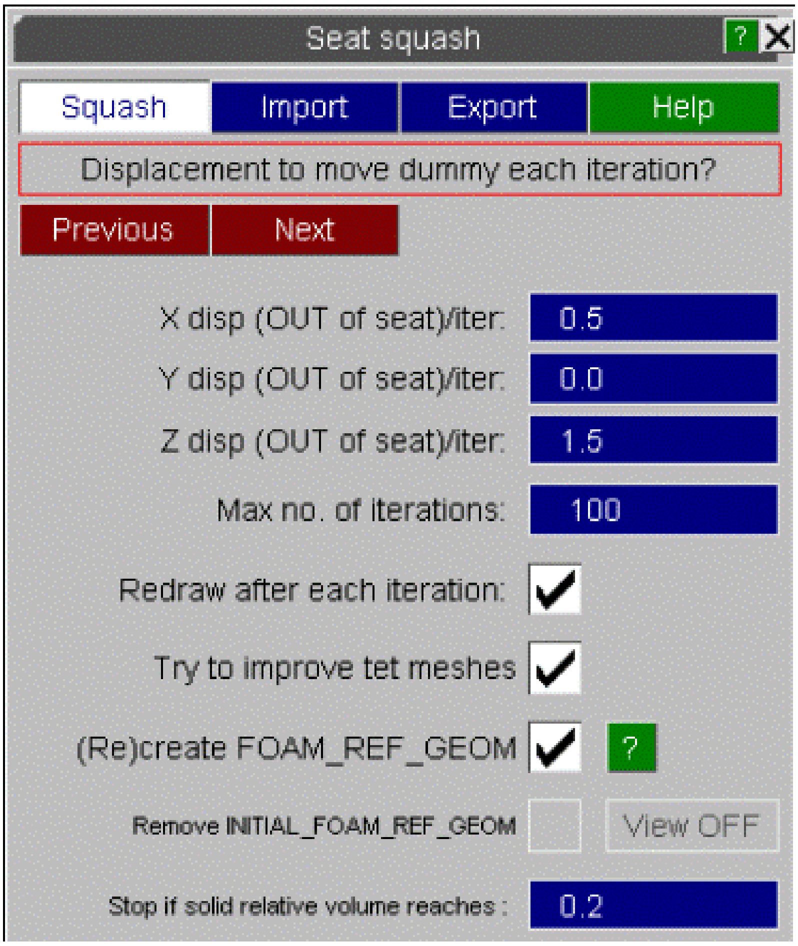
Step 8



To select the contact between the seat foam and the dummy. You can use the standard popup functions (right click). You can then Pick, Select an existing contact or Create a new one. Make sure that your contact uses a sensible thickness. This is what PRIMER will use when pushing the dummy into the seat. If the thickness is very small then you will have to have a small increment per iteration.

Once you have selected/created the contact press **Next**

Step 9



Define the dummy displacement increment at each de-penetration iteration. Give the X, Y, and Z displacements that the dummy will move per iteration to move the dummy out of the seat. Once PRIMER has moved the dummy out of the seat enough to eliminate any penetrations it will reverse the motion, squashing the dummy back into the seat to the original position.
If the displacement per iteration is bigger than the contact thickness chosen in the previous step, PRIMER will scale it down.

You can set the maximum number of iterations that PRIMER will try to do when moving the dummy out of the seat.

If you want to see the progress of the seatsquash then select the **Redraw after each iteration** checkbox. This will make the process much slower so if you want, you can turn it off.

Some tet meshes can be very badly deformed making it very easy to make badly deformed elements. PRIMER can try to ’smooth’ tet meshes to make them better. This may help if you are having problems squashing a dummy into a tet meshed seat.

You can opt to create *INITIAL_FOAM_REFERENCE_GEOMETRY cards for the nodes in the seat foam before the deformation. This is only available for hyperelastic materials and certain solid element formulations. Note the REF field on the appropriate material card will be set to 1.0 upon creation of the *INITIAL_FOAM_REFERENCE_GEOMERTY cards.

The minimum value of relative volume for the seat foam solid elements is by default set to 0.2. If any solid element becomes excessively deformed and reaches this threshold, the seat squash process will stop. You can modified this value if you wish.

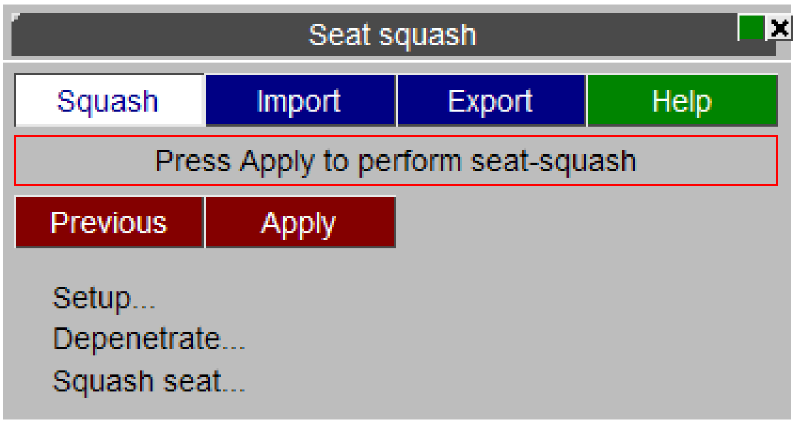
Then press **Next**

Step 10

Press **Apply** to start the process. First, the dummy will move away from the seat according to the displacement you prescribed until the contact between seat and dummy is fully de-penetrated.

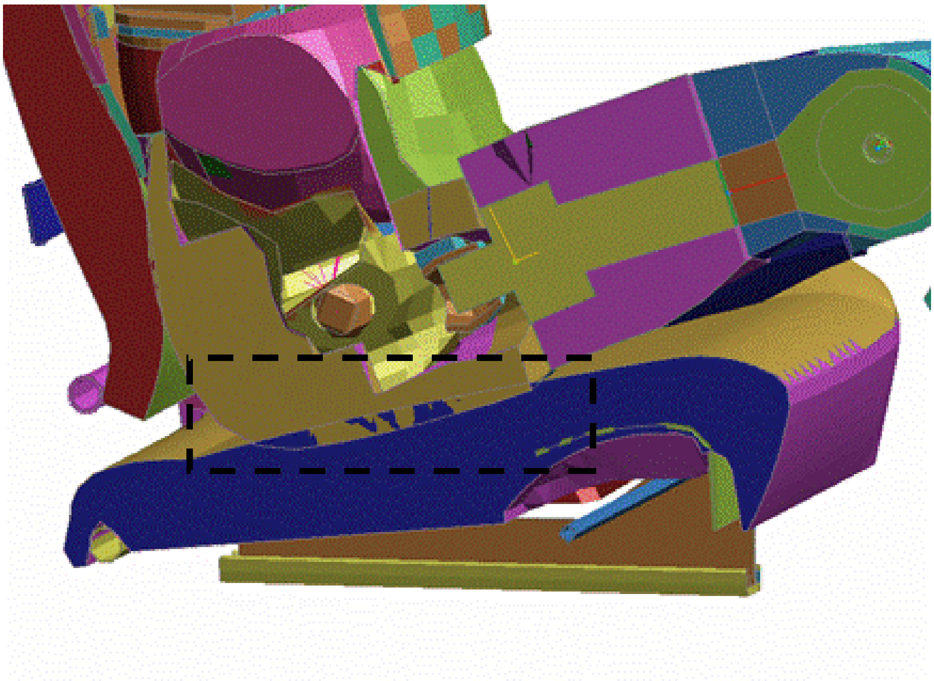
The dummy is then moved back to its original H-point position while compressing the seat foam.

Once finished you can save your model and/or export coordinates if required for use in other analyses.



LS-DYNA seat squash method

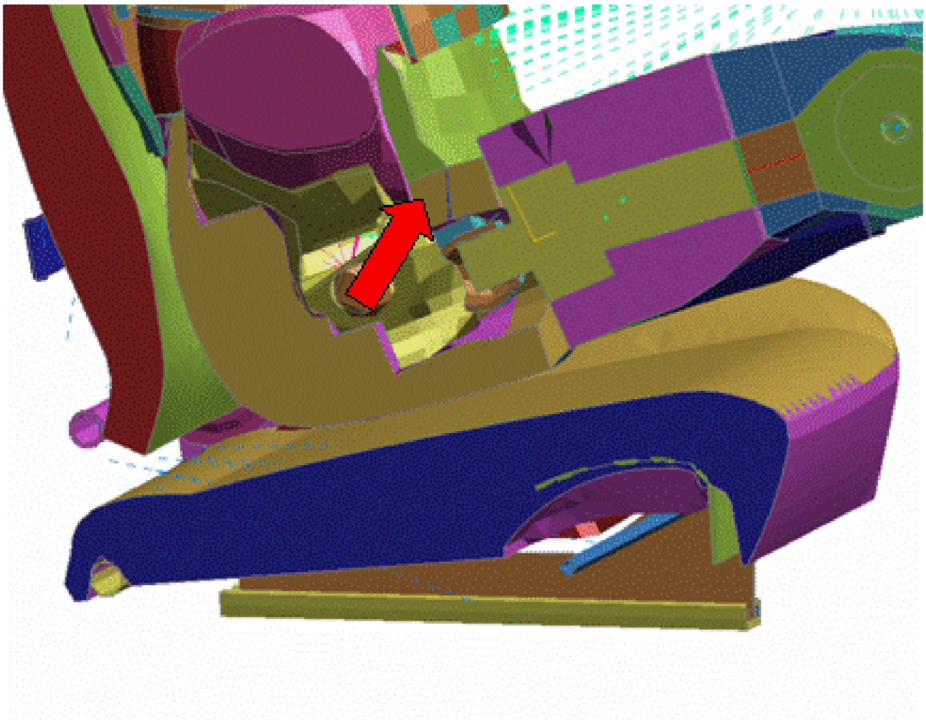
Process description



Initial state: model with penetration

Cut section through dummy and seat showing initial penetration between dummy components and seat foam.

The dummy is at the correct H-point but penetrates the seat.



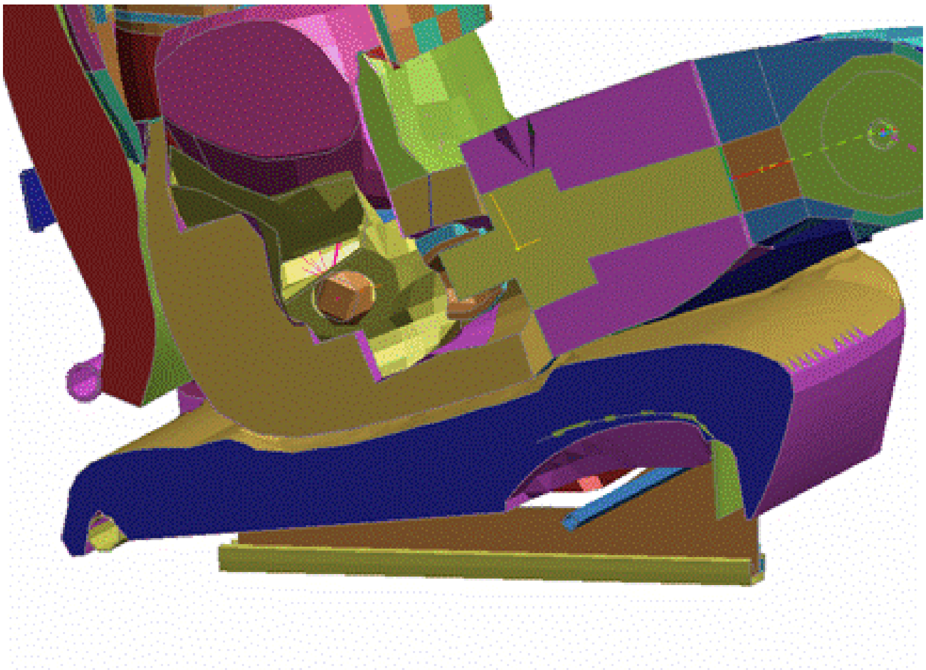
The dummy is moved from its initial H-point position, following the direction prescribed by the user until the contact with the seat top shell part has no penetration.

The dummy and the seat components that you defined as non-deformable are then rigidified.

Any contacts that you identify as redundant are then deleted.

The complete input deck for the LS-DYNA seat-squash analysis is set-up.

The user tidies, checks and modifies the LS-DYNA input file as required and then runs the analysis.



Final state: Seat foam compressed

When the LS-DYNA analysis terminates it will write a dynain file that contains the coordinates and initial stresses for the seat foam (and possibly other parts too)

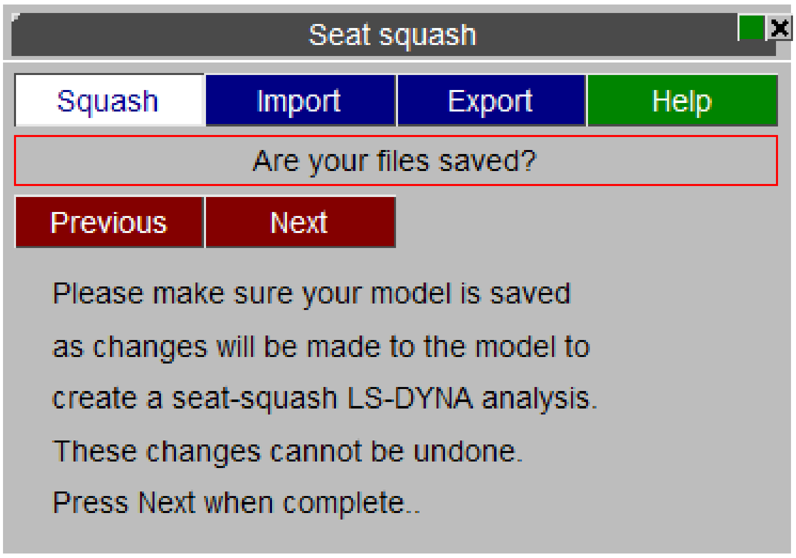
The user imports the data written out in the dynain file. The model now contains the deformed geometry and initial stress of all the **DEFORMABLE** parts

The seat foam components are now in their compressed state and the contact between seat and dummy is de-penetrated.

Step 1

Before you go any further you should save your model. PRIMER Will prompt you to save your model as the seat squash changes are irreversible (although you can import coordinates from a dynain file to effectively do an 'undo' See section 6.29.1 for more details).

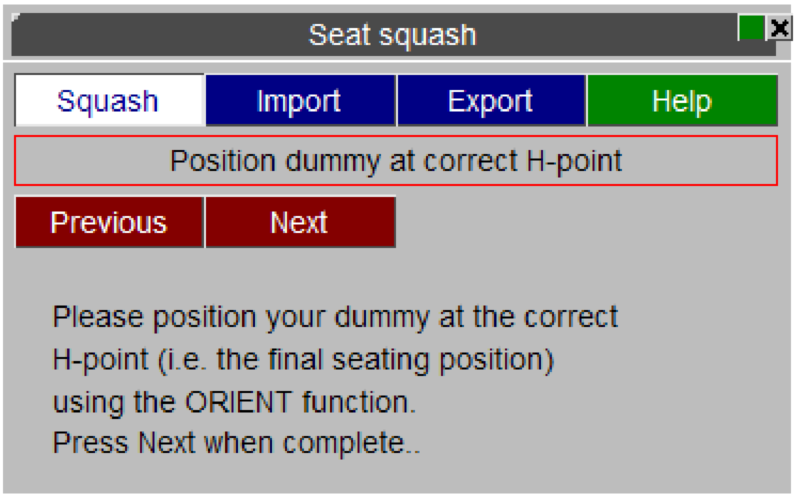
Once you have saved your model press **Next**



Step 2

You will be asked to move your dummy to the correct H-point location. Position the dummy by either using the orient menu or dummy positioning menu.

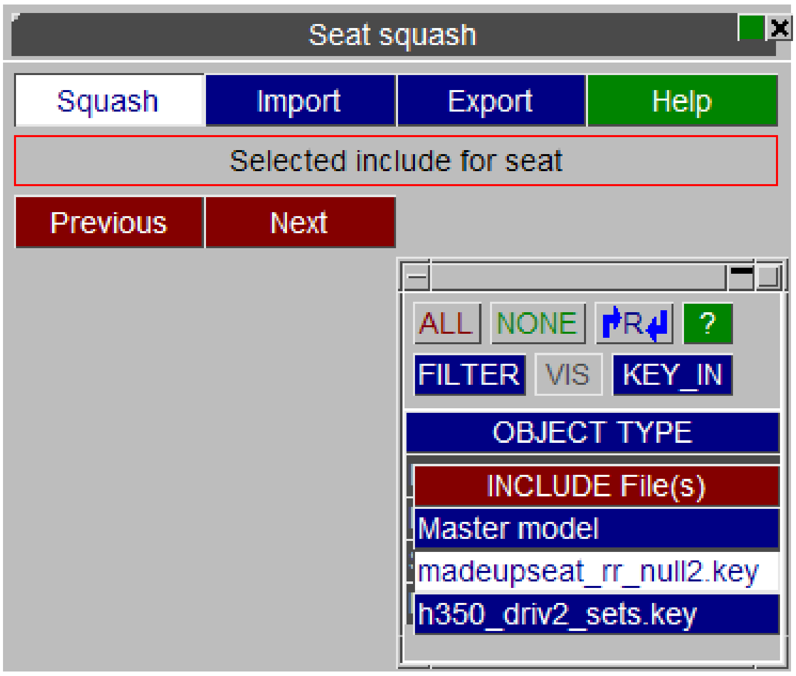
Once this is done press **Next**



Step 3

Select all of the parts that make up the seat using the standard object menu.

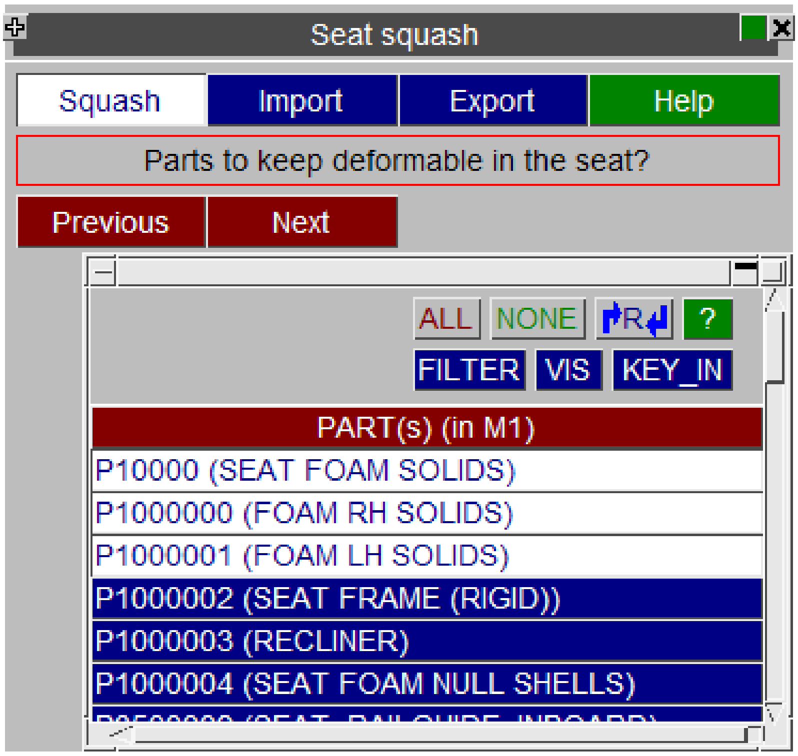
To continue press **Next**



Step 4

Select the all the **DEFORMABLE** parts of the seat structure using the standard object menu. PRIMER Will automatically select any parts that use a foam material. Typically, you should select the foam components and null shells on the surfaces. You can add and/or change this selection as required. PRIMER Will rigidify any parts that are not deformable to make the LS-DYNA analysis quicker.

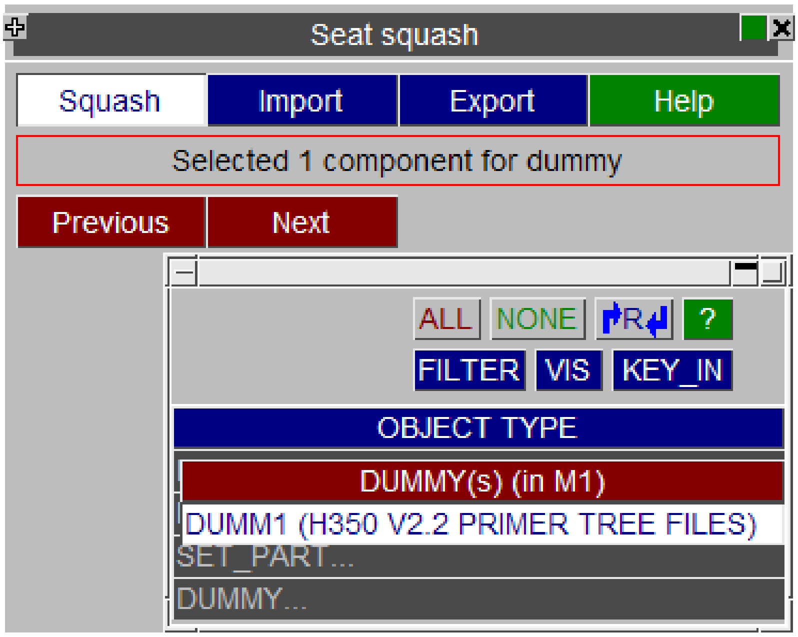
To continue press **Next**



Step 5

Select the dummy components. You can use the **DUMMY...** option in the standard object window.

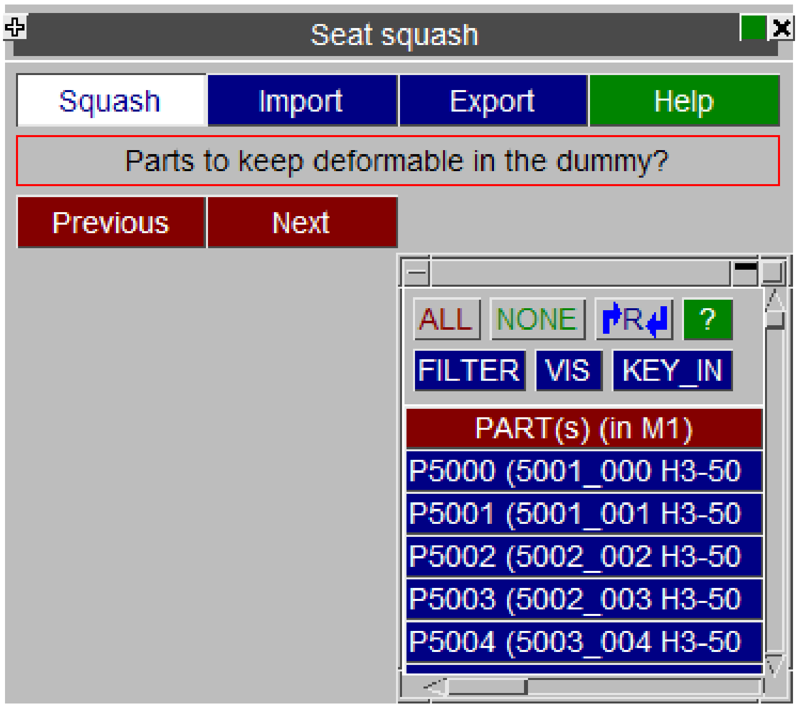
To continue press **Next**



Step 6

Select any parts of the dummy that you want to keep deformable using the standard object menu. Typically, you would not select any parts so the entire dummy is rigidified. However, you may want to keep some parts deformable so you can change this selection as required. PRIMER Will rigidify any parts that are not deformable to make the LS-DYNA analysis quicker.

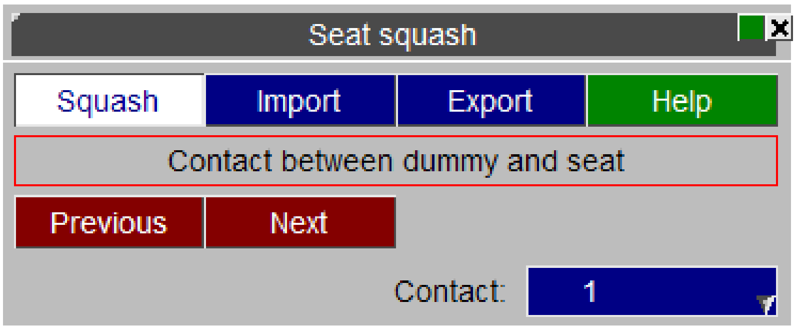
To continue press **Next**



Step 7

To select/create the contact between the seat foam and the dummy, you can use the standard popup functions (right click). You can then Pick, Select an existing contact or Create a new one.

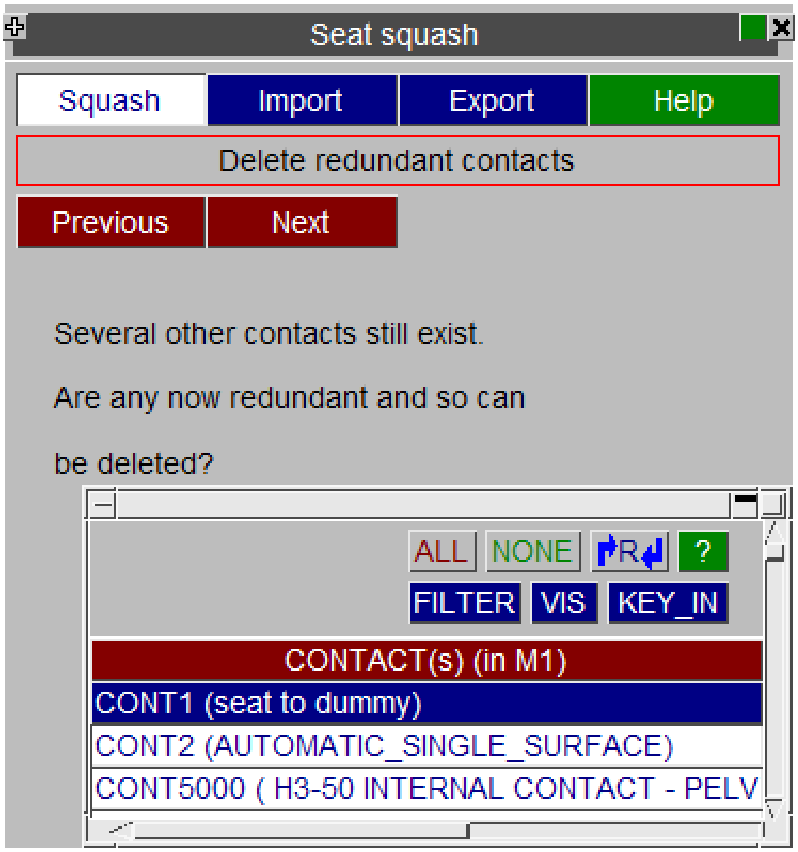
Once the contact has been selected/created press **Next**



Step 8

In the simplest seatsquash model the only contact that you will need is between the dummy and the seat. Any other contacts that are present in the model will just slow the analysis down. PRIMER Will prompt you to delete any contacts which it thinks are unnecessary. By default, all contacts except the contact defined in step 7 are chosen. Change this as required.

When the relevant contacts are selected press **Next**



Step 9

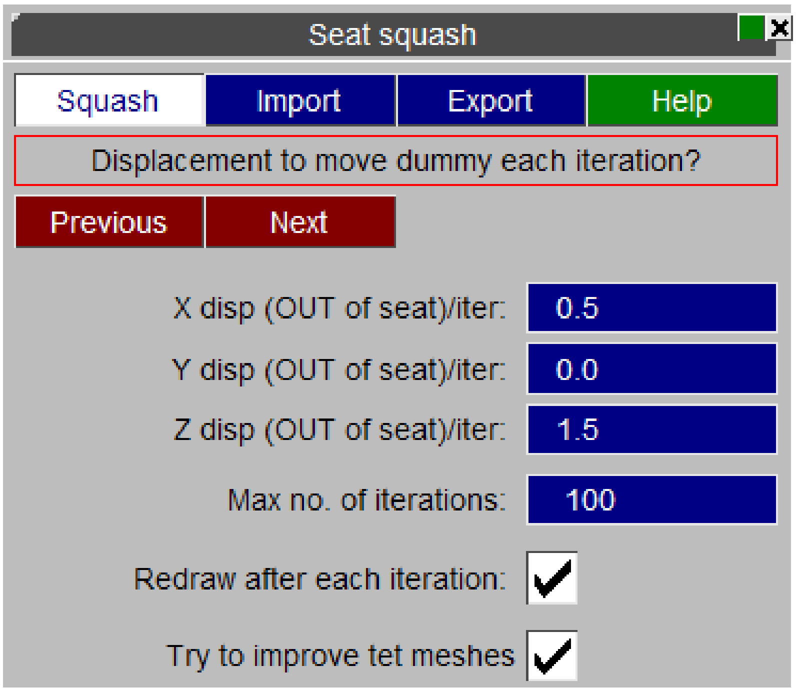
Define the dummy displacement increment at each de-penetration iteration. X, Y, and Z increments are given for each iteration to move the dummy out of the seat. As this is only going to be used to move the dummy out of the seat the iteration can be as large as you like.

You can set the maximum number of iterations that PRIMER will try to do when moving the dummy out of the seat.

If you want to see the progress of the seatsquash then select the **Redraw after each iteration** checkbox. This will make the process much slower so if you want, you can turn it off.

Some tet meshes can be very badly deformed making it very easy to make badly deformed elements. PRIMER Can try to 'smooth' tet meshes to make them better. This may help if you are having problems squashing a dummy into a tet meshed seat.

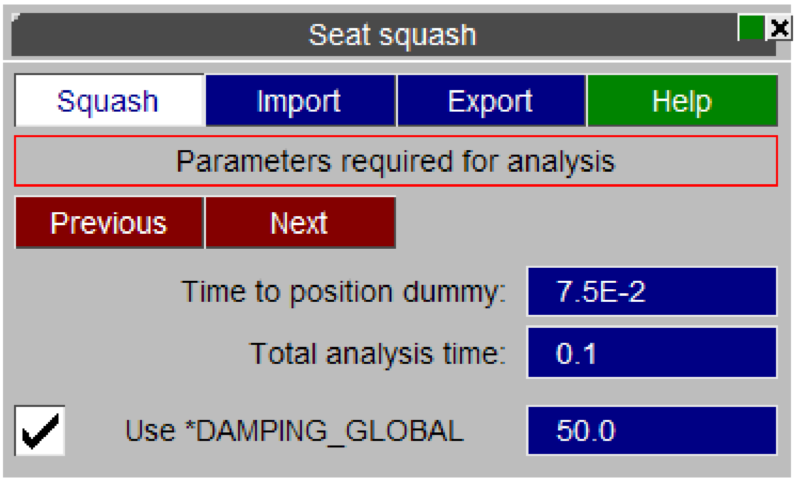
To continue press **Next**



Step 10

Setup the parameters of the LS-DYNA seat-squash analysis. In most cases, the default values setup in Primer should be appropriate.

To continue press **Next**



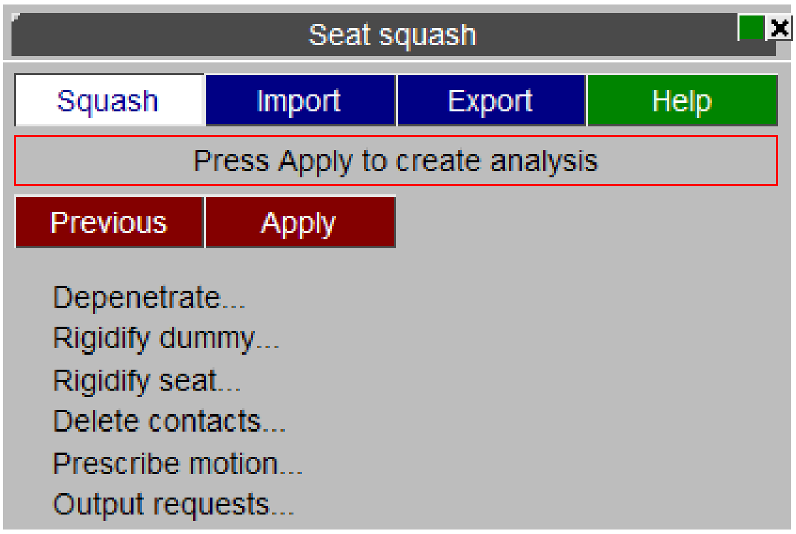
Step 11

Press **Apply** to start the process. First, the dummy will move away from the seat according to the displacement you prescribed until the contact between seat and dummy is fully de-penetrated.

The dummy and the seat components that you defined as non-deformable are then rigidified.

The contacts that you identified as redundant are then deleted.

The complete input deck for the LS-DYNA seat-squash analysis is setup



Step 12

Now review the LS-DYNA input deck that PRIMER has created, making any amendments you wish. Run the analysis using LS-DYNA and then import the required coordinates and initial stresses to your main model.

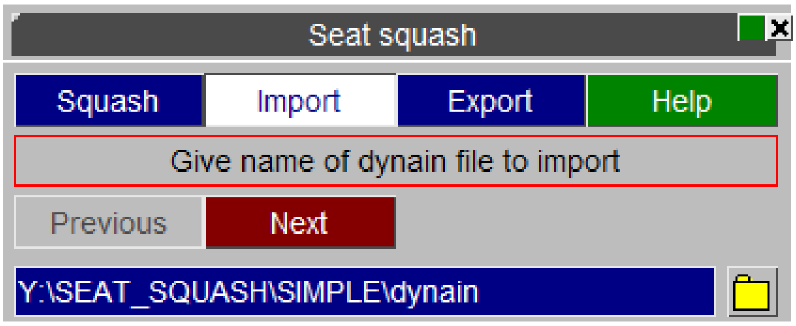
6.29.3 **Import** option

Step 1

Once the analysis is finished, read your original model, in which dummy and seat foam components have penetration.

Go back to the Seat squash panel, click on the **Import** button and read in the dynain file that LS-DYNA has written out. This file contains the final geometry and initial stress data of all the **DEFORMABLE** parts that you define at **STEP 4.**

To continue press **Next**

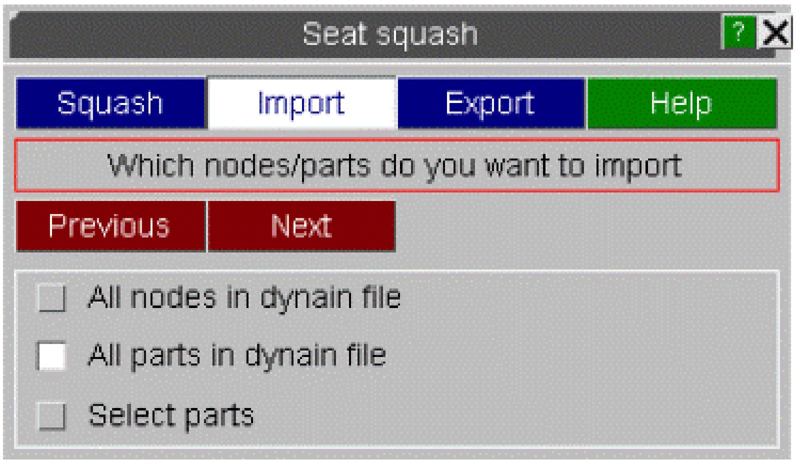


Step 2

PRIMER will then read the file into a temporary model.

Either read the data for **All nodes in dynain file, All parts in dynain file** or **Select parts** to choose the parts in the dynain file to read data from

To continue press **Next**

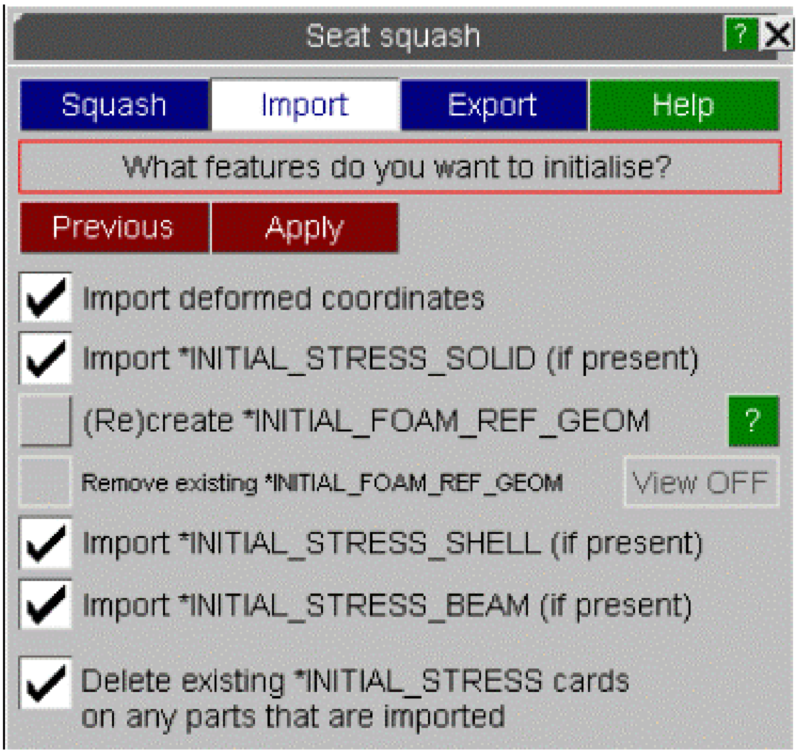


Step 3

Select the features that you want to initialise. (Initial stress)

To continue press **Apply** and **Finish**.

The seat foam components are now in their compressed state and the contact between seat and dummy is de-penetrated.



6.29.4 **Export** option

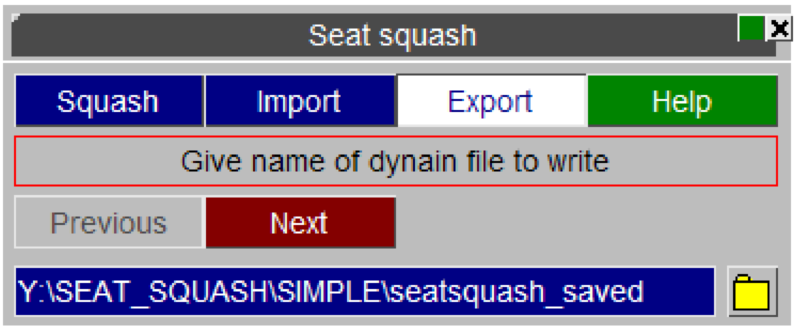
Step 1

The **Export** option allows you to write out nodal coordinates and initial stress data for parts in the seat.

There are three steps to follow to write out the data:

First select the name of the file that you want to write by either typing the name in the textbox or using the file selector.

Once done, press **Next**.

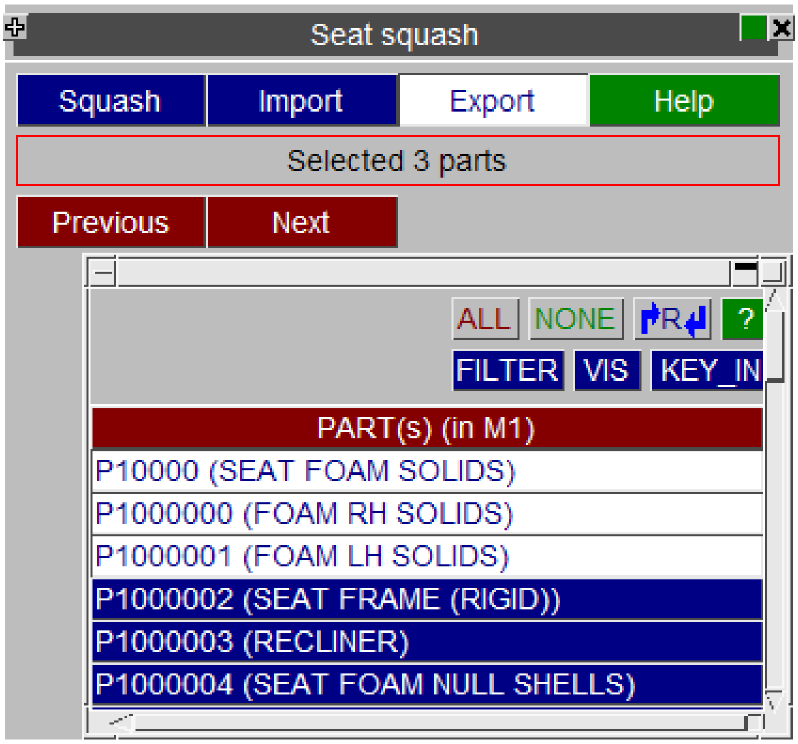


Step 2

Second, select the parts that you want to write to the file.

PRIMER Will show an object menu allowing you to select the parts you want. All of the normal options for object menus will be available. For example, if the seat is in an include file it may be helpful to first filter by include file so that only the parts for the seat are shown. Then if you only want to write out the data for the solid parts you could then additionally filter by element type SOLID or by a material type to further limit what is shown in the object menu.

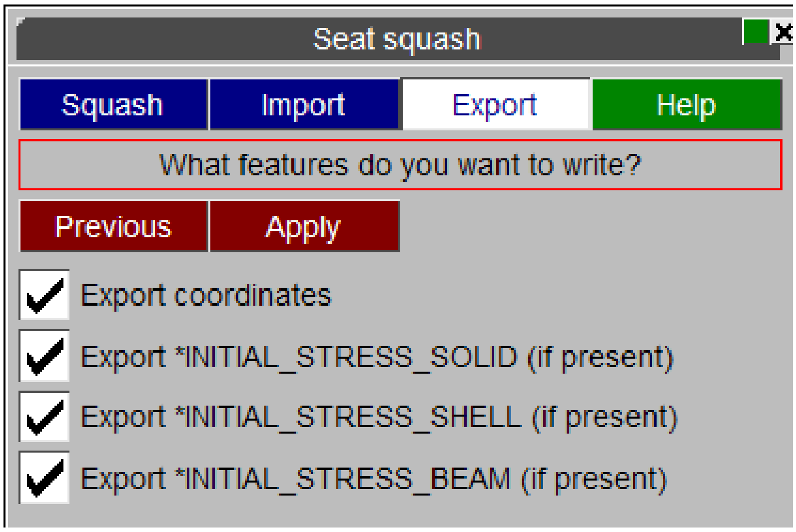
Once all of the required parts are selected press **Next**.



Step 3

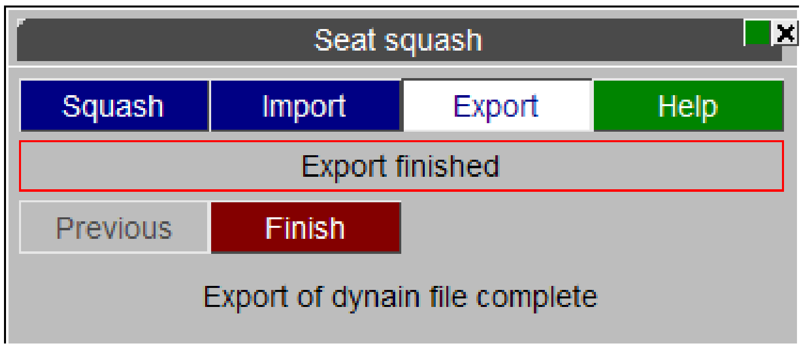
Now select what you want to write to the file by using the checkboxes.
A basic file could just contain the nodal coordinates. This could also be used as a way of undoing a seatsquash. If you save the coordinates of the seat foam nodes before doing a seatsquash you can import them again if required to 'undo' the seatsquash.

When you have the categories you want selected press **Apply** to write the file.



Step 4

When the file has been written press **Finish**.

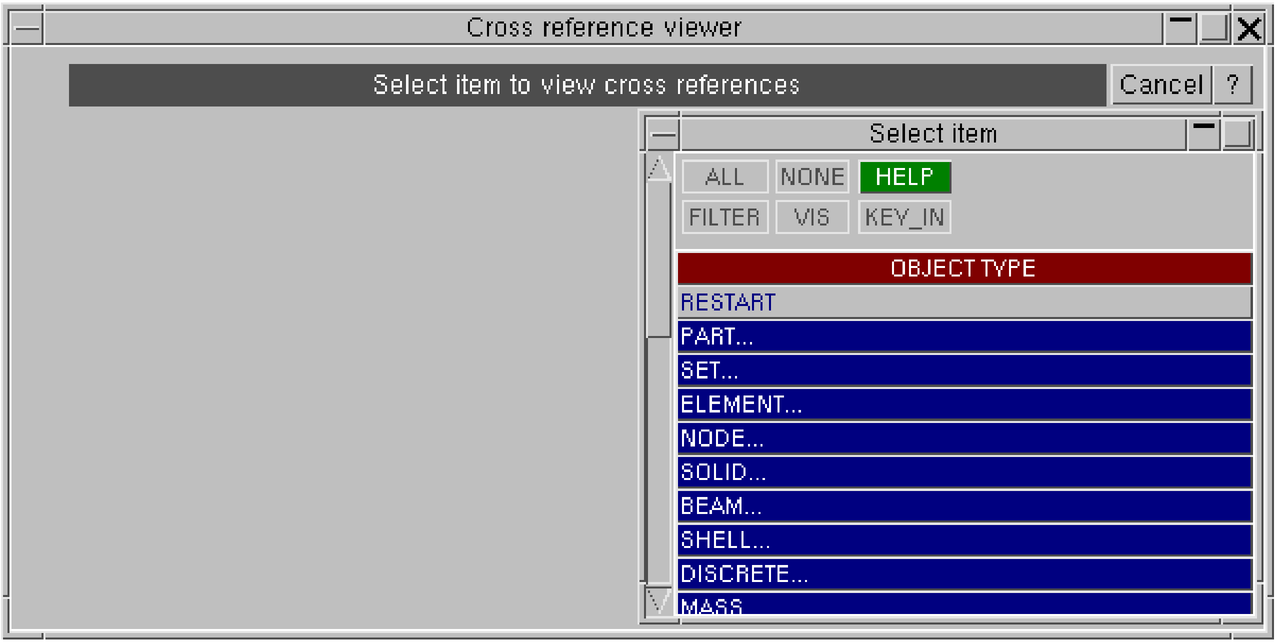


6.30 XREFS Cross references viewer

The cross reference viewer allows you to look at how a specific keyword is used. The viewer is started from the main **Tools** panel by the **Xrefs** button or in editing panels by the **VIEW_XREFS** button.

6.30.1 Selecting an item in the viewer

If the viewer is started from the main **Tools** panel you need to select the item to display. An object menu allows you to choose an item using the normal methods (selecting from list in menu, picking, filtering etc.)

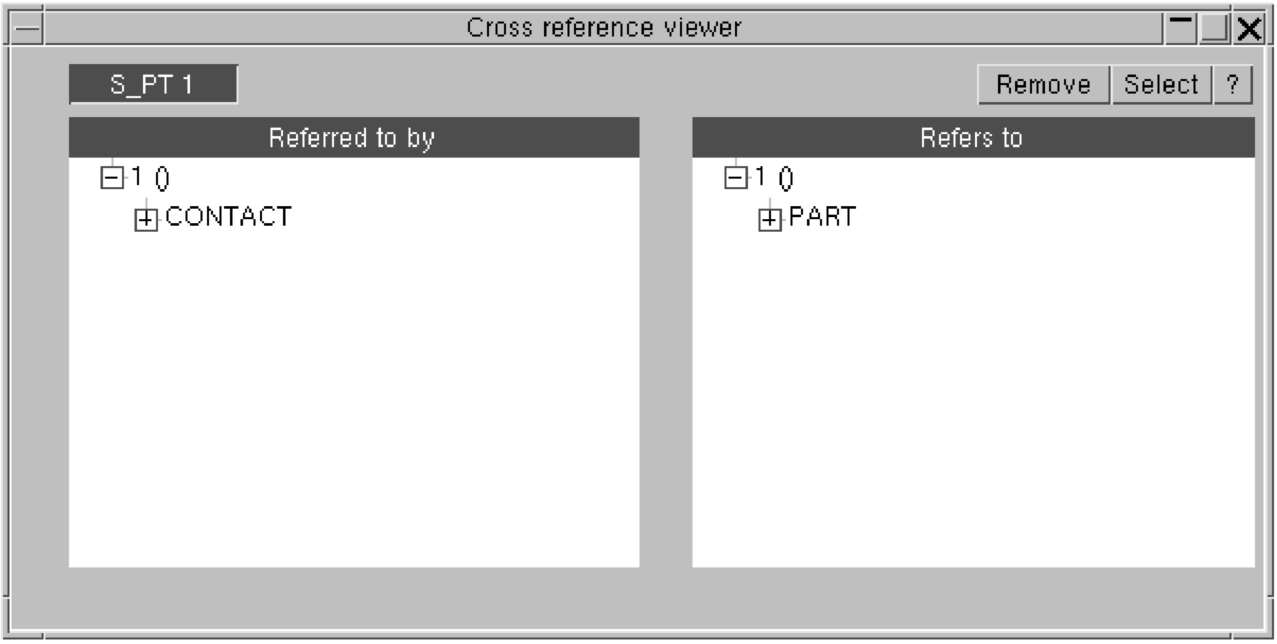


Once you have chosen an item the viewer will display the references for that item. **Cancel** quits the object menu and returns you to the normal cross reference display.

6.30.2 How cross references are displayed

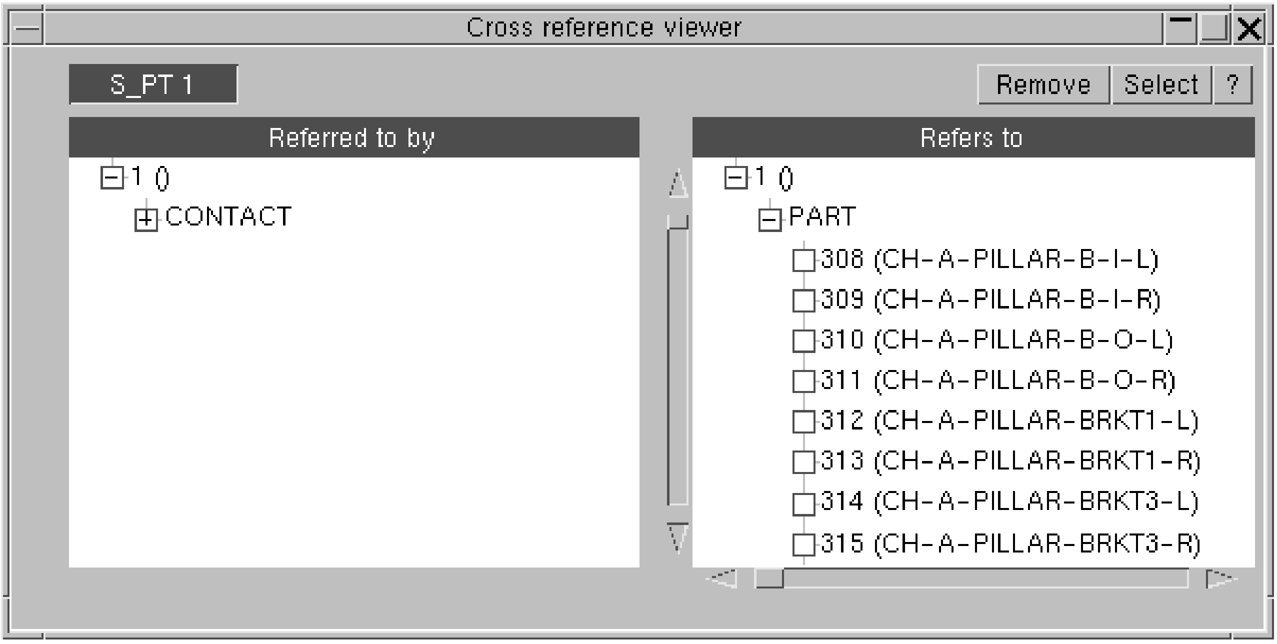
Cross references for an item are displayed using 2 'tree' views.
The left hand tree shows the items that are **referred to by the selected item**. For example below `*SET_PART 1` is referred to by `CONTACT` cards (as there is a `CONTACT` branch in the left hand tree).
The right hand tree shows the items that the **selected item refers to**. For example below `*SET_PART 1` refers to `PART` cards (as there is a `PART` branch in the right hand tree).

A tab is shown (**S_PT 1**) at the top of the panel for each item (only `*SET_PART 1` in this example). If multiple tabs are shown you can switch between them by pressing the tabs.

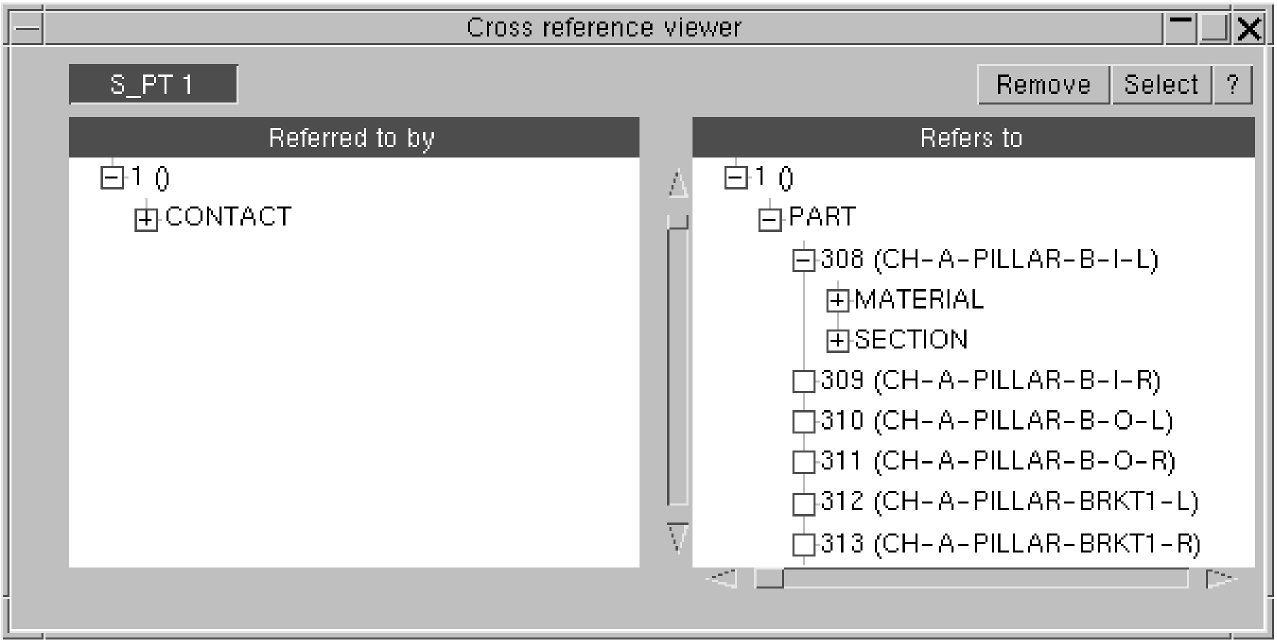


To expand a branch click on the ⊞ symbol. For example to see the parts that `*SET_PART 1` refers to click on the ⊞ on the **PART** branch. The branch is expanded and the parts are shown.

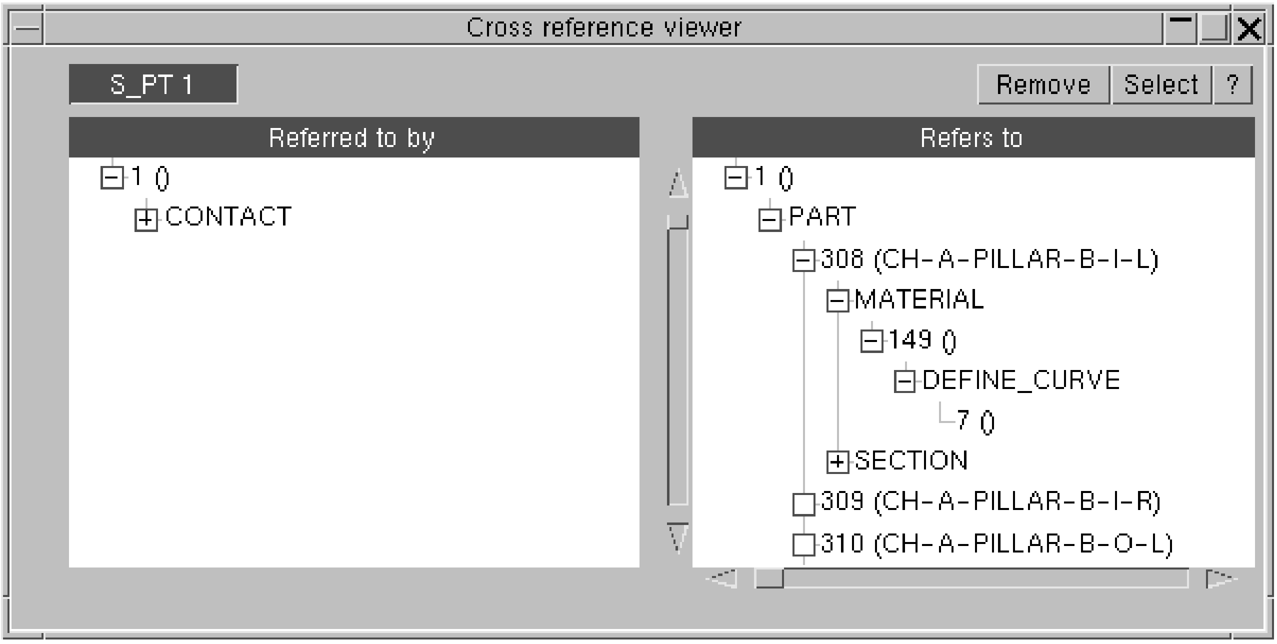
The cross reference viewer allows you to carry on looking for references, expanding branches as required. So, for example, we may want to look to see what PART 308 refers to. Does it have any references? At the moment we don't know. Primer does not calculate all the references in the tree at the beginning as this could take a long time for a large model. Instead it looks for cross references when each branch is clicked on. When Primer does not know if a branch has cross references a □ symbol is shown instead of a ⊞ symbol. So, in the example below we do not know if any of the parts have cross references yet.



To see if part 308 has references click on the □ symbol. In this example part 308 does have references (`MATERIAL` and `SECTION` references) so the branch is expanded. If the branch did not have any references the □ symbol would just disappear.



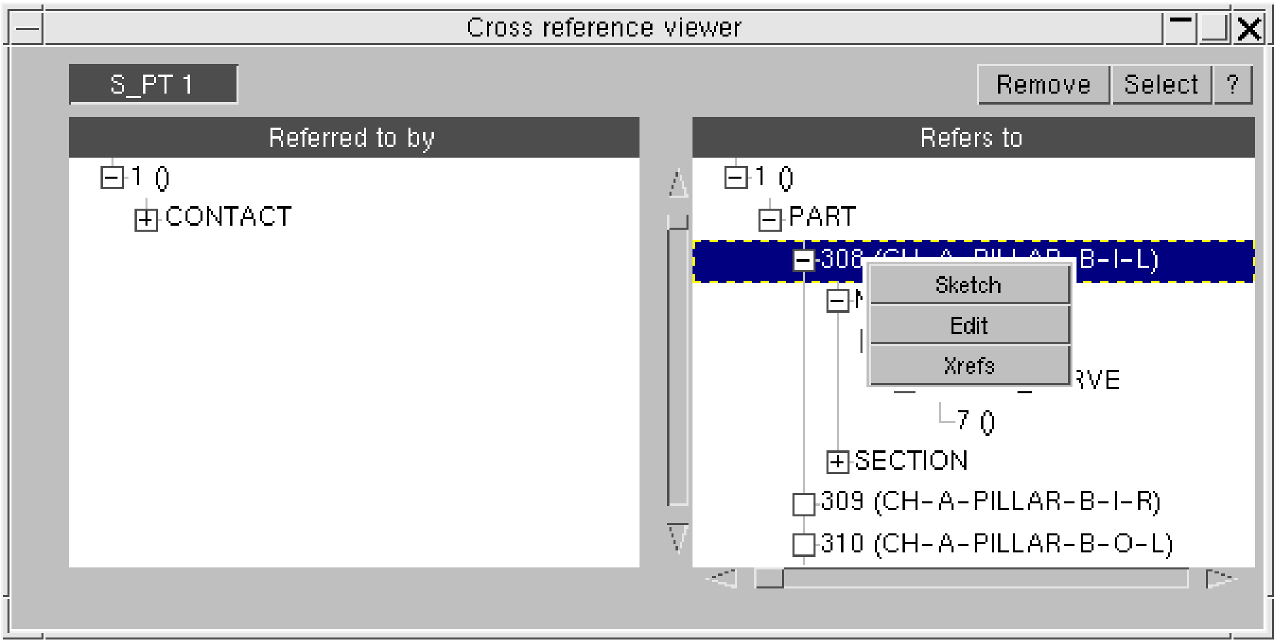
We can continue along the branches as required. In the example below `SET_PART 1` refers to `PART 308` which refers to `MATERIAL 149` which refers to `DEFINE_CURVE 7`. Curve 7 is the end of the branch as there is no □ symbol.



6.30.3 Sketching, editing and references for items

Right clicking on an item in either tree brings up a popup menu. If the item is sketchable a **Sketch** button is available, and if there is an edit panel for the item an **Edit** button is available that will allow you to edit the item.

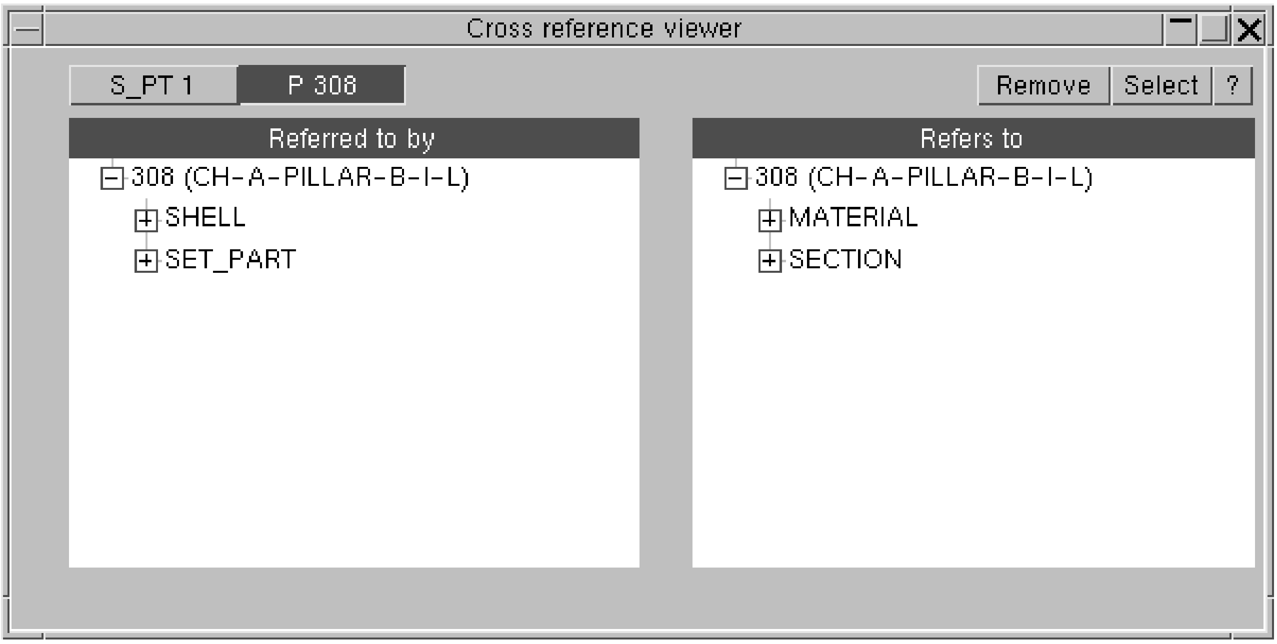
The **Xrefs** button in the popup creates another tab in the viewer for that item. For example, below selecting **Xrefs** for PART 308 will create another tab with PART 308 at the top of the tree.



6.30.3 Multiple tabs

The cross reference viewer enables you to have multiple tabs in the same window. Each tab can have a different item at the top of the tree.

For example, in the above images we have been looking at the references to SET PART 1 and we have found that it contains (**refers to**) PART 308. We now want to see what elements are in PART 308. i.e. we want to see which shells PART 308 is **referred to by**. This means we need to use the left hand (**referred to by**) tree. In the image above we right click on PART 308 and select **Xrefs**. This makes a new tab for PART 308 (image below) and we can now see the shells in the tree.



Remove will delete the current tab from the window. **Select** maps an object menu and allows you to select another item. Another tab will be created for the item.

7 Part Tree and Table

7.0 Part tree and table.

The part table gives a clear and intuitive method for viewing part properties and manipulating part or parts quickly and efficiently see section 7.1 for more details

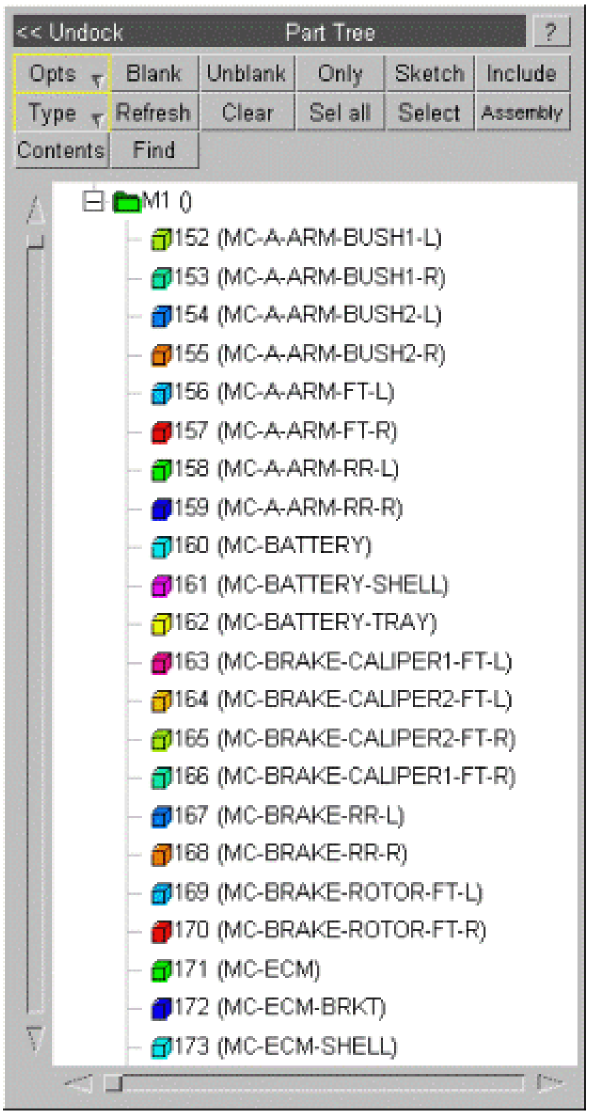
The part tree is a powerful visualisation and manipulation tool based on the organisation of parts within the model see section 7.2 for more details

Part compare is a function which drives the part table to find differences in properties between ostensibly matched parts, see section 7.3 for more details.

7.1 PART TREE

This enables quick navigation around a Model. It is possible to manipulate, view and edit entities in a quick and easy fashion. The part tree is focused around part manipulation, but it is also possible to view other types - e.g. materials, shells. There are various types of view expansion and contraction available, multiple selection and key viewing functions to hand. This makes the part tree a very powerful Primer tool and therefore it is constantly available through the tabbed area below the tools and keywords on the right hand side:

The part tree defaults to a view of the parts within the model, a typical display is shown here (with parts and materials displayed):

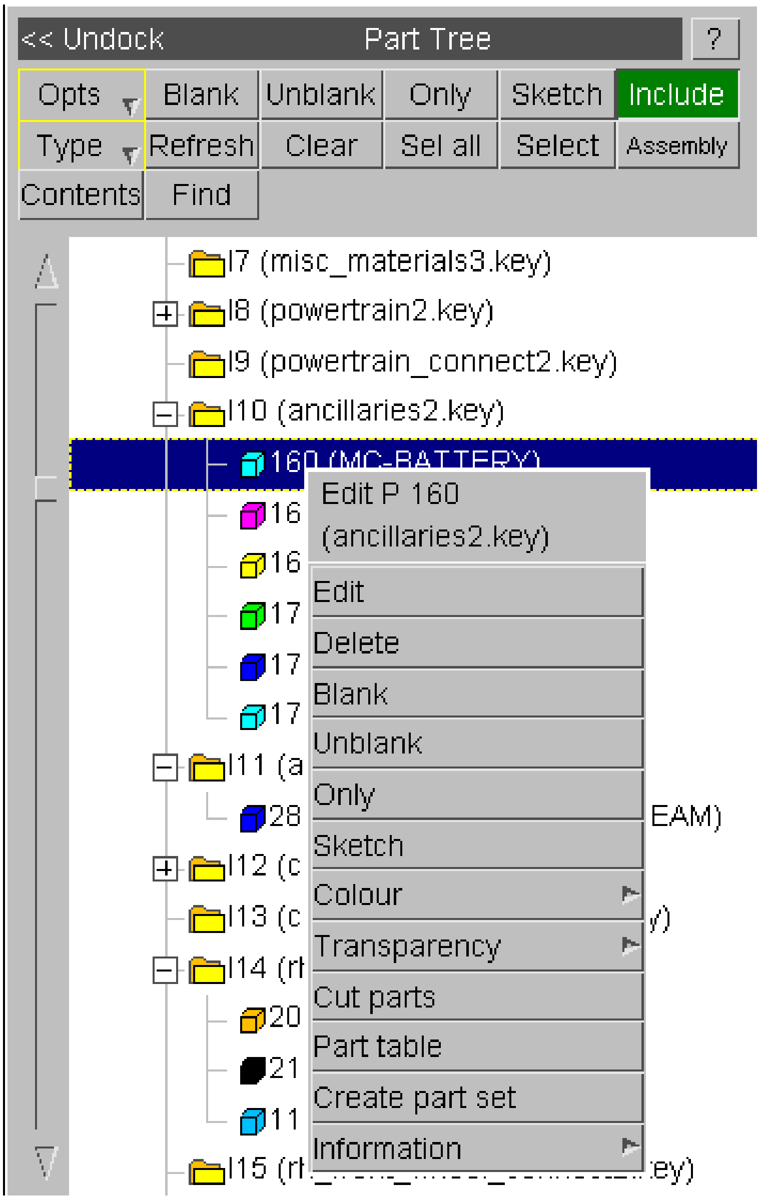


Part Tree Behaviour

Items can be selected by left-clicking anywhere on their row. Where selecting more than 1 item would be valid you can hold <ctrl> whilst clicking to select multiple items. Alternatively the <click> (start of range) .. <shift><click> (end of range) method (cf Windows) may be used.

Clicking on the [-] button next to models / include files / assemblies will collapse branches. Collapsed branches will have a [+] button which when clicked will expand the branch.

Right-clicking on an item or a selection of items produces a pop-up menu with the options shown on the right (not all of these options will be available for some selections).



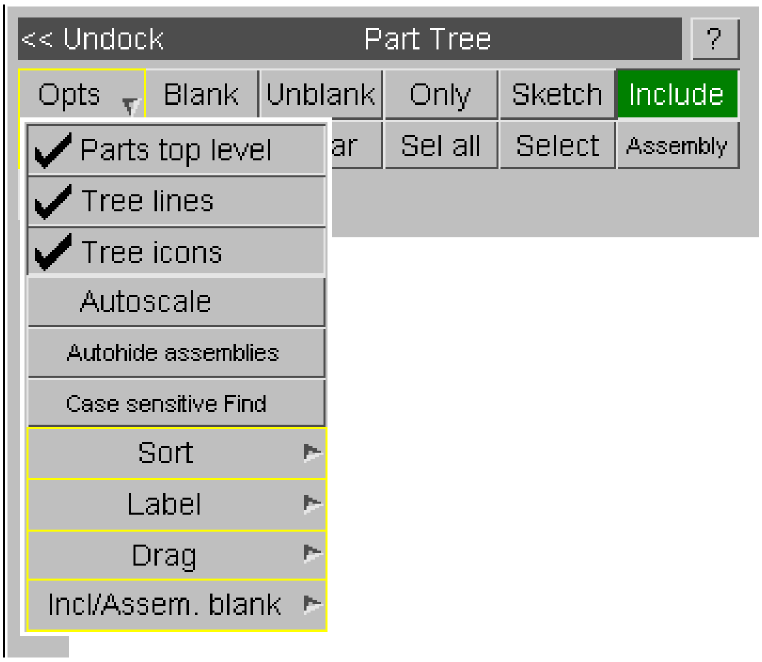
Edit	Brings up the standard editing panel for that item
Delete	Deletes the item
Blank	Blanks the item
Unblank	Unblanks the item
Only	Blanks all other items and unblanks the item
Sketch	Sketch the item
Colour	Colours the items (or elements associated with the item)as selected
Transparency	Sets the transparency the items (or elements associated with the item)as selected
Cut parts	Marks the part as those to be moved upon receiving a paste command
Part table	Brings up the Part table for the selected parts
Paste parts	Moves the last cut parts into the selected Include file or assembly

Create part set	Create a part set which the currently selected parts as the contents
Create part set and link	Available for assemblies. Creates a part set which the currently selected parts as the contents. If the assembly contents are updated, the part set is also updated
Information	Display information on the currently selected part
Make current layer	Makes the selected include file the current layer into which newly created entities will be put
Read Assembly file	Read an assembly file (applies to a model only)
Write Assembly file	Write out an assembly file (applies to a model only)
Create Assembly	Create an assembly in the the selected model or as a child of the selected assembly
Delete Assembly	Remove the assembly (but not the contents)
Rename Assembly	Rename the assembly
Assembly C of G	Give information on assembly C of G
Select parts to add	Opens an object menu to select parts to add to an assembly
Make current assembly	Makes the selected assembly the current layer into which newly created entities will be put
Clear current assembly	Clears the current assembly (newly created entities will not go into an assembly)
Add to Clipboard	Add assembly contents to the clipboard
Remove from Clipboard	Remove assembly contents from the clipboard
Replace Clipboard	Empty the clipboard contents and replace with assembly contents
Flatten all assemblies	Flatten all assemblies in the selected model (confirmation will be required)
Edit comments	Edit the header comments for a model or include file
Rename Include file	Changes an include file name

Part tree top menu bar

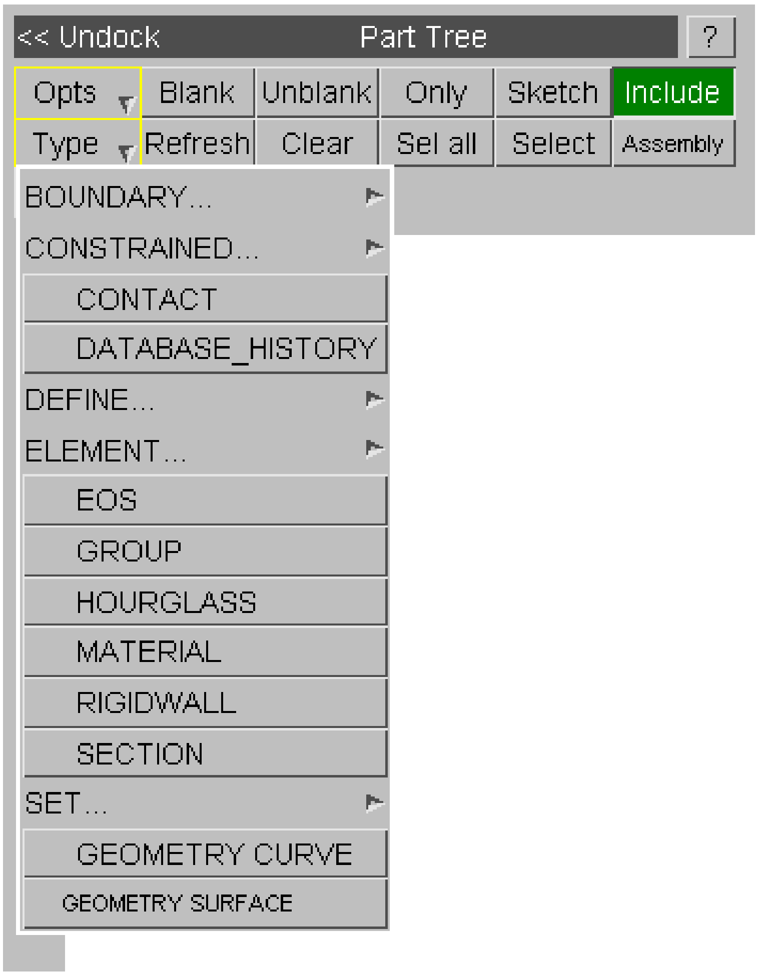
The top menu bar allows quick access top common Primer functions, as well as controlling how the part tree behaves and is displayed.

Opts



There is a range of options for controlling the part tree available via the **Opts** pop-up menu. These include how items are labeled and ordered as well as whether assemblies, tree lines and icons are drawn. The **Drag** option controls which items are moved when using the part tree to move items between include files/assemblies. For example when moving parts to a different include file you may wish to move the sections and materials as well. The **Incl/Assem blank** option allows you to control how items are treated when using blank/unblank/only on the part tree. For example, the elements within a part may be in a different include file to the part, but you may still wish to blank/unblank/only the elements.

Type



From the Type pop-up menu it possible to select a variety of different item types to be displayed in the tree in addition to parts. These appear below the parts in the tree, and most of the options (edit, blank etc.) are available through the "right-click" menu.

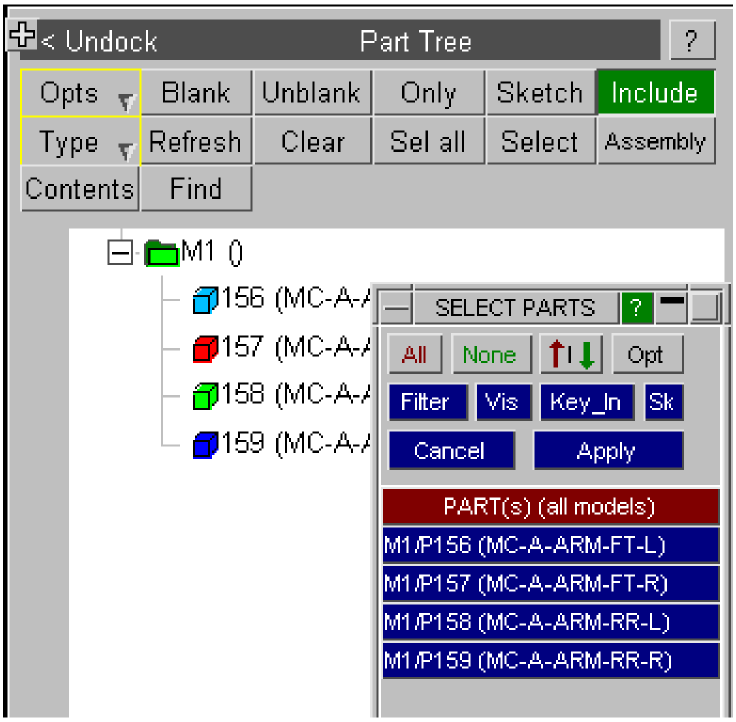
Blank / Unblank / Only / Sketch

One use of the part tree is as easy way access to blanking commands. **Blank**, **Unblank**, **Only** (blank all other items) and **Sketch** commands can be applied to the currently selected items.

Sel all / Clear

The **Sel all** and **Clear** buttons can be used to select all items and empty the selection respectively.

Select

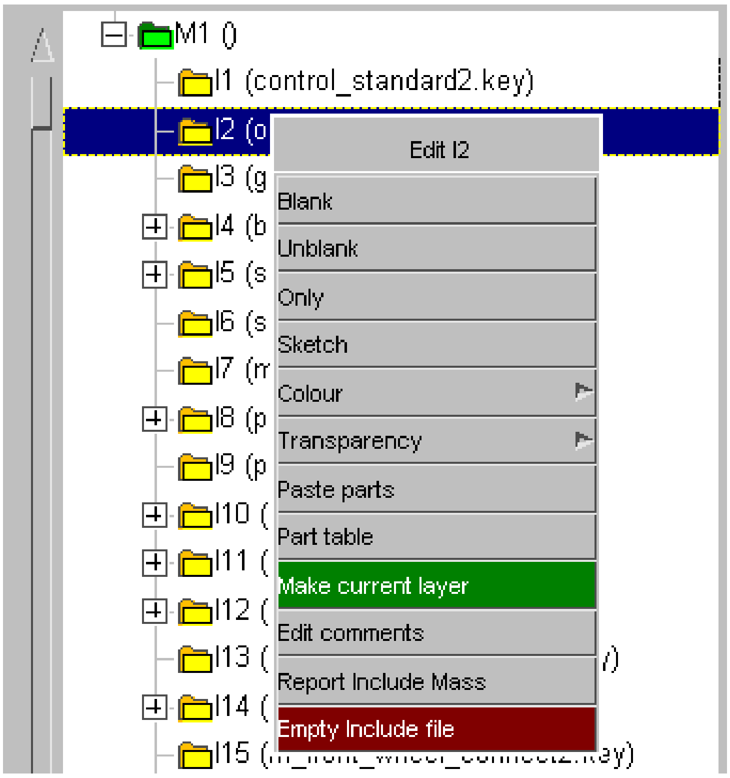


The **Select** button invokes an object menu for selecting parts. Selection can also be made via the Quick Pick option "Locate in Tree".Note that selections can be Include files, Assemblies or Models as well as parts. For example, click Only then an Include file to display only that Include file.

Include

The Include and Assembly buttons determine what type of hierarchy is displayed.

When in Include mode, the Include file structure of the model is shown. Parts can be dragged from one Include to another- this has the same effect as putting the parts on the Clipboard and moving to an include file with the "find referenced items" option (nodes and elements are moved in addition to the part cards). It is also possible to "Make current layer" which sets the current layer to the relevant include file (the layer is where Primer creates new entities). *INCLUDE_STAMPED_PART definitions are also shown in the part tree with **S.P.** shown before the filename and a grey coloured folder icon. The stamped part definitions cannot be dragged and no options are available when right clicking on them.

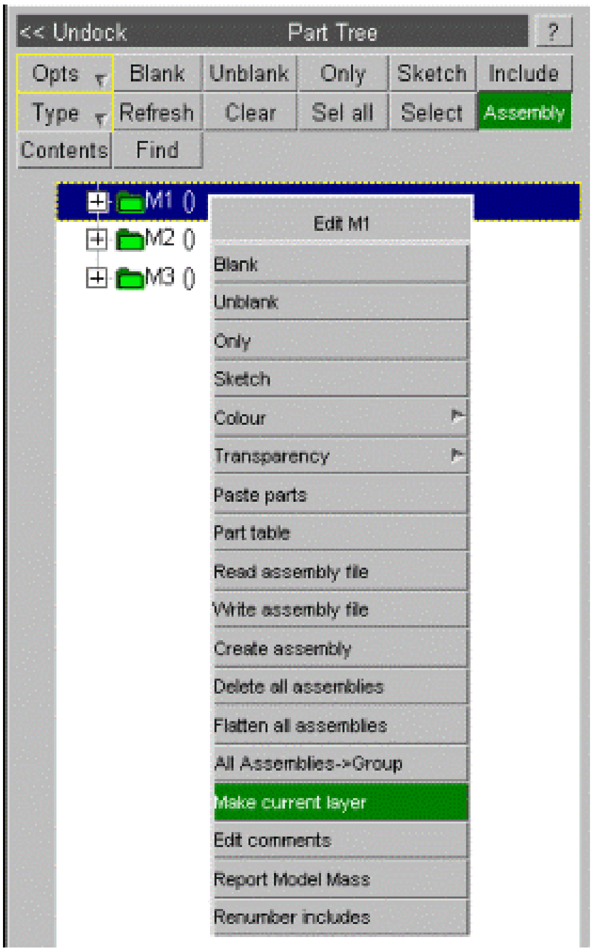


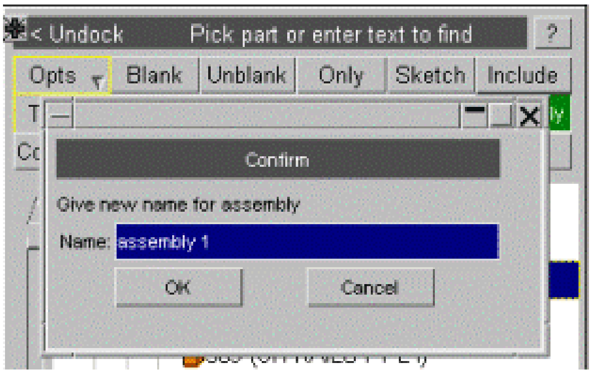
You can also cut and paste parts using the "right-click" menu. When a single or multiple selection of parts is shown, right click and cut parts. Then right-click over the include file and paste parts (shown above).

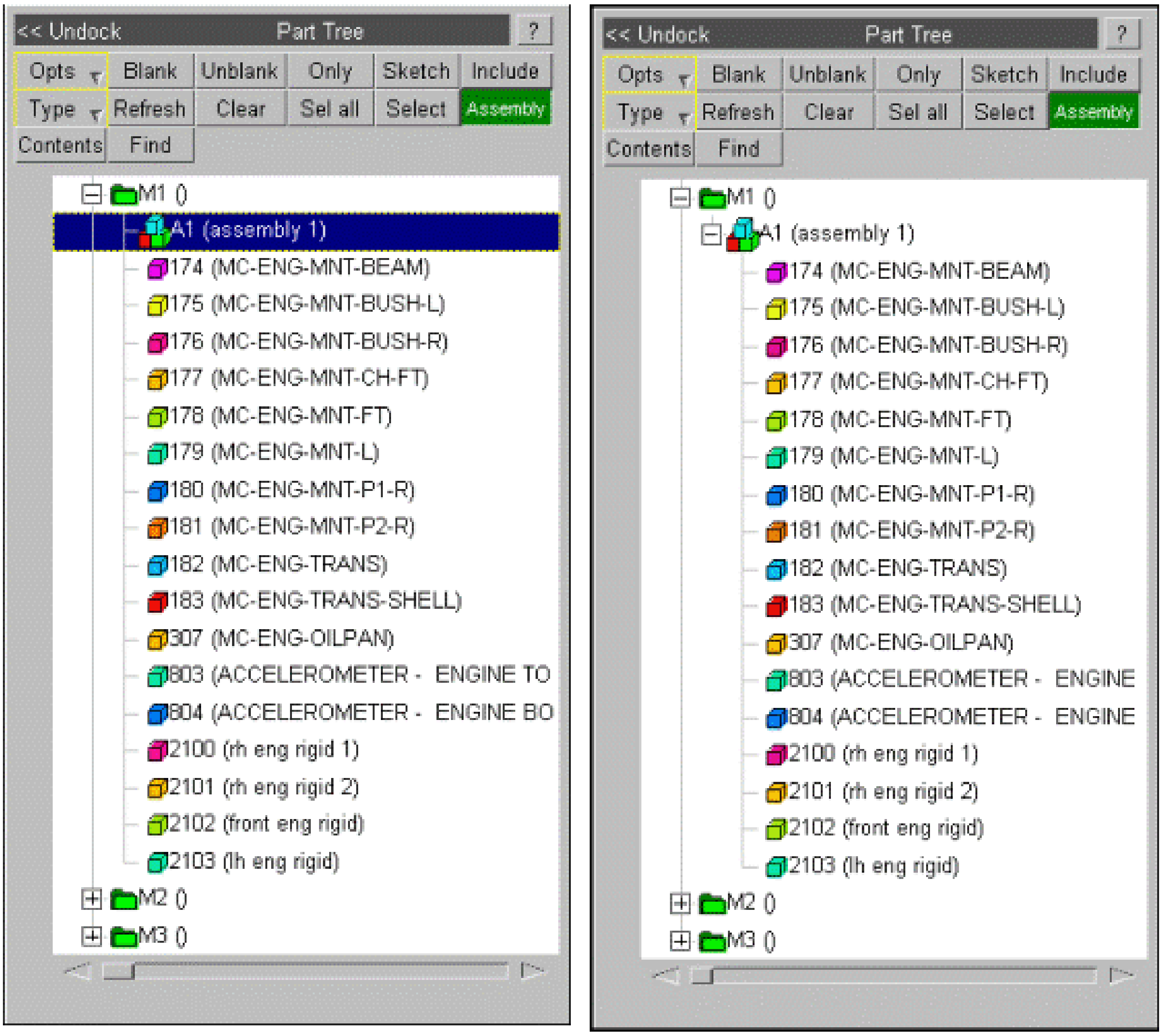
Assembly

Assemblies are user-defined hierarchical groupings of entities. They exist only in Primer and do not effect the output analysis file (they are written in comment lines). Assemblies provide a way of grouping entities together to enable quick model manipulation and viewing. Entities from different include files can be grouped together, and the hierarchy is stored in what is known as an **assembly file**. Note in earlier versions of Primer, only parts could be placed in assemblies. Now, any entity type can be placed in an assembly.

The assemblies are created by right-clicking on a model (or existing assembly) in the part tree. Parts can then be dragged into the assembly as shown below. The clipboard can also be used to move entities into an assembly.







When the keyword file is written out, the assembly to which each entity belongs will be written with the entity data as a comment. The hierarchical structure can be written only as a separate assembly file (right-click on the model in the part tree) - this is to avoid duplicated or missing assembly hierarchies when working with the model in Include files. If the keyword file is written but not the assembly file, when the model is next read in, the part tree will show a list of any assemblies that contain entities but flattened into a 1-layer-deep structure.

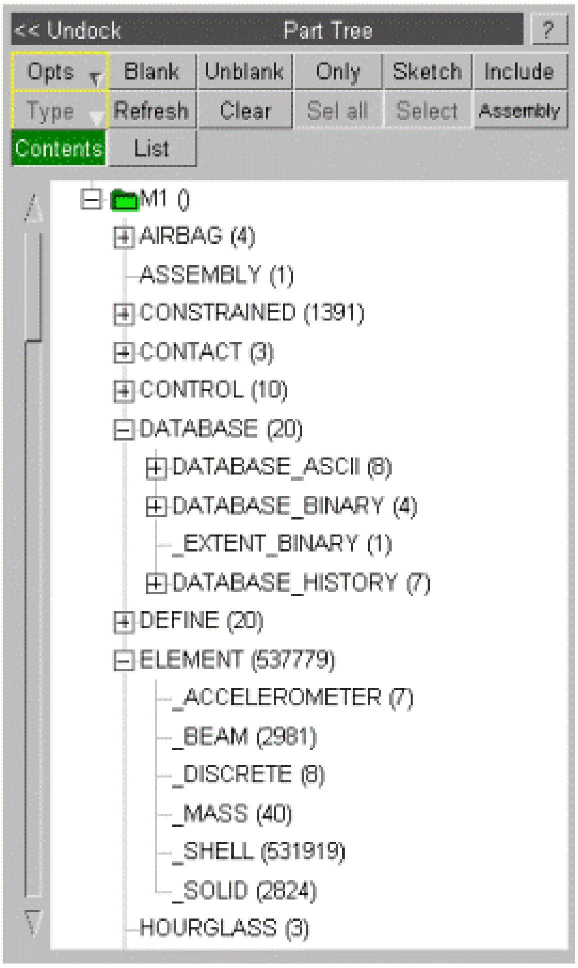
Note that assembly hierarchies are automatically created from Hypermesh and ANSA hierarchy comment data should this exist in the input deck. Upon keyout you can choose if you wish to save the assembly hierarchy data in Primer, Hypermesh or ANSA format.

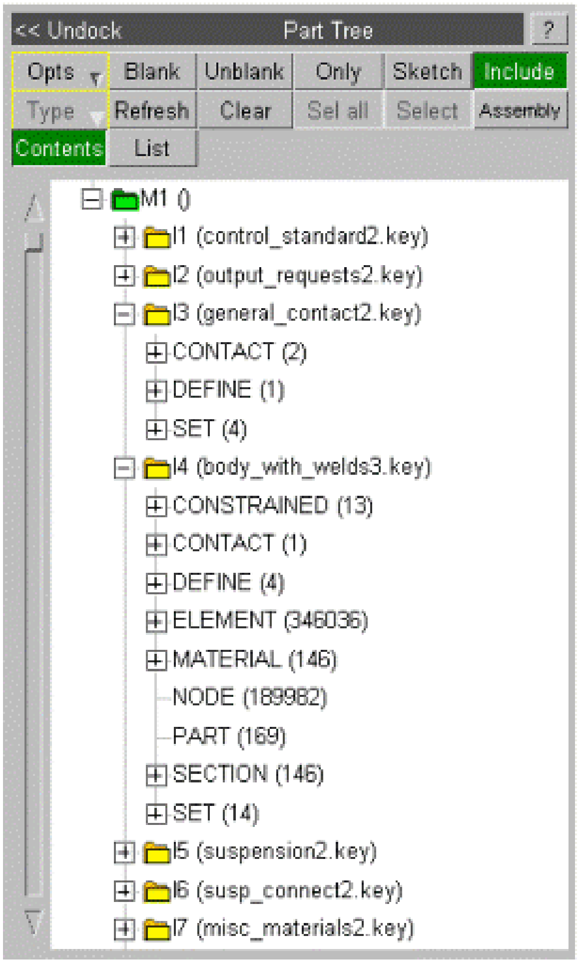
Find

The **Find** button is available only when **Contents** is switched off, and it gives a search option. Text or an ID number is entered in the text field. Primer finds a part whose title contains the text, or a part with an ID matching the number. The arrows determine whether the search direction is up or down from the current selection. **Next** will find the next matching item. The search will only find matches for currently enabled options (i.e. if id is disabled and items are labeled by name only a search for part 15 will return no matches regardless of its presence in the tree).

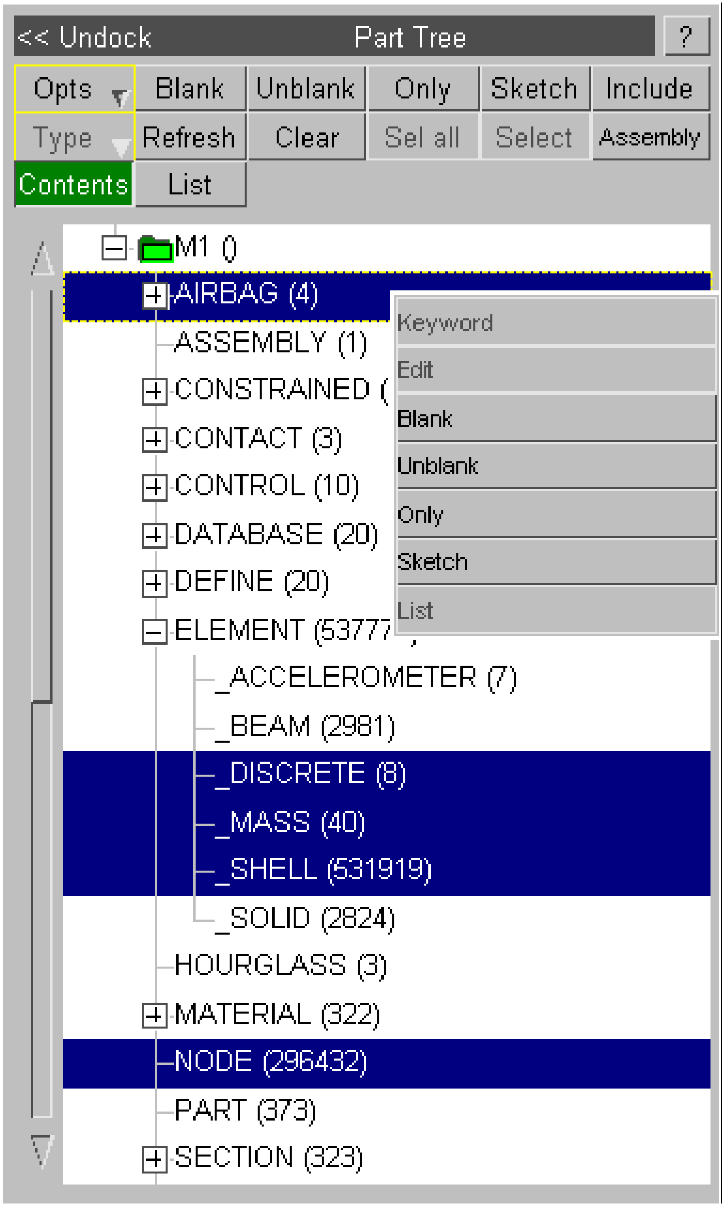
Contents

When **Contents** is switched on, the part tree displays each keyword in the model along with the number of entities of that particular keyword type arranged in alphabetical order of keywords. If **Include** is switched on in conjunction with **Contents**, the part tree displays the keywords and their numbers by include files. On the other hand, if **Include** is switched off, the keywords and their numbers are displayed for the entire model. The following illustrations depict the part tree in **Contents** mode with **Include** switched off and on respectively:





Branches can be selected from the tree and operations such as blanking, unblanking, etc., can be performed. Not all operations are permitted for all keywords, and the buttons for the non-permitted operations are greyed-out in the popup box as seen below:

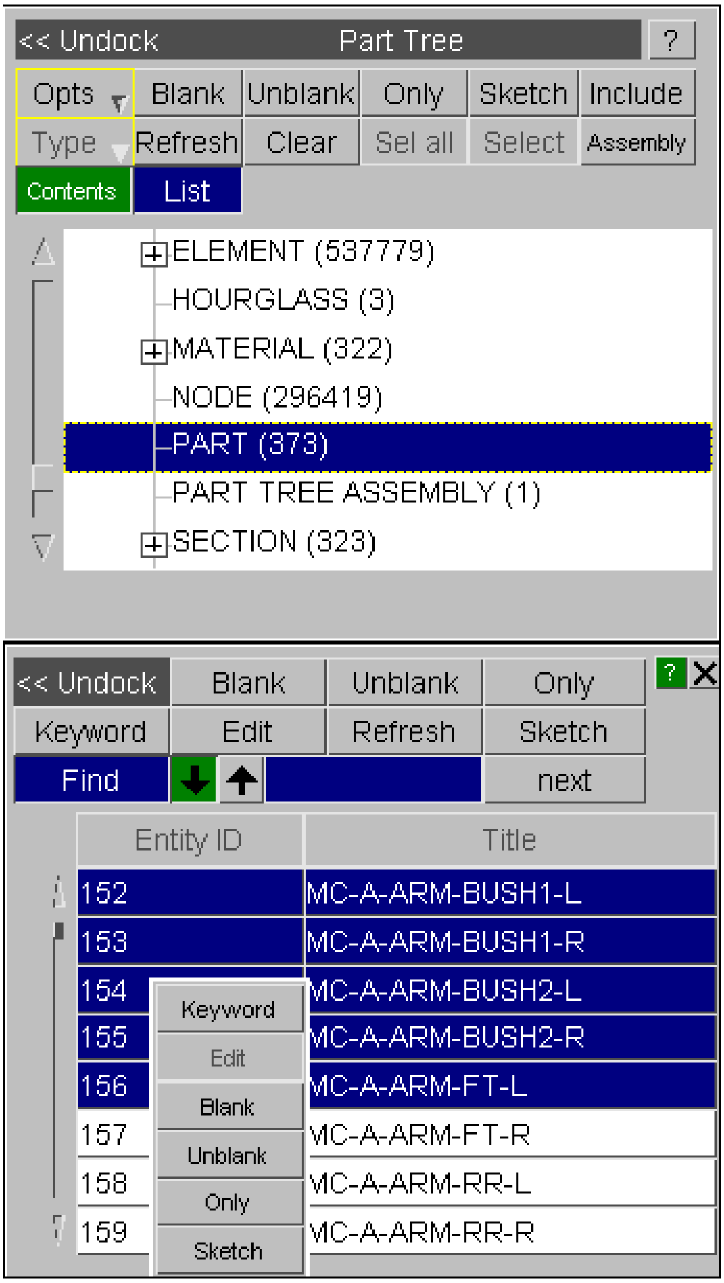


In addition to the options available at the top of the tree, the pop-up box offers the following new options when the part tree is in **Contents** mode:

- **Keyword**: This option is available if the entities that belong to a selected keyword can be manipulated via a generic keyword editor in Primer. When clicked, this option invokes the generic keyword editor containing all the entities that belong to the selected branch. This option is obviously not available when multiple keyword are selected in the part tree.
- **Edit**: If only one keyword is selected from the part tree, and if only one entity of the selected keyword type exists in the model, the editing panel for the selected entity can be directly invoked using this option.
- **Cut/Paste**: Allows you to move entities to different include files.
- **List**: This button invokes the **Contents List** window for the selected keyword. The Contents List window is described in the following section.

List

This button replaces the **Find** button when **Contents** is switched on, and is therefore available only when the part tree is in **Contents** mode. If multiple entities of a keyword type are permitted in a model, the entities of that type can be listed in a single **Contents List** window by means of the **List** button. Keywords such as *CONTROL_ and *DATABASE_ASCII_ are exceptions as only one entity of each of such keyword types are allowed in a model. The **Contents List** window can alternatively be invoked by means of the **List** option in the pop-up box for the keyword selected in the part tree. The **Contents List** window can be seen in the illustration below:



The entries in the Contents List window can be selected and subjected to operations such **Keyword**, **Edit**, **Blank**, **Unblank**, **Only** and **Sketch**. Note that not all operations are available for every keyword type, and that the buttons for the non-permitted operations are greyed-out in the popup box as seen above.

Just as in the part tree itself, option **Keyword** invokes the generic keyword editor for the selection (if permitted by Primer), and option **Edit** invokes the editing panel for one single entity selected in the **Contents List** window.

The **Contents List** window also has a **Find** button that looks and functions in a similar fashion to the one in the main **Part Tree**.

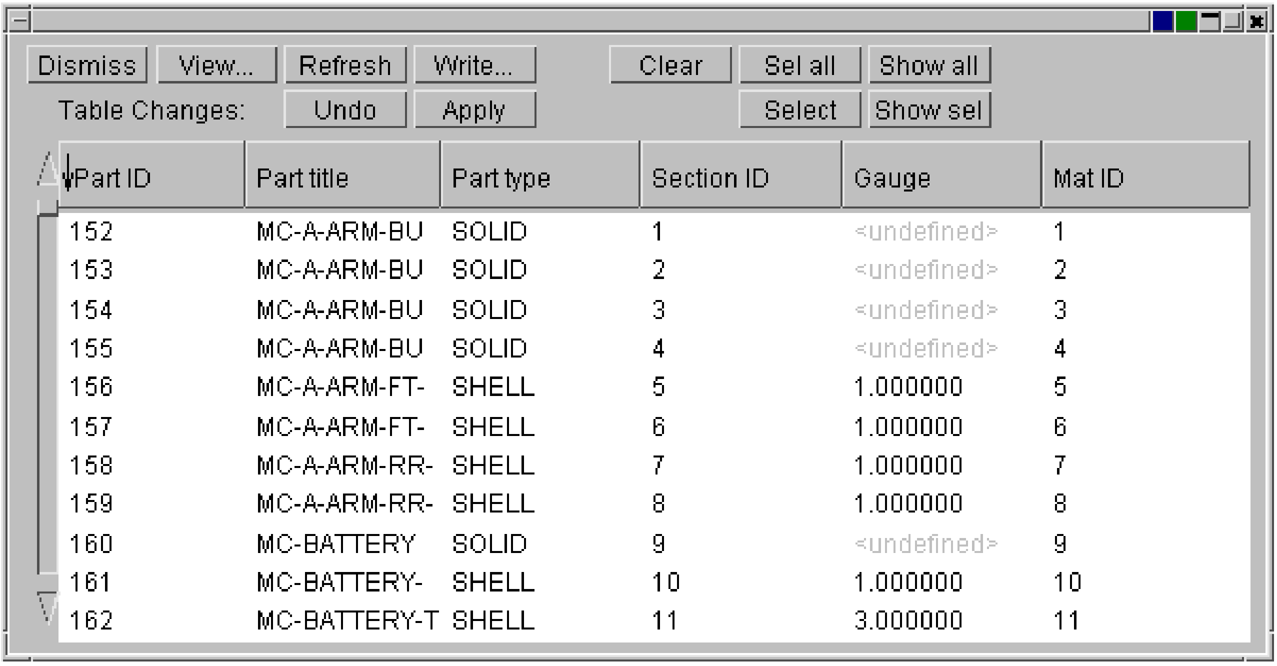
7.2 PART TABLE

The part table allows you to easily view and change information for parts in your model. The table can be started by either:

- Selecting parts/include files/assemblies/models in the **part tree**, right clicking and selecting **Part table** from the popup menu.
- Using the **Table** option in The **Part** menu.
- Using quick picking for PART with **Part Table** mode selected.

Each row in the table represents one part. By default the rows are sorted by ascending Part ID. The column used for sorting can be changed by clicking on the appropriate column header. Additionally clicking on a column header again will sort by descending order instead of ascending order. The column which is currently used for sorting has an arrow drawn on the header (in the example below it is Part ID).

If a column is invalid for a part **<undefined>** is shown. For example, below, **Gauge** is not valid for solid parts.



Part ID	Part title	Part type	Section ID	Gauge	Mat ID
152	MC-A-ARM-BU	SOLID	1	<undefined>	1
153	MC-A-ARM-BU	SOLID	2	<undefined>	2
154	MC-A-ARM-BU	SOLID	3	<undefined>	3
155	MC-A-ARM-BU	SOLID	4	<undefined>	4
156	MC-A-ARM-FT-	SHELL	5	1.000000	5
157	MC-A-ARM-FT-	SHELL	6	1.000000	6
158	MC-A-ARM-RR-	SHELL	7	1.000000	7
159	MC-A-ARM-RR-	SHELL	8	1.000000	8
160	MC-BATTERY	SOLID	9	<undefined>	9
161	MC-BATTERY-	SHELL	10	1.000000	10
162	MC-BATTERY-T	SHELL	11	3.000000	11

The part table will resize as required as the window size is changed.

Changing which columns are shown

There are many different fields that can be shown. To add or remove a column press **View...** which will bring up the list of field types as shown below. The fields which are currently shown will have a tick symbol next to them

Model	HG Coeff	NS Mass	Blanking	FD	Save Settings
✔ Part ID	Mat ID	✔ Dyna Part Mass	Colour	DC	Dismiss
Part title	Mat title	Component Mass	Transparency	VC	
Part type	✔ Mat type	✔ C of G	Style	OPTT	
Section ID	Density	Inertia (XX YY ZZ)	Include	SFT	
Section title	Modulus	Inertia (XY XZ YZ)	Numel	SSF	
Gauge	Yield	✔ Lumped Mass (def)	Smallest TS	Merge status	
NIP	Fail strain	✔ NRB Mass	Smallest elem	CON1	
Elform	EOS ID	✔ Transferred Mass	Part Inertia	CON2	
HG ID	Struct Mass	Dyna Added Mass	Part contact	Stamped part	
HG Type	Assign Mass	%Added Mass	FS	Encrypted material	

Changing the table columns

The following columns are available for display in the part table under **View...**

Column	Explanation
Model	Model label
Part ID	Part label
Part title	Part title
Part type	Type of part (shell, solid etc.)
Section ID	Section ID part uses
Section title	Title of section part uses
Gauge	Gauge or thickness of part
NIP	Number of integration points
Elform	Element formulation
HG ID	Hourglass ID part uses
HG Type	Hourglass type
HG Coeff	Hourglass coefficient
Mat ID	Material ID part uses
Mat title	Title of material part uses
Mat type	Type of material part uses
Density	Density of material part uses
Modulus	Young's Modulus of material part uses
Yield	Yield stress of material part uses
Fail strain	Failure strain of material part uses
EOS ID	EOS ID part uses
Struct Mass	Structural mass (rho x vol)
Assign Mass	Mass added through lumped mass belonging assign mass
NS Mass	Non-structural mass (section card or *Element_mass_part)
Dyna Part Mass	Part mass (see below)
Component mass	Part mass (see below)
Lumped Mass	Mass applied through lumped masses on nodes of deformable part (including assign mass)
NRB Mass	Mass of nodes of deformable part that is attributed to NRB
Transferred Mass	Mass lost/gained where node is shared by both deformable & rigid element
Added Mass	Timestep Added mass (true value applicable for solid spotwelds see appendix 17)
Added Mass %	Percentage added mass
C of G	Centre of gravity
Inertia (XX YY ZZ)	Part inertia tensor
Inertia (XY XZ YZ)	Part inertia tensor
Blanking	Displays whether part is blanked or not
Colour	Colour of part
Transparency	Transparency status of part
Style	Current style of the part
Include	Include file the part resides in
Numel	Number of elements contained within the part
Smallest TimeStep	id & timestep of element with smallest timestep in this part

Smallest elem	id & characteristic length of element with smallest characteristic length
Part Inertia	Is _INERTIA applied to the part?
Part Contact	Is _CONTACTapplied to the part?
FS	_CONTACT field. Static coefficient of friction
FD	_CONTACT field. Dynamic coefficient of friction
DC	_CONTACT field. Exponential decay coefficient
VC	_CONTACT field. Coefficient of viscous friction
OPTT	_CONTACT field. Optional contact thickness
SFT	_CONTACT field. Option thickness scale factor.
SSF	_CONTACT field. Contact stiffness scale factor
Merge status	Rigid body merge status (master or slave)
CON1	CON1 field from any applicable rigid material card
CON2	CON2 field from any applicable rigid material card
Stamped part	Displays whether the part is in an INCLUDE_STAMPED_PART definition or not
Encrypted material	Displays YES if the part references an encrypted material card, otherwise displays NO

Changing the default table columns

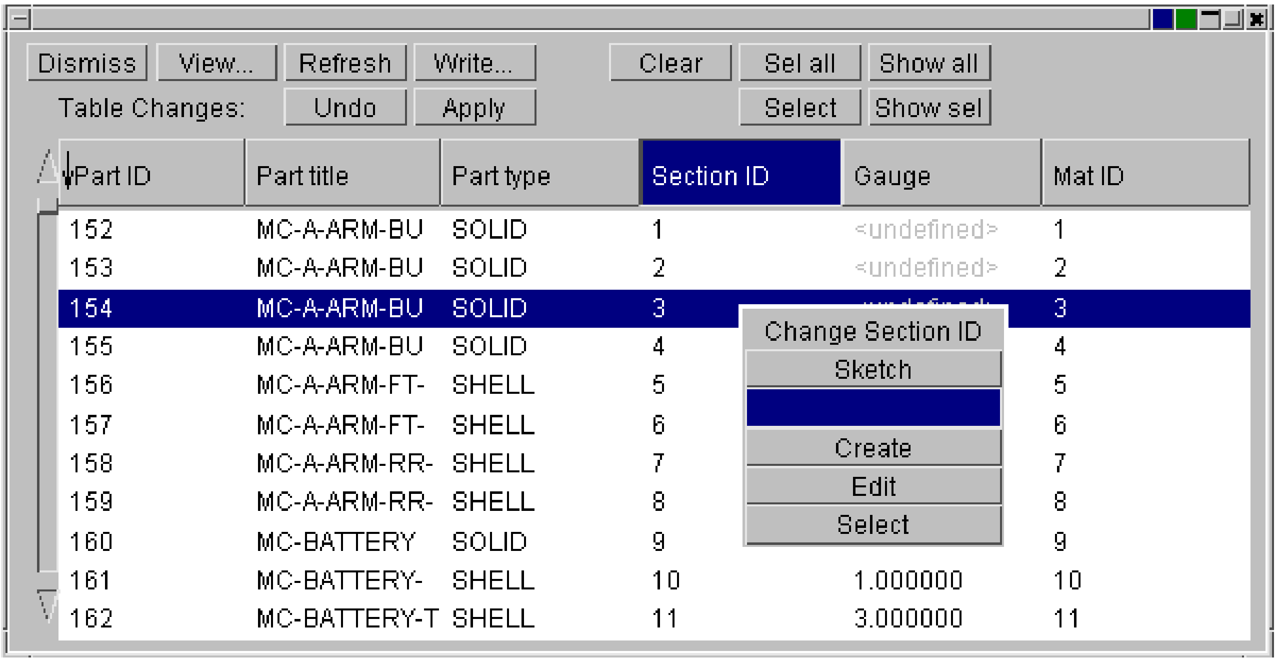
By default the part table will show the Part ID, Part title, Part type, Section ID, Gauge and Mat ID columns. If you want to change which columns are shown by default then change the columns shown to be the ones you want and press **Save Settings** in the **View...** popup. This will automatically add a preference `primer*part_table_columns` to your home oa_pref file with the appropriate columns.

Selecting rows in the table

Rows can be selected in the table by clicking with the mouse. To select multiple rows use the **`Ctrl`** key while clicking and to select a range of rows use the **`Shift`** key while clicking. There are also buttons at the top of the table to aid you in selecting and viewing parts in the table. The **Clear** button clears all current selections. The **Sel all** button selects all parts currently displayed on the table. **Select** will bring up an object menu and allow you to select parts using that method (i.e. being able to use various filters to select parts). **Show sel** will display in the table only those parts currently selected. **Show all** will bring back and display the original parts on the table.

Changing a value in the part table

To change the value for a field, select the parts that you want to change and then right click on the column that you want to change. This shows a popup menu allowing you to change the value. e.g. below Section ID is being changed for part 154.

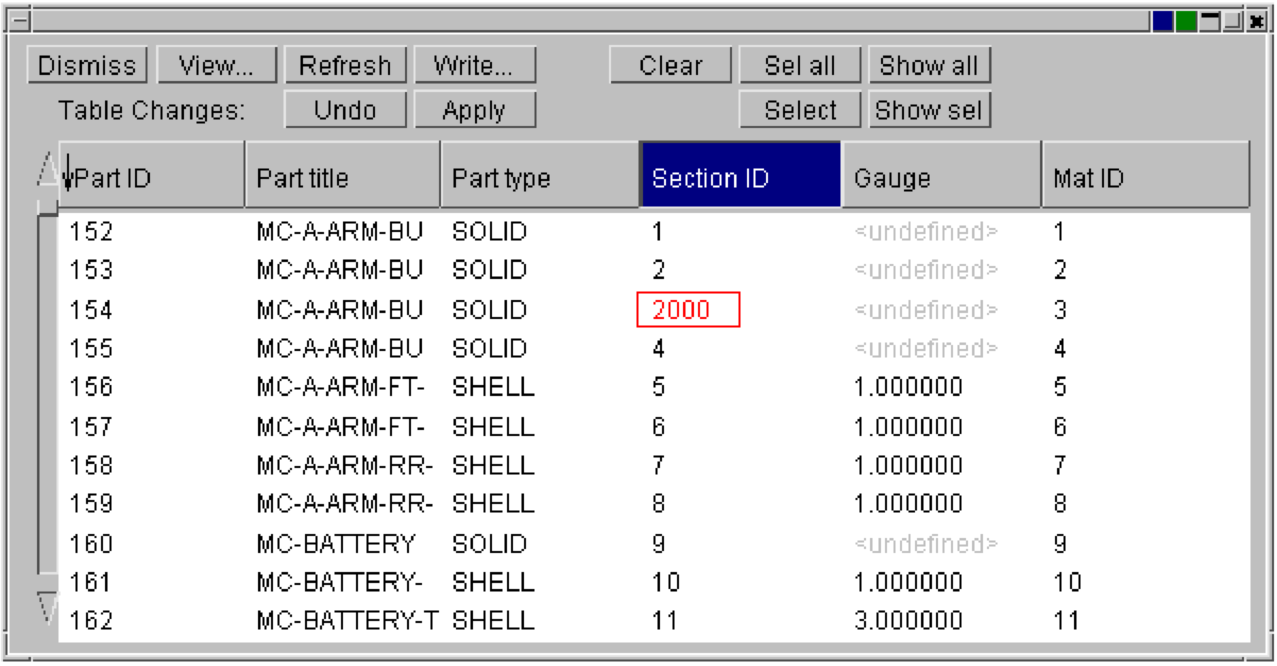


To change the value, type in the new value in the text box. Alternatively, if the field refers to an item (e.g. section ID, material ID) you can use **Select** to choose the item or **Create**/**Edit** to make a new item or edit the item respectively. Note for materials in the table there is the option to **Select from table**. This restricts the selection to only materials currently referenced by parts on the table.

Sketch will draw the currently selected parts.

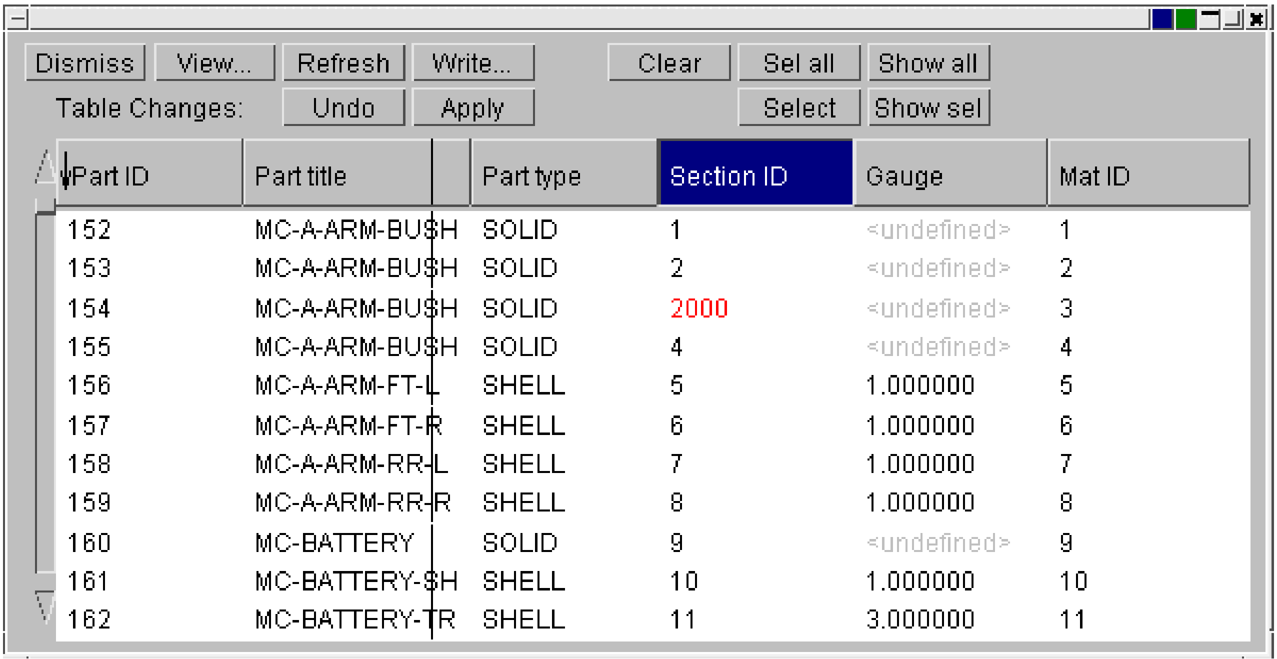
If the table is used to change Part colour, transparency, blanking and drawing mode the new settings are applied immediately. For other values (e.g. title, gauge etc.) the values will not be updated for the parts until the **Apply** button is pressed. To indicate that a value has changed but not applied it will be shown in red. e.g. below, the section ID for part 154 has been changed to 1000 and so is shown in red. Changes can be undone by pressing **Undo**.

If the same section card applied to multiple parts and the user changes a value (such as gauge) on one part (or some but not all of the parts) Primer will create a copy of the original section card and modify that.



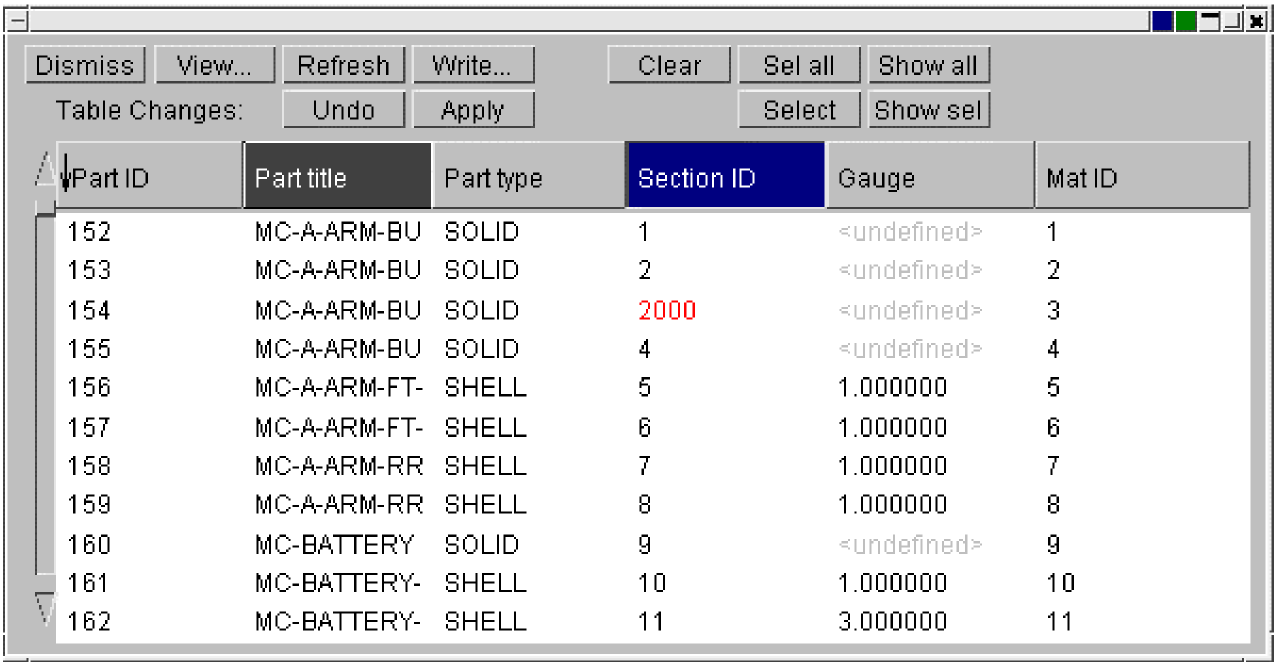
Changing the width of a column

Columns can be made wider or narrower by clicking on the edge of a column header and dragging the mouse. The cursor symbol will change to a double ended arrow while you are dragging. E.g. below the use has clicked on the header between **Part title** and **Part type** and is dragging to the left.



Changing the order of columns

The order of the columns can be changed by clicking on a column header and dragging the column left or right to a new location. When you are at a valid location the cursor will be a **+** symbol and when you are at an invalid location the cursor will be a **X** symbol. E.g. below the user has clicked on Part title and has started dragging the column. If (s)he wanted to move the column to be between the Gauge and Mat ID columns (s)he would drag the header until the cursor symbol changed to **+** between the 2 columns.



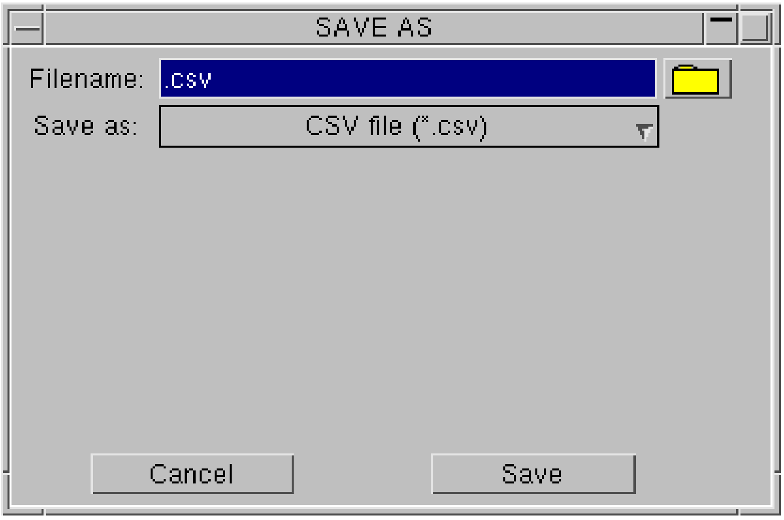
Saving part table information to file

The information in the part table can be saved to a file. Currently there are 3 different formats:

- A csv file suitable for importing into a spreadsheet such as Excel.
- A HTML file suitable for a web page, viewable by any web browser.
- A postscript file suitable for printing to a postscript printer or viewing with a postscript previewer such as ghostscript.

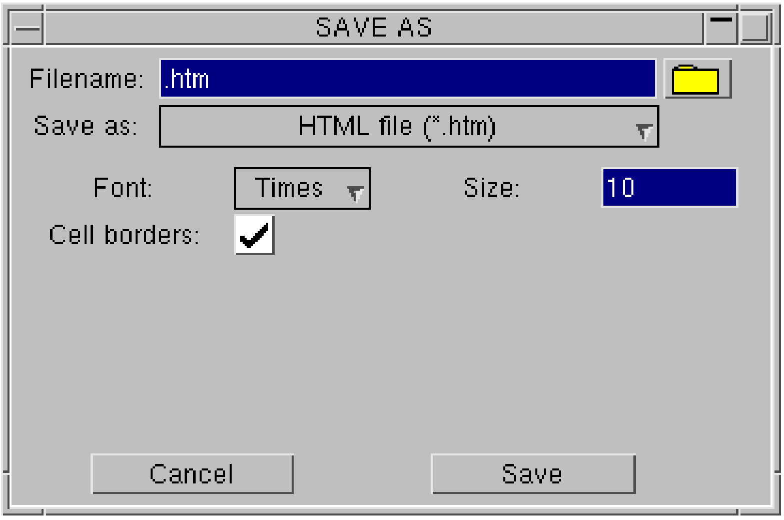
Choose the appropriate format using the **Save as:** popup.

CSV file



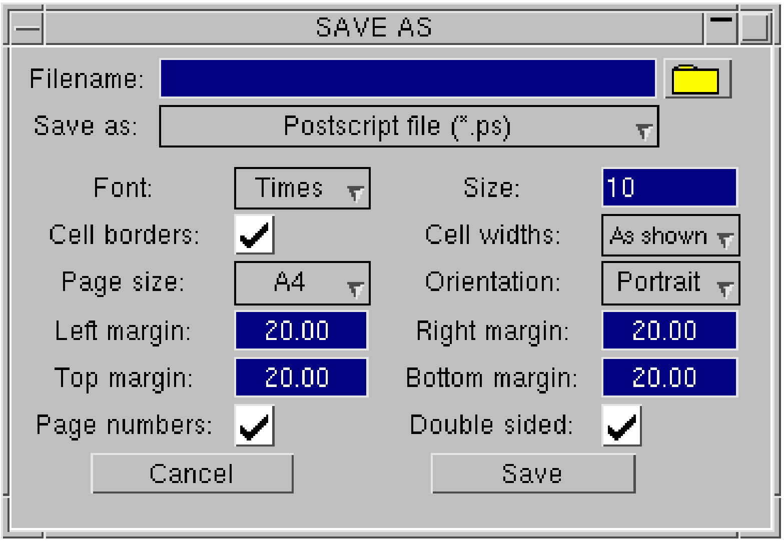
There are no options for CSV files. Give the name of the file to save.

HTML file



For html files you can choose the font, font size and if table cell borders are shown.

Postscript file



For postscript files you can choose the font, font size, table cell visibility, page size and orientation, margins, page numbering and double sided printing. Additionally you can choose to make all columns have equal widths or to use the column widths as displayed in the part table.

Mass in part table

The Part table allows display of different kinds of part mass, namely Structural mass, assign mass, nonstructural mass, dyna part mass, component mass, lumped mass, added mass and percentage added mass.

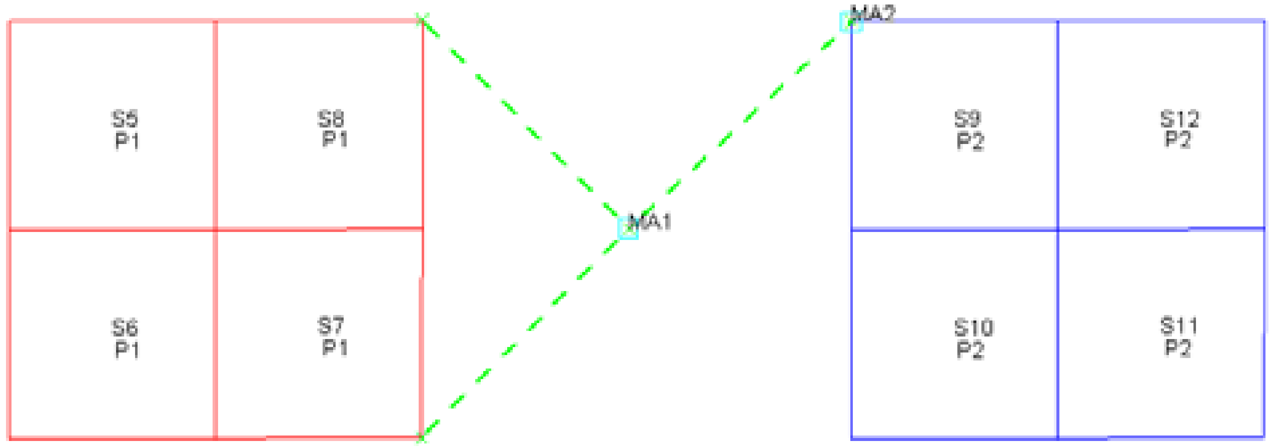
NS mass (non-structural mass) is the mass that applies on shell or beam parts as a result of mass per unit area setting (MAREA) on the section card. It may also be applied using the *ELEMENT_MASS_PART card.

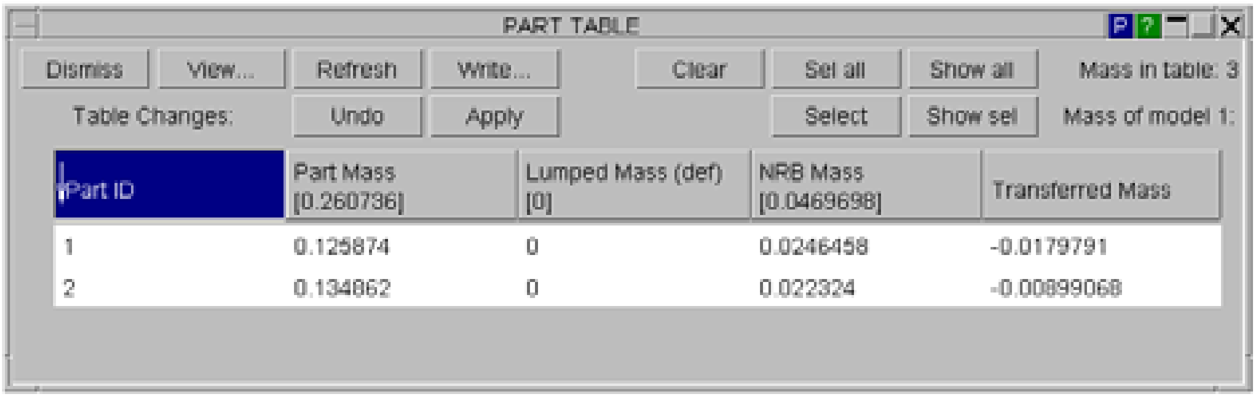
Lumped mass (def) is the sum of lumped masses attached to the nodes of the part, including assigned mass. For rigid parts the lumped mass (including masses on constrained extra nodes) is included in the Dyna Part mass, so it is not included in the column total (though the sum per part is listed for information).

Dyna part mass tries to use the same mathematical formulation as LS-Dyna. It is the sum of structural & non-structural mass belonging to nodes of part including lumped mass for rigid part (unless it is Part_Inertia). A deformable part ’loses’ mass where nodes attach to rigid part/nrb. A rigid part ’gains’ mass where it attaches to deformable nodes. Slaves in rigid body merges get zero mass, master parts acquire the mass of the slave(s).

Component mass is an attempt to describe the "engineering" mass of a part. This is the sum of structural & non-structural mass belonging to nodes of part including lumped mass for both deformable & rigid parts (unless Part_Inertia). Mass is NOT transferred from deformable to rigid parts/nrbs. Also in this context rigid body merges are ignored. The total of this column should be the model mass (without added mass).

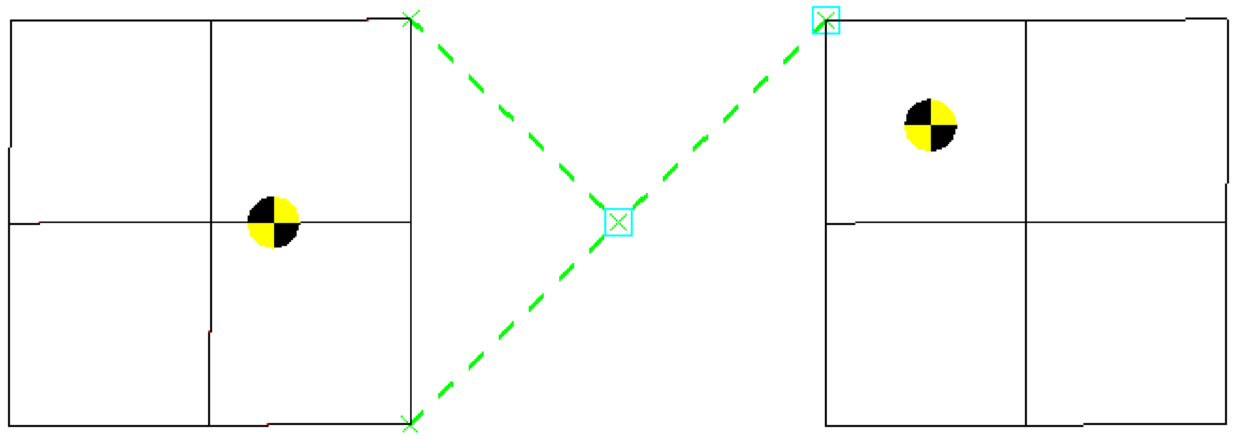
Note on masses on nodes of nodal rigid bodies. Masses on nodes of nodal rigid bodies attached to a part will be included in the NRB mass column for the part. If the mass is on a node which does not directly attach to a part its mass will be shared amongst the nodes which do attach.


S5
P1
S8
P1
S6
P1
S7
P1
MA1
MA2
S9
P2
S12
P2
S10
P2
S11
P2

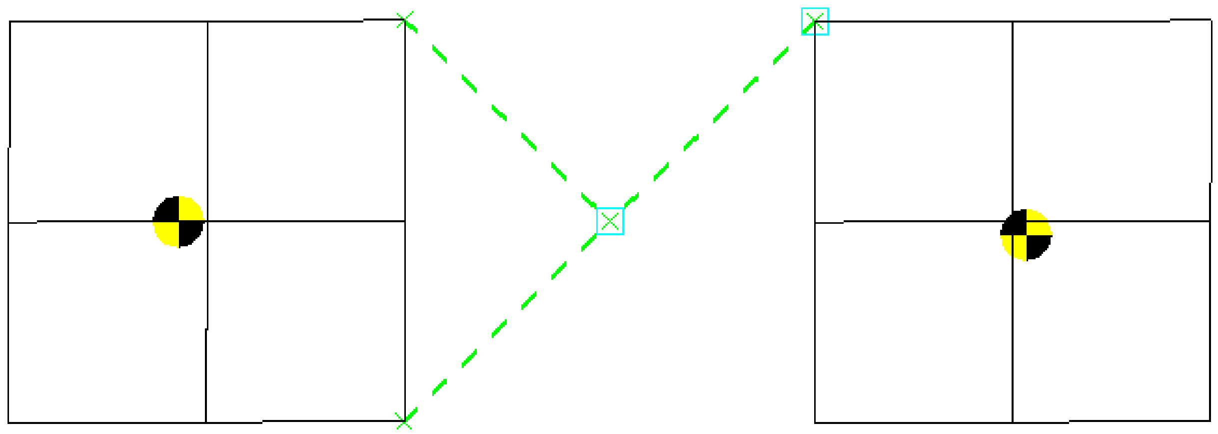

PART TABLE
Dismiss
View...
Refresh
Write...
Clear
Sel all
Show all
Mass in table: 3
Table Changes:
Undo
Apply
Select
Show sel
Mass of model 1:
Part ID
Part Mass [0.260736]
Lumped Mass (def) [0]
NRB Mass [0.0469698]
Transferred Mass
1
0.125874
0
0.0246458
-0.0179791
2
0.134862
0
0.022324
-0.00899068

In this example NRBmass for Part 1 will include mass 2/3 of mass of MA1 and 2 quarter element shares. Mass for Part 2 will include 1/3 of mass of MA1 and all of mass MA2 and 1 quarter element share.

If mass properties (CofG and Inertia) are activated in the table as above, they will include Lumped Mass and NRB mass.



If these columns are not displayed, the calculation will ignore the mass associated with the NRB and should give the same result as reported in the LS-Dyna otf (d3hsp) file. This treatment de-couples the NRB from the deformable parts.



Added mass is the timestep added mass on deformable parts that arises due the model mass scaling (DT2MS < 0.0).The percentage added mass is the ratio of added mass to part mass.

Structural mass is the sum of the structural element masses, except for rigid Part_Inertia, where is the raw value <TM>.

Part mass is defined by Primer as follows:

For a deformable part it is the sum of the structural mass, the assigned mass and the nonstructural mass.

For a rigid part that is not _INERTIA it is the sum of structural mass, the assigned mass, the nonstructural mass and the attached lumped mass. This will include mass on constrained extra nodes. If it is a master part, the mass of its slaves will be added in. If it is a slave itself its part mass will be reported as zero.

For a rigid part_inertia it is the true LS-DYNA part inertia value, that includes adjustment for rigid body merges and constrained extra nodes which carry the inertia flag (IFLAG). If the part is slaved itself its part mass will be reported as zero.

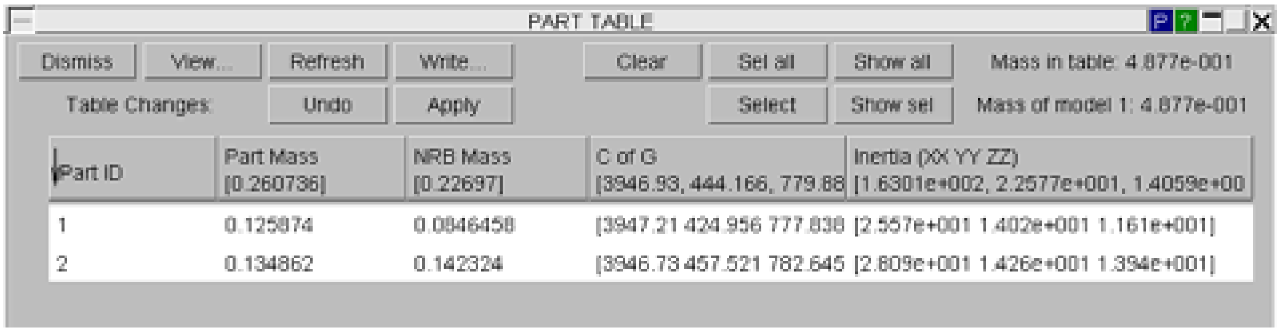
C of G and Inertia in part table

The C of G and Inertia tensor of individual parts may be displayed by using the drop down from the appropriate row.

If multiple parts are selected, the combined C of G will be displayed. These values are echoed in the dialogue box.

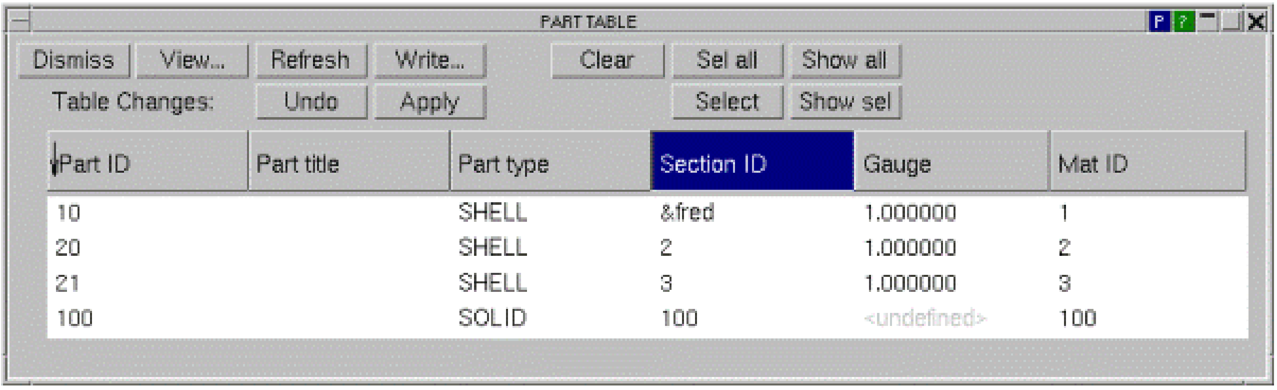
The value given on the top row is the combined C of G and combined Inertia for the parts displayed on the table.

If NRB mass/Lumped mass/Added mass columns are displayed, these masses will be included in the mass property calculations.

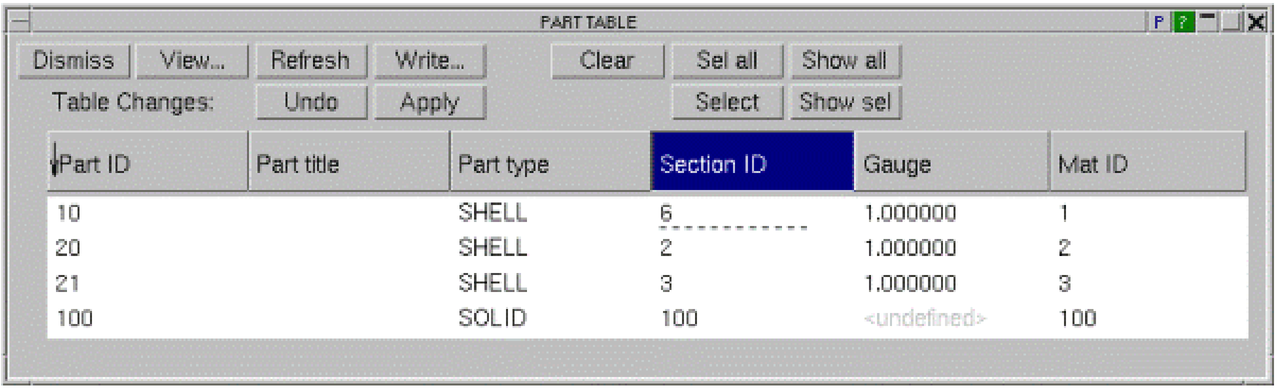


Parameters in the part table

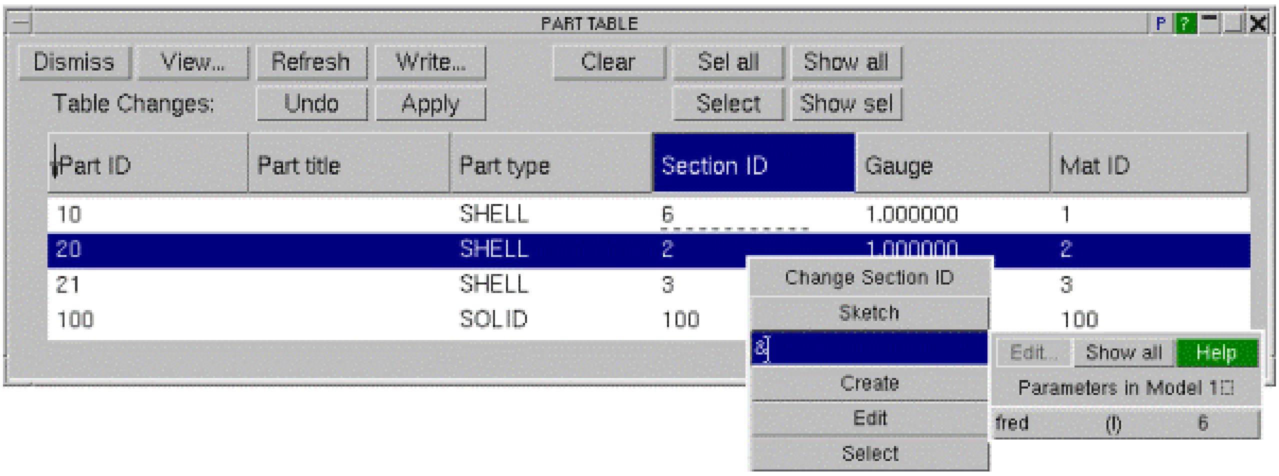
Parameters can be displayed in the part table. This works in a very similar way to how parameters are displayed in keyword edit panels. If a field contains a parameter, the parameter name is displayed with an "&" at the front. The example below shows a part table containing a part where the section ID is defined using a parameter, "fred".



The parameter value can be viewed by clicking on the "P" button in the top right of the part table. This button can be used to toggle between displaying parameters or their value. If the parameter value is displayed, then the value will be underlined to indicate it as a parameter value.

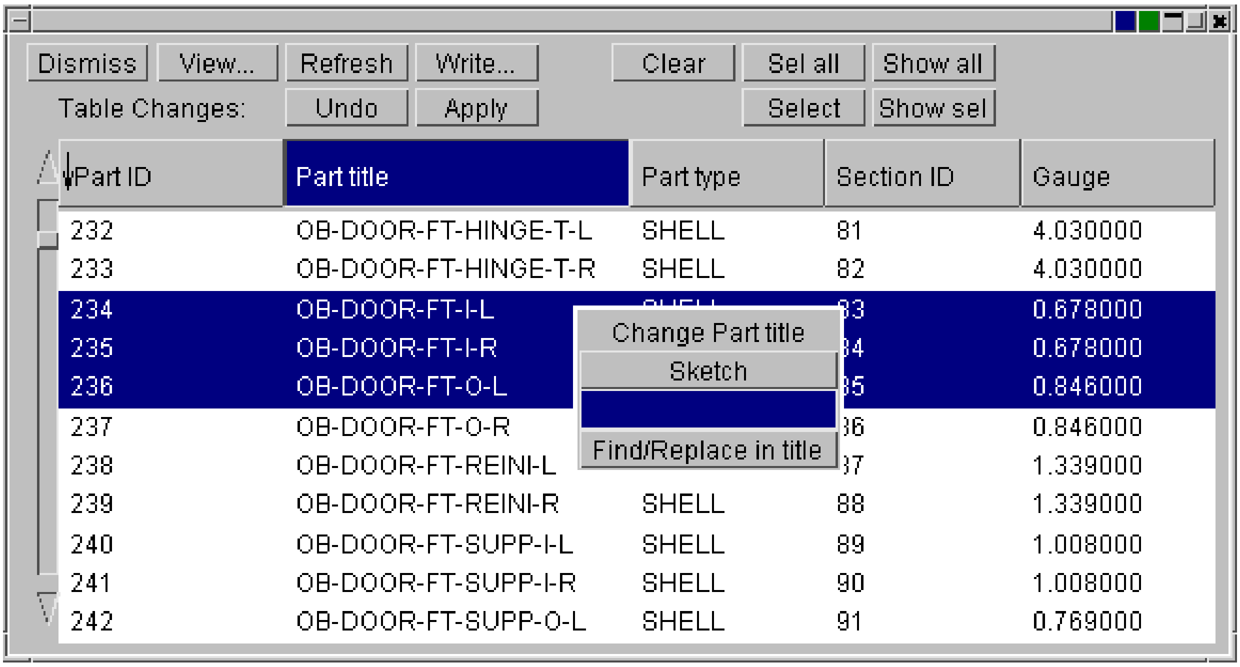


For editable fields, parameters can be added/edited by right clicking on the field and typing an "&" into the input box on the popup. This works in the same way as inputting/editing parameters in edit panels. Typing in & will also bring up a further popup for selection of parameters should they already exist in the model. See section 2.10 for more details.

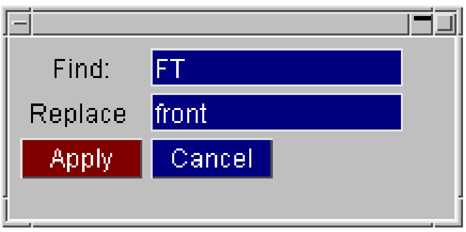


Find/Replace in part title

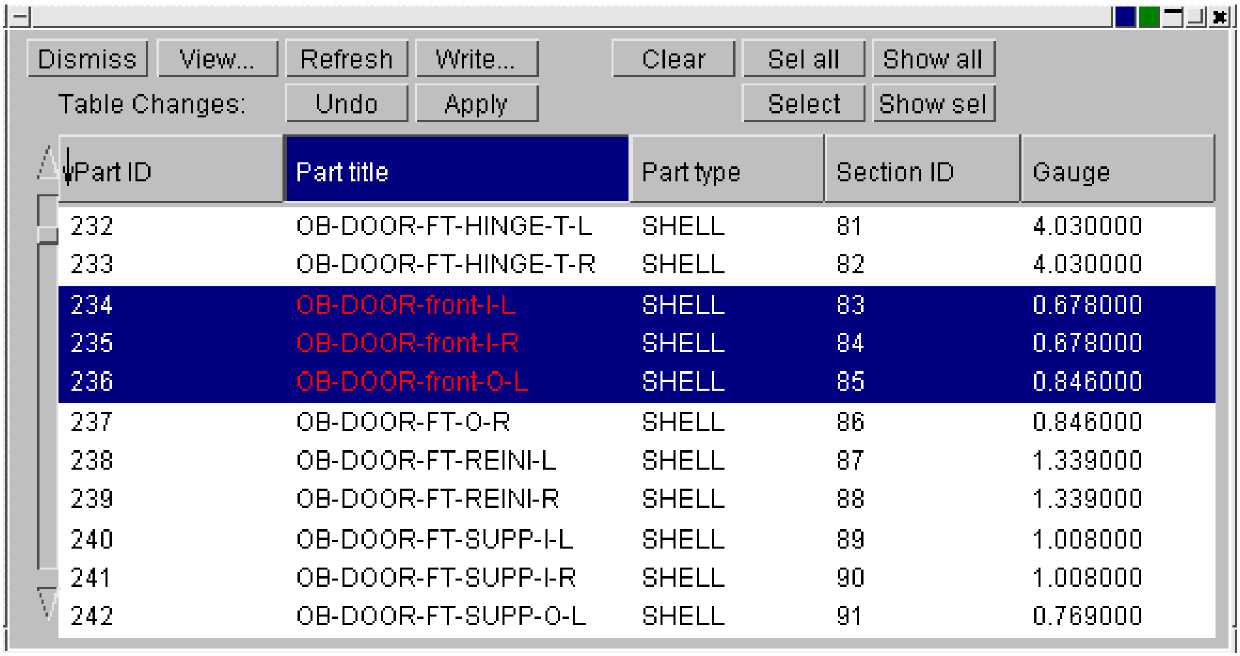
It is possible to search for particular text strings within part titles and replace the text string with another specified text string. This is done through the popup invoked by right clicking over the part title field of selected parts on the part table.



For part titles, the **Find/Replace in title** button is available. Clicking on this will open the find/replace panel.



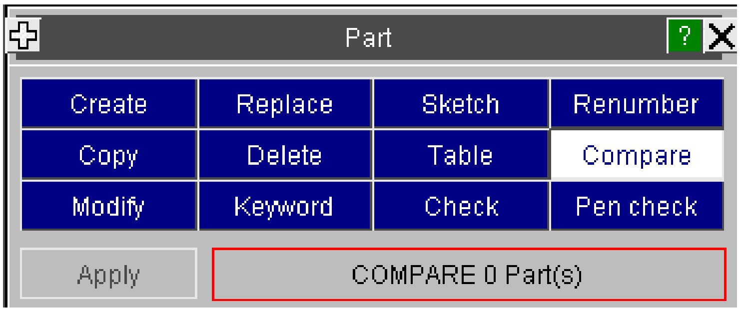
This is used to specify the text sting to search for, and also the text string to replace it with. Clicking **Apply** will apply the title modifications to the part table. Note that the search is case sensitive.



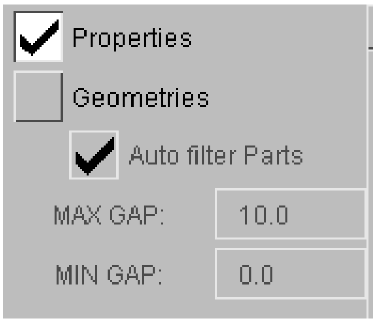
Part ID	Part title	Part type	Section ID	Gauge
232	OB-DOOR-FT-HINGE-T-L	SHELL	81	4.030000
233	OB-DOOR-FT-HINGE-T-R	SHELL	82	4.030000
234	OB-DOOR-front-I-L	SHELL	83	0.678000
235	OB-DOOR-front-I-R	SHELL	84	0.678000
236	OB-DOOR-front-O-L	SHELL	85	0.846000
237	OB-DOOR-FT-O-R	SHELL	86	0.846000
238	OB-DOOR-FT-REINI-L	SHELL	87	1.339000
239	OB-DOOR-FT-REINI-R	SHELL	88	1.339000
240	OB-DOOR-FT-SUPP-I-L	SHELL	89	1.008000
241	OB-DOOR-FT-SUPP-I-R	SHELL	90	1.008000
242	OB-DOOR-FT-SUPP-O-L	SHELL	91	0.769000

As with other changes to the data in the part table, they are shown in red and not applied to the stored part data until **Apply** is clicked.

7.3 PART COMPARE



Part compare runs in one of two modes



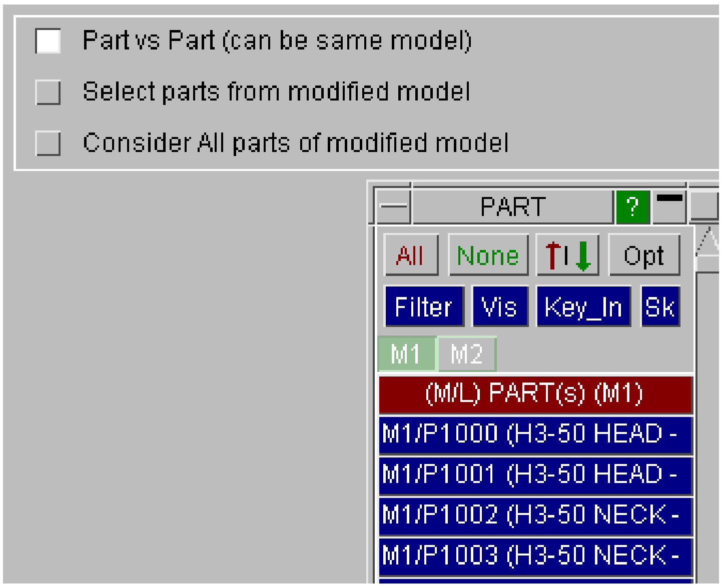
Properties uses the functionality of the part table to make a comparison between the properties of parts. Any parts which do not match for all the criteria tested will be displayed on the table.

Geometries will run a contact type check to detect gaps (using defined min/max values) between a pair of shell parts.

7.3.1 Selection of parts to compare

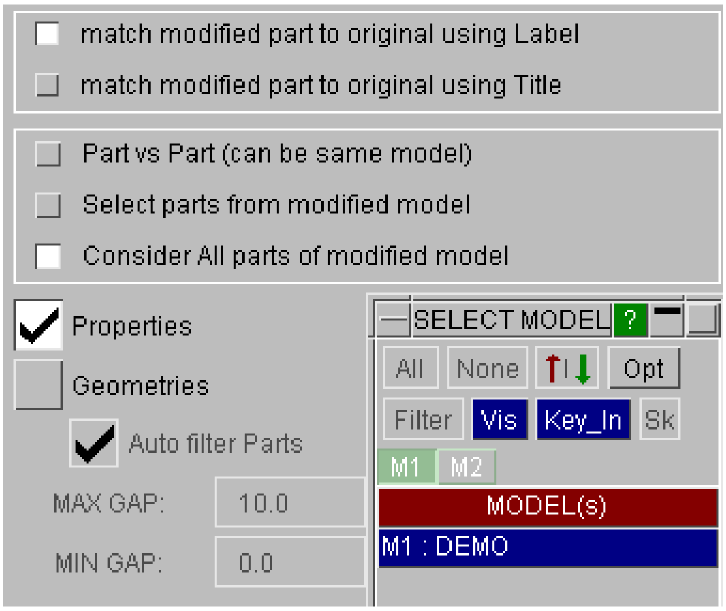
Comparing two parts

In its simplest mode the user chooses two parts top be compared directly. These may be in different models or the same model and will be compared in **Properties** mode.

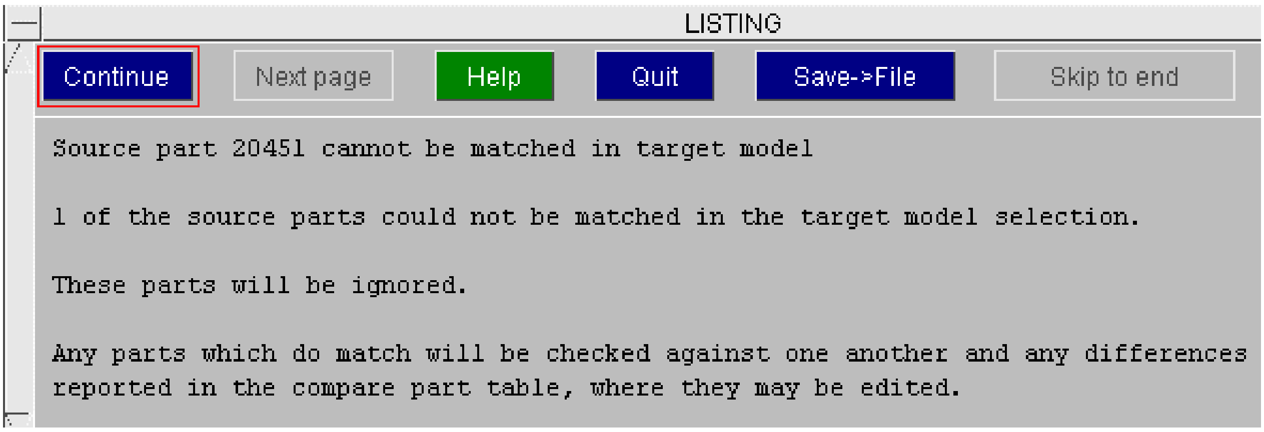


Comparing all parts in one model to another

If multiple models are in memory, the function can be used to compare multiple parts across models. In the mode **Consider all parts of modified model** the user selects the matching method (label or title) and then selects the modified model. If there are two models in memory the other model is automatically taken as the original model, if more than two, the user will need to select the original model.



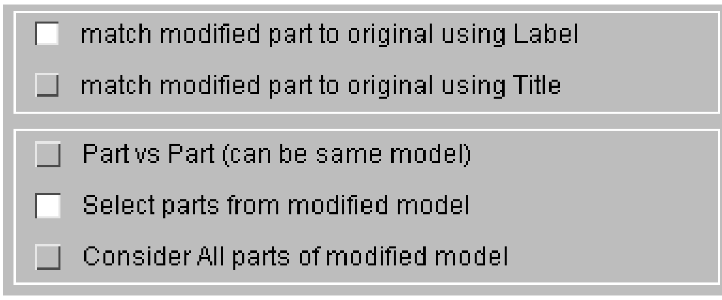
Primer then attempts to match all parts to those of the source model and if applicable will report any that failed to find a match in an information panel.

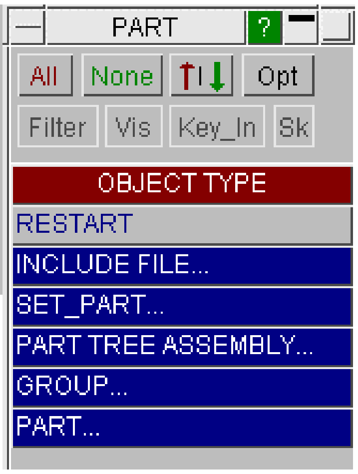


Comparing some parts in one model to another

This is similar to the above, but after selection of the modified model, another object menu will allow you to select a subset of parts in the modified model, for example by include file(s) or one or more assemblies. Once the section is complete, pressing **Next>** will start the comparison process. For large models this may be considerably quicker than comparing all parts (particulary wrt the mass calculation).

Primer will attempt to match the selected parts of the modified model to parts in the original model, using label match or title match as per the option.





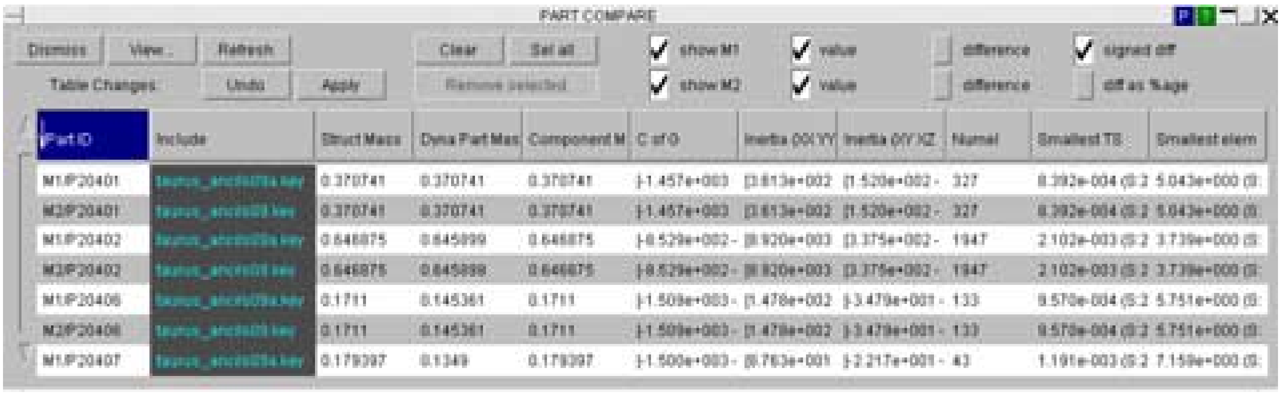
7.3.2 Display of different properties

Properties mode uses the part table.

All the matched pairs are then checked against one another for all the values that the part table treats (see section 7.2) and a part table is constructed for the part pairs which show differences.

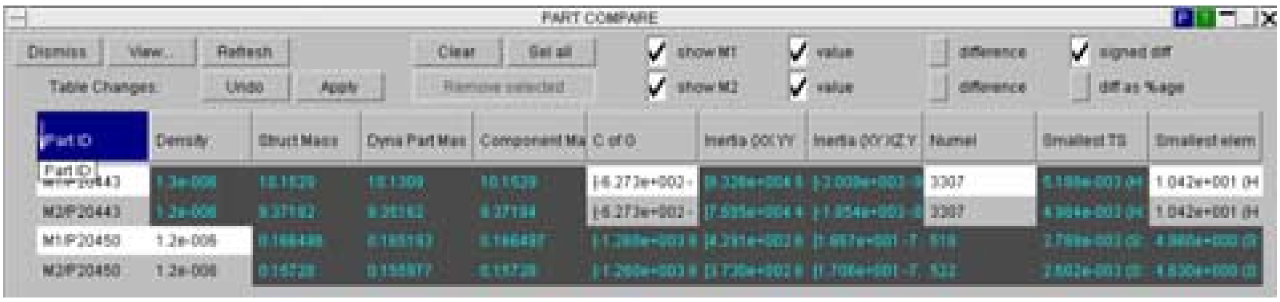
The parts are listed in the form M1/Px, M2/Px, M1/Py, M2/Py, etc and the sorting of the Part ID column will always restore this order.

All the appropriate columns are displayed and the difference is highlighted. In this example, many parts have changed include file.

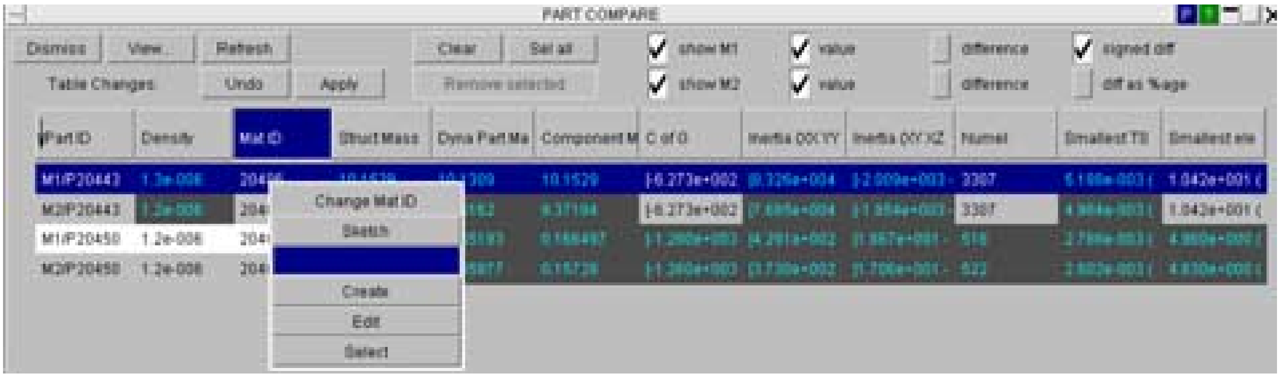


You may use shift-select to select unwanted part pairs and then apply **Remove Selected**.

If we are not interested in the include change, we can use **View..** to de-activate that column and **Refresh** to rebuild the table. Any part pairs from which the **only** diference is their include are now removed from the data stacks.



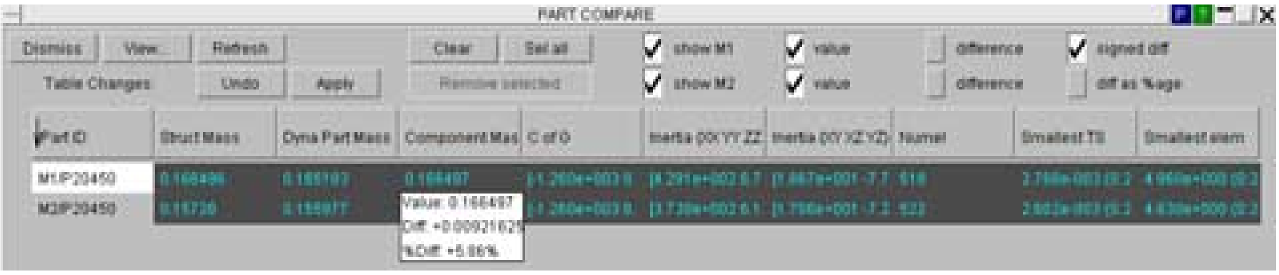
We may then use the part table functionality to investigate and edit data as appropriate. For example, by using **View...** to activate display of **Mat ID** and editing the material for M1/P20443 to correct the density from 1.3e-6 to 1.2e-6.



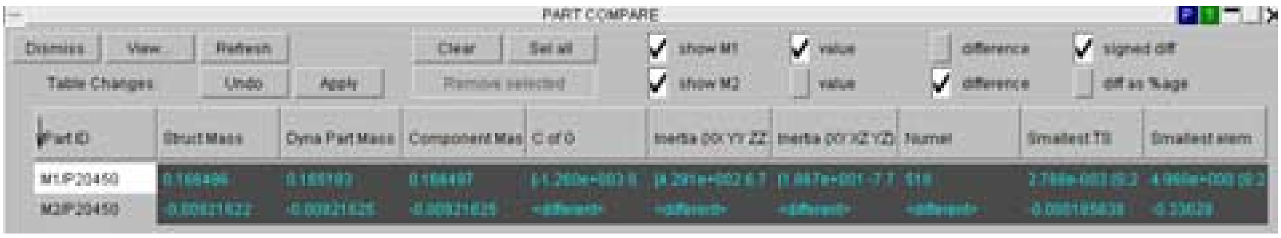
Refresh of the table will then remove part 20443 as the data is consistent across models

Showing the difference

By default the table shows values with hover text to show the absolute and percentage differences wrt the other value.

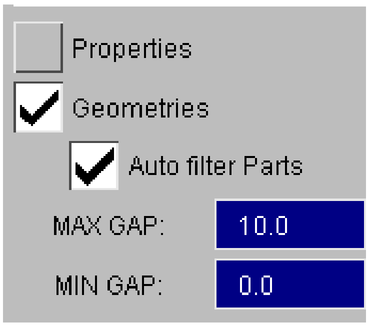


By activating the difference switch you may show the absolute difference for floating point numbers. For other types the string <different> will be written.



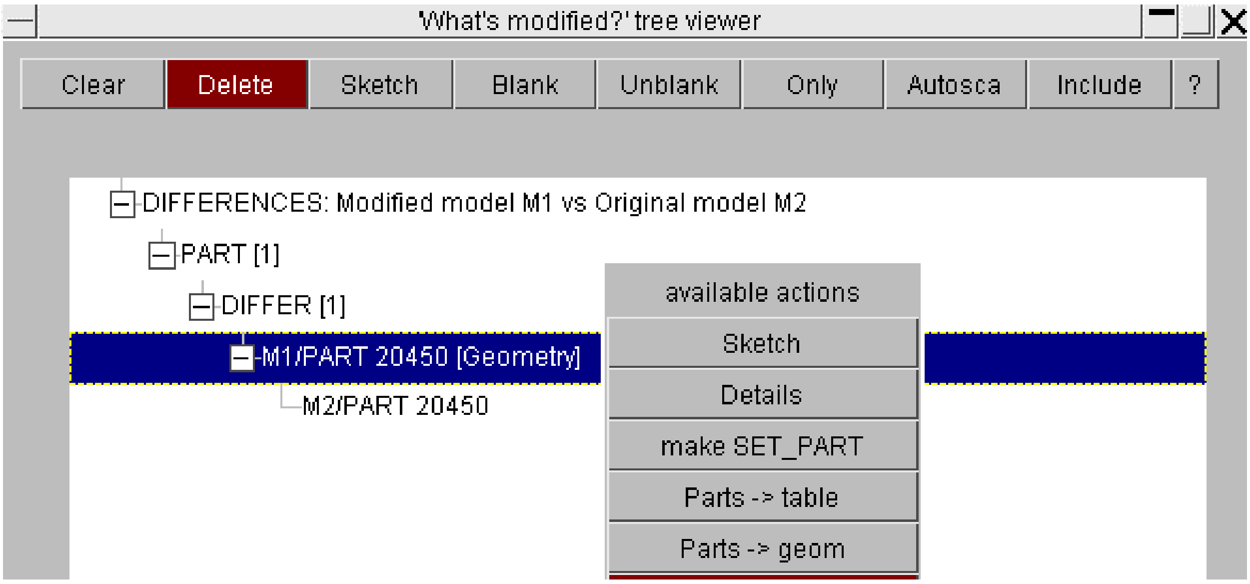
The difference for floating point numbers may also be usefully expressed as a percentage.

7.3.3 Display of different geometries

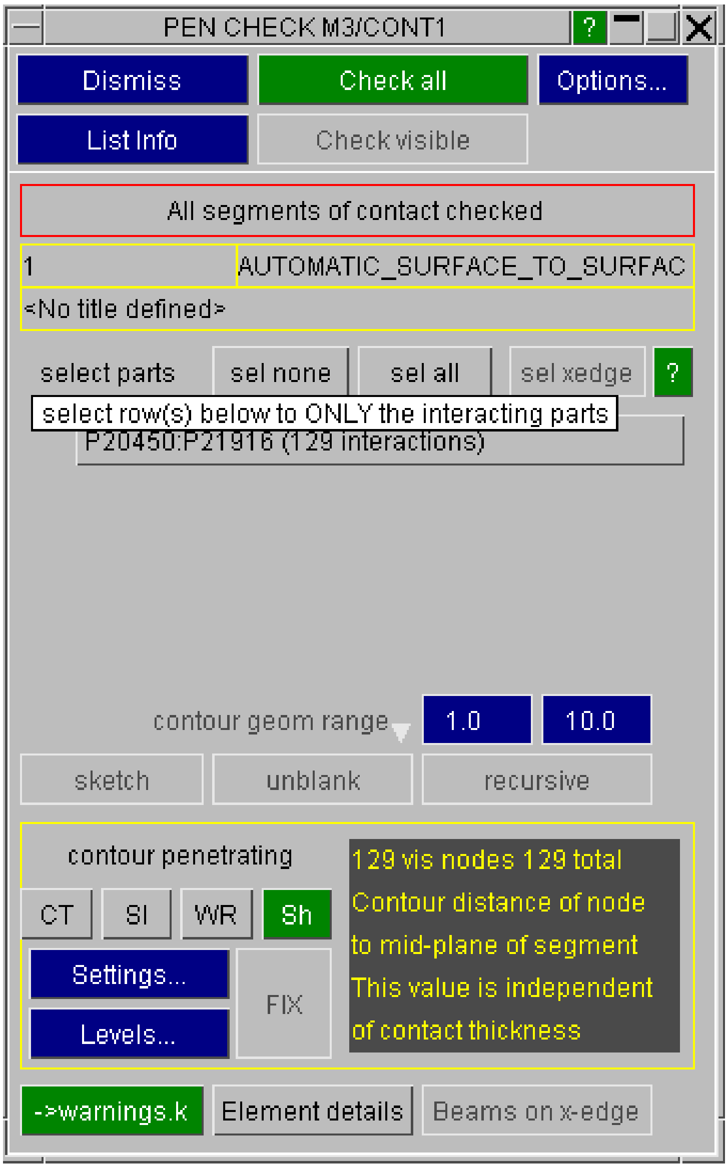


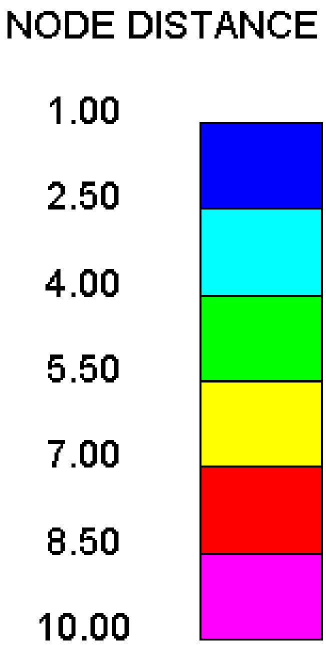
In **Geometry** mode, matched part pairs will be compared to one another using a contact type check which will detect gaps within user defined min/max values (default 0-10mm). If many parts are being checked, the option **Auto filter parts** is recommended to block the test (which can take a few secs per contact check) for part pairs which are unlikely to be geometrically different (i.e. they have the same element count, same geometric CofG and same surface area).

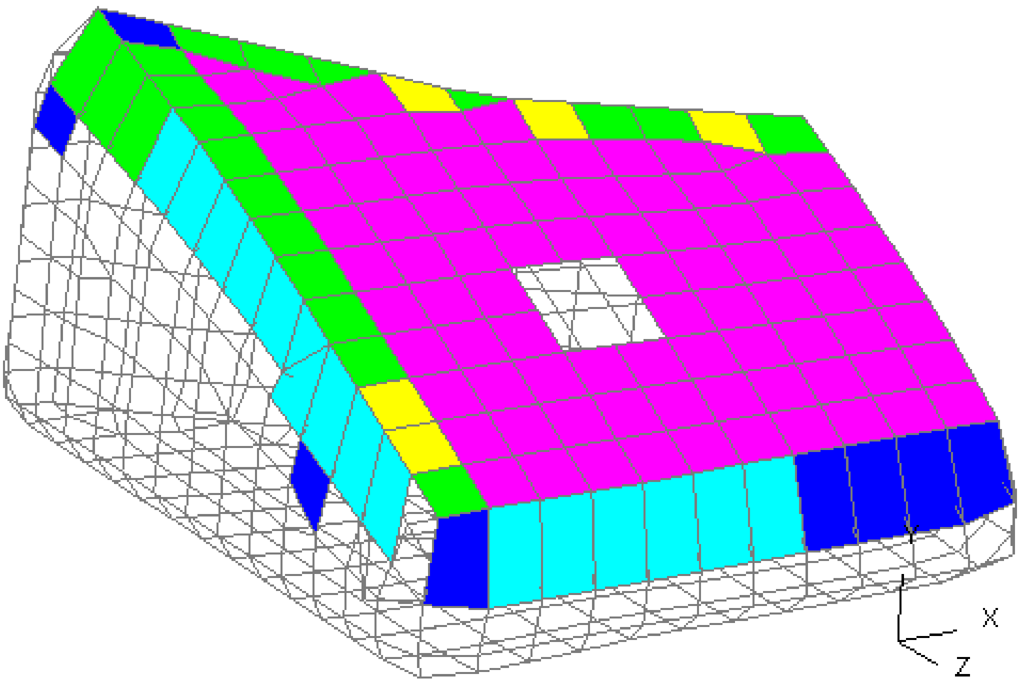
Any part pairs found to be geometrically different are sent to the 'What's modified?' tree viewer where they can be investigated in detail by using **Parts -> geom.**



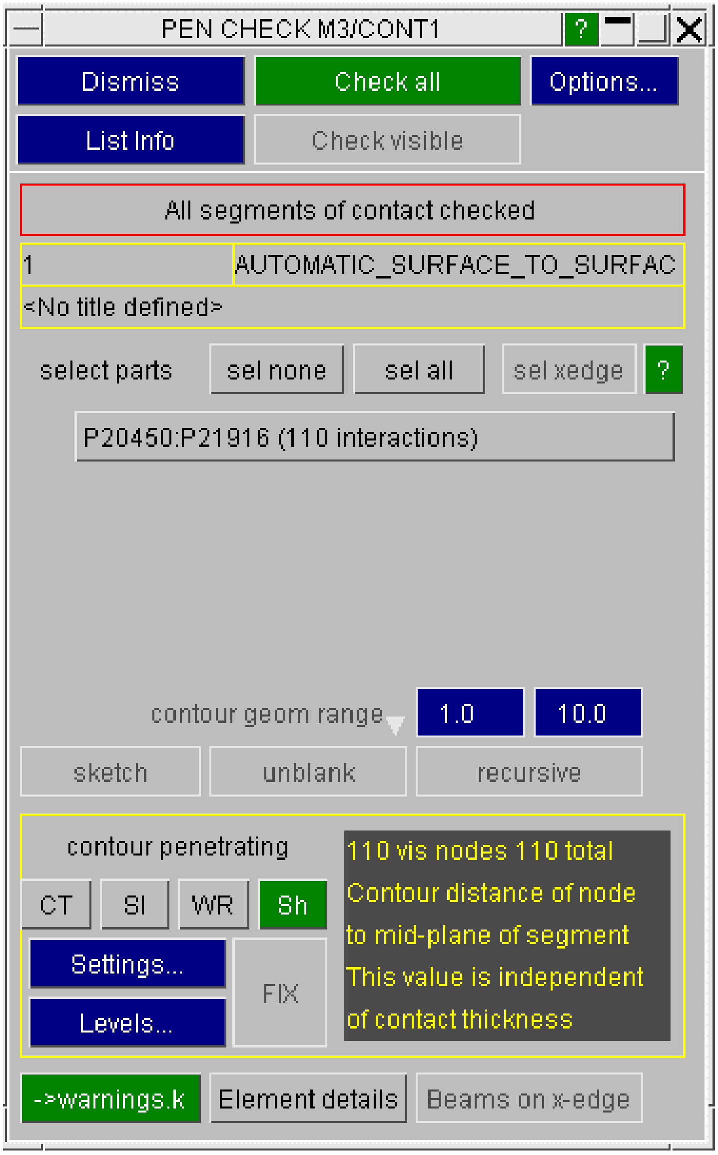
This will copy the parts into a separate model and create a surface to surface contact between them. This can be used to contour the distance away of the nodes of one part to the segments of the other by using **CT** button.

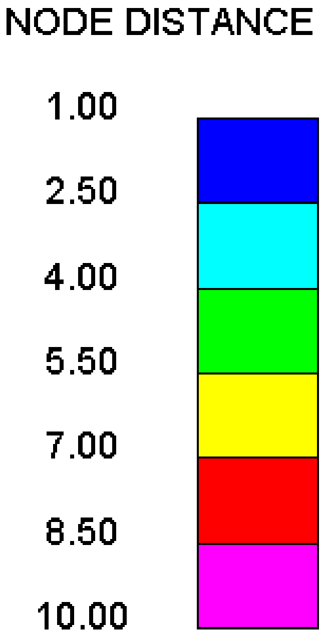

PEN CHECK M3/CONT1
Dismiss
Check all
Options...
List Info
Check visible
All segments of contact checked
1
AUTOMATIC_SURFACE_TO_SURFAC
<No title defined>
select parts
sel none
sel all
sel xedge
select row(s) below to ONLY the interacting parts
P20450:P21916 (129 interactions)
contour geom range
1.0
10.0
sketch
unblank
recursive
contour penetrating
CT
SI
WR
Sh
Settings...
Levels...
FIX
129 vis nodes 129 total
Contour distance of node
to mid-plane of segment
This value is independent
of contact thickness
->warnings.k
Element details
Beams on x-edge

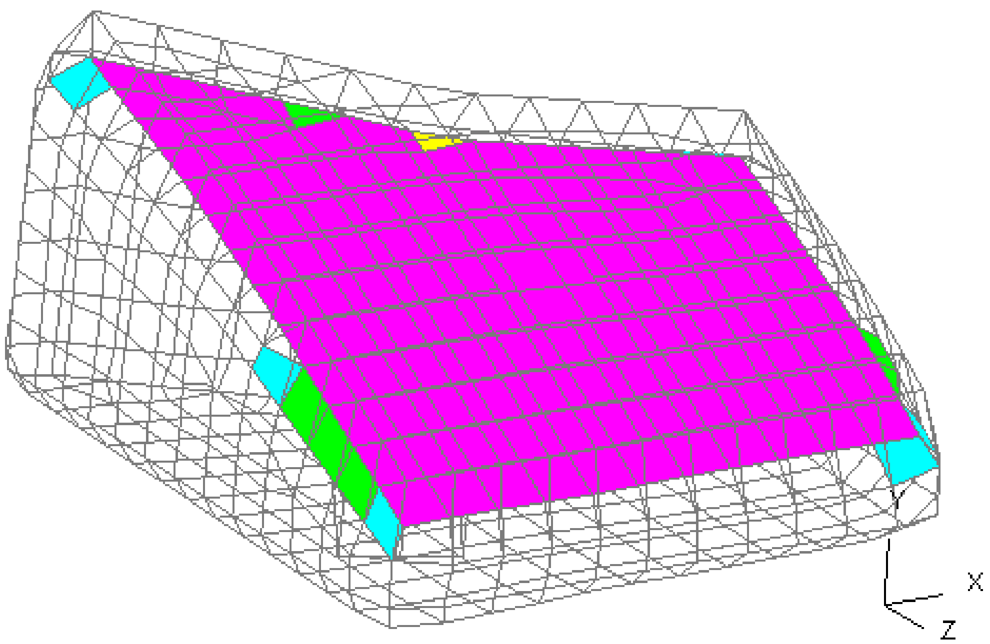

NODE DISTANCE
1.00
2.50
4.00
5.50
7.00
8.50
10.00


X
Z

Observe that the contact is deliberately reversed if **Parts -> Geom** is selected for M2.


PEN CHECK M3/CONT1
Dismiss
Check all
Options...
List Info
Check visible
All segments of contact checked
1
AUTOMATIC_SURFACE_TO_SURFAC
<No title defined>
select parts
sel none
sel all
sel xedge
?
P20450:P21916 (110 interactions)
contour geom range
1.0
10.0
sketch
unblank
recursive
contour penetrating
CT
SI
WR
Sh
Settings...
Levels...
FIX
110 vis nodes 110 total
Contour distance of node
to mid-plane of segment
This value is independent
of contact thickness
->warnings.k
Element details
Beams on x-edge


NODE DISTANCE
1.00
2.50
4.00
5.50
7.00
8.50
10.00


X
Z

Which view is more informative will depend on the geometries involved.

Because this is a rather non-standard use of the contact checker, the user is restricted to operations within this panel until Dismiss is pressed and he is returned to the modified tree. Unfortunately this inhibits all functions in the View box, so a shaded image draw button **Sh** has been added to the check panel. Dynamic viewing operations (rotate and zoom) are not affected by the restriction.

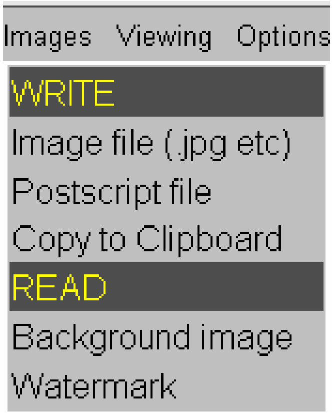
8 Images

Primer can read images in as the background or watermark.

It can also capture graphics and copy to file in two ways:

- By Laser plotting, generating a Postscript file, using the **POSTSCRIPT file** command. Sections 8.0 to 8.4 below
- By a screen grab, generating Bitmap or JPEG files, using the **Image file (.jpg etc)** command in Section 8.5 onwards

Both of these commands are found under the **IMAGES** button



8.0 **LASER**: Introduction to Laser Plotting

By default all graphics images generated by PRIMER are sent only to the screen, but you can choose to copy them to laser files (postscript and pdf files for a laser printer).

This is done by pressing the "Plot" button when you are in Postscripts/PDF.

8.0.1 Laser language and file format used.

At present PRIMER writes Postscript laser files, using PS ADOBE level 2.0 commands, and PDF files. These are ASCII files that can be viewed and edited using any common editor.

"Encapsulated" Postscript files are not written, but later in Section 8.3 the very simple edits required to convert a file to encapsulated form are given.

Laser output is switchable between A4 (297 x 210mm), A3 (296 x 420mm) and US "letter" (11" x 8.5") paper sizes. The Postscript language makes it easy to edit files to fit other sizes.

The laser driver defaults to "PDF" file format, you can opt for "Postscript" laser file.

8.0.2 Number and orientation of plots on a page.

The laser driver defaults to "landscape" orientation, with one plot per page. You can opt for "portrait" orientation and, in both cases, put multiple plots on a page in a variety of layouts.

8.0.3 Resolution setting.

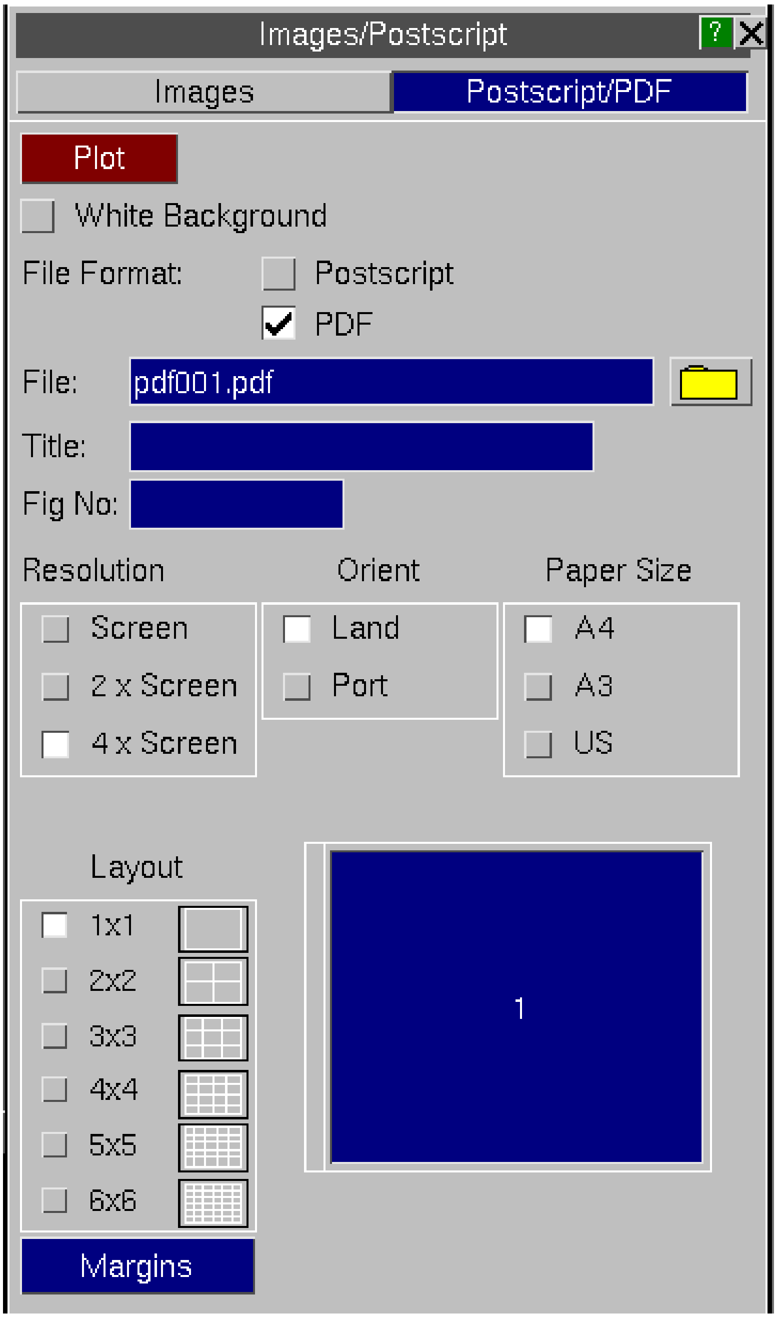
Postscript and pdf files generated can have higher than current screen resolution. Screen resolution, twice and four times screen resolutions are all available.

8.1 Controlling laser plotting using the Laser Plotting panel

This figure shows the basic laser plotting panel.

This is invoked by the Postscript/pdf command under Images->Write in the top menu box.

It both controls and shows the status of the current laser file (if any).



8.1.1 Plot button

Press "Plot" button when you want to plot the current view on the screen. With the White Background option switched on, images will be plotted with a white background. Entity labels and screen text will be switched to black. Once the image has been captured the screen will return to its original colours.

Any plot directed to laser file is sent by default to the next free sub-image (if the file has multiple plots per page), or file (if only a single image per file, or the multiple page is full).

When multiple sub-images in a file are in use the next image to be written is shown by depressing the appropriate icon in the file layout panel. You can override this and choose a different sub-image: see Section 8.1.5 below.

8.1.2 Choosing the laser filename

When no file is currently in use the **File:** entry box will be available. You can give any valid filename for the next laser file to be written, or let PRIMER choose one for you. You can also use the button to select a file via the standard file filter box.

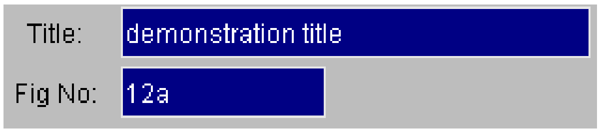
If the file already exists you will be queried to check that you genuinely want to overwrite it: you cannot append to existing laser files.

The default naming convention used by PRIMER for postscript laser files is **`postNNN.ps`**, where:

> **`NNN`** is a 3 digit number (with leading zeros if required) in the range `001 - 999`.

Any existing files are skipped when the next file in the sequence is computed.

8.1.3 Defining a label and figure number for laser plots.



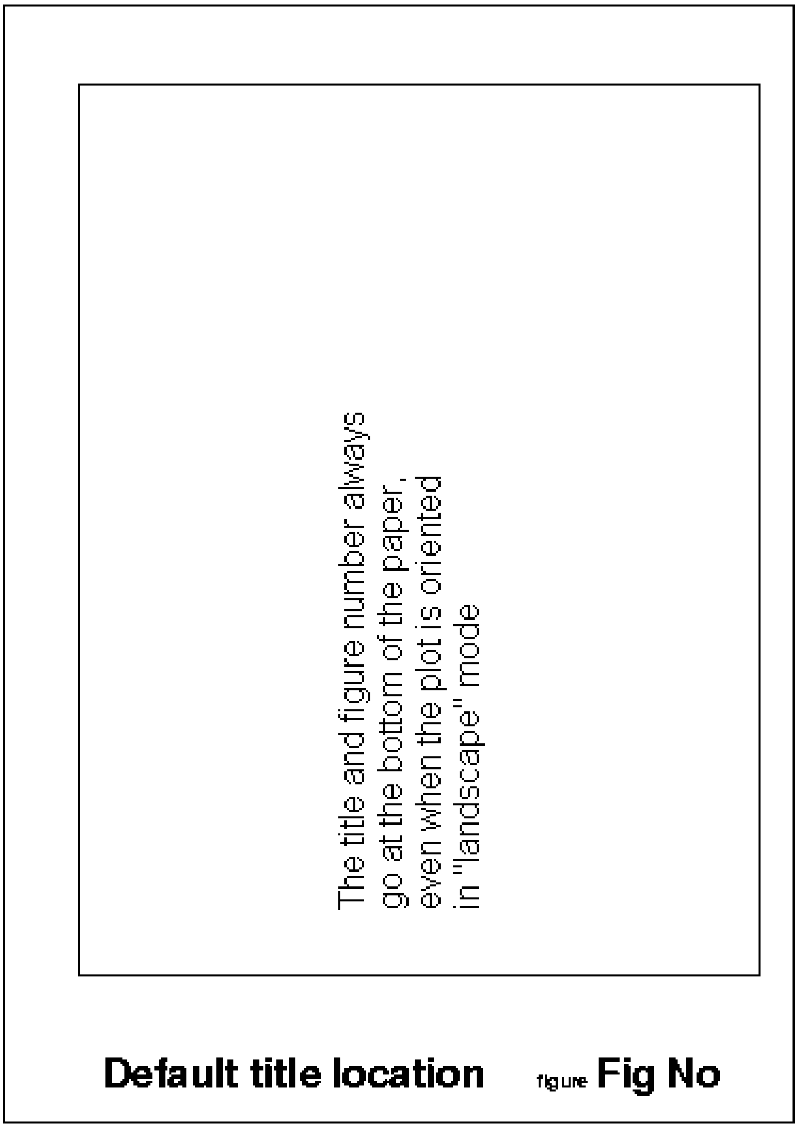
By default laser files are not labelled and have no figure number, but you may add either or both of these. They are always put at the bottom of the page, along the short edge, regardless of the orientation used for plots.

This figure shows the standard locations for title and figure number on laser plots.

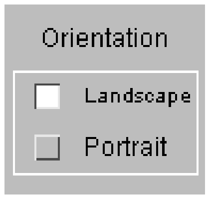
The title may be up to 80 characters long, and is split over two lines if necessary by PRIMER.

The figure number may be any string (not just a number), and is preceded by the word "figure". It is suggested that it is 6 characters or less long: here "12a" was used.

This plot is written in "landscape" format, and reinforces the point that the title and figure number always go at the bottom of the paper, regardless of the orientation of the plot contents.

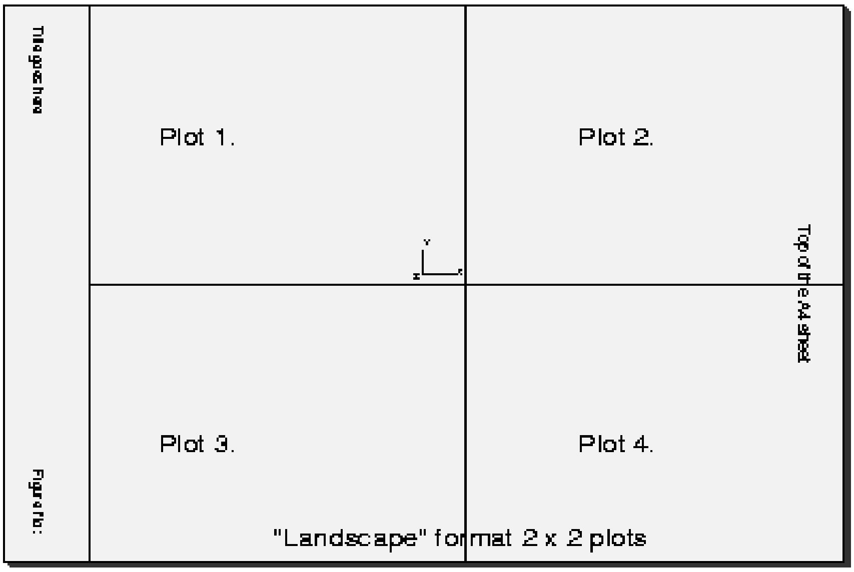


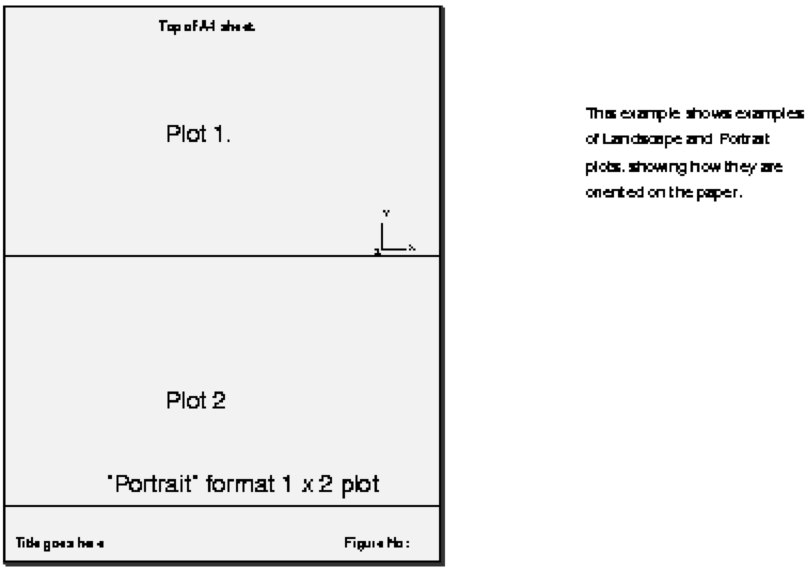
8.1.4 **Orientation** Setting Landscape or Portrait plot orientation.



By default plots are in "Landscape" orientation, with the long side of the plot aligned with the long side of the paper, but you can choose "Portrait" format instead.

The figure below shows examples of both landscape and portrait format plots, showing how they are aligned on the paper.



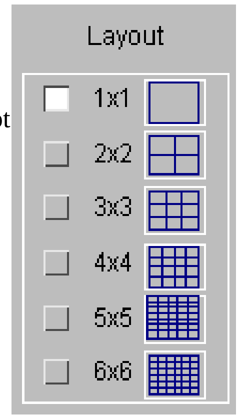


8.1.5 Layout

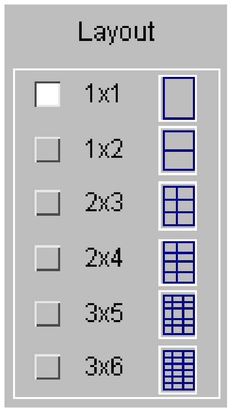
In both landscape and portrait formats it is possible to have more than one plot on a page.

Various pre-programmed permutations of <#x> x <#y> plots are available as shown here.

Each individual plot on a page will be referred to from now as a "sub-image".

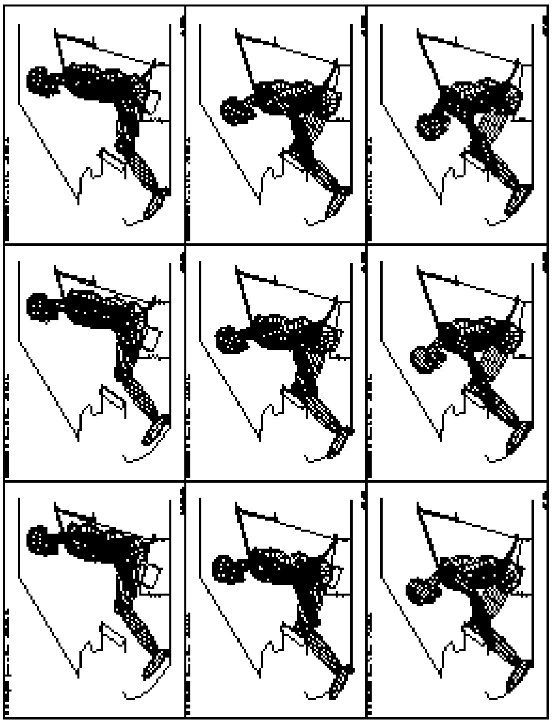


LANDSCAPE

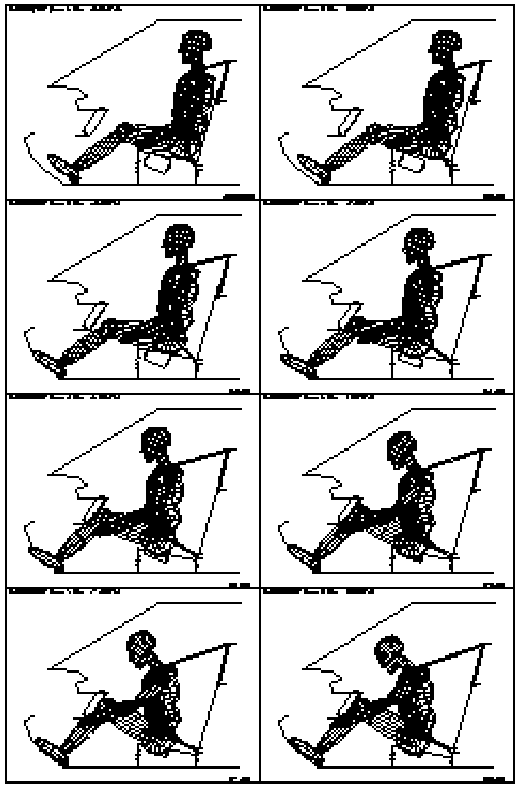


PORTRAIT

The figures below show examples of 3x3 Landscape and 2x4 Portrait multiple plots.



EXAMPLE OF 3 x 3 LANDSCAPE OUTPUT figure 7.1.5a



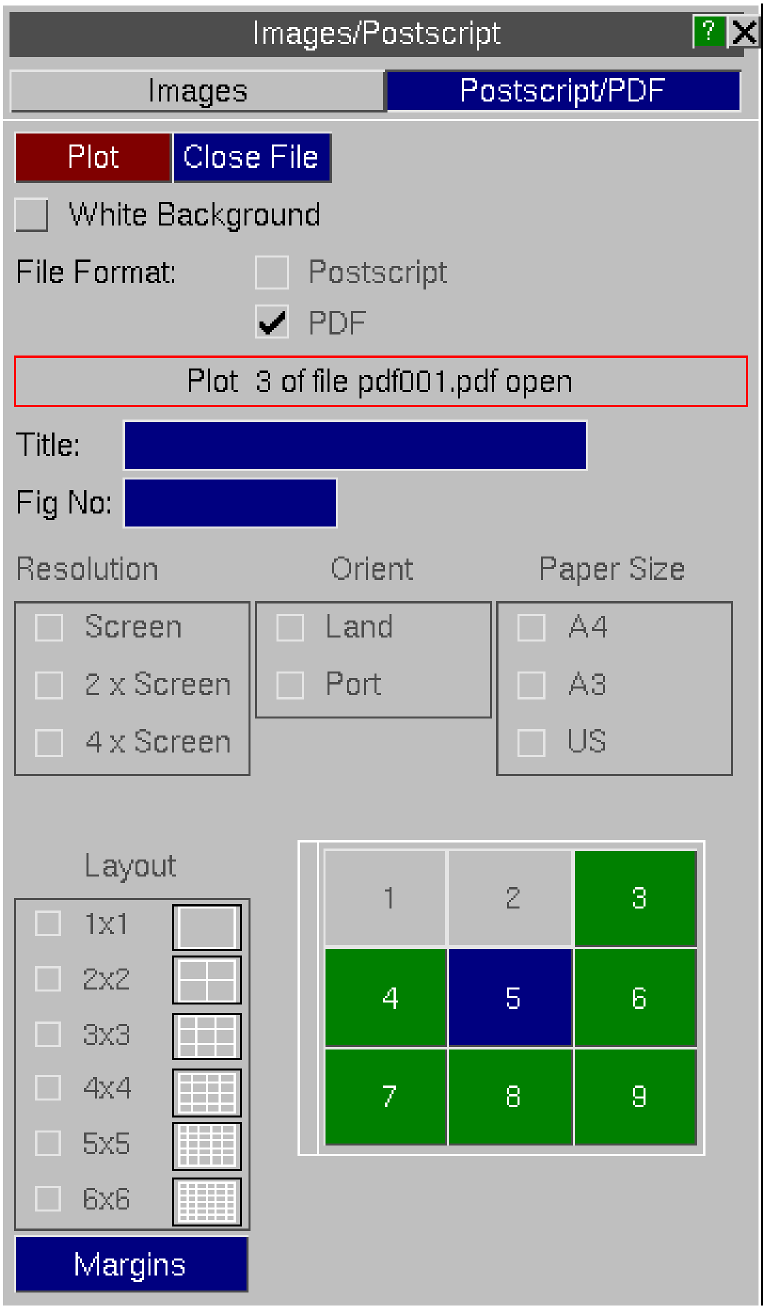
EXAMPLE OF 2 x 4 PORTRAIT OUTPUT figure 7.1.5b

Controlling the order in which multiple plots are drawn.

The right hand menu shows a typical laser panel for a 3x3 portrait plot in which sub-images 1 and 2 are complete.

Normally sub-images are written in the order #1 to #n, but if the user wanted the next plot to be drawn to sub-image #5 instead of #3, he would click on the [5] icon where the button gets coloured in blue instead of the [3] icon as it normally would.

Next sub-image would be the next free one, i.e. #3 to receive the next plot. The [3] icon will be coloured in blue.



The status of files, and sub-images within files.

PRIMER laser files, and sub-images within files, have one of four possible states.

Inactive	**Green**	No graphics written yet, and not selected for the next plot.
Selected	**Blue**	No graphics written yet, but selected to receive the next plot.
Closed	Greyed out	File/sub-image complete, and cannot receive any more information.

The colours referred to above are used for the button icons on multiple sub-image panels, as shown in the figure above. Only green icons (ie those which are currently inactive) may be selected to receive the next image.

How sub-image status affects the destination of graphics.

(1) If no graphics have been written to a sub-image then the next plotting command will send laser output to the the sub-image currently "selected".

By default this will be the lowest numbered sub-image that has not yet been written to, but you can choose another as described above.

(2) Once graphics have been sent to the sub-image its status changes to "closed". This means that it cannot receive further graphics.

Interaction between sub-images and files

A file with only a single image in it is treated in exactly the same way as an individual sub-image above, except that it is (implicitly) always "selected" for plotting until something is drawn in it.

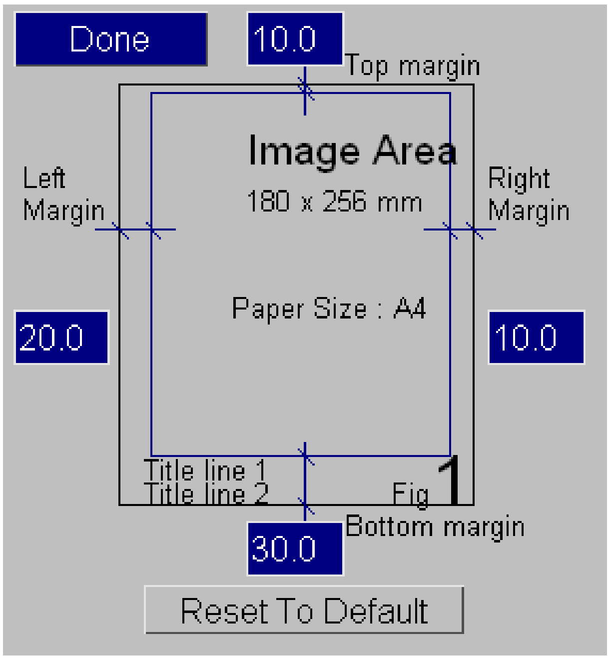
A file with sub-images remains current (ie open) until all of the sub-images in it have been "closed", or the user closes it prematurely with a **CLOSE FILE** command. Then PRIMER defaults to the next default filename as defined in Section 8.1.2 above.

The importance of closing files.

While a file is still current it is still connected to the programme, and at least some of its contents will still be held in system buffers. If you want to send it to a printer you **must** close it first using a **CLOSE FILE** command.

This flushes any remaining data to disk and disconnects the file from the programme.

8.2 MARGINS... Modifying laser paper size on the page.



The **MARGINS** button in the laser control panel gives a special sub-menu that allows you to select the margins on all sides:

The margins will only apply to the axis of the plot that comes closest to the paper borders; the other axis margins will be overridden to maintain the correct aspect ratio of plots (ie no image distortion).

8.3 Creating Encapsulated Postscript (EPS) files.

EPS format is used by many software packages to import postscript images. The laser files written by PRIMER are not in EPS format, but only two very simple edits at the top of the file are required to change this.

The first seven lines of any PRIMER laser file look like this:

To convert it to EPS format you must add a "**%%BoundingBox:**" line, and delete the "**statusdict**" line. Thus this file becomes:

```
%!PS-Adobe-2.0
%%EndComments
%%Pages: 1
%%Page: 1 1

statusdict begin
/a1test save def
```

```
%!PS-Adobe-2.0
%%BoundingBox: 0 0 595 842
%%EndComments
%%Pages: 1
%%Page: 1 1

/a1test save def
```

The arguments of the "BoundingBox" line are the Postscript coordinates:

<lower left> <lower right> <upper left> <upper right>

These must be expressed in raw Postscript space of 72 points per inch, and they assume that the paper is in portrait format with its origin at its lower left corner.

The values in the example above refer to A4 format: 210 x 297 mm = 595 x 842 points; US "letter" paper would give 8.5" x 11" = 612 x 792 points. Clearly a smaller bounding box would select only a subset of the image.

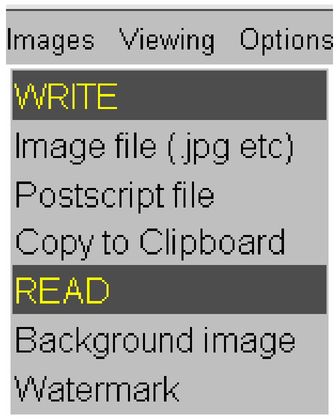
For more information on encapsulated postscript see the "PostScript Language Reference Manual, 2nd edition" by Adobe Systems Incorporated. (Published by Addison Wesley, ISBN 0-201-18127-4)

8.4 Notes on laser plotting

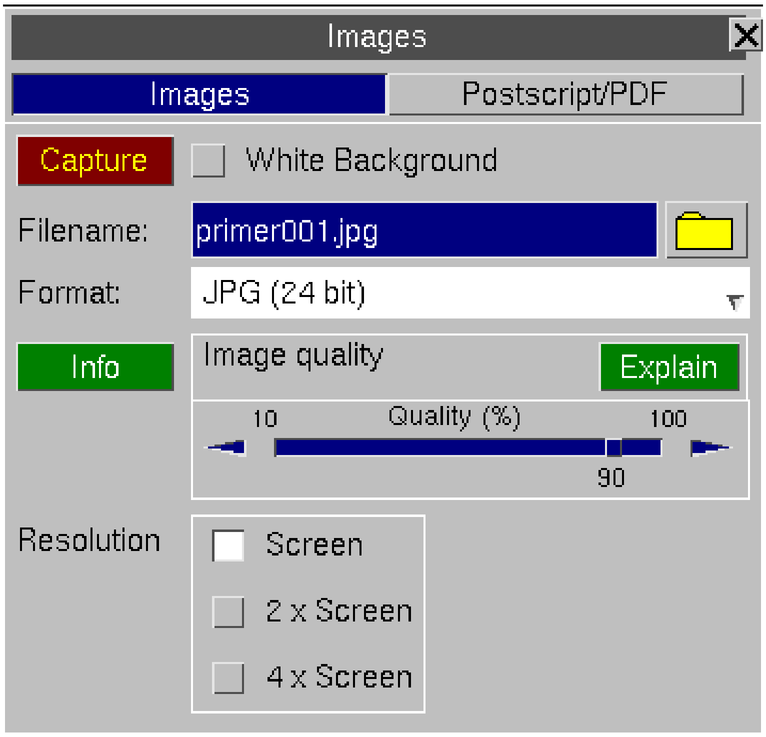
- Users on 3D devices should note that turning the laser on will temporarily force the graphics mode back to 2D. This is because a laser plot is intrinsically a 2D image and is computed in software.
- Transient graphics added "dynamically" to the screen are never copied to laser files. Examples are cursor-pick symbols, and also the information added interactively with the **DYNAMIC_LABEL** function.
- If an attempt to open a laser file fails because the file/directory refuses "write" permission, or the disk is full, you are warned and laser output is switched off.
- You can switch laser output **off** and **on** at will in the course of assembling a file with multiple images. Sub-images will only be written when the laser is on.
- Some of the defaults here may be preset outside PRIMER via preferences in the `.oa_pref` file: see Appendix II.

8.5 Raster Images

The function is invoked by selecting **Image file** from the **Images** option in the main menu.



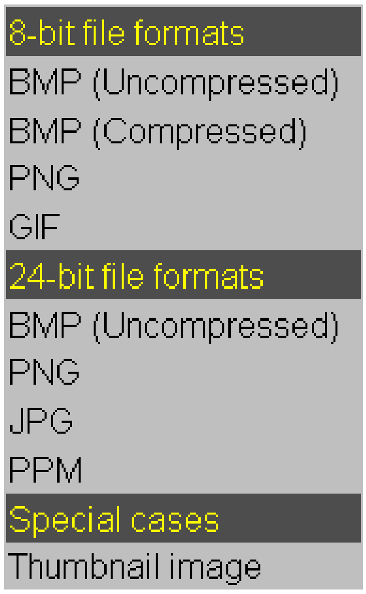
8.5.1 CAPTURE: Copying the current image to file



Captures a single static frame in one of the following formats:

With the White Background option switched on, images will be captured with a white background. Entity labels and screen text will be switched to black. Once the image has been captured the screen will return to its original colours.

8-bit file formats



BMP Uncompressed Uncompressed 8 bit Microsoft windows bitmap. The approximate size of the file is [image width * image height] bytes.

BMP Compressed 8 bit runlength encoded (RLE) Microsoft windows bitmap.

PNG 8 bit lossless compressed **P**ortable **N**etwork **G**raphics image.

GIF 8 bit lossless compressed **G**raphics **I**nterchange **F**ormat.

24-bit file formats

BMP Uncompressed 24 bit Microsoft windows bitmap. The approximate size of the file (in bytes) is file size = 3 * image width *image height

PNG 24 bit lossless compressed **P**ortable **N**etwork **G**raphics image. PNG offers the similar degree of compression as GIF but has better colour quality.

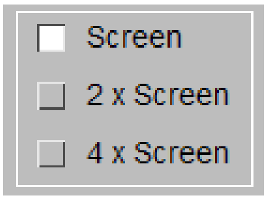
JPG Joint Photographic Experts Group compressed format. This gives image quality nearly comparable to 24 bit-plane bitmaps, but with a file of < 5% the equivalent size. JPEG format is supported by all common visualisation packages and is recommended for all applications unless image quality is of paramount importance.

PPM 24 bit uncompressed **P**ortable **P**ix**M**ap. The approximate size of the file is [3 * image width *image height] bytes.

Various `.bmp` formats are available, and there are Controls for the dithering of the 8 bit-plane variants and palette optimisation .

RESOLUTION

All images can be output at either the screen resolution or at a resolution of either 2 or 4 times the screen resolution.



8.5.2 Controls on the quality of 8 bit-plane bitmap files.

24 bit BMP files tend to be huge, and the space saved by using the compressed 8 bit format is attractive. The trouble is that without further processing the image quality obtained when 24 bit images are truncated to 8 bits is definitely not!

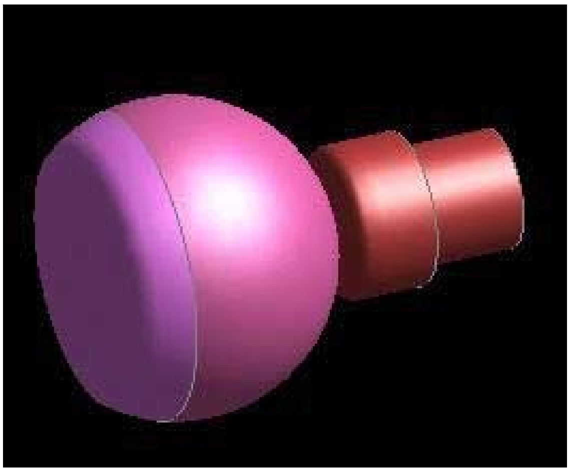
The problem is the number of colours available in the 8 bit format is 2**8 = 256, and this gives rise to "banding" when the least significant bits of the original colour definitions are lost as the original 16 million colours are truncated.

Dither image

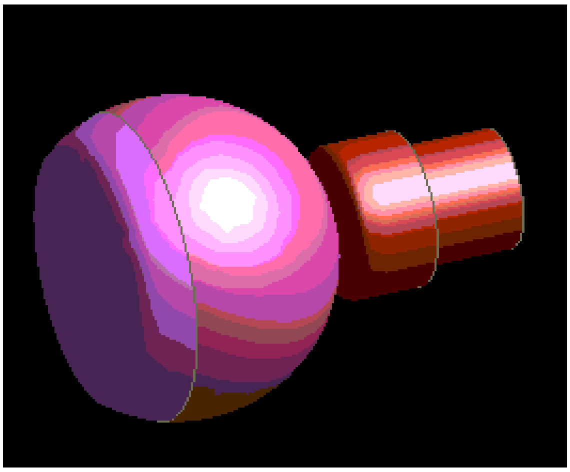
The following sequence of images show how the different levels of dithering affect the quality and file size of a compressed 8 bit image, comparing it with the JPEG equivalent.

For static images there is no advantage in using BMP files over JPEGs: they are larger and of inferior quality. However if you are unable to use MPEG animation, and have to revert to AVI format, the various permutations of image quality and filesize below will be of interest when trying to obtain the best compromise between image quality and overall file size.

The ideal would be an AVI file composed of JPEGs, or MJPEG format. Sadly all such formats are proprietary and, perhaps as a consequence, are not supported by the typical players currently available. Hopefully this will change in the future.



Here is the original 24 bit-plane image, saved as a JPEG file. **Size 5.1kB**



This is the undithered equivalent bitmap image. **Size 7.3kB**.

Note how the discretising affect of mapping onto a limited colour palette has caused "banding" which makes the image almost unusable. However at least the files are small!

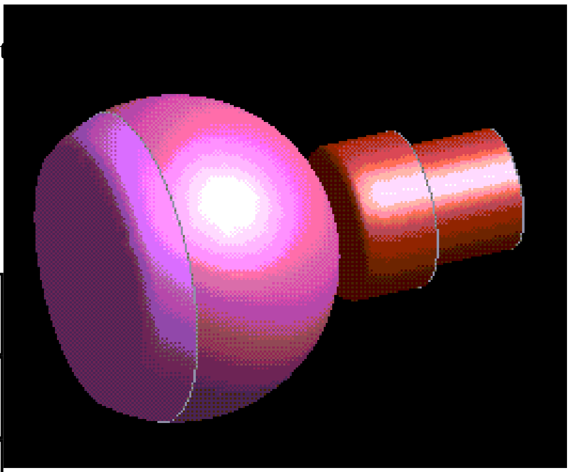
"Dithering" is a technique in which an ordered pattern of noise, **xxxxx** in the table below, is added to the least significant bits of a colour value to make it alternate between two adjacent shades.

Consider the bits for a single colour in this image that have been truncated from 24 bits (8 bits each of Red, Green and Blue). Truncated bits are shown in lower case.

Original Red byte `001?????`	truncates to	`001ooooo`
Adding the dither pattern **oooxxxxx** to the bottom 5 bits of the original byte gives		
`001????? + oooxxxxx`	giving either or	001ooooo 010ooooo

The result may be truncated to `001ooooo`, or the increased to the next shade up `010ooooo`, depending on the trailing bits `?????` and the noise value **xxxxx** at that pixel.

The effect is to produce a composite shade that is somewhere between the two originals.



This is the "Shifted 2x2" dithered bitmap image. **Size 19kB**.

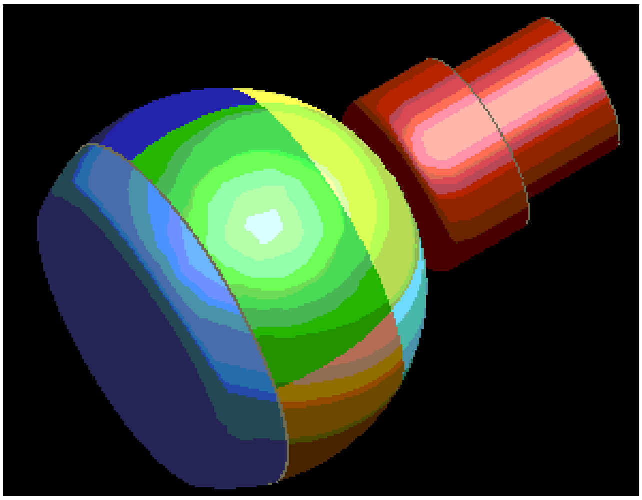
Some reduction in banding is evident, but the quality is still poor and not worth the saving in file size over normal 2x2 unless you are desperate for space.

Palette Optimization

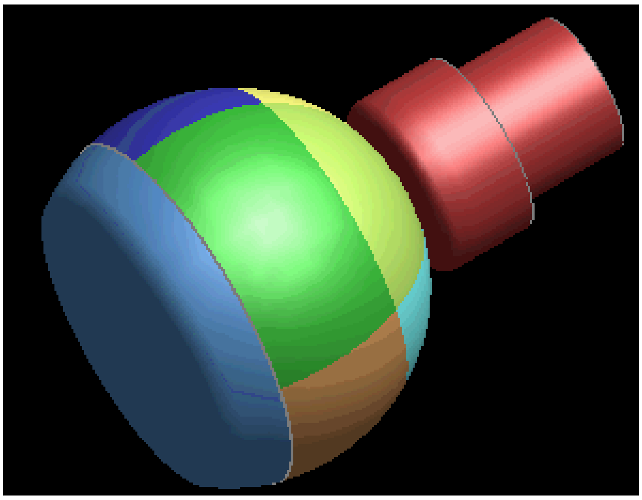
When 8 bit images are produced the 24 bit palette has to be reduced to only 256 colours. To do this the best way is to use Palette Optimization to choose the most representative colours used in the image.

Without Palette Optimization 256 colours can be chosen uniformly along the original 24 bit palette, missing out important colours.

The following figures show the differences in images with Palette Optimization.



This is the original image, saved as a GIF, with no dithering or palette optimization. **Size 6.1kB**

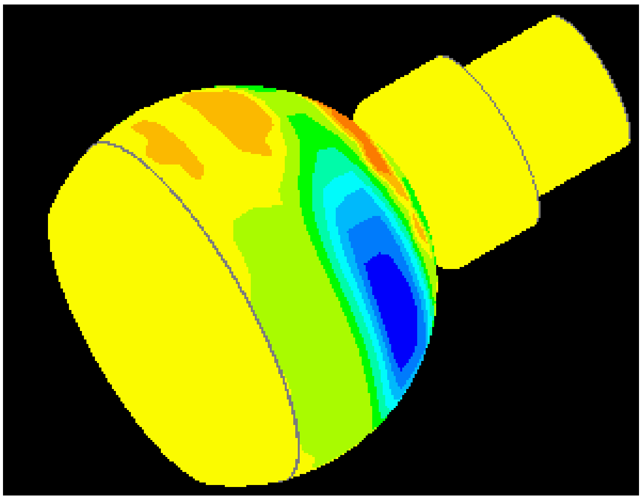


This is the image, saved as a GIF, with palette optimization. **Size 12.6kB**

Note that whilst there are still bands, the coarseness of them has diminished.

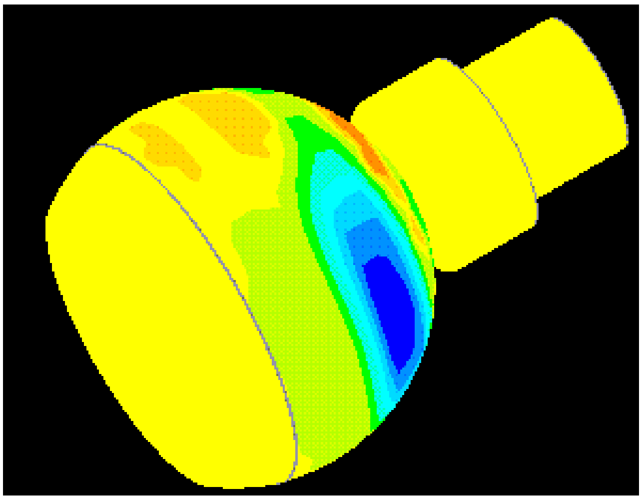
The two images above were created from a shaded image plot in PRIMER and therefore contained a lot of different colours. The banding is present because the palette has been reduced to the 256 most representative colours in the image.

The following figures are images of a contour plot in PRIMER.



This is the image, saved as a GIF, with palette optimization. **Size 3.1kB**

Note that due to the smaller number of colours in the image, there is no banding.



This is the image, saved as a GIF, with dithering. **Size 3.7kB**

Dithering is not well suited to images with distinct colours since by its nature it produces colours that are somewhere in between neighbouring colours. This is effective with shaded images, but not with images where there are sharp changes of colour.

8.5.3 INFO: Further online help about formats

This gives an online "help" message explaining what the various formats are.

8.5.4 Capturing the contents of "menu" windows.

The **IMAGES** command only captures the contents of the graphics window. To copy any other window on the menu interface to a bitmap file use the **SAVE->BITMAP** option in the popup menu belonging to the `[-]` button at its top left corner. (See section 2.4.1).

This distinction is required because the "menu" windows are typically running in X11 window manager overlay planes, whereas the graphics window may be X11 or OpenGL, and is generally located in the screen’s image planes. Trying to capture an image which is a composite of different windows, bit-plane depths, physical location in the hardware and graphics type is possible but difficult!

8.5.5 Capturing the contents of all the Primer windows

If you want to grab an image of the whole Primer window contents: graphics, menus, the lot, then:

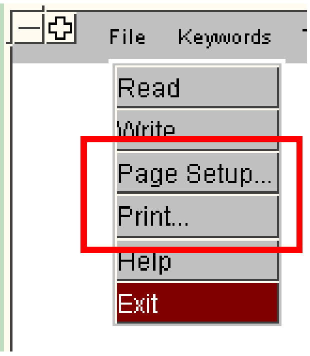
On Windows platforms:
- Use <Alt><Print Screen> with the mouse inside the Primer window
- This will place the image in the Windows "paste" buffer.
- It can then be pasted into other applications.

On Unix platforms:
- Use "xwd -out <filename>" and click inside the Primer window.
- This will create and "xwd" format file which can be used elsewhere.
- Some Unix platforms may have other, non-standard screen grabbing software.

If some windows appear to have the wrong colours this will be because the X "visuals" in the various windows are different, which is invariably the case when OpenGL graphics are used, and this has confused the screen-grabbing programme. To fix this:
- Set the **`SM_USE_VISUAL`** environment variable to "default" (eg **`setenv SM_USE_VISUAL default`**)
- Select device **XMENU...** when starting Primer, (see section 1.2), and choose the visual marked "default"

These two actions will force both menus and graphics to be drawn in the default visual of the window manager, which will maximise your chance of getting a sensible screen grab. However the graphics window will be using X11 (2D) rather than OpenGL (3D) graphics, and the resulting image may be inferior.

8.5.6 PRINT (Windows only)

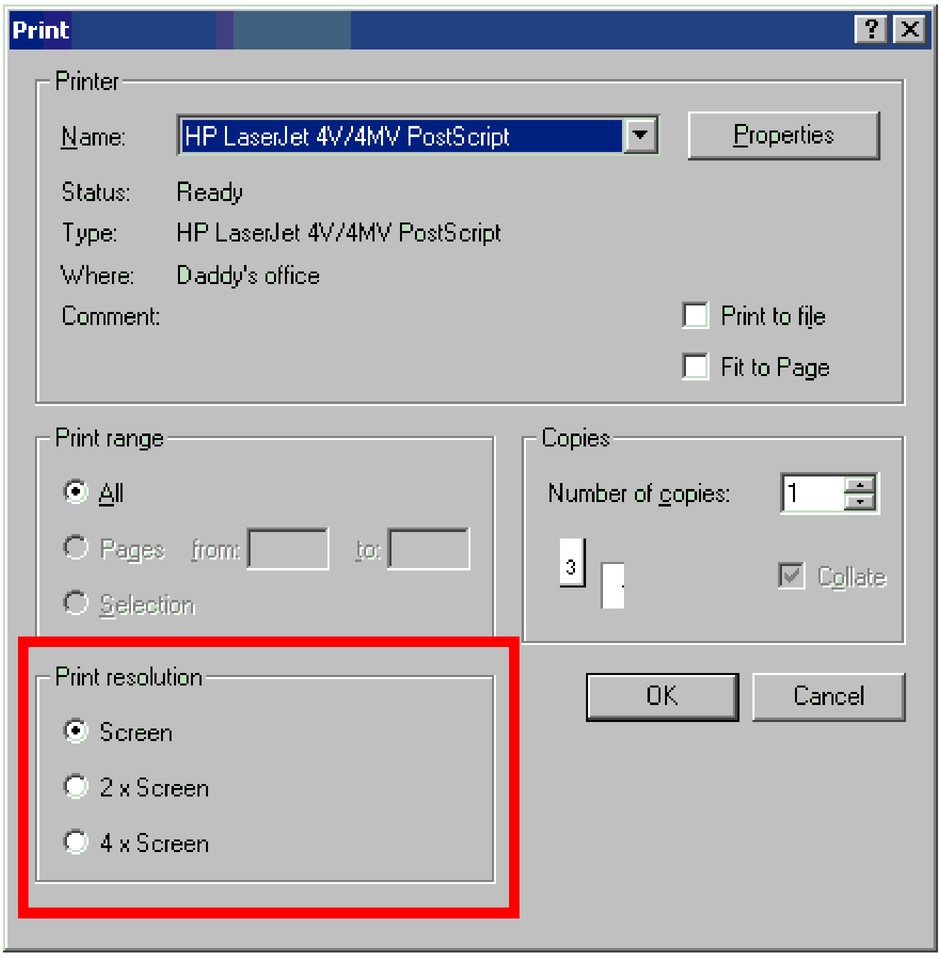


On Windows platforms only the graphics window image may be sent directly to a printer using **FILE > Print...** and **FILE > Page Setup...**

These map the standard Windows printer panels, with the addition of an optional print resolution button (see below).

(***Unix and Linux users*** will need to save a .bmp, .jpg or Postscript file and queue it externally for printing.)

Print Resolution



By default "Print" output will be at the screen resolution, but you can choose the options of 2x and 4x screen resolution. Higher resolution output may be preferable if printing to larger paper sizes.

Note that higher resolution files will be much larger (4x and 16x respectively) and may take correspondingly longer to processes and print.

8.6 Read background image and watermark

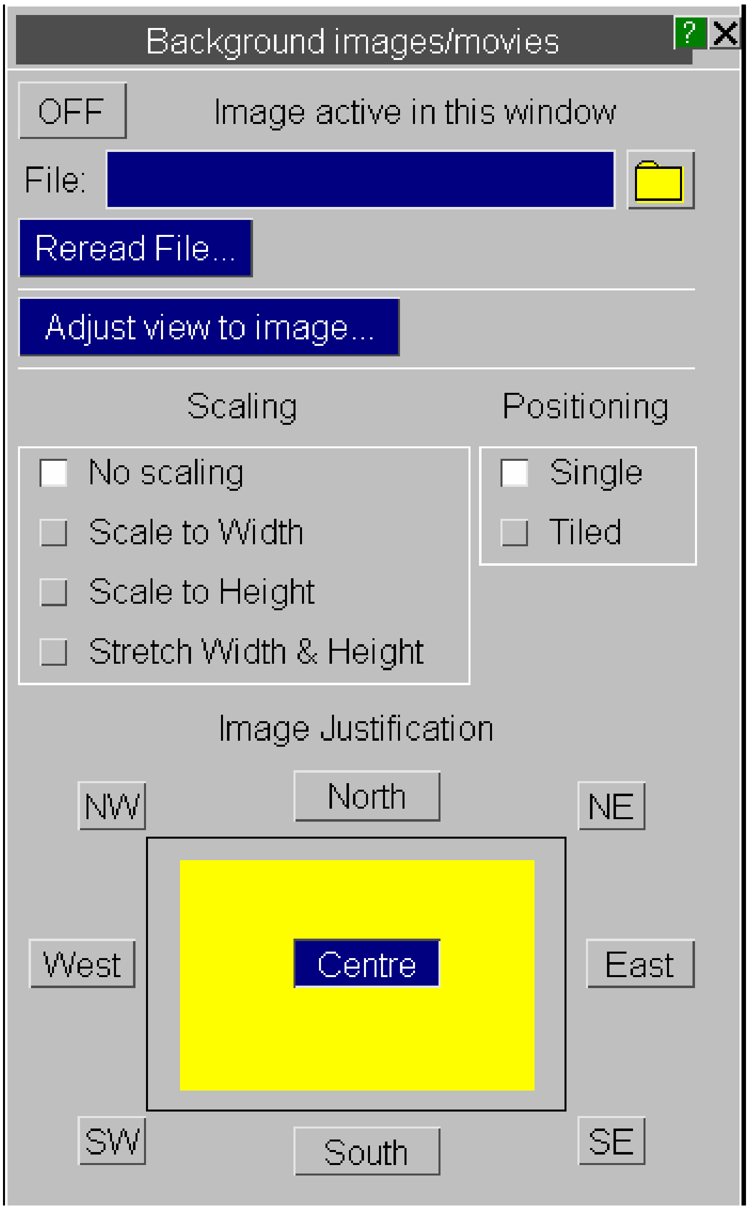
Read an image file to display as a background image behind a model instead of a solid background colour.

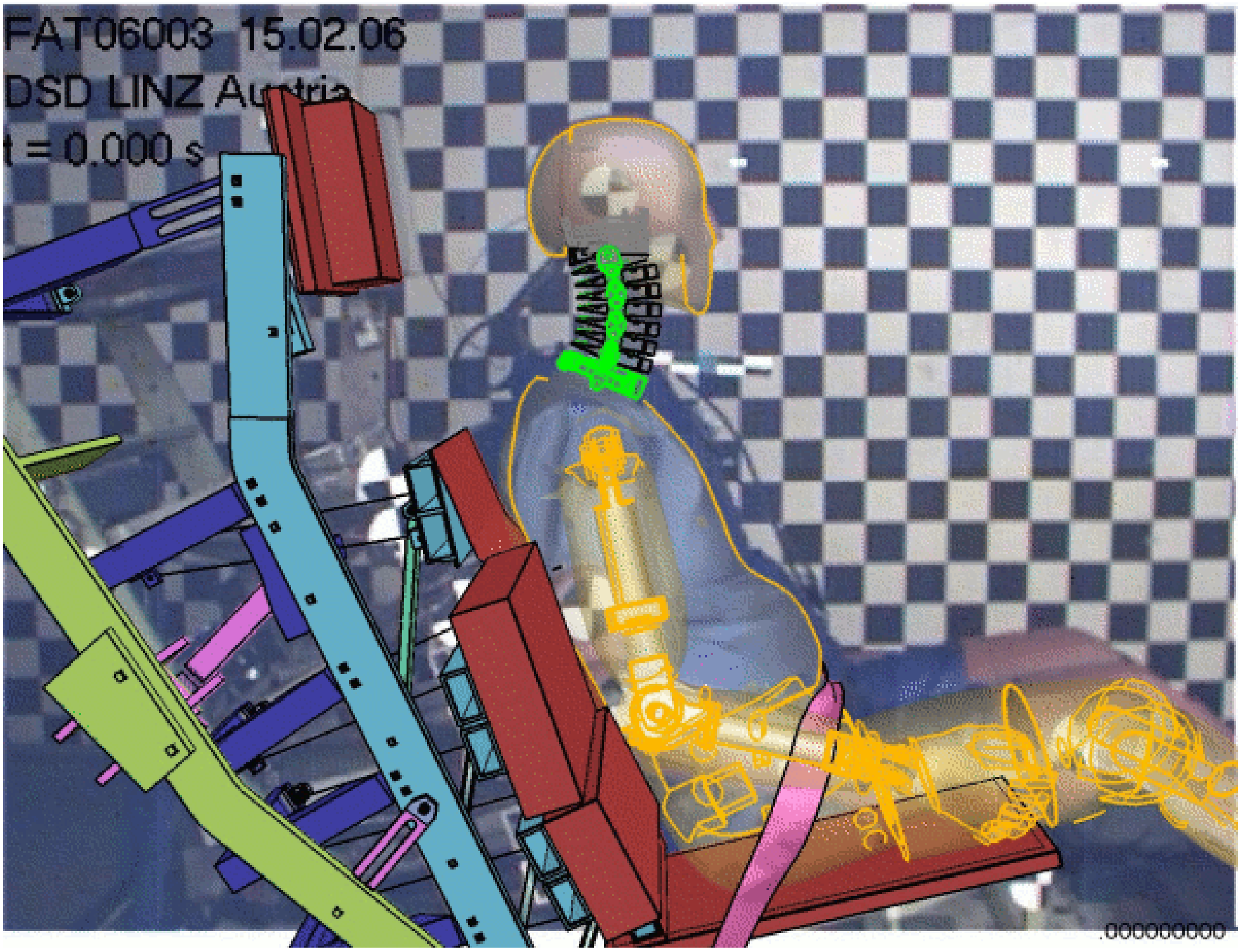
The formats we support are the same as we are able to write, see Capture.

Scaling Options

If the image dimensions do not match the graph window dimensions then the image can be scaled to fit or it can be tiled.

Below is an example background with the model overlayed on top.

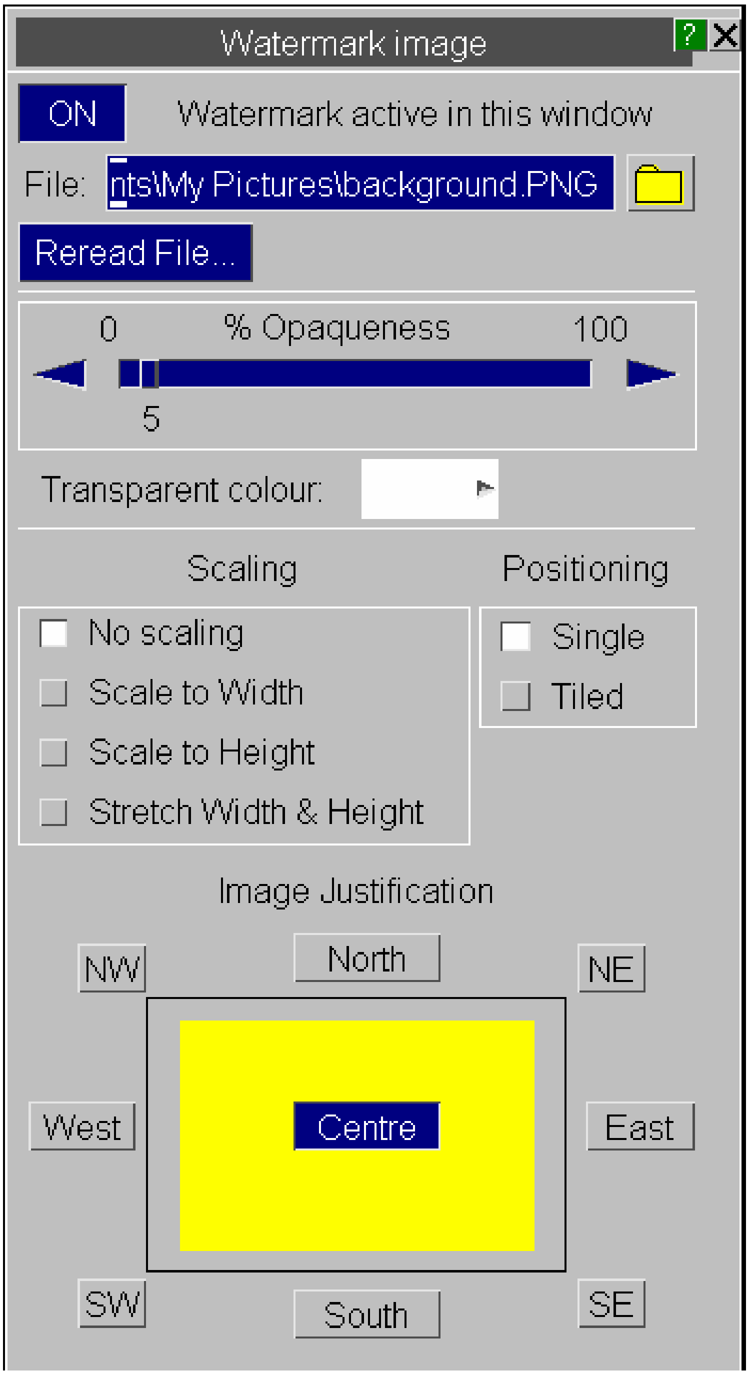


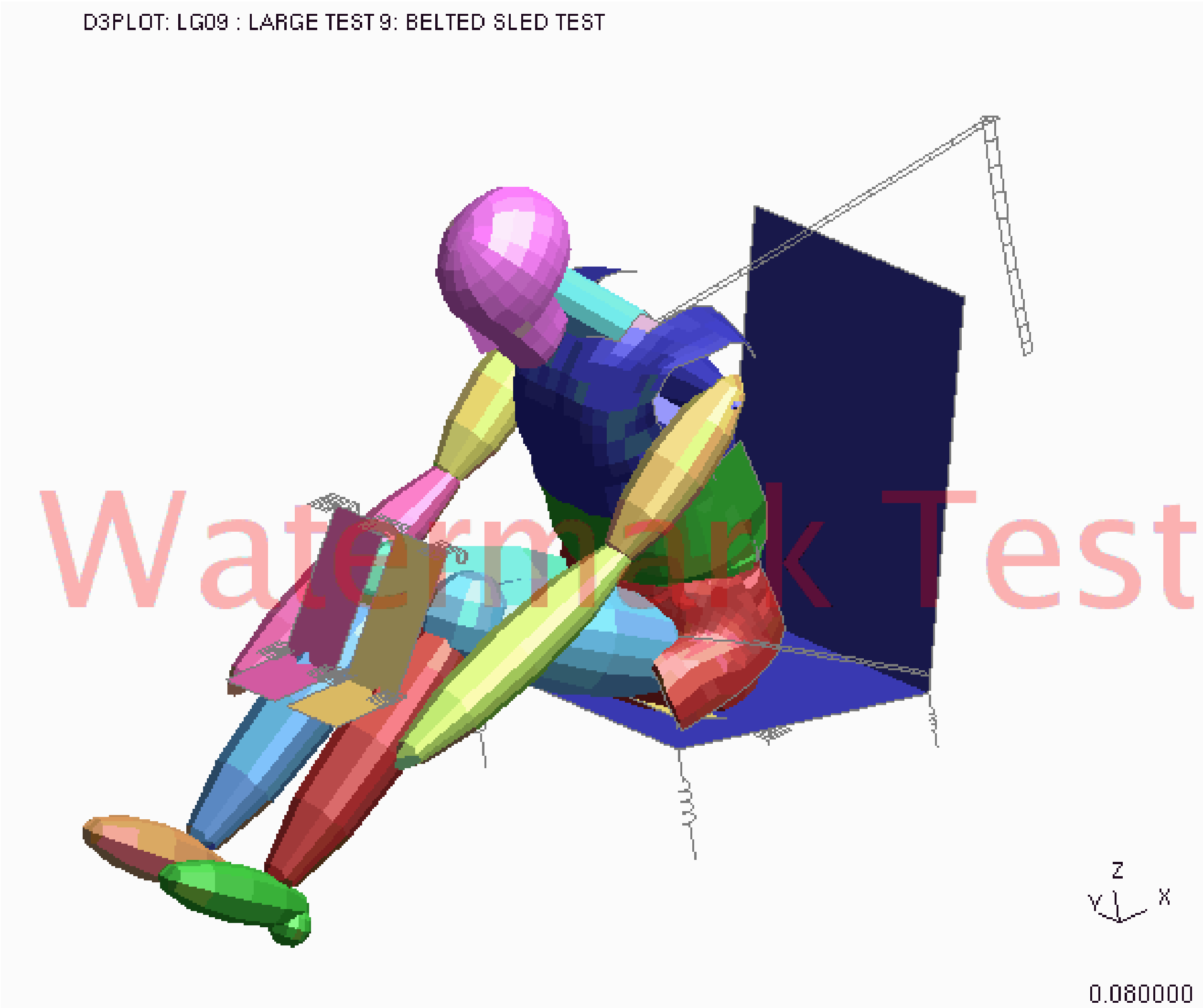

FAT06003 15.02.06
DSD LINZ Austria
t = 0.000 s
.000000000

It is possible to add a "watermark" to a plot. Simply load in an image file in the watermark panel and set its transparent colour and overall transparency.

It will be drawn in front of the normal image, using the transparency settings you have defined. The position and size can also be set.

Below is an example with black as the transparent colour (the image was created on a black background) with 20% opaqueness.

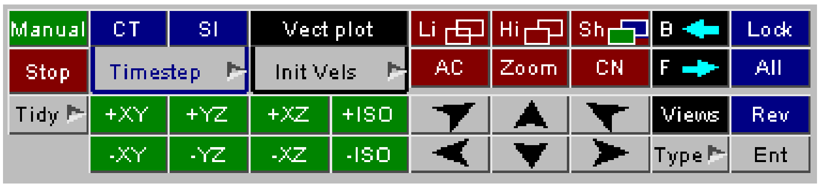



D3PLOT: LG09 : LARGE TEST 9: BELTED SLED TEST
Z
Y
X
0.080000

9 Viewing Controls

9.0 VIEWING CONTROL

"Viewing" refers to the manipulation and presentation of images, rather than their actual generation. All basic commands live in the "Viewing and Drawing Control" box, located at the bottom right hand corner of the screen. The remaining commands can be found in the **Viewing** option in the main menu.

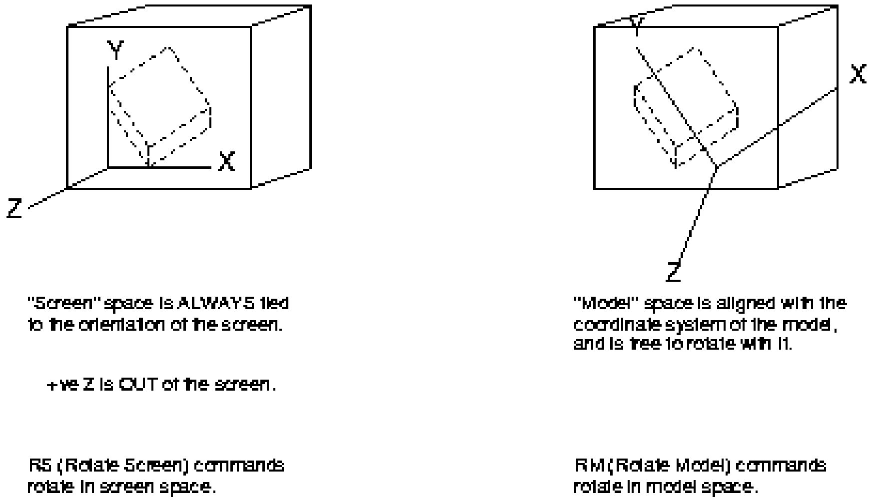


Drawing and Viewing box.

Viewing Menu

9.1 PRIMER coordinate space systems and view layout.

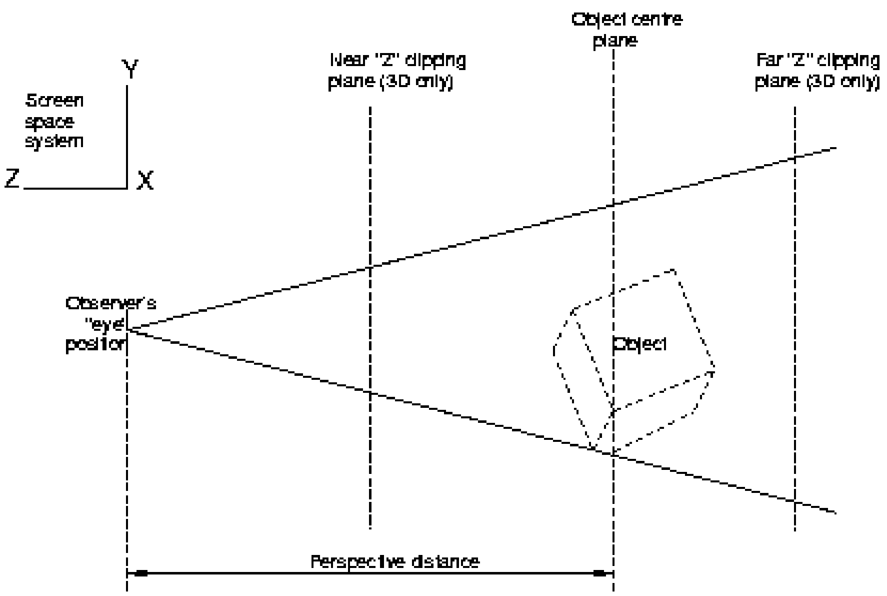
PRIMER uses two right-handed coordinate space systems for viewing: "screen" space, and "model" space. This is illustrated in figure 9.1(a):



Initially the two space systems are coincident, ie the initial view on the model is a plan on XY looking down the Z axis. Transformations to the current view can be applied in either space system, with the result that the model coordinate system will rotate with respect to the (fixed) screen system.

The current model orientation, (ie the axis system in the right hand side of figure 9.1(a) above), is shown by the "triad" in the bottom right of each plot.

The object exists at a point in screen space, and is seen through a viewing "frustrum" as shown in figure 9.1(b) below. The observer's (your) eye point is located at the vertex of a rectangular section frustrum, with the object some distance away in the -ve Z screen space system. The screen image is a 2D projection of what the eye sees: the sides of the frustrum clip the view to the left/right and bottom/top edges of the screen.

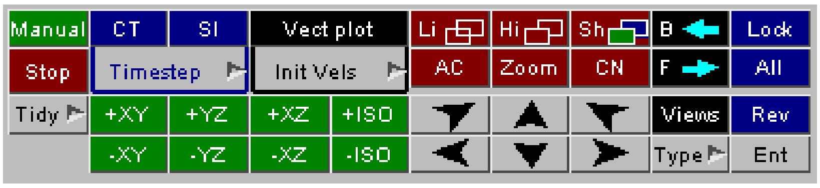


It is important to note the following:
1. Rotation of the image always takes place about the point that is the XY centre of the screen (in screen space coordinates), at the "object centre plane" (which gives the screen Z location). Thus in the example above roughly at the "O" of "Object".
2. The example above implies a perspective projection. In fact PRIMER defaults to a parallel (orthographic) projection, in which the sides of the frustrum are parallel and perspective is irrelevant. Nevertheless the object

centre plane is still significant, since this still gives the screen Z coordinate about which rotations take place. You can turn perspective on/off and alter its distance at will.

3. You can change the scale ("zoom" in/out) of your image. This effectively changes the angle of view of the frustrum above: zooming in makes it narrower, zooming out wider. But note that this does not change your distance from the object: changing the scale is like putting a lens of a different focal length on your camera, to change your distance from the object you must alter the perspective distance.
4. The near and far "Z" clipping planes shown here only apply on 3D hardware that is capable of this. See section 9.6.

9.2 The Viewing Control box

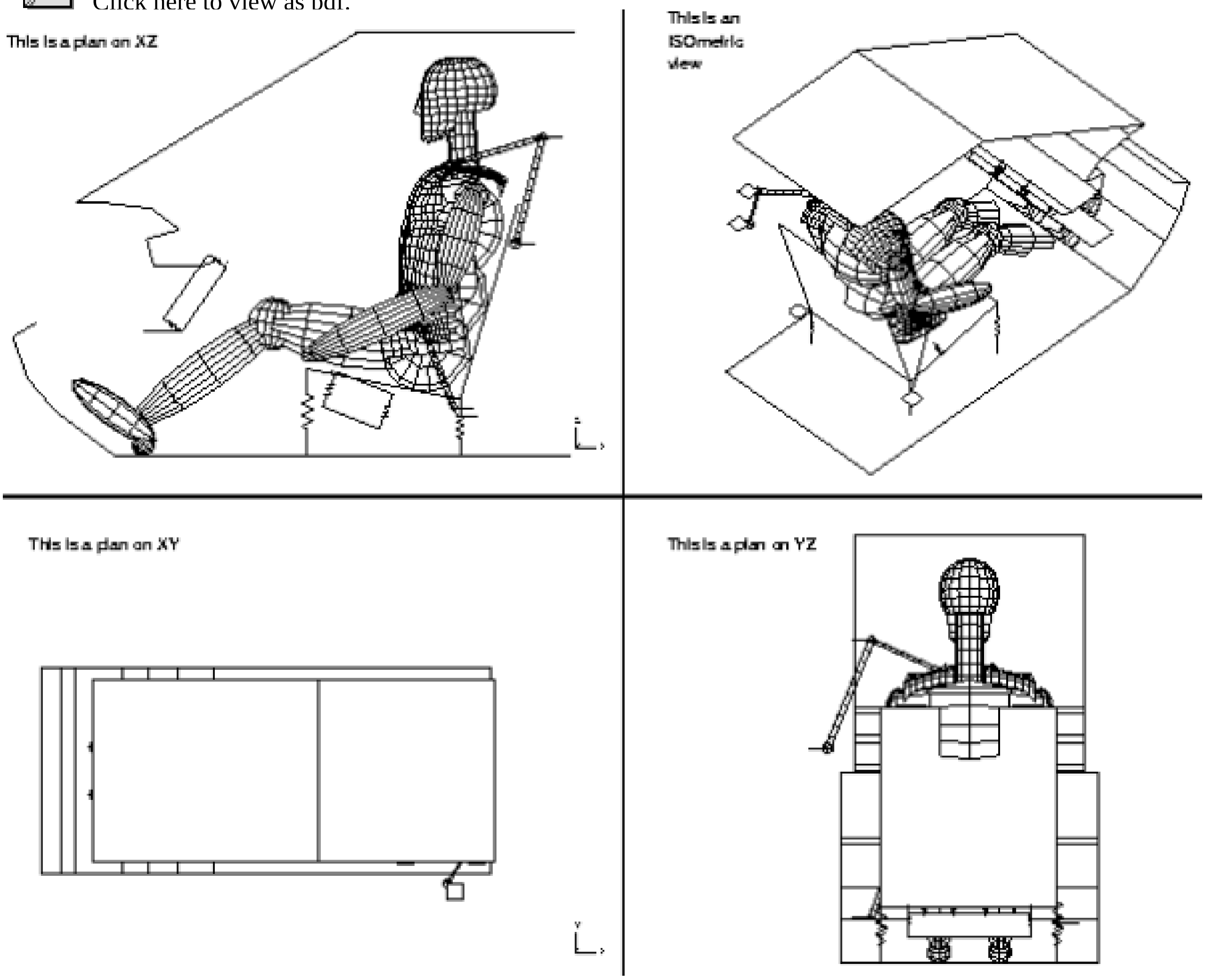


All aspects of viewing are controlled from within the "Viewing and Drawing Control" box. Its layout is shown on the right.

9.2.1 Pre-programmed views.

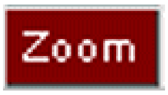
+/-XY Is a plan on model XY, looking down/up Z.
+/-XZ Is a plan on model XZ, looking down/up Y,
+/-YZ Is a plan on model YZ, looking down/up X,
+/-ISO Is an isometric view (equal angles for X,Y,Z).
These views are also available from shortcut keys 1,2,3 etc.

Click here to view as pdf.



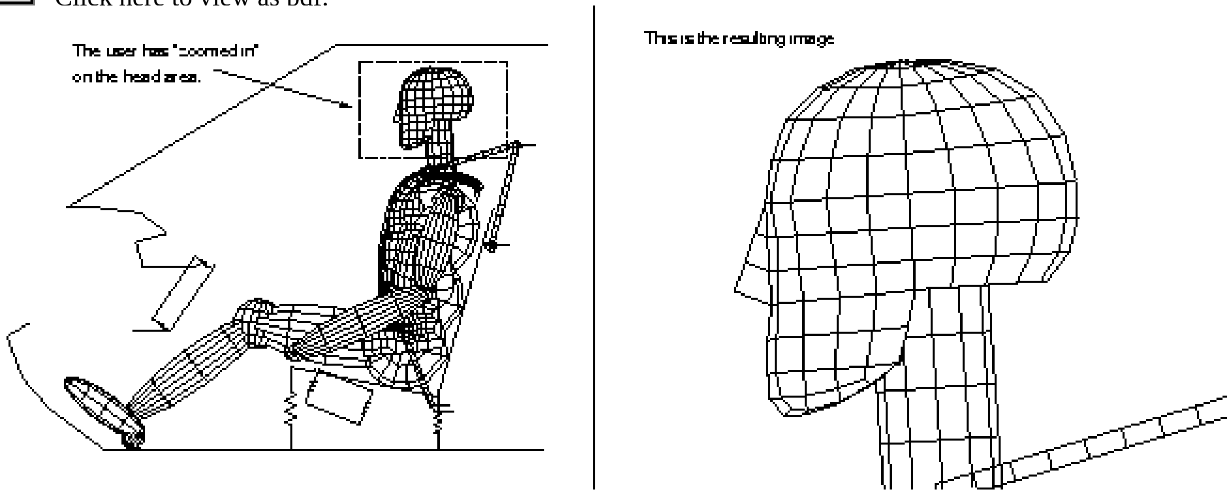
These views only apply rotations, they do not affect scale or position.

9.2.2 "Zooming in" (magnifying an area).



ZMZooms in by using the cursor to pick a rectangular screen area that is to be enlarged to fill the screen (also available from shortcut key Z).

Click here to view as pdf.



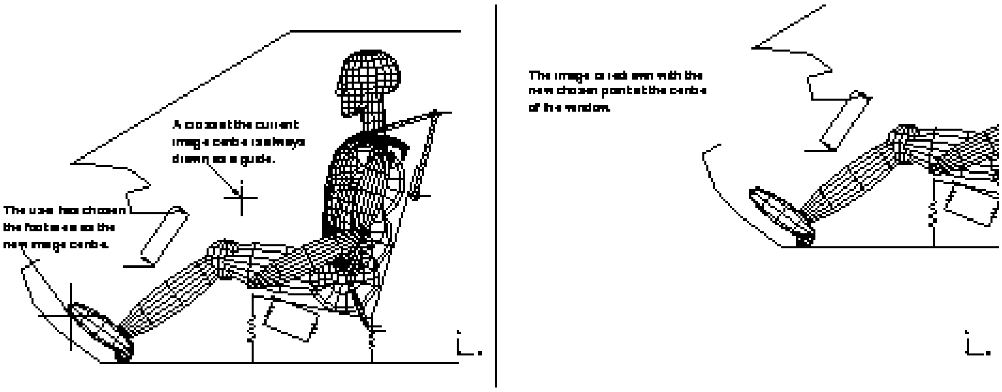
Both the scale and the current image centre are changed.

When defining the rectangle with the cursor you can "rubber-band" the box by picking its first point, then pressing the button and moving to see the effect of the second point.

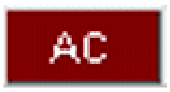
9.2.3 "CeNtre" (Centre Node).

CN Lets you choose a new node that will become the new image centre and about which rotations will be performed. The point chosen with the cursor moves to the window centre, but the scale is not changed.

Click here to view as pdf.



9.2.4 AC (Autoscale Current) Scaling the view to fit the screen



ACCalculates the correct scale and centre position required to make the current image fit neatly onto the screen (also available as shortcut key A). This takes account of blanking, clipping, deformations, etc.

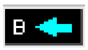
9.2.5. Zooming using +/-

Shortcut keys + and - magnify/reduce the current view.

9.2.6 Zooming using mouse wheel

The current view can be magnified by moving the mouse wheel down (towards the user). The view can be reduced by moving the mouse wheel up.

9.2.7 Scrolling through previous views

The 

and 

buttons allow scrolling through previous views.

9.2.8 Blanking control buttons

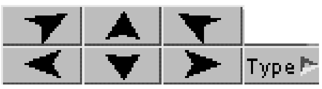
Rev reverses all blanking (or use shortcut key R). **Lock** stores the current blanking status. **All** restores the last stored blanking status.

9.2.9 Other buttons on the Viewing Control panel

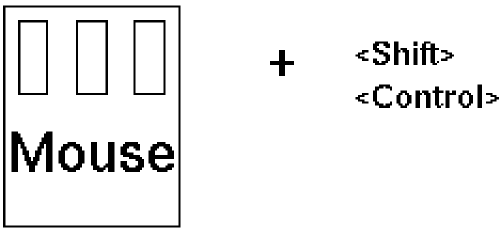
Manual invokes the on-line manual. **Stop** interrupts the current operation. Click here to see a description of **Tidy**.

9.2.10 Alternatives to the commands in this section.

The commands listed here are useful for making explicit changes to the view of your model, for example to set it to a view exactly co-incident with axis orientation or locate its centre at a specific position. However there are two other ways of making these changes which, in some contexts, may be better:



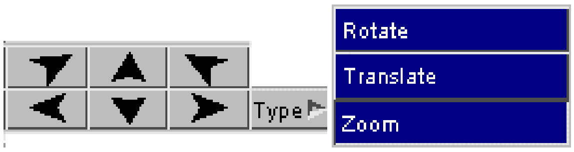
Compass Rose



The "Compass Rose" is described in section 9.3, and dynamic viewing in 9.4.

9.3 Using the "Compass Rose"

The "Compass Rose" provides three sets of buttons that allow the model to be rotated, translated and scaled with single mouse clicks.



9.3.0 General information on using the Compass Rose

The Compass Rose operates in one of three modes selected by the **Rotate**, **Translate** and **Zoom** buttons available from the **Type** pop-up menu as shown in the example above (**Rotation** mode is selected). The options provide:

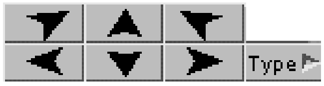
Rotate Arrow buttons provide pre-defined increments of rotation about each of the X,Y,Z axes in screen or model space.

Translate Arrow buttons provide pre-defined increments of translation in each of the X,Y,Z axes in screen or model space.

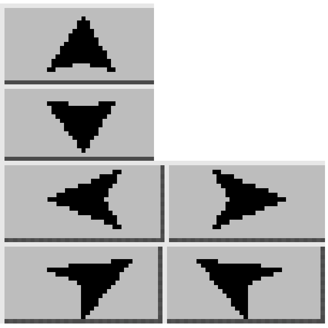
Zoom Buttons provide pre-defined increments of magnification and reduction of viewing scale.

Also, a single click on a function button will deliver a single instance of that function; but holding the button down will, after an initial delay (see Settings), cause it to repeat its function at a pre-defined rate. If command-file recording is on each such repeat is recorded as a separate button click: this makes it possible to record and replay sequences of view changes.

9.3.1 Rotation functions.



The arrow buttons have the following meanings:



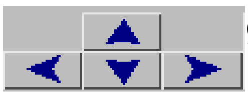
Rotates about *x* axis

Rotates about *y* axis

Rotates about *z* axis

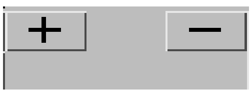
Each click generates an increment of rotation about the relevant axis or, if held down, continuous rotation.

The sign of the rotation is intuitive for system rotations, for example => gives clockwise (+ve) rotation.



9.3.2 Translation functions.

The Translation functions are very similar to the rotation ones: translation defaults to screen space, and uses a similar system of arrow buttons.



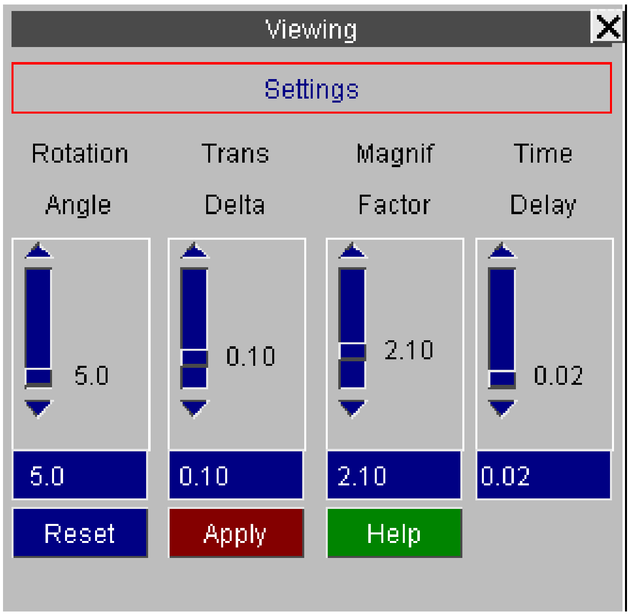
9.3.3 Magnification functions

The Magnification function is very simple:

+ Increases the size of the image by the pre-defined increment;
-- Decreases it by the same increment.
The image is scaled about the current window centre.

9.3.4 Setting attributes

In each case the the option **Settings** in the **Viewing** menu gives access to further options which allow you to modify the pre-defined settings (angular increments, time delays, etc).



The attributes that can be set are:

Rotation Angle Angular increment in degrees. The default is 5 deg, but any value 0.0 < value < 180.0 is valid. The **ANGLE** increment of five degrees is a reasonable value for single clicks, but is really too large to give the impression of smooth rotation under continuous motion: you will probably find that a value of 1 or 2 degrees is better for that.

Trans Delta The translation increment as a fraction of screen span. Default is 0.1 (ie 10%), but any value 0.0 < value < 0.5 is valid.

Magnif Factor The magnification factor. Default is 1.1 (ie 10%), but any value 1.0 < value < 5.0 is valid.

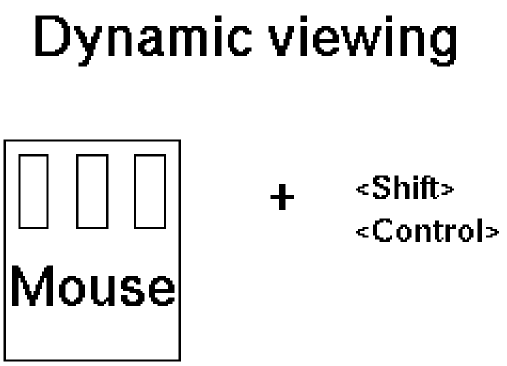
Time Delay Is the time delay (in seconds) between continuous transformations when a button is held down. The default is 0.02s, but values 0.0 <= value < 0.5 are valid. The **Time Delay** is the minimum permitted time delay between frames. If the hardware is taking longer than this to render each frame it does not add to the delay, it simply pads it out if the inter-frame time is shorter than this interval.

You may be tempted to cut the time delay between transformations down to zero, and on very fast hardware (typically 3D machines) this gives good results, but on slower hardware you may find that this gives uneven results as competing system demands lead to variable elapsed intervals between frames. It is for this reason that the delay of 0.02 seconds is the default: it barely slows transformation (50 frames per second is twice as fast as a TV set scans!), but it does even out the delay time between frames giving a smoother result - especially under X-Windows. Experiment on your hardware.

9.4 Dynamic Viewing (Using the mouse to change views).

"Dynamic" viewing is the name given to the process in which you perform viewing transformations by moving the mouse around the screen. This is the final, and probably most useful, way of controlling views described in this section.

The three classes of transformation available in the compass rose (rotation, translation and scaling) are all available in dynamic viewing as well. However rotation and translation are only available in "screen" space: it would be too confusing to try to relate cursor movement to "model" space transformations.



9.4.0 Graphics modes during dynamic viewing

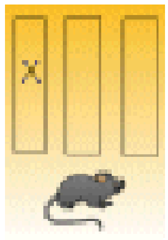
All dynamic viewing operations require a combination of two screen "meta" keys, (**<left control> and <left shift>**), and mouse buttons. The meta key(s) used dictates the graphics mode in which the image is transformed as follows:

<left shift> + <mouse>	Transforms the image in the current graphics mode. For example if it is a hidden-line plot, then dynamic viewing will take place in hidden-line mode.
<left control> + <mouse>	Transforms the image in "wire-frame" mode for the duration of the drawing operation. (i.e. no hidden-surface removal or lighting.)
<Left shift> &}<left control>} + <mouse>	Transforms the image in pre-computed free-edge mode for the duration of the drawing operation. (i.e. wire-frame of free edges only, no hidden-surface removal or lighting.)

In the latter two cases the original drawing mode is always returned to at the end of the dynamic viewing operation. The wire-frame and free edge modes are provided to make transformations quicker for very large models and/or slow computers: free edge is very fast.

For the last case, with **<left shift>** & **<left control>** held down together, the order of pressing and releasing the meta-keys matters: press **<left shift>** before **<left control>,** and release in the opposite order, otherwise you will (correctly) get the image redrawn in wire-frame mode as the **<left control>** key is pressed and released.

9.4.1 Dynamic Rotation.



Dynamic rotation uses **<left mouse>** + **<left shift>** &/or **<left control>**

(The distinction between the keyboard meta-keys is explained in section 9.4.0 above.)

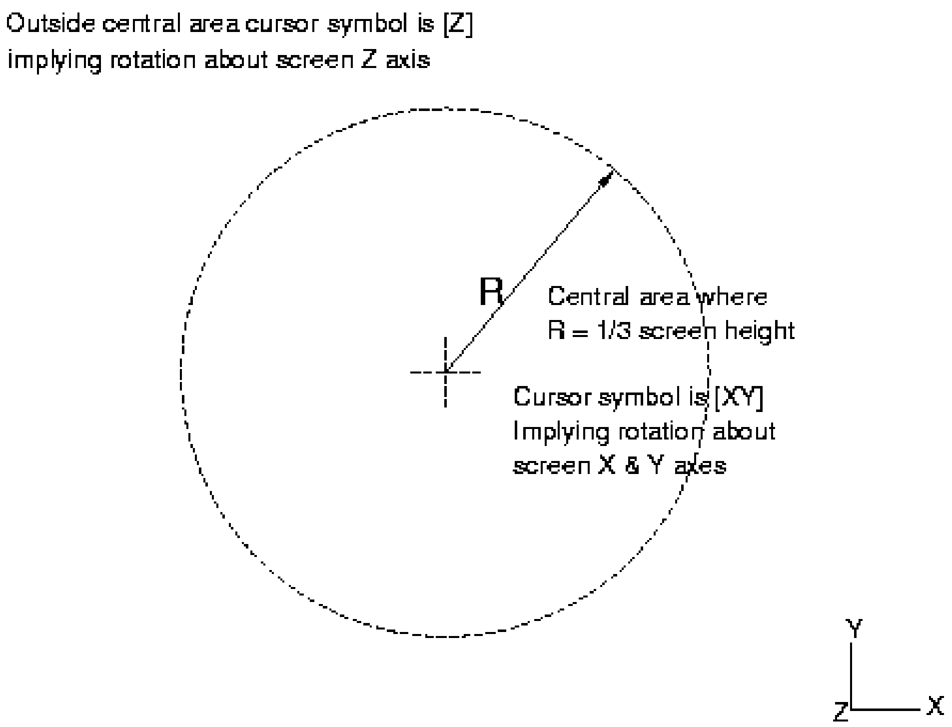
Rotation always take place in the screen coordinate system, and may be about the XY axes or Z: this depends upon the starting position of the mouse. This is shown in figure 9.4.1:

If the mouse initial position is *inside* the central circle (radius (screen height/3)) then rotation is about screen XY axes.

If the initial position is outside this circle then rotation will be about screen Z.

You can tell which mode you are in by the cursor symbol. This red, and:

XY rotation uses `[XY]`
Z rotation uses `[Z]`

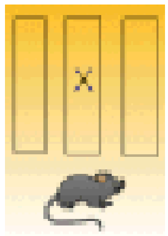


The relationship between mouse and image motion is intuitive in both modes. It is as if you had grabbed a point on the object near you, (this side of the object centre plane), and used this to move the image about its centre:

XY mode Moving the mouse left/right rotates about the screen Y axis;

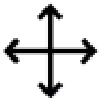
Moving the mouse up/down rotates about the screen X axis.
Z mode Moving the mouse in a circular direction rotates about the screen Z axis.
Rotation remains locked in its initial XY or Z mode for the duration of a dynamic viewing operation, regardless of where you subsequently move the cursor to, until you release a mouse or keyboard button.

9.4.2 Dynamic Translation.



Dynamic translation uses **<mid mouse>** + **<left shift>** &/or **<left control>**

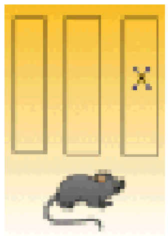
(The distinction between the keyboard meta-keys is explained in section 9.4.0 above.)

The cursor symbol is yellow, and looks like: 

Translation always take place in the screen coordinate system, in the X and Y directions.

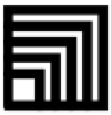
The relationship between mouse and image motion is intuitive: the object tracks the mouse motion in the screen XY plane. The initial position of the mouse is irrelevant.

9.4.3 Dynamic Magnification (Scaling).



Dynamic scaling uses **<right mouse>** + **<left shift>** &/or **<left control>**

(The distinction between the keyboard meta-keys is explained in section 9.4.0 above.)

The cursor symbol is green, and looks like: 

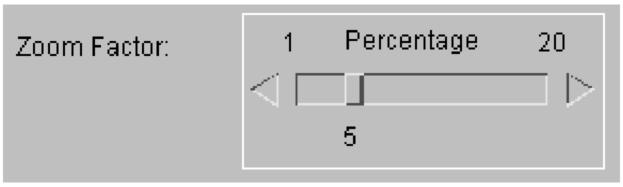
Mouse motion to the right and up makes the image larger, left and down smaller. The initial position of the mouse is irrelevant.

Dynamic magnification using the mouse scroll-wheel

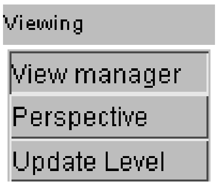
If your mouse is equipped with a scroll-wheel then it will also perform dynamic magnification in the graphics window in which the cursor is present.

- Magnification is centred at the current cursor position unless "**CN** Centre node" has been used to lock centring on a node.
- Scrolling towards you magnifies the image scale.
- Scrolling away from you reduces the image scale.

By default each scroll wheel "click" will change the magnification factor by +/- 5%, but this can be changed using the **Options > Menu attributes** panel, and altering the **Zoom factor.**



9.5 Further commands in the Viewing Menu



The following commands with sub-menus in this box are described:

View manager... Storing and retrieving views (this can also be accessed from the button **Views** in the Drawing and Viewing Box).

Perspective... Controlling perspective, also "locate target and eye"

Update Level... Setting plot update frequency

9.5.1 VIEWS... Storing and retrieving "view" information.

What is a view?

A "view" is all the information required to set up the current view of the object. In practice this means:

- The current rotation matrix (3 direction cosines).

- The current image centre location in space (x,y,z coordinate).

- The current magnification scale.

- The current perspective distance.

Up to 100 such views may be stored and retrieved at will from a file, and any number of such files may exist. A view is given a name and number when it is stored, and these are used when retrieving it

Up to and including release 9.3: Views are stored parametrically.

What this means is that views are not tied to a particular model, they will work for any model of similar dimensions. So if you are working on a set of variants of an analysis you can share the views on file between them: this is why they are stored in a separate, model-independent file. It is only when the shape and/or size of a model differs wildly from the original from which the view was created that this shareability fails.

From release 9.3.1 onwards: Views are stored explicitly

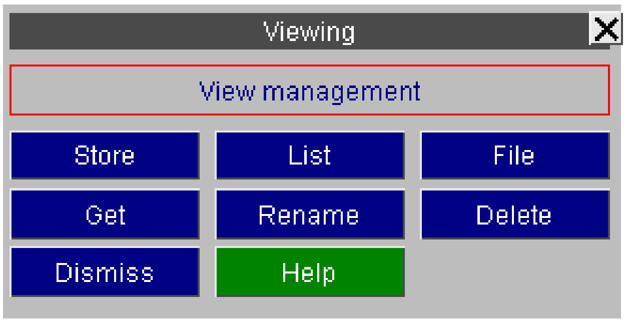
The parametric method described above was not a success, as users wishing to compare models visually found it misleading. Therefore from release 9.3.1 onwards views are now stored explicitly, and no account is taken of model size or position. Put qualitatively: the camera now stays in the same place with the same settings.

Retrieval of views is backwards-compatible. A view stored prior to V9.3.1 will read successfully into V9.3.1 onwards, but will be converted to "explicit" format if subsequently saved.

Using views

PRIMER always has a current "view" definition. This dictates how the image will appear when a drawing command is issued. You can save the current view to file at any time. Likewise you can retrieve a stored view to replace the current one at any time.
The current view only exists in memory, and changing it has no influence on any views stored on file. (Indeed you don't need to have a stored view file: the default is none.)

Managing views

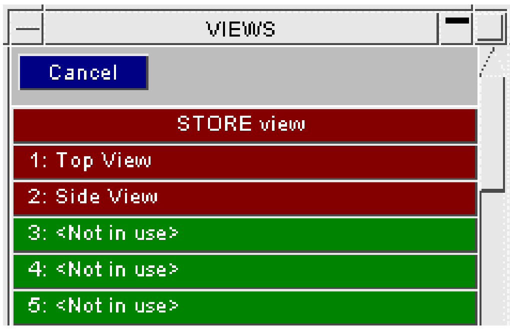


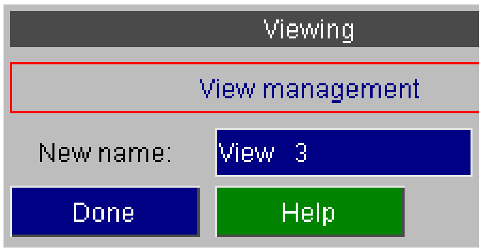
When you press the **Views**... button you get the View Management panel shown in figure 9.5.1(a).

The controls are described below.

9.5.1.1 **STORE** Storing the current view on file.

You are presented with the **STORE** view menu showing views 1 to 100, and you must choose which one this view is to be stored as.



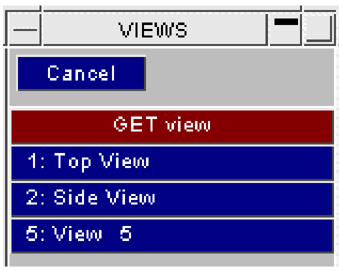


Views currently in use are red, with their current names shown; unused views are green, and marked **<Not in use>**.

Once you have defined the view number, then give a name. A default name of **"View <n>"** is provided, but any name is valid.

Up to 100 views can be stored in a file, and views can be overwritten at will. If no explicit file has been opened the default file plot.view is opened automatically and used.

9.5.1.2 **GET** Retrieving a view from file.



You can only retrieve from file views that already exist.

You are presented with the **GET** view menu of stored views, and must pick one. In this example three views are available.The attributes of the stored view are converted from parametric form to your model’s coordinate system, and then become the current view.

If the `UPDATE_LEVEL` is 2 or above (see section 9.5.3) then the view takes effect immediately, otherwise it becomes effective the next time you issue a drawing command.

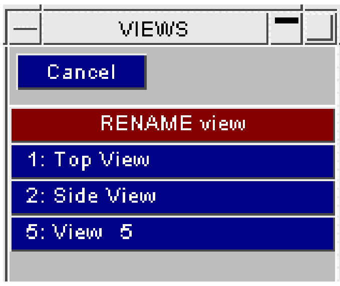
9.5.1.3 **RENAME** Renaming stored views.

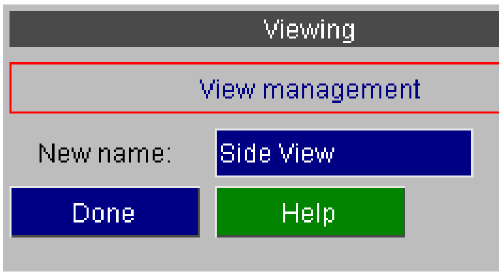
You can rename any stored view.

Select a view from the **RENAME view** menu, then give it a new name.

Any (or no) name is acceptable. This is simply a label by which the view is known.

Here the user has chosen to rename view #2, currently called "Side view".

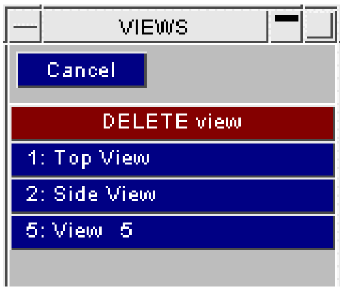




9.5.1.4 **DELETE** Deleting stored views.

You can only delete existing views. Select a view from the **DELETE** view menu, and it will be deleted.

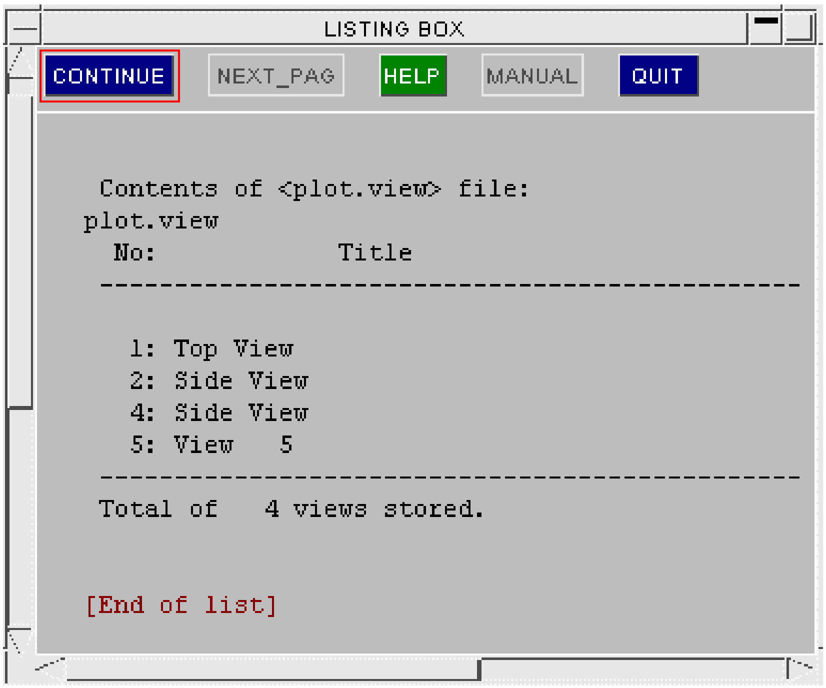
There is no warning or confirmation dialogue, so make sure that you want to do this! **CANCEL** can be used to abort the operation.



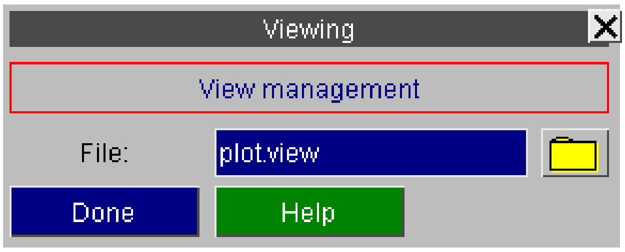
9.5.1.5 **LIST** Listing stored views

You can list information about stored views to screen with the **LIST** option.

If you have a lot of views this is a better way of listing them than trying to use the menus in a confined space.



9.5.1.6 **FILE** Selecting/Defining a View Storage filename

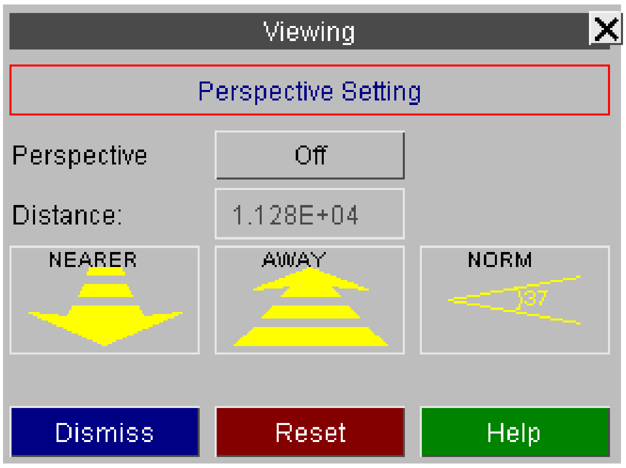


By default views are stored in a file called plot.view, and generically the view storage file is referred to as the **`<plot.view>`** file.

However you may choose any filename, and you may have any number of view storage files. To open a new or existing **`<plot.view>`** file use the **FILE** command, and enter a new filename.

To use the file filter click the button to the right of the field.

9.5.2 **PERSP**... Setting Perspective Attributes.



By default perspective in PRIMER is off.

Figure 9.5.2(a) shows the perspective control panel it this default state. Note that the distance changing options are greyed out as a consequence of perspective being turned off.

Figures 9.5.2(b) and (c) show the effect of turning perspective OFF and ON for a rectangular box. In the left image, where perspective is off, the image is foreshortened and looks strange; in the right image, with it on, the box looks more normal.

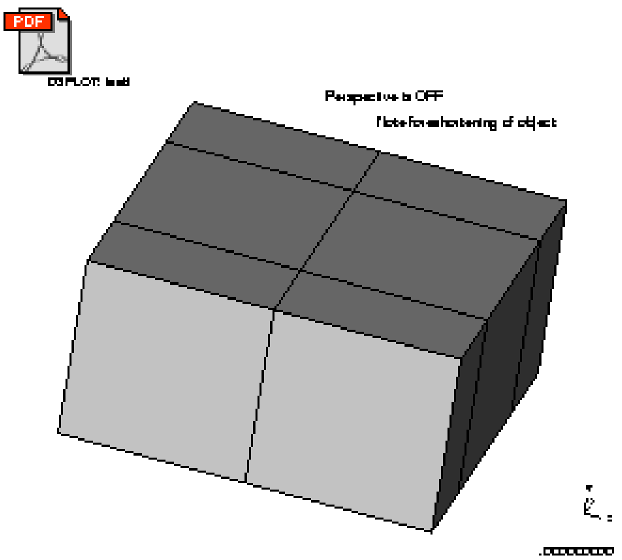


Figure 9.5.2(b): Perspective OFF

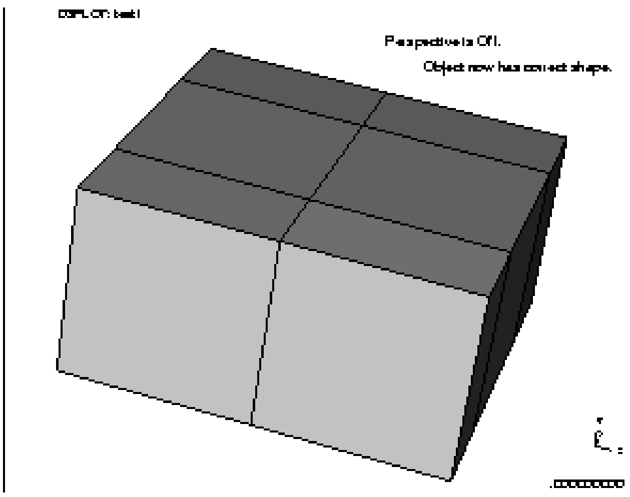
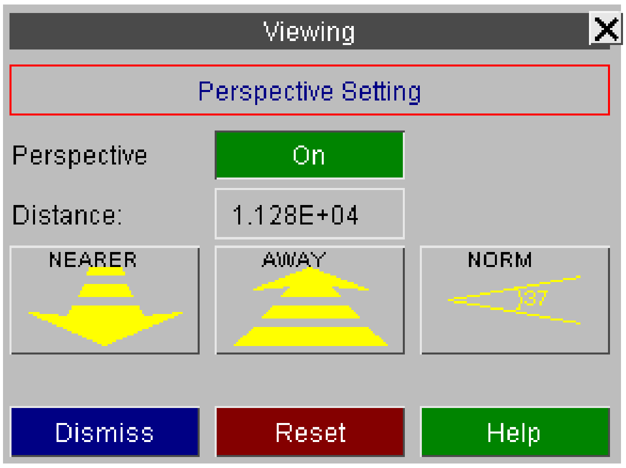
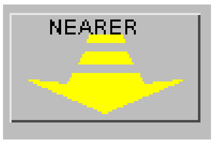


Figure 9.5.2(c): Perspective ON

Figure 9.5.2(d) shows the control panel when perspective is turned **ON.**

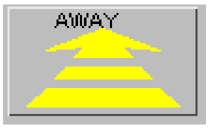
Note that the distance changing options are now live. These are used as follows.





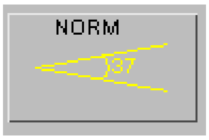
NEARER Reducing perspective distance.

Clicking on this button reduces the perspective distance by 5%. This brings the viewer closer to the object.



AWAY Increasing perspective distance.

Clicking on this button increases the perspective distance by 5%. The takes the viewer further away from the object.



NORM Restore perspective distance to its **NORM**al value

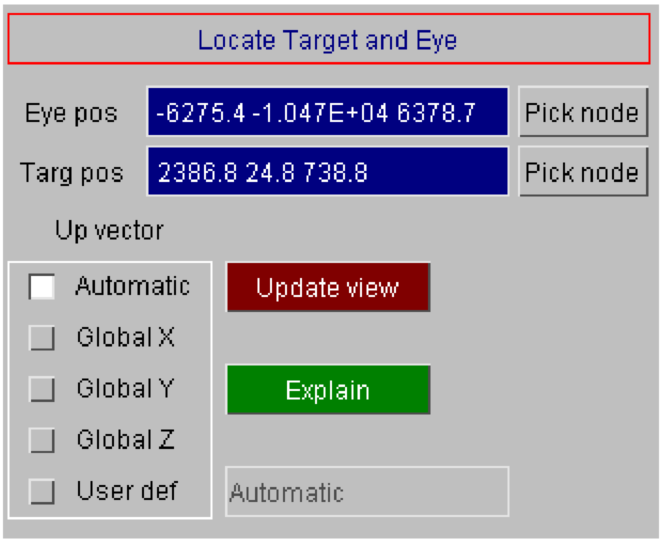
This button restores the perspective distance to its standard setting, which gives a field of view of about 37 degrees.

For all three buttons above the effect is immediate if the **UPDATE_LEVEL** (see 9.5.3) is 2 or greater. In addition holding down a button gives a repeated action after an initial delay, so that you can, in effect, see the effect dynamically as you change the distance.

Setting the distance explicitly.

You can type in an explicit Distance from the observer.

9.5.2.1 Locate Target and Eye

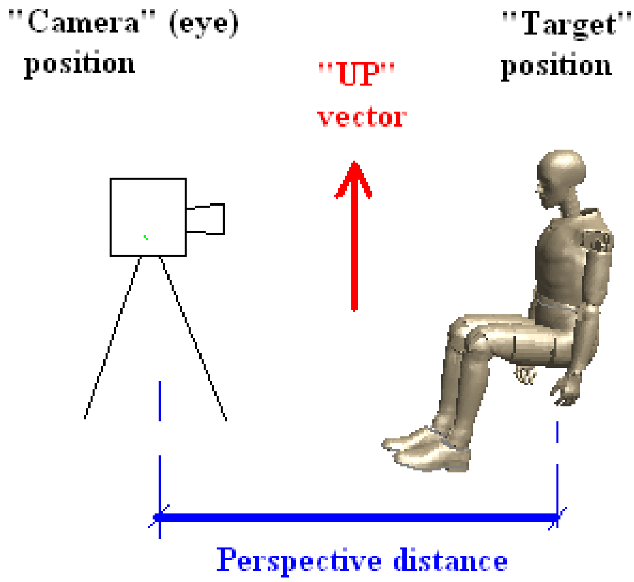


Normal PRIMER viewing effectively positions the model in front of a stationary camera, then rotates, pans and enlarges it to place the desired region in the field of view of the lens.

However it is possible to set the "eye" (camera) position and also the "target" point on the structure at which the camera is pointing, and PRIMER will compute the viewing transformation required to give the image from this point.

There are three components in a "Locate target and eye" definition:

Target position	This is the coordinate in space at which the camera is pointing.
Eye position	This is the coordinate in space at which the camera (eye) is located
"Up" vector	This is the vector defining "which way is up". Panning the camera up and down would move it up and down this axis



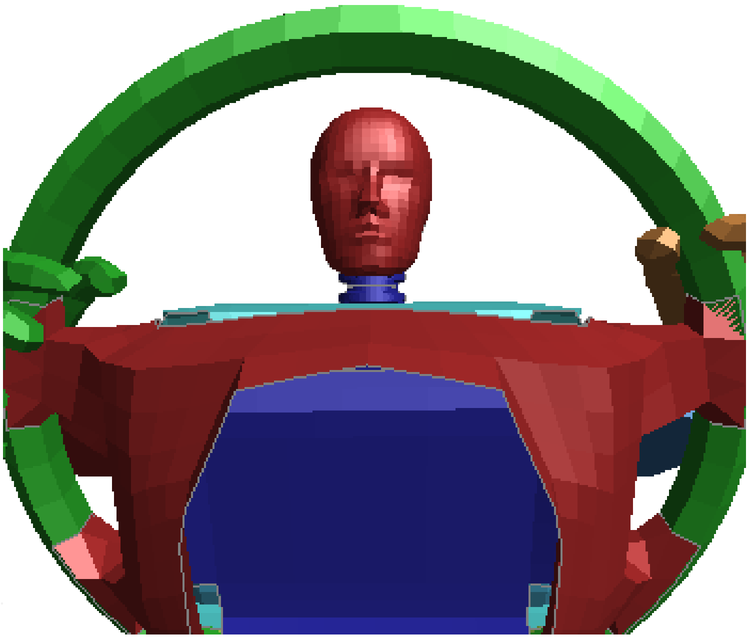
The distance between the camera (eye) and target points is implicitly the current perspective distance, and this is reset when you **Update** the view. Perspective is switched on automatically if this is not already the case.

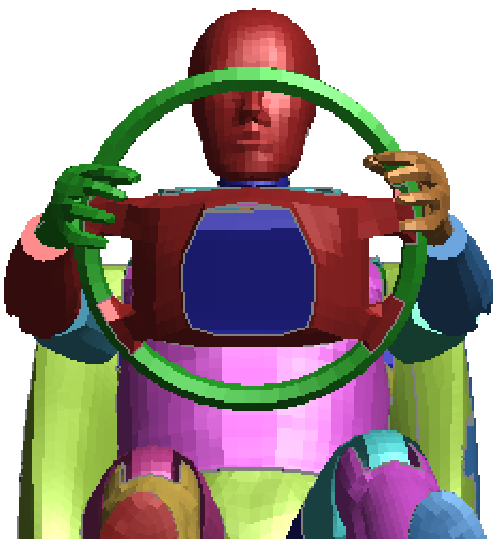
Both target and eye positions may be defined explicitly as coordinates in space, or you may screen-pick a node and its coordinate will be extracted.

By default PRIMER tries to deduce the "Up" vector automatically, but you can override this by choosing a global vector, or by defining your own arbitrary vector.

The relationship between Perspective Distance and Scale.

If you use the "locate target and eye" feature you will almost certainly position your eye fairly close to the structure, which will bring you much closer than the normal perspective distance set by PRIMER which is 3x the diagonal of the bounding box around the model. When the perspective distance becomes small the fore-shortening effect it causes becomes much more obvious





In this image the target point is the dummy's nose, and eye point has been placed on the steering column just behind the wheel.

In this image the target point is the same, but the perspective distance has been increased by a factor of three, effectively moving the eye point backwards out of the paper.

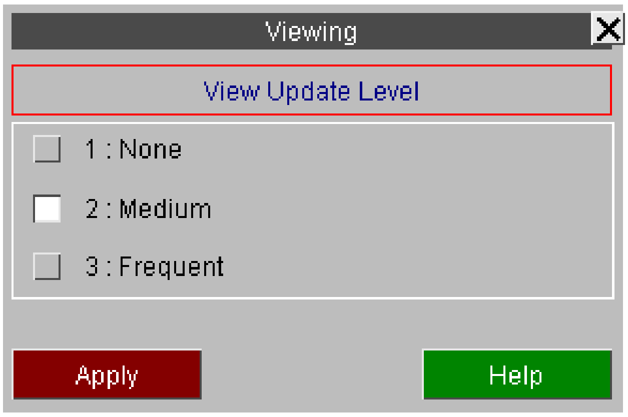
Photographers will recognise that the perspective distance is, quite literally, the distance between subject and camera, whereas the scale is the "zoom power" (or, more precisely, focal length) of the lens on the camera. Both images above show the dummy head at approximately the same ***scale***, but the difference in ***perspective distance*** gives rise to very different images.

If you are attempting to select viewing attributes to match an existing image you may find this quite difficult to achieve by hand since there are 11 independent variables to match in such an operation:

- Camera position (x,y,z coordinate = 3 variables)
- Subject position (ditto = 3 variables)
- "Up" vector (ditto = 3 variables)
- Scale (1 variable)
- Perspective distance (1 variable)

The **Match Image** function below will calculate this for you when given at least four points on the image and structure to match.

9.5.3 UPDATE... Controlling the View updating frequency.



PRIMER has an **UPDATE_LEVEL** setting which dictates how often the view is updated following commands that change it.

Figure 9.5.3 shows the **UPDATE** panel and its three settings. These have the following meanings:

UPDATE_LEVEL = 1 **No updates**

The plot is never updated automatically. Changes only become apparent when you issue an explicit drawing command, eg **DR, CT**, etc.

UPDATE_LEVEL = 2 **Medium updates**

The plot is updated immediately when any view control command is given.

The current image is amended as necessary following blanking, clipping, etc if any viewing command, including dynamic viewing, is used. In other words a viewing change command is tantamount to an explicit redraw command in the current mode which would, of course, reflect any changes in the model geometry.

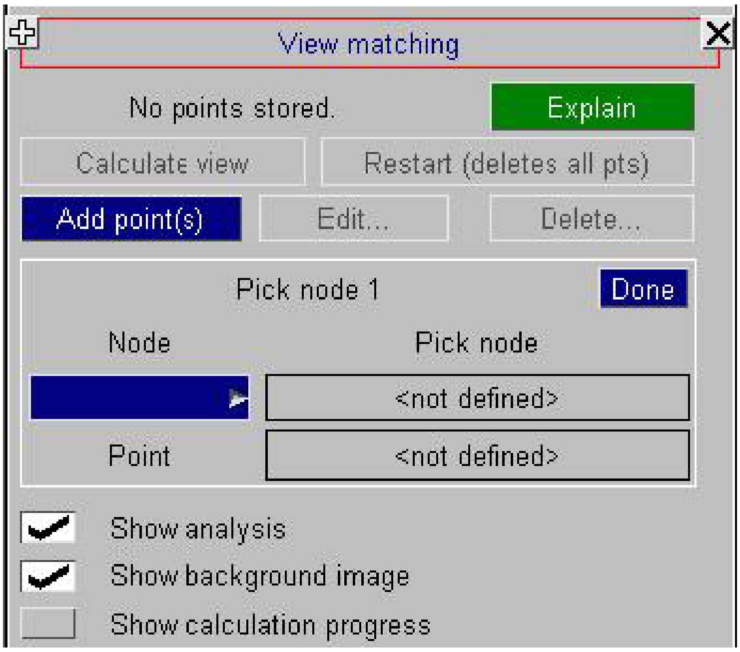
UPDATE_LEVEL = 3 **Frequent updates**

The plot is updated immediately as at level 2 above, but also following any blanking, clipping, etc, command that would change the image if explicitly redrawn.

Therefore the effects of blanking, etc are seen immediately.

Note 1 The default setting is 2 on a windows device.
Note 2 Level 3 is only recommended if you have a very fast display and/or a small model since it requires frequent redraws.
Note 3 Users with slow devices and/or with large models may find that level 1 is preferable to decrease redrawing effort.

9.5.4 **Match Image**



Automatically aligns the current model image with the background by calculating the transformation parameters required.

Lining up an image requires the calculation of 11 unknowns:

- The camera position (3 coordinates)
- The direction in which the camera is pointing (3 vector terms)
- The "Up" axis of the camera (3 vector terms)
- The distance of the object from the camera, ie perspective distance (1 term)
- The focal length of the camera lens, ie image scale (1 term)

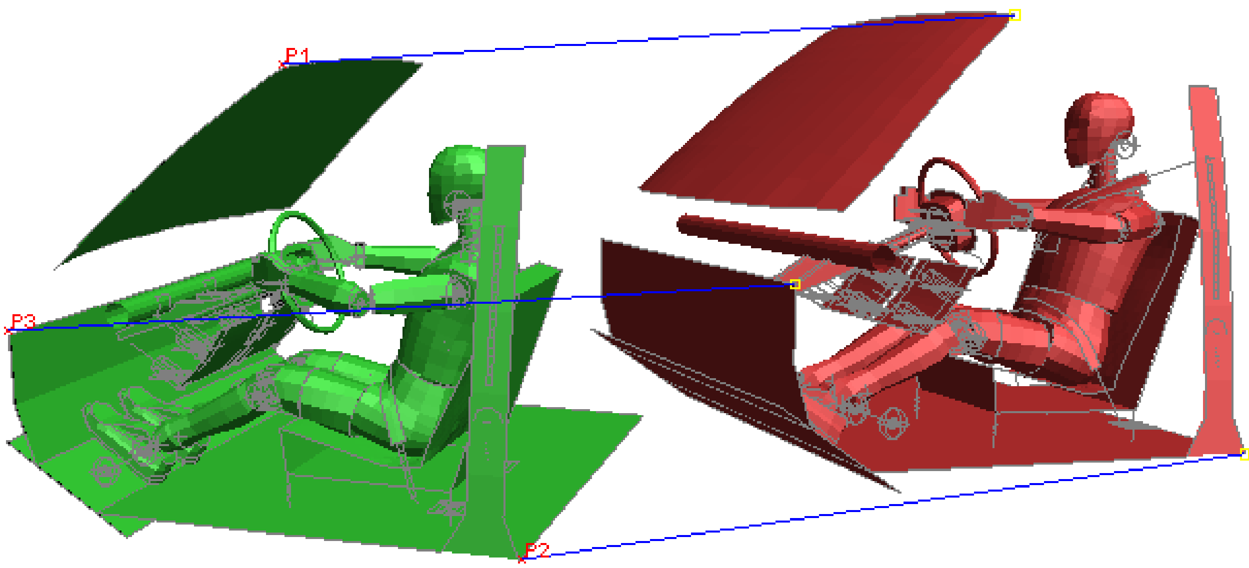
(In the orthographic case, where the object is viewed in a parallel sided frustrum, the perspective distance can be omitted leaving only 10 values to be computed.)

This calculation can be performed by Primer if four or more nodes on the model are matched to their corresponding points on the image. Generally 5 or 6 points are required for a good match.

Add point(s) Defining <node : point> pairs for matching.

In the (artificial) example below the green image on the left has been read in as a background image, and the task is to get the red analysis image on the right to lie on top of it.

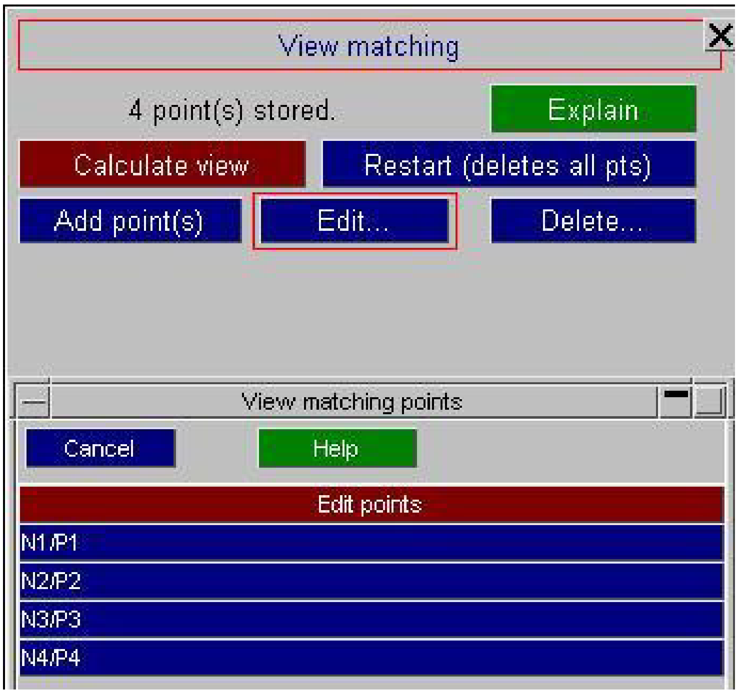
The user has defined 3 points so far: the nodes, identified by yellow pick symbols on the right, correspond to their matching points (red symbols and labels) on the left; the blue line shows which points and nodes are associated. These are screen-picked by selecting first the node, and then the corresponding point, and so on for the next pair.



Calculate: aligning analysis with image.

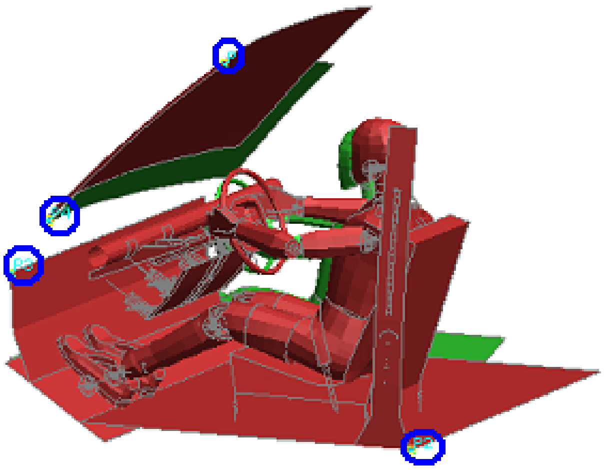
Once four or more <node : point> pairs have been defined it is possible to calculate the revised view. This will calculate the revised viewing parameters and update the image immediately. If the images can be matched and the points have been well chosen then the analysis should lie exactly over the target image.

Edit...: correcting poorly chosen points.



In the example below points have deliberately been chosen badly to obtain a poor match. (The error here is choosing points, ringed in blue, that lie more or less in a plane, making it difficult to calculate perspective distance correctly. In addition choosing only four points is often inadequate, and more can be required for a good solution.)

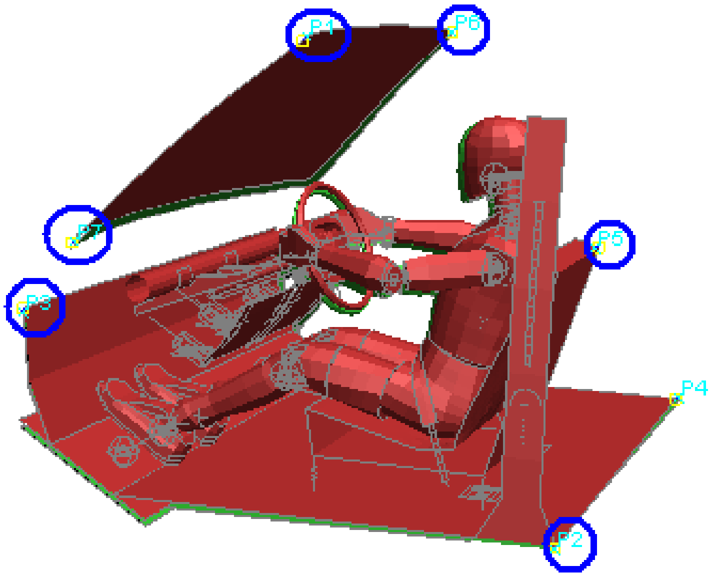
To edit a point screen-pick either its node or point (or select it from the menu), then repick its node or point.



Delete and Restart: Deleting points.

Delete allows you to delete individual points by selecting them as above. Each point is deleted immediately.
Restart deletes all points letting you make a fresh start.

You can **Add**, **Edit** and **Delete** points in any order. Here is the example above with 6 points (circled in blue) chosen rather more judiciously, and it can be seen that the correspondence is now very good.

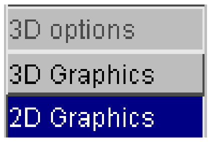


What is stored for matching.

"Node" data is stored as a reference to a node in a model.

"Point" data is stored as a parametric (x,y) screen space coordinate, so points will remain valid so long as the aspect ratio of the window remains the same. However in most cases if a window is resized it is best to delete all the points and start again if further matching is required.

9.6 Special 3D graphics driver options.



On a 3D graphics driver special 3D-only viewing options become available, (greyed out under 2D), as shown here.

9.6.0 Brief description of 3D vs. 2D graphics.

In 2D mode PRIMER treats the display device as a dumb 2D device on which lines, polygons and text can be drawn. All coordinates are expressed in 2D integer space, ie [x,y] only, and all calculation of hidden-surface removal, lighting, etc must be done in software. Dynamic viewing relies on the software recalculating and redisplaying images quickly.

In 3D mode much more intelligence is available in the graphics driver, and much of the effort of computing images can be shifted from the software to the hardware. In particular:

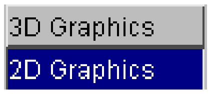
- Graphics coordinates exist in 3D [x,y,z] space, and the hardware does the transformation and projection onto the 2D screen. The software only has to provide the raw coordinates for an image once, and thereafter to change the view only a new scale, centre and rotation matrix.
- The hardware can compute shading, lighting and hidden-surface removal. So, again, the software only needs to provide raw coordinates, topology, light source data, etc, and then just ask the hardware to render it.
- The hardware can provide functions, such as Z-clipping, that are not available in software.

So 3D devices, especially those with hardware acceleration, give much faster graphics.

However there are also drawbacks to using 3D graphics: more memory is required since the full scene has to be sent to the driver using [x,y,z] floating-point coordinates. In addition laser plots cannot be generated by the 3D driver, so the capability to switch temporarily back to 2D mode has to be preserved.

Therefore there are options to control aspects of 3D graphics, and also the ability to switch back and forth between 3D and 2D modes.

9.6.1 Switching between 3D and 2D modes.



You can switch explicitly between 2D and 3D modes using the **3D Graphics** and **2D Graphics** buttons.
Some other graphics options also cause a switch.

On a 3D graphics driver the default mode is 3D, but certain graphics operations will switch the mode back to 2D. These are:

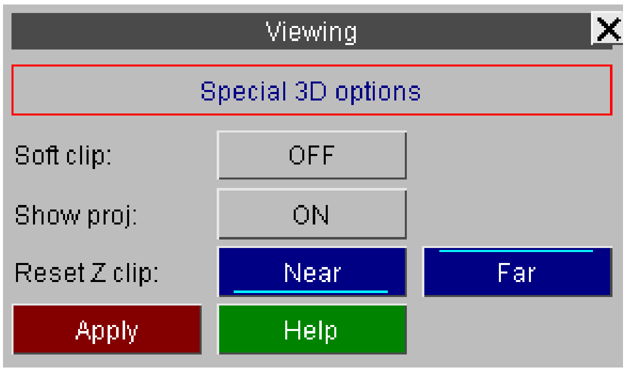
- Switching on dithered shading mode: continues until you switch it back explicitly.
- Plotting with laser output turned on. 2D mode is only transient during the course of the plotting operation, it is switched back to 3D automatically after each plot.

There are other circumstances when you might also want to switch explicitly to 2D mode:

- When producing LC line-contour plots the result in 3D mode can be a bit patchy, with contour lines dropping in and out of view. This is a function of the Z buffering in hardware, and software (2D) images look much better.
- When using the OPACITY switch for contact surface and beam plotting. This works after a fashion when in 3D mode, but the transparent structure overlay does not use proper hidden-surface removal. The results are better in 2D mode where more control is available in the software.
- When animating large models. The amount of data stored for a 2D animation can be far less than for 3D, and can get round memory shortage problems. (However you would do better to use the X-Windows driver in this situation: see 4.4.5.4.)

9.6.2 **3D_OPTS..**. Further 3D options.

The **3D_OPTS..**. button gives a control panel for further 3D options.



The Special 3D options panel is shown in figure 9.6.2.

These options are described below.

9.6.2.1 **Soft clip** Clipping graphics outside the current screen window.

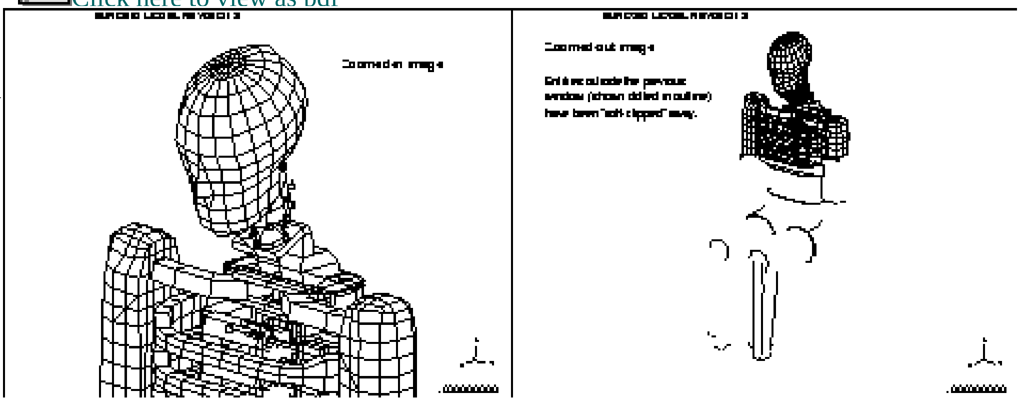
If you are dealing with a very large model, but are only looking at a small part of it, the 3D graphics driver can work unnecessarily slowly in its default mode of operation. This is because the whole model is sent to and manipulated by the graphics driver, despite the fact that you are only looking at a small part of it, in anticipation of your wanting to zoom out to see the whole of it.

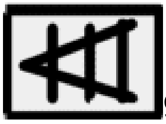
If you turn Soft Clip on, and redraw the image, the graphics will run faster. This is because the software has "clipped" (ie removed) those parts of the image not visible in the current window before sending it to the 3D graphics driver, so the 3D driver has to process fewer graphics entities. However this also means that if you zoom out those parts of the image outside the previous window will not be there. This is illustrated in figure 9.6.2.1(a) and (b).

In this example the user has zoomed in on the neck and upper chest region of a side-impact dummy (left hand image), and then zoomed out to what should show the full dummy. This exposes the jagged edges left by the 3D clipping algorithm.

To see the missing elements you need to issue an explicit drawing command at the new scale to recalculate the clipping and send more elements to the 3D graphics driver.

Click here to view as pdf





9.6.2.2 **SHOW_PROJ** Showing the viewing frustrum

On 3D devices it is possible to show the current viewing "frustrum" at the bottom left corner of the plot by turning **SHOW_PROJ** on.

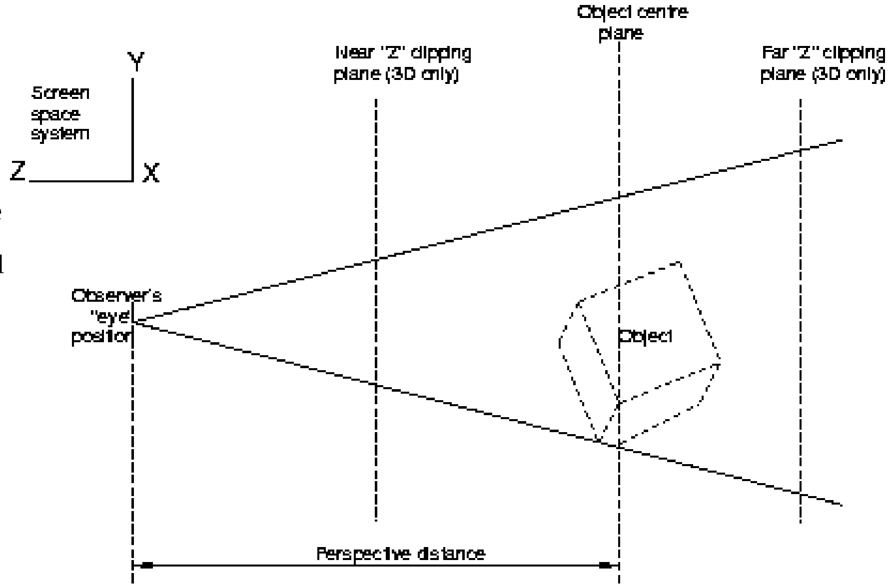
This shows the information in figure 9.6.2.2 (a copy of figure 9.1(b)).

The frustrum shown here assumes perspective projection.

The Z clipping plane locations are shown when **SHOW_PROJ** is on, and this can be very helpful when using Z clipping, as otherwise it is easy to "lose" the clipping planes.

The default near and far plane positions are drawn in green, and the plane locations in blue. So you can visualise movement relative to initial locations.

Click here to view as pdf



9.6.2.3 Using the Z clipping planes

The Z clipping planes are shown in figure 9.6.2.2. There are two planes: a "near" and a "far" one, which the hardware uses to clip the image in the +/- screen Z axis.

By default they are set just outside the +/-Z limits of the structure (shown as green lines in the projection box), so that no clipping takes place, but you can move them (shown as blue lines in the box) using the following mouse and keyboard meta-key combination:

<right shift> + <left mouse> Moves the near clipping plane.

Cursor symbol is 

<right shift> + <right mouse> Moves the far clipping plane

Cursor symbol is 

<right shift> + <mid mouse> Moves the both clipping planes

Cursor symbol is 

In all cases moving the mouse up moves the plane(s) away from you, and down moves towards you. This is a form of dynamic viewing: the planes move and the image gets updated as the cursor moves. It is recommended that you turn the **SHOW_PROJ** switch, described above, on as this will enable you to see the planes moving in the projection box.

To reset the planes to their default positions use the Reset Z clip **NEAR** and **FAR** buttons. This will reset them to their initial positions (shown by the blue lines in the projection box).

10 Scripting

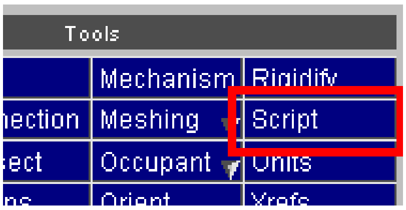
Introduction

Javascript is a freely available scripting language that is normally found performing the "work" behind interactive web pages, however its syntax and structure also make it an excellent tool for providing an externally programmable interface to programmes in general.

Within PRIMER it is implemented as an Application Programming Interface (API) which provides a range of functions that allow you to extract data from and poke it into the database, open windows, generate plots, and so on. This is documented in a separate Javascript API Manual.

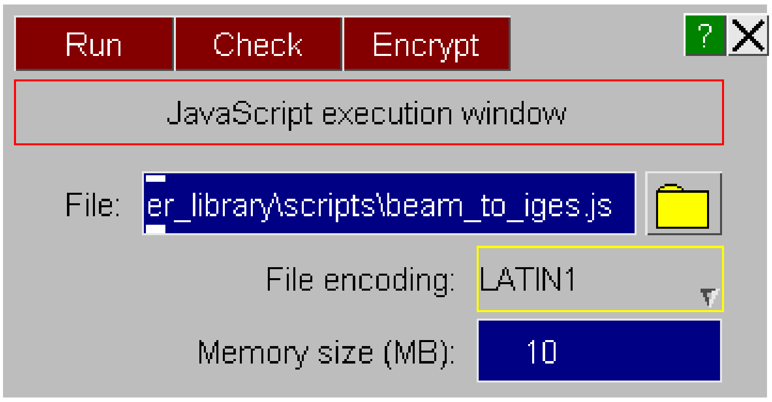
Anyone familiar with C or shell script programming will find existing Javascripts are instantly readable, and can be given minor edits without further ado. For those who are more ambitious a good guide to the language is "**Javascript, A definitive Guide**" by David Flanagan, published by O’Reilly, ISBN 0596101996. A brief tutorial on their use is given below.

10.1 Using Javascript in PRIMER.



Javscripts need to be ***compiled***, meaning turned from something human-readable into a set of instructions that a computer can understand; and then ***run*** in their compiled form. They can be changed and rerun in their modified form at any time without having to exit and re-enter PRIMER, making the "write, test, modify, re-test" development cycle very quick and easy.

10.1.1 Compiling and Running a script



Run Will both compile and run the script unless it contains syntax errors, in which case it stops with an error message when compilation fails.

Check Only compiles the script, reporting any errors found, and does not run it.

Encrypt A script can be encrypted so that the source code is hidden but the script can still be run (when compiling and running the script PRIMER decrypts the file in memory). Once encrypted the source code cannot be retrieved by an ordinary user so make sure that you keep the original file somewhere safe. As a last resort contact OASYS Ltd who can decrypt the script if required.

Memory size is the memory allocated for garbage collection in the JavaScript engine. Please see the garbage collection section for more details.

File encodings for scripts

Version 10.0 of PRIMER introduced the ability for unicode text to be used on widgets created in a script. Previous versions of PRIMER only supported English text so the default ASCII encoding was used for script files (this is still the default encoding for script files).

If you want to use unicode text in widgets then you must use a file encoding that is capable to representing the unicode ’chracters’ you require. The **File encoding** popup allows you to change the file encoding used when reading the script file. PRIMER supports the following file encodings:

Encoding	Description
LATIN-1	Default ’ASCII’ encoding
BIG5	Taiwan/Hong Kong (traditional)
EUC-CN	Extended unix code (Simplified Chinese)
EUC-JP	Extended unix code (Japanese)
EUC-KR	Extended unix code (Korean)
GB	Chinese (simplified)
GBK	Chinese
ISO-2022-CN	Chinese
ISO-2022-CN-EXT	Chinese (extended)
ISO-2022-JP	Japanese
ISO-2022-JP-2	Japanese (extended)
ISO-2022-KR	Korean
JOHAB	Korean
SHIFT-JIS	Japanese
UTF-8	Should NOT have a byte order mark (BOM).
UTF-16	Should have a byte order mark (BOM). If not present assumes big endian
UTF-16LE	Little endian with or without byte order mark (BOM)
UTF-16BE	Big endian with or without byte order mark (BOM)
UTF-32	Should have a byte order mark (BOM). If not present assumes big endian
UTF-32LE	Little endian with or without byte order mark (BOM)
UTF-32BE	Big endian with or without byte order mark (BOM)

Please contact Oasys Ltd if you have problems or require another encoding to be supported.

To show the unicode text the appropriate font must be used. This can be set using the preferences `primer*cjk_unix_font` and `primer*cjk_windows_font`.

10.1.2 Dealing with errors in scripts

Script errors come in two forms:

`Syntax errors` Are mistakes of Javascript grammar or spelling, resulting in error messages during compilation.

These are easy to detect and correct since the line number and offending syntax are both described by the compiler. The script needs to be edited to correct the problem and then recompiled. Sometimes several iterations of the compile/edit cycle are required to eliminate all errors from a script.

`Run-time errors` Are errors of context or logic in scripts that are syntactically correct, and thus have compiled, but which fail at some stage when being run.

A typical example of a run-time error is an attempt to divide a value by zero, yielding the illegal result infinity. More subtle errors involve passing an invalid value to a function, accessing an array subscript that is out of range, and so on.

10.1.3 Setting the Garbage Collection Memory Size

(This is an advanced topic, and you don’t need to understand it.)

Javascripts execute inside a memory "arena", allocated dynamically from the operating system, which grows in size as storage is requested within the script. This growth occurs due to requests for "new" variables within the script and also when API functions allocate and return values and objects, and it is limited only by what the operating system can deliver.

The nature of Javascript means that objects frequently become redundant, and it is wasteful not to reuse the storage that they occupy, therefore there is a "Garbage Collection" process running behind the scenes which periodically checks storage and releases that which is no longer needed. This process is automatic and hidden from the user, it just "happens".

However Garbage Collection is quite a CPU-hungry process, so it is only carried out periodically when a certain threshold is reached. This can sometimes be observed during script execution as a periodic "pause for thought", and if you are monitoring memory usage with a system tool you may see it drop during these pauses.

Clearly this threshold value must be large enough not to trigger excessively frequent (and costly) garbage collections, while at the same time not being so large that scripts build up large amounts of excess memory to the detriment of the rest of the programme.

The **Memory size** value in the JavaScript panel is the amount of memory allocated for garbage collection. Every time a new object, array, string or double precision number is used a garbage collection 'thing' is also allocated. The Memory size is the total memory for these 'garbage collection things', **NOT** the total memory for the script. The total memory for the script could be significantly higher than this value. e.g the memory required for a Model object could be several kbytes but the memory for the 'garbage collection thing' for the Model object will something like 10 bytes for a 64bit operating system.

When the memory used for garbage collection 'things' reaches a significant proportion of **Memory Size** (normally about 2/3) then garbage collection will take place to try to reclaim memory. If no memory can be reclaimed and the total memory used for garbage collection reaches **Memory size** then the script will terminate with an error.

If your script has to retain a large number of objects, arrays, strings etc in memory then you may have to increase the value for **Memory size**. This can also be done using the `primer*javascript_memory_size` preference.

To recap:

- This threshold does ***not*** limit the memory the script can use, that is limited only by the operating system.
- It sets the memory for Garbage Collection 'objects'.
- Scripts which allocate a lot of memory, and which exhibit frequent pauses, ***may*** run faster with a larger value.

- ... and finally:

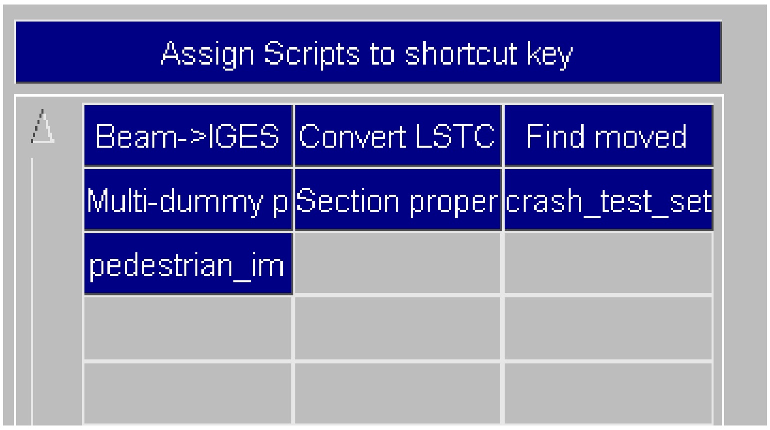
  If you don't understand this topic don't worry. Most scripts will run quite happily with the default value, and you can ignore this setting unless they appear to be struggling, in which case try raising it. (As good an approach as any is to keep on doubling this value until the script works, but don't use very large sizes unnecessarily.)

10.1.4 Assigning Javascripts to shortcut keys

Scripts can be assigned to shortcut keys to make them quick and easy to run. To do this use the **Assign Scripts to shortcut key** button (or press the shortcut key '**?**')

Using the above button will only assign the script to the shortcut key for this session of PRIMER. If you want to always assign the script to a shortcut key use the **`primer*F1_key`** preferences etc in the **`shortcut_keys`** branch for PRIMER in the oa_pref preference file. Note that the preference can be used to run either a shortcut, a script or a macro. For PRIMER to know that the preference should run a script and not a macro, the script **MUST NOT** have the extension **prm**.

10.1.5 Maintaining a library of Javascripts



When PRIMER starts it automatically looks for scripts in the directories:

- $OA_ADMIN/primer_library/scripts (if $OA_ADMIN is defined)
- $OA_INSTALL/primer_library/scripts
- $OA_HOME/primer_library/scripts

Each script that is found is assigned to a button in the script panel.

The directory that PRIMER looks in for script files can be changed in the oa_pref files in $OA_ADMIN, $OA_INSTALL and $OA_HOME by using the `script_directory` preference.

For example if you change the **`script_directory`** preference in the oa_pref file in the $OA_INSTALL directory to /test/primer_scripts then PRIMER will look for script files in the directories:

- $OA_ADMIN/primer_library/scripts (if $OA_ADMIN is defined)
- /test/primer_scripts
- $OA_HOME/primer_library/scripts

Using the "description:" comment at the top of a script to identify its purpose.

To help to identify scripts special comments are searched for in the top 10 lines of each script, and if **`description:`** is found, for example the comment line:

```
// description: Some description of the script's purpose
```

Then the description line is shown as hover text when the mouse is placed over that filename. In the example above the "**demo**" script has the line

```
// description: Javascript demonstration file
```

Using the "name:" comment at the top of a script to change its name

Normally the name shown for a script will be its filename, stripped of any leading pathname and trailing ".js" extension.

However if the string **`name:`** is found in the first ten lines of the script, then the following name will be used instead. For example the line:

```
// name: temporary
```

Will result in the script appearing with the name "**`temporary`**" in tha Javascript panel. This does not affect the actual name of the script, only the name on its library button.

10.1.6 Running a Javascript in "batch" mode.

All the above assumes that Javascripts will be run interactively from the user interface, however it is also possible to run a script in "batch" mode using the command line interface. The relevant command-line commands are:

`/SCRIPT`	**`READ`** `<script>`	Read, compile and execute <script>

To run a Javascript from batch these commands need to be placed in a command file and run using the command line "**`-cf=`***`command filename`*" option. For example the command file might be:

```
... some other commands
/SCRIPT READ my_script.js
```

...some further commands

And the command line required to run PRIMER might be something like:

$OASYS/PRIMER93.exe -d=default -cf=command_file **-exit** analysis_name

Obviously multiple script invocations may be placed in a command file. For more information see:

Using command files	Describes command files, and explains how to create and use them
Command line arguments	Describes the various command line arguments, and how to use them

10.2 A Brief Tutorial on Javascript in PRIMER

While most people associate JavaScript with web pages and html it is a full-featured programming language. Additionally JavaScript is not Java! JavaScript is completely unrelated to Java.

Hopefully, enough people are familiar enough with JavaScript through the internet to be able to use it in PRIMER. JavaScript has all of the functionality you would expect from a programming language, such as:

- variables (strings, numbers, booleans, objects, arrays)
- functions
- control flow statements such as `if`, `while`, `do`, `for`, `switch` etc.
- objects
- arrays
- regular expressions
- maths functions (sin cos, log, sqrt etc)

Additionally, PRIMER extends JavaScript by defining several new object classes specifically for PRIMER. A detailed reference on these classes is given in the separate Javascript API Manual available from Oasys Ltd. Over time this functionality will be extended. If you need to do something which is not possible with the current functionality then contact Oasys Ltd.

Probably the best way to see what sort of things are easily possible in PRIMER using JavaScript is to look at the example scripts which are given out with PRIMER in the **$OASYS/primer_library/examples** directory. However, additionally a few example scripts are given here and documented. For more details on the classes described in these examples see the separate Javascript API Manual.

10.2.1 Example scripts

Example 1: Make a 10x10 mesh of nodes and shells

Problem

How can you make a grid of new nodes and create a mesh of shells?

Solution

```
// Declare variables
var x, y, i, j;
// Create a new model using Model constructor
var m = new Model();
// Give message saying what we are doing
Message("Making nodes");
// Loop over y nodes
for (y=0; y<11; y++)
{
// Loop over x nodes
    for (x=0; x<11; x++)
    {
// make node in model m using Node constructor
        var n = new Node(m, 1+x+(y*11), x*10, y*10, 0);
    }
}
```

```
Message("Making shells");
for (i=1; i<=10; i++)
{
    for (j=1; j<=10; j++)
    {
// Make shell using Shell constructor
        var s = new Shell(m, i+(j*10), 1, ((i-1)*11)+j+0, ((i-1)*11)+j+1,
                                          ((i-0)*11)+j+1, ((i-0)*11)+j+0);
    }
}
// Update the graphics in model
m.UpdateGraphics();
// View XY plane
View.Show(View.XY);
// Autoscale view
View.Ac();
```

Discussion

To make the nodes and shells we first need to choose the model that they will be made in. The Model class in PRIMER gives you access to any existing models and also allows you to make new models. The line
`var m = new Model();`

creates a new model in PRIMER and m is then a Model object which you can use. We then need to make a grid of nodes. To make a 10x10 mesh of shells we need to make 11 nodes in the x direction and 11 in the y direction so we use 2 'for' loops to make the nodes. Just as a model can be made using the Model constructor a node can be made with the Node constructor. The line
`var n = new Node(m, 1+x+(y*11), x*10, y*10, 0);`

makes a new node in model m. The other arguments are the x, y and z coordinates of the node. We can make the shells in the same way using the Shell constructor. The arguments required are the model, shell number (EID), the part to create the shell in (PID - in this case we are using part 1) and the 4 nodes (N1-N4).

Once we have made all of the nodes and shells we have to tell PRIMER that things have been made and so the graphics need updating. This is done with the UpdateGraphics method for the model. Once we have refreshed the graphics we select the view we want to see and autoscale the plot.

The source code for this example is available here.

Example 2: Change the number of integration points on a *SECTION_SHELL card depending on thickness

Problem

You want to loop over all of the sections in your model (model 1) giving more integration points to the thicker sections.

Solution

```
// Get the model object for model 1
var m = Model.GetFromID(1);
// Get the first section card in the model
var s = Section.First(m);
// While there is a section card look at it
while (s)
{
// If this section card is a section shell type then look at it
    if (s.type == Section.SHELL)
    {
// If thickness is < 1 then assign 2 ipts, < 2 assign 3 ipts, otherwise 5 ipts
        if      (s.t1 < 1.0) s.nip = 2;
        else if (s.t1 < 2.0) s.nip = 3;
        else                 s.nip = 5;
    }
// Get the next section after this one
    s = s.Next();
}
```

Discussion

```
var m = Model.GetFromID(1);
```

retrieves the model pointer for model 1. Once we have the model pointer we can then look at the contents of the model.

```
var s = Section.First(m);
```

will return the section object for the first section in the model (or null if there are no sections). Once we have a section object then it is simple to look at the properties of the object and change them as required. First we look to see if the section type is correct (i.e the section is a shell section). If it is we look at the thickness and set the number of integration points.

```
s = s.Next();
```

will return the next section after this one or null if one does not exist. The while loop will keep looping until s is not ’true’ so when we get to the last section `s.Next` will return null and we will break out of the while loop.

Note that this is a very simple example and we have not done any error checking. For example it is possible that one (or more) of the section cards in the model has been refeered to by a part card but has not actually been read in or created. In this case PRIMER creates a ’latent’ definition. The example should check for this problem. This can easily be done by looking at the exists property of the section. The check on the section type could be extended by doing:

```
if (s.type == Section.SHELL && s.exists)
```

The source code for this example is available here.

APPENDICES

APPENDIX I: Standard Object Names and Acronyms

Inside PRIMER every class of object (nodes, parts, solids, etc...) has a standard "acronym" that is used when labelling items on the screen, and which can sometimes be needed when the user types in a specific object label. For example the acronym for a node is "N", thus node 27 will always be labelled as "N27".

In addition, when PRIMER has more than one model in memory it is necessary to prefix the object label with its model number. The acronym for a model is "M", thus if node 27 exists in both models 1 and 3 the two labels will be respectively:

M1/N27	(Model #1, Node #27)
M3/N27	(Model #3, Node #27)

You only have to remember these when using the **Key in** method of defining objects (section 6.2), and even then only when the object type is not implicit. For example to select node 10 in model #1 you will need to "key in":

10	If both object type (NODE), and the model id (1) are preset.
N10	If object type is ambiguous, but model id is preset.
M1/N10	If neither object type or model id are preset

In most situations the <model> and <object type> are implicit: either because of the context of the operation, or because of prior selections, or because of "filter" settings, and only numbers are required.

As well as acronyms every object has a "formal" name that is used when referring to it in error messages, diagnostic output, panel buttons, etc. This is the same as its LS-DYNA keyword where relevant, although PRIMER adds a few categories which are not found in LS-DYNA input.

The complete list of object types, their standard acronyms and their formal names is:

Object type		Standard Acronym	Formal name
Model		M	MODEL
Include File		INC	INCLUDE FILE
Airbag definition		ABAG	AIRBAG
	Interaction	AINT	AIRBAG_INTERACTION
	Reference Geometry	ARDT	AIRBAG_REFERENCE
	Shell Reference Geometry	ASRG	AIRBAG_SHELL_REFERENCE
ALE Arbitrary Lagrange/Euler		ALEX	ALE
	Multi-Material Group	ALMM	ALE_MULTI-MATERIAL_GROUP
	Reference System Curve	ALRC	ALE_REFERENCE_SYSTEM_CURVE
	Reference System Node	ALRN	ALE_REFERENCE_SYSTEM_NODE
	Reference Syetem Switch	ALRS	ALE_REFERENCE_SYSTEM_SWITCH
	FSI Switch MMG	ALFS	ALE_FSI_SWITCH_MMG
Boundary conditions (general)		BNDY	BOUNDARY
	Prescribed Motion	BPRM	PRESCRIBED_MOTION
	Spc	BSPC	SPC
Component		COMP	COMPONENT
	Gebod	CGBD	COMPONENT_GEBOD
	Hybrid III	CHB3	COMPONENT_HYBRIDIII
Constrained (generic types)		CNST	CONSTRAINED
	Generalized Weld	GWLD	GENERALIZED_WELD
	Interpolation	ITRP	INTERPOLATION
	Joint	JNTC	JOINT

	Joint Stiffness	JSTF	JOINT_STIFFNESS
	Lagrange in Solid	LAIS	LAGRANGE_IN_SOLID
	Linear	LINC	LINEAR
	Nodal Rigid Body	NRBC	NODAL_RIGID_BODY
	Node Set	NSET	NODE_SET
	Points	PNTS	POINTS
	Rivet	RIVT	RIVET
	Spotweld	SWLD	SPOTWELD
	Spline	SPLN	SPLINE
Generalized stiffnesses		JSTF	GENERALIZED
Contact (generic)		CGEN	CONTACT
	"Sliding" (general 3D)	CONT	CONTACT_SLIDING
	Geometric	CENT	CONTACT_GEOMETRIC
	Gebod	CGEB	CONTACT_GEBOD
	Interior	CINT	CONTACT_INTERIOR
	Rigid Surface	CRIG	CONTACT_RIGID_SURFACE
	1D (rebar)	C_1D	CONTACT_1D
	2D (slide lines)	C_2D	CONTACT_2D
	Auto Move	C_AM	CONTACT_AUTOMOVE
	Coupling	C_CO	CONTACT_COUPLING
	Guided Cable	C_GC	CONTACT_GUIDED_CABLE
Control cards (all)		CTRL	CONTROL
Damping (general)		DAMP	DAMPING
	Global	GDMP	DAMPING_GLOBAL
	Modal	MDMP	DAMPING_MODAL
Database (general)		DBAS	DATABASE
	Ascii	DASC	DATABASE_ASCII
	Binary	DBIN	DATABASE_BINARY
	Extent ascii	DAEX	DATABASE_EXTENT_ASCII
	Extent binary	DBEX	DATABASE_EXTENT_BINARY
	Extent ssstat	DASS	DATABASE_EXTENT_SSSTAT
	History	DH	DATABASE_HISTORY
	<Scalar Item>	DSCA	DATABASE_<scalar item>
	Cross-section	XSEC	DATABASE_CROSS_SECTION
	Nodal force group	NFGR	DATABASE_NODAL_FORCE_GROUP
	PWP Flow	PWPF	DATABASE_PWP_FLOW
	Tracer particles	TRAC	DATABASE_TRACER
Define (generic)		DEFN	DEFINE
	Alebag Bag	ALBG	DEFINE_ALEBAG_BAG
	Alebag Hole	ALHL	DEFINE_ALEBAG_HOLE
	Alebag Inflator	ALIN	DEFINE_ALEBAG_INFLATOR
	Box	BOX	DEFINE_BOX
	Connection Properties	CPRP	DEFINE_CONNECTION+PROPERTIES
	Coordinate System	CSYS	DEFINE_COORDINATE
	Contact Volume	CVOL	DEFINE_CONTACT_VOLUME
	Staged Construction Part	DSCP	DEFINE_STAGED_CONSTRUCTION_PART
	Construction Stages	DSTG	DEFINE_CONSTRUCTION_STAGES
	Death Times	DTIM	DEFINE_DEATH_TIMES
	Friction	FRIC	DEFINE_FRICTION
	Hex Spotweld Assembly	HSWA	DEFINE_HEX_SPOTWELD_ASSEMBLY
	Load Curve	LC	DEFINE_CURVE
	Curve Entity	LENT	DEFINE_CURVE_ENTITY
	Curve Compensation	LCMP	DEFINE_CURVE_COMPENSATION
	Curve Feedback	LFBK	DEFINE_CURVE_FEEDBACK
	Curve Trim	LTRM	DEFINE_CURVE_TRIM
	Spring/Damper Orientation Vector	SDOV	DEFINE_SD_ORIENTATION
	Set Adaptive	STAD	DEFINE_SET_ADAPTIVE
	Transformation	TFRM	DEFINE_TRANSFOR ATION

	Table	TABL	DEFINE_TABLE
	Vector	VECT	DEFINE_VECTOR
	Spotweld Failure Resultants	SWFR	DEFINE_SWFR
	Spotweld Rupture Parameter	SWRP	DEFINE_SWRS
	Spotweld Rupture Stress	SWRS	DEFINE_SWRP
Deformable to Rigid		DTOR	DEFORMABLE_TO_RIGID
Element (generic)		EL	ELEMENT
	Solid	H	SOLID
	Beam	B	BEAM
	Shell	S	SHELL
	Shell Source Sink	SHSS	SH_SS
	Thick shell	T	TSHELL
	Discrete (spring/damper)	D	DISCRETE
	Lumped inertia	IN	INERTIA
	Lumped mass	MA	MASS
	Mass Part	MP	MASS_PART
	Seatbelt	SB	SEATBELT
	Accelerometer	ACC	ACCELEROMETER
	Pretensioner	PRET	PRETENSIONER
	Retractor	RETR	RETRACTOR
	Sensor	SENS	SENSOR
	Slipring	SLIP	SLIPRING
	SPH	SPH	SPH
	Trim	TRIM	TRIM
Encrypted		CRYP	ENCRYPTED
Equation of State		EOS	EOS
Hourglass		HG	HOURGLASS
Initial conditions (generic)		INIT	INITIAL
	Stress Section	INSS	INITIAL_STRESS_SECTION
	Axial Force Beam	IAFB	INITIAL_AXIAL_FORCE_BEAM
Integration Beam		INTB	INTEGRATION_BEAM
Integration Shell		INTS	INTEGRATION_SHELL
Interface		IFCE	INTERFACE
Loads (general)		LOAD	LOAD
	ALE Convection	LALE	LOAD_ALE_CONVECTION
	Moving Pressure	LMOV	LOAD_MOVING_PRESSURE
	Body	LBOD	LOAD_BODY
	Segment	LSEG	LOAD_SEGMENT
	Segment Set	LSSG	LOAD_SEGMENT_SET
	Shell	LSHE	LOAD_SHELL
	Segment Nonuniform	LSGN	LOAD_SEGMENT_NONUNIFORM
	Segment Set Nonuniform	LSSN	LOAD_SEGMENT_SET_NONUNIFORM
	Thermal Variable Shell	LTVS	LOAD_THERMAL_VARIABLE_SHELL
Material (structural)		MAT	MATERIAL
	(Thermal)	TMAT	THERMAL_MATERIAL
Node		N	NODE
	Node Scalar	N_SC	NODE_SCLAR
	Node Rigid Surface	N_RS	NODE_RIGID_SURFACE
	Node Transform	N_TR	NODE_TRANSFORM
Parameter		PARM	PARAMETER
Part		P	PART
Part Adaptive Failure		PADF	PART_ADAPTIVE_FAILURE
Part Modes		PMOD	PART_MODES
Part Senor		PSEN	PART_SENSOR
Part Move		PMOV	PART_MOVE
Perturbation		PERT	PERTURBATION
Rail (general)		RAIL	RAIL
	Train	RTRN	RAIL_TRAIN

	Track	RTRK	RAIL_TRACK
Rigid (stone) walls		WALL	RIGIDWALL
Section		SECT	SECTION
Segment (for contact, etc)		SEG	SEGMENT
Sensor (general)		SNSR	SENSOR
	Control	SCON	SENSOR_CONTROL
	Switch	SSWT	SENSOR_SWITCH
	Define	SDEF	SENSOR_DEFINE
Set (generic)		SET	SET
	Beam	S_BM	SET_BEAM
	Discrete	S_DS	SET_DISCRETE
	Multi-Material Group	S_MM	SET_MULTI-MATERIAL_GROUP
	Node	S_NO	SET_NODE
	Part	S_PT	SET_PART
	Segment	S_SG	SET_SEGMENT
	2D Segment	S_2D	SET_2D_SEGMENT
	Shell	S_SH	SET_SHELL
	Solid	S_SO	SET_SOLID
	Thick shell	S_TS	SET_TSHELL
Termination		TERM	TERMINATION
Translate (Nastran, etc)		TRAN	TRANSLATE
User-defined data/subroutines		USER	USER

The following object names are unique to PRIMER, although they can be written to an LS-DYNA input deck after the *END card:

Object type		Standard Acronym	Formal name
Airbag "Origami"		ORIG	ORIGAMI
	Fold defn	FOLD	FOLD
	Orientation	ORNT	ORIENTATION
Assembly (subset)		SASS	ASSEMBLY
Assign Mass		ASSM	ASSIGN MASS
Connection		CONX	CONNECTION
Dummy definition		DUMM	DUMMY
	Assembly	ASSY	ASSEMBLY
Headform		HEAD	HEADFORM
	Target Point	TARG	TARGET POINT
	Headform Position	POSN	HEADFORM POSITION
IP Pendulum		IPPI	IP Pendulum
Group		GROP	GROUP
Mechanism		MECH	MECHANISM
	Connection	MCON	CONNECTION
	Point	MPNT	POINT
	Child	MMCH	CHILD
Seatbelt fitting		BDEF	SBELT Defn

APPENDIX IIa: Dummy "tree" file format.

A "dummy" is not a special model, rather it is a normal model in which the connectivity of parts is defined in a specific hierarchy using a "tree" file.

The following terminology needs to be understood:

"Dummy" Is the name of a model, or subset of a model, which constitutes a connected series of assemblies which can be manipulated in a hierarchical fashion. In fact it need not be a dummy (occupant) at all: any mechanism could be represented in this way so long as its assemblies form a discrete "tree" (or trees).

"Assembly" Is one or more PARTs and/or SETS of parts. These are manipulated as a rigid body for the purposes of positioning, although they may contain both rigid and/or deformable materials. Assemblies are connected via JOINT_ STIFFNESS definitions in a strictly defined parent/child hierarchy.

"Point" A location attached to an assembly that can be used to drive mechanism-style positioning, and also to impose restraints at a point. (Added in release 9.3, Feb 2007)

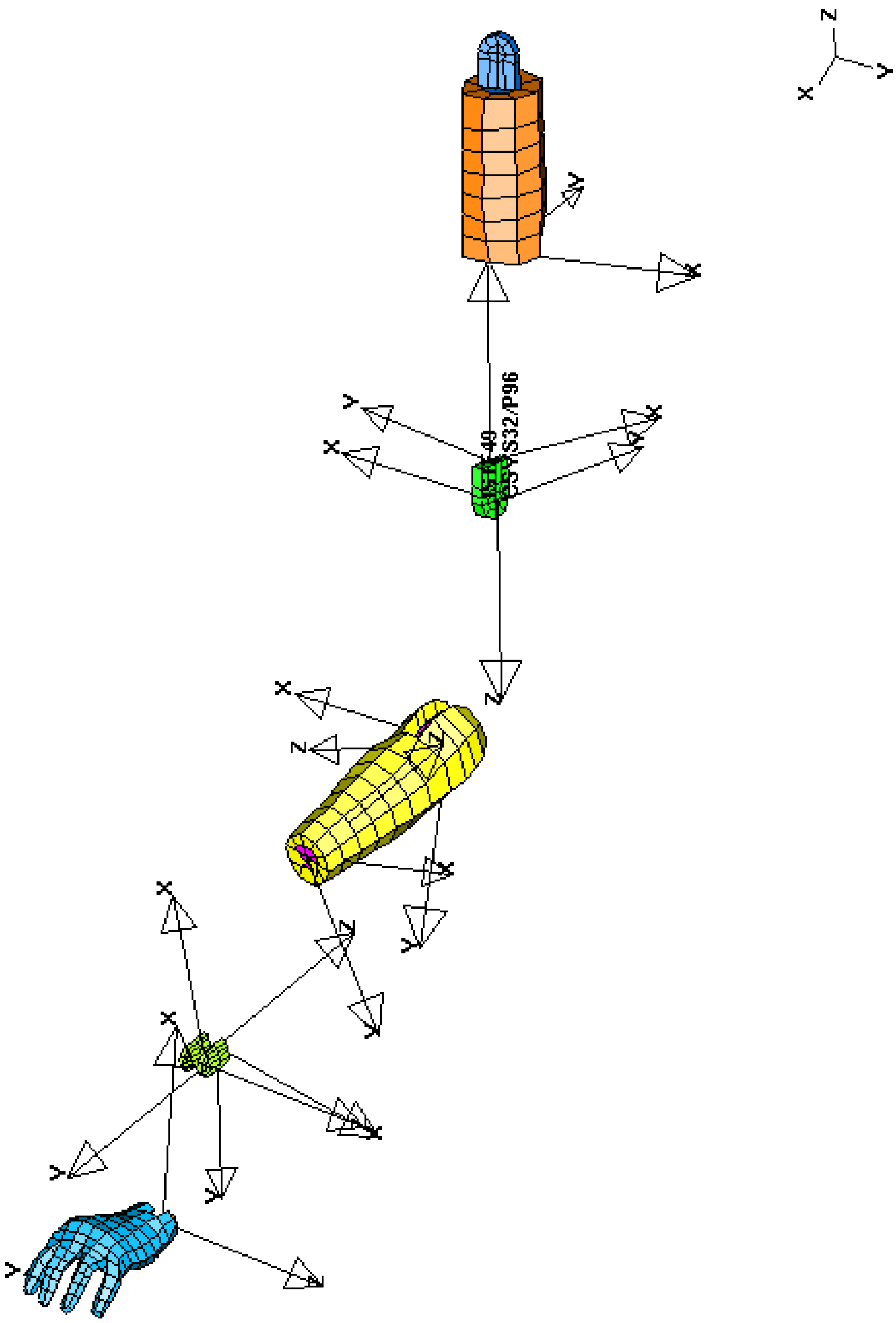
"Tree" Assemblies must be connected in a strict hierarchy. The "root" assembly is grand^{N}-parent to all other assemblies, which are arranged in a parent/child order. For example the pelvis is parent to the upper leg, which in turn is parent to the lower leg, with the foot being the youngest child.

"Parent/child" This terminology is used to represent the relationship between assemblies. A "child" assembly always follows the motion of its "parent", to the <nth> generation. For example waggling the upper leg also moves the lower leg and foot, all in a rigid-body sense. An assembly may have any number of children, and either zero or one parents. A child with no parents (an orphan) is usually the "root" of a dummy, but it is legal to have more than one "orphan" in a model.

However it is strictly illegal for a child to have more than one parent, as this would create a dynamically indeterminate mechanism requiring compatibility equations for its solution. Put another way assemblies may not connect to themselves, either directly or indirectly.

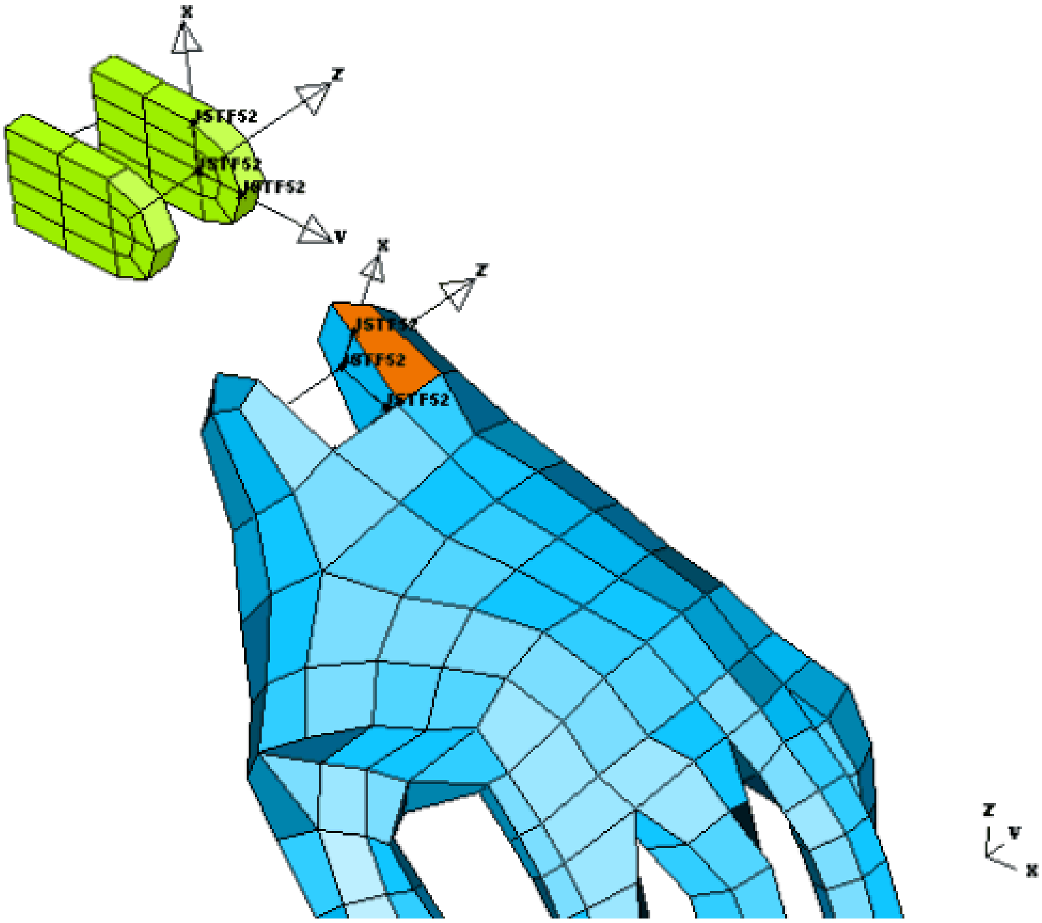
A "tree" file is made up of an extra set of keywords, and one such file for each dummy in a model is appended to the normal LS-DYNA (keyword) file after its *END card, where it is ignored by LS-DYNA, but read by PRIMER. A model may contain any number of dummies, and each must have a unique label within that model.

The purposes of appending the "tree" section to the analysis input deck are to try to make sure that a dummy and its tree data don’t get separated; that any renumbering which takes place is consistent in both dummy and tree file; and to permit adjustment of a dummy in a complete deck without having to go through the rigmarole of reading it in and repositioning it from scratch each time.



This figure illustrates "tree" file usage by showing an exploded diagram of the arm assemblies in a typical dummy. The assemblies are "upper arm", "elbow", "lower arm", "wrist" and "hand"; and the joint stiffnesses between them are shown.

It can be seen from the geometry of the connections that in this particular dummy rotation can only take place about one axis at each joint, and that the coordinate systems of the joint stiffnesses are aligned to these. It is also clear that angular rotations must be limited, which is done by using "stop angles" in the joint stiffnesses.



Here the connection between wrist and hand has been enlarged (still artificially separated) to show its organisation in more detail.

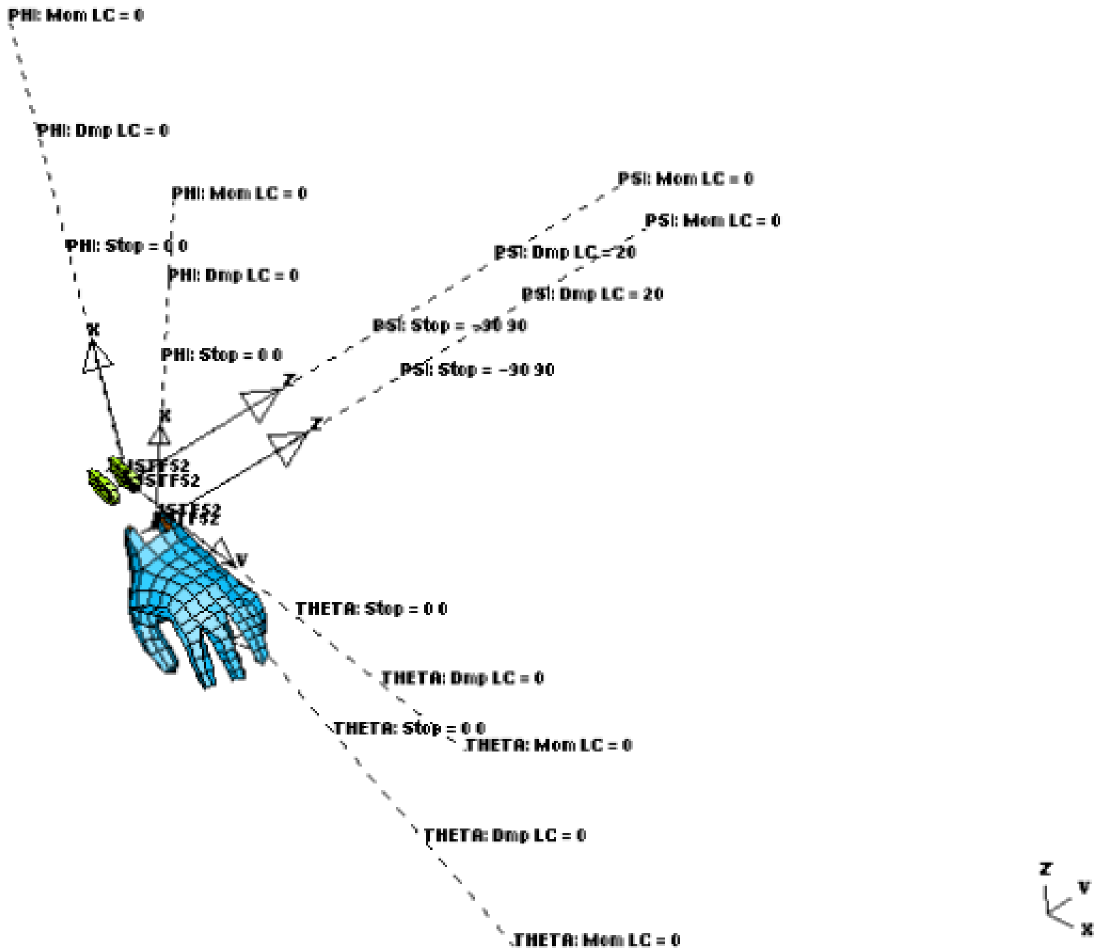
Clearly rotation of the hand can only take place in the "up and down" direction, which is about the local Z (Psi) axis of the joint stiffness.

The two sides of the joint stiffness definition can be seen in terms of their coordinate systems, and it is clear from this image that the following sensible modelling rules have been followed for them:

(1) Both coordinate systems have their origins at the same point in space. LS-DYNA does not require this, but it is difficult to visualise a connection clearly unless it is the case.

(2) Both coordinate systems are defined in terms of nodes attached to parts on their respective assemblies: either structural or extra nodes on rigid bodies. In this way the coordinate systems move with their parent parts.

(3) The node at the origin of the "parent" coordinate system on the wrist is the designated node about which the hand rotates. Again, not required but sensible.



Here the **NOTATE** option in VIS_2 has been turned on to add to the joint stiffness graphics their loadcurves and stop angles. This shows that parameters have only been defined for rotation about local Z (Psi), that the "stop angles" are +/-90 degrees, and that only damping is applied (so the initial angle between the coordinate systems generates no moment).

There is a separate revolute joint, not shown, through the wrist which constrains rotation to this axis only, so the joint stiffness does not need to restrict others. However to prevent PRIMER allowing positioning about axes other than Psi the "tree" file restricts rotation at this joint to local Z (axis 3). Here is the fragment of tree file defining wrist to hand connection:

```
*ASSEMBLY
        16Wrist left
         1         0         1
         9
$ Child #1 is the left hand (Assy #18, Joint stiff #52, rotates
$ about node #3980, rotation restricted to Z (3) axis)
        18        52      3980         3
```

The syntax of a "tree" file is as follows. All cards other than *DUMMY_START and *DUMMY_END are optional, and may appear in any order within these two _START/_END cards.

```
*DUMMY_START
<Label> <Title>
```

<Label>	I10	Must be unique within a model, as this identifies the dummy.
<Title>	A70	An arbitrary character string describing the dummy.

```
*H_POINT
<Hx> <Hy> <Hz> <Root Assembly label> <Root initial X angle> <Root initial Y
angle> <Root initial Z angle>
```

The following 3 lines are **optional**. Either they must be omitted, or all 3 lines must be supplied:

```
<XX Cosine> <XY Cosine> <XZ Cosine>
<YX Cosine> <YY Cosine> <YZ Cosine>
<ZX Cosine> <ZY Cosine> <ZZ Cosine>
```

<Hx>	E10.0	X coordinate of dummy H-Point
<Hy>	E10.0	Y ditto
<Hz>	E10.0	Z ditto
<Root assembly label>	I10	***Optional***: The label of the assembly below that will be treated as containing the H point, and to which initial dummy orientations will apply. If this is omitted the H-Point will be assigned automatically to the lowest numbered assembly that has no parent.
<Initial X angle>	E10.0	***Optional***: Initial rotations about [X,Y,Z] axes for root assembly (degrees) Initial X,Y,Z angles are simply a numeric offset applied to the root assembly angles, which otherwise will be initialised to [0,0,0]. **From V9.3 onwards the use of this feature is deprecated**, as the orientations of the dummy as a whole and each of its constituent assemblies are now stored as direction cosines, and it can cause confusion if further numeric offsets are added to the values calculated from these cosines.
<Initial Y angle>	E10.0	
<Initial Z angle>	E10.0	

The following 3 rows of **optional** direction cosines were added in release 9.3 to define the orientation of the Dummy as a whole.

They will only be written out if the Dummy’s cosines are not a unit matrix, which will only be the case of the Dummy has been rotated either via Orient or by using the Rotate or Reflect options in the Dummy positioning panel itself. (Positioning individual Dummy assemblies will not modify these overall Dummy cosines.)

If they are not present a set of unit cosines is assumed, implying that the Dummy as a whole has not been rotated.

<X Cosines>	3E20.0	The X row of cosines: XX, XY, XZ
<Y Cosines>	3E20.0	The Y row of cosines: YX, YY, YZ
<Z Cosines>	3E20.0	The Z row of cosines: ZX, ZY, ZZ

The H-Point need not be defined. If this definition is absent then [0,0,0] is assumed. Rotations of any orphan assemblies, generally the root assembly, take place about the H-Point.

```
*UNITS
<mass unit> <length unit> <time unit>
```

<mass unit>	A10	A valid mass unit name: kg, Te, lb
<length unit>	A10	A valid length unit name: m, mm, in
<time unit>	A10	A valid time unit name: s

Units may be upper or lower case, anywhere in the A10 field. Units need not be consistent, eg kg, in, s is legal (if daft!). If this definition is absent then no units are assumed and the dummy is dimensionless.

```
*AXES
<Coord system label>
```

Csys label	I10	(Optional) An existing *DEFINE_COORDINATE system label.

This defines a local axis system for the dummy, which is used when calculating the rotation angles of the"root" part. If this definition is absent then the internal reference axes of the dummy will be aligned with the global cartesian axes when initially read in, so this card may be omitted if you want your root part’s angles to be expressed as rotations about the global axes.

Some sort of reference system for root assembly rotations is required since this needs to rotate with the dummy if the whole dummy is oriented, otherwise root angles would change in a confusing manner.

```
*DYNA_POSITION
<node_1> <node_2>
```

Node label <node_1>	I10	Node at base of dummy neck
Node label <node_2>	I10	Node on dummy left or right hip

These nodes are used during the "Dyna method" positioning process. This keyword may be omitted if only PRIMER positioning is to be used.

```
*ASSEMBLY
<label>     <Title>
<#SET_PARTs> <#PARTs> <#children>  <#SET_NODES>  <locked>   (<csys>) <#contacts>
<dyna_pos>
<SET_PART_1> <SET_PART_2> ...      <SET_PART_n>
<PART_1>     <PART_2> ...          <PART_n>
<SET_NODE_1> <SET_NODE_2> ...      <SET_NODE_n>
<Part set>   <Box>      <tk factor><active> ... (1 line per contact)
<CHILD_1>    <JSTF_1>   <NODE_1a>  <dof codes>  (<NODE_1b>)
<CHILD_2>    <JSTF_2>   <NODE_2a>  <dof codes>  (<NODE_2b>)
 : : :
```

<Label>	I10	Label number for this assembly. This must be unique within this dummy, (but assemblies are "local" to a dummy, so the same label may occur in different dummies).	
<Title>	A70	Arbitrary name for this assembly.	
<#set_parts>	I10	The number of *SET_PARTs in this assembly	
<#parts>	I10	The number of *PARTs in this assembly	
<#children>	I10	How many "child" assemblies this assembly is "parent" to.	
<#set nodes>	I10	The number of *SET_NODES in this assembly	
<locked>	I10	Locked degrees of freedom during mechanism-style positioning. Any permutation of 123456, or 0 for none.	
<csys>	I10	Optional local coordinate system for assembly restraints during mechanism-style positioning	
<#contacts>	I10	Number of contact definitions	
<dyna_pos>	I10	Dyna position data flag (1 if data to be read, 2 if assembly is also flagged to be rigidified)	
<set_part_1...	8I10	Define <#set_parts> entries	8 entries per line, using as many lines as required
<part_1 ...	8I10	Define <#parts> entries	8 entries per line, using as many lines as required
<set_node_1...	8I10	Define <#set_nodes> entries, 8 to a line *(The option of Node Sets in assemblies is new in PRIMER release RC2)*	8 entries per line, using as many lines as required

An assembly may be made up of any number of SET_PARTs and/or PARTs and/or SET_NODEs, whichever is more convenient. Parts may be defined more than once, ie occur both explicitly and in sets, only a single instance will be used. For visualisation purposes you should have at least one part since node sets are not easy to interpret visually.			
<Part set>	I10	Part set for contact	<#contacts> lines of data, each definition starts a new line.
<Box>	I10	Optional box to delimit contact	
<tk factor>	E10.0	Factor on true shell thickness for contact purposes	
<active>	I10	Flag to denote contact active (1) or inactive (0)	
<child_n>	I10	The <nth> child assembly label. Required.	
<jstf_n>	I10	The GENERALIZED_STIFFNESS connecting parent assembly to child <n>. If omitted free rotation about dummy axes is assumed unless node NB is defined.	
<node_na>	I10	The NODE on the parent assembly about which child <n> rotates. If omitted the child rotates about the dummy H-Point.	
<dofs_n>	I10	The degrees of freedom about which child <n> may rotate with respect to the parent. The codes are "1", "2", "3" for local Phi, Theta, Psi (x,y,z) axes respectively. Given in any order, for example 13 = rotation permitted about Phi and Psi. If omitted no rotation about any axes will be permitted.	
<node_nb>	I10	Optional second node. If no joint stiffness is defined then if NB is defined the rotation axis will be about the vector NA - NB.	
The <child> <jstf> <node_a> <dofs> <node_b> line is repeated for <#children> child assemblies.			
<node_1>	I10	Dyna positioning node #1	<dyna_pos> nust be set to read this line of data.
<node_2>	I10	Dyna positioning node #2	
<node_3>	I10	Dyna positioning node #3	

```
*POINT_NODE
<title>
<assembly id> <node id> <restrained DoFs> (<csys>) (<h_pt>)
```

<title>	A80	Title for the point
<assembly id>	I10	Label of assembly to which point is attached
<node id>	I10	Label of node from which point coordinates are taken
<restrained DoFs>	I10	A restraint code made up of any permutation of 123456, or 0 for none
<csys>	I10	Optional: a coordinate system to give restraint in local axes.
<h_pt>	I10	Optional: set to 1 if this is an automatically generated point at the dummy H-Point

```
*POINT_LOCATION
<title>
<assembly id> <px> <py> <pz> <restrained DoFs> (<csys>) (<h_pt>)
```

<title>	A80	Title for the point
<assembly id>	I10	Label of assembly to which point is attached

<px>	E10.0	X coordinate of point
<py>	E10.0	Y coordinate of point
<pz>	E10.0	Z coordinate of point
<restrained DoFs>	I10	A restraint code made up of any permutation of 123456, or 0 for none
<csys>	I10	Optional: a coordinate system to give restraint in local axes.
<h_pt>	I10	Optional: set to 1 if this is an automatically generated point at the dummy H-Point

Points are optional. They are relevant only during mechanism-style positioning, and provide two related functions:

- By restraining degrees of freedom at a point an assembly can be given a "point" restraint. This can be in a local system if <csys> is defined for the point.
- Points can also be used to "drive" mechanism-style movement by giving them new target coordinates and letting PRIMER iterate to achieve these.

Any number of points may be defined for a dummy, and an assembly may have any number attached to it.

***POSITION**

Any number of positions may be stored for a dummy, and position information is identical for dummies and mechanisms. A description of these and their card format is given below.

***DUMMY_END**

Terminates the dummy definition. This is assumed if a physical <end of file> is found, but is mandatory if a second dummy definition (or other keyword) is to follow.

Rules for "tree" files.

Syntax:

*- Syntax is based on LS-DYNA keyword format. Thus numbers should be right-justified in their respective fields, and comment lines ($....) may be inserted anywhere. Character strings should be left justified.

*Any **`*DUMMY_...`** cards must appear after the **`*END`** card in an input file, as LS-DYNA will not recognise them.

*- Only the keywords described above may appear between the **`*DUMMY_START`** and **`*DUMMY_END`** cards.

Numbering:

*There are no restrictions on the labels for dummies or assemblies, other than that they must be valid, positive integers.

*Dummy numbers must be unique: you cannot have two dummies numbered "2" within a model.

*Assembly numbers within a dummy must be unique: you cannot have two assemblies numbered "10" within a dummy. Assembly numbers are "private" to their parent dummy definition. Thus if you have two dummies in a model both may have an assembly numbered "8" without there being any conflict.

*Thus when dummies are merged into a model, or models already containing dummies are merged, the dummy numbers may need to be incremented but the assembly numbering within a model will not be changed. Note that the merge operation may result in the numbering of part/set/node/etc entities being changed, this will be reflected in the dummy definition too.

*Assemblies can only refer to PARTs, SET_PARTs, NODEs, JSTFs, etc that are within their current model. Thus a dummy tree file definition cannot be used independently of a model and, more importantly, when transferring dummy definitions between input decks by hand take great care to ensure that the node/part/set/etc numbering in the new deck matches that in the old one. (If PRIMER is used to merge decks this is handled automatically.)

Modelling rules:

Problems are likely to arise if "free-standing" items such as **`*DEFINE_BOX`**, **`*DEFINE_COORDINATE_SYSTEM`**s defined other than by nodes, and such like, are used.

The reason is that these items do not belong unambiguously to PARTs, so they may not be flagged for rotation/translation/reflection when a part is so operated upon. As a consequence orienting a dummy or its assemblies may move items in or out of them. Therefore it is recommended that:

`*DEFINE_BOX`	Should not be used, since this may not be oriented correctly. Contacts, walls, etc should avoid selecting slave entities by volume, and use instead PART, SET, SEGMENT or NODE ids.
`*DEFINE_VECTOR`	Should also not be used, for the same reasons.
`*DEFINE_COORDINATE`	Should only be used in its _NODES variant, and the nodes employed should be attached to the parent PART that is supposed to control its orientation. The attachment method doesn't matter: explicit nodes on elements, extra nodes on rigid bodies, rigid body merges, nodal rigid bodies, etc; so long as the nodes are subordinate to the PART in question.
`*DEFINE_SD_ORIENT..`	Can be used so long as some thought is given to its use. These orientation vectors are unambiguously subordinate to their parent DISCRETE elements, so they meet the requirement that they are attached to PARTs. But, obviously, problems will arise if a single orientation vector is used for springs in different PARTs which may be rotated independently. You should use separate orientation vectors for all springs unless it is clear that a group of springs will always be rotated together (effectively rigidly). It is not mandatory to use the two node method for defining these vectors, although this may help to make the display of them clearer since it will both locate them in space and give them an explicit length**.**

The restrictions on **`BOX`**es and **`VECTOR`**s may be eased in the future if a method of defining them by nodes becomes available. This will move them to the same status as the **`*DEFINE_COORDINATE_NODES`** option.

`*MAT_RIGID`	May be used, but if the options to constrain the material or request output in local coordinate systems are used, then separate **`*MAT_RIGID`** definitions should be repeated for each assembly of parts. For restraining motion of rigid bodies it might be preferable to use the **`*BOUNDARY_PRESCRIBED_ MOTION_RIGID`** method, but not its **`_LOCAL`** variant!
`*MAT_xx orthotropic`	Where orthotropic materials are used problems will only arise if global orthotropicity is defined. This presents the same problem as above in a different guise, since different PARTs sharing a common material can only have a single set of material axes defined. Again, the solution is to have separate material definitions for each part (or rigidly connected set of parts).

These two restrictions on the use of material models may be removed in the future, as PRIMER may detect this conflict and generate separate material definitions automatically; but for the time being they stand, and represent good modelling practice anyway.

A **`PART`** may not appear in more than one assembly in a dummy, otherwise a conflict will arise when it is positioned because more than one assembly will be trying to update the part's nodal positions. This will be detected as an error and the dummy definition will be rejected.

Similarly a **`NODE`** should not have its motion "driven" during positioning by more than one assembly. This could happen if a part defined in assembly A extended to include elements and/or nodes common with parts in assembly B. This too will be detected and flagged as an error. However there are some specific exceptions to this rule:

1. **Extra nodes on a rigid part in assembly A may legally be part of structure in assembly B.**

   This is common modelling practice where two assemblies need to be able to articulate during positioning, but to be clamped together during analysis. The positioner detects this situation and ensures that motion of assembly A does not update the coordinates of its "extra" nodes in assembly B.

2. **Rigid parts in assembly B that are slaves to a master part in Assembly A are permitted.**

   This is also common modelling practice to achieve articulation during positioning, but a rigid connection during analysis. Normally PRIMER carries motion of a master rigid part through to its slaves, but in the positioner this is detected and the slave part's nodes will not be updated if they are found to be present on a different assembly to that of the master part.

3. **If a part consisting of null shells (material type 9) is used to "skin" the Dummy in order to give a continuous surface for contact it is legal for this part to cover more than one assembly.**

   Such a part will not be included when nodal coordinates are updated following positioning, so the assemblies in question should also include "structural" parts (as would be required during analysis anyway). However such a part may still only be defined once in the dummy, even if its connectivity spans multiple assemblies. However there is an exception to this exception:

Exception: "Isolated" null shell parts *are* included when an assembly is positioned.

Experience with working dummies has shown that some modellers build target markers into their dummies by creating targets made out of null shell parts, attached to the dummy by tied contacts. The "exclude null shell parts" rule implicit in exception (3) above meant that these target markers got left behind when the dummy moved since their nodes were not on parts in the assembly.

To fix this problem null shell parts ***are*** moved with an assembly if their nodes are not common to any "structural" (meaning on non-null parts or on node sets) nodes in the assembly. The restriction that such parts should not span more than one assembly remains, meaning that target markers defined in this way must have separate null shell parts for each assembly's markers.

4. **Orientation ("3rd") nodes on beams in assembly A that are located in assembly B may *optionally* be ignored.**

   Normally it would be both illegal and also extremely poor modelling practice to locate the 1st two nodes of a beam in one assembly and its "3rd node" in another one. However there are occasions when this might occur, for example when beams are circular in section so their orientation is not important, and any old node will suffice for orientation.

   The pre-positioning check will detect this and warn you, and strictly speaking you should correct the model to eliminate the problem. However there is an "**Ignore B3**" option following the check which, if chosen, will suppress propagation to 3rd nodes of beams. This setting is "remembered" for the Dummy or Mechanism for the duration of this PRIMER session so that the warning will not pop up again each time you perform a positioning operation.

   This should be regarded as a quick and dirty way to deal with an annoying problem, and not as a solution. In particular if you are preparing models for others to use you should not rely on this flag since your future users will have to select it every time they position your model.

5. **PRIMER connections between assemblies may *optionally* be ignored**

   Connections in PRIMER may be "made" if they have been explicitly realised, or "potential" if they would be realised via a pass through the Connection menu. The Dummy and Mechanism positioner considers a "made" connection between two assemblies to be something that links them together, and it will give a warning about this in the pre-positioning check. However it will not consider "potential" connections in the same location.

   You can choose to ignore connections in the pre-positioning check via the "**Ignore Conn**" option, but you need to think through what this will imply. In particular a "made" connection which straddles two assemblies may be pulled apart when they are positioned causing it to become invalid. For example a spotweld beam may not longer be tied to its surface, or a weld may become too long.

   On the whole it is probably best to avoid joining Dummy and Mechanism assemblies via PRIMER connections, but if it is to be done this way then it is recommend that they are rechecked when the positioning process is completed.

The list above is almost certainly not exhaustive: builders of dummy models should think through carefully what happens when assemblies of PARTs are oriented independently of each other.

Example of a "tree" file

This example refers to a typical dummy. Each assembly below has been defined (in the main body of the input deck) using **`*SET_PART`**s.

```
$
$
*DUMMY_START
1This is a test DUMMY definiton
$
*H_POINT
       0.0         0.0         0.0
$
*UNITS
        te          mm           s
$
*ASSEMBLY
         1Neck, Thorax, Torso
         3           0           5
```

```
        21         22          23
$
$ Child #1 is the head
         2          9       20589        123
$ Child #2 is the left yoke
         9          4       10597        123
$ Child #3 is the right yoke
        10          5       14963        123
$ Child #4 is the Left upper leg
         3         16        2252        123
$ Child #5 is the Right upper    leg
         4         17        2044        123
$
$
*ASSEMBLY
         2Head
         1          0           0
        20
$
$
*ASSEMBLY
         3Upper leg left
         1          0           1
        24
$
$ Child #1 is the left knee/lower    leg
         5          2        4329        123
$
$
*ASSEMBLY
         4Upper leg right
         1          0           1
        25
$
$ Child #1 is the left knee/lower    leg
         6          1        4320        123
$
$
*ASSEMBLY
         5Lower leg left
         1          0           1
        26
$
$ Child #1 is the left ankle &    foot
         7         22          4562        123
$
$
*ASSEMBLY
         6Lower leg right
         1          0            1
        27
$
$ Child #1 is the right ankle    & foot
         8         23          5437        123
$
$
*ASSEMBLY
         7Foot left
         1          0            0
        28
$
*ASSEMBLY
         8Foot right
         1           0            0
        29
$
*ASSEMBLY
         9Yoke left
         1           0            1
        30
$
```

```
$ Child #1 is the upper left arm
        11          18        18559        123
$
$
*ASSEMBLY
         10Yoke right
         1           0             1
        31
$
$ Child #1 is the upper right    arm
        12          19         19281        123
$
$
*ASSEMBLY
         11Upper arm left
         1           0             1
        32
$
$ Child #1 is the left elbow
        13          11         19138        123
$
$
*ASSEMBLY
         12Upper arm right
         1           0             1
        33
$
$ Child #1 is the right elbow
        14          14         19860        123
$
$
*ASSEMBLY
         13Elbow left
         1           0             1
        34
$
$ Child #1 is the lower arm left
        15          10         19136         123
$
$
*ASSEMBLY
         14Elbow right
         1           0             1
        35
$
$ Child #1 is the lower arm right
        16          13         19858         123
$
$
*ASSEMBLY
         15Lower arm left
         1           0             1
        36
$
$ Child #1 is the left wrist
        17          12         19140         123
$
$
*ASSEMBLY
         16Lower arm right
         1           0             1
        37
$
$ Child #1 is the right wrist
        18          15         19862         123
$
$
*ASSEMBLY
         17Wrist left
         1           0             1
        38
```

```
$
$ Child #1 is the left hand
        19          20         18888         123
$
$
*ASSEMBLY
         18Wrist right
         1           0             1
        39
$
$ Child #1 is the right hand
        20          21         19610         123
$
$
*ASSEMBLY
         19Hand left
         1
        40
$
*ASSEMBLY
        20Hand right
         1
        41
$
*DUMMY_END
```

APPENDIX IIb: Mechanism file format.

Mechanisms are similar to Dummies in many ways: they share a common Assembly and Point definition syntax, and when a Dummy is "free dragged" in mechanism-style positioning PRIMER automatically builds an internal mechanism to perform the drag.

However there are also some significant differences:

- Mechanisms do not have the hierarchical parent/child structure of dummies, as their connectivity can be completely arbitrary. Therefore although their keyword card formats are similar it would be misleading to refer to them as a "tree file".
- Because of the absence of a hierarchy Mechanism assemblies are a little different different from Dummy ones, in particular they do not define "child" assemblies.
- Mechanism assemblies are joined together by "Connections" which have to be defined explicitly, whereas Dummy connectivity is defined by the "child" lines on Assembly cards.
- Finally Mechanisms can have child mechanisms or dummies, whereas dummies cannot.

It is not expected that mechanisms will be created outside PRIMER as defining them interactively within the programme is very easy, so the card formats are included mainly for completeness.

```
*MECHANISM_START
<label> <title>
```

<Label>	I10	Unique label for the mechanism
<Title>	A70	Optional title

This card starts a new Mechanism definition, giving its label and title. All cards between this and the corresponding **`*MECHANISM_END`** are "private" to this mechanism definition.

```
*ASSEMBLY
<label>     <Title>
<#SET_PARTs> <#PARTs>    <unused>    <#SET_NODEs>  <locked>   <csys>
<#contacts>
<SET_PART_1> <SET_PART_2> ...        <SET_PART_n>
<PART_1>     <PART_2> ...            <PART_n>
<SET_NODE_1> <SET_NODE_2> ...        <SET_NODE_n>
<Part set>   <Box>       <Tk factor> <active>
```

: : :

<Label>	I10	Label number for this assembly. This must be unique within this mechanism, (but assemblies are "local" to a mechanism, so the same label may occur in different mechanisms).	
<Title>	A70	Arbitrary name for this assembly.	
<#set_parts>	I10	The number of *SET_PARTs in this assembly	
<#parts>	I10	The number of *PARTs in this assembly	
<unused>	I10	This field is unused for mechanism definitions (see note 1 below)	
<#set_nodes>	I10	The number of *SET_NODE definitions in this assembly	
<locked>	I10	Locked degrees of freedom during positioning. Any permutation of 123456, or 0 for none.	
<csys>	I10	Optional local coordinate system for assembly restraints during positioning	
<#contacts>	I10	Number of contacts between assemblyand fixed "structure"	
<set_part_1...	8I10	Define <#set_parts> entries.	8 entries per line, using as many lines as required
<part_1 ...	8I10	Define <#parts> entries.	8 entries per line, using as many lines as required
<set_node_1...	8I10	Define <#set_nodes> entries	8 entries per line, using as many lines as required
Note 1:	PRIMER version 9.3RC1 contained Nodal Rigid Bodies in this "slot". These have been withdrawn and replaced with Set Nodes since the latter are more flexible and permit nodes to be defined explicitly in assemblies. Models which contain Nodal Rigid Bodies in assemblies may be converted to the new format by replacing the nodal rigid body labels with the labels of their node sets, which in many cases will be identical. The behaviour of the assembly during mechanism analysis will be identical.		
An assembly may be made up of any number of SET_PARTs and/or PARTs and/or SET_NODEs, whichever is more convenient. Parts may be defined more than once, ie occur both explicitly and in sets, only a single instance will be used. Assemblies should contain at least one part otherwise visualising and dragging them may prove difficult.			
<Part set>	I10	Part set for contact	<#contacts> lines of data, each definition starts a new line.
<Box>	I10	Optional box to delimit contact	
<tk factor>	E10.0	Factor on true shell thickness for contact purposes	
<active>	I10	Flag to denote contact active (1) or inactive (0)	

```
*CONNECTION_PIN
<Title>
<assy_1> <assy_2> <node> <locked>
```

<title>	A80	Optional title for connection

<table>
<tr><td><assy_1></td><td>I10</td><td colspan="2">Assembly #1</td></tr>
<tr><td><assy_2></td><td>I10</td><td colspan="2">Assembly #2</td></tr>
<tr><td><node></td><td>I10</td><td colspan="2">Node at connection position.</td></tr>
<tr><td><locked></td><td>I10</td><td colspan="2">0 for unlocked joint, 1 for locked.</td></tr>
<tr><td><cx></td><td>E10.0</td><td>Connection position X coord</td><td rowspan="3">Note: These fields are only present in V10 onwards.

From V10.0 onwards a pin location may be defined either by a node or by an explicit position. If a node is defined it is used, regardless of any position <cx,cy,cz>, otherwise the stipulated position is used.</td></tr>
<tr><td><cy></td><td>E10.0</td><td>Connection position Y coord</td></tr>
<tr><td><cz></td><td>E10.0</td><td>Connection position Z coord</td></tr>
</table>

```
*CONNECTION_LINE
<Title>
<assy_1>  <assy_2>  <node_1>     <node_2> <pos_slide> <neg_slide> <cur_dist>
<locked>/<-1>/<-2>
<pos_rot> <neg_rot> <curr_angle> <locked> <a3_active> <assy_3>    <factor_1>
<factor_2>
<c1_x>    <c1_y>    <c1_z>       <c2_x>   <c2_y>      <c2_z>
```

<table>
<tr><td><title></td><td>A80</td><td colspan="2">Optional title for connection</td></tr>
<tr><td colspan="4"></td></tr>
<tr><td><assy_1></td><td>I10</td><td colspan="2">Assembly #1</td></tr>
<tr><td><assy_2></td><td>I10</td><td colspan="2">Assembly #2</td></tr>
<tr><td><node_1></td><td>I10</td><td colspan="2">First node on line</td></tr>
<tr><td><node_2></td><td>I10</td><td colspan="2">Second node on line</td></tr>
<tr><td><pos_slide></td><td>E10.0</td><td colspan="2">Permitted slide distance in +ve direction</td></tr>
<tr><td><neg_slide></td><td>E10.0</td><td colspan="2">Permitted slide distance in -ve direction</td></tr>
<tr><td><cur_dist></td><td>E10.0</td><td colspan="2">Current slide distance</td></tr>
<tr><td><locked>
or
<-1>
or
<-2></td><td>I10</td><td>0 for unlocked joint, 1 for locked. (9.3RC1 format)

-1 to signify continuation in 9.3RC2 format

-2 to signify continuation in 10.0 format</td><td>Note: Format of this card changed between 9.3RC1 and RC2. And changed again with 10.0</td></tr>
<tr><td colspan="4"></td></tr>
</table>

<pos_rot>	E10.0	Permitted +ve rotation (degrees: 0 to +180.0)	**Note**: This continuation line is 9.3RC2 format & later only
<neg_rot>	E10.0	Permitted -ve rotation (degrees 0 to -180.0	
<curr_angle>	E10.0	Current rotation angle (degrees)	
<locked>	I10	0 for unlocked joint, 1 for locked.	
<a3_active>	I10	1 if Assembly #3 active	**Note**: these fields are only present from V10.0 onwards
<assy_3>	I10	Assembly #3	
<factor_1>	E10.0	Factor on <assy_1> motion	
<factor_2>	E10.0	Factor on <assy_2> motion	
<c1_x>	E10.0	Point 1 X coordinate	**Note**: This card is only present from V10.0 onwards, signified by <-2> in column 8 of the 1st card. From V10.0 onwards either or both locations may be defined either by a node or by an explicit position. If a node is defined it is used, regardless of any position <cx,cy,cz>, otherwise the stipulated position is used.
<c1_y>	E10.0	Point 1 Y coordinate	
<c1_z>	E10.0	Point 1 Z coordinate	
<c2_x>	E10.0	Point 2 X coordinate	
<c2_y>	E10.0	Point 2 Y coordinate	
<c2_z>	E10.0	Point 2 Z coordinate	

```
*CONNECTION_HINGE
<Title>
<assy_1>  <assy_2>  <node_1>     <node_2> <locked>/<-1>/<-2>
<pos_rot> <neg_rot> <curr_angle> <locked> <a3_active> <assy_3> <factor_1>
<factor_2>
<c1_x>    <c1_y>    <c1_z>       <c2_x>   <c2_y>      <c2_z>
```

<title>	A80	Optional title for connection	
<assy_1>	I10	Assembly #1	
<assy_2>	I10	Assembly #2	
<node_1>	I10	First node on line	
<node_2>	I10	Second node on line	
<locked> or <-1> or <-2>	I10	0 for unlocked joint, 1 for locked. (9.3RC1 format) -1 to signify continuation in 9.3RC2 format -2 to signify continuation in 10.0 format	**Note**: Format of this card changed between 9.3RC1 and RC2. And changed again with 10.0

<pos_rot>	E10.0	Permitted +ve rotation (degrees: 0 to +180.0)	**Note**: This continuation line is 9.3RC2 format only
<neg_rot>	E10.0	Permitted -ve rotation (degrees 0 to -180.0	
<curr_angle>	E10.0	Current rotation angle (degrees)	
<locked>	I10	0 for unlocked joint, 1 for locked.	
<a3_active>	I10	1 if Assembly #3 active	**Note**: these fields are only present from V10.0 onwards
<assy_3>	I10	Assembly #3	
<factor_1>	E10.0	Factor on <assy_1> motion	
<factor_2>	E10.0	Factor on <assy_2> motion	
<c1_x>	E10.0	Point 1 X coordinate	**Note**: This card is only present from V10.0 onwards, signified by <-2> in column 5 of the 1st card. From V10.0 onwards either or both locations may be defined either by a node or by an explicit position. If a node is defined it is used, regardless of any position <cx,cy,cz>, otherwise the stipulated position is used.
<c1_y>	E10.0	Point 1 Y coordinate	
<c1_z>	E10.0	Point 1 Z coordinate	
<c2_x>	E10.0	Point 2 X coordinate	
<c2_y>	E10.0	Point 2 Y coordinate	
<c2_z>	E10.0	Point 2 Z coordinate	

```
*POINT_NODE
<title>
<assembly id> <node id> <restrained DoFs> (<csys>)
```

<title>	A80	Title for the point
<assembly id>	I10	Label of assembly to which point is attached
<node id>	I10	Label of node from which point coordinates are taken
<restrained DoFs>	I10	A restraint code made up of any permutation of 123456, or 0 for none
<csys>	I10	Optional: a coordinate system to give restraint in local axes.

```
*POINT_LOCATION
<title>
<assembly id> <px> <py> <pz> <restrained DoFs> (<csys>)
```

<title>	A80	Title for the point
<assembly id>	I10	Label of assembly to which point is attached
<px>	E10.0	X coordinate of point
<py>	E10.0	Y coordinate of point

<pz>	E10.0	Z coordinate of point
<restrained DoFs>	I10	A restraint code made up of any permutation of 123456, or 0 for none
<csys>	I10	Optional: a coordinate system to give restraint in local axes.

***CHILD_DUMMY** *or* ***CHILD_MECHANISM** *(Card format is the same for both)*

```
<title>
<parent assy> <child label> <nslaved> <linked DoFs> <locked>
<child assy #1>
<child assy #2>
: : :
```

<title>	A80	Optional title
<parent assy>	I10	Label of "driving" assembly in parent mechanism
<child label>	I10	Label of "child" mechanism or dummy definition.
<nslaved>	I10	Number of assemblies in child that are "slaved" to this master
<linked DoFs>	I10	Any permutation of 123 giving degrees of freedom linking master to child.
<locked>	I10	1 if child is fully locked to master
<child assy *n*>	I10	Linked child assemblies 1 to <nslaved>, 1 per line.

***POSITION**

Any number of positions may be stored for a mechanism, and position information is identical for dummies and mechanisms. A description of these and their card format is given below.

***MECHANISM_END**

Terminates the mechanism definition.

APPENDIX IIc: "Positions" in Dummy and Mechanism data.

Both Dummies and Mechanisms may have any number of saved "positions" stored within their data cards. The format used is identical for both types, and all the information below applies equally to both.

Position card format

```
*POSITION
<title>
```

then for each assembly

```
<Assembly id #1>
<Cx> <Cy> <Cz>
<Xx> <Xy> <Xz>
<Yx> <Yy> <Yz>
<Zx> <Zy> <Zz>
```

... and so on for assemblies #2 to #n in this Dummy or Mechanism

<title>	A80	Title for position (required, must be unique)

<Assembly id>	I10	Assembly label
<Cx> <Cy> <Cz>	3E20.0	Notional centroid of assembly
<Xx> <Xy> <Xz>	3D20.0	X components of direction cosines
<Yx> <Yy> <Yz>	3D20.0	Y components of direction cosines
<Zx> <Zy> <Zz>	3D20.0	Z components of direction cosines

Data stored for Positions

As the card format above shows a "Position" stores a title, then a centroid and 3x3 set of direction cosines for each assembly in the Dummy / Mechanism. The direction cosines are written in double precision.

The first time a position is **Saved** an extra "reference position", the current configuration, is created and all user-defined positions follow this. The reference position is hidden from the user, and updated whenever an assembly is re-positioned.

Positions are relative within the Dummy / Mechanism

Positions are always stored relative to the "reference position", making it possible for **Retrieve** to restore any previous configuration.

This also means that "global" orientations of the whole Dummy / Mechanism are effectively cancelled out, making saved positions "local". For example if a saved position raises a left hand relative to the wrist this will still be the case even if the Dummy as a whole has been translated and rotated to some totally different location.

Re-using Position data when geometry changes

Position data generates a notional centroid, the average coordinate of the bounding box round an assembly, the first time a position is saved and thereafter this value is "remembered". Since this value is non-structural it does not matter if the geometry or content of an assembly are modified: this centroid remains unchanged and saved positions will still work. (Technically ill-conditioning could occur if the assembly's dimensions are made wildly different, but this is unlikely to happen in practice.)

However if nodes on the assembly used for **Connections** with its neighbours are moved it is possible that retrieving previously saved position will result in these **Connections** becoming separated. A moment's thought reveals why: the location of such a node in a previously saved position is unlikely to match up with the relevant connection position on the assembly on the other side of the connection.

Therefore the general rule is that saved positions will continue to function despite changes or additions/subtractions to assemblies so long as the locations of nodes used in connections are not modified.

Re-using Position data in different contexts

Dummies and Mechanisms can be slaved to masters as "children", whereupon position **Save** / **Retrieve** operations will operate on all assemblies in master and child(ren).

This is not a problem since positions are referenced purely by name, therefore:

- During a **Save** operation each assembly will acquire another entry of the specified name.

  Care should be taken not to use the same name in different contexts, leading to two or more positions for an assembly having the same name. While **Save** will work, a subsequent **Retrieve** may pick up the wrong position.

- During a **Retrieve** operation each assembly is scanned for a previously saved position of the given name. If no such position is found then that assembly's geometry is not altered.

As explained above if an assembly contains two or more positions of the same name the first encountered will be used, which may cause unexpected outcomes.

Positions are an attribute of assemblies, so those saved while positioning a Dummy or Mechanism in isolation will be "visible" and available for use when it is used as a "child", and vice-versa.

Editing position data by hand

Essentially don't try it!

When a position is first saved an extra "reference" position, not visible to the user, which contains the assembly's current configuration is saved first, then user-defined positions follow. This reference position is updated whenever the assembly is repositioned, and all transformations are stored relative to this.

In addition any **Orient** type transformations of the whole Dummy or Mechanism will result in all saved positions being updated accordingly.

It would be virtually impossible to track all these transformations by hand, and it is strongly recommended that the only hand-editing carried out on these data should be limited to total deletion of a position should this be required.

APPENDIX IId: The Dummy Angles File (.daf)

From release 9.3RC1 onwards PRIMER can read and write "Dummy Angle" files, extension .daf.

The following formatting rules apply:

- These are simple ASCII files, intended to be editable by hand if required.
- All lines starting with **$**, **%** or **#** are treated as comment lines.
- All numerical data is in free format, however each section of data should be on a single line.
- All angles are in degrees, and should be in the range +/-180.0

The file format is: (note that this format evolved during release 9.3 development, see Format Changes below for details)

<table>
<tr><td colspan="4">$ Dummy Angles File. Generated on <current date>
$ Model title: <current model title>

<... any number of comment lines starting with $, % or # ...>

These comment lines are remembered, and will be copied to any newly written file for this model.
All comment lines up to, but not including, the "<b>$ H-Point ..." line are saved.</td></tr>
<tr><td colspan="4">The H-Point location. Note that its "label" field is always zero.</td></tr>
<tr><td>$ H-Point</td><td>X position</td><td>Y position</td><td>Z position</td></tr>
<tr><td>0</td><td><X coord></td><td><Y coord></td><td><Z coord></td></tr>
<tr><td colspan="4">The "whole dummy" angles. Note that their label field is always -1.</td></tr>
<tr><td>$ Whole Dummy</td><td>X Angle</td><td>Y Angle</td><td>Z Angle</td></tr>
<tr><td>-1</td><td><Theta X></td><td><Theta Y></td><td><Theta Z></td></tr>
</table>

<table>
<tr><td colspan="4">Then each assembly's title, label and angles are listed.
• These do not have to appear in numerical order, although this is recommended for clarity.
• The title is a comment line for ease of reading: it is ignored when the file is read, and need not match the existing title of assembly <n>
• If an assembly does not have an angle specified in this file its current angle will not be changed when the file is read.</td></tr>
<tr><td colspan="4">$ Assembly #1 title</td></tr>
<tr><td>1</td><td><Theta X></td><td><Theta Y></td><td><Theta Z></td></tr>
<tr><td colspan="4">$ Assembly #2 title</td></tr>
<tr><td>2</td><td><Theta X></td><td><Theta Y></td><td><Theta Z></td></tr>
<tr><td colspan="4">And so on for all assemblies in the Dummy</td></tr>
</table>

Here is an example of an actual file:

```
$ Dummy Angles File. Generated on Wed Aug 27 13:46:49 2008
$ Model title: FT-ARUP HYBRID III 50TH - VERSION 5.1S2 (MM, TON, S)
$
$ The comments on this line, and all following up to the "H-Point"
$ line below will be saved and written out again.
$
$ H-Point X position Y position Z position
0 6.8459E+002 0.0000E+000 0.0000E+000
$
$ Whole Dummy X Angle Y Angle Z Angle
$
-1 0.0000E+000 1.0000E+001 1.8000E+002
$
$ Assembly X Angle Y Angle Z Angle
$
$ Lower Torso
1 0.0000E+000 0.0000E+000 0.0000E+000
$ Thorax
2 0.0000E+000 0.0000E+000 0.0000E+000
$ Head & Neck
3 0.0000E+000 4.9999E+000 0.0000E+000
$ Upper leg left
4 0.0000E+000 1.8107E+001 0.0000E+000
$ Upper leg right
5 0.0000E+000 0.0000E+000 0.0000E+000
$ Lower leg left
6 7.1490E-004 5.0000E+001 0.0000E+000
$ Lower leg right
7 5.2281E-004 2.7816E+001 0.0000E+000
$ Foot left
8 -1.6916E-003 1.7390E+001 0.0000E+000
$ Foot right
9 -1.6445E-003 0.0000E+000 -1.4572E-004
$ Yoke left
10 3.8573E-004 2.1671E-004 -5.0000E+001
$ Yoke right
11 -3.8573E-004 -1.5796E-004 -5.0000E+001
$ Upper arm left
12 1.2956E-004 -1.5649E-004 1.0004E-001
$ Upper arm right
13 -4.7659E-004 2.2656E-004 -1.5000E+001
$ Elbow left
14 3.8414E-004 -1.4245E-004 5.5000E+001
$ Elbow right
15 0.0000E+000 0.0000E+000 5.5000E+001
$ Lower arm left
```

```
16 6.8736E-004 0.0000E+000 9.2774E+001
$ Lower arm right
17 -1.0686E-004 -4.4135E-004 6.1678E+001
$ Wrist left
18 -1.4033E-004 -1.2171E-003 -9.0001E+001
$ Wrist right
19 2.6753E-004 0.0000E+000 -9.0000E+001
$ Hand left
20 -4.7274E-004 0.0000E+000 -1.9801E+001
$ Hand right
21 0.0000E+000 1.6804E-004 2.3132E+001
$ <End of file>
```

Format changes during V9.3 development

The format of this file evolved during the development of PRIMER 9.3 as described below.

Reading of earlier formats is automatic and no user intervention is required to read a 9.3RC1 or RC2 format .daf file into release 9.3, however if you propose to hand-edit older files you may need to consider the information below.

Direction cosines instead of Euler angles in 9.3RC1

The original format of this file, in 9.3RC1, used direction cosines instead of Euler angles to record assembly orientations; however the Euler angles were written as comment lines above these cosines. PRIMER 9.3 will read these files, but will write the new format described above using Euler angles.

If you have such an "old" file it is recommended that you read it into release 9.3 and write it out again immediately to convert it to the current format.

"Whole dummy" angles not present in 9.3RC2

PRIMER 9.3RC2 used Euler angles as described in the format above, but did not include the "whole dummy" orientation angles. These have been added in release 9.3 using the "label" -1.

If this line is omitted PRIMER assumes that no "whole dummy" orientation is required, making the change backwards-compatible.

APPENDIX III: Origami "tree" file example

The following is an Origami "tree" file example

```
*END
$
$
$ ============
$ ORIGAMI data
$ ============
$
$
*ORIGAMI_START
         1Origami 1
         1         2
$
*AXES
         1
$
*OPTIONS
        80         2         2         2         1
         2        -1         4         5         6         3         7
 1000000.0       1.1       0.9       0.0
$
$ =============
$ List of FOLDs
$ =============
$
$ LINE 1 (Basic data)
$     FIELD 1: LABEL
$     FIELD 2: TYPE
$              =0: NULL
$              =1: THIN
$              =2: THICK
$              =3: TUCK
$              =4: SPIRAL
$              =5: SCRUNCH
$              =6: ALIGN
$     FIELD 3: UPDOWN
$              =0: UP
$              =1: DOWN
$     FIELD 4: RIGHTLEFT
$              =0: RIGHT
$              =1: LEFT
$     FIELD 5: REFERENCE COORDINATE    SET FLAG
$     FIELD 6: LAYERS FOR TUCK FOLD
$     FIELD 7: SUBSET FOLDING
$     FIELD 8: CREATE ALIGN FOLD TRAM    LINES
$
$ LINE 2 ( Reference nodes)
$     FIELD 1: FOLD_NODE
$     FIELD 2: TUCK_ZSPLIT_N1
$     FIELD 3: TUCK_ZSPLIT_N2
$     FIELD 4: LAYER_ZMIN_N1
$     FIELD 5: LAYER_ZMIN_N2
$     FIELD 6: LAYER_ZMAX_N1
$     FIELD 7: LAYER_ZMAX_N2
$
$ LINE 3 (Sets, etc.)
$     FIELD 1: NODES_LEFT
$     FIELD 2: NODES_CENTER
$     FIELD 3: NODES_RIGHT
$     FIELD 4: SHELL_LEFT
$     FIELD 5: SHELL_CENTER
$     FIELD 6: SHELL_RIGHT
$     FIELD 7: TUCK FOLD TYPE
$
$ LINE 4 (Positions, etc.)
$     FIELD 1: THICKNESS
```

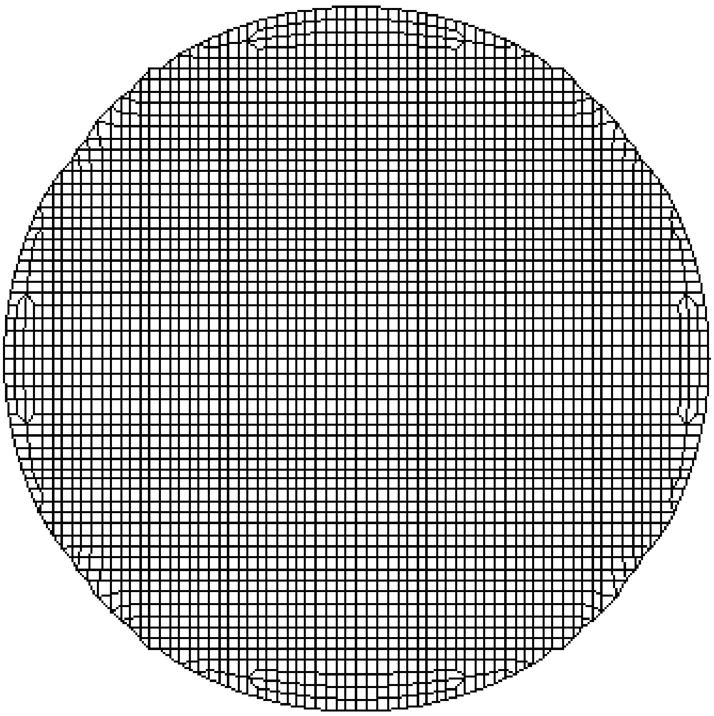
```
$     FIELD 2: FOLD_XPOS
$     FIELD 3: FOLD_XTOL
$     FIELD 4: INPLANE_ANGLE
$
$ LINE 5 (Fold specific data)
$     FIELD 1: Factor for unused portion    of spiral.
$     FIELD 2: Out-of-plane fold angle.
$     FIELD 3: Scale factor for fold    point separation.
$     FIELD 4: Location of ZSPLIT    for tuck.
$
$ LINE 6 (Layering and align data)
$     FIELD 1: Minimum value for layer.
$     FIELD 2: Maximum value for layer.
$     FIELD 3: Tramline offset distance    for align.
$
$ LINE 7 (Reference point.)
$     FIELD 1: X.
$     FIELD 2: Y.
$     FIELD 3: Z.
$
$ LINE 8 (Local X vector.)
$     FIELD 1: X.
$     FIELD 2: Y.
$     FIELD 3: Z.
$
$ LINE 9 (Vector in X-Y plane.)
$     FIELD 1: X.
$     FIELD 2: Y.
$     FIELD 3: Z.
$
$*FOLD
         1         3         0         0         1         0         0         0
       314         0         0         0         0         0         0
         2         2         2         1         1         1         0
                 5.0           114.50000                 0.0                 0.0
          0.10000000               180.0                 1.0                 0.0
      -1.0000000E+20       1.0000000E+20                 0.0
                 0.0                 0.0                 0.0
                 1.0                 0.0                 0.0
                 0.0                 1.0                 0.0
$
*FOLD
         2         3         0         1         1         0         0         0
      1729         0         0         0         0         0         0
         2         2         2         1         1         1         0
                 5.0          -114.50000                 0.0                 0.0
          0.10000000               180.0                 1.0                 0.0
      -1.0000000E+20       1.0000000E+20                 0.0
                 0.0                 0.0                 0.0
                 1.0                 0.0                 0.0
                 0.0                 1.0                 0.0
$
*FOLD
         3         1         0         1         1         0         0         0
      2343         0         0         0         0         0         0
         2         2         2         1         1         1         0
                 5.0          -36.750000                 0.0                90.0
          0.10000000               180.0                 1.0                 0.0
      -1.0000000E+20       1.0000000E+20                 0.0
                 0.0                 0.0                 0.0
                 1.0                 0.0                 0.0
                 0.0                 1.0                 0.0
$
$ ===============
$ List of ORIENTs
$ ===============
$
$ LINE 1
$     FIELD 1: LABEL
$     FIELD 2: TYPE
$              =0: TRANSLATION
```

```
$               =1: ROTATION
$               =2: SCALE
$      FIELD 3: TRANSLATE/ROTATE TYPE
$               =0: X
$               =1: Y
$               =2: Z
$               =3: vector
$               =4: N1->N2
$      FIELD 4: SCALE TYPE
$               =0: X,Y,Z
$               =1: N1,N2,N3
$      FIELD 5: N1
$      FIELD 6: N2
$      FIELD 7: N3
$      FIELD 8: CENTRE NODE
$
$ LINE 2
$      FIELD 1: TRANSLATE DISTANCE    TYPE
$               =0: MAGNITUDE OF VECTOR
$               =1: USER DEFINED
$      FIELD 2: USER DEFINED DISTANCE
$      FIELD 3: ROTATE/SCALE CENTRE    TYPE
$               =0: GLOBAL AXIS
$               =1: COORDINATE
$               =2: NODE
$               =3: N1
$      FIELD 4: CENTRE[X]
$      FIELD 5: CENTRE[Y]
$      FIELD 6: CENTRE[Z]
$      FIELD 7: ANGLE
$
$ LINE 3
$      FIELD 1: VECTOR[X]
$      FIELD 2: VECTOR[Y]
$      FIELD 3: VECTOR[Z]
$      FIELD 4: SCALE[X]
$      FIELD 5: SCALE[Y]
$      FIELD 6: SCALE[Z]
$
$
*ORIENT
         1         0         0         0         0         0         0         0
         0     100.0         0       0.0       0.0       0.0       0.0
       1.0       0.0       0.0       1.0       1.0       1.0
$
*ORIENT
         2         1         0         0         0         0         0      5726
         0       0.0         2       0.0       0.0       0.0      45.0
       1.0       0.0       0.0       1.0       1.0       1.0
$
*ORIGAMI_END
```

It is strongly recommended that you don't attempt to edit Origami files by hand, as it can be very hard to identify exactly what the individual numbers mean. To change folds or orients read them back into PRIMER and edit them there.

Also, try not to separate Origami definitions from their parent input decks: they reference SET and other entities within these decks, and confusion will arise if these labels are not treated consistently.

APPENDIX IV: Airbag Folding example



The following example is of a somewhat simplified geometry of an airbag. This is designed to provide a fairly complete demonstration of the capabilities to make a folded airbag. It is not designed to represent a realistic fold pattern or to represent accurate deployment of an airbag. In fact, there is no *AIRBAG card in this model so it will not deploy.

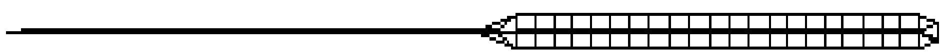
Figure A4.1 shows the starting geometry for this model. The units are in millimetres and the fabric is 0.25mm thick. The airbag is a simple drivers side (or pancake) airbag.

For reference, the model is provided with the PRIMER program and is entitled "airbag_folding_example.key". This should be available in the PRIMER examples directory.

The first issue in folding this model is to create an ORIGAMI. This is done by selecting **DEFINE_ORIGAMI** and then **CREATE.** When choosing the materials, it is important to select only the materials which are to be folded. In this case the whole model is wanted for folding so **WHOLE MODEL** can be used to select the entire model. **SELECT** the ORIGAMI and press **SET_FOLD** to start folding the airbag.

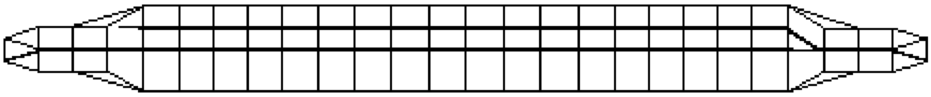
The folding pattern consists of 9 folds. The folds are:

1. Tuck fold along x-axis
2. Tuck fold along x-axis (interferes with first fold)
3. Thin fold along y-axis
4. Thin fold along y-axis using subset folding
5. Thin fold along y-axis using subset folding
6. 90 thick fold along y-axis
7. Align fold using tramlines
8. 90 thin fold along y-axis
9. Spiral fold using a local coordinate system and layers



The first fold is a tuck fold. The fold is in the x direction and in the xy plane (the default folding plane). The fold point is at 114.5 and the direction is from right to left. The fold point is defined by selecting a node using the FOLD_POINT button. By default the folder chooses all of the airbag to the right of the fold line. As this is what we want this is OK. Figure A4.2 shows a side view of the airbag after the tuck fold. In this example the fold separation has been set very high (5.0mm) so you can see the tuck fold. In reality the separation would be much smaller (probably the same order as the fabric thickness).

The second fold is defined in exactly the same way except that the direction is from left to right and the fold point is at -114.5. By default the folder chooses all of the airbag to the left of the fold line which is what is required.



A side view after FOLD 2 is shown in Figure A4.3. As the two tuck folds interfere with each other the first tuck fold has been moved so that no penetrations occur. As the fold separation is very large this effect has been exagerated. In reality the amount would be much smaller.

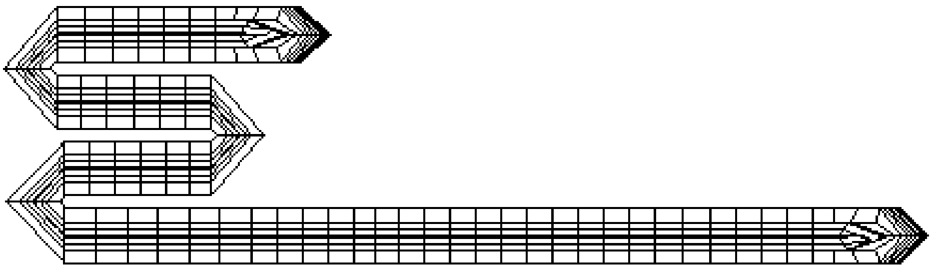
Folds 3 to 5 will show how you can use subset folding to quickly fold an airbag. Fold 3 is a thin fold. We want to fold it in the y direction soa 90 fold angle needs to be selected. Figure A4.4 shows the airbag after this fold has been done.

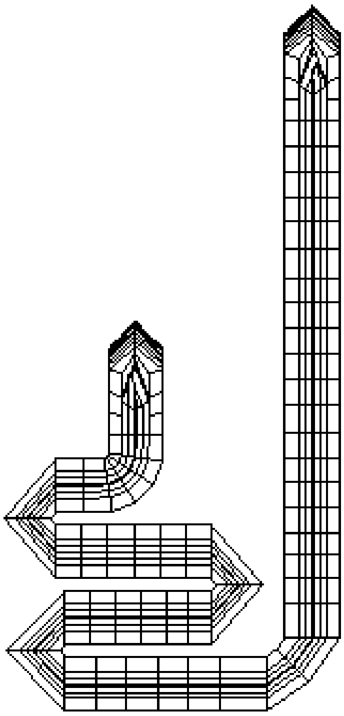
The fourth fold is a thin fold in the opposite direction. By default the folder will fold all of the airbag which is on one side of the fold line to the other side. In this case we only want to fold the top layer of the airbag in figure A4.4. We have three ways of performing this fold.

1. Defining a set to fold rather than the whole origami
2. Using layers to select a specific vertical range of the airbag to fold.
3. Using subset folding.

Subset folding is the easiest option to use. We can use this because all the nodes which we want to fold in this fold (4th fold) have been folded in the previous fold (fold 3). i.e these folds are a subset of the previous fold nodes. Press the **SUBSET FOLDING** button. The node selection should autoamtically update to the folds we need. The previous fold was in the left to right direction. This fold needs to be right to left.

The 5th fold is done in exactly the same way. As we are already using subset folding everything (including the fold direction) will be set correctly. Figure A4.5 shows the airbag after the first 5 folds.





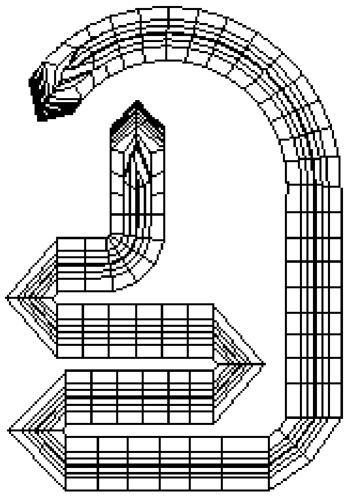
Fold 6 can also be done with subset folding. This is to show that subset folding is not just for thin folds. It also works for thick folds. We only want to fold this by 90 instead of 180. This is easily changed by using **THICK FOLD OPTIONS** and changing the angle. Apart from this complication the process is identical to folds 3 to 5.

Fold 7 is an **ALIGN** fold. This is used if your nodes are not exactly where you want them to be. In this case we want to make adjacent nodes to the fold line a constant distance of 5mm from the fold line. This is done by setting the fold point as normal. As we were using subset folding in the previous folds we need to make sure that it is turned off as these folds are no longer a subset of the previous fold. To set the align fold options use **ALIGN FOLD OPTIONS** and make sure that the **MAKE TRAMLINES** option is set with a constant distance of 5mm.

Fold 8 is a thin fold of 90. This presents no real problems and is similar to previous fold definitions. Figure A4.6 shows the airbag after the first 8 folds have been done.

The last fold is a spiral fold. This presents problems because the part of the airbag we want to fold does not lie on the xy plane which is the default folding plane for the folder. This can be overcome by defining a local coordinate system for this fold. To do this press **DONE** to exit the **SET_FOLD** menu and return to the main folding window. **LOCAL_SYSTEM** allow you to define a new folding plane. We define a plane which is in the global XZ axis. This is done by selecting N1 somewhere on the vertical part of the airbag where we want to do the spiral fold. N2 is selected so that local x points in the global Z direction. N3 is now chosen so that the 3 points give the local xy plane.

This allows us to define the plane for the spiral fold but if we try to fold this we will fold other parts of the airbag which are on the same side of the fold line. We need to either define a subset of the bag to fold or use layers to specify a range of the airbag to fold. In this case it is easier to define the **UPPER LAYER** so that anything above that local z coordinate is discarded when folding.



Once the airbag has been folded it can be positioned to the desired location using the **POSITION FOLDED BAG** option. This example model has 4 orientations stored with the origami so that the airbag can be repositioned as needed. They can be modified or deleted easily or new orientations can be added just like folds. The 4 orientations that are stored are:-

1. Translation, 100mm along the X axis
2. Rotation of 45 about X, centred on node 5726
3. Translation along vector using N1->N2 method with distance 50mm
4. A scale in a local coordinate system of 0.8 in the Y direction only.

As all of the parameters are already stored in the example, airbag_folding_example.key, it will not necessarily amount to much experience gained from looking at this file. The best way is to make a copy; edit the file; and remove all of the ***ORIGAMI, *FOLD** and ***ORIENT** definitions at the end of the file. Working from this, it should be possible to learn many of the important folding parameters.

APPENDIX V: Seatbelt "Tree" File Structure

A seatbelt definition is not a special file, rather it is an ordinary LS-DYNA input deck to which an extra section has been added after the ***END** card which allows PRIMER to recognise and manipulate seatbelt data.

The description "tree" file is misleading, although it is adopted for historical reasons, since there is no hierarchy as such. Rather the file contains extra information about belt geometry, fitting, meshing and contact, and it is organised as follows.

```
*BELT_START
<label>       <Title>
<Solid set>   <Shell set>   <Tshell Set>   <Segm set>
<#rows>       <width>       <length>       <thickness>
<iter>        <conv_tol>    <tk_flag>      <tk_scale> <pen_dist> <overlap>
<proj_dist> <curve_ang>
```

<table>
<tr><td>Label</td><td>I10</td><td colspan="2">The label of this belt definition</td></tr>
<tr><td>Title</td><td>A70</td><td colspan="2">The title of this definition (optional)</td></tr>
<tr><td colspan="4"></td></tr>
<tr><td>Solid set</td><td>I10</td><td>A set of solid elements defining "structure"</td><td rowspan="3">The "structure" can be any combination of these sets, and defines the dummy and vehicle elements contacted by the seatbelt. Not all sets need to exist, but at least one is required to give a structure definition.</td></tr>
<tr><td>Shell set</td><td>I10</td><td>A set of shell elements ditto</td></tr>
<tr><td>Tshell set</td><td>I10</td><td>A set of thick shell elements ditto</td></tr>
<tr><td>Segm set</td><td>I10</td><td colspan="2">The set of segments that PRIMER creates from the three element sets above, used for contact between belt and structure during fitting.</td></tr>
<tr><td colspan="4"></td></tr>
<tr><td>#rows</td><td>I10</td><td colspan="2">The number of rows of 2D elements across the belt. This must lie in the range 0 (for all 1D belt elements) to 20. Default = 1</td></tr>
<tr><td>width</td><td>E10.0</td><td colspan="2">The overall belt width. Each 2D element wuill be <width> / <#rows> wide. Default = 50.0</td></tr>
<tr><td>length</td><td>E10.0</td><td colspan="2">The characteristic length of each belt element. Default = 25.0</td></tr>
<tr><td>thickness</td><td>E10.0</td><td colspan="2">The thickness of 2D belt elements. Default = 1.0</td></tr>
<tr><td colspan="4"></td></tr>
<tr><td>iter</td><td>I10</td><td colspan="2">The number of fitting iterations between contact bucket resorts. Default = 25</td></tr>
<tr><td>conv_tol</td><td>E10.0</td><td colspan="2">The convergence tolerance at which fitting halts. Default = 1.0e-5</td></tr>
<tr><td>tk_flag</td><td>I10</td><td colspan="2">Thickness used during fitting: 0 (default) = use true thickness; 1 = use true thickness * factor; 2 = use neutral axis (no thickness)</td></tr>
<tr><td>tk_scale</td><td>E10.0</td><td colspan="2">Factor used when <tk_flag> = 1. Default = 1.0</td></tr>
<tr><td>pen_dist</td><td>E10.0</td><td colspan="2">Maximum penetration distance considered for contact into solid & thick shell elements. Default = 25.0, but also limited to shortest side length * 0.4.</td></tr>
<tr><td>overlap</td><td>E10.0</td><td colspan="2">Fraction by which facets are extended during contact checking to stop nodes "falling into gaps". Default = 0.05</td></tr>
<tr><td>proj_dist</td><td>E10.0</td><td colspan="2">Initial projection distance by which belt path is "thrown outwards" at start of fitting. Default = 35.0</td></tr>
<tr><td>curve_ang</td><td>E10.0</td><td colspan="2">Maximum permitted transverse belt curvature in degrees. Default = 0.0, meaning no limit, permitted values are 0 to 180 deg.</td></tr>
</table>

***BELT_MESH**

```
<node_set>    <nsbo_set>    <elem_set>
<sbelt_f>     <sbelt_l>     <retr_f>     <retr_l>     <slip_f>     <slip_l>
<nrb_set_f>   <nrb_set_l>
<pid_1d>      <slen_1d>
<pid_sh>      <t1_sh>       <t2_sh>      <t3_sh>      <t4_sh>      <psi_sh>
<pid_2d>      <t1_2d>       <t2_2d>      <t3_2d>      <t4_2d>      <psi_2d>
```

node_set	I10	Set of all nodes in seatbelt	These sets are only created if the option to generate a contact for the belt is used. They define the contact between the belt and the structure, the latter being defined via segment set <segm set> on the *BELT_START card.	
nsbo_set	I10	Set of nodes on 1D seatbelt elements only		
elem_set	I10	Set of shell or 2D seatbelt elements		
sbelt_f	I10	First 1D seatbelt element id	Since there is no *SET_SEATBELT card in LS-DYNA the seatbelt fitter stores the first and last 1D belt element ids, assumes that elements are contiguous in this range and that it "owns" all these elements.	
sbelt_l	I10	Last 1D seatbelt element id		
retr_f	I10	First retractor id	Likewise it assumes that the list of retractors is contiguous between first and last, and that it owns all of these.	This is true for both 1D and 2D slipring and retractor elements.
retr_l	I10	Last retractor id		
slip_f	I10	First slipring id	Likewise it assumes that the list of sliprings is contiguous between first and last, and that it owns all of these.	
slip_l	I10	Last slipring id		
nrb_set_f	I10	First nodal rigid body set id	There is no *SET_NODAL_RIGID_BODY card, so it is assumed that this list is contiguous, as above, and that the belt "owns" all these definitions.	Belt fitting in PRIMER pre-dates the optional Nodal RB labels in LS-DYNA, so set ids are used instead for backwards compatibility.
nrb_set_l	I10	Last nodal rigid body set id		
pid_1d	I10	The part ID for any 1D seatbelt elements		Only need to be defined if the belt contains 1D belt elements
slen_1d	E10.0	The initial slack length for any 1D seatbelt elements		
pid_sh	I10	The part id of any shell elements		Only need to be defined if the belt contains any conventional shell elements
t1_sh *to* t4_sh	4E10.0	***Optional*** thicknesses at the 4 nodes of these elements		
psi_sh	E10.0	***Optional*** orthotropic element angle for these elements		

pid_2d	I10	The part id of any 2D seatbelt elements	Only need to be defined if the belt contains any 2D seatbelt shell elements
t1_2d *to* t4_2d	4E10.0	***Optional*** thicknesses at the 4 nodes of these elements	
psi_2d	E10.0	***Optional*** orthotropic element angle for these elements	

```
*BELT_CONTACT
<no_su_cont>   <su_su_cont>
```

no_su_contact	I10	Label of Nodes to Surface contact used when contact between nodes on 1D belt elements and dummy structure is defined.	This card is only used if the option to create a contact between belt and dummy "structure" has been used.
su_su_contact	I10	Label of Surface to Surface contact used when contact between belt shells / 2D belt elements and dummy structure is defined.	

```
*BELT_PATH
<npts>
<bits>       <x1>          <y1>          <z1>     (<nid>)
(Optional row 1 of further data depending on <bits>)
(Optional row 2 of further data depending on <bits>)
 : : : :
<bits>       <xn>          <yn>          <zn>     (<nid>)
(Optional row 1 of further data depending on <bits>)
(Optional row 2 of further data depending on <bits>)
```

npts	I10	Number of points in the basic "line" belt path	
There then follow <npts> cards of the following data:			
bits	I10	Bitwise encoded integer containing information about the attributes of this point	This encoding is internal to PRIMER
X, Y, Z	3E10.0	X,Y,Z coordinates of point <n>	Either or both of [X,Y,Z] and <nid> may be defined. However if <nid> ***is*** defined its coordinates are used and [X,Y,Z] is ignored.
nid	I10	***Optional***: Node at point <n>	
Optional row #1	I10, 3E20.0	Encoded data defining belt orientation at this point	These two rows of data are only written if the twist at this belt point has been controlled during belt fitting. Row #1 defines the twist on side A, and row #2 on side B.
Optional row #2	I10, 3E20.0		

The format of the *BELT_PATH row entries is complex, involving internal bit fields designating retractor, slipring, etc. It may also contains normal information path twist that is similarly bit-packed into data words. These formats are internal to PRIMER and may change in subsequent releases.

It is ***strongly*** recommended that you do not attempt to edit these values.

```
*BELT_END
```

Users should not attempt to edit this file by hand unless they are very certain about what they are they doing.

Deleting this file from an input deck will not affect its behaviour during analysis, but will cause PRIMER to lose the ability to remesh the belt.

APPENDIX VI: Format translation during MODEL> READ

NASTRAN Bulk data file format

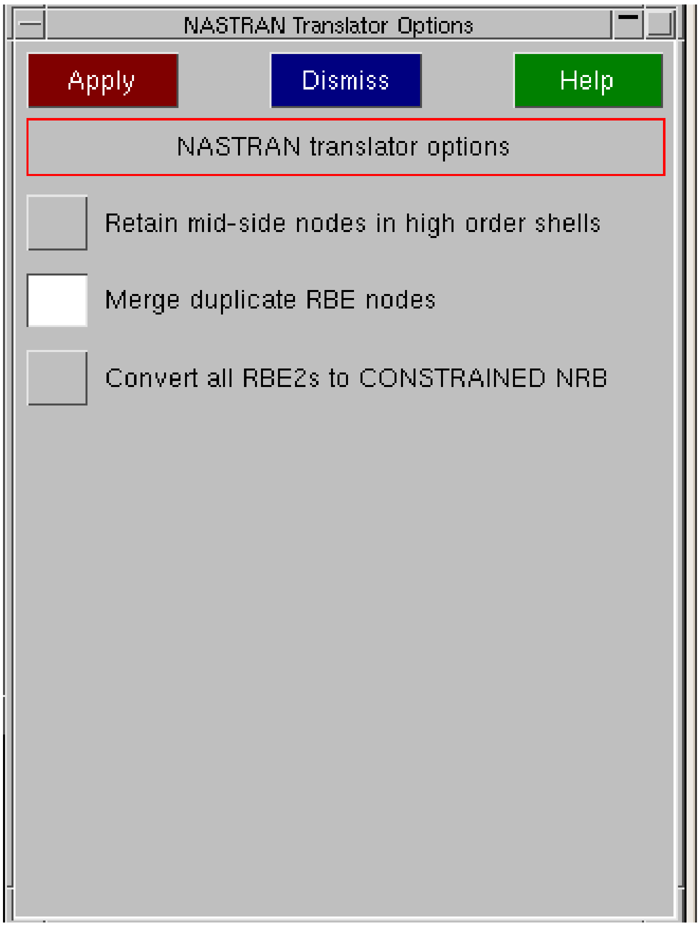
The following table shows the Nastran keywords supported, and how they are translated to internal LS-DYNA data when read in.

Nastran keyword	**Internal LS-DYNA keyword**
CBAR (+BAROR) **CBEAM (+BEAMOR)** **CROD** **CTUBE**	***ELEMENT_BEAM** (+3rd nodes as required)
CDAMP1 **CDAMP2** **CELAS1** **CELAS2** **CVISC**	***ELEMENT_DISCRETE** (with appropriate attributes as deduced from PELAS, PDAMP and PVISC cards).
CHEXA **CPENTA** **CTETRA**	***ELEMENT_SOLID**
CONM2	***ELEMENT_MASS** *or* ***ELEMENT_INERTIA**
CORD1R	***DEFINE_COORDINATE_NODES**
CORD2R	***DEFINE_COORDINATE_SYSTEM**
CQUAD4 **CQUAD8** **CQUADR** **CTRIA3** **CTRIA6** **CTRIAR**	***ELEMENT_SHELL**
FORCE	***LOAD_NODE**
GRID	***NODE** (includes restraints if specified)
INCLUDE	***INCLUDE**
MAT1	***MAT_ELASTIC**
MAT8	***MAT_ENHANCED_COMPOSITE_DAMAGE**
PBAR **PBEAM** **PROD** **PTUBE**	***PART** ***SECTION_BEAM**
PBARL (Currnetly "BAR", "BOX", "BOX1", "I", "I1", "ROD" and "TUBE" TYPEs are supported.)	***SECTION_BEAM** ***PART**
PCOMP	***PART_COMPOSITE**
PDAMP **PVISC**	***PART** ***SECTION_DISCRETE** ***MAT_DAMPER_VISCOUS**

PELAS	***PART** ***SECTION_DISCRETE** ***MAT_SPRING_ELASTIC**
PLOAD	***LOAD_SHELL**
PLOTEL	***ELEMENT_PLOTEL**
PSHELL	***PART** ***SECTION_SHELL**
PSOLID	***PART** ***SECTION_SOLID**
RBAR **RBE1** **RBE2**	***CONSTRAINED_NODAL_RIGID_BODY** *or* ***CONSTRAINED_SPOTWELD** (RBE2 only.) (+sets of nodes [***SET_NODE**] as required)
RBE3	***CONSTRAINED_INTERPOLATION**
SPC **SPC1**	***BOUNDARY_SPC**
TEMP	***LOAD_THERMAL_CONSTANT_NODE**

Notes:

1. NASTRAN **RBE2** cards with only two fully constrained nodes are translated to LS-Dyna ***CONSTRAINED_SPOTWELD** by default. If these two nodes are not fully constrained, then it is convereted to LS-Dyna ***CONSTRAINED_NODAL_RIGID_BODY**. Now there is a new option **"Convert all RBE2s to CONSTRAINED NRB"** available while reading the NASTRAN input file, which allows all two noded NASTRAN **RBE2** cards to be converted to LS_Dyna ***CONSTRAINED_NODAL_RIGID_BODY** cards. For this option there is a preference **primer*convert_rbe2_cnrb:**also, which is set FALSE by default.



2. Continuation characters are fully supported by the NASTRAN input translator. Even those cards that span multiple lines but do not contain explicit continuation characters are now translated properly.
3. Include files are fully supported by the input translator, and the include file structure is preserved in memory after the translation process is complete. Hence, if the model is written out in a format that supports include files, the resulting model will be written out across include its corresponding include files.
4. Both SMALL and WIDE format cards are supported by the input translator.
5. The "extra" data on certain cards that are defined in the LS-DYNA keyword manual under **`*TRANSLATE_NASTRAN`** (cards **`CELAS1, PSHELL, PSOLID`**) is not supported by the PRIMER translator. These values will be ignored if found.

I-DEAS Universal File Reader

The I-DEAS Universal file reader reads and processes the following I-DEAS Universal file modules.

Module Number	**Contents**	**Data Processed**
151	Title	
755	Restraints	Translational and rotation restraints in the global axis system.
773, 1710 & 1714	Materials	ID and Name
776	Beam Cross Sections	ID and Name (see below for more details)
780 & 2412	Elements	(see below for more details)
781 & 2411	Nodes	ID, Global Coordinates
789, 2437, 2448	Physical Sections	ID and Name (see below for more details)

Element Types

I-DEAS TYPE	**Colour**	**LS-DYNA Type**	**Additional Data / Comments**
21	Any	2 Noded Beam	Beam Orientation Node
91	Any	3 Noded Shell	
94	Any	4 Noded Shell	
101	Any	6 Noded Thick Shell	
104	Any	8 Noded Thick Shell	
111	Any	4 Noded Solid	
112	Any	6 Noded Solid	
115	Any	8 Noded Solid	
122	Cyan	*CONSTRAINED_RIVET	Only first 2 nodes processed
122	Red	*CONSTRAINED_SPOTWELD	Only first 2 nodes processed
122	Yellow	*CONSTRAINED_NODAL_ RIGID_BODY	I-DEAS element ID is used as the LS-DYNA node set ID.
122	Green	*CONSTRAINED_NODE_SET	I-DEAS element ID is used as the LS-DYNA node set ID
122	Blue	*CONSTRAINED_GENERALIZED_ WELD_SPOT	I-DEAS element ID is used as the LS-DYNA node set ID

136	Any	Translational Spring	
137	Any	Rotational Spring	
138	Any	Grounded Translational Spring	
139	Any	Grounded Rotational Spring	
141	Any	Translational Damper	
161	Any	Lumped Mass	

Physical Sections

IDEAS TYPE	LS-DYNA Type	LS-DYNA Data Processed
90 (Thin Shell)	*SECTION_SHELL	ID, Name, Shell thickness
100 (Thick Shell)	*SECTION_TSHELL	ID, Name
110 (Solid)	*SECTION_SOLID	ID, Name
133 (Translational Spring)	*SECTION_DISCRETE	ID, Name
134 (Rotational Spring)	*SECTION_DISCRETE	ID, Name
161 (Lumped Mass)	*ELEMENT_MASS	Mass

Beam Cross Sections

I-DEAS TYPE	LS-DYNA Type	LS-DYNA Data Processed
0 (Keyed In)	Belytschko-Schwer	Area, Iss, Itt, Irr , Shear Area = Area
1 (Rectangular)	Hughes-Lui (Rectangular)	TS1, TS2, TT1, TT2
3 (Circular)	Hughes-Lui (Circular)	TS1, TS2, TT1, TT2
4 (Pipe)	Hughes-Lui (Circular)	TS1, TS2, TT1, TT2
999 (General)	Belytschko-Schwer	Area, Iss, Itt, Irr, Shear Area = Area

In I-DEAS an elements properties are defined by both a set of Material data and a set of Physical. In LS-DYNA an elements properties are defined by a single PART number which then references a Material and a Section. When importing a Universal file PRIMER will automatically generate PARTS for all the combinations of Materials and Physicals used in the universal file. The ID's assigned to each PART can be made equal to either the elements Material ID or the elements Physical (Section) ID.

If PART numbers are based on Material ID and a Material is used with more than one Physical (Section) then the numbering scheme described above can not be followed. If this is the case then the PART using the Material which has the lowest Physical (Section) ID will be assigned the and ID equal to the Material ID and any other PARTs using the same material will be given ID's greater then the highest PART ID. Similarly if PART numbers are based on Physical (Section) ID and a Section is used with more than one Material then unique PART numbers will be generated for each PART.

The following table shows the difference in basing the PART number on the material or section property ID. Note that as there is only a clash with the material numbers then the option of basing the PART number on the section ID still results in the PARTS being given ID's the same as the section properties.

Element ID	**Material ID**	**Section ID**	**Part Number based on**	
			Material ID	**Section ID**

1	1	101	1	101
2	1	102	4	102
3	1	103	5	103
4	1	104	6	104
5	2	201	2	201
6	2	202	7	202
7	3	301	3	301

SPRING ELEMENTS

In I-DEAS spring elements are only assigned a Physical (Section) ID. PRIMER assumes that the Material ID of a spring element is the same as the Physical (Section) ID. Spring elements should NOT therefore be defined using Physical (Section) ID's that are the same as Material ID's used by other element types.

BEAM ELEMENTS

In I-DEAS the section properties for beam elements are defined using a Beam Cross Section. This means it is possible for beams to have the same Material and Physical ID's but to have different Beam Sections. When translating beam elements for I-DEAS PRIMER ignores the Physical ID and uses the Beam Cross Section ID as the LS-DYNA section ID. Beam Cross Sections should NOT therefore be defined using the same ID's as Physical Sections used by element types that are not beams.

ABAQUS Input file format

The following table shows the Abaqus keywords supported, and how they are translated to internal LS-DYNA data when read in.

Abaqus keyword	**Internal LS-DYNA keyword**
`*BEAM GENERAL SECTION` (Currently only " SECTION=GENERAL" supported.)	**`*SECTION_BEAM`** (ELFORM = 12)
`*BEAM SECTION` (Currently only "SECTION=CIRC" supported.)	**`*SECTION_BEAM`** (ELFORM = 1)
`*CONTACT PAIR` (Currenlty only supproted for optional parameter SMALL SLIDING) **`*SURFACE`** **`*SURFACE INTERACTION`**	**`*CONTACT_NODES_TO_SURFACE`**
`*DISTRIBUTING COUPLING`	**`*CONSTRAINED_INTERPOLATION`**
`*ELEMENT`	**`*ELEMENT`** (Currently only beam elements (TYPE=B31), shell elements and 8 nodes solid elements supported)
`*ELSET`	**`*SET_SHELL`** *or* **`*SET_SOLID`**
`*HEADING`	**`*TITLE`**
`*KINEMATIC COUPLING`	**`*CONSTRAINED_INTERPOLATION`**

***MATERIAL** ***DENSITY** ***ELASTIC**	***MAT_ELASTIC**
***MPC** (Currently only BEAM type supported.)	***CONSTRAINED_NODAL_RIGID_BODY**
***NODE**	***NODE**
***NSET**	***SET_NODE**
***PLASTIC**	***MAT_PIECEWISE_LINEAR_PLASTICITY**
***SHELL SECTION**	***SECTION_SHELL**
***SOLID SECTION**	***SECTION_SOLID**

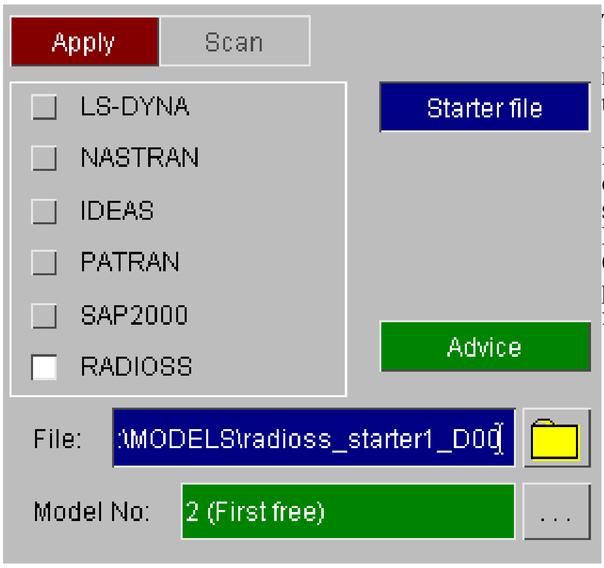
Patran Neutral File (.ntl) Reader

The following modules from Patran "neutral" files up to Patran level 2.5 format are read. All others are ignored.

Patran Module id	**Data type**	**Stored as**
25	Analysis title	Analysis title
1	Nodes	Nodes
2	Elements	Shape <iv> = 2 (bar), #nodes = 2 • <config> 0, 1, 2 become a beam element • <config> 5 becomes extra nodes on rigid parts • <config> 7 becomes a lumped mass • <config> 8 becomes a spherical joint • <config> 30 becomes a seatbelt element • <config> 31 becomes a retractor • <config> 32 becomes a slipring The following configurations becomes springs/dampers, grounded if <n1> = <n2>: • <config> 6, 10 becomes a translational spring • <config> 11 becomes a rotational spring • <config> 20 becomes a translational damper • <config> 21 becomes a rotational damper Shape <iv> = 2 (bar), #nodes > 2 • If #nodes = 4 become revolute joints • If #nodes = 6 become translational joints. Shape <iv> = 3 (tria): • <config> 0, 3 become 3 noded shells (n3 = n4) Shape <iv> = 4 (quad): • <config> 0, 3 become 4 noded shells Shape <iv> = 5 (tetra): • <config> = 0, 4 become 4 noded solid tetrahedra (n4=n5=n6=n7=n8) Shape <iv> = 7 (wedge): • <config> = 0, 4 become 6 noded solid wedges (n5=n6, n7=n8) • <config> = 1 become 6 noded thick shells (n3=n4, n7=n8) Shape <iv> = 8 (hex): • <config> = 0, 4 become 8 noded solid hexahedra • <config> = 1 become 8 noded thick shells All other element types are ignored.

RADIOSS fixed file format

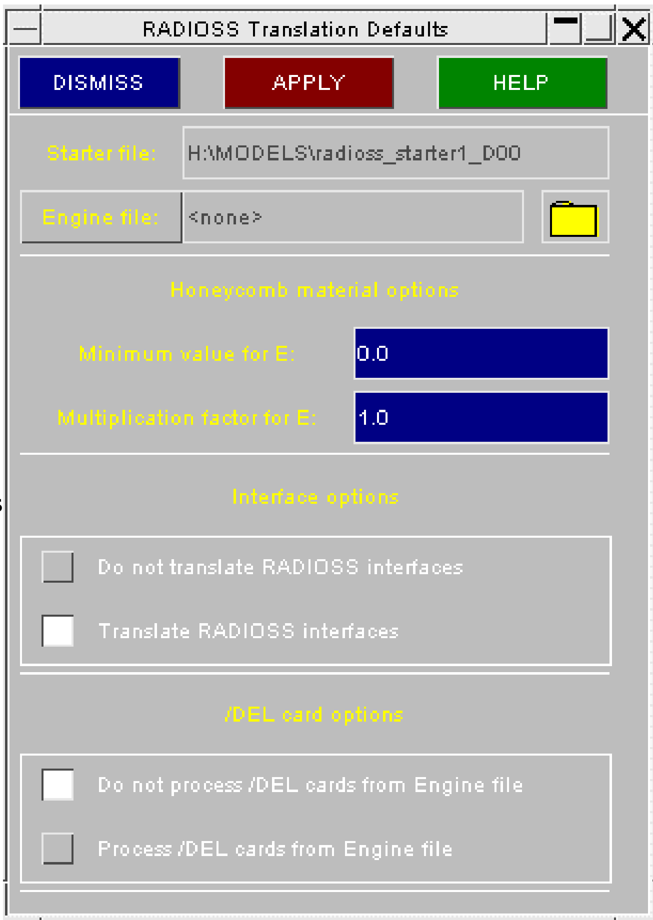
The RADIOSS Translator function is aimed specifically at translating RADIOSS starter and engine files into LS-DYNA keyword format data.



The RADIOSS translator is invoked identically to all other formats read into PRIMER. Please refer to section 3.2 of the main manual for details on the **READ** function, remembering to select the RADIOSS sub-type.

For a translation the starter file must be present. An engine file can optionally be selected. The generic file extension for a starter file is 00 (Radioss starter files usually end with d00 or D00), but this can easily be modified in the file selector panel. Once the correct file has been selected and the **APPLY** button pushed, another window will be created which allows an engine file to be selected and some options to be set.

The buttons in the RADIOSS translation defaults box are as follows:-



DISMISS terminate the translation process, returning to the generic READ panel.

APPLY accept the defaults in the panel and proceed with the translation.

HELP will create a message box full of useful information about the function of this panel.

ENGINE FILE is by default greyed out and the text box next to the button reads <none>. If you have a RADIOSS engine file to go with the starter file you are translating pressing this button will then allow an engine file to be selected by either typing in the name in the text box or pressing the button which will bring up the file selection box. Engine files by default have the extension 01 (Radioss engine files usually end with d01 or D01)

MINIMUM VALUE FOR E and **MULTIPLICATION FACTOR FOR E** are used to set options for translating honeycomb materials. When honeycomb materials are defined in RADIOSS the material curve can be steeper than the Young's modulus. This is not allowed in LS-DYNA. These options allow a minimum permissible Young's modulus to be set.

INTERFACE OPTIONS

The radio buttons under Interface Options can be used to either translate or skip any interfaces (contacts) that are in the starter file.

/DEL CARD OPTIONS

The radio buttons under /DEL card options can be used to either process or skip any /DEL cards that are present in the engine file.

TRANSLATOR FUNCTIONALITY

This section of the appendix is meant to give a brief insight into the methods that the translator uses to process and store data, and the types of RADIOSS data which are understood.

Engine File

The following keywords are translated:

`/DTIX` Initial timestep translated as **`DTINIT`** on **`*CONTROL_TIMESTEP;`** maximum timestep ignored

`/DT` Scale factor on timestep translated as **`TSSFAC`** on **`*CONTROL_TIMESTEP`**. Minimum timestep converted to a mass-scaling timestep (-ve value of **`DT2MS`** on **`*CONTROL_TIMESTEP`**). Any other keywords (e.g. specifying a particular element type) are ignored.

`/GFILE` T_{freq} translated as interval between plot file outputs (**`DT`** on **`*DATABASE_BINARY_D3PLOT`**). T_{start} and type of file ignored.

`/RFILE` N_{cycle} translated as number of cycles between restart dumps (**`CYCL`** on **`*DATABASE_BINARY_D3DUMP`**). File types ignored.

`/RUN` T_{stop} translated as termination time (**`ENDTIM`** on **`*CONTROL_TERMINATION`**). Run name and number ignored.

`/TFILE` T_{his} translated as interval between time history file outputs (**`DT`** on **`*DATABASE_BINARY_D3THDT`**). Type of file ignored

`/VERS` Tells Primer which version of RADIOSS the starter file is

The following keywords are translated, but for their effect upon the model see the relevant sections:

`/DEL` See below

`/VEL` See below

The following keywords are ignored:

`/ANIM`

`/DYREL`

`/INTER` (but birth & death times will be read from Starter file)

`/KEREL`

`/PATRAN`

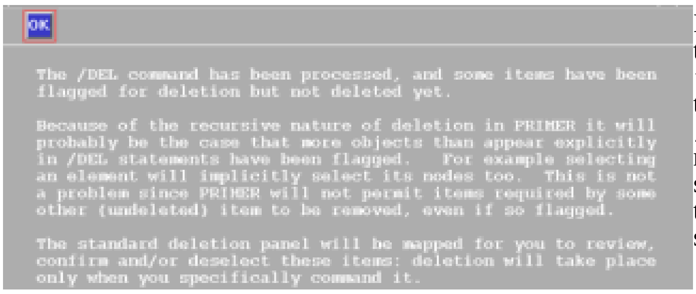
`/PRINT`

`/TITLE` (but title will be read from starter file)

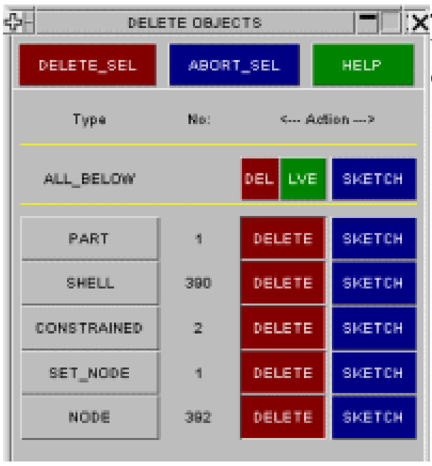
`/BCS & BCSR` cannot read restraints from the engine file, only from the Starter file.

`/RBODY`

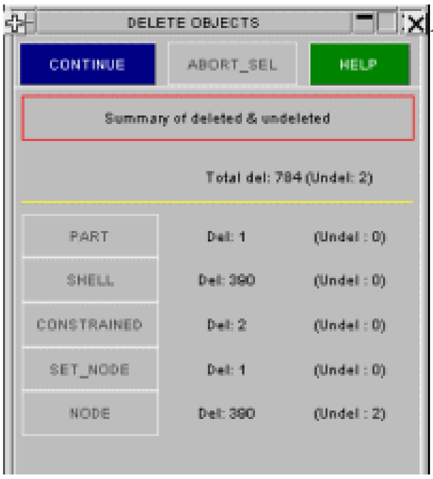
`/DEL CARDS`



If there are any **`/DEL cards`** in the engine file and they have been selected to process by using the 'Process /**`DEL cards`**' radio button Primer will allow the user to delete or leave the elements selected in the **`/DEL cards`**. If there are any elements that are marked for deletion, two windows will appear on the screen after the translation has finished. The first is a text window explaining what is happening and the second shows what objects will be deleted.



The selected elements can now either be deleted by pressing the **DELETE SEL** button or left by pressing the **ABORT SEL** button. Alternatively if only some of the elements need to be deleted the appropriate elements can be selected.



After deleting the window shows if the deletion has been successful.

/VEL CARDS

In RADIOSS, a set of nodes can be given the same translational velocity by using a **/VEL/TRA** card. Similarly they can be given the same rotational velocity by using a **/VEL/ROT** card. Each card can be used in any combination of X, Y and Z. Many of these combinations have no equivalent in LS-DYNA so some translations are not exact.

If a set of nodes occurs on a **/VEL/TRA** card only it is translated as a ***CONSTRAINED_NODE_SET**. This is an exact translation.

If a set of nodes on a **/VEL/TRA** card have a translational degree of freedom XYZ and they also appear in a **/VEL/ROT** card with degree of freedom XYZ, the set is translated to a ***CONSTRAINED_NODAL_RIGID_BODY**. This is an exact translation.

If a set of nodes appears on a **/VEL/TRA/** and a **/VEL/ROT** card with any other degree of freedom there is no equivalent in LS-DYNA. The rotational degrees of freedom are ignored and a ***CONSTRAINED_NODE_SET** is created with the translational degree of freedom.

Starter File

The table below summarises which features can be translated. For more details on each feature see the relevant section.

Feature	Translated	Ignored
Header		
Title & Control Data	title, gravity load	all other data
Materials	Types 1,2,3,4,14,19,21,22,23,27 & 28 (see below)	All other materials are translated as *MAT_NULL (see below)
Mats for time history		
Nodes	fully supported	
Nodes for time history	fully supported	
Skew frames for node time history		no equivalent

Boundary Conditions	all except GRILAG	GRILAG
Skew Frames		
Elements	all except 2-D solid and 3 noded springs	
Properties	all except type 12 (see below)	
Functions		
Concentrated loads		Sensor not translated
Pressure loads		Sensor not translated
Initial velocity	all except I_{vel}	I_{vel}
Accelerometers		
Interfaces	all types & most data - see below	
Rigid Walls	Plane, parallelogram - all data except as in "ignored" box	Node (moving wall); dist for search; direction of init. vel; prlgrm converted to rectangle
Rigid Body	see below	
Rigid Body for time history		
Added Mass		
Fixed Velocity		
Rivets spotwelds		
Sections		
Cylindrical joints		
Monitored volumes		

Header and control cards

The first line in the file must start **RADIOSS STARTER** and the INVERS number (columns 17 to 24) must be 31. If this is not the case, Primer will assume that the file is not a Radioss version 3.1 starter file and will not be able to translate the file.

CARD1 - TITLE is used by Primer for the analysis title.

IGRAX, IGRAY and IGRAZ are translated to LOAD_BODY_X, _Y and _Z respectively.

Materials

Only some materials in the RADIOSS starter file can be translated. This is because there are some RADIOSS materials that have no equivalent in LS-DYNA.

RADIOSS material law 1: Elastic

RADIOSS material type 1 translates directly to an elastic material (***MAT_ELASTIC**) in LS-DYNA

RADIOSS material law 2: Elastic/plastic

RADIOSS uses a power-law equation to describe the stress-strain behaviour. There is no exact equivalent in LS-DYNA so a piece-wise linear approach is used (material type ***MAT PIECEWISE_LINEAR_PLASTICITY**). The stress-strain curve is created using the power law and input parameters. Similarly, the rate effect equation is implemented by creating a further curve for LS-DYNA.

Primer tests to see if LS-DYNA material type ***MAT_PLASTIC_KINEMATIC** can be used instead (bi-linear elastic plastic). This will happen if the RADIOSS power is 0.0 (no strain hardening), or if the power is 1.0 and no maximum stress has been defined (bi-linear power law).

Because of the look-up table operations, the LS-DYNA model will not run very quickly; if exact correlation with RADIOSS is not required then we recommend that the materials be converted by hand back to bi-linear elastic-plastic (type 3).

RADIOSS material law 21: Foam/Honeycomb

The most direct equivalent of this RADIOSS material is LS-DYNA material type ***MAT SOIL_CONCRETE (78).** This material allows a curve of pressure vs compaction while still allowing a pressure-sensitive yield criterion. However, there is a special case where the behaviour can be replicated more simply using the crushable foam material ***MAT CRUSHABLE_FOAM(63)** - see final paragraph in this section.
The pressure-vs-volume strain relationship is converted to be suitable for LS-DYNA (LS-DYNA true strain = -log(1/(1.+RADIOSS strain))) and the maximum volume strain is included as a steep slope at the end of the curve.

In RADIOSS, the curve can be steeper than the elastic modulus: often the RADIOSS models have very low Young's moduli. This is not permitted in LS-DYNA: it would cause the timestep to be reduced and there would be energy gain caused by hysteresis loops going the wrong way. Therefore we must raise the Young's modulus or change the curve. Primer checks each section of the curve, and if steeper than the bulk modulus, revises the curve and warns the user. This avoids the problems listed above but because the LS-DYNA curve is then softer than the RADIOSS curve, unwanted bottoming-out behaviour may occur.

As an option, the user may request that all RADIOSS materials which use law 21 have their Moduli multiplied by a factor (e.g.10) - the curve segments will then be permitted to be stiffer. The new modulus will then be checked against a user-supplied minimum modulus, and raised to match it if necessary. To do this, use the options in the RADIOSS translation window. Set values for the multiplication factor and minimum value for Young's Modulus.

The yield vs pressure relationship is determined by a quadratic power law with plateau value (A0, A1, A2 and AMAX). These must be converted by Primer to a load curve for LS-DYNA, and this in turn requires a sensible range of pressures to form the x-axis of the load curve. Primer uses a range of -5 to +50: this is sensible for foams in the usual millimetre units but would be very inappropriate if the units are not millimetres and Newtons.

The RADIOSS tensile and unloading bulk moduli are ignored (in LS-DYNA, assumed to be defined by Young's modulus and Poisson's ratio)

If A0=0.0,A1=0.0,A2=3.0 then the behaviour is that of crushable foam. Primer translates the material as type 63 (***MAT_CRUSHABLE_FOAM**). The curve is converted to stress-strain using the following assumptions:

LS-DYNA volumetric strain = 1.0 - (1.0/(RADIOSS strain + 1.0))

LS-DYNA direct stress = 3 x pressure

Again, the slopes of the curve are checked against the Young's modulus and revised if necessary to maintain the timestep.

RADIOSS material law 19: Elastic orthotropic

This material is translated to LS-DYNA material ***MAT_ORTHOTROPIC_ELASTIC (2).** The stiffness reduction in compression parameter (Re) is ignored.

RADIOSS material laws 3,4,22,23,27

These RADIOSS materials are similar to material law 2 with some extra parameters and are translated in the same way, so some information may be lost (for example damage parameters). The user should check these material models after translation to satisfy themselves that a suitable translation has been achieved. If not some editing of the keyword deck may be required. Also in some circumstances the LS-DYNA material type ***MAT_ENHANCED_COMPOSITE_DAMAGE** may be more appropriate.

RADIOSS material law 28

This material is translated to LS-DYNA material 26 (***MAT_HONEYCOMB**). The load curves are translated to volumetric strain for LS-DYNA. RADIOSS allows the user to specify the local coordinate system of each element individually, with respect to a reference plane defined on property set type 6. In LS-DYNA there is no similar function, but by swapping the values for each direction on the material card in LS-DYNA it is possible to achieve the same effect. Currently the swapping has to be done by hand since Primer cannot do this automatically. It should be also noted that the orthotropic angle referred to in the RADIOSS property set 6 cannot be incorporated into the LS-DYNA keyword deck.

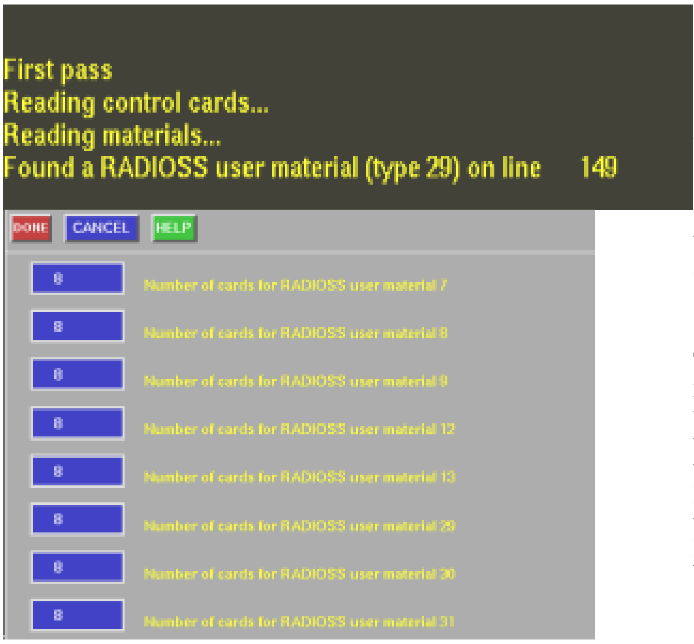
Material law 36

This is translated into material 24 in LS-DYNA (***MAT_PIECEWISE_LINEAR_ PLASTICITY**). Only the first function is translated so any multi-functional parts are lost.

Unsupported Materials

Where an unsupported material model is encountered Primer will translate them as **`*MAT_NULL`**. The user then has to change the material model and input parameters by hand to a suitable material model before running the LS-DYNA model. All translated **`*MAT_NULL`** materials have zero density to act as a error trap in LS-DYNA.

User defined materials



In a RADIOSS starter file, material laws 7, 8, 9, 12, 13, 29, 30 and 31 are user defined materials. As the length of a user defined material in RADIOSS is unknown some information is needed by the translator when one is encountered. A warning will be printed in the dialogue box.

Another window will also be displayed on the screen showing all the user defined material numbers in RADIOSS and how many cards each one requires. The default number for each material law is set to 8 cards (each material in RADIOSS has 7 compulsory cards). The user needs to edit the number of cards for the required user defined material in this window. Once this has been done and the **DONE** button pressed the translation will proceed. The user defined materials will be translated to user defined materials in LS-DYNA as shown in the following table. However, there will be no material parameters in the materials in LS-DYNA. The user will need to edit the material back to something sensible.

RADIOSS material law	LS-DYNA user defined material
7	41
8	42
9	43
12	44
13	45
29	46
30	47
31	48

Equivalent RADIOSS and LS-DYNA user defined materials.

Skew Frames

Moving skew frames are translated to **`*DEFINE_COORDINATE_NODES`** (fully supported)

Fixed skew frames are translated to **`*DEFINE_COORDINATE_VECTOR`** (fully supported)

Property sets

Type 0 (void) Skipped

Type 1 (shells) Supported. Only N, Thick and A_{shear} are translated.

Type 2 (truss) Supported. Initial gap is not translated.

Type 3 (beam) Supported. Rotations at nodes are ignored.

Type 4 (spring) RADIOSS spring properties are defined by a number of parameters describing force as a function of deflection and deflection rate, with various options for treating unloading. In the general case, there is no equivalent in LS-DYNA, but if certain combinations of parameters have been left zero in the RADIOSS deck, a close equivalent in LS-DYNA does exist.

Primer identifies the following cases:

a) If curve N1 is zero (linear spring) but K is non-zero, the spring is assumed to be linear elastic. This translation is exact when C is also zero; if C is non-zero then the rate effect will be missing in LS-DYNA.
b) If curve N1 is zero (linear spring) and K is also zero, a linear damper is assumed with coefficient C. This is an exact translation.
c) Otherwise the spring is assumed to be nonlinear elastic, with rate effect: curve N1 is force-deflection and curve N2 is rate effect. This translation is exact only when H, A and B are zero. If H is non-zero, the unloading response is missing in LS-DYNA; this could lead to serious errors since no permanent deflection could occur.

Type 5 (Rivet and Spotweld) Supported. Maximum normal and tangential forces are translated.

Type 6 (Orthotropic solid element) Translated to a normal solid element property. Any orthotropic angles will be lost.

Type 7 (Airbag) Translated into a Wang Nefske airbag in LS-DYNA. In LS-DYNA the ambient density is required when creating a control volume. Primer attempts to create a suitable value for the density by looking at the values given for the Gas constant and the specific heat. This value may not be correct and will need checking. Functions used for airbags in Radioss use total mass vs. time. In LS-DYNA the function is mass flow rate vs. time. The Radioss function is differentiated to get the mass flow rate curve for LS-DYNA.

Type 8 (General spring). The best equivalent of a general spring in LS-DYNA is a discrete beam element. As with property type 4 springs only some combinations of values can be translated correctly. A linear discrete beam will be created if the properties in all 6 degrees of freedom are linear, and a non-linear discrete beam will be created if all are non-linear. If a mixture of properties are found 2 discrete beams need to be created - one linear and one non-linear - with the appropriate properties for each degree of freedom. As the spring is changed into a beam element in LS-DYNA the element number will change to avoid any potential clashes.

Type 9 (Orthotropic Shell) and **type 10 (Composite shell)**.These are translated as a normal shell. Any orthotropic properties will be lost.

Type 11 (Composite shell). Material angles will be lost but a user defined integration law is created with the correct layer thicknesses.

Type 12 (3 noded springs). Not supported. These will be skipped.

Type 13 (beam type spring). Translated in the same way as type 9 general springs.

Type 14 (Solid) Supported

Accelerometers

Accelerometers in RADIOSS are defined by one node and a skew system (coordinate system). In LS-DYNA they are defined using 3 nodes which define a local triad. These nodes must also be on one rigid body. To translate the accelerometers two extra nodes are created which together with the first node make up this triad. A rigid part is then created which contains a beam representation of this triad. Finally the 3 nodes are used to create the accelerometer. N.B as any skew system can be used in RADIOSS to define an accelerometer, if a moving skew frame (defined by nodes) is used which is defined with 3 nodes somewhere else in the model the accelerometer in LS-DYNA will give different results. This is because the accelerometer in RADIOSS will give results depending on how the moving skew frame rotates. The accelerometer for LS-DYNA will give the results depending on how the rigid triad rotates.

Interfaces

Interfaces in RADIOSS are defined in a very similar way to LS-DYNA and so can be translated. The following table shows which contact types in LS-DYNA the RADIOSS interfaces are translated to.

RADIOSS Interface type	LS-DYNA contact type
2 TIED	***CONTACT_TIED_NODES_TO_SURFACE or** *CONTACT_TIED_SURFACE_TO_SURFACE
3 SLIDE/VOID	***CONTACT_AUTOMATIC_NODES_TO_SURFACE or** *CONTACT_AUTOMATIC_SURFACE_TO_SURFACE
4 SLIDE/VOID	***CONTACT_AUTOMATIC_SINGLE_SURFACE**
5 SLIDE/VOID	***CONTACT_AUTOMATIC_NODES_TO_SURFACE or** *CONTACT_AUTOMATIC_SURFACE_TO_SURFACE

6 SLIDE/VOID	***CONTACT_RIGID_BODY_TWO_WAY_TO_RIGID_** BODY
7 SLIDE/VOID	***CONTACT_AUTOMATIC_NODES_TO_SURFACE or** *CONTACT_AUTOMATIC_SURFACE_TO_SURFACE or *CONTACT_AUTOMATIC_SINGLE_SURFACE
8 SLIDE	***CONTACT_DRAWBEAD**
10 TIED/VOID	***CONTACT_AUTOMATIC_NODES_TO_SURFACE or** *CONTACT_AUTOMATIC_SURFACE_TO_SURFACE

Surface input types 1 (segments), 2 (nodes), 4 (shell property sets) and 5 (shell elements) are translated directly. Input type 3 (shell materials) has no equivalent in LS-DYNA so it is translated to a list of parts which use these materials. Interfaces in RADIOSS can have the same entities in the slave and the master side of the contact. This could cause problems in LS-DYNA so for each contact any entities which occur in the slave and master surfaces are removed from the master side.

For RADIOSS interface type 4, any entities which are left in the master side of the contact after this process are added to the slave side and a single surface contact created.

Interface type 7 can be used to replace more than one type of contact in RADIOSS. If a type 7 contact is found which does not use nodes for the slave it is translated to a surface to surface contact and a single surface contact.

Rigid Walls

Rigid walls in RADIOSS are translated to LS-DYNA rigid walls as follows

RADIOSS Type	**LS-DYNA type**
1 infinite plane	***RIGIDWALL_PLANAR_OPTION**
2 infinite cylinder	***RIGIDWALL_GEOMETRIC_CYLINDER**
3 sphere	***RIGIDWALL_GEOMETRIC_SPHERE**
4 parallelogram	***RIGIDWALL_PLANAR_FINITE_OPTION**

The OPTION on the LS-DYNA rigidwalls can be MOVING if the rigid wall in RADIOSS has a velocity. However in RADIOSS a rigidwall can have a velocity in any direction. In LS-DYNA velocity is restricted to the direction of the normal defining the rigidwall. The vector magnitude of the velocity from the RADIOSS rigidwall is used and so the direction and magnitude of the velocity may be incorrect.

Rigid Bodies

Rigid bodies in RADIOSS can be defined by either nodes or property sets. The equivalent of a rigid body defined by nodes in LS-DYNA is a ***CONSTRAINED_NODAL_RIGID_BODY**. However if this is used then no external motions can be applied to the rigid body (e.g. ***BOUNDARY_PRESCRIBED_MOTION_RIGID**). To overcome this, if a rigid body in RADIOSS is defined by nodes a new part, beam section and rigid material are created. Beams with negligible mass are created from the primary node on the rigid body to all other nodes. This also helps in visualisation of the rigid body.

If the primary node on a rigid body is at (0,0,0) then RADIOSS will calculate the centre of mass and inertia for the rigid body on initialisation and then move the node to the centre of mass. If this node was included in the rigid body definition in LS-DYNA the inertia and centre of mass would be incorrect. If the master node is found to be at the origin it is not translated and the next node in the rigid body is used as the master node.

Rigid bodies defined by property sets are translated as rigid parts by Primer with ***CONSTRAINED_RIGID_BODIES** to merge the parts together. If the master node is not at the origin it is added to the rigid body by using a ***CONSTRAINED_EXTRA_NODE** card.

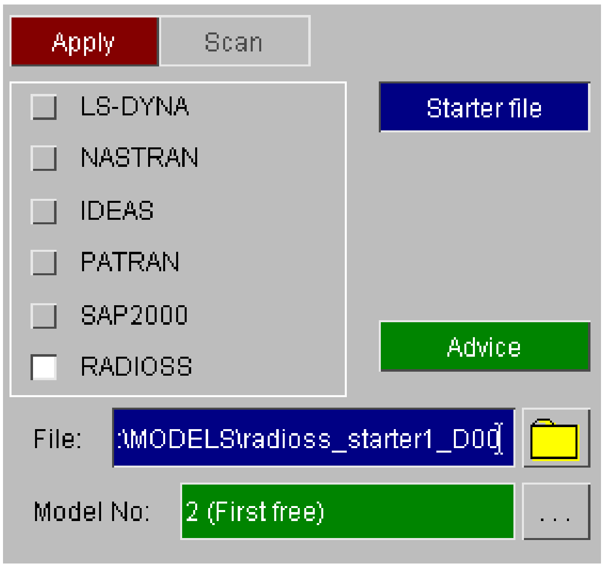
If there is a mass and inertia tensor for the rigid body it is translated and a ***PART_INERTIA** card defined. If there is no mass and inertia a ***PART** card is defined and LS-DYNA will calculate the mass and inertia tensor. If there is only one of the mass and inertia present a warning will be printed.

Fixed velocities

Fixed velocities in RADIOSS are translated to ***BOUNDARY_PRESCRIBED_MOTION_ NODE** unless the node is a primary node on a rigid body. If this is the case it is translated to ***BOUNDARY_PRESCRIBED_MOTION_RIGID**.

RADIOSS block format

The RADIOSS Translator function is aimed specifically at translating RADIOSS starter and engine files into LS-DYNA keyword format data. The block format is the default format for the RADIOSS translator.



The RADIOSS translator is invoked identically to all other formats read into PRIMER. Please refer to section 3.2 of the main manual for details on the **READ** function, remembering to select the RADIOSS sub-type.

For a translation the starter file must be present. An engine file can optionally be selected. The generic file extension for a starter file is 00 (Radioss starter files usually end with d00 or D00), but this can easily be modified in the file selector panel. Once the correct file has been selected and the **APPLY** button pushed, another window will be created which allows an engine file to be selected and some options to be set.

The buttons in the Radioss Translation Defaults box are as follows:

APPLY accept the defaults in the panel and proceed with the translation.

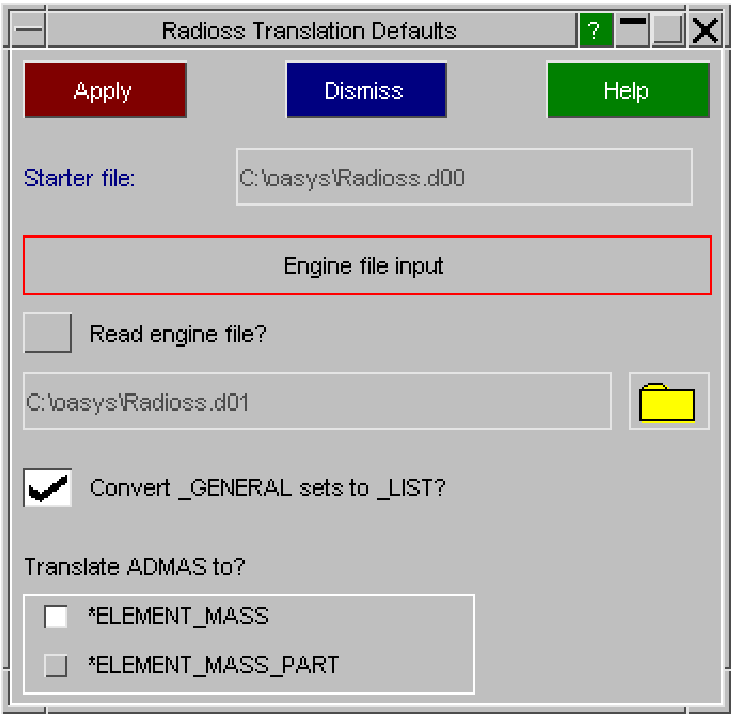
DISMISS terminate the translation process, returning to the generic READ panel.

HELP will create a message box full of useful information about the function of this panel.

READ ENGINE FILE button can be checked to select an engine file by either typing in the name in the text box or by pressing the button which will bring up the file selection box. Engine files by default have the extension 01 (Radioss engine files usually end with d01 or D01). The engine file is translated by default as long as a *01 file exists that corresponds to the *00 starter file.

CONVERT _GENERAL sets to _LIST can be used to convert sets that use the _GENERAL option to _LIST sets.

Translate ADMAS to? can be used to specify the method of translation for /ADMAS cards.



Current known issues

- Not all options are supported for defining groups and surfaces.
- Only a small number of materials are translated at present.

Notes on specific keywords

Please see the following table which uses the following colours to indicate how well supported the keyword is.

Not supported
Limited support. A small subset of the options is supported (details given)
Reasonable support. Most options supported (details given)
Radioss 4.4 only - Reasonable support (details given)
Fully supported
Radioss 4.4 only - Fully supported

Starter file

Keyword	Notes
/ACCEL	Accelerometers are translated to *DATABASE_HISTORY_NODE_LOCAL_ID. **Note that Fcut is not translated and the output will not be filtered.**
/ADMAS	Masses are translated to *ELEMENT_MASS. Additionally so you can tell which masses are created for each original /ADMAS card, a *GROUP is created for each /ADMAS containing the *ELEMENT_MASS masses created.
/ANALY	Not supported.
/BCS	/BCS cards using a node group will be translated to *BOUNDARY_SPC_SET_ID. Any secondary nodes will be translated to *BOUNDARY_SPC_NODE.
/BEAM	Translated to *ELEMENT_BEAM.
/BRICK	Translated to *ELEMENT_SOLID.
/BRIC20	Translated to *ELEMENT_SOLID. Midside nodes are ignored.
/CLOAD	/CLOAD cards using a node group will be translated to *LOAD_NODE_SET. Any secondary nodes will be translated to *LOAD_NODE_POINT. **Note that the sensor input is not translated**.
/CYL_JOINT	Not supported.
/DEF_SHELL	Element formulation value is set to 2 (Belytschko-Tsay). If Ishell in input deck <> 1 (Belytschko), a warning is issued.
/DEF_SOLID	A warning is issued and the element formulation value set to 1.
/END	Anything after /END is ignored.
/FUNCT	Translated to *DEFINE_CURVE.
/GRAV	Not supported.
/GRBEAM	All types are supported. BEAM is translated to a *SET_BEAM. PART is translated to a *SET_BEAM_GENERAL using option PART. BOX and BOX2 are translated to a *SET_BEAM_GENERAL using option BOX (box is created). SUBSET, MAT and PROP are translated to a *SET_BEAM_GENERAL using option PART (an equivalent list of parts is generated).

/GRBRIC	All types are supported. BRIC is translated to a *SET_SOLID. PART is translated to a *SET_SOLID_GENERAL using option PART. BOX and BOX2 are translated to a *SET_SOLID_GENERAL using option BOX (box is created). SUBSET, MAT and PROP are translated to a *SET_SOLID_GENERAL using option PART (an equivalent list of parts is generated).
/GRNOD	All types **except SURF and NODENS** are supported. NODE is translated to a *SET_NODE. GENE is translated to *SET_NODE_GENERATE. PART is translated to a *SET_NODE_GENERAL using option PART. BOX and BOX2 are translated to a *SET_NODE_GENERAL using option BOX (box is created). SUBSET, MAT and PROP are translated to a *SET_NODE_GENERAL using option PART (an equivalent list of parts is generated).
/GRQUAD	All types are supported. QUAD is translated to a *SET_SHELL. PART is translated to a *SET_SHELL_GENERAL using option PART. BOX and BOX2 are translated to a *SET_SHELL_GENERAL using option BOX (box is created). SUBSET, MAT and PROP are translated to a *SET_SHELL_GENERAL using option PART (an equivalent list of parts is generated).
/GRSH3N	All types are supported. SH3N is translated to a *SET_SHELL. PART is translated to a *SET_SHELL_GENERAL using option PART. BOX and BOX2 are translated to a *SET_SHELL_GENERAL using option BOX (box is created). SUBSET, MAT and PROP are translated to a *SET_SHELL_GENERAL using option PART (an equivalent list of parts is generated).
/GRSHEL	All types are supported. SHEL is translated to a *SET_SHELL. PART is translated to a *SET_SHELL_GENERAL using option PART. BOX and BOX2 are translated to a *SET_SHELL_GENERAL using option BOX (box is created). SUBSET, MAT and PROP are translated to a *SET_SHELL_GENERAL using option PART (an equivalent list of parts is generated).
/GRSPRI	All types are supported. SPRI is translated to a *SET_BEAM. PART is translated to a *SET_BEAM_GENERAL using option PART. BOX and BOX2 are translated to a *SET_BEAM_GENERAL using option BOX (box is created). SUBSET and PROP are translated to a *SET_BEAM_GENERAL using option PART (an equivalent list of parts is generated).
/GRTRUS	All types are supported. TRUS is translated to a *SET_BEAM. PART is translated to a *SET_BEAM_GENERAL using option PART. BOX and BOX2 are translated to a *SET_BEAM_GENERAL using option BOX (box is created). SUBSET and PROP are translated to a *SET_BEAM_GENERAL using option PART (an equivalent list of parts is generated).
/IMPDISP	If a node group is used it is translated to *BOUNDARY_PRESCRIBED_MOTION_SET with VAD=2. Secondary nodes are translated to *BOUNDARY_PRESCRIBED_MOTION_NODE with VAD=2. **Note that the sensor input is not translated**. If a skew system is used a *DEFINE_VECTOR card is created using the skew system and direction and the DOF is set to 4.
/IMPVEL	If a node group is used it is translated to *BOUNDARY_PRESCRIBED_MOTION_SET with VAD=0. Secondary nodes are translated to *BOUNDARY_PRESCRIBED_MOTION_NODE with VAD=0. **Note that the sensor input is not translated**. If a skew system is used a *DEFINE_VECTOR card is created using the skew system and direction and the DOF is set to 4 for translation or 8 for rotation.
/INISTA	Not supported.
/INIVEL	If a node group is used it is translated to *INITIAL_VELOCITY. Secondary nodes are translated to *INITIAL_VELOCITY_NODE. **Note that the title will be lost as the LD-DYNA keywords do not support it**.

/INTER	TYPE2: Converted to *CONTACT_TIED_SHELL_EDGE_TO_SURFACE_OFFSET. This is used instead of *CONTACT_TIED_NODES_TO_SURFACE as it is better for tying spotweld beams onto shells (the rotational dof is handled correctly with tied shell edge, with tied nodes to surface it is not). The _OFFSET option is used so that the contact will work correctly if any nodes on the contact segments are in constraints (e.g. nodal rigid bodies TYPE 3: Converted to *CONTACT_AUTOMATIC_SURFACE_TO_SURFACE TYPE5: Converted to *CONTACT_AUTOMATIC_NODES_TO_SURFACE TYPE6: Converted to *CONTACT_RIGID_BODY_TWO_WAY_TO_RIGID_BODY TYPE7: Converted to *CONTACT_AUTOMATIC_NODES_TO_SURFACE Types 8, 10, 11 and 14 not supported. **Note that some contacts will be better translated as *CONTACT_AUTOMATIC_SINGLE_SURFACE. The user will need to review the contacts and decide this**.
/IOFLAG	Not supported.
/LINE	Only type SEG is supported. It is translated to a *SET_SEGMENT.
/MADYMO	Not supported.
/MAT	ELAST (law 1): Translated to *MAT_ELASTIC. PLAS_JOHNS (law 2): If m is zero then there are no temperature effects and *MAT_SIMPLIFIED_JOHNSON_COOK is used. Otherwise *MAT_JOHNSON_COOK is used. PLAS_BRIT (law 27): Translated to *MAT_098 (Simplified Johnson Cook). HONEYCOMB (law 28): Translated to either *MAT_HONEYCOMB or to *MAT_MODIFIED_HONEYCOMB. If iflag1 != iflag2 - Translated to *MAT_126 along with a warning message. If iflag1= iflag2=0 - x = x/(1+x) ; New entry is inserted into beginning of load curve if current 1st entry is positive; Translated to *MAT_026. If iflag1 = iflag2 = 1 - Translated to *MAT_126. If iflag1 = iflag2 = 1 - Translated to *MAT_126; SFA = 1; AOPT, MACF set to 0. Checks are thrown in to see whether any MAT_MODIFIED_HONEYCOMB types use a solid section. If yes, and if no other material type uses that section, element formulation value is set to 9. A warning is issued and the element formulation value is retained at 1 if other types use the section as well. PLAS_TAB (law 36): Translated to *MAT_024 (Piecewise Linear Plasticity). Material card refers to a table that contains relevant load curves and strain rates if multiple load curves are specified. *MAT_ADD_EROSION is invoked if failure is specified. VISC_TAB (law 38): Translated to *MAT_057 (Low Density Foam).
/MEMORY	Not supported.
/MONVOL	**Types AREA and COMMU are not supported**. Type PRES is translated to *AIRBAG_SIMPLE_PRESSURE_VOLUME. Type GAS is translated to *AIRBAG_ADIABATIC_GAS_MODEL. Type AIRBAG is translated to *AIRBAG_SIMPLE_AIRBAG_MODEL. **No attempt is made to translate the airbag properties**. Only the surface is translated (to a *SET_SEGMENT).
/NODE	Translated to *NODE
/PART	Translated to *PART
/PENTA6	Translated to *ELEMENT_SOLID
/PLOAD	Not supported.

/PROP	/PROP/SHELL (pid 1) translated to *SECTION_SHELL /PROP/TRUSS (pid 2) translated to *SECTION_BEAM with ELFORM = 3 (truss). /PROP/BEAM (pid 3) translated to *SECTION_BEAM with ELFORM = 2 (resultant beam). /PROP/SPRING (pid 4) translated to *SECTION_BEAM with ELFORM = 6 (discrete beam). A *MAT_ELASTIC_SPRING_DISCRETE_BEAM material is created for linear and non-linear elastic springs. For elasto-plastic springs a *MAT_INELASTIC_SPRING_DISCRETE_BEAM material is created. Note that if B is non-zero the log term used in the Radioss and Dyna formulations will be used and this is not the same. Note that hardening types > 1 have no equivalent. /PROP/RIVET (pid 5) properties are copied to any *CONSTRAINED_SPOTWELD elements created from /RIVET elements. Fn is translated to SN. Ft is translated to SS. N and M on the *CONSTRAINED_SPOTWELD are set to 1.0. **The maximum length and rotation flag are not supported**. /PROP/SOL_ORTHO (pid 6) translated to *SECTION_SOLID. Relevant *ELEMENT_SOLID cards are converted to *ELEMENT_SOLID_ORTHO cards. Local material directions are also inserted. /PROP/SPR_GENE (pid 8) and /PROP/SPR_BEAM (pid 13) translated to *SECTION_BEAM with ELFORM = 6 (discrete beam). A *MAT_GENERAL_SPRING_DISCRETE_BEAM material is created. Note that if B is non-zero the log term used in the Radioss and Dyna formulations will be used and this is not the same. Note that hardening types > 1 have no equivalent. SCOOR is set to 2 if all beams that refer to parts that, in turn, refer to this section are of finite length SCOOR is set to 0 if all beams that refer to parts that, in turn, refer to this section are of zero length SCOOR is set to 2 if we have a mixture; zerolength beams are then added to a set and a warning issued. /PROP_SH_SANDW (pid 11) translated to *SECTION_SHELL. *PART cards are created for each layer. Relevant *INTEGRATION_SHELL card written. /PROP/SOLID (pid 14) translated to *SECTION_SOLID. Calls DEF_SOLID if elform is set to 0.
/QUAD	Translated to *ELEMENT_SHELL
/RANDOM	Not supported.
/RBODY	Translated to *CONSTRAINED_NODAL_RIGID_BODY with a PNODE node. Note values of ICOG other than 1 are ignored (there is no equivalent in LS-DYNA). Mass and inertia properties are ignored (there is no equivalent in LS-DYNA). INERTIA option is now invoked if positive components of inertia tensor are input along with positive value of mass. If the tensor is spherical a small perturbation is added to Ixx. If a local coordinate system is given and the tensor has off diagonal terms this is illegal for IRCS=1 in LSDYNA. In this case the tensor is rotated back to the global system and IRCS, CID and CID2 are all set to zero.
/REFSTA	Not supported.
/RIVET	Translated to *CONSTRAINED_SPOTWELD
/RLINK	Not supported.
/RWALL	/RWALL/CYL is translated to *RIGIDWALL_GEOMETRIC_CYLINDER. /RWALL/SPHER is translated to *RIGIDWALL_GEOMETRIC_SPHERE /RWALL/PLANE is translated to *RIGIDWALL_PLANAR /RWALL/PARAL is translated to *RIGIDWALL_PLANAR_FINITE Moving rigid walls are only supported for the PLANE and PARAL types.

/SECT	Translated to *DATABASE_CROSS_SECTION_SET cards. **Note that the 3 nodes defining the plane are ignored and the output will be in the global coordinate system. To get output in a local coordinate system in LS-DYNA an *ELEMENT_SEATBELT_ACCELEROMETER would need to be made with the 3 nodes (which would be referenced on the *DATABASE_CROSS_SECTION_SET card). This is not done because the *ELEMENT_SEATBELT_ACCELEROMETER needs to be on a rigid body. Making the 3 nodes part of a rigid body could significantly alter the results**. If a triangle group is used as well as a shell group the shells in the triangle group will be added to the shell group. Check that this will not cause problems elsewhere. If secondary nodes are used as well as a node group the secondary nodes will be added to the node group. Check that this will not cause problems elsewhere.
/SENSOR	Not supported.
/SH3N	If the shell has a thickness it is translated to *ELEMENT_SHELL_THICKNESS, otherwise *ELEMENT_SHELL. It is assumed that there are no label clashes with /SHELL elements.
/SHELL	If the shell has a thickness it is translated to *ELEMENT_SHELL_THICKNESS, otherwise *ELEMENT_SHELL
/SKEW	/SKEW/FIX translated to *DEFINE_COORDINATE_VECTOR /SKEW/MOV translated to *DEFINE_COORDINATE_NODES
/SPMD	Not supported.
/SPRING	Translated into *ELEMENT_BEAM (discrete beams). Numbering may change to prevent clashes with beam and truss elements
/SUBSET	Subset hierarchy and part contents are reproduced in Primer (see the part tree). Each subset is also translated to a *SET_PART. Note that as *SET_PARTs cannot be nested like subsets they will just be expanded to the list of parts. e.g. if subset 1 has child subsets 2 and 3, *SET_PART 2 will contain the parts in subset 2, *SET_PART 3 will contain the parts in subset 3, but *SET_PART 1 will contain the parts in subsets 1, 2 and 3.
/SURF	SEG is translated to a *SET_SEGMENT in DYNA. PART is translated to a *SET_SEGMENT_GENERAL using option PART. BOX and BOX2 are translated to a *SET_SEGMENT_GENERAL using option BOX (box is created). SUBSET, MAT and PROP are translated to a *SET_SEGMENT_GENERAL using option PART (an equivalent list of parts is generated) **SURF, GRSHEL, GRSH3N, ELLIPS and MDELLIPS are not supported.**
/TETRA4	Translated to *ELEMENT_SOLID
/TETRA10	Translated to *ELEMENT_SOLID
/TITLE	Translated to *TITLE
/TH	Only types NODE, SHEL, SH3N, BRIC, QUAD, BEAM, TRUSS and SPRING (i.e. nodes and elements) are translated. For type NODE, if there is a skew frame it is translated to *DATABASE_HISTORY_NODE_LOCAL_ID. Everything else is translated to *DATABASE_HISTORY_xxx_ID where xxx is the appropriate type. *DATABASE_RWFORC and *DATABASE_SECFORC files are used by both the /TH cards and the RADIOSS Engine keyword /ANIM/VECT/FOPT. The /TH keywords grab their DT value from the /TFILE keyword (if present). The /ANIM/VECT/FOPT keyword gets its DT value from the /ANIM/DT keyword (if present). In case of a clash, the /TH and /TFILE keywords take preference over the /ANIM keywords.
/TRUSS	Translated into *ELEMENT_BEAM (truss). Numbering may change to prevent clashes with beam and spring elements

Engine file

Engine file Keyword	Notes
/ANIM	The following options are translated: /DT - Start time is ignored; Frequency of output is specified /BRICK/TENS/STRAIN /SHELL/TENS/STRAIN /NODA/DT /NODA/DMAS /VECT/CONT /VECT/FOPT The following options are handled by default: /MASS /BEAM/FORC /TRUS/FORC /SPRING/FORC
/BCS	Modifies or inserts new *BOUNDARY cards.
/BCSR	Modifies or inserts new *BOUNDARY cards.
/DEL	Initiates a delete panel that permits the user to leave or delete elements specified in the /DEL card.
/DELINT	Not supported .
/DT	Cycle frequency (scale factor) is read in and translated. deltatmin is ignored. Options are ignored.
/DT1	Not supported.
/DTIX	Not supported.
/DYREL	Not supported.
/FUNCT	Redefines existing *DEFINE_CURVE cards. Inserts new card if the given curve Ifunc does not exist.
/GFILE	Not supported.
/INIV	Redefines existing *DEFINE_CURVE cards. Inserts new card if the given curve Ifunc does not exist.
/INTER	Not supported.
/KEREL	Not supported.
/KILL	User gets a warning to the effect that the restart file is written anyway (as in STOP)
/MADYMO	Not supported.
/MON	/OFF option warns the user that CPU time information is written by default.
/OUTP	Not supported.
/PARITH	Not supported.
/PATRAN	Not supported.
/PRINT	Not supported.
/PROC	Translated to *CONTROL_PARALLEL.
/RBODY	The RADIOSS Starter keyword is supported. The Engine keyword, however, is not.

/RFILE	Translated to *DATABASE_BINARY cards.
/RUN	Mandatory keyword translated to *CONTROL_TERMINATION.
/SHFRA	Not supported.
/STOP	Translated to *CONTROL_TERMINATION.
/TFILE	Translated to *DATABASE_BINARY cards. (refer /TH in the Starter file section).
/@TFILE	Not supported.
/TH	Version 31 is not currently supported. The VERS option is hence ignored.
/TITLE	Supported.
/VEL	Not supported.
/VERS	Mandatory keyword - 31 option warns the user that the translator was written to support version 4.1

Other notes

*DATABASE_ABSTAT
*DATABASE_GLSTAT
*DATABASE_MATSUM
*DATABASE_RCFORC
*DATABASE_RWFORC
*DATABASE_SECFORC
cards are automatically generated (as there is no LS-DYNA equivalent for most of the RADIOSS time history output (e.g. /TH/RWALL) unless one or more card has already been generated during translation..

Additionally a *CONTROL_CONTACT card is created with RWPNAL=1 so nodes on rigid bodies can interact with rigid walls.

During translation any keywords that are not supported will be written to a file <*filename*>.skipped. Any warnings that are generated will be written to a file <*filename*>.warnings

SAP2000 file format

The SAP2000 Translator function is aimed specifically at translating the SAP2000 ASCII input file (.s2k) into LS-DYNA keyword format data.

Briefly, SAP2000 (viz "Structural Analysis Program") marketed by CSI (Computers & Structures Inc.) is used for linear and simple non-linear analyses of building structures.

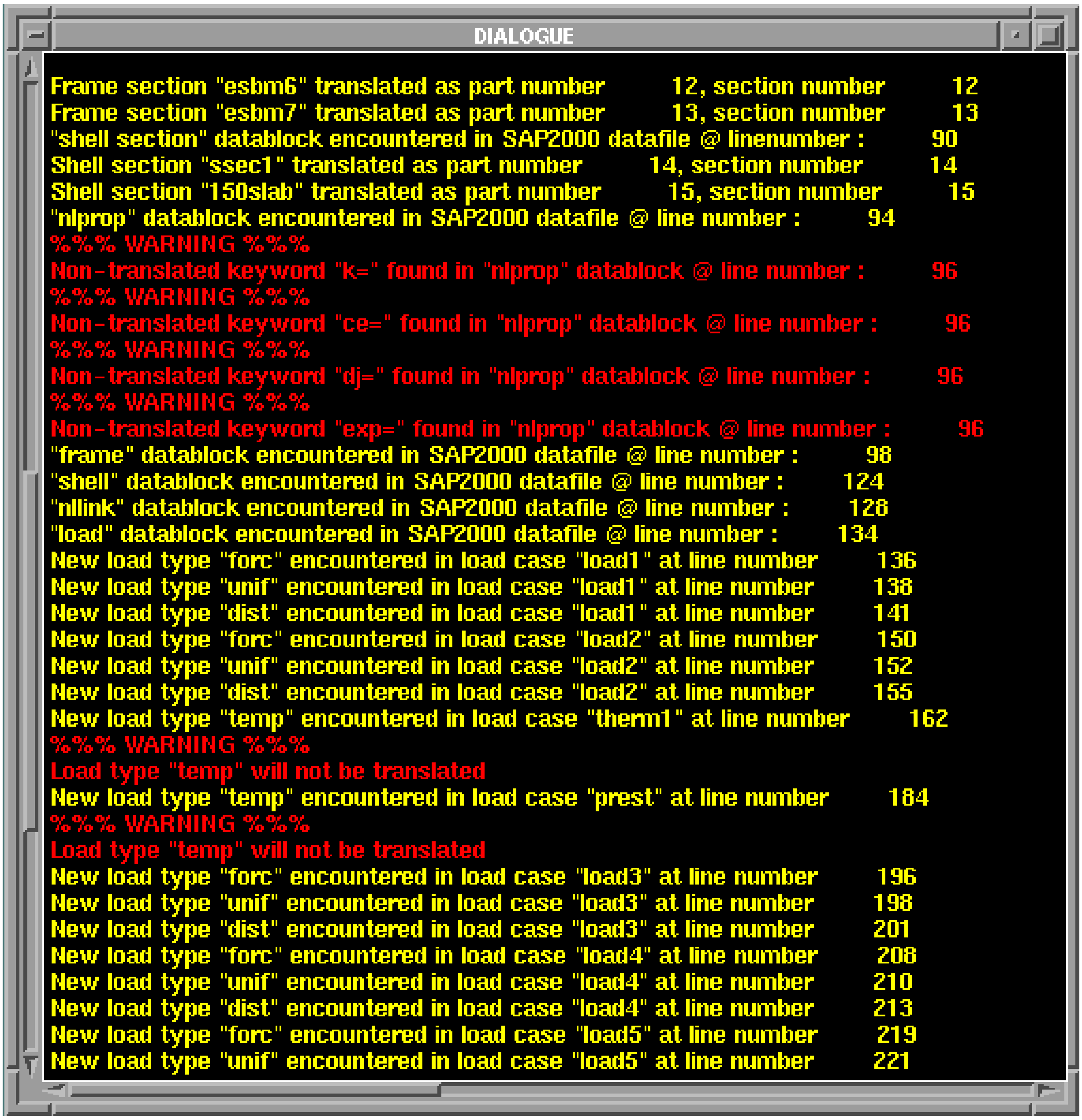
LS-DYNA is well suited to the cutting edge performance based earthquake engineering and enables extremely complex structures to be analysed against time history data recorded from past seismic events. In this way damage to the structure can be quantified and the most effective structure designed as a result of LS-DYNA analyses.

The SAP2000 translator is invoked identically to all other formats read into PRIMER. Please refer to the main manual for details on the **READ** function, remembering to select the SAP2000 sub-type,

The generic file extension expected is .s2k, but this can easily be modified in the file selector panel). Once the correct file has been selected and the **APPLY** button pushed, the SAP2000 translator will begin.

There are several items of note that the user should be aware of prior to use of the translator function:

1. The translator program creates several binary scratch files during the translation process. You should ensure that sufficient space is available on your disk. As a general guide you may expect to require 10 Mb of space for a complex model containing 10000 frame elements. These files will disappear as soon as the translation is completed.
2. The speed of the translation is not impressive. Please be patient whilst frame elements are read into PRIMER, as these will most likely be the largest portion of your model. There is a running commentary in the DIALOGUE panel which will present the current status of the translation. If an excessive number of frame rigid offsets or end-releases are specified in the model this will considerably increase the amount of time required to complete the translation.



3. The DIALOGUE panel will contain useful information about the status of the translation and what the translator is doing. It is advisable to enlarge the panel prior to beginning the translation process so that the data can be viewed more easily. Any warnings printed in this panel may be important, so please check anything printed.

Main translation panel

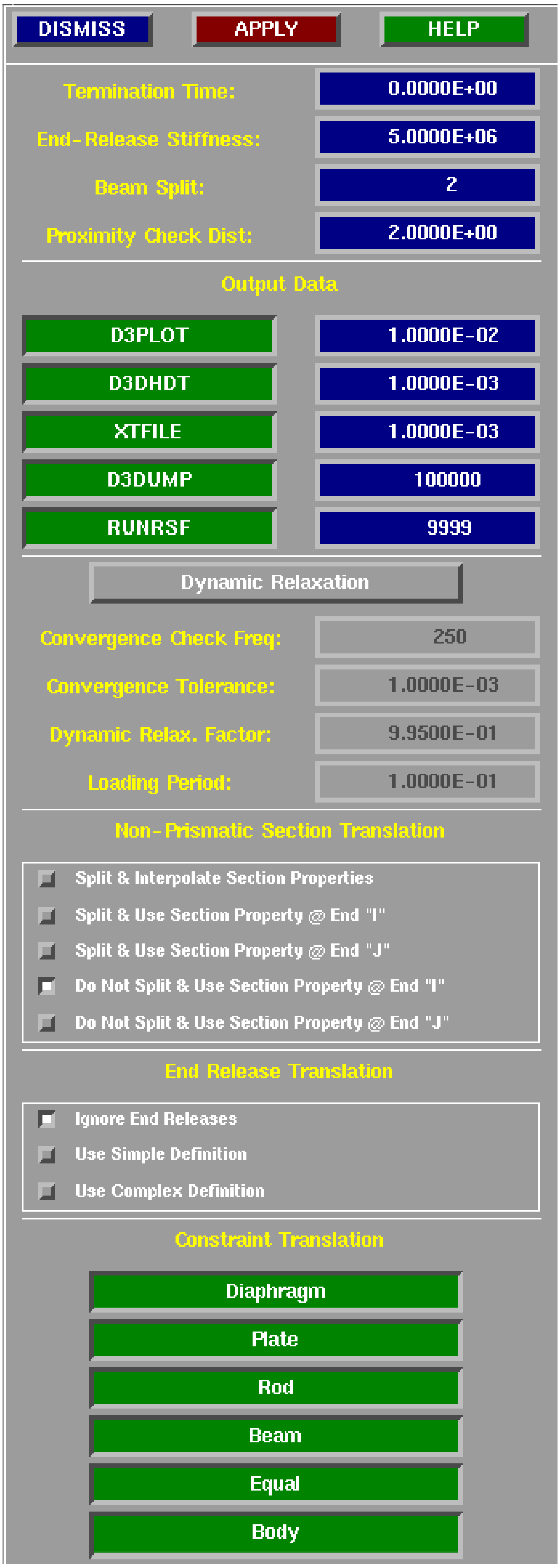
Before any data is read, a SAP2000 defaults panel will appear on the screen. This panel allows some options to be set prior to the translation process:

DISMISS terminate the translation process, returning to the generic READ panel.
APPLY accept the defaults in the panel and proceed with the translation.
HELP will create a message box full of useful information about the function of this panel.

Termination Time will set the final termination time of your transient LS-DYNA analysis. This value is also used to define loadcurve timing values.
End-Release Stiffness defines the stiffness of the discrete springs used represent pin releases at the end of frame elements. It is important the spring elements do not control the time step of the LS-DYNA analysis. If this occurs, the model may terminate due to numerical instabilities. To avoid allowing springs to dictate the time step of the model the stiffness should be small. However, this must be balanced against the need for the spring to keep the pinned nodes together. Refer to LS-DYNA User Manual for more data.
Beam Split defines the default number of beams into which a SAP2000 frame will be split if required - refer to the OPTIONS panel.
Proximity Check Dist defines a linear distance in the model units. If two nodes in the same rigid offset group are further apart than this distance a warning will be produced. This option is useful for checking the model.

Output Data allows the LS-DYNA DATABASE keyword cards which control the output frequency of results to be created in PRIMER. Each individual card can be switched on and off (green highlight implies on). The output frequency can only be modified when the card is 'on', else it is greyed-out.

Dynamic Relaxation allows dynamic relaxation (DR) options to be turned on and off (green highlight implies 'on'). When the DR option is selected, the following items can be edited:
Convergence Check Freq Number of cycles between DR convergence checks.
Convergence Tolerance value at which convergence is achieved.
Dynamic Relax Factor DR damping factor. This is a scale factor on velocity.
Loading Period Period over which the load which is being relaxed onto the model (i.e. self weight) is ramped to the full value. This 'ramping' avoids dynamic shock and oscillation in the model.



Non-Prismatic Section Translation
the concept of non-prismatic frame sections does not easily translate into LS-DYNA. For this reason a choice of several options is required. SAP2000 allows the section type to be defined at either enc of the non-prismatic frame element. These sections are then used to define the intermediate section properties. The options are as follows:

- Split the frame element into ’nseg’ beam elements (nseg is defined on the SAP2000 frame element card), and interpolate the section data at either end of the frame creating new frame sections. Note that this can become overly messy if nseg is set to greater than 2. Also note that nseg can be overridden by the beam splitting options in the OPTIONS panel.
- Split the frame element as before but use the section data from end ’I’ all the way through the frame.
- Split the frame element as before but use the section data from end ’j’ all the way through the frame.
- Do not split the non-prismatic frames (unless specified otherwise) and use section ’I’.
- Do not split the non-prismatic frames (unless specified otherwise) and use section ’j’.

Refer to the Frame Element section of this appendix for more details on frame splitting and non-prismatic frames.

End Release Translation end releases (i.e. pin ends) also do not have an equivalent in LS-DYNA. Hence there are three options:

- Ignore end releases entirely.
- Use the simple definition.
- Use the complex definition.

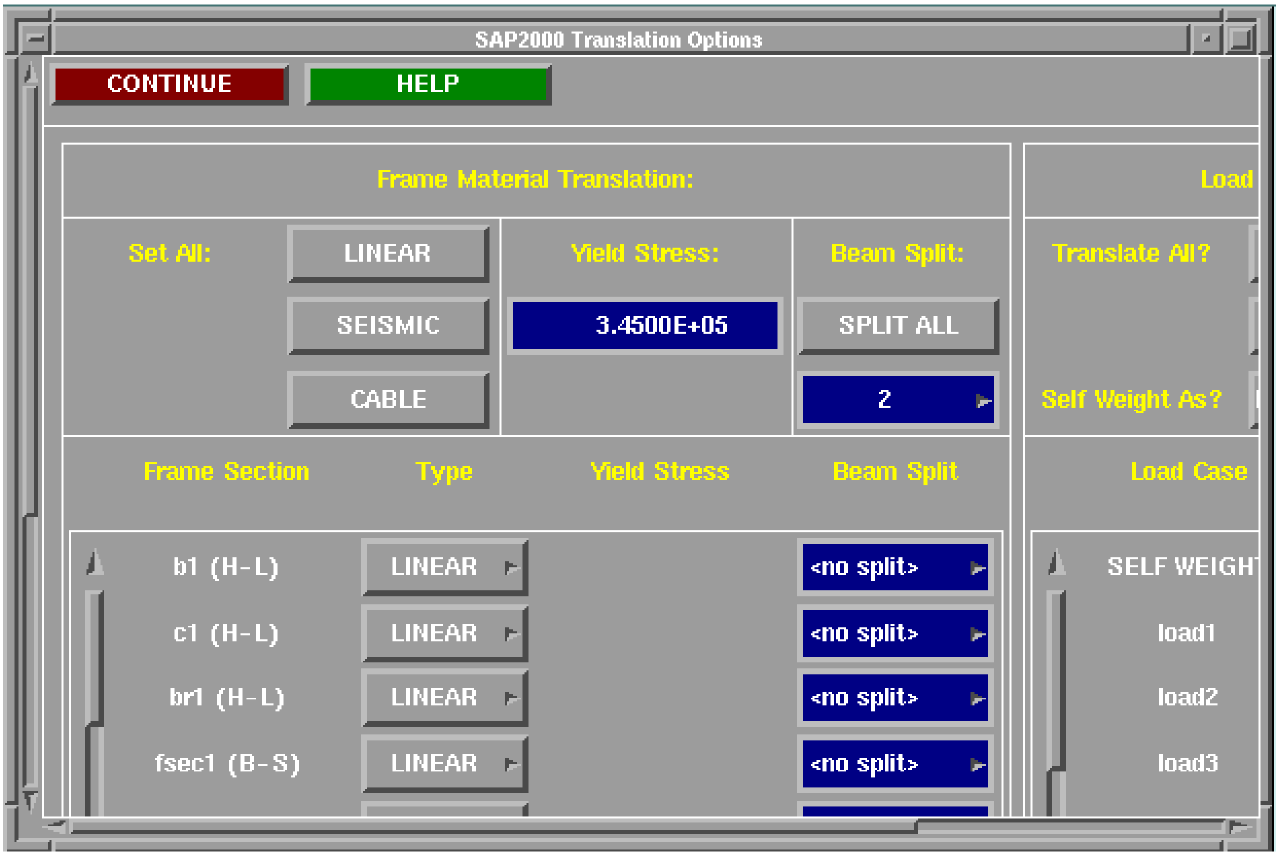
Please refer to the Frame Element section of this appendix for more details.

Constraint Translation Many of the constraint types in SAP2000 cannot be converted into LS-DYNA. This portion of the panel allows each type of constraint to be selected individually:

On (green highlight); all constraints of this type will be converted into nodal rigid bodies with all degrees of freedom connected.
Off (greyed out); all constraints of this type will be ignored.

Once the **APPLY** button from the main panel has been pushed the translation process will begin.

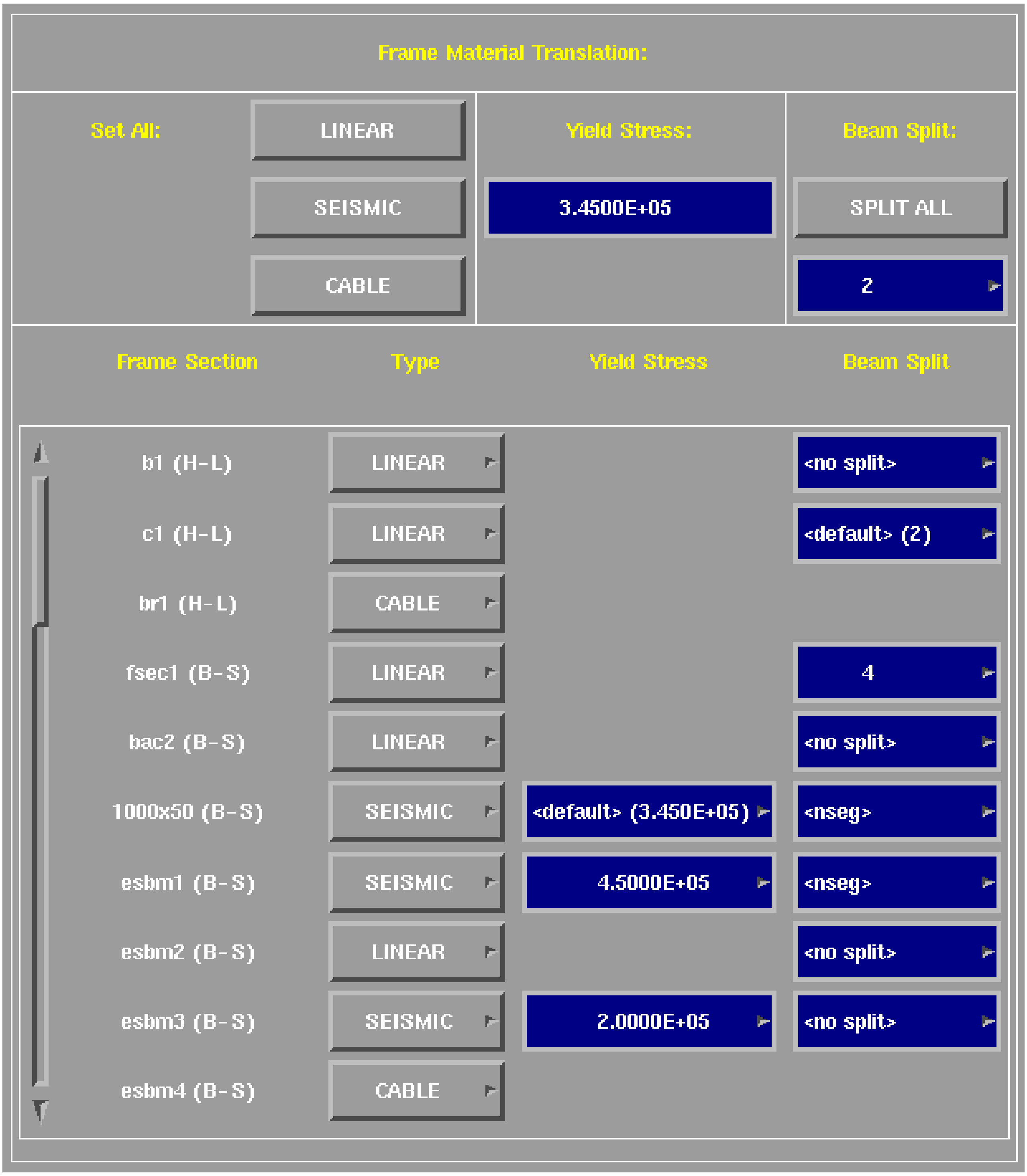
Stage one of the process reads all of the data from the SAP2000 ascii file and stores the majority of the data in the binary scratch files. Once stage one is completed a second window panel is created: SAP2000 OPTIONS - see figure below.



This panel is split into three areas; the static top area (**CONTINUE** - proceed with translation, and **HELP**), a frame options region and a loading options region.

Frame Translation Options

this panel allows the user to modify the material type of a frame component (part), or decide whether to split frame elements on an individual part basis.



The material options are as follows:

LINEAR: standard linear elastic material.

SEISMIC: will convert the standard linear material into a non-linear 'seismic beam' material capable of modelling plastic hinges and failure. The plastic moments of these frame components are automatically calculated provided that the geometry of the frame section was included in the SAP2000 ascii file. This shape is combined with the yield stress of the material to calculate the plastic properties. Where frame elements are expected to fail due to applied moments it is advisable to split seismic beams into at least 2 beams in order to capture the correct moment at each end. Note that;

seismic beams can only be created from Belytschko-Schwer beam sections (B-S).

CABLE: converts the standard frame sections to linear elastic discrete springs. These elements cannot be split and will ignore any end release definitions on frames in a ’cable’ component - end releases have no meaning!

SET ALL enables all frame components to be set to a specific material type. The global **Yield Stress** and **Beam Split** options are adjacent. These global values become default values which apply to the entire model.

In the global **Beam Split** area all beams can be split with the current default value (i.e. the value from the main panel, ’nseg’ (defined for each frame element in the SAP2000 ascii file) or some other even number typed in). A further option **SPLIT NONE** is available if no beams are to be split. A pop-up box can be displayed (by clicking the right mouse button) from the text box in order to define the default value.

Below the ’global’ area is a list of frame component names, followed by data relevant to each individual component. If more than 10 components exist in the model the slider can be used to scroll through the components.

Each frame component name is followed by ’(H-L)’ implying Hughes-Lieu beam section or ’(B-S)’ for Belytschko-Schwer beam sections. As mentioned earlier, this enables the user to distinguish between components which can and cannot be modified to seismic beams. Refer to the Frame Section part of this appendix for details on how the frame section type is chosen in the translation.

The material **Type** is displayed in the next column. This can be chosen by cycling through the material options by clicking the left mouse button with the curser positioned over the screen button, or using the right mouse button to invoke a pop-up box.

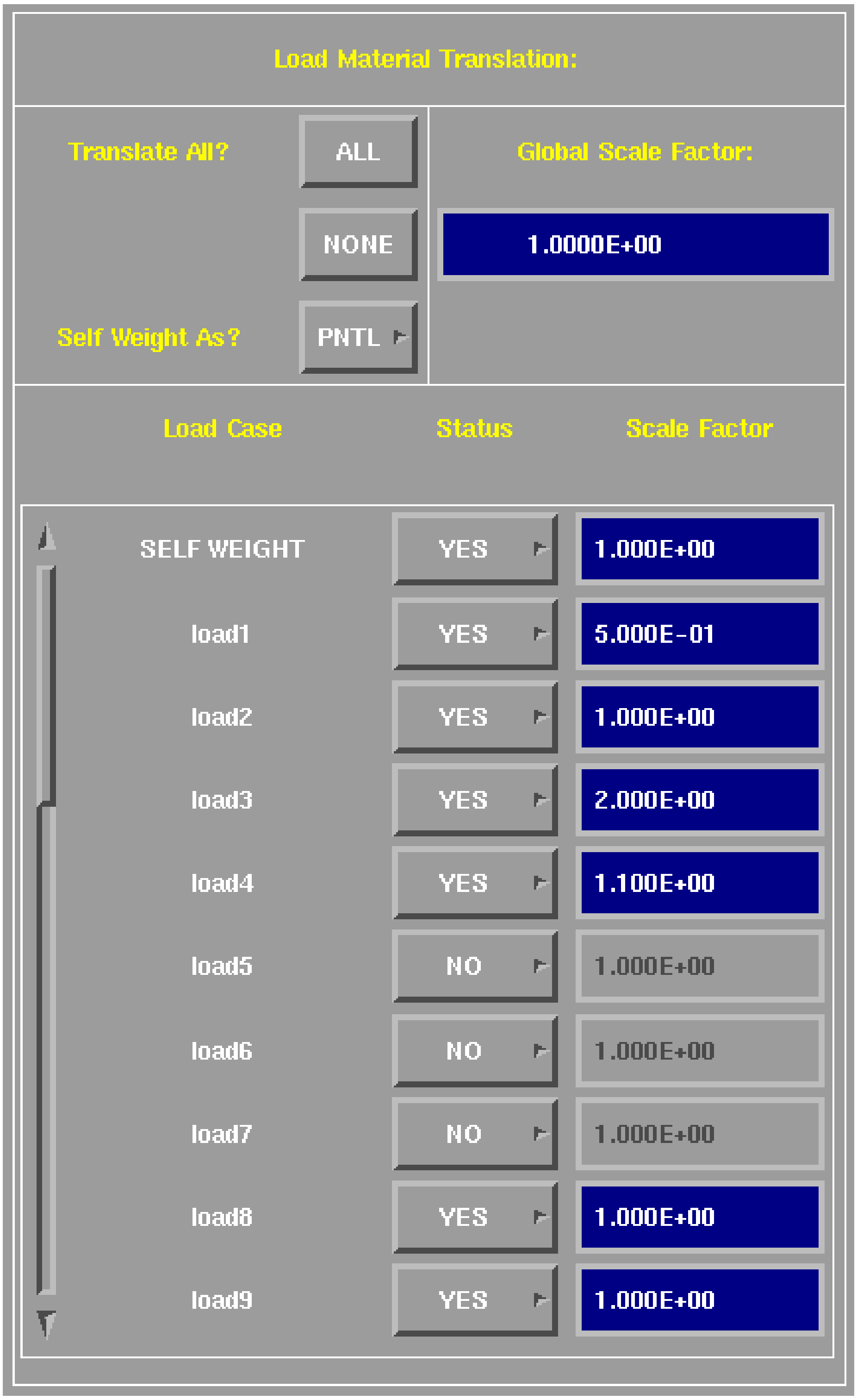
Yield Stress is only appropriate for seismic materials, and will only become available if a seismic material is chosen. The yield stress value can be typed in directly or the right mouse button can be used to invoke a pop-up which will reset the current default value.

The last column contains the **Beam Split** option for individual components. The options are: type a value in directly (even numbers only) or use the right mouse button to invoke a pop-up to choose <no split>, <nseg> or <default> for ’do not split this component’, ’split using the nseg value on the frame element card’ or ’split using the default value above’. The beam split option will be unavailable for cable materials as it is inappropriate.

Load Translation Options

Translate All allows the user to request all or none of the load cases defined in the SAP2000 ascii file to be translated.

Self Weight As? allows the self weight of the model to be represented in either of two ways: **PNTL** (POINT LOADS) where each node in the model that has weight associated with it will have a point load applied to it loading in the negative (downward) Z direction, or **GRAV** (GRAVITY) where the loading is applied as a gravity acceleration to the mass of the model. Use the self weight scale factor to multiply by the gravitational constant (i.e. 9.81 m/s2) for the **GRAV** option. The **PNTL** / **GRAV** option is chosen by invoking a pop-up box with the right mouse button.



A **Global Scale Factor** for all load cases can also be set in this window. Note that each load case that is selected for translation will be multiplied by the product of its own individual scale factor and the global scale factor.

The names of all load cases that are not empty are printed in a list in the panel. If there are more than 10 load cases then a slider will be created to enable the user to scroll through the load cases. Each separate load case can be selected and deselected by changing the load **Status** to **YES** (select) or **NO** (deselect). Where a load is selected for translation, an individual **Scale Factor** can be typed directly in. If the load is not selected, then the individual scale factor will be greyed out.

Once happy with the chosen options, the **CONTINUE** button should be pushed to complete the analysis.

Translator Functionality

This section of the appendix is meant to give a brief insight into the methods that the translator uses to process and store data, and the types of SAP2000 data which are understood.

Alternative Coordinate Systems

Alternative coordinate systems are supported by the translator in cartesian, cylindrical and spherical forms. Internally these are all mapped onto a cartesian system. The coordinate system will only be installed in the PRIMER database if it is specifically referenced by an entity in the model; i.e. a nodal restraint.

Joints

Joints (or nodes) can be read into PRIMER in cartesian, cylindrical or spherical coordinates systems (global or alternative). The joint coordinates are always transformed and stored in a global cartesian system. Note that the joint number will be retained in the final LS-DYNA model.

SAP2000 JOINT = LS-DYNA `*NODE`

Local Joint coordinate Systems

These are supported, but only when defined in a cartesian coordinate system. These joint local systems are stored identically to normal coordinate systems.

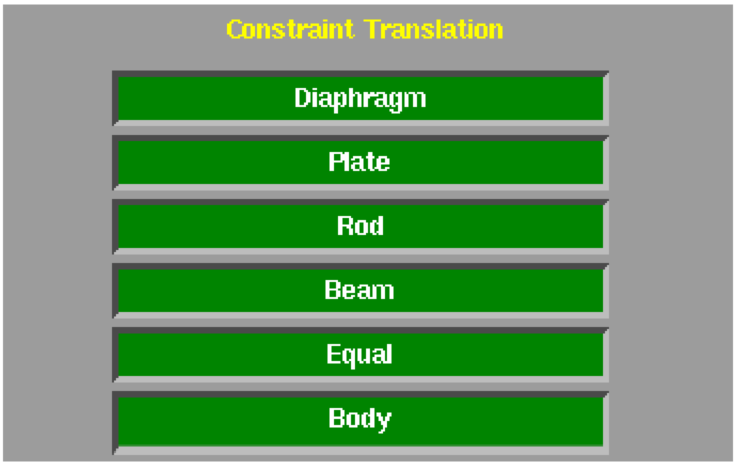
Restraints

Restraints are supported in either local, alternative or global coordinate systems (cartesian only). If a joint references a coordinate system, the coordinate system will be installed in the PRIMER database.

SAP2000 RESTRAINT = LS-DYNA `*BOUNDARY_SPC`
SAP2000 COORDINATE SYSTEM = LS-DYNA `*DEFINE_COORDINATE_SYSTEM`

Constraints

The following constraint types are recognised by the translator - the rest are ignored:



Each constraint type will be translated into PRIMER as a nodal rigid body where all degrees of freedom are constrained together for all nodes in the constraint group irrespective of the constraint type. Alternatively, it is possible to turn off and ignore constraints of a specific type.

SAP2000 CONSTRAINT = LS-DYNA `*CONSTRAINED_NODAL_RIGID_BODY`

Please refer to the section on Rigid Links etc for more details.

Welds

Welds are supported. If the nodes in the group are within the required distance, the nodes are grouped together in a nodal rigid body.

SAP2000 WELD = LS-DYNA `*CONSTRAINED_NODAL_RIGID_BODY`

Please refer to the section on Rigid Links etc for more details.

Patterns

Both gradient and standard pattern types are supported, but are stored in a binary scratch file. These are never installed in the PRIMER database as there is no LS-DYNA equivalent.

Spring Elements

Spring elements in SAP2000 are roughly equivalent to grounded springs in LS-DYNA, having only one node.

Spring elements in LS-DYNA cannot be coupled across freedoms, hence only the diagonal terms of the stiffness tensor of a coupled spring are translated. Each different component is translated as an individual element.

A new node is created adjacent to the original node. The new coexistent node is then fully restrained using a `*BOUNDARY_SPC`.

The orientation of each spring component is preserved by creating spring-damper orientation vectors (`*DEFINE_SD_ORIENTATION`). The translator will first check to see if an existing vector is parallel to the required direction. If a parallel vector is found it is referenced by the discrete element. Else, a new sd-orientation vector is created.

SAP2000 SPRING ELEMENT = LS-DYNA collection of `*ELEMENT_DISCRETE`s

Mass Elements

Mass elements are translated, and can be defined in global, local or alternative coordinate system space.

The translational mass defined in SAP2000 has directional components. This is not supported in LS-DYNA, hence the average value is used. Rotational masses are transformed into a global inertia tensor.

SAP2000 MASS ELEMENT = LS_DYNA `*ELEMENT_MASS` & `*ELEMENT_INERTIA`

Materials

Elastic isotropic materials are the only material type translated into LS-DYNA (Young’s Modulus, Poisson’s Ratio, mass density and weight density). All thermal materials are ignored.

SAP2000 MATERIAL = LS-DYNA `*MAT_ELASTIC`

Frame Elements & Sections

Frame Sections

PRISMATIC FRAMES:

The `*SECTION_BEAM` formulation used for a particular frame section depends on the data available: if the area (A) and the second moments of area (Iss, Itt & J) are present in the SAP2000 ascii file, a Belytschko-Schwer (resultant) beam formulation will be used (note that the shear area defaults to half the geomertic area, and that if different values of shear area are defined for the ’s’ and ’t’ axes an average value will be used), else the section geometry will be used to define a Hughes-Lieu (integrated) beam formulation. If there is inadequate data for either resultant or integrated formulations are present, then the translator will error terminate. Also note that for the more complex section shapes, if a Hughes-Lieu beam section is created, the associated `*INTEGRATION_BEAM` cards will also be generated. The component (`*PART`) will tie together the frame section name, the newly created section card, and the associated material.

NON-PRISMATIC FRAMES:

These frame sections are defined using existing frame sections. In these cases two new parts are created which reference the existing beam section definitions. Both parts will use the same frame section name.

SAP2000 FRAME SECTION = LS-DYNA `*SECTION_BEAM` (& `*INTEGRATION_BEAM` where required) & `*PART`

Frame Orientation

The beta angle system of orienting frame elements in SAP2000 is converted into a third node system in LS-DYNA. In the process of conversion the element axes become reversed (i.e. if the strong axis direction in SAP2000 was axis 1, then it would become axis 2 in LS-DYNA). This does not cause concern though as all frame elements will retain the correct orientation with respect to strong and weak axes.

Rigid Offsets

Where horizontal beams interface with vertical columns, the centre line geometry attributed to a one dimensional beam element is incorrect. The horizontal beam elements should terminate at the outer face of the vertical column. To facilitate this rigid links are created.

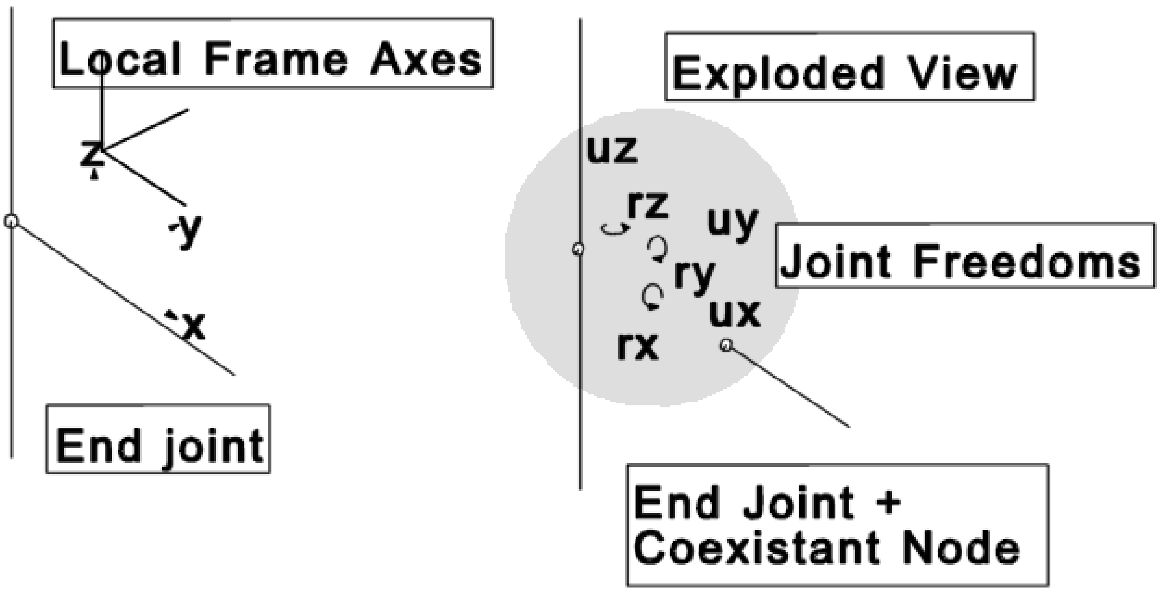
These rigid offsets are modelled in LS-DYNA as groups of rigid beam elements with small mass. Each rigid offset group will have its own component part. In this way the separate rigid offset groups can move independently. If the master node (i.e. the node at the centre of the rigid offset) has any kind of restraint associated with it, the rigid material definition associated with that particular rigid offset will also be restrained in a similar manner. However, these master node restraints must be in the global axis system.

The proximity check distance can be used as a model checking feature.

If any two nodes in the same rigid offset are further apart than the specified distance a warning will be printed in the dialogue box.

Please refer to the section on Rigid Links etc. for more details on how the rigid offsets are dealt with.

End Releases



End releases are used to 'release' a selection of degrees of freedom at the end of a frame element. Note that if the frame element has a rigid offset, the end release will occur at the point that the flexible region of the frame attaches to the rigid region (i.e. at the free end).

The translation of end releases into LS-DYNA will vary according to which translation option is selected in the DEFAULTS panel.

1. Simply ignore the end release definitions.
2. Use the ’simple’ definition of end releases. This is a fairly crude translation, but utilises the fact that the majority of end releases are simple pins; i.e. all rotational freedoms (rx, ry & rz) are released leaving the translational freedoms fixed (ux, uy & uz). The simple end release definition separates the released joint into two coexistent nodes. These two nodes are then connected by a single translational spring with a stiffness defined on the DEFAULTS panel.

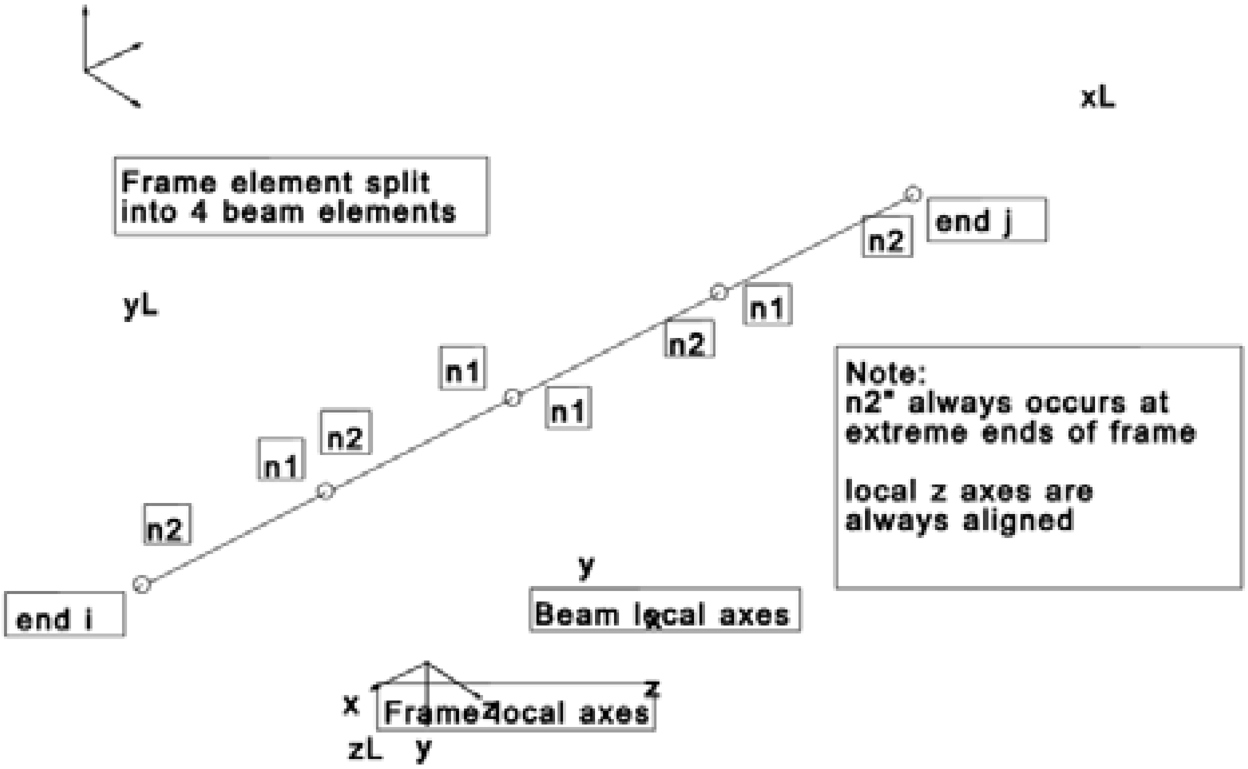
   The spring has no orientation vector and merely holds the two nodes together while allowing the two nodes to rotate freely relative to each other. This form of end release, while fast to translate and simple to understand, it does not read which freedoms have been released: it always assumes a pure pin.
3. The ’complex’ end release definition is by far the most correct method, but it can become very slow and difficult to implement. With this in mind, it should not be used where there are a huge number of end releases to be translated in the model. The end release itself is a combination of linear constraint equations, nodal restraints and oriented discrete springs. The following table outlines what is used where and when:
   - ux, uy & uz are not released;
     The LS-DYNA `*CONSTRAINED_LINEAR` keyword is used to create a set of linear constraint equations between the two coexistent nodes linking all translational freedoms. Note that if either of the two nodes is included in any rigid part or an existing constraint a series of oriented springs will be used - see below.
   - rx, ry & rz are not released;
     As type (I) but for all rotational freedoms.
   - uy & uz are not released, ux is released;
     One translational spring defined to work in the plane normal to the axis of the beam.
   - ry & rz are not released, rx is released;
     One rotational spring defined to work in the plane normal to the axis of the beam.
   - rx is not released;
     A nodal restraint is used to prevent the node on the beam side of the pair of nodes from twisting. A coordinate system is created to ensure that the restrained freedom is aligned with the axis of the beam.
   - all other combinations;
     Translational or rotational springs using spring-damper orientation vectors are used to hold the non-released freedoms together.

Note that where oriented springs (using spring-damper orientation vectors) or the coordinate system in the fifth case are used, the program searches through all existing vectors or coordinate systems to locate a parallel vector or coordinate system. Only if a parallel vector / system cannot be found will a new vector / system be created. This searching for parallel vectors can take a considerable amount of time on large models.

Frame Splitting

It is often desirable to split a frame element into a number of beam elements. Where splitting occurs one of the beam elements within the frame will use the element number of the original frame element. Also the two extreme node numbers will be retained.

Frames can only be split into an even number of beams, and the mid node will always be equally spaced between the two end nodes even if different length end releases are specified for the frame.



The geometry of the split frame is shown above. As shown in the figure beams are oriented in such a way that the 'end 2' node of a beam always occurs at the extreme end of the frame. This is because seismic beams only develop plasticity at end 2. Hence if a seismic beam is split, it can develop a plastic hinge at both ends. The necessity of having end 2 nodes at the extreme ends of the frame requires the local beam coordinate system to be swapped halfway along the frame. Note that the local 'z' axis (generally the strong axis) is maintained in the same direction whilst the local 'y' axis is modified.

Defining which beams are to be split is done on a component by component basis. Each frame component can be split into a different number of beam elements. The basic choices are as follows:

1. Split into 'nseg' (or nearest even number), as defined for the frame element in the SAP2000 ascii deck.
2. Split into the default number of beams which is set on the SAP2000 main panel.

3. Split into an arbitrary number of beam elements as defined on the frame translations options panel.

A fourth option 'no split' can be specified. If this option is selected the frame component will not be split.

Non-prismatic sections which are split require an additional piece of information to be defined: what section properties are to be used. The choices are:

1. Use the section properties from end one for all beams.
2. Use the section properties from end two for all beams.
3. Interpolate the section properties along the length of the frame.

This option is a global parameter set for the whole model on the SAP2000 main panel.

Where the section data is interpolated from end one to end two, entirely new parts and sections are created. Clearly this has the potential for becoming very messy if the frame is split into too many beams each with their own section.

Frame Material Conversion

The basic translation converts all frame elements into linear elastic beams. However, two further options are available:

Seismic beams (only available for Belytschko-Schwer beam formulations)

A new `*MAT_SEISMIC_BEAM` material is generated and the plastic data calculated from the yield stress and the section geometry data defined in the SAP2000 ascii deck. If no geometry data is present then default unit values will be used for the plastic data which must be modified prior to running the analysis.

Cables

Frame components flagged as cable elements will be turned into discrete spring elements. Currently the spring materials are linear elastic with stiffness as the product of Young's modulus and cross sectional area. Each element, however, has a scale factor on the stiffness of the inverse of the element length. Hence the element stiffness is EA/L. Note that cables cannot be split and that end releases are ignored as they have no meaning in this context.

Mass and Self Weight

Additional non-structural mass is distributed applied to the model as lumped mass elements at the free ends of the frame, and where the frame is split also at all mid-nodes.

A similar method is used to distribute the self-weight and additional non-structural self-weight. Point loads are created at the free-ends of the frame and, if split, at the mid-node. If the frame has been split then the point loads will be arranged such that the correct moment will be achieved at the free ends of the frame - refer to the loading section for more details on this arrangement.

Shell Elements & Sections

Shell Sections

The SAP2000 'Shell' and 'Membrane' formulation are identically translated into LS-DYNA. However the 'Plate' formulation does not exist. In this case the translator employs a standard shell formulation.

SAP2000 SHELL SECTION = LS-DYNA `*SECTION_SHELL`

Shell Elements

Shell elements are directly translated into LS-DYNA, although the topology for the element has to be modified. The same element number is used in LS-DYNA as for SAP2000.

SAP2000 SHELL ELEMENT = LS-DYNA `*ELEMENT_SHELL`

Self Weight

The weight associated with each node on the shell element is estimated, and applied to the element as a series of point loads in the negative z-direction.

Solid Elements & Sections

Solid Sections

SAP2000 does not define any section properties for solid elements. Instead all the 'section' data is lifted directly from the material definition. LS-DYNA requires that these sections are defined.

SAP2000 [SOLID SECTION] = LS-DYNA `*SECTION_SOLID`

Solid Elements

Solid elements are translated directly into LS-DYNA, although the topology of the element has to be modified. The shorthand definition of the solid element is also supported by the translator.

SAP2000 SOLID ELEMENT = LS-DYNA `*ELEMENT_SOLID`

Self Weight

The weight associated with each node on the solid element is estimated, and applied to the element as a series of point loads in the negative z-direction.

Nllink Elements & Nlprop Data

Nonlinear properties (Nlprop)

The nonlinear properties are defined in terms of shear properties are non-shear properties. The various property types and their LS-DYNA counterparts are list below:

SAP2000 NLPROP (no type) = LS-DYNA `*MAT_SPRING_ELASTIC` (stiffness = ke, effective elastic stiffness from SAP2000 definition) (shear & non-shear)

SAP2000 NLPROP (damper) = LS-DYNA `*MAT_SPRING_ELASTIC` (stiffness = ke, effective elastic stiffness from SAP2000 definition) & `*MAT_DAMPER_VISCOUS` (damping coefficient = c, from SAP2000 definition) (shear & non-shear)

SAP2000 NLPROP (gap & hook) = LS-DYNA `*MAT_SPRING_ELASTIC` with clearance defined on the discrete section card (stiffness = ke, effective elastic stiffness from SAP2000 definition) (shear & non-shear)

SAP2000 NLPROP (plastic1) = LS-DYNA `*MAT_SPRING_ELASTIC` & `*MAT_SPRING_ELASTOPLASTIC` (These two springs combine to give a kinematic hardening hysteresis response. The elastic spring controls the post-yield stiffness and the elastoplastic spring defines the yield strength. The pre-yield strength of the elastoplastic spring when added to the elastic spring gives the correct pre-yield stiffness, ke. The post-yield stiffness of the elastoplastic spring is zero.) (shear and non-shear)

SAP2000 NLPROP (isolator1) = LS-DYNA `*MAT_SPRING_ELASTIC` (stiffness = ke, effective elastic stiffness from SAP2000 definition) (non-shear) LS-DYNA `*MAT_SPRING_ELASTIC` & `*MAT_SPRING_ELASTOPLASTIC` (as for 'plastic1' type) (shear)

SAP2000 NLPROP (isolator2) cannot be translated

Nonlinear Link Elements

A single nllink element will be translated into a collection of discrete elements using spring-damper orientation vectors to imitate the local coordinate system to the nllink element.

The first element discrete element defined will use the same element number as the nllink element.

Where zero length single node elements have been defined in SAP2000, the translator will generate a second node and then fully restrain it.

SAP2000 NLLINK = LS-DYNA `*ELEMENT_DISCRETE` (collection of ...)

Mass, Inertia & Self Weight

The mass associated with nllink elements can be defined for each translational direction. This is not possible in LS-DYNA, so the average mass is used and distributed evenly between the two nodes defining the nllink element.

The inertia of the nllink element is converted into a global inertia tensor and then evenly distributed between the two nodes defining the nllink element.

The self weight of the nllink element is distributed evenly between the two nodes and represented as point loads.

Loading

Each load case defined in the SAP2000 model can be individually selected or deselected. Each selected load case has a scale factor applied to it (default = 1.0) for the translation. The translator will then take all load cases that have been selected and apply them to the model. A global scale factor is also defined which is applied to every load in the model on top of the individual scale factor discussed previously. Note that each load case that is selected will be associated with its own load curve. This enables the scale factor on each load case to be changed at a later date.

The self weight load case is a special case. The self weight loading can either be applied to the model as a series of point loads which have been defined according to the self weight of each element in the model (which may or may not be equivalent to the product of mass and gravitational acceleration), or as a acceleration body load to the whole structure (`*LOAD_BODY_Z`). If the later is the case, the scale factor on the self weight load case should be the gravitational acceleration parameter (i.e. 9.81 m/s2). The point loads should need no additional scale factor as these will already be defined in terms of structure weight.

Joint Forces

These are simple point loads. Each point load defined will be split into a global load vector comprising of up to three point loads, each parallel with a global axis.

SAP2000 JOINT FORCE = LS-DYNA *LOAD_NODE_POINT

Joint Displacement

These are nodal displacement boundary conditions. Each displacement boundary condition is split into a global load vector, each non-zero component of the vector having a separate displacement boundary condition.

SAP2000 JOINT DISPLACEMENT = LS-DYNA *BOUNDARY_PRESCRIBED_MOTION_NODE

Spring Displacement

As for Joint Displacement, but for the grounded node of a spring element.

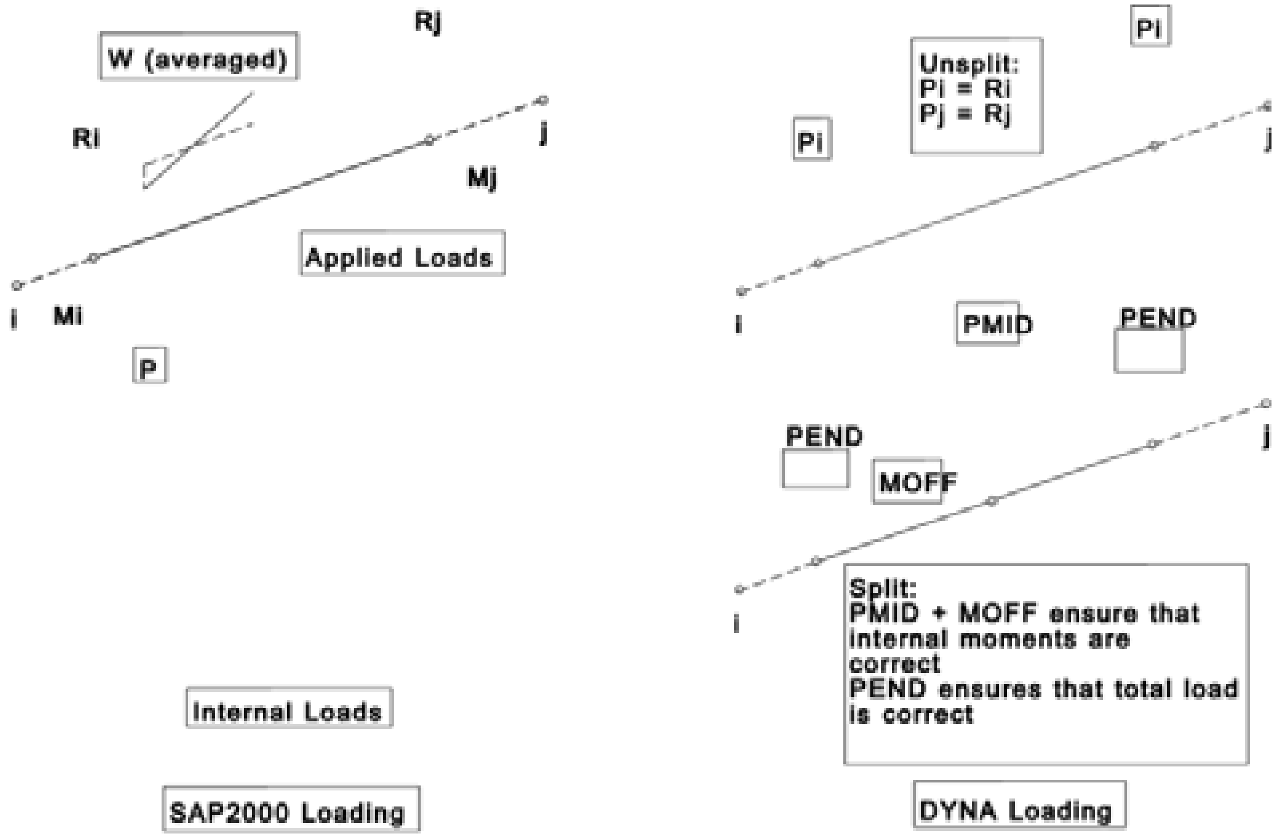
SAP2000 SPRING DISPLACEMENT = LS-DYNA *BOUNDARY_PRESCRIBED_MOTION_NODE

Concentrated & Distributed Span Loads

All frame loading types are supported by the translator, converting the applied load into a global load vector. Each non-zero component of the load vector is then split into a collection of point loads.

If the frame element to which the load is applied has not been split, two point loads are applied to the free ends of the frame. The point loads are equal and opposite to the reaction force the beam would create if it were fully fixed.

If the frame element has been split then the loading arrangement attempts to ensure that the correct moment is achieved at the free ends of the frame. A point load and moment are applied to the mid-node of the frame which create the same end moment conditions as the original loading condition. Two additional point loads are used to balance up the total load on the frame.



SAP2000 SPAN LOADING = LS-DYNA *LOAD_NODE_POINT

Uniform Shell Loading

All forms of uniform shell loading are translated. The applied load is split into a global load vector. The area associated with each node on the shell element is estimated and for each non-zero component, a point load is applied to each node on the element.

SAP2000 UNIFORM SHELL LOADING = LS-DYNA `*LOAD_NODE_POINT`

Surface Loading

The surface loading for shells (faces and edges) and solids (faces) is applied normal to the face of the element. This load is then converted into a global load vector and a point load applied to each node on the face / edge for each non-zero component.

SAP2000 SURFACE LOADING = LS-DYNA `*LOAD_NODE_POINT`

Dynamic Relaxation Loading

If dynamic relaxation is required then an additional set of loads will be created for dynamic relaxation only, except for any displacement loads. The load curve used to apply dynamic relaxation loads will ramp the loading from zero to the full working load. This ramping period is used to prevent dynamic shock to the model caused by instantaneous application of loading.

Rigid Links and Rigid Bodies

A node in LS-DYNA cannot be a member of more than one rigid body or constraint system. If such a case did occur, LS-DYNA would fail to initialise. To avoid such a problem the translator performs a check to ensure that no nodes occur in more than one nodal rigid body or offset. If a clash is detected, then the two rigid groups of which the offending node is a member will merged into one new rigid group.

IGES file format

The International Graphics Exchange Specification (IGES) is a standard format for model geometry. Primer will read fixed ASCII file format 5.3 IGES files.

The following IGES entity types are read by PRIMER.

Entity type	**Entity description**	**Notes**
100	Arc	
102	Composite curve	
104	Conic arc	Forms 2 (hyperbola) and 3 (parabola) not supported.
106	Copius data/Linear path/Simple closed planar curve	Forms 1, 2, 11, 12 and 63 supported
108	Plane	
110	Line	Forms 1 (semi bounded) and 2 (unbounded) not supported
112	Parametric spline curve	
116	Point	
118	Ruled surface	
120	Surface of revolution	
122	Tabulated cylinder (extruded surface)	
124	Transformation matrix	
126	Rational B-Spline curve	
128	Rational B-Spline surface	
141	Boundary	
142	Curve on parametric surface	
143	Bounded surface	
144	Trimmed (parametric) surface	
308	Subfigure definition	
314	Colour	
406	Property	Only forms 3 (level function) and 15 (Name) supported
408	Singular subfigure instance	

All other entity types are ignored.

APPENDIX VII: Format translation during MODEL> WRITE

ABAQUS "Input" file output.

The following table shows the limited set of DYNA keywords that are supported for conversion to Abaqus .inp file format. The translator leaves the internal LSDYNA data unchanged.

Internal LS-DYNA keyword	**Abaqus written output**	**Note**
*CONSTRAINED_INTERPOLATION	*ELEMENT,TYPE=DCOUP3D, *ELSET= **KINEMATIC COPULING** or **DISTRIBUTING COPULING**	
*CONSTRAINED_NODAL_ RIGID_BODY	*MPC	[1]
*CONTACT_NODES_TO_SURFACE	*CONTACT_PAIR *SURFACE *SURFACE_INTERACTION	
*ELEMENT_BEAM	*ELEMENT	[2]
*ELEMENT_SHELL	*ELEMENT	[3]
*ELEMENT_SOLID	*ELEMENT	[4]
*MAT_ELASTIC	*MATERIAL *DENSITY *ELASTIC	
*MAT_PIECEWISE_LINEAR_PLASTICITY	*PLASTIC	
*NODE	*NODE	
*SECTION_BEAM (ELFORM = 1)	*BEAM SECTION	[5]
*SECTION_BEAM (ELFORM = 12)	*BEAM GENERAL SECTION	[6]
*SECTION_SHELL	*SHELL SECTION	
*SECTION_SOLID	*SOLID SECTION	
*SET_NODE	*NSET	
*SET_SHELL	*ELSET	
*SET_SOLID	*ELSET	
*TITLE	*HEADING	

Notes:

1. *CONSTRAINED_NODAL_RIGID_BODY cards are converted to *MPC of **BEAM** type only.
2. *ELEMENT_BEAM crads are converted to *ELEMENT with **Type B31** only.
3. **Three noded shells** are written as *ELEMENT, **TYPE=S3R** while **four noded shells** are written as *ELEMENT, **TYPE=S4.**
4. Four noded, six noded and eight noded solids are converted to types C3D4, C3D6 and C3D8 respectively.
5. All *SECTION_BEAM cards with elform =1 are converted to ***BEAM SECTION with SECTION=CIRC**.
6. All *SECTION_BEAM cards with elform =12 are converted to ***BEAM GENERAL SECTION with SECTION=GENERAL**.

IDEAS (Master Series) Universal file format.

This section shows which LS-DYNA keywords are supported, and how they are translated to entities in an I-DEAS universal file.

Primer uses the same conventions as N/CODE for the universal file so N/CODE can be used to re-translate the universal file back into a keyword deck.

Primer carries out several checks when writing a universal file to ensure that there are no problems when importing the universal file into I-DEAS.

These are:

- I-DEAS needs unique labels for every element. During output of the universal file Primer will check and fix any element clashes which occur, printing a warning about what has been renumbered. If this happens the model in Primer is renumbered, not just the output in the universal file. If you do not want the elements renumbered in your original LS-DYNA deck then ensure the keyword deck is saved before writing the universal file.

- I-DEAS does not allow Physical names longer than 40 characters in the universal file and does not allow duplicate names. As LS-DYNA allows part names up to 72 characters long there is a potential problem with names being duplicated. Primer truncates any part names in the LS-DYNA deck to 36 characters and then checks to see if any duplicate part names exist. If there are duplicates then '_1' is added to the first duplicate, '_2' to the second etc. to ensure names are unique.

- Node sets in I-DEAS (used for rivets, spotwelds, constrained node sets and generalised welds) need an independent node and at least one dependent node. Primer checks to see if any have been defined that use less than 2 nodes. If any have a warning is displayed and the node set is not translated.

- Springs in I-DEAS master series do not allow a material to be associated with them. To deal with this when translating a universal file to a LS-DYNA keyword deck N/CODE attempts to create a material with the same id as the physical id. This can cause problems so when Primer creates a universal file it checks any parts which contain discrete elements. If a material already exists with the same id as the discrete part then the part is renumbered so that N/CODE will not encounter any problems. This renumbering will not occur if the discrete part refers to a material with the same id.

The universal file import in Primer is very basic. At present only nodes and elements are read. To read data from a universal file N/CODE should be used to create an LS-DYNA keyword deck. This can then be read into Primer.

Supported keywords

*Materials. Structural and thermal material tables are written to the universal file (module 773). In addition to the structural materials dummy materials are written for lumped masses, sliprings, retractors, joints, stonewall plates, contact surfaces and airbag segments. Materials for conventional elements get written as isotropic materials with null properties. Materials for springs get written out as null materials with no properties.

*
Physical property tables (module 772 for I-DEAS V, module 778 for I-DEAS VI). Solid, beam, shell, thick shell, seat belt properties are written. The property number in I-DEAS is created from the part number in the LS-DYNA keyword deck. Dummy physical properties are written for lumped mass, slip rings, retractors, joints, extra nodes on rigid bodies, stonewalls, contacts and airbags

*
Beam cross section properties (module 776) are written for rectangular or circular section Hughes-Liu beams. A, I_{yy}, I_{zz} and J are written for Belytschko-Schwer beams

*
Nodes

*
Solid, beam, shell, thick shell, discrete and lumped mass elements are all translated (module 780)

*
Seat belts, retractors and sliprings are translated as green, read and blue rod elements (module 780) consistent with N/CODE

* Contact surfaces defined by either parts or segments are translated (module 2417). Groups are created in I-DEAS called CONTACT_SLAVE_n and CONTACT_MASTER_n. If the contact is a nodes to surface contact the slave group is called CONTACT_NODES_n or if the contact is a single surface algorithm a single group is created called CONTACT_SINGLE_n. Segments are translated as red and blue plate elements (slave and master). For contacts defined by parts one element from each part is placed in the group. This is consistent with N/CODE
* Extra nodes on rigid bodies are translated as pipe elements connecting the extra node to a node on the first element found in the rigid part. (module 780)
* Node and element time history blocks are translated (module 2417) by creating a group in I-DEAS called NODE_BLOCK or ELEMENT_BLOCK containing the history entities
* Restraints (**`*BOUNDARY_SPC`**) (module 755)
* Constraints (**`*CONSTRAINED_SPOTWELD, *CONSTRAINED_RIVET, *CONSTRAINED_NODE_SET, *CONSTRAINED_GENERALIZED_WELD_SPOT`**) (module 754)
* Point loads (module 782)
* Displacement, velocity and acceleration boundary conditions on nodes (module 2417). A group in I-DEAS is created called BC_DISP_n:(dof i), BC_VELO... or BC_ACCE... where n is a unique group number and i is the degree of freedom (1 to 6). Boundary conditions on rigid bodies cannot be written
* Nodal force groups (**`*DATABASE_NODAL_FORCE_GROUP`**) are written (module 2417) by creating a group in I-DEAS called REACTION
* Nodal rigid bodies are written (module 2417) by creating a group in I_DEAS called NODAL_RIGID n
* Joints (module 780)
* Initial velocities (module 2417) are written by creating I-DEAS groups called VELOCITY x y z, where x, y and z are the components of the initial velocity
* Rigidwalls are written (module 2417) by creating a group in I-DEAS called RIGIDWALL n. All the slave nodes for the rigid wall are placed in this group. If the rigidwall is finite in size a yellow plate element is created which represents the extent of the stonewall and this is also placed in the group. Only planar rigidwalls are written.
* Airbags are written (module 2417) by creating a group in I-DEAS called AIRBAG n. If the airbag is defined by segments they are translated as cyan plate elements. If it is created by parts one element from each part is placed in the group (compatible with N/CODE).

For further details on how LS-DYNA entities can be created in I-DEAS and how they are represented in a universal file see the N/CODE user manual.

Patran "Neutral" file output

Patran "neutral" (.ntl) file data can be written for both level 2.5 and level 3 versions.

As with universal file output elements of different types cannot share the same label, so prior to output a check is made for clashes and element labels are renumbered if required.The following data are output:

* **Analysis title** is translated directly as module 25

* **Nodes** are translated directly as module 1 (extra nodes are generated for grounded springs).

* **Elements** are translated as follows:

Dyna element type	**Level 2.5 output**	**Level 3 output**
8 noded solids (hexa)	<iv> = 8, config = 0	<iv> = 8, config = 4
8 noded thick shells (quad)		<iv> = 8, config = 1
6 noded solids (wedge)	<iv> = 7, config = 0	<iv> = 7, config = 4
6 noded thick shells (tria)		<iv> = 7, config = 1
4 noded solids (tetra)	<iv> = 5, config = 0	<iv> = 5, config = 4
4 noded shells (quad)	<iv> = 4, config = 0	<iv> = 4, config = 3
3 noded shells (tria)	<iv> = 3, config = 0	<iv> = 3, config = 3
Beams	<iv> = 2, config = 0, 1, 2 (= elform)	
Translational spring	<iv> = 2, config = 10	
Rotational spring	<iv> = 2, config = 11	
Translational damper	<iv> = 2, config = 20	
Rotational damper	<iv> = 2, config = 21	
Seatbelt element	<iv> = 2, config = 30	
Retractor	<iv> = 2, config = 31	
Slipring	<iv> = 2, config = 32	
Lumped mass	<iv> = 2, config = 7	

* **Structural materials** are translated as module 3:

type = 1 for deformable materials.
type = 2 for rigid ones in level 2.5 format, type = 1 in level 3 format.

Thermal materials are always translated as type = 1.

All material data fields are set to zero. "Fake" (empty) materials are also generated for those element types (eg lumped masses) that don’t have materials in Dyna, but do in Patran.

* **Part and section data** are written out as Patran "properties", packet type 4. The Dyna part id becomes the property id, and other data are generated as follows:

Element type	Common data	Level 2.5	Level 3.0
Solids:	<shape> = 8 <nvals> = 1 (material label)	config = 0	config = 4
Shells	<shape> = 4 <nvals> = 5 (matl label, t1... t4)	config = 0	config = 3
Thick Shell	<shape> = 8 <nvals> = 1 (material label)	config = 1	
Beams	<shape = 2> <nvals> = 5 (matl label, ts1 .. tt2)	config = 0	
Discrete	<shape> = 2 <nvals> = 1 (material label)	config = 20 (springs) config = 21 (dampers)	
Seatbelt	<shape> = 2 <nvals> = 1 (material label)	config = 30	

Where materials are not defined, for example in "latent" parts, "fake" (empty) material ids are generated and specified.

In addition a single property for each of the following element types is created, if they exist in the dyna model. Again "fake" (empty) materials are also generated for these:

Retractors	<shape> = 2, config = 31
Sliprings	<shape> = 2, config = 32
Lumped masses	<shape> = 2, config = 7

NASTRAN output

The following table shows the limited set of DYNA keywords that are supported for conversion to a Nastran bulk data file. The translator leaves the internal LSDYNA data unchanged.
For property cards $HMNAME comments are written, as are the $HMMOVE comments which maintain the PBAR collectors.
See special note on rigid parts.

Internal LS-DYNA keyword	Nastran written output	Note
*BOUNDARY_SPC_NODE	SPC	[2]
*BOUNDARY_SPC_SET	SPC1	[3]
*CONSTRAINED_INTERPOLATION	RBE3	
*CONSTRAINED_NODAL_ RIGID_BODY	RBE2	
*CONSTRAINED_SPOTWELD	RBE2 or Solid welds	[10]
*DEFINE_CORDINATE_NODES *DEFINE_COORDINATE_xxx (except _NODES)	CORD1R CORD2R	
*DEFINE_SD_ORIENT	CORD2R	[1]
*ELEMENT_BEAM	CBAR	[8]
*ELEMENT_DISCRETE	CELAS1, CDAMP1	[9]

*ELEMENT_MASS *ELEMENT_INERTIA	CONM2	
*ELEMENT_PLOTEL	PLOTEL	
*ELEMENT_SHELL	CQUAD4, CQUAD8, CTRIA3, CTRIA6	[7]
*ELEMENT_SOLID	CHEXA, CTETRA, CPENTA	
*INCLUDE	INCLUDE	
*LOAD_NODE	FORCE	
*LOAD_SHELL	PLOAD	
*LOAD_THERMAL_CONSTANT_NODE	TEMP	
*MAT (all structural exc. discrete, composite)	MAT1	
*MAT_ENHANCED_COMPOSITE_DAMAGE	MAT8	
*NODE	GRID	
*PART_COMPOSITE *MAT_54 (with *INTEGRATION)	PCOMP	
*SECTION_BEAM (exc. type 6)	PBAR, PBARL	[5]
*SECTION_DISCRETE	PELAS, PDAMP	[6]
*SECTION_SHELL	PSHELL	[4]
*SECTION_SOLID	PSOLID	
*TITLE	TITLE =	

Notes:

1. To avoid label clashes these will be labelled at the *DEFINE_SD_ORIENT label offset with the highest *DEFINE_COORDINATE label.
2. Local coordinate systems on SPCs are maintained.
3. *BOUNDARY_SPC_SET cards defined in a local coordinate system are automatically written out as SPC cards instead of SPC1 cards.
4. Thickness on the PSHELL card will be taken as the T1 value on the *SECTION_SHELL card. Shells of variant thickness must be set up on the *ELEMENT_SHELL card.
5. For DYNA integrated beams the values of A,I1,I2,J and shear factor (if SHRF=0) are calculated from the section defined.
6. Values of k and dc are read from *MAT_SPRING_ELASTIC and *MAT_DAMPER_ VISCOUS. For other discrete materials the illegal field ’xxxxxxx’ will be written to the Nastran file.
7. For shells _THICKNESS and _BETA attributes are translated directly.
8. DYNA beam release codes, orientation vectors and global offset vectors are translated.
9. For a discrete element or set of coincident discrete elements, new RBE2s are created at each end in order to convert the discretes to zero length (if necessary) and to generate nodes to which the CORD2Rs may be referenced. All nastran elements will reference component number 1 (translational) or 4 (rotational). If no *DEFINE_SD_ORIENT vector exists for a non-zero length discrete, the CORD2R will be created from the two nodes.
10. *CONSTRAINED_SPOTWELD cards are translated to NASTRAN RBE2 cards if option "Convert spotweld beams to hexa" is not set prior to the write operation. Also see section Conversion of spotweld beams below.
11. In PRIMER release 9.3 the option of "SMALL" and "WIDE" format output is provided. Note that all GRID (node) cards are written out in the "WIDE" format even if the user opts to write the deck out in the "SMALL" format. This is done in order to preserve the precision of nodal coordinates. When the "WIDE" option is chosen by the user, only those cards which contain floating point fields are written out in the "WIDE" format. Cards containing only integer data are written out in the "SMALL" format always.

Special note on rigid parts:
Rigid parts will be represented by the generation of an RBE2 over the nodes of the part itself, any parts slaved to it and any *CONSTRAINED_EXTRA_NODES.

An additional node, created at the centre of mass, serves as the independent node and will carry an SPC if there is any

DYNA material constraint (CMO.ne.0).
If the part is of type *PART_INERTIA a CONM2 element will be added at the defined centre of mass and element printing will be suppressed.
Local coordinate systems for the PART_INERTIA (IRCS=1) and for DYNA material constraint(CMO=-1) are treated appropriately.

Conversion of spotweld beams:
If the option **Convert spotweld beams to hexa** is activel, existing LS-Dyna beam type spotwelds will be written out as Nastran solid weld elements. Note that the beams need to be created and correctly projected in the Dyna model before exporting the Nastran file.

The area of the NASTRAN spotweld is the same as the area of the DYNA beam.

Each LS-Dyna beam will be converted to a Nastran CHEXA solid element of equivalent cross-sectional area, the nodes of which are tied using RBE3 elements to the appropriate shells. If a solid node lies directly over a shell node an RBE2 is used instead as Nastran does not permit RBE3s with zero weighting. Solid elements which are too close to free edges or facets which exceed 30 degrees will be moved inboard to enable the nodes to be tied on.

Typical shell edge length value is set to 10.0 which is appropriate for typical crash models with units in mm. If Primer reports that it **"***Could not generate solid elements for n spotweld beams***"** this value may be increased to capture more of the welds.

If any spotwelds fail to convert, a set of failed beams is created.

Conversion of tied contacts:
If the option **Convert tied contact to RBE3** is active, CONTACT_TIED_NODES_TO_SURFACE in LS-Dyna will be translated for Nastran output.

The conversion process assumes that a set of slave nodes (defined by node set/part/part-set) is to be tied to a set of master shells (defined by part/part set/shell set).

The Primer contact checker will be used to determine which nodes actually tie in the LS-Dyna model. **Only these nodes** will be tied in the Nastran output.

In default mode, each slave node (this is typically a node on a solid) will be tied as a dependent using an RBE3 element to the nodes of the corresponding master shell. However, Nastran does not appear to permit zero weighting in RBE3, so if the nodes coincide (or project directly onto one another) an RBE2 will be made.

Use RBE2 to project. This option may be used if the slave nodes are excessively distant from their master shells. It will generate an RBE2 on the slave node which projects to the master shell.

On completion of the contact a node set will be created called "*Slave nodes tied. Contact n*".

APPENDIX VIII: "Curve" file formats

(1) T/HIS curve file format.

This section describes the format of a T/HIS "curve" file.

Line 1	Title
Line 2	X axis label
Line 3	Y axis label
Line 4	Curve label
Line 5	X, Y point 1
Line 6	X, Y point 2
...	
Line n+4	X, Y point n

The X and Y values can be in any format as long as the two values are separated by a space or a comma.

A comment line may be included anywhere in the file by starting the line with a '$'.

Several curves can be put in one file sequentially, separated by the word CONTINUE.

The title and three label lines must be present for each curve.

(2) "raw" x,y file format.

This section describes the format of a "raw" x,y data file.

A raw x,y data file contains lines which have a pair of x,y data points on them.

Line 1X, Y point 1
Line 2X, Y point 2
...
Line nX, Y point n

The X and Y values can be in any format as long as the two values are separated by a space, a comma or a tab.

A comment line may be included anywhere in the file by starting the line with a '$'.

APPENDIX IX: Primer database format

Database files in Primer are a powerful way of organising and retrieving data. For databases to function correctly in Primer two types of files are required.

‘database’ files

‘index’ files

database files

Primer looks for databases by looking for the existence of a file called ‘oa_database’ in the directories $OASYS (the directory where the Oasys Ltd. LS-DYNA environment is installed) and $HOME (the home directory of the user. (‘.database’ can also be used for backwards compatibility with versions of Primer previous to version 8.1). In the instance where a ‘.database’ file is found in both these directories then the entries from both files are included.

The databases in the file from the directory $OASYS are available to every user. If the file was in $HOME then the databases would only be available to that user.

Any line in the file which begins with a $ (dollar) is treated as a comment.

Each line in a ‘database’ file refers to a database and contains 3 fields separated by a space, a comma or a tab character.:-

1. **Database type**. This indicates what type of entities are in this database. **At present the only 2 types are available; ‘LCUR*’ and ‘MATL*’**. The database type must end with an asterisk (*), for example LCUR*. Databases are given a type so that only databases which are relevant are displayed when editing a particular entity.
2. **A directory**. The directory points to where the actual database information is stored. This directory must contain an index file which explains how the database is formatted.
3. **A name**. The database names will be used in windows when selecting a database.

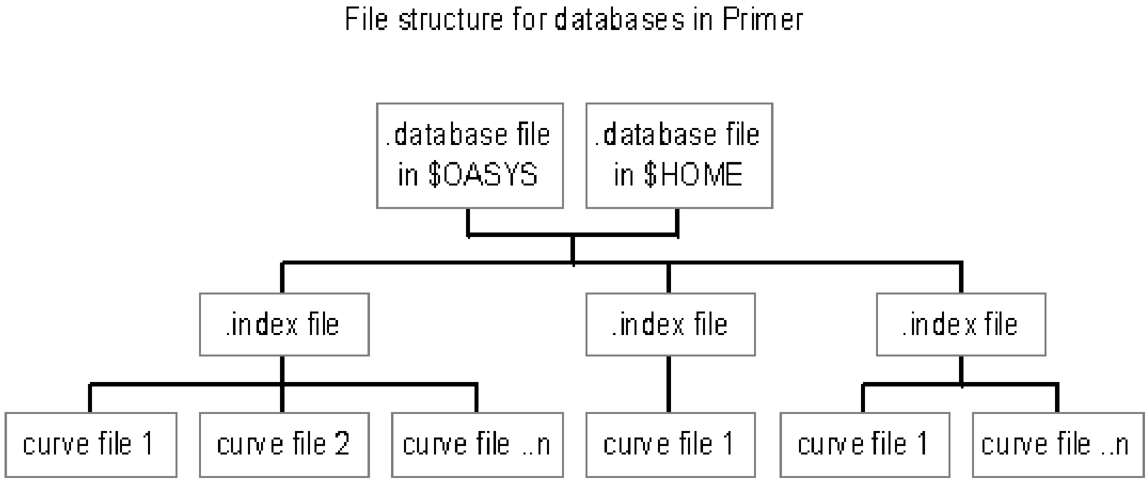
An example of a ‘database’ file taken from an $OASYS directory is given below containing 2 LCUR and 2 MATL databases.

```
$ database file for Primer in $OASYS
$
$ Any databases which are defined in here will be available to all users
$=======================================================================
$
$ type directory to find .index file in     database name
$ ==== ================================     =============
LCUR*  /disk/database/loadcurve/seismic     Seismic loadcurve database
LCUR*  /disk/database/loadcurve/material    Material loadcurve database
MATL*  /disk/database/material/steel        Steel Database
MATL*  /disk/database/matreial/aluminium    Aluminimum Database
```

index files

A index file contains information on how a database is formatted. One index file called ‘oa_index’ is needed for each database entry in the ‘database’ file and it must be present in the directory specified for each database. (*‘.index’ can also be used for backwards compatibility with versions of Primer previous to version 8.1*).

The .index file then describes this database and gives the filenames of all the entries in this database. For example the LCUR* database ‘Seismic loadcurve database’ in the above database file points to the directory /disk/database/loadcurve/seismic. An oa_index file must be present in this directory to describe how the database ‘Seismic loadcurve database’ is formatted and how many entries there are. This is summarised in figure A9.1.



Any line in the file which begins with a $ (dollar) is treated as a comment. The first uncommented line in an oa_.index file **MUST** contain how many fields there are for each database. The next lines in the file then give the headings for each field.

There are some limitations on the fields in an oa_.index file.
1. There is a maximum limit of 20 fields.
2. **The first field for each entry must be the filename which contains the data**. (*The filename given is automatically appended to the full path of the directory the index file is located in.)*
3. Each entry in the .index file is limited to 40 characters in length

An example of a oa_index file is given below.
```
$ Example oa_index file
$
$ NUMBER OF COLUMNS IN DATABASE
$
  3
$
$==================================================================
$ THE COLUMN NAMES *** THE FIRST COLUMN MUST BE THE FILENAME ***
Filename
Units
Curve Description
$==================================================================
$
$ THE DATABASE ENTRIES
$
kobe_x-accel.cur
m/s2
X Acceleration - Kobe
$
kobe_y-accel.cur
m/s2
Y Acceleration - Kobe
$
kobe_z-accel.cur
m/s2
Z Acceleration - Kobe
```

In this example above there are 3 fields called
- filename
- units
- curve description

MATL* - Material Databases

There is a maximum limit of 256 Material Databases. The entries in a oa_index file for a MATeriaL database **MUST** be formatted using 3 fields as follows.
- filename
- material name (this will be used to automatically match the material in a model with a material in the database

after it has been converted to upper case)
- LS-DYNA material model type (without the ***MAT_** prefix)

i.e

```
3
$
Filename
Material Name
LS_DYNA Material Type
$
steel_yield-200.key
   STEEL : YIELD 200 MPa
   PIECEWISE_LINEAR_PLASTICITY
$
steel_yield-250.key
   STEEL : YIELD 250 MPa
   PIECEWISE_LINEAR_PLASTICITY
```

Each data file should contain a single LS-DYNA material input card (in the KEYWORD format) along with any ***DEFINE_CURVE** and ***DEFINE_TABLE** definitions that are referenced by the material card. The material should be defined in the database file as material number 1 and any ***DEFINE_CURVE** and ***DEFINE_TABLE** cards should also be numbered sequentially from 1.

When a material is imported from a database file the following rules are used :
- If the file contains more than 1 material then only the material with the lowest material ID will be imported from the file.
- All ***DEFINE_CURVE** and ***DEFINE_TABLE** definitions will be imported .
- Any other LS-DYNA keyword data cards will be ignored.

APPENDIX X: Headform "tree" file example

```
*HEADFORM_START
$
$ Headform is at: 0.0 horizontal and 0.0 vertical
$ Reference point is at: 0.0 x 0.0 y 0.0 z
$
$  <label><title>
         1FTA FREE MOTION HEADFORM
$
*REF_POINT
$   <node>
         1
$
*UNITS
$<mass>   <length>  <time>
TE        MM        S
$
*AXES
$<coordID>
         1
$
*COMPONENTS
$ part_set cont_part   cont_no
         1         1         1
$
*TARGET
$<current>
$ No target point defined
$
*HEADFORM_END
```

APPENDIX XI: Target and Position "tree" file example

```
*TARGET_POINT_START
$  <label><posn>    <title>
         1AP2       Example target point
$
$ Target point coordinates
$       <x>       <y>       <z>
     500.0     500.0       0.0
$
$ Horizontal angles
$     <min>     <max>
      25.0      90.0
$
$ Velocity, contact part set and current position
$
$     <Vel><Cont set><head_pos>
   5364.48         2         1
$
*HEAD_POSITION
$  <label><name>
         1max h= 90.0
$
$ H-point position, headform angles and relation to target point
$       <x>       <y>       <z>   <horiz>    <vert>  <maxmin>
     500.0     500.0       0.0      90.0       0.0       max
$
*TARGET_POINT_END
```

APPENDIX XII: Dialogue (typed in) Command Syntax

Primer has a limited "dialogue" command set that can be used in any of three ways:

1. In graphical (screen menu) mode commands can be typed into the "Dialogue Box" at any time.
2. In non-graphical (text only) mode commands are typed in at the terminal prompt.
3. In command files, run either interactively or in batch, commands are executed as if typed in.

In all cases the command ***input*** syntax is identical, although there are minor differences in output between "screen menu" and "text-only" modes: in the latter case all output has to go to the controlling terminal ("stdout"), whereas in the former separate windows are used for "help", "listing" and other output.

The Dialogue command structure

The command structure forms a hierarchical "tree", with the top-level **PRIMER_MANAGER** at its "root".

The following rules apply:

- Command words may be abbreviated to any degree so long as:
  - they are unique in the context of their current menu
  - they must have at least their first two characters given

  For example **`READ DK_DYNA_KEYWORD`** may be abbreviated to **`RE DK`**.
- Navigation up and down menu levels is performed as follows:
  - <command> takes you to that command's (sub-)menu level
  - Forward slash "/" takes you back to the top **PRIMER_MANAGER** level before executing the following command(s)

  For example **`READ`** above takes you down into the **READ_MODEL** sub-menu
  The command **`/WRITE DK`** would work at the **READ_MODEL** prompt because it would return to the top level before parsing the **`WRITE`** command
- There is also a "global menu" of commands which is available at any (sub-)menu prompt.
  - These are primarily graphics commands that do not require a context.
  - The commands can be listed with the **`GM`** (for Global Menu) command
- Any command can be aborted by typing **`Q`**(uit). This will return control to the next highest command prompt in the "tree".
- At any prompt you can type **`H`**(elp) to receive advice about what to do next.

Commands that perform model-based operations:

READ

Reads in models in the following range of formats:

DK_DYNA3D_KEYWORD		Reads LS-DYNA keyword input decks
BDF_NASTRAN_BULK_DATA	TRANS	Nastran "Bulk data" (.bdf) file
UNV_UNIVERSAL_FILE	TRANS	Ideas/Master Series "universal" (.unv) files
RADF_RADIOSS_FIXED_FORMAT	TRANS	Radioss fixed format "starter" (00) files
RADB_RADIOSS_BLOCK_FORMAT	TRANS	Radioss block format "starter" (00) files
PNF_PATRAN_NEUTRAL_FILE	TRANS	Patran 2.5 "neutral" (.ntl) file
SAP_2000	TRANS	SAP 2000 input (.s2k) files
ABAQUS	TRANS	Abaqus keyword input decks

In all cases the syntax is **READ <format> <filename> <target model id>**
for example: **READ DK test.key 2**

All the "non-DYNA3D" options imply a degree of translation, click on the TRANS buttons in the table above to view the relevant sections in Appendix VI which describe translation during input.

WRITE

Writes out models in the following range of formats:

DK_DYNA3D_KEYWORD DB_DYNA3D_KEYWORD DM_DYNA_MERGE_INCLUDES DO_DYNA_OMIT_INCLUDES		All four options write out LS-DYNA keyword files: • DK writes separate include files • DB writes out the master file only • DM merges include data into a single master file • DO omits data in include file
BDF_NASTRAN_BULK_DATA	TRANS	Writes out the model in Nastran Bulk Data File (BDF) format.
I5_IDEAS_LEVEL_5 **I6_IDEAS_LEVEL_6** **MS_IDEAS_MASTER_SERIES**	TRANS	All three options write Ideas or Master Series universal files, but the modules used reflect the version chosen.
P25_PATRAN_2.5_NEUTRAL **P3_PATRAN_3_NEUTRAL**	TRANS	Some of the <config> arguments change between Patran levels 2.5 and 3 neutral files.
ZTF_FILE		Writes the <name>.ztf file for post-processing in D3PLOT
GROUP_FILE		Writes <namennn>.bin file(s) to make *GROUP data available for post-processing in D3PLOT
PARTS_GROUP		Expanded ascii parts group file
SF_SUMMARY_FILE		Writes a summary file giving the main parameters of the model

In all cases the syntax is **WRITE<format> <filename> <target model id>**
for example: **WRITE DK test.key 2**

The Ideas/Master Series, Patran and Nastran output formats require some translation, and sometimes some adjustments to element labels. Click on the TRANS buttons in the table above to view the relevant sections in Appendix VII which describe translation during output.

COPY

The syntax is **EXECUTE <list> <target>**

This copies the model(s) in **<list>** into the **<target>** model, which must not already exist.

DELETE

The syntax is **EXECUTE <list>**

This deletes the model(s) in **<list>**.

ORIENT

Applies orientation to models, or subsets of them.

The syntax is: **<orientation type> <object type> <list of objects> <orientation parameters>**

Where:

<orientation type>	Is one of **TRANSLATE, ROTATE, REFLECT, SCALE**
<object type>	Is one of: • **MODEL** • **PART** • **<element type> (e.g. SOLID, SHELL,...)** • **NODE**
<list of objects>	Is a valid **<list>** of the relevant object type
<orientation parameters>	Depend on the <orientation type>: **TRANSLATE** **[dX, dY, dZ]** translation distances **ROTATE** **[rX, rY, rZ]** rotation angles in degrees **[cX, cY, cZ]** centre of rotation **REFLECT** **X** or **Y** or **Z** reflection axis **<Distance along axis>** **SCALE** **[sX, sY, sZ]** scales in each of X,Y,Z **[cx, cY, cZ]** centre of scaling

These commands should not all be given on one line! Give each section separately to avoid confusion, for example:

```
PRIMER_MANAGER >>> ORIENT TRANSLATE
ORIENT WHAT? MODEL
Give MODEL <list>: 1 2 3
Give X,Y,Z translations: 1.0 2.0 0.0
```

DT_DATA_TRANSFER

Transfers selected data from a "source" model to a "target" one, according to a range of parameters.

The syntax is:

SOURCE <model id>	Required. Defines the source model <model id> from which data are extracted.
TARGET <model id>	Required. Defines the target model <model_id> into which data will be transferred.
DATA_TYPE <type(s)>	Required, no default. Defines the type(s) of data which will be transferred. Valid <type>s are:

MAT	Structural materials
EOS	Equations of state
SECTION	Element section data
HOURGLASS	Hourglass definitions
TMAT	Thermal materials

Selections are additive. Thus you could give the commands:

DATA_TYPE MAT SECTION TMAT

To transfer those three types of data.

You may also give the commands:

NOT <type>	(e.g.**NOT EOS**) to remove a selection from the current list
CLEAR	to clear the current selection
STATUS	to show what is currently selected.
DONE	to return to **DT_DATA_TRANSFER** prompt.

MATCH_BY <method>	Optional, defaults to "match by **NAME**". Defines how objects in the source file are matched for data transfer. Valid <methods> are:

ID	If labels match
NAME	If **<source>** name (i.e. **_TITLE**) is equal to or a subset of **<target>** name.
BOTH	First by **ID**, then by **NAME**.
ALL	All the data in the <source> is transferred to the <target>
STATUS	to show what is currently selected.
DONE	to return to **DT_DATA_TRANSFER** prompt.

ACTION <action>

Optional, defaults to action **CS_COPY_TO_SEPARATE**.

Defines how data transferred from source to target models is stored in the target model. Valid actions are:

CS_COPY_TO_SEPARATE	This causes transferred data to be stored in a separate include file in the target model. This include file will be given the name "dt_transfer_from_<*target*>.key" This is the default behaviour, and is recommended since it will make it easy to identify data that has been transferred.
CO_COPY_TO_ORIGINAL	The include file of the destination in the target model is unchanged by the transfer operation. For example if a material in include aaa.key is overwritten it remains in that file.
CM_COPY_TO_MASTER	Any data transferred into the target file is made to exist in the master file.
RO_READ_ONLY	This is rather different to the options above: • It is assumed that the <source> file will be included verbatim via an *INCLUDE keyword at run-time; • It is also assumed that the contents of this file will not be changed. Therefore the following actions are taken: • Any data transferred into the <target> file is placed in a special "Include" file which is marked "read only". • When the <target> model is finally written out this file is not included, rather the original <source> file is referred to via an *INCLUDE statement, using its full original pathname.
STATUS	Shows the current status of **ACTION**.
DONE	Returns to **DT_DATA_TRANSFER** prompt.

NAME_MATCH <method>

Optional, defaults to **EITHER**.

Defines how names in source and target models are matched when the **MATCH_BY** method is **NAME**.

T_IN_S	**T**arget ***I***n ***S***ource. A match is made if the target's title string is equal to or a subset of (<=) the source title.
S_IN_T	**S**ource **I**n ***T***arget. A match is made if the source's title string is equal to or a subset of (<=) the target title.
EITHER	A match is made if either **S_IN_T** or **T_IN_S** is true.
EXACT	A match is made only if the target and source titles are identical.
STATUS	Shows the current status of **NAME_MATCH**
DONE	Returns to **DT_DATA_TRANSFER** prompt.

Note that name matching is also subject to the following rules:

- Matching is case ***IN***sensitive. So **ABC = abc = ABC = ABC**
- Leading and trailing spaces are ignored. (But intermediate spaces between words are significant,

  thus

  " **ABCDEF** " matches "**ABCDEF**"
  "**ABC DEF**" does not match "**ABCDEF**"

SUPERSEDED

Optional: defaults to **SAVE**.

This controls how objects that are overwritten ("superseded") in the target model are handled.

SAVE	Relabels the "old" objects and transfers them to include file "dt_renumbered.key". They will not be referenced by anything and can be deleted by a **CLEANUP_UNUSED** operation.
DELETE	The incoming definition supersedes the object in the target model, and the original definition is lost.
STATUS	Shows the current status of **SUPERSEDED**
DONE	Returns to **DT_DATA_TRANSFER** prompt.

FEEDBACK Optional: defaults to PART_STATUS in non-graphical mode.

"Feedback" controls how much information about what happened during a data transfer operation is given to the user. Any permutation of the options below can be selected.

SKETCH	Sketches all updated objects on the current plot. In non-graphical mode this option is ignored.
PART_STATUS	Creates two tables: the first lists all parts that were affected by this data transfer operation, the second lists those that were not.
NOT <option>	Unsets <option>. For example **NOT SKETCH** means that sketching will not take place.
CLEAR	Clears all options leaving none selected.
STATUS	Shows the current status of **FEEDBACK**
DONE	Returns to **DT_DATA_TRANSFER** prompt.

STATUS Lists the status of all subcommands and settings in the **DT_DATA_TRANSFER** menu.

APPLY Checks the input parameters, and carries out the actual transfer operation.

Several of these settings can also be preset in the "oa_pref" file. See the data transfer section.

BOM (Bill of Materials)

Allows reading or writing of a Bill of Materials file. There are 5 options available:

READ_BOM	Read a bill of materials file. The columns in the file will be auto-detected from the headers and the file read.
NODE/MA/EL	Node/Mass/element renumbering activated
CREATE	Create mode activated
DEFAULT	Default (no create) mode without renumbering
WRITE_BOM	Write a bill of materials file

For all options the syntax is **<filename> <model id>.** Where:

<filename>	Is the filename of the Bill of materials to read/write
<model id>	Is the number of the model to read the BOM into/write the BOM from.

PART_INFO

Allows writing of a part information .csv file for all parts in a model. There are 2 options available:

WRITE	Write information in full with mass properties
NOMASS_PROPS	Write information excluding mass properties

For all options the syntax is **<filename> <model id>.** Where:

<filename>	Is the filename of the part information file to write
<model id>	Is the number of the model to write the part information file from.

ASSIGN_MASS

The syntax is **APPLY_ALL <model id>**

This applies all the assign mass definitions in model number **<model id>**.

MD_MATERIAL_DATABASE

Performs a material database "import" operation.

The syntax is **<model id> <database id>.** Where:

<model id>	Is the number of the model in which material properties will be updated.
<database id>	Is the <i_th> *MATL database in the predefined database file.

Once these two parameters have been defined a "MATCH_ALL" operation is carried out.

SPOTWELD Performs spotweld functions. A number of options are available for this menu.

READ Read spotweld menu options, four file types are supported:

SPOTWELD	Read Primer spotweld file
CATIA	Read Catia spotweld file
UG	Read a UG weld file
CONNECTION	Read Primer XML connection file

The following syntax is **<filename> <model number> <part for welds>**. Where:

FILENAME	Connection file name
MODEL NUMBER	Model number to read connections into
PART FOR WELDS	Part for the spotweld beams/solids to go into. This must be a valid spotweld part: *SECTION_BEAM with ELFORM=9 (for beams) and material type *MAT_SPOTWELD

After the spotweld file is read, PRIMER tries to make the connections. Any bad welds are reported to the screen and written to a PRIMER spotweld file called **<inout weld file name>_bad**.
For example: 5 bad welds from **example.weld** would be written to the file **example.weld_bad**.

An example input line is: **/SPOTWELD READ CATIA example.weld 1 40000**
This would read the Catia spotweld file **example.weld** into model 1, part 40000.

WRITE Write spotweld menu options, four file types are supported:

SPOTWELD	Write Primer spotweld file
UG	Write a UG weld file
CONNECTION	Write Primer XML connection file
USER	Write a user defined connection file

The following syntax is **<filename> <model number>** . Where:

FILENAME	Connection file name
MODEL NUMBER	Model number to write connections out of

TYPE Here you can set the type of spotweld that will be created when reading in a connections file. Note this does not affect the connections read in from a Primer XML connections file, as this will contain the spotweld type already. Available types are:

BEAM	Create spotweld beams
HEXA	Create a single spotweld solid
4_HEXA	Create a solid spotweld nugget with 4 solids
8_HEXA	Create a solid spotweld nugget with 8 solids
12_HEXA	Create a solid spotweld nugget with 12 solids
16_HEXA	Create a solid spotweld nugget with 16 solids

DIAMETER Here you can set the diameter of the spotweld that will be created when reading in a connections file. Note this does not affect the connections read in from a Primer XML connections file, as this will contain the spotweld diameter already.

DUMMY and **MECHANISM**

ASSEMBLY	Select an assembly by name or number, then perform one of the following operations upon it:	**FIX** *dof code* — Restrain the assembly in degrees of freedom *dof code* **TRANSLATE** *dx, dy, dz* — Translate assembly ***by*** amount *dx,dy,dz* **RX** or **RY** or **RZ** *theta* — Rotate assembly ***to*** angle *theta* degrees about x/y/z **RESET** — Undo all dummy transformations and return to initial state **DONE** — Finish with assembly and return to **DUMMY >** prompt
POINT	Select a point by name or number, then perform one of the following operations upon it: (Note: moving the point implicitly moves its "owner" assembly.)	**FIX** *dof code* — Restrain the point in degrees of freedom *dof code* **TRANSLATE** *dx, dy, dz* — Translate point assembly ***by*** amount *dx,dy,dz* **POSITION** *x, y, z* — Translate point assembly ***to*** coord *x, y, z* **RESET** — Undo all dummy transformations and return to initial state **DONE** — Finish with point and return to **DUMMY >** prompt
CONNECTION	Select a connection by name or number	**SLIDE** *distance* — Applies to **LINE** connections only, and will slide the joint by *distance* down its AB axis. **ANGLE** *theta* — Applies to **LINE** and **HINGE** connections only, and rotations the assemblies to achieve angle *theta* (in degrees) about the AB axis.
POSITION	Specify a position *name*	Retrieves and applies the stored position *name*
H_POINT	Specify coordinate *x, y, z*	Will move the Dummy H-Point ***to*** coord *x, y, z*
READ_CONFIG	Specify a *filename*	Retrieves a free-standing dummy configuration file (the keywords and data between ***DUMMY_START** and ***DUMMY_END**). *Filename* will usually have the extension .dcf
READ_DUMMY_ANGLE	Specify a *filename*	Retrieves and applies the overall orientation, H-Point and joint angles stored in a Dummy Angles File (usually extension .daf).

`ACCEPT`	Accept the current dummy position, save its updated geometry and return to the main [Primer >] prompt.
`RESET`	Undo all transformations and restore the initial geometry of the dummy, remaining at this prompt level.
`QUIT`	Undo all transformations and restore the initial geometry of the dummy, then return to the main [Primer >] prompt.

Meanings of terms in the table above	
dof code	Is a numeric Degree of Freedom code made up of any permutation of 123456, where 1 = Tx, 2 = Ty, 3 = Tz, 4 = Rx, 5 = Ry, 6 = Rz For example code **`136`** means restraint in Tx, Tz, Rz Code **`0`** may also be used, meaning "free all restraints"
dx, dy, dz	Is a translation vector, ie a relative movement from the current position, made up of three numbers. For example **`10.0 20.0 30.0`** means translate 10.0 in X, 20.0 in Y, 30.0 in Z. "Wildcard" syntax is permitted: any number entered as an asterisk ("*"), and omitted trailing digits, are treated as "free" values. For example: **`10.0`** means translate 10.0 in X, but permit Y and Z to adopt any value. **`* * 20.0`** means translate 20.0 in Z, but permit X and Y to adopt any value
x, y, z	Is an absolute coordinate. For example **`10.0 20.0 30.0`** means coordinate X=10, Y=20, Z=30. Wildcards as for translations above are permitted
theta	Is an angle in degrees for the given degree of freedom. In a dummy model angles are absolute values expressed in the coordinate system of the connection between this assembly and its parent. In most cases this will mean the system implied by the local axes of the joint stiffness definition at the joint.

BUILD

Allows the user to access Primer's model build functionality. Available options are:

DATABASE	To select a database file for build use:
	The following syntax is **DATABASE <filename.dba>**
TEMPLATE	To select a template file for build use:
	The syntax is **TEMPLATE <filename.tpl>**
SIMPLE **RIGOROUS** **MASTER**	You apply the build process by selecting the build mode: **SIMPLE** will build without checks or renumbering **RIGOROUS** will build with checks and renumbering **MASTER** <a.key> will just write out a masterfile to a.key (no orient)

An example of the syntax is:

BU DATAB all_models.dba TEMPL odb_40.tpl SIMP

Alternatively you can set up multiple models using a build file:

READ	To set up multiple models using a build file:
	The syntax is **READ <build.csv>**

Before READ you may set the following:

KEEP	Keep each Primer model after keyout
DEL	Delete Primer model after keyout (default)
ECHO	Echo each line from input file
NOECHO	Do not echo (default)

To sketch target points either before or after building:

SKETCH	Sketch target points in file
	The syntax is **SKETCH <build.csv>**

An example of the syntax is:

BU KEEP RE headpos.csv
The build mode is **SIMPLE** in this case

PREFERENCE

Read the preference file. Available options are the location of the preference file:

SYSTEM	look for the preference file in the $OASYS directory
HOME	look for the preference file in the home directory
PATH	use file path as supplied

After the option is chosen, you are prompted for the filename. An example of the syntax is:

HOME oa_pref

CHECK

This function will check the model specified. Syntax is:

MODEL n CHECKFILE filename APPLY

The name of an output file is specified with the CHECKFILE option. For example, to run an error check on model 2:

```
MODEL 2
CHECKFILE nyname.txt
APPLY
```

Omitting the checkfile option will give the default filename derived from the root of the dyna key file - <file>.check:

MODEL 2 APPLY

The **MODEL** option may be omitted if only one model in memory.

Before applying you may set the format option to **FULL_LIST** or **SHORT_LIST** (default). Just type **FULL_LIST** or **SHORT_LIST** to activate the option.

FULL_LIST will ensure that label lists are written for all item types. By default these are not written for the potentially highly populous types (NODES, ELEMENTS etc.).

AUTOFIX

This function will auto fix all errors/warning in the model. Syntax is:

MODEL n APPLY

The **MODEL** option may be omitted if only one model in memory.

CONNECTION

The option allows the user to remake connection in the model. Available options are:

ALL_WELDS	remake all welds
ALL_GLUE	remake all adhesive
ALL_BOLTS	remake all rigid bolts
UNMADE_WELDS	remake unmade welds
UNMADE_GLUE	remake unmade adhesive
UNMADE_BOLTS	remake unmade rigid bolts

You can also modifiy the pitch of spotwelds by selecting the following option:

MODIFY_PITCH Change the pitch of spotwelds.

Required inputs for this operation are:

SELECT_SPOTWELDS	Select the spotwelds you wish to modify
NEW_PITCH	The new pitch value you wish to apply

After choosing **SELECT_SPOTWELDS** above, the user is prompted to select the type of entity through one of a number of different methods:

ALL	Select all spotwelds in the model
BY_LAYER_PART	Select the parts that have a layer that contains the selected parts
ATTACHED_TO_PANEL	Select spotwelds that are attached to the selected parts
BY_SINGLE_SEAM	Select spotwelds that are attached to ALL the selected parts
BY_MULTIPLE_SEAM	Select spotwelds that are attached to ANY combination of the selected parts
QUIT	Quit the selection process

When selecting parts for some of the above, you first define the type then the labels of items of the type. For example, to select a part range type **PART 10 to 19** or to select a part set type **PART_SET 5**.

Optional things to modify are:

BREAK_ANGLE	Used to determine how the lines of spotwelds are split (default 45.0 degrees)
MAX_PITCH	Maximum distance allowed between two spotwelds so that they can be considered on the same line (default 80.0mm)
SIMILAR_PITCH_ON	Group connections by similar pitch
SIMILAR_PITCH_OFF	Do not group connections by similar pitch (default)

See section 6.10.12 for more information on these inputs.

Finally:

APPLY	Apply the new pitch to the selected spotwelds.
DONE	Quit without applying the new pitch

RENUMBER

Allows you to renumber entities within the model. Available options are

MODEL	Select the model you want to operate on, for example MODEL 2
SELECT/DESELECT	Select or deselect items to renumber. After choosing select or deselect, you then must select the type of entity, and finally the labels of the items you wish to renumber. The type is one of: • **MODEL** • **PART** • **<element type> (eg SOLID, SHELL,...)** • **NODE etc.**
APPLY	Choose a start point and apply the renumbering

Syntax is **MODEL n SELECT type**, followed by **APPLY.**

For example:

```
model 2
sel part
10 to 19
apply
```

MODEL option may be omitted if only one model in memory.

SCRIPT

Reads a Primer javascript. Syntax is **READ <filename>**.

BELT

Refit an existing seatbelt to a dummy. Primer will select a seatbelt definition in the model automatically. Available options are:

SELECT	Select a different belt definition for refitting
REFIT	Refits the current belt definition to its dummy
PR_BASIC	Retrieve and update basic belt dimensions and form-finding parameters
PR_REFIT	Retrieve and update belt refit parameters
DONE	To return to the main menu prompt

SEATSQUASH Impliment auto seatsquash function on a model. Available options are:

PRIMER The simple Primer method of seatsquash.

Required inputs for this method are:

SEAT_FOAM	Select the seat foam
SEAT_TOP	Select the top of the seat
SEAT_BOTTOM	Select the bottom of the seat
DUMMY	Select the dummy
CONTACT	Select the contact between the dummy and the seat
X	X displacement per iteration
Y	Y displacement per iteration
Z	Z displacement per iteration

After choosing **SEAT_FOAM**, **SEAT_TOP**, **SEAT_BOTTOM** or **DUMMY** above, the user is prompted to select the type of entity, then the labels of items of the type. For example, to select a part range type **PART 10 to 19**, o select a part set type **PART_SET 5** or to select a dummy type **DUMMY 17**.

Optional inputs for this method are:

MAX_ITER	Maximum number of iterations (default 100)
SOLID_REL	Stop if the solid element relative volume reaches this value (default 0.2)

Finally:

STATUS	Report what still has to be defined
APPLY	Initiate the seatsquash

DYNA The Dyna method of seatsquash. Sets up an LS-DYNA analysis that carries out the seatsquash operation.

Required inputs for this method are:

SEAT_ALL	Select the seat parts
SEAT_DEFORM	Select the seat parts to remain deformable
DUMMY	Select the dummy
CONTACT	Select the contact between the dummy and the seat
X	X displacement per iteration
Y	Y displacement per iteration
Z	Z displacement per iteration

After choosing **SEAT_ALL**, **SEAT_DEFORM** or **DUMMY** above, the user is prompted to select the type of entity, then the labels of items of the type. For example, to select a part range type **PART 10 to 19**, o select a part set type **PART_SET 5** or to select a dummy type **DUMMY 17**.

Optional inputs for this method are:

MAX_ITER	Maximum number of iterations (default 100)
DUMMY_POS_TIME	Time to position the dummy (default 0.075)
ANALYSIS_TIME	The termination time of the created analysis (default 0.1)
DAMPING	Global damping (default 50.0)
CONT_KEEP	Select contacts to keep in the analysis (by default all apart from the one selected above will be deleted)

Finally:

STATUS	Report what still has to be defined
APPLY	Initiate the seatsquash

IMPORT Import a Dynain file

EXPORT Export a Dynain file

DYNA_DUMMY Impliment Dyna dummy positioning method. Available options are:

POSITION Sets up a dyna analysis to position a dummy. When entering the POSITION setcion you will be asked if you wish to automatically determine assembly nodes. This is recommended if you have not already set these nodes up within PRIMER and saved the information to the dummy tree keywords in the keyword file associated with the dummy. Se section 6.12 for more information of how to do this.

Required inputs for this method are:

NECK_NODE	Node at the base of the neck (this may be set and saved to the keyword file pria to command line positioning)
HIP_NODE	Node on the left or right hip (this may be set and saved to the keyword file pria to command line positioning)
POSITION	Choose the desired position. This should be setup pria to command line positioning. See section 6.12.2 for information on saving positions.

Optional inputs/things to modify for this method are:

RIGID_ON	Select assemblies to rigidify
RIGID_OFF	Select assemblies to turn off rigidification
DEFORM_PARTS	Select parts to remain deformable in rigidified assemblies
ANALYSIS_TIME	The end time of the analysis (default 250.0)
DAMPING	Global damping (default 50.0)
DAMPING_OFF	Turn global damping off
CABLE_DAMPING	In-line damping applied to cables (default 0.5)
FORCE_RAMP	Force ramp up time in cable (default 10.0)
CABLE_FORCE	Force in cables (default 1.0)

Finally:

APPLY	Initiate setup

IMPORT This imports a dynain file produced in an LS-DYNA dummy positioning analysis.

Required inputs for this method are:

FILENAME	Select the dynain file to import
POSITION	Choose the desired position. This should be setup pria to command line positioning. See section 6.12.2 for information on saving positions.

Optional inputs for this method are:

COORDINATES	Import the deformed coordinates
SOLID_INIT	Import *INITIAL_STRESS_SOLID
SHELL_INIT	Import *INITIAL_STRESS_SHELL
BEAM_INIT	Import *INITIAL_STRESS_BEAM
DELETE_INIT	Delete existing *INITIAL_STRESS cards
ALL_OFF	Turn all options off

Finally:

APPLY	Import the data

Dialogue commands that control viewing - the "Global Menu"

The following commands can be used to control graphics and viewing. When running in "text-only" mode they are ignored and will have no effect.

<table>
<tr><td>Drawing commands:
• LINE
• HIDDEN_LINE
• SHADED</td><td>These commands generate plots of the following types:
• Line gives a wireframe plot (no hidden surface removal)
• Hidden-line shows element edges, but with hidden surfaces removed
• Shaded produces a shaded and lit plot, implicitly with hidden surfaces removed.</td></tr>
<tr><td>Commands which control the current view:
• SXY, SYZ, SZX
• ISOMETRIC
• RS <x y z>
• RM <x y z>
• ZM
• CENTRE
• MG <scale>
• AU_AUTOSCALE
• AC_AUTOSCALE_CURRENT
• ZERO_VIEW</td><td>These commands control how the image appears on the screen.
• SXY is a view on the XY plane, down Z; SYZ on YZ down X; SZX on ZX down Y
• Isometric is a view down the vector X=Y=Z (at 45 deg to each axis)
• RS <x y z> rotates the image by <x> degrees about screen X, <y> about Y etc
• RM <x y z> rotates the image by <x> degrees about model X, etc
• ZM "zooms" in to the rectangle dragged out by the cursor
• CENTRE centres the image at the cursor location
• MG <scale> magnifies the image by <scale>. Values less than 1.0 reduce its size.
• AU autoscales on the complete model, ignoring the effects of blanking etc (ie as if all unblanked)
• AC autoscales on what is currently visible, taking into account blanking etc
• ZERO_VIEW resets the viewing routines to their default state: Autoscaled view on SXY.</td></tr>
<tr><td>Other commands:
• GM_GLOBAL_MENU
• GH_GLOBAL_HELP</td><td>Commands to do with the global menu
• GM lists the commands above, with a brief description
• GH gives more general help on the "global" menu.</td></tr>
</table>

APPENDIX XIII: Summary of "oa_pref", command-line and Environment Variable settings

The data below has been described elsewhere in this manual, but is summarised below to concentrate it all in one place.

"oa_pref" files	Allow customisation of Primer via site-wide, user and directory specific settings.
Command-line arguments	May be appended to the line which executes Primer to specify operation modes, graphics drivers, etc
Environment variables	Can be used to modify behaviour and to preset many attributes.

The "oa_pref" preferences file.

This file contains code-specific preferences that can be used to modify the behaviour of Oasys Ltd products. It is optional and, where entries (or the whole file) are omitted Primer will revert to its default settings.

"oa_pref" naming convention and locations

The file is called "oa_pref", optionally with a leading "." (.oa_pref) on Unix systems.

It is looked for in the following places in the order given:

- The top level configuration directory **$OA_ADMIN_xx** (where xx = 10 for release 10, etc)
- The installation directory **`$OA_INSTALL`**
- The user's home directory: **`$HOME`** (Unix/Linux) or **`$USERPROFILE`** (Windows)
- The current working directory
- from any directory of the user's choosing - see custom oa_pref file

Files do not have to exist in any of these locations, and if none exists the programme defaults will be used.

If the same preference is set in more than one place the last occurence will dominate, thus a setting read from a pref file in the user's home directory will over-rule a setting in a pref file in the installation directory.

That said, preferences (excepting those that control the graphics) in the higher level directories may by locked by using # instead of * in the syntax. In this case the locked setting will win.

On Unix and Linux:

`$HOME` on Unix and Linux is usually the home directory specified for each user in the system password file. The shell command "**`printenv`**" (or on some systems "**`setenv`**") will show the value of this variable if set. If not set then it is defined as the "**`~`**" directory for the user. The command "**`cd; pwd`**" will show this.

On Windows:

`$USERPROFILE` on Windows is usually **`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

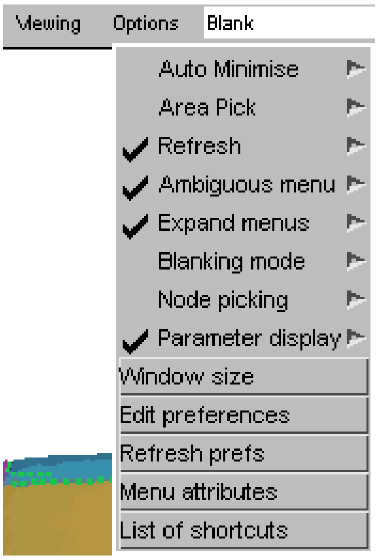
- Organisation-wide options in the version in OA_INSTALL,
- User-specific options in $HOME / $USERPROFILE
- Project-specific options in the current working directory.

Re-reading the "oa_pref" file

You no longer need to quit and restart Primer to return your settings to the state they are in on initial start up.

Primer can now restore settings to their default value and re-read the system and home oa_pref files using the feature

Refresh Preference Settings. It is available under CHECK > OPTIONS and from the main Options drop-down.



"oa_pref" file syntax

The syntax used for Primer is: **primer*<keyword>: <argument>** or for a locked pref **primer#<keyword>: <argument>**

for example:

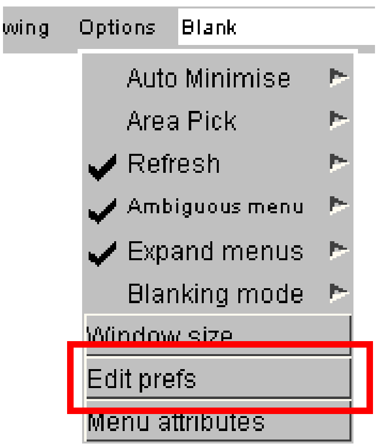
`primer*initial_plot_model: SHAD`

The rules for formatting are:

- The **`<programme>*<option>:`** string must start at column 1;
- This string must be in lower case, and must not have any spaces in it.
- The **`<argument>`** must be separated from the string by at least one space.
- Lines starting with a "**#**" are treated as comments and are ignored.

The interactive Preferences Editor

You are free to edit oa_pref files by hand, but there is an interactive "Preferences Editor" that may be called from within PRIMER that makes the job much easier.



It is started by **Options, Edit Prefs:**

The preferences editor reads an XML file that contains all possible preferences and their valid options, and allows you to change them at will. In this example the user is changing the background colour in PRIMER.

Note that changes made in the Preferences editor will not affect the current session of PRIMER, they will only take effect the next time it is run.

If you have write permission on the oa_pref file in the $OASYS directory you will be asked if you want to update that file, otherwise you will only be given the option of updating your own file in your $HOME / $USERPROFILE directory.

In this example the user is changing the background colour.

The option is "active" (ie present in the oa_pref file) and currently is set to BLACK.

Usage is:

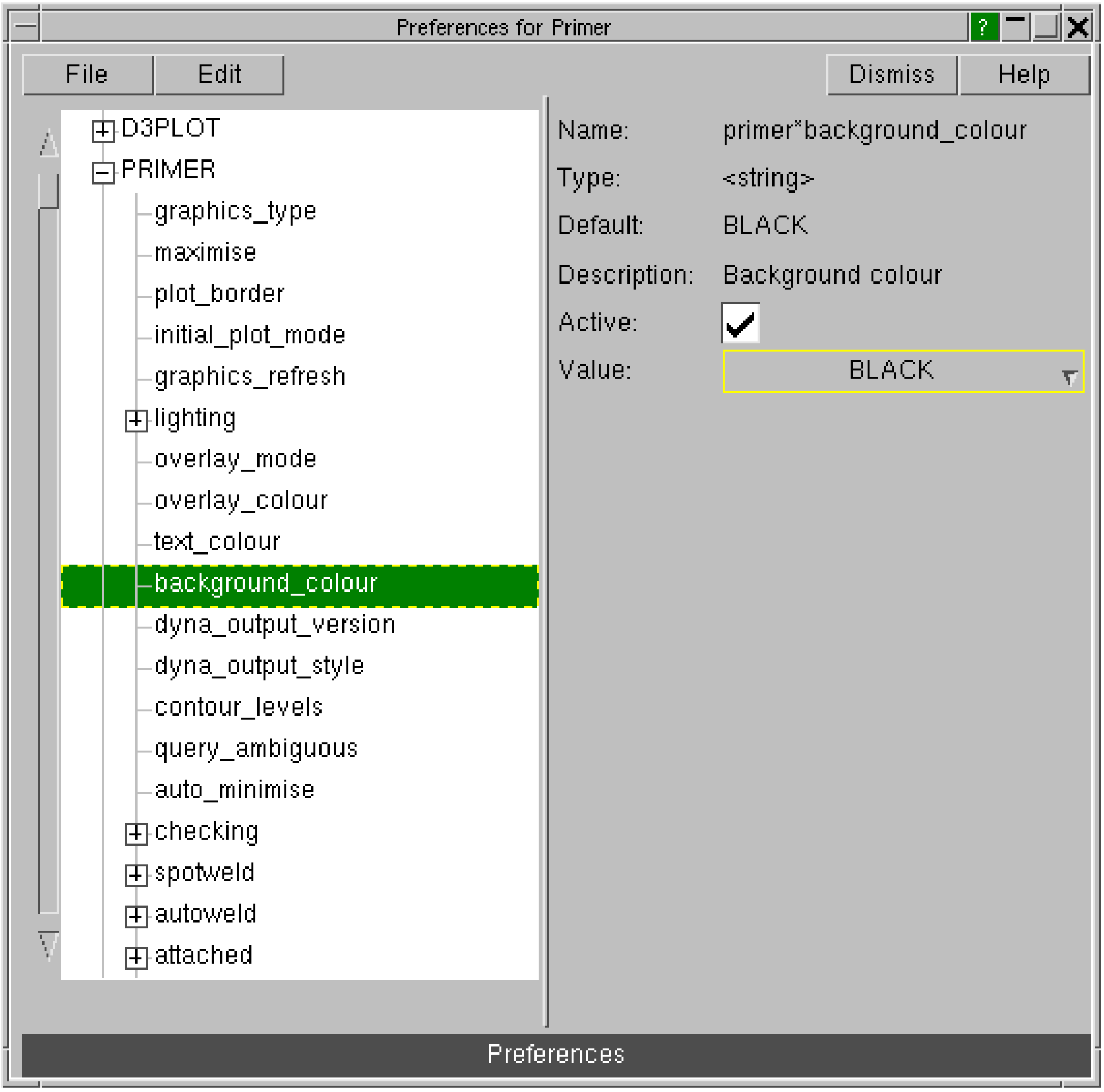
- Select an option in the Tree on the left hand side
- Make it active / inactive
- If active select a value from the popup, or type in a value if necessary

The colour of the highlighting in the left hand side tree is significant:

Green	Means that the option has been read from your $HOME file.
Red	Means that the option has been read from the $OASYS file.

In either event, regardless of the data source, the updated option will be written to the file chosen when you started the preferences editor.

Because of the order of file reading (see above), and option read from the master $OASYS file, amended, and written to your local $HOME file will take precedence when you next run PRIMER.



"oa_pref" arguments valid for Primer

Preference	Type	Description	Valid arguments	Default
information_mass_display	<string>	switch to turn on/off mass information in the quick-pick information panel	OFF, ON	ON
conx_table_columns	<string>	Columns initially shown in connection table		<none>
part_table_columns	<string>	Columns initially shown in part table		<none>
start_in	<string>	Directory to start Primer in		<none>
checkpoint_dir	<string>	Directory for checkpoint files, or "none" to suppress them altogether		<none>

The following controls the default ascii file output

Preference	Type	Description	Valid arguments	Default
ABSTAT_asc	<string>	ABSTAT ascii file selected	ON, OFF	OFF
BNDOUT_asc	<string>	BNDOUT ascii file selected	ON, OFF	OFF
DEFGEO_asc	<string>	DEFGEO ascii file selected	ON, OFF	OFF
DEFORC_asc	<string>	DEFORC ascii file selected	ON, OFF	OFF
ELOUT_asc	<string>	ELOUT ascii file selected	ON, OFF	OFF
GCEOUT_asc	<string>	GCEOUT ascii file selected	ON, OFF	OFF
GLSTAT_asc	<string>	GLSTAT ascii file selected	ON, OFF	OFF
H3OUT_asc	<string>	H3OUT ascii file selected	ON, OFF	OFF
JNTFORC_asc	<string>	JNTFORC ascii file selected	ON, OFF	OFF
MATSUM_asc	<string>	MATSUM ascii file selected	ON, OFF	OFF
NCFORC_asc	<string>	NCFORC ascii file selected	ON, OFF	OFF
NODFOR_asc	<string>	NODFOR ascii file selected	ON, OFF	OFF
NODOUT_asc	<string>	NODOUT ascii file selected	ON, OFF	OFF
RBDOUT_asc	<string>	RBDOUT ascii file selected	ON, OFF	OFF
RWFORC_asc	<string>	RWFORC ascii file selected	ON, OFF	OFF
SECFORC_asc	<string>	SECFORC ascii file selected	ON, OFF	OFF
RCFORC_asc	<string>	RCFORC ascii file selected	ON, OFF	OFF
SBTOUT_asc	<string>	SBTOUT ascii file selected	ON, OFF	OFF
SLEOUT_asc	<string>	SLEOUT ascii file selected	ON, OFF	OFF
SPCFORC_asc	<string>	SPCFORC ascii file selected	ON, OFF	OFF
SPHOUT_asc	<string>	SPHOUT ascii file selected	ON, OFF	OFF
SWFORC_asc	<string>	SWFORC ascii file selected	ON, OFF	OFF
TPRINT_asc	<string>	TPRINT ascii file selected	ON, OFF	OFF
TRHIST_asc	<string>	TRHIST ascii file selected	ON, OFF	OFF

The following relate to assemblies

Preference	Type	Description	Valid arguments	Default
assembly_read_mode	<string>	Action for existing assemblies when reading assembly data in part tree	RESET, OVERWRITE, ASK	ASK

The following relate to assign mass function

Preference	Type	Description	Valid arguments	Default
assign_mass_percent_error_tolerance	<real>	Error tolerance (percent) for Assign Mass Calculation		5.0
assign_mass_includes_timestep_mass	<string>	massing up function includes timestep added mass	OFF, ON	OFF

The following control attributes of the attached function

Preference	Type	Description	Valid arguments	Default
attached_beam_pid	<string>	Attach through beam PIDs	ON, OFF	OFF
attached_tied_contact	<string>	Attach through tied contacts	ON, OFF	ON

attached_shortcut_tied	<string>	Attach through tied contacts when using the attached shortcut button	ON, OFF	ON

The following controls the default binary file output

Preference	Type	Description	Valid arguments	Default
ABSTAT_bin	<string>	ABSTAT binary file selected	ON, OFF	OFF
BNDOUT_bin	<string>	BNDOUT binary file selected	ON, OFF	OFF
DEFGEO_bin	<string>	DEFGEO binary file selected	ON, OFF	OFF
DEFORC_bin	<string>	DEFORC binary file selected	ON, OFF	OFF
ELOUT_bin	<string>	ELOUT binary file selected	ON, OFF	OFF
GCEOUT_bin	<string>	GCEOUT binary file selected	ON, OFF	OFF
GLSTAT_bin	<string>	GLSTAT binary file selected	ON, OFF	OFF
H3OUT_bin	<string>	H3OUT binary file selected	ON, OFF	OFF
JNTFORC_bin	<string>	JNTFORC binary file selected	ON, OFF	OFF
MATSUM_bin	<string>	MATSUM binary file selected	ON, OFF	OFF
NCFORC_bin	<string>	NCFORC binary file selected	ON, OFF	OFF
NODFOR_bin	<string>	NODFOR binary file selected	ON, OFF	OFF
NODOUT_bin	<string>	NODOUT binary file selected	ON, OFF	OFF
RBDOUT_bin	<string>	RBDOUT binary file selected	ON, OFF	OFF
RWFORC_bin	<string>	RWFORC binary file selected	ON, OFF	OFF
SECFORC_bin	<string>	SECFORC binary file selected	ON, OFF	OFF
RCFORC_bin	<string>	RCFORC binary file selected	ON, OFF	OFF
SBTOUT_bin	<string>	SBTOUT binary file selected	ON, OFF	OFF
SLEOUT_bin	<string>	SLEOUT binary file selected	ON, OFF	OFF
SPCFORC_bin	<string>	SPCFORC binary file selected	ON, OFF	OFF
SPHOUT_bin	<string>	SPHOUT binary file selected	ON, OFF	OFF
SWFORC_bin	<string>	SWFORC binary file selected	ON, OFF	OFF
TPRINT_bin	<string>	TPRINT binary file selected	ON, OFF	OFF
TRHIST_bin	<string>	TRHIST binary file selected	ON, OFF	OFF

The following control attributes of the model checking function

Preference	Type	Description	Valid arguments	Default
model_keyout_check	<string>	Generate check info on model keyout	ON, OFF	OFF
separate_check_file	<string>	write check info into fname.check	ON, OFF	OFF
error_configuration_file	<string>	user file to configure error/warning/ignore status		<none>
error_tags	<string>	show Primer error code for every error and warning	ON, OFF	OFF
contact				
contact_check_shells_for_thinning	<string>	report shells where LS-Dyna contact thickness less than 90% of actual	ON, OFF	OFF
contact_check_master_slave_contents	<string>	check contents of master and slave side of contact	ON, OFF	OFF
contact_stiffness_check	<string>	Check for contact stiffness mismatch	ON, OFF	OFF
max_allowable_contact_stiffness_ratio	<real>	Max allowable ratio of contact stiffnesses		100.0
sliding_contact				
contact_penetration_checks	<string>	Contact penetration checking	ON, OFF	OFF
contact_penetration_max_allowable_value	<real>	allowable penetration expressed as value		0.0
contact_penetration_min_remaining_depth	<real>	allowable penetration expressed as remaining segment depth		0.0
contact_penetration_min_remaining_depth_factor	<real>	allowable penetration expressed as remaining factor on segment depth		0.0
tied_contact				
contact_check_constrained_clash	<string>	check for segment clashes between constrained contacts	ON, OFF	ON

contact_check_all_connection_nodes_tie	<string>	all nodes of connections must tie	ON, OFF	ON
contact_check_tied	<string>	check that all tied contacts tie at least one node	ON, OFF	OFF
contact_check_all_slave_nodes_by_part_tie	<string>	all slave nodes defined by part must tie	ON, OFF	OFF
contact_check_all_slave_nodes_by_node_set_tie	<string>	all slave nodes defined by node set must tie	ON, OFF	OFF
contact_check_all_slave_nodes_on_shell_edge_tie	<string>	all slave nodes on shell edge must tie	ON, OFF	OFF
contact_check_all_slave_nodes_on_solid_face_tie	<string>	all slave nodes on face of solid must tie	ON, OFF	OFF
element_quality_checks				
element_quality_checks_active	<string>	Element quality check settings at program start	ON, OFF, PREF	PREF
shell_thickness_check	<string>	On section update warn if *ELEMENT_SHELL_THICKNESS is set	ON, OFF	ON
structural_element				
element_length_check	<string>	Element quality length checking	ON, OFF	OFF
element_min_length	<real>	Element quality minimum length		5
element_warpage_check	<string>	Element quality warpage checking	ON, OFF	OFF
element_max_warpage	<real>	Element quality maximum warpage	0.0 - 180.0	20
element_skew_check	<string>	Element quality skew checking	ON, OFF	OFF
element_max_skew	<real>	Element quality maximum skew	0.0 - 180.0	60
element_aspect_ratio_check	<string>	Element quality aspect ratio checking	ON, OFF	OFF
element_max_aspect_ratio	<real>	Element quality maximum aspect ratio		5
element_tria_angle_check	<string>	Element quality tria internal angle checking	ON, OFF	OFF
element_max_tria_angle	<real>	Element quality tria maximum internal angle	0.0 - 180.0	120
element_min_tria_angle	<real>	Element quality tria minimum internal angle	0.0 - 180.0	30
element_quad_angle_check	<string>	Element quality quad internal angle checking	ON, OFF	OFF
element_max_quad_angle	<real>	Element quality quad maximum internal angle	0.0 - 180.0	140
element_min_quad_angle	<real>	Element quality quad minimum internal angle	0.0 - 180.0	40
tet_skew_check	<string>	Skew check on Tet elements	ON, OFF	OFF
elem_free_end_check	<string>	Both nodes of 1D elements should be structural	ON, OFF	OFF
part_quality_check	<string>	Part quality check	ON, OFF	OFF
part_quality_min_elem	<integer>	Min required number of elements for part quality check		1
part_quality_percent_threshold	<real>	Threshold for part quality check		10.0
<<<<<<< .mine				
weighting_factors				
quality_wt_leng	<real>	Weighting factor for length		1.0
quality_wt_aspr	<real>	Weighting factor for aspect ratio		1.0
quality_wt_warp	<real>	Weighting factor for warpage		1.0
quality_wt_skew	<real>	Weighting factor for skew		1.0
quality_wt_lang	<real>	Weighting factor for min angle		1.0
quality_wt_uang	<real>	Weighting factor for max angle		1.0
history				
database_node_check	<string>	Check for absence of database history node	ON, OFF	OFF
database_shell_check	<string>	Check for absence of database history shell	ON, OFF	OFF
database_solid_check	<string>	Check for absence of database history solid	ON, OFF	OFF
database_beam_check	<string>	Check for absence of database history beam	ON, OFF	OFF

database_sbelt_check	<string>	Check for absence of database history seatbelt	ON, OFF	OFF
database_discrete_check	<string>	Check for absence of database history discrete	ON, OFF	OFF
include				
part_element_include_check	<string>	Check elements in same include as part	ON, OFF	OFF
element_node_include_check	<string>	Check nodes in same include as element	ON, OFF	OFF
mass_node_include_check	<string>	Check ELEMENT_MASS(_NODE) in same include as node	ON, OFF	OFF
section_include_check	<string>	Check sections in same include as part	ON, OFF	OFF
material_include_check	<string>	Check materials in same include as part	ON, OFF	OFF
joint				
minimum_joint_mass_value	<real>	minimum value of mass on joint node		0.0
minimum_cylindrical_joint_size	<real>	minimum value for size of cylindrical joint		0.0
material				
mat24_vp_check	<string>	Check VP set when strain rate is active	ON, OFF	OFF
mat24_strain_check_limit	<real>	Limiting strain value for Matl curve and table check		100.0
mat24_required_table_curve_separation_factor	<real>	Minimum required table curve separation factor		0.01
mat24_check_curve_discretization	<string>	Check for accuracy loss from discretization of MAT24 curves	ON, OFF	OFF
model_quality_checks				
model_quality_checks_active	<string>	Model quality check settings at program start	ON, OFF, PREF	PREF
model_timestep_check	<string>	Model timestep check	ON, OFF	OFF
model_min_timestep	<real>	Model minimum timestep		1.e-6
model_max_timestep	<real>	Model maximum timestep		<none>
model_added_mass_check	<string>	Model added mass check	ON, OFF	OFF
model_max_added_mass	<real>	Model maximum added mass		<none>
model_max_added_mass_percent	<real>	Model maximum added mass percent		5.0
part_added_mass_check	<string>	Part added mass check	ON, OFF	OFF
part_max_added_mass	<real>	Part maximum added mass		<none>
part_max_added_mass_percent	<real>	Part maximum added mass percent		5.0
spotweld_part_max_added_mass	<real>	Spotweld Part maximum added mass		<none>
spotweld_part_max_added_mass_percent	<real>	Spotweld Part maximum added mass percent		5.0
element				
element_timestep_check	<string>	Element timestep checking	ON, OFF	OFF
element_min_timestep	<real>	Element minimum timestep		1.e-6
spotweld_min_timestep	<real>	Spotweld minimum timestep		1.e-6
element_added_mass_check	<string>	Element added mass check	ON, OFF	OFF
element_max_added_mass	<real>	Element maximum added mass		<none>
element_max_added_mass_percent	<real>	Element maximum added mass percent		5.0
spotweld_max_added_mass	<real>	Spotweld maximum added mass		<none>
spotweld_max_added_mass_percent	<real>	Spotweld maximum added mass percent		5.0
element_overlap_check	<string>	Element overlap checking	ON, OFF	OFF
part_checks				
empty_part	<string>	check for empty parts	ON, OFF	ON
def_part_continuity_check	<string>	Deformable part mesh continuity check	ON, OFF	OFF
rigid				
nodal_rigid_body_minimum_mass	<real>	minimum mass required for nodal rigid body (0.0 for off)		0.0

rigid_part_minimum_mass	<real>	minimum mass required for rigid part (0.0 for auto)		0.0
accelerometer_minimum_mass	<real>	minimum mass required for accelerometer		0.0
rigid_body_merge_check	<string>	Rigid body merge checking	ON, OFF	OFF
rigid_body_merge_max_separation	<real>	Maximum size for rigid body merge		100.0
rigid_body_continuity_check	<string>	Rigid body mesh continuity check	ON, OFF	OFF
rigid_body_min_elem	<integer>	Min required num of elems for rigid body continuity check		3
nodal_rigid_body_size_check	<string>	Check size of nodal rigid bodies	ON, OFF	OFF
maximum_nodal_rigid_body_size	<real>	Maximum size for nodal rigid body		100.0
other				
allow_unused_nrb_master_node	<string>	allow unused nodal rigid body master node	ON, OFF	OFF
control_solution_undef_isnan_check	<string>	Flag undefined ISNAN parameter as an error	ON, OFF	OFF
warn_param_8_chars	<string>	Warn if 8 character parameter names found	ON, OFF	ON

The following apply to the connections function

Preference	Type	Description	Valid arguments	Default
allow_mig_weld_to_feature_line	<logical>	TRUE if a MIG welds can be created on feature lines as well and free edges	TRUE, FALSE	FALSE
automatically_create_ connections_from_welds	<string>	automatically make connections from existing welds	ON, OFF	ON
connection_allow_clinch	<logical>	TRUE if shell elements of different parts meshed together forming a clinch can be connected	TRUE, FALSE	FALSE
connection_file_type	<string>	Default spotweld file type	PRIMER_SPOTWELD, PRIMER_CONNECTION, CATIA, UG, USER	PRIMER_CONNECTION
connection_read_script	<string>	User defined JavaScript to read connections		<none>
connection_write_script	<string>	User defined JavaScript to write connections		<none>
connection_same_part	<logical>	TRUE if a part can be joined to itself with a connection entity	TRUE, FALSE	FALSE
connection_node_ element_numbering_rule	<string>	default rule for numbering of connection nodes and elements	HIGHEST_PLUS_ONE, LAYER_FIRST_FREE, LAYER_HIGHEST_FREE, LAYER_HIGHEST_PLUS_ONE	LAYER_HIGHEST_ PLUS_ONE
connection_general_ item_numbering_rule	<string>	default rule for numbering of connection general items	HIGHEST_PLUS_ONE, LAYER_FIRST_FREE, LAYER_HIGHEST_FREE, LAYER_HIGHEST_PLUS_ONE	LAYER_HIGHEST_ PLUS_ONE
contact_check_all_ connection_nodes_tie	<string>	all nodes of connections must tie	ON, OFF	ON
spot_max_scanlines	<integer>	Max number of lines displayed for spotweld read panel		50

warn_of_connection_ attached_to_shell_on_edge	<string>	warn if connection node attaches to shell with free edge	ON, OFF	OFF
adhesive				
adhesive_solid_percentage	<real>	Max percentage of solids created for adhesive. Anything less is invalid	0 - 100	50.0
autoweld				
autoweld_diff_seam_proximity	<real>	Proximity check for different autoweld seams	0 - 1	0.5
autoweld_same_seam_proximity	<real>	Proximity check for same autoweld seams	0 - 1	0.5
bolts				
use_parent_layer_for_bolts	<logical>	bolt connection items put into parent layer where possible	TRUE, FALSE	TRUE
strict_layer_method_for_bolts	<logical>	abort bolt creation if include range undefined	TRUE, FALSE	FALSE
bolt_entity_numbering_rule	<string>	layer rule for numbering of bolt FE (over-ruling setting)	CONNECTION_LABEL_RULE, LAYER_FIRST_FREE, LAYER_HIGHEST_FREE, LAYER_HIGHEST_PLUS_ONE	CONNECTION_ LABEL_RULE
adjust_bolt_mass_on_create	<logical>	on bolt creation add mass for stability if needed	TRUE, FALSE	FALSE
add_database_history_ beam_for_bolts	<logical>	for beam type bolts create DTHB	TRUE, FALSE	FALSE
bolt_angle_tolerance	<real>	shell angle tolerance when creating bolts is max(bolt_ang, spot_ang)		0.0
spotweld				
spotweldbeam_length_check	<string>	Spotweld beam length checking	ON, OFF	ON
spotweldbeam_min_length	<real>	Spotweld beam minimum length		0.5
spotweldbeam_max_length	<real>	Spotweld beam maximum length		5.0
spotweldbeam_max_total_length	<real>	Spotweld beam maximum total length		10.0
spotweldbeam_panel_check	<string>	Spotweld beam panel checking	ON, OFF	ON
spotweldbeam_max_panels	<integer>	Spotweld beam maximum number of panels		5
spotweldbeam_distance_check	<string>	Spotweld beam pitch checking	ON, OFF	ON
spotweldbeam_min_distance	<real>	Spotweld beam minimum pitch		10.0
spotweld_warpage_check	<string>	solid spotweld warpage checking	ON, OFF	OFF
spotweld_max_warpage	<real>	solid spotweld maximum warpage	0.0 - 180.0	20

spotweldbeam_pid	<logical>	TRUE if _PID option is used for spotweld beams	TRUE, FALSE	TRUE
solid_spotweld_consider_ free_edges	<string>	consider free edges for orientation when creating/remaking solid spotwelds	ON, OFF	ON
contact_penetration_checking				
contact_penchk_dup_shells	<string>	Allocation of contact segments to duplicate shells	AUTOMATIC, THINNEST, THICKEST	AUTOMATIC
constructed_properties				
construct_material_card_for_latent	<string>	Construct material card for latent materials in model	ON, OFF	OFF
constructed_material_card_density	<real>	Default density for material cards constructed by Primer		7.85e-9
constructed_material_card_stiffness	<real>	Default stiffness for material cards constructed by Primer		200000
constructed_material_card_spring_stiffness	<real>	Default stiffness for spring material cards constructed by Primer		1.0
constructed_section_card_thickness	<real>	Default thickness for section shell cards constructed by Primer		1.0

The drive mappings allow PRIMER to convert equivalent folder names from Windows to Unix and visa versa

Preference	Type	Description	Valid arguments	Default
drive_a	<string>	Mapping from Windows drive A: to unix path		<none>
drive_b	<string>	Mapping from Windows drive B: to unix path		<none>
drive_c	<string>	Mapping from Windows drive C: to unix path		<none>
drive_d	<string>	Mapping from Windows drive D: to unix path		<none>
drive_e	<string>	Mapping from Windows drive E: to unix path		<none>
drive_f	<string>	Mapping from Windows drive F: to unix path		<none>
drive_g	<string>	Mapping from Windows drive G: to unix path		<none>
drive_h	<string>	Mapping from Windows drive H: to unix path		<none>
drive_i	<string>	Mapping from Windows drive I: to unix path		<none>
drive_j	<string>	Mapping from Windows drive J: to unix path		<none>
drive_k	<string>	Mapping from Windows drive K: to unix path		<none>
drive_l	<string>	Mapping from Windows drive L: to unix path		<none>
drive_m	<string>	Mapping from Windows drive M: to unix path		<none>
drive_n	<string>	Mapping from Windows drive N: to unix path		<none>
drive_o	<string>	Mapping from Windows drive O: to unix path		<none>
drive_p	<string>	Mapping from Windows drive P: to unix path		<none>
drive_q	<string>	Mapping from Windows drive Q: to unix path		<none>
drive_r	<string>	Mapping from Windows drive R: to unix path		<none>
drive_s	<string>	Mapping from Windows drive S: to unix path		<none>
drive_t	<string>	Mapping from Windows drive T: to unix path		<none>
drive_u	<string>	Mapping from Windows drive U: to unix path		<none>
drive_v	<string>	Mapping from Windows drive V: to unix path		<none>
drive_w	<string>	Mapping from Windows drive W: to unix path		<none>
drive_x	<string>	Mapping from Windows drive X: to unix path		<none>

drive_y	<string>	Mapping from Windows drive Y: to unix path		<none>
drive_z	<string>	Mapping from Windows drive Z: to unix path		<none>

The following options control many aspects of image appearance. The `overlay_<xxx>` options affect only the hidden-line overlay on shaded and all data-bearing plots. They have no effect on wireframe (**LINE**) or hidden-line plots.

Preference	**Type**	**Description**	**Valid arguments**	**Default**
auto_minimise	<string>	If an edit panel obscures the graphics window, it will automatically minimise when the mouse moves off the panel	OFF, ON, PICK, ALWAYS	OFF
background_colour	<string>	Background colour	WHITE, GREY, BLACK, RED, GREEN, BLUE, CYAN, MAGENTA, YELLOW	BLACK
backing_store	<string>	Backing store refresh switch and method	OFF, ON, PIXMAP, PBUFFER	ON
contact_shaded_display	<string>	How contact segments are rendered in shaded display mode	SOLID_HATCHED, SOLID, STIPPLE_1, STIPPLE_2, STIPPLE_4, STIPPLE_8, STIPPLE_16	STIPPLE_1
contact_colour	<string>	Colour used to draw contact surfaces	DEFAULT, BY_SIDE, WHITE, GREY, BLACK, RED, GREEN, BLUE, CYAN, MAGENTA, YELLOW	DEFAULT
contour_levels	<integer>	Number of contour (CT/SI/VEC) levels	2 - 13	6
existing_panel_action	<string>	Action for existing panels when new one mapped	NONE, IN_SITU, TIDY	NONE
graphics_refresh	<string>	Refresh graphics window when exposed	OFF, ON	ON
graphics_type	<string>	Graphics format to start Primer with	X8, X24, X, Opengl, Default	<none>
initial_plot_mode	<string>	Initial drawing mode	LINE, HIDDEN, SHADED	SHADED
maximise	<logical>	Maximise window when Primer started	TRUE, FALSE	FALSE
overlay_colour	<string>	Overlay colour	WHITE, GREY, BLACK, RED, GREEN, BLUE, CYAN, MAGENTA, YELLOW, ELEMENT	GREY
overlay_mode	<string>	Overlay drawn	OFF, FREE, FEATURE, ALL	FREE
overlay_line_width	<integer>	Width of overlay lines	1 - 10	1
panel_placement	<string>	Where new floating panels are located	LEFT, R_BORD, RIGHT, TOP, B_BORD, BOTTOM, FREE	FREE
placement	<string>	Location for initial window on multi-screen display	LEFT, RIGHT, BOTTOM, TOP, LEFT_BOTTOM, LEFT_TOP, RIGHT_BOTTOM, RIGHT_TOP	<none>
plot_border	<string>	Border drawn on plot	OFF, ON	ON
sketch_colour	<string>	Background colour	WHITE, GREY, BLACK, RED, GREEN, BLUE, CYAN, MAGENTA, YELLOW	WHITE
spotweld_size	<integer>	Spotweld beam graphics size	1 - 100	40
text_colour	<string>	Text colour	WHITE, GREY, BLACK, RED, GREEN, BLUE, CYAN, MAGENTA, YELLOW	WHITE
image_format	<string>	Default image format	BMP_8_C, BMP_8_UN, PNG_8, GIF_8, BMP_24_UN, PNG_24, JPG_24, PPM_24	JPG_24
white_background_image	<logical>	Write images with white background	TRUE, FALSE	FALSE

The following control settings and warnings when reading files

Preference	**Type**	**Description**	**Valid arguments**	**Default**
convert_rbe2_cnrb	<logical>	Action to convert RBE2 nastran card to CONSTRAINED NRB	TRUE, FALSE	FALSE
correct_seatbelt_topology	<string>	Action to correct errors in 1D seatbelt topology on input	AUTOMATIC, MANUAL	MANUAL

duplicated_keyword_warning	<string>	give ’printg’ style warning whenever duplicated keyword is read	ON, OFF	ON
extensions_for_file_read_on_windows	<string>	considered extensions for file read on windows (e.g. .key;*.dyn)		*.k;*.key
extension_for_file_read_on_unix	<string>	considered extension for file read on windows (e.g. .k* or *)		*.k*
find_data_for_scan	<logical>	Find missing parameter and/or include transform data during model scan	TRUE, FALSE	TRUE
input_buffer_size	<integer>	File buffer size for reading ascii files	2 - 2147483646	4096
input_echo_frequency	<integer>	Progress echo interval when reading ascii files	1 - 2147483646	1000
read_duplicates	<string>	How duplicate labelled items are handled during keyword input	ALL, NONE, LS971, NODE	LS971
read_embedded_comments	<logical>	Read and store embedded comments in the keyword file	TRUE, FALSE	TRUE
suppress_text_box_messages_on_keyin	<string>	suppress all text box messages on keyin (just write to log)	ON, OFF	OFF
warn_parameter_order	<logical>	Warn if parameters in INCLUDE_TRANSFORM used before being defined	TRUE, FALSE	TRUE
read_hm_comments	<string>	read HM comments upon input (comments used to construct assemblies, colours and in some cases, titles)	ON, OFF	ON
read_ansa_comments	<string>	read ANSA comments upon input (comments used to construct assemblies)	ON, OFF	ON
copy_hm_comment_title	<string>	Set material and section titles to HM comment titles if no current title is set	ON, OFF	ON

These settings control the attributes and behaviour of Primer’s generic Keyword editor.

Preference	**Type**	**Description**	**Valid arguments**	**Default**
kw_edit_init_defs	<integer>	Number of definitions to show initially	1 - 100	5
kw_edit_init_rows	<integer>	Maximum number of rows to show initially	1 - 100	10

The following control element labelling

Preference	**Type**	**Description**	**Valid arguments**	**Default**
max_labels	<integer>	Maximum number of labels to display	1 - 2147483646	1000
label_warning	<logical>	Display a warning if the maximum number of labels is reached	TRUE, FALSE	TRUE
NODE_labelled	<string>	Nodes labelled	ON, OFF	OFF
SOLID_labelled	<string>	Solids labelled	ON, OFF	OFF
BEAM_labelled	<string>	Beams labelled	ON, OFF	OFF
SHELL_labelled	<string>	Shells labelled	ON, OFF	OFF
TSHELL_labelled	<string>	Thick shells labelled	ON, OFF	OFF
DISCRETE_labelled	<string>	Springs/dampers labelled	ON, OFF	OFF
MASS_labelled	<string>	Lumped masses labelled	ON, OFF	OFF

The following strings and values control laser plotting setup

Preference	**Type**	**Description**	**Valid arguments**	**Default**
laser_paper_size	<string>	Default paper size	US, A4	A4
laser_orientation	<string>	Default page orientation	Portrait, Landscape	Landscape
laser_mode	<string>	Default laser mode	Colour, Greyscale	Greyscale
laser_insert_file	<string>	Valid filename		<none>
laser_top_margin	<real>	Top margin size in mm		10
laser_bottom_margin	<real>	Bottom margin size in mm		30
laser_left_margin	<real>	Left margin size in mm		20
laser_right_margin	<real>	Right margin size in mm		10

The following options control image lighting

Preference	Type	Description	Valid arguments	Default
shaded_ambient	<real>	Percentage ambient light (0-100)	0.0 - 100.0	30
shaded_diffuse	<real>	Percentage diffuse brightness (0-100)	0.0 - 100.0	70
shaded_shininess	<real>	Percentage specular brightness (0-100)	0.0 - 100.0	70
shaded_saturation	<real>	Percentage colour saturation (0-100)	0.0 - 100.0	50

The following options control display of local material directions

Preference	Type	Description	Valid arguments	Default
locaxes_type	<string>	Set size of triad as a function of raw or screen coordinates	SCREEN_SPACE, MODEL_SPACE	SCREEN_SPACE
locaxes_size	<real>	Triad size		75
locaxes_colour	<string>	Display triads for composite parts	DEFAULT, MTL_COLOUR	DEFAULT

The following options relate to macros

Preference	Type	Description	Valid arguments	Default
macro_auto_record	<string>	Filename to automatically record macro to		<none>
macro_directory	<string>	Directory in which PRIMER looks for macros		$OA_INSTALL/primer_library/macros

The following options affect the appearance and behaviour of the graphical user interface (see section 2.4.4.1), left handed support, and the mouse

Preference	Type	Description	Valid arguments	Default
display_factor	<real>	Factor on display size (0.5-2.0)	0.5 - 2.0	1.0
display_brightness	<real>	Menu brightness (0.0-1.0)	0.0 - 1.0	1.0
display_saturation	<real>	Menu colour saturation (0.0-1.0)	0.0 - 1.0	1.0
dv_left_shift	<string>	Dyn view action for shift + Left mouse	ROTATION_XYZ, ROTATION_XY, ROTATION_Z, ROTATION_SPHERE, TRANSLATION, ZOOM_UP_+VE, ZOOM_DOWN_+VE, UNUSED	ROTATION_XYZ
dv_middle_shift	<string>	Dyn view action for shift + Middle mouse	ROTATION_XYZ, ROTATION_XY, ROTATION_Z, ROTATION_SPHERE, TRANSLATION, ZOOM_UP_+VE, ZOOM_DOWN_+VE, UNUSED	TRANSLATION
dv_right_shift	<string>	Dyn view action for shift + Right mouse	ROTATION_XYZ, ROTATION_XY, ROTATION_Z, ROTATION_SPHERE, TRANSLATION, ZOOM_UP_+VE, ZOOM_DOWN_+VE, UNUSED	ZOOM_UP_+VE
dv_left_ctrl	<string>	Dyn view action for ctrl + Left mouse	ROTATION_XYZ, ROTATION_XY, ROTATION_Z, ROTATION_SPHERE, TRANSLATION, ZOOM_UP_+VE, ZOOM_DOWN_+VE, UNUSED	ROTATION_XYZ
dv_middle_ctrl	<string>	Dyn view action for ctrl + Middle mouse	ROTATION_XYZ, ROTATION_XY, ROTATION_Z, ROTATION_SPHERE, TRANSLATION, ZOOM_UP_+VE, ZOOM_DOWN_+VE, UNUSED	TRANSLATION
dv_right_ctrl	<string>	Dyn view action for ctrl + Right mouse	ROTATION_XYZ, ROTATION_XY, ROTATION_Z, ROTATION_SPHERE, TRANSLATION, ZOOM_UP_+VE, ZOOM_DOWN_+VE, UNUSED	ZOOM_UP_+VE
dv_left_both	<string>	Dyn view action for shift+ctrl + Left mouse	ROTATION_XYZ, ROTATION_XY, ROTATION_Z, ROTATION_SPHERE, TRANSLATION, ZOOM_UP_+VE, ZOOM_DOWN_+VE, UNUSED	ROTATION_XYZ
dv_middle_both	<string>	Dyn view action for shift+ctrl + Middle mouse	ROTATION_XYZ, ROTATION_XY, ROTATION_Z, ROTATION_SPHERE, TRANSLATION, ZOOM_UP_+VE, ZOOM_DOWN_+VE, UNUSED	TRANSLATION

dv_right_both	<string>	Dyn view action for shift+ctrl + Right mouse	ROTATION_XYZ, ROTATION_XY, ROTATION_Z, ROTATION_SPHERE, TRANSLATION, ZOOM_UP_+VE, ZOOM_DOWN_+VE, UNUSED	ZOOM_UP_+VE
dv_shift_action	<string>	Dynamic viewing mode for shift + mouse button	CURRENT, WIREFRAME, FREE_EDGE, UNUSED	CURRENT
dv_ctrl_action	<string>	Dynamic viewing mode for ctrl + mouse button	CURRENT, WIREFRAME, FREE_EDGE, UNUSED	WIREFRAME
dv_both_action	<string>	Dynamic viewing mode for shift+ctrl + mouse button	CURRENT, WIREFRAME, FREE_EDGE, UNUSED	FREE_EDGE
font_size	<string>	Menu font size	SMALL, DEFAULT, LARGE	DEFAULT
font_type	<string>	Menu font typeface and strength	HELVETICA, HELVETICA-BOLD, TIMES, TIMES-BOLD, COURIER, COURIER-BOLD	HELVETICA
left_handed	<string>	Left handed switching of mouse and/or keyboard	NONE, MOUSE, KEYBOARD, ALL	NONE
zoom_factor	<real>	Zoom Factor for mouse wheel (0.01-0.2)	0.01 - 0.2	0.05
czoom_factor	<real>	Factor for right mouse dynamic zoom (0.01-0.2)	0.01 - 0.2	0.05
kzoom_factor	<real>	Factor for +/- keyboard short-cut keys	0.01 - 100.0	2.0

The following settings control meshing and hole properties

Preference	**Type**	**Description**	**Valid arguments**	**Default**
hole_element_method	<string>	Elements around hole specified by number/size	NUMBER, SIZE	NUMBER
hole_element_number	<integer>	Number of elements around hole	3 - 1000	4
hole_element_size	<real>	Size of elements around hole		10.0
hole_diameter	<real>	Hole diameter		10.0
hole_rotate_angle	<real>	Rotation angle		10.0
remesh_area	<logical>	Remesh area when removing hole	TRUE, FALSE	TRUE
washer	<logical>	Create washer elements around hole	TRUE, FALSE	TRUE
washer_diameter	<real>	Washer diameter		20.0
washer_elements	<integer>	Number of washer elements	1 - 5	1
mesh_element_size	<real>	Element size for shells		10.0
mesh_feature_line	<logical>	Limit mesh area by feature lines	TRUE, FALSE	FALSE
mesh_feature_line_angle	<real>	Mesh feature line angle		20.0

The following affect model build

Preference	**Type**	**Description**	**Valid arguments**	**Default**
set_database_dir	<string>	Default directory for databases when using model database		<none>
set_template_dir	<string>	Default directory for templates when using model database		<none>
model_build_delete_latent_items	<string>	on build remove missing items if possible	ON, OFF	ON
move_include_to_master	<string>	on build move extra data file to master if any delete	OFF, ON, ASK	ASK
extensions_for_database_from_dir	<string>	considered extensions for database from dir		.key;.k;.dyn

The following control settings when merging models

Preference	**Type**	**Description**	**Valid arguments**	**Default**
merge_range_location	<string>	Choose the model to take any master file numbering range information from upon model merge	FIRST, SECOND, NEITHER	FIRST
merge_thumb_location	<string>	Choose the model to take any master file thumbnail information from upon model merge	FIRST, SECOND, NEITHER	NEITHER

merge_incl_path_location	<string>	Choose the model to take any master file INLCUDE_PATH information from upon model merge	FIRST, SECOND, NEITHER	FIRST

The following options relate to model modified

Preference	Type	Description	Valid arguments	Default
threshold_for_modified_nodal_coordinates	<real>	Threshold difference for comparing nodal coordinates		0.0
sig_fig_for_modified_floats	<integer>	Number of significant figures for comparing real numbers	1 - 6	6
check_for_modified_header_comments	<string>	Check for differences in header comments	OFF, ON	ON

If a selection menu is not wide enough to display all the contents, it can be expanded automatically by the following (see section 2.4.4.3)

Preference	Type	Description	Valid arguments	Default
menu_expand	<string>	Automatic menu expansion on/off switch	OFF, ON	ON
menu_expand_delay	<real>	Factor on delay time before expansion	0.1 - 5.0	1.0
menu_expand_speed	<real>	Factor on menu expansion speed	0.1 - 5.0	1.0
menu_sketch	<string>	Whether or not to show sketch menu items when cursor hovered over menu row	OFF, ON	ON
menu_label	<string>	Whether or not menu sketching also shows item labels	OFF, ON	ON

These settings apply to object picking

Preference	Type	Description	Valid arguments	Default
predictive_pick	<string>	Whether or not to show what will be picked based on the current cursor position	OFF, ON	ON
predictive_label	<string>	Whether or not predictive picking also shows item labels	OFF, ON	ON
query_ambiguous	<string>	If screen picking is ambiguous, ON will offer the selection menu, OFF will select nearest	OFF, ON	ON

The following control settings when writing files

Preference	Type	Description	Valid arguments	Default
ascii_file_format	<string>	Output format for ascii files	NATIVE, UNIX	NATIVE
assembly_output_format	<string>	Format for writing post *END assembly data	PRIMER, HYPERMESH, ANSA, CUSTOMER	PRIMER
customer_comment_output_name	<string>	Name that appears on customer comment output button. Limited to 10 characters		customer
autocreate_ztf	<logical>	Write out jobname.ztf file for D3PLOT	TRUE, FALSE	FALSE
dyna_output_version	<string>	The version of dyna used when writing a keyword file	940, 950, 960, 960+, 970v3858, 970v5434, 970v6763, 971R2, 971R3, 971R4, 971R5, 971dev	971R4
dyna_output_style	<string>	The floating point format used when writing a keyword file	native, common, rounded	common
add_dyna_key_file_extension	<logical>	Add file extension to dyna keyword files if unspecified	TRUE, FALSE	TRUE
dyna_key_file_extension_master	<string>	file extension for Dyna master keyword files		.key
dyna_key_file_extension_include	<string>	file extension for Dyna include files		.key
element_mass_part_970	<logical>	TRUE if *ELEMENT_MASS_PART can be written to a 970 deck	TRUE, FALSE	FALSE
include_file_paths	<string>	Write absolute/relative pathnames in INCLUDE statements	ABSOLUTE, RELATIVE	ABSOLUTE

include_file_method	<string>	Default method of writing include files	IN_SUBDIR, IN_SAME_DIR, SELECT_FILES	IN_SUBDIR
directory_name_for_include_keyout	<string>	Directory name for include keyout IN_SUBDIR mode		INCL
include_mass_comment	<string>	Write the mass of an include file as a comment	ON, OFF	OFF
output_buffer_size	<integer>	File buffer size for writing ascii files	2 - 2147483646	4096
output_echo_frequency	<integer>	Progress echo interval when writing ascii files	1 - 2147483646	1000
output_os	<string>	Operating system type for include file naming	NATIVE, WINDOWS, UNIX	NATIVE
model_mass_comment	<string>	Write the model mass and C of G as a comment	ON, OFF	OFF
parameter_in_string	<string>	Processing of parameters in text and titles during keyout	AUTOMATIC, REPLACE_AMPERSAND, INSERT_VALUE, VERBATIM	AUTOMATIC
skip_undefined_message	<string>	Preference for skipping the writing of undefined item message	ON, OFF	OFF
solid_two_line_output	<logical>	Write out all solids in LS Dyna version 970 and above 2 line format	TRUE, FALSE	FALSE
suppress_keyout_all_connections	<string>	suppress write of all connections	ON, OFF	OFF
suppress_keyout_ autocreated_connections	<string>	Don’t write connections created from welds by Primer	ON, OFF	OFF
suppress_keyout_geometry	<string>	Don’t write geometry to keyword file	ON, OFF	OFF
write_data_field_headers	<logical>	Write comment line of data field acronyms to keyword file	TRUE, FALSE	FALSE
write_embedded_comments	<logical>	Write embedded comments to keyword file	TRUE, FALSE	TRUE
write_hm_comments	<logical>	Write HM comments to keyword file	TRUE, FALSE	FALSE
write_primer_part_colours	<logical>	Write primer part colour comments to keyword file	TRUE, FALSE	TRUE
write_thumbnails	<logical>	Write any include file thumbnail images to keyword file	TRUE, FALSE	TRUE

Options to control the behaviour and appearance of the Part Tree.

Preference	**Type**	**Description**	**Valid arguments**	**Default**
ptree_parts_top_level	<logical>	If TRUE parts are always expanded at the top level	TRUE, FALSE	TRUE
part_tree_blank_mode	<string>	Sets the mode for blanking/unblanking/only for includes/assemblies in the part tree	PART, ELEMENT, LEGACY	ELEMENT
part_tree_drag_material	<string>	Take or leave materials referenced by the part card when moving parts in the part tree	TAKE, LEAVE	TAKE
part_tree_drag_section	<string>	Take or leave sections referenced by the part card when moving parts in the part tree	TAKE, LEAVE	TAKE
part_tree_drag_hourglass	<string>	Take or leave hourglass cards referenced by the part card when moving parts in the part tree	TAKE, LEAVE	LEAVE
part_tree_drag_equation_of_state	<string>	Take or leave equation of state cards referenced by the part card when moving parts in the part tree	TAKE, LEAVE	LEAVE
part_tree_drag_other	<string>	Take or leave junior items that reference parts when moving parts in the part tree	TAKE, LEAVE	LEAVE
ptree_show_beam	<logical>	If TRUE a Beam category will be included in the tree	TRUE, FALSE	FALSE

ptree_show_box	<logical>	If TRUE a Define box category will be included in the tree	TRUE, FALSE	FALSE
ptree_show_coordinate	<logical>	If TRUE a Define coordinate category will be included in the tree	TRUE, FALSE	FALSE
ptree_show_contact	<logical>	If TRUE a Contact category will be included in the tree	TRUE, FALSE	FALSE
ptree_show_curve	<logical>	If TRUE a Define curve category will be included in the tree	TRUE, FALSE	FALSE
ptree_show_database_history	<logical>	If TRUE a Database history category will be included in the tree	TRUE, FALSE	FALSE
ptree_show_eos	<logical>	If TRUE an Equation of state category will be included in the tree	TRUE, FALSE	FALSE
ptree_show_function	<logical>	If TRUE a Define function category will be included in the tree	TRUE, FALSE	FALSE
ptree_show_group	<logical>	If TRUE a Groups category will be included in the tree	TRUE, FALSE	FALSE
ptree_show_hourglass	<logical>	If TRUE an Hourglass category will be included in the tree	TRUE, FALSE	FALSE
ptree_show_joint	<logical>	If TRUE a Joints category will be included in the tree	TRUE, FALSE	FALSE
ptree_show_mass	<logical>	If TRUE a Mass category will be included in the tree	TRUE, FALSE	FALSE
ptree_show_nodal_rigid_body	<logical>	If TRUE a Nodal RB category will be included in the tree	TRUE, FALSE	FALSE
ptree_show_prescribed_motion	<logical>	If TRUE a Boundary prescribed motion category will be included in the tree	TRUE, FALSE	FALSE
ptree_show_pretensioner	<logical>	If TRUE a Pretensioner category will be included in the tree	TRUE, FALSE	FALSE
ptree_show_retractor	<logical>	If TRUE a Retractor category will be included in the tree	TRUE, FALSE	FALSE
ptree_show_rigidwall	<logical>	If TRUE a Rigidwall category will be included in the tree	TRUE, FALSE	FALSE
ptree_show_sbelt	<logical>	If TRUE a Seatbelt category will be included in the tree	TRUE, FALSE	FALSE
ptree_show_sd_orientation	<logical>	If TRUE a Define spring/damper orientation category will be included in the tree	TRUE, FALSE	FALSE
ptree_show_section	<logical>	If TRUE a Section category will be included in the tree	TRUE, FALSE	FALSE
ptree_show_set_beam	<logical>	If TRUE a Set beam category will be included in the tree	TRUE, FALSE	FALSE
ptree_show_set_discrete	<logical>	If TRUE a Set discrete category will be included in the tree	TRUE, FALSE	FALSE
ptree_show_set_node	<logical>	If TRUE a Set node category will be included in the tree	TRUE, FALSE	FALSE
ptree_show_set_part	<logical>	If TRUE a Set part category will be included in the tree	TRUE, FALSE	FALSE
ptree_show_set_segment	<logical>	If TRUE a Set segment category will be included in the tree	TRUE, FALSE	FALSE
ptree_show_set_shell	<logical>	If TRUE a Set shell category will be included in the tree	TRUE, FALSE	FALSE
ptree_show_set_solid	<logical>	If TRUE a Set solid category will be included in the tree	TRUE, FALSE	FALSE
ptree_show_set_tshell	<logical>	If TRUE a Set tshell category will be included in the tree	TRUE, FALSE	FALSE
ptree_show_shell	<logical>	If TRUE a Shell category will be included in the tree	TRUE, FALSE	FALSE
ptree_show_slipring	<logical>	If TRUE a Slipring category will be included in the tree	TRUE, FALSE	FALSE
ptree_show_solid	<logical>	If TRUE a Solid category will be included in the tree	TRUE, FALSE	FALSE
ptree_show_spc	<logical>	If TRUE a Boundary SPC category will be included in the tree	TRUE, FALSE	FALSE
ptree_show_spring	<logical>	If TRUE a Spring category will be included in the tree	TRUE, FALSE	FALSE
ptree_show_surface	<logical>	If TRUE a (contact) Surface category will be included in the tree	TRUE, FALSE	FALSE
ptree_show_table	<logical>	If TRUE a Define table category will be included in the tree	TRUE, FALSE	FALSE
ptree_show_tshell	<logical>	If TRUE a Thick Shell category will be included in the tree	TRUE, FALSE	FALSE

ptree_show_vector	<logical>	If TRUE a Define vector category will be included in the tree	TRUE, FALSE	FALSE

The following control permissions when writing files

Preference	Type	Description	Valid arguments	Default
directory_permission	<integer>	Octal permission code for new directories	0 - 777	744

The following control settings when renumbering entities

Preference	Type	Description	Valid arguments	Default
database_norenumber	<string>	If set to ON, DATABASE_HISTORY cards will not be renumbered	OFF, ON	OFF
range_norenumber	<string>	If set to ON, labels in the specified range will not be renumbered	OFF, ON	OFF
norenumber_min	<integer>	Min label that will not be renumbered		1
norenumber_max	<integer>	Max label that will not be renumbered		1
edit_panel_ numbering_rule	<string>	default rule for numbering of created entities in edit panels	HIGHEST_PLUS_ONE, LABEL_FIRST_FREE, LABEL_FIRST_LATENT, LAYER_FIRST_FREE, LAYER_HIGHEST_FREE, LAYER_HIGHEST_PLUS_ONE	LAYER_HIGHEST_ PLUS_ONE

The following affect scripting

Preference	Type	Description	Valid arguments	Default
script_directory	<string>	Directory for PRIMER to look for scripts in		$OA_INSTALL/primer_library/scripts
javascript_memory_size	<integer>	Maximum memory allocated for garbage collection		10

The following affect how *SET_GENERATE are treated in PRIMER

Preference	Type	Description	Valid arguments	Default
check_set_generate_on_renumber	<string>	check SET_GENERATE content on renumber selected	ON, OFF	ON
highest_label_considers_set_generate_entries	<string>	Consider SET_GENERATE entries when determining highest label or not?	TRUE, FALSE, PROMPT	PROMPT

Keys can have functions assigned to them:

Preference	Type	Description	Valid arguments	Default
F1_key	<string>	Shortcut for F1		<none>
F2_key	<string>	Shortcut for F2		<none>
F3_key	<string>	Shortcut for F3		<none>
F4_key	<string>	Shortcut for F4		<none>
F5_key	<string>	Shortcut for F5		<none>
F6_key	<string>	Shortcut for F6		<none>
F7_key	<string>	Shortcut for F7		<none>
F8_key	<string>	Shortcut for F8		<none>

F9_key	<string>	Shortcut for F9		<none>
F10_key	<string>	Shortcut for F10		<none>
F11_key	<string>	Shortcut for F11		<none>
F12_key	<string>	Shortcut for F12		<none>
A_key	<string>	Shortcut for A		AUTOSCALE
B_key	<string>	Shortcut for B		BLANK
C_key	<string>	Shortcut for C		CLOSE_ALL
D_key	<string>	Shortcut for D		DRAG_CUT
E_key	<string>	Shortcut for E		ENTITIES
F_key	<string>	Shortcut for F		FRINGE
G_key	<string>	Shortcut for G		<none>
H_key	<string>	Shortcut for H		HIDDEN
I_key	<string>	Shortcut for I		ICONISE
J_key	<string>	Shortcut for J		ATTACHED
K_key	<string>	Shortcut for K		RESET_VIS
L_key	<string>	Shortcut for L		LINE
M_key	<string>	Shortcut for M		MEASURE
N_key	<string>	Shortcut for N		CUT_PLANE
O_key	<string>	Shortcut for O		DISPLAY
P_key	<string>	Shortcut for P		TOGGLE_ALL_PP
Q_key	<string>	Shortcut for Q		QUICK_PICK
R_key	<string>	Shortcut for R		REVERSE
S_key	<string>	Shortcut for S		SHADED
T_key	<string>	Shortcut for T		TIDY_MENUS
U_key	<string>	Shortcut for U		UNBLANK
V_key	<string>	Shortcut for V		VIEW_MENU
W_key	<string>	Shortcut for W		IMAGE_MENU
X_key	<string>	Shortcut for X		CUT_SECTION
Y_key	<string>	Shortcut for Y		CYCLE_OVERLAY
Z_key	<string>	Shortcut for Z		ZOOM
a_key	<string>	Shortcut for a		AUTOSCALE
b_key	<string>	Shortcut for b		BLANK

c_key	<string>	Shortcut for c		CLOSE_ALL
d_key	<string>	Shortcut for d		DRAG_CUT
e_key	<string>	Shortcut for e		ENTITIES
f_key	<string>	Shortcut for f		FRINGE
g_key	<string>	Shortcut for g		<none>
h_key	<string>	Shortcut for h		HIDDEN
i_key	<string>	Shortcut for i		ICONISE
j_key	<string>	Shortcut for j		ATTACHED
k_key	<string>	Shortcut for k		RESET_VIS
l_key	<string>	Shortcut for l		LINE
m_key	<string>	Shortcut for m		MEASURE
n_key	<string>	Shortcut for n		CUT_PLANE
o_key	<string>	Shortcut for o		DISPLAY
p_key	<string>	Shortcut for p		TOGGLE_CURR_PP
q_key	<string>	Shortcut for q		QUICK_PICK
r_key	<string>	Shortcut for r		REVERSE
s_key	<string>	Shortcut for s		SHADED
t_key	<string>	Shortcut for t		TIDY_MENUS
u_key	<string>	Shortcut for u		UNBLANK
v_key	<string>	Shortcut for v		VIEW_MENU
w_key	<string>	Shortcut for w		IMAGE_MENU
x_key	<string>	Shortcut for x		CUT_SECTION
y_key	<string>	Shortcut for y		CYCLE_OVERLAY
z_key	<string>	Shortcut for z		ZOOM
SPACE_key	<string>	Shortcut for space		<none>
ONE_key	<string>	Shortcut for 1		VIEW_P_XY
TWO_key	<string>	Shortcut for 2		VIEW_P_YZ
THREE_key	<string>	Shortcut for 3		VIEW_P_XZ
FOUR_key	<string>	Shortcut for 4		VIEW_P_ISO
FIVE_key	<string>	Shortcut for 5		VIEW_N_XY
SIX_key	<string>	Shortcut for 6		VIEW_N_YZ
SEVEN_key	<string>	Shortcut for 7		VIEW_N_XZ

EIGHT_key	<string>	Shortcut for 8		VIEW_N_ISO
NINE_key	<string>	Shortcut for 9		<none>
ZERO_key	<string>	Shortcut for 0		<none>
EXCLAMATION_key	<string>	Shortcut for !		<none>
DOUBLEQUOTE_key	<string>	Shortcut for "		<none>
HASH_key	<string>	Shortcut for #		<none>
DOLLAR_key	<string>	Shortcut for $		<none>
PERCENT_key	<string>	Shortcut for %		<none>
AMPERSAND_key	<string>	Shortcut for &		<none>
SINGLEQUOTE_key	<string>	Shortcut for ’		<none>
LEFTBRACKET_key	<string>	Shortcut for (		<none>
RIGHTBRACKET_key	<string>	Shortcut for)		<none>
ASTERISK_key	<string>	Shortcut for *		<none>
PLUS_key	<string>	Shortcut for +		ZOOM_IN
COMMA_key	<string>	Shortcut for ,		<none>
MINUS_key	<string>	Shortcut for -		ZOOM_OUT
DOT_key	<string>	Shortcut for .		<none>
SLASH_key	<string>	Shortcut for /		SHORTCUT
COLON_key	<string>	Shortcut for :		<none>
SEMICOLON_key	<string>	Shortcut for ;		<none>
LESSTHAN_key	<string>	Shortcut for <		<none>
EQUALS_key	<string>	Shortcut for =		ZOOM_IN
GREATERTHAN_key	<string>	Shortcut for >		<none>
QUESTIONMARK_key	<string>	Shortcut for ?		SHORTCUT
AT_key	<string>	Shortcut for @		<none>
LEFTSQUAREBRACKET_key	<string>	Shortcut for [		<none>
BACKSLASH_key	<string>	Shortcut for \		<none>
RIGHTSQUAREBRACKET_key	<string>	Shortcut for]		<none>
CIRCUMFLEX_key	<string>	Shortcut for ^		<none>
UNDERSCORE_key	<string>	Shortcut for _		ZOOM_OUT
BACKTICK_key	<string>	Shortcut for ‘		<none>
LEFTCURLYBRACKET_key	<string>	Shortcut for {		<none>

PIPE_key	<string>	Shortcut for |		<none>
RIGHTCURLYBRACKET_key	<string>	Shortcut for }		<none>
TILDE_key	<string>	Shortcut for ~		<none>

The following settings control the interactive text editor, and the contents of files.

Preference	**Type**	**Description**	**Valid arguments**	**Default**
text_editor	<string>	Text editor to use for editing comments		<none>
text_edit_show_names	<logical>	Whether to show field header names in file to be edited	TRUE, FALSE	TRUE
text_edit_show_rules	<logical>	Whether to show rules about editing files as comments	TRUE, FALSE	TRUE

The following parameters can be used to preset **TRANSFER_DATA** operations

Preference	**Type**	**Description**	**Valid arguments**	**Default**
transfer_source_file	<string>	Source model filename		<none>
transfer_data_type	<list>	One or more datatypes to be matched from the list	MAT, SECTION, EOS, HOURGLASS, TMAT	<none>
transfer_match_by	<string>	The method for matching items between source and target models	ID, NAME, BOTH, ALL	NAME
transfer_action	<string>	Where in the target model to copy transferred data	CS, CO, CM, RO	CS
transfer_name_match	<string>	The name matching method used	T_IN_S, S_IN_T, EITHER, EXACT	EITHER
transfer_superseded	<string>	What happens to superseded data	SAVE, DELETE	SAVE
transfer_missing	<string>	Transfer missing items into target model	TRUE, FALSE	FALSE
transfer_populate	<string>	items for populate during transfer	DISCRETE, JOINT, NONE	NONE

The following control undo functionality

Preference	**Type**	**Description**	**Valid arguments**	**Default**
undo_enabled	<logical>	Undo enabled	TRUE, FALSE	TRUE
undo_max_percent_memory	<integer>	Maximum percentage of memory to use for storing undo info	0 - 100	5

The following control treatment of unicode

Preference	**Type**	**Description**	**Valid arguments**	**Default**
cjk_unix_font	<string>	Font to use for CJK text on unix machines		-misc-fixed-medium-r-normal-*-12-*-*-*-*-*-*-*
cjk_windows_font	<string>	Font to use for CJK text on windows machines		MS Gothic 12
file_encoding	<string>	Character encoding for script files	Latin-1, BIG5, EUC-CN, EUC-JP, EUC-KR, GB, GBK, ISO-2022-CN, ISO-2022-CN-EXT, ISO-2022-JP, ISO-2022-JP-2, ISO-2022-KR, JOHAB, Shift-JIS, UTF-8, UTF-16BE, UTF-16LE, UTF-16, UTF-32BE, UTF-32LE, UTF-32	Latin-1

The following control element visibility

Preference	Type	Description	Valid arguments	Default
NODE_drawn	<string>	Nodes drawn	ON, OFF	OFF
SOLID_drawn	<string>	Solids drawn	ON, OFF	ON
BEAM_drawn	<string>	Beams drawn	ON, OFF	ON
SHELL_drawn	<string>	Shells drawn	ON, OFF	ON
TSHELL_drawn	<string>	Thick shells drawn	ON, OFF	ON
DISCRETE_drawn	<string>	Springs/dampers drawn	ON, OFF	ON
MASS_drawn	<string>	Lumped masses drawn	ON, OFF	OFF
segment_hatching	<string>	Contact and other segment hatching	OFF, ON	ON

Global preferences.

Global preferences that apply to all programs can be specified using **`"oasys"`** as the program name.

`oasys*<keyword>: <argument`

The following global preferences can be defined.

Preference	Type	Description	Valid arguments	Default
file_names	<string>	Controls input filename syntax. LSTC = d3*, OASYS = job.ptf*	OASYS, LSTC	OASYS
html_application	<string>	Location of HTML browser		<none>
image_format	<string>	Default image format	BMP_8_C, BMP_8_UN, PNG_8, GIF_8, BMP_24_UN, PNG_24, JPG_24, PPM_24	JPG_24
maximise	<logical>	Maximise window when Program is started	TRUE, FALSE	FALSE
placement	<string>	Location for initial window on multi-screen display	LEFT, RIGHT, BOTTOM, TOP, LEFT_BOTTOM, LEFT_TOP, RIGHT_BOTTOM, RIGHT_TOP	<none>
pdf_application	<string>	Location of PDF browser		<none>
start_in	<string>	Directory to start Program in		<none>

The following control directories

Preference	Type	Description	Valid arguments	Default
home_dir	<string>	"home" directory for user		<none>
install_dir	<string>	Directory Oasys Ltd software is installed in		<none>
manuals_dir	<string>	Directory user manuals are installed in		<none>
temp_dir	<string>	temporary directory for user		<none>
checkpoint_dir	<string>	Directory for checkpoint files, or "none" to suppress them altogether		<none>

The following control laser options

Preference	Type	Description	Valid arguments	Default
laser_paper_size	<string>	Default paper size	US, A4	A4
laser_orientation	<string>	Default page orientation	Portrait, Landscape	Landscape
laser_top_margin	<real>	Top margin size in mm		10
laser_bottom_margin	<real>	Bottom margin size in mm		30
laser_left_margin	<real>	Left margin size in mm		20
laser_right_margin	<real>	Right margin size in mm		10

The following control menu and mouse attributes

Preference	Type	Description	Valid arguments	Default
display_factor	<real>	Factor on display size (0.5-2.0)	0.5 - 2.0	1.0
display_brightness	<real>	Menu brightness (0.0-1.0)	0.0 - 1.0	1.0

display_saturation	<real>	Menu colour saturation (0.0-1.0)	0.0 - 1.0	1.0
dv_left_shift	<string>	Dyn view action for shift + Left mouse	ROTATION_XYZ, ROTATION_XY, ROTATION_Z, ROTATION_SPHERE, TRANSLATION, ZOOM_UP_+VE, ZOOM_DOWN_+VE, UNUSED	ROTATION_XYZ
dv_middle_shift	<string>	Dyn view action for shift + Middle mouse	ROTATION_XYZ, ROTATION_XY, ROTATION_Z, ROTATION_SPHERE, TRANSLATION, ZOOM_UP_+VE, ZOOM_DOWN_+VE, UNUSED	TRANSLATION
dv_right_shift	<string>	Dyn view action for shift + Right mouse	ROTATION_XYZ, ROTATION_XY, ROTATION_Z, ROTATION_SPHERE, TRANSLATION, ZOOM_UP_+VE, ZOOM_DOWN_+VE, UNUSED	ZOOM_UP_+VE
dv_left_ctrl	<string>	Dyn view action for ctrl + Left mouse	ROTATION_XYZ, ROTATION_XY, ROTATION_Z, ROTATION_SPHERE, TRANSLATION, ZOOM_UP_+VE, ZOOM_DOWN_+VE, UNUSED	ROTATION_XYZ
dv_middle_ctrl	<string>	Dyn view action for ctrl + Middle mouse	ROTATION_XYZ, ROTATION_XY, ROTATION_Z, ROTATION_SPHERE, TRANSLATION, ZOOM_UP_+VE, ZOOM_DOWN_+VE, UNUSED	TRANSLATION
dv_right_ctrl	<string>	Dyn view action for ctrl + Right mouse	ROTATION_XYZ, ROTATION_XY, ROTATION_Z, ROTATION_SPHERE, TRANSLATION, ZOOM_UP_+VE, ZOOM_DOWN_+VE, UNUSED	ZOOM_UP_+VE
dv_left_both	<string>	Dyn view action for shift+ctrl + Left mouse	ROTATION_XYZ, ROTATION_XY, ROTATION_Z, ROTATION_SPHERE, TRANSLATION, ZOOM_UP_+VE, ZOOM_DOWN_+VE, UNUSED	ROTATION_XYZ
dv_middle_both	<string>	Dyn view action for shift+ctrl + Middle mouse	ROTATION_XYZ, ROTATION_XY, ROTATION_Z, ROTATION_SPHERE, TRANSLATION, ZOOM_UP_+VE, ZOOM_DOWN_+VE, UNUSED	TRANSLATION
dv_right_both	<string>	Dyn view action for shift+ctrl + Right mouse	ROTATION_XYZ, ROTATION_XY, ROTATION_Z, ROTATION_SPHERE, TRANSLATION, ZOOM_UP_+VE, ZOOM_DOWN_+VE, UNUSED	ZOOM_UP_+VE
dv_shift_action	<string>	Dynamic viewing mode for shift + mouse button	CURRENT, WIREFRAME, FREE_EDGE, UNUSED	CURRENT
dv_ctrl_action	<string>	Dynamic viewing mode for ctrl + mouse button	CURRENT, WIREFRAME, FREE_EDGE, UNUSED	WIREFRAME
dv_both_action	<string>	Dynamic viewing mode for shift+ctrl + mouse button	CURRENT, WIREFRAME, FREE_EDGE, UNUSED	FREE_EDGE
font_size	<string>	Menu font size	SMALL, DEFAULT, LARGE	DEFAULT
font_type	<string>	Menu font typeface and strength	HELVETICA, HELVETICA-BOLD, TIMES, TIMES-BOLD, COURIER, COURIER-BOLD	HELVETICA
left_handed	<string>	Left handed switching of mouse and/or keyboard	NONE, MOUSE, KEYBOARD, ALL	NONE
zoom_factor	<real>	Zoom Factor for mouse wheel (0.01-0.2)	0.01 - 0.2	0.05
czoom_factor	<real>	Factor for right mouse dynamic zoom (0.01-0.2)	0.01 - 0.2	0.05
kzoom_factor	<real>	Factor for +/- keyboard short-cut keys	0.01 - 100.0	2.0

.

Command-Line arguments to Primer.

"Command-line" arguments are added to the execution line itself as extra arguments to the code. Where they conflict with settings in the "oa_pref" file the command-line arguments take precedence.

Command-line argument syntax

- There should be no spaces between keyword, "=" and argument.

  For example "**`-d=opengl`**" is valid, but "**`-d = opengl`**" is not.

- Fixed arguments (such as **`opengl`**) are not case sensitive.

  Filenames are case-sensitive on Linux and Unix operating systems, but not on Windows.

Command-line arguments valid in Primer

Function	Format	Options
Setting the graphics device By default no graphics device is defined, and the device selection panel is mapped. These options can be especially useful if you want to bypass the device selection panel and always start Primer with a particular graphics driver.	**-d=<*device*>**	**-d=opengl** : Use OpenGL 3D graphics **-d=x24** : 24 bit-plane X-Windows graphics **-d=x8** : 8 bit-plane X-Windows graphics **-d=x** : **X24** if available, otherwise **X8** **-d=default** : Whichever is available in the order **OpenGL**, **X24**, **X8** **-d=batch** **-d=tty** : No graphics - text-only mode
Specifying "full screen" mode on startup Normally Primer occupies about 70% of the display when it starts, the "maximise" argument changes this to become the full screen.	**-maximise**	
Specifying window placement on a multi-display desktop By default the top right corner of the desktop is used. The most common arrangement is two screens side by side, for which "left" and "right" may be used. However "top" and "bottom" are also available for the case of two screens one above the other, and the options may be concatenated for a 2x2 display. These options can be combined with **-maximise** to fill the relevant screen. Users on Windows platforms where tools such as NVidia's "NView" are available may find that it is better to leave window placement to that tool, so that Primer's windows behave in a fashion consistent with other application windows.	**-placement=<*where*>**	This option is intended for use where the desktop is spread as a "Single Logical Screen" over multiple monitors. **<*where*>** values — **Meaning** **left** — Left hand monitor **right** — Right hand monitor **top** — Upper monitor **bottom** — Bottom monitor The above may be concatenated for a 2x2 display, for example **top_left** — Top left monitor **bottom_right** — Bottom right monitor
Defining a command file name By default no command file is assumed.	**-cf=<*filename*>**	<*filename*> can be any text file containing valid commands.

Defining a macro file name By default no macro file is assumed.	`-macro=<filename>`	<*filename*> can be any text file containing valid macro commands.
Requesting termination at the end of a command or macro file This is ignored if no command or macro file is defined	`-exit`	
Requesting batch creation of ZTF and group files This generates both <*filename*>.ztf and groups <*filename*>.bin files for subsequent post-processing in D3PLOT. When combined with "`-d=batch`" then: • ZTF and group (.bin) files are created, then Primer exits • No licence to run Primer is required	`-ztf=<filename>`	<*filename*> must be a valid LS-Dyna keyword (.key) file, with or without the ".key" extension.
Specifying the directory in which to start. PRIMER will make this your "current working directory", so that all files which do not have explicit pathname prefixes are assumd to be in this directory.	`-start_in=<directory>`	<*directory*> must be a valid directory name on your system.
Specifying a custom "oa_pref" file This causes an extra, optional "oa_pref" file to be read.	`-pref=<filename>`	<filename> must be a valid "oa_pref" file. If it has no path prefixed, the file is assumed to be in the OA_INSTALL directory. Any legal filename may be used.

Redirecting console output to a file
 	
This option is only available on windows. For unix use standard shell redirection instead.
	`-eo`
	`-eo=default`
	`-eo=<filename>`
		If <filename> is given then it is used as the filename to write the output to. In order to permit multiple sessions to coexist on the same machine the process id will be appended to the main part of the filename. For example if <filename> is "primer_output.log" then the actual filename will be "primer_output_<pid>.log.
		If no filename is given or the filename is "default" then filename generation is automatic, and the first valid of:
		%TEMP%\primer_log_<pid>.txt
		%TMP%\primer_log_<pid>.txt
		%HOMESHARE%\primer_log_<pid>.txt
		%USERPROFILE%\primer_log_<pid>.txt
		will be used.

Environment variables that affect Primer.

Environment variables are set at the both at the operating system and user levels, and can be used to influence the behaviour of Oasys Ltd products. Generally they are better suited to site-wide customisation in the Shell when the software is installed, but users are free to make their own local settings.

Unix/Linux systems running "C" shell (/bin/csh) or its derivatives such as /bin/tcsh:

The format of the command is:

setenv *<parameter> <argument list>*

For example:

```
setenv DISPLAY my_machine:0
setenv SM_USE_VISUAL default
setenv DISPLAY_FACTOR 1.2
```

(Note that the "**oasys_xx**" shell is written using C shell syntax, so if it is amended the format above should be used.)

Unix/Linux systems running "Bourne" (/bin/sh) or "Korn" (/bin/ksh) shells

The format of the command is:

<parameter>*=*<argument list>*; export *<parameter>

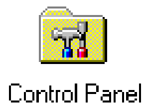
For example:

```
DISPLAY=my_machine:0; export DISPLAY
SM_USE_VISUAL=default; export SM_USE_VISUAL
DISPLAY_FACTOR=1.2; export DISPLAY_FACTOR
```

Windows systems

Choose the "**`System`**" panel

==>



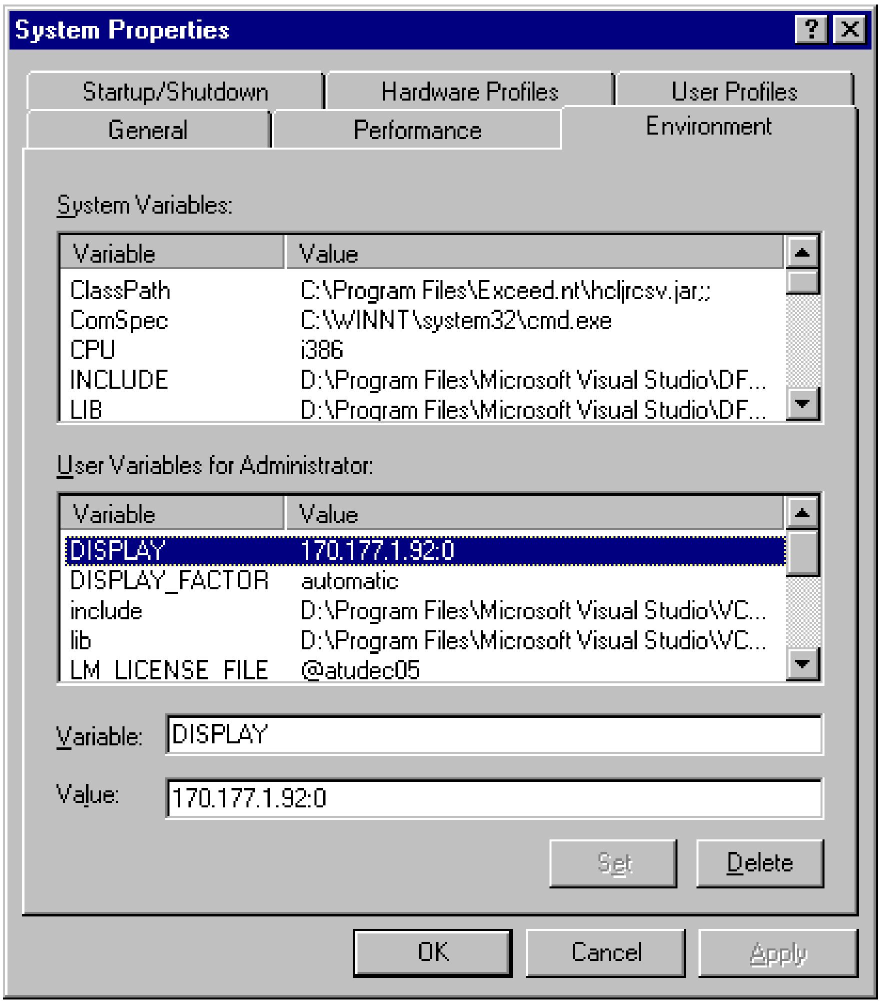
==>

On Windows NT:
On Windows 2000 / XP:

==> Properties ==> Advanced

Select the **`Environment`** tab.

(This example shows the panel from Windows NT4. That from Windows 2000 and XP is very similar.)



Then insert the relevant **`Variable`** and **`Value`** strings into the User or System settings as desired.

In this example it can be seen that iser Administrator has set the **`DISPLAY`** environment variable to **`170.177.1.92:0`**.

Environment variables that control the behaviour of PRIMER.

Variable name	Description	Possible Values	Default
The following variables control the graphics and attributes of the display window and menu system.			
DISPLAY	The X11 display id on which graphics will be drawn. If this is not defined (most systems initialise this to "**:0**") then no connection can be made to an X server, and no graphics will be drawn.	(<*machine name*>)**:<*server id*>**(<.*screen id*>)	**:0**
DISPLAY_SATURATION DISPLAY_BRIGHTNESS DISPLAY_FACTOR	Saturation controls the colour saturation (intensity) of menus Brightness controls the colour brightness of menus "Factor" sets the relative display scale, and can range from 0.5 (making menus larger) to 2.0 (making them smaller). It may also be set to "**automatic**" which derives a factor from the physical screen dimensions.	0.0 to 1.0 0.0 to 1.0 0.5 to 2.0, or **automatic**	**1.0** **1.0** **1.0**
SM_USE_VISUAL	Sets the X11 "visual" id to be used for screen menus. Where a graphics display provides "overlay" planes these should normally be used, otherwise this should be left undefined or set to "default". Using an explicit visual id is possible, and this should be defined in hexadecimal (eg 0xf16). Experience has shown the on some Silicon Graphics systems using the "overlay" planes can result in very strange colours in other windows, in which case "default" should be used. Also on some W2000 and graphics board combinations problems may also arise with overlay planes and, again, "default" should be used.	**overlay** **default** <*visual id*> in hex	**overlay**
SM_AUTO_CONFIRM	This variable is often used when replaying command files which, when recorded, paused and asked the user to confirm things. (For example HELP and Warning messages.) If the variable is set (true) then these will not pause and will behave as if the user had pressed "OK" - meaning that command files can play back without user intervention.	**true** or **false**	**false**
USE_PIXMAPS	Controls whether or not the menus use "pixmaps" (off-screen memory) to produce smooth scrolling. Turning this off (false) will save memory, and may help memory problems on a display that has only limited memory available for the X server, but will give slightly jerky window scrolling.	**true** or **false**	**true**

PRIMER_NO_PIXMAP **PRIMER_NO_PBUFFER**	Section 2.4.4.2 shows a new method that has superseded these variables. May be used to suppress backing store redraws for the OpenGL graphics window. Should be used on OpenGL / X graphics combinations only if you receive errors starting "GLX ...", and then only after consultation with Oasys Ltd.	**true** or **false**	**false**
SAVE_UNDER	This flag was introduced to fix a specific bug on Compaq Alpha OSF4.x operating systems. Normally the window manager requests a redraw of windows that have been updated, even when they are currently obscured by something else. However the OSF4 window manager series failed to do this, leading to "bare" patches underneath popup menus when these were unmapped. Setting this flag to **false** results in more redraws on these systems since it suppresses the default "save under" property of X11 windows, but it does at least prevent windows getting bare areas. Compaq have fixed the bug in OSF5, and possibly in later releases of OSF4.	**true** or **false**	**true**
CP_FILE_FILTER	Used during checkpoint file replay to override any file and pathname stored in the checkpoint file, bringing up the file filter instead. This allows checkpoint files to be replayed on different systems.	**true** or **false**	**false**
The following two variables apply on **Windows platforms only**, and should only be used if the menu system is clearly obtaining the wrong display size from the system, as evidenced by fonts and menus being very much the wrong size.			
DISPLAY_HEIGHT	Set an explicit display height in millimetres	<height in mm>	<none>
DISPLAY_WIDTH	Set an explicit display width in millimetres	<width in mm>	<none>
The following variables affect the functioning of the code:			

PRIMER_FILE_FORMAT	An alternative way of controlling the format of ASCII files written on Windows systems. Windows has the peculiarity that, by default, it writes both <carriage return> and <line feed> characters at the end of each line, whereas Unix and Linux platforms only write a <line feed>. The presence of the <carriage return> can confuse some software on Unix/Linux, and its absence can confuse some software on Windows, so in a mixed machine environment there is - unfortunately - no single answer that is "best". PRIMER offers the following options: **native**: Uses the default for the machine's architecture, which adds <carriage return> on Windows. **unix**: Suppresses the <carriage return> (makes no difference on Unix/Linux machines.) This setting has the same effect as the **ascii_file_format** preference above, and is provided for users who wish to set file output format on a "per machine" basis rather than globally with the oa_pref file. If the oa_pref option is used it will supersede this setting.	**native** or **unix**	**native**
The following controls the display of on-line manual pages on Unix systems only. (Windows systems use the default web browser.)			
NETSTART	Command string to start Netscape on Unix/Linux hosts. This is used to fire up the Netscape browser in order to read manual pages from within PRIMER.	Any valid Unix command string.	<*none*>
The following variables are provided for debugging purposes only, and should not normally be used.			
DB_POINTER_CHECK	Runs a check during every internal database allocation and return operation to scan for duplicated or erroneous pointers. This will result in very much (potentially 100x) slower operation of internal memory management, and is normally only used to track down internal errors.	**false**, **1** or **2** (Turned off, level #1 or level #2 checking)	**false**
XSYNC	Runs the X server in "synchronised" (unbuffered) mode. This will give woefully slow graphics, and is used for debugging purposes only.	**true** or **false**	**false**
WARN_REDEFINE	Makes the menu system issue a warning if a button is redefined. Again this is normally only used for debugging purposes.	**true** or **false**	**false**

APPENDIX XIV: Automated model build from command line

Primer offers the ability to build multiple models from a csv input file.

The command line syntax is simply: **BUILD READ filename.csv**. This may be typed onto the command line of the Primer dialogue box or into a file which Primer reads in batch mode.

Each csv file contains a header line which determines the type of build and consequent format.

Currently the following types are supported:

DATABASE uses the database/template method as used by MODEL->BUILD function
IHI interior head impact
PEDHEAD pedestrian head impact
PEDHEAD_ANGLE pedestrian head impact with approach angle
PEDLEG_LOWER pedestrian lower leg impact
PEDLEG_UPPER pedestrian upper leg impact

And the following general formats:
GENERAL_TRANSLATE
GENERAL_TRANSLATE_ROTATE
GENERAL_TRANSLATE_VECTOR
GENERAL_TRANSLATE_TRIAD

Here are examples of each type, with $comments included.

DATABASE

DATABASE
Database, vehicle.dba
$
$ following 2 lines are optional
$ tag to specify root directory for output files
Rootdir, /data/DEMO
$ tag for reporter summary template
Reporter_summary, /data/reporter/summary.opt
$
$ the following loadcases consist of
$ directory to be created for output, template mask, output filename, optional reporter individual template
ODB, odb.tpl, odb_model.key, reporter_odb.opt
SIDE, side.tpl, side_impact.key, reporter_side.opt
Etc…

Primer will build the models for each load case and write dyna keyword master files to /<rootdir>/ODB/odb_model.key, /<rootdir>/SIDE/side_impact.key, etc. Component files will be referenced by *INCLUDE or *INCLUDE_TRANSFORM (see below)

Also in /data/DEMO a job submission file (.lst) will be written, which can be read by the submission shell. The shell will submit each job to a node on the processing cluster, where on the completion of each the individual reporter template will be run. When the last job is finished the summary template will be run.

Note on automatic positioning using database/template method.

See also orienting include files for model database.

As some component files (e.g. barriers) are unlikely to be in exactly the correct position, Primer provides an automated method for positioning them on a per build basis.

This positioning is achieved by matching named slave and master orient points which are contained within the database and templates. Typically a master point is kept in a template (associated with a loadcase) but may also reside in the database. A slave point must reside in the database as it is associated with a component file to be oriented.

Suppose, for example, the template odb.tpl contains a master orient point named "od_barrier" with an associated node id (master origin node) and a part name/id (e.g. "bumper skin") to be used for contact purposes. Similarly in the database, on the entry where the odb barrier is kept, a slave orient point of the same name exists, which refers to two nodes (origin and O-X nodes) and a part name/id (e.g. "barrier null shells").

On completion of the build Primer will associate the matching master and slave points. The odb barrier will be translated such that the slave origin node is moved to the position of the master node. A temporary contact will then be created between the master part(s) and the slave part(s) and the odb barrier will be translated into position using the slave vector (defined by the slave origin and O-X nodes). If the initial position gives contact penetration the barrier will be translated against the direction of the vector until just out of contact, otherwise it will be moved along the vector until just about to contact.

Interior Head Impact

IHI
Model, /data/DEMO/interior.key
Impactor, fmh.key
$
$ following lines up to loadcase are optional
$ tag to activate depenetration function
$ method = contact, contact name/id, dof <x, xz or xyz>
Depenetrate, contact, head to trim, XZ
$
$ tag to activate auto-vertical angle positioning
$ chin shell set name/id, dof <x, xz or xyz>
Vertical, 5000, XZ
$
$ tag to set root directory for output files
Rootdir, /data/DEMO
$ tag to set root name for output files
Rootname, ihi_test
$ tags for reporter templates
Reporter, individual.opt
Reporter_summary, summary.opt
$
$ loadcase lines consist of
$ directory, x, y, z target coords, horizontal angle, vertical angle, velocity for each
AP1, 1897.38, 602.244, 1205.9, 140, 40, 5364.0
AP3, 1679.84, 661.876, 1087.14, 120, 30, 5364.0
BP1, 2495.27, 578.925, 1237.79, 90, 23, 6705.0
$ for automatic vertical angle positioning, the loadcase line contains two more entries,
$ AUTO to tell PRIMER this is the automatic case, and the angle to rotate back by
$ after the chin has touched the trim. In this case, the vertical angle is the
$ maximum vertical angle the headform can rotate to.
BP2, 1997.65, 567.356, 13455.8, 160, 50, 5364.0, AUTO, 10.0

It is also possible to define a positional node for each loadcase line. This headform node is used to position the headform. In the absence of a positional node, Primer will just use the headform default node. The positional node is always defined after the velocity on the loadcase line, so the following lines are all valid:

BP2, 1997.65, 567.356, 13455.8, 160, 50, 5364.0, 1000000 - (positional node defined, no automatic vertical angle positioning)
BP2, 1997.65, 567.356, 13455.8, 160, 50, 5364.0, AUTO, 10.0 - (positional node not defined, but the automatic vertical angle positioning has been activated)
BP2, 1997.65, 567.356, 13455.8, 160, 50, 5364.0, 1000000, AUTO, 10.0 - (Both positional node and automatic vertical angle positioning have been activated)

In the case of the Oasys Ltd free motion headform the base coordinates of the impactor are taken from the head coordinate system and so do not need to be defined here.

For each loadcase the horizontal and vertical angles of the impactor are set by rotating about the base node, then the head is moved such that the base node is on the target point.

If the depenetrate option is active and the contact is valid, i.e. the latent ids of the trim parts have been added to the slave side, Primer will automatically depenetrate the head. The motion during depenetration is controlled by the dof setting:
Dof = X, head is locked onto the local X axis and depenetrated without changing local y or z coordinates
Dof = XZ, (recommended) head is locked in the local XZ plane and depenetrated without changing the local y coordinate
Dof = XYZ, head is moves freely during depenetration
The advantage of using the xz or xyz methods is that the additional freedom will allow the initial impact point to be achieved more closely to the target point.

For more information on the auto vertical angle positioning, see section 6.13.3. The CSV model build method works in the same way as the method within PRIMER’s FMH positioning panel.

On completion of the build, a master dyna keyword will be output to directories, /<rootdir>/AP1/ihi_test.key, etc. in which the main model is referenced by *INCLUDE and the impactor by *INCLUDE_TRANSFORM.

Additionally, in <rootdir> a submission (.lst) file will be written which may be read by the Shell.

By default reporter variables XCOORD, YCOORD,ZCOORD,H_ANG,V_ANG,VELOCITY are passed into the file. These may be accessed by the individual reporter templates.

Automated positioning methods use Include_Transform. For any method which involves rotational transforms of arbitrary angles, the use of *DEFINE_BOX inside the include file is to be discouraged. LS-Dyna rotates boxes by rotating the two vertices, consequently the process may change the shape and volume of the box adversely.

Pedestrian Head Impact

```
PEDHEAD
model, /data/DEMO/PEDESTRIAN_HEAD/biw.key
impactor, child_head.key
$define 3 coordinates on impactor either by method=define or method=nodes
$orient, define, <name/id of csys>
$orient, nodes, <name/id>, <name/id>, <name/id>
orient, nodes, base node, x node, y node
$
$tag to activate depenetration
$ method = contact, contact name/id, dof <x, xz or xyz>
$ method = partset, partset name/id, dof
depenetrate, contact, head to bonnet contact, XZ
$optional offset for deployable bonnet, applied along line of flight
offset, <distance>
$ tag for root directory for output
rootdir, /data/DEMO/NCAP_RUN_2
$ tag for root name for output files
rootname, childhead
$
reporter, individual.opt
reporter_summary, summary.opt
$
$ loadcase lines consist of
$ directory name, zone name(as dir if blank), X coord, Y coord, (optional Z coord)
C1A, C1A, 899.98401, 1393.1749
C1A_2, C1A, 889.98401, 1393.1749
C1B, , 841.03717, 1276.2445
C2A, , 804.94501, 1171.8967
Etc.
```

The pedestrian head model, unlike the free motion headform, does not carry a specified base coordinate system. Thus the "orient" line is necessary to define

- the base coordinate (at the centre of the head)
- the X coordinate (to give the line of flight pointing in the correct direction)
- the Y coordinate (which defines the normal to the XZ plane for the impactor).

The three points may be defined by the name or id of a DEFINE_COORDINATE_SYSTEM or by defining three nodes.

If these are DATABASE_HISTORY_NODE_ID, they may be defined by name as alternative to label, which has the advantage that they will not be affected by renumbering.

Depenetration is activated either by referencing a surface-surface contact between the head skin and the bonnet or a single part set comprising both. In the latter case, Primer will generate a contact between impactor and target parts.

Finally, the loadcase lines consist of a unique directory name for output key file, the zone name (which if blank, will default to same as directory name), and an X coordinate and Y coordinate. If the Z coordinate is not defined, as is usually the case, it will be calculated by projection determining the target geometry from the depenetrate information. For this reason, if depenetrate is not active, the Z coordinate must be defined explicitly.

Primer will generate a master keyword file in the output directory, which will reference the main model with *INCLUDE and the impactor with *INCLUDE_TRANSFORM.

In the root directory a submission file (.lst) will be written which may be used by the submission shell.

By default reporter variables ZONE, XCOORD, YCOORD and ZCOORD are passed into the file. These may be accessed by the individual reporter templates.

Note: Optional fields such as zone name may be specified by including a **space** between commas (,<space>,). Omitting the space might result in erroneous processing.

Pedestrian Head Impact with Approach Angle

PEDHEAD_ANGLE
model, /data/DEMO/PEDESTRIAN_HEAD/biw.key
impactor, child_head.key
$define 3 coordinates on impactor either by method=define or method=nodes
$orient, define, <name/id of csys>
$orient, nodes, <name/id>, <name/id>, <name/id>
orient, nodes, base node, x node, y node
$
$tag to activate depenetration
$ method = contact, contact name/id, dof <x, xz or xyz>
$ method = partset, partset name/id, dof
depenetrate, contact, head to bonnet contact, XZ
$
$ tag for root directory for output
rootdir, /data/DEMO/NCAP_RUN_2
$ tag for root name for output files
rootname, childhead
$
reporter, individual.opt
reporter_summary, summary.opt
$
$ loadcase lines consist of
$ directory name, zone name(as dir if blank), X coord, Y coord, angle of flight
C1A, C1A, 899.98401, 1393.1749, 30
C1A_2, C1A, 889.98401, 1393.1749, 40
C1B, , 841.03717, 1276.2445, 50.5
C2A, , 804.94501, 1171.8967, 60
Etc.

This is identical to the Pedestrian Head Impact model with the addition that an angle of rotation can be specified for each loadcase line. This angle will be applied to rotate the line of flight in the XZ plane of the impactor. This angle is specified as the fifth field. The Z coordinate, in this case, is not defined.

Automated positioning methods use Include_Transform. For any method which involves rotational transforms of arbitrary angles, the use of *DEFINE_BOX inside the include file is to be discouraged. LS-Dyna rotates boxes by rotating the two vertices, consequently the process may change the shape and volume of the box adversely.

Pedestrian Lower Leg Impact

PEDLEG_LOWER
model, /data/DEMO/PEDESTRIAN_LEG_LOWER/biw.key
impactor, child_head.key
$define 3 coordinates on impactor either by method=define or method=nodes
$orient, define, <name/id of csys>
$orient, nodes, <name/id>, <name/id>, <name/id>
orient, nodes, base node, x node, y node
$
$ following lines up to loadcase are optional
$tag to activate depenetration
$ method = contact, contact name/id, dof <x, xz or xyz>
$ method = partset, partset name/id, dof
depenetrate, contact, head to bonnet contact, XZ
$
$ z coordinate
z, 200.0
$
$ tag for root directory for output
rootdir, /data/DEMO/NCAP_RUN_2
$ tag for root name for output files
rootname, childhead
$
reporter, individual.opt
reporter_summary, summary.opt
$
$ loadcase lines consist of

$ directory name, zone name(as dir if blank), X coord, Y coord
C1A, C1A, 899.98401, 1393.1749
C1A_2, , 889.98401, 1393.1749
C1B, , 841.03717, 1276.2445
C2A, C2A, 804.94501, 1171.8967
Etc.

The pedestrian lower leg impact model involves an impactor translation without rotation. The "Orient" line is used to specify base, X, and Y coordinates.

"Depenetration" is specified as in the Pedestrian Head Impact case.

A constant Z can be specified here. The impactor will not be moved in the Z direction.

Reporter templates, root directory, and name may also be specified.

Finally, the loadcase lines consist of a unique directory name, an X coordinate, and a Y coordinate.

Pedestrian Upper Leg Impact

PEDLEG_UPPER
model, /data/DEMO/PEDESTRIAN_LEG_UPPER/biw.key
impactor, child_head.key
$define 3 coordinates on impactor either by method=define or method=nodes
$orient, define, <name/id of csys>
$orient, nodes, <name/id>, <name/id>, <name/id>
orient, nodes, base node, x node, y node
$
$tag to activate depenetration
$ method = contact, contact name/id, dof <x, xz or xyz>
$ method = partset, partset name/id, dof
depenetrate, contact, head to bonnet contact, XZ
$
$ tag for root directory for output
rootdir, /data/DEMO/NCAP_RUN_2
$ tag for root name for output files
rootname, childhead
$
reporter, individual.opt
reporter_summary, summary.opt
$
$ loadcase lines consist of
$ directory name, zone name(as dir if blank), X coord, Y coord, Angle, Velocity, optional Mass value, optional Mass parameter
C1A, , 899.98401, 1393.1749, -40, 5400.0, 5.1, andy
C1A_2, ,889.98401, 1393.1749, -50.5, 5800.5
C1B, C1B, 841.03717, 1276.2445, -45.0, 4200.3, 5,2, bob
C2A, C2A, 804.94501, 1171.8967, -30.5, 4800.9, 5,3, fred
Etc.

Impact angle and velocity can be specified for each loadcase. Additionally the added mass value and an associated parameter can optionally be specified.

The Z-coordinate will be calculated by projection determining the target geometry from the depenetrate information as in the Pedestrian Head impact case.

Automated positioning methods use Include_Transform. For any method which involves rotational transforms of arbitrary angles, the use of *DEFINE_BOX inside the include file is to be discouraged. LS-Dyna rotates boxes by rotating the two vertices, consequently the process may change the shape and volume of the box adversely.

General Translate

GENERAL_TR
model, interior.key
impactor, impactor.key
$orient tag uses method=nodes or method=define
orient, nodes, 4675, 4685, 4679
$

$ following lines up to loadcase are optional
$ depenetrate tag, method = contact or method = partset
depenetrate, partset, 10, XZ
$
$ tag to set root directory for output files
Rootdir, /data/DEMO
$ tag to set root name for output files
Rootname, ihi_test
$ tags for reporter templates
reporter, individual.opt
reporter_summary, summary.opt
$
$ loadcase lines consist of
$ directory, X, Y, Z target cords, optional velocity
RUN1, 1897.38, 602.244, 1205.9, -2000.0
Etc.

If the impactor is to be translated without rotation, this format is applicable. The impactor is moved such that the base coordinate (node 4675) lies on the the target point and then is depenetrated appropriately.

General Translate Rotate

GENERAL_TR_ROTATE
Model, interior.key
Impactor, impactor.key
.
. as above
.
$ loadcase lines consist of output directory, X, Y, Z target coords, angle about X, Y, Z, optional velocity
RUN1, 1897.38, 602.244, 1205.9, 0, -40, 140, 5340.
RUN2, 1679.84, 661.876, 1087.1, 0, -30, 120, 5340.

If the impactor is to be rotated into position as well as translated this format is applicable. For each loadcase, three global rotations are defined. These will be applied to the impactor before it is translated. The centre of rotation is the impactor base coordinate.

General Translate vector to vector

GENERAL_TR_VECT
Model, interior.key
Impactor, impactor.key
.
. as above
.
$ loadcase lines consist of output directory, X, Y, Z, X’, Y’, Z’ coords, optional velocity
RUN1, 1897.38, 602.244, 1205.9, 1900.0, 620.0, 1210.0
RUN2, 1679.84, 661.876, 1087.1, 1700.0, 670.0, 1090.0
Etc.

For the vector to vector transformation format, a vector PP’ is defined for each loadcase. The impactor is rotated so that the line of flight vector coincides with the PP’ vector. The axis of rotation is the normal to the two vectors.

General Translate triad to triad

GENERAL_TR_TRIAD
Model, interior.key
Impactor, impactor.key
.
. as above
.
$ loadcase lines consist of output directory, X, Y, Z, X’, Y’, Z’, X", Y", Z" coords, optional velocity
RUN1, 1897.38, 602.244, 1205.9, 1900.0, 620.0, 1210.0, 1910.0, 620.0, 1210.0
RUN2, 1679.84, 661.876, 1087.1, 1700.0, 670.0, 1090.0, 1710.0, 680.0, 1090.0
Etc.

For the triad to triad transformation, a triad PP’P" is defined for each loadcase. The impactor is rotated such that its base triad (as defined by the orient tag) aligns with the target triad.

APPENDIX XV: Finding model mass properties

include file mass, C of G, Inertia

You can obtain include file mass properties in the following ways:

(1) Using **INCLUDE MASS** off the include tree popup

(2) Writing summary file (**MODEL**->**UTILITIES**->**Write Summary file**) with **Write mass for includes** active

(3) Writing out model with **write mass to each include file** set in the Pre-Output check panel (or as an oa_pref setting).

(4) Using **report include mass** from part tree icon

(5) putting all parts of include on Part table and looking at part mass, C of G and Inertia on top row

Methods 1, 2, 3 are all consistent in their handling of Rigid Body merges, giving:

(A) a sum of part masses which only includes effect of merges where both parts are in the include and

(B) a sum of part masses with consideration of all merges.

Method 4 just gives the answer using the “correct” method (B). This also gives C of G and Inertia tensor for parts in include.

Model mass, C of G, Inertia

You can get model mass, C of G, and Inertia:

(1) Using **report model mass** from part tree icon, this will also give timestep added mass adjusted values

(2) writing summary file (**MODEL**->**UTILITIES**->**Write Summary file**), the data also appears in dialogue box

(3) writing out model with **write model mass** option active, the data also appears in dialogue box

(4) putting all parts on part table and looking at part mass, C of G and Inertia totals on the top row

APPENDIX XVI: XML format for model build

Database file

Element tag	Attributes	Comment
<PRIMER_DATABASE	category = 'database name'	opening tag
	protected = 'no'	

- **Version data**

<DEFAULT_VERSION	id = '<id>'	if not specified highest available version if default
<VERSION	descr = 'version name'	
	id = '<id>'	<id> is integer version id

- **Component data**

<H1	category = 'title'	
	subcategory = 'subtitle'	
	thumbnail = 'filename'	optional
	owner = 'name of owner'	optional
<COMPONENT	file = 'filename'	component file with absolute or relative path
	version = '<id>'	version of this component
<EXTRA_DATA	file = 'filename'	file for contact, etc which apply across components
<RENUMBERING	nelidlow = '<n>'	lower id for nodes/elem/nrb/nsets
	nelidup =	upper id
	idlow =	lower id for other types
	idup =	upper id
	frozenlow =	lower id for range where labels are frozen
	frozenup =	upper
		entity renumber is optional
<CONNECTION_FILE	file = 'filename'	name of xml connection file
	version = '<id>'	version of this file
<CONNECTION_SETTINGS	target_title = '<title>'	destination component for connections
	target_subtitle = '<subtitle'	

- **Orientation of component by master and slave point**

<MASTERPOINT	name = 'point name'	lower id for nodes/elem/nrb/nsets
<SLAVEPOINT	Origin = '<x><y><z>'	coordinate
	Origin_node = 'node'	may be id or name of node
	Ox = '<x><y><z>'	2nd coordinate to define depenetration vector
	Ox_node = 'node'	
	Vector = '<x><y><z>'	depenetration vector
	Rotate = '<rx><ry><rz>'	rotation angles (defined for masterpoint only)
	Part = '<id>'	part for contact
	Partname = 'name'	
	Partset = '<id>'	part set for contact
	Partsetname = 'name'	

Here is an axample of a simple database file to show the correct nesting of the elements.

```
<PRIMER_DATABASE category = 'New database' protected='no'>

<VERSION descr = 'Version-1' id = '1' />
<VERSION descr = 'Version-2' id = '2' />
<VERSION descr = 'Version-3' id = '3' />

<H1 category = 'New database' subcategory = 'aaa' owner = 'fred bloggs' >
<COMPONENT file = 'Component_files/a1.key' version = '1' />
<RENUMBERING nelidlow = '100000'
nelidup = '199999'
idlow = '1000'
idup = '99999'
frozenlow = '1'
frozenup = '1000' />
```

```
<SLAVEPOINT name = ’point A’
Origin_node = ’3000’
Vector = ’1 0 0’
Part = ’1000’ />
</H1>

<H1 category = ’New database’ subcategory = ’bbb’
<COMPONENT file = ’Component_files/b1.key’ version = ’1’ />
<COMPONENT file = ’Component_files/b2.key’ version = ’2’ />
<COMPONENT file = ’Component_files/b3.dat’ version = ’3’ />
<EXTRA_DATA file = ’Component_files/extra1.k’/>
<EXTRA_DATA file = ’Component_files/extra2.k’/>
</H1>

<H1 category = ’New database’ subcategory = ’ccc’ >
<CONNECTION_FILE file = ’connection.xml’ version = ’3’ />
<connection_settings target_title = ’New database’ target_subtitle = ’aaa’/>
</H1>

</PRIMER_DATABASE>
```

Template File

<PRIMER_TEMPLATE		opening tag

- **Selecting a component**

<SELECTED	category = ’title’	identifies matching component
	subcategory = ’subtitle’	

- **Orientation**

<MASTERPOINT	(as defined above)	master point may be defined in template

Here is an example of template file format.

```
<PRIMER_TEMPLATE>
<SELECTED category = ’New database’ subcategory = ’aaa’/>
<SELECTED category = ’New database’ subcategory = ’bbb’/>
<SELECTED category = ’New database’ subcategory = ’ccc’/>
<masterpoint name = ’point A’
Origin = ’1000 200 110’
Rotate = ’0 30 0’
Part = ’1200’
/>
</PRIMER_TEMPLATE>
```

APPENDIX XVII: MAT100 <DT> added mass for solid spotwelds

LS-Dyna has a special method of adding mass to MAT100 spotweld solids, which may result in a rather higher true %age added mass than the user is lead to expect.

The method is only applied when the parameter <DT> on the MAT100 card is greater than zero.

In this case, in addition to the normal calculation of added mass (let's call it X), LS-Dyna scales the density of the MAT100 solids on a per element basis and reports this extra added mass (let's call this Y) in the otf (d3hsp) file as

"added mass for type 100 hexahedron spot welds=Y"

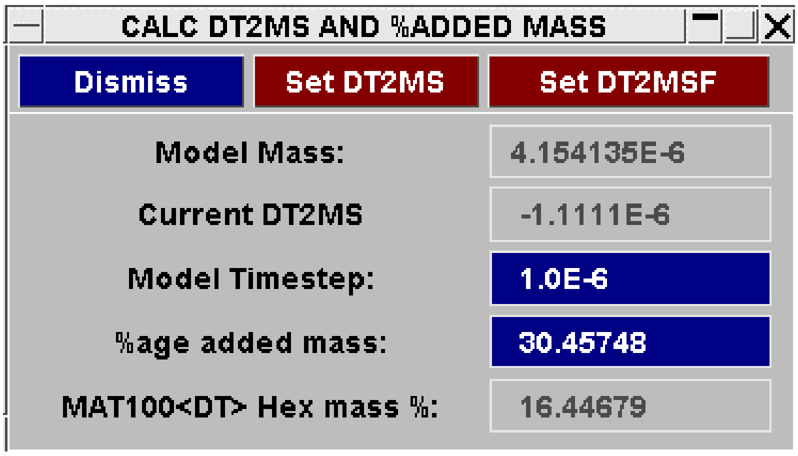
The true added mass ratio is **(X + Y) / physical mass.**

LS-Dyna (LS971R4), however, takes the physical mass as original physical mass (before density adjusted) + Y, and reports the added mass ratio as **X / physical mass**.

Primer contour functions and added mass reports on the part table will use the true added mass ratio and, consequently, will give higher values than the otf file.

The **CALC DT2MS** function accessed under KEYWORD > CONTROL will report the MAT100 DT added mass as a %age of model mass. It is already included it in the "%age added mass" given.

It is worth noting that changing DT2MS on the *CONTROL_TIMESTEP card will not affect the MAT100 added mass..



Technical Topics to do with Graphics

This section attempts to explain some of the technical aspects of using software and graphics on networked computers. It covers both Unix and Windows operating systems, and explains how the Oasys Ltd. LS-DYNA environment suite utilises these.

If all you want to know is how to set up the DISPLAY environment variable click here.
If you want to troubleshoot X11 graphics problems click here

Otherwise read on...

1. Window Managers

1.1 What is a window manager?

All modern computers provide an interface with the user that is based on "windows":

- On Unix (and Linux) operating systems this will be based on the X11 graphics system;
- On Microsoft Windows operating systems this will use the native Windows.

(This terminology leads to confusion: windows with a lower case "w" refers to any window on any operating system; Windows with an upper case "W" refers to the generic Microsoft Windows operating systems [NT, 2000, 95, 98, Millenium, CE, ...]).

The functionality in both cases is quite similar: both user and software can request windows on the screen that are dedicated to a particular application. These windows can be resized at will, and their appearance modified in a range of ways.

Clearly something has to mediate the demands of the user and applications, reconcile them with the hardware that is available, manage the mouse and keyboard inputs, decide which windows lie on top of others (the "stacking order") and so on. We will refer to this as the Window Manager.

The way Window Managers work under Microsoft Windows and X11 are radically different, although the functionality presented to the user is very similar in both cases. This document will not attempt to explain their innards, only their characteristics in so far as they affect the Oasys Ltd. LS-DYNA environment.

1.2 What are the main differences between X11 and "Windows" window managers?

X11

- Is ubiquitous on "engineering workstations" running all variants of Unix, and also on Linux-based systems.
- Is intrinsically operating system, network and hardware transparent. Any machine running the X11 protocol can, via a network, display windows on any other machine also running X11. This capability is independent of hardware type, manufacturer and screen attributes.
- Is "open source" provided by the "X Consortium" (http://www.x.org). Anyone can download, examine and modify the source code and, more importantly, its openness almost guarantees that X-based applications will continue to operate across a range of hardware platforms in the future.
- Supports accelerated 3D graphics by "overloading" the X protocol with the extra information required. The OpenGL graphics library has become the industry standard for 3D graphics, and all modern X11 based window managers support this. Consequently 3D graphics may also be run over networks under X11.

Microsoft Windows

- Is provided only on machines running some version of Windows.
- Is not really designed to operate over networks. Inter-operability between machines and operating systems is nearly impossible without 3rd party software.
- Is proprietary to Microsoft.
- Also supports 3D graphics (OpenGL and others) via direct access to the graphics hardware.

1.3 What inter-operability is there between X11 and Microsoft Windows?

Sadly there is no intrinsic support for the one system in the other, nor any signs of there being any in the forseeable future. Given the gradual convergence of "workstation" and "PC" hardware this seems strange, but no-one ever said that the world had to make sense!

However it is possible via 3rd party software packages to emulate Windows under X11, and also X11 under Windows, and users with mixed hardware are currently forced to take this approach.

1.4 What window managers will the Oasys Ltd. LS-DYNA environment run under?

The Oasys Ltd. LS-DYNA environment suite is developed primarily on Unix-based machines using X11. It is also developed on PCs running Microsoft Windows, but at present it requires an X11 emulator to be running on Windows-based platforms.

"Native" Windows support is being developed (slowly) and will be available at some time, as yet unknown, in the future.

2 Graphics

2.1 A brief description of how modern graphics hardware works.

All modern workstations and PCs separate the main processor, the graphics card and the display. At its simplest:

The main processor (CPU, etc)	*Sends graphics requests to*	**The graphics card**. Which has its own memory and, in the case of better cards, considerable "intelligence" for 3D graphics (lighting, shading, hidden surface removal, etc)	*Sends video output to*	**The Display.** **Cathode ray tube, LCD, plasma, projector, etc**

2.2 The graphics card

There is a huge range of graphics cards available which are grouped here by approximate category. However graphics cards tend to be optimised for particular applications, and in practice these categories will be blurred:

A base-level card	Will typically have 8 bit-planes, giving an acceptable colour range (2^8 = 256 colours) - but no more. Hidden-surface removal, shading and lighting will typically have to be done in software, probably quite slowly. Double-buffering may be available using 4:4 bit-planes, but shading will be poor (2^4 = 16 colours in each buffer).
A higher specification card	Will typically have 24 bit-planes, giving a "true" colour range (2^{24} = 16777216 colours). Such a card may have a hardware Z-buffer, giving faster hidden-surface removal. It may also have some hardware support for shading and lighting. Double-buffering will typically be available using 12:12 bit-planes, giving good shading (2^{12} = 4096 colours in each buffer, which in practice is virtually indistinguishable from 24 bit-plane "true" colour).
A top specification card	Will probably have at least 48 bit-planes, and maybe some overlay planes as well. Hardware Z-buffering, lighting and shading will be available - and extremely fast (many such cards have several processors dedicated to this). Texture mapping in hardware will tend also to be available. Double buffering using 24:24 bit-planes will be "True" colour, and "overlay" planes will allow user-menu and graphics to be separate, reducing the number of image redraw evnts required.

An explanation of some of the terms above:

Bit-planes

After the two width and height dimensions this is the third: "depth". Put simply the more bit-planes a screen has the more colours it can display at the same time.

More specifically the number of colours that can be displayed simultaneously is 2 to the power of the number of planes, thus typical arrangments are:

An 8 bit-plane screen

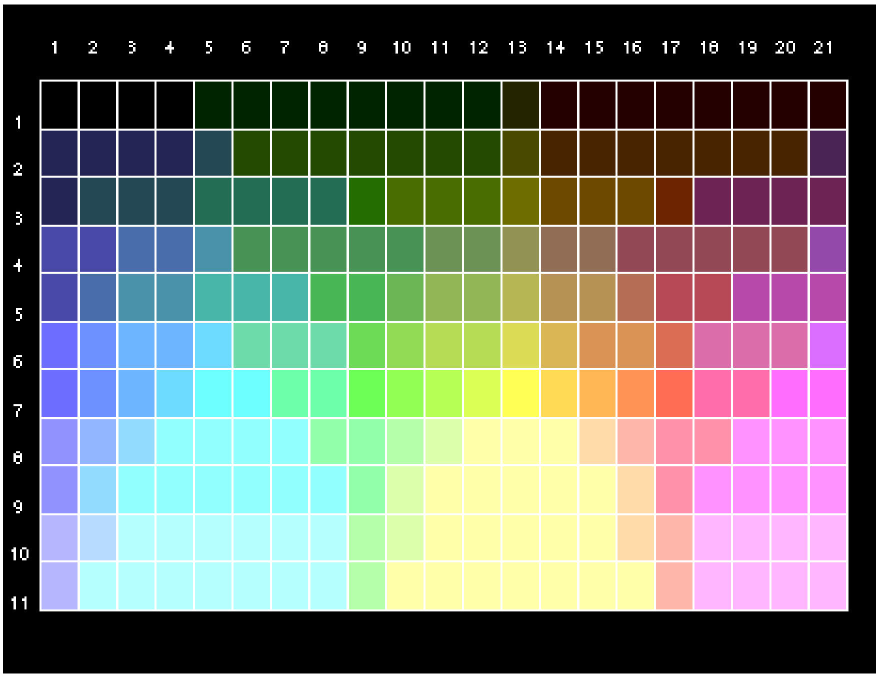
Has 2^8 = 256 colours, usually arranged as:

3 bits Red	= 8 shades
3 bits Green	= 8 shades
2 bits Blue	= 4 shades

The colourmap opposite shows the typical colour ramp available on an 8 bit-plane display.

Note how some of the adjacent shades show poor gradation.

This works surprisingly well, since the human eye is less sensitive to blue than it is to red or green.
Double-buffering such a display to 4:4 bit-planes tends to give poor shading since, even if dithering is used to give more shades, the colour range available is still poor.



A 24 bit-plane screen

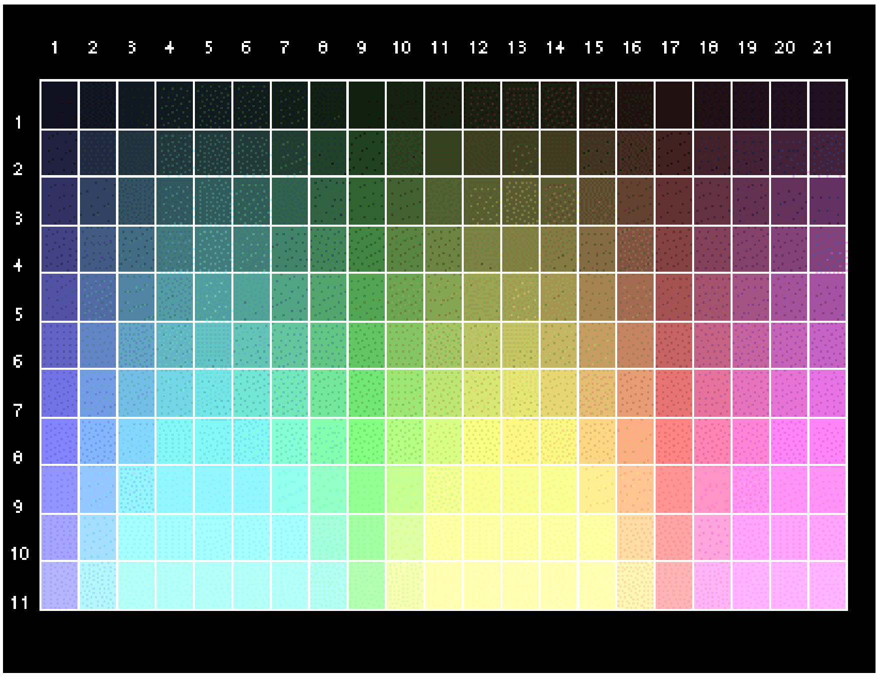
Has 2^{24} = 16777216 colours, arranged as:

8 bits Red	= 256 shades
8 bits Green	= 256 shades
8 bits Blue	= 256 shades

The colourmap opposite shows a typical 24 bit-plane colour ramp.

Note how all shades vary smoothly: even in the very dark and light regimes.

This is often referred to as "true" colour since the human eye can only resolve about 100 shades of each primary colour, and this system produces over twice as many shades. Double-buffering to 12:12 bit planes gives 4096 shades per buffer (4 bits each of red, green and blue) which in practice, with hardware dithering, is difficult to distinguish from full 24 bit-plane colour.



Hidden-surface removal, and "Z" or "depth" buffering.

Most 3D computer graphics is made up of many facets (polygons) which must be displayed correctly such that those nearest the observer obscure those which are further away. This is referred to as hidden surface removal.

The most common method of performing this is to have an auxiliary buffer which is never drawn, but which contains the depth relative to the viewer. This is the Z dimension, where X is width and Y is height, hence "Z-buffer" or "Depth buffer". A pixel in an incoming polygon is only drawn to the display if its depth dimension is closer than the depth currently stored for that pixel in this buffer.

Better graphics perform this in hardware, with little or no speed penalty, but on primitive displays this must be performed in software resulting in a dramatic reduction in "hidden surface" display speed.

Shading and Lighting

Most images that attempt any degree of realism (which is not always the case in engineering plots) require some degree of lighting, which results in a colour variation across facets.

Better graphics cards can be pre-programmed with light source and intensity information, and they will compute the lighting parameters of each incoming facet given its position in space. They will then shade it as required, computing all the colour variations themselves.

More primitive systems may require this to be carried out in software, again giving a dramatic reduction in speed of shaded and lit plots.

Double-buffering

When smooth animation or image rotation/scaling etc is required it is necessary to perform "double-buffering".

The display memory is split into two buffers: one, the front buffer, currently visible; the other, the back buffer, not displayed. Generally each buffer will use half the total graphics bit-planes available, trading off colour resolution against speed. The notation **N:N** user here (eg **12:12**) implies double-buffering using **N** planes in each buffer.

The new image is drawn (invisibly) into the back buffer, and when it is complete the front and back buffers are swapped over exposing the new image, and the process is repeated. The buffer swap operation is performed in hardware, and can usually be done in one vertical retrace of the display - which is faster than the eye can detect - so the transition appears to be smooth.

Texture Mapping

Is a bit like wall-papering. A pre-defined pattern (or "texture"), such as a brick pattern for a wall, is "mapped" onto each facet with the appropriate lighting and shading superimposed.

This tends currently not to be used for engineering applications as, historically, it has been a slow process. However as graphics cards get faster expect this to appear as a rendering option.

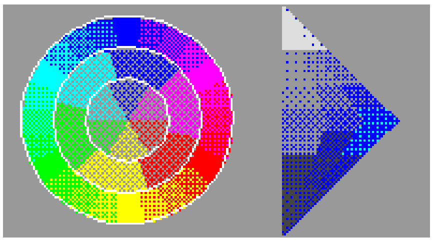
Overlay Planes

Many modern graphics cards permit their memory to be partitioned so that the bit-planes used for the "image" (perhaps 24 planes) are separate from those used for other windows and menus (typically 8 planes). The latter bit-planes are given the additional, special attribute that one (reserved) colour is "transparent", which makes them usable as an "overlay".

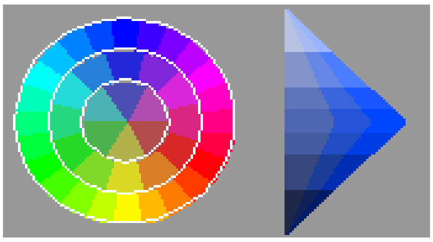
Imagine a transparent sheet that covers your image, which can be wiped clean and redrawn without affecting the image underneath: this is how overlay planes function. The advantage is that other windows, popup menus, etc can be drawn, mapped and unmapped in the overlay planes without corrupting the image planes beneath them, removing the requirement for the (potentially slow) redraw of that image.

From release 8.1 the "screen menu" component of the Oasys Ltd. LS-DYNA environment may be run in the overlay planes where the hardware supports these.

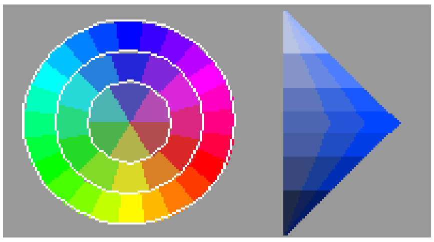
The following images (from the colour "palette" in D3PLOT) demonstrate the effect on the colours available of running the screen menu system in visuals of different depths. (The difference between 8 and 24 bit-plane depths will be undetectable in most browsers.)



4 bit-planes. Dithering is required to achieve most shades.



8 bit-planes. Nearly all shades are directly available

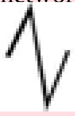


24 bit-planes. All shades are directly available.

2.3 Graphics over a network

In most cases you will be drawing graphics on the display of the machine you are using, and this will almost always be the fastest method since the processor and graphics card can communicate directly at high speed and, if necessary, share memory.

This need not be the case under X11, where networked graphics may also be used:

The main processor (CPU, etc)	*Sends graphics requests ....*	Over a network 	*... to a second computer*	**The graphics card**. Which has its own memory and, in the case of better cards, considerable "intelligence" for 3D graphics (lighting, shading, hidden surface removal, etc)

The network

- Is almost always some form of ethernet running TCP/IP (***T***erminal ***C***ontrol ***P***rotocol / ***I***nternet ***P***rotocol)
- But other mechanisms, for example "DecNet", are possible.
- May be of any speed, but obviously the faster the better. Recent machines will have 100 MBit/second ("100 Base T") network ports.

Under X11 networked graphics are used in"client/server" mode by:

- Opening a connection to a remote machine, by setting the DISPLAY environment variable to a "server:screen" on that machine.
- The "client" (application) software sends its graphics requests down the network to that machine.
- The "server" software on that machine, generally part of the window manager, translates these into graphics on the remote machine.

The 3D OpenGL protocol operates over a network under X11 in exactly the same manner:

- A remote connection is opened as before.
- The client application sends graphics requests, this time 3D ones, to the remote machine.
- The server software on that machine (which must include the GLX extension) turns these into 3D graphics requests on that machine.

Special aspects of networked graphics:

If every image redraw, rotation operation or animation frame requires all the graphics information to be sent down the network between client and server machines, networked graphics can become quite slow - especially if the network is low speed. Both "raw" X11 and OpenGL protocols recognise this, and provide a means of storing graphics information in the remote server to speed up redraws.

2D X11 images:

- Can be stored in off-screen "pixmaps", which are effectively copies of the screen in memory. They can be mapped to the screen sufficiently quickly to provide good animations.
- These are suitable for image redraws; and for pre-computed animation frames, where an unchanging sequence of images is played repeatedly.
- They do not help for rotation, scaling or translation since a 2D image must be rebuilt from scratch to show the new appearance.

3D OpenGL graphics imformation:

- Can be stored in "objects" in the server. An "object" contains all the 3D information required to describe a graphical scene.
- Objects can be used for image redraws and for animations, by sending them to the graphics card for re-rendering.
- They can also be used for rotation, scaling and translation since enough 3D information is present to rebuild the image in its new form: only the new rotation and scale information (perhaps a dozen numbers) need be sent down the network to transform a scene of any complexity.

Both mechanisms provide a means, in 2D and 3D respectively, of maximising graphics display speed in some circumstances. If the client machine is doing the "thinking", and the server machine the "drawing", both can operate at full speed at their respective tasks. This can be an efficient means of displaying data from very large databases, since it splits the data retrieval task from the rendering one, although in most cases display on the local machine will be the fastest.

2.4 More about graphics hardware

This section is optional reading for those who want to understand more about the innards of graphics, and find out how to use their machinery more effectively.

What are the "client" and "server"?

The "client" is the process making the graphics requests, for example D3PLOT, T/HIS, etc.

- You own this process - on Unix systems a "ps" command will show it as belonging to your user id.
- In order to display graphics it must open a display connected to a "server": either locally or over a network.

The "server" is the process running either on this machine (the local host) or remotely elsewhere on a network that receives graphics requests from client processes and turns them into graphics on a screen.

- The server is owned by the operating system of the machine - on Unix systems a "ps" command will show it belonging to user id "root".
- It serves all graphics requests that come to it, not just your client's: it is a shared resource.
- You request facilities (eg windows, colourmaps, backing store) but since it is a shared resource these may not always be available.

In most cases a machine will be running a single X server, by convention #0. However there is a special case of the "X Virtual Frame Buffer" (Xvfb) that is used for batch mode graphics which, by convention, will run as server #1 - see section 2.5 below.

What the ?"!!##? are X11 "visuals"?

The X11 protocol has to operate on a wide range of hardware ranging from old "black and white" displays to modern colour ones, and it is fact of life that the hardware on these machines varies. Some machines support a range of graphics options simultaneously: for example they can have some windows operating in "greyscale" mode alongside others in full colour; they may also provide "overlay" planes. Most modern machine provide a range of visual types and depths (#bit-planes).

To make some sort of sense of all this the X11 protocol groups graphics capability into six categories of "visual": two greyscale and four colour.

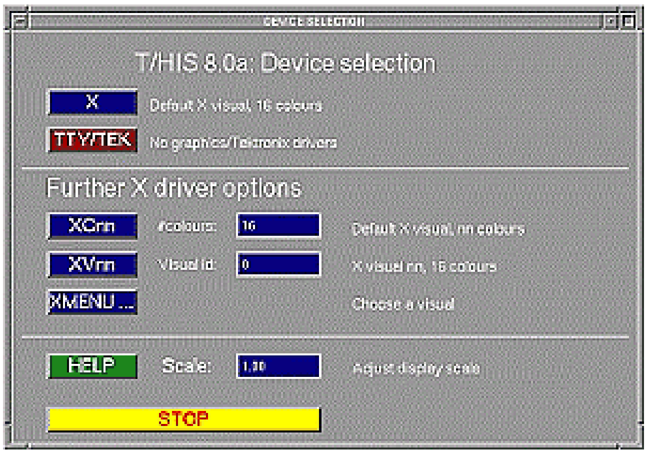
Visual type	What it is	Description and usage
StaticGray	Preset ("static") range of greyscale shades	• Primitive and seldom seen today. Offers only a fixed range of grey shades. • Colour graphics are rendered as shades of grey.
GrayScale	Configurable range of greyscale shades	• Also seldom seen today. Grey shades can be defined by the application. • Colour graphics are rendered as shades of grey.
StaticColor	Preset ("static") range of colour shades	• Seen on older colour displays: colours can only be selected from the pre-defined ones set up. • Usually 4 or 8 bit-planes deep. • Oasys Ltd. LS-DYNA environment uses this for graphics if nothing better is available.
PseudoColor	Configurable range of colour shades	• Found on almost all colour displays and used typically for menus and user interfaces. • Usually 4 or 8 bit-planes deep, but rarely also 12 bit-planes (4096 cols). • Each shade can be defined individually by the application, for example 100 shades of pink • Is used for graphics by Oasys Ltd. LS-DYNA environment, by setting up a colour "cube", if no better visual is available.
TrueColor	Preset linear ramp of red, green and blue shades	• Found on most colour displays. Each colour (r,g,b) has a linear ramp forming a colour "cube" from which shades can be selected. The resolution of each axis of the cube is $2^{\#bits}$ of that colour. • Usually 8, 12 or 24 bit-planes deep, #bits being organised R|G|B as 3|3|2, 4|4|4 and 8|8|8 respectively. • The 24 bit-plane case is "true" colour in that it provides more shades (16777216) than the eye can resolve. This is the preferred visual/depth combination for graphics in the Oasys Ltd. LS-DYNA environment. • The 8 bit-plane case gives only a limited range of shades (256) limiting its usefulness for graphics

DirectColor	Configurable ramps of red, green and blue shades	• Each colour (R,G,B) has its own configurable "ramp", permitting a non-linear and dynamically changeable colour cube. • Invariably 24 or more planes deep. • Needed only for specialised applications. • The Oasys Ltd. LS-DYNA environment only uses this if nothing better is available, and configures linear RGB ramps - effectively TrueColor.

What visuals does the Oasys Ltd. LS-DYNA environment use?

This depends upon the application, whether you are working in 2D or 3D, and what is available on the selected display.

T/HIS 2D "XY" plotting using X11 graphics only.



For graphics

The "**X**" option selects the default visual of the display, which is that in which the main window manager is running. Typically only 16 colours will be requested, since T/HIS graphics are very simple, and these can be shared with other applications. The advantage of this is that screen dumps (eg "xwd") will work correctly giving the proper colours in all windows and sub-windows.

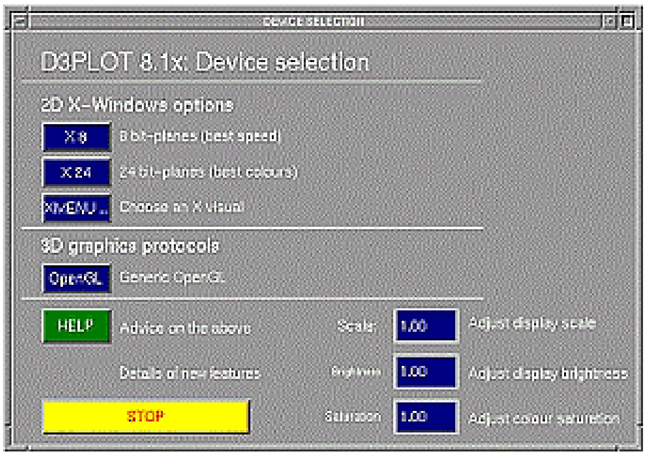
The default visual is usually either 8 bit-plane PseudoColor, or 24 bit-plane TrueColor. Since the demands of the application are so simple either will generally provide all that is required.

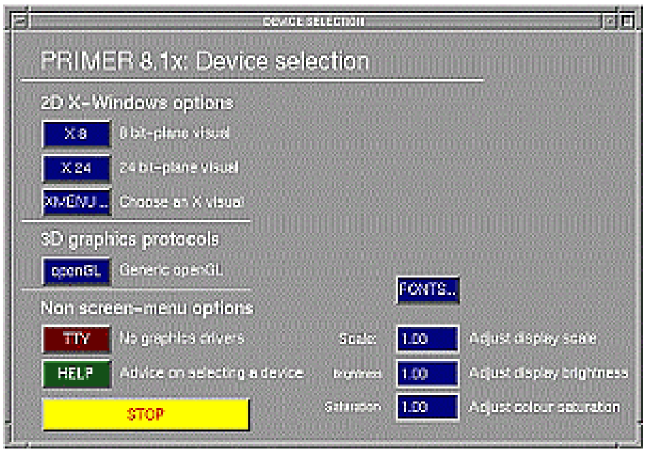
The other options **XCnn, XVnn, XMENU...** will only very rarely be required on modern displays, contact Oasys Ltd for advice if the default **X** doesn't work.

For menus

The menu interface will also run in the default visual, unless explicitly forced elsewhere with the **SM_USE_VISUAL** environment variable.

D3PLOT and PRIMER 3D graphics using OpenGL and X11.





For graphics

When "3D graphics" is used (the recommended choice) OpenGL chooses its own visual from those available on the display. There is no user control over this.

The "2D X11 graphics" options should select:

X8

If animation and dynamic rotation speed is critical, and some loss of image quality during shading can be tolerated.

This will choose an 8 bit-plane visual, usually TrueColor or PseudoColor, with only 256 colours - hence the poor shading. However since animation and dynamic viewing are performed using off-screen "pixmaps" (memory containing copies of the screen image), the smaller depth means faster transfer of data from memory to screen.

X24

If shaded image quality is more important than animation and dynamic rotation speed.

This will choose a 24 bit-plane visual (usually TrueColor) with 16777216 colours, so shading should be excellent. However the amount of data to be transferred from Pixmap to Screen is three times as much as in the 8 bit-plane case, and is correspondingly slower. Memory consumption in the X11 server is also greater.

XMENU..

Lets the user select a specific visual.

This is seldom needed - contact Oasys Ltd for advice if the default choices do not give satisfactory results.

For menus

The menu interface will also run in the deepest overlay visual that exists on the server, or in the default visual if none exists, unless explicitly forced elsewhere with the **SM_USE_VISUAL** environment variable.

How does OpenGL work with X11?

Initially the client process opens a conventional X11 connection with the server, then it requests a further OpenGL connection on top of this. Broadly speaking:

- The X11 server manages the windows themselves (sizing, placement, redraw requets), and all mouse and keyboard events.
- The OpenGL connection provides a parallel path for rendering high speed graphics in the window(s).

Where the graphics are being displayed on the local machine a "direct" OpenGL rendering connection is used to give the fastest possible graphics data transfer between client process and display. This bypasses the X11 server.

Where OpenGL graphics are being rendered over a network then an "indirect" OpenGL rendering connection is used:

- The remote X server must have the OpenGL/X (GLX) extension installed. If it doesn’t have this the connection will be refused.
- X11 protocol requests are "overloaded" with OpenGL rendering data and sent to the remote server.
- The GLX extension in that server turns them into local rendering requests on the remote machine.

Using OpenGL over a network generates more traffic than "raw" X11 graphics. This isn't surprising: a coordinate in X11 is an (X,Y) data pair expressed as two "short" (16 bit) integers. The same coordinate in OpenGL is an (X,Y,Z) triplet expressed as 3 "floating" (32 bit) numbers - 3 times as much data. For this reason networked OpenGL graphics is best used with "objects" stored in the server, as this requires geometry information to be sent only once. "Objects" are described below.

What are OpenGL "objects"?

Normally OpenGL rendering requests are sent to the display, drawn and then forgotten. If the image needs to be redrawn, for example due to rotation, then the requests must be regenerated, resent, and so on. When rendering takes place on a local display, via a "direct" rendering connection, then this method gives the best trade-off between speed and memory consumption.

However if you are displaying remotely over a network then the continual re-transfer of the same data in order to draw successive frames is slow, and a method of storing the graphics data in the server would give an obvious efficiency gain. OpenGL "objects" provide this capability:

- Graphics data requests ("move to a point", "draw a line", "change colour", etc) are stored locally in an "object".
- To display that object again the only command required is "redraw the object".
- Redrawing from a stored object tends to be faster than from explicit requests.

If it is faster why don't all OpenGL applications use objects all the time, even when rendering locally?

- Objects use a ***lot*** of memory.

  The application programmer has no control over how graphics data are stored in objects, and cannot therefore practise any storage economies that are made possible by his knowledge of how his data are organised.

  It is not difficult to write a data storage scheme that is much (eg 2x or 3x) more compact than that afforded by objects. This translates into huge savings of memory which, if they stop the machine using virtual memory (swapping), more than offset any speed gains from using objects.
- Objects are slow to create initially.

  The first pass, in which the data is sent to populate the object, is quite slow. This is a considerable price to pay if the image is only to be displayed once, as any savings only come from the 2nd and subsequent renderings of it.

Of the Oasys Ltd. LS-DYNA environment only D3PLOT permits the use of "objects" for data display, and it is recommended that this mode is used only when OpenGL graphics are being used on a remote server over a network. The default "vector" mode in D3PLOT, in which it manages the storage of its own graphics data "vectors", should be used on local displays.

2.5 The "X Virtual Frame Buffer" (Xvfb) server

So far it has always been assumed that graphics requests, however generated, will end up as images on a screen somewhere. However this is not always what is required: in the special case of batch mode running we wish to generate images for capture as bitmaps, but we don't want to have to display them in order to do this. Most conventional X11 servers will only permit access to clients when someone is logged in at their screen and has permitted remote access, which is not much use for batch jobs.

This is where the X Virtual Frame Buffer (Xvfb) server is used.

- It is an X11 server process just like any other.
- It can co-exist with an ordinary X11 server on a machine (it is distinguished by its #server id).
- It accepts and processes X requests from clients just like an ordinary server.
- But it generates images internally in memory, not on a physical display.
- It can act autonomously, not requiring a user to be logged on at a display.
- In fact it can run on a machine that has no display or graphics hardware.

Where facilities for batch graphical processing have been provided by Oasys Ltd the Xvfb server will also be provided, and will have to be running somewhere on an accessible machine (or locally). By default it will be started with the properties:

- Server id #1 (hence the **DISPLAY** variable will be **<*machine name*>:1**)
- 1280x1024x8 (width x height x bit-planes depth) resolution

However these properties may be modified if required. For more information please contact Oasys Ltd.

3 Controlling Graphics in Oasys Ltd. LS-DYNA environment

All Oasys Ltd graphical programmes (PRIMER, D3PLOT and T/HIS) use the same menu interface; however the "graphics" display (of data) varies according to the specialised nature of each programme:

- D3PLOT has a sophisticated 2D (X11) and 3D (OpenGL) rendering capability, which is optimised for speed.
  - Networked graphics are supported efficiently via Pixmaps (2D X11) and Objects (3D OpenGL).
  - Graphics and other memory usage can be controlled.

- PRIMER also has 2D (X11) and 3D (OpenGL) rendering capability, but optimised to display a wide range of entity types - speed being less of an issue.
  - Performance over a network is adequate, but no special provision is made for this.
  - No user control over graphics memory is provided.

- T/HIS has only 2D (X11) rendering capability, as this is all that is required for XY graph plots.

All the Oasys Ltd. LS-DYNA Environment software requires an X11 based window manager or emulator to be running (even when OpenGL is used for rendering). The following topics related to this are described below:

- Defining the DISPLAY environment variable, which determines where graphics are drawn.
- Configuring the parameters of the Oasys Ltd "menu interface".

3.1 Opening an X11 connection to a display: the DISPLAY environment variable.

The syntax of the DISPLAY variable
The simple case: displaying on this machine <= **Probably all you need to know**
Examples of networked graphics setup
Configuring under Unix/Linux
Configuring under Microsoft Windows.
Troubleshooting X11 graphics

The syntax of the DISPLAY environment variable

The X11 protocol requires that an "address" and "screen" number are nominated for its graphical output. This done with the **DISPLAY** environment variable, which has the form

Entries in ***(..)*** can be omitted.

(<*address*>):<*server*>(.<*screen*>)

- The **<*address*>** is a computer name or, more precisely, a network address. If omitted it means the local machine.
- The **<*server*>** is the X11 "server" process. This is typically server #0, but it is possible to have more than one server running.
- The **<*screen*>** is the screen number, starting at 0, on that computer (some machines have > 1 screen). It can be omitted if the display has only one screen.

The simple case: displaying on the screen attached to this computer.

In the vast majority of cases all you will want to do is to display graphics on the screen attached to this computer. Therefore you need default "address", "server" #0, default "screen", which is achieved by setting:

DISPLAY = "**:0**"

Sone examples of achieving this under different operating systems:

Unix/Linux	C shell (/bin/csh, /bin/tcsh)	**setenv DISPLAY :0**
	Bourne/Korn shell (/bin/sh, /bin/ksh)	**DISPLAY=:0; export DISPLAY**
Windows	In the System Properties panel	Variable = **DISPLAY** Value = **:0**

If you are not interested in networked graphics then this is all you need to know about establishing a connection, and you can ignore the rest of this section.

The following examples show how **DISPLAY** could be set to display graphics in a range of local and networked locations:

DISPLAY	Where the graphics will appear	Comments
tigger:0	Server #0 (screen 0) on machine "tigger"	The IP (***I***nternet ***P***rotocol network) address of "tigger" must be known to your system
170.177.15.2:0	Server #0 (screen 0) on the machine with the IP address 170.177.15.2	Since the IP address is given explicitly the remote machine name need not be known.
:0.1	Screen #1 on server #0 on this machine	For a system with 2 screens, this will display on the second.
rainbow:0.1	Screen #1 on server #0 on machine "rainbow"	The IP address of machine "rainbow" must be known, and it is assumed to have at least two screens.

The way the **DISPLAY** variable is set, and remote machine names are mapped to IP addresses, depends upon the operating system in use.

Defining DISPLAY on Unix and Linux systems:

The **DISPLAY** enviroment variable (**$DISPLAY**) is set by:

C shell (/bin/csh /bin/tcsh)	**setenv DISPLAY rainbow:0** **setenv DISPLAY 170.177.15.2:1**	**Server #0 (screen 0 on machine "rainbow"** **Server #1 (screen 0) on machine 170.177.15.2**
Bourne/Korn Shell (/bin/sh /bin/ksh)	**DISPLAY=rainbow:0; export DISPLAY** **DISPLAY=170.177.15.2:1; export DISPLAY**	Ditto

To save the **DISPLAY** variable a given user could place it in their startup files: "~/.cshrc" (for C shell) or "~/.login".

Machine name (hostname) to IP address resolution is defined in the file **/etc/hosts**. (This is owned by root, and requires superuser privileges if it is to be updated.)

This has any number of rows of the form:

<ip address>	**<name>**	**(<name>)**	**(<name>) ...**	IP addresses must have at least one "name", but may have any number of alternative aliases

For example:

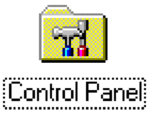
170.177.15.2	**atuhp002**	**atghp002**	**fred**	(This machine is known by any of three names)
193.20.116.16	**abcsgi16**			
193.20.116.20	**abcsgi20**			

Defining DISPLAY on Windows systems:

The **DISPLAY** enviroment variable (**$DISPLAY**) is set in the **System Properties** panel.

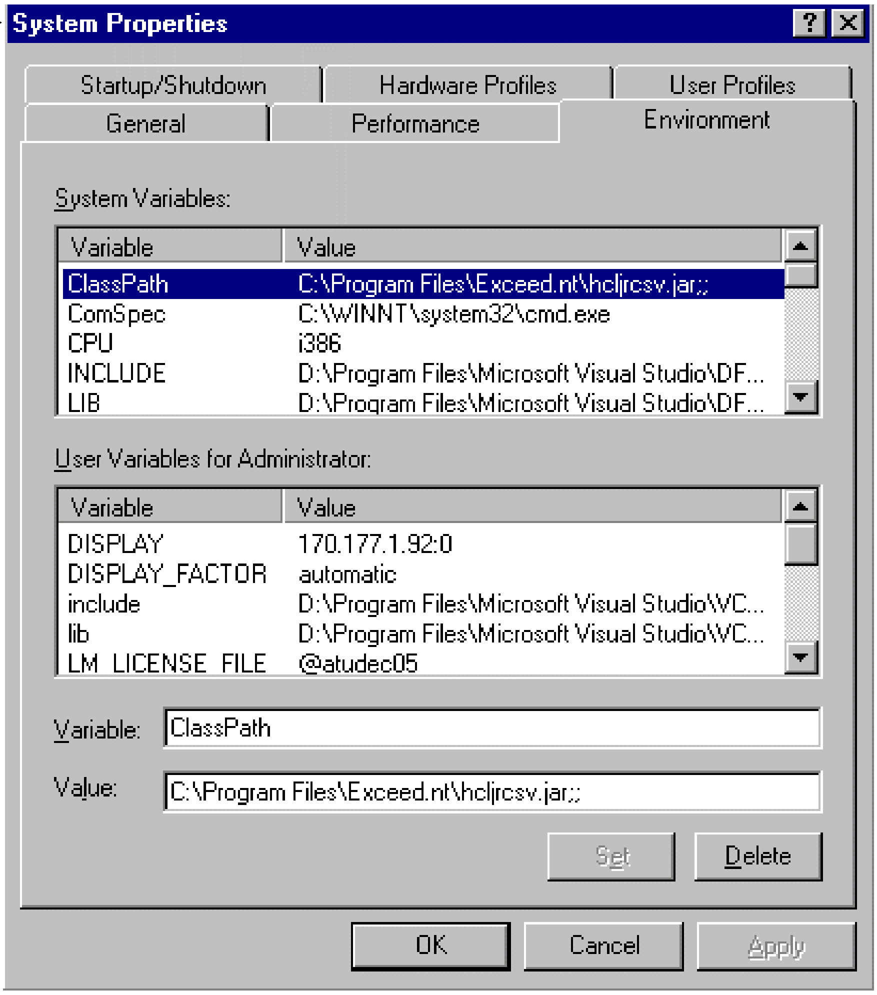
The example below is from a Windows NT 4 system, but other variants of Windows will be similar.

This is accessed by:



=>

=>



In the **System Properties** panel select the **Environment** tab, as shown here.

Click on the **User Variables for <userid>** (here **Administrator**) and insert:

Variable:	**DISPLAY**	
Value:	***170.177.1.92:0***	Insert your own IP address or name

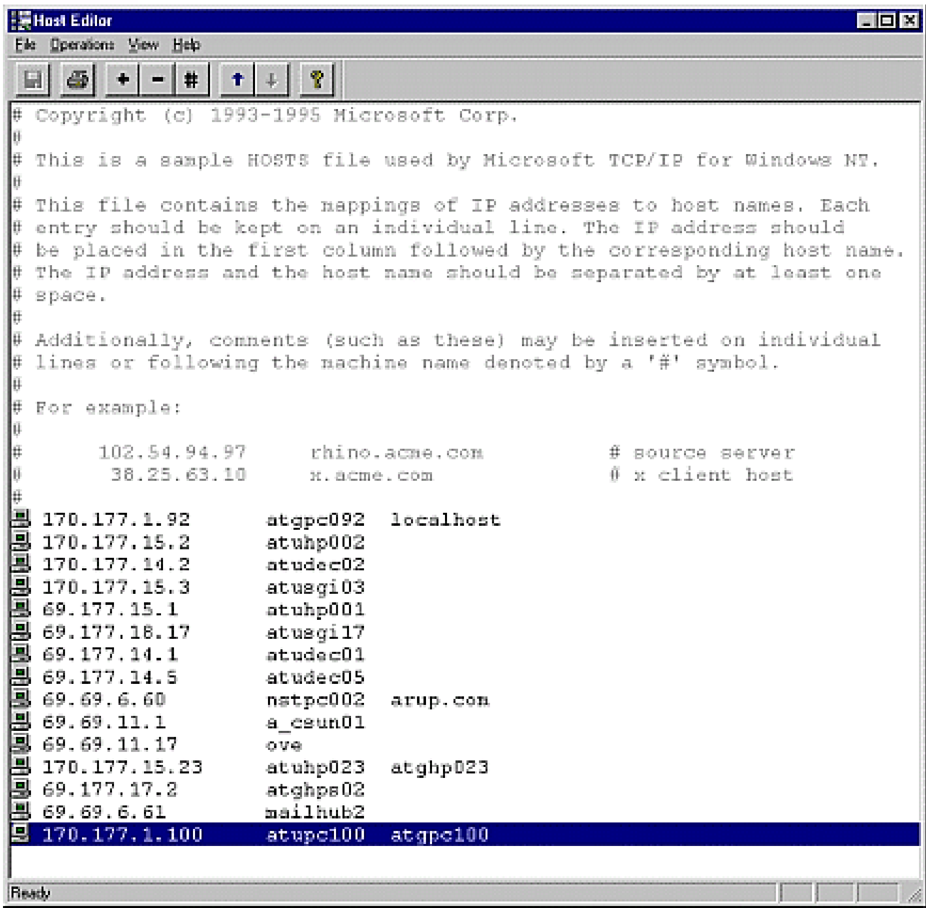
The click on **Set** to add it to your environment variable list.

Machine name (hostname) to IP address resolution is provided by a "Hosts" file (on Windows NT in C:\WINNT\SYSTEM32\drivers\etc) that has the same format as the Unix version above, ie:

<ip address> <name> (<name>) (<name>) ...

This may be updated using a text editor (eg WordPad).

Or on the Hummingbird Exceed™ emulator, which Oasys Ltd recommends, it may be maintained via a **Host Editor:**



The + and - buttons are used to add/remove entries in this panel

Troubleshooting X11 graphics.

Problem	Possible resolution
When you try to start an application you get the message: **`Could not open display`**	This means that the **DISPLAY** variable has not been defined. See above for how to define it on Unix/Linux systems and Windows systems.
When you try to start an application you get the message: **`Could not open display`** ***`<name>:<server>.<screen>`***	This means that your machine (the client) cannot "see" the server machine, or the X11 server on that machine. 1. Is machine <*name*> correct? Check its spelling, and remember that it is case-sensitive. 2. Is <*name*> in your /etc/hosts file (unix) or Hosts file (windows)? 3. Is the IP address in there for <*name*> correct? 4. Is the network path to <*name*> working correctly? (Try "ping"ing it.) 5. Is machine <*name*> turned on and plugged into the network? 6. Is the X11 window manager running on machine <*name*>? 7. Does <*server*>.<*screen*> exist on machine <*name*>?

When you try to start an application you get the message: **Xlib: connection to** ***<name>:<server>.<screen>*** **refused by server** **Xlib: Client is not** **authorised to connect to** **Server**	This means that your client process has made contact with the server's X11 window manager, but has been refused permission to open a window on it. This is a security feature of the X11 system: server window managers must grant permission for clients to open windows, which may be done as follows: On a transient basis, not "remembed" once you log out: Type "**xhost +**" in any window on the server. This will grant permission for any remote client to open windows on this display. To be more selective about which remote clients you will allow to open windows on a display type "**xhost +*name***" where <*name*> is a remote computer name. On a permanent basis, "remembered" across logout/login.: In the file **/etc/X0.hosts** add a list of computer names (each on a new line) that are permitted to open windows on this display. (This is for server #0, for server #1 put it into file **/etc/X1.hosts,** etc.) To allow access to any host put a **"+"** into this file. For more information (on a Unix/Linux host) type "**man xhost**" which describes X11 access control.
The application appears to start, but then fails with a message along the lines of: **X connection to** ***<name>:<server>.<screen>* broken** **(explicit kill or server shutdown)**	This usually means that you have forgotten to reset your **DISPLAY** variable, and have popped up a window on someone else's screen. They, understandably, have got annoyed and killed the window (an "explicit kill"). Check that you are displaying graphics where you intended! If this isn't the problem it may indicate that the server to which you are trying to connect is in distress and can't cope with the extra workload - see below.
The X11 server gets very slow, or locks up completely. Normally there are no error messages, but a heavily overloaded server may produce "synchronisation" errors or other symptoms of its impending demise.	This can happen occasionally when the window manager on a server fails to cope with the load placed on it, typically during animation, and dies (a "server shutdown"). Server shutdowns may also occur if they run out of memory: usually caused by performing large "pixmap" or "object mode" animations in D3PLOT which cause the server to grab lots of memory. (Under Unix/Linux memory cannot easily be returned to the system's free pool once it has been allocated so, like middle-aged spread, memory consumption of a process will tend to grow but never diminish. This is not such a problem under Windows.) An X server in distress may be shut down and restarted by the following methods: • Log out from the console, then select "command line", or "no windows", or some similar option (this will depend on the vendor and operating system) for a new login. Log in, then straight out again, and resume the normal "windows" login. This will shutdown then restart the X11 server, which usually sorts out problems. • If the display has locked up (no response to mouse or keyboard) then log in from a remote machine as "root", and kill the window manager process explicitly. ("**ps -ealf | grep X**" will usually find the process, and "**kill -9 <process id>**" will zap it.). • If you can't log in remotely, or don't have root access, reboot or turn the machine off and on again! Cruel but effective.

OpenGL Extension missing on remote server. `Xlib: extension "GLX" missing on "<name>:<server>.<screen>`	You may see this if you try to open a remote OpenGL window on a local server that does not have the OpenGL/X extension "GLX" installed. You will not be able to open remote OpenGL clients until it has been installed. Note that you may still see this message on a machine that is able to display OpenGL graphics locally. This means that it can handle a "direct" OpenGL connection from a local client (which largely bypasses the X server), but that it does not have the ability to render "indirect" OpenGL requests. See "How does OpenGL work with X11?" for more information. But in the meantime you will have to use a **X..** option to display remote graphics.

Other errors, typically:
`X Error of failed request:`
`Major opcode ...`
`Minor opcode ...`
`Resource id ...`
	These usually suggest an error in the Oasys Ltd. LS-DYNA environment. Please make a copy of the error message and contact Oasys Ltd for help and advice.

3.2 The Oasys Ltd "Menu Interface"

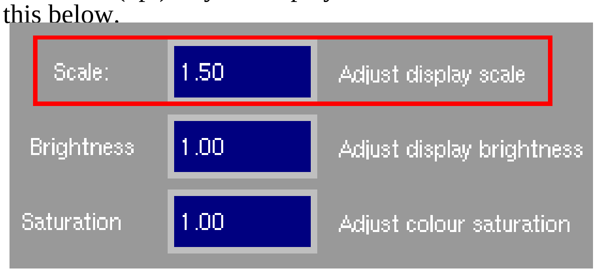
All graphical Oasys Ltd. LS-DYNA environment software uses a menu interface, which is effectively a window manager that runs in the application’s own window.

Its default configuration should work satisfactorily in most cases, but there are some environment variables that can be set to change its behaviour:

Configuring Menu Interface environment variables.

Setting the correct physical resolution for your display.

Environment Variable	Possible values	What it does
USE_PIXMAPS	**true**[default] or **false**	Uses "pixmap" backing store to make the scrolling of windows smoother. If memory consumption in the X server is a problem, which sometimes occurs in older X terminals and emulators, this may be set to **false** to conserve memory. You will notice that scrolling menu windows gives a little bit of "flicker", but functionality is unaffected.
SM_USE_VISUAL	**overlay**[default(1)] **default** ***<visual id>***	Defines which bit-planes the menu system runs in. • "**overlay**" puts it into the deepest overlay planes that exist on the display. • "**default**" uses the default planes. • ***<visual id>*** is the visual id (in hexadecimal) in which it is to run - ask Oasys Ltd for advice first! On some older displays, especially SGis, using the overlay planes may result in strange colours elsewhere on the screen. And on seriously old machines the overlay planes may only be 2 bits deep resulting in the menu system appearing only in shades of grey. In these cases set this variable to **default** to force the menu system into the default (image) planes. (1) "**overlay**" is the default for PRIMER and D3PLOT to avoid conflict with the image planes, but "**default**" is used for T/HIS to make screen grabs simpler. See the notes on "Overlay Planes" above for more information about this.

SAVE_UNDER	**true**[default] or **false**	Determines whether or not the "save under" property of the window manager is requested for the area under popup menus. This property "saves" the portion of the image under these sub-windows, meaning that they do not have to be redrawn explicitly when made visible again. On Compaq OSF 4 operating systems the default behaviour can result in an empty grey area appearing under these menus. This is due to a bug in the Compaq implementation of the X server which appears to be fixed in OSF 5. Setting this variable to **false** will cure the problem on these platforms, although this can lead to a lot of image redraws if the visual used for the menu system is not **overlay** (or the machine has no overlay planes).
DISPLAY_FACTOR	<*undefined*>[default] **automatic** 0.5<**value**<2.0	The Oasys Ltd menu system tries to present a consistent appearance over a range of display resolutions and sizes. To do this it attempts to determine the physical resolution in dots per inch (dpi) of the display and, in conjunction with the pixel resolution (eg 1280x1024), to provide the best appearance it can. Unfortunately not all X servers are correctly configured to provide this information, and there is no standard way of performing this configuration. If the menus on your display seem too big or too small, or just plain "wrong", the following process is recommended: Set **DISPLAY_FACTOR** to **automatic**, then run D3PLOT to see how it looks. If that is still unsatisfactory then try a range of **Scale** values in the front panel to try to arrive at a satisfactory value. If that still doesn't work properly it is probably because the physical resolution (dpi) of your display has not been set correctly see how to do this below. 

For examples of how to set environment variables see Defining **DISPLAY** on Unix and Linux systems: or Defining **DISPLAY** on Windows systems:. The method is identical.

Setting the correct physical resolution for your display.

This is only necessary if you think that your display is assuming the wrong physical properties for the screen, as evidenced by fonts and/or windows coming out the wrong size. Unfortunately there is no standard way of doing this, and the following hardware-dependent solutions may not meet your case. If you still have problems please contact Oasys Ltd for help.

Windows platforms
HP-UX systems
Compaq OSF systems
PCs running Windows with the Exceed emulator.

Windows platforms (2000, XP, Vista)

In most cases the operating system should "know" the physical attributes of the display, but we have seen a few cases (especially on Vista) where this is not the case and it is necessary to give the Oasys Ltd. LS-DYNA environment this infiormation explicitly. This is done via the two environment variables:

`DISPLAY_HEIGHT`	The height of the display in millimetres
`DISPLAY_WIDTH`	The width of the display in millimetres

It is best to set these as "system" variables, as then they will apply for all users on that machine; but if you only have permission to set environment variables for your user name that will suffice. Once set close down and restart the Oasys Ltd. LS-DYNA environment, and it will use these values rather than those supplied by the operating system.

HP platforms (HP-UX operating) system.

Method 1:

Edit the /usr/lib/X11/X0screens file directly.

As user root edit this file and add this line to the end of the file

`MonitorSize` ***`nn`*** **`Inches`**

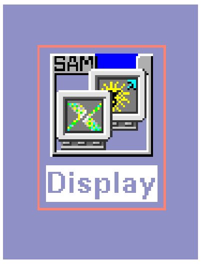
Where <*nn*> is your monitor size, eg 20.

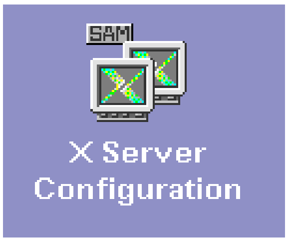
Then logout completely from the console, revert to "command line" login, and allow the window manager to restore the "desktop login" prompt. This will restart the X server with the new properties.

Method 2:

Use the SAM utility to change this file.

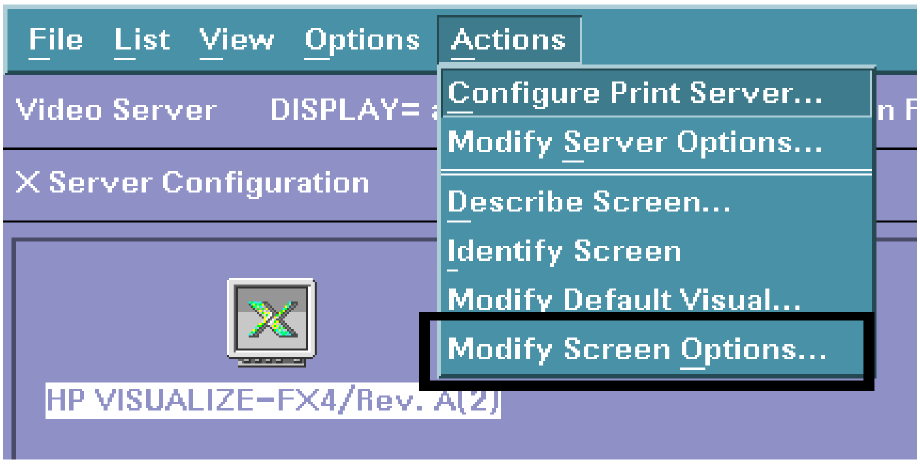
As user root start "sam".

Select **Display** 

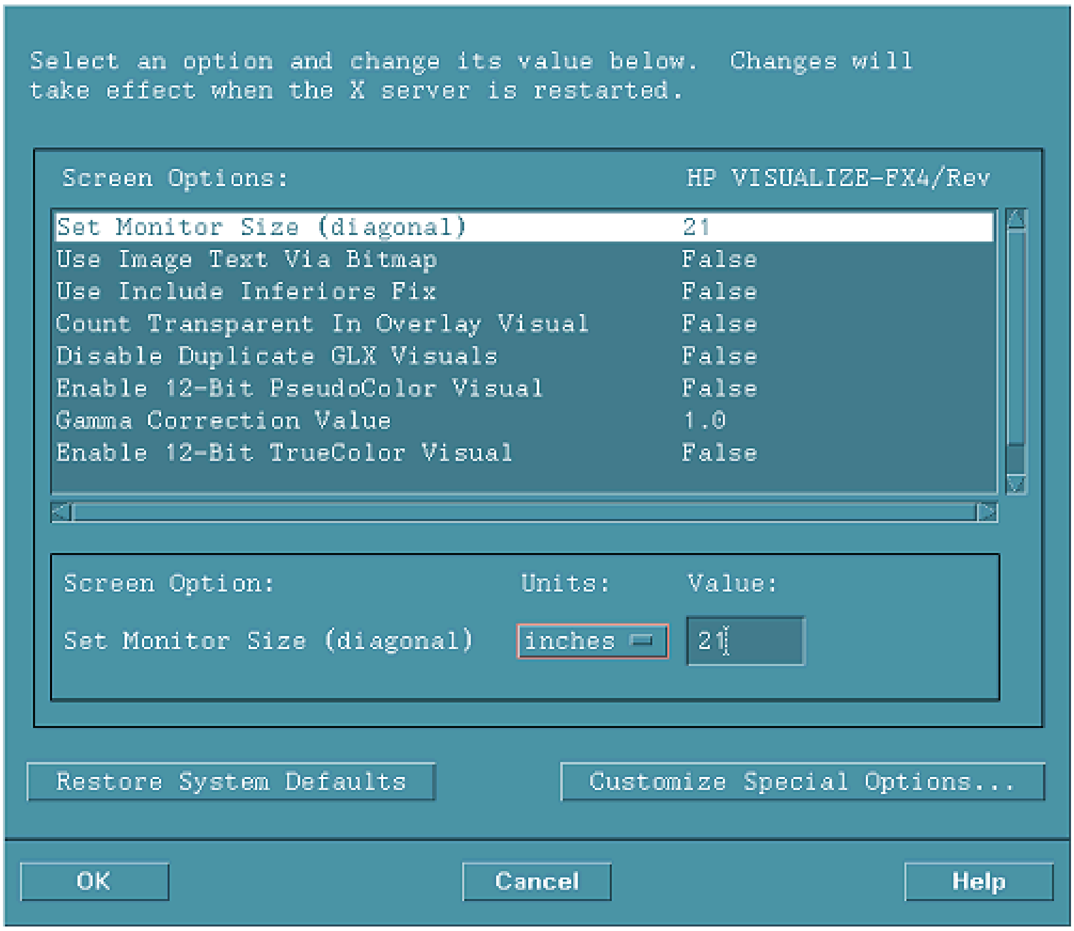
=> **X Server Configuration** 

Select your graphics card, and from the **Actions** pull-down menu select:

Modify Screen Options



Set the monitor size to the required dimensions, and follow the instructions about restarting.



On Compaq OSF 4/5 systems

Edit the /usr/lib/X11/Xserver.conf file
Note: On OSF 5 systems I have found that the Xserver.conf file existed in several different places (eg /etc/X11, /var/...). It was clear from the documentation which one it should have been using, and it was equally clear from trial and error that it was using a different one.

I would suggest that you do a "find" on your system to find all occurrences, ie:

```
find / -name Xserver.conf -print
```

and perform the edits given here in each in turn until you find one that works.

Thanks Compaq!

As root edit this file.
Look for the line starting

```
args <
```

Add to the end of the following argument list

```
-dpi nn
```

Where <*nn*> is the screen resolution in dots per inch.

For example on my machine these lines now read:

```
args <

-pn -nice 2 -wm -dpi 90

>
```

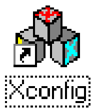
To establish the <nn> value in dots per inch:

- Find the screen resolution from xdpyinfo if unknown (eg 1280 x 1024)
- Measure the vertical and horizontal visible dimensions in inches with a ruler (eg 15" x 12")
- Calculate the average (here of 1280/15 and 1024/12) to get a "dots per inch" value (here 85)

Finally logout of the console, taking it right back to command line mode, then allow the X server to restart the graphical login with the new properties.

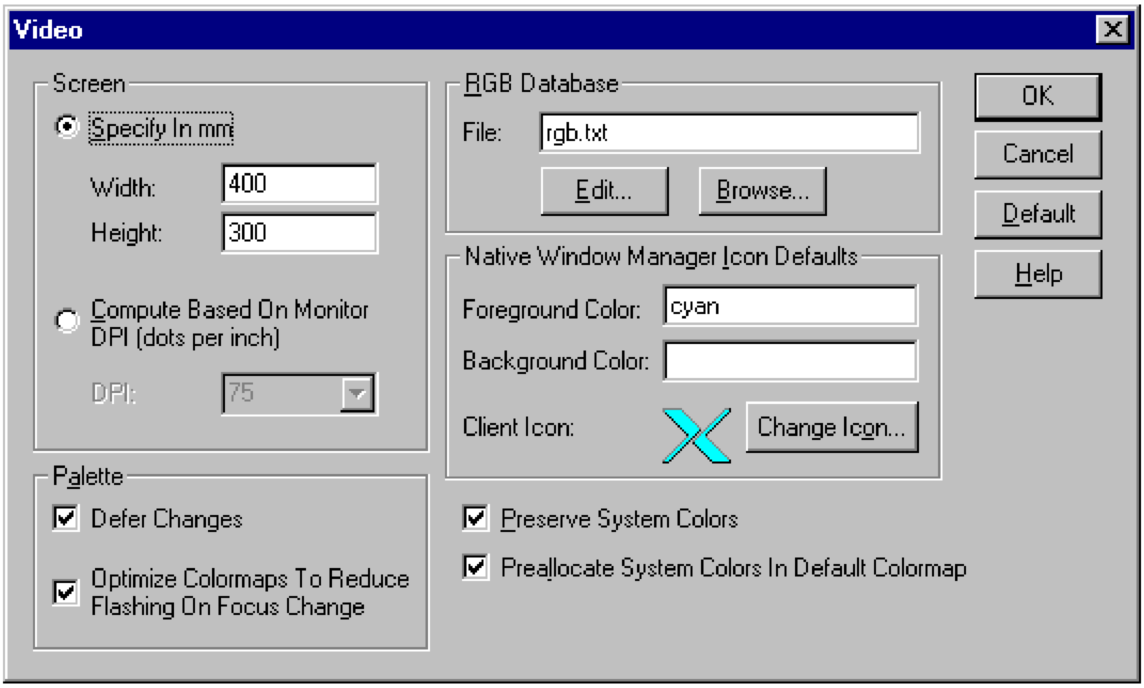
On PCs running Windows using the Exceed emulator.

From the main Exceed directory use



=>

=>



In the **Video** panel set the measured (this time in mm) screen dimensions.

Here my (21") display is 400x300mm.

You will need to restart Exceed to make these settings take effect.

Installation organisation

The version 10 installation can be customised to try and avoid a number of issues that often occur in large organisations with many users.

- Large organisations generally imply large networks, and it is often the case that the performance of these networks can be intermittent or poor, therefore it is common practice to perform an installation of the software on the local disk of each machine, rather then having a single installation on a remote disk.

  This avoids the pauses and glitches that can occur when running executable files over a network, but it also means that all the configuration files in, or depending upon, the top level "Admin" directory have to be copied to all machines and, more to the point, any changes or additions to such files also have to be copied to all machines.

- In larger organisations the "one person per computer" philosophy may not apply, with the consequence that users will tend to have a floating home area on a network drive and may not use the same machine every day.

  This is not usually a problem on Unix/Linux where the "home" directory is tied to the login name not the machine. However on Windows platforms it means that %USERPROFILE%, which is typically on the local C drive of a machine, is not a good place to consider as "home" since it will be tied to a given computer, therefore a user who saves a file in his home directory on machine A may not be able to access it from machine B.

- In a similar vein placing large temporary files on the /tmp partition (Unix/Linux) or the C: drive (Windows) may result in local disks becoming too full, or quotas exceeded.

This section gives only a brief summary of the installation organisation, and you should refer to the separate Installation Guide if you want to find out more about the details of installation, licensing, and other related issues.

Version 10.0 Installation structure

In version 10.0 the option is provided to separate a top-level ’administration’ directory from the ’installation’ one where the executables are located.

For large installations on many machines this allows central configuration and administration files to exist in one place only, but executables to be installed locally on users’ machines to give better performance. Version 10.0 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`** =10 for release 10.0, thus **`OA_ADMIN_10`**) Admin level oa_pref file Other configuration files Timeout configuration file	

OA_ADMIN	*Optional*	Same as **OA_ADMIN_10**, 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_10** *is not defined then this non-release specific version is checked.*	
OA_INSTALL_xx	*Optional*	(**xx** =10 for release 10.0, thus **OA_ADMIN_10** All executables Installation level oa_pref file	**oasys*install_dir: <*pathname*>**
OA_INSTALL	*Optional*	Same as **OA_INSTALL_10**. 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_10** *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** (Unix/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** (Unix/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 UNIX/LINUX systems the "oasys_10" script that starts the SHELL automatically sets this Environment Variable and passes it to any application started from the SHELL. If you run applications directly from the command line and bypass the SHELL then you should set **OA_INSTALL_XX** so that the software can locate manuals and other required files.

OA_ADMIN_XX

Users wishing to separate configuration and installation directories will be able to do so by making use of the new top level **OA_ADMIN_xx** directory.

Installation Examples

The following diagrams illustrate how the installation might be organised in various different scenarios..

a) Single user installation on one machine

There is no need to worry about separating administration and installation directories, and the default installation of all files in and below the single installation directory will suffice.

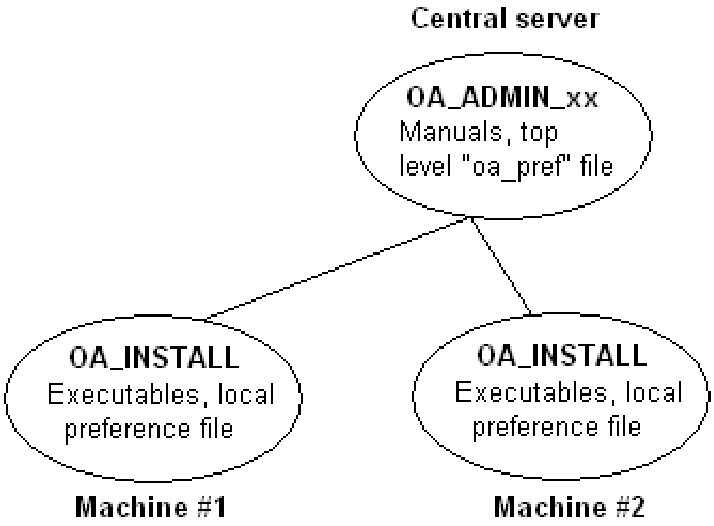
It is suggested that the `_xx` version suffix of `OA_INSTALL_xx` is used in order to keep parallel installations of different releases of the Oassys Ltd software separate on the machine.

b) A few machines on a small network, each user has his own machine

The top level administration directory can be installed on a network server, possibly also locating the manuals centrally.

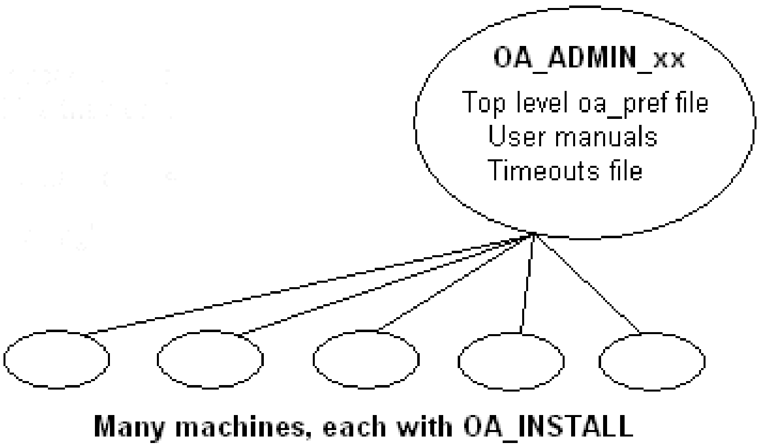
Each user's machine has its own 'installation' directory to give good performance, but there is no need to manage home or temporary directories centrally since each user 'owns' his machine.

If network performance is good an alternative would be to install executables on the central server, meaning that local OA_INSTALL directories are not required.



c) Large corporate network

There is no need to worry about separating administration and installation directories, and the default installation of all files in and below the single installation directory will suffice.



Dynamic configuration using the top level oa_pref file.

A further improvement is that all environment variables below **`OA_ADMIN_xx`** may either be set explicitly, or dynamically using the options in the oa_pref file at the top **`OA_ADMIN_xx`** level. This permits parallel installations of different versions of the software to co-exist, with only the top level administration directory names being distinct. For example:

Release 10.0	**Release 10.1**
Top level directory **`OA_ADMIN_10`**	Top level directory **`OA_ADMIN_101`**
oa_pref file in **`OA_ADMIN_10`** contains: **`oasys*install_dir:`** *<pathname for **10.0** installation>* **`oasys*manuals_dir:`** *<pathname for **10.0** manuals>* **`oasys*home_dir:`** *<pathname for home directory>* **`oasys*temp_dir:`** *<pathname for temporary files>*	oa_pref file in **`OA_ADMIN_101`** contains: **`oasys*install_dir:`** *<pathname for **10.1** installation>* **`oasys*manuals_dir:`** *<pathname for**10.1** 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.