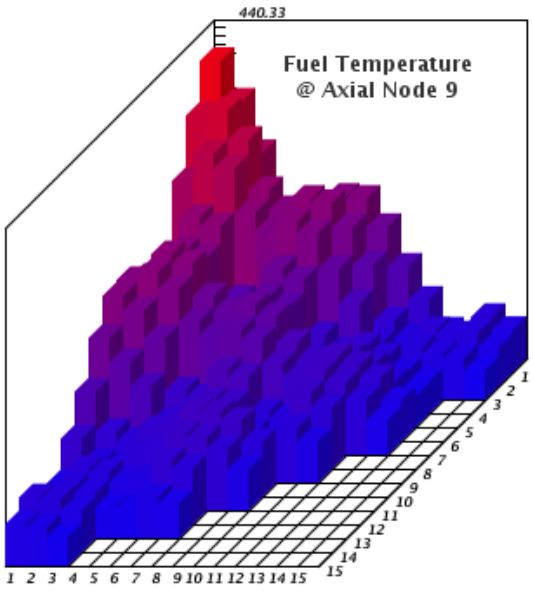
Symbolic Nuclear Analysis Package (SNAP)

User's Manual



Version 2.2.1 - October 25 2012

Applied Programming Technology, Inc.

240 Market St., Suite 208
Bloomsburg PA 17815-1951

Symbolic Nuclear Analysis Package (SNAP) User's Manual

Applied Programming Technology, Inc.

Table of Contents

Chapter 1. Introduction

The Symbolic Nuclear Analysis Package (SNAP) consists of a suite of integrated applications designed to simplify the process of performing engineering analysis. SNAP is built on the Common Application Framework for Engineering Analysis (CAFEAN) which provides a highly flexible framework for creating and editing input for engineering analysis codes as well as extensive functionality for submitting, monitoring, and interacting with the codes.

SNAP currently includes support the CONTAIN, COBRA, FRAPCON-3, MELCOR, PARCS, RADTRAD, RELAP5 and TRACE analysis codes. Each code is supported by a separate plug-in. The list of currently installed plug-ins can be found in the Chapter 2, *The Model Editor* **About** dialog by pressing the **Plugins** button.

The SNAP Application Suite

The SNAP application suite includes the Model Editor, JobStatus and the Configuration Tool client applications as well as the Calculation Server. In this context, a client application is one that typically is run on the local machine and provides a graphical user interface. A server application is one that runs in the background or on a different computer to provide job processing or access to data.

The Model Editor is the primary SNAP client-side user interface. It is responsible for the development and modification of input models for the supported analysis codes. The Model Editor is also responsible for animating the results of those analyses using the Animation plug-in. In addition, the Model Editor is responsible for the creation and design of job streams: arbitrarily complex application flows that can be run on a Calculation Server or High Performance Computing (HPC) environment.

The Configuration Tool is used to specify properties for the SNAP client applications as well as to startup, shutdown and configure a local Calculation Server. Job Status is used to display the status of jobs on a server, import local data files as new jobs, and in general monitor job stream execution. The Calculation Server provides control of and communication with active and completed calculations and streams.

Chapter 2. The Model Editor

The Model Editor is the primary SNAP client application. With it, a user may both design input for analysis codes, display animated representations of calculation results, and build and submit job streams (see Chapter 5, *Job Streams*). The Model Editor provides a consistent interface regardless of the analysis code in question.

2.1. The Model Editor User Interface

SNAP's Model Editor user interface is illustrated in Figure 2.1, "Model Editor UI". The primary components are labeled and a brief description of each follows.

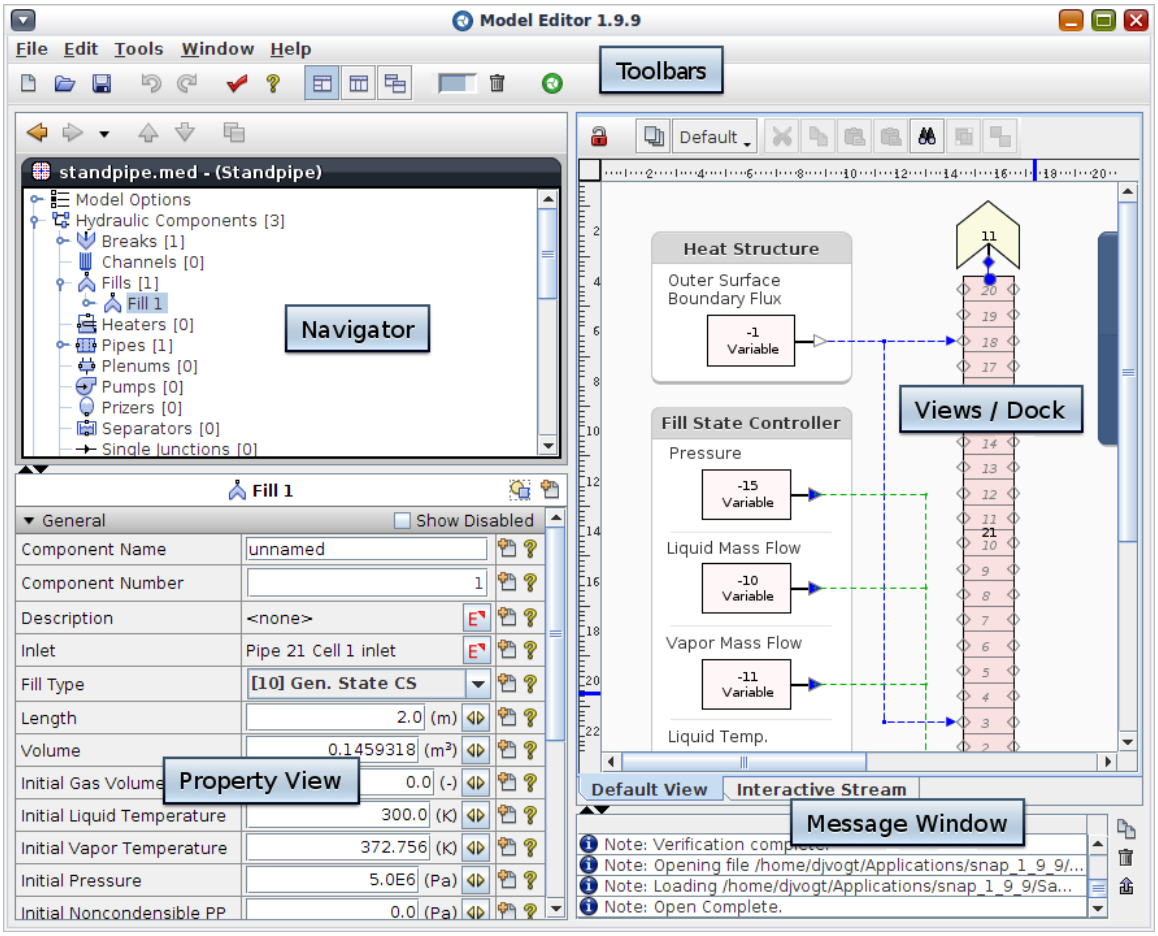


Figure 2.1. Model Editor UI

2.1.1. The Main Toolbars

The main toolbars include basic file operations, model operations, and undo/redo (see Figure 2.2, "Model Editor Main Toolbar"). These toolbars can be individually enabled or disabled using a right-click pop-up menu located on the main toolbar panel.

Buttons for the basic file and model operations include **New**, **Open** and **Save** as well as shortcuts for **Undo**/**Redo**, **Check Model**, and **About**.

The Window Mode buttons can be used to change the basic layout of the major Model Editor components. The first displays the Navigator above the Property View. The second displays the Navigator, Property View, and Views all horizontally aligned, with the Message Window underneath. The third is a multi-window mode, where each major component receives its own window.

The memory toolbar displays information on the current memory usage.

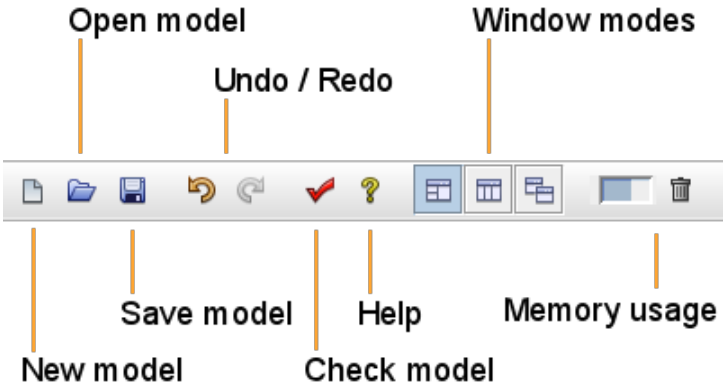


Figure 2.2. Model Editor Main Toolbar

2.1.2. Main Menu Items

The main menu contains several menus of commonly used actions.

The **File** menu contains the following items:

- **New** - Creates a new model (see Section 2.3, “Creating a Model”). The Ctrl-N shortcut (Command-N on Mac OS) is supported on most platforms.
- **Open** - Opens a model (see Section 2.3.1, “Opening a New Model”). The Ctrl-O shortcut (Command-O on Mac OS) is supported on most platforms.
- **Open Recent** - Opens a recently used model, broken down into sub-menus by plug-in.
- **Save** - Saves the model (see Saving a Model). The Ctrl-S shortcut (Command-S on Mac OS) is supported on most platforms.
- **Save As** - Saves the model with a new name (see Saving a Model). The Ctrl-Shift-S shortcut (Command-Shift-S on Mac OS) is supported on most platforms.
- **Close** - Closes the current model, removing it from the Navigator. If the model has unsaved changes, a prompt will be displayed asking whether the model should be saved.
- **Close All** - Closes all open models. If any of the models have unsaved changes, a prompt will be displayed asking which models should be saved.
- **Import** - Imports a model. This provides a menu of the file formats that can be imported by the currently loaded plug-ins. In the case of COBRA, the **COBRA ASCII** item will create a new COBRA model and import the given file as COBRA input.

- **Export** - Exports the current model. This menu lists the file formats that can be exported by the current model, defined by its plug-in.

- **Exit** - Closes the Model Editor. If any open models have unsaved changes, a prompt will be displayed asking which models should be saved.

The **Edit** menu contains the following items:

- **Undo** - Reverts the last modification made to the current model. Successive undo actions revert additional changes. The Ctrl-Z shortcut (Command-Z on Mac OS) is supported on most platforms.

- **Redo** - Reverts the last undo, effectively restoring the change. An additional redo action becomes available for each undo performed on the model. Making any change to the model clears the list of available redo actions. The Ctrl-Y shortcut (Command-Y on Mac OS) is supported on most platforms.

- **Preferences** - Opens the Model Editor preferences window (see Section 2.2, "User Preferences").

- **Plugin Manager** - Opens a dialog listing all available SNAP plug-ins. Toggling the check-box next to each plug-in allows for loading and unloading the plug-in.

The **Tools** menu contains the following items:

- **Check Model** - The Check Model button and menu item both perform a model validation check on the current model. The checks performed for a model are defined by its plug-in.

- **Submit Job** - A job stream can be submitted by selecting the **Submit Job** menu item from the main **Tools** menu. Refer to Section 5.4, "Submitting Job Streams" for more information on the job submission process.

- **Export to jEdit** -This item exports the current model to the jEdit text editor. Refer to Section 3.3.1, "General Properties" for more information about configuring jEdit for use by the Model Editor.

- **Configuration Tool** - Opens the Configuration Tool SNAP application (see Chapter 3, *The Configuration Tool*).

- **Job Status** - Opens the Job Status SNAP application (see Chapter 4, *Job Status*).

- **Model Note Viewer** - Opens the Model Note Viewer (see Section 2.4, "Model Notes").

The **Window** menu contains one item, **Commands**, that opens a **Commands** dialog, which can be used to process Model Editor batch commands (see Section 2.9, "Batch Command Syntax"). When other Model Editor windows are open, this menu will also list those dialogs. Selecting an open window menu item will bring that window to the foreground.

The **Help** menu contains the following items:

- **Help Contents** - Opens the SNAP User's Manual as an interactive help system.

- **About** - Displays information about the Model Editor, installed SNAP plug-ins, included technologies, contact information, etc..

2.1.3. The Navigator

The Navigator is used to select models and access their contents (see Figure 2.3, “Example Navigator View”). The Navigator can be said to consist of three major components: the toolbar, the accordion, and the model contents. Each of these is discussed below.

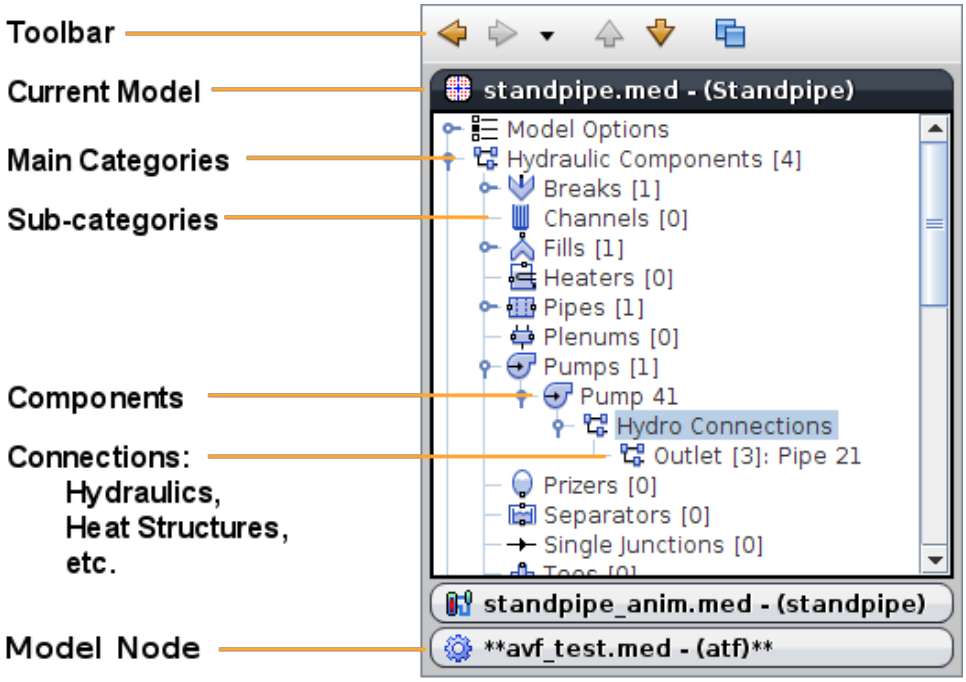


Figure 2.3. Example Navigator View

The *model contents* provides a logical, hierarchical representation of the current model's components and views. Each model is broken down into categories of components. The majority of these categories are plug-in specific, however some (such as views) are shared between all plug-ins. All of the model's components, including non-visual elements such as Model Options, are accessible from the Navigator.

The Accordion

The *accordion* organizes models and displays their contents. It consists of everything below the Navigator toolbar. In the example figure, the Model Nodes labelled **W4Loop.med**, **typpwr_anim.med**, and **avf_trace.med** each represent an open model. Model nodes can be expanded to display their contents (**W4Loop.med** in the figure), or contracted (**typpwr_anim.med** and **avf_trace.med**).

When a model is expanded, at most two models to either side are displayed as contracted model nodes. Other models are hidden from view until the expansion changes. Clicking a collapsed node expands it. Double-clicking an expanded node contracts it. A single click on an expanded node displays a model panel in the Property View if the plug-in defines such a panel. The constantly expanding and contracting behavior of model nodes is where the term "accordion" originates.

Exactly one model can be expanded at a time. This expanded node is the *current model*. Contracting the current model with a double-click does not clear its status as the current model. Instead, collapsing the current model provides a complete view of all open models.

Note Only those 2D Views associated with the current model are displayed in the *Views* area. When a view is undocked, it remains visible regardless of which model is selected. However, activating an undocked view selects its parent model in the Navigator.

Navigator Toolbar

The *Navigator toolbar* provides access to several navigation shortcuts. These are, from left-to-right:

1. **Go Back** - Every time the selection changes in the *model contents* changes, a step is added to a selection history. This button returns the selection to the most-recent step in the history, moving the current selection into a forward history. This can change the current model if the previous selection is not in the selected model.

2. **Go Forward** - Similar **Go Back**, but in the other direction.

3. **History** - Displays a pop-up with the last several selections in the Navigator selection history. Selecting an item restores that selection, moving all more recent selections into the forward selection history. Right-clicking on either the Go Back or Go Forward buttons brings up a similar list for the indicated direction.

4. **Up** - Expands the model above the current model.

5. **Down** - Expands the model below the current model.

6. **List** - Displays a pop-up menu listing all open menus. Selecting a model in the pop-up expands it.

Other Navigator Tools

A right-click on nodes in the Navigator opens a pop-up menu. Items in this menu can be used to add, delete or edit components, as well as to perform operations on the model.

The **Show ASCII** item will open an ASCII View showing what the corresponding model's ASCII deck would look like for the selected component. This non-editable view is handy for those analysts who are familiar with ASCII decks and want to verify that they are creating viable ASCII input for their codes.

Note The number of components within a category or sub-category is displayed in brackets following the node name. Notice in Figure 2.3, “Example Navigator View” that there are 4 hydraulic components, and one of these components is a pump.

Note A model's name will appear surrounded by a pair of stars if that model has changes that need to be saved. Notice that in Figure 2.3, “Example Navigator View” that the model **avf_test.med** has unsaved changes while **standpipe.med** and **standpipe_anim.med** do not.

The **Ctrl-Tab** shortcut key can be used to quickly switch between models. Press and hold **Ctrl**, then press **Tab**, to display a model selection pop-up, as shown in Figure 2.4, “Model Selection Pop-up”. Repeatedly pressing **Tab** will cycle through the history of selected models. When the **Ctrl** key is released, the highlighted model is expanded in the accordion.

The list in the model selection pop-up is not displayed in the same order in which the models were opened, but rather the order in which they were most recently selected. Furthermore, the second entry in the list is selected when the **Ctrl-Tab** shortcut is pressed, so that the two most recently used models can be quickly swapped.

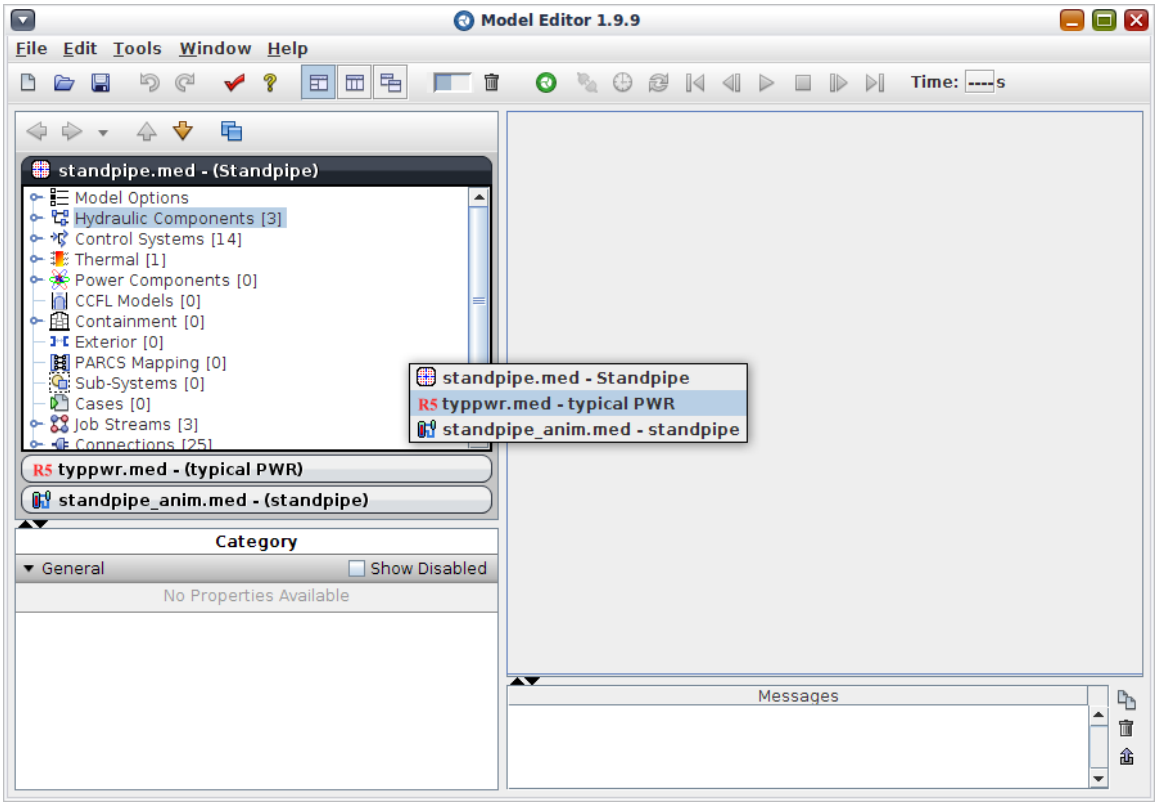


Figure 2.4. Model Selection Pop-up

2.1.4. The Main Property View

The Main Property View, shown in Figure 2.5, “Example Property View”, provides the central point for viewing and editing properties of model components, Display Beans, etc.. It displays the properties of the current selection from either the Navigator or a 2D View. Changes to these properties will immediately be reflected in all other open views (2D, ASCII, Property, etc.). An example of a Property View is shown in Figure 2.5, “Example Property View”.

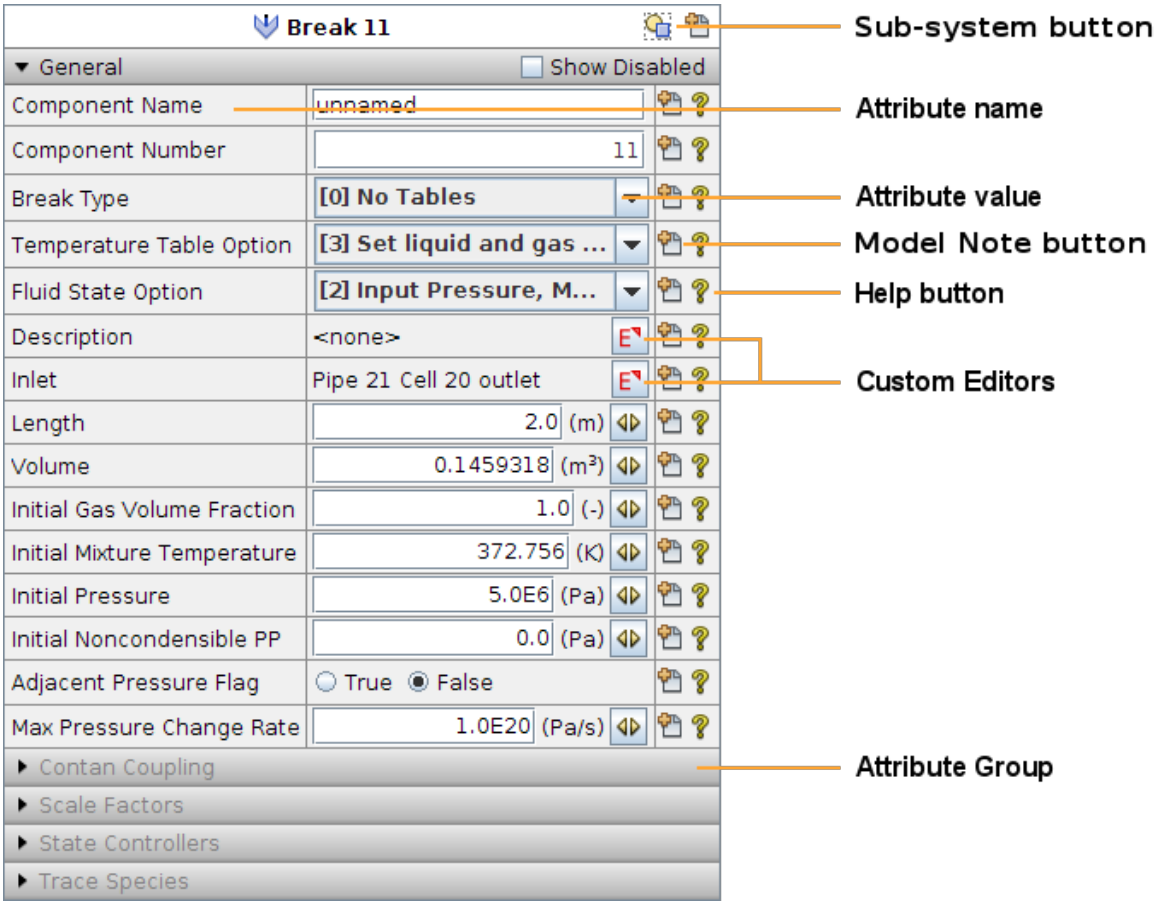


Figure 2.5. Example Property View

Attribute Groups are used to organize the properties of a component within the Property View. Clicking anywhere in the Attribute Group title area will expand or collapse the list of attributes. Each group has a 2 column table of the attributes (description and value). Every object has a **General** attribute group as its first section; other objects add groups as needed.

The **Sub-system Button** opens a pop-up menu containing items related to sub-systems for the select component(s). If the component has been placed into a sub-system, this item can be used to select the parent sub-system in the Navigator. Otherwise, the component can be placed into an existing sub-system from the menu. Sub-systems are explained in greater detail in Section 2.7, “Sub-System Integration”.

Help Button - A help button appears next to each property. Pressing this button will display a more detailed description of the attribute, as shown in Figure 2.6, “Sample Attribute Description”.

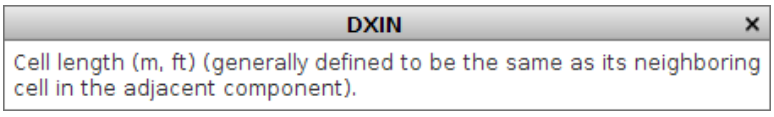


Figure 2.6. Sample Attribute Description

Show Disabled Check Box - This check box (shown in Figure 2.5, “Example Property View”) is used to activate the display of disabled properties in each of the attribute groups. The **Show Disabled** check box always appears in the title area for the **General** attribute group.

Note Disabled properties do not appear by default in the property view. Enabling and disabling properties is the means by which the Model Editor hides attributes of a component that are not relevant given its current configuration.

Custom Editor (also referred to as the **Edit** button) and Component Selection (also referred to as **Select**) buttons are located adjacent to some attribute values. These buttons open detailed custom dialogs for editing components and for selecting other model components where appropriate. These icons are used throughout the SNAP to indicate that pressing the button will open another dialog to either edit or select a value.

Note A separate Property View can be opened for individual components by double-clicking on the component in either the Navigator or a 2D View. The same separate Property View can also be opened by selecting the **Properties** item from the right-click pop-up menu of the component in either location.

The size and content of custom dialogs varies between plug-ins and components. In general, however, they are either "Modal" dialogs whose changes can be cancelled without affecting the mode or "Non-Modal" dialogs whose changes affect the model immediately. The simplest way to distinguish between the two is the buttons that appear at the bottom. If the dialog includes an **OK** and a **Cancel** button then it is "Modal" and its changes will not be applied to the model until the **OK** button is pressed. Most "Non-modal" dialogs will instead have a **Close** button that simply closes the dialog. Some very complex "Non-modal" dialogs will not include a close button at all but will rely on the close button provided by the Operating System (Windows) or Window Manager (UNIX).

2.1.5. 2D Views

2D Views provide a mechanism to visualize models. Components can be placed on a view and arranged in logical ways. Once placed, creating and editing connections between components in a view is as simple as a few clicks with the Connection Tool. Views also support various annotations, such as text, simple graphic shapes, and complex polygons, allowing the user to build an image that represents the model of interest clearly and appealingly (see Section 2.3.4, “Creating Annotations” for more information on view annotations).

A 2D View can be embedded in other views to allow the user to drill-down into more complex parts of the model shown in other views. Once added to another view, these embedded views will be represented by the **Display Icon** specified in the view properties. The **Add To View** menu in the 2D View's right-click pop-up menu is used to embed a 2D View into another 2D View.

2.1.5.1. View Toolbars

The view toolbars located above the 2D View include the main view toolbar, a toolbox toolbar, and optional plug-in specific toolbars used to insert components into the view. The main view toolbar contains buttons for cut, copy, paste, paste special, find, group, and ungroup operations. Paste special can be used to paste multiple copies of a copied set (refer to Figure 2.7, “2D View Toolbars”). The **Lock View** button is used to lock the views in the model. Locking a view prevents the manipulation of its contents.

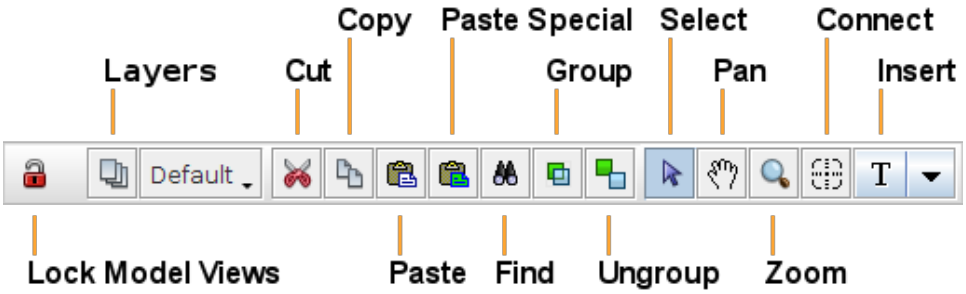


Figure 2.7. 2D View Toolbars

The toolbox toolbar contains buttons used for manipulating the View in various ways. These include the following:

Layers Button

The Layers button opens the Layer Manager, described in Section 2.1.5.4, “2D View Layers”. The second button can be used to select the current default layer.

Select Tool

The select tool is used to select, move, and manipulate elements of a View. Components may be selected using the left mouse button. Clicking an unselected element while holding Ctrl will add it to the selection; clicking a selected element while holding Ctrl will remove it from the selection. The Model Editor also supports "rubber band" selection: pressing the mouse button in an empty area of the View, then dragging the cursor, will draw a rubber band used to select a group of components.

Pan Tool

The pan tool is used to change the visible portion of a View. This feature can be used to maneuver around a large model by pressing the left mouse button and dragging. The displayed portion of the model will move with the mouse.

Zoom Tool

The zoom tool changes the position and scale of the View. Clicking in the View will zoom in a set amount; holding the shift-key and clicking will zoom out the same fixed amount. Clicking and dragging to select a region (drawing a box) will zoom in to the selected region. Right-clicking in the View with the zoom tool will show a menu for selecting a specific zoom position or fitting the entire View to the window.

Connect Tool

This tool is used to create connections between components. To connect two components, place the pointer over a **from** connection point and click the left mouse button, then move the pointer to the desired **to** connection point on another component and left click again. The mouse pointer will turn into a blue dot when a viable **to** connection point is passed over. A line representing the connection will then be drawn in the View.

Insert Tool

The insert tool includes a button to activate the tool and a drop down menu to select the type of annotation or component to insert. Once a type has been selected, left-clicking anywhere in the view will create an object of the selected type. After the insertion is complete the insertion tool will be deactivated.

To insert multiple components of the same type hold the **Ctrl** key during the insertion click. This will keep the insertion tool active and allow multiple inserts.

2.1.5.2. View Menu Items

View menu items can be accessed via the right-click pop-up menu of the View and the menu bar for undocked views. The actual pop-up menu displayed will depend on the item type (and quantity) currently selected in the view. The specific code plug-ins dictate what items the right-click menus contain, however certain menu items appear frequently and are described here. Many of the menu items available from the right-click menu in the 2D view perform identical functions to those in the Navigator.

- **Hide Menubar / Show Menubar** - When a view is undocked, its menubar can be hidden with the **Hide Menubar** item. Once hidden, the right-click pop-up menu for the view will display the **Show Menubar** item to restore the menu.

- **Properties** - This item opens a separate property view for the selected object. This separate property view is functionally equivalent to the main property view shown in Figure 2.5, “Example Property View”.

- **Show ASCII** - The **Show ASCII** menu item will open a window showing what the corresponding model's ASCII deck would look like for the selected component. This view is handy for analysts who are familiar with ASCII decks and want to verify that they are creating viable ASCII input for their codes.

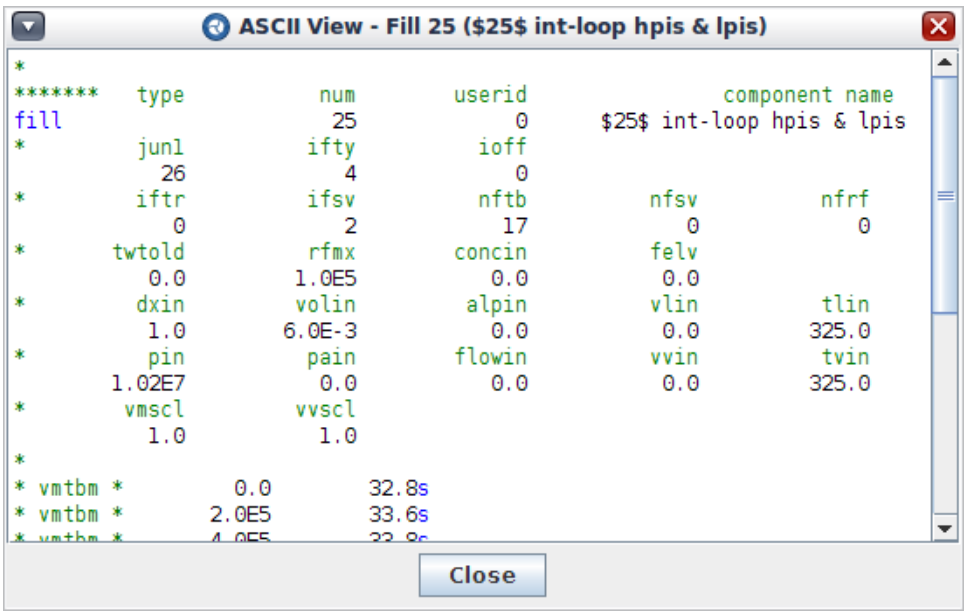


Figure 2.8. ASCII View

- **Reference Docs** - The **Reference Docs** menu item opens the analysis code manual in another window, scrolled to the relevant portion for the selected component.

- **Copy** - Copies the selected objects to the clipboard. Whenever a new cut or copy operation is performed, the copy/paste buffer's contents is overwritten with the new cut/copy object and the old object is lost. The **Ctrl-C** keyboard shortcut (Command-C on Mac OS) is supported on most platforms.

- **Cut** - Copies the selected objects to the clipboard and then removes these objects from the view. This process *does not remove components from the model* but merely removes the graphical representation of those components from the view. The **Ctrl-X** keyboard shortcut (**Command-X** on Mac OS) is supported on most platforms.

- **Paste** - Pastes the graphical items (not the model components) previously copied to the clipboard. The **Ctrl-V** keyboard shortcut (**Command-V** on Mac OS) is supported on most platforms.

  Note A component can only be represented in a view by a single drawn component.

- **Paste Special** - Opens the Paste Special dialog to allow duplicates of the copied *model components* to be pasted. This dialog is a plug-in specific feature, however, most current plug-ins allow pasting multiple copies and renumbering the components as they are pasted. The **Shift-Ctrl-V** keyboard shortcut (**Shift-Command-V** on Mac OS) is supported on most platforms.

- **Group / Ungroup** - The grouping feature of 2D views allows the user to create a "visual group" for a set of selected visual objects (annotations, display beans, etc.). Once created, a visual group is treated as a single object in the view. Selecting any part of the group will select the entire group. Moving, cutting, copying, etc. affects the entire group. Groups can be dissolved at any time by pressing the **Ungroup** button.

  Note Groups can themselves be grouped. When placed into another group, they retain their composition. For example, if two components and a group are combined into another group, performing an ungroup action will yield two components and a group.

- **Delete** - This item removes the currently selected item(s) from the model. A confirmation dialog is presented before the item is deleted. The **Delete** keyboard shortcut is supported on most platforms.

  Note Deleted objects are not saved on the clipboard.

- **Align** - The Align sub-menu contains items can be used to line up objects in a view by the objects' top  , bottom  , left  , or right  faces, as well as by their horizontal  or vertical  centers.

- **To Front / To Back** - These items move the selected object either to the "front" or "back" of the view, relative to other components. **To Front** ensures that no other objects will appear over the selected beans. **To Back** ensures that the selected objects will not appear over others.

- **Scale Drawing** - This item allows the user to change the default scaling of a drawn component. Drawn components may be scaled by length, width, or both. Adjusting the sliders in the Scale

Drawing dialog will cause the on screen display of the object to reflect the new scale. The user may also choose to enter the scaling factors by hand in the available text boxes. This scaling only affects the visual display of the component, the underlying geometry data is unchanged.

Note In some cases it may be more convenient to use the 2D View scaling described in Section 2.3.5, “Component Display Scaling”.

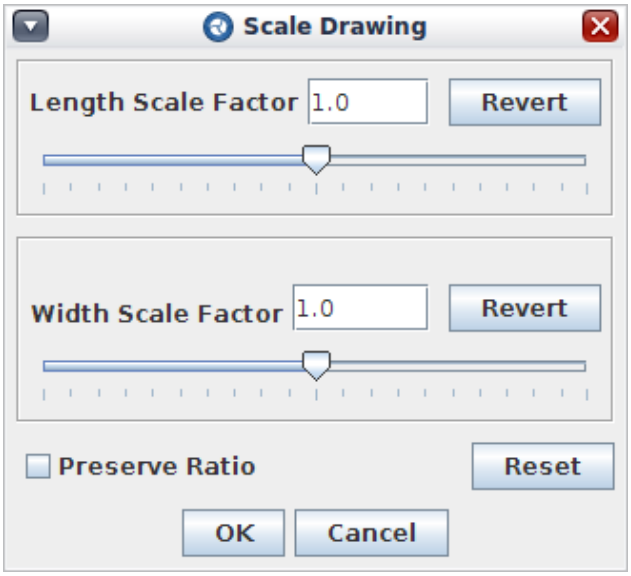


Figure 2.9. Scale Drawing Dialog

- **Organize** - If more then one object is selected, the **Organize** menu item may be used. This item will apply the code plug-in's layout algorithm to arrange the selected items in a visually clear way. Drawing organization is a plug-in specific operation that may not be supported.

- **Redraw** - Sometimes the graphical views' display is not automatically updated when a change to the model is made in another view. This item will cause the current view to be redrawn to reflect the current state of the model and its components.

- **Print View Menu** - The items in this menu allow the view to be printed. The **Entire View** item will print the entirety of the view, regardless of the current zoom scale or pan location. The **Current Perspective** item will print the only the visible portion of the view using the current zoom scale and pan location.

- **Export Image Menu** - The items in this menu are used to export the view as an image. The following formats are currently supported: JPEG, PDF, PNG, SVG, TIFF. Items are provided to export either the **Entire View** or the **Current Perspective**, similar to the Print View Menu above.

- **Select Menu** - The select menu can be used to select a sub-set of the objects in a view by type. Items are provided in the sub-menus for selecting every type that can be added to the view.

- **Zoom Menu** - This menu includes items for setting the current zoom to either **Fit To View** or the following preset scales: **%10**, **%25**, **%50**, **%75**, **%100**, **%150**, **%200**.

- **Undock View** - Undocking a view removes it from the Model Editor main window and opens it in a separate window. This feature can be especially useful for presentations, as the undocked

view can then be maximized to fill the entire screen. Closing and re-opening the view will return it to the docked state.

- **Close** - Closes the current view. The view can be re-opened by selecting the **Open** item in the right-click pop-up menu of the view's node in the Navigator, or by double-clicking it.

2.1.5.3. View Tools Menu

The **Tools** menu of the view's right-click pop-up contains a set of useful operations that are used less often. The following are some of the more frequently used menu items in the **Tools** menu.

Note Plug-ins can add items not described here to the Tools menu.

- **Add / Remove Components** - This item will open a dialog that allows the user to choose from a list of components in the model, shown in the left hand column, to add to the view. The user may remove components from the view with this dialog. Components are added to the view by selecting them in the **Components** list then pressing the >> button, and vice versa.

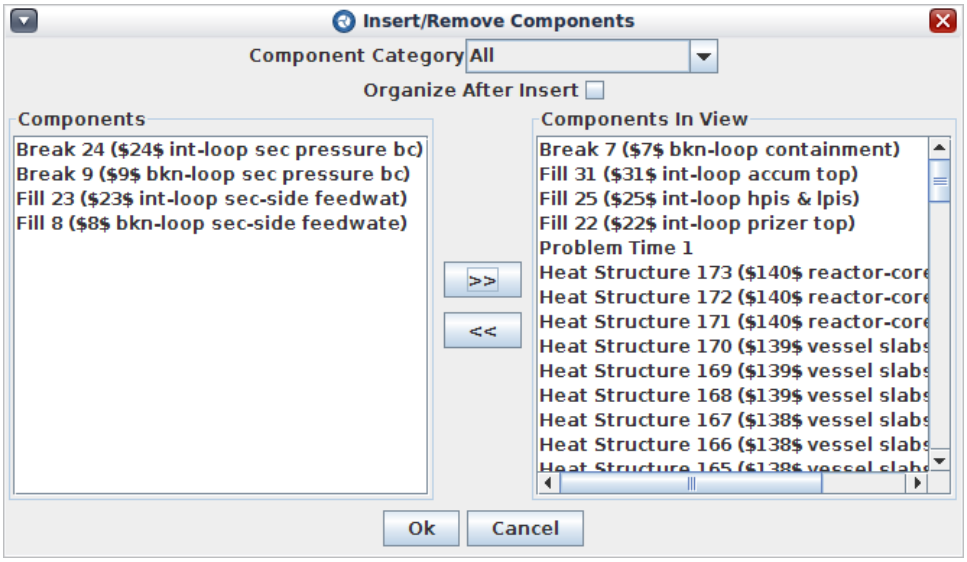


Figure 2.10. Add / Remove Components Dialog

- **Show All Connections** - When components are added to a view, drawn connections are created to represent connections between the components. The drawn connections can be **Cut** from the display to simplify the view. **Show All Connections** will restore any drawn connections missing from the view.

- **Reset Connections** - This will cause all of the selected components connections to be "reset". A connection that has been reset discards any user-specified modifications to the connection in favor of the default path.

- **Import / Export View Template** - These items allow view templates to be imported and exported for a view. A View Template is a file that contains the annotations, component locations, etc. required to reproduce a view inside of another, similar model. This allows for easy duplication of detailed 2D views between models. View templates also allow preserving complex views when a model must be modified outside of the Model Editor and re-imported.

- **Trim Excess Canvas** - This action removes the unused canvas space inside a view. Trimming the canvas is provided so that a view can be set to a very large "working area" while designing the view, then quickly resized to the appropriate dimensions when complete.

- **Export to jEdit** - This item exports the current model to the jEdit text editor. Refer to Section 3.3.1, “General Properties” for more information about configuring jEdit for use by the Model Editor.

2.1.5.4. 2D View Layers

Layers are provided as a tool for organizing and manipulating the contents of a 2D View. Layers are essentially a named collection of zero or more display elements, with each layer belonging solely to its parent 2D View. Each layer may be individually hidden or locked. Hiding layers is a simple visibility toggle: the components in a hidden layer are not drawn in the view. Layer locks function similarly to view locks: the contents of a locked layer cannot be selected in the view. Hiding and locking layers is provided to selectively reduce clutter and complexity in heavily-populated views.

Note Unlike the layers common to graphics applications, view layers do not affect the stacking, or "depth", of elements in a view. Moving an element between layers will not affect its display relative to other components. This design allows the fluid movement of display elements between layers based solely on role and logical grouping. As a side effect, this allows for non-destructive layer sorting. Layers are automatically sorted by name to ease layer management.

All 2D Views contain a 'Default' layer which cannot be removed or renamed. Additional layers can be added using the Layer Manager, available from the **Layers** button on the 2D View toolbar (see Figure 2.11, “The Layer Manager Window”, described below). Layers can also be created by using the **Move to Layer** -> **New** menu item in the right-click pop-up menu of a View.

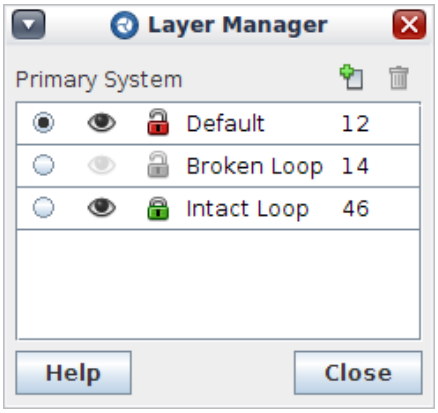


Figure 2.11. The Layer Manager Window

As shown in the figure, the **Layer Manager** is where most layer activity occurs. Layers may be added with the **Create Layer** button at the top of the dialog, and selected layers may be removed with the **Remove Layer** button. The central list details each layer with visual, interactive status indicators. The radio button on the left of each entry controls which layer is "active": all new elements added to the view are automatically associated with the active layer. The **visibility**

button (in the form of an eye icon), controls whether the layer is hidden. This indicator is a toggle button for layer visibility: a fully opaque eye represents a visible layer, while a transparent, faded eye indicates the layer is hidden. The **locked** indicator is another toggle button, switching to and from a locked and unlocked status. The fourth element displays the name of the layer, which can be edited with a double-click. The fifth and final portion of each entry is a non-interactive tally of the elements in the layer.

Once an element is placed in the view, it is placed in the currently active layer. To change the layer to which one or more display elements belong, select the elements in the View and open the right-click pop-up menu. Select **Move to Layer -> <Layer name>** to move the selected elements to that layer. The drop-down next to the **Layers** button in the 2D View toolbar can be used to select the active layer.

2.1.6. The Message Window

The Message Window displays a running list of error, warning, alert, and notice messages. Processes such as saving a file or checking a model will produce messages in this window. Buttons located along the right side of this window are used to clear the window, export selected messages to a file, or copy selected messages to the clipboard.

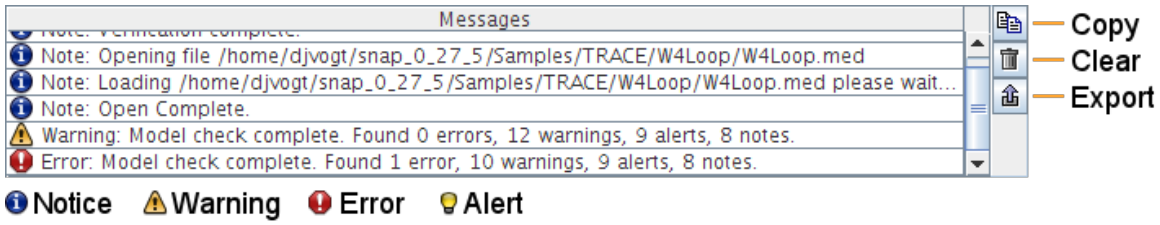


Figure 2.12. Message Window

2.1.7. Mini Navigator

The Mini Navigator provides a detailed look at selected model components. The navigator portion of the editor lists the selected component and any associated connections. The lower portion of the editor provides a editable Property View of the component selected in the upper navigator view. The Mini-Navigator Property View provides the same editors and help documentation as the main Property View.

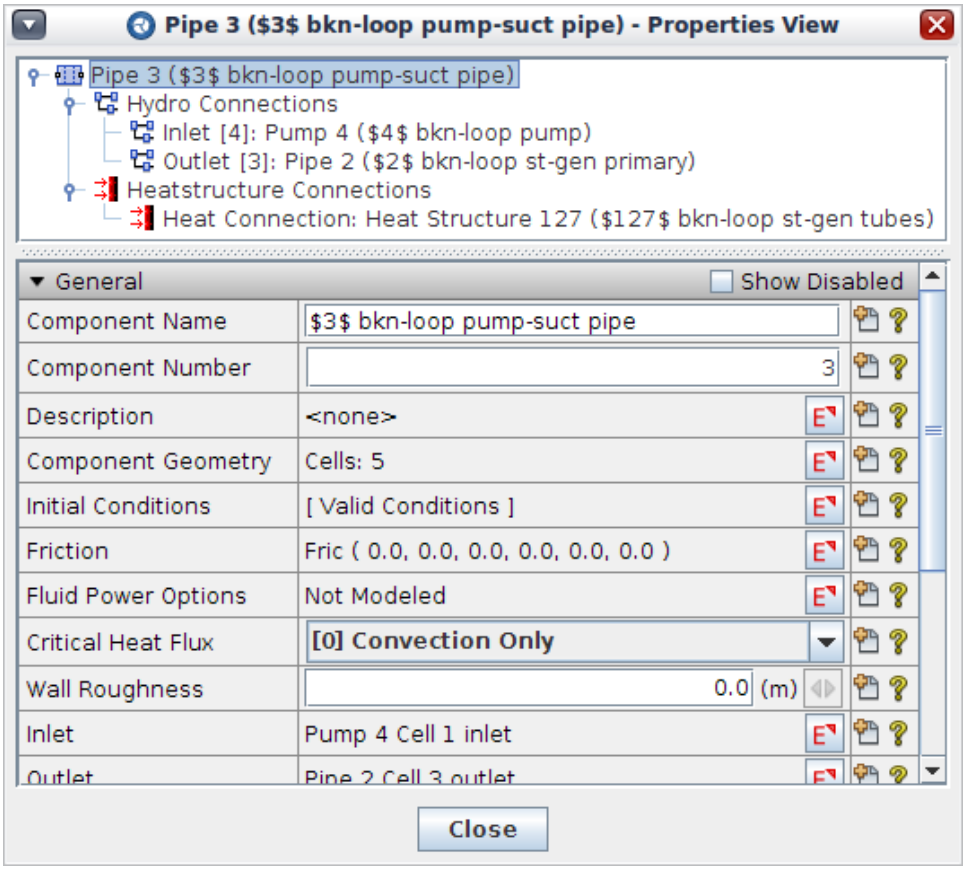


Figure 2.13. Mini-Navigator

2.1.8. The Component Differencing Utility

The component differencing utility provides a means of comparing the ASCII output of two component selections. Component selections can vary in size, ranging from a single component to entire models, where all components of two models can be compared side-by-side.

2.1.8.. Comparing Single Component Selections

The component differencing utility provides two methods for comparing selections of single components; the target components view menu, and the right-click menu of a component's navigator node. Choosing the **Select Left Side to Compare** item adds the selected component to the left side of a difference viewer. If a component has already been added to the left side of a comparison, the **Compare to...** menu item will add the currently selected component to the right side of a difference viewer. The differences between the ASCII output of the components will be calculated, and a new Component Difference Viewer will be displayed, as shown in Figure 2.14, “Component Difference Viewer”.

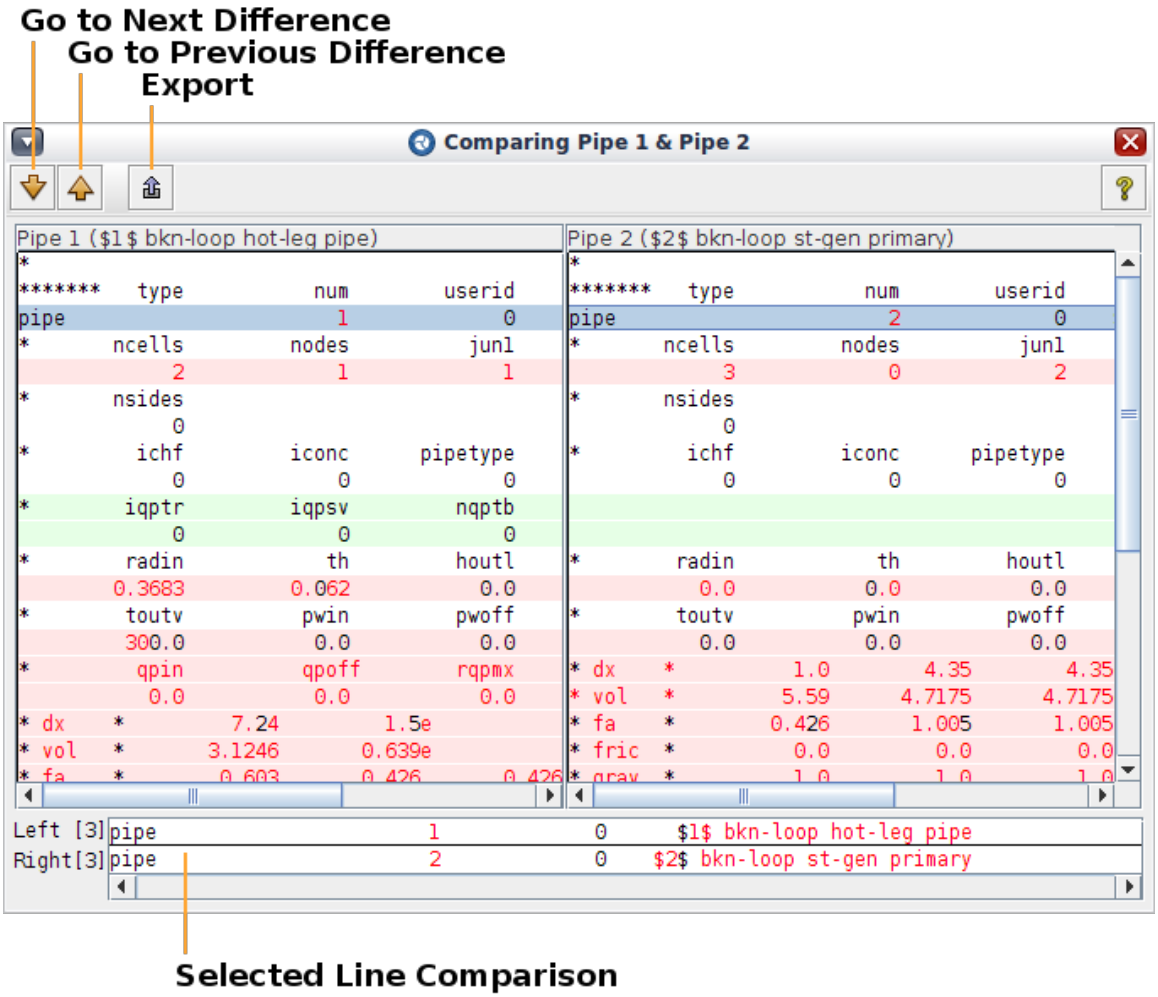


Figure 2.14. Component Difference Viewer

The Component Difference Viewer displays the ASCII decks of two selected components side-by-side, with special formatting added to each side to indicate the detailed differences between the sources. Lines which are identical in both sources are reproduced in black text with a white background. When the line is present in both sides but has differences, the background color is set to light red. The specific characters which are different are displayed in a red font. Added and deleted lines have their backgrounds colored light blue and light green, respectively. To ensure that the source lines "line up" properly in the difference viewer window, blank lines are inserted in certain locations of the sources. These added blank lines have gray backgrounds and are not part of the component's actual ASCII output: they are included to enhance the visibility of the differencing utility.

The Component Difference Viewer provides buttons for navigating through the differences. Clicking on one of these buttons causes the viewer to navigate to and select the first line of the next (or previous) contiguous set of differences. Additionally, an export button is included which allows the user to export the differences between the two sources to an external text file. The export output format is similar to that of the GNU diff utility.

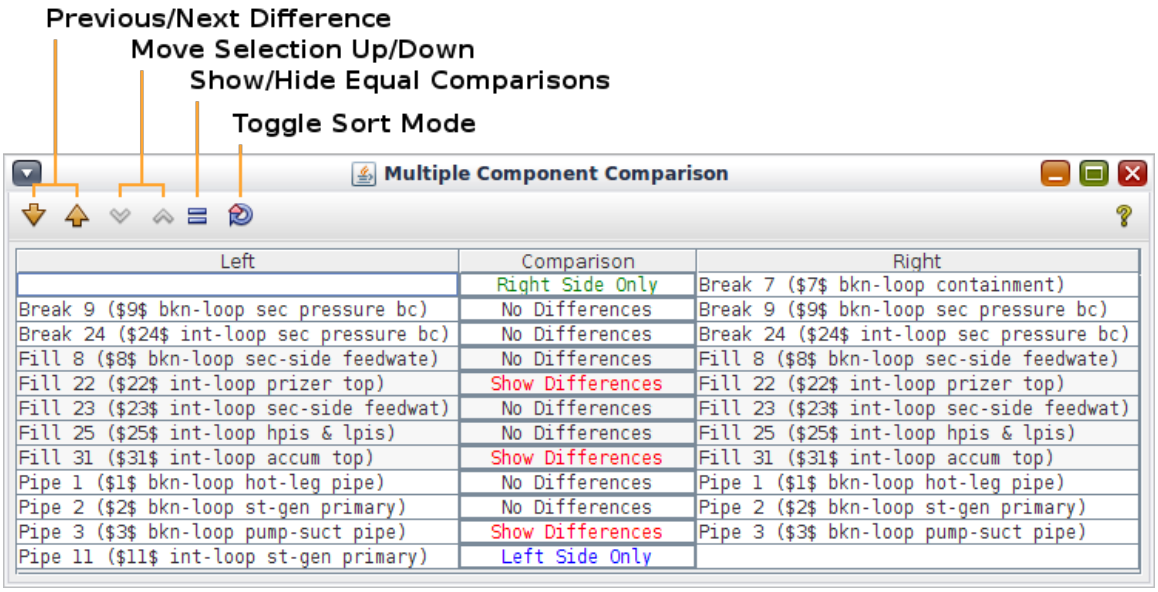
The bottom panel of the Component Difference Viewer contains a vertical comparison of the left and right source lines at the currently selected row. Character-specific differences between the two lines are colored red.

When a component is compared to itself, the left side of the Difference Viewer caches the current ASCII lines of the component. If the component is changed from within the Model Editor, the right side of the Difference Viewer updates its ASCII lines to reflect the new state of the component's output. This allows the user to view the differences made to a component's ASCII output by changing a certain value in the Model Editor.

2.1.8.2. Comparing Multiple Component Selections

The Component Differencing Utility can also be used to compare differences between two selections of multiple components. Initialization of the Multiple Component Comparison Window can be accessed by either the view menu for a multiple selection, the right-click menu for a selection of multiple navigator nodes, or the right-click menu of a model navigator node. Once left and right selections have been defined, a new Multiple Component Comparison Window is provided.

Note Multiple component comparisons can only be performed on component selections from the same plug-in type.



Left	Comparison	Right
	Right Side Only	Break 7 (7 bkn-loop containment)
Break 9 (9 bkn-loop sec pressure bc)	No Differences	Break 9 (9 bkn-loop sec pressure bc)
Break 24 (24 int-loop sec pressure bc)	No Differences	Break 24 (24 int-loop sec pressure bc)
Fill 8 (8 bkn-loop sec-side feedwate)	No Differences	Fill 8 (8 bkn-loop sec-side feedwate)
Fill 22 (22 int-loop prizer top)	Show Differences	Fill 22 (22 int-loop prizer top)
Fill 23 (23 int-loop sec-side feedwat)	No Differences	Fill 23 (23 int-loop sec-side feedwat)
Fill 25 (25 int-loop hpis & lpis)	No Differences	Fill 25 (25 int-loop hpis & lpis)
Fill 31 (31 int-loop accum top)	Show Differences	Fill 31 (31 int-loop accum top)
Pipe 1 (1 bkn-loop hot-leg pipe)	No Differences	Pipe 1 (1 bkn-loop hot-leg pipe)
Pipe 2 (2 bkn-loop st-gen primary)	No Differences	Pipe 2 (2 bkn-loop st-gen primary)
Pipe 3 (3 bkn-loop pump-suct pipe)	Show Differences	Pipe 3 (3 bkn-loop pump-suct pipe)
Pipe 11 (11 int-loop st-gen primary)	Left Side Only	

Figure 2.15. Multiple Component Comparison Window

The Multiple Component Comparison Window consists of two sets of selected components (the left side and right side) and a group of results comparison buttons. Initially, the components of each selection are sorted by category name and component number. Components with the same category name and number are horizontally aligned in the comparison window. If a component exists on one side and not the other, then it is aligned with a blank line. Components can also be sorted by category alone by clicking the **Toggle Sort Mode** button. When components are sorted by category alone, components can be aligned on the same row as components with different component numbers, and all required blank lines are added at the end of each group of components that share the same category. The background of the table cells of components in every other category group are tinted.

The comparison buttons indicate the differences between components on the same row. If a component is aligned with a blank, then the comparison button indicates which selection the

component belongs to, and clicking on that button launches a Component Difference Viewer with text only on one side. If a component is aligned with another component, then the comparison button displays whether the components contain differences in their ASCII output, and clicking the comparison button will display an appropriate Single Component Difference Window. The **Show/Hide Equal Comparisons** button controls whether or not comparisons without differences are displayed in the table.

The ordering of components within each selection can be modified manually by using the **Move** buttons. Clicking a **Move** button will reposition the selected components up or down a row. Comparisons between the newly-aligned components are computed, and the Difference Viewer Launch Buttons are updated accordingly. **Move** operations are restricted to the bounds of each selected component's category group. For example, in the figure above, a break cannot be moved down to be compared to a fill. The move buttons are only enabled when the sort mode is set to sort by category and component number.

When a Multiple Component Comparison Window is created, the left selected components are cached. If a component contained in the left selection is modified from the Model Editor, the changes will not be reflected in the left side of the Multiple Component Comparison Window (or any Single Component Difference Viewer initialized from the Multi-Component Window). Conversely, if a component exists in the right selection, and is changed from the Model Editor, the Multiple Component Comparison Window (and any resulting Single Component Difference Viewers) will update to reflect the changes. This functionality allows the user to compare different "versions" of the same components and models, and can be a useful tool for learning how changing values in the Model Editor interface impacts a model's ASCII deck.

2.2. User Preferences

The SNAP Model Editor user preferences are available from the main **Edit** menu. These options are presented as a set of General Preferences (such as **Handle Size**) used by all plug-ins in addition to plug-in specific configuration.

The General Preferences are divided into two Attribute Groups: **General** and **Colors**. These are described in more detail below.

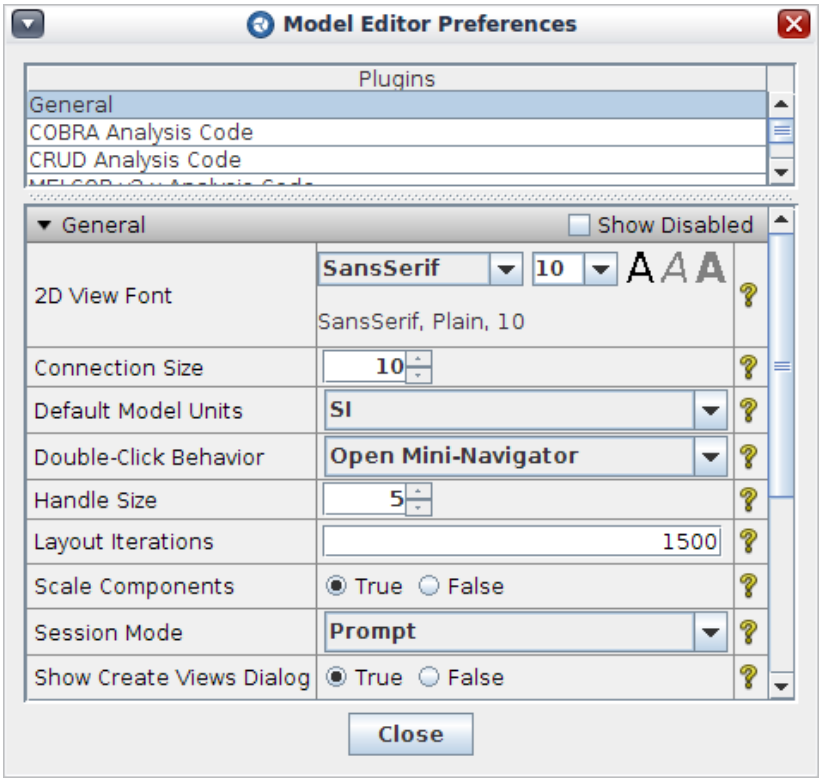


Figure 2.16. Model Editor Preferences Dialog

2.2.1. General Preferences

Preferences in the General category refer to a number of miscellaneous preferences, some of which shall be applied to all plug-ins.

- **2D View Font** - This value sets the font used in automatic text annotations in 2D views, such as the CC numbers displayed in the middle of most components.

- **Connection Size** - This value sets the size of the connection points, which defaults to 10 pixels. Connection points are graphic shapes that indicate where a connection to a component can be made. Different shapes imply different connection semantics on a plug-in specific basis. For instance, in the TRACE plug-in, a circle is used to indicate a point that may be used to start a connection. A diamond is used to indicate a point that may be used to end a connection. Refer to Figure 2.17, “Connection points used in the TRACE plug-in”.

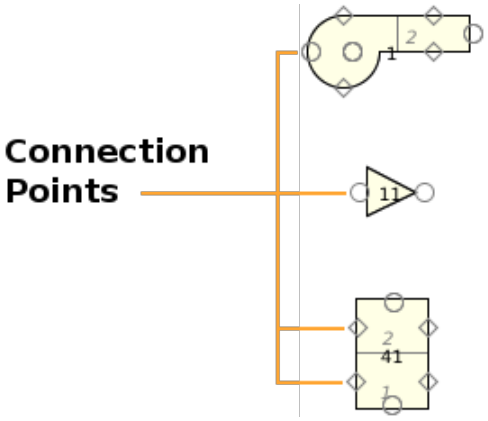


Figure 2.17. Connection points used in the TRACE plug-in

- **Default Model Units** - Selects whether models default to using SI or British units.
- **Double Click Behavior** - When a user double clicks an object in a view, the Model Editor will either select the object in the Navigator view or open a Section 2.1.7, “Mini Navigator” view. This option controls which behavior will occur.
- **Handle Size** - This value sets the size of “handles,” or the boxes that are drawn in the selection frame. It defaults to **5**. See Figure 2.18, “Handles and Selection Frame”.

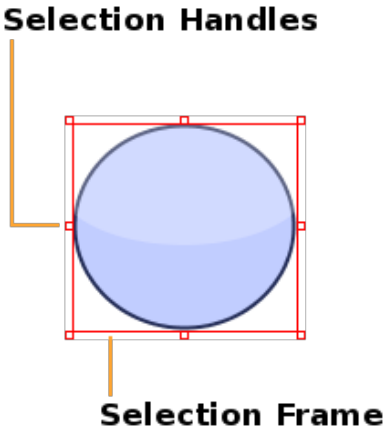


Figure 2.18. Handles and Selection Frame

- **Layout Iterations** - A user may choose to “organize” a model (or selection of the model) in a graphical view. In short, organize operations attempt to reposition the components on the canvas in a logical and “clean” fashion. The iteration number entered here determines how much time should be spent on this process. Generally, higher numbers produce better layouts, but cause the organization process to take longer. The default value is 1500.

  Note The Layout Iterations preference is used as a hint for the layout systems included with each plug-in. Thus, the effect of this preference varies from plug-in to plug-in.

- **Scale Components** - When this option is selected, components shown in a graphical view are scaled in size based on the data entered for that component (volume, length, etc.). Because small components can become difficult to see with much larger components, the user may choose to turn off scaling when components are of drastically different size.

- **Show Create Views Dialog** - When set to true, a Create Views dialog will be displayed for each model imported into the Model Editor.
- **Show Welcome Dialog** - This option controls the appearance of the welcome dialog at the Model Editor startup.
- **View Tab Layout** - Determines the display mode for view tabs that take more horizontal space than allocated to the view area. **Scroll Tabs** provides a scrolling mechanism that hides buttons until scrolled into the viewable area, where **Wrap Tabs** displays all tabs across several rows.
- **Temporary Folder** - This optional property can be used to specify the location in which temporary files are generated, particularly when submitting job streams.
- **Always Use Anti-Aliasing** - When set to true, all drawings in 2D Views will be anti-aliased (some drawings are always anti-aliased). Anti-aliasing greatly improves the appearance of most drawings by smoothing curves and rough edges. Anti-aliasing incurs a performance cost, so turning this off may result in faster 2D View scrolling and repaints.
- **View Selection Model** - Determines how selections made in the Navigator affect 2D Views. **Independent**, the default, states that Navigator selections do not effect views. **Follows Navigator** indicates that a selection made in the Navigator causes that component to be selected in the View.
- **Xpdf Executable** - Sets the location of the xpdf executable used to display PDFs on UNIX systems. This option is not available on Windows installations of SNAP.

2.2.2. Color

This section lists the different colors assigned to various components inside 2D views. There are two ways to modify a color preference. The first is to click on the box displaying the color as it would appear and select a new color from the drop down menu. Second, a user can click on the text field to the left of the color box, which displays the color as three integers under the RGB color scheme. Changing these numbers and pressing Enter will change the color to the corresponding color in the RGB scheme.

Note Color preferences are used as hints for plug-in 2D View rendering and are not supported by all plug-ins.

2.3. Creating a Model

Assisting the creation of models is the primary purpose of the Model Editor. Models are typically classified in one of two categories: pre-processing and post-processing. A pre-processing model is generally comprised of components used to form the basis of a calculation. Alternatively, a post-processing model is used to display either the results of a calculation or some other form of data; in SNAP, this usually takes the form of an Animation. This section shall focus on functionality common to pre-processing models. Refer to Chapter 6, *The Animation Plugin* for more information about post-processing in SNAP.

Note Pre-processor models are submitted as job streams (see Chapter 5, *Job Streams*).

2.3.1. Opening a New Model

Pressing the **New** button on the main toolbar will open the model type selection dialog as shown in Figure 2.19, “Example New Model Dialog”. This dialog allows the selection of any of the currently loaded plug-ins for a new model. After selecting the desired type a new model will be created and added to the Navigator. A single empty view will also be created and opened.

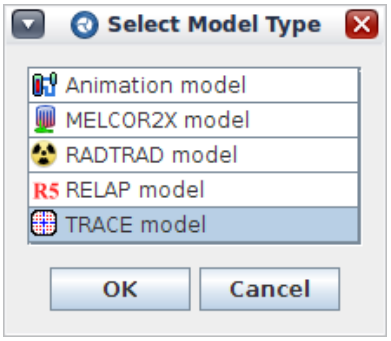


Figure 2.19. Example New Model Dialog

After creating a new model, special attention should be paid to its **Model Options** node in the Navigator, as it contains several important model specific properties (refer to Figure 2.20, “Model Options”).

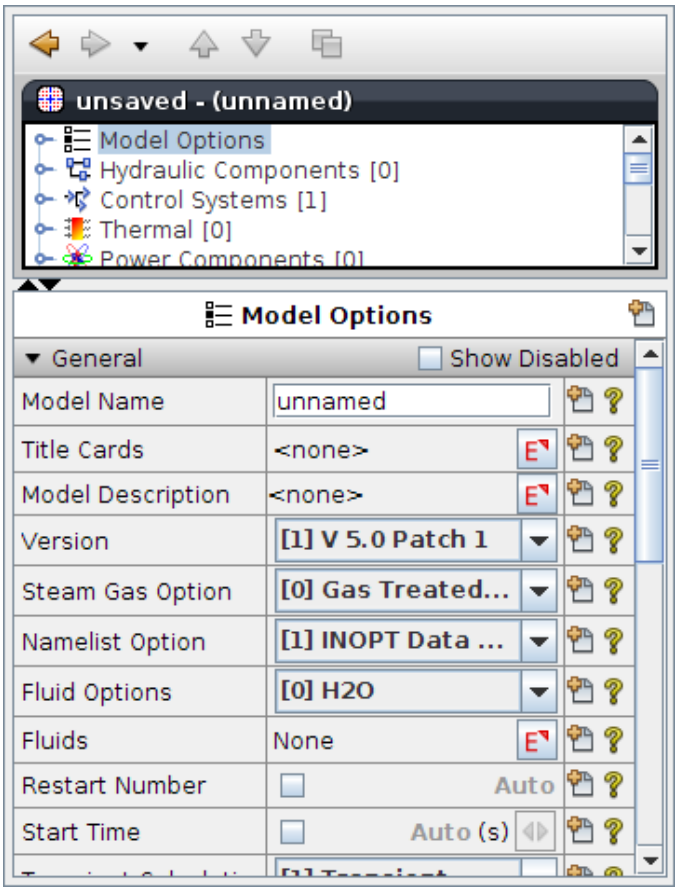


Figure 2.20. Model Options

2.3.2. Creating and Editing Components

SNAP provides several methods of creating and editing components. The most common method is to use the component insertion button located above the 2D View (see Section 2.1.5.1, “View Toolbars”). This button opens a pop-up menu that contains all of the component types organized by category as shown in Figure 2.21, “Adding TRACE Components from the View Toolbar”. As an example, when editing TRACE models, Pipes are located in the **Hydraulic Components** category, while user-defined numerics are located under **Numerics**.

Once the appropriate component type has been selected, the button to the left of the drop down menu button changes to the icon of that component type and the view cursor changes into a cross-hair. A left-click within the 2D View will insert the component. Adding a component may open a completion dialog for the component before it is inserted into the model. The Select Tool will be automatically activated after the insertion is complete to allow moving or resizing the newly created component. Holding the **Ctrl** key when inserting a component will leave the Insert Tool active, allowing multiple components to be created without reactivating the Insert Tool for each.

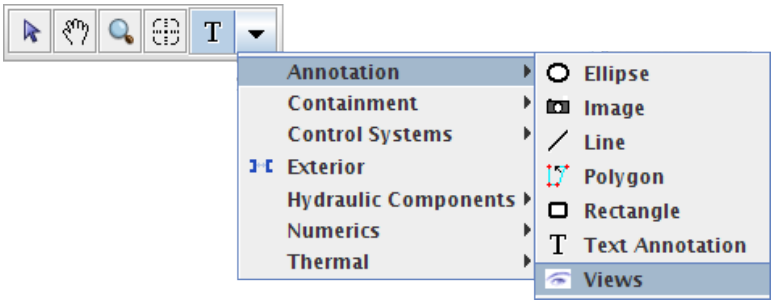


Figure 2.21. Adding TRACE Components from the View Toolbar

The second method of adding a component involves the Navigator, as illustrated for a Fill component within a TRACE model in Figure 2.22, “Adding TRACE Components from the Navigator”. First, expand the component category of the desired component; in the example, this is the **Hydraulic Components** category. Next, right-click on the desired component type to display a pop-up menu and select **New**. This creates a new component of the specified type with a default name.

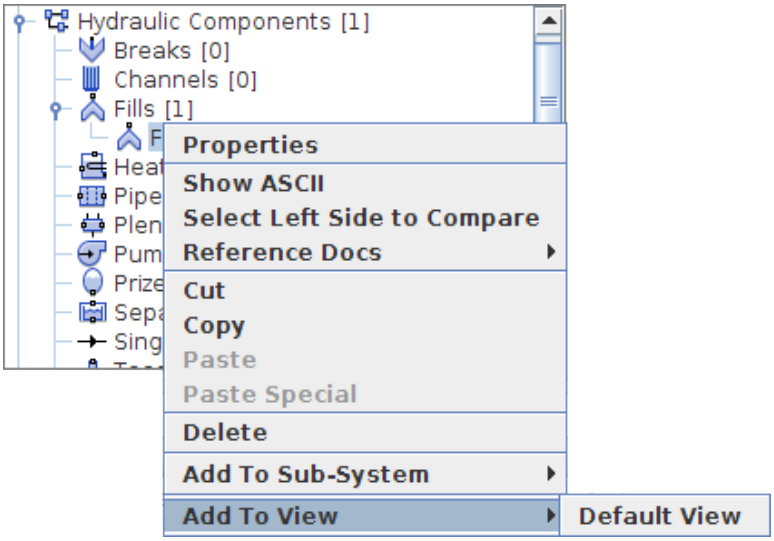


Figure 2.22. Adding TRACE Components from the Navigator

To add the newly created component to a 2D View, right-click on the component in the listing under the component type to display a pop-up menu. From this menu, select the desired View to which the component is added from the **Add to View** sub-menu. The component will be placed within the selected View. Alternatively, the component can be added to a 2D View by clicking and dragging the component from the Navigator into the View.

To edit a component, select the component, either in the 2D View or in the Navigator. The properties for that component will then appear in the Main Property View. See Section 2.1.4, "The Main Property View" for more information on editing properties.

2.3.3. Creating Drawn Connections

Some components placed in the 2D view can be connected via the connection tool. Placing the connecting tool over a valid connection point causes the tool to change into a cross-hair. Clicking on the connection point starts a connection, shown on the screen as a red line drawn from the connection point to the moue cursor. Clicking on another compatible connection point completes the connection between the components. When hovering over the second connection point, the cross-hair will display a blue dot in the cursor. The compatibility of connection points varies between plug-ins.

Once connections between components are constructed, they can be selected and edited in the Property View. The type of connection created depends on the connected components and the points from which the connection was made. A sample drawn connection between two pipes is shown below:

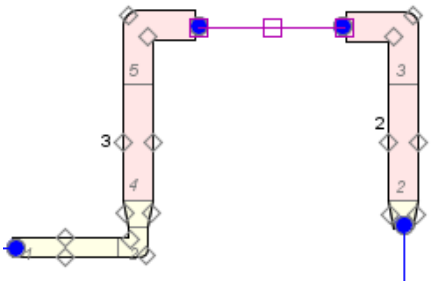


Figure 2.23. Drawn Connection

Drawn Connections have a series of options which are made available by bringing up the right-click pop-up menu on a given connection. These options are as follows:

- **Cut** - This option provides the ability to remove a Drawn Connection from a view. Cutting a connection does not delete the selected connection. The points which are involved in a connection that was cut remain filled with blue to indicate a connection still exists at that point.
- **Reset** - Selecting this option will reset a modified connection path back to its original position.
- **Add/Remove Point** - Drawn Connection paths can be adjusted by dragging adjustment points located on the connection. These points can be added or removed in order to define a connection path appropriately. The position where the right-click occurs determines where the point will be added or removed. A point is only removed if one existed at the position of the mouse click.
- **Straight Line** - Setting this option provides a straight Drawn Connection path. All previously set path adjustment points will be removed.
- **Properties** - This provides a pop-up Property View listing of the specific connection properties. The pop-up Property View is identical to the view provided by clicking on the connection.
- **Disconnect** - Deletes the selected connection.
- **Drawing Style** - The drawing style of a connection is determined by this property. The following describes the available options:
  - **Single Line** - The default drawn connection display.
  - **Source Marker** - A marker which denotes the name of the source of the selected connection.

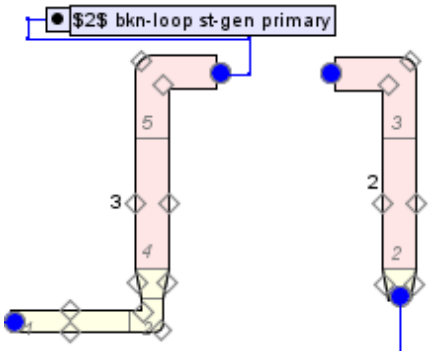


Figure 2.24. Source Marker Connection Style

- **Target Marker** - A marker which denotes the name of the target of the selected connection.

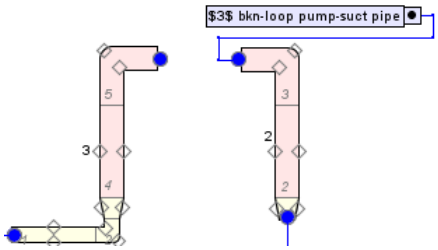


Figure 2.25. Target Marker Connection Style

- **Source and Target** - Two markers which denote the name of the source and target of the selected connection.

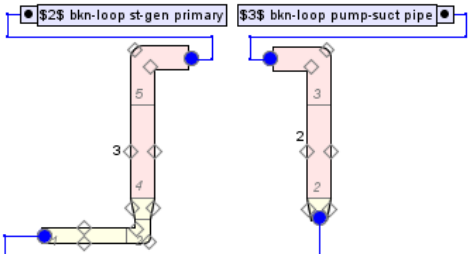


Figure 2.26. Source and Target Connection Style

2.3.4. Creating Annotations

Annotations can be used to mark-up a view with static information about its contents. For example: a rectangular annotation with a titled border can be used behind a group of components to visually separate them from the surrounding components. Annotations are created in a view using the Insertion Tool. All Annotations appear in the Insert Tool drop-down menu under the **Annotations** sub-menu.

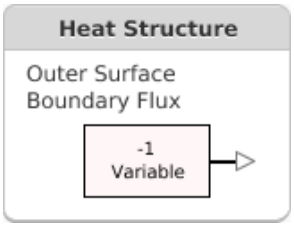


Figure 2.27. Compound Annotation Example

The example shown in Figure 2.27, “Compound Annotation Example” illustrates a complex object built from 2D View annotations. The main body consists of gray and white rectangular polygons with curved segments at the corners. A highlight at the top is created from a single, semi-transparent white line. A duplicate of the lower white polygon, with its outline disabled and its fill color set to a semi-transparent black, comprises the shadow at the bottom. The highlighted "Heat Structure" text is composed of two text annotations with a one pixel difference in placement, one white and one black.

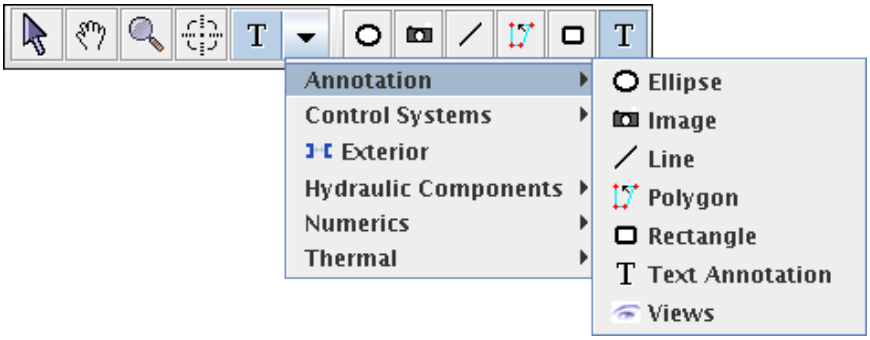


Figure 2.28. Adding Annotations from the View Toolbar

The following types of annotations are included with the Model Editor and available for use in any view. More information about the specific properties of each annotation type can be found in the pop-up help buttons to the right of each property in the Main Properties View.

Note The **Views** item also appears in the Annotation menu for convenience. This item creates an embedded view icon that can be used to *drill-down* to the view it represents.

Ellipse

The ellipse (or Elliptical Annotation) is a very simple rounded shape with an optional filled center. When inserting an ellipse, left-clicking will insert an ellipse of the default size. To insert an ellipse of a specific size, left-click and drag inside a 2D View to create a rubber-band box that is approximately the desired size, then release the mouse button.

The following properties can be specified for an Elliptical Annotation:

- **Height -** The vertical height of the annotation (in pixels).
- **Width -** The horizontal width of the annotation (in pixels).
- **Fill Background -** When this property is set to **True**, the ellipse will paint its center using the **Background Color** specified. When this property is set to **False** the center of the ellipse will be transparent.
- **Background Color -** The color used to fill the center of the ellipse when **Fill Background** is True.
- **Outline Color -** The color used to paint the outline of the ellipse.
- **Line Thickness -** The thickness of the line drawn around the outline of the ellipse.

Image

The image annotation allows an image to be displayed inside a view and it is inserted with a simple left-click. Newly created image annotations will display the default light bulb image. The following properties can be specified for an Image Annotation.

- **Height -** The vertical height of the annotation (in pixels).
- **Width -** The horizontal width of the annotation (in pixels).

- **Border -** An optional border can be specified for an image annotation. Etched, line and beveled borders are available using the provided border editor.
- **Image -** The **Image** property is used to select the image to display. Once selected, this image data will be loaded directly into the annotation and stored with the view.
- **Tooltip Text -** The image annotation includes optional tool-tip text. This text is displayed when hovering the mouse over the image. Some simple HTML tags may be specified for this property, but must be surrounded by <html></html> tags.

Line

Line annotations are solid or dashed lines with optional arrow heads. The line thickness, line color, and relative arrowhead size can all be specified.

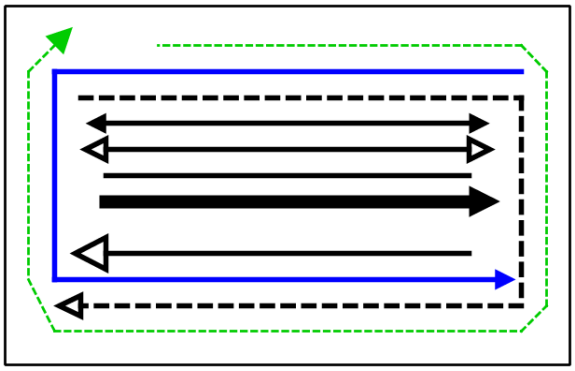


Figure 2.29. Line Annotation Examples

Line annotations are defined by a set of segments. Creating a line a annotation involves defining the anchor points of the line as a series of clicks. The first left-click will begin the line. Moving the mouse will show a rubber-band line drawn to the previous point in the line. Left-clicking again at another location will create a line segment to that location. Additional left clicks will create more line segments. Double-clicking with the left mouse button will complete the last segment and insert the line annotation using the defined segments.

At any point during the insertion process, the right mouse button can be used to remove the previously added point. If there is only a single point defined then this will cancel the line insertion.

Once inserted, the points that make up a line may be relocated by left-clicking and dragging the red rectangular handles around the points. In addition, the right-click pop-up menu of the line includes the **Add Point** and **Remove Point** menu items. The **Add Point** item can be used to add a new relocatable point anywhere on the line. The **Remove Point** item can be used to remove all but the first and last points of a line. Note that the **Remove Point** item is only available when right-clicking on (or very near) a point.

- **Arrow Size -** The size of the annotation's arrow heads relative to the specified line thickness.
- **Dashed -** When this property is **True** the line will be drawn with a dashed pattern.
- **Color -** The color used to draw the line.

- **First / Second Arrow Head -** The style used to draw the line's arrow heads. Each head style can be specified individually. Available styles include None (no head), Filled (a filled triangle) and Hollow (an unfilled triangle).

- **Thickness -** The thickness (in pixels) of the line. This value also affects the size of the arrows on either end of the line.

- **Tooltip Text -** The line annotation includes optional tool-tip text. This text is displayed when hovering the mouse over the annotation. Some simple HTML tags may be specified for this property but must be surrounded by <html></html> tags.

Polygon

The polygon annotation is a 2D closed figure made up of any number of line segments and curves.

Inserting a polygon annotation is a process of defining the line segments that will make up the polygon. The first left-click will begin the polygon. Moving the mouse will show that a rubber-band line is drawn to the previously defined point. Left-clicking again at another location will create a line segment and begin drawing the rubber band line to that location. Additional left clicks will create more line segments. Double-clicking with the left mouse button will complete the last segment, close the shape, and insert the polygon annotation using the defined segments.

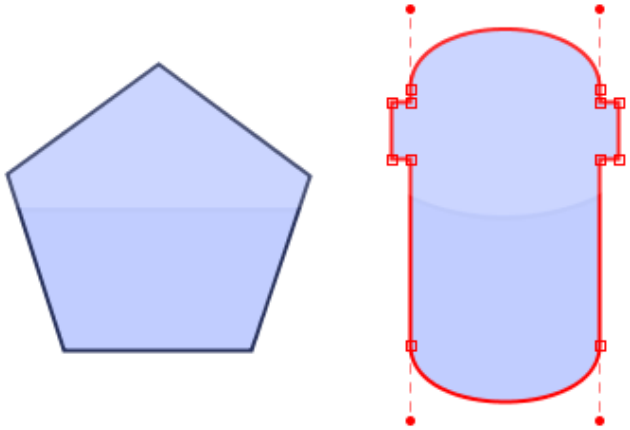


Figure 2.30. Polygon Annotation Examples

At any point during the insertion process, the right mouse button can be used to remove the previously added point. If there is only a single point defined then this will cancel the polygon insertion.

Once inserted, the points that make up a polygon may be relocated by left-clicking and dragging the red rectangular handle around the point. The vertical and horizontal line segments in a polygon may also be relocated by clicking left-clicking and dragging the line. Any curved segments may be left-clicked and dragged to move the entire polygon. Also, left-clicking and dragging with the shift button pressed will drag the polygon regardless of whether lines or points are clicked.

The right-click pop-up menu of a polygon includes the following additional items which are used to manipulate the following figure:

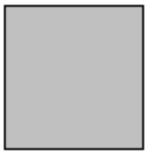


Figure 2.31. 2D View Polygon

- **Flip Horizontal** / **Flip Vertical** - These features are used to horizontally or vertically mirror the points about the center of a polygon.

- **Reshape Polygon** - This feature is used to completely reshape a polygon into a regular polygon with a user specified number of sides and outer radius. The number of sides and outer radius is requested by an input dialog.

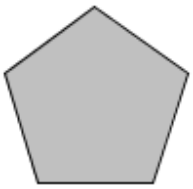


Figure 2.32. Reshaped Polygon

- **Rotate Polygon** - This rotates the polygon points clockwise a user specified number of degrees. The number of degrees is requested by an input dialog.

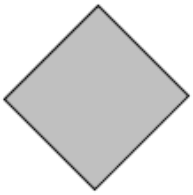


Figure 2.33. Rotated Polygon

- **Scale Polygon** - This feature scales the points of the polygon by a user specified fraction. The scale fraction is requested by an input dialog.

Figure 2.34. Scaled Polygon

- **Curve Segment / Straighten Segment** - Sets the closest segment to be curved if it is currently straight, or straight if it is curved. Curved segments can be manipulated by dragging the control points that appear when the polygon is selected.

- **Hide Segment** - Hides the line segment or curve closest to the mouse cursor.

The following properties can be specified for a Polygon after insertion:

- **Fill Style** - The style used to fill the contents of the polygon: **Unfilled**, **Solid** (color), or **Pattern**. If **Unfilled** is selected the inside of the polygon will be transparent.

- **Fill Color** - The color used to fill the contents of the polygon when the Solid style is selected.

- **Fill Pattern** - The pattern used to fill the contents of the polygon.
- **Pattern Foreground/Background** - The colors used to paint the foreground and background of the pattern filling the polygon.
- **Outline Style** - The style of line that will be used to outline the polygon. If **None** is selected no outline will be painted.
- **Outline Color** - The color used to outline the polygon.

Rectangle

The rectangle annotation creates a rectangle with an optional filled background and border. By default the rectangle has squared corners but can be rounded by setting the **Rounded Corners** property to **True**.

When inserting a rectangle, left clicking will insert a rectangle of the default size. To insert a rectangle of a specific size, left click and drag to create a rubber-band box that is approximately the desired size and release the mouse button.

The following properties can be specified for a rectangular annotation:

- **Height -** The vertical height of the annotation (in pixels).
- **Width -** The horizontal width of the annotation (in pixels).
- **Rounded Corners** - When set to **True** the corners of this rectangular annotation will be rounded using the **Rounded Arc Height** and **Width**.
- **Rounded Arc Height / Width -** The height and width (in pixels) of the arc used to round the corners of the annotation when **Rounded Corners** is true.
- **Fill Background -** When this property is set to **True**, the annotation will paint its center the **Background Color** specified. When this property is set to **False** the center of the annotation will be transparent.
- **Background Color -** The color used to fill the center of the annotation when **Fill Background** is True.
- **Outline Color -** The color used to paint the outline of the annotation.
- **Line Thickness -** The thickness of the line drawn around the outline of the annotation.
- **Border -** An optional border can be specified for a rectangular annotation. Etched, line and beveled borders are available using the provided border editor.

Text

The text annotation is used to label or describe portions of a view and can be inserted with a left-click. Like other annotations, the text annotation can optionally fill its background with a specified color and has an optional border.

The content of a text annotation can be edited directly by double clicking it in the 2D View. This will open a text field for editing the content of the annotation.

The following is a list of the properties of a text annotation:

- **Tooltip Text -** The text annotation includes optional tool-tip text. This text is displayed when hovering the mouse over the annotation. Some simple HTML tags may be specified for this property, but must be surrounded by <html></html> tags.
- **Foreground Color -** The color used to paint the text entered for this annotation. Note that HTML tags specified in the **Text** of this annotation may override this color value.
- **Fill Background -** When this property is set to **True**, the annotation will paint its background the **Background Color** specified. When this property is set to **False**, the center of the annotation will be transparent.
- **Background Color -** The color used to fill the background of the annotation when **Fill Background** is True.
- **Border -** An optional border can be specified for a text annotation. Etched, line and beveled borders are available using the provided border editor.
- **Text Margin** - The amount of space used to surround the text in the Text Annotation.
- **Text -** The text displayed in the text annotation. Figure 2.35, “Text Annotation Example” is an example of a text annotation using the optional HTML tag support.

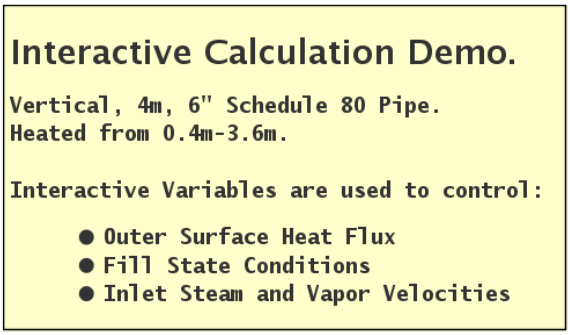


Figure 2.35. Text Annotation Example

HTML tags can be entered directly into the text of a text annotation. These tags will be interpreted by the built-in Java HTML renderer. This allows tables, bulleted lists, font colors, font sizes, etc. to be specified directly in a text annotation. The above example can be seen by entering the following text:

```
<h2>Interactive Calculation Demo.</h2>
<code>
  Vertical, 4m, 6" Schedule 80 Pipe.<p>
  Heated from 0.4m-3.6m.<p><p>
  Interactive Variables are used to control:
  <ul>
    <li>Outer Surface Heat Flux</li>
    <li>Fill State Conditions</li>
    <li>Inlet Steam and Vapor Velocities</li>
  </ul>
</code>
```

Additional Properties

When in a pre-processor model, annotations also provide an **Associated Component** property. This property allows specifying a reference to a model component represented by the annotation. In a locked view, clicking on an annotation will select its associated component in the Navigator.

2.3.5. Component Display Scaling

The scaling of components within a 2D View is controlled by the **Pixels Per Meter** property of the view. The initial default value of 20 pixels per meter is designed to display complete model diagrams for large, full plant models. If view components appear too small to identify key details, the relative size of the components may be increased by adjusting the **Pixels Per Meter** property.

Note To display the View properties within the Main Property View, switch to the Selection Tool and select any point in the view not occupied by a component.

A View's **Width Scale Factor** defines relative width to height scaling for components placed in the view. Adjusting this property will make components appear wider or thinner.

Note The 2D drawing of components can also be scaled individually using the Scale Drawing menu item, as described in Section 2.1.5.2, “View Menu Items”.

2.3.6. Checking a Model for Errors

Once a model has been completed, it can be checked for errors. To do so, select either the **Check Model** button from the main toolbar, or the **Check Model** menu item located under the **Tools** menu. A model report dialog will be displayed, such as in Figure 2.36, “Model Report Window”.

Within this dialog, the toggle buttons near the top can be used to enable/disable the display of the corresponding type of message. The **Refresh** button in the upper right will force another model validation, the results of which will be displayed in the current model report dialog. Finally, the **Export** button can be used to write an HTML copy of the report to a file.

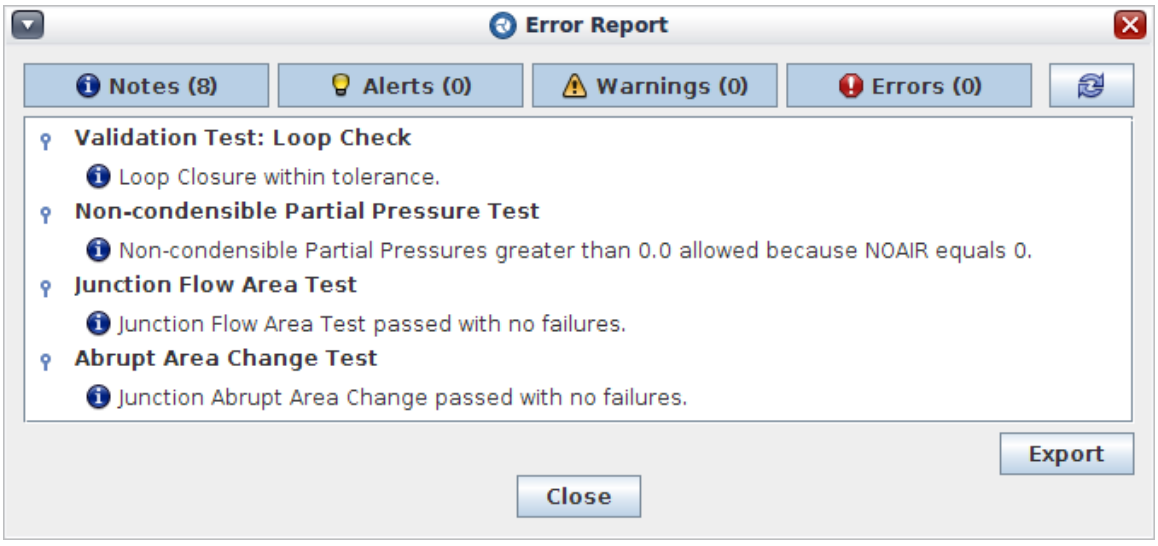


Figure 2.36. Model Report Window

If the model is free of errors, the model report dialog will display the message **`"Error check complete. No errors found."`** If errors are discovered, the problematic component can be selected immediately by double clicking on the error message.

Model validation tests, such as the elevation checker, may be enabled and disabled by editing the **Model Validation** property of the **Model Options** navigator node.

2.3.7. Saving a Model

A model can be saved by using either the **File** -> **Save** menu item, or the **Ctrl-S** key combination shortcut (**Command-S** on Mac OS). If the model has not yet been named, a Save dialog will be displayed within which the name and location of the model may be selected. The **File -> Save As** menu item, or the **Ctrl-Shift-S** shortcut (**Command-Shift-S** on Mac OS) can be used to save the model to another location.

Note Model Editor documents have a default extension of `.med`.

2.4. Model Notes

The Model Notes feature provides a simple way to add informative messages to one or more components. These notes consist of a title, a user-defined type, and an HTML document. Any number of notes can be associated with a component, and a given note can be associated with any number of components.

The basic per-component notes capability is available for all Model Editor plug-ins. Some plug-ins (such as RELAP5 and TRACE) provide more extensive support, including references to notes at the attribute level.

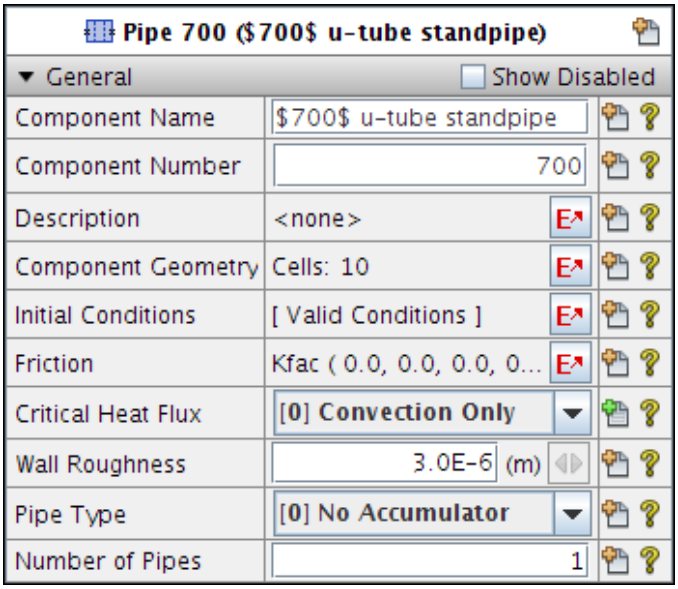


Figure 2.37. Property View Model Note Button

Model notes are viewed and managed primarily with the Model Note Viewer. It can be used to view or edit every note in the model or those associated with selected set of components.

The Model Notes viewer is available either from the **Tools** menu or a Properties View and is shown in Figure 2.38, “Model Note Viewer - Notes View” and Figure 2.39, “Model Note Viewer - Components View”. The viewer display will be customized to show a different set of components and properties depending on which button/item is used.

- The **Tools** -> **Model Note Viewer** menu item opens the model note viewer for the entire model and shows all notes and components available.
- The Properties View's **Model Note Viewer** button (to the right of the property view title) opens the note viewer for the currently selected objects (Pipe 700 in Figure 2.37, “Property View Model Note Button”). This is the same viewer used to display all notes. However, the **Category** will be set to **< Selected Components >** and it will display notes only for the object currently displayed in the property view.
- Each property's **Model Note Viewer** button (to the right of each property) opens the note viewer showing only that property. For example, the notes attached to the **Critical Heat Flux** property of all pipes could be displayed by selecting all of the pipes and pressing the note viewer button for that property.

The model note viewer displays in one of two modes. The "Notes View" shows a list of notes in the model on the left and where they're "Linked To" (i.e. used) on the right. The "Components View" shows a four column view of the components and their attributes. Each of these modes is a separate tab in the model note viewer. At the bottom of both views is a set of optional search parameters that can be used to narrow the displayed notes and links.

The notes view is made up of the list of notes (left), search options (bottom), and links to components (right), and a preview of the currently selected note (center).

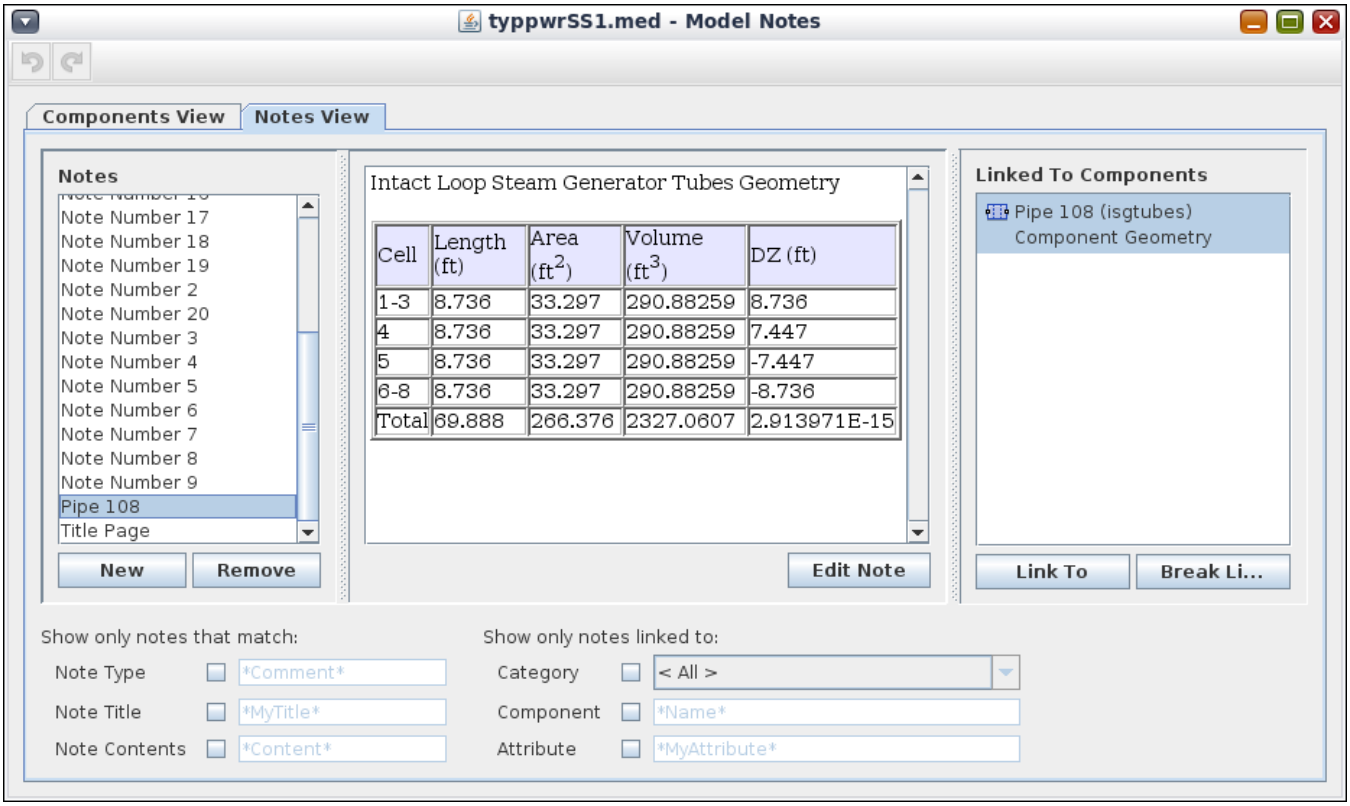


Figure 2.38. Model Note Viewer - Notes View

Two sections of search options are available. The options on the left are used to narrow the list of notes and those on the right match the components and/or attributes linked to notes. Each parameter can be enabled by the check-box to the right of the option's name. All of the enabled search parameters must match for the note to be displayed.

The **Notes** list on the left contains all of the notes in the model that match the current search parameters. The **New** and **Remove** buttons are used to create new notes and and remove the selected notes from the model. The **Edit Note** button at the bottom of the note display opens the note editor for the selected note.

The **Linked To Components** list to the right shows the list of components and attributes that refer to the selected note. **Link To** and **Break Link** buttons are provided to add or remove references.

The components view is made up of a set of optional search parameters and a list of the components and documentable attributes that match the selected search parameters. If the attribute does not reference a note, the **Note** column will show **-not set-**, otherwise the note's title will be shown. If multiple notes are referred to by an attribute, then the attribute will be displayed in several rows, once for each note.

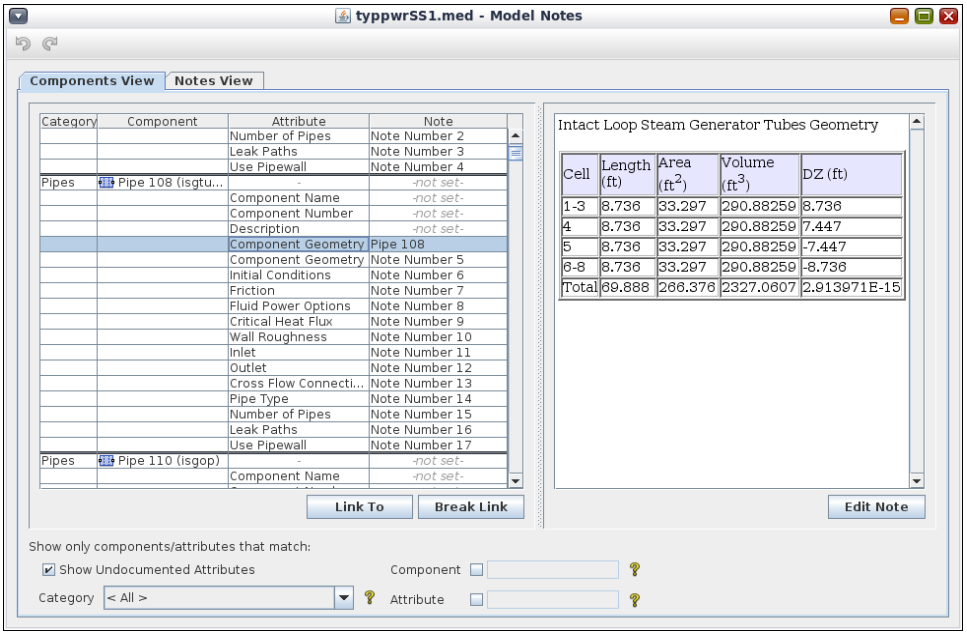


Figure 2.39. Model Note Viewer - Components View

A new note (or reference to a note) can be added either by double-clicking on the attribute entry in the list or by pressing the **Link To** button. Notes can be removed or edited by pressing the **Break Link** or **Edit Note** buttons respectively. The **Show Undocumented Attributes** check-box is used to show or hide those attributes that do not refer to any notes.

Each of these views uses the note editor shown in Figure 2.40, “Model Notes Editor” to display and edit model notes. This editor supports basic HTML formatting such as bold, italic and bulleted lists, as well as image links and hyper-links to external documents. These features, and several others, are available as buttons on the note editor toolbar.

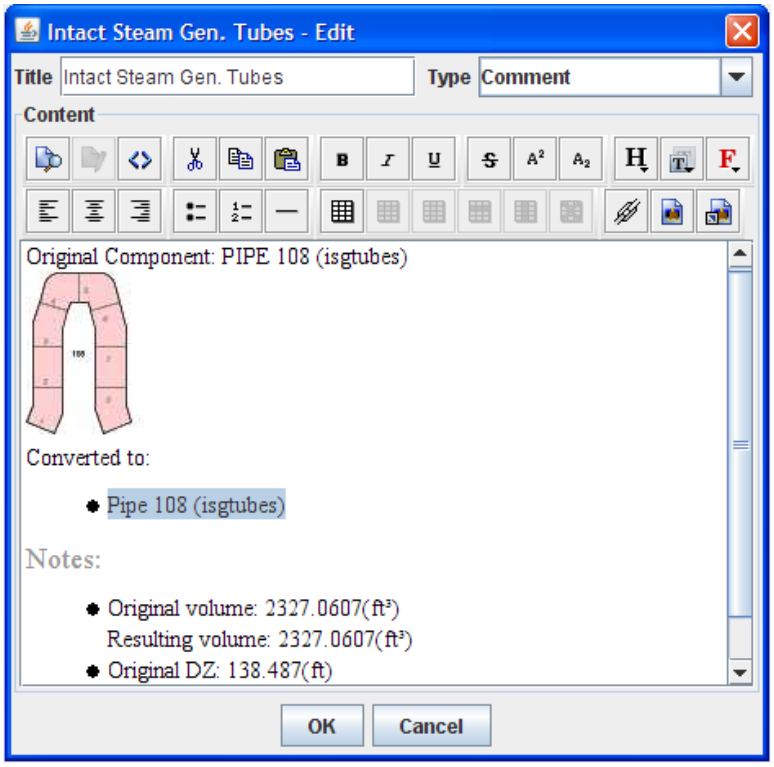


Figure 2.40. Model Notes Editor

In addition to the toolbar provided options, the note editor also supports pasting images and formatted text into a note from applications such as Adobe Acrobat Reader and Microsoft Word. Images pasted in this way are stored directly in the note rather than using an external link.

2.5. User-Defined Numerics

The SNAP User-Defined Numerics (UDN) feature is designed to allow properties of a model to be modified and/or calculated outside the normal input for the model. To this end, a user may create shared reals, integers, booleans, tables, and functions. The use and capability of each is detailed in the following sections.

Numerics serve the following general roles:

- Store one or more values referenced at other locations in the model.
- Compute a value based on other values and user-defined code.
- Define parametric value ranges that allow exporting and submitting a version of the model for each permutation of the parametric values.

All numerics are found under the **Numerics** category, typically found just before the **Views** category toward the bottom of the Navigator. The **Numerics** sub-categories, **Integers**, **Reals**, **Booleans**, **Tables**, and **Functions**, provide access to the numerics themselves. Numerics are created in the standard fashion: right-click the category of the desired numerics type and select **New** from the resulting pop-up menu.

Note Upon creating a new user-defined Real, a prompt will appear requesting the units of the new value. The available units vary by plug-in (although most plug-ins have a "no unit" type, often displayed as **No Unit (-)**). When assigning a reference to a Real value (discussed in the next section), the units must match between the referencing value and the user-defined numeric.

2.5.1. Single-Value Numerics: Reals, Integers, Booleans

Of the five numerics categories, integers, reals, and booleans all represent a single-value. These numerics share a number of common attributes with little variation in semantics. The properties of a real are displayed in Figure 2.41, “Numerics Properties”.

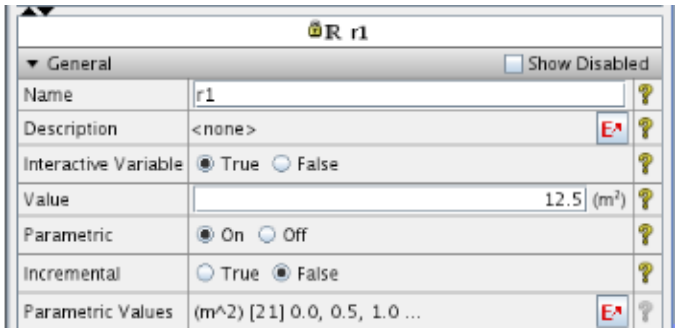


Figure 2.41. Numerics Properties

There are two ways to assign user-defined numerics to a value. The first (and only method for table cells that support numerics), is to right-click in the editor for that value and select the **Select Shared Value** item from the pop-up menu. The second is to press the **Select a shared value** button () on the right side the property editor. A dialog will be displayed to allow selecting the desired numeric.

Figure 2.42. Assigning a User Defined Numeric

All single-value numerics share the following properties.

- **Name** - the name of the parametric. There are two restrictions on the names of numerics:
  - A valid number should never be used.
  - Duplicate names *across the entire Numerics category* are not allowed (this is required to avoid undefined results when retrieving values in user-defined functions, discussed later).

  When choosing a numeric in an editor, the choices are listed alphabetically, so wise choice of names can save time and avoid confusion.
- **Description** - an optional and arbitrary user-defined description.
- **Interactive Variable** - determines whether the numeric can be modified by functions. When set to **True**, a lock appears next to the numeric icon, and attempting to modify the numeric in

a function will exit with an error. Conversely, setting this attribute to **False** removes the lock and allows setting the numeric value in functions.

- **Value** - the current value of the numeric. Any attribute or value that references the numeric will use this value instead of its own. When **Interactive Variable** is set to **False**, this value is uneditable and is set by function evaluation.
- **Initial Value** - if the numeric is not an interactive variable, the initial value is copied into **Value** at the beginning of function evaluation (see Section 2.5.4, “User-Defined Functions”). This defines the base-point of the numeric so that repeated function evaluation achieves consistent results. This attribute is only shown when **Interactive Variable** is set to **False**.

Note When toggling **Interactive** from **True** to **False**, the contents of **Value** are copied into **Initial Value**, and vice versa. This behavior is intended to prevent accidentally losing a value when switching the variable type.

Parametrics

Single-value numerics support a type of value ranges called parametrics. Exports may utilize parametric values to export the model multiple times: once for each value defined in a range or array. When multiple parametric numerics are used as the basis of an export, a model is generated for each permutation of the parametric values.

Integers and reals share the following properties related to parametrics. For booleans, **Parametric** is a simple "On/Off" flag that indicates whether parametric exports and submits should see true and false variations. Only input variables may be specified as parametric.

- **Parametric** - the flag that determines whether the numeric is parametric.
- **Incremental** - determines whether the parametric values are defined as a range or an explicit value array. Setting **Incremental** to **True** enables the **Parametric Values** attribute, while **False** enables **Start Value**, **End Value**, and **Increment**.
- **Parametric Values** - the parametric value array.
- **Start Value** - the start value for a parametric value range.
- **End Value** - the end value for a parametric value range.
- **Increment** - the increment of a parametric value range.

Note **End Value** is not guaranteed to be included in parametric exports and submits: it is the maximum bound at which a parametric value may exist. For a numeric integer with a start value of 0, an end value of 9, and an increment of 2, a simple parametric export would yield five exported models with the values {0,2,4,6,8}.

Integer Properties

Integers have the following exclusive properties related to defining an integer numeric as an enumeration.

- **Type** - determines whether the integer is an enumeration (**Enum**), or a standard single-value **Scalar**. When set to **Enum**, all other properties except those listed below are disabled.
- **Enumerated Values** - defines the values in the enumeration. Each value is a named integer, the order and number of which are completely arbitrary.
- **Value** - the enumerated value acting as the current numeric value. The drop-down allows selecting from the list of enumerated values in the numeric.

2.5.2. Numerics in a View

User-defined numerics can be added to a 2D view in the same way as other components: the **Add To View** menu in the numeric's Navigator node pop-up menu, the **Create View** item from the Numerics Navigator node, etc.. These values are represented in a view as a **Drawn Numeric**. Drawn Numerics that correspond to integers and reals appear as a text label and optional text field (depending on the **Editable** option, described below) that displays the name of the numeric, current value, and units if applicable (e.g. "Area 1" in Figure 2.43, "User Defined Numerics Example"). Booleans appear as check-boxes with a text label that displays the name of the numeric.

Note Drawn Numerics for user-defined tables and functions are not currently supported.

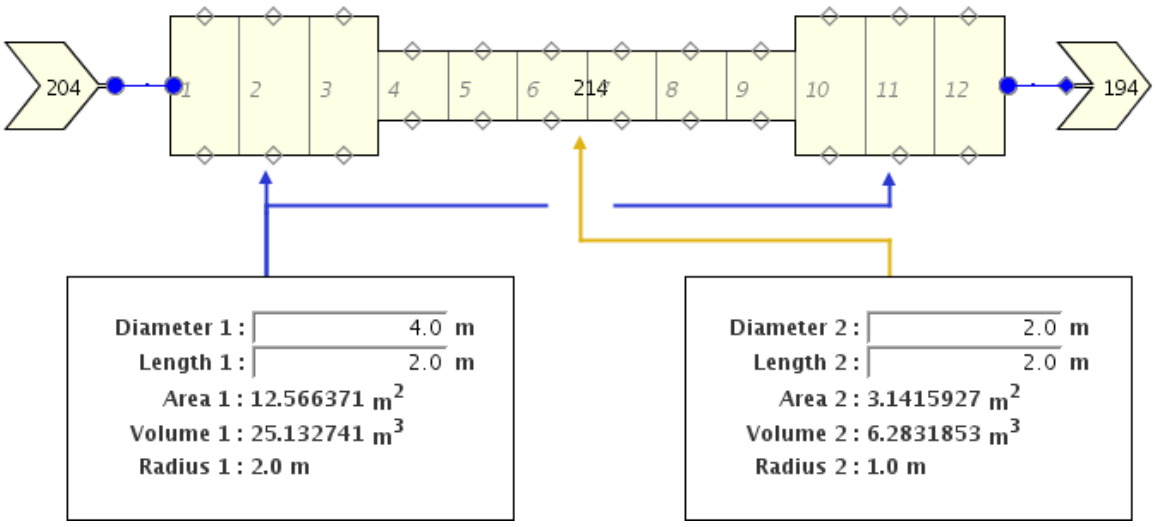


Figure 2.43. User Defined Numerics Example

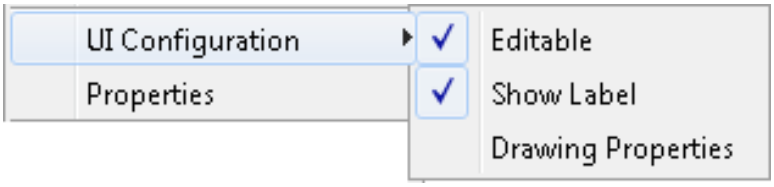


Figure 2.44. Drawn Numeric Menu

The appearance and capability of a drawn integer or real can be changed by using the following items in its right-click pop-up menu, as shown in Figure 2.44, "Drawn Numeric Menu".

- **Editable** - This check-box item enables editing an input variable by entering a new value into the Drawn Numeric (e.g. "Diameter 1" in Figure 2.43, "User Defined Numerics Example" above). Entering a value into a Drawn Numeric and pressing **Enter** will change the value in the same way as editing the numeric properties directly.

  Note Numerics can be edited in a locked 2D View. Locking the View makes editable Drawn Numerics easier to use, as they can no longer be selected by mistake.

- **Show Label** - This check-box item enables and disables the display of the user numeric name in the Drawn Numeric.

- **Drawing Properties** - The drawing properties item opens the Drawn Numeric Properties dialog. This dialog can be used to modify the drawn numeric's label text (replacing the name of the numeric), font, foreground and background colors, and border

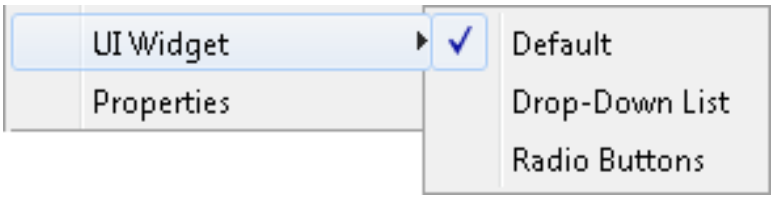


Figure 2.45. Drawn Numeric Enumerated Integer Menu

The drawn numeric enumerated integer provides three seperate editing modes. A default text field is provided which allows hand entered values where only valid enumerated values are accepted. A drop-down list is available where each of the available enumerated items are selected from a list in a vertical menu. Finally, a radio button display is provided which allows the selection of a single enumerated value. These drawing modes are displayed in the following figure:

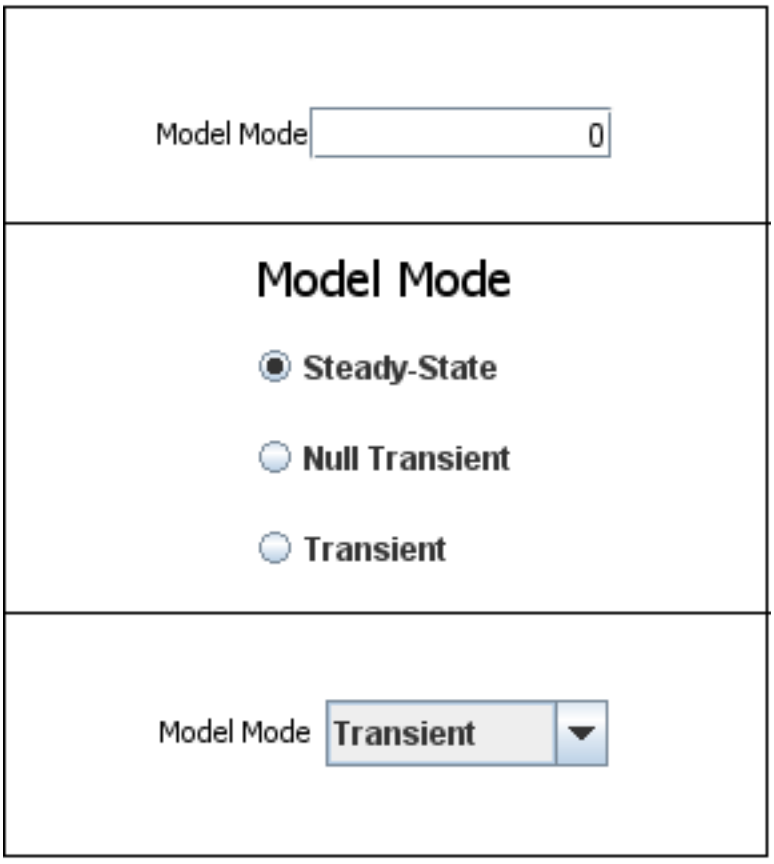


Figure 2.46. Enumerated Integer Drawing Modes

2.5.3. User-Defined Tables

User-defined tables allow arbitrary data tables which may be accessed in user-defined functions and referenced by various tables elsewhere in the model. Upon creating a user-defined table, a **Define Table Columns** dialog is displayed, as shown in Figure 2.47, “Numerics Table

Initialization". This dialog is used to determine the names, types, and order of the columns in the table. Once created, this *column signature* cannot be modified.

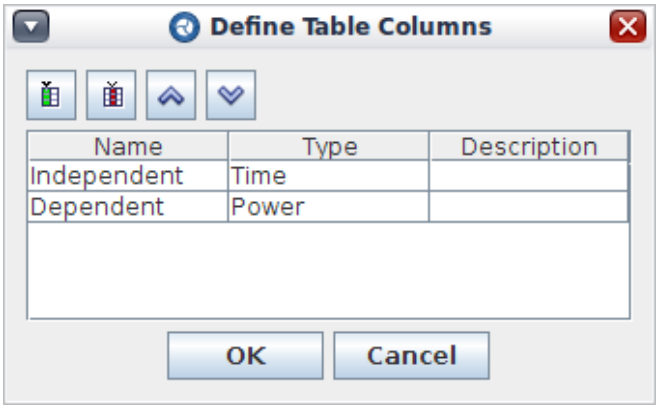


Figure 2.47. Numerics Table Initialization

Once the table is created, its data can be edited through the **Table Data** attribute. This opens the **Edit Table Data** dialog (see Figure 2.48, "Table Data Dialog"). This dialog allows adding and removing rows, moving rows up and down, and editing cell values.

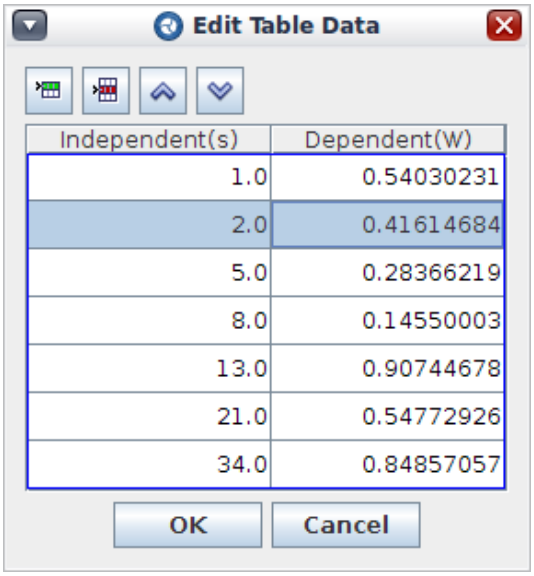


Figure 2.48. Table Data Dialog

Shared tables may be used as the data source for tables that support references. Most attributes that support references will use the standard table editor shown in Figure 2.49, "Table Editor with Reference". The **Shared Table** field at the top of the editor is used to set or clear the referenced numeric table. When a numeric reference is applied, its data is displayed in the table area.

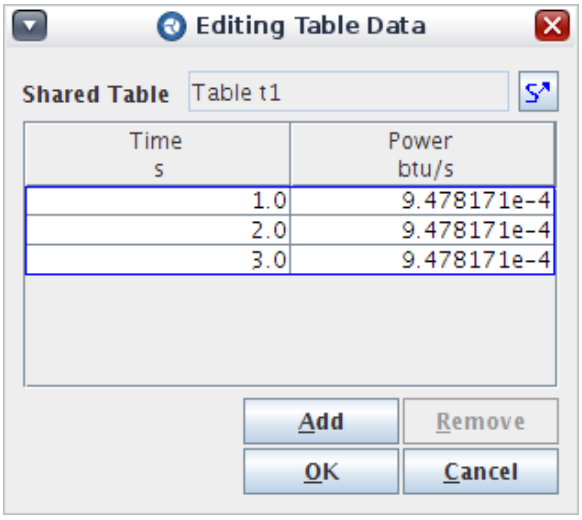


Figure 2.49. Table Editor with Reference

2.5.4. User-Defined Functions

User-defined functions provide a means of working with user-defined numerics from high-level code and external applications. User-defined functions support the Python programming language, Mathcad, and Matlab.

Note To use Mathcad or Matlab, the application installation location must be specified in the SNAP configuration. Refer to Section 3.3.1, "General Properties" for more information.

The properties of a user-defined function are displayed in Figure 2.50, "User-Defined Function Properties".

- **Function Type** - this property defines the external application or language that will interface with a selection of numerics.
- **Enabling Boolean** - this property determines whether the function is invoked during function execution. When set to reference a user-defined boolean numeric, the function is only executed when the boolean value is set to **True**.
- **External Function** - edits the function contents, as described below.

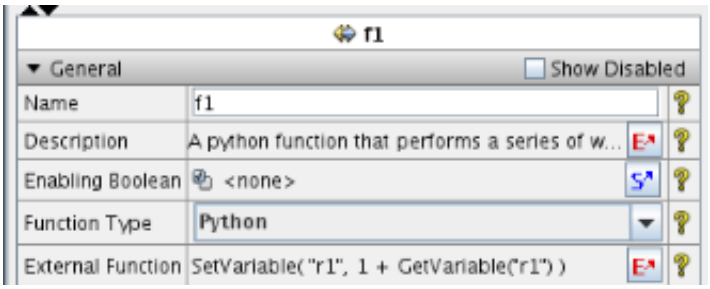


Figure 2.50. User-Defined Function Properties

User-defined functions are defined by selections of input and output variables and (in the case of external applications) a reference to a file that contains the function that will be run by the

application. For Matlab and Mathcad functions, the editor button of the function's **External Function** property opens an **External Function Dialog**, as shown in Figure 2.51, "External Function Dialog".

Note The following description describes the External Function dialog when used in combination with external application functions. When opened for a Python function, the dialog differs slightly, replacing the **External Function File** field on the top with a code panel on the right, as described in Section 2.5.4.4, "Python User-Defined Functions".

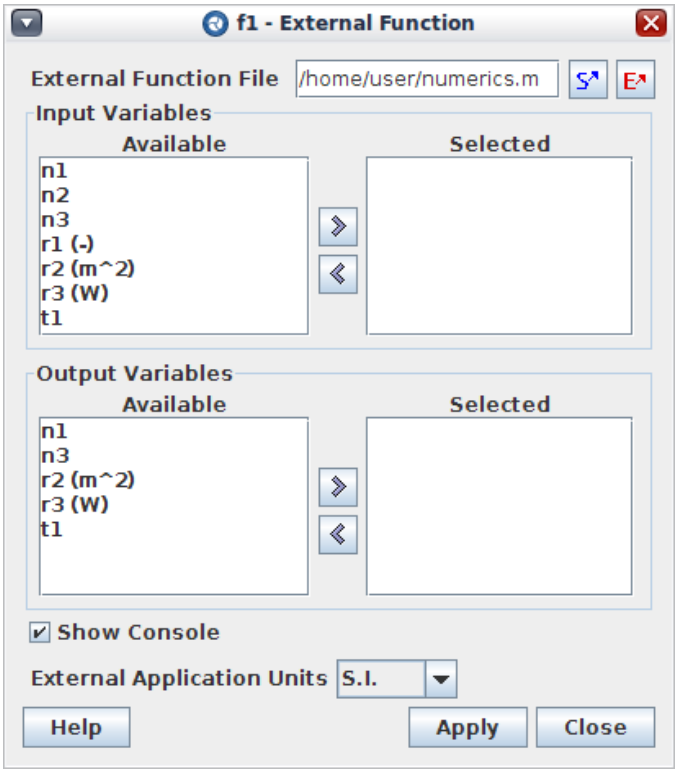


Figure 2.51. External Function Dialog

The **External Function File** field at the top of the dialog indicates the location of the file processed by the external application. The **Select** button to the right of the field can be used to select the file with a file browser. The **Edit** button further right will launch the external application, opening the indicated file.

The **Input Variables** and **Output Variables** segments of the dialog are used to indicate the numerics available to the external application. Numerics can be moved between the **Available** and **Selected** lists with the central **Left** and **Right** buttons. When executing a user-defined function, numerics in the **Input Variables Selected** list are available to the external application as an input: their value can be retrieved for use in calculations. Similarly, numerics in the **Output Variables Selected** list are available as outputs: their value can be modified by the external computation.

Note In Figure 2.51, "External Function Dialog", the list of available **Input Variables** and **Output Variables** varies slightly. The disparity is caused by the **Interactive Value** property of the missing numerics. Only those numerics with **Interactive Value** set to **False** can be used as an output in a user-defined function.

The **External Application Units** combination box controls the units of values passed to the external application. If the unit set specified in the combination box is different from the current model units, the appropriate conversions are made, the converted values are passed to the external application, and the resulting values are converted back before updating the numerics.

The **Show Console** check-box, when selected, gathers all output from the external applications (or Python execution) and displays it in a pop-up dialog when the function has completed execution, as shown in Figure 2.52, "External Function Output Console"

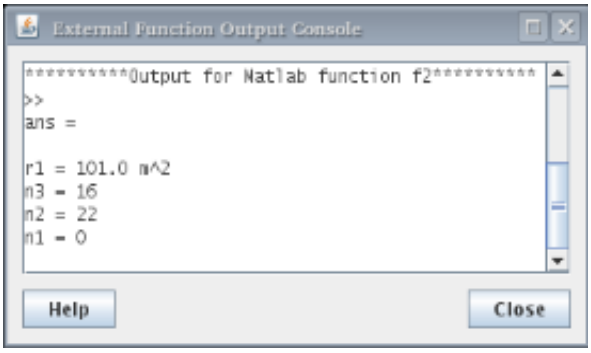


Figure 2.52. External Function Output Console

Finally, the **Apply** button commits all changes in the dialog to the user-defined function. As implied by the name of the button, the dialog is non-modal: other dialogs and Model Editor functionality can be used while this dialog is open, including editing dialogs for other functions.

2.5.4.1. Function Execution and Category Properties

Selecting the **Functions** category displays the properties displayed in Figure 2.53, "Functions Properties".

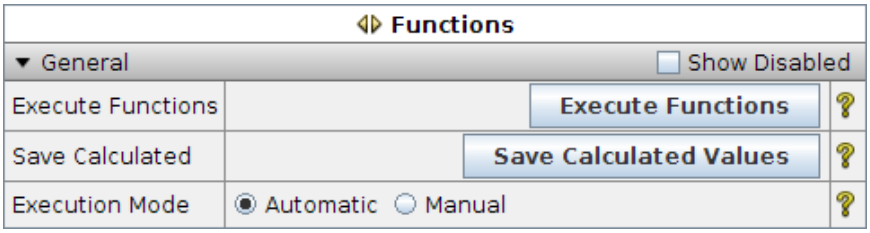


Figure 2.53. Functions Properties

Functions can be explicitly executed in one of the following ways:

- The **Execute Functions** property editor in the **Functions** category properties.
- The right-click pop-up menu on the **Functions** category contains an **Execute Functions** item.

Before function execution, all non-interactive numerics have their current **Value** replaced by their **Initial Value**. Afterwards, functions are executed in the sequence they appear in the Navigator; this ordering can be customized via the **Move Up** and **Move Down** items found in the right-click pop-up menu for numeric functions. As described in Section 2.5.4, "User-Defined

Functions", functions that reference an **Enabling Boolean** will only execute when the referenced boolean evaluates to **True**.

After function execution, the **Save Calculated Values** editor button of the **Function** category properties can be used to copy current numeric **Values** into their **Initial Values** field.

The **Execution Mode** property can be used to control how and when functions are executed in the model. When set to **Automatic**, certain actions, such as submitting a stream or exporting a parametric model, will cause function evaluation. When set to **Manual**, only the explicit function execution methods described above will cause function evaluation.

2.5.4.2. The Matlab-UDN Interface

When an instance of Matlab is opened from the Model Editor, Matlab commands that initialize the Matlab-UDN interface are automatically executed. This interface provides a Java object in Matlab that contains methods for accessing and modifying user-defined numerics that were passed into Matlab as input and output variables. The following interface functions are available for use in Matlab M-Files.

Note As a convention, the interface object is always named `cafean`. Therefore, calls to interface methods must be of the form `cafean.method()`, where `method` can be any of the following methods and their appropriate signatures.

- `getModelName()` - returns the name of the SNAP model associated with this interface instance.

  ```
  M = cafean.getModelName()
  ```

- `addMessage( 'message' )` - adds a message to be displayed in the Message Window when the function completes execution.

  ```
  cafean.addMessage( 'Default Message' )
  ```

- `getMessages()` - returns a newline-delimited list of messages currently set to be displayed in the Message Window.

  ```
  S = cafean.getMessages()
  ```

- `getReturnStatus()` - gets the return status for this interface instance. A return status of `0` indicates normal behavior, a non-zero return status will cause the Model Editor to not update any output variables that were set to be updated.

  ```
  i = cafean.getReturnStatus()
  ```

- `setReturnStatus( value )` - sets the return status for this interface instance. This method does its best to coerce `value` into an integer.

  ```
  cafean.setReturnStatus( 0 )
  ```

- `listVariables()` - returns a newline-delimited list of variables that have been added to the current Matlab session by the interface. Each variable is represented in the format "name = value".

```
cafean.listVariables()
```

- `getVariable( 'name' )` - checks the list of variables for variable `name` and, if found, returns the current value of that variable. If the variable could not be found, an error message is added to the Message Window.

  ```
  A = cafean.getVariable( 'Area_1' )
  ```

- `setVariable( 'name', value )` - sets the value of the indicated numeric. If the indicated numeric is not found, or the input value cannot be coerced into the appropriate type, an error message is added to the Message Window.

  ```
  cafean.setVariable( 'Area_2', A )
  ```

- `getTableValue( 'name' , row, column )` - gets the value of the specified table at [row, column]. Table indexes are consistent with the Matlab convention, and thus start at 1.

  ```
  X = cafean.getTableValue( 't1', 1, 1)
  ```

- `setTableValue( 'name', row, column, value )` - sets the table value at the given cell. Makes the same attempt at type coercion as `setVariable.`

  ```
  cafean.setTableValue( 't1', 1, 1, X)
  ```

Note Suppressing output from a Matlab command is achieved by appending a semicolon to the end of the command. Suppressed Matlab output will not be displayed in the External Function Output Console.

2.5.4.3. The Mathcad-UDN Interface

When functions of type **Mathcad** are executed, a Matlab session is opened with the Mathcad worksheet specified in the **External Function File** field of the External Function Editor. The specified input and output variables are loaded into the worksheet automatically, and are ready for use by the worksheet. When the worksheet finishes executing, the values of the output variables are updated accordingly. The **Show Application Window** check-box in the External Function Dialog for Mathcad functions controls whether or not the Mathcad window is displayed.

2.5.4.4. Python User-Defined Functions

The Python External Function dialog functions similarly to the Section 2.5.4, “User-Defined Functions”, with the **External Function File** at the top of the dialog replaced by a Python code panel on the right, and the dialog receives a menu bar giving access to additional functionality. The dialog is shown in Figure 2.54, “Python External Function Dialog”.

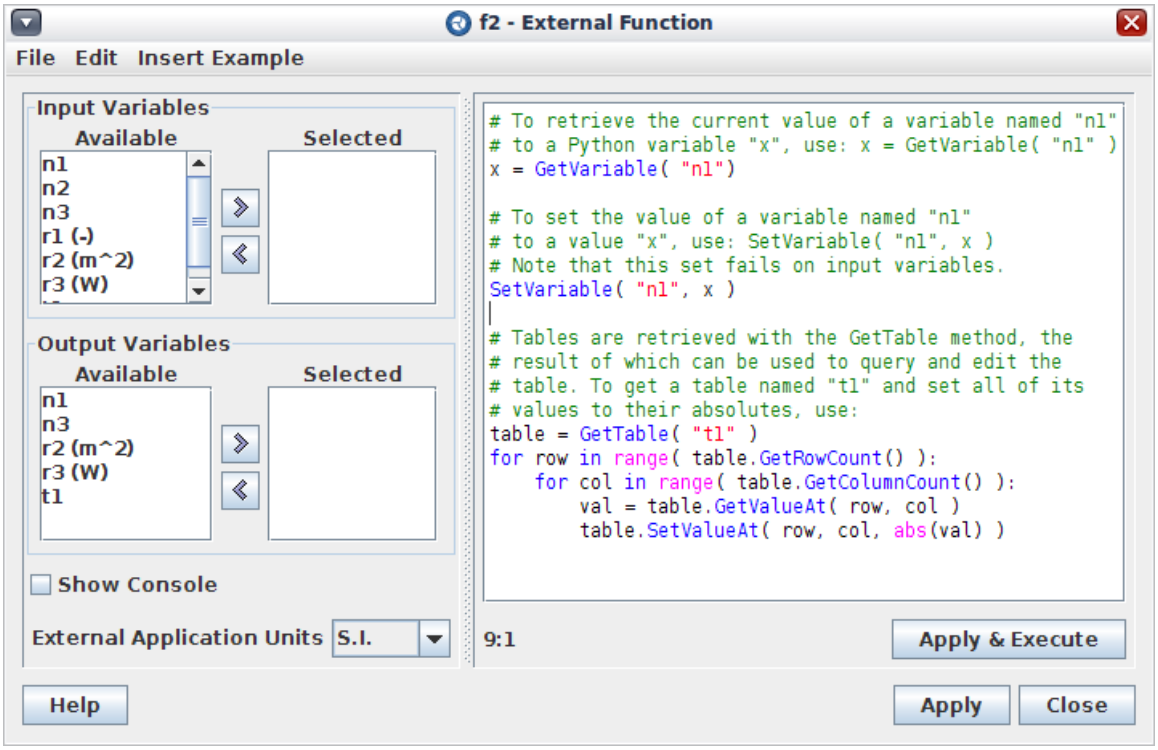


Figure 2.54. Python External Function Dialog

Python Code

The function retrieves data from numerics and/or external sources, computes values, and places data back into numerics. Standard Python can be used in this logic, including the definition and usage of functions, classes, etc..

The following custom functions have been provided to access numerics from user-code.

- `GetVariable( name )` - returns the value of the indicated numeric.

  ```
  X = GetVariable( "Symbolic Name" )
  ```

- `SetVariable( name, value )` - sets the value of the indicated numeric.

  ```
  SetVariable( "Symbolic Name", X )
  ```

  `SetVariable` does its best to coerce values into the appropriate type. In the case of booleans, all non-zero values passed to a boolean numeric are converted to `True`; all zero values are converted to `False`. Similarly, boolean values copied into other numerics are treated as `0` for `False` and `1` for `True`.

- `GetTable( name )` - returns a table object that can be used to query dimensions of a table, as well as get and set its values. Table objects have the following methods.

  - `GetRowCount()` - returns the number of rows in the table.
  - `Get ColumnCount()` - returns the number of columns in the table.
  - `GetValueAt( row, column )` - gets the table value at the given cell. Indexes start at 0.

- `SetValueAt( row, column, value )` - sets the table value at the given cell. Indexes start at 0. This function makes the same effort as `SetVariable` to coerce values to the appropriate type.

The following is an example of table usage.

```
table = GetTable( "t1" )
for row in range( table.GetRowCount() ):
    for col in range( table.GetColumnCount() ):
        val = table.GetValueAt( row, col )
        table.SetValueAt( row, col, abs( val ) )
```

Additional Dialog Functionality

The following additional features are available in the Python editing dialog.

- **File** menu - this menu contains entries dedicated to manipulating the Python code itself. The function code can be written to a file with the **Export File** item. The **Import File** menu item will replace the Python logic with the contents of the selected text file. **Append File** imports the contents of the selected file to the end of the existing code; **Prepend File** imports at the beginning.

- **Edit** menu - this menu contains items centered around the system clipboard. Items are present to **Cut** and **Copy** the currently selected Python code, and to **Paste** the contents of the clipboard into the code panel.

- **Insert Example** menu - items in this menu do exactly what the name implies: insert Python samples into the code panel.

2.6. Component Sub-Systems

Component sub-systems are a convenient way to improve the logical layout of a model by allowing the components that make up physical systems (such as steam generators) to be grouped together. Component sub-systems appear in the Navigator under the **Sub-Systems** category node and can be created by selecting **New** from the category's right-click pop-up menu.

Sub-systems also support component-level and attribute-level documentation links and can be used as a central location for documentation that applies equally to an entire sub-system (see Section 2.4, “Model Notes”).

The list of components included in a sub-system can be changed by editing the **Components** property of the sub-system. A sub-system can contain any number of components as well as other sub-systems. A component or sub-system can only be included in a single sub-system.

Component sub-systems currently support the following additional features:

- **Copy/Paste Special** - When a sub-system is copied, the contents of that sub-system are copied as well. Use the **Paste Special** feature to paste the sub-system within a model or to a different model. This will allow the newly pasted components to be renumbered.

- **Create View / Add To View** - The right-click pop-up menu of any Sub-System can be used to either create a new 2D View with the included components or add them to an existing 2D View.

- **Show ASCII** - The **Show ASCII** feature can be used with sub-systems in the same way as other components. The ASCII view for a sub-system includes the ASCII that would be displayed for each of the included components in the order they appear in the sub-system.
- **Component Differencing** - Much like **Show ASCII** above, the **Component Differencing** feature can also be used with sub-systems. The differenced ASCII will be that of the included components in the order that they appear in the sub-system.
- **Exportable** *(Optional)* - The **Exportable** property is used to specify those sub-systems that are complete "Composite Components" that will be used with the Sub-System Integration feature described below. This feature is not supported by all plug-ins.

Note Component Sub-Systems is an optional feature that may not be supported by all pre-processor plug-ins.

2.7. Sub-System Integration

The sub-system integrator (or "Master Integrator") allows a sub-system in one model to be imported into another model. Once imported, the integrator can also be used to *update* the imported sub-system without requiring the components to be reconnected manually.

Note In this section the descriptions refer to the "Composite Component" model (the model that a sub-system will be copied from) and the "Full Plant" model (the model that the components will be copied into). However, sub-system integration is not limited to this situation alone.

For example, a standalone steam generator model could be created with all of the necessary boundary conditions to run to a steady state. The hydraulic components and heat structures that make up the steam generator can then be added to an *Exportable* sub-system. This sub-system could then be imported into a full plant model and connected to the appropriate locations. Later, when changes are made to the steam generator model the full plant model can be updated to include these changes. All of the connections made in the full plant model will be automatically reconnected.

Using the Sub-System Integrator

The following steps are required to import a sub-system.

1. Create a "Composite Component" model.
2. Create a sub-system in the "Composite Component" model.
3. Set the sub-system property Exportable to Yes.
4. Create or open the "Full Plant" model.
5. Right-click on the **Sub-Systems** node in the Navigator and select the **Import Sub-System** menu item.
6. Select the MED (Model Editor Document) file containing the "Composite Component" model. This will then open the Select Sub-Systems dialog to allow the desired sub-system to be selected.

7. Select the desired exportable sub-system from the combo box in the top left-hand corner of the Select Sub-Systems dialog.

8. Select **<Create New Target>** from the combo box in the top right-hand corner of the Select Sub-Systems dialog.

9. Press **Next** to complete the import. This will display a the message "Sub-system integration complete." The imported sub-system can then be connected to the "Full Plant" model.

The following steps are required to update an imported sub-system.

1. Right-click on the sub-system's Navigator node and select the **Update Sub-System** menu item.

2. Select the MED (Model Editor Document) file containing the "Composite Component" model. This will then open the Select Sub-Systems dialog shown in Figure 2.55, “Select Sub-Systems Window”.

3. Select the desired sub-system in each of the combo boxes at the top of the dialog.

4. Press **Next** to continue the update.

   If any interconnections are discovered that cannot be supported by the update, a message window will be displayed listing the details for each connection. Pressing **Cancel** at this point will stop the process without modifying the model.

   If all connections are supported, the Display Cross References dialog will be displayed as shown in Figure 2.56, “Cross-Reference Display”.

5. Examine the interconnections listed in the dialog to ensure that the correct sub-system has been selected, then press **Finish** to complete the update. Pressing **Cancel** instead at this point will stop the process without modifying the model.

The Select Sub-Systems Window

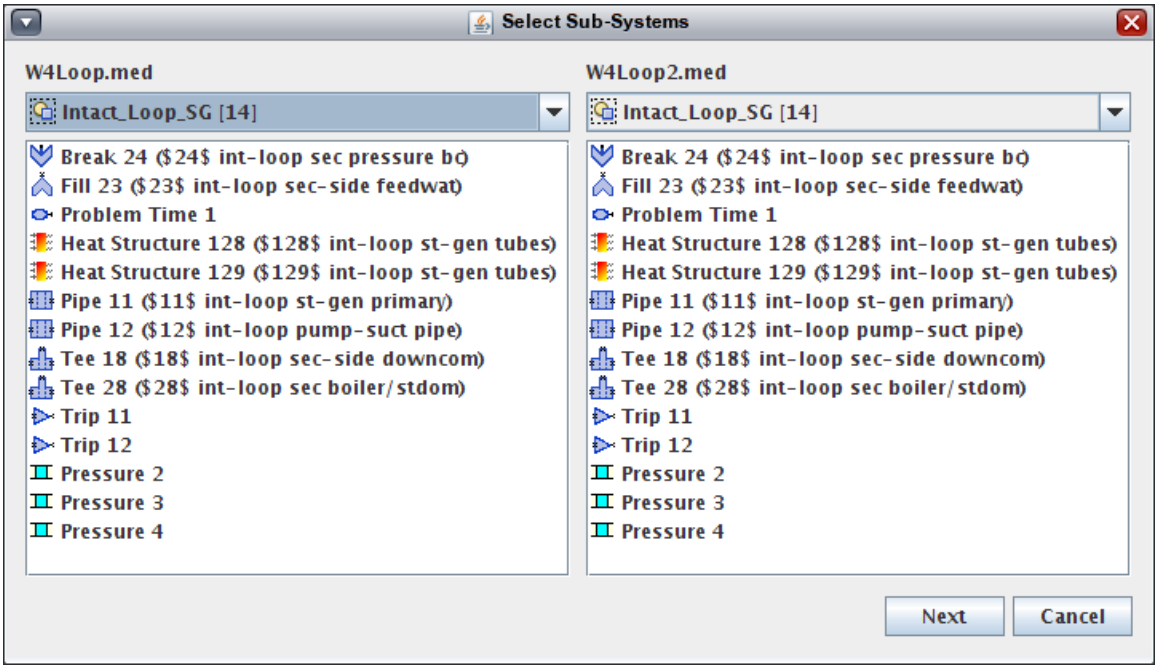


Figure 2.55. Select Sub-Systems Window

The Select Sub-Systems Window is used to select which sub-system in each model (full plant vs. composite component) will be used in the integration. On the left side of the window is the "composite component" model, a drop-down list of the exportable sub-systems in that model, and the list of components in that sub-system. Similarly, the "full plant" model is in right side of the window.

The Cross-Reference Display

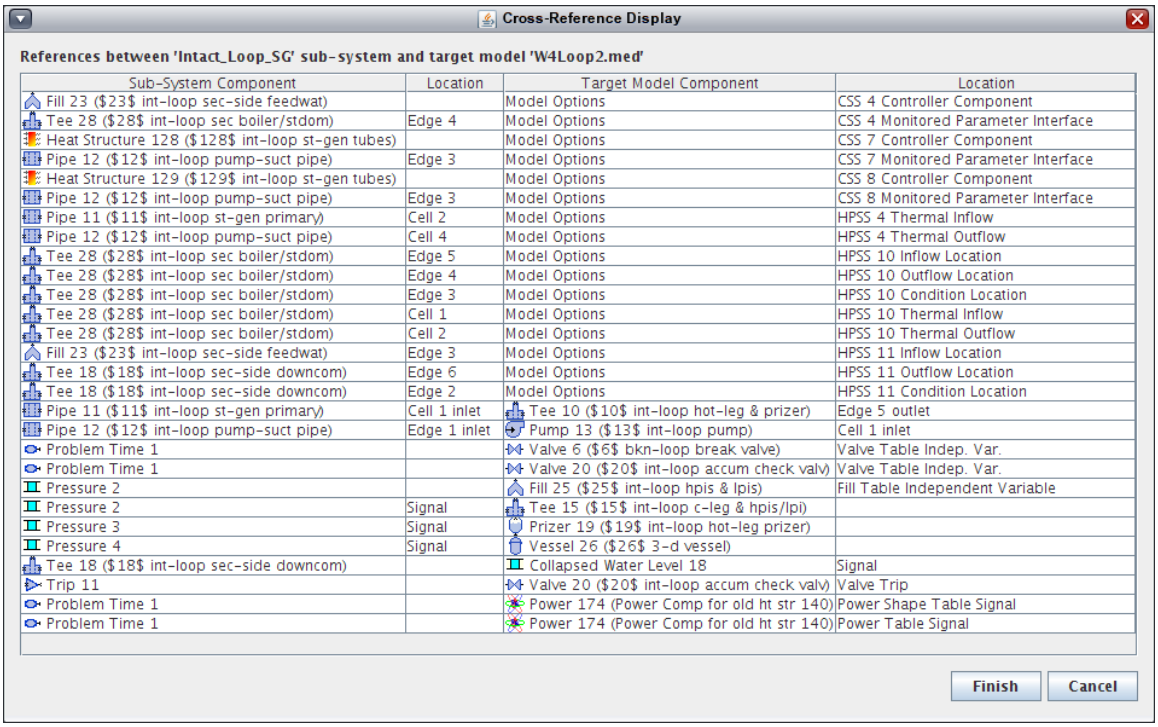


Figure 2.56. Cross-Reference Display

The cross-reference display contains a list of all of the interconnections between the "full plant" model and the sub-system that will be handled during the integration process. The table in the center of the window is made up of 4 columns. The two left-most columns list the components inside the sub-system. The two right-most columns list the components outside the sub-system. The location columns indicate where in the listed component the interconnection occurs. For example: In Figure 2.56, “Cross-Reference Display” above, the inlet **Pipe 11 cell 1** is connected to the outlet of **Tee 10 edge 5**.

Note This window is not displayed when importing a sub-system for the first time as there are no interconnections to handle.

2.8. Command Line Usage

The SNAP installation includes platform specific launchers for each of the included applications. For Windows a set of executables are included. For UNIX type systems (including Mac OS X) shell scripts are included. These launchers are placed in the `bin/` directory of the SNAP installation and can be executed directly from the command line. This allows the Model Editor to be executed from other applications or scripts. When combined with batch commands (Section 2.9, “Batch Command Syntax”) this allows the Model Editor to be used by automated systems such as analysis code test suites.

A special version of the Model Editor, called MEBatch, can be used for executing batch scripts. MEBatch runs in "headless mode", which allows it to run on machines without a graphical environment, such as a typical server. To invoke MEBatch, use the Windows `mebatch.exe`

executable of the `mebatch.sh` shell script, both of which are in the `bin/` directory of the SNAP installation.

Note Some commands, such as **`SHOW`** and **`HIDE`** are will not function in headless mode.

When run from the command line, certain options can be specified for the Model Editor. The command line options can be displayed by running the Model Editor from the command line with the option --**`usage`** as shown below.

```
Symbolic Nuclear Analysis Package (SNAP) Model Editor
Version: 1.0.0

Command line usage:
  snap [--batch filename] [--debug] [--nosplash]
       [--noscreen] [--remote] [--usage] [--version]
  where --batch identifies a batch file to process.
        --debug turns on debug mode.
        --nosplash turns off the splash screen.
        --noscreen turns off screen file logging.
        --remote disables double buffering for remote X displays.
        --usage prints this message and exits.
        --version prints the version info and exists.
```

When running MEBatch, the command line syntax is as follows. The key difference between invoking the Model Editor from a command line and MEBatch is that MEBatch requires a batch file (specified without **`--batch`**) somewhere in its list of arguments.

```
Command line usage:
  mebatch <batchfile> [--debug] [--usage] [--version]
  where <batchfile> identifies a batch file to process.
        --debug turns on debug mode.
        --usage prints this message and exits.
        --version prints the version info and exists.
```

`--batch <filename>`	The Model Editor supports a batch command language. This argument is used to specify a batch file that will be run before the Model Editor is displayed. Refer to Section 2.9, "Batch Command Syntax" for a detailed explanation of the Model Editor batch command syntax.
`--debug`	This option enables various debugging features of the Model Editor and is recommended that this parameter only be used by plug-in developers.
`--noscreen`	This option disables console logging, instead telling SNAP to write its output directly to the console.
`--nosplash`	This option disables the splash window display on startup.
`--remote`	The default JAVA double-buffering scheme causes a significant UI performance loss on some systems when displaying the Model Editor from a remote machine. This option disables the double-buffering and may improve performance in these circumstances.
`--usage / --help`	This option displays the usage information shown above and exits.
`--version`	This options displays the current Model Editor version and exits.

2.9. Batch Command Syntax

The following is the standard batch command set. Each command is listed with optional fields shown in between less than < and greater than > signs, while required fields are shown between square brackets [ ].

Note Not all commands are supported when running with MEBatch, which runs in headless mode (see Section 2.8, "Command Line Usage"). These commands are noted as not supported in headless mode.

- **SHOW** - The show command is used to display the primary user interface. This allows for batch commands to execute in such a way that the user receives immediate visual feedback. If this option is not used, the Model Editor will run without a graphical interface. This command is not supported in headless mode.

- **HIDE** - This command hides the primary user interface. This command is not supported in headless mode.

- **MESSAGE [message]** - Where `message` contains a quoted string to be displayed to the used.

  The **MESSAGE** command can be used to display a single quoted message to the user in a pop-up dialog with an OK button. This batch command will stop the batch file process until the user presses OK. In headless mode, this command writes a message to the console; no user interaction is necessary.

- **CHECK_MODEL** - This performs a model check in an identical manner to the Check Model button on the main toolbar. This is useful for a batch verification of a large number of models.

- **EXIT** - The exit command is used to close the Model Editor. This should be used at the end of a batch file to ensure that the Model Editor process does not continue running. If the batch file contains the **SHOW** command, this will close the user interface as well.

- **OPEN <Mn> [filename]** - Where **Mn** is defined as one of the ten valid model labels numbered **M0-M9**.

  The **OPEN** command is used to open a MED (`*.med`) file in the Model Editor. Upon opening, the new model becomes the current model. Optionally, the user may declare that the model inside that file should be identified with one of the ten labels. Acceptable values for the label field are M0 through M9. Until that model is closed, it may be accessed using its label in subsequent batch commands. The last parameter is the file name and should include the full path to the file.

- **SAVE <Mn> [filename]** - Where **Mn** is defined as one of the ten valid model labels numbered **M0-M9**.

  The **SAVE** command is used to save a model in MED format. Optionally, the user may specify the model to be saved by that model's predefined label. Acceptable values for the label field are **M0** through **M9**. If no label is given, the current model will be saved. The last parameter is the file name and should include the full path to the save file.

- **CLOSE <Mn>** - Where **Mn** is defined as one of the ten valid model labels numbered **M0-M9**.

The **CLOSE** command is used to close a currently open model. Optionally, the user may specify the model to be closed by that model's predefined label. Acceptable values for the label field are **M0** through **M9**. If no label is given, the current model will be closed. Any unsaved changes to the given model will be lost when closed.

- **SETCONST <Mn> [name] [value]** - Where **Mn** is defined as one of the ten valid model labels numbered **M0-M9**.

  This command is used to set the current value of a User Defined Constant to the specified value. The name of the constant must be enclosed in double-quotes if the name contains any spaces.

- **SETVAR <Mn> [name] [value]** - Where **Mn** is defined as one of the ten valid model labels numbered **M0-M9**.

  This command is used to set the current value of a User Defined Numeric to the specified value. The name of the constant must be enclosed in double-quotes if the name contains any spaces.

  Note This command only sets the value of interactive variables, and is meant to replace the **SETCONST** command.

- **EXECUDF <Mn>** - Where **Mn** is defined as one of the ten valid model labels numbered **M0-M9**.

  Executes the User Defined Functions in the specified model (if given) or the current model if none is specified.

- **EXPORT_SMZ <Mn> [job stream name] [filename]** - Where **Mn** is defined as one of the ten valid model labels numbered **M0-M9**.

  This command is used to export a full Stream Manager ZIP file including the stream definition and all input models to the given file. The resulting SMZ cannot be used to execute the stream directly. It can, however, be very useful when testing application-specific and site-specific modules.

- **SET [property] [value]** - The set command is used to set a property to a specific value. Where **[property]** is defined as:

  - **CURRENT** - The current model. Valid values of **CURRENT** are **M0-M9**.

  Note At this time, only the property **CURRENT** (the current model) can be changed.

- **LOGFILE [filename]** - The logfile command is used to specify a file where messages are logged. The filename should include the full path to the log file.

- **MACRO <Mn> [filename]** - Where **Mn** is defined as one of the ten valid model labels numbered **M0-M9**, and `filename` is the file name which should include the full path to the file in quotes.

  This command executes the Python macro contained in the specified file.

- **LOCK_VIEWS <Mn>** - Where **Mn** is defined as one of the ten valid model labels numbered **M0-M9**.

This command will lock all of the views for the specified model. The current models views will be locked if the model label is not specified. This command is not supported in headless mode.

- **UNLOCK_VIEWS <Mn>** - Where **Mn** is defined as one of the ten valid model labels numbered **M0-M9**.

  This command will unlock all of the views for the specified model. The current models views will be unlocked if the model label is not specified. This command is not supported in headless mode.

Plug-in Specific Commands

In addition to those commands listed above, each Model Editor code plug-in can support plug-in specific batch commands. Plug-in specific batch commands must be prefixed with the corresponding plug-in ID. For example, a TRACE import command would resemble the following:

```
TRACE IMPORT ASCII_FULL "trace_model.inp"
```

Inline File Arguments

Some plug-in specific commands require a way to specify more complex multi-line input for some commands. An *Inline File* is intended as a way to place a larger, more strictly formatted set of data as a single value argument. An *Inline File* argument must be the last argument to a batch command.

```
PLUGIN SETDESCRIPTION <
This is an example of using an Inline File to set the
description of a model.
This description can span multiple lines and will maintain
its
spacing
and formatting.
.
```

Note that the *Inline File* begins with the **<** (less than) character and ends with a **.** (period). The **<** should be at the end of the line preceding the Inline File and the **.** should be on the line after the *Inline File* on a line by itself.

Chapter 3. The Configuration Tool

The Configuration Tool is used to configure global properties for the SNAP client applications. This tool is also used to startup, shutdown, and configure the local SNAP Calculation Server.

3.1. Menu Items

The Configuration Tool includes a menu bar which contains access to various functions outlined below.

- **File -> Save** - Saves the current configuration and notifies the running calculation server that the configuration has changed and should be refreshed. This also informs any open Model Editor or Job Status applications that the configuration has changed and should be reloaded.
- **File -> Export Public Keys** - Exports the public encryption keys used for connecting to the local Calculation Server from a remote location. Using this menu item opens a file selection dialog which may be used to specify the name and location of the key.

  Note The Outlook email application may not allow filenames that end with the .key extension.
- **File -> Generate Server Keys** - Regenerates the public encryption keys used for connecting to the Calculation Server from a remote location. All previously exported keys will be invalidated.
- **Help -> Check for Updates** - Displays a login dialog which can be used to login to the SNAP web server and check for updates and optionally retrieve and install them.

3.2. Main Window

The Configuration Tool main window includes the same Navigator and Property View interface layout used by the Model Editor (see Section 2.1.3, “The Navigator” and Section 2.1.4, “The Main Property View”). The Navigator on the left displays a tree of nodes related to specific configurations: selecting a node displays its editable properties in the Property View on the right. Standard SNAP functionality is provided, including multi-selection edit, Undo/Redo (via toolbar buttons), Cut/Copy/Paste/Delete (via toolbar buttons and right-click pop-up menus), and pop-up help for each property.

The **Check Configuration** button is available on the main toolbar of the Configuration Tool. Pressing this button will open a list of the errors, warnings, and other messages resulting from a check of the current SNAP configuration.

The Navigator has two root nodes: one for **Personal Settings** stored in the user's home folder, and one for **Global Settings** stored in a separate, locally accessible folder. Each of these nodes has a **Platforms** node containing the defined platforms and an **Applications** node containing the defined applications. Both the general SNAP settings and local calculation server settings are accessed from the **Personal Settings** node.

The **Global Settings** node contains a **Groups** node that contains additional grouped sets of platforms and applications. Portions of the **Global Settings** that are read-only (i.e. the files

containing them are read-only) are displayed with a lock icon. The properties of read-only nodes are displayed in the Property View but cannot be changed.

Note The **Global Settings** only appears when a global configuration has been specified as part of the SNAP configuration. Consult your local system administrator for information related to specifying a global stream configuration.

The Configuration Tool allows a user to add or remove **Platforms**, **Applications** and **Groups** from the current site configuration. The parent tree node of each of these types includes the **New** pop-up menu item that can be used to create new instances of each type. They can then be removed by selecting the **Remove** item from the right-click pop-up menu of the **Application**, **Group**, etc..

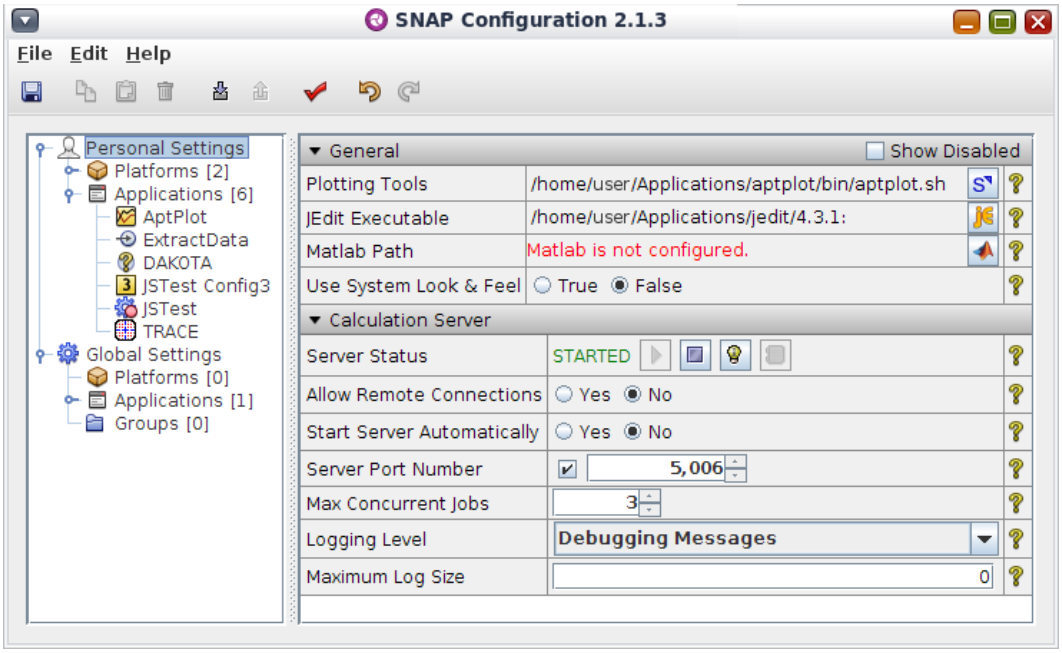


Figure 3.1. Configuration Tool Main Frame

3.3. Personal Settings

Personal Settings specify local configuration which is stored in the user's `.snap` folder. Available settings range from the installation paths of utility applications used by SNAP to the configuration of the implicit local calculation server.

3.3.1. General Properties

The **General Properties** group includes options pertinent to all applications in the SNAP suite.

Plotting Tool

This property defines the installation path of the plotting tool used by client applications to plot data. For example, right-clicking on an animation component in the Model Editor will display a pop-up menu with a **Plot Data** menu item. Selecting this item will launch another dialog within which Data Channels specified for that component are listed. After selecting one or more channels and pressing the **OK** button, the specified **Plotting Tool** will be launched. Data specific

to the selected channels will be provided to this application for plotting. SNAP currently supports the AcGrace and AptPlot plotting packages.

Note For AcGrace under Windows, cygwin's bin and usr\X11R6\bin directories must be included in your system path, i.e. "`c:\cygwin\bin;c:\cygwin\usr\X11R6\bin;`" without quotes.

jEdit Editor

This property defines the installation path of jEdit, used by SNAP for displaying ASCII data. jEdit is a pure-Java, programmer's text editor that was chosen for use with the SNAP system because of its stability and multitude of features.

Matlab Editor

The location of a Matlab installation used by SNAP for interfacing the Matlab Computational Engine with SNAP User-Defined Numerics (see Section 2.5, "User-Defined Numerics"). Upon selecting a directory, the Configuration Tool checks to see if the specified folder is valid. A Matlab installation directory is valid if it contains a subdirectory named **`bin`**, which in turn contains a directory whose name depends on the current operating system and architecture (e.g. the target folder for a 32-bit architecture running Windows is **`win32`**). System-specific shared objects necessary for proper function of the Matlab-UDN Interface are contained in this directory. A few common Matlab installation directories are listed below.

- Windows: `C:\Program Files\Matlab`, `C:\MATLAB`
- Linux/UNIX/Mac: `userhome/matlab`, where `userhome` is the path to the current user's home directory.

Use System Look & Feel

When set to **Yes**, SNAP applications will be displayed with system-specific window decorations. When set to **No**, the platform independent look and feel will be used. Turning the system look & feel **On** or **Off** requires a restart of all client applications.

Use System Look & Feel is only available on platforms for which the Configuration Tool is able to detect a system-specific look and feel, such as Microsoft Windows®.

3.3.2. Calculation Server Properties

These properties configure the implicit local Calculation Server. The Calculation Server status can be controlled using the **Server Status** property described below.

Server Status

The Server Status property can be used to start or stop the Calculation Server, as well as obtain information about an active server. The Server Status editor is displayed in Figure 3.2, "Stopped Calculation Server" and Figure 3.3, "Started Calculation Server". The buttons on both properties are, from left to right: **Start Server**, **Stop Server**, **Show Server Status**, and **Open Log Viewer**.

Figure 3.2. Stopped Calculation Server

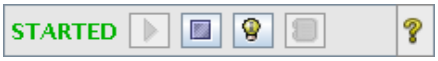


Figure 3.3. Started Calculation Server

- **Start Server** - This button activates the Calculation Server. It is only available when the Calculation Server is stopped.
- **Stop Server** - This button deactivates the Calculation Server. It is only available when the Calculation Server is running.
- **Show Server Status** - This button displays a dialog containing information about client applications connected to the currently active Calculation Server. It is only available when the Calculation Server is running.
- **Open Log Viewer** - *This feature is not yet implemented.*

Allow Remote Connections

When set to **Yes** the Calculation Server will allow connections from other users and other machines. To allow remote connections, first change **Allow Remote Connections** property to **Yes**. Then, export the server's public key to a convenient location by selecting the **Export Public Key** item from the **File** menu. This key can then be transferred to those users who require access to the server.

Note The Outlook email application may not allow filenames that end with the .key extension.

Start Server Automatically

When activated (**Yes**), the **Start Server Automatically** preference will enable the local server to startup when needed by local client applications. When not activated (**No**), the user will be prompted to startup the server instead. Client applications are considered local only if they are running on the same computer as the server (referred to as "localhost") and running as the same user as the server.

This property will enable the local server to startup automatically:

- when submitting a stream
- when browsing the local calculation server via Job Status
- when selecting a run for an animation from the local server
- when connecting an animation to the local server

Server Port Number

Defines the port number that the Calculation Server will listen on. Changing this property does not take effect until the Calculation Server has been restarted.

Max Concurrent Jobs

Sets the maximum number of tasks that the Calculation Server will execute simultaneously. Any other tasks submitted after this maximum has been reached will wait in a queue for the first available opening.

Logging Level

This parameter determines the level of detail used in writing information to the Calculation Server log file.

Maximum Log Size

The maximum size (in bytes) of the log files created by the calculation server. After reaching this limit, the current log file will be renamed **`calculation_server_1.log`** and logging will continue in a new **`calculation_server.log`**. An unlimited log file size can be set by specifying 0 for the maximum size.

Changing this property does not take effect until the Calculation Server has been restarted.

Start Server at Login

When set to **Yes** a shortcut will be placed in the **Startup** group of the user's **Start Menu**. This option is only available on Windows platforms.

3.4. Platforms

Platforms represent computer systems where applications may be executed. Each platform definition indicates either a calculation server or a High Performance Computing (HPC) environment, such as a cluster. A calculation server is a single machine that handles the execution of all tasks. An HPC environment distributes tasks to specific computing resources, typically nodes in a cluster.

3.4.1. Platform Creation

When a new platform is created, the platform creation dialog is displayed (see Figure 3.4, “Platform Creation Dialog”). This window is used to define the platform location and type. The new platform is added to the tree after pressing the dialog's **OK** button.

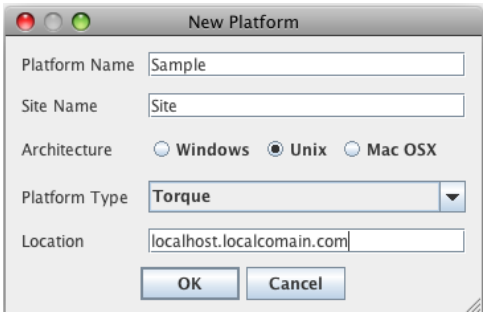


Figure 3.4. Platform Creation Dialog

Note Consult your local system administrator for the correct values to enter in this dialog.

Architecture

The architecture (i.e. operating system) of the computing system represented by this platform. Available selections include Windows, Unix, and Mac OS X. This must be correctly specified so that the stream manager can use the correct line endings when writing files and the correct file path separators when locating files.

Type

The name of the Platform Module that will be used to manage access to this platform. This can either be a calculation server or one of the HPC-specific platform modules loaded by the Configuration Tool at start-up.

Location

The location of this platform as defined by the Platform Module. For a calculation server, this is the host name and port on which the server is accessed, in the form "**`hostname:port`**" (sans quotes). Otherwise, this value is interpreted in a manner specific to the selected platform module. Typically this is the host name or IP address of the HPC resource. For the Torque platform reference implementation, this value is the hostname of the main cluster node.

3.4.2. Platform Properties

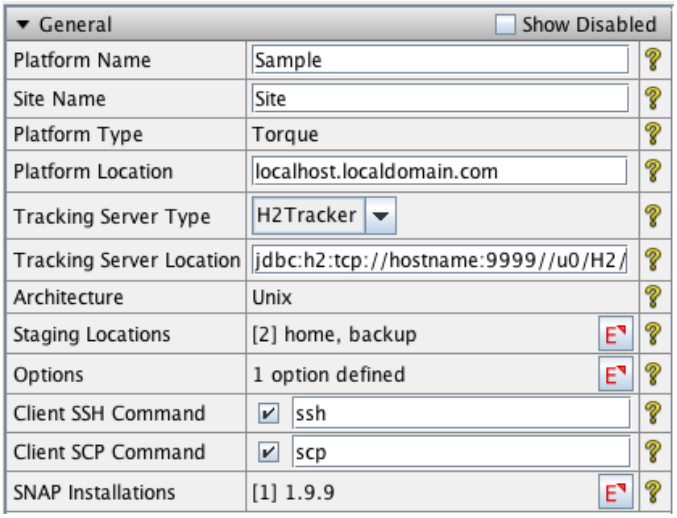


Figure 3.5. Platform Properties

Note Consult your local system administrator for the correct values to enter for these properties.

Platform Type

The type of platform module that will be used to manage access to this platform. This is defined when the platform is created and cannot be changed afterwards. See Section 3.4.1, “Platform Creation” for more information on this property.

Platform Location

The location of this platform as defined by the Platform Module. See Section 3.4.1, “Platform Creation” for more information on this property.

Tracking Server Type

Defines the type of tracking database server associated with this platform. As a stream executes, the tracking server is updated with status information. The H2Tracker reference implementation

is provided with Job Stream; this tracker stores stream status information in an H2 database. This property is not available for Calculation Server platforms.

Tracking Server Location

The JDBC URL that identifies the location of the tracking database server. This property is not available for Calculation Server platforms.

Staging Locations

The locations where intermediate files, such as the stream definition, will be stored while a job stream is running. The **Edit** button for this property opens the Staging Locations dialog, shown in Figure 3.6, "Staging Locations Dialog". This dialog allows the user to define the available staging locations for the platform. The **`${USERID}`** keyword can be used to indicate the current user name.

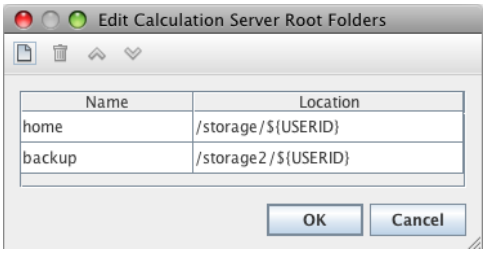


Figure 3.6. Staging Locations Dialog

3.5. Applications

Applications represent executables that may be launched as part of a job stream. Each application corresponds to a job step that may be added to the stream. Consequently, each step represents the execution of one or more tasks utilizing that executable. When adding a new application, the **Create Application** dialog is displayed, as shown in Figure 3.7, "Create Application Dialog". Within this dialog, the type of application being created is selected. Available application types are loaded from the SNAP installation `plugins` directory.

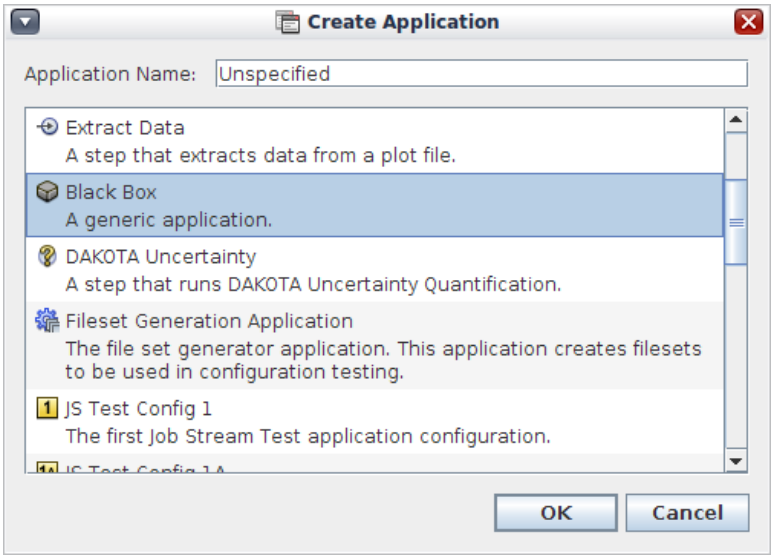


Figure 3.7. Create Application Dialog

Applications have the following properties as shown in Figure 3.8, “Application Properties”.

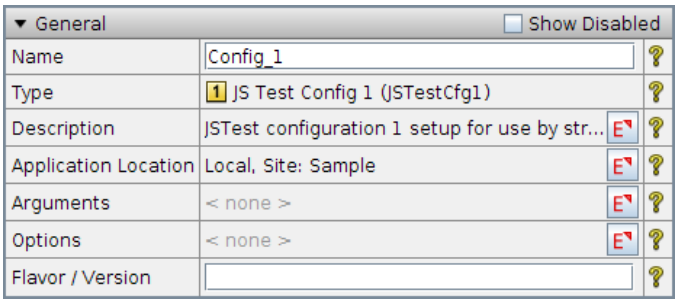


Figure 3.8. Application Properties

Name

The name by which the application is identified within a stream.

Note Steps identify the application used to run them by name. Changing the name of an application referenced by a step will invalidate streams used in this SNAP configuration until the application references are manually adjusted.

Type

The application type selected when the application was created. This value cannot be changed.

Description

An arbitrary description of the application. This text is shown in dialogs used to select the application associated with a particular job step.

Arguments

This is the list of command-line arguments that will be applied to the application when executed.

Options

Application plug-ins may provide additional options that can be configured through this property. Each additional option is specified as a name/value pair, which is then interpreted by the step as needed. Consult the application documentation for any additional options provided for the step.

Flavor/Version

The flavor and/or version of the application. This optional value serves as an additional indication to the application of how this executable should be handled.

Application Locations

The **Application Locations** attribute group holds all the properties related to specifying the location of the application, both on the local machine and on other platforms.

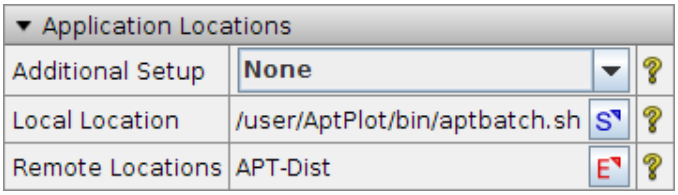


Figure 3.9. Application Location Properties

The following properties are available.

- **Additional Setup** - Specifies any additional setup required for the application. It has the following values:
  - **None** - The most common setup, this indicates an application that will be executed from its specified location.
  - **Retrieve Executable** - The executable at the specified location will be copied to either the working directory of associated steps or the staging location (pending **Target Location**).
  - **Extract ZIP Archive** - The ZIP archive at the specified location will be extracted to either the working directory of associated steps or the staging location (pending **Target Location**).
  - **Run Installer** - The installer at the specified location will be executed to either the working directory of associated steps or the staging location (pending **Target Location**).

- **Local Location**, **Retrieve From**, **ZIP Location**, **Installer Location** - The location of the executable, ZIP archive, or installer specified by the application.

- **Target Location** - Indicates whether the file specified by the Location property above is managed. With **Working Directory**, the file will be copied to, extracted at, or installed at the working directory of associated steps. For **Staging Location**, the file will be handled at the stream staging location. This property is only available for **Additional Setup** values other than **None**.

- **Executable Name** - The name of the executable. This value has different meanings depending on the value of **Additional Setup**. For **Retrieve Executable**, this is the name that the executable will be given when copied to its destination. For **Extract ZIP Archive** and **Run Installer**, this value specifies a relative path to the executable (including the name of the executable itself) that will be created after extraction or installation. This property is only available for **Additional Setup** values other than **None**.

- **Remote Locations** - Specifies the **Location** of the executable, archive, or installer on other platforms. This property is only available when at least one Platform is defined whose **Platform Type** is not set to **Calculation Server**.

3.6. Black Box Applications

Black box applications are used to add simple applications, such as PERL scripts or other custom executables, to a job stream without building a full application plug-in. Black boxes are intended to be extremely simple. Anything that requires more complicated behavior should be built as an application plug-in.

Black Box application definitions have the following properties, as shown in Figure 3.10, “Black Box Properties”. All properties serve the same purpose as those described in Section 3.5, “Applications”, with the following additional attributes.

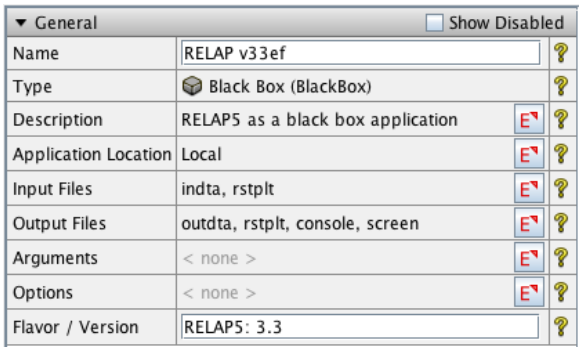


Figure 3.10. Black Box Properties

Input Files

The input files must be defined for a black box application. Each input file required by the black box application must be defined, with a label, runtime file name, and file type specified.

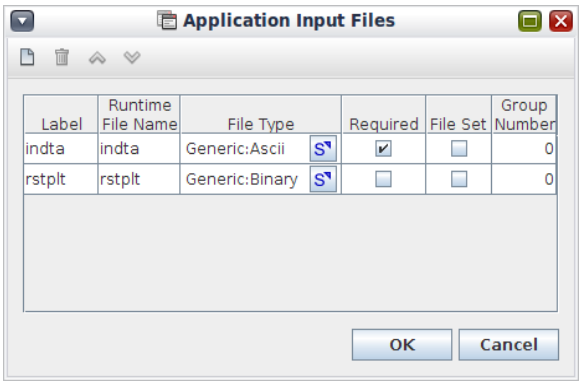


Figure 3.11. Black Box Inputs

Output Files

The output files dialog is used to specify the application outputs for a black box. The output files from the application may be used in downstream job steps and can be copied to a custom storage location after the job step completes. Each output file of a black box must define the output label, runtime filename, and file type.

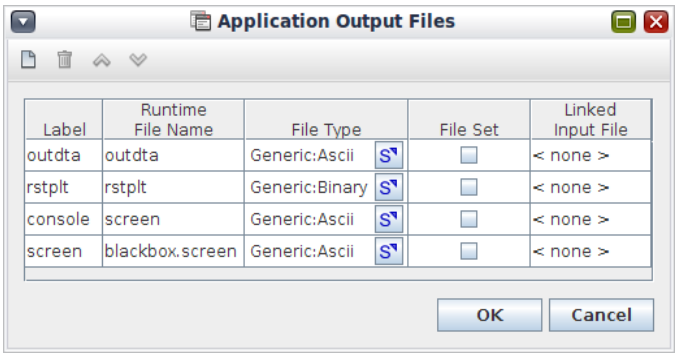


Figure 3.12. Black Box Outputs

Chapter 4. Job Status

The Job Status application is a client application which details the status of jobs submitted to a Calculation Server. It also provides support for adding connections to remote Calculation Servers and importing local files into the list of jobs available on the local Calculation Server.

4.1. Job List

The Job Status User Interface (UI) is composed of a Job Navigator on the Left and a Job Panel on the right. These are placed in a **Job List** tab at the top of the application, used to differentiate between the core Job Status UI and Job Consoles (see Section 4.3, “Job Consoles”), which are opened as additional tabs. This section describes the **Job List** tab.

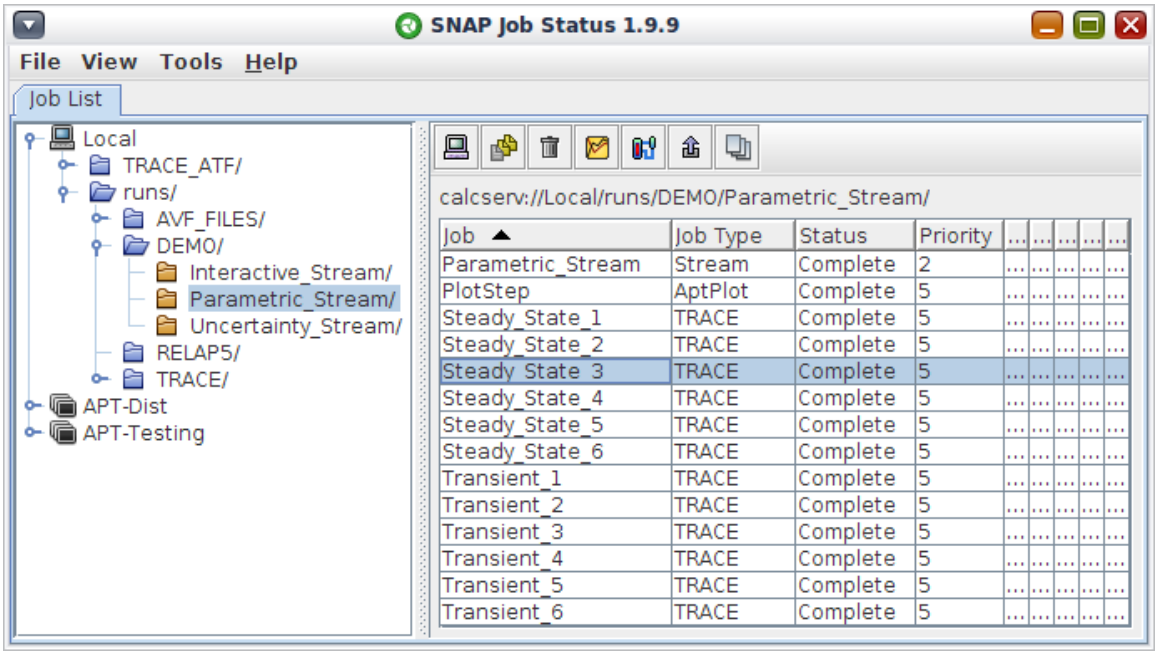


Figure 4.1. The Job Status UI

The Navigator to the left represents a logical hierarchy of platforms (a.k.a. servers) and folders. Each platform can be expanded into a list of mounted folders, which can in turn be expanded into a list of sub-folders. Selecting an item in this view changes the contents of the Job Panel, displaying the jobs run at that location.

Note Platforms are added and removed in the Configuration Tool (see Section 3.4, “Platforms”).

Right-click pop-up menus are available for each platform and folder displayed in the Job Navigator. The contents of each of these menus are described below.

The Calculation Server's pop-up menu contains the following menu items:

- **Connect** - Attempt to make a connection to the server. This may show a failed icon if the server is unavailable or the specified server key is invalid.
- **Disconnect** - Disconnects from the server.

- **Reconnect** - Disconnects from the server and attempts to reconnect. This option is equivalent to using the **Disconnect** and **Connect** options above.

- **Mount Root Folder** - Opens a file selection dialog for specifying a folder on the local Calculation Server which should be mounted as a root folder. For more information on this functionality, refer to Section 4.2, “Root Folders and Sub-Folders”.

  Note Root folders cannot be mounted or unmounted for remote Calculation Servers.

The folder pop-up menu contains the following menu items:

- **Create Folder** - Creates a new sub-folder within the selected folder. Selecting this option opens a dialog which requests a name for the new folder.

- **Rename Folder** - Opens a dialog for renaming the folder. This option is not available for root folders.

- **Remove Folder** - Removes a folder from the Calculation Server. This option is not available for root and non-empty folders.

- **Import Completed Job** - Imports a job from the selected folder into the Calculation Server. For more information, refer to Section 4.5, “Importing Jobs”.

- **Unmount [folder name]/** - Unmounts the folder, removing it from the Job Navigator. This option is only available to folders that have been mounted as root folders; it is not available for subfolders.

4.1.1. Menu Items

The Job Status menu bar contains several menus, described here. The **File** menu contains only one item, **Exit**, which closes JobStatus. The **View** menu contains one check-box per column in the Job Panel table. Each check-box toggles the corresponding column's visible status.

The **Tools** menu contains entries specific to the currently selected job. It has the following items:

- **Console Output** - Displays the Console View detailing the status of the selected job. This option is only available for calculations that are currently active (i.e. running, interactive, paused, etc.)

- **View Files** - this menu is used to view or open files associated with the given job. The secondary sub-menus, described below, organizes the file types supported by Job Status.

  - **Documents** - word processor documents, such as OpenDocument Format (ODF) files. Files opened from this category are opened in the system's native document viewer associated with the given file's type.

  - **Images** - image files, such as JPG, GIF, and PNG files. Images are opened with the system's native image viewer.

  - **PDFs** - Portable Document Format files, a platform-independent file type used to preserve exact document formatting between machines, operating systems, and individual viewers. PDFs are opened with the system's native PDF viewer.

- **Plot Files** - job data files, containing results of the calculation as multiplexed or demultiplexed plot data. Plot files are opened in AptPlot (see Section 3.3.1, "General Properties" for info on configuration AptPlot with SNAP). After loading the file, AptPlot will display a list of channels in the file.

- **Text Files** - ASCII text files. Opening a text file opens the File Viewer, described in Section 4.4, "File Viewer".

Note With the exception of **Text Files**, files cannot be displayed for remote Calculation Servers.

- **Plot** - Opens the selected job file for plotting. The application used to open the file is specified by the Plotting Tool property in the Configuration Tool (see Section 3.3.2, "Calculation Server Properties"). This option is only available when the job has been loaded.

- **Send Command** - Displays a dialog for sending an interactive command to the selected job. This option is only available for interactive calculations.

- **Terminate** - Terminates the selected job, ending the underlying process. This action sends an early termination signal to the process and does not attempt to shut it down gracefully. This option is only available for jobs that are currently active.

- **Change Priority** - Changes the priority of the selected job. This priority determines the order in which jobs are started by the Calculation Server. Otherwise, jobs are selected based on the order of their submission into the queue. Larger numbers correspond to higher priorities. This option is only available for jobs waiting to be started in the job queue.

- **Load Data** - Loads a completed job into memory. Most options in the **Tools** menu can only be activated after a job has been loaded.

- **Unload Data** - Unloads a job, freeing any system resources, (such as cached plot records) used by the job.

- **Animate** - Animates the selected job in the Model Editor with its associated post-processing MED. Animate functions identically to **Animate With** when no animation model has been associated with the currently selected job.

- **Animate With** - Animates the selected job in the Model Editor with a post-processing MED selected in a file browser. After selecting a file via Animate With, the same MED can be animated for that task with the **Animate** item.

- **Release** - Releasing a job removes the Calculation Job File (.cjf) but does not remove any of the job's data files.

- **Delete** - Removes the job from the Calculation Server. This removes the Calculation Job File (*.cjf) as well as any data files associated with the job.

Note Most **Tools** options require that a job first be loaded into the calculation server.

The **Help** menu contains the following items:

- **Help Topics** - Displays the Job Status SNAP help set.

- **Included Technologies** - Displays a dialog listing various technologies included in SNAP and their corresponding licenses.

4.1.2. Job Panel Toolbar

The toolbar in the Job Panel provides several tools for the currently selected job in the table. The provided tools can change depending on the currently selected job, its type, and its status. Below is a description of the available tools.

 **Job Console** - Opens a Job Console for the selected job. See Section 4.3, “Job Consoles” for a description of the Job Console.

 **Terminate** - Terminates the selected job, ending the underlying process. This action sends an early termination signal to the process and does not attempt to shut it down gracefully. This option is only available for jobs that are currently active.

 **Send Command** - Displays a dialog for sending an interactive command to the selected job. This option is only available for interactive calculations.

 **Release** - Releasing a job removes the Calculation Job File (.cjf) but does not remove any of the job's data files.

 **Delete** - Removes the job from the Calculation Server. This removes the Calculation Job File (*.cjf) as well as any data files associated with the job.

 **Plot** - Opens the selected job file for plotting. The application used to open the file is specified by the Plotting Tool property in the Configuration Tool (see Section 3.3.2, “Calculation Server Properties”). This option is only available when the job has been loaded.

 **Animate** - Animates the selected job in the Model Editor with its associated post-processing MED. If no suitable MED has been associated with the job, a file browser will be displayed to select an appropriate model.

 **Load Data** - Loads a completed job into memory. Most options in the **Tools** menu can only be activated after a job has been loaded.

 **Unload Data** - Unloads a job, freeing any system resources, (such as cached plot records) used by the job.

 **View Files** - opens a pop-up menu used to view or open files associated with the given job. See Section 4.1.1, “Menu Items” for a description of the **View Files** menus.

4.1.3. Status Indicators

Several status indicators can be placed over folders in the Navigator to indicate activity, either in the folder or in some sub-folder.

 **Running** - At least one task is running in the indicated folder. This status has the highest priority of the various status indicators: other jobs in the folder could be paused or waiting, and sub-folders could be active.

Paused - At least one interactive task is paused in the indicated folder. This status has higher priority than **Waiting** and **Active**:

Waiting - At least one task is waiting to begin execution in the indicated folder. This status has higher priority than **Active**.

Active/Loaded - Indicates that a job in the folder has either been loaded or that some sub-folder has some type of activity, such as running, paused, waiting, or loaded jobs. This is the lowest priority indicator: any folder containing running, paused, or waiting jobs will not display the **Active** status.

Folders without any of these indicators are *inactive*: no jobs in the folder or any of its sub-folders are running, paused, waiting, or loaded.

4.2. Root Folders and Sub-Folders

The Calculation Server includes support for multiple Root Folders (with sub-folders) that can contain any number of jobs. These Root Folders can be mounted in Job Status by selecting the Mount Root Folder item from the right-click pop-up menu of the local Calculation Server node, **Local**. These folders can be unmounted by selecting the **Unmount** item from the right-click pop-up menu for the folder. Unmounting a folder does not remove any data contained in that folder.

To mount or unmount folders, the server's node must first be expanded to make a connection to the server. If no root folders are mounted on the Local server, a prompt will be shown allowing the user to mount one (Figure 4.2, “No Root Folders Prompt”).

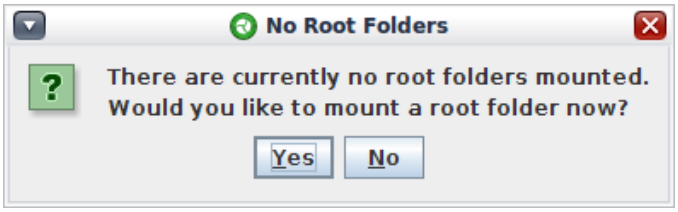


Figure 4.2. No Root Folders Prompt

Sub-folders can be created by selecting the parent folder and selecting the **Create Folder** item from the right-click pop-up menu. These sub-folders can also be renamed or deleted by selecting the **Rename Folder** or **Delete Folder** item from the right-click pop-up menu of the folder.

Note Only completely empty folders can be removed through Job Status.

4.3. Job Consoles

A Job Console for a specific task can be opened by pressing the **Job Console** button in the toolbar, or by double-clicking the row in the Job Panel table. A sample console is shown in Figure 4.3, “Sample Job Console for a TRACE Job”. The central area is the heart of any Job Console, displaying the console output of the job as it runs. Additional features are described below.

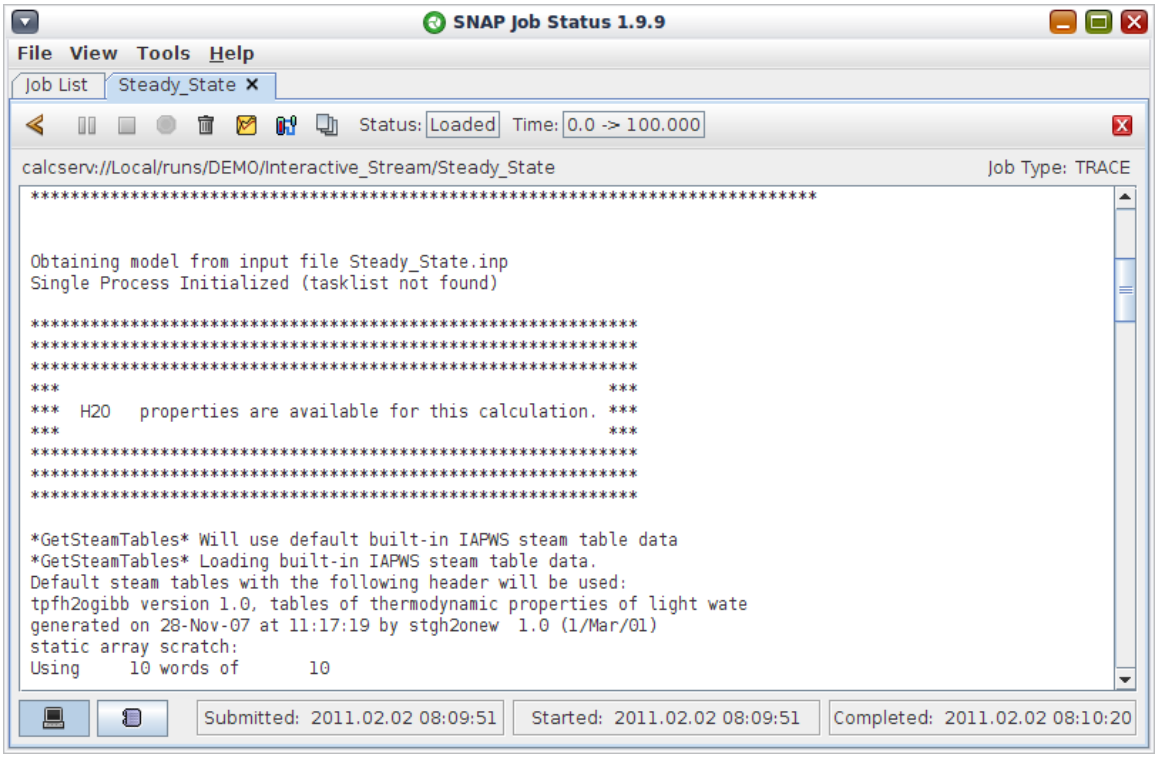


Figure 4.3. Sample Job Console for a TRACE Job

The actions in the console toolbar function identically to those described in Section 4.1.2, "Job Panel Toolbar", with the following additional items.

Select Job - This button causes Job Status to switch to the **Job List** tab, expand the folder path to location of the job represented by this console, and select the job in the Job Panel table.

Play - Resume calculation for paused jobs. This button is only available for interactive jobs.

Pause - Suspends calculation for running jobs. This button is only available for interactive jobs.

Stop - End the current job. Unlike terminate, this button sends an interactive command to the job telling it to complete, rather than killing the process. This button is only available for interactive jobs.

Close Console - Closes this Job Console tab. Consoles can also be closed with the **X** button in the console tab. Closing the Job Console does not terminate or stop the underlying job.

Job Consoles for most tasks will display a pair of toggle buttons in the lower left corner used to control the output displayed in the central text area.

Screen - When selected, the console output of the underlying job process will be displayed.

Task - When displayed, the 'task log' will be displayed. This log details the execution of the task relative to its parent job stream.

4.4. File Viewer

When displaying a text file, the File Viewer is shown, as seen in Figure 4.4, "File Viewer displaying a TRACE Output". This window can display the contents of text files on local or remote calculation servers, and provides the additional functionality described below.

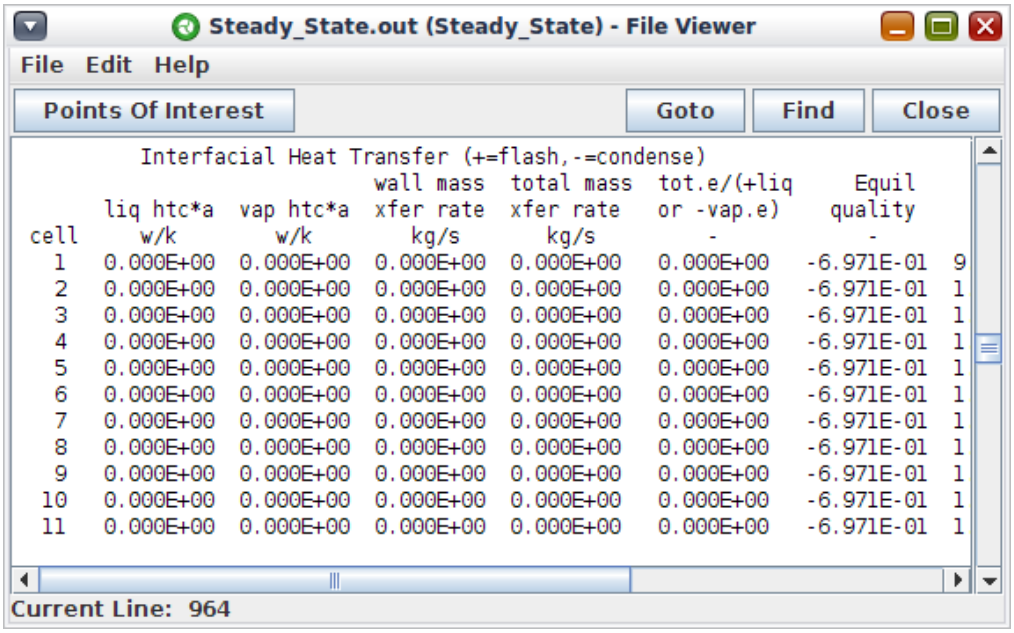


Figure 4.4. File Viewer displaying a TRACE Output

The **Points of Interest** button opens a window used to jump to areas of interest in the text file. Points are interest are defined on a plug-in by plug-in basis. The **Goto** button can be used to jump to a specific line in the file, with line numbers starting at 1. The **Find** button allows searching the file for a specific piece of text.

4.5. Importing Jobs

Job Status allows importing jobs for use by SNAP. The primary motivation for importing a job is typically to use it as a source of data in a post-processing model. In addition, all other job management features provided by SNAP are available for imported jobs.

To import a job, its corresponding files must first reside in a root folder or one of its sub-folders. The actual names of the files and their organization vary by plug-in, but typically the files should all reside in a sub-folder with the name of the job that will be imported. So if the hypothetical job `SteadyState` in the folder `DEMO` is to be imported, its files will be in the folder `DEMO/SteadyState`.

Selecting **Import Completed Job** from the parent folder's right-click pop-up menu begins the import process, opening the dialog shown in Figure 4.5, "Job Import: First Step". This dialog is a *wizard*: choices made at each step determine the results of the next step. At any time, steps can be traversed with the **Back** and **Next** buttons.

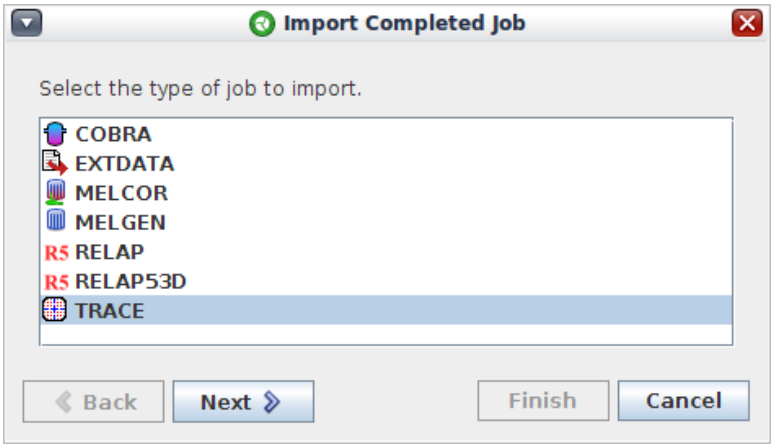


Figure 4.5. Job Import: First Step

The first step of importing a job is to indicate its type. All plug-ins that support importing jobs will be listed. Pressing the **Next** button moves to the second step, as shown in Figure 4.6, "Job Import: Second Step".

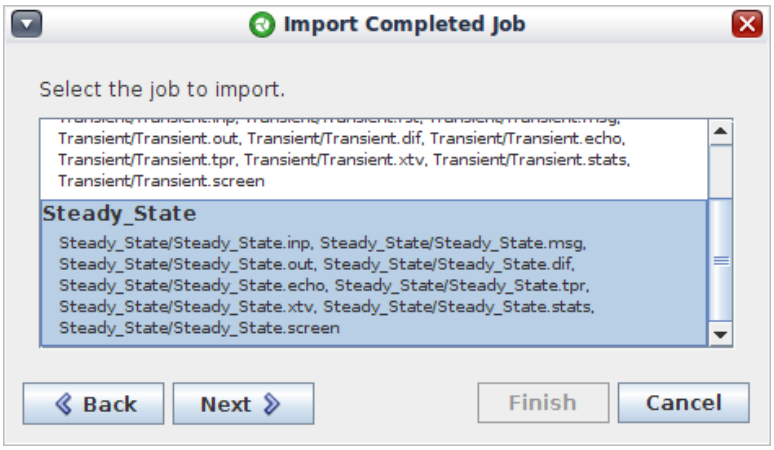


Figure 4.6. Job Import: Second Step

In the second step, any combination of files that the plug-in job importer recognize as a potential job will be listed. In the figure, a job titled **Steady_State** has been recognized by the TRACE importer, and all the corresponding files in the `State_State` folder are listed. The **New_Undefined_Job** entry at the top of the list can be used to enter a job manually. Undefined jobs will not have the benefit of the default values created by selecting a job from the list: each entry will have to be manually specified.

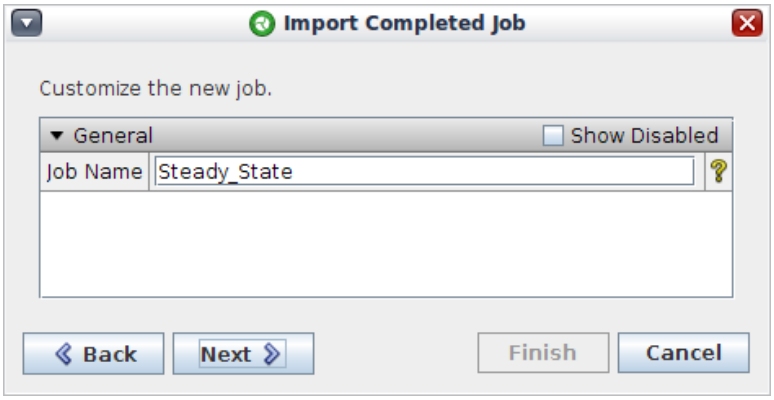


Figure 4.7. Job Import: Third Step

The third step, shown in Figure 4.7, “Job Import: Third Step”, is used to specify the name of the new imported job. Any additional properties required by the plug-in job importer will also be displayed here. Unless importing a new, undefined job, changing the name of the imported job is strongly recommended against.

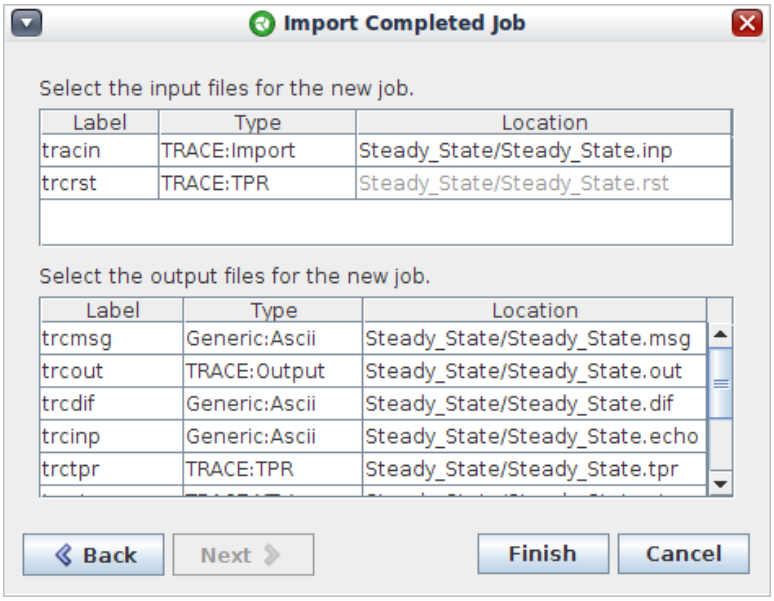


Figure 4.8. Job Import: Final Step

The fourth and final step, shown in Figure 4.8, “Job Import: Final Step”, is used to specify the files that are part of the imported job. The two tables indicate the files defined as inputs and outputs. A grayed-out **Location** indicates that the expected file is not present.

The **Location** column can be used to change the file for each entry. Pressing the **Select** button in the Location cell editor opens **Select File** dialog, shown in Figure 4.9, “Job Import: Select File Dialog”, used to select which available file is used for the given input/output. Note that files can be left unspecified as needed.

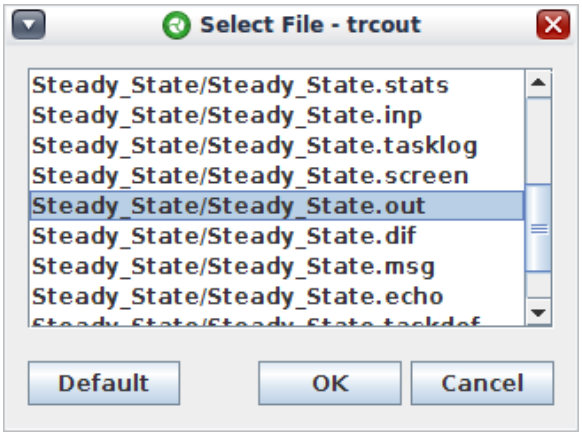


Figure 4.9. Job Import: Select File Dialog

Once files have been specified as needed for the new job, pressing the **Finish** button will complete the import.

Chapter 5. Job Streams

A Job Stream is a component in a pre-processor plug-in model that defines the work flow for executing analysis applications. A Job Stream is defined by stream steps (single application executions), files (primarily used as application input), switches (logical operations used to control job stream work flow), and reference models. A model may have more than one job stream, each stream defining a different set of applications to be executed.

5.1. Job Stream Basics

This section describes basic job stream concepts referred to regularly throughout this chapter.

5.1.1. Stream Steps

Stream steps represent applications that will be executed. Each application has a number of input files that must be supplied, and produces a number of output files after execution. The outputs of a stream step can be used to supply the inputs of another application. The resulting chain of applications is what makes up the job stream. The input files for the step can come from previous job steps, external file references, or from the model node. The stream element that represents the file inside the job stream is called the file source.

The inputs for a job step may be single files, or file sets. Each input is identified with a label that appears inside the step when it is rendered inside the 2D view, and when the inputs are displayed inside a property dialog. The file type on each input is used to restrict what files may be selected as a file source.

Parametric Stream Steps

If a single file input of a step is supplied multiple file sources, the step will be executed multiple times. This is called a parametric step. Each of the resulting executions of the step are called tasks. The properties that define the differences between the parametric tasks are the parametric keywords. These keywords are used to define the different tasks on the job step. The total listing of tasks for a job step, organized by parametric keyword, is called the file sequence of that step. Individual tasks may be filtered out from execution. A specific combination of parametric keywords may be excluded directly, or a numeric filter can be used to allow a shared numeric value to define which tasks should be executed.

5.1.2. External File Sources

Job streams require the ability to select files from outside of the Model Editor to include as the inputs for job steps. There are two external file sources: external file, and file set. The external file represents a single file that will be included in the job stream submission. The file set represents multiple files joined together. The protocols for selecting the files use a customizable interface. By default the local file selection and calculation server file selection routines are included.

A file set can be used to generate a parametric job step by using it as the file source for a single file input. The files that are included in the set are used to generate the file sequence that defines the parametric keywords in this case.

5.1.3. Model Nodes

Job streams use model nodes to include the files generated by the model editor for a model. Simple job streams will only require a single model node. Inside an engineering template model, these nodes are used to specify the base models. In this case the model nodes represent the reference models, not the model in which the job stream is defined. Additionally, a model node may be used to represent a restart case.

The model nodes are used to supply the parametric model files produced when doing a parametric study using the different stream types. The different stream types will produce a set of input files for each of the files produced by a model. The properties used to define the parametric model define the parametric keywords that will be included with any job step that uses a parametric model node as a file source. This list of parametric keywords is used to generate the file sequence from parametric model nodes.

5.1.4. Input Switches

Input switches allow a job stream to determine which file source to apply to a job step based on the value from a shared numeric. The switch consists of a set of input branches, that must each define the same type of input file. Each input branch defines the set of shared numeric values when that branch should be used. The first input branch whose numeric conditions evaluates to TRUE will be used as the input for any stream step that uses the switch as a file source. In the case where a parametric stream type is defined, the switch will be evaluated for each of the generated cases.

5.2. Creating Job Streams

Creating a simple Job Stream can be summed up with the following process:

1. Create a Job Stream.
2. Create one or more Stream Steps in the stream.
3. Connect the model to the first Stream Step.
4. Connect the output of each Stream Step to the input of the next Stream Step.
5. Connect the required external files to each Stream Step.

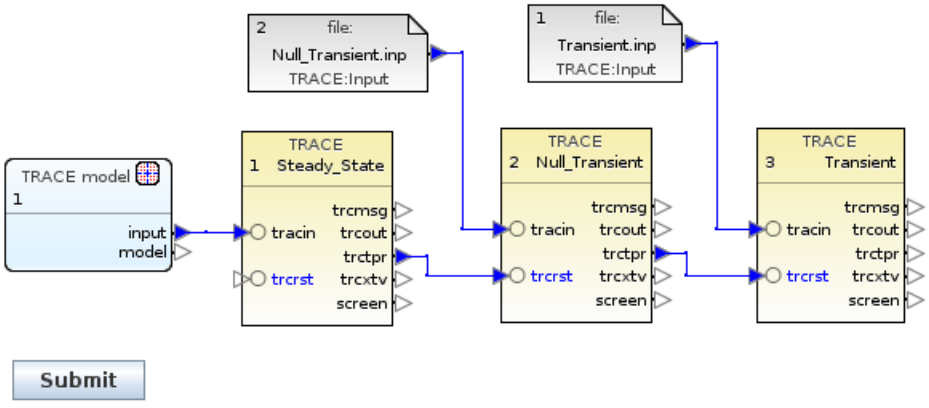


Figure 5.1. Simple TRACE Job Stream

This section will focus on how this process works in the context of a simple TRACE Job Stream. The stream in Figure 5.1, “Simple TRACE Job Stream” is made up of three executions of the TRACE analysis code in a series. The **Null Transient** and **Transient** executions use an output file from the previous step (trctpr) as an input (trcrst).

5.2.1. Creating a Job Stream

Job Streams can either be created in the Navigator or inserted directly into a 2D view. To create a Job Stream in the Navigator, right-click on the **Job Streams** node and select the **New** menu item. To insert one directly into a 2D view, first press the down-arrow next to the insertion tool to pop up the list of available component types, then select **Job Streams** from the list. This will activate the Insertion Tool for inserting Job Streams. Click anywhere in a view and a Job Stream will be inserted with a job stream drawing (a customizable submit button).

The **Select Stream Type** dialog will appear as soon as a Job Stream is created. This dialog lists the available stream types (Basic, Numeric Combination, etc.). A stream's type determines the kinds of parametric studies that can be performed using the stream. The **Basic** type is a good first choice as it does not include the extra complexity of parametrics. A different type may be selected at any time. Refer to section Section 5.3.1, “Job Streams” for a detailed description of the available stream types.

After a type is selected (and the **OK** button is pressed) the newly created stream will be added to the model and selected. The properties of the new stream will be displayed in the Main Properties View, as shown below.

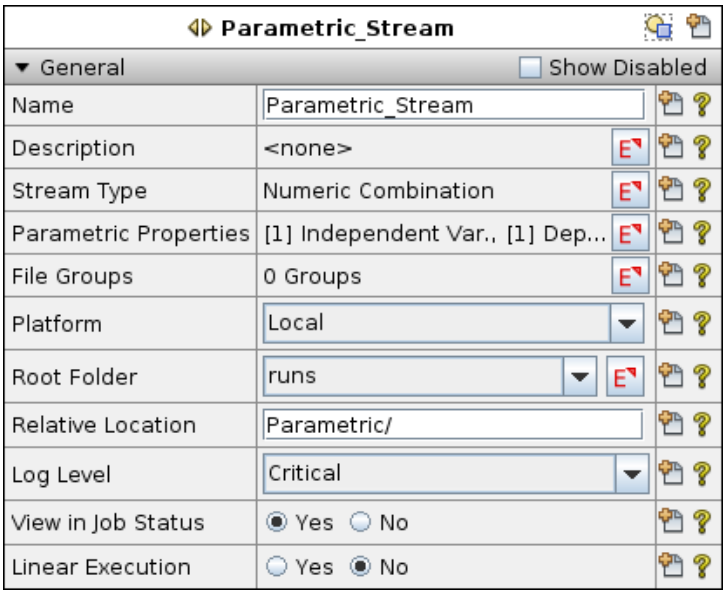


Figure 5.2. Job Stream Properties

Complete descriptions of these properties are available in Section 5.3.1, “Job Streams”, however, the following properties require special note.

- **Name** – The files and folders created for the stream are named using the Job Stream's name as the base. It should be descriptive enough to identify the purpose of the stream without being too long for a folder name.

- **Platform** – The computing platform that this stream will be submitted to by default. For most users only the **Local** platform (the local Calculation Server) will be available.
- **Root Folder** – The location on the Calculation Server where this stream will be placed. The **E** (Edit) button to the right of the root folder selector can be used to add/remove folders to the local Calculation Server.
- **Relative Location** – The location of the stream within the selected Root Folder (above). This relative location can be used to organize the set of streams submitted to a particular location. Relative Location paths must use the slash character '/' as a folder separator; a backslash cannot be used.

5.2.2. Creating Stream Steps

Like Job Streams, Stream Steps can be created either in the Navigator or within a 2D View. In either case, steps must be created within the context of a Job Stream.

To create a new Stream Step in the Navigator, first expand the **Job Streams** node. This will show the list of Job Streams in the model. Then expand the node for the stream to which the step should be added. Finally, right-click on the **Stream Steps** node and select the **New** menu item.

Inserting a new Stream Step into a 2D view is much the same as inserting a stream. First, press the down-arrow next to the insertion tool to pop up the list of available component types. Then select **Stream Steps** from the list. This will activate the Insertion Tool for inserting new Stream Steps. Click anywhere in a view and a new Stream Step will be inserted.

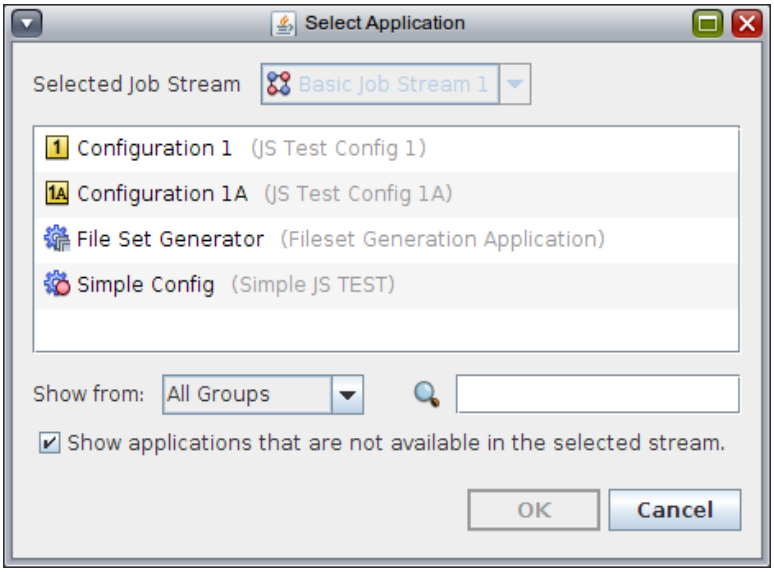


Figure 5.3. Insert Job Step Dialog

The **Insert Job Step** dialog will appear as soon as the Stream Step is created. This dialog lists the applications that are available to the selected Job Stream. New applications can be defined in the Configuration Tool as described in Section 3.5, "Applications". The search field at the bottom of the dialog limits the displayed applications to those that match the search string. Selecting an application and pressing the **OK** button will insert a new Stream Step into the Job Stream shown at the top of the dialog.

A complete description of the properties of a Stream Step is given in Section 5.3.3, “Stream Steps”.

5.2.3. Connecting the Model

The current model (i.e. the TRACE model in this example) is represented in a Job Stream by a "Model Node." Streams can have any number of model nodes that each provide a different view of the model. For example: if a TRACE model included several restart "cases", then a model node could be created for each case. This would allow a Stream Step to use a case by connecting to the model rather than referencing an external file. Newly created Job Streams start with one Model Node. Additional nodes can be added at any time. Model Nodes are discussed in greater detail in Section 5.3.2, “Model Nodes”.

Model Nodes can be added to a view from the Navigator by using drag-n-drop. First, expand the **Job Streams** node, then the node for the stream. Expand **Model Nodes** under the stream, then drag the Model Node to the 2D view.

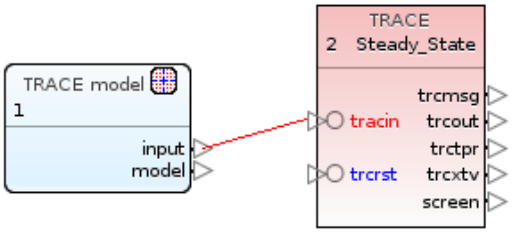


Figure 5.4. Connecting a Model Node

Once the Model Node is added to a 2D view it can easily be connected to the input of a Stream Step. Just activate the **Connection Tool** in the view, click on the model node's connector then click on the step's connector. In this example, first click on the model node's **input** connector and then on the TRACE step's **tracin** (i.e. "TRACE Input") connector. Once connected, the TRACE step will retrieve its input directly from the current model.

5.2.4. Connecting Stream Steps

The essence of a Job Stream is the flow of data through a set of Stream Steps. Each step takes its inputs, executes an application or process, then passes one or more of its outputs to the next step in the stream. In the example below, two of the TRACE steps get one of their inputs from the previous step. More specifically they use the previous step's **trctpr** (TRACE Portable Restart) file as their **trcrst** (TRACE Restart File) input.

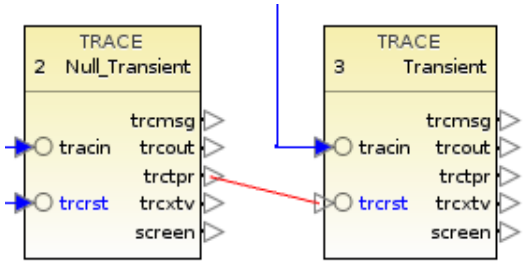


Figure 5.5. Connecting Stream Steps

Stream Steps are connected to steps in the same way that model nodes are connected to steps. Activate the **Connection Tool** in the view, click on the first step's output connector, then click on the next step's input connector. In this specific example, the output connector is **trctpr** and the input is **trcrst**. Because they are connected, the second step is now dependent on the first. This means that the second step will not execute until the first is complete and the trctpr file is available.

5.2.5. Connecting External Files

External file components are a Job Stream's method of handling files that are not managed directly by a model and cannot be represented by a model node. These files are represented as either an External File (for a single file) or a File Set (for multiple files).

External files can be created in the same way as Steps or model nodes. That is, either using the New menu item in the Navigator or the Insertion Tool directly in a 2D View. Once created, external files can be connected to steps in the same way as Model Nodes or other steps; connect the output connector of the file to the input connector of the step.

The initial connection from an External File to a step also sets the **File Type** property of the file. This type determines what inputs the file can be connected to as well as how the file will be handled by the Job Stream system.

Each External File should be named by editing its **Name** attribute. Note that this is the name of the External File component, not the name of the file it refers to. This allows the External File **Name** to indicate the function of the file rather than the on-disk name. For example: the External File can be named "Null_Transient_Input" rather than "w4loopnt.inp."

A complete description of the properties of an External File is given in section Section 5.3.4, “Files and File Sets”.

5.3. Job Stream Components

This section focuses on the components that make up a job stream and the properties of each. These components are covered individually in the following sections.

1. **Job Streams** - The central component of the job stream system. Each of the following elements are created and used as part of a particular Job Stream component. The properties of a Job Stream component are described in Section 5.3.1, “Job Streams”.

2. **Model Nodes** - Model Nodes represent a currently open model used by a Job Stream. In most cases, this is also the model that contains the Job Stream. The Model Node component is described in Section 5.3.2, “Model Nodes”.

3. **Stream Steps** - Stream steps represent applications that will be executed by the job stream. The properties of stream steps are described in Section 5.3.3, “Stream Steps”.

4. **Files and File Sets** - External File and File Set components represent files and sets of files that are used in a job stream but are not provided by a model. These external file references are described in Section 5.3.4, “Files and File Sets”.

5. **Input Switches** - Input Switches allow the flow of a job stream to branch based on the value of a user-defined numeric. The creation and use of input switches is explained in Section 5.3.5, “Input Switches”.

6. **Stream Types** - The stream type is a direct property of the Job Stream component that determines the types of parametric manipulations that can be performed on the stream. The Stream Type property of the Job Stream component is described in Section 5.3.1, "Job Streams" below. The available stream types are described in Section 5.3.6, "Stream Types".

Refer to Section 5.2, "Creating Job Streams" for an overview of how to create and connect complete job streams.

5.3.1. Job Streams

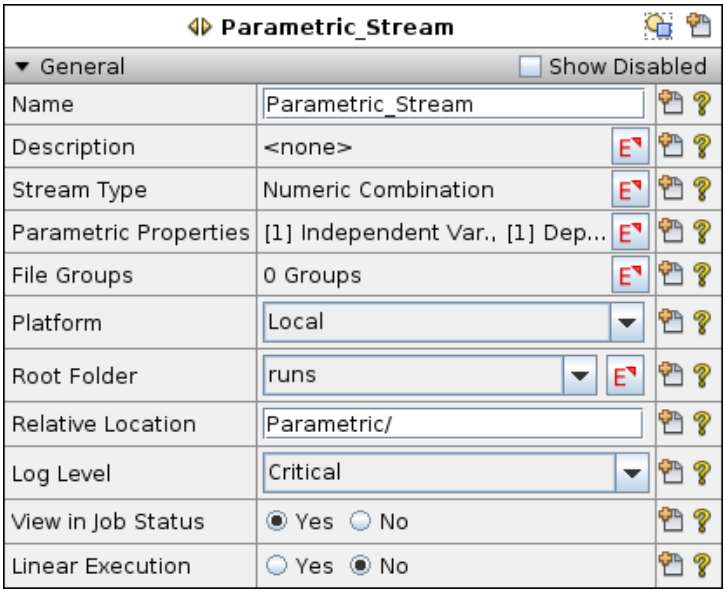


Figure 5.6. Job Stream Properties

Job Stream components are the central piece of the job stream system. They include the set of steps, the files that will be used, and the conditional logic that determines which steps will be executed. Figure 5.6, "Job Stream Properties" shows the properties of a typical a Job Stream. Properties in this list are shown or hidden based on the values of other properties or the stream type. Each of these properties will be described in detail below.

Stream Type

A stream's type determines the types of parametric manipulations that can be performed on the stream.

The Stream Type Selection Dialog opens when a new job stream is created. This dialog allows the user to select the type of job stream by displaying a list of the supported stream types that are currently loaded.

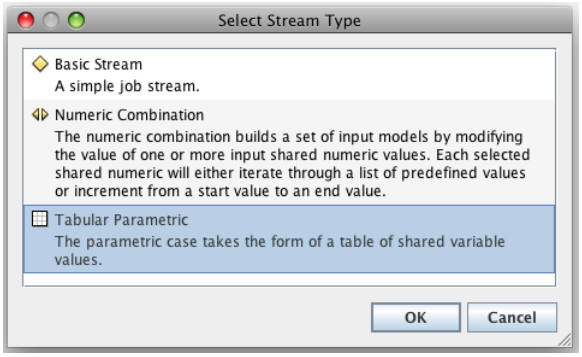


Figure 5.7. Stream Type Dialog

The following stream types are always available to new or existing Job Streams. Additional stream types may be available depending on the SNAP installation.

- **Basic** - This is the simplest of the stream types and does not allow parametrics of any sort. This type can be used when parametric features are not required.

- **Numeric Combination** - A Numeric Combination stream builds a set of input models by modifying the value of one or more input shared numeric values. This stream type is described in detail in section Section 5.3.6.1, “Numeric Combination”.

- **Tabular** - A Tabular Parametric stream defines parametric tasks using a table of shared variable values. This stream type is described in more detail in section Section 5.3.6.2, “Tabular Parametric”.

Note The Stream Type can be changed at any time.

Parametric Properties

The Parametric Properties editor provides access to the additional parametric stream configuration provided by the stream type. The editor launched by pressing the **Edit** button varies based on the selected type. Parametric properties are described in detail in Section 5.3.6, “Stream Types”. This property is not available for Basic streams.

File Groups

File groups are used to associate the File Sequence produced by two or more input sources together. When two parametric file sources with associated file sequences are connected to the input of a job step, the entries of their file sequences are combined one-to-one. This means that instead of the step including a task for the combination of each entry in both file sequences, the step will contain a task for the first entries in both sequences, then a task for the second entries, etc.. File sequences associated in this manner must have the same number of entries.

The editor for the **File Groups** property allows the user to associate different file sources together. The File Sequence Groupings dialog is populated with the list of all the file sets, and parametric model nodes in the job stream. A group number can be set for each file sequence via the drop-down editor in that row's **Group Number** column. The file sequences generated by the file sources with the same group number are associated together.

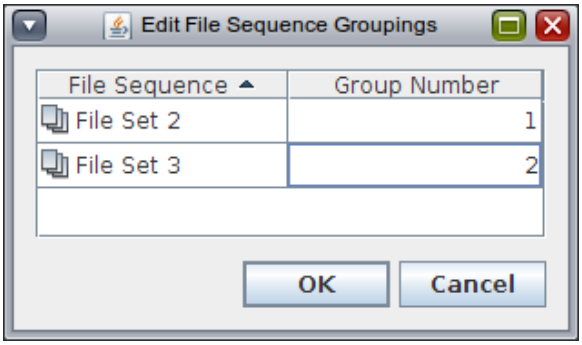


Figure 5.8. File Sequence Groupings

Platform

The platform (i.e. computer system) that the stream will be submitted to by default. This selection determines how the stream's platform related properties will be displayed. For example: the **Root Folder** property is only displayed when the "Local" (local Calculation Server) platform is selected.

This editor allows the user to select from the platforms that have been defined in the Configuration Tool as described in Chapter 3, *The Configuration Tool*. Changing the platform may require that the **Submission Properties** be redefined for each stream step.

Platform Properties

This property is used to customize the way the stream is submitted to the selected Platform. As these properties are very specific to the type of platform selected, each platform type provides its own platform properties dialog. For example: The "Torque/Maui" platform properties dialog includes the "Use local staging location mapping" option.

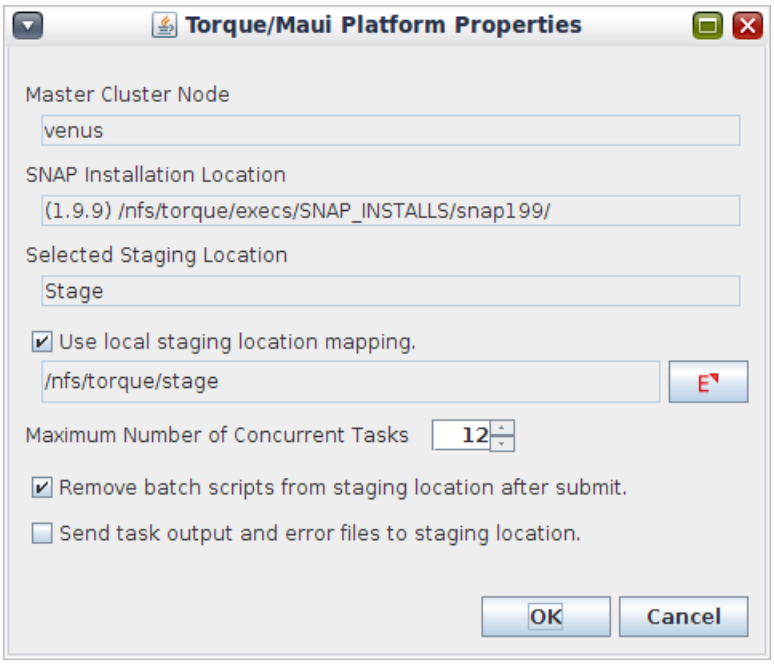


Figure 5.9. Torque/Maui Platform Properties

Note Some platform types (such as the Calculation Server) do not use Platform Properties.

Root Folder

This property determines where on the local Calculation Server the stream will be executed. The **E** (Edit) button to the right of the root folder selector can be used to add/remove folders for the local Calculation Server.

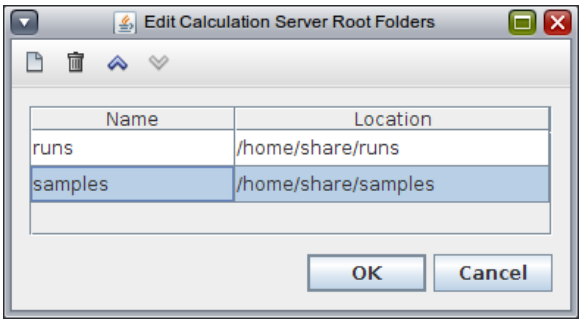


Figure 5.10. Calculation Server Root Folders

Note Only the "Local" platform (local Calculation Server) uses the Root Folder property.

SNAP Installation

The SNAP installation that will execute this stream on the selected platform. The list of SNAP installations is specified for the Platform in the Configuration Tool. This allows a single Platform to provide access to current versions of SNAP without removing access to older versions.

Note The Calculation Server does not support the SNAP Installation property.

Staging Location

The location on the selected platform where the stream's "intermediate" files are placed. Intermediate files are the files required either by the entire stream (i.e. the Stream Definition) or by down-stream tasks. For example: the output file produced by one task would be placed on the staging location at the end of its execution so that the next task in the stream would be able to use it as an input.

By default, any remaining intermediate files are removed at the end of stream execution.

Note The local Calculation Server uses the Root Folder property in place of the Staging Location.

Relative Location

The location of the stream within the selected Root Folder or Staging Location. This relative location can be used to organize the set of streams submitted to a particular location. If the relative location does not exist, it will be created during the submission process. The job stream will create a directory inside the relative folder where the stream steps will be executed.

Note The slash character '/' must be used as a folder separator for relative locations. The backslash character cannot be used.

Submission Properties

This property is used to customize the way the Stream Manager job (the central job stream process) is submitted to the selected Platform. These properties are specific to the type of selected platform; the available properties are identical for each step. Each platform type provides its own submission properties dialog.

For example: The "Torque/Maui" platform properties dialog includes the "Process Priority" and "Maximum CPU Time" options.

Log Level

This enumeration defines the level of details that will be included in the job stream log files. The selected value represents the minimum message level to display. For example, selecting **Information** defines that all messages flagged as Information, Warning, or Critical will be included, but not messages flagged as Debug.

View In Job Status

This flag indicates whether job status should be opened, and the job stream automatically selected when the job stream is submitted. If **Yes** is selected, Job Stream will be opened, and brought to the front of the screen. If an instance of Job Status is already available, that instance will be brought to the front of the screen. The submitted job stream will be selected either in the calculation server or the Tracking server.

Linear Execution

Job Streams using Linear Execution are executed one task at a time, prioritized by their Step Number. In a normal (non-linear) Job Stream tasks will be executed as soon as their dependencies are available (i.e. up-stream tasks are complete) up to the maximum number allowed by the platform.

This option should only be used in situations where the stream must execute one task at a time and each task must complete before another can be be executed.

In most cases this property should be left at its default value: *No*

5.3.2. Model Nodes

A model node represents a model currently loaded into the Model Editor. For most job streams, this will be a reference to the current model.

The outputs for each model node are the different files that may be exported by the model editor for referenced model. Model nodes may be set to parametric, which indicates that the model node will use the current stream type to generate a set of input files. The properties of a model node are shown in Figure 5.11, “Model Node Properties”.

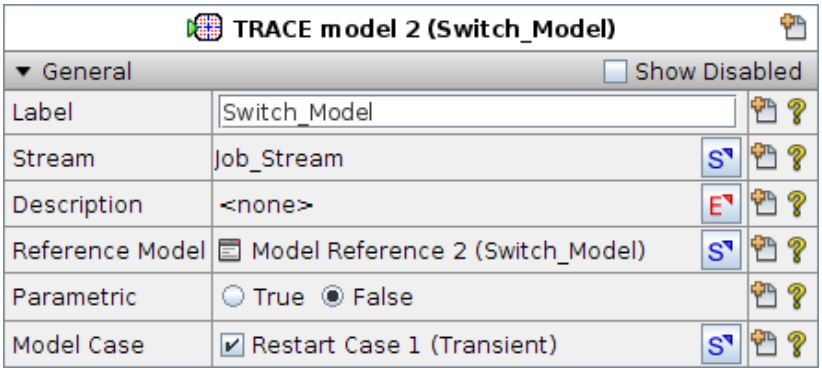


Figure 5.11. Model Node Properties

Model nodes have the following properties.

Label

The label by which the model is referred to in the stream. This need not be the name of the actual model. The model label may is used to name folders created during stream execution. It should be descriptive enough to identify the purpose of the model without being too long for a folder name.

Stream

The Job Stream to which the model node belongs. This property can be edited to move the model node between streams.

Description

This property allows an arbitrary description of the model node to be entered.

Reference Model

This property is used to indicate which Reference Model is represented by this model node. This property is only displayed for model nodes in an Engineering Template stream.

Parametric

Determines whether the model's outputs will be a set of parametric input files. This property is only enabled when the model node is part of a stream with a parametric Stream Type, such as Numeric Parametric. Toggling this property will change the appearance of the node in a 2D View to more easily distinguish between parametric and non-parametric models.

Model Case

When the model represented by the model node has one or more restart cases, this property can be enabled to select a restart that will further clarify the role of the node. By using restart cases, a single model can specify multiple restarts, allowing model node references to that model to use the restarts in a stream without relying on external files.

5.3.3. Stream Steps

Stream steps represent applications that will be executed by the job stream. Steps have inputs and outputs representing files that are used by or created by the application. Outputs may be connected to down-stream steps as inputs. The creation of stream steps is discussed in Section 5.2.2, “Creating Stream Steps”.

Job steps that have multiple input sources for an input file may be parametric. Parametric steps create a separate task for each combination of parametric inputs. A detailed discussion of parametrics is given below, in "Parametric Tasks".

Stream steps each include the properties shown in Figure 5.12, “Stream Step Properties”. Each of these properties is described in detail below.

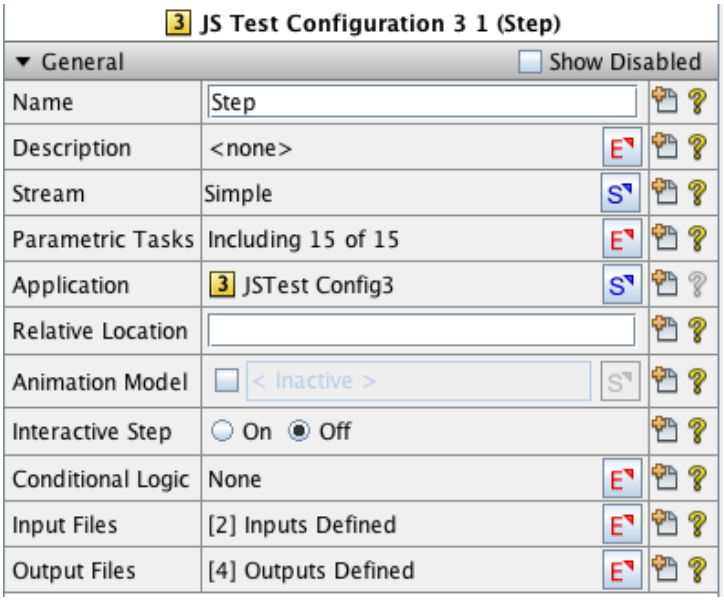


Figure 5.12. Stream Step Properties

Name

The files and folders created for stream steps are named using the step's name as the base. It should be descriptive enough to identify the purpose of the step without being too long for a folder name. Consequently, steps with the same relative location must have unique names (unique step names are advised regardless, if only for clarity).

Stream

This property indicates the job stream in which the step is contained. The step can be moved to a different stream by pressing the S (Select) button to the right and selecting a different stream. Note that changing the stream that a step is contained in does not affect other steps and/or files that it may be connected to.

Parametric Tasks

Parametric cases can be generated in two ways. The first method is to create multiple connections to a single input node. The second is by connecting a output-set of a stream component to a

non-set input of another stream component. In a stream step that satisfies one of these conditions, the **Parametric Tasks** property is enabled. Pressing the **E** (Edit) button at the right of the editor opens the Parametric Tasks dialog.

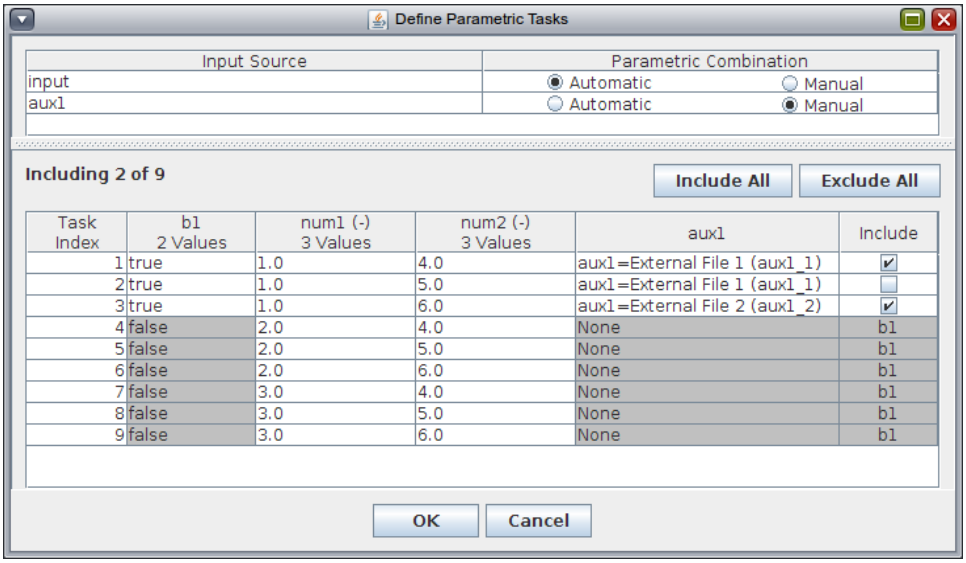


Figure 5.13. Parametric Tasks Dialog

The Parametric Tasks Dialog lists the parametric tasks that will be generated for a stream step. The top portion of the dialog (shown inFigure 5.13, “Parametric Tasks Dialog”) allows the user to specify which input sources will be combined manually. By default, all of the input sources will be combined automatically. If there is only one input source defined for this step, the top portion of the dialog will be hidden.

Each row in the task table (the table in the bottom portion of the dialog) represents a single task that will be generated. The columns represent a single variable name of the variables/files that make up the parametric set. Manually combined parametric inputs get a single column, regardless of how many variables make up that parametric input. The **Include** column allows the user to specify which parametric tasks will be included in the job submission.

If a disabling variable has been added to the stream type properties for this stream, the **Include** column for all affected rows in the task table will be uneditable; it will instead specify the name of the variable controlling its inclusion or exclusion. The task table can be sorted on any column by clicking on the a column header. A label included in the dialog indicates how many tasks are currently included, and buttons for including or excluding all tasks are also provided.

Application

This property determines which application will be executed by this step's tasks. The list of available applications is determined by those defined in the SNAP Configuration that have the same type as the step.

Relative Location

The location of this step in relation to the location of the entire stream. This relative location can be used to organize the set of steps within the stream. Relative Location paths must use the

slash character '/' as a folder separator. Note: Step names must be unique within a given relative location.

Animation Model

This property allows selecting the location of an Animation MED file used to animate the step. Editing the value opens a file browser used to select the MED.

Open Animation

This property defines how the step's **Animation Model** is used during stream execution. When set to **Immediately**, the selected **Animation Model** will be opened and connected to the task as soon as it begins to execute.

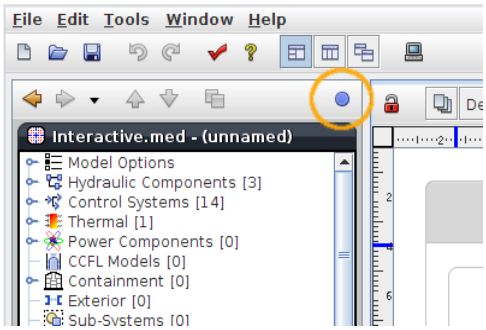


Figure 5.14. Available Jobs Indicator

When set to **Prompt**, the user will be notified when the task is available for animation. As soon as any jobs are available, the **Available Jobs** indicator will appear on the right side of the Navigator toolbar. To animate one of the waiting jobs, click on the indicator and a list of the available jobs will be displayed. Selecting any job from the list will immediately open the selected **Animation Model** and connect to the job.

Interactive Step

When this property is **On**, this step's tasks will connect to the Calculation Server the task was submitted to. This connection will allow clients to communicate with the task to send interactive commands and retrieve the current calculation time, multiplexed plot data, console output, etc..

Note The Interactive Step option is only available when submitting to a Calculation Server.

Start Paused

This property instructs this step's tasks to pause immediately after starting. This property is only available for interactive steps (see above) when submitting to a Calculation Server.

Conditional Logic

The user-defined numeric conditions that determine whether this step should be included in the job stream. These conditions are evaluated during submission and can be used to exclude specific

tasks of a parametric step. Editing the property displays the dialog shown in Figure 5.15, “Define Variable Range Dialog”.

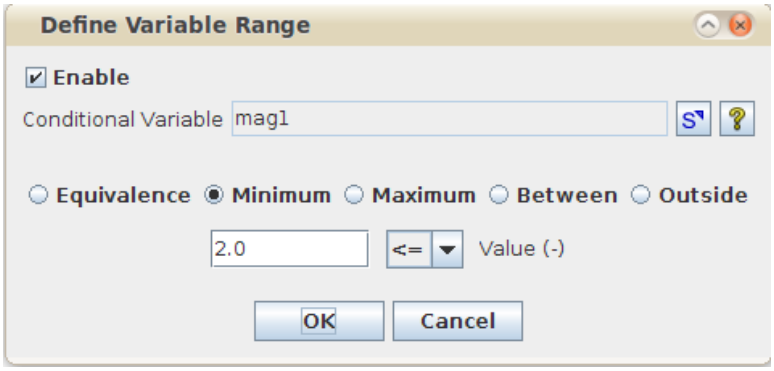


Figure 5.15. Define Variable Range Dialog

The **Enable** check-box controls whether conditional logic is applied to the step. When enabled, the **Conditional Variable** field can be used to select a user-defined numeric upon which the logic will be based. Finally, once a numeric reference has been defined, the variable range upon which the conditional is defined can be specified. The valid set may be defined as: a single value, all the values above or below a set-point, all the values between two set points, or all the values outside of two set points. Each set-point can be either inclusive (<=) or exclusive (<).

Input Files

This property represents the "inputs" to a step which correspond to the input files that the application expects when it is executed. The source of each input can be selected using the "Define Input Files" dialog available by pressing the **E** (Edit) button to the right of the property. This dialog lists all of the inputs to the step in a table as shown below.

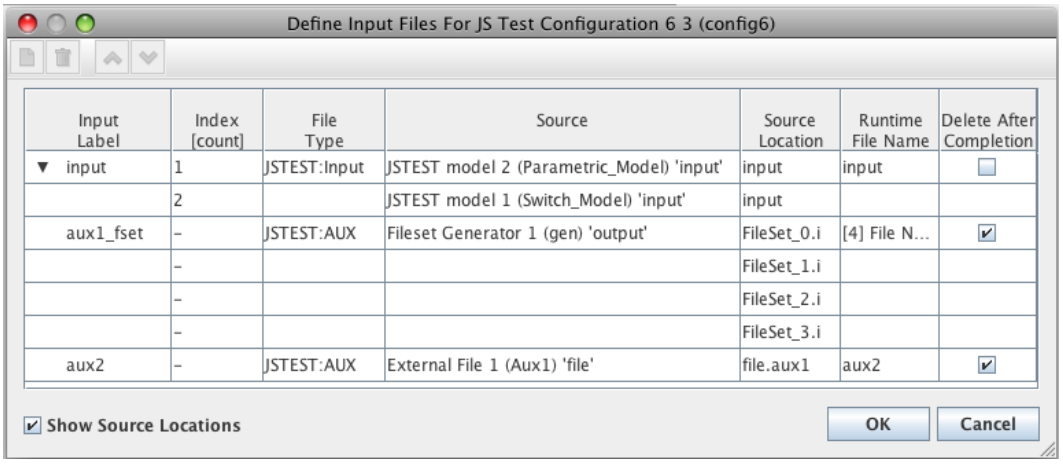


Figure 5.16. Input Files Dialog

Each input is represented by one or more rows in the table with the following columns.

- **Input Label**

The short, human-readable "label" for the input. This is not a file name used at runtime, but rather a short description displayed in 2D views and lists to uniquely identify this input.

When multiple input connections exist for a given input, this column can expand the table to show all the sources.

- **Index [count]**

  If this step is parametric, the **Index** column displays the total number of parametric sources for each parametric input. If the label is expanded, then this column contains the index of the input source.

- **File Type**

  The type of file that this input expects. File types are typically listed as `[Type Source]:[Type Name]`. For example: the TRACE ASCII output file (trcout) is of type `TRACE:Output`.

- **Source**

  The **Source** column allows the user to define the file source for the input. The editor for the source column allows the user to select from any of the inputs in the job stream that have a type matching that input's **File Type**.

- **Source Location** (optional)

  The **Source Locations** column displays the absolute location of the source file specified in the **Source** column. This column is only displayed if the **Show Source Locations** check-box is checked at the bottom of the dialog.

- **Runtime File Name**

  The name(s) that will be used for this input when this step's tasks are executed.

- **Delete After Completion** (Calculation Server only)

  The check-boxes in this column indicate which inputs will be explicitly deleted after the application executes. This column is only available when submitting to a Calculation Server.

- **Save to Staging Location** (all but Calculation Server)

  The check-boxes in this column indicate which inputs will be saved to the selected Staging Location after the application executes. This column is not available when submitting to a Calculation Server.

Output Files

This property represents the "outputs" of a step which correspond to the output files that the application creates when it is executed. The **Output Files** property of a stream step is edited via the Output Files Dialog.

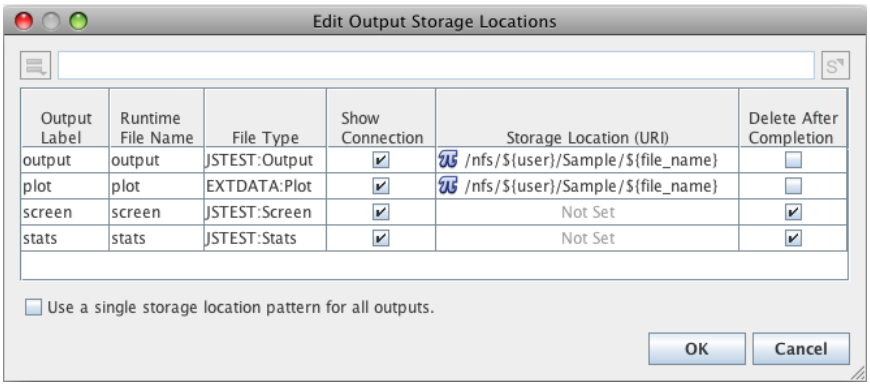


Figure 5.17. Output Files Dialog

The Output Files Dialog lists the output files that will be produced by the application. Each output is represented by a single row in the table with the following columns.

- **Output Label**

  The short, human-readable "label" for the output. This is not a file name used at runtime but rather a short description displayed in 2D views and lists to uniquely identify this output.

- **Runtime File Name**

  The name(s) that will be used for this output when this step's tasks are executed.

- **File Type**

  The type of file that this output creates. File types are typically listed as `[Type Source]:[Type Name]`. For example: the TRACE ASCII output file (trcout) is of type `TRACE:Output`.

- **Show Connection**

  The check-boxes in this column indicate which outputs will be displayed as connectors for this step in 2D Views.

- **Storage Location** (URI)

  This column defines where the output file(s) will be stored after the task is completed. This field is optional and if left blank, the output file will not be stored. The storage location of the currently selected output can be edited in the text field at the top of the dialog.

  Storage locations can also contain patterns that can create file names based on system-specific or model-specific tokens. To insert a storage location token into a storage location, position the cursor in the desired insertion position in the storage location text field and click the **Insert Storage Location Token** button to the right of the text field. The list of available tokens and a description of each is given below.

  Note The storage location must contain a well-formed URI (Uniform Resource Identifier).

- **Delete After Completion** (Calculation Server Only)

The check-boxes in this column indicate which outputs will be explicitly deleted after the application executes. This column is only available when submitting to a Calculation Server.

- **Save to Staging Location** (all but Calculation Server)

  The check-boxes in this column indicate which inputs will be saved to the selected Staging Location after the application executes. This column is not available when submitting to a Calculation Server.

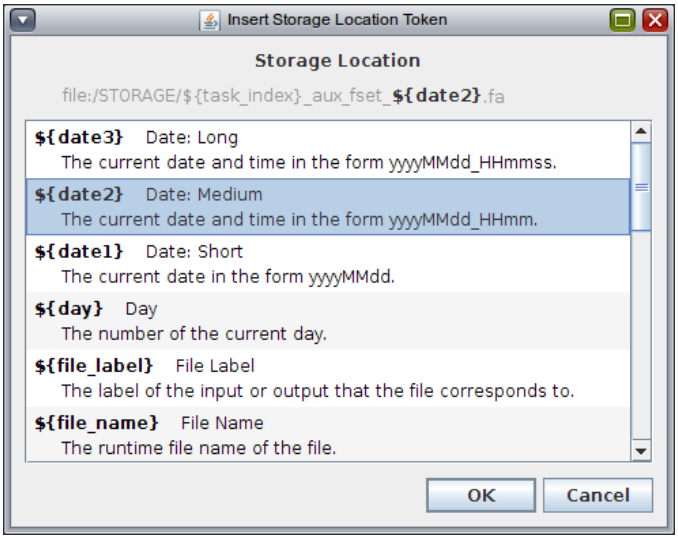


Figure 5.18. Storage Location Tokens Dialog

Figure 5.18, "Storage Location Tokens Dialog" shows the Storage Location Tokens Dialog. This dialog contains the list of available storage tokens, as well as a preview of the storage location with the currently selected token added in the specified position. The complete list of available storage location tokens is as follows:

- **`${date1}`** Date Short- The current date and time in the form yyyyMMdd.
- **`${date2}`** Date Medium- The current date and time in the form yyyyMMdd_HHmm.
- **`${date3}`** Date Long- The current date and time in the form yyyyMMdd_HHmmss.
- **`${day}`** Day- The number of the current day.
- **`${file_label}`** File Label- The label of the input or output that the file corresponds to.
- **`${file_name}`** File Name- The runtime file name of the file.
- **`${file_index}`** File Set Index- The index of the file within its file set.
- **`${kw_n: <keyword>}`** Keyword Name- The name of a keyword, if it is present.
- **`${kw_v: <keyword>}`** Keyword Value- The value of a keyword, if it is present.
- **`${month}`** Month- The number of the current month.

- **`${pkw_n: <keyword>}`** Parametric Keyword Name- The name of a parametric keyword if it is present.
- **`${pkw_v: <keyword>}`** Parametric Keyword Value- The value of a parametric keyword if it is present.
- **`${step}`** Step Name- The user-specified name of the Job Step.
- **`${steploc}`** Step Name- The user-specified relative location of the Job Step.
- **`${stream}`** Stream Name- The user-specified name of the Job Stream.
- **`${streamloc}`** Stream Location- The user-specified relative location of the Job Stream.
- **`${task_ident}`** Task Ident- The ID number of the task in the form XXYYYYY where X is the job step number and Y is the task index.
- **`${task_index}`** Task Index- The index of the task.
- **`${task}`** Task Name- The name of the task as generated from the step name and task index.
- **`${user}`** User ID- The user ID of the current user.
- **`${year}`** Year- The number of the current year.

Note To use a storage location token that includes the characters "<keyword>", replace "<keyword>" with the keyword, no additional quotations or brackets are necessary (Sample usage: ${pkw_n: r1}).

5.3.4. Files and File Sets

Two types of file components are supported in Job Streams: External Files and File Sets. Upon creation of a new file, the File Completion dialog is shown and allows the user to choose to create a single file or a set of files.

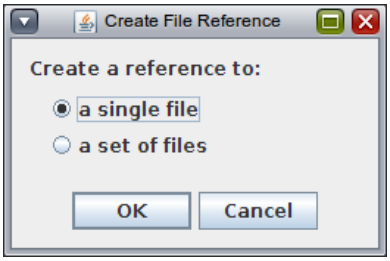


Figure 5.19. File Completion Dialog

5.3.4.1. External Files

An External File represents a single file that either is locally accessible to the Model Editor, or can be selected by one of the available file selection implementations.

External File Properties

The general properties for an external file are shown in Figure 5.20, “External File Properties”.

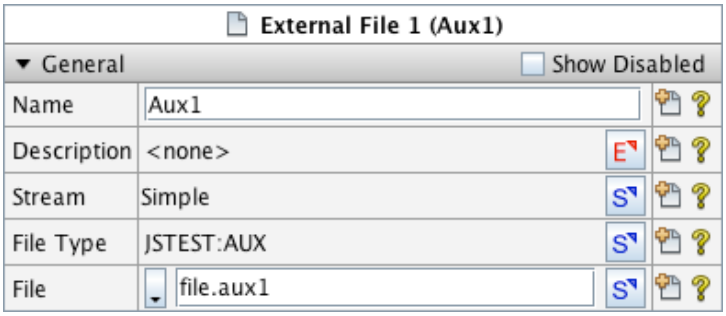


Figure 5.20. External File Properties

Name

The name by which the file component is referred to within a stream. Note that this is the name of the External File component, not the name of the file it refers to. This allows the External File **Name** to indicate the function of the file rather than the on-disk name. For example: the External File can be named "Null_Transient_Input" rather than "w4loopnt.inp."

Description

This property allows for an arbitrary description of the external file.

Stream

The parent stream to which the external file belongs. The external file can be moved between streams by editing this property.

File Type

The **File Type** property specifies the type of file(s) represented by this File component. The File Type Selection Dialog displays the list of available file types. Each available type is described by a source, name, and short description.

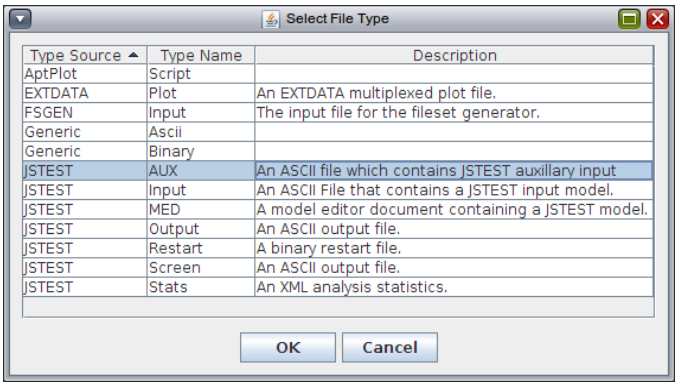


Figure 5.21. File Type Selection Dialog

File

The **File** property represents the physical location of the selected file displayed as a URI. The drop-down button on the left side of the editor is used to select the protocol that handles locating the file. The **Local File** protocol is always provided; additional protocols may be available. Pressing the editor **S** (Select) button opens a dialog used to select the file based on the specified

protocol. For Local File, this opens a standard file browser. Other protocols provide similar selectors tailored to their specific requirements.

5.3.4.2. File Sets

A file set is used to define a set of multiple files that are either locally accessible to the Model Editor, or can be selected using an available file protocol.

File Set Properties

The general properties for file sets are shown in Figure 5.22, "File Set Properties". File Sets have the same properties as External Files, with the exception of the **Files** property (see Section 5.3.4.1, "External Files" for the definition of other File Set attributes).

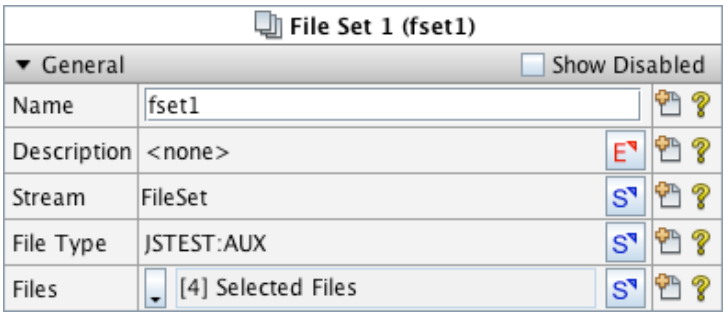


Figure 5.22. File Set Properties

Files

The **Files** property of a file set defines the physical locations of the files selected for this file set. Similar to the **File** property of an External File, files in a File Set use a protocol to specify how all files in the set are located. The protocol can be selected with the drop-down button on the left side of the editor. Pressing the **Select** button opens the Select Files dialog, shown in Figure 5.23, "File Set Selection Dialog". Individual files can be added, removed, and reordered via this dialog.

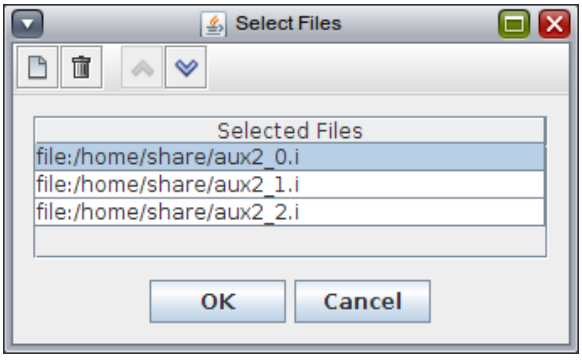


Figure 5.23. File Set Selection Dialog

Adding a file to the set opens the file selection UI specific to the selected protocol. For the Local File protocol, this is a standard file browser.

5.3.5. Input Switches

In essence, an Input Switch takes multiple inputs and selects only one of these to pass through as a single output. It makes this selection based on the value of a user-defined numeric (or "shared

value"). To do this, a switch has a set of Input Branches that correspond to input connections. Each of these input branches has conditional logic based on the selected user-defined numeric. When the stream is submitted, each of the Input Branches is evaluated in order to determine which input source will be used as the output of the switch.

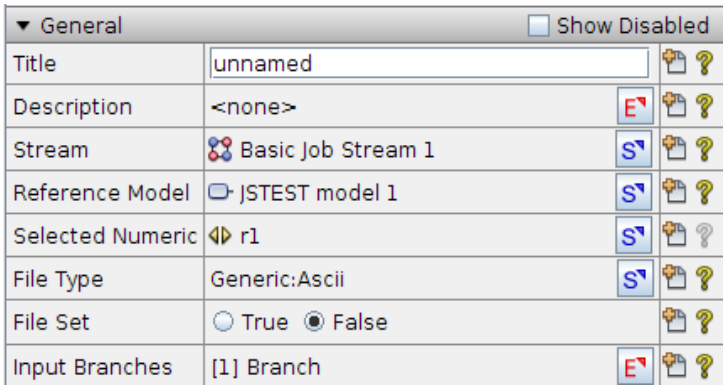


Figure 5.24. Input Switch Properties

Figure 5.24, "Input Switch Properties" shows the properties for an input switch. The **Selected Numeric** property determines the user-defined numeric whose value will be used when evaluating the **Input Branches**. The selected numeric can be a shared Real, Integer, or Boolean.

Note The **Selected Numeric** and **Input Branches** properties are disabled while the value of the **Reference Model** property is **<none>**. Also, the **Input Branches** property is not editable while the value of the **Selected Numeric** property is **Unspecified**.

Input Branches

Input Branches represent the different input sources that may be used for the output of this switch. The Numeric Branches Dialog allows the user to specify the branches for a particular switch. Each branch corresponds to a distinct value of the numeric specified in the switch's **Selected Numeric** property. Branches are evaluated in the order they appear in this dialog, and the first branch whose conditions evaluate to true will be executed. The order of the branches can be modified via the **Up** and **Down** buttons located at the top of the dialog. The conditions column of each branch is edited via the Branch Conditions Dialog. For a detailed description of the Branch Conditions Dialog, see the section called "Branch Conditions".

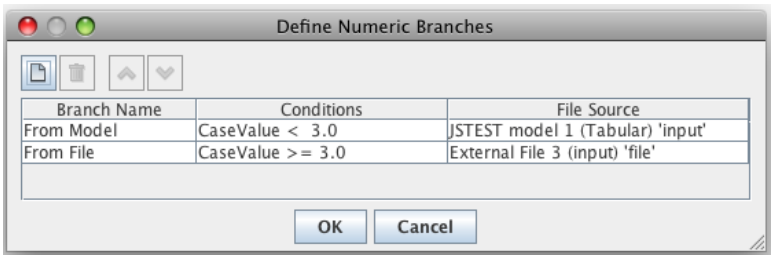


Figure 5.25. Input Switch Numeric Branches Dialog

Branch Conditions

Each branch of an input switch must have numeric conditions which specify when a particular input branch is active. For Real and Integer numerics, these conditions are defined as a set of

valid numeric values. The valid set may be defined as: a single value, all the values above or below a set-point, all the values between two set points, or all the values outside of two set points. Each set-point can be either inclusive (<=) or exclusive (<).

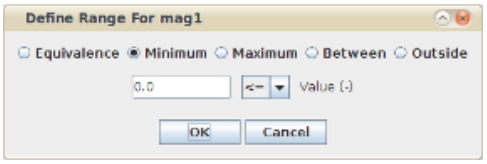


Figure 5.26. Numeric Branch Condition Dialog

5.3.6. Stream Types

Job Streams that are set to have a type other than **Basic** can be used to define parametric cases. The parametric cases depend on the type of stream selected. Once the parametric properties are defined, the model nodes inside the job stream may be set to parametric. The two types of parametric stream types available are Numeric Combination, and Tabular Parametric.

Parametric tasks produced by a parametric stream type are defined in terms of parametric keywords. Each task has a unique combination of name=value keywords that describe the contents of the tasks. For example, within a Numeric Combination stream, the keywords listed for each task will be named after the Independent or Dependent Variables, and the value of each keyword will be the corresponding value of that variable for the task.

5.3.6.1. Numeric Combination

A Numeric Combination Job Stream builds a set of input models by modifying the value of one or more shared numeric values. Each selected numeric is modified by either a set of explicit values or a range of values defined by a start, end, and increment. The complete combination of independent variable values, coupled with the dependent values evaluated at those combinations, defines the set of parametric tasks.

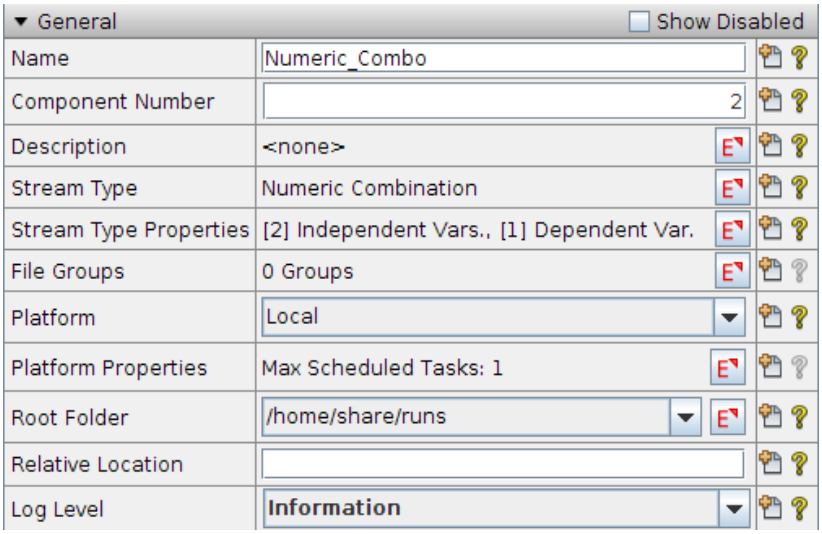


Figure 5.27. Numeric Combination Stream Properties

Editing the Stream Type Properties opens the **Select Numeric Variables** dialog, shown in Figure 5.28, "Numeric Variables Dialog - Independent Variables Tab". This dialog allows the user to specify the independent and dependent variables that vary over the parametric tasks, and to filter specific values from the results.

Independent Variables

The list of **Independent Variables** will be explicitly modified for each case. This will produce one case for each combination of independent variable values. For example, if independent variable **num1** is an explicit list of four values, and independent variable **num2** is a range that produces three values, the resulting stream will have 12 parametric tasks as a base.

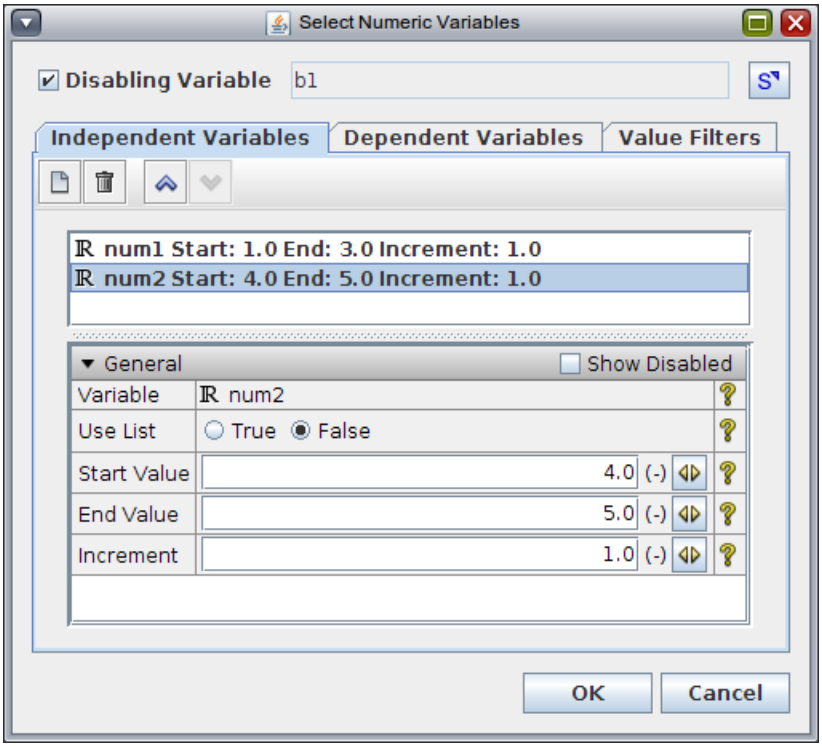


Figure 5.28. Numeric Variables Dialog - Independent Variables Tab

Within the **Independent Variables** tab, single-value numerics can be added to the list, removed, or re-ordered. Selecting a value in the list will display properties related to the parametric cases defined for the numeric. The **Use List** property controls whether the parametric cases are defined as an explicit list of values (**True**) or as a range of values (**False**). When **Use List** is **True**, the **Value List** property is used to define the explicit values. Otherwise, the range is described by the **Start Value**, **End Value**, and **Increment** properties.

Each value described by an Independent Variable will be used to create parametric keywords for the stream. These keywords take the form of `name=value` pairs. In the above example, one such keyword for a task would be `num2=4.0`.

Dependent Variables

Dependent Variables provide a means of adding additional parametric keywords to a Numeric Combination.

Each Dependent Variable is specified as a reference to a shared numeric value. User-defined functions are evaluated after the independent variable values have been set for each case. The user-defined numeric referenced by the Dependent Variable is then stored as a parametric keyword value for that case.

Dependent variables cannot be used to add parametric cases, they exist solely to provide streams with additional meta-data.

Value Filters

Value Filters are used to exclude specific parametric cases from the list of cases created by the numeric combination. The filter is specified as a numeric range, and can be configured to define a specific numeric value, a minimum value, a maximum value, the set of values between two numbers, or the set of values outside of two numbers. The value filters table contains the values for the selected numeric. For each value, a check-box allows each individual case to be excluded. Any filtered value will have the background of its table row shaded grey. Filtered values can be hidden from the table by selecting the **Hide Filtered Values** check-box.

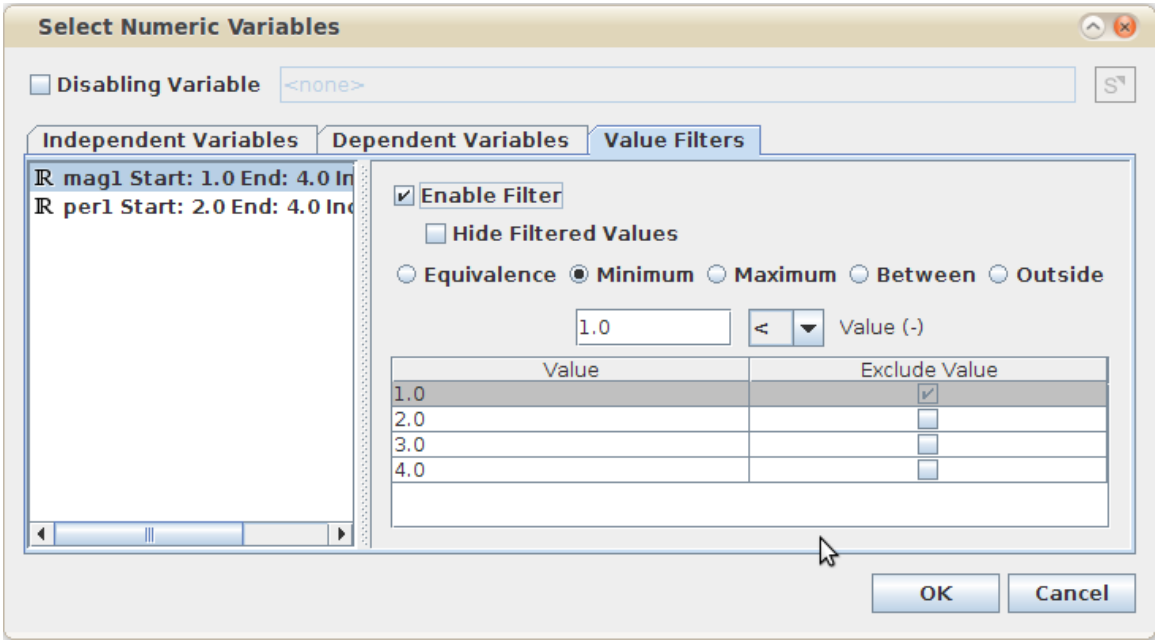


Figure 5.29. Numeric Variables Dialog- Value Filters Tab

5.3.6.2. Tabular Parametric

A Tabular Parametric stream is a specialized Numeric Combination Job Stream that defines parametric tasks using a table of shared variable values.

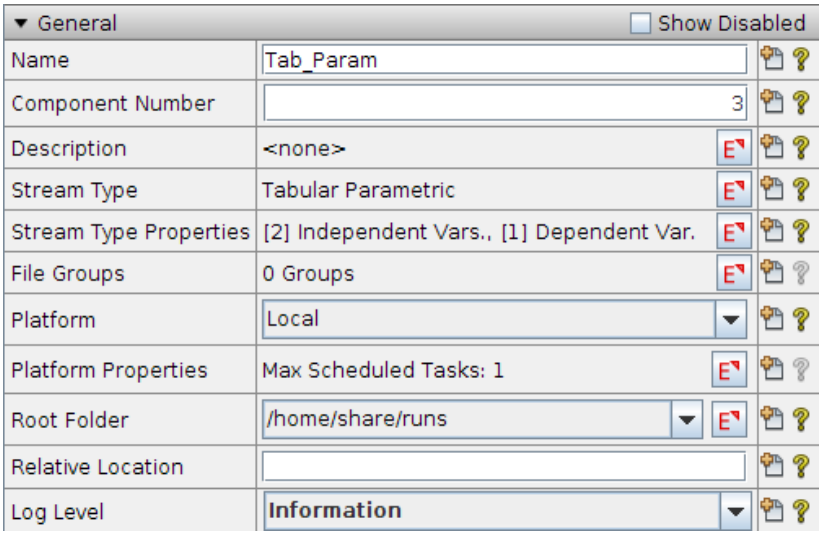


Figure 5.30. Tabular Parametric Stream Properties

Stream Type Properties

Independent variables can be added, removed, and re-ordered in the Table Variables tab. Dependent variables are additional user numerics that are included in the parametric keywords.

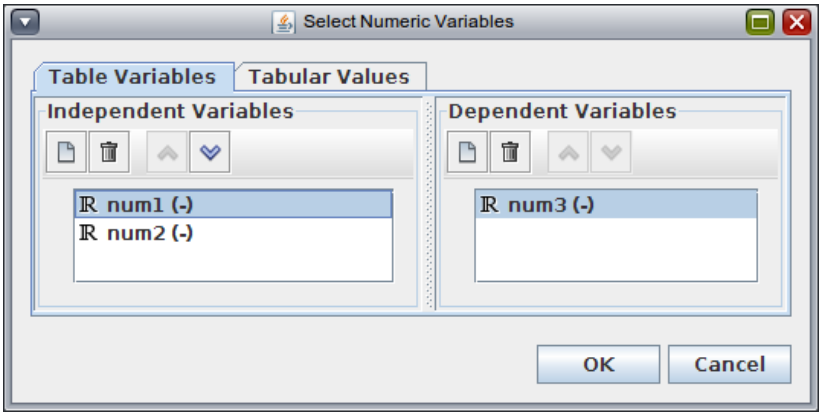


Figure 5.31. Numeric Variables Dialog- Table Variables Tab

For tabular parametric streams, parametric cases are defined via the Tabular Values tab of the Numeric Variables Dialog. The structure of the tabular values table is defined by the independent variables specified in the Table Variables tab of the dialog. The table contains a column for task index and a column for specifying the value of each independent value. Parametric cases can be added, removed, and re-ordered.

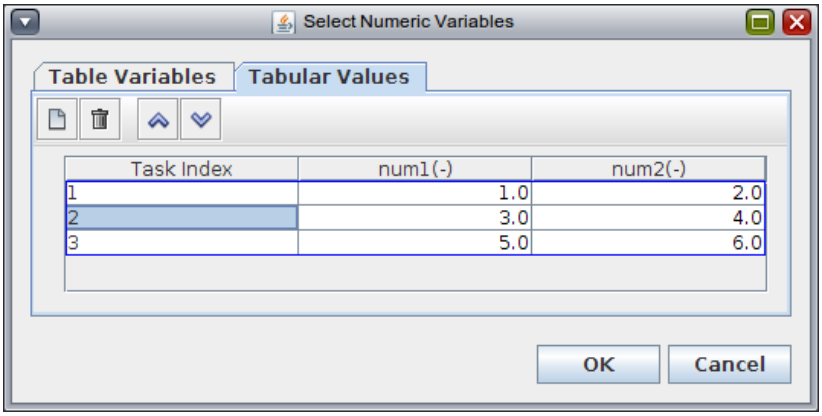


Figure 5.32. Numeric Variables Dialog- Tabular Values Tab

5.4. Submitting Job Streams

Submitting a job stream is the process of sending it to a Platform (such as a Calculation Server) so that it can be executed.

Job streams can be submitted for execution in the following ways:

- Select "**Submit Job...**" from the **Tools** menu. This opens the dialog shown in Figure 5.33, "Submit Job Stream Dialog". This dialog allows the user to select both the stream to submit and the platform to which it will be submitted. Pressing the **OK** button submits the stream.
- Add the stream to a 2D View by either selecting **Add To View** from its right-click pop-up menu, or by dragging the stream into the 2D View. This will add a button to the View used to execute the stream. This button can only be used when its parent 2D View is locked. The stream is submitted to the **Platform** selected in the stream's properties.
- Right-click on the stream in the Navigator, and select **Submit Stream to <Platform>**, where the stream's selected **Platform** is displayed.

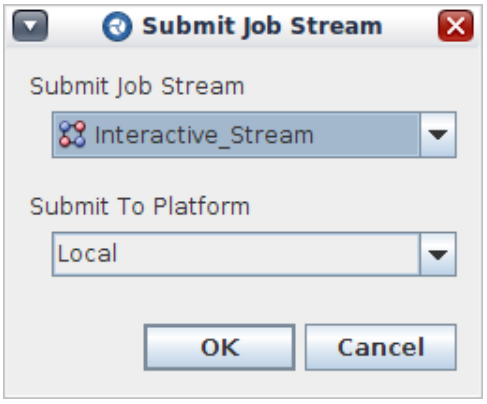


Figure 5.33. Submit Job Stream Dialog

Regardless of which method is used the Model Editor will export and/or bundle up all of the necessary files and send a complete description of the stream to the selected platform. If the stream's **View in Job Status** property is set to **Yes**, Job Status will open and highlight the submitted stream.

5.5. Diagnosing Submitted Job Streams

SNAP Applications write messages to screen and log files that can help diagnose issues. Job streams and their tasks also write to log files. The log for the entire job stream (the *Stream Log*) contains messages indicating where the stream executed, and what staging location it used. This log also indicates which tasks were submitted, started, and finished (or failed). The log files for each task contain information more specific to the task being executed. This includes messages for each input file retrieved and each output file stored as well as any errors that might have occurred. The stream and task log files are available in *Job Status* via either the File Viewer or Job Consoles.

If the stream is not available on a Calculation Server then the log files must be examined directly rather than using Job Status. The log file for the entire stream is written to the stream's Staging Location during execution and is named [Stream Name].streamlog. The individual log files for the tasks in the stream are written to each task's folder under the stream's staging location with the name [Task Name].tasklog.

When submitting to the Calculation Server it may sometimes be necessary to examine its log file to determine how or why a stream failed. The Calculation Server log files are written to the current user's SNAP preferences folder. A new log file is created each time the server is started. Up to 9 previous log files are retained.

```
<home>\.snap\2.0\log\calculation_server.log
```

The SNAP application screen files are also stored in the user's SNAP preferences folder. These files may contain additional information that can be used to diagnose an application failure. They are not typically necessary.

- `Calculation Server`

  ```
  <home>\.snap\2.0\cs.screen
  ```

- `Job Status`

  ```
  <home>\.snap\2.0\js.screen
  ```

- `ModelEditor`

  ```
  <home>\.snap\2.0\me.screen
  ```

The current user's home directory (<home> above) can be found in:

- Windows 7 or Vista

  ```
  C:\Users\<username>\.snap\2.0\
  ```

- Windows XP

  ```
  C:\Documents and Settings\<username>\.snap\2.0\
  ```

- UNIX / Linux

  ```
  $HOME/.snap/2.0/
  ```

5.6. AptPlot Step

A specialized AptPlot step is included with the Model Editor. A single AptPlot step can specify any number of plots, all of which are generated every time its parent stream is run.

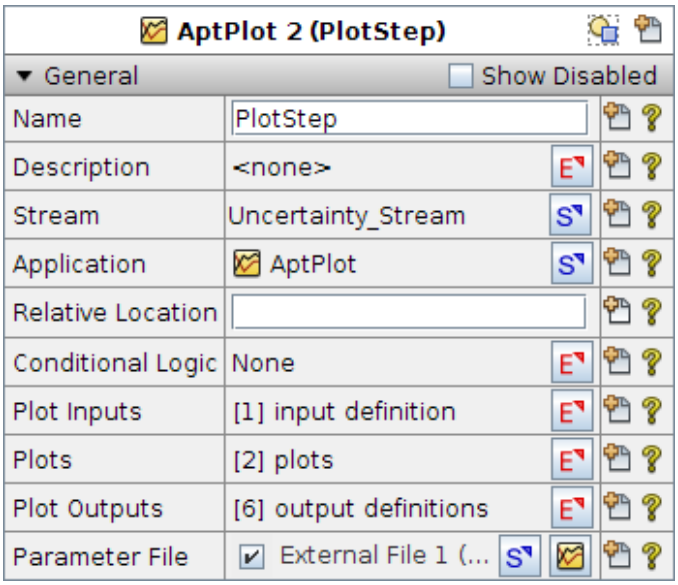


Figure 5.34. AptPlot Step Properties

An AptPlot step has the standard properties found in a step, with the following additional attributes.

- **Plot Inputs** - the inputs to the step, see Section 5.6.2, “AptPlot Step Inputs”.
- **Plots** - the plots created by the step, see Section 5.6.3, “AptPlot Step Plots”.
- **Plot Outputs** - the images and data output by the step, see Section 5.6.4, “AptPlot Step Outputs”.
- **Parameter File** - an optional parameter file (see Section 5.6.1, “Parameter Files”). This is the *global* parameter file: it is used by all plots unless they explicitly specify a parameter file of their own.

5.6.1. Parameter Files

A parameter file is a series of AptPlot batch commands that formats a plot. A Parameter File property is used to specify the optional parameter file for a step or plot definition. When such a **Parameter File** property is enabled, an optional input is added to the step.

The **Parameter File** property editor has a **Select** button and an **AptPlot Preview** button. The **Select** button can be used to make connections to the parameter file input. Pressing it opens the pop-up menu shown in Figure 5.35, “Parameter File Select Menu”. From the menu, an existing External File can be selected for the selection, or a new External File can be selected. The **AptPlot Preview** button is enabled when the param input is connected to an External File available on the local machine. Pressing the **Preview** button will open AptPlot with the indicated parameter file.

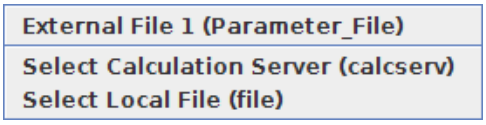


Figure 5.35. Parameter File Select Menu

5.6.2. AptPlot Step Inputs

As AptPlot supports a host of input formats which can be opened in arbitrary numbers, an AptPlot step must define the inputs which will be used to generate its plots. The step **Plot Inputs** property is used to specify these inputs. Pressing the **Edit** button in the **Plot Inputs** property editor opens the **Edit Plot Inputs** dialog, shown in Figure 5.36, "AptPlot Step Edit Inputs Dialog". Note that this dialog is *non-modal*; other dialogs and Model Editor functions can be used while this dialog is open. Changes made in this dialog can be undone and redone with the standard undo/redo buttons and menu items.

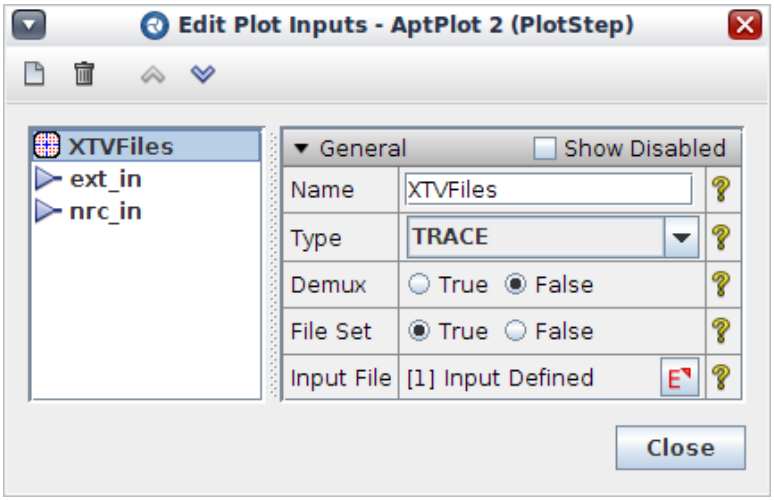


Figure 5.36. AptPlot Step Edit Inputs Dialog

On the left side of the dialog is a list of inputs available to the AptPlot step. On the right side of the dialog are the properties for the selected input. The toolbar at the top can be used to add, remove, and reorder inputs. Each input added in this dialog adds another input to the step, visible on its left side when displayed in a 2D View.

Inputs have the following properties.

- **Name** - The name of the input. Names must be unique among inputs in a particular AptPlot step. This name specifies the label for the corresponding input on the step.

- **Type** - Indicates the type of file. In the figure, a single TRACE input has been specified, which can be connected to any TRACE:XTV output (be it from a TRACE step output, an external file, etc.). Most supported types correspond to one of the analysis code files supported by AptPlot. However, the following additional types are supported.

  - **ASCII** - generic, space-delimited columns ASCII data as supported by AptPlot. The file type of ASCII inputs is **`AptPlot:ASCII Data`**.

- **Variables ASCII** - AptPlot Equation Interpreter saved variables in ASCII format. The file type of Variables ASCII inputs is **`AptPlot:Variables`**.
- **Variables PIB** - AptPlot Equation Interpreter saved variables in PIB format. The file type of Variable PIB inputs is **`AptPlot:PIB-Variables`**.

- **File Set** - Indicates that this input is a file set input, allowing parametric steps to *fan-in* to the input. When a parametric step or external file set is connected to a file set input, any data sets that reference the input will create one set per parametric task (see Section 5.6.3, "AptPlot Step Plots").
- **Input File** - Indicates the external file or step output connected to the AptPlot step input. The **Edit** button opens the **Define Input Files** dialog used to manually edit the input (see the section called "Input Files").

5.6.3. AptPlot Step Plots

The **Edit** button for the step **Plots** property opens the **Edit Plot Properties** dialog, shown in Figure 5.37, "AptPlot Step Edit Plot Properties Dialog". This editing dialog is the heart of the AptPlot step. Within it, the plots, graphs, data sets, and annotations created by the step are added, edited, and removed. Note that this dialog is *non-modal*; other dialogs and Model Editor functions can be used while this dialog is open. Changes made in this dialog can be undone and redone with the standard undo/redo buttons and menu items.

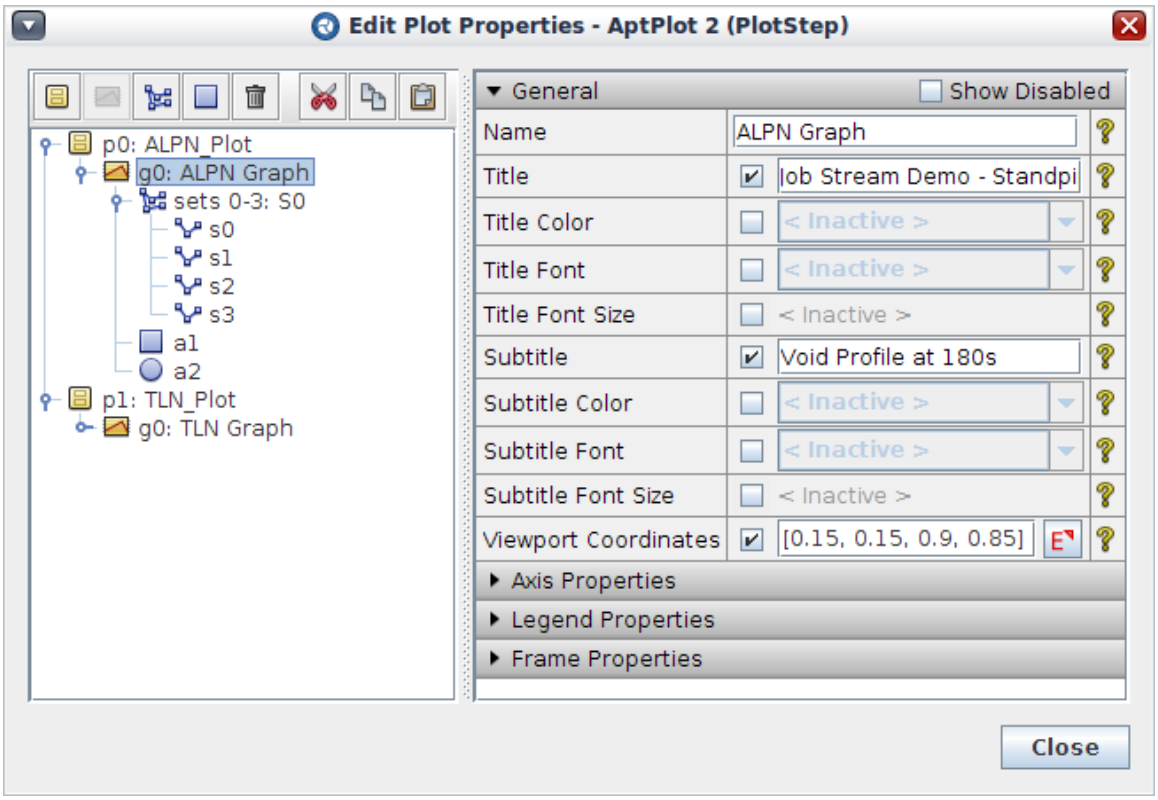


Figure 5.37. AptPlot Step Edit Plot Properties Dialog

On the left side of the dialog is a tree of AptPlot definitions. The contents of the tree are as follows.

1. At the top level of a tree is the list of *plots* in the step. Each plot is a unique "canvas" upon which graphs are drawn.

2. Each plot may contain any number of *graphs*. A graph is a region of the plot that displays any data set values within its coordinate space.

3. Each graph may contain any number of *data sets* and *annotations*. Data sets are the coordinates displayed as points or lines in the graph. Annotations are rectangles, ellipses, lines, and strings of text used to mark the plot.

4. Data sets connected to a file set input may contain a generated list of *sub-sets*, described below.

The toolbar over the can be used to add or remove plots, graphs, data sets, and annotations. Graphs can only be added when the parent plot is selected; data sets and annotations can only be added when the parent graph is selected. Additionally, cut, copy, and paste actions are available on the right side of the toolbar. All of these actions are also available from the right-click pop-up menu of the various entries in the tree.

Selecting a definition in the tree displays its contents in the Property View to the right. The definition is edited in this Property View. The **Edit Plot Properties** dialog supports multi-edit: selecting several components will display their shared properties in the Property View.

Most properties in the definitions are *namelist properties*: their values are optionally defined. Any given value is only specified in the generated plots if its value is enabled and defined. This allows the AptPlot step to use parameter files to define the plot, while individual parameters in the plot definitions *overrides* the parameter file formatting.

5.6.3.1. Plots

Plot definitions may specify a parameter file, as described in Section 5.6.1, “Parameter Files”.

5.6.3.2. Data Sets and Sub-Sets

Each data set definition in a graph describes a set of independent and dependent data extracted from an input source. Data set definitions have the following properties related to specifying the displayed data.

- **Input** - defines a reference to the input from which this data set definition reads data.

- **Plot Type** - defines the type of data read into the set. The following types are supported.
  - **Time** - a set of values is extracted from the input, with time (the independent values in the data) automatically retrieved based on the specified dependent data.
  - **Time Point** - same as the **Time** type, but interpolated to a single point at a specified time.
  - **Parametric** - two sets of values are extracted from the input, specifying both the independent and dependent values.
  - **Parametric Point** - same as the Parametric type, but interpolated to a single point at a specified time.
  - **Axial** - allows for the creation of axial profile plots at a specified time. Axial plots are described in greater detail below.

- **Independent Data** - the source independent data. This channel has different semantics depending on the selected input and the value of **Independent Type**.

- For plot file inputs, such as a TRACE XTV file, this denotes a data channel name. The syntax of data channel names varies by type (consult the AptPlot documentation for more info).
- For ASCII inputs, this denotes index of the data column to use.
- For Variable inputs (both ASCII and PIB), this denotes the name of a vector variable in the file.

Independent Data can only be specified for **Parametric** and **Parametric Point** data sets.

- **Independent Type** - determines whether **Independent Data** specifies a single channel or a complex expression. When set to **Expression**, **Independent Data** is interpreted as an AptPlot Equation Interpreter expression, with one distinction. Data channel names must be surrounded by curly-braces, i.e. **`{CHANNEL_NAME}`**. For example, the TRACE channel **`pn-400A01`** must be written in the expression as **`{pn-400A01}`**. A simplistic sample expression:

  ```
  {pn-400A01} - {pn-400A02}
  ```

  To use the braces '{' and '}' within an expression, escape them with a backslash '\', so that '{' becomes '\{' and '}' becomes '\}'. To use a backslash, escape it as '\\'.

  Independent Data can only be specified for **Parametric** and **Parametric Point** data sets.

- **Dependent Data** - similar to **Independent Data** for the dependent portion of the data set.

  Dependent Data can be used for axial profile plots. Axial **Dependent Data** values use the construct %2N, %3N, etc. to indicate the portion of the channel name substituted with axial mesh indexes. The integer in the construct determines how many padding zeros are applied for indexes less than the ceiling index.

- **Dependent Type** - similar to **Indpendent Type** for the dependent portion of the data set.

- **Time** - specifies the time at which **Time Point**, **Parametric Point**, and **Axial** type data sets are based.

- **Axial Locations** - defines the axial or z-direction locations for a series of mesh locations in a single component along the direction of flow for the data channel being plotted. Axial plots are described in greater detail below.

Sub-Sets

When a data set references a file set input connected to a parametric output, sub-sets are generated, as shown in Figure 5.37, “AptPlot Step Edit Plot Properties Dialog”. As shown, four generated subsets, **s0**-**s3**, represent the four parametric tasks connected to the input. A sub-set indicates that the data-set will be generated once for the corresponding parametric task. These sub-sets can be used to override the formatting of the parent data set. This is useful when each set should have a different line color or symbol, for example.

Sub-sets cannot change how data is read from the parent input, they can only adjust display properties.

Axial Profile Plots

Axial profile plots allow for plotting the value of numerous data channels at specific axial locations (typically an elevation) at a given time. When **Plot Type** is set to **Axial**, the **Dependent**

Data specifies a channel pattern rather than a specific channel. **Dependent Data** values use the construct %2N, %3N, etc. to indicate the portion of the channel name substituted with axial mesh indexes. The integer in the construct determines how many padding zeros are applied for indexes less than the ceiling index.

The **Axial Locations** property defines the extracted values added to the data set. This dialog can be used to add and remove axial locations, and enable and set an optional axial index override. Axial locations can be added, removed, and reordered with buttons at the top of the dialog. **Index Offset** determines the first axial index substituted into the **Dependent Data** pattern.

In the table, **Axial Location** defines an independent data point in the set, and may be set to the desired value. The **Override Index** column can be used to manually specify the axial index for the data channel associated with the given axial location. When **Override Index** is selected, **Channel Index** can be defined as the substituted index. Finally, **Channel Name** displays the result of substituting the axial index (either automatic or overridden) into the **Dependent Data** pattern.

To help visualize the use of axial plots and the definition of axial locations, consider Figure 5.38, "Edit Axial Locations Dialog". Assume that the data set **Time** value has been set to 180.0. Consider the first row in the table. In the resulting set, the independent data value will be set to 0.1, the axial location. Channel "alpn-21A01" will be read from the input source, and its dependent data value at time 180.0 will be interpolated from the data. The resulting value is used as the dependent data value. This is repeated for each row in the table.

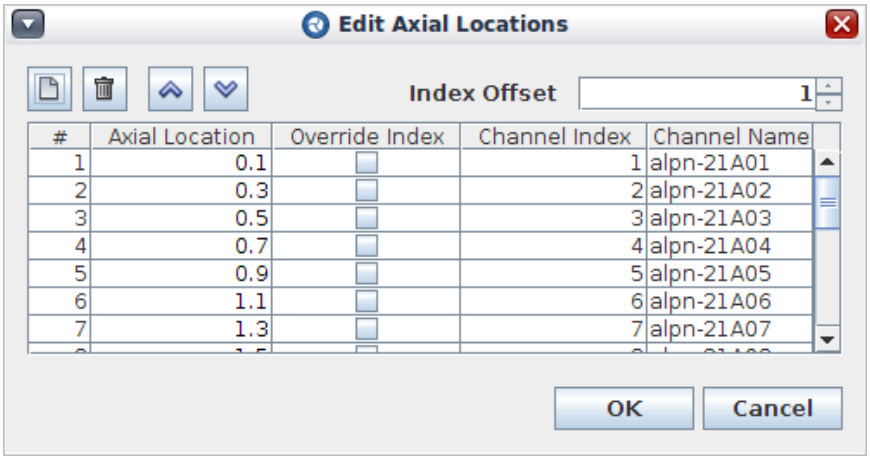


Figure 5.38. Edit Axial Locations Dialog

When the **Dependent Data** channel represents a channel with fine-mesh renodalizations built into the plot file, such as **`rftn`** in TRACE, the need to specify axial indexes is unnecessary. Instead, data is retrieved at the appropriate axial location automatically using the AptPlot **`@elevation`** construct, as shown in Figure 5.39, "Edit Axial Locations Dialog with Dynamic Elevations".

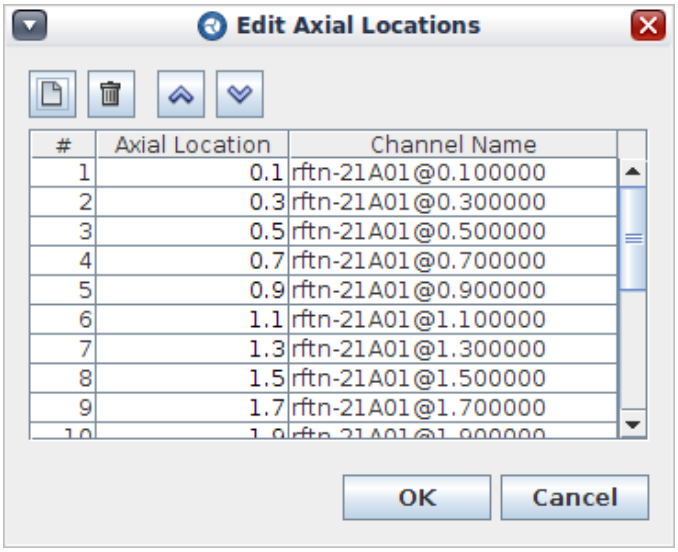


Figure 5.39. Edit Axial Locations Dialog with Dynamic Elevations

5.6.4. AptPlot Step Outputs

Similar to plot inputs, an AptPlot step must define the outputs used to save the contents of generated plots. The step **Plot Outputs** property is used to specify these outputs. Pressing the **Edit** button in the **Plot Outputs** property editor opens the **Edit Plot Outputs** dialog, shown in Figure 5.40, "AptPlot Step Edit Outputs Dialog". Note that this dialog is *non-modal*; other dialogs and Model Editor functions can be used while this dialog is open. Changes made in this dialog can be undone and redone with the standard undo/redo buttons and menu items.

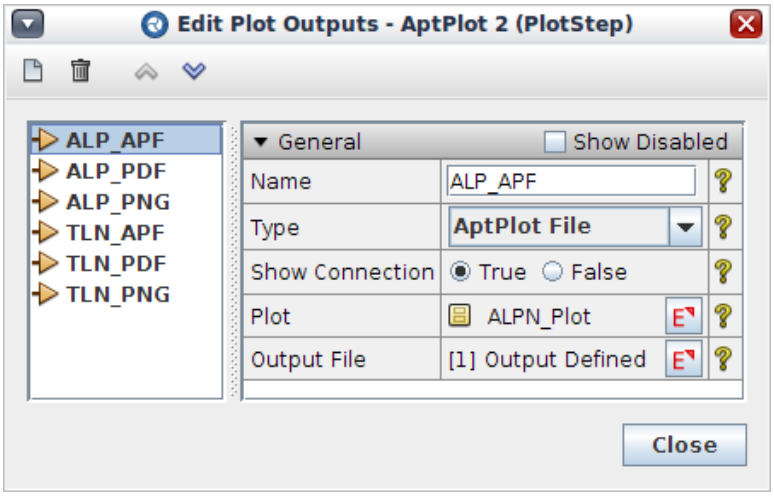


Figure 5.40. AptPlot Step Edit Outputs Dialog

On the left side of the dialog is a list of outputs created by the AptPlot step. On the right side of the dialog are the properties for the selected output. The toolbar at the top can be used to add, remove, and reorder outputs. Each output added in this dialog adds another output to the step, visible on its right side when displayed in a 2D View.

Outputs have the following properties.

- **Name** - The name of the output. Outputs must be unique among outputs in a particular AptPlot step. This name specifies the label for the corresponding output on the step.

- **Type** - Indicates the type of generated file. The standard image outputs supported by AptPlot are provided. The following additional output types are provided.
  - **AptPlot File** - creates an AptPlot File (APF) that can be used to recreate the plot in AptPlot.
  - **ASCII** - creates an ASCII file of space-delimited columns of data.
  - **Variables ASCII** - creates an AptPlot Equation Interpreter variables file in ASCII form.
  - **Variables PIB** - creates an AptPlot Equitation Interpreter variables file in PIB form.

- **Show Connection** - determines whether the connection for the output is displayed on the step.

- **Plot** - the plot represented by this output. For image types, this defines the completed plot output to the image. For others, it defines which data sets are available for export.

- **Exported Set** - When **Type** is set to **ASCII**, this indicates the data set written to the ASCII data file.

- **Exported Sets** - When Type is set to **Variables ASCII** or **Variables PIB**, this is used to determine the data sets written to the variables file, as well as the names of the vectors assigned to each set.

- **Outputs File** - Indicates the external file or step output connected to the AptPlot step output. The **Edit** button opens the **Define Outputs Files** dialog used to manually edit the output (see the section called “Output Files”).

Chapter 6. The Animation Plugin

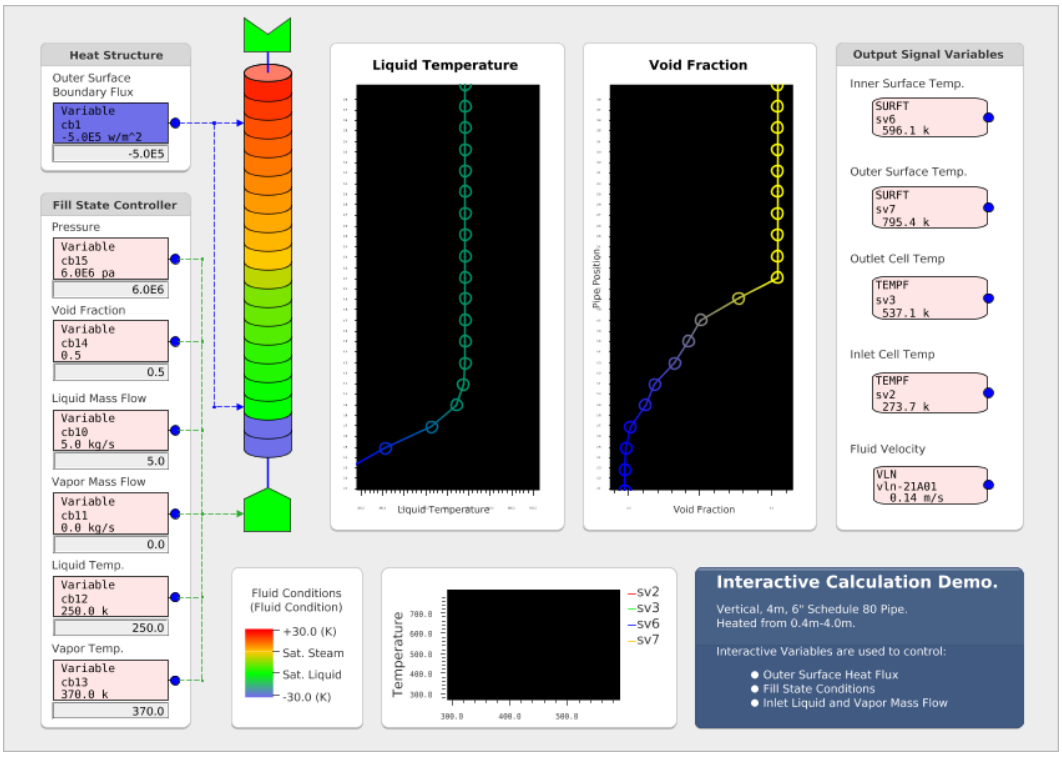


Figure 6.1. Example Animation Mask

SNAP's interactive and post-processing capabilities are predominately realized within its animation displays. Within such a display, the results of a calculation may be animated in a variety of ways.

Animation models (or "masks") are composed of Views containing a number of visual elements, each of which contain properties that may be edited in the Main Property View. An animation display retrieves data from a Calculation Server and represents it visually in some fashion. This data can be from actively running calculations, completed calculations, imported EXTData files, etc.

6.1. Creating a New Animation Model

Pressing the New button on the main toolbar will open the model type selection dialog as shown in Figure 6.2, “New Model Dialog”. Select the Animation model option and press OK. This will create a new Animation model and add it to the Navigator. A single empty 2D view will also be created and opened.

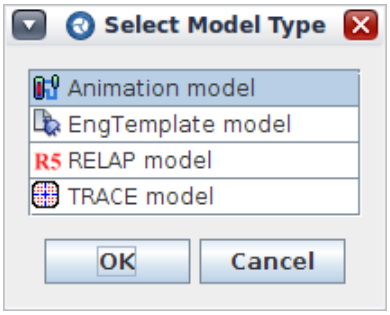


Figure 6.2. New Model Dialog

6.2. Animation Components

Animation models contain several component types: Data Sources, Color Maps, Plot Definitions, and Views. Creating and editing these components is identical to other components in the Model Editor. (Refer to Section 2.3.2, "Creating and Editing Components" for more information.)

The majority of the objects in an Animation model are display beans and Annotations within 2D views. Refer to Section 6.8, "Display Beans" and Section 2.3.4, "Creating Annotations" for more information on the creation and use of display beans and Annotations.

6.3. Data Sources

An animation requires one or more Data Sources to animate data. A Data Source is most often a reference to a job on a Calculation Server. While animating, the Model Editor will retrieve data from the Calculation Server and display visual representations of that data. This process repeats until either the data is exhausted or the user interrupts the animation.

Each Animation Model requires one *Master* Data Source and may have any number of *Slave* Data Sources. Slave sources are other sources of data from a Calculation Server whose data will be interpolated relative to the Master Data Source.

To specify a job for a data source, first select the Data Source from the Data Sources category in the Navigator. Then locate the **Source Run URL** property in the Main Property View and press the red E button to launch the Select Data Source dialog (shown in Figure 6.3, "Select Data Source Dialog").

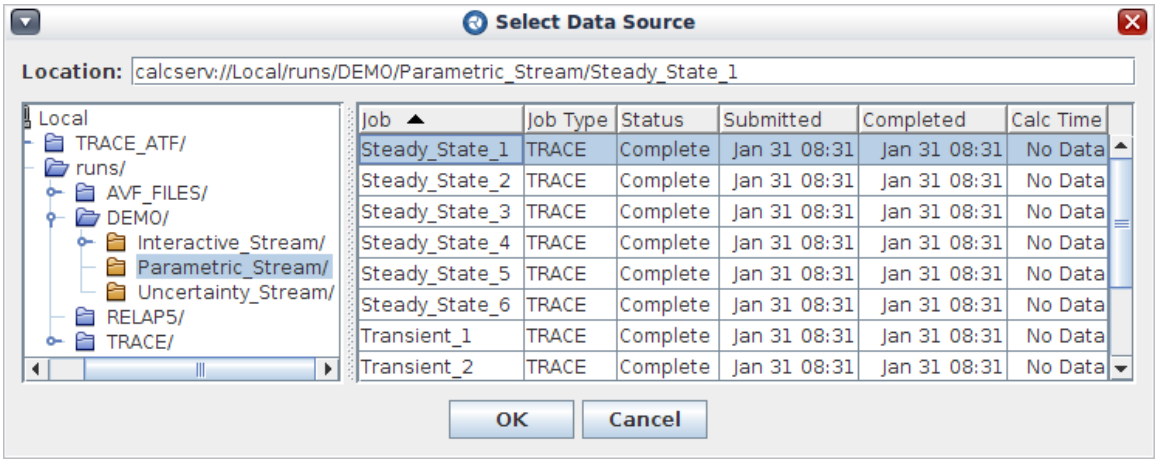


Figure 6.3. Select Data Source Dialog

Expanding each Calculation Server node in the job tree (on the left) will show the root folders for that server. Expanding each folder will show any contained sub-folders and jobs. Selecting a folder will display the list of jobs in that folder (to the right). Selecting a single job in the tree will show the details of that job to the right. Note that the **Location** field is updated when selecting jobs. This location is the URL to the job that will be used for the Data Source. For example: Figure 6.3, "Select Data Source Dialog" shows a selected job (Steady_State_1) that is the result of a parametric TRACE run.

Note Unlike other Animation components, Data Sources cannot be added to a view.

EXTDATA Channel Name Patterns

The EXTDATA Channel Name Patterns group contains a set of patterns used to generate channel names from Volume IDs for the Display Beans connected to this Data Source. The resulting channels can then be used with Color Maps to display Liquid Temperature, Quality, etc., based on the bean's selected Volume ID. This allows EXTDATA jobs that follow an appropriate naming convention to be used with Display Beans that use Volume IDs in the same way as TRACE, RELAP5, or MELCOR jobs.

Patterns can be entered for liquid temperature, vapor temperature, saturation temperature, pressure, void fraction, and quality. Note that Color Maps of type Fluid Condition require only four of these patterns: liquid temperature, vapor temperature, saturation temperature, and void fraction.

The string '%V' is used to represent the volume ID in these patterns. For example: The pattern 'prefix-%V-suffix' would result in the channel 'prefix-301A01-suffix' for volume ID 301A01.

Note These patterns are only used when an Experimental Data (EXTDATA) job is selected for this Data Source.

6.4. Sequenced Data Sources

The Animation Plug-in's Data Source component allows multiple Source Run URLs jobs to be specified as a sequence. To enable this feature, set the **Number of Source Runs** property to the

number of jobs in the sequence (up to 4) then select each job in the sequence (as described in Section 6.3, “Data Sources”) for the First, Second, Third and Fourth run URL properties.

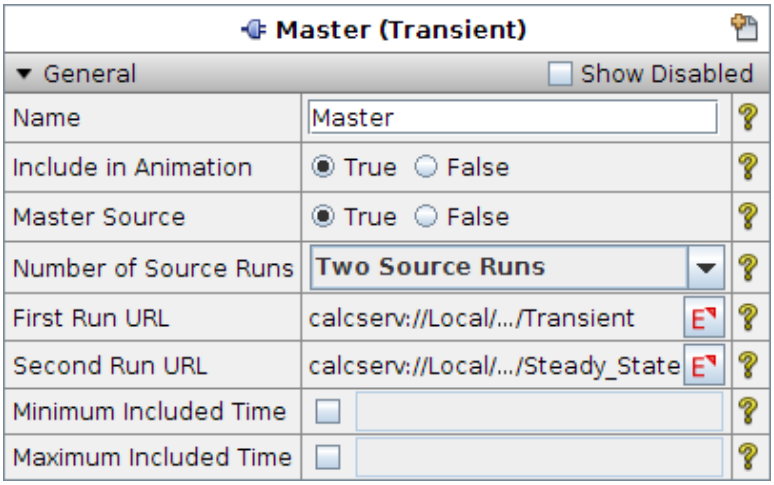


Figure 6.4. Data Source Properties

When using a sequence of jobs the playback time will begin at the first job's start time with the first job as the current job. When the playback time reaches the start time of the next job it will become the current job and re-initialize all display beans to animate that job's data. The same process will occur at the start time of the third job, and so on.

Note Only the last job in a sequence can be actively running. The preceding jobs must be completed.

The channel name list for a sequence is the union of the channel lists of the jobs included in the sequence. Channels that are not available in the current job during an animation will behave in the same manner as other undefined channels. (For example: The single volume bean is colored flat gray.) When the current job changes to a job that includes the previously undefined channel it will be animated with the currently available data as normal.

6.5. The Python Data Source

The Python Data Source is used to animate data values that must be calculated based on data from one or more Data Sources. This Data Source contains two segments of user-defined Python code that are executed at different times.

- The **Initialization Source** is the block of Python source code executed upon activation of this Python Data Source. This code should include:
  - Registering required channels from other Data Sources.
  - Setting initial values for Virtual Data Channels in this Python Data Source.
  - Defining methods and global variables to be used in the Transient Source during animation.
- The **Transient Source** is the block of Python source code executed after each timestep of data is retrieved from the master data source. This code should include:

- Retrieving current values of required data channels in other Data Sources.
- Setting new values for Virtual Data Channels in this Python Data Source with setChannel calls.

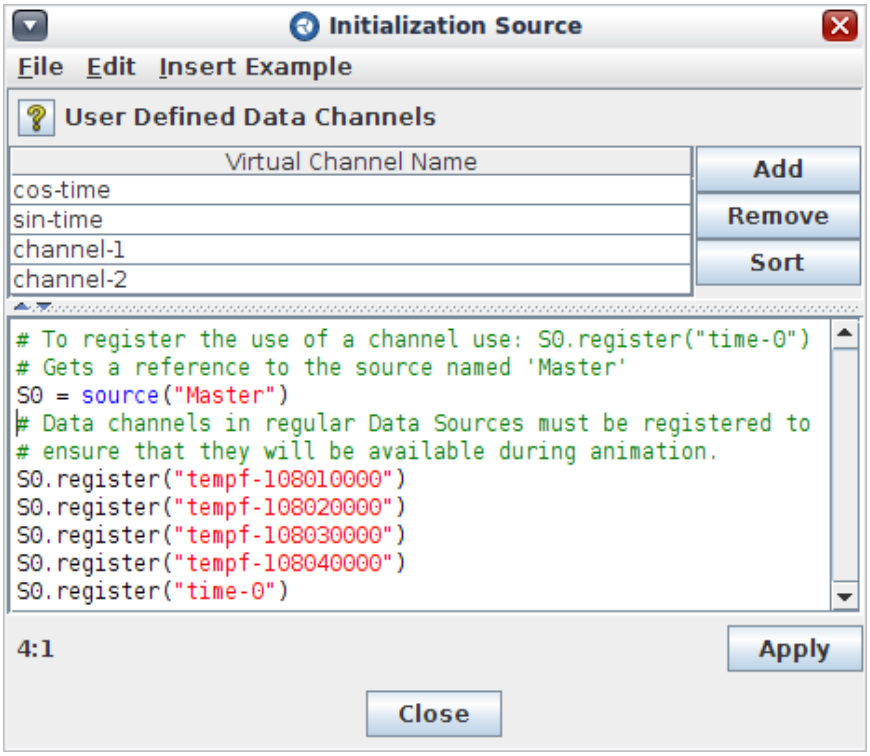


Figure 6.5. Python Data Source Source Editor

Both of these source segments are edited by using the source editor shown in Figure 6.5, "Python Data Source Source Editor". The source editor provides a syntax highlighted text editor for editing Python code. The list of virtual data channels available from this Python data source is at the top of the source editor. Any data channels that will be made available from the Python source must be added to this list.

Example Python code for the various data source specific methods can be found in the **Insert Example** menu. Included in this menu are: **Get a Source**, **Register a Data Channel**, **Get a Channel Value** and **Set a Channel Value**.

6.6. Color Maps

A Color Map is a user defined range of colors and a corresponding user defined range of values used for display bean animation. During an animation, a typical behavior of a display bean is to change color based on the current value of its associated data channel. For example, a Single Element display bean is depicted as a rectangle. If it is tied to a generic color map with a single segment ranging between values of 0 and 100, with respective colors of red and blue, when animating a value of 75, the rectangle will appear as a deep red. Alternatively, when animating a value of 25, the rectangle will display a deep blue. Any values exceeding the minimum and maximum values of the color map will be displayed by the colors defined for the minimum and maximum values (alternatively, colors can be specified for out or range values).

As Color Maps are themselves components, any number can be created and optionally added to a 2D View (see Figure 6.6, “Generic Color Map”).

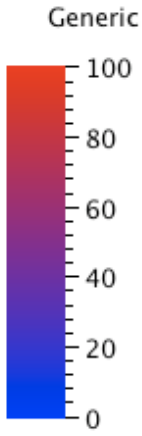


Figure 6.6. Generic Color Map

The properties available for a Color Map are dependent on its current **Color Map Type**. Also, the properties available for each type are completely separate from those for other types. For example: the **Segments** specified for the **Temperature** type are independent of the **Segments** specified for the **Liquid Temperature** type. This means that changing **Color Map Type** will alter what property is being shown in the display beans using that map.

The **Segments** of a color map are consecutive regions of increasing value that may have different starting and ending colors. This allows for different regions of the overall color map to have separate color values. The dialog for editing a color map's segments is displayed in Figure 6.7, “Color Map Segments”.

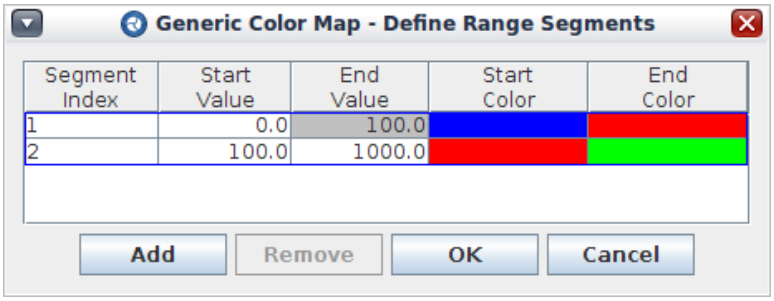


Figure 6.7. Color Map Segments

Temp / Pressure / Quality / Void Fraction

These color map types work with beans that use either a Volume ID or a Data Channel to retrieve animation data. These types use a sets of min/max values and min/max colors to determine the displayed color for each bean. For beans that use a Volume ID, the channel that is displayed will be determined by the animation module, which loads the data from a graphics file.

Generic

This color map type allows minimum/maximum values (and colors) to be specified along with SI and British units for these values.

Generic is the most flexible color map type for display beans that use the **Volume ID** property. When these beans use a color map of type Generic they will use a user-speficied pattern to determine which data channel to animate. These patterns can be specified for each plug-in currently loaded (i.e. one for TRACE, another for RELAP5). Refer to the pop-up help for the **Channel Name Patterns** property for more information on the pattern syntax.

Fluid Condition

This color map type displays the current fluid conditions of a volume (by **Volume ID** only) as a set of color ranges. Separate color ranges are specified for sub-cooled, saturated and superheated liquid. For this color map type, the superheat and subcool values are specified as a number of degrees from the saturation temperature.

Note The Fluid Condition color map type can only be used with beans that use a **Volume ID**.

6.7. Animation Playback

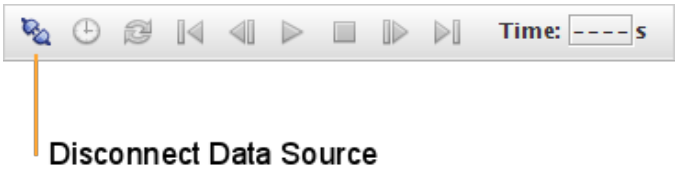


Figure 6.8. Playback Controls

When an Animation Display is open, the Animation Playback Controls will become available in the Main Toolbar. The playback controls are a collection of buttons used to drive an animation. They are illustrated in Figure 6.8, “Playback Controls” and Figure 6.9, “Playback Controls: Connected”.

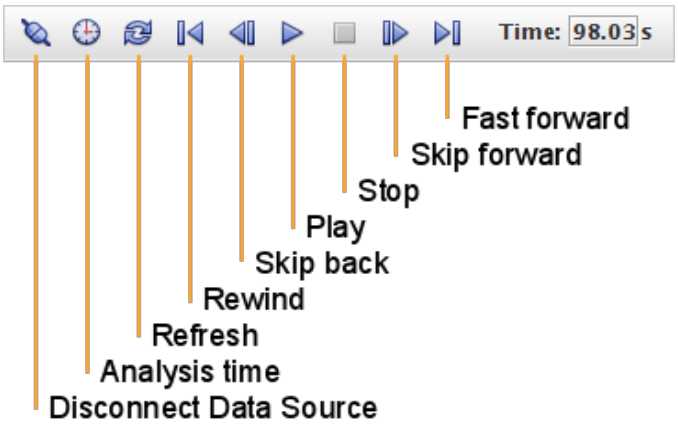


Figure 6.9. Playback Controls: Connected

- **Connect Data Source** - Connects the Animation to the currently defined Data Sources. Once connected to a Data Source, this button becomes the **Disconnect Data Source** button. Only one Animation Model may be connected at a time.

- **Disconnect Data Source** - Disconnects from the current Data Sources.

- **Analysis Time** - Opens the Analysis Time dialog. This dialog is used seek a specific time in the calculation, as well as to configure various time-related settings for the animation.
- **Rewind** - Moves the animation back to the first available time value.
- **Skip Backward** - Moves the animation backward by a fixed number of time steps. This number of time steps can be configured in the Analysis Time dialog by changing the **Skip Forward/Back Steps**.
- **Play Animation** - Begins playback of the animation. Once pressed, this button becomes a **Pause** button which can be used to pause playback.
- **Stop Animation** - Halts the currently animating job. The **Stop** button is only available when the job is currently active (executing) and the current playback is at the last available timestep.
- **Skip Forward** - Moves the animation forward by a fixed number of time steps. This number of time steps can be configured in the Analysis Time dialog by changing the **Skip Forward/Back Steps**.
- **Fast Forward** - Moves the animation forward to the last available time value.

Note Skip Forward and Skip Backward are not supported by all plug-ins.

6.8. Display Beans

Display beans are objects that can be added to a 2D view to display values retrieved from a job on a Calculation Server or an external data source. Display beans are created using the Insertion Tool in the exact same way as Annotations. Display beans can also be resized, reshaped and moved in a similar fashion to annotations.

Note For most plug-ins the quickest way to create a simple display is to copy the contents of a 2D View (e.g. a set of interconnected pipes or control systems) and paste into an Animation view.

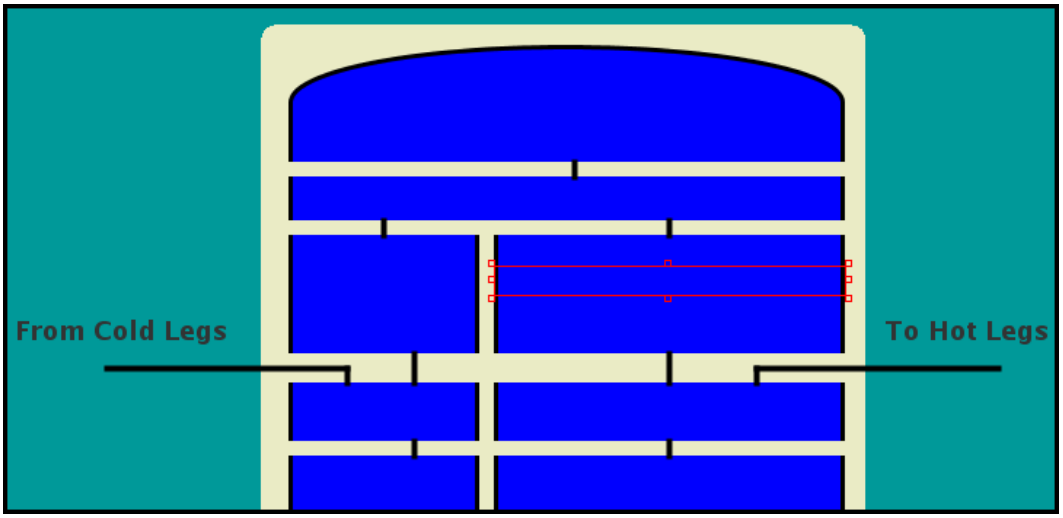


Figure 6.10. display beans Example

Each bean displays its data in a different way but they all share a common set of functionality. The following is a list of features supported by most display beans. In the case that a beans does not support one of these features it will be noted in that bean's description.

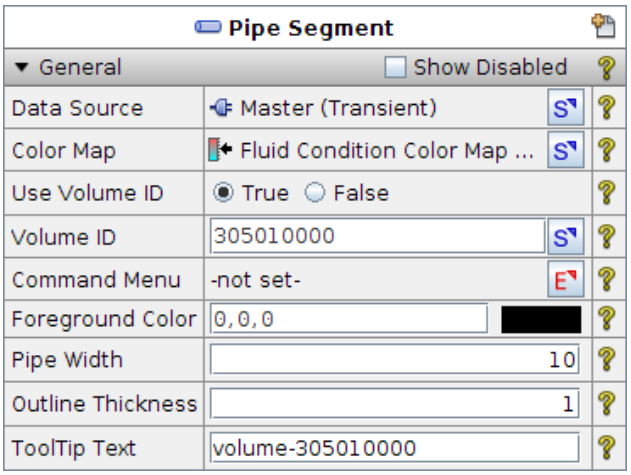


Figure 6.11. Sample Properties

- **Data Source** - All display beans connect to at least one Data Source, some more than one. Those beans that use more than one Data Source include pop-up explaining what aspects of the bean will use each source. Refer to Section 6.3, "Data Sources" for more detailed information on Data Sources and Data Source selection.

- **Color Map** - A Color Map is used to map a data value (or values) to a color for animation. Refer to Section 6.6, "Color Maps" for more detailed information on Color Maps and Color Map selection.

- **Channel Name** - The name of the data channel (or often data channels) that will be animated by the bean. Display beans that use a single data channel will most often use the channel selector shown in Figure 6.12, "Channel Selection Dialog". Beans that use multiple data channels (such as the Axial Plot bean) include more customized selectors.

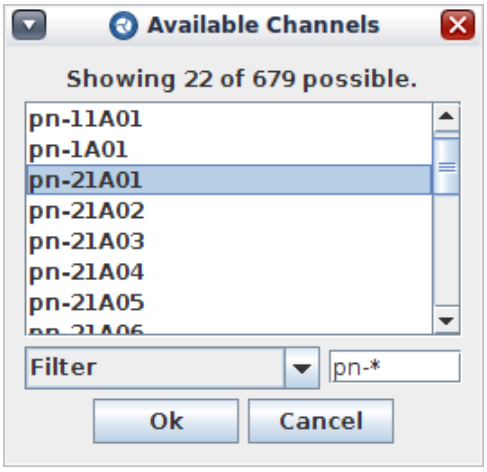


Figure 6.12. Channel Selection Dialog

- **Volume ID** - The Volume ID property is used by some display beans to animate sets of data channels based on the current type of the Color Map selected for the bean. This is most often

used for properties such as void fraction, liquid temperature and fluid conditions where the data channel names can be implied based on the Volume ID and color map type. In these cases, the animation module (for the analysis code being animated) is responsible for determining the required data channels.

Note A bean will only use either channel names or volume IDs to identify data values. Some beans allow switching between the two.

- **Tooltip Text** - This is the tool-tip that is displayed when the mouse is over the bean. Optionally, this tool-tip text can be a valid HTML document but if so it must include the `<html>` and `</html>` tags.

- **Command Menu** - The command menu an easy way to send interactive commands to a calculation during animation. This feature allows the user select an interactive variable and specify a list of values (and corresponding text for each name) for a display bean. Once defined, these values can be selected from the **Commands** sub-menu of the right-click pop-up menu of the bean. Selecting any of the menu items in the **Commands** menu will send the corresponding value to the calculation as an interactive command.

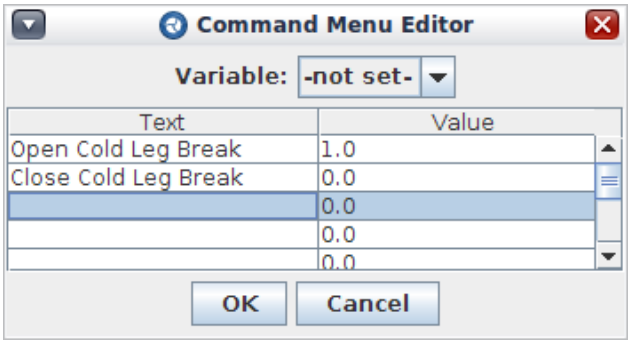


Figure 6.13. Command Menu Editor

The following sections describe the display beans included with the Animation plug-in broken down by their sub-menu location in the Insertion Tool pop-up menu. Additional beans may be installed separately or included with other pre-processor plug-ins. Refer to the individual plug-in users' manuals for more information on the beans they include.

6.8.1. Control System

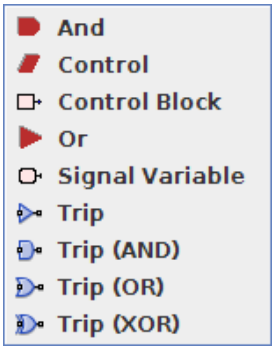


Figure 6.14. Control System Beans

The control systems beans are intended for use in creating logical representations of control systems for animation. These beans do not actually perform any logic operations but merely display the value of their selected data channels.

6.8.1.1. And / Or / Control

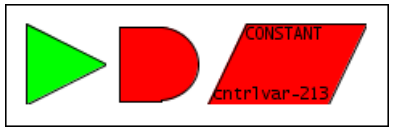


Figure 6.15. And / Or / Control Beans

The And, Or and Control beans (shown above) are more customized versions of the Single Element bean that can be used to layout rough control systems for animation.

6.8.1.2. Control Block / Signal Variable / Trips

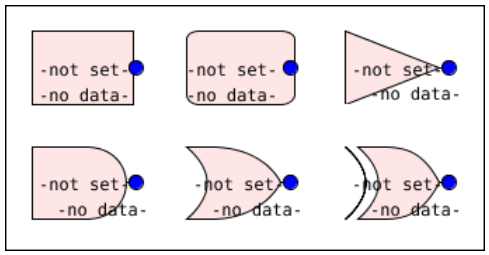


Figure 6.16. Control Block, Signal Variable, and Trip Beans

The Control Block, Signal Variable and Trip beans are intended can be used to layout logical control systems such as those in the TRACE and RELAP plug-ins. These beans can be created via the Insertion Tool or by pasting a control system type component into an animation view from a code plug-in that supports control system display bean creation (such as TRACE or RELAP5).

These beans support the standard single data source and data channel along with a numerical format (for the float display) and a label to replace the data channel name when animating. In addition, SI and British units strings can be provided for display depending on the units setting for the animation model.

6.8.2. Indicators

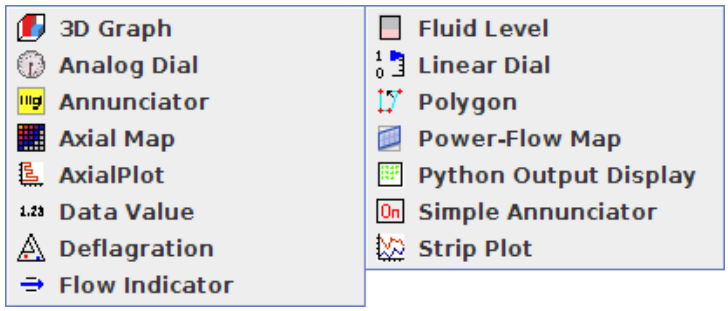


Figure 6.17. Indicator Beans

The Indicators group contains a set of more specialized beans for plotting and data display.

6.8.2.1. 3D Graph

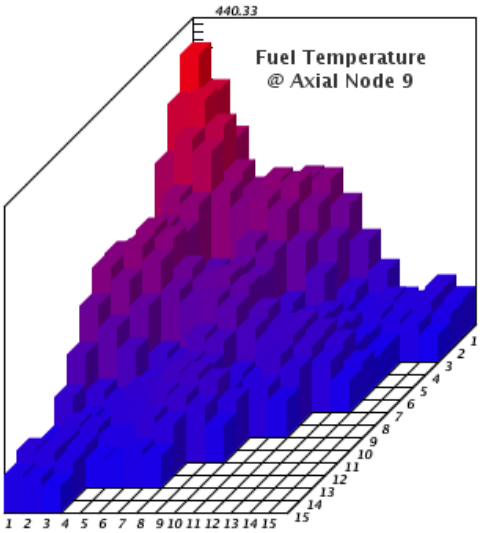


Figure 6.18. 3D Graph Bean

The 3D Graph bean is a three dimensional representation of data values similar in nature to the Stacked Element bean (see Stacked Element). Individual cells in the grid may be disabled, allowing pattern-based selection of data channels to skip unneeded cells. Orientation, grid spacing, and relative lengths of the X, Y, and Z axes can be customized. The example shown in Figure 6.18, "3D Graph Bean" displays fuel temperatures, where all empty spaces in the grid are disabled cells. The following steps are required to setup a new 3D Graph Bean:

1. Insert a new 3D Graph bean using the Insert Tool.

2. Set the number of columns and rows in the bean with the **Dimensions** property.

3. Disable any unneeded cells by editing the **Enabled Cells** property. This displays a dialog of toggle buttons which can be used to enable or disable each cell in the grid.

4. Specify data channel names for each element in the bean.

   Open the *Channel Nodes* dialog by editing the **Data Channels** property of the bean. The table at the top of this dialog contains the channel names for each element in the bean arranged by row and column in the same order as they will be rendered by the bean. These channels can be input individually (by editing each table cell) or input in groups by selecting the **Use Pattern** check box and specifying a **Pattern**.

5. Select an appropriate **Color Map** for the bean.

In addition to the normal bean customization, the 3D Graph bean provides the following additional properties:

- **Orientation** - Determines the relative angles and lengths of the X, Y, and Z axes.
- **Grid Origin** - Determines the position of the grid along the Z axis.
- **Display Wire Mesh** - When set to *True*, a wire-frame mesh is used to display data. When set to *False*, the default bar graphs are displayed.

- **Wire Thickness** - The thickness of lines used in displaying a wire mesh.
- **Column Spacing** and **Row Spacing** - Determines the relative widths of columns and rows. Editing either property opens the *Grid Spacing* dialog, which displays the current spacing values.
- **Display Grid** - When set to *True*, lines indicating the borders of cells in the grid are displayed.
- **Display Opaque Grid** - When set to *True*, the cells of the grid are filled with the color specified by the **Grid Base Color** property.
- **Display Labels** - When set to *True*, numbers indicating the index of cells in the grid are displayed along the X and Y axes. If the **Position Labels on Axes** property is set to *False*, the labels will be displayed along the bottom and right lines of the grid parallel to the X and Y axes.
- **Display Ticks** - When set to *True*, tick marks will be displayed along the Z axis. The spacing of these tick marks is determined by the specified Color Map.

6.8.2.2. Analog Dial

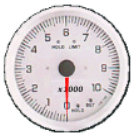


Figure 6.19. Analog Dial Bean

The Analog Dial is a 0-10 representation of a single data channel as a dial. A scale factor must be set to translate the value of the data channel into the 0-10 range.

6.8.2.3. Annunciator

Figure 6.20. Annunciator Bean

The Annunciator bean is a more complete version of the Simple Annunciator that includes High, High-High, Low and Low-Low values.

6.8.2.4. Axial Map

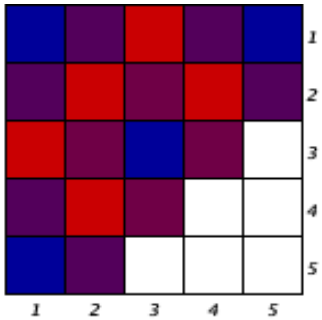


Figure 6.21. Axial Map Bean

The Axial Map bean is a specialized case of the 3D Graph bean. Its functionality is identical to a 3D Graph bean with a Z axis length of 0 and both X and Y angles set to 0. See the section 3D Graph more information on properties specific to this bean.

6.8.2.5. Axial Plot

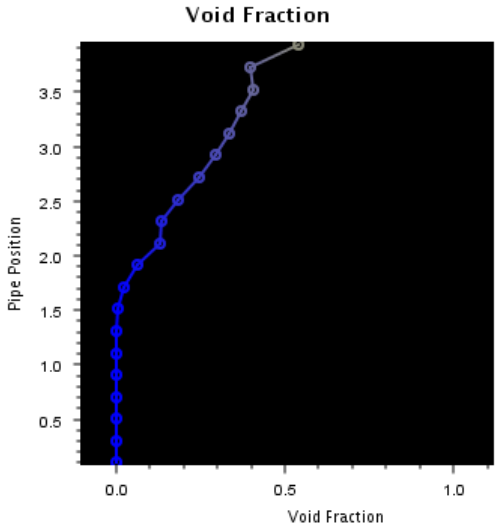


Figure 6.22. Axial Plot Bean

The Axial Plot bean displays a line plot based on a set of data channels for each timestep rendered. If desired, multiple data channel sets can be input to display a line plot for each set. Each data channel set consists of a number of x/y pairs of data channels that determines the location of each point on the plotted line. Optionally, the x channel may be input as Fixed X Positions for situations where there is no x data available as data channels or the x data does not change over time.

A large number of properties are available in the Axial Plot bean to customize the plot, line and axis appearance. Refer to the pop-up help available next each property for more information on axial plot customization.

6.8.2.6. Data Value

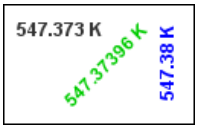


Figure 6.23. Data Value Bean

The Data Value bean simply displays the numerical value of the selected data channel. The font, foreground color, border, etc. can be customized like most display beans. Also, a numerical format may be specified (in C printf format) to control how the data value is displayed.

The Data Value bean also supports displaying the selected channel using user-defined custom units. Once the Custom units display is selected a conversion factor, offset value, and units string

(such as ft/s), can be specified that will be used to display the value. Separate conversion settings are available to display the value in SI or British units.

6.8.2.7. Deflagration

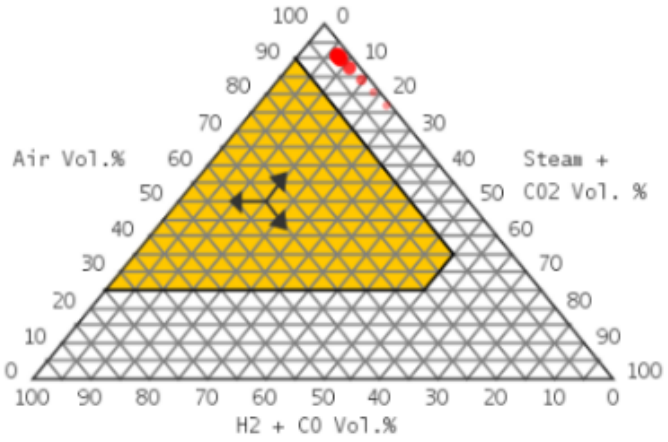


Figure 6.24. Deflagration Bean

The Deflagration bean is used to monitor the concentrations of combustible gases in a containment volume. This data is displayed on a trilinear graph. Trilinear graphs are used to plot three variables that sum to 100% at any location on the graph. The three axes used for the deflagration bean are Air Volume Percentage, H2 + CO Volume Percentage, and Steam + CO2 Volume Percentage. A Deflagration bean requires at least three data channels (**Steam Channel**, **O2 Channel**, and either **H2 Channel** or **CO Channel**), and can optionally include data channels for CO2 and H2/CO. The ignition region (shown in the above figure in orange) is drawn based on the properties listed below, as well as the values of the data channels.

- **Min. O2 Mole Frac**.(XO2IG)- Minimum oxygen mole fraction limit for ignition (Default = 0.05).
- **Diluent Mole Frac. Limit**(XMSCIG)- Maximum diluent (Steam + CO2) mole fraction for ignition (Default = 0.55).
- **H2 Mole Frac. Limit**(XH2IGN)- Hydrogen mole fraction limit for ignition (Default = 0.1).
- **CO Mole Frac. Limit**(XCOIGN)- Carbon monoxide mole fraction limit for ignition (Default = 0.167).

The ignition region depends on time-dependent gas concentrations within each containment volume, and may change during animation. The equations used for determining the boundaries of the ignition region are as follows:

- Air Axis: XO2IG*((X02+XN2)/X02), where X02 is the value of the data channel specified in the **O2 Channel** property, and XN2 is (1.0-XO2-XH2-XCO-XH2O-XCO2), where XH2, XCO, XH2O, and XCO2 correspond to the current values of their associated data channels.
- H2+CO Axis: XH2IGN + (XCO*((1.0-XH2IGN)/XCOIGN)), where XCO is the value of the data channel specified in the **CO Channel** property.

- Steam+CO2 Axis: XMSCIG.

The default behavior of the Deflagration bean (when no data source is connected or the proper data channels have not been initialized) is to fill the ignition region gray, at locations based on default values (25% Air, 55% Steam+CO2, 10% H2+CO). When the data channels are properly initialized and the bean is animating, the ignition region is drawn based on the above equations.

Note The ignition region will disappear for low O2 concentrations.

During animation, the containment volume marker is displayed in the Deflagration bean as a point on the tri-gram whose coordinates correspond to the volume's current gas concentrations. The marker's color can be customized via the **Marker Color** property. A trail of the hydraulic volume's marker can be drawn so that as the marker moves in the bean, its travel path can be displayed as a series of circles that become increasingly smaller and transparent. The length of the hydraulic volume's marker trail, in seconds, can be customized via the bean's **Marker Trail Length** property. The three small arrows drawn in the bean indicate how to "read" values from the deflagration bean. For example, the Air Volume Percentage of the containment volume is found by following the horizontal lattice line to the left side of the bean. The color of the lattice can be modified by changing the value of the **Lattice Color** property.

6.8.2.8. Flow Indicator

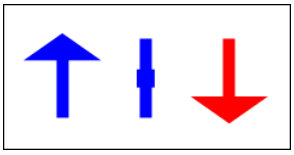


Figure 6.25. Flow Indicator Beans

The flow indicator bean is used to indicate flow beyond a forward and reverse threshold pair. When the data value is above the forward threshold the indicator will display an up arrow. When the data value is below the reverse threshold the indicator will display a down arrow. When the data value is between the two thresholds the indicator will be displayed as a single line. The flow indicator can be customized with a foreground color for forward and backward indication, a border and an orientation (north, south, east or west).

6.8.2.9. Fluid Level

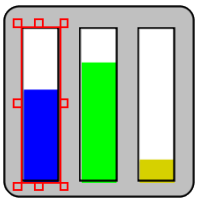


Figure 6.26. Fluid Level Bean

The Fluid Level bean is a specialized version of the Single Volume bean. This bean has a **Level Data Channel** that is used to indicate the liquid fluid level (using specified minimum and

maximum values) and a **Volume ID** used to determine the color of the filled region. Like many other display beans, this bean includes a background color, border, tool-tip text, etc.

6.8.2.10. Linear Dial

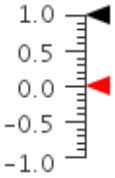


Figure 6.27. Linear Dial Bean

Linear Dials are vertical axes where a "needle" moves along the meter to indicate the current value. Up to two channels can be displayed per linear dial (**Data Channel 1** and **Data Channel 2**). The color used for the channel value needle can also be customized via **Channel 1 Color** and **Channel 2 Color**.

In addition to the normal font, background color, etc. this bean can be customized by setting the following plot appearance related properties.

- Orientation. Sets which side of the bean contains the value needles.
- Maximum Value. The maximum value displayed in the dial.
- Minimum Value. The minimum value displayed in the dial.
- Scale Factor. Incoming values are multiplied by this scale factor.
- Minor Ticks Per Major. The number of smaller ticks displayed between major ticks.
- Major Tick Count. The number of major ticks in the dial. Each major tick also displays the value at that location.

6.8.2.11. Polygon

The Polygon is the same object as the Polygon Annotation described in the section called “Polygon”. When placed in an animation model the polygon includes a **Data Source**, **Color Map** and an either a **Data Channel** or a **Volume ID**.

Polygon beans also support arbitrary levels in a manner similar to the Linear Dial bean (see Section 6.8.2.10, “Linear Dial”). When level tracking is turned on, a horizontal threshold will fill the polygon from the bottom to the top, depending on how close the current value of **Level Data Channel** is to the specified **Maximum Level** and **Minimum Level**.

The following properties relate to displaying level data in a polygon.

- **Use Level Data Channel** - Determines whether this polygon will use level information.
- **Level Data Channel** - The channel that drives the displayed level in the polygon.

- **Level Foreground** - The color of the filled in level region.
- **Maximum Level** - The value of **Level Data Channel** that completely fills the polygon.
- **Minimum Level** - The value of **Level Data Channel** that completely empties the polygon.

6.8.2.12. Power Flow Map

The Power-Flow Map display bean animates percent power and flow rate values over a set of user specified mapping lines. The Power-Flow Map takes data channel values and calculates percentage values against the specified maximum rated percentages. The map provides a series of configurable properties including the ability to specify the maximum and minimum displayed boundary values. A map line editor is provided which creates lines based on the percent values specified. The map fonts, colors, and maximum/minimum displayed boundary values are all configurable through the bean's property view. Scaling factors are provided to allow specific conversions on data values based on current model units.

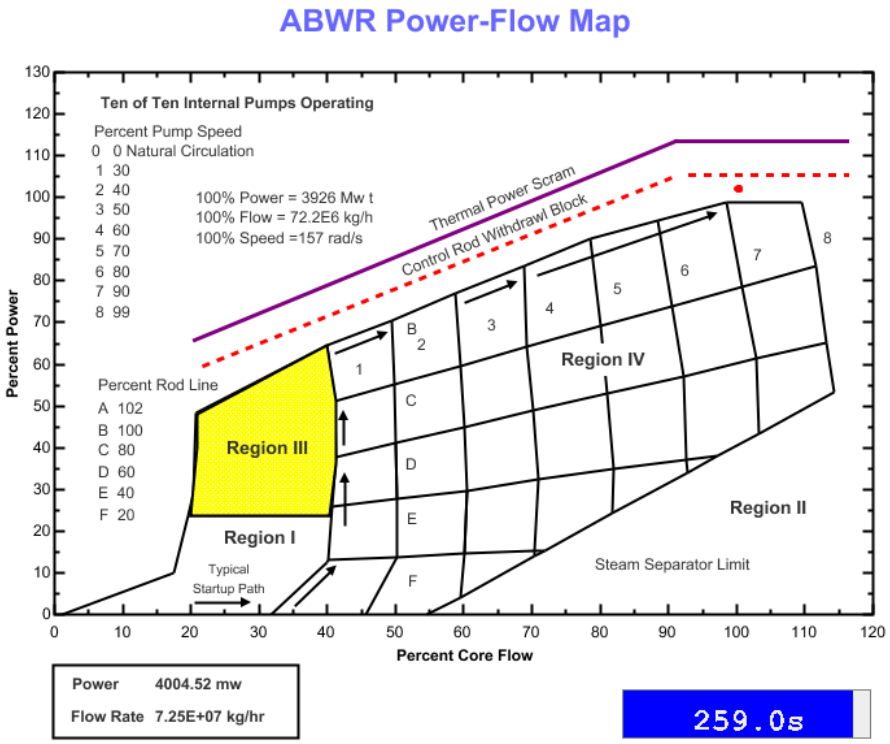


Figure 6.28. Power Flow Map

6.8.2.13. Python Output Display

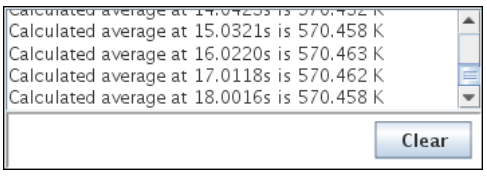


Figure 6.29. Python Output Bean

The Python Output bean is a specialized bean for displaying the printed output and error streams of the Python Data Source (Section 6.5, “The Python Data Source”). The font, background color, border, and error message display properties can be used to customize this bean.

6.8.2.14. Simple Annunciator

Figure 6.30. Simple Annunciator Bean

The Simple Annunciator bean is used to indicate 'On' when the data value has increased beyond the specified **On Threshold** and then 'Off' when the value has decreased beyond the specified **Off Threshold**. The text displayed ('On' above) can be specified by changing the **On Text** property. When off, no text is displayed. Also, like many other display beans, this bean includes a background color, border, tool-tip text, etc.

6.8.2.15. Strip Plot

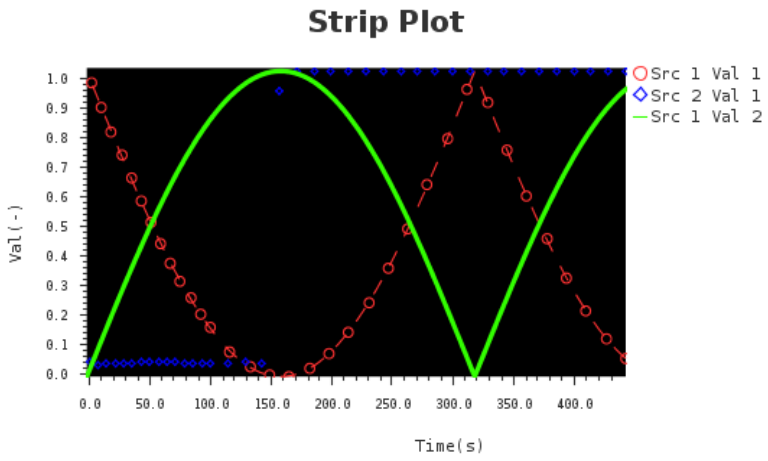


Figure 6.31. Strip Plot Bean

The Strip Plot bean is a color coded line plot of multiple data sets. For each data set, a single plot point is added to the line for each animated timestep. Rewinding time will clear all plotted points and restart the plot from the new current time. A data set is defined by a **Data Source** and a **Data Channel**. The data channel describes the dependent variable for the data set, with the independent variable being the time variable from the associated data source. Data sets also contain properties that control how its values are drawn. The drawing properties for each data set are:

- **Label** - The label used to identify this data set in the plot legend. When set to blank, the data channel name appears in the legend.
- **Line Type** - Controls the style of the line that connects this data set's values. Line can either be drawn dashed or solid.
- **Line Width** - The thickness (in pixels) to draw the line.
- **Line Color** - The color to draw the line.

- **Symbol Type** - The symbol that is drawn for each point.
- **Symbol Size** - The size (in pixels) to draw each symbol.
- **Symbol Skip Factor** - The number of points to skip between each symbol that is drawn.

In addition to the normal title, font, background color, etc. this bean can be customized by setting the following plot appearance related properties.

- **Y-Axis Scaling** - When set to *Logarithmic*, the logarithm (base 10) of all data is plotted instead of the actual values, and the Y-Axis is adjusted to display only major ticks that are powers of 10.
- **Autoscale Y-Axis** - When *True* the Y-Axis minimum,maximum and tick spacing will be automatically determined. When *False* the Y-Axis boundaries are determined by the values of the **Maximum Y Value** and **Minimum Y Value** properties.
- **Plot Background Color** - The background color of the plot region.
- **Show Label** - When set to *True* the legend (shown at the top right) will be displayed showing the channel names and their corresponding line colors.
- **X-Axis Label** - The label used to describe the x-axis.
- **Y-Axis Label** - The label used to describe the y-axis.
- **Time Base** - Determines the x-scaling of the plot and how much data is visible at one time.
- **Time Base Units** - The units for the time base property. Changing the time base units will convert the current time base value to the new units.
- **Major Tick Number Format** - The pattern used to format the x and y axis. This formatting pattern follows that defined for the C function printf.

6.8.3. Interactive

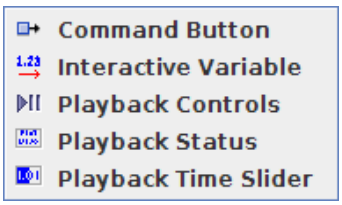


Figure 6.32. Interactive Beans

The **Interactive** category provides display beans used to drive an animation or send commands to an interactive job.

6.8.3.1. Command Button

This bean provides a 2D View button that activates a set of predefined interactive commands when pressed. The **Label** attribute can be used to specify the text displayed by the button. The

Commands property allows specifying which interactive variables are modified and to what values. The **Commands** editor opens a dialog for adding and removing commands. When the model is connected to a data source, new commands can be selected from a drop down list of known interactive variables.

6.8.3.2. Interactive Value

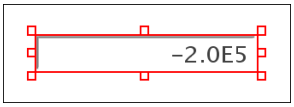


Figure 6.33. Interactive Value Bean

This bean is designed as a controller for an interactive variable inside a calculation. In essence, it is an enhanced Data Value that allows the retrieved value to be edited and sent as an interactive command. To use this bean, first specify a **Data Channel** to be displayed. Then, specify the **Variable Name** of the interactive variable in the calculation. During animation the data value will be displayed in red for a brief time each time the value changes.

6.8.3.3. Playback Controls

The Playback Controls bean is identical to the controls available on the main toolbar. Refer to Section 6.7, “Animation Playback” for more information on these controls.

6.8.3.4. Playback Status

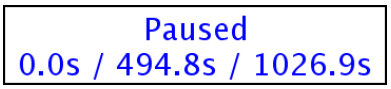


Figure 6.34. Playback Status Bean

The playback status bean shows the status of the current animation including the current, start and end times. The time display can be customized with a numerical format as well as the standard font and foreground color properties.

6.8.3.5. Playback Time Slider

Figure 6.35. Playback Time Slider Bean

This bean can be used to quickly scroll through to an approximate location in a calculation. Clicking inside the bounds (while the view is locked) will skip the playback to a time relative to the clicked position in the slider. Clicking and dragging to interactively skip the animation following the mouse. The time display can be customized using a numerical format, font and foreground color or hidden entirely.

6.8.4. Plant Components

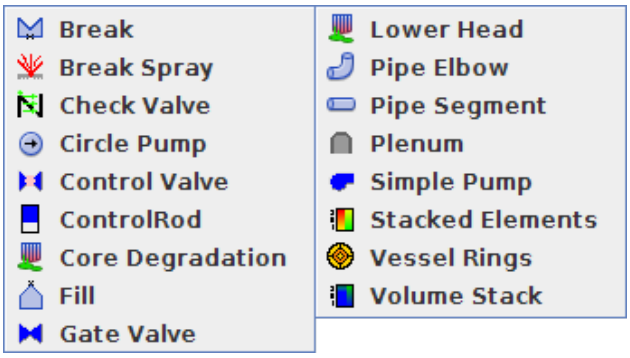


Figure 6.36. Plant Component Beans

6.8.4.1. Break

Figure 6.37. Break Bean

The Break bean is a simple indication of whether a data value is above a specified **Threshold Value**. The break display can be customized by its **On Color**, **Off Color**, **Orientation** and **Line Width**.

6.8.4.2. Break Spray

Figure 6.38. Break Spray Bean

The Break Spray is intended to represent a break in a fluid system with a color change and a spraying animation. The properties available for this bean are nearly identical to the Break Bean with the exception of the **Line Width** used to control the thickness of the spray lines.

6.8.4.3. Check Valve

The Check Valve bean represents a valve that can be fully open or fully closed. When the data value is greater than or equal to the **threshold** the valve is considered open is drawn with arrows colored in the **Open Color**. When the data value is less than the **threshold** the valve is considered closed and is drawn with the **Closed Color**. The Check Valve is driven by a single data channel and allows for rotation in any direction.

Figure 6.39. Check Valve Bean

6.8.4.4. Circle Pump

The Circle Pump bean is a component which has a circular shape and indicates flow direction using both forward and reverse flow indication arrows. The circle pump is driven by a single data channel. When the value of the channel is less than 0.0 - the **Threshold Value**, the forward flow indicator is drawn. This indicator is drawn at the specified **Reverse Flow Angle** colored using the **Reverse Flow Color**. When the value of the channel is greater than 0.0 - the **No-Flow Threshold** value and less than 0.0 + the **No-Flow Threshold** value, an indicator is drawn using the **No-Flow Threshold Color** at the specified **Forward Flow Angle**. When the value of the channel is greater than 0.0 + the **No-Flow Threshold** value an indicator is drawn using the **Forward Flow Color** with the specified **Forward Flow Angle**. The Circle pump provides a **Flip Horizontally** right click pop-up menu item which will swap the flow angles horizontally. Additionally, the pump allows the specification of the **Arrow Outline Color**, **Outer Outline Color**, **Inner Outline Color**, **Inner Circle Color**, and **Background Color**.

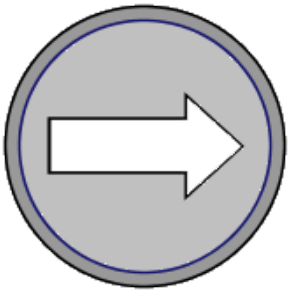


Figure 6.40. Circle Pump Bean

6.8.4.5. Control Rod

Figure 6.41. Control Rod Bean

The Control Rod bean is intended to represent a control rod's location relative to a set of bounds. The **Inserted Position** defines the location that corresponds to the control rod being fully inserted. The **Withdrawn Position** defines the location that corresponds to the control rod being fully withdrawn. The length of the control rod relative to the overall size of the bean can be customized via the **Rod Proportion** property. The control rod is drawn using the Rod Color. The **Rod Orientation** property controls the vertical orientation of the control rod. Up oriented indicates the control rod is inserted from the bottom, and down oriented means the control rod is inserted from the top.

6.8.4.6. Control Valve

Figure 6.42. Control Valve Bean

The Control Valve bean represents a valve that can be partially open. When the data value is greater than or equal to the **Open Value** the valve is considered open is drawn using the **Open Color**. When the data value is less than or equal to the **Closed Value** it is considered closed and is drawn using the **Closed Color**. Values in between the two are considered partially open and an interpolated color between the two colors is used.

6.8.4.7. Core Degradation

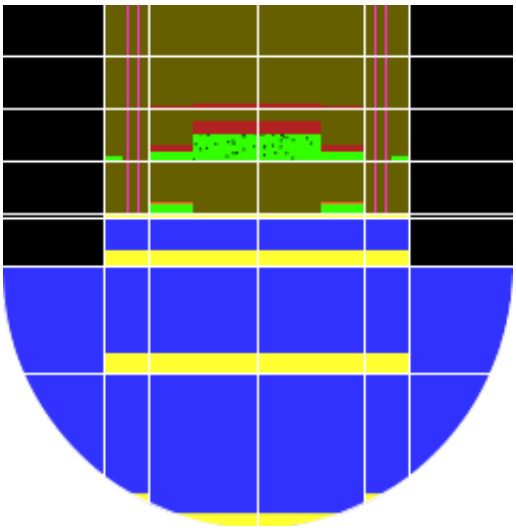


Figure 6.43. Core Degradation Bean

The Core Degradation bean represents a Core component based on its geometry and the volume percentages of certain properties in its cells. The Core Degradation Bean monitors the volume percentages of water (blue), fuel (pink), particulate debris (green), molten pools (red and orange), supporting and non-supporting structures (yellow) in both the channel and bypass portions of each core cell. Unmodeled cells (cells with elevations greater than the difference between the **Core Support Plate Elevation** property and the **Lowest Vessel Elevation** property, and inner radii greater than the value of the **Core Radius** property) are shown in black. The lower plenum curve is defined by the **Vessel Inner Radius** property. The geometry of the bean can be modified via the **Axial Lengths** and **Radial Lengths** properties.

Each cell must be mapped to a control volume that provides water level and volume data for the channel and bypass portions of each cell. The cell-control volume maps for the channel and bypass portions are editable via the **Channel-Control Volume Map** and **Bypass-Control Volume Map** properties. Only control volumes specified in the **Control Volumes** property can be mapped to cell channel and bypass portions; control volumes specified in this property also have maximum and minimum elevations specified. Geometry and control volume data are only

input for one "half" of the core; the data is mirrored for the other side. The **Drawing Mode** property controls whether the whole core, the left half, or the right half are drawn. The black "specks" in the green particulate debris beds indicate the porosity of the debris beds. The **Lower Head Type** property controls the drawing of the lower plenum curve, when this property is set to **Cylindrical**, the lower plenum is modeled as a rectangle. The Reactor Type property can be used to limit the drawing of the bypass portions of core cells. When the value of **Reactor Type** is set to **PWR**, the bypass portion of core cells is only drawn in the outermost radial ring, otherwise the bypass is drawn for all cells.

Note The Core Degradation Bean currently supports only MELCOR data sources.

6.8.4.8. Fill

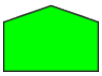


Figure 6.44. Fill Bean

This bean differs from the Break Bean only in its shape (see Break Bean).

6.8.4.9. Gate Valve

Figure 6.45. Gate Valve Bean

The Gate Valve bean represents a valve that can only be fully open or fully closed. It differs from the Break Bean only in its shape.

6.8.4.10. Lower Head

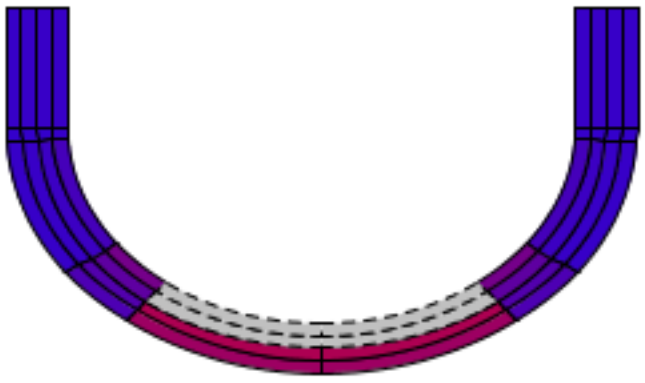


Figure 6.46. Lower Head Bean

The Lower Head bean represents a Core component's lower head based on its geometry and the values of its associated plot data. The geometry of the Lower Head bean is defined by the **Lower**

Head Type, **Inner Radius**, **Lower Head Thickness**, **Thickness Scale Factor**, and **Lower Head Height** properties. The outer radius of the lower head is determined by multiplying the Lower Head Thickness by the Thickness Scale Factor. The lower head is divided radially into rings, and the thicknesses of these rings are editable via the **Ring Thicknesses** property. Each ring thickness is multiplied by the Thickness Scale Factor to determine its relative thickness. The lower head bean is divided into a number of segments, whose lengths are defined by the **Segment Radials** property. Each length specifies the inner radius of the corresponding segment. Duplicate entries in the Segment Radials array indicate a vertical segment of the lower head. Vertical segments of the lower head are equally spaced based on the difference between the Lower Head Height property and the largest Segment Radial value. As in the Core Degradation bean, input is entered for only one "half" of the lower head, and the data is mirrored for the other side. The **Drawing Mode** property controls whether the whole lower head, left half, or right half are displayed. The radial rings are indexed from the outside in, meaning that radial 1 is the outermost ring.

The Lower Head Bean compares the values of plot data associated with each lower head node against failure criterion. During animation, each node's area is filled in using a color determined by the current data value and the color map specified in the **Color Map** property (Note: the Lower Head Bean currently does not support Color Maps of type Fluid Condition). The **Lower Head Failure Type** property defines the plot variables that are associated with this bean, and can be set to **Temperature** or **Plastic Strain**. When the Lower Head Failure Type is set to Temperature, the value of each lower head node's associated temperature plot variable is checked against the value of the **Lower Head Failure Temperature** property. If the node's temperature exceeds the failure temperature, the node's outline is drawn using a dotted line, and the node's area is greyed out. If the Lower Head Failure Type is set to Plastic Strain, each node's plastic strain is compared against the value of the **Plastic Strain Failure Threshold** property to determine node failure.

Note The Lower Head Bean currently supports only MELCOR data sources.

6.8.4.11. Pipe Elbow

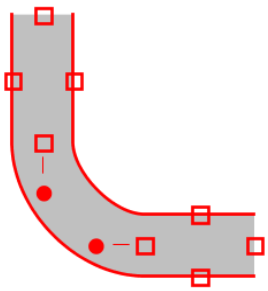


Figure 6.47. Pipe Elbow Bean

The pipe elbow bean is an indicator intended to resemble a piping elbow. The elbow is defined by four points that must be placed on the View when the bean is created. These form the center "line" of the pipe, with the bend placed between the second and third points.

Pipe Elbow beans have the following custom properties:

- **Use Volume ID** - When set to **True**, the pipe may specify a **Volume ID** to select the data to animate. When set to **False**, a single **Channel Name** is specified instead.

- **Pipe Width** - The width of the pipe from the center line. This value may also be edited directly in the view, as described below.

- **Outline Thickness** - The thickness of the outline (in pixels).

When a pipe elbow selected, as shown above, a number of control points become available for editing the bean directly in the View. The points in the pipe path may be repositioned by dragging any of the four square control handles in the center of the pipe. The width of the pipe may be directly manipulated by dragging the control handles on the sides of the pipe. Finally, the circular control points extending from the second and third points in the path control how far the pipe bends in that direction; they may be dragged into and out of the anchor point to adjust the curve.

6.8.4.12. Pipe Segment

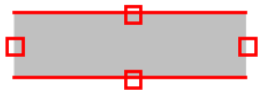


Figure 6.48. Pipe Segment Bean

The pipe segment bean is an indicator intended to resemble a single, straight length of pipe, but may also be used for any rectangular value indicator. The segment is defined by two points that must be placed on the View when the bean is created. The two control points form the center line of the pipe.

Pipe Segments beans have the following custom properties:

- **Use Volume ID** - When set to **True**, the pipe may specify a **Volume ID** to select the data to animate. When set to **False**, a single **Channel Name** is specified instead.

- **Pipe Width** - The width of the pipe from the center line. This value may also be edited directly in the view, as described below.

- **Outline Thickness** - The thickness of the outline (in pixels).

When a pipe segment selected, as shown above, a number of control points become available for editing the bean directly in the View. The points in the pipe path may be repositioned by dragging either of the square control handles in the center of the pipe. Additionally, the width of the pipe may be directly manipulated by dragging the control handles on the sides of the pipe.

6.8.4.13. Plenum

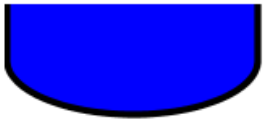


Figure 6.49. Plenum Bean

The plenum bean is a more customized version of a Single Volume bean that is intended to resemble an upper or lower plenum. Like the Single Volume, the this bean uses a **Volume ID**

to select the data to animate. In addition to the normal Font, Foreground Color, Border, etc. this bean's shape and appearance can be customized with the following properties.

- **Plenum Type** - The plenum orientation type. (Upper or Lower)
- **Curved Fraction** - The fraction of the plenum that is curved. (0.0 - 1.0)
- **Outline Width** - The thickness of the outline (in pixels).

6.8.4.14. Simple Pump

Figure 6.50. Simple Pump Bean

The Simple Pump bean is a simple indication of whether a data value is above a specified **Threshold Value**. The pump display can be customized by its **On Color**, **Off Color** and **Facing East** properties.

6.8.4.15. Single Element

Single Element has been replaced by the Pipe Segment bean. Single Elements saved in existing MED files are automatically converted to pipe segment beans upon opening the model.

6.8.4.16. Single Volume

Single Volume has been replaced by the Pipe Segment bean. Single Volumes saved in existing MED files are automatically converted to pipe segment beans upon opening the model.

6.8.4.17. Stacked Elements

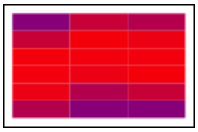


Figure 6.51. Stacked Element Bean

The Stacked Element bean is a 2 dimensional stack of Single Element beans designed to simplify the process of visualizing areas of dense nodalization or long stretches of sequential channel numbering. The example shown in Figure 6.51, “Stacked Element Bean” has 6 rows and 3 columns of fuel rod temperatures.

To create a complete Stacked Element bean:

1. Insert a new Stacked Element bean using the Insert Tool.
2. Set the number of columns in the bean by adding entries to the **Column Widths** table.

This table is used (in normalized form) to scale the size of each column in the bean. The number of entries in this table determines the number of columns and the value for each determines its relative width.

3. Set the number of rows in the bean by adding entries to the **Row Heights** table.

   This table, like the Column Widths table, is used to determine the number of rows and their relative heights.

4. Specify data channel names for each element in the stack.

   Open the *Select Channels* dialog by editing the **Node Volumes** property of the bean. The table at the top of this dialog contains the channel names for each element in the bean laid out by row and column in the same order as they will be rendered by the bean. These channels can be input individually (by editing each table cell), selected individually (by selecting the **Channel Selection** radio button) or input in groups by specifying a **Pattern**.

   Refer to the pop-up help of the Select Channels dialog for a more detailed discussion of channel selection individually and by pattern.

In addition to the normal bean customization, the stacked element bean provides 2 stack-specific properties:

1. **Opaque Nodes** - When this is set to False, nodes that do not have channels specified will be transparent.

2. **Draw Outline** - When this is set to true, a line border will be drawn in the foreground color around each element in the stack and around the entire stack.

6.8.4.18. Vessel Rings

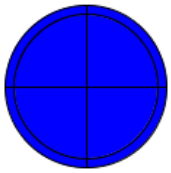


Figure 6.52. Vessel Rings Bean

The Vessel Rings Bean is a series of concentric circles each divided into azimuthal sectors. The Rings bean may have any number of rings, and each ring may have any number of azimuthal sectors. Each azimuthal sector is assigned a volume number, and the first sector may include an offset angle indicating where it begins around the circle. Optionally, regions that have no volume number may be transparent, appearing as a void in the volume.

6.8.4.19. Volume Stack

This bean differs from the Stacked Element bean only in that **Volume ID**s are selected for each element rather than channel names (see Stacked Element).

6.8.5. TRACE 3D Vessel

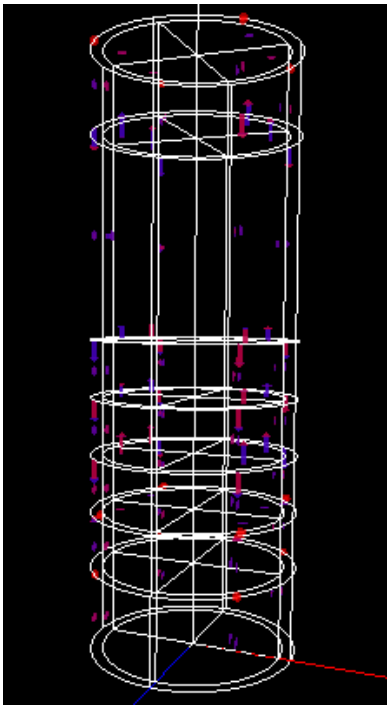


Figure 6.53. TRACE 3D Vessel Bean

The TRACE 3D Vessel Bean is a 3D representation of a TRACE Vessel component. The **Vessel Type** property determines the data the bean will visualize. When this property is set to **Solid**, each vessel cell is rendered in a solid color correspond to that cell's current fluid conditions. When the Vessel Type is set to **Wireframe**, the bean provides a wire-frame view of a Vessel component similar to the one displayed in the TRACE Vessel Geometry dialog. In wireframe mode, the velocities of each vessel cell are displayed as arrows. For each vessel cell, flow indicator arrows display both liquid phase and vapor phase flow along each axis. The colors used to render these arrows are editable via the **Liquid Flow Indicator Color** and **Vapor Flow Indicator Color** properties. The magnitude and orientation (positive or negative along each axis) of each flow indicator arrow is controlled by the value of the corresponding vessel cell's plot data during animation. Scale factors can be applied to the magnitude of flow indicator arrows along each axis via the **Flow Indicator X Scale Factor**, **Flow Indicator Y Scale Factor**, and **Flow Indicator Z Scale Factor properties**.

Note The TRACE 3D Vessel Bean currently supports only TRACE data sources, and can only be created via the **Copy as Bean** button found in the TRACE Vessel Component's Geometry Dialog.

6.8.6. TRACE 3D Vessel Rings

The TRACE 3D Vessel Rings bean is used to visualize the data for a TRACE vessel. The bean is displayed as a stack of discs broken down into cells by radial rings and azimuthal sectors, with each disc representing a paritcular axial level. 3D Vessel Rings can be created one of two ways. The bean can be created via the Insertion Tool, and is found under the TRACE Beans category. Also, when in a TRACE model, you can right-click a Vessel component and select **Copy as 3D**

Rings Bean from the pop-up menu. Pasting into the view of an animation model will then create a 3D Vessel Rings bean using the geometry and component number of the selected vessel.

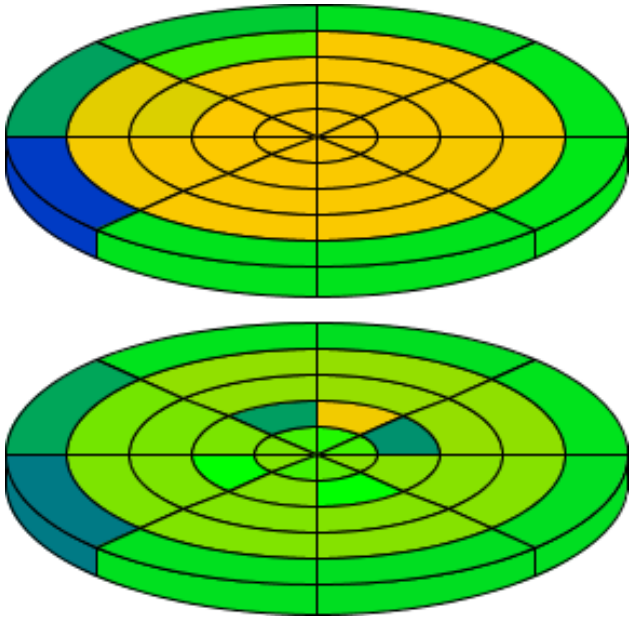


Figure 6.54. TRACE Vessel Rings Bean

The 3D Vessel Rings bean has the following custom properties:

- **Vessel CC Number** - the component number for the vessel represented by the bean. The bean uses the geometry, axial orientation and referenced color map to automatically determine the volume IDs and channel names required to animate the vessel.
- **Axial Orientation** - determines whether the axial levels are ordered bottom-up or top-down.
- **Vessel Geometry** - used to set the overall structure of the vessel. The editor opened for this property is described below.
- **Tilt** - controls the tilt of the vessel rings towards the viewer. At 0 degrees, the rings are seen from the top. At 90 degrees, the rings are fully tilted up and only the edge is visible.
- **Rotation** - defines a rotation for the rings around their midpoint. Valid values range from -360 to 360 degrees, with 0 degrees indicating no rotation.
- **Disc Distance** - the distance between disc center points as a factor of overall disc-width. Distance is defined as a factor so that resizing the disc does not fundamentally alter the positioning of rings within the bean.
- **Edge Height** - the maximum height of the ring edge in pixels.
- **Draw Labels** - a flag that determines whether axial level labels are drawn next to each ring.

3D Vessel Rings Geometry Editor

Editing the Vessel Geometry property opens the Vessel Geometry editor.

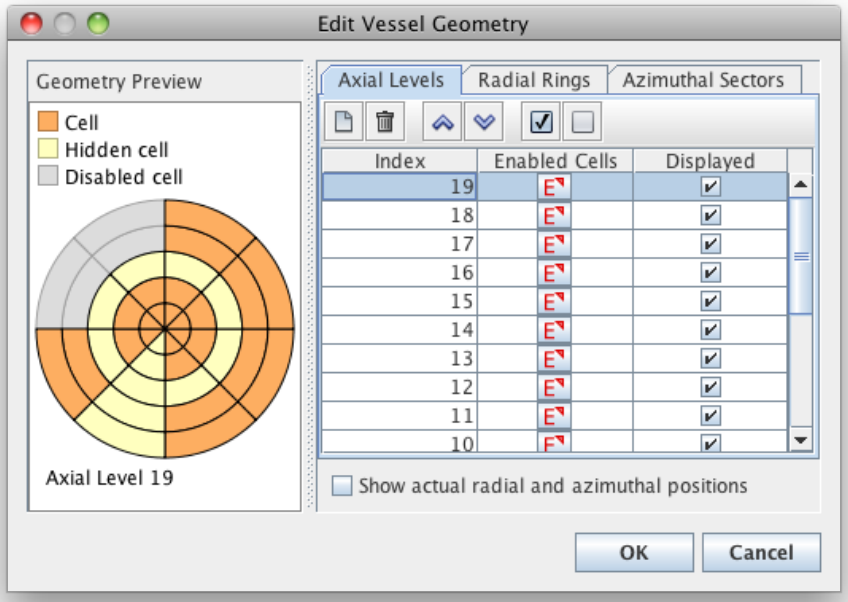


Figure 6.55. Vessel Geometry Editor

The tabbed panel on the right is used to edit the geometry. Each tab contains a table with three columns, where the first column always lists node indexes. In the Axial Levels tab, the central column edits the enabled cells for each axial level (described in greater detail below). In the Radial Rings tab, the central column edits the radii for the vessel rings. In the Azimuthal Sectors tab, the central column edits the azimuthal angle for each sector. In all three tabs, the right-most column toggles the visibility of that level, ring or sector. When radial rings or azimuthal sectors are disabled, the hidden region is shown as an empty area in the 2D view. When axial levels are hidden, their corresponding slices are simply not shown, and the remaining slices are shifted upward or downward depending on the axial orientation.

The check at the bottom of the table toggles whether radial and azimuthal positions are shown as actual distances from the origin or as delta values from the previous node. For example, a vessel with radii of 0.75ft, 1.0ft, and 1.5ft will show radii of 0.75ft, 0.25ft, and 0.5ft when this check-box is deselected.

Note When editing with actual positions, all radii and azimuthal angles must be greater than or equal to zero and monotonically increasing. At least one value must be defined for both. Azimuthal angles cannot exceed 360 degrees. The preview area will indicate any violations of these constraints.

The toolbar at the top of each table contains the following operations. Press the **New** ( ) button to add a row to the table. Press the **Remove** ( ) button to delete the selected rows. Reorder the selected cells with the **Move Up** ( ) and **Move Down** ( ) buttons. Toggle the visiblity of all nodes with the Show All ( ) and Hide All ( ) buttons. Finally, the additional buttons in the Azimuthal Sectors tab can be used to quickly normalize all nodes to equal angles in a full-core (360 degrees), half-core (180 degrees) and quarter-core (90 degrees) configuration.

To the left of the tabbed panel is a preview area which displays the general structure of the vessel geometry. The preview is color coded based on the hidden and disabled status of the individual cells. Making a selection in the any of the tables will highlight the selected level, ring or sector in the preview area. Conversely, nodes can be selected in the preview with the left mouse button. Making a selection in the preview area will update the selection in the Radial Rings and Azimuthal Sectors tables to match the chosen cells. In the figure above, radial ring 2 and azimuthal sector 3 have both been hidden, while the cells in the range azimuthal 5-6 and radial 3-4 have been disabled.

The enabled status of cells is edited from the central column of the Axial Levels table, which opens the Enabled Cells window. This shows a table of values broken down into columns by azimuthal sector and rows by radial ring. The enabled flag controls whether a cell can be shown on that level; an enabled cell can still be hidden the visibility flag on a radial ring or azimuthal sector.

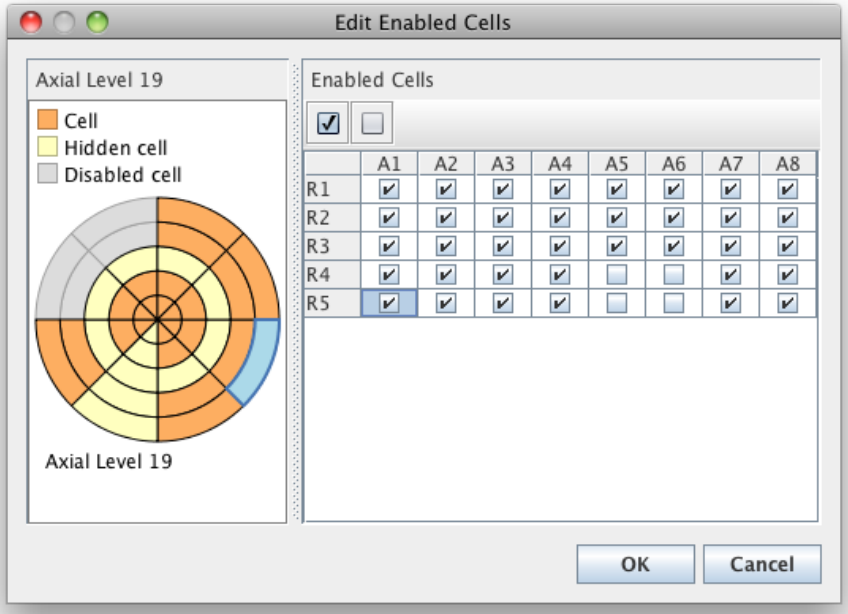


Figure 6.56. Enabled Cells Window

Press the **OK** button to store the changes made to the geometry. The edit can be discarded at any time by pressing the **Cancel** button or closing the geometry editor.

6.9. Plot Definitions

Animation models may specify a number of predefined plot definitions for use in plotting data with specific formatting. Plot definition components are created in the **Plot Definitions** category. The following sections describe plot definition properties, how they tie into AptPlot, and view integration.

It should be noted that throughout these sections, references will be made to animation model Data Sources. Plot definition functionality is exclusive to *local* data sources: jobs run on a

calculation server executing on the user's machine. Trying to use display a plot with a remote Data Source will omit any remote sources and display an error.

6.9.1. Plot Definition Properties

Selecting a plot definition in the Navigator will display its properties in the Property Editor (Figure 6.57, "Plot Definition General Properties").

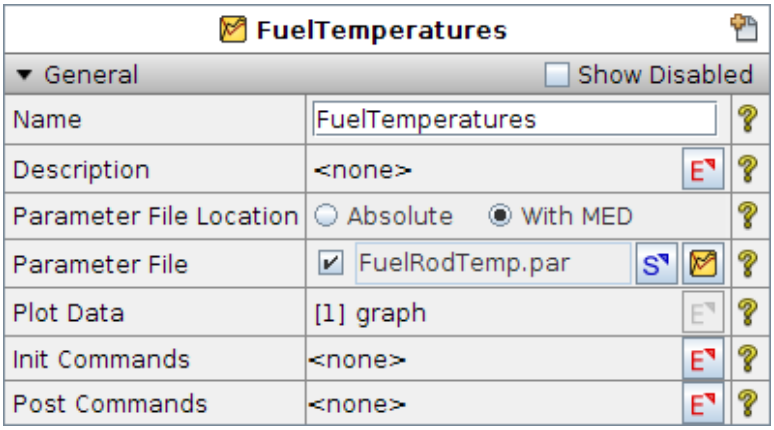


Figure 6.57. Plot Definition General Properties

The following properties are available in the **General** attribute group.

- **Name** - An arbitrary name. The name controls how the plot is labelled in the Navigator.
- **Description** - An arbitrary description for the plot definition.
- **Parameter File Location** - Determines how the AptPlot parameter file (described below) is located. When set to **Absolute**, the complete path to the file will be specified. **With MED** indicates that the plot file will reside in the same directory as the model MED file. The latter option cannot be selected for new, unsaved models.

  Some notes on changing this property when **Parameter File** is already set. When changing from **Absolute** to **With MED**, the folder path of the parameter file is removed automatically, and the file is assumed to exist in the same folder as the MED. If the file does not exist in this location (and exists in the original path), a prompt will be displayed asking whether the file should be copied to the MED folder. When changing from **With MED** to **Absolute**, the file name is prefixed with the complete path to the MED file. Warnings will be displayed in the message window if any of the indicated files do not exist.

- **Parameter File** - The location of an AptPlot parameter file. This is the main source of plot customization. An AptPlot parameter file typically contains commands that define the plot, from its background color to the graph count to the formatting of individual data sets. A parameter file does not normally contain data set values.

  The **Select** button on the editor opens a file chooser for selecting the parameter file. When **Parameter File Location** is set to **With MED**, this button opens a file selector limited solely to the folder housing the model MED. The **Plot** button in the Parameter File editor will launch AptPlot and load the indicated parameter file without providing plot data.

Note Parameter files can be explicitly created in AptPlot by selecting **Plot** > **Save parameters...** from the main menu. The result is a file selection dialog that determines where the parameter file is stored and which graphs it includes.

- **Plot Data** - Defines the graphs in the plot and which data channels are displayed in each. Editing this attribute is described below.
- **Init Commands** - An arbitrary series of AptPlot batch commands that will be executed before data is plotted.
- **Post Commands** - Similar to **Init Commands**, but executed after data is plotted.

The following properties are available in the **DIsplay** attribute group.

- **Display Label** - An optional label attached to the plot definition when displayed in a View. This is described in further detail in Section 6.9.3, “Plot Definitions in a View”.
- **Label Font** - Defines the font of the **Display Label** text in a View.
- **Button Contents** - Determines whether the plot definition display button contains an icon, text, or both.
- **Button Icon** - The icon displayed in the button. The editor can be used to select an image file for use as the icon. Any image selected by the property editor will be saved directly in the MED. This property is only enabled when **Button Contents** is set to display the icon.
- **Button Icon Size** - The size of the icon displayed in the button. The value can be entered in the form "**`[width, height]`**" (sans quotes). Neither the square brackets nor the comma in the text value are necessary when entering the width manually. This property is only enabled when **Button Contents** is set to display the icon.
- **Button Text** - Defines the text displayed in the button. This property is only enabled when **Button Contents** is set to display text.
- **Font** - The font used to display text in the button. This property is only enabled when **Button Contents** is set to display text.
- **Foreground Color** -

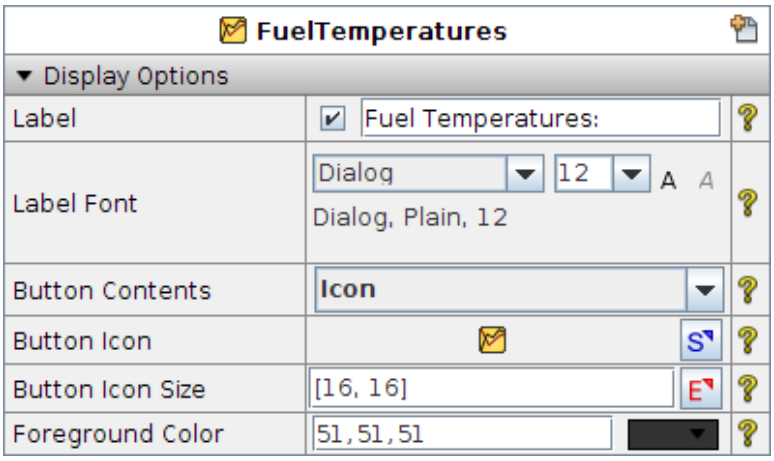


Figure 6.58. Plot Definition Display Properties

Pressing the **Edit** button in the Plot Data property editor will open the dialog shown in Figure 6.59, "Edit Plot Data Dialog". Plot definition graphs are added, removed, and edited in this window. Each row in the table indicates a selected data channel: the columns indicate the graph and set that will contain the channel data, the Data Source from which the channel retrieved, the name of the data channel, and an optional legend entry. The **Graph/Set**, **Data Source**, and **Channel** columns are not directly edited in the table: their contents are automatically determined by the assignment of channels to graphs, outlined below. The **Legend Text** column is editable.

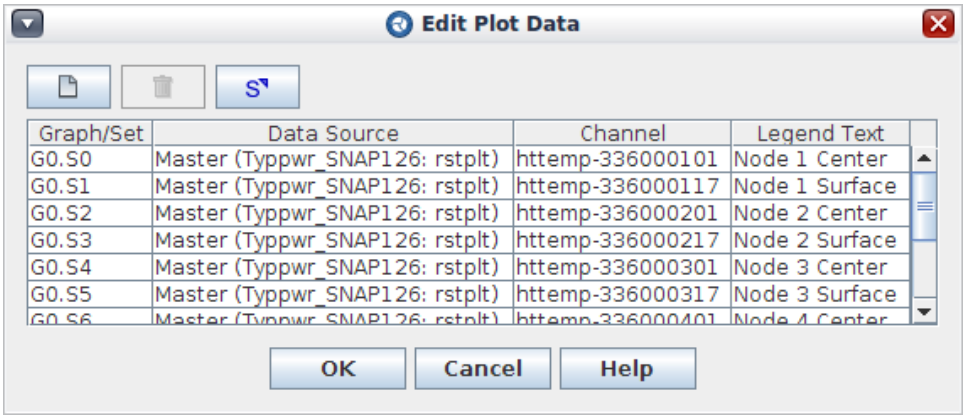


Figure 6.59. Edit Plot Data Dialog

Pressing the **Select** button in the **Edit Plot Data** dialog opens the dialog shown in Figure 6.60, "Select Channels Dialog". Data channels are mapped on a per graph and Data Source basis. The **Graph** list controls which graph is being edited, while the **Data Source** list selects which channels are displayed. Channels can be moved from the **Available** list to the **Selected** list (and back) via the arrow buttons in the center of the dialog: channels in the **Selected** list are plotted.

The channels lists may be filtered with the fields below each list: the asterisk (*) character can be used to match any chunk of text within the filter. For example, the filter **p*** shown in the figure will match any channel name starting with the character p.

The **Select Channels** dialog may also be used to set custom titles and subtitles for individual graphs. To do so, specify the desired title or subtitle in the **Graph Title** and **Graph Subtitle** fields when the appropriate graph is selected.

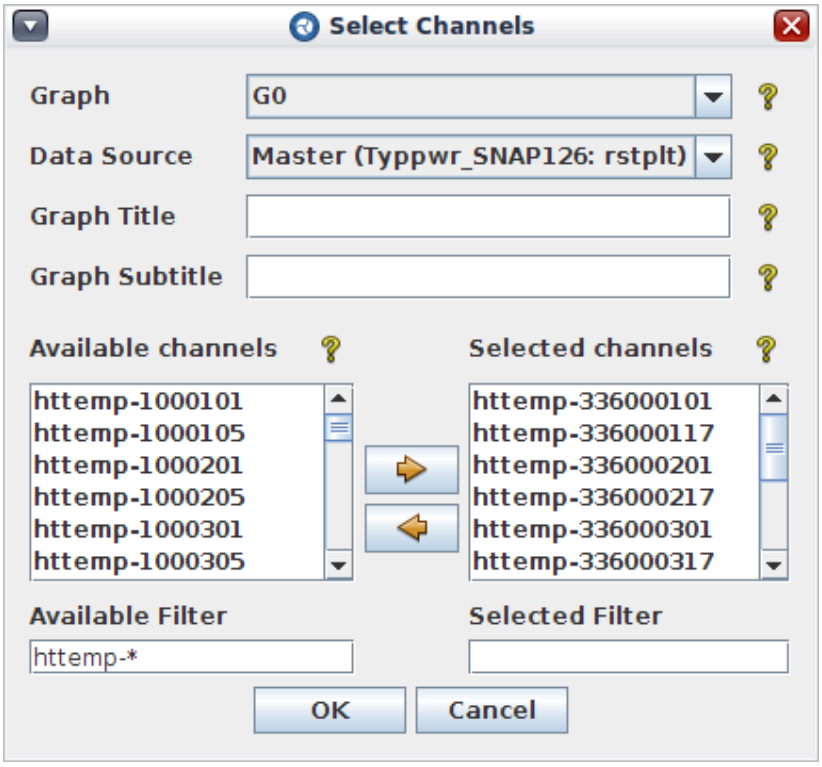


Figure 6.60. Select Channels Dialog

Selecting the **Plot Definitions** category in the Navigator will display the property shown in Figure 6.61, "Plot Definitions Category Properties". In this context, the **Parameter File Location** and **Parameter File** properties function similarly to those described above, specifying an optional parameter file that is used by *all* plot definitions. If a plot definition specifies a parameter file when a category file is defined, both are used: the category file is read first, then the plot definition file. This behavior allows defining general formatting for all plots while allowing plot-specific overrides.

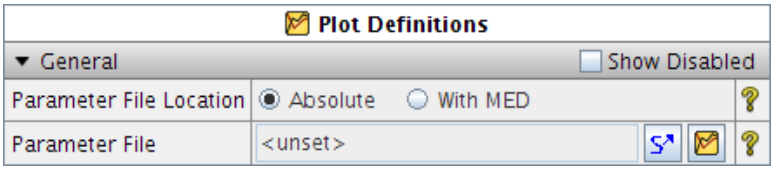


Figure 6.61. Plot Definitions Category Properties

6.9.2. Plot Definitions and AptPlot

Plot definitions utilize AptPlot to plot channel data. The next section describes how to plot data and create plot batch scripts from plot definitions. It should be noted that the following two conditions must hold to perform any action described in this section.

1. AptPlot must be installed and correctly configured in the SNAP configuration (see Section 3.3.1, "General Properties"). AcGrace is not supported, and may not work correctly.

2. A Data Source in the animation model must be connected (see Section 6.7, "Animation Playback").

A plot definition's right-click pop-up menu contains a **Display in AptPlot** item (Figure 6.62, "Plot Definition Right-Click Pop-up Menu"). Selecting this menu item will launch AptPlot and display the plot.

Note The process of launching AptPlot and sending it plot commands may complete much more quickly than AptPlot's actual start-up. When selecting the menu-item, if the menu disappears and nothing seems to happen, wait a few seconds before trying again.

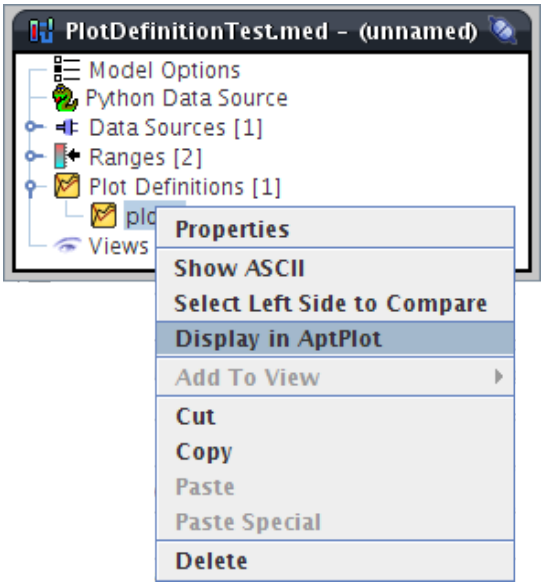


Figure 6.62. Plot Definition Right-Click Pop-up Menu

To see the batch commands used to create a plot, right-click on a plot definition and select **Show ASCII**. This will display the **ASCII View** dialog shown in Figure 6.63, "Plot Definition Batch Commands".

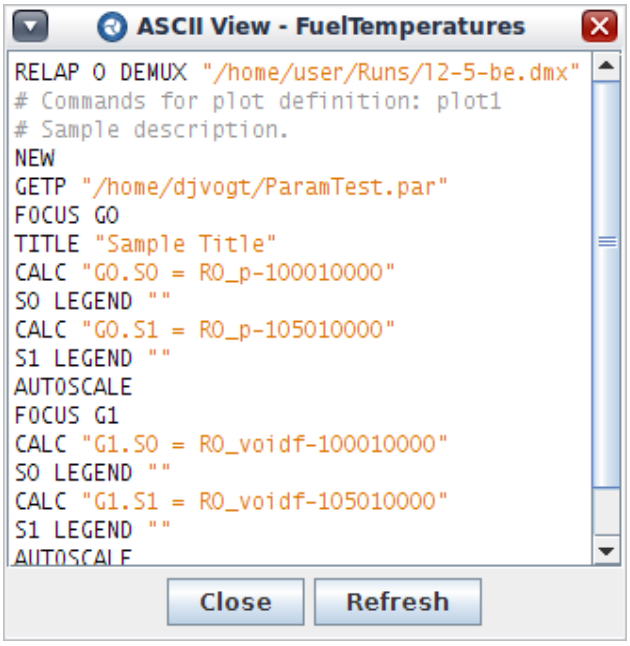


Figure 6.63. Plot Definition Batch Commands

Right-clicking on the **Plot Definitions** category will open a pop-up menu containing the items **Export Plot Images** and **Export Plot Commands** (among others). Both items open the export dialog shown in Figure 6.64, "Plot Definitions Export Dialog". When **Export Plot Images** is chosen, the dialog is used to select a directory where images will be created, the image format, and the plots to be exported as images. Pressing the OK button will launch AptPlot in the background and generate the images. When **Export Plot Commands** is selected, the dialog functions similarly, except that user enters a file where the batch commands for the selected plots are written, but not executed.

Note When exporting images, each plot is named after the plot definition.

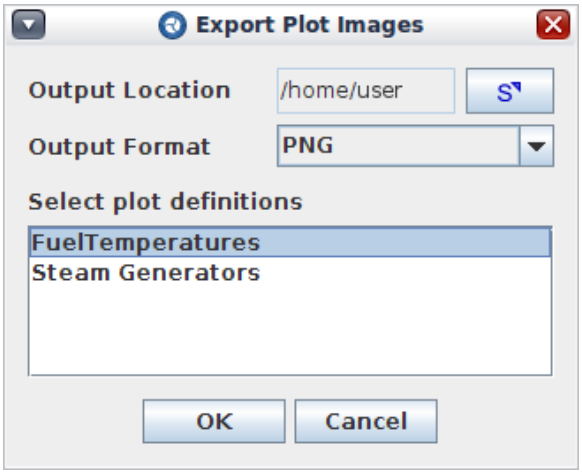


Figure 6.64. Plot Definitions Export Dialog

6.9.3. Plot Definitions in a View

Plot Definitions can be placed in a view to facilitate quick plotting. To do so, either drag the plot definition from the Navigator to the view, or right-click the definition and select **Add to View** > **[Desired View]**. In the view, a plot definition is represented by a single button: pressing this button when the view is locked will launch AptPlot and display the plot. The optional **Display Label** property in a plot definition can be used to specify a label displayed next to the button.

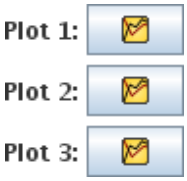


Figure 6.65. Plot Definitions in a View

Chapter 7. Using the jEdit Plug-in

jEdit is a pure-Java, programmer's text editor that was chosen for use with the SNAP system because of its stability and multitude of features. The built-in support for syntax highlighting definitions and extensible plug-in architecture make it a very valuable tool when working with ASCII formatted input files. For a detailed listing of jEdit's available features see the jEdit homepage at http://jedit.sourceforge.net.

7.1. Installing the jEdit Plug-in

The current jEdit plug-in for SNAP is designed to work with jEdit version 4.2 or higher. Older versions of jEdit will not work properly with the SNAP plug-in. To acquire and install jEdit, consult the Download section of the jEdit homepage at http://jedit.sourceforge.net.

Once jEdit is installed, the SNAP plug-in can be installed through the Configuration Tool. In **Personal Settings**, the **jEdit Executable** property defines where jEdit is located. In the property editor, pressing the jEdit configuration button () opens a file browser used to select the jEdit installation directory. For most, this would be:

```
<path to install>/jEdit/4.2/
```

Selecting a file will both set the jEdit location in the SNAP configuration, and install the jEdit SNAP plug-in.

7.2. Using jEdit with SNAP

The jEdit application is used primarily for the **Export to jEdit** item in the **Tools** menu. This item is available in most SNAP plug-ins. This feature is used to export the ASCII input for the selected object (model, component, etc.) to a temporary file which is then opened in jEdit. There, the exported ASCII can be reviewed or copied and pasted into other input files.

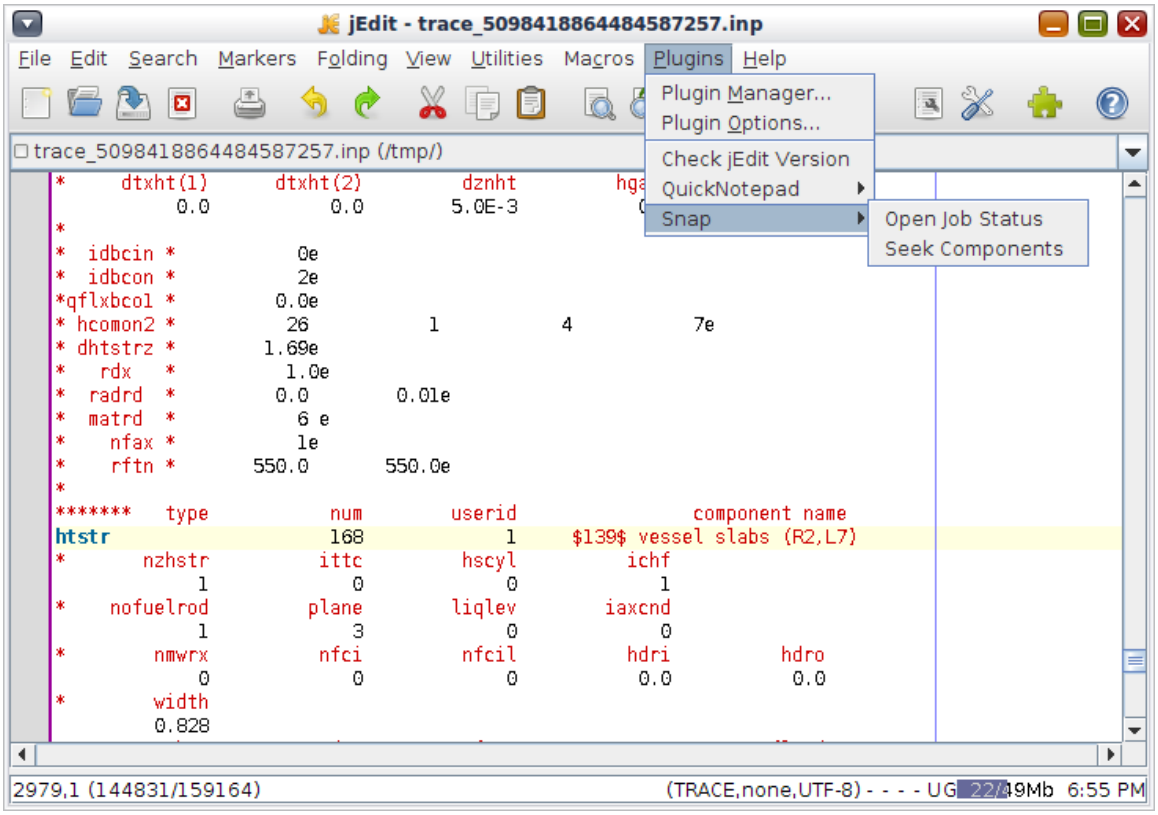


Figure 7.1. The jEdit Editor

The SNAP jEdit Plug-in provides the following additional features to jEdit:

- **Syntax Highlighting** - The jEdit plug-in currently supports syntax highlighting for the TRACE and RELAP5 analysis codes.

- **Seek Components** - The Seek Components dialog (shown in Figure 7.2, "The Seek Components Dialog") is used to locate components within the input deck. The type drop-down contains a list of component types found in the model along with the number of each type located in parentheses. The components for the selected type are displayed in the list. Selecting a component in the list will result in that component being scrolled into view. The Seek Components dialog currently supports the TRACE and RELAP5 analysis codes.

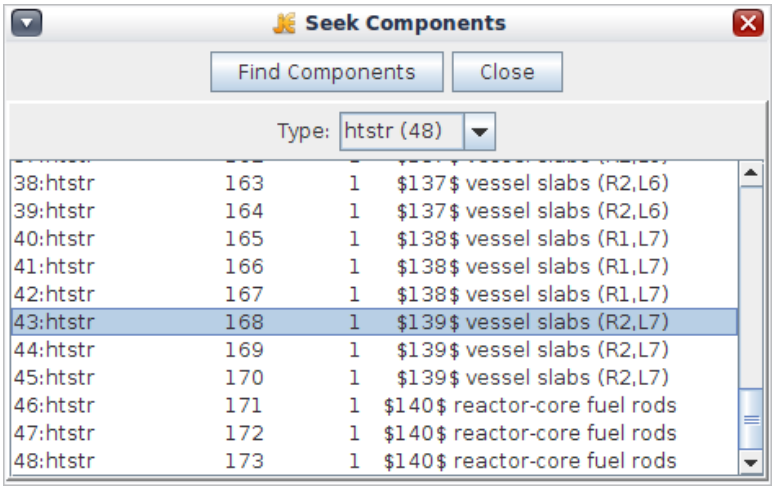


Figure 7.2. The Seek Components Dialog

Index