# User Manual

## Temerity Pipeline™ 2.0.8

Draft 7 - 03.12.2006

Editor's Note: This draft version of the manual is a work in progress and is therefore incomplete and may contain inaccuracies and other minor errors. Please feel free to report any errors you find or any suggestions you may have to the *Bugs & Features* Forum on the Temerity Software web site. Use a the "Documentation" Project and "User Manual" Component when posting to this forum about this document.

# Table of Contents

# Typographical Conventions

A brief note on the writing conventions and approaches used in this manual. Even though the graphical user interface of Temerity Pipeline is fairly straightforward compared to most high level applications familiar to artists in the CG industry, Pipeline can be difficult to explain to new users because it is based upon several highly interrelated core concepts. No direct catalogue of its buttons and menus would serve very well to explain these underlying concepts. Instead, the we have taken the approach of introducing these concepts through a series of practical examples. Often only the most important aspects of a particular feature are explained when first introduced in order to make the workflow of the examples manageable. In later sections, these features will be explored in more detail as users become familiar with the overall landscape of the application.

When it is important to mention a note containing details tangental to the normal flow of the current example, it will be highlighted in a blue box. These notes can safely be ignored on first reading, but may provide a more in depth understanding of the issues related to the thread of discussion in the nearby normal text. Here is an example of one of these notes:

This text would normally explain something of interest related to the nearby paragraphs, but which is not essential to understanding the main text.

There is another type of note encountered sometimes encountered in the text highlighted in red. These notes provide warnings about common mistakes or misconceptions related to the main text. These notes should be read carefully in order to avoid undesirable consequences. A warning note looks like this:

Warnings of this sort should be ignored at your peril. They will contain information which is vital to understanding an important concept related to the nearby text.

This is a left margin note. Normally it would summarize the paragraph to the right.

In general, pages are laid out in a two page side-by-side style where the left page contains the main text of the manual and the right page contains screen shots, tables and diagrams related to this text. You will also encounter notes written in read text in the left and right margins of these pages. The left margin notes provide summaries of the main concepts in the nearby text, while the right margin notes describe the contents of the nearby images or diagrams.

When mentioning the literal names of files, nodes or other string data related to Pipeline, the text will be enclosed in parentheses such as (foo.0123.tif). When some portion of this text can be many different values, italic text will be used to delineated the portions which are literal from those which are variable. For example, preferences are stored in the directory (*home-dir*/.pipeline) where the first portion of the name denotes the path to the users home directory.

The three mouse buttons (left, middle and right) may sometimes be referred to as (Mouse1), (Mouse2) and (Mouse3) respectively. When describing the use of these buttons in combination with keyboard modifier keys, all keys being pressed may be listed in parentheses separated by addition signs. For instance, pressing the left and middle mouse buttons while holding down the (Shift) and (Alt) modified keys on the keyboard would be described as (Shift+Alt+Mouse1+Mouse2).

# Revision Control Concepts

This section will cover some basic concepts necessary for understanding Pipeline's unique revision control system. Those readers who are familiar with the concepts of revision control and directed node-based tree data structures may wish to just skim this section.

Revision control provides a way to manage all changes made to an asset.

Revision control is the process of maintaining multiple successively edited versions of some data, called an asset, in a centralized repository. Each asset is identified by a unique name and the different revisions of an asset are specified using a revision number. Other useful information is also recorded about each version of an asset such as the date that it was added to the repository, the name of the user who created it and a brief message by this user explaining the modifications to the asset since the previous version.

The act of creating a new version of an asset in the repository is called a "check-in". Conversely, the act of obtaining a copy of an asset from the repository which can possibly be modified is called a "check-out". This naming is derived from the idea of how a people interact with a library when they want to obtain a particular book. However, the library metaphor is a little misleading in the case of Pipeline where there can be many checked-out copies of the same asset. With Pipeline, checking-out an asset is more like getting a photocopy of a book from the library than the actual book itself.

All permanent versions of an asset are stored in the central repository. Users make all changes in a private sandbox called a working area.

When a user checks-out particular version of an asset in Pipeline, a copy of the asset is placed in one of their working areas. A working area can be thought of as a private sandbox where a user can make changes to the asset without having to worry if those changes will other users. When the user has reached a state where they are happy with changes they have made to an asset in their working area, they can check-in these changes to create a new version of the asset the repository. Once a new version has been created, it becomes visible all other users who may then check-out the newly created version of the asset to use or modify in their own working areas. In this way, the repository acts as the mediator in all assets exchanges between different users. Each user only works in their own working areas and must go through the repository in order to access assets created or modified by other users.

Working areas are represented by Pipeline using seperate directory structures which are writable only by the owner of the working area. The current working area is stored in an environmental variable called ($WORKING). This variable is useful for a couple of reasons. It provides a convienent shortcut for navigating to the current working area when working from the shell or in file browsers of other software. But more importantly, it provides a mechanism for making files used with Pipeline truly portable between different user's working areas. Instead of using absolute directory paths, the ($WORKING) variable can be used to specify file locations. When data is then moved between different working areas, this path will point to the correct location within the current working area.

Versions are identified by unique revision numbers.

Pipeline uses three part version numbers of the form (*major.minor.micro*). Each place in the number corresponds to the significance of the change in an asset from the previous version. See the table of the right for some examples. This level of change is purely a human heuristic used to indicate to other users how much you modified the node. Pipeline only distinguishes the order that versions were created in based on the revision number, but ignores the significance of the difference between revision numbers. In order for this aspect of revision numbers to be practically useful, users must adopt some shared conventions about what qualifies as a major, minor or micro level change.

Editor's Note: Put a diagram here which helps explain relationship of working areas and the repository and the check-in/check-out operations in an abstract manner.

| Revision Level | Before Check-In | After Check-In |
|---|---|---|
| Major | 1.2.3 | 2.0.0 |
| Minor | 1.2.3 | 1.3.0 |
| Micro | 1.2.3 | 1.2.4 |

Examples of how the significance of a change is represented by the revision numbers of checked-in versions of an asset.

For example, a Micro change could be something which does not affect the behavior of the scene in any way, like changing the names of objects within a scene for organizational purposes. A Minor change might be something like tweaking some vertices of a model, smoothing some animation curves, or making subtle adjustments to shader parameters. Large changes to the rigging of a character, significant amounts of reanimation or other far reaching modifications could be considered a Major change. The meaning of these terms should established by each studio to suit their particular needs and conventions.

Pipeline uses nodes to represent a production process and the file assets created by this process.

Traditional revision control systems normally manage simple assets such as single data files. Some systems also associate metadata (data about data) properties with these files to store other useful information. Pipeline takes this concept a step farther by managing not just data, but also proceduralism. Assets in Pipeline are called nodes. A node is associated with a collection of related data files and all essential information about how these files where created. This includes any dependencies on other nodes upstream in the production process.

For example, a node representing a sequence of rendered images might depend on a node representing the Maya scene being rendered and nodes for all of the texture maps required to perform the render. So in this example a version of the rendered images node would include the images themselves, how to perform the rendering and a list of all of the nodes required to perform the renderer. These nodes may in turn depend on other nodes, so that a version actually represents a tree-like structure of nodes rooted at a node which represents the product of some production process like rendering.

In Pipeline, all production processes are represented by one of these tree-like hierarchical structures composed of series of linked nodes. The most downstream node (the result of the process) in one of these node trees is called the root node. The most upstream nodes, which have no dependencies on other nodes, are called leaf nodes. The intermediate nodes between the root and leaf nodes are called branch nodes. Leaf nodes tend to be associated with human created assets such as painted texture maps, models and key-framed animation data. The root nodes represent the final products of a production such as final rendered images or composited image sequences. The branch nodes typically represent some automated intermediate process such as running simulations (particle, cloth or other dynamics), automated rigging, dynamic scene building, RIB processing and other steps in the production process.

When you check-in a node, the state of the entire tree of nodes upstream of the node being checked-in is saved as part of the newly created node version. When a node version is later check-out, this operation also restores all of these nodes and their associated files in exactly the same state as when they where checked-in. When you wish to restore a particular version of a node, you are not just restoring that node, but the entire process that generated it.

The ability of Pipeline to save and restore entire production processes is important for several reasons. By recording all of the exact details of how the production actually works, artists are able to clearly understand their relationship to the rest of the production and better coordinate their efforts. Supervisors and producers are provided with essential information about the ramifications for the entire production of a making changes to shared assets. It also greatly simplifies tracking down the particular asset which needs to the modified in order to accomplish a requested creative change and to identify the artists who are responsible for making these changes. Finally, whenever a change is made to a leaf asset, such as repainting a texture map or adjusting animation curves, all of the steps required to recreate the final rendered or composited images are well understood and automated to the greatest possible extent.

Editor's Note: Put a diagram here which shows a symbolic node tree data structure without any node symbols or other complicating UI graphics.

# Basic Layout Operations

The layout and arrangement of windows and panels can be customized by each user to suit their needs.

When you first open start Pipeline, you will see an empty panel. The initial step in using the program is creating a useful layout. A layout in Pipeline is a collection of one or more windows, the panels in those windows and the nodes displayed in those panels. The initial window displayed by Pipeline, called the main window, is special and can be identified by the progress message bar located along the bottom edge of the window. Closing the main window will cause Pipeline to exit. This section describes the tools provided by Pipeline for creating, customizing and saving these panel layouts.

The empty window to the right contains the basic elements common to every Pipeline panel. In the upper left hand corner is the Main Menu button. This menu contains controls for the layout related commands, user preferences, and all the administrative tools. Next to the menu button is the Channel button, which determines how the panel is linked to other panels. In the center is the Working Area title bar which identifies the current user and working area being displayed by the panel. In this case, the (default) working area of the user (jesse). In the upper right-hand corner of the panel, there is an X-shaped button which can be used to close the panel.

There are three different ways you can modify a layout: adding tabs, splitting panels, and creating new windows. The controls for all of these are located in the main menu. Choosing New Window will create a new top-level window which can be moved around independently of the other Pipeline windows, minimized, maximized, or moved to separate desktops. You might prefer having all your node related panels in one window and all your queue related panels in a separate window.

The arrangement of panels within windows can be specified using Tabs and Spits.

Tabs and split panels provide a way to organize the contents of a window and can be nested inside each other. Tabs make it possible to quickly switch between alternate sets of panels which use the same screen real estate. You can also split panels both horizontally and vertically to subdivide an area of a Pipeline window between different types of panels. For example, choosing Add Left will create a vertical split with the current panel occupying the right and the newly created panel on the left. Tabs and splits can be nested inside each other. You can add tabs, split one the tabs vertically, and then add more tabs to one the split panels.

Layouts for different kinds of tasks or for working on specific shots can be saved and restored.

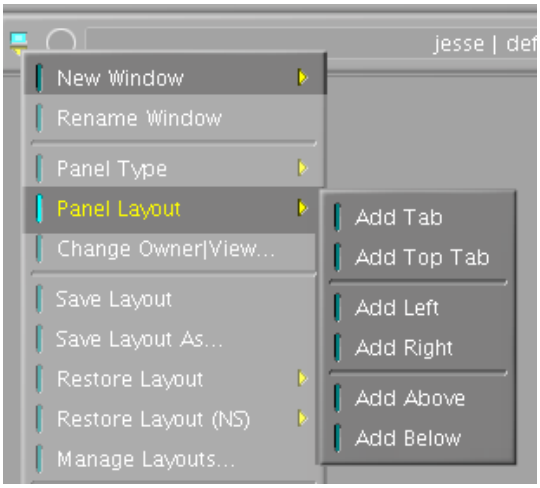
Slightly further down on the main menu are the items that allow you to save, restore, and manage layouts. The Save Layout item will save the current layout of windows and panels replacing any previously stored version of the current named layout. The name of current layout is displayed in the title bar of the main Pipeline window. The Save Layout operation is useful if you are tweaking the current layout and wish to just quickly save your changes. To create a new layout, use the Save Layout As item. You will be prompted for a new name for your layout.

Be careful when using Save Layout. Since it saves over the current layout, you can accidentally overwrite a saved layout. Make sure to use Save As to create a new layout.

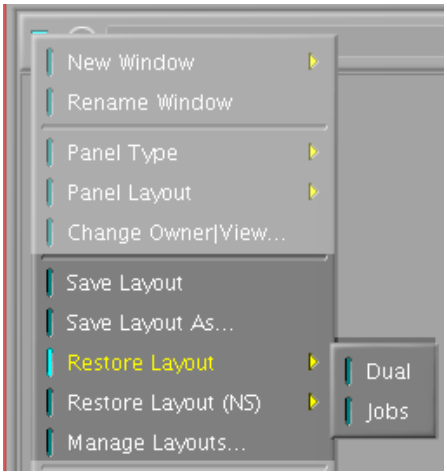
The two Restore Layout options will cause your current layout to be replaced by a previously saved layout you specify. In addition to saving the arrangement and positions of panels, a layout also remembers the selected nodes and jobs at the time the layout was saved. The Restore Layout (NS) option will ignore any saved node or job selections when restoring the layout. This can be useful when you want to restore the arrangement of panels contained in a layout but are not concerned with any node or job selections which may have been saved with the layout.


The default Pipeline window containing a single empty panel. The standard controls common to all panels are highlighted at the top. They are from left to right: the Main Menu button, the Channel button, the Working Area title bar and he Close Panel button.

Progress messages are displayed in the main window in the high-lighted area near the bottom of this window.




The menu items of the Main Menu controlling new window creation and panel layout within windows.




The Main Menu items controlling saving and restoring saved panel layouts. The Restore Layout and Restore Layout (NS) submenus display lists containing the names of all previously saved layouts.

Finally, the Manage Layouts item displays a dialog that allows you to edit your existing layouts. There are three actions you can take here. Rename allows you to change the name of the currently selected layout (highlighted in yellow). Delete removes the currently selected layout. Default will make the selected layout the initial layout displayed the next time Pipeline is restarted.

One more option related to layouts can be found in the Preferences dialog. The Save Layout On Exit option can be enabled, which will save your layout whenever you exit Pipeline. When you restart the program, that layout will be restored, allowing you to continue working where you left off.

## Panel Types

There are nine different panel types in Pipeline, six of which are related to nodes and three of which are related to the queue and jobs.

There are nine different types of panels in Pipeline which we will discuss briefly. Each type of panel will be explained in detail in later sections. This section will only introduce the various types of panels and give an overview of how they relate to each other.

The Node Browser displays a directory tree representing all the nodes managed by Pipeline. When you wish to work on or view a node you select it in the Node Browser. These selected nodes are displayed in the Node Viewer. This window shows all selected nodes, their dependencies and a graphical representation of their revision control and queue status. The Node Viewer panel is where a majority of the user interaction takes place. There are four panels which provide extra information about a node, collectively referred to as the detail panels. The Node Details panel shows all the properties associated with a node and provides ways of comparing these properties with those associated with any repository version of a node. The Node Files panel provides information about the status of the individual files associated with a node, as well as showing the detailed history of changes to these files associated with all the repository versions of the node. The Node Links panel displays information about dependencies on other nodes upstream in the production process, including the specific versions of these nodes. The Node History panel lists all the versions of a node stored in the repository showing the date and time when these versions were created, the user who created them, and a brief check-in message explaining the nature of the changes contained in the node version.

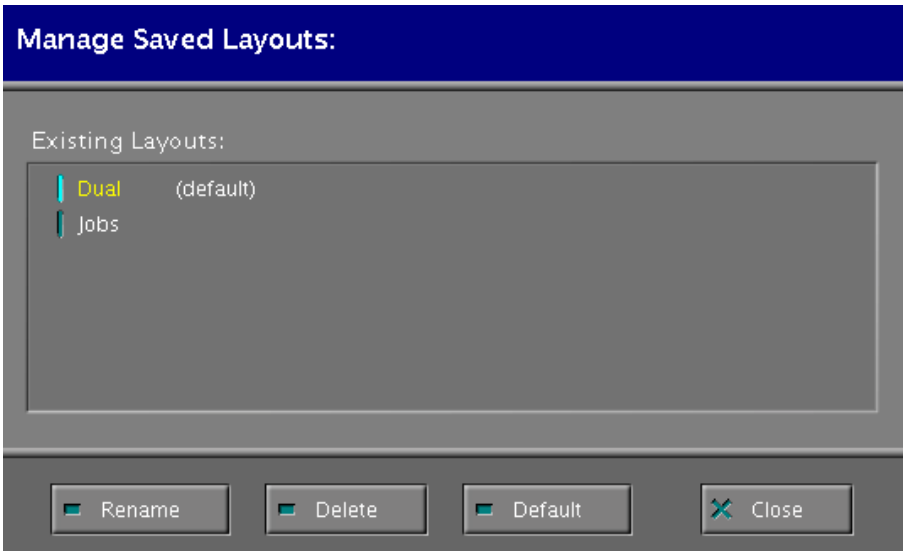
The Job Browser panel allows you to monitor the state of all the machines participating in the distributed job queue. This panel also contains tabs for managing all of the job groups being processed by the queue. A job group is a collection of interdependent jobs submitted together to accomplish a specific goal. Job groups selected in the Job Browser are displayed in the Job Viewer panel. This panel displays the state of individual jobs and their dependant jobs and provides controls for managing specific jobs.. The Job Details panel displays execution time, resource usage and all progress and error messages generated for a selected in the Job Viewer.

There are two items in the Main Menu which allow you to adjust panel type. The New Window menu item creates a new window containing a single panel of the selected type. The Panel Type item will replace the current panel with a different panel of the given type.

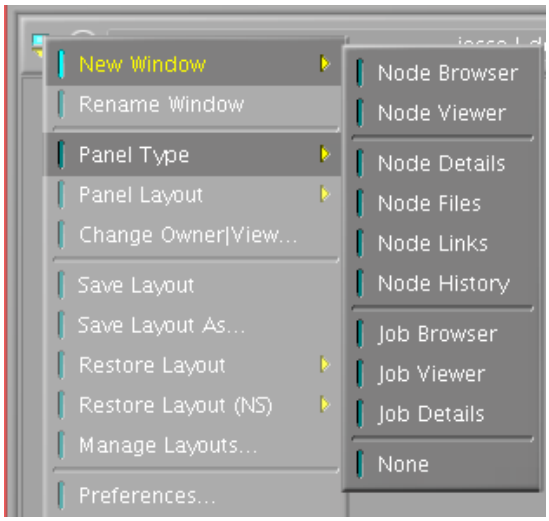
## Update Channels

Linking is a way for Pipeline to have multiple panels of the same type but make sure that commands are passed to the correct ones by putting them on separate channels.

It is possible for a layout to contain multiple panels of the same type. Since panels are designed to be used together in combination with panel of other types, Pipeline needs a way of specifying how information flows from one panel to another. For this reason, each panel communicates on a specific channel to all other panels which share that channel number. Each panel has a small circle in the upper left hand corner which controls its channel selection. Pressing this button, displays a menu which allows the channel for a panel to be changed.




The Manage Layout dialog.




The New Window menu option, showing a list of all the different panel types that can be created. The Panel Type option will display the same list.



The panel Channel menu and a panel with channel (1) selected.

You can only have one panel of a particular type on a channel.

An empty circle means that the panel is not linked to any other panels while a number indicates the selected channel. All panels on the same channel communicate with all other panels on the same channel. For example, a node selected in the Node Browser it will show up in the Node Viewer sharing the same channel. If no Node Viewer shares the channel, then nothing will happen.

Given the channels are used by Pipeline to direct the flow of information, there can only be one panel of each type on a channel. When you create a new panel from an existing panel, it will attempt to use the same channel. However, if a panel of the same type already exists on that channel, then the new panel will not be assigned any channel. When you attempt to switch the channel of an existing panel, any channel with already has a panel of that type will be grayed out.

You should be aware that linking too many panels on the same channel can decrease the responsiveness of the user interface. Every time you perform a status update Pipeline has to acquire status information for every linked panel. If your Job and Node panels are linked together, updating the node status requires you to wait for the queue status as well. It is usually a good idea to keep those panels on separate channels allowing everything to move as quickly as possible.

## Common Panel Layouts

The composition of the ideal layout is up to the artist, but these are some suggestions towards building an efficient layout.

What makes a good layout is really up to individual artists to determine, but the next section will be a brief look at several layout types to provide some inspiration.

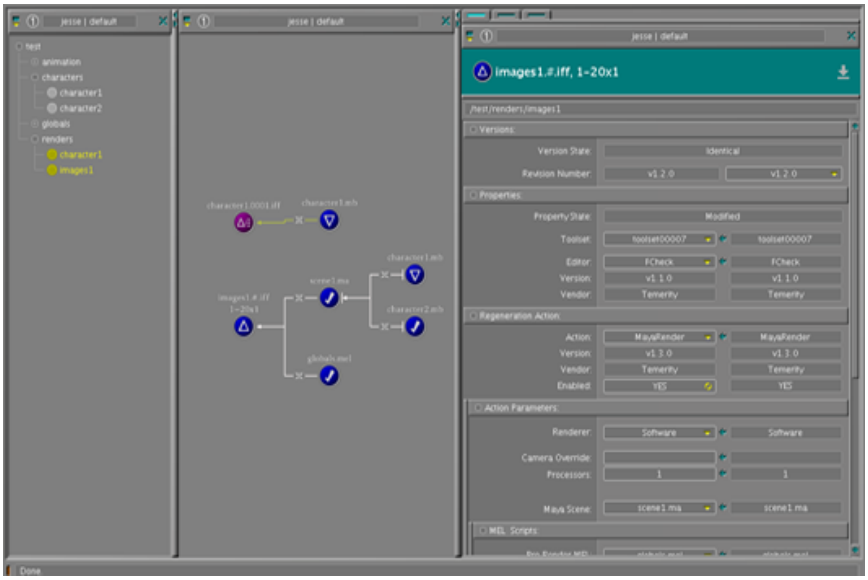
Most layouts feature a side-by-side Node Browser and Node Viewer (linked of course). The position of the Details panels usually differs depending on personal preference, however most layout s arrange them together in a series of tabs. The two most common locations to place these node detail panels is either in a split panel to the right of the Node Viewer or in a separate window which floats in the background. Having them next to the Viewer panel is convenient, but restricts the size of the Node Viewer–a drawback when viewing complicated node trees. Placing them in their own window permits more space for the Viewer, but makes it less convenient to interact with the Details panels. Some users prefer to simply create a new window for a specific node details panel whenever they want to view or edit this information and then close this window when done. There are customizable hot-keys for creating new windows and changing panels which make this convenient.

It is fairly common to omit the Node Links panel from a saved layout. This is due to the fact that the Links panel requires a significantly longer update time, given the amount of information it must acquire from the database to display. Also, the information included in this panel usually is not necessary very often and is only useful when trying to debug the details of how links have changed between node versions.

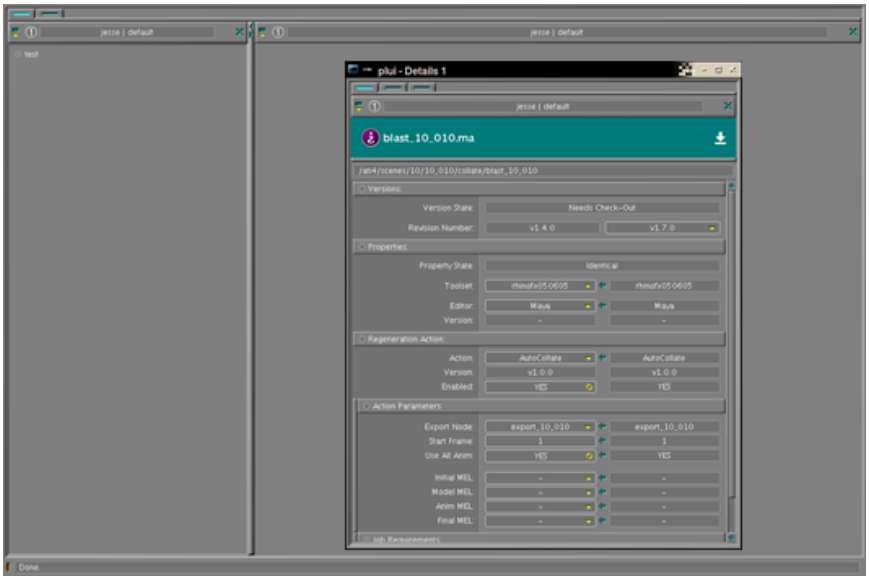
The job panels are usually placed together, either in their own window or in a top-level tab behind the node panels. A common arrangement for these panels places the Job Browser on top, while the Job Viewer and the Job Details share the space underneath.

## Naming Windows

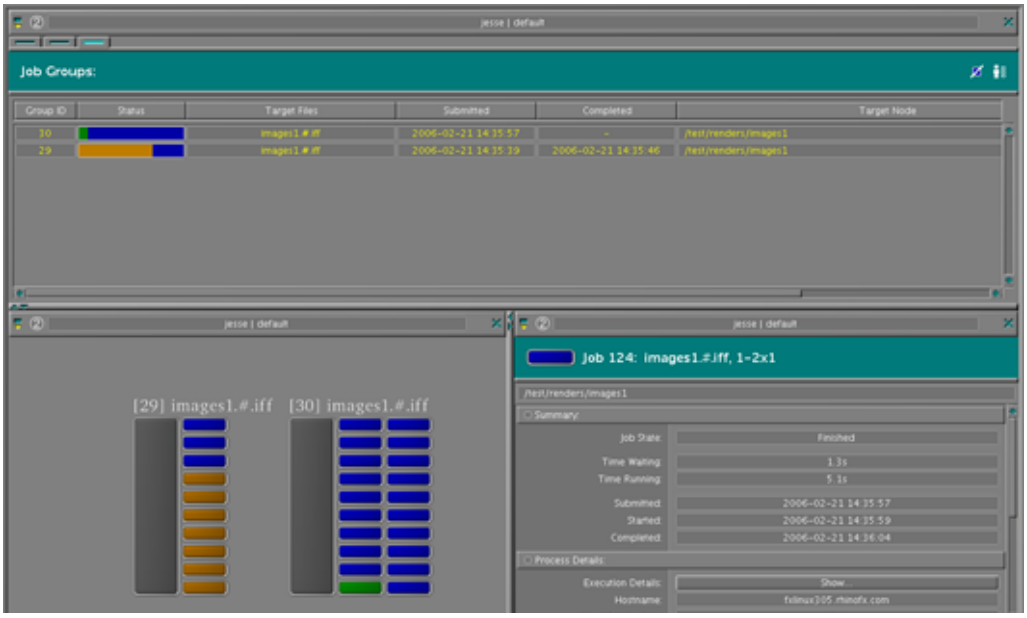
If multiple windows are used, it may be desirable to name each window something different, so that it is easy to select the correct window from a taskbar. To name a window, select the Rename Window menu item from the Main Menu in the window that you wish to rename. Type in a name for that window and press the Apply button. Window names are saved along with layouts.




The main window is split into three. The Node Browser is on the far left with the Node Viewer in the middle. The Details panels are arranged in tabs on the right.




The main window is split into two. The Node Browser is on the far left with the Node Viewer on the right. The Details panels are placed in tabs in a separate window floating above.




The Job Browser panel is spread across the top of the window, with the bottom part split between the Job Viewer on the left and the Job Details panel on the right.

# Basic Node Operations

New nodes are created by registering a file or sequence of files with Pipeline.

In order to add a node into the Pipeline system, it is necessary to register the primary sequence of files associated with the node. In this example, we are going to register a single Maya file ($WORKING/test/characters/character1.mb). The first step is to create the file and save it in an appropriate location under the root working area directory ($WORKING). Once that is done, press and hold down (mouse3) over an empty area of the Node Browser and select the Register item. This will display the Register New Node dialog.

Press the Browse button at the bottom of the dialog, which will open up the Select File Sequence dialog. This allows you to browse through the directory structure of your working area and select a file (or file sequence) to associate with a new node. In our case, we browse down through the (test) and (character) directory until we see the (character1.mb) file. Select this file and then press the Select button at the bottom of the dialog.

Pipeline will fill in all the appropriate fields in the Register dialog, setting the node name and extension based on the name of the file. It will also choose an initial Editor plugin for the node, based on the extension. Verify that everything that Pipeline has filled in looks correct and press the Confirm button to register the new node.

Pipeline guesses the appropriate Editor for your node based on the file's extension and the default Editor for that extension set in the Default Editor dialog. When you register a sequence of images, Pipeline will fill in the frame range and padding information for you as well.

Node icons provide information about how the working version of the node compares with the latest version of the node in the repository.

You should now see your new node displayed in the Node Browser. The name of the primary file sequence (character1.mb) is displayed above the node. The node icon itself provides basic information about the status of the node in the Pipeline system. In this case, the Blue color of the node icon means that the newly created node has a Queue state of Finished. This means that no jobs need to be run to regenerate the files associated with the node. For this particular node this is because it does not have an Action plugin assigned to regenerate its files. Actions and their relationship to jobs and node's Queue state will be explained in detail later.

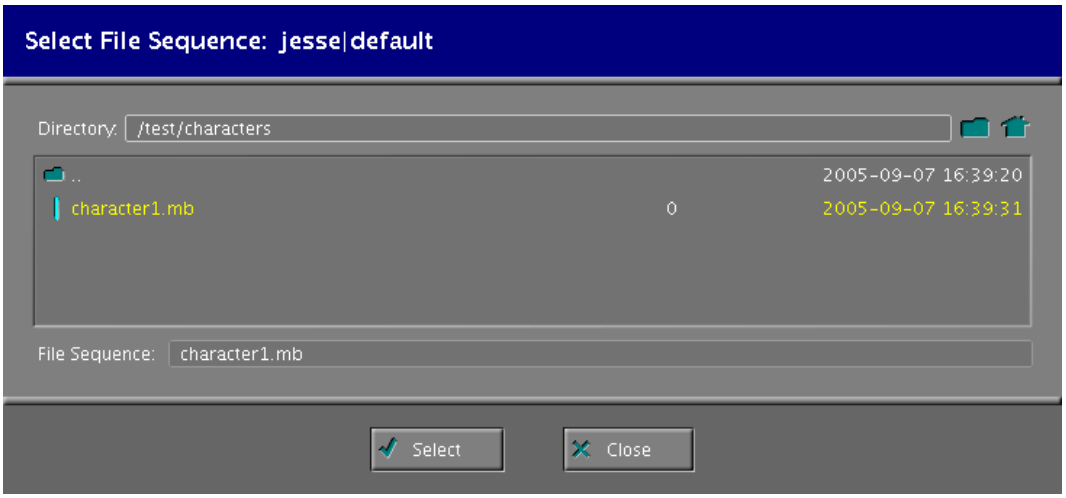
The information displayed above the node can be customized using the Preferences dialog. Options include: no label, node name only, primary file sequence pattern or sequence pattern and frame range informations.

The arrow symbol pointing to the left for our example node means that the node has never been checked-in. This state is called Pending. When a node is checked-in, a new version is created in the repository, which becomes visible to other users. However, until that happens the node only exists in your working area.

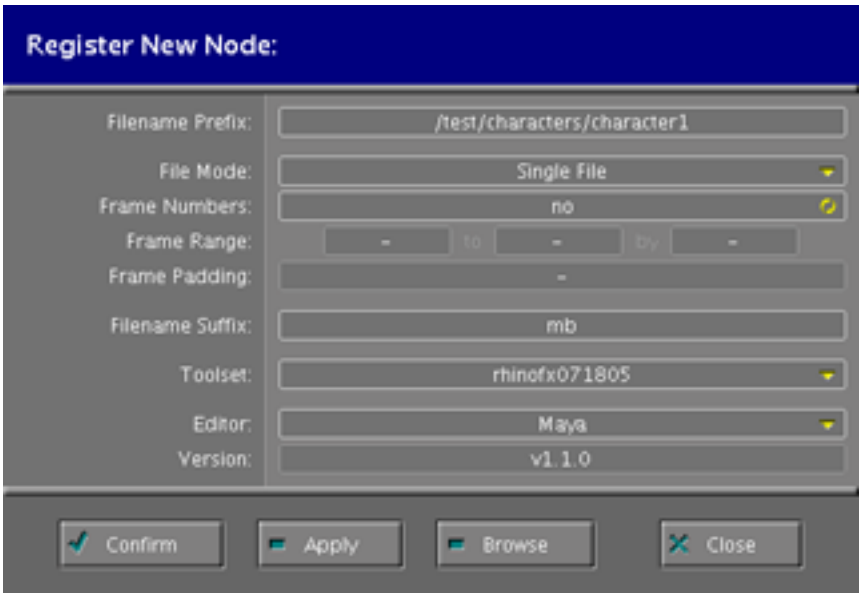
It is important to remember that Pending nodes have not yet been preserved in the repository. If you delete the files associated with a Pending node, they are gone forever since there is no way to restore them from the repository.

The Check-In operation copies any modifications that a user has made to a node in their working area into the repository, creating a new version of the node.

With that in mind, lets go ahead and check in the newly created node now. Press and hold (mouse3) over the node and select the Check-In menu item from the Node Menu. This will display the Check-In dialog which allows you to specify a version number which corresponds to the significance of you Check-In and a brief message explaining the nature of these changes. In this case the New Version is disabled and does not allow you to make any choices except (v1.0.0) for the initial version. The Check-In message provides other users with a summary of what changes you made to the node

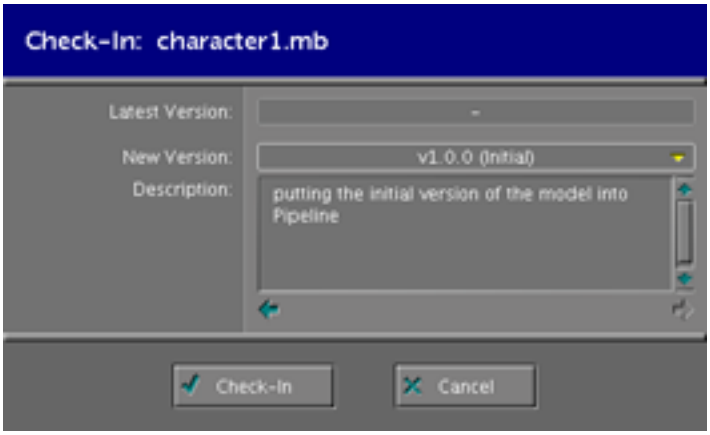



The Select File Sequence dialog box, showing the file (character1.mb)




The Register New Node Dialog after (character1.mb) was selected. You can see that Pipeline has made a guess as to the correct editor to assign to this node, as well as filled in the all the fields related to the file name.


The (character1.mb) node showing the left-pointing arrow, indicating that the node is Pending.




The Check-In dialog. Note that is says Initial in the New Version Option Box and that there is no Latest Version displayed. This is because the node has never been checked-in.

without having to inspect any of the associated files. In this case, since it's the initial version, there are no changes to record so a simple message such as "initial version" will suffice. Enter the message and press the Confirm button.

The arrow buttons at the bottom of the Check-In dialog allow you to scroll back and forth through all previous check-in messages created in the current Pipeline session. Press the left-pointing arrow to go back to the previous message and the right arrow to go forward. The first message in the list is always "initial version" which is inserted automatically by Pipeline. It is often quicker when creating the initial version of a node to just use the left arrow to select this message than to type in your own initial message.

Pipeline will create an initial version of the node in the repository and then update the status of the Node Viewer panel and any other panels sharing the same channel. You'll notice that the symbol displayed by the node has changed to a check mark, which indicates the Identical node state. The Identical state means the version of the node in your working area is identical to the latest version in the repository.

Lets say we now want to make some changes to the file associated with our example node. Choose the Edit menu item from the Node Menu. This will cause the primary file associated with the node to be opened using the Editor plugin you selected when registering the node. In this case the Maya plugin is used. For the purpose of demonstrating Pipeline's revision control features, go ahead and make some tweaks to the model, save the scene, and quit Maya. Going back to Pipeline, press and quickly release the Spacebar in the Node Browser (or Node Viewer) to trigger a status update. After the update completes, you will notice that the node icon has changed into a delta symbol (a right-side-up triangle). This symbol represents the Modified state, meaning that the version in your working area is based on the latest repository version but has been locally modified.

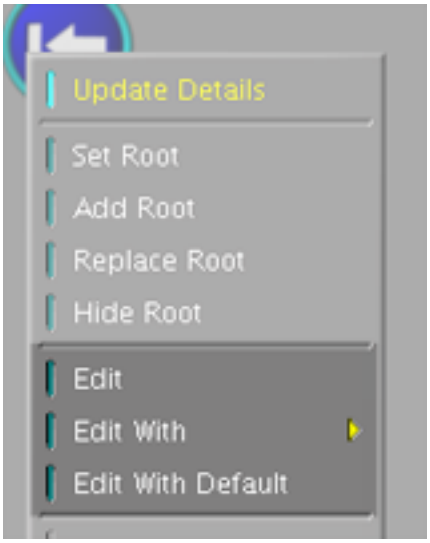
There are three menu items on the Node Menu that relate to editing the files associated with a node.. The Edit item uses the Editor plugin associated with the current working version of the node to edit the files. You can change the Editor plugin associated with a node in the Node Details panel. The Edit With option will display a submenu containing all the available Editor plugins provided by the node's current Toolset. This makes it possible to select an Editor plugin other than the default for some temporary purpose without needing to modify the node's Editor property. For example, the Terminal editor launches a shell with the current directory set to the the location where the files associated with the node reside. There is no particular reason to ever have this plugin as the assigned Editor for a node, but it is often very useful. The Edit With Default options item uses filename extension of the node's primary files to select an Editor plugin. This choice of Editor plugin can be customized using the Default Editor dialog reachable from the Main Menu. This is a useful shortcut when the assigned Editor for the node is different from the default Editor.

The Enter key is the default hotkey for Edit. Just make sure the mouse pointer is positioned over the node you wish to edit and press the Enter key.

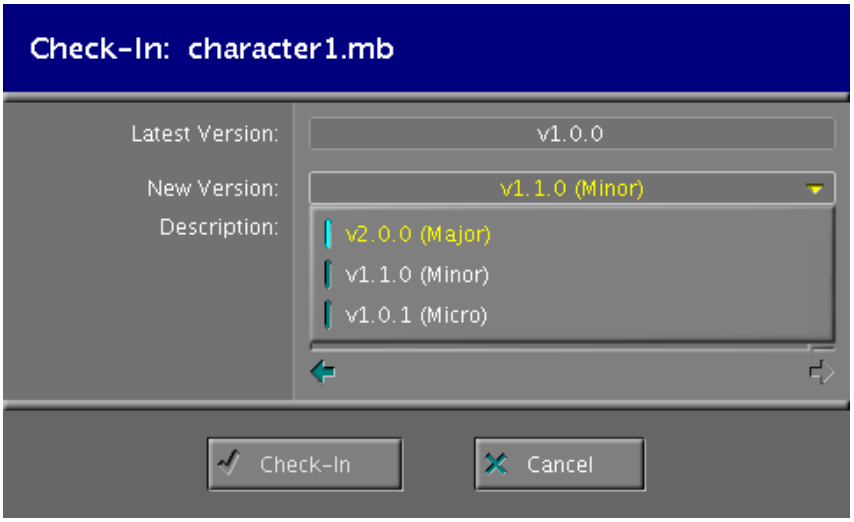
If you are satisfied with these modifications, then you will want to check them in. This will not only version control your changes, but will also allow anyone else working on the project to view them. Again, choose the Check-In menu item. This time the New Version option box is enabled and provides three choices: Major, Minor, and Micro. These indicate to other people how significant the changes to the node are compared to the previous version. In this case, the changes made were pretty insignificant, so we will use the Micro option. As you can see, this will cause the version to increment to 1.0.1. A reasonable message might be something like, "Made some small changes to the model geometry."


The check mark symbol indicates that the working version is identical to the latest repository version.


The delta symbol represents the Modified state, indicating the changes have been made to the node which have not yet been checked-in.




The Node menu, with the three Edit menu items.




The New Version option box is now enabled. The Latest Version line displays the newest repository version.

# Links and Actions

Actions are used to regenerate the files associated with nodes in Pipeline representing automated steps in the production process.

We will now demonstrate some of the essential workflow management features of Pipeline by linking two nodes and using an Action plugin to generate the files associated with the downstream node. This time, instead of registering existing files as a node, we are going to register a node and let Pipeline generate the files for us. Select the Register item again from the Panel Menu to display the Register New Node dialog. The dialog should still contain all the information from the last node you created. Change the name of the node from (/test/characters/character1) to (/test/renders/render1). Set the Frame Numbers option to YES. Set File Mode to File Sequence and the frame range to 1 to 10 by 1. Make sure that the Filename Suffix is set to (iff) and the Editor is set to (FCheck) and then press Confirm. This time the symbol displayed by the newly created node the node icon is a question mark. This symbol indicates the Missing node state which means that one or more files associated with the node is missing from the current working area.

The question mark symbol will appear whenever a file is missing from a node. If the node represents a file sequence, the Missing icon will be displayed even if only one of the files is missing. If the node has secondary sequences, the question mark will show up if any of the secondary sequence files are missing. It is important to remember that the Missing state does not mean all the files are missing. In order to determine which files are actually missing, the Node Files panel can be consulted.

## Node Links

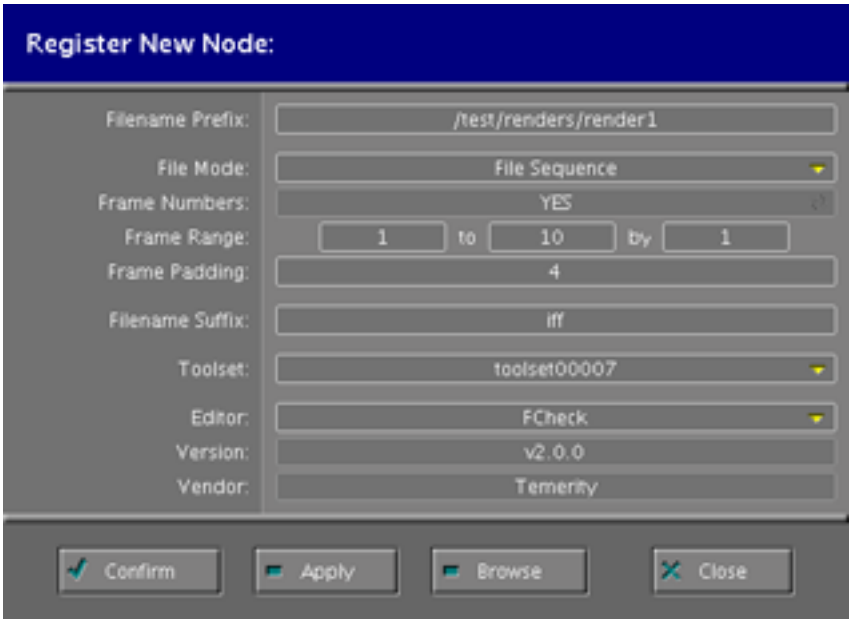
Links are used to specify the dependencies between the files associated with nodes.

We are now going to create a link between (character1) and (render1). Make sure that both nodes are visible in the Node Viewer. If they are not, go to the Node Browser, hold down the shift key, and select both nodes. Pipeline will wait until you release the shift key before updating the Node Browser and Node Viewer panels to display the selected nodes.

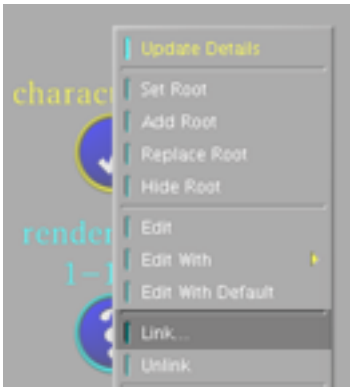
In the image on the right, you can see that the node names and borders have been highlighted. When you select a node or a group of nodes in the Node Viewer, they are highlighted in yellow. These selected nodes will become the source node for any Pipeline operation which requires both source and target nodes such as Link. In this example, the source node is being linked to the target node. If you had more source nodes selected, they would all be linked to the target node.

Once both nodes are displayed by the Node Viewer, select the (character1) node and then press and hold the right mouse button over the (render1) node to show the Node Menu. Select the Link item from the menu to bring up the Link dialog. This dialog allows you to specify the properties of the link. In this case the defaults are fine. Press the Add button to create the link.

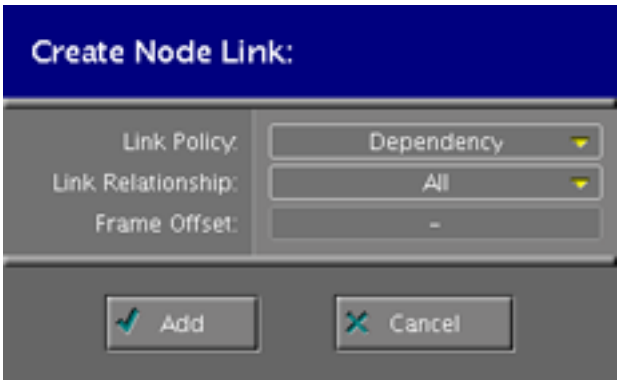
What do these default options mean? Link Policy specifies how the source node influences the regeneration of the target node. In this case, a Dependency policy means that the (render1) node will need to be regenerated every time that (character1) is changed. In this situation, this is exactly what we want. The Link Relationship setting is only meaningful in situations where a node associated with a multiple file sequence is being linked to a node which is also associated with a multiple file sequence. Since this example only involves single files, the Link Relationship setting does not particularly matter.




The Register dialog for the (render1) node.




Selecting the Link item from the Node Menu. This menu appears when you press the right mouse button over the target node.




The Create Node Link dialog. The default options are fine for now. The other options will be discussed in the next section.

The (character1) node should now be connected to the (render1) node by a white arrow. This arrow indicates that there is a link between the (character1) and (render1) nodes and that the (render1) files depends upon (character1.mb) in some way. This means that a change to (character1) will require that the (render1) images be regenerated to reflect these changes. However, since the (render1) node currently does not have an assigned Action, Pipeline does not yet know how to regenerate the file.

## Regeneration Actions

Actions are assigned to nodes and tell Pipeline how to regenerate the files associated with that node.

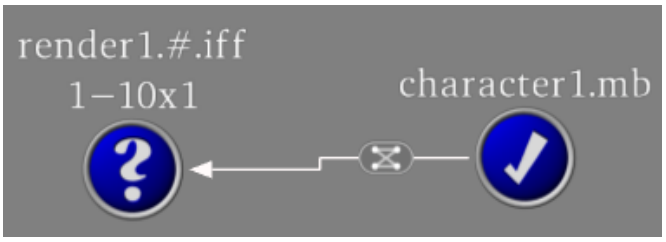
We are going to add an Action to the (render1) node which will make use of the newly created link. Select Update Details from the Node Menu for the (render1) node. This will cause that node's properties to be displayed by any linked details panels. See the Update Channels section for more information. For the sake of this exercise, make sure that you have Node Details panel displayed which shares the same channel as your Node Viewer. If this is not the case, select the New Window –> Node Details item from the Main Menu of the Node Viewer panel.

In the Action Parameters section of the Node Details panel, you will find a field for the Action node parameter. Click on the field and select the MayaRender Action from the drop down menu. The Node Details panel will change and now display all the parameters associated with the MayaRender Action. Click on the field for the Maya Scene parameter and select (character1.mb). Finally, click on the Apply button (white arrow) in the upper right-hand corner of the Node Details panel. This causes any changes that you made to be applied to the working version of the node. If you exit the Node Details panel or load up another node's parameters without first pressing the Apply button, none of your changes will be saved. After applying your changes, Pipeline will update the associated node panels.

After the update, the (render1) node will have turned purple. The purple color indicates that the node has a queue state of Stale, which means one or more jobs need to be run to regenerate the node's files. In this case the node became Stale because we assigned a new Action to it. Pipeline will not consider the node to be Finished until the Action has been run to successfully regenerate the files associated with (render1). Right-click on the node and select Queue Jobs, from the Node Menu. This will cause Pipeline to generate and submit all jobs required to regenerate any Stale or Missing files associated with the node and any of its upstream dependencies. In this case only the (render1) node requires regeneration and a single job will be submitted which regenerated the (render1.#.iff,1) files. After submitting the jobs, the node panels will be automatically updated.

You may see any of three different queue states after the status updates finishes, depending on the state of your queue and how quickly the status update occurred. The node may be teal indicating that jobs are waiting to be run, green indicating that jobs are currently running, or blue indicating that all jobs have finished running. You may press the spacebar to update the status of the Node Viewer panel to monitor the progress of the jobs associated with the (render1) node.

Once all jobs have finished successfully the (render1) node will become blue. If remains light blue indefinitely, then there is either something wrong with the queue configuration or the queue is currently busy processing other jobs. You can skip ahead to the sections on managing jobs to see how to check the status of jobs in the queue or talk to your system administrator. Once the (render1) node is finished, go ahead and view the images using the Edit menu item or by pressing Enter over the node. If everything looks good, check-in the (render1) node




The two nodes after the link has been made.


The Node Details panel showing the Action Parameters that are created after the MayaRender action is selected. The Maya Scene parameter has been set to (character1.mb), telling the Action which files to render.


The apply button on the far right. When it is white, there are unapplied changes in the panel.


The node has turned purple, indicating it is Stale and needs to be queued in order to regenerate the files associated with the node to make them up-to-date.


The node has turned cyan, indicating jobs associated with the node are waiting to run.


After all jobs associated with the node have finished running and the node is ready to be checked in.

# A Closer Look at Nodes

Nodes are the basic building blocks of Pipeline.

In the last two example, we demonstrated how to register a node in Pipeline, check it in to the repository, create a second node, link them together, and use an Action to generate a node's files. In this next section we will examine these steps and the underlying Pipeline concepts in more detail.

At its heart, Pipeline is a version control system based around the idea of nodes. Instead of simply version controlling files, as many existing systems do, Pipeline also preserves all information required to regenerate these files in exactly the same manner as they where originally created. all dependencies on nodes upstream in the production process required by this Action. The Action associated with a node preserves the specific programs executed and all the details of their execution such as command line arguments. Pipeline considers a change to any of this information to be modification of the node even if none of the files associated with the node have been changed. This fact is central to the goal of Pipeline to preserve the files and the exact process of how they were created so they can be regenerated later. It might not be obvious why this is necessary considering that Pipeline is already preserving the files generated by this process. However, if changes are later made to any of the nodes upstream in the production process, it will be critical to know precisely how to regenerate the files associated with the node in order to incorporate these changes correctly.

We will now explain the various properties of a node and discuss how they work together to precisely specify a production process.

## File Sequences

A node in Pipeline is associated with at least one file sequence which can contain one or more files. The names of the files which make up a file sequence composed of three parts: a prefix, an optional frame range, and an optional suffix. The tables to the right give some examples of legal and illegal Pipeline file sequences.

The fully resolved name of the node determines the directory path and the prefix of the primary file sequence. The directory path will be relative to the root directory of the working area containing the node ($WORKING). For example, a node with the name (/test/images/image1) is associated with a file named something like (image1.jpg) located in the directory ($WORKING/test/images). Every fully resolved node name in Pipeline must be unique. Once a node has been created with a certain name, this name cannot be reused by any other user or in another working area, even if the node has not yet been checked in. The newly created node will be visible in the Node Browser, even before it is checked in, allowing other users to see that it already exists.

Secondary sequences can be used for nodes who's Actions generate multiple files in one pass.

Sometimes it can be advantageous to associate multiple files or multiple sequences of files with a single node. This is the case when the Action associated with a node regenerates all of these files at once. For example, multiple rendering passes (beauty, diffuse, ambient, specular, etc...) for the same frame could be generated by a single invocation of a render. If a separate node was associated with the files from each pass, then Pipeline would have to run a separate job to render each one, resulting in much wasted computation. When multiple files can be generated simultaneously, it is much more efficient for this to be accomplished by one node. Pipeline uses secondary file sequences to specify additional file sequences beyond the mandatory primary file sequence associated with all nodes.

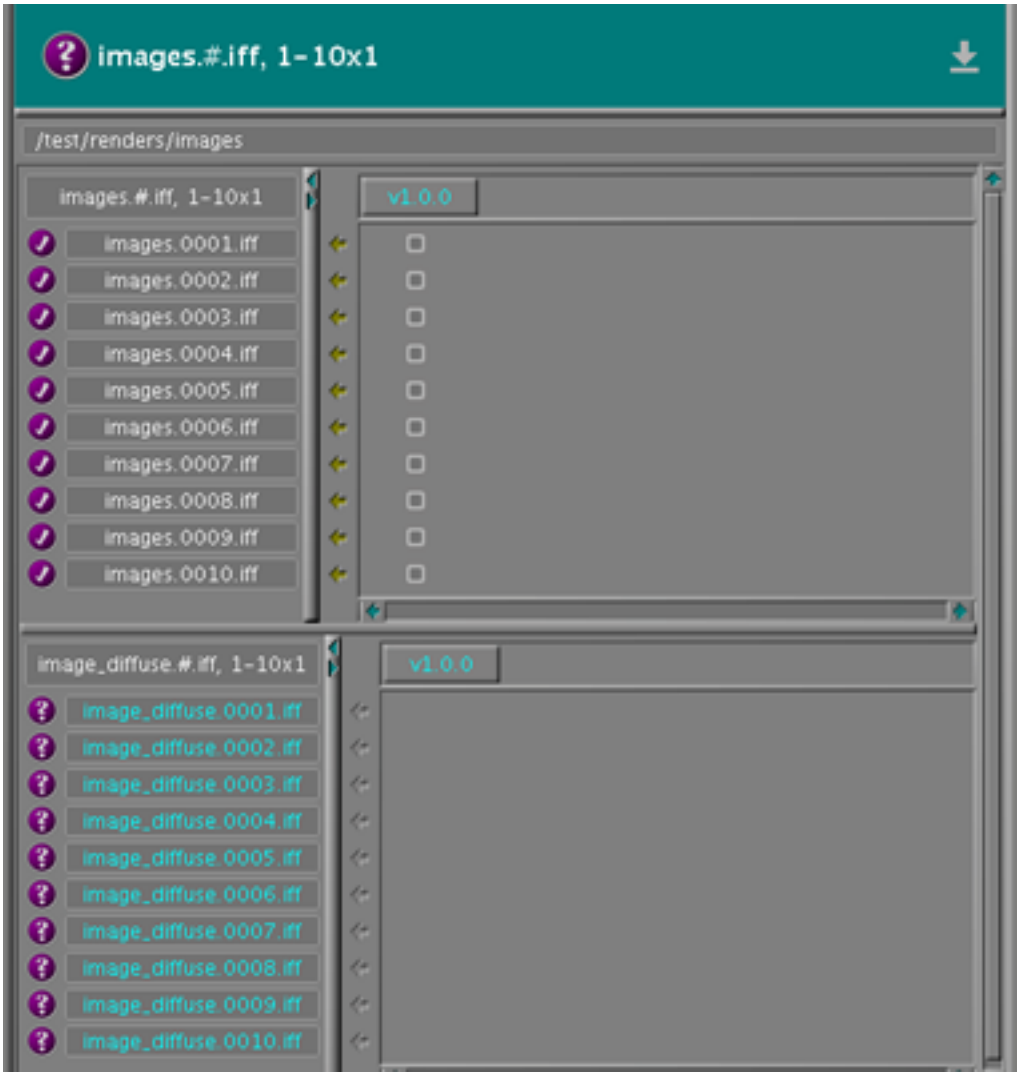
There are some additional restrictions on the files which can be used as secondary file sequences. Secondary file sequences must reside in the same directory as the primary sequence and must have the same number of frames as the primary file sequence. However, it is not necessary that the primary and secondary file sequences have exactly the same frame range. Secondary file sequences must have either a different filename prefix or suffix than the primary file sequence and other secondary sequences. For example, a depth pass for a render might have a primary file sequence

| Filename | Prefix | Frame Range | Padding | Suffix |
|---|---|---|---|---|
| example.txt | example | none | none | txt |
| images.#.iff, 1-20x1 | images | 1 to 20 by 1 | 4 | iff |
| example | example | none | none | none |
| images.@@@, 1-10x3 | images | 1 to 10 by 3 | 3 | none |
| image.0001.tiff | image | 1 | 4 | tiff |

Examples of file sequences which satisfy Pipeline's file naming conventions.

| Filename | Problems |
|---|---|
| images.depth.#.iff | Pipeline does not allow a period in the prefix |
| images.iff.# | Pipeline does not support frame numbering after the suffix. |
| images#.iff | Pipeline requires a period between the prefix and the frame number. |

Some examples of file sequence which cannot be used with Pipeline.




The Node Files showing a secondary sequence. At the top is the primary sequence, with the name of the files matching the name of the node.

Below that is the secondary sequence. It has the same number of files as the primary sequence, and in this case has the same frame range as well. The Missing symbol is next to the images in the diffuse pass indicates that they have not been generated.

named (images.#.iff) and the secondary file sequence named (images.#.z). In another example where secondary sequences are used for shading passes, the primary file sequence could be named (images_diffuse.#.iff) and a secondary sequence named (images_specular.#.iff).

As noted above, secondary file sequences do not have to have the same frame range as the primary file sequence as long as they have the same number of frames. It is perfectly legitimate for the primary sequence to go from frames 1-20x1, while the secondary sequence goes from 56-75x1 or from 2-40x2 since all of these sequences contain exactly 20 frames.

## Actions

Most leaf nodes in Pipeline will not have an Action associated with them since the files associated with these nodes are generally created manually by artists. We saw this kind of node in the first example, where the (character1) node had no Action because the associated Maya scene was created directly by the user. Actions become necessary any time trees of nodes are created to represent a production process and Pipeline is responsible for generating the files associated with the nodes downstream of these leaf nodes. Some examples of production processes automated by Actions are listed on the right.

Actions insure consistency by codifying how a production process is performed and guaranteeing that the same results will be produced every time an Action is run as long as all of its upstream dependencies remain the same. Actions can define parameters which can be used by artists to control their behavior. Ultimately, an Action plugin is responsible for executing some combination application binaries or other utilities in a precise way based on the nodes parameters and upstream dependencies.

When a node has an Action, the Action plugin is entirely responsible for creating the files associated with a node. For this reason, the files associated with any node with an Action have read-only permissions. If an artist were allowed to modify the files after the Action ran and then check-in those files, another user would not be able to regenerate the same results using the Action alone.

However, it is sometimes desirable to make changes to the files associated with a node which has an Action. For example, take the case where an animation file is generated by an expensive dynamics simulation. When you inspect the results of this simulation, everything appears to be satisfactory except for two objects which need to be adjusted. You might be able to correct this problem by modifying the input parameters and running a new simulation. However, clearly this is not the most efficient approach possible. It might be far simpler and faster to manually edit the animation for these two objects. Pipeline will allow you to edit the animation file if you first disable the Action associated with the node. A node which has a disabled Action still retains the information about how the files associated with the node were originally generated, but allows manual modifications to be made to the node's files. A disabled Action also serves as a warning to anyone else looking at the node in the future that the results of the Action have been manually modified.

When disabling an action to edit a file, it is often wise to check-in the node with the Action enabled, then disable it, make your changes, and the check it in again. This way, if you ever want to return to the original data, there is no need to regenerate it since it exists in the repository. If all you checked-in was the edited version of the node with a disabled action, you would not be able to get the original results back without rerunning the Action which could take a long time.

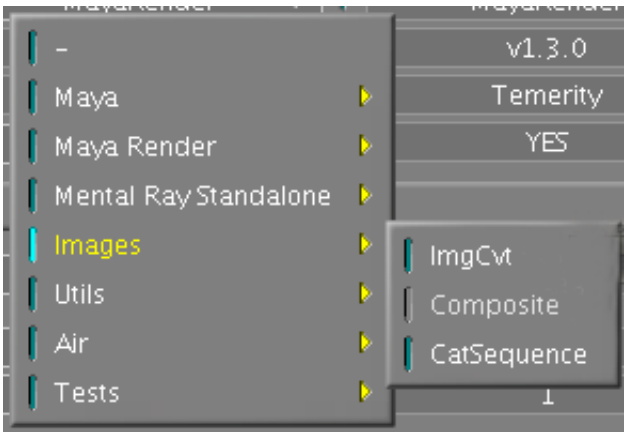
Pipeline ships with a default set of Actions that allow it to perform most common operations involving industry standard tools. It also features a full Java API allowing you to add custom Actions as different needs arise. Any program that has command-line functionality can be added as an Action. Consult the development docs for more information.

**Examples of production processes automated by Actions:**

- Building dynamics scene from models and saved animation
- Running a cloth, particle, or RBD simulation
- Applying MEL scripts to a Maya scene.
- Combining Houdini CHOPs.
- Exporting RIB's or other renderer input files from 3D scenes.
- Batch rendering a sequence of images using any command line renderer.
- Compositing one image sequence on top of another sequence.
- Performing Image or geometry format conversions.
- Pre-filtering textures, environment maps or building shadow maps.
- Building HDR images.
- Converting camera data from motion-capture rigs to 3D.
- Running perl, python or, shell scripts.
- Renaming files or resizing images.


Part of the Node Details Panel, showing the Enabled property for the Action.


Part of the Node Details Panel, showing the Toolset and the Editor fields.




Notice how the Composite Action is disabled. This is because the current Toolset does not support this Action.

## Editors

Editor plugins launch interactive applications used to modify files associated with nodes.

Editors are another type of plugin for Pipeline. An Editor specifies how to launch interactive applications capable of viewing and modifying the files associated with a node. For example, the Maya Editor plugin launches the Maya to edit a scene file and the Emacs Editor launches the popular text editor of the same name to edit the files associated with a node. Editors are also commonly used for interactive programs which simply display images or geometry. Editors can even be used to inspect the read-only files associated with checked-in versions of nodes. As with Actions, additional Editor plugins can be added to Pipeline to extend its functionality.

## Toolsets

Toolsets define the environment used to run Editor and Action plugins associated with a node.

A Toolset is essentially a named collection of environmental variables managed by Pipeline. When an Action plugin is run to regenerate the files associated with a node or when an Editor plugin is used to launch an interactive program, any processes started by these plugins are executed under an environment defined by the Toolset property of the node. This insures that these processes are executed in a completely consistent and reliable manner. The way nodes behave is therefore completely independent of any per-user or even site wide shell configurations. Without this guarantee, a version of a node check-in by one user might behave differently when checked-out by another user or even at a later date by the same user.

In order to fully understand the role played by Toolsets it is important to realize that plugins such as the Maya Editor do not specify an absolute the location of the (maya) executable. Instead, the (PATH) environmental variable defined by the Toolset associated with the node is used locate the binary. This means that one Toolset might cause the Maya Editor to launch Maya-6.5 while another Toolset with a different value for (PATH) might cause the same Editor plugin to launch Maya-7.0. This is also true of environmental variables specifying the directories to search for plugins, scripts and other resources which may change the behavior of the program when launched. The behavior of Action plugins is dictated by the Toolset environment in the same way.

# Node Status

A Node's status has two parts, the node state and the queue state.

In the Node Viewer and node detail panels, Pipeline uses a collection of colorful icons that provide quick visual summary of both revision control and queue related status information for node. The revision control status, called node state, is represented by the symbol displayed by the node icon, while the color of the node icon background indicates the queue state.

## Node State

The node state compares the Node's files and metadata to the repository and determines how the working version of the file relates.

Node state is determined by comparing the properties, files and links of the version of the node in the current working area with versions of the node in the repository. There are two specific repository versions used in this comparison. These are the latest version of the node in the repository and the version of the node that was checked-out into the working area. Many times the latest and base versions will be the same version. When Pipeline goes to calculate status, it compares the properties and the files in your working area against these two versions and comes up with the node state.

For more information on resolving Conflicted states, see the section on Evolve Node in the Node Viewer section.

A node can potentially have many nodes states, but Pipeline can only display one at a time. Pipeline has a priority scale which determines which state is most important to show the user and use that scale to pick from the variety of node states a single node can have. The Conflicted state is considered to be the most important state, since that signals a serious situation where changes to a node may not be preserved correctly in the revision control system. The Conflicted state means that you edited a version of the node which was not the latest. Often this can occur without you even realizing it. You check-out the latest version of a file and start editing it. When you finally go to check it in, you realize that it is now in the Conflicted state because someone else has checked-in a version since you last checked it out. This situation needs to be resolved by communication between the two users who are in conflict. It usually comes down to whose changes will take longer to recreate. The person with the less complicated task will have to redo their work. More information and an example on how Conflicted nodes can be dealt with can be found in the Resolving Node Conflicts section.

Right below Conflicted in priority is the Missing state. The Missing state indicates that at least one of the node's files does not exist. This is a situation that shows how node state can be effected by just one file in a sequence. If a node represents a thirty frame sequence and even one of those frames is not present, then the node state will always be Missing. Missing also indicates that the working version of the node is based on the latest repository version. If this isn't the case, then the node will have the Missing Newer state instead. The Missing New state shares the same priority with the Missing state.

The three states mentioned already all indicate problems that require user intervention to resolve. The Pending state simply indicates that the node has never been checked-in. There are only two states possible for an non checked-in node, Pending and Missing. Nodes that have been checked in and that are not in a problem state, have a node state that reflects how the working version relates to the latest repository version. There are three states that deal with modifications to your working version, if the working version was based on the latest working version. In order of priority they are Modified, Modified Links, and Modified Locks. The priority system means that is a node is in a situation where is could be both Modified and Modified Links, since both the node and one of its sources were edited, then Pipeline will display the Modified node state for that node.

The name Needs Check-Out can actually be a bit misleading. This state indicates that the node is identical to a checked-in version which is not the latest version. There is a gut reaction when seeing this state to immediately check-out the node and get the latest version. While it is nice to always have the latest version of node, it is often undesirable to immediately check it out. If you wish to edit that particular node, then checking it out is necessary. Editing a node in the Needs Check-Out state will force it into the Conflicted state.

## Node States

**Identical**
The properties, upstream links and associated files of the node in the current working area are identical to those of most recently checked-in version of the node in the repository. In addition, the working version of the node is based on the latest checked-in version of the node.

**Modified**
The version of the node in the current working area is based on the latest version of the node in the repository. However, one or more of the node properties or associated files is not identical to this latest version.

**Modified Links**
The upstream links of this node are not identical to those of the latest checked-in version of the node. This may be due to links which have been added to or removed from the working version or differences in link properties. This state may also due to differences in the versions of the upstream nodes connected to the working version and the latest checked-in version. Alternatively, this state may be caused by one or more of the upstream nodes having a Node State of Modified or Modified Links.

**Modified Locks**
The only difference between the working version of the node and the latest checked-in version is the locked state of one or more of the upstream nodes. One of these nodes has been locked or unlocked since the working version was last checked-out. Alternatively, one or more of the upstream nodes has a Node State of Modified Locks.

**Needs Check-Out**
The version of the node in the current working area is based on an older version than the latest version of the node in the repository. This signifies that a newer version of the node is available. Changes to the current working version of the node will produce conflicts. The three symbols associated with this state represent the magnitude of the difference in revision numbers between the latest checked-in version and the checked-in version upon which the working version is based.

**Conflicted**
The version of the node in the current working area is different than the latest checked-in version of the node. These differences are due both to changes of the working version and the creation of new checked-in versions since the time the working version was last checked-out or created. These conflicts between the working version and the latest checked-in version must be resolved before the node can be checked-in.

**Missing**
One or more of the files associated with the working version are missing from the current working area.

**Missing Newer**
Identical to Missing, except that version of the node in the current working area is based on an older version than the latest version of the node in the repository.

**Pending**
The node has been registered in the current working area, but has not yet been checked-in.

**Checked-In**
One or more checked-in versions of the node exist in the repository, but no version of the node has yet been checked-out into the current working area.

However if you are not editing that node, but merely using it as a source for another node you are working on, you may not want to check it out. Checking out a newer version will causing any nodes with Actions that depend on it to become Stale. Those stale nodes will have to be regenerated, forcing you to wait until you can continue working. Sometimes the modifications will be so significant that it is worth taking that extra time. One way that this is communicated is through the three different icons that can represent Needs Check-Out. The idea is that when a user checks in a node, they select the level of change that has occurred. If the change is not likely to have any real effect, the Micro option should be selected and other users will see the small inverted delta. If the change is so large that doing work without it is a waste of time, then the Major option should be selected and node icon will have the dot about the delta. Anything between those extremes will be a Minor change and the normal delta symbol.

So the icon is the first step in determining whether you want to check out the latest version. It is also helpful to consult the Node History panel. This should provide a summary of the changes that were made and can help determine if the changes matter to you. For example, if you are animating a character and notice it has a Major change, you may figure you should check it out. However, upon consulting the history you realize the Major change was a complete overhaul of the texturing and shading of the character. While that is definitely a large change, it will not impact your animation and you can continue working without wasting time to check it out.

The last node state is Identical. This is usually the state that you'll see after you've checked the latest version of a node out of the repository and is always the state that will appear after you've checked-in a node.

## Queue State

Queue state is determined by any jobs currently running or, if there are no current jobs, whether the node needs to have a job run to regenerate its files.

Queue state is determined by the node's Action, whether that Action is enabled or disabled, by the status of other nodes which are linked to the node, and by any jobs in the queue that exist for the node.

Like node state, queue states have a priority and they also operate on the idea that even one file with a particular queue state will impart that state to the entire node. First come the states that indicate that there are active jobs. The Paused state has top priority, followed by Running and then Queued. Next come the states that indicate that not all the jobs associated with the node successfully completed. These states are Failed and Aborted, with Failed taking precedency. Next is the Stale state, which indicates that at least one of the files associated the node need to be regenerated. The Finished state has the lowest priority and indicates that all the files associated with the node are up to date.

The two states that are primarily dealt with are Finished (blue) and Stale (purple) and it is important to understand what makes a node switch from Finished to Stale and back again.

**A Node will be in the Finished state because:**

- It has no action. If a node does not have an action, then there is no way for it to become stale.
- It has an action, but the action is disabled.
- The node has an action, but the working version of the node is identical to the repository copy of that same version. If the node has not been changed at all since it was checked-out, then there is no reason to have to regenerate the file.
- The node has an action, the working version of the node has changed, but the action has been re-run and completed successfully.

## Queue States

**Finished**
All files associated with the working version of the node exist and are up-to-date. If there is an enabled Action associated with the node, the last job executing this Action completed successfully. Nodes must be in a Finished state before they may be checked-in.

**Stale**
One or more of the files associated with the node is missing or requires regeneration. Files require regeneration when the node has an enabled Action and one or more of the critical properties of the node have been modified since the time the file was created. Files may also require regeneration when any of the upstream file which they depend upon are newer or are missing. Depending on the link properties, these upstream files may be associated with nodes more than one link upstream.

**Queued**
A job has been submitted to the queue which will regenerate the files associated with the node, but has not yet begun execution.

**Running**
A job which is currently executing will regenerate the files associated with the node.
Paused, A job has been submitted to the queue which will regenerate the files associated with the node, but was paused before it began execution.

**Paused**
A job has been submitted to the queue which will regenerate the files associated with the node, but was paused before it began execution.

**Failed**
The last job submitted to the queue which would have regenerated the files associated with the node failed to complete successfully. The job may have failed during job preparation or one or more OS level processes started by the job may have exited with a non-zero exit code. The job may also have been killed manually by a user after it began execution.

**Aborted**
The last job submitted to the queue which would have regenerated the files associated with the node was aborted (cancelled) before it began execution.

**Undefined**
The state is undefined because no working version of the node exists in the current working area (Checked-In) or the state of the node is not known (None).

**A node will be Stale for the following reasons:**

- An action was just added to the node.
- The node has an action, which has just been enabled.
- The node has an action and one of its parameters has been changed.
- The node has an action and one of the nodes it is linked to has been changed (see the next section for more on linking).
- A change has occurred in the nodes upstream that has not yet been reflected in files associated with the node.

## Node Link Properties

Actions which regenerate files associated with a node usually need some form of input data in order to proceed. For example, a node which renders a sequence of images needs to know which scene to render. All links between nodes in Pipeline are directed and acyclic meaning that information flows in a one-way direction from source node to target node and circular links are prohibited. The target (downstream) node is said to depend on the source (upstream) node. When an Action runs, it is provided with information about all the nodes it depends on and uses some or all of these nodes to regenerate the target files. There are two different types of links in Pipeline, references and dependencies, which differ in how they affect the target node's queue state. There are also two different ways for links to specify the relationship between files in the target and the source node, All or 1-to-1.

A Reference is an indirect relationship between nodes.

A reference link represents an indirect relationship between the target and source. Changes to the source node of this type of link has no effect on the queue state of its target node, but will affect the target's node state. For example, modifying the source node may cause the target node's node state to change from Identical to Modified Links, but will not make the queue state switch from Finished to Stale. References are used to describe situations where a change in the source node will be automatically reflected in the target node due to the contents of the files associated with the target node. For example, the latest texture images referenced by a model file are automatically used whenever the model is loaded. Because there is no need to regenerate the model file, no Action must be run, so there is no need to make the model node stale. This is because the model file only refers to the location of the texture and does not actually contain a copy of the texture image data.

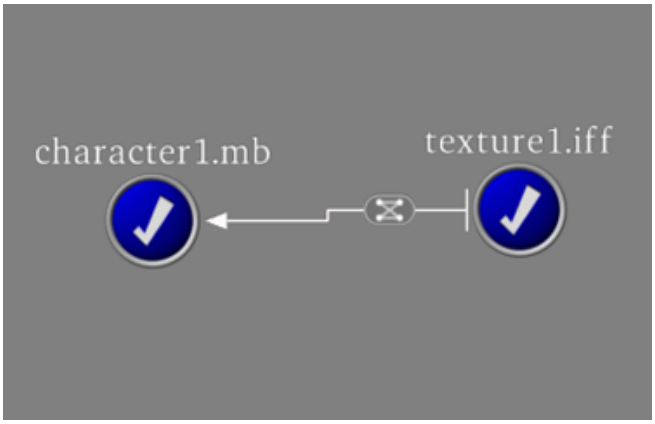
A Dependency is an direct relationship between nodes.

A dependency link represents a direct relationship between the target and the source. Any change to the source node of the link will affect both the queue state and the node state of the target node. A change in the source node may cause the target node's node state to switch from Identical to Modified Links and it's queue state to switch from Finished to Stale. Dependencies are used to represent situations where a change in the source node requires that the that the files associated with the target node be regenerated in order to reflect the changes made to files upstream. For example, an image sequence generated by a render will have to be re-rendered when it's source scene is changed.

References and Dependencies can work together in sometimes subtle ways to determine which nodes need to be regenerated. Take an example of the nodes on the right. The root node is a sequence of rendering images that depends on a Maya scene (character1), which in turn references an image file (texture). When the image file is modified, it indirectly causes (images) to become Stale. You'll also notice that both downstream nodes have a Modified Links node state because of this change. So while the change to the image file did not make (character1) Stale, the changes propagated downstream and resulted in the (images) node needing to be regenerated. Highlighted (yellow) links show the path of the propagation of this node staleness from the texture node down to the (images) node. This correctly reflects that a change in the texture does require the images to be re-rendered but does not require any changes to the Maya scene referring to this texture.

Links between nodes with multi-file sequences can be 1-to-1.

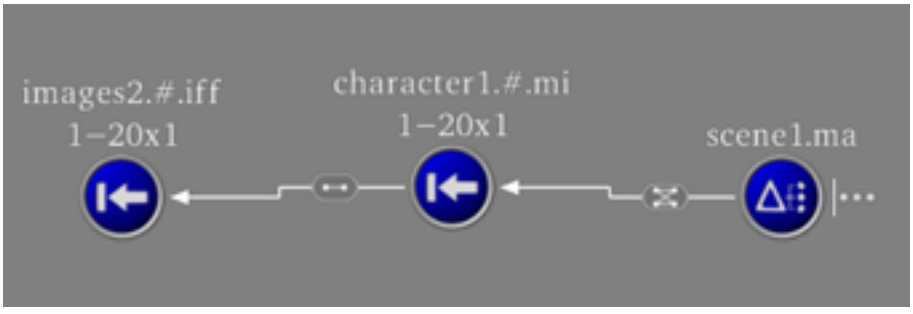
Link Relationships become important when dealing with file sequences that contain more than one file. A link relationship of All means that every file in the target depends on every file in the source. Therefore, if any file associated with the source changes, every file associated with the target may potentially require regeneration, depending on the type of link involved. In a 1-to-1 relationship, each of the target files depends upon exactly one source file, meaning a change to a single source file will only cause at most one of the target files to be regenerated. For example, if the target is a rendered image sequence and the source is a sequence of per-frame input MI files (or RIB's), then a link type of 1-to-1 would mean that a change to a single input file would only require a single image to be re-rendered.




A reference link. The vertical line at the beginning of the link indicates that it is a reference.


A dependency link. Notice the lack of any vertical lines at the start of the link.


This example shows staleness passed through a reference link.




An example showing the visual difference between an All and a 1-to-1 relationship.

Ordinarily, in a 1-to-1 relationship, the file sequence associated with the target node contains the same number of files as the file sequence associated with the source node. The first target file would be linked to the first source file and so on. However it is possible that the file sequences can contain different numbers of files and that the link relationship between their files may be offset. An offset of 5, for example, would mean that frame 1 of the target would depend on frame 6 of the source. It is possible that the frame index offset of the link may cause some of the target files to be linked to source files outside the frame range of the source node. The Node Details contains a setting called Overflow Policy in the Queue Settings that controls Pipeline's behavior in these cases

# Building a Simple Node Tree

We will now link several nodes in a small node tree and use Maya to render an animation scene containing two characters. We will be using the (/tests/characters/character1) node we created earlier as starting point. We add a node for a second character, combine these two characters into a rendering scene, create a MEL script specifying render settings and finally render some images using these settings. Along the way we will demonstrate some more node operations and examine the use of Actions in more detail.

## Cloning

The first step will be to clone the (character1) node to create a second character. If the (character1) node is not already visible in the Node Viewer, select it in the Node Browser. In the Node Viewer, press the right mouse button over the (character1) node to display the Node Menu and select the Clone item. This will display the Register Cloned Node dialog. You'll notice that the Filename Prefix field is already filled in with the name of the node plus "-clone." Change that to (/test/characters/character2). The rest of the default options should be fine. Go ahead and press the Confirm button. The newly created (character2) node has identical node properties to the original (character1) node. In addition a (character2.mb) file as been created which is a copy of the (character1.mb) file associated with the original node.

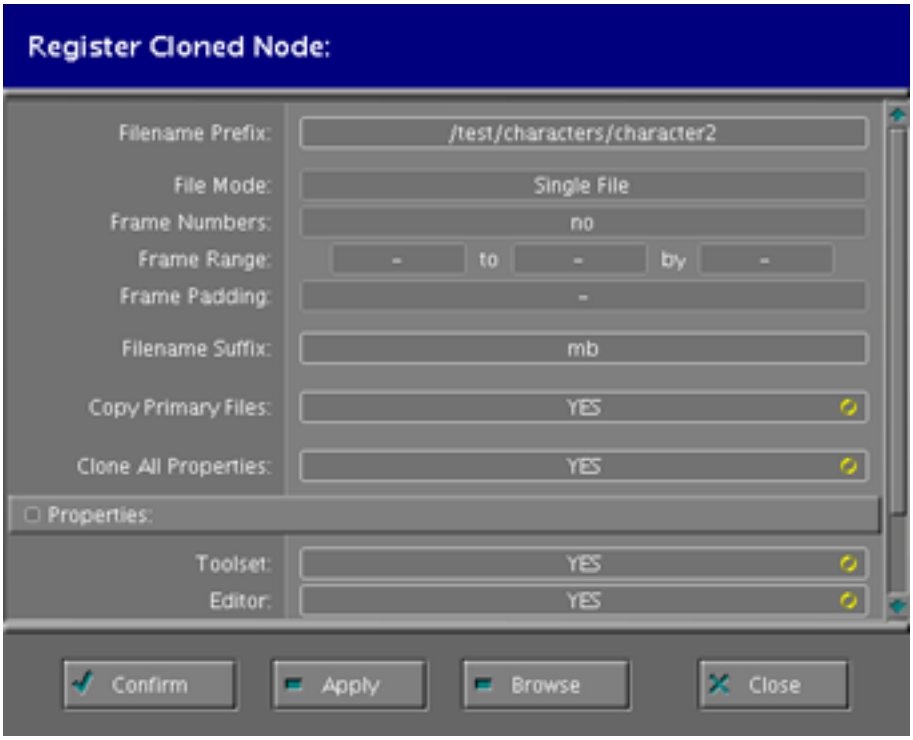
Cloning is a very powerful tool in Pipeline. It is the fastest way to create a variation of an existing node. Say you have a node with an Action which has a rather large list of action parameters that have been carefully specified. You then wish to create several nodes which are almost identical to this node but with slight variations in a few of these parameters. Instead of registering each node and having to specify each of those parameters from scratch, you can Clone the node and duplicate all of the properties. You'll probably want to set the Copy Primary Files field in the Register Cloned Node dialog to 'no.' You can then tweak the Action parameters you wish to vary and regenerate the files.

The Clone All Properties setting provides a quick way to switch all the Properties options from 'YES' to 'no' or vice-versa. Say you only wish to clone the Toolset property and nothing else. Simply set Clone All Properties to 'no' and then scroll down and set the Toolset property to 'YES.'

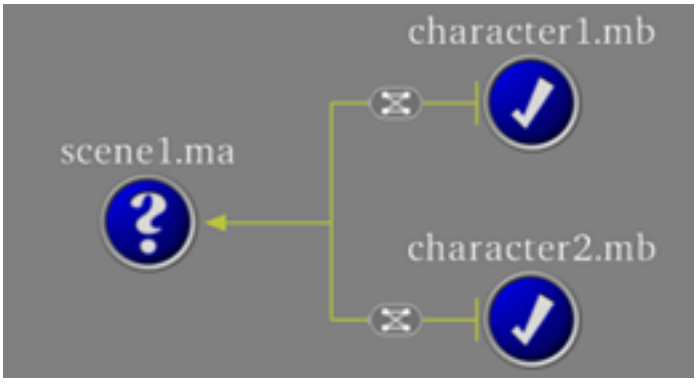
In addition to cloning node properties, Actions and action parameters, the Clone operation can also be used to copy the upstream links of the original node. This will be demonstrated in later examples.

## Reference Links and Source Parameters

Next we will use Pipeline to create an animation scene. Register a new node with the name (/test/animation/scene1) and the (ma) suffix. Make sure that you have both character nodes and the new animation node displayed together in your Node Browser.




The Register Cloned Node dialog. It looks similar to the Register Node dialog, but it has more extensive options below, controlling exactly what Properties are cloned.




The nodes after they have been linked. Note the vertical lines just to the left of the character nodes which that these are Reference links.

Select the two character nodes, press the right mouse button over the (scene1) node to display the Node Menu, and select the Link item. We will be using Reference links here because we will be importing the character model scenes into (scene1) using Maya referencing. Select Reference from the Link Policy and press Add.

In order to create the animation scene file (scene1.ma), we will use an Action called MayaReference. This action creates a new Maya scene which contains one or more of the model files (imported as Maya references) associated with the source nodes linked to the target node. Double-click on the (scene1) node to load its properties into the Node Details panel. Assign this node the MayaReference Action. You'll notice that once you select this Action a new field called Source Parameters appears in the Node Details. Source Parameters are similar to the Action parameters we saw before (called Single Parameters) except that these parameters can be specified for each source node file sequence. Pressing the Edit button next to the Source Parameters property will display the Edit Source Parameters dialog. In this case, the MayaReference action only defines one Source Parameter, called Prefix Name, but Actions can define any number of Source Parameters. If our example character source nodes had secondary file sequences, then there would a separate entry for each sequence and the sequence name would appear in the File Sequence column. Since neither of the two source nodes in our example has a secondary sequences, that column is blank.

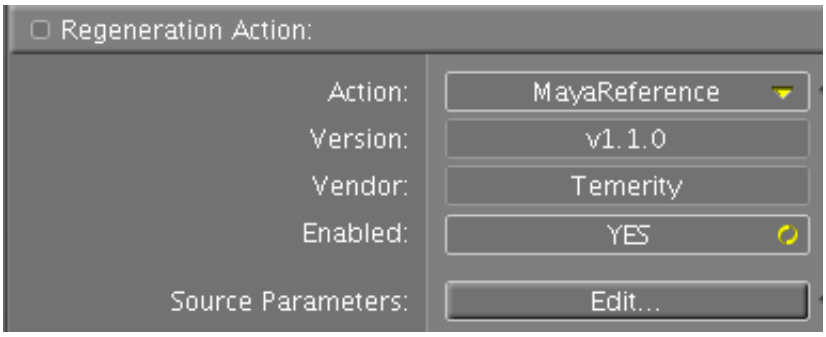
The Prefix Name parameter in the MayaReference Action specifies the namespace that will be used when the files are referenced into the target scene. First we have to tell Pipeline that we want to create Source Parameters for both source. Holding down (Shift), use (mouse1) to select both rows and press the Add button. Then type in the values 'char1' and 'char2' for the respective nodes. Press Confirm, to accept the changes you have just made. Then press the Apply button (the white arrow in the upper left hand corner of the Node Details Panel) to apply the changes to the node's Action property and Source Parameters. Finally, select the Queue item from the Node Menu for the (scene1) node to create the animation scene.

## Disabled Actions

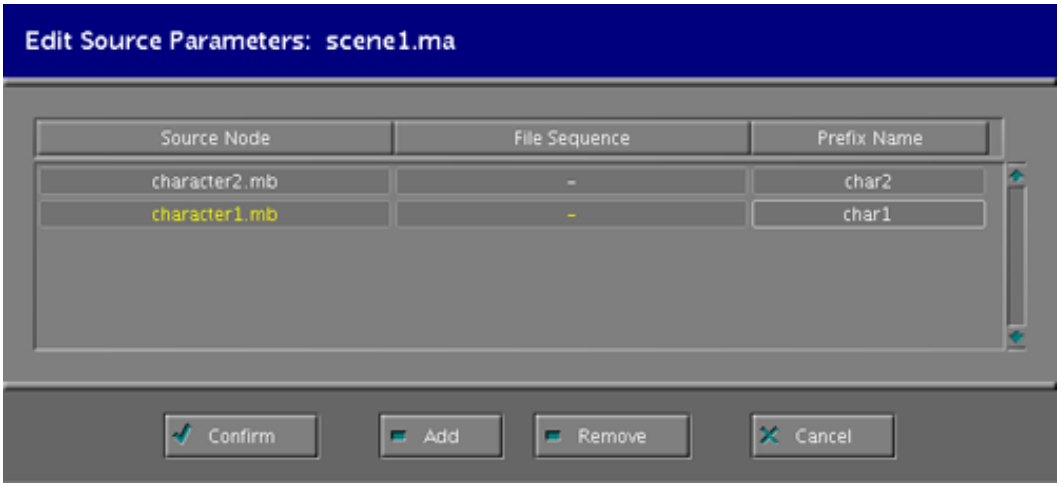
Disabled Actions allow manual editing of generated files, while preserving all the settings used to generate the files.

Once the job for the (scene1) node has finished running, there should now exist a Maya scene (scene1.mb) that references the two character models. However, at this point you cannot edit this scene because it has an Action leading Pipeline to make the files associated with the node read-only. In order to be able to modify the scene, you will need to disable the Action. In the Node Details panel, click on the Enabled option, changing it to 'no' and press the panel Apply button. In the Node Viewer panel, you should now see a vertical line to the right of the (scene1) node. This indicates that the scene has a disabled Action. You can edit the animation scene, to add some animation to the characters. Once you are done, save the scene and check-in your changes.

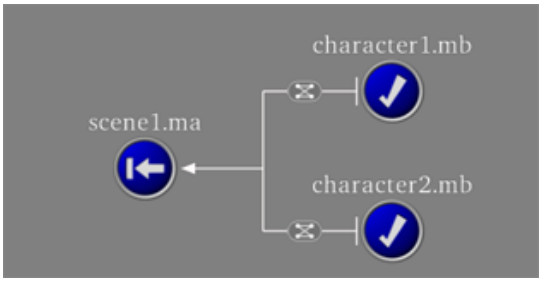
There are two things here which bear a little more examination. The first was touched on before, that is it sometimes a good idea to check-in a node before disabling its Action to preserve a copy of the generated files. However, in this case the cost of regenerating the scene is so minimal it is not really necessary to save a clean version. Also, it is highly unlikely that you will ever want to rerun this Action. Since all the Action does is create a scene with a couple models in it, there is no benefit to actually rerunning the Action.

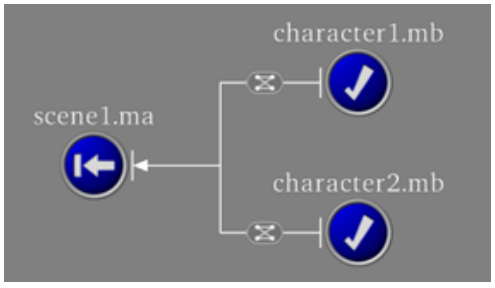



The Source Parameter field indicating that the MayaReference Action defines parameters for each source node file sequence. Clicking the edit button will open the Edit Source Parameters dialog shown below.

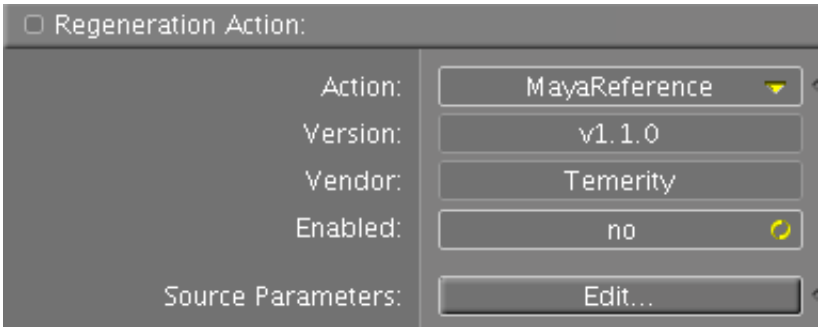



The Edit Source Parameter dialog, showing the Prefix Name parameter set for both nodes.




The nodes after the Action has finished and then after the Action has been disabled. The vertical line just to the right of the (scene1) node indicates the disabled Action.




A close-up of the Node Details panel showing the Enabled attribute set to 'no'.

The other point is why not remove the Action from the node entirely if you never intend to regenerate the Maya scene. There are actually two answers this. The first is that it is always good to know how the file was created. Disabling the Action preserves all the information that was used to generate this file. It is possible that will you wish to Clone this node or Export its settings to another node. If so, all of this information is still available. Second, if the node has no Action, then there is no visible display of that fact in the Node Viewer. Since the normal assumption is that branch nodes are always being regenerated, seeing a node without the Action Disabled line next to leads people to assume that the node has a valid Action. Leaving the Action associated with the node, but disabled provides better visual indicators to other people who might work with this tree.

## Leaf Nodes with Actions

It is rare in Pipeline to have a leaf node which has an Action, but it is useful occasionally. Register a new node with the name (/test/globals/globals) and the suffix (mel). Make the Editor a text editor (Emacs is fine). Double-click on the new node and in the Node Details panel assign it the MayaGlobals Action. The MayaGlobals action generates a mel script containing a list of render global settings which can be applied to a Maya scene before rendering to control the render settings. You'll notice that this Action defines a rather large list of Single Parameters. Scroll down and set the render globals to something that seems suitable to you and then press the apply button. The (globals) node should now be stale. Queue it and check it in when it is finished.

## Job Submission Details

File Sequences have queuing options that allow you to use different batching methods to generate their files.

Finally, we will create an image sequence to render out our animation. Register a new node with the name (/test/renders/images). Change the File Mode to File Sequence and set the Frame Range values to 1 to 10 by 1 and the Frame Padding to 4.

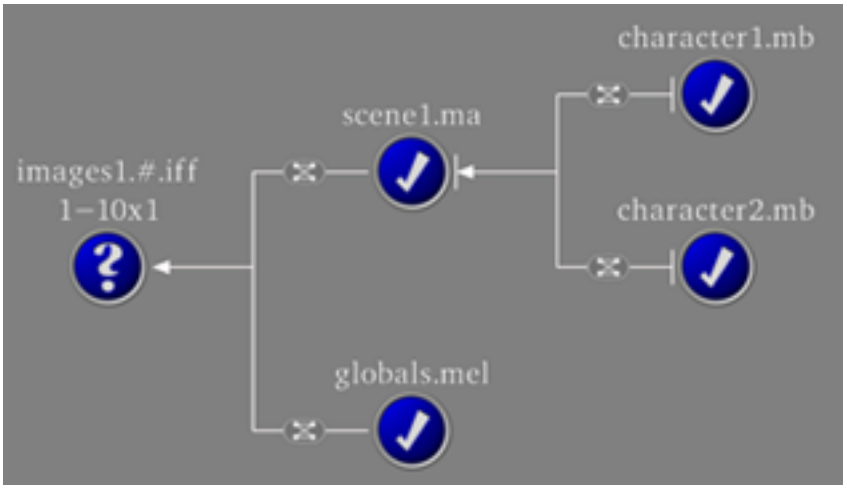
Next, link the (scene1) and (globals) nodes to the (images1) node as Dependency links. They are Dependency links because every time there are animation changes or render setting changes, we will want to re-render the images.

In the Node Details panel for the (image) node select MayaRender as its Action. This will display a set of new Single Action Parameters which correspond to different options that can be passed to the Maya command line renderer. There are two parameters that we need to specify. Set the Maya Scene parameter to (scene1.mb) to specify which scene to render. Set the Pre Render MEL parameter to (globals.mel). This will cause the render settings MEL script to be executed before Maya starts the render, applying your globals settings. Confirm they are both set correctly and apply your changes.

Before queuing this node, we will specify the method for assigning frames to jobs. Under Job Requirements is a list of parameters that control how jobs are are submitted to regenerate the files associated with the node. Currently, Execution Method is set to Serial. Serial means that whenever the Node is queued, a single job will always be submitted which generates all the frames on the same machine from start to end. This Execution Method is typically used for Actions which run some form of simulation that each frame must be computed in sequential order. For example: particle systems, cloth or other dynamics simulations.


Selecting the Maya Globals Action from the list of available Actions. Your list may look different depending on what Actions are installed in your version of Pipeline.


The Register Node dialog for the image sequence, showing the Frame Range fields filled in with the chosen values.



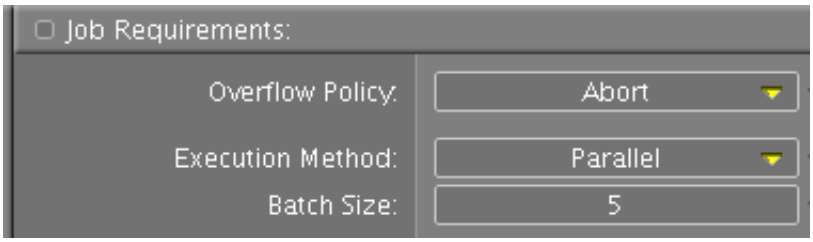

The node layout after the animation node and the globals are linked to the images node.

In this case, rendering an image sequence does not require this strict execution order since each frame can be rendered independent of the others. Therefore, we will switch the Execution Method to Parallel. Parallel means that multiple jobs can be created for the node and executed independently and possibly simultaneously. With Parallel, we can specify a Batch Size which determines the maximum number of frames allocated to each job. In this case, we will set the Batch Size to (5) which will usually create two render jobs which each render (5) frames. See the note below for an explanation of why this might not always be the case.

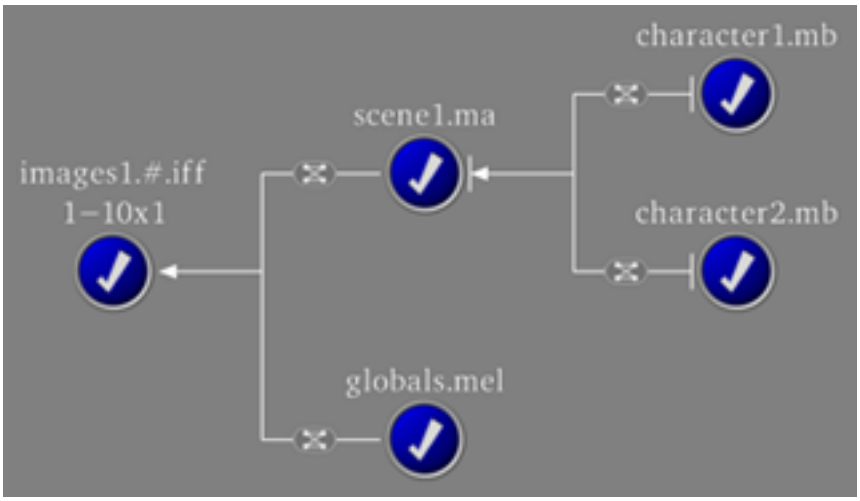
Apply the changes and queue the job. When it finishes, check in the rendered images.

If Batch Size is set to (0), as many frames as possible will be assigned to each job created. When all frames need regeneration, this would behave similarly to using Serial with one job rendering all of the frames. When only some of the frames need regeneration, Parallel with a Batch Size of (0) acts very differently from Serial. Imagine that after rendering the sequence we discover that the frames (2, 3, 4, 5, 7, 9, 10) contain some missing scanlines due to a network outage during the render. If we requeue this node with Parallel and Batch Size (0), it will create (3) jobs assigning frames (2-5, 7 and 9-10) to the jobs respectively. A Batch Size of (2) would create (4) jobs with the following frame assignments (2-3, 4-5, 7, 9-10).

There is also a third Execution Method called Subdivided which assigns jobs similar to Parallel with a Batch Size of (1), but changes the order in which jobs are processed. Subdivided recursively subdivides the frame range rather than increment through the frame range. For example, for a file sequence with a frame range of (1-8), Subdivided would process the frames in the order (1, 5, 3, 7, 2, 4, 6, 8).




The Execution Method is now set to Parallel, allowing you to specify a Batch Size.




The node layout after everything has finished running and has been checked in.

# Node Browser Panel

The Node Browser provides a catalogue of all nodes managed by Pipeline.

The Node Browser is used to navigate the nodes registered in Pipeline. It takes the form of a tree, mirroring the directory structure of the repository. Directories are expanded using the little icons to the left of the directory names. When you reach a level that cannot be expanded any further, you have found a node. Nodes have no expansion icon to the left of their name, but instead have a small circular icon indicating node presence in the repository and the current working area.

Selecting a node in the Node Browser will cause that node to be displayed in a linked Node Viewer. If no Node Viewer shared the same channel as the Node Browser clicking on a node has no effect. Selecting a node by clicking on it will clear all the nodes currently being displayed in the Node Viewer. To add to the currently displayed nodes, hold down the (Shift) key while selecting nodes in the Node Browser. Holding down the (Shift) key will also delay the panel update until the (Shift) key is released. You can use this to quickly select multiple nodes in the Browser panel. All nodes currently being displayed as root nodes are highlighted in yellow.

Holding (Shift) and clicking on a directory expansion icon will cause the complete expansion or collapse of all of the sub-directories below that directory.

## Releasing Nodes

Take a look at your current Node Browser, under the (/tests/characters) directory. The (character1) and (character2) nodes are both displayed there. Select the (character1) node. In the Node Viewer, choose the Release item from the Node Menu for (character1). The Release operation removes the node from the current working area. Pipeline will ask if you wish to remove the files as well. Go ahead and choose (YES).

What would have happened if you hadn't chosen to remove the files? Pipeline would have removed the node from your working area, but left the files associated with the node in the working area directory. One thing to keep in mind is that not every file in a working area is necessarily being tracked by Pipeline. Only files associated with checked-out nodes are managed by Pipeline. In general, it is not a good idea to leave important files not managed by Pipeline in a working area directory. You may inadvertently either overwrite or delete these unmanaged files when performing Pipeline operations in the future. For instance, whenever you check-out a node, any previously existing files with the same names as those associated with the node would be overwritten.

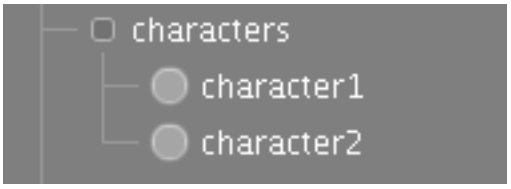
Releasing a file that has never been checked-in means that it is gone forever. Only release a Pending node if you wish to completely remove it from the Pipeline system.

## Node Browser Symbols and Filtering

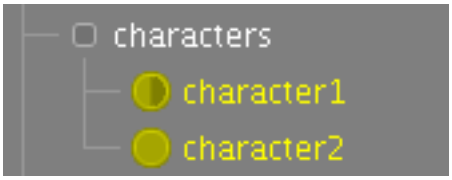
The nodes displayed by the Node Browser can be filtered based on the node's existence in the repository and user working areas.

In the Node Viewer, the (character1) node has turned grey and the symbol changed to Checked-In indicating that the node no longer exists in your working area. Also notice that the small symbol to the left of the (character1) icon in the Node Browser has changed as well. There is a chart below that explains what the different symbols mean.

Hold down the right mouse over the Node Browser and select the Node Filter menu item. This will display the Node Browser Filter dialog. This dialog allows you to hide nodes, based on the state represented by their icon. You can tell Pipeline to display or hide each of the five different states.

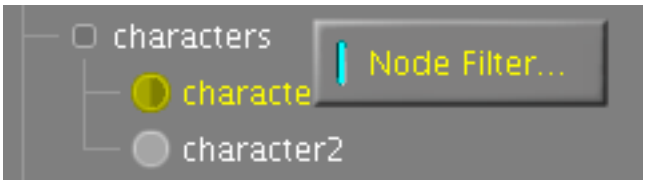



The character directory in the Node Browser in its original state.

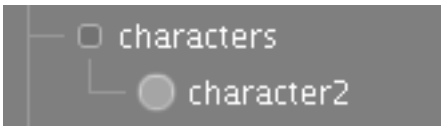


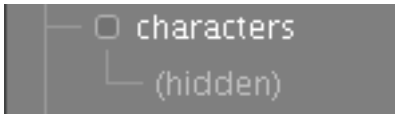

Selected nodes are highlighted in yellow.


The (character1) node after the node has been released.

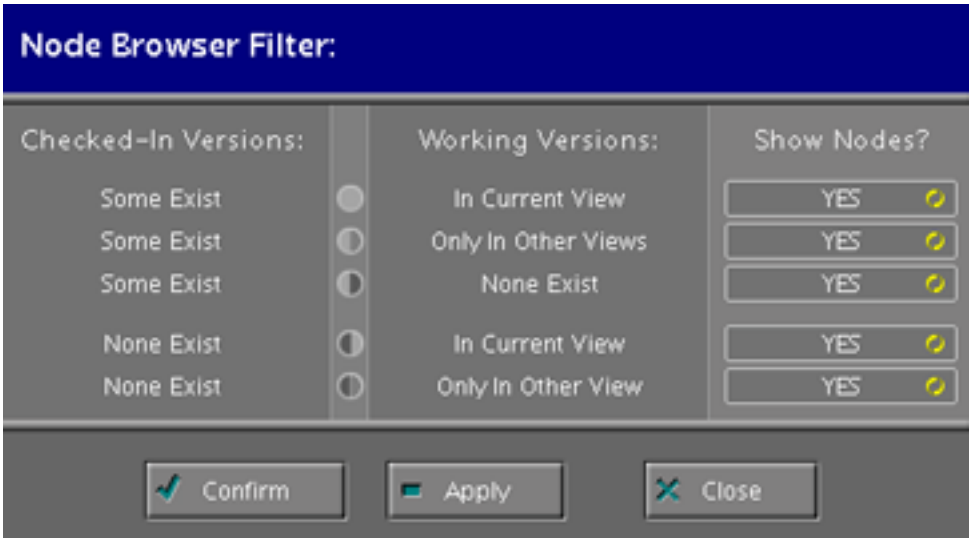



Selecting the Node Filter menu option. Notice that the icon next to (character1) has changed now that you have released the node.




The Node Browser, first with just (character1) hidden and then with both of the nodes hidden.




The Node Browser Filter dialog.

To demonstrate how this works, select (no) for all of the states except the top most option and press the Confirm button. When you look at the Node Browser, you'll notice that (character1) is no longer being displayed. Return to the Filter dialog and select (no) for all of the states. Now there are no nodes displayed under the characters directory, just the text "(hidden)" which tells you that at least one node exists in that directory, but all of the nodes are being hidden by current node filter settings. Return to the Node Browser Filter dialog and select (YES) for all of the states.

The circular icon next to each node is composed of two halves. The left half represents the node's existance in the repository while the right half displays node existence in working areas. There are three different colors used in these displays: light, dark, and medium grey. The first two colors are used for repository states (left side) while all three are used for working area states (right side).

- **Dark Grey**: This color means the node doesn't exist in the context in question. On the left side, it means that the node has never been checked-in (no repository version exists). On the right hand side, it means that no working areas contain a version of that node. It is impossible for a node to have dark grey on both sides since a node must exist in either a working area or the repository to be displayed at all.

- **Light Grey**: This color means that the node does exist. On the left side, it means that at least one checked-in version of the node exists. On the right side, it means that the current working area contains a version of the node.

- **Medium Grey**: This color only applies on the right hand side. It means that the node exists in at least one working area, but does not exist in the current working area.

# Node Viewer Basics

The Node Viewer is the most frequently used panel in Pipeline.

The Node Viewer is the heart of the Pipeline system and you will spend most of your time interacting with this panel. This is where node related operations are performed. It is also where the links between nodes can be added, removed and visualized. Nodes selected in the Node Viewer can be examined more thoroughly in the Node Details, Node Files, Node Links, and Node History panels. This is also the place where user created Tool plug-ins can be run, adding customized workflow enhancements.

## Viewer Navigation

Pan: (Alt+Mouse2)
Zoom: (Alt+Mouse1+Mouse2)

Navigation in the node viewer is accomplished using the mouse and the (Alt) key. The view of the panel can be panned by holding down the (Alt) key and dragging with the left mouse button. Dragging with both left and middle mouse buttons while holding down the (Alt) key will cause the view to be zoomed in and out. The Frame Selection and Frame All options adjust the view framed by the Node Viewer. Frame Selection causes the view to pan/zoom so that it shows the smallest possible area that contains all of the currently selected nodes. Frame All behaves similarly except that the view shows all of the displayed nodes. Hide All Roots causes no nodes to be displayed in the Node Viewer.

It is important to remember, though, that every node displayed in the Node Viewer adds to the time it takes to calculate status, so limiting the number of roots can speed up interactivity.

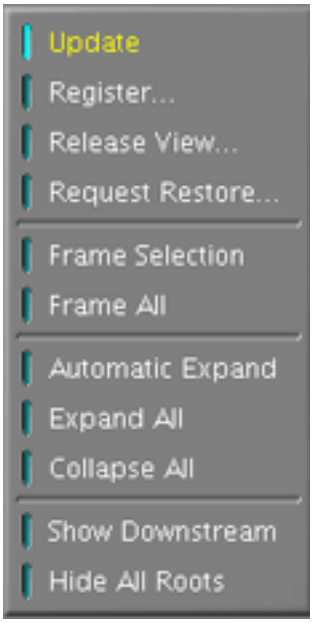
The Node is in the both the current working area and the repository.

The Node is not checked out in the current working area, but does exist in some working area and is also in the repository.

The Node is in the repository, but does not exist in any working area.

The Node exists in the current working area but has not yet been checked into the repository.

The Node does not exist in the current working area or in the repository, but does exist in some other working area.




The Node Viewer menu. This menu is activated by right-clicking on any blank spot in the Node Viewer.

## Controlling the Node Displayed

Node trees in Pipeline are displayed in the Node Viewer with the node most downstream in the production process (roots) on the left with the nodes upstream in the process arranged to the right. Nodes are drawn everywhere that they are used as an upstream dependency of nodes contributing to the currently displayed root nodes. A single node may be used in multiple places in a tree, it will be drawn multiple times. Nodes may be displayed as both the root node of a tree of nodes and as one of the branch or leaf nodes of another node tree. By controlling the root nodes displayed by the Node Viewer panel, artists can inspect the links between nodes from several independent vantages simultaneously.

The Show Downstream command allows you to see the tree of nodes which depend on the currently displayed root nodes.

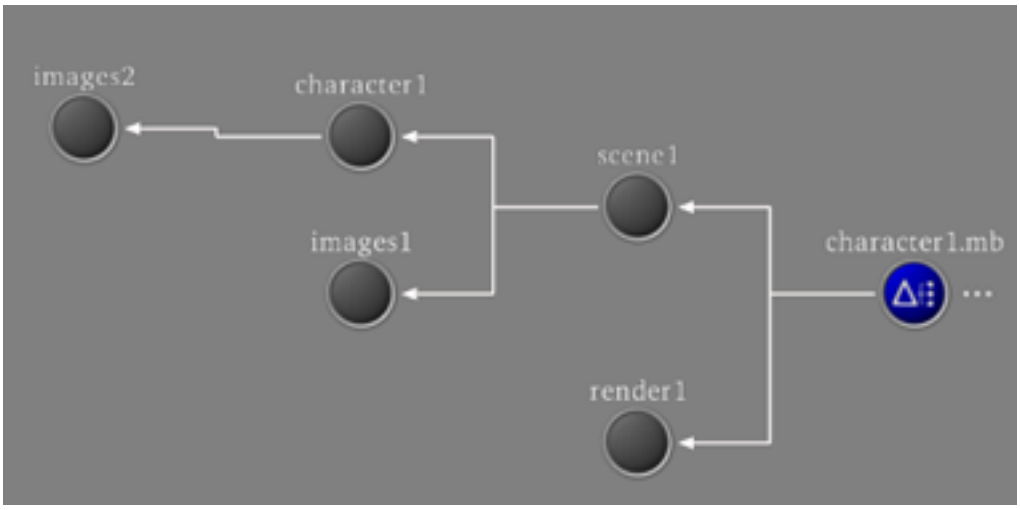
For example, select the (character1) node in the Node Browser. If it is not checked out, then Check-Out the latest version of the node. In the previous tutorials, we linked this node as a source to several other nodes. In order to have Pipeline display the nodes downstream in the production process select the Show Downstream menu item from the Panel Menu. All the nodes that depend on (character1) are now displayed to the left of the root. Pipeline shows the node connectivity, but will not calculate the status of these downstream nodes. This step is skipped because calculating the status of all downstream nodes could be incredibly time consuming. Instead the downstream nodes are displayed with the with a blank grey icon representing the Undefined state. To see the status of these downstream nodes, it is necessary to make them one of the root nodes or one of the upstream dependencies of one of these root nodes displayed by the Node Viewer. Downstream nodes can be hidden again using the same menu item which toggles between Show and Hide Downstream.

There are several operations available in the Node Menu which can be used to adjust the root nodes.

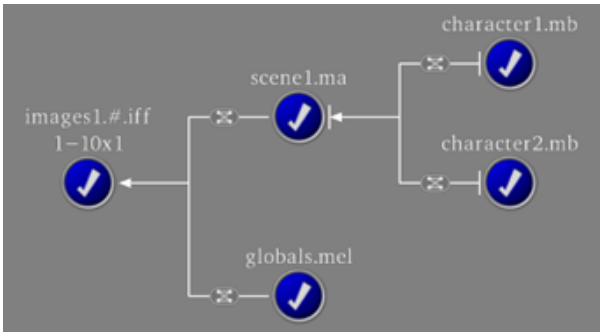
Click on the (images1) node in the Node Browser and select the Set Root menu item. All the other root nodes are cleared from the viewer. If you were to run Set Root on an existing root node it will simply clear any other root nodes. You can now click on Hide Downstream, since we no longer need to see parent nodes.

Right-click on the (scene1) node and select the Add Root menu item. This will add (scene1) as a new root node. If you selected Add Root on (images1) nothing would happen since it is already a root. What you should now see is Pipeline displaying the (scene1) node in two different places. In one tree it is a root node and in the other a branch node. A node can only show up as a root node once, but may appear an unlimited number of times as a source node, including multiple times in the same tree. However, Pipeline only calculates the status for the node once no matter how many times it appears in the Node Viewer.

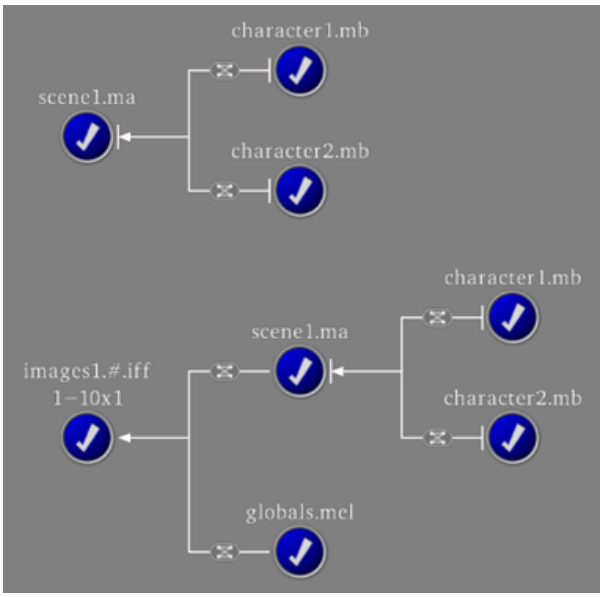
Now right-click on the (character1) node in the (images1) tree. Select the Replace Root option from the menu. Notice that (images1) has disappeared and (character1) is now being displayed as a Root. Running Replace Root on an existing root node will have no effect.




character1 with it's downstream links displayed. They are all grey, with no node status icon. Because a node might have hundreds of downstream connections, calculating a status for all of them would take a prohibitive amount of time.




The display after Set Root was run on (images1).




The display after Add Root was run on (scene1). Notice how (scene1) now shows up twice in the Node Viewer.

## Collapsing/Expanding Node Trees

Pipeline has a set of controls that can be used to manage how much of a tree is displayed.

Often you will have deep node trees that can be difficult to view in their entirety in the Node Viewer. For this reason, Pipeline allows you collapse and expand different branches of node trees. Select the (images1) node in the Node Browser. In the Node Viewer, middle-mouse click on the (scene1) node. Notice that all of the nodes underneath it disappear and are replaced with an ellipsis. Middle-mouse click again on the node and its source nodes will reappear.

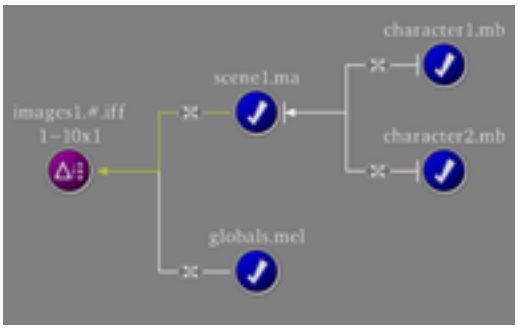
These include middle-mouse clicking on nodes, commands in the Node Viewer menu, and hotkeys.

There are several commands in the Node Viewer menu, accessed by right-clicking in the Node Viewer, that control how nodes trees are displayed. Expand All and Collapse All cause all displayed trees to be fully expanded or collapsed When a tree is fully collapsed, only the root node is shown. If you run Collapse All now, only (images1) will be displayed. Run Expand All to return the tree to its default state.

Automatic Expand causes all nodes to be fully expanded the first time they appear in a tree and collapsed every other time. So if you have a tree of nodes which is used as a source four times in a large tree, then that source sub-tree will be fully expanded the first time is appears, but fully collapsed the other three times.

There are also a series of hotkeys that allow you to quickly expand and collapse trees. The hotkeys are Alt and any of the number keys, with the number corresponding to the number of tree levels that are expanded. For example, (Alt-1) is the same as Collapse All, only showing the top most level. (Alt-2) will show two levels of the trees. In our example, this is the images node and then the animation node and the globals node. These hotkeys can be changed in the Preferences dialog.

Pipeline remembers the expansion information for each tree, so if you hide a root and then view it again later, it will remember what branches were expanded and which were collapsed.




The node tree fully expanded.


The same node tree after middle-mouse clicking on (scene1). The two model nodes have been hidden and replaced with the ellipsis.

# Check-Out (Overwrite All) and Evolve

The Overwrite All option for Check-Out will copy the entire checked-in tree out of the repository overwriting any local copies of the checked-out nodes.

The Check-Out operation is one of the more complicated node operations. Care should be take when using this operation because there are some potentially destructive consequences for nodes and files in your current working area. Knowing when and why to check-out nodes is an important part of efficiently using Pipeline. For these examples we are going to be using (scene1), the animation scene we created before, which has the two model dependencies, (character1) and (character2).

When we start, all of the example nodes we have been using should be in the Finished/Identical state. Open up (character1) and make some modifications to it. After you save the changes and update the status in Pipeline, you'll notice that (character1) is Modified and (scene1) is Modified Links. Go ahead and check in (scene1).

## Node Files Panel

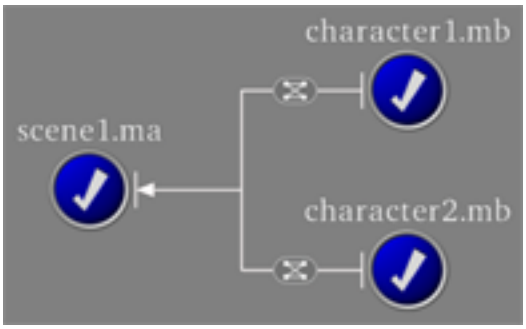
The Node Files panel allows you to see where files actually changed between versions, as opposed to node metadata.

Before we start examining the details of Check-Out, we are going to take a brief look at the Node Files panel. The Node Files panel allows you to see the specifics of changes made to the files associated with a node as opposed to the node metadata. When we checked-in (scene1), the associated (scene1) file had not actually changed at all. The new node version was based completely on one of its sources changing. When you are trying to track down a change that occurred, it can be helpful to know if the underlying file have actually changed.

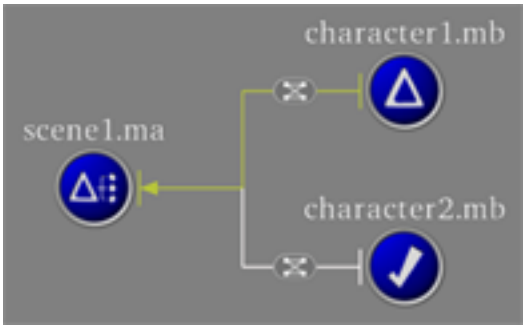
Make sure that you have a Node Files panel linked to your Node Browser and Node Viewer and double-click on (character1). If you look at the Node Files panel, you'll see each version of the file listed across the top with a small checkbox underneath it. The checkbox means that the files associated with each node version are different from the previous version. At some point the file was opened and changes were made. Now double-click on the (scene1) node. The list of versions is still across the top, but there is only one checkbox underneath the oldest version and a line extending to the left ending in a vertical line under the newest version. This graphic means that the associated file did not change between those two version. A result like this indicates that any change in the node's behavior is due either to change in its metadata (which can be examined in the Node Details panel) or in its links (which can be examined in the Node Links panel).

Return to the Node Viewer and run Check-Out on (character1). Under Check-Out Version, select (v1.0.1). This will check-out the older version of the node into your working area. When you look at the tree, you will see that (character1) is now in the Needs Checkout state and that (scene1) is in the Modified Links state. Go ahead and check-in (scene1), with a check-in message that mentions that you rolled-back (character1) to an earlier version.

It may be a little confusing that Pipeline has just let you check in something that isn't the latest version or based on the latest version. It's important to remember that you are not creating a new version of (character1) in this situation. You are only creating a new version of (scene1) which is linked to an older version of (character1). This does not cause any problems. There are plenty of situations where you might wish to check-in a tree that contains an older version of a source node. One common occurrence is when you have an overnight render based on a group of models, some of which will be edited by other people before the render finishes. If you could only check-in trees where every node was based on the latest version, you would have to check-out all the models and then rerun the render, clearly not something you'd want to do. Pipeline's rules about check-in are as follows. The root node of a check-in must be based on the latest version. There are no exceptions to this rule. Any source nodes can be older versions (in the Need Checkout state) as long as they have not been modified (which would put them in the Conflicted state).

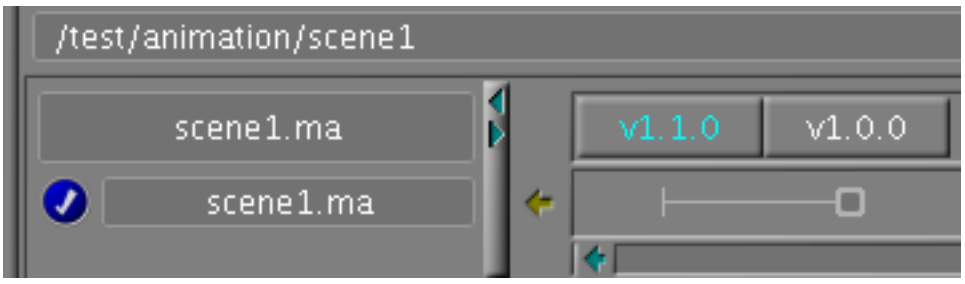



The beginning state for the nodes.

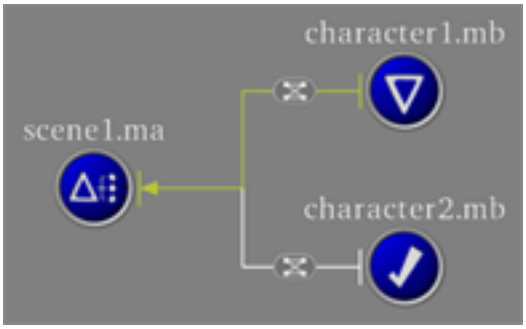



The node status after (character1) was modified.


The Node Files panel for (character1). The row of checkbox means that each file is physically different from the previous version.




The Node Files panel for (scene1). The line from v1.0.0 to v1.1.0 indicates that those two files are bit-for-bit identical.




(scene1) now depends on an older version of (character1). This is an acceptable state.

## Node Links Panel

The Node Links panel displays how the links of a node have changes between versions.

Lets look at the Nodes Links panel and see how Pipeline records this information. Create a new Nodes Link panel if you don't have one already and load up the details for (scene1). Each source node has its own section in this panel. Across the top of each section is a list of the versions of (scene1). Underneath that is a table showing the version, link policy, link relationship, and frame offset for the source node. If you look at the table for (character1) you'll notice the change in Reversion Number between v1.1.0 and v1.2.0 of the animation scene. This is the change we just made when we checked out the older version of (character1) and then checked in a new version of (scene1) based on that older version of (character1). If you look down at (character2) you see the same horizontal line structure that you saw in the Node Files panel. Like in the Node Files panel, the line means that no changes were made to this link between the different version: v1.0.0 of (character2) has always been linked.

## Check-Out (Overwrite All)

Back in the Node Viewer, check-out the latest version of (character1). This will cause (scene1) to change to the Modified Links state. Remember that the latest checked-in version of (scene1) is based on v1.0.1 of (character1). Checking out a different version of that model, whether it is older or newer, will always cause the animation scene to become modified. Now select the Check-Out menu item on the (scene1) node. Change the Check-Out Mode to Overwrite All and click the Check-Out button. You'll notice that the newer version of (character1) has been overwritten with v1.0.1. This exemplifies how the Overwrite All setting operates.

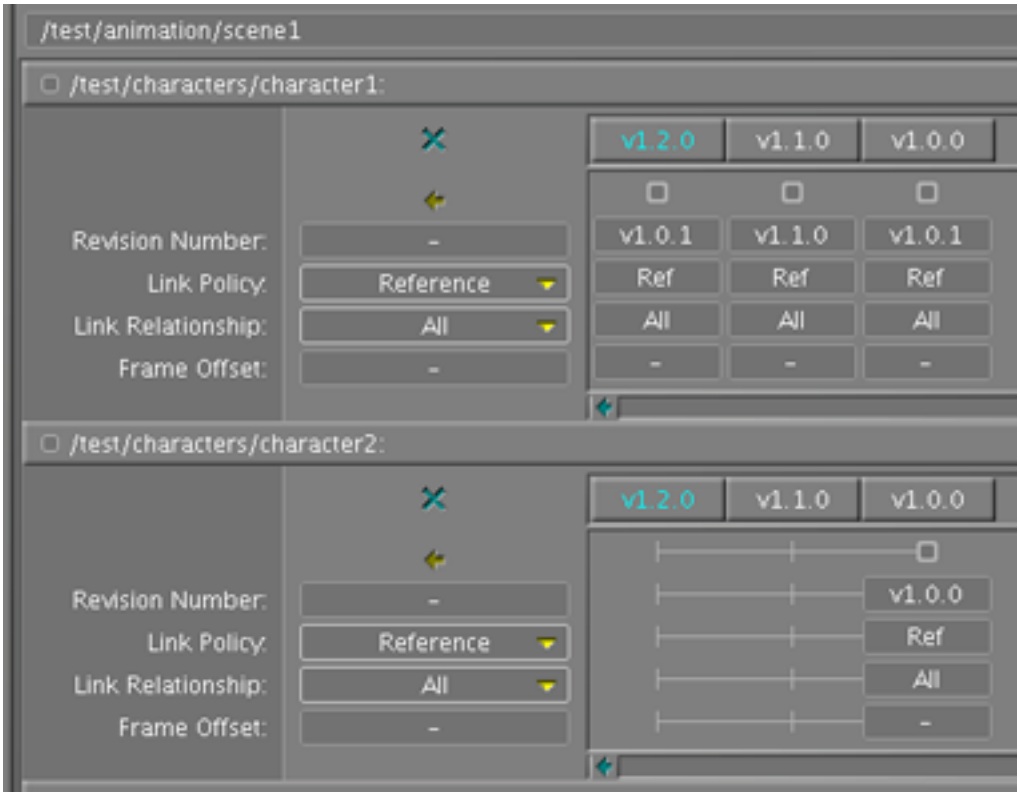
Overwrite All is the most basic and destructive check-out option. Check-out does not blindly copy the latest version of every node in the tree into the working area. Instead, the exact versions of each node at the time that the root node version was created are checked-out. If you have other versions of nodes contained in the tree, they will be overwritten with the versions associated with the root checked-out node. This can result in older versions of nodes being copied into the working area.

This is the check-out mode that guarantees that the entire tree will be in a Finished queue state after check-out. Since everything matches the repository exactly, there is no reason for anything to be requeued. The other check-out option frequently violate this guarantee.

## Resolving Node Conflicts

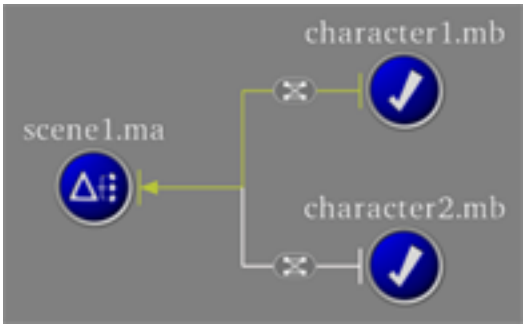
Now say that you wish to make some modifications to (character1). Use the Edit menu item to open the scene, make some changes, save the file, and update the Pipeline status. You'll immediately notice that the node status of the newly edited node has changed to Conflicted. This will happen any time you modify an older version of a node or whenever you modify the newest version of node, but some other user checks in a new version of that node before you can check in your changes.

It is important to note that Pipeline uses a non-locking method of version control. Version control systems come in two main flavors, locking and non-locking. In a locking system, when a user checks out a file, no one else can modify that file. The file is considered locked until the user who locked it releases it. This clearly will stop two people from working on the same file at the same time. However it also becomes possible for people to lock files and not remember to unlock them. In either method, conflict-resolution becomes a central issue. The non-locking methodology provides more flexibility.

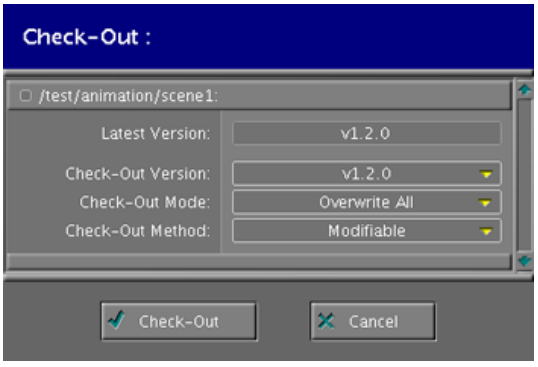



The Node Links panel for (scene1). Note that the current version of (scene1) is shaded cyan in the list of versions.

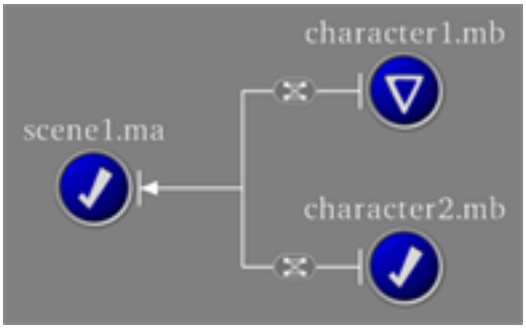
The horizontal lines under (character2) indicate that nothing changed from the prior version.




Now that (character1) has been checked out, (scene1) is in a modified state.




The Check-Out dialog with Check-Out Mode set to Overwrite All.




After the Overwrite All Check-Out. Notice that (character1) has now reverted back to the older version.

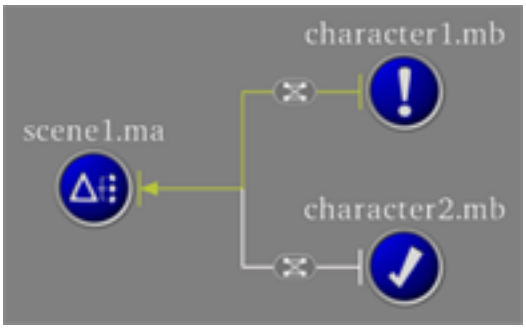
There are times when it might be desirable to have the ability to have two people making changes to a file at the same time. Imagine a situation where one user is engaged in a lengthy modification process to a complex character rig. While they are working on that, an easily correctable error is discovered in the latest checked-in version of the rig. If this was a locking system, then all the other users would have to wait until the lengthy modifications were done to apply a fix. In the case of Pipeline, the fix can be made immediately and users can continue working. Of course this will put the original user's rig node in the Conflicted state. But when the modifications are done and it is time to check-in, they can simply look at the Node History, notice that the newest version was simply a quick fix, Evolve their changes over top of that, and check-in the new rig. No one was inconvenienced and all the users were able to continue doing their work with the least amount of pause.

There are several ways to resolve this problem. In this case, we are going to assume that we want the changes we just made to v1.0.1 to be the newest version, to basically throw away all the changes made in version 1.1.0. To do this, we use the Evolve menu item. Right-click on the (character1) node and select the Evolve option. The drop-down menu will allow you select any version to evolve to. Normally you will want to evolve to the latest version, which is the default.

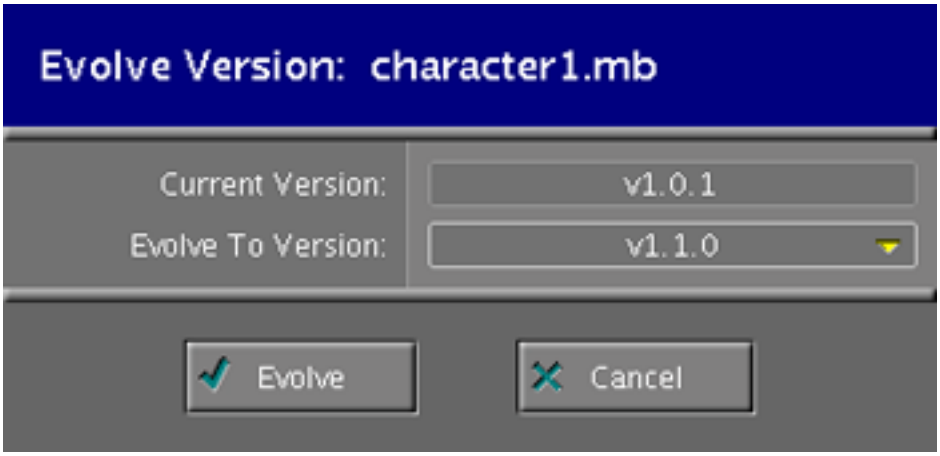
Evolve simply changes the repository version number that your current working version is based upon. If you have a Conflicted node and evolved to anything other than the latest version, it would still be Conflicted!

Click the Evolve button. (character1) is now in the Modified state and you could now check-in the tree if you wished. There is another way to accomplish something similar without having to evolve a node. Check-out the latest version of (character1). Again, lets assume you want to make the version 1.0.1 file the file associated with the latest version. Instead of checking out that older version and having to evolve your changes, load up the Node Files panel for the node instead. Click on the checkbox beneath the v1.0.1 heading and then click the apply button in the upper right corner. This will copy the file from v1.0.1 into your current working area, over-writing whatever file exists there. However, it does not change the version the node is based upon. The node remains based on the latest revision. You can now edit (character1) and then check-in your changes.

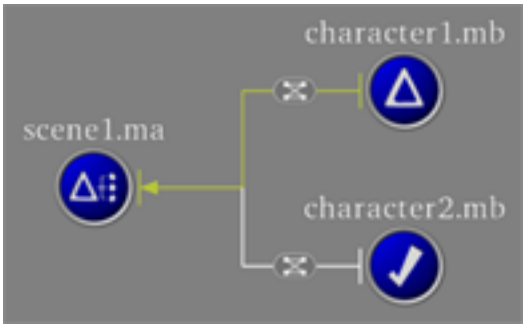
Whether using Evolve or the Node Files panel, it is important to note whatever changes you made in the check-in message so that other people know that you've rolled back to an older version. So when you go to check-in (scene1) make a note in the check-in message that you rolled character1 back to version v1.0.1.




Editing (character1) when it was in the Needs Check-Out state has caused it to become Conflicted. This state needs to be resolved before you can check-in this node.




The Evolve Version dialog. The Evolve To Version dropdown is always set to the latest version by default. There is never a need to change it.




After the Evolve, (character1) is now based on the latest version and is in the Modified state.


Using the Node Files panel to copy an older version of a file out of the repository.

# Check Out (Keep Modified)

Local changes to nodes can be preserved during a check-out by using the Keep Modified mode.

We will now look at an alternative Check-Out Mode called Keep Modified. This mode does not necessarily check-out the exact tree as it was checked-in unlike Overwrite All. Instead, it will avoid checking out nodes which have been modified in the current working area or which are newer than the versions of the node that would have been checked-out.

At this point, (scene1) and all its sources should all be in the Finished/Identical state. Edit (character1), make some changes to it, and save the file. Updating the status in Pipeline will show (scene1) in the Modified Links states and (character1) in the Modified state. Now perform a Check-Out on (scene1) with the Overwrite All option selected. The changes that you had made to (character1) have been lost. This is not desirable.

Make the same changes to (character1) again. After saving the file, perform a Check-Out on (scene1) with the Keep Modified option selected. This time, Pipeline will not overwrite the changes you made to (character1). This is useful in a situation where you where working on (character1) and noticed that another user had checked-in a new version of (scene1). Since your changes to (character1) have not yet been checked-in, the other user would have used a different version of (character1) during their check-in of (scene1). By using Keep Modified, you are able to incorporate the changes made to (scene1) by the other user while preserving your modifications to (character1). The only other way to do this would be to check in your changes to (character1), check-out (scene1) with the Overwrite All option and then check the latest version of (character1) out. Checking out with the Keep Modified option is clearly the simpler approach.

It is not uncommon to see either stale sources nodes with a queue state of Missing after a check-out with the Keep Modified option. Often, nodes will end up in the Stale state due to a unique combination of your local changes and the versions of nodes you checked-out. When a node will require regeneration, Pipeline skips copying the files and removes the working copies since they are no longer valid and will need to be replaced by running the node's Action.

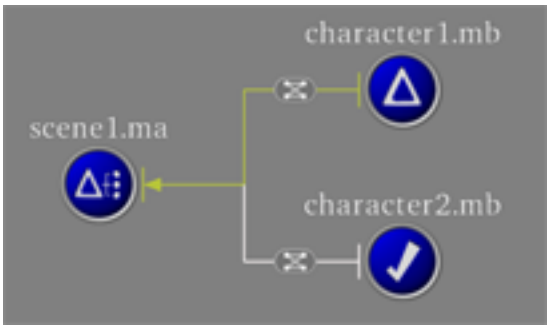
## Frozen Nodes

Frozen nodes are checked-out in a read-only state and access files in the repository directly through symbolic links instead of making a local copy.

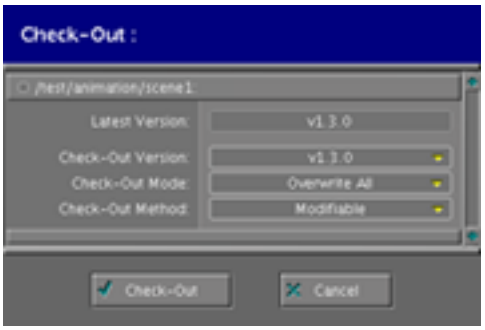
Press (mouse3) on Select (scene1) and select the Check-Out menu item. Set Check-Out Mode to Overwrite All, the Check-Out Method to All Frozen, and press the Confirm button. You'll notice that the nodes are now a less intense white color, an indication that the nodes are Frozen.

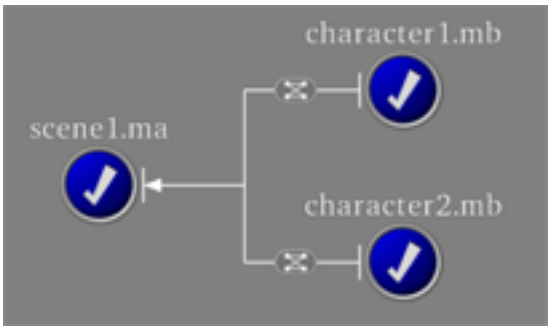
The files for frozen nodes are not copied from the repository into your working area. Instead, a symbolic link is created in your working area which points into the file in the repository. While this makes it impossible to edit the file, frozen nodes can greatly speed-up status updates. Status updates have to check single file associated with every node visible in the Node Viewer to ensure that they have not changed, which can take a fair amount of time for complex node trees. When a file is directly linked to the repository, there is no way it could be changed, so there is no reason for status to check the associated files for modifications. In situations where you are not going to edit a node (or any of its sources) the frozen state is preferable.

If you right-click on a frozen node, you'll notice that there is a reduced menu of commands available. One important thing to note is that you cannot Check-In a frozen node. However, you can Check-In a node with frozen source nodes. To return (scene1) to a form where it can be edited, check it out again with the Overwrite All and the Frozen Upstream options. This will cause (scene1) to be modifiable, but will freeze both the model nodes.

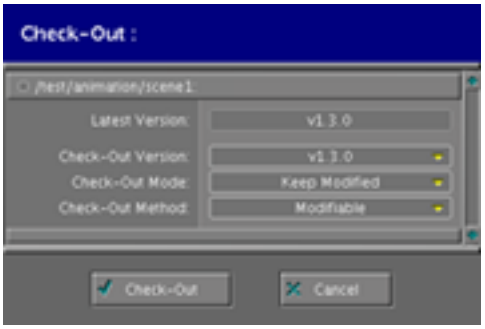


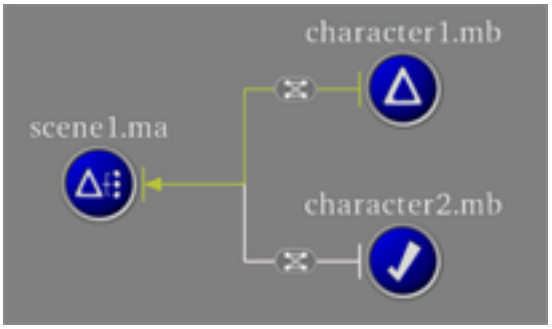

The starting state for the nodes in this example.



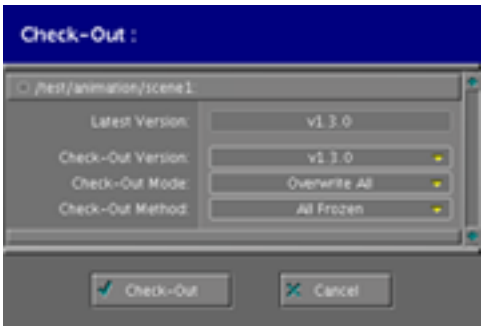



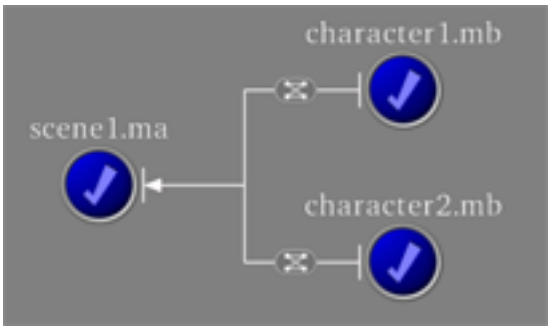

The result of doing a check-out with the Overwrite All option. Notice that the changes that were made to (character1) have been lost.



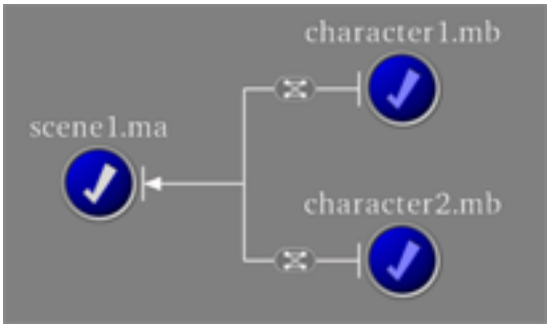




The result of doing a check-out with the Keep Modified option. Nothing has changed in this setup because Pipeline did not overwrite modified nodes.




The result of doing a check-out with the All Frozen option. Notice how the intensity of the nodes has been diminished, providing a visual cue to their frozen state.




The result of doing a a check-out with the Overwrite All and the Frozen Upstream options.

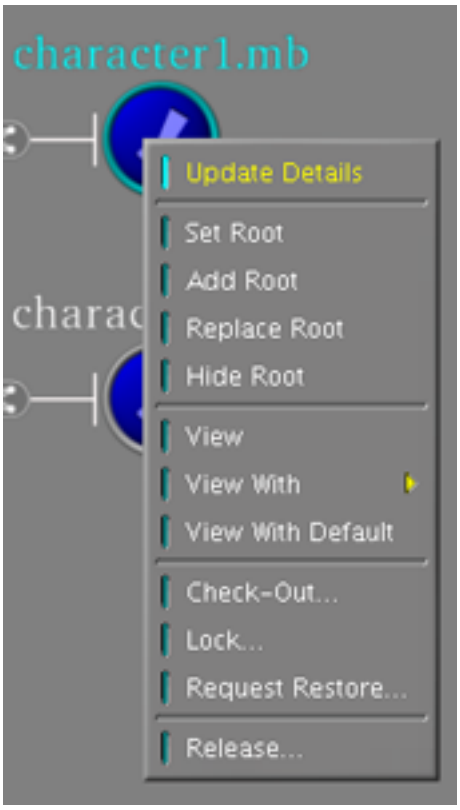
The Frozen Upstream option will check-out the entire tree as frozen, except for the node that you clicked-on, which will be modifiable. This is useful when you want to edit a particular node, but know that you are not going to be modifying any of its source files.

The Preserve Frozen option will cause all nodes that are currently frozen to stay frozen, while making anything new that is checked-out modifiable. This is useful if you have a complex network setup that involves some selectively frozen sources and some modifiable sources: Preserve Frozen will keep those settings while updating the overall version of the network.

To finish up, just run a Check-Out with Overwrite All and Modifiable to unfreeze all the nodes.

Both Check-Out Methods involving freezing can cause unwanted problems when used with Keep Modified. Freezing a node is deciding that you do not want to make any changes and the node cannot be checked-in or queued. However Keep Modified implies, in its very nature, that there have already been changes made among the source nodes. These changes will most likely lead to node staleness.

Stale/Frozen nodes are a common result of checkouts that combine Keep Modified and one of the Frozen options. These nodes cannot be queued because they are Frozen and cannot be checked-in because they are Stale. There are three ways to resolve this issue: check-out the offending node as Frozen with Overwrite All, check-out the node as Modifiable with Frozen Upstream, or just check out the offending node using Modifiable method. While this situation does not cause permanent damage done, it can be a confusing result of freezing nodes and using the Keep Modifiable check-out mode.




The reduced menu of items available for a frozen node in the Node Viewer panel.

Editor's Note: Add Snow flake picture for frozen node here.

## Node History

The Node History panel displays useful information for tracking changes to a node.

The Node History panel is used to review all the different versions of a node and the messages that were entered when they were checked-in. There are several other pieces of information that are also found here. The name in the middle indicates who checked in the particular version. This can be useful when investigating the source of a problem encountered in production. Once you find the version of the node where the problem first occurs, you can use the node history to find out who checked it in and, if the check-in message is descriptive, what exactly was done that might have caused the problem. The information also includes the date and time of the check-in.

It is impossible to emphasis enough the importance of descriptive check-in messages. Quite often multiple versions of a node are created daily. When attempting to track down the source of a problem, the worst approach is having to check every version. Messages which accurately document what was changed in each revision provide important information on where to start looking.

The last piece of information displayed is the Check-In Root. Often a node will get checked-in as part of a larger check-in operation. For example, the (character1) node has been checked-in as part of the (scene1) check-in. The Check-In Root field will always display the root node of the check-in and the version of that root node. If the node you are viewing was the root node, then Check-In Root will be displayed. Otherwise the name of the node that was the root will be displayed.

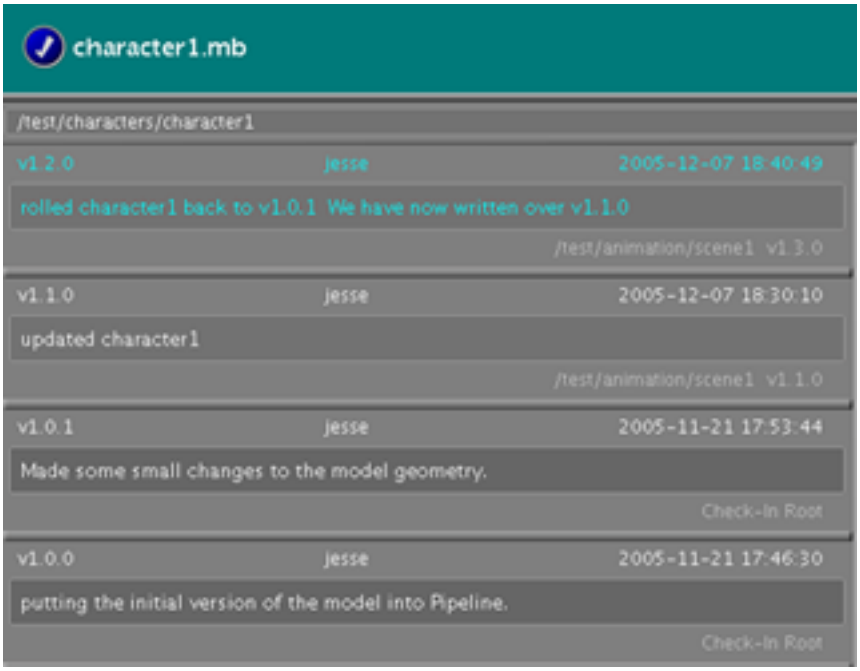
## Check-In Details

New node versions are created by the Check-In operation. This operation is not atomic in nature and may succeed on a subset of the selected nodes.

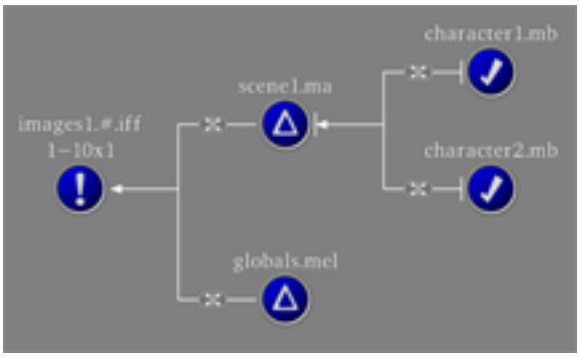
When you check-in a tree of nodes, Pipeline starts at the leaves of the tree and works its way down to the root node. Any nodes downstream of the root node in the Modified, Modified Links, or Pending state will be checked-in as well. All nodes checked-in as part of the same check-in operation will be assigned the same check-in message and same version increment as the root node.

Check-in is not an atomic operation. If there is a problem checking-in one of the leaf or branch nodes, the check-in operation will abort. When this happens, some of the nodes will have been checked-in and will be in the Identical state, while others may still be Modified or Conflicted. This is not a big deal. You simply have to fix the cause of the problem and then check-in the remaining nodes.

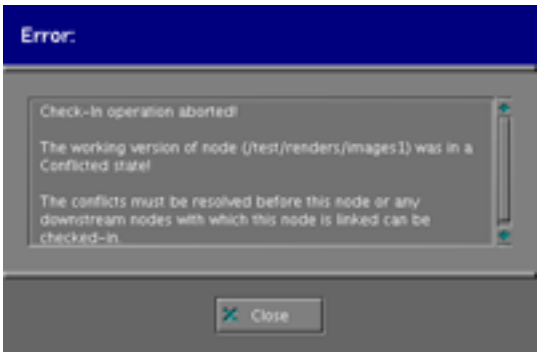
What might cause the check-in to fail? There are numerous conditions which can cause a check-in to fail. If a node has any jobs which are in any state except Finished it cannot be checked-in. If a node is missing even one file, it cannot be checked-in. If the node is in a Conflicted state it cannot be checked in. A Modified node that is Frozen cannot be checked in.



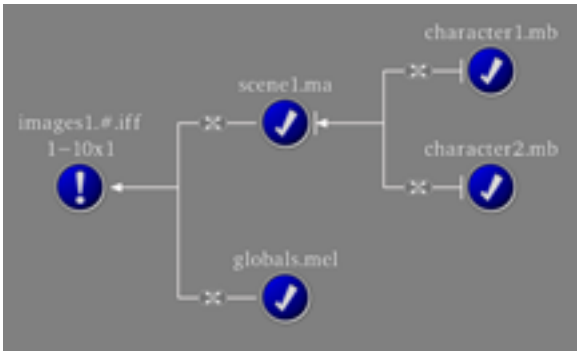

The Node History Panel. The version of the node in your working area is highlighted in cyan. For versions 1.0.0 and 1.0.1 (character1) was the root of the check-in. For the last two versions, (scene1) was the root. All versions where checked-in by the user (jesse).




What happens if you attempt to check-in (images1) even though it is Conflicted?




The error message you get telling you that you cannot check-in a Conflicted node.




Notice that both (scene1) and (globals) were checked-in, even though the check-in of (images1) failed. This is because Pipeline starts at the leaves of the tree and works its way up to the root node.

# More Node Viewer Operations

## Renumbering Sequences

The frame range of the file sequences associated with a node can be changed using the Renumber operation.

Right now the files associated with the (images1) node have a frame range of (1-10x1). You can use the Renumber command to change the range. Before beginning, check-out the latest version of the (images) node with the Overwrite All option. Press (mouse3) on the images node and select the Renumber menu item. This will bring up the Renumber Node dialog. Here you can change the start and end frame of the sequence as well as the step. There is also the Remove Obsolete Files option. If you set this to YES, then Pipeline will remove any frames outside the current range from your working area. If you had a sequence that was 1-10x1 and you changed it to 1-5x1 with that option set to YES, then frames 6-10 would be deleted. Otherwise those extra frames are left in your working area, but Pipeline ignores them. In this case set the range to 1 to 20 by 1. When you go back to the Node Viewer, you'll notice that (images1) is now in the Missing node state. The frames we just added do not exist and even one missing frame will cause a node to be in the Missing state. Open up the node in the Node Files panel if you wish to see which files are missing.

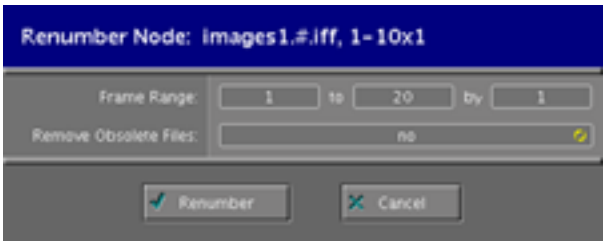
Queue the node and check-it in when it is finished running. Now renumber it again, this time as 1 to 5 by 1. Load the (images1) details and look at the Node Files panel. The first 5 images in the list have status symbols next to them. These are the frames that are in the node's current frame range. All the frames with Undefined statuses are frames that exist in at least one checked-in version, but not in your current working area. The checkboxes under each column show which files were in each version. For example, version 1.0.0 only has checkboxes for frames 1-10 because that was the frame range when that version was checked in. The checkbox with a line through it means that the frame does not exist in the current working area. Just like with the single file earlier, you can click on any of the normal checkboxes to copy frames out of the repository into your working area.

## Node Locking

Node Locking provides a way of using some types of node versions without being exposed to the details of how they where created.

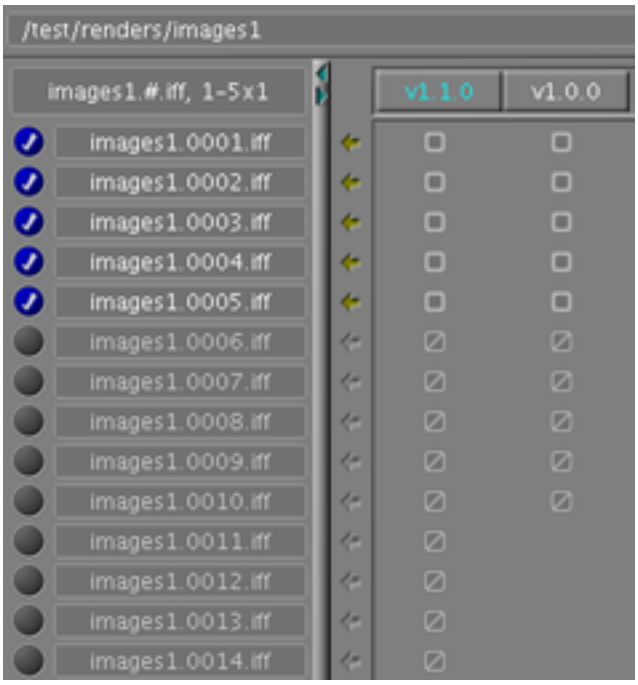
Node Locking allows users to work with nodes that share dependencies, even if the desired versions of those nodes are based on different versions of the source nodes. One possible situation like this is when a compositor wants to work with latest version of a series of renders. However those renders do not all share the same versions of their source nodes. If a model changes it is often not possible to re-render every shot that uses that model in a timely fashion. But if the compositor attempted to check-out all those renders, a majority of them would be stale preventing the compositor from checking in their work until the Stale nodes where regenerated. In a situation like this, the compositor does not need to know about the entire tree that was used to generate those images, only the images themselves. Node Locking provides a way to ignore a node's entire tree of dependencies and simply use a repository version directly without checking-out these upstream nodes. In our example, the compositor can simply lock all their image nodes and then update them accordingly as other artists check in new versions without ever having to worry about how these images where created and whether they have any shared dependencies that would force regeneration.

Load (images1) into the Node Viewer, right-click on it, and select the Lock Node menu item. The Lock Dialog will pop up, allowing you to select which version you want to lock the node to. Select the latest version and click on the Lock button. All of the node's sources will disappear, the node is now shaded like a Frozen node, and the lock icon appears to the right of the node. To change the version that is being pointed to, simply choose Lock Node again and pick a different version. To unlock the node, simply perform a check-out. Go ahead and check-out the latest version of node now to unlock it.

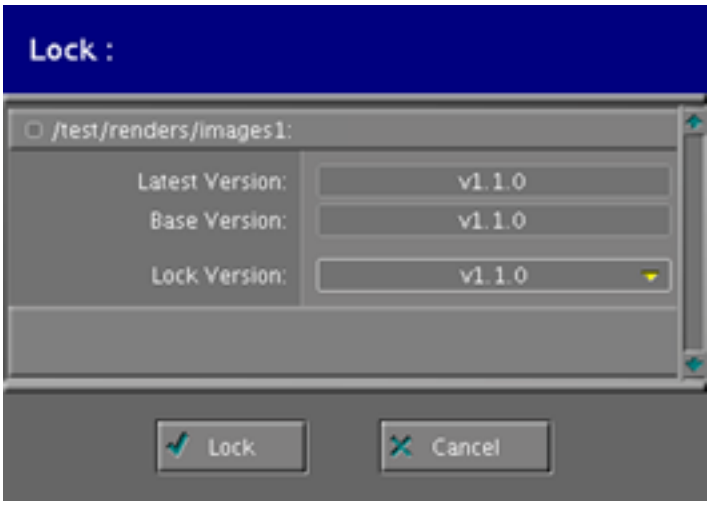




The Renumber Node dialog and the Image node after the renumbering when it is now in the Missing state.




The Node Files panel for the images node. The frames with an actual Status icon are the ones included in the frame range of your current working version. All the other frames exist in repository version but not in the current version.





The Lock dialog. The Lock Version allows you to select which repository version you want. And the (images1) node after it was locked. The image of the padlock to the right of the node provides a visual clue that the node is locked.

You can only lock nodes which either have no sources or which have only dependency links to their sources. Reference links imply that the node actively makes use of that source. If you are ignoring all a node's sources, like locking does, there is no guarantee that the node's sources are the right version or that they even exist in the current working area. This could lead to queue failures or other anomalous behavior. For that reason, node locking is not an option for all nodes.

Locking a source node can lead to a downstream node's state changing to Modified Locks. Lock the (globals) file and notice how the status of (images1) changes. Locking information is stored in the repository, so if you check-out a node which was checked-in with locked sources, those sources will be locked when someone checks them out. Therefore locking a source node makes it target appear in the Modified Links state, letting you know that you've changed some of that node's metadata. Check-out (globals) to return it to its unlocked state.

Node locking should be used sparingly and as a last resort. When nodes are locked, the details of how they where created are hidden. This may be necessary in some cases, but generally is a bad idea that can lead to inconsistent results when the locked nodes have some shared dependencies. For instance, consider two render passes which use the same model node as part of the scene being rendered. If the rendered nodes are locked, they might not be using the same version of the model. This could lead to a situation where they rendered images don't line up in the composite. With the nodes are locked, there is no simple way for the compositor to know that the model versions are different. For this reason, you should use locking only when it saves considerable time or processing resources but realize that these savings are at the expense of consistency.

## Node Details

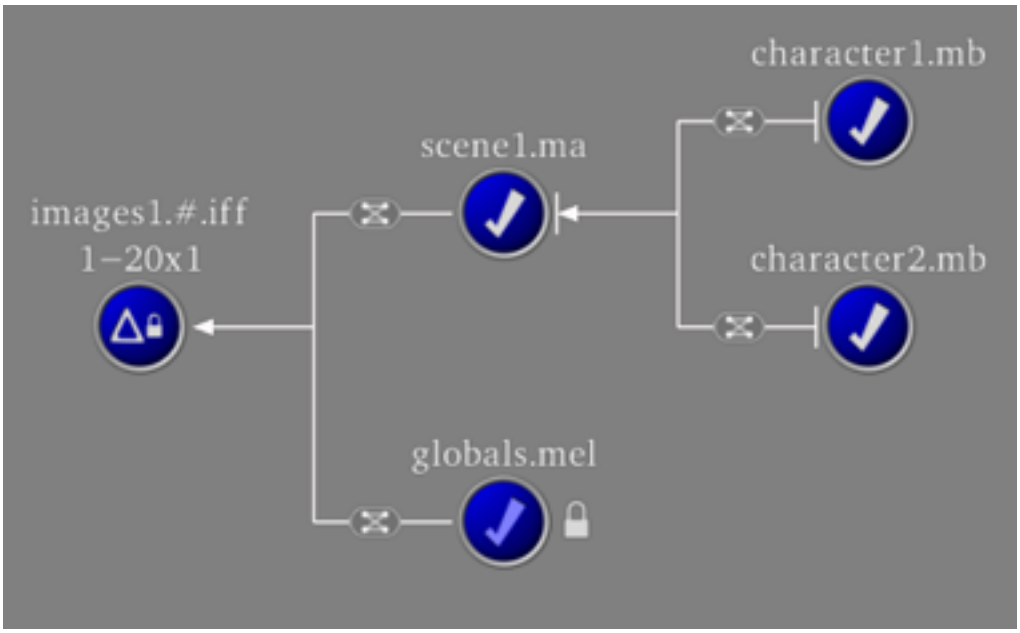
The Node Details panel lets you adjust the Node metadata and copy parameters from checked-in version of the node into the current working version.

The Node Details panel allows you to view and edit the metadata information associated with a node. It also allows you to view the settings that were used in any repository version and even copy those settings to the current version. Select the (globals) node and load its details. Make some changes to your globals. In this example, the Anti-Aliasing settings were changed. Anything that you've changed from the last checked-in version will show up in blue. Apply your changes and queue the node. Once it finished running, check-in it in.

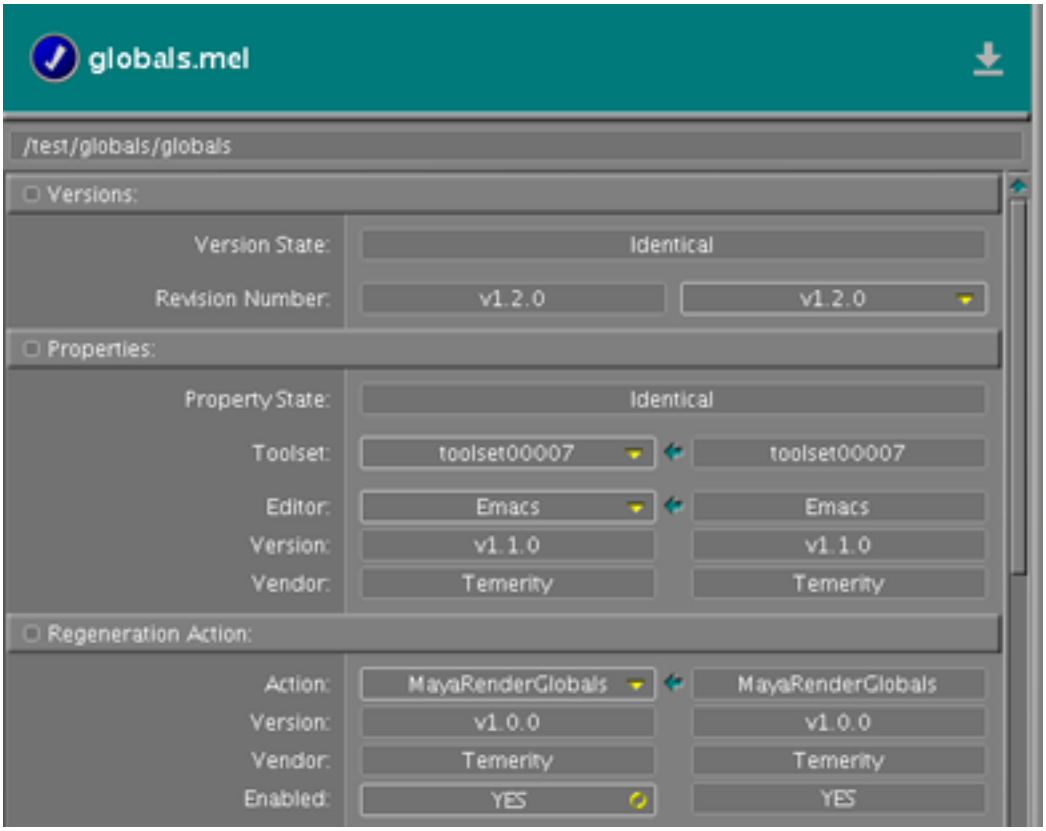
The top part of the Node Details panel displays the name of the primary file sequence and frame range, the full node name, and a small icon representing the status of the node. The rest of the panel is divided into three columns. The leftmost column is simply a key, providing a description for each of the fields. The middle column shows the node properties for the working version of the node. The right column displays the node properties for any repository version of a node: this makes it easy to compare the current node settings against the settings used to generate another version. To switch between different repository versions, simply click on the dropdown menu at the top of the rightmost column in the Revision Number row. This will display a list of all the versions that exist in the repository. Selecting any of them will load those settings into the righthand column.

Use that dropdown menu to select version v1.0.0 of the globals. You'll see some of the parameter values in the righthand column change values. Action Parameters which differ between your current working version and the repository version being displayed will be shaded teal.

It is also easy to copy settings from a repository version to the current version. Many of the fields have green arrows between the current version and the repository version. Clicking that arrow will cause the value of the field in the repository to be copied over the current value. It is important to note that this can cause changes to more fields than just the current field. If an action is copied from a repository version, then all the action's parameters are copied as well. Go ahead and copy some




(images1) in the Modified Locks state after globals is locked. Even though the link between the images and the globals is a dependency, changing the locked state does not make (images1) stale.




The Node Details panel, showing the different sections. The Properties Section contains the toolset and the Editor assigned to the node. Underneath that is the Action and all the Parameters that pertain to that Action.


Selecting the version to view in the righthand column.

parameters from the older version to the newer version. Just like all changes to parameters these modifications are not saved until the Apply button is clicked. To return the node to its original state simply perform a status update in the Node Viewer before applying any changes. This will reload the Node Details panel, removing any unsaved changes.

The topmost piece of information provided is the Version State. This compares the version number of the working version against the latest version in the repository and indicates whether the node is Identical, Needs Checkout, or Modified. Underneath that is the Revision Number.

The next section is Properties. The first field indicates the state of the properties. If this reads Identical, then all the node properties of the working version are identical to the node properties of whatever repository version is selected on the right. Below that is the Toolset field. When the dropdown menu is clicked in the working version, you can select a new toolset from a list of all the Active Toolsets. The list will also contain the current toolset, even if it is not Active. However, once you switch the toolset away from that older toolset and apply the changes, that older toolset will no longer be available as a choice.

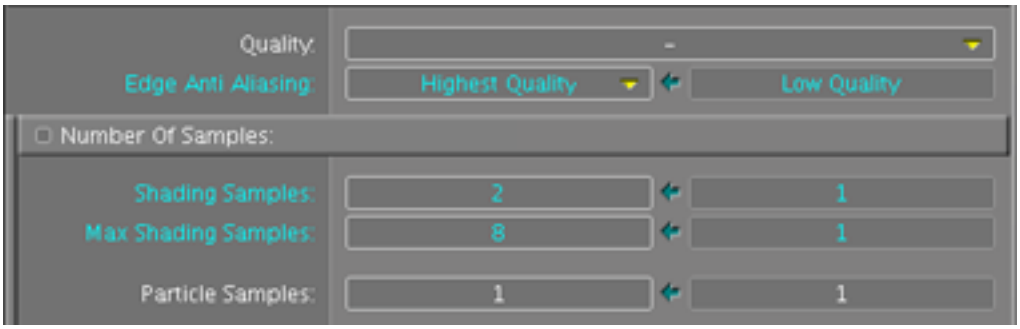
When you change the toolset it can cause the current Editor or Action to become invalid. If that is the case, then the appropriate field in the Node Details Panel will turn Purple to indicate that it is invalid. You will not be able to queue or check-in the node when it is in this state. You must either choose a new toolset that supports the Action or choose a new Action that is supported by the toolset.

Underneath that is the Editor field. When this is clicked on, it will display a list of all the Editors that are associated with the current toolset. When a valid editor has been selected, then the name of the editor plug-in (which maybe different from the menu item) will be displayed in the editor field and the version number in the version field.

Below this is the Action Section. Actions, as discussed earlier, are plug-ins responsible for regenerating stale nodes. Like Editors, the list of available Actions is determined by the node's toolset. Selecting an Action from the list will cause the action parameter section to change to reflect the set of parameters that the new Action defines. Underneath the action and the action version is the Enabled Check-Box. A node with an enabled action will become stale and will not be writable by the user. A node with a disabled action will not become stale and will be user-writable, but will never be regenerated by Pipeline. Below this are all the action parameters. Actions can define as many parameters as they need to: there are six different categories of parameters.

- Boolean Parameters: The value will either be (YES) or (no) and it will be displayed as a clickable field. Clicking on the field will flip its value.
- Double Parameters: The value can be any real number and it will be displayed as a text field. Any non-numeric entries will be ignored.
- Enumerated Parameters: The value will be selected from a plug-in specified list of options and will be displayed as a dropdown menu containing all possibilities.
- Integer Parameters: The value can be any integer and will be displayed as a text field. Any non-numeric entry will be ignored and any decimal value will be thrown away.
- Link Parameters: The value will either be null or any of the sources of the node. It is displayed as a dropdown menu containing a null value and then a list of the names of the primary file sequences of all the node's sources.
- String Parameters: The value can be any string and is be displayed as a text field.

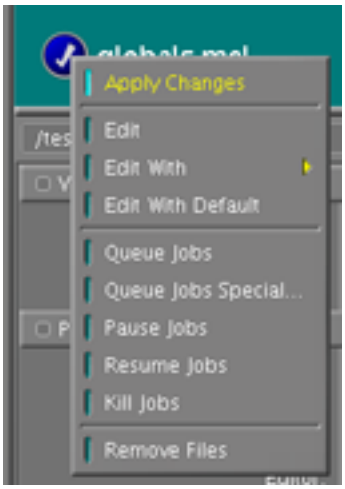
Pressing (mouse1) on the Node Status Icon in the upper left hand corner of the details panel brings up a menu that gives you access to the Node Details menu. This menu allows you to apply your changes (same as clicking on the white arrow), edit the node, control its jobs, and remove its files.




Parameters which are different between the left and right columns are shaded in teal.


Clicking the arrow will copy a value from right to left.


A Test Action showing all the different type of parameters. From top to bottom: String, Double, Integer, Link, and Enum.




This menu is accessible at the top of the Node Details panel by right-clicking on the status icon. It contains a subset of the commands available in the node menu in the Node Viewer.

## More Cloning

When cloning a node you can include its Action, Action Parameters and links.

The Clone tools allows you register a new node based on an existing node. Select the (images1) node and check it out with the Overwrite All. Then right-click on it and select the Clone menu item. The Register Cloned Node dialog that appears is similar to the Register dialog.

In the Filename Prefix textfield, change the name from the default (/test/renders/images1-clone) to (/test/renders/images2). Underneath that you can make changes to the Frame Range and Padding. Notice that you cannot change the File Mode or the Frame Numbers fields.

The Copy Primary Files setting is one you'll usually want to pay attention to. Setting it to (YES) will copy the primary files from the existing node to the new node. This is great in situation where you are not going to regenerate the files. However if your reason for cloning the node is to create an alternative version and you are going to be regenerating the files anyway, it is not worth copying the files. In these cases make sure to set this option to (no).

The Clone All Properties switch allows you to quickly set all the option boxes that appear in the lower part of the window. These cover all the properties of the node, including toolset, editor, action, action parameters, queue settings, and links. Setting Clone All Properties to (YES) makes everything below it (YES), while setting it to no makes everything below no as well. Once this is done, it is possible to go down the list by hand and adjust individual settings to your liking.

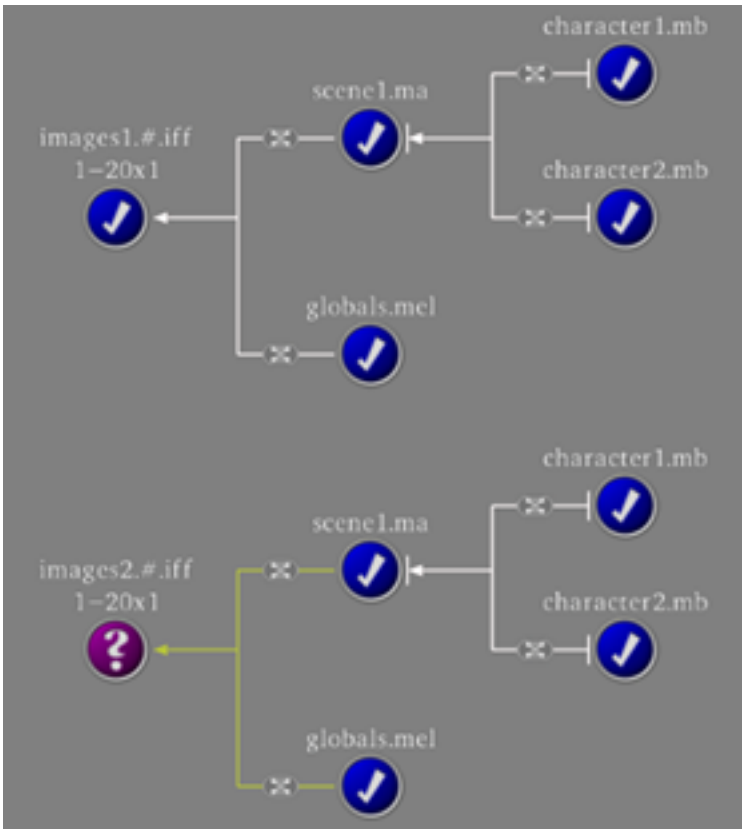
At the bottom of the dialog is a list of all the node's sources. Setting these to (YES) will cause the new node to be linked to the same nodes as the existing node. If you set one to (no) then that link will not be present in the new node. In this case the defaults are fine. Click the Confirm button. You'll now see the new (images2) node appear in the Node Viewer with the same links as (images1). If you open (images2) up in the Node Details you'll see all the parameters are identical as well.

If you click the Apply button to register the cloned node instead of the Confirm button the dialog will stay open and you can clone more nodes.

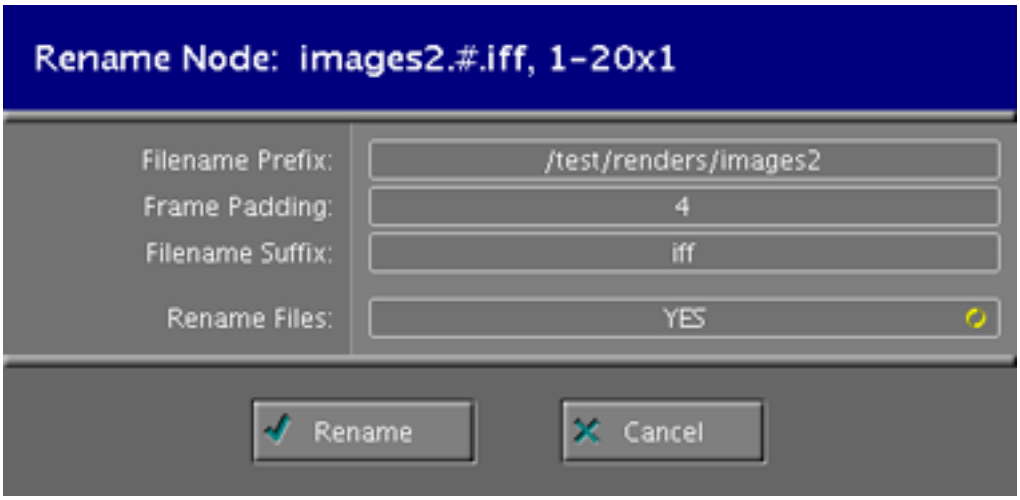
## Renaming a Node

You can use the Rename Node menu item to rename a node that hasn't been checked in.

Selecting the Rename menu item from the Node Menu will open up the Rename Node dialog. Only nodes which have not been checked-in can be renamed. The rename screen will let you enter a new prefix which can have the effect of changing the node's directory depending on what part of the prefix you change, a new suffix, and a new padding setting, provided the node uses frame numbering. The last option is the Rename Files option. With this set to (YES), any files which currently exist for the node will be renamed to match the new name. If it is set to (no), then the old files will be left where they are and the node will now be in the Missing state.




The Node Viewer after the node has been cloned. Since we left all the link options set to 'YES' the links are identical.




The Rename Node dialog.

## Export

The Export menu item allows you to copy properties from one node to a group of other nodes.

Export allows you to copy properties from one node to other nodes. It can be as simple as copying an action from one node to another or as complex as copying an Action, all its parameters, queue settings, and links from one node to a group of other nodes.

When using the Export tool, the last node selected is the target node and is always the node whose settings are being exported. All the other nodes are the source nodes and will receive the exported information.

Register two additional image nodes, (/test/images/images1_globals1) and (/test/images/images1_globals2), with the same length as (images1). Select both the new nodes and then right-click on the (images1) node and select Export to open the Export dialog. The Export dialog is quite similar to the Clone dialog, except for the fact that it does not include controls for renaming the nodes.

The Export All Properties button allows you to quickly set all the option boxes that appear in the lower part of the window. These cover all the properties of the node, including toolset, editor, action, action parameters, queue settings, and links. Setting Export All Properties to (YES) makes everything below it (YES), while setting it to (no) makes everything below (no) as well. Once this is done, it is possible to go down the list by hand and adjust individual settings to your liking. In this case, the defaults are fine, so just press the Export button.

You'll notice that the two image nodes now have the same links as the (images1) node. If you load up either new node in the Node Details panel you'll see that all its Action Parameters are identical to the original's as well.

## Releasing a Node

Releasing a node removes it from the current working area.

To get rid of those two new image nodes, select both of them and then select the Release menu item from the Node Menu. The Release Multiple Nodes dialog will appear. It has one option to it, asking if you wish to remove the working files. Selecting (YES) will cause all the file sequences associated with the node to be deleted from your working area. Selecting (no) will remove the node, but will leave the files where they currently are. In our example,since these nodes have never been checked in they will be completely gone from Pipeline after you release them. Press the Release button and Pipeline will prompt you to ensure that you want to release these nodes. Press Yes and the nodes will be removed. Release the (images2) node as well. Since it is a single node, this will display the Release Node dialog, which is identical to the Release Multiple Nodes dialog.

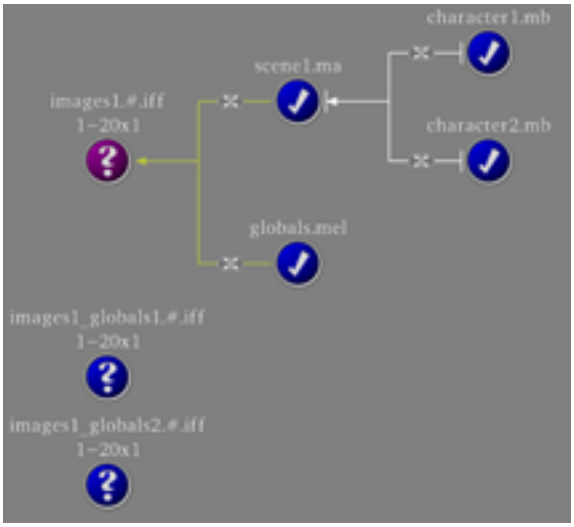
After a node has been checked in, releasing it will only remove your working copy of the node. It will not effect the versions in the repository.

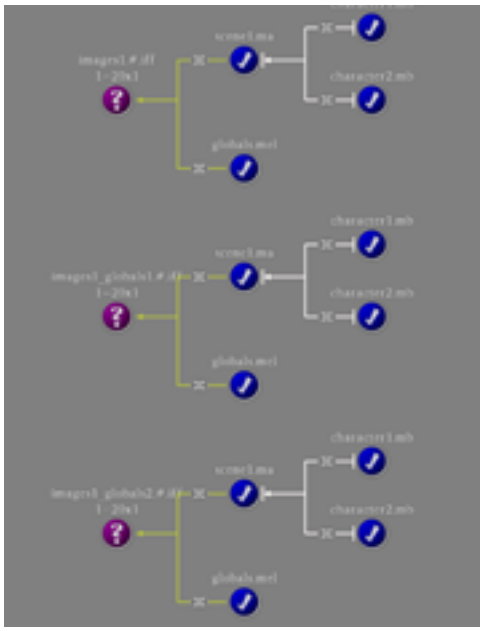
> Releasing an unchecked in node is the only way for ordinary users to remove nodes from Pipeline: once something is checked in only an admin can delete it, which makes it important to ensure correctness when you check in a node for the first time.

## Deleting a Node (Administrator)

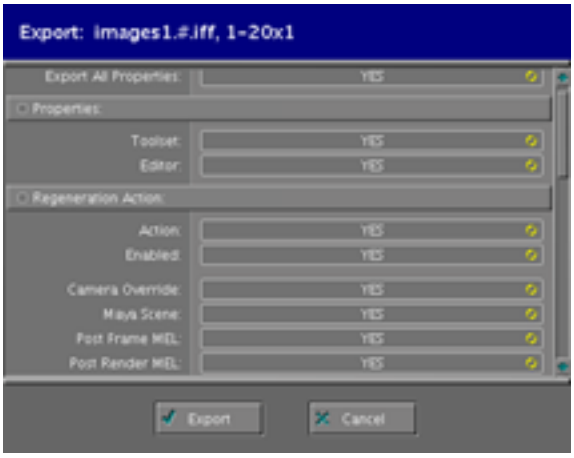
Deleting a node removes a checked-in node completely from the Pipeline system.

The Delete Node menu item is found at the bottom of the Node Menu. Selecting this option on a node will remove the node completely from the Pipeline system. Only nodes which have never been used as source nodes can be deleted. This option is only available to administrators and is not undoable, so it should be used with extreme caution. The primary use for this option is to remove nodes that were registered with incorrect names.

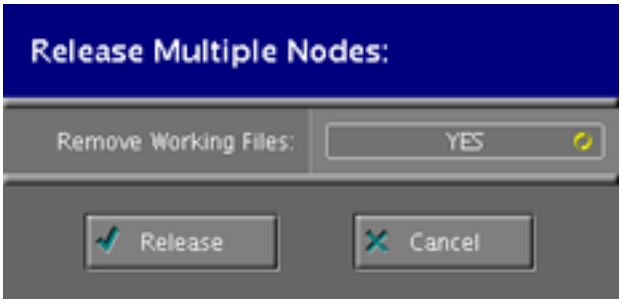


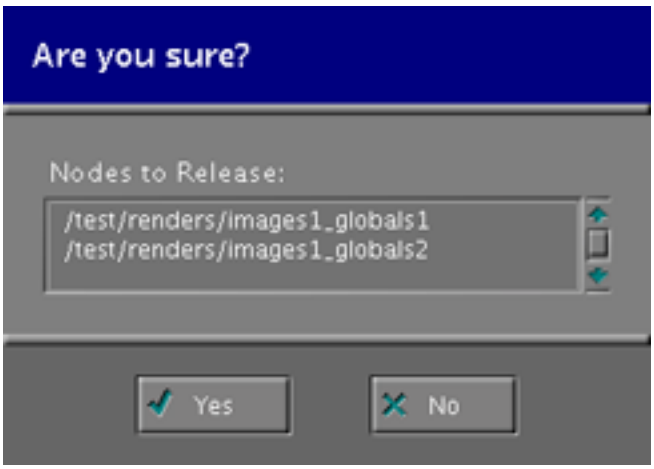
The Node Viewer before and after the Export. Notice how the links have been exported to both the new nodes.




The Export dialog. The Export All Properties button will set the value for all the fields below it.




The Release Multiple Nodes dialog.




Double-checking that you really want to do this.

## Removing Files

The Remove Files command is a way of making a node stale when its files were incorrectly generated, but the program used to generate them reported no problems.

There are times when it is desirable to delete some or all of the files associated with a node. One way Pipeline handles this with the Remove Files option on the node menu. This option deletes all the file sequences associated with the node from the current working area. Removing files from any node with an action automatically cause that node to become Stale. Removing files from a node without an action can be dangerous, because there is no way to regenerate the removed files. There are several useful applications for this option. Go ahead and remove the files from (images1).

The primary use for this tool is removing files that Pipeline thinks have been generated correctly but which in actuality are not correct. This can be the result of a rendered frame where several buckets did not finish correctly, but the renderer did not report an error. It is impossible to just re-queue these files, since they have a queue state of Finished. Removing the files makes the node stale and allows Pipeline to regenerate the files. It is often the case that only a small number of the files are bad and need to be removed. In this case it is more efficient to remove the individual files using the Node Files panel.

The Remove Files option is not undoable and should be used carefully.

## More Node Files Panel

The Node Files panel allows more finely grained control over a node, letting you queue and remove individual frames rather than the entire node.

The Node Files panel allows for various actions to be performed upon a subset of the node's files. Individual files can be selected from the list on the left side by clicking on them. Standard file selection controls apply for selecting multiple files, with shift-clicking allowing the selection of a contagious range and ctl-clicking allowing individual selection of multiple files. When you select a file, its name turns yellow. Once the desired frames are selected, right-clicking on any of the selected frames will bring up the options menu.

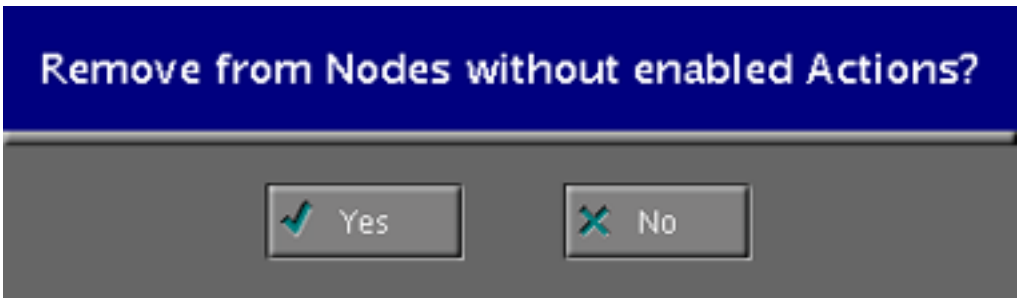
Load the details for (images1). Right-click on the first file in the list and select Queue Jobs from the menu. You'll notice that only that file is submitted to the queue: all the others stay in their stale state. Once the job finishes, right-click on the first file again and select the Remove Files menu item. Now use the (Shift + mouse1) to select files two through six. Right-click on the first file and notice that the first file turns yellow as well. Select the Queue Jobs option again

More information on the options available under Queue Jobs Special can be found later in this document in the Queue section.

Queue Jobs and Queue Jobs Special cause the currently selected frames to be submitted to the queue for rebuilding. Queue will use the submission options specified in the Node Details, while Queue Jobs Special will pop up a small window allowing the submission options to be over-ridden.

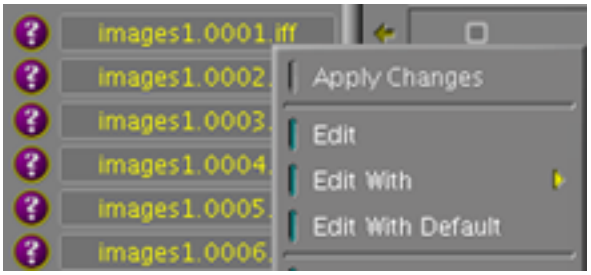
Pause Jobs, Resume Jobs, and Kill Jobs all operate on any jobs that contain the selected frames. If you run Kill Jobs on a single frame, for example, that is being rendered as part of a batch, then the entire batch will be killed. Remove Files causes all the currently selected files to be deleted from the current working area. This action is not undoable: anything that is removed is permanently deleted.

Right-clicking on any of the checkboxes that denote repository versions brings up a menu allowing that file to be viewed or compared. View, View With, and View With Default all work the same as the trio of Edit commands discussed earlier. The only difference is that since the files being viewed are in the repository none of them are writable. Below the view options is the Compare With menu item. This allows the selection of any of the Comparator plugins associated with the node's toolset and will compare the selected version of the file against the current working version.




If you attempt to remove files from a node without an Action, Pipeline will prompt you before actually deleting the files. It is rare to remove files from a node without an enabled Action, since there is no way to regenerate those files.


The selected file in the Node Files panel is highlighted in yellow. This shows the full menu of options available for effecting selected files.




Multiple selected files are all shaded yellow.

### Batching when Queuing Individual Files

How the individual files are batched depends on two things: the order of the selected files and the submission options set on either the node or in the Queue Jobs Special panel.

- Serial When the submission type is serial, queuing any file will cause the entire sequence to be queued.
- Subdivided: In this case, the job will be submitted as if it was set to Parallel(1).
- Parallel(x): x represents the size of the batch. If all the selected files are contiguous, then they will be submitted in batches of size x or smaller. If the files are not all contiguous, then any contiguous files will be submitted in batches of size x or smaller and non-contiguous files will be submitted as individual jobs. For example, assume the batch size equals 5. If 10 contiguous files are selected, they will be submitted as two batches of 5. If 3 contiguous frames and 2 non-contiguous frames are selected, they will be selected as batch of 3 and two batches of 1. If 7 contiguous frames and 2 non-contiguous frames are selected, they will be submitted as a batch of 5, a batch of 2, and 2 batches of 1.

## Comparators

Comparators are another group of plug-ins for Pipeline, similar to Actions and Editors. The purpose of Comparator plug-ins is to compare one version of a file against another. You might want to compare to different versions of a script in xdiff to see where changes were made or compare two versions of a texture in gimp. Comparators are selected through the Compare With menu item found in the Node Files panel. When you right-click on a repository version in the panel and select a Comparator it will compare the currently selected repository version again the current working version. As with Actions and Editors, the Pipeline API supports adding your own Comparators, allowing you to create your own methods of comparing two files.

# More Node Links Panel

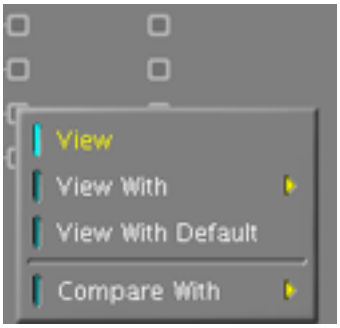
The Node Links Panel can be used to copy information about links from repository versions to the current working version.

Just like the Node Files panel can be used to copy files from the repository to the current working area, the Node Links panel can be used to copy link information from a checked-in version to the current working version.

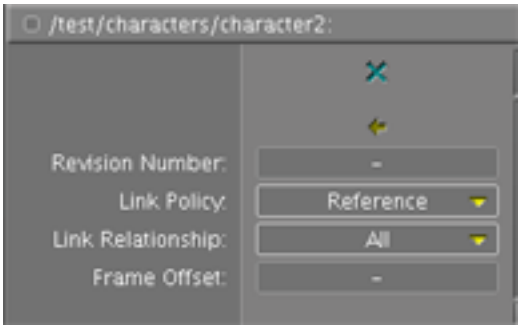
Load the (scene1) node into the Node Viewer and check it out with the Overwrite All option. Make sure you have a linked Node Links panel open and then update the details for (scene1). Underneath each link in the panel is a small teal 'X'. Clicking this 'X' will break the link. Go ahead and click the 'X' underneath the (character2) node. Notice that in the Node Viewer (character2) is no longer linked to (scene1). Also notice that in the Node Links panel that there is no information listed in the left hand column under (character2). This indicates that the link does not exist in the current working version.

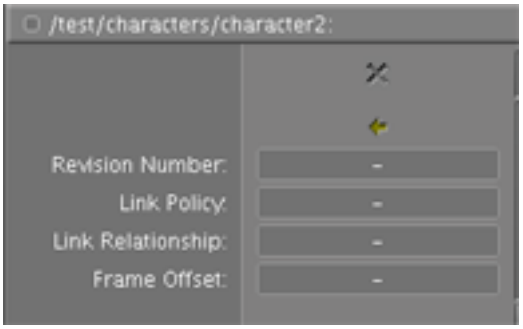
To get the link back, we can copy the link information from an older version of the node. In the Node Links panel, click on the (character2) checkbox underneath v1.0.0. The arrow in the left column turns yellow, indicating that you are copying information for that link. Click on the apply button in the upper right hand corner of the panel. The restoration of the link will be visible in the Node Viewer.


After the queue command, those 6 images queue status is now Pending, represented by the Teal color.

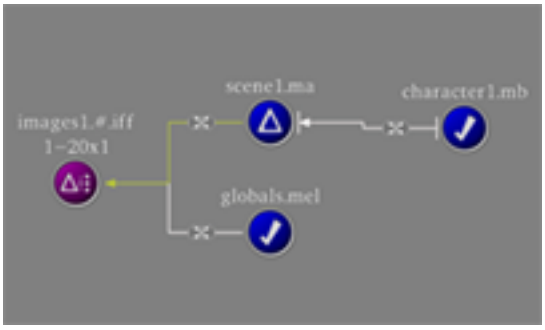



The menu available when right-clicking on one of the checkboxes representing a repository file.

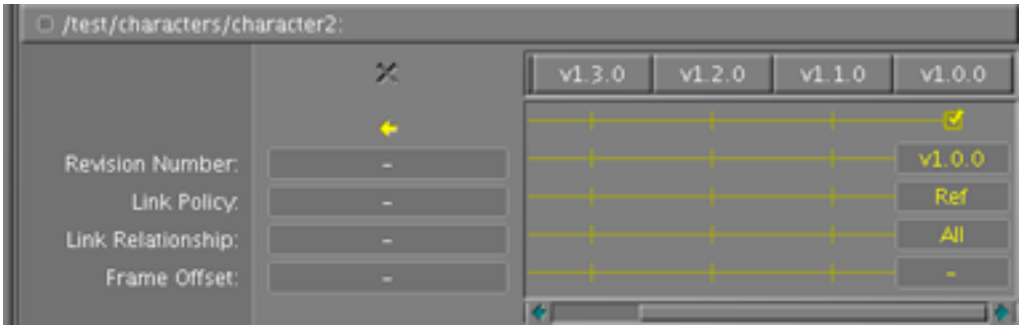



The (character2) entry in the Node Links panel before and after the 'X' was clicked. Notice that all the information about Link Policy and Relationship vanishes. This indicates that the node is no longer linked in the current working version.




The Node Viewer after the link has been broken




Node Links, showing a repository version of the link selected and ready to copy to the current working version.

Copying information about links from the repository only copies information about the link, not the actual verison of the node used in the link. For example, if the checked-in version of the node was linked to version 1.0.0 of a source, Pipeline will not check-out version 1.0.0 of the source node when you copy the link. Instead it will just link the current working copy of the source node, whatever its version. The only information that the Node Links panel copies is the existence of the link, the link policy (Reference or Dependency), the link relationship, and the frame offset.

## Adding and Removing Secondary File Sequences

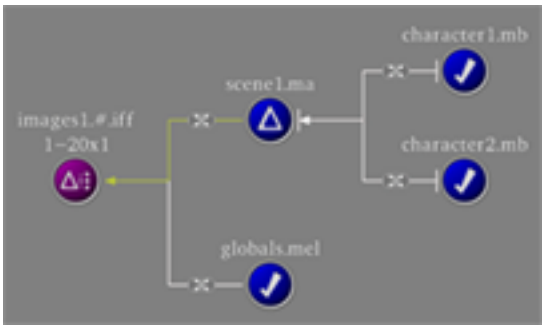
You add and remove secondary sequences using the menu items in the node menu.

Selecting the Add Secondary Sequence menu item from the Node Menu will cause the Add Secondary File Sequence dialog to appears. Go ahead and do this for the (images1) node. The Add Secondary dialog fills in all the fields relating to sequence length, padding, and extension based on the primary file sequence. You need to fill in a Filename Prefix. This can be any name you wish that does not conflict with another node. It can share the same prefix as the primary file as long as you change the suffix. In this case, we can just register a secondary file sequence named (images1_diffuse).

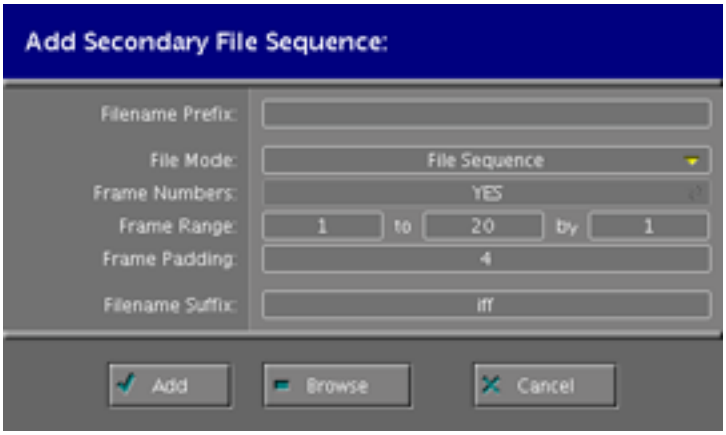
If you look in the Node Files panel, you will see that the new sequence is now listed underneath the primary sequence. The node is stale because the new files needs to be generated.

If you change the frame range of the sequence, you have to make sure the new frame range has the same number of frames as the primary sequence. The exact range and step do not matter as long as the totally number of frames matches.

Since the MayaRender action is not set up to actually generate secondary sequences, if we queued the node now, the Action would run and the node would stay in the Missing state since none of the secondary files would be created. Because of this, lets remove the sequence. Right-click on the node again and select the Remove Secondary menu item. This will display a submenu listing all the secondary sequences associated with the node. Select the (images1_diffuse) sequence from the submenu and it will be removed. Remove Secondary does not cause the files to be deleted from disk, however. The only way to remove those files using Pipeline is in the Node Files panel or by selecting the Remove Files command from the Node Menu.

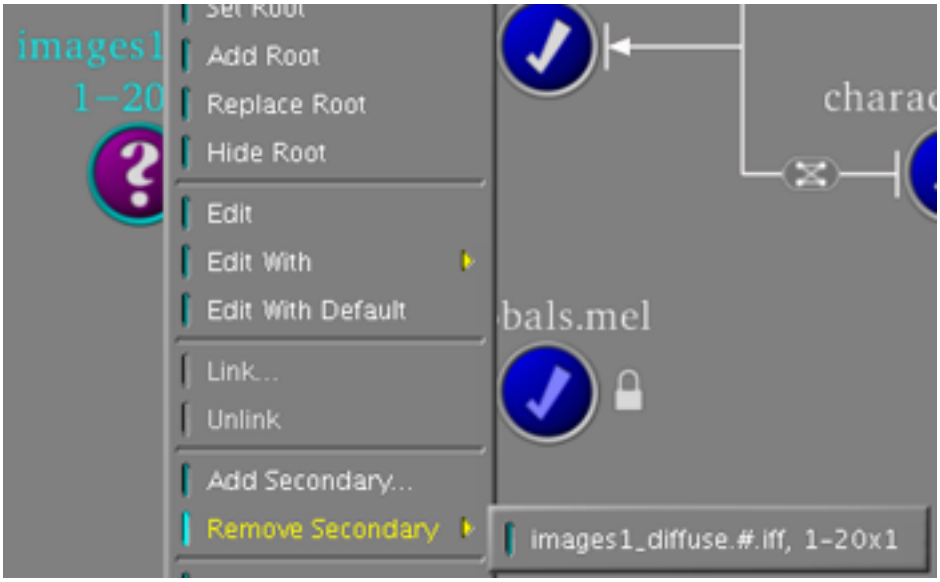



The Node Viewer after the link has been copied back.




The Add Secondary File Sequence dialog. Notice that everything accept the prefix is filled in for you based on the primary file sequence.


The secondary sequence showing up in the Node Files panel. This is listed underneath the primary sequence. Notice that no repository versions exist for the secondary sequence.




The Remove Secondary menu item showing a list of all the secondary sequences associated with that node. Clicking on a sequence will remove it from the node.

# Multiple Working Areas

Multiple working areas permit a single user to use different version of the same file simultaneously.

So far all the actions that we have taken have been in a single working area. While each user has their own distinct working area, it is also possible for individual users to have multiple working areas. Each working area operates as a distinct sandbox, in the same way that different users' working areas are isolated from each other. Changes made to a node in one working area will not effect nodes in the user's other working areas.

This functionality can greatly improve speed and flexibility. Assume that you want to run two versions of a render, each using different versions of linked source nodes. One solution would be to create a second version of every node that was different and every node that depended on a node that was different. But this would quickly become inefficient and confusing, not to mention losing many of the advantages of version control.

Consider a situation where you wish to run two versions of a render, one using version 1.0.0 of a model and one using version 1.1.0. There is no way to have both versions of this node checked out in a single working area. In order to do this in a single working area, it is necessary to create a second node which actually contains the files associated with one of the two desired versions. Then, it is necessary to clone the animation scene that depends on that model, and point the cloned scene to the new model file. Then clone the lighting scene and repoint it as well, and then the images node as well.

By the time you are done, you have created an entirely new tree, identical to the original tree. But now if you wish to do something such as change the animation, you need to change both animation scenes and make sure that you make the same changes to both. Not only that, but you've now split the version control information, so that your new tree has none of the history that explains where it came from.

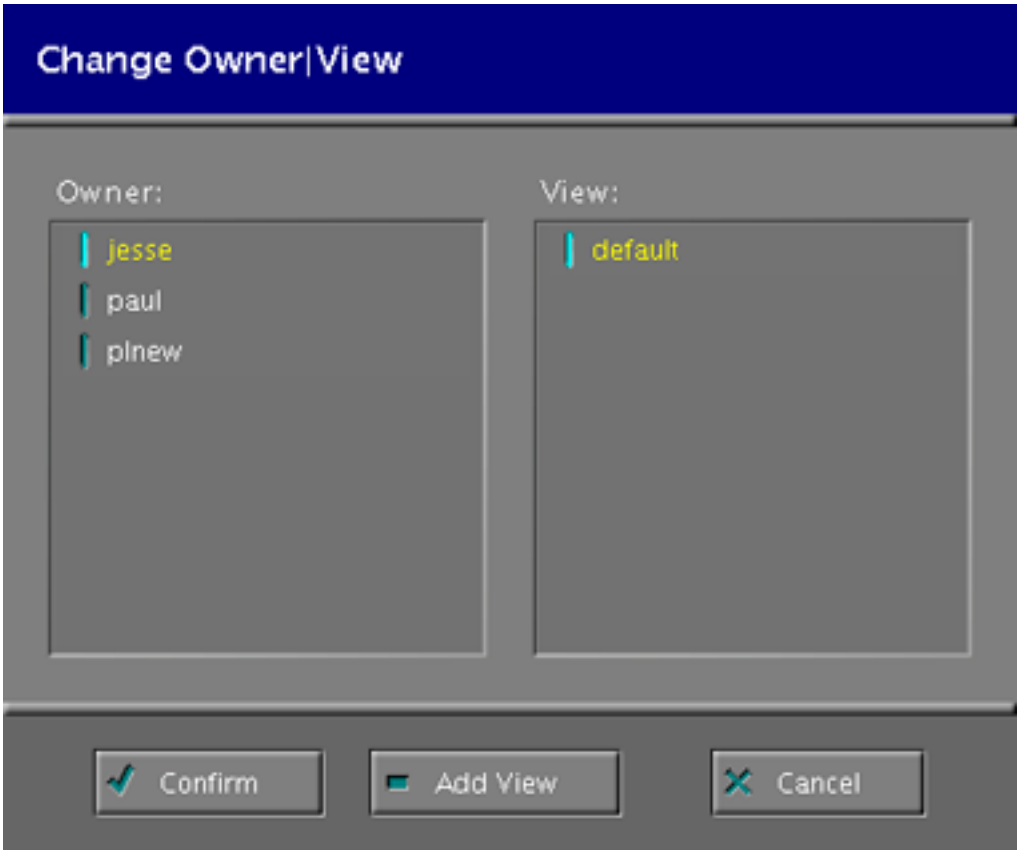
The other, and much efficient, solution is to simply use two separate working areas. In one working area, check-out version 1.0.0 of the model and in the other working area check-out version 1.1.0. Both renders can run simultaneously. Of course, it will be impossible to check-in both renders without making use of the Evolve option, since checking in one of the versions will make the other version Conflicted, but that is something easily dealt with if there is a need for both versions to be checked-in.

This flexibility can also be useful in other situations. It allows you to continue editing a leaf node, even when that leaf node is being used in jobs that are currently running. For example, assume you wish to make changes to a model file which is being used as a source node in several running renders. If you made changes to the model file in the same working area that the renders are queued in, the changes will make all the render nodes stale, not to mention the possibility negatively effecting the renders. Making those same changes in another working area will allow the renders to finish successfully and allow you to continue working.

## Creating and Switching Working Areas

You can use the Change Owner|View menu to switch between different working areas and create new ones.

In order to create a second working area, select the Change Owner/View menu item from the Main Menu found by clicking on the yellow triangle in the upper left hand corner of a node panel. For this example, make sure that you active the menu in the Node Browser.




The Change Owner|View dialog.

Pipeline has a hierarchical setup when changing views in one panel and having other panels change their view as well. The Node Browser is the top of the hierarchy. Changing the working area of the Node Browser will change the working area of every other panel on the same channel. Changing the Node Viewer will cause all the Details panels to change as well. Changing the Job Browser will cause all the other linked job panels to change as well. Changing the Job Viewer panel will change a linked Job Details panel as well. This can occasionally lead to unexpected behavior. If you change the working area in the Node Viewer and then click on a node in the Node Browser, the working area in the Node Viewer will switch back to the working area that the Node Browser is set to.

The Change/Owner View dialog will appear. This dialog allows you to do two things. You can move between different working areas, both your own other peoples. It also allows you to create new working areas for yourself.

Pipeline allows you to browse other peoples working areas. Switching to someone else's working area is exactly like switching to one of your own working areas, except that you cannot make changes to the nodes in someone else's working area. It does make it possible, however, to see changes that someone else has made to nodes.

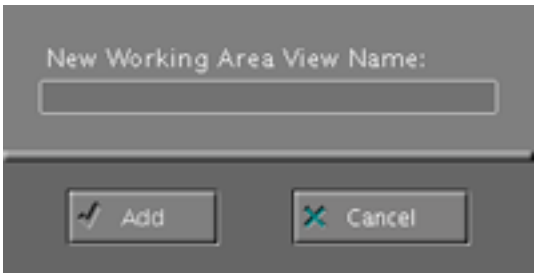
To create a new working area, press the Add View button in the dialog. Pipeline will prompt you for a name for the new working area. It's usually a good idea to give your working areas descriptive names, so that in the future when you are looking for a particular tree you have a good idea of where to find it. In this case, the new working area is only for demonstrative purposes, so 'test' will be an apt name. Once the working area is created, select it from the list in the Change/Owner View dialog and press the Confirm button.

The bar at the top of all your panels should now have changed to read (*user-name* | test). If you look through the Node Browser, you'll notice that the icons next to all the node names indicate that none of the nodes exist in the current working area. When you create a new working area, nothing is checked out. Check-out (character1), open the file, edit it, and check-it back in.

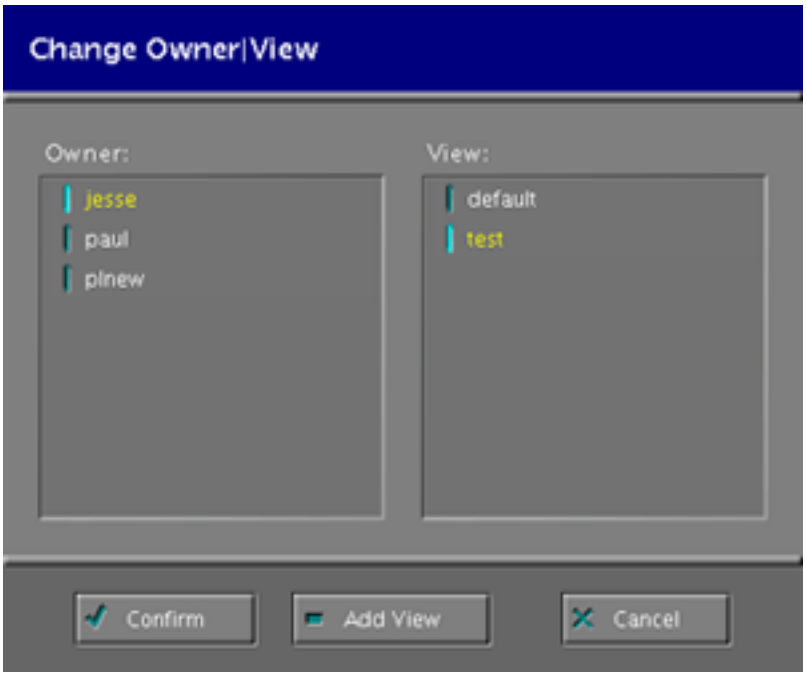
Switch your working area back to your default working area. You will now see that in the default working area, (character1) has the Needs Checkout status.

Users with Administrator privileges can interact with nodes in any user's working area. This makes is possible for an administrator to build a tree, assign Actions, tweak parameters or render settings, add secondary sequences, check-in or out nodes, and any other command that has to do with nodes. The one thing that a user with Administrative privileges may not be able to do is open a file in another user's working area, edit it, and then save it. Working area files are only writable by their owner. In order to save changes in a user's working area, the administrator either needs to get the user to change the file permissions or needs to be able to change file permissions at the operating system level.

Users with the Node Manager privilege can interact with the nodes of any user who is a member of a group that they are a manager of. See the section on User Permissions for a longer explanation of how that works.




This dialog prompts you for a name for the new working area.




The new working area being displayed in the list.


Once you've switched to the new working area, you'll see that the title bar now contains the name of the new working area. The (character1) node is not checked out in this working area yet.

# Release View

Release view provides an easy way to release a large number of nodes after a project is done or a working area is no longer needed.

Earlier you saw how to release individual nodes, removing the current working version. This can be useful to free up disk space and clean up after a project has finished. But releasing hundreds of nodes by hand is too tedious to be useful for this sort of large-scale operation. Simply removing the files that these nodes point to does not solve the problem, as those nodes will still exist in your working area, but will all just have a node state of Missing.

Pipeline provides the Release View menu item, found in the Node Viewer menu which allows for the large scale removal of nodes or even the removal of an entire working area. Selecting this item from the menu causes the Release View dialog to appear. The first option allows you to specify what nodes you want to release. Selecting Entire Working Area will cause every node in the current working area to be released. The other option is Matching Pattern. This allows you to enter a regular expression on the line underneath it and only nodes which match that regular expression will be released from the working area. This is a good way to release a single project from a working area. Entering a regular expression in the form (/projectname/.*) will cause all nodes that live underneath the projectname directory to be selected for releasing. Any sort of standard regular expression can be used here to specify a list of nodes to be released.

The Remove Working Files option controls whether the files represented by the released nodes are deleted. Setting it to (YES) will delete the files. However it will not delete files that are in the same directories but which are not part of Pipeline. If Remove Working Area is set to (YES) then Pipeline will completely remove the working area from your list of working areas. It will not delete files in the working area directory that are not part of Pipeline. Remove Working Area can only be set if the Release Nodes option is set to Entire Working Area.

Switch your current working area back to the test working area. Select the Remove Working Area option from the Node Viewer menu. Make sure the Entire Working Area option is selected, and both the Remove Working Files and Remove Working Area options are set to YES. Click on the Release button. If you now return to the Change Owner/View menu, you'll notice that the test working area no longer shows up on the list of available working areas.

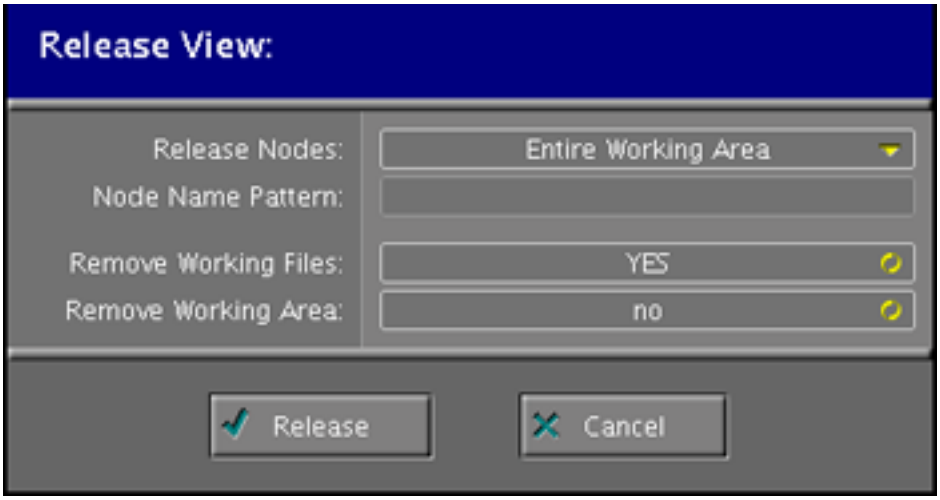
# Offline Files

Pipeline keeps track of which files have been moved offline and allows users to request that certain files are restored.

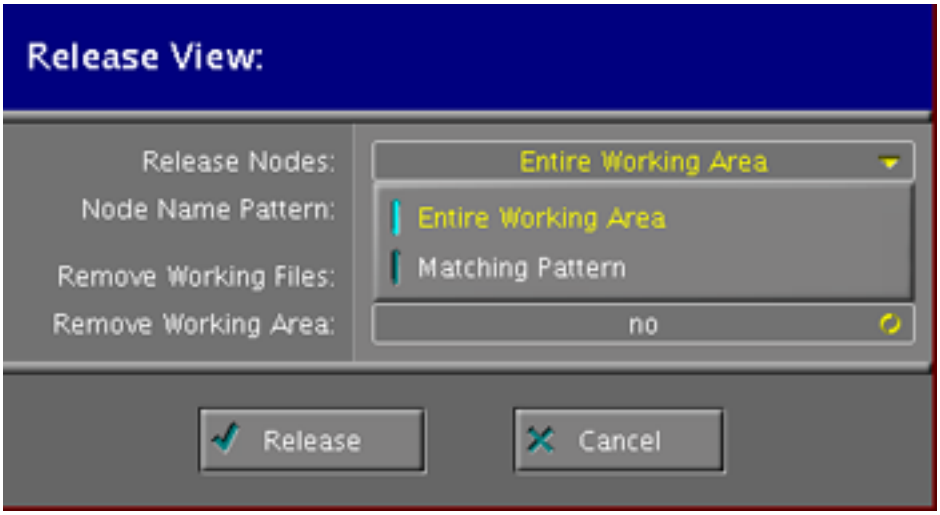
Pipeline also controls the backing up and restoring of files. Most of these processes do not fall under the auspices of this manual since they are purely administrative functions. The one area where normal users interact with this system is to request the restoration of files that have been moved offline. Since there is no way for you to simulate this on your own, this section will not include any practical examples, but will simply explain the options that are available.

When the files for a particular version of a node have been moved offline, you are no longer able to checkout that version of the node. In the Checkout Node dialog it will display the word Offline next to any version which has been removed and will not let you check it out. If you wish to have access to offline files you need to request that they be restored.

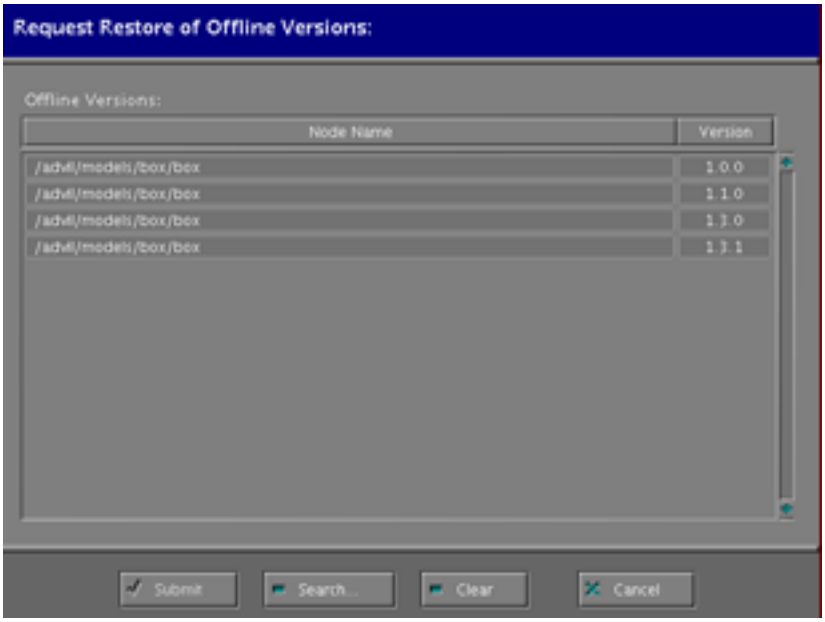
There are two places where the Restore Request menu item can be accessed. The first is in the Node Viewer menu. Selecting the Restore Request item here will display the Request Restore dialog which will contain a list of all the versions which have been offlined. The second is through the node menu in the Node Viewer. In this case, the Request Restore dialog will only show offlined versions of the target node. Pressing the Search button displays another dialog which allows you to




The Release View Dialog.




Selecting Matching Pattern option will let you enter a regular expression in the Node Name Pattern text box. This will allow the selective removal of nodes from the working area. Pipeline will ask you to approve the list of nodes to be removed before it actually removes them.




A list of offline versions of a file. Selecting several from the list and clicking the Submit button will cause a restore request to be sent to the system administrators.

enter a regular expression to search the list. Clicking on one or more versions in the list and hitting submit will submit a restore request for the selected nodes. Use shift-clicking to select a contiguous list of versions and ctrl-click to select non-contiguous versions. Once the restore request is submitted, it is up to the system administrators to process the request and get the node version back online.

## The Help Menu

The help menu contains links to all the reference information related to Pipeline.

There are help options located in the main menu, under the help option. The following help options are available.

About Pipeline: Provides the basic information about the version of Pipeline being run, including the version number.

Quick Reference: Opens up the quick reference page in mozilla or firefox. The quick reference page provides a basic explanation for all the different node symbols and colors.

User Manual: Open this document.

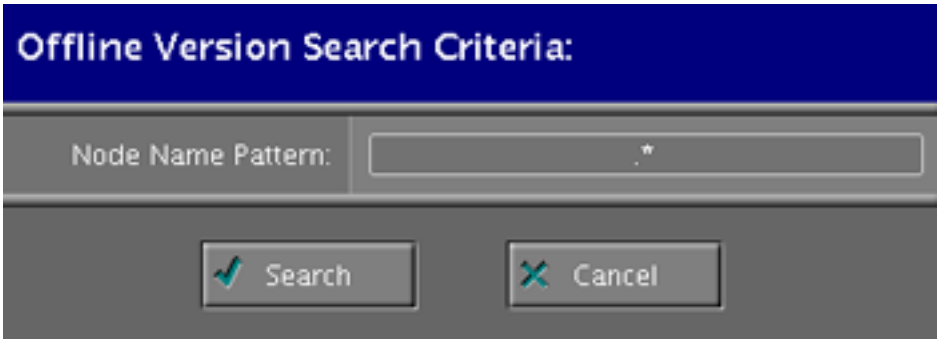
Home Page: Take you to the Temerity Software home page.

Support Forums: Take you to the top level of the Temerity Software forums.

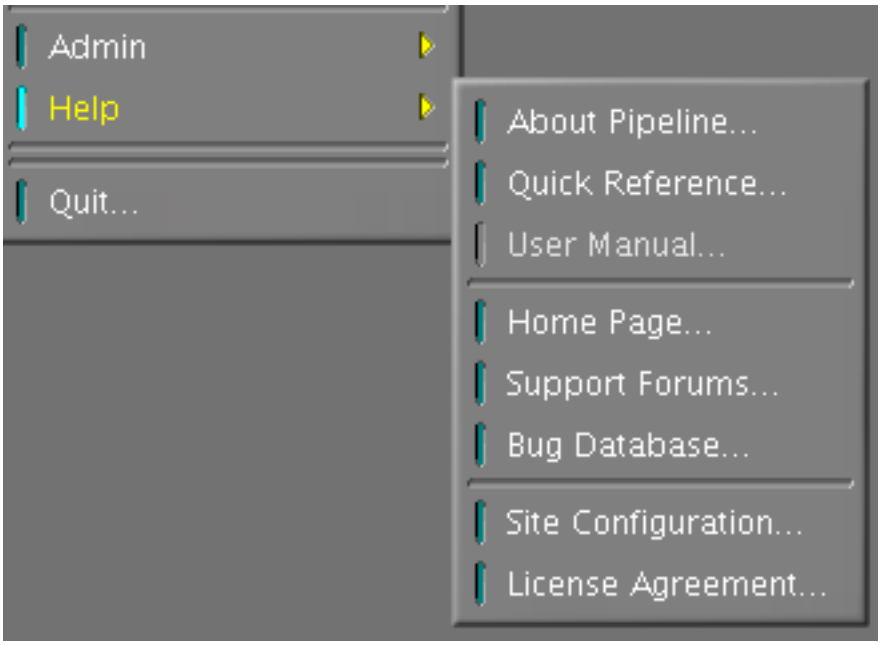
Bug Database: Take you to the Temerity Software forum dedicated to bugs and feature requests.

Site Configuration: Take you to the webpage where you can enter your site configuration to generate a new license for Pipeline.

License Agreement: Displays the Pipeline licensing agreement.




When you click the search button, you will be presented with a text field allowing you to enter a regular expression to prune the list of offline files.




The Help submenu, located in the main menu.

# The Queue

The queue manager uses a number of distinct parameters to control which jobs run on which machines.

Any time user queues a job, the execution of that Action is controlled by the queue manager. Pipeline runs a separate queue manager, distinct from its main manager, to distribute jobs to render machines. The goal of the queue system in Pipeline is to guarantee the most efficient possible utilization of all the available machines, while still permitting important jobs to be prioritized. To accomplish this, the queue server uses a complicated score system to figure out which jobs should be running on which machines. The next section will explain how this scoring system works, and will be followed by a examples focused on interacting with the queue.

## Queue Score Parameters

There are a large number of different parameters which factor into the calculations which determine which job will run on which particular machine. A number of these parameters are assigned to machines in the render queue. Normal users have the ability to change some of those parameters on their local machine. User with the Queue Administrator privilege can change those parameters on any machine in the queue.

• Order: The Order is a per-machine parameter that determines the order in which the queue manager attempts to assign jobs. The higher the order, the lower the machine ranks on the queue manager's list. Among machines with the same order, the queue server simply uses alphabetical order to differentiate between the machines. Order is useful when you have certain machines that you wish to use before others. For example, if a render farm has 20 old machines and 20 new machines, the new machines might be assigned a lower order to ensure that they are used first. That way, when a small number of jobs are submitted, they'll attempt to run on the best machines first.

• Slots: The number of slots is a per-machine parameter that specifies the maximum number of jobs that can ever run on the machine. No matter how many other conditions are met, there can never be more jobs on a machine than the number of slots.

• Reservations: Reservations are a per-machine parameter which restricts a machine to only running jobs submitted by a particular user. If none of that user's jobs qualify on the machine, then no jobs will be run on it. A normal user can change the reservation status of their local machine.

• Dynamic Resources: The queue manager tracks the CPU load, free memory, and free disk space for every machine in the queue. Every Action has a CPU, memory, and disk setting as part of its submission options. For a job to run on a machine, the CPU load on the machine must be below the CPU setting, while the free memory and free disk space must be above their respective parameters. If these conditions are not met, then the job will not be run on that machine. This keeps jobs from being run on computers which do not have the resources to complete them.

• Selection Keys: Selections keys are assigned on a per-machine basis and are either null or a number value ranging from [-100] to [100]. Selection keys control which jobs run on which machines. Each node in Pipeline can have one or more selection keys assigned to it, either in the Node Details pane or at queue time using the Queue Special option. For an Action to run on a machine, the machine must have all the keys that the Action has. So if the Action specifies Key A and Key B, then it can only run on a machine that also specifies Key A and Key B. The numeric value of each key specifies the preference of the machine for that sort of job. If a machine has a Key A value of 90 and a Key B value of 10, it will attempt to run a job with Key A before it runs a job with Key B.

The Selection Key values only guarantee that when two jobs with identical parameters compete for the same machine, the one with the selection key that has a higher value on the machine will run. Selection keys can be used to simply divide the queue up into different parts, allowing certain jobs to run on certain machines or it can be used to represent machines with node-locked licenses. For example, if there are only ten machines that have a node-locked renderer on them, then only

those ten machines will get a selection key representing that renderer and all jobs that need that renderer must specify that selection key. A machine's selection keys are specified by the Group that the machine belongs too.

• Groups: Instead of directly assigning selection keys to machines, Pipeline uses a layer of abstraction called Groups. A render machine is assigned to a particular render group, which specifies keys and their values for all the machines in those groups. Groups have the additional property that they can be set to ignore reservations. This can be useful for overnight rendering, ignoring reservations that were used by artists during the day. What group a machine belongs to can be changed in two ways. A user with the Queue Administrator privilege can change the group manually or they can setup a schedule which specifies what group a machine should belong to based upon the current time and date. For example, a schedule might specify that a group of machines should be removed from rendering during the day and used only for running quick jobs for artists, but then be switched back to rendering overnight.

• License Keys: License keys are used to control floating licenses. There are three different type of license keys that Pipeline supports. Per machine keys are based on machine usage. As soon as a machine starts using a key, a license is deducted from the pool. This means that multiple slots on the same machine can use a license without a problem. Per slot keys mean that each slot that is running a job deducts a license from the pool. Per slot per machine keys limit the number of licenses in two ways. The total number of per machine licenses controls how many machines can run jobs using that license, same as per machine keys. The per host slot settings controls how many jobs using that license can be running on one machine at once. When a job with a license key is being considered for pick-up, it is first evaluated in every other category. If it is selected to run, the queue managers attempts to acquire a license. If it is successful, then the job runs. If there is no license available, the job is passed over and must wait for the next free slot to try again.

• Priority: This is a per-node setting. Every Action has a priority setting. Priority is used to arbitrate when two (or more) jobs are equally eligible for a free slot. This means that all the selection key values are equal and that all the jobs meet the machine stat requirements. In these cases, the job with the highest priority will get the free slot. This allows differentiation between jobs which are competing for the same group of machines. If everything is equal about the jobs, including the priority, then the job that has been waiting longest will run.

Those parameters are applied in a discrete order, gradually winnowing the list of possible jobs until the best one is found.

When the queue manager decides what jobs are going to be distributed to what machines, it starts with a list of machines sorted by order and a list of all the jobs that are currently waiting. The machine with the lower order will be the first one it attempts to find a job for. The first thing the queue manager does is check to see if the machine has a reservation. If it does, then it will eliminate all the jobs not submitted by others users. The queue manager will then examine the dynamic resources for the machine and eliminate any jobs that require more resources than the machine currently has available.

What is now left is a list of jobs that satisfy all the basic requirements of the current machine. These jobs are then sorted, based first on their selection score (generated from their selection keys). If the machine does not have all the selection keys that a job requires, the job is removed from consideration. Selection score ties are arbitrated by the job's priority and then finally by the time the job was submitted, with older jobs taking priority over more recently submitted jobs. The job that comes out on top will be the first job that the queue manager will try to assign to the machine. If the job does not require a license key, then it is simply assigned to the machine. However, if it does require a license, then the queue manager attempts to get a license. If it successfully reserves the license, then the job is dispatched. Otherwise, it discards that job and continues to the next job on the list. It will continue though all the jobs in the list until it either finds a job that can run or it runs out of jobs. The queue manager then proceeds to the next machine and repeats the same process until it has gone through all the machines.

# Interacting with the Queue

## Queue Settings in the Node Details

The bottom section of the Node Details panel contains the per-node queue controls. Select any node and load up details. The first control is the Overflow Policy. Overflow Policy applies in situations where the node has sources with a Link Relationship of 1-to-1. Situations where a source node has fewer frames than the target node are going to result in problems when attempting to regenerate the target frames without a correlated source node frame. There are two different ways that Pipeline can handle this. When Overflow policy is set to Abort then any job that contains frames without a match in the source will not execute (the job status will be Aborted). If the Overflow Policy is set to Ignore then the jobs that involve frames without a match will attempt to run despite missing their source file. It is possible that the Action will fail anyway, if it attempts to access a non-existent frame.

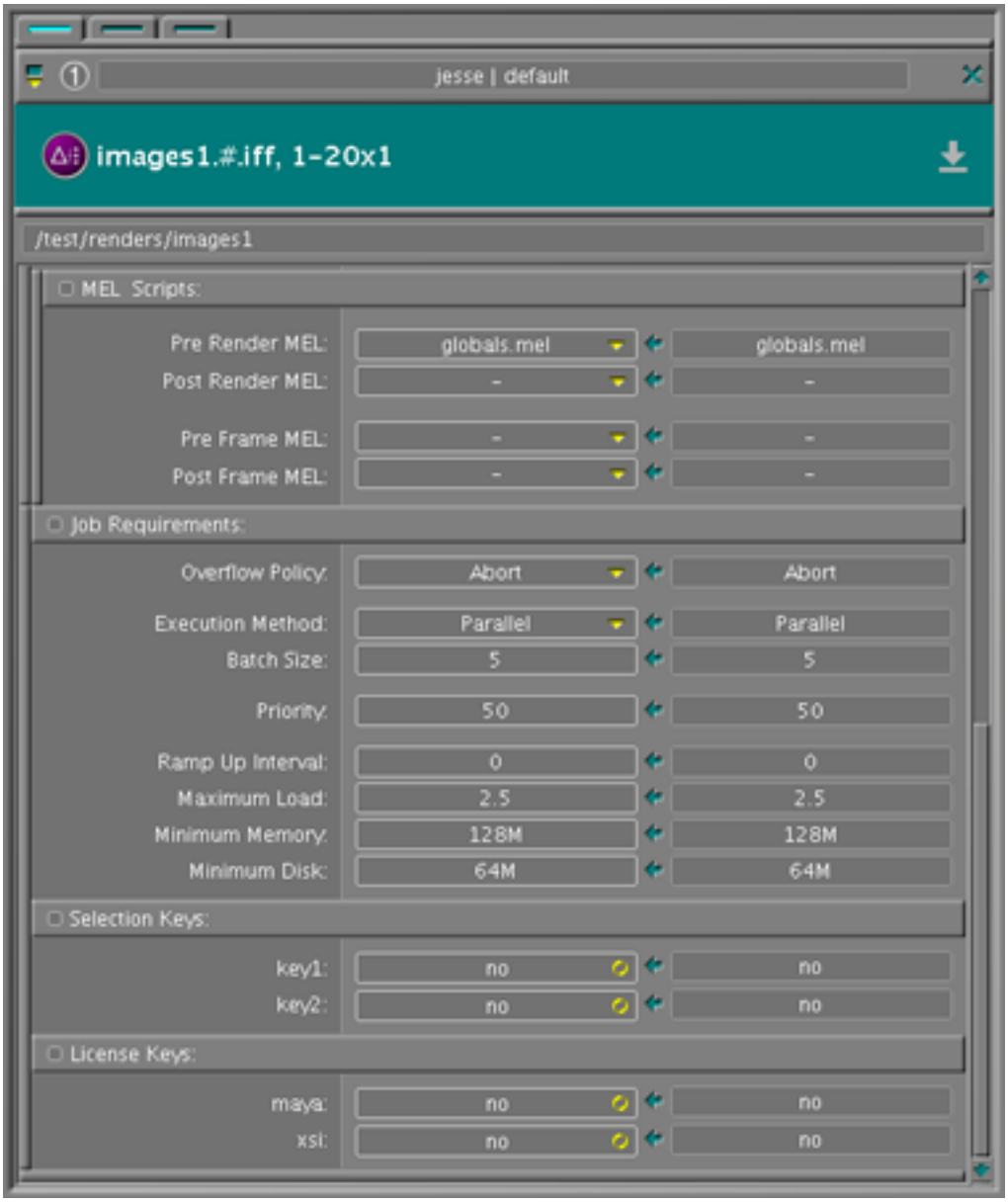
Below the Overflow Policy are the Execution Method and Batch Size settings. These were described earlier in the manual, but will be discussed here for completeness sake. Nodes which are set to Serial must be run from the beginning, with each file being generated after the one before it as a single job. This means that if a single file is missing from the sequence, the entire sequence must be regenerated to fill in that gap. This is useful for procedural events, where the results of any particular frame depends on all the frames that come before, such as particle systems, dynamic setups and cloth simulations. The Subdivided setting causes each frame to be evaluated separately from every other frame. Instead of running the frames sequentially, the order of execution is determined by recursively subdividing the frame range. For example, for a file sequence with a frame range of (1-8), Subdivided would process the frames in the order (1, 5, 3, 7, 2, 4, 6, 8).

When a node is set to Parallel each frame can be evaluated separately from every other frame. However, the Parallel setting makes use of the Batch Size setting located right beneath this option. A Batch Size of 0 will cause all the contiguous files that need to be generated to be run on the same machine. This is not the same as Serial. If the node was set to Serial, then files that already existed and did not need to be regenerated would be regenerated anyway. In Parallel(0), only the files which need to be regenerated will be queued. If the range is non-contiguous, then Pipeline will make the groups as large as possible. A Batch Size of more than one will cause file to be grouped into batches of up to the value of batch size. A batch will be smaller than the batch size if there are not enough sequential files needing generation to make a full batch. In these cases Pipeline will create a batch that contains this smaller set of files.

Underneath that is the Priority setting. Jobs with higher priorities will run before jobs with lower priorities assuming both jobs are equally qualified for an available machine slot. Priority can range from 0 to 999.

The Ramp Up Interval sets the number of seconds that a machine will wait once it starts this job before it attempts to pick up another job. This is useful when you know that a job will initially use a small amount of resources but will then expand its requirements once the job has been running for a while. By forcing the machine to wait before picking up another job, it allows the dynamic resource readings to be more accurate. This means that there is less of a chance of a machine running too many complex jobs at once and running out of resources.

Finally there are the dynamic resource settings. Maximum Load represents the maximum system load that a machine can have and still be able to run this job. If a machine has a load over this level, then the job will not be dispatched to that machine. The Minimum Memory setting represents the minimum amount of free RAM that a machine must have to be run this job. If a machine's free memory level is below this setting, then the job will not be dispatched to that machine. Minimum Disk represents the minimum amount of free hard disk space that a machine must have to be run




The Job Requirements section of the Node Details panel.

this job. If a machine's disk space level is below this setting, then the job will not be dispatched to that machine.

The Selection Keys section is right below the Job Requirements. Each key that is set to YES is a key that is required on a machine that is going to run the job. If three keys are turned on, then a machine must have all three keys. Turning on more keys does not necessarily increase the chances of the job of running.

At the very bottom is the License Key section. Each one of these set to YES means that a job server

## Queue Jobs and Queue Jobs Special

Queue Jobs and Queue Jobs Special are the two commands used to send jobs to the queue. They can be found in many of the menus in Pipeline. They exist in the Node Menu in the Node Viewer, in the menus in the Node Files and Node Details panels accessed through pressing (mouse3) on the node status symbol in the upper left hand corner, in the menu accessed by pressing (mouse3) on a file in the Node Files panel, when pressing (mouse3) on a job group in the Job Browser, and when pressing (mouse3) on a job or job group in the Job Viewer. The commands do something similar everywhere, create a job group to regenerate frames. When either command is run in the node panels, all Stale frames will be queued. When it is run in the job panels, only frames from jobs which have a queue state of Failed will be re-queued.

The Queue Jobs command will submit the jobs with whatever queue settings are saved on the node. Queue Jobs Special, however, will pop up a dialog allowing you override many of the job submission parameters. You can edit the Dynamic Resource parameters, the Job Priority, the Run-up Time and change the Selection keys assigned to the job. Only the nodes that you have selected will be queued with these new parameter settings. Jobs generated for stale upstream nodes will be still be queued with their saved job settings. Queue Jobs Special comes in handy in the job panels as well. If there is a crucial job which has been waiting for a long time to run, you can kill it, and then use Queue Jobs Special to resubmit it different parameters which will cause it to pick up more quickly.

## The Job Groups Tab

The Job Groups panel displays job groups, giving a high-level look at all the work that queue is currently doing, finished doing, and waiting to do.

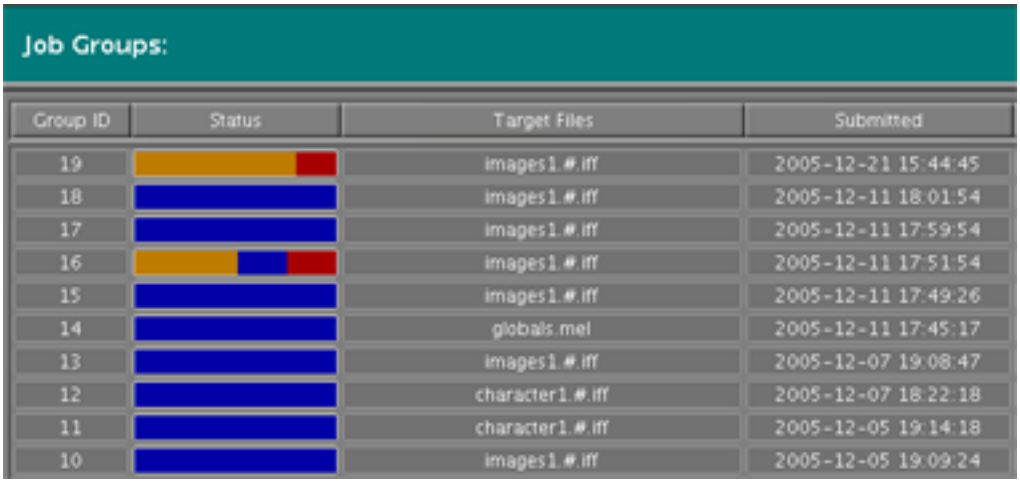
It is now time to look at the job panels. If you haven't already, create a linked Job Browser, Job Viewer, and Job Details panels. You can link them on the same channel as your node windows or use a separate channel for them.

You'll notice that the Job Browser is sub-divided in to the tabs. The first tab shows all the machines that are in the queue and all the settings that control what jobs those machines can pick up. The second tab provides a listing of all the slots available on all the machines and what jobs are currently running in those slots. The third tab provides a listing of all the job groups and their statuses.

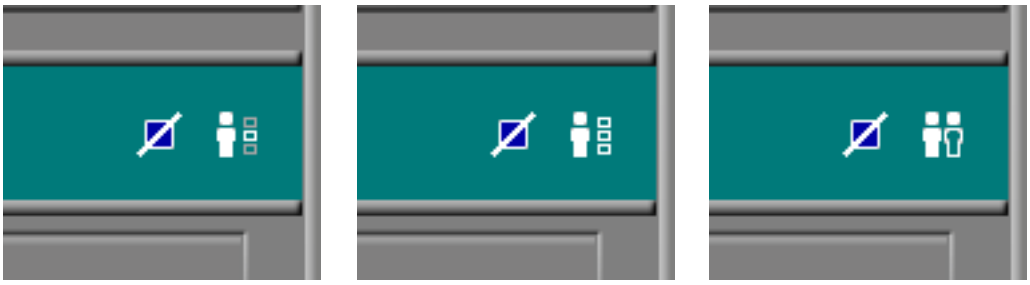
If you click on the third tab now, you'll probably see a long list of job groups from all the previous times that you have queued nodes. To clear all displayed jobs in which all the frames have finished executing either successfully or unsuccessfully, click on the button in the upper right hand corner that looks like a square with a line through it. All the job groups on the list should now disappear.

The button to the immediate right of the clear jobs button controls what jobs are visible in the Job Browser. By default, only job groups that were submitted by the current user from the current working area are visible. Pressing the button once will change the mode so that all job groups submitted by the current user, regardless of the working area, are displayed. Pressing it a second time will

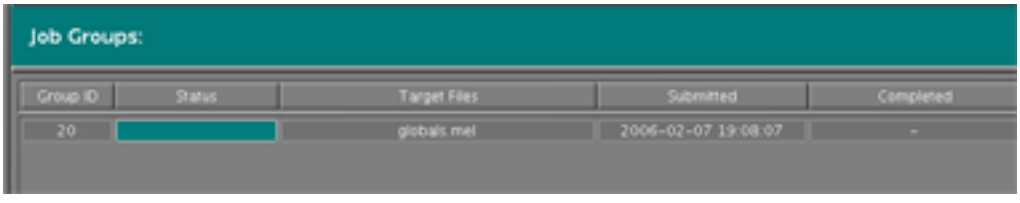

The Queue Jobs Special dialog.




The right hand side of the third tab in the Job Browser. It shows a list of all the job groups, sorted by Group ID. Next to the ID is the status bar, showing the individual statuses of each job in the job group.



The upper right hand corner of the Job Groups tab. The icon on the left will clear all finished job groups from the queue. The icon on the right represents the different display states. From the left: only job groups associated with current owner and view; job groups associated with the current owner; all jobs.




The Job Groups tab after (globals) has been queued. The status bar is teal, indicating that all the jobs in this group (which is only one in this care) are waiting to get picked up.

display all submitted job groups from all users. As with the Job Browser and other panels, you can use the Change Owner/View command found in the Main Menu to change what working area you are viewing.

It is important to note that this window only displays job groups and not actual jobs. A job group is created every time a node is queued and contains one or more jobs. A job is created for the node that was queued and for each of its sources that needed to be queued. Plus for any nodes which use batching, a separate job is created for each batch. Doing something simple like queuing a render results in a single job group that may contain hundreds of jobs.

Return to the Node Browser and select the (/test/globals/globals) node. Press the (mouse3) button on it and select the Remove Files menu item to make the node stale. Queue the node and return to the Job Browser. If your Job Browser is linked to your Node Viewer, then the status should have already updated and your job group should show up. If they are not linked, then you need to right-click on the teal area at the top of the Job Browser and select the Update Status. Once it is updated, you will see a job group displayed.

The status bar for each job group on this tab is subdivided based on the number of jobs. Each job subdivision displays a color based on the status of that job. Looking at the status bar gives an overall idea of the status of every job in the group, without providing any specifics about the individual jobs. Each color used here to represent statuses are the same as those used in Node Viewer to show queue state with the exception that there are no Stale or Undefined jobs.

An entirely blue bar would mean that every job in the job group was in the Finished state. A half green and half blue bar would mean that half the jobs were in the Running state and half were in the Finished state. An entirely red bar would mean that every job in the group a state of Failed and so on. Next to the status bar is the name of the files being generated and then the time at which the job group was submitted. If all the jobs in the group have finished running (either successfully or failing), then the time that the last job finished will be displayed under Completed time.

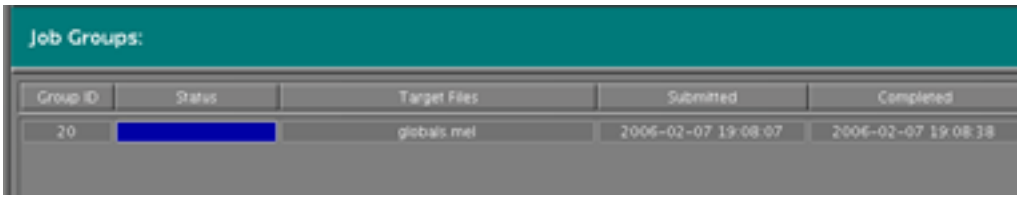
This value is not necessarily a measure of how long the jobs took to run, since it includes time during which the jobs could not run because there were no machines available. In order to get a better idea of how long jobs are taking to run, it is necessary to get information about individual jobs. Next is the target node name, which provides the full path of the node allowing you to differentiate between jobs which might have the same file sequence. Finally on the far right is the user and working area from which the job was submitted.

In the case of (globals) there is only one job in the job group, so the entire bar will reflect the status of that single job. Right after the node is submitted, the bar will be teal, reflecting the Queued state. Once a machine picks up the job, the bar will change to Green and then finally to Blue once the job successfully finishes.

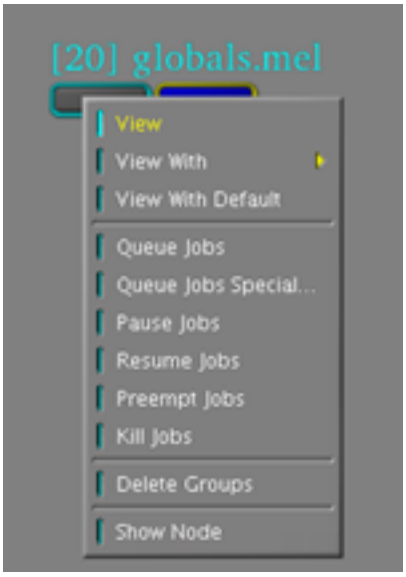
The Job Viewer allows you to see all the specific jobs that make up a job group, including the status of each job.

To view more specific information about a job, it is necessary to use the Job Viewer and the Job Details panels. Click on the (globals) job group and you'll see it show up in the Job Viewer. The job group number and the name of the source node are displayed across the top of the job group. Underneath that and on the left is a grey rectangle that represents the job group as a whole. Pressing (mouse3) on that rectangle will bring up a menu of actions that will apply to the entire group.

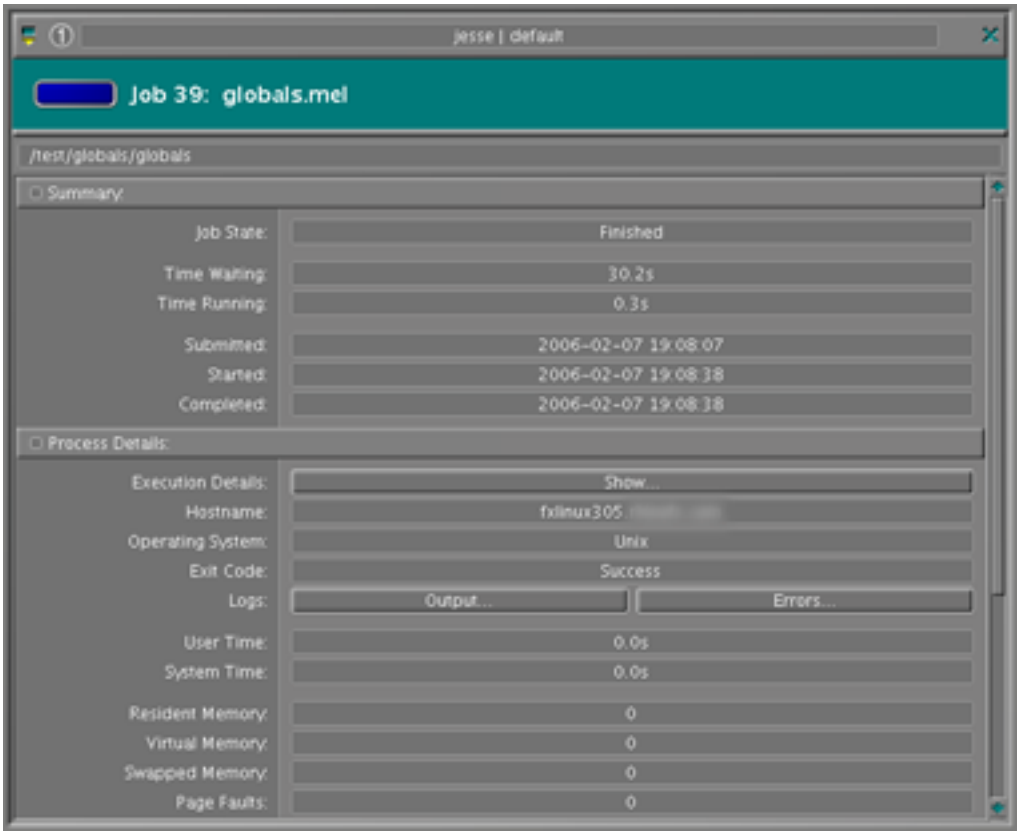
To the right of this rectangle, each individual job is displayed. Jobs are displayed as colored rectangles with the color representing the queue state of that particular job. In this case since there is only one job, there is only one rectangle. It should be blue, since the (globals) job completed successfully. If you double-click on an individual job, then its details are loaded in the Job Details panel. The Job Details panel will provide you with all the specific execution details about that job.




The same tab after the job has been picked up and run. The status bar is now blue.


The (globals) job displayed in the Job Viewer. The grey rectangle on the left allows you to run commands on the entire job group. Or you can click on an individual rectangle on the right to effect just that job.




The Job Details panel.

The Job Details panel displays all the job specific execution information about a job selected in the Job Viewer.

Double-click on the (globals) job in the Job Viewer to load up its details. The top part of the panel provides information about the job's time. It displays how long the job had to wait before running and how long it took the job to complete once it started running. The time that the job was submitted, started running, and completed are also shown. Underneath this are the details of the process that Pipeline created to run the job.

If you press the Show button next to Execution Details, then the Execution Details panel will be displayed. At the top of this panel is the exact command line that was run by the job. Underneath that is a list of all the environment variables defined in the toolset that was used to run that process.

Back in the Job Details panel, the next line indicates what render machine the process was run on, followed by the Exit Code that the process returned. The next line contains the log files for the process. Pipeline redirects the Standard Out and Standard Error streams to files locally on the render machine. Pressing the Output button will display any output from Standard Out, while the Errors button will show the contents of the Standard Error log.

The last four lines provided detailed information about memory usage by the process. The next part of the Job Details displays the Job Requirements that the job was submitted with. The priority, ramp-up interval, and dynamic resource settings are displayed first, followed by the selection keys and then the license keys. The last section covers the file sequences, first showing the target sequence that the job was generating and then all the source sequences that were used in generation.

Pipeline handles this logging in an efficient manner. All output from the process is captured locally and saved in a text file. When you request the log file, Pipeline reads directly from the text file, never bothering the process that is actually running the job. When you have the Errors or Output dialog window open, Pipeline will regularly check the file for additional information and update the log with new any new data. This means you can use those windows to watch a job's progress, without it actually slowing down the job.

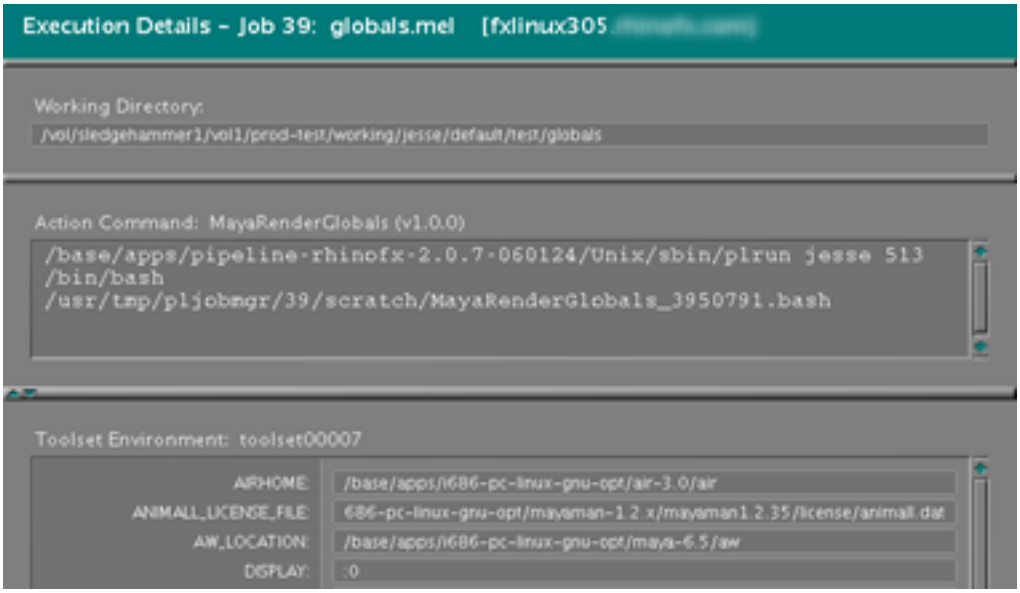
## The Job Slots Tab

The Job Slots tab provides information about which job is running on which machine and also lets you control individual jobs.

Go back to the Node Browser and select the (images1) node. In the Node Details panel, set the Set the Execution Mode to Parallel and the Batch Size to 2 and press the Apply button. Then remove its files so we can queue it. Queue the job and return to Job Browser. There should be a new job listed for the (images1) node. Notice that the status bar now contains various colors. Some of the packets should be green representing jobs that are currently in the Running state, some should be teal and in the Queued state, and some will be blue as they reach the Finished state. If you click on the job group, you'll see all the jobs displayed in the Job Viewer. Each rectangle represents one job and the color of the rectangle the queue state of that job. You can double-click on any of the jobs to load its details into the Job Details panel.

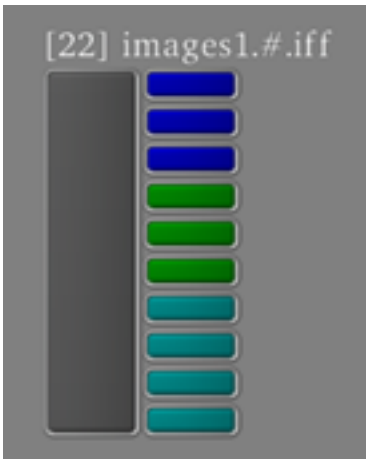
In the Job Browser, switch to the middle tab at the top to view the slots list. This should show a representation of all the machines in the queue along with any of the jobs that are running on those machines. Each line represents a slot on a machine. Any machine which has more than one slot will appear more than once on this list. Next to the name of each machine is the Job ID and the file sequence that is being run on that machine. It is here that you can see what particular frames are running on a particular machine. The On Hold field is connected to the Ramp Up parameter that is set on a node. Ramp up specifies the length of time a machine must wait after starting a job to acquire a new job. On Hold displays the amount of time left before an empty slot starts looking for a job to run. After this is the time the job started running and then how long the job has been running.


The top of the Output Panel. In the case of this job, there was no output.




The Execution Details panel. The actually command that Pipeline ran is at the top. Underneath that is a list of all the environmental variables that are in the toolset that is being used to run the job.


The Status bar as the jobs progress. As more of the jobs finish, more the bar turns blue.




The job group displayed in the Node Viewer. Here you see each job as an individual rectangle with the color representing the status of that job. Right-clicking on a job allows you to issue commands to just that job. Right-clicking on the grey rectangle will issues commands to every job in the group.

This make it easy to see jobs that are taking an abnormally long time to run. Finally the target node name and the user and view who submitted the job are displayed.

If you press (mouse3) on a job in the slots window you get a list of operations you can perform on that group. The Update option will update all the slots with the latest information. The three view options available are identical to those found in the Node Viewer, except they only operate on the files generated by that job and not all the files associated with the node. The Kill Jobs option will causes all the selected jobs to terminate. The Preempt Jobs option will cause all the selected jobs to be preempted.

## Preemption

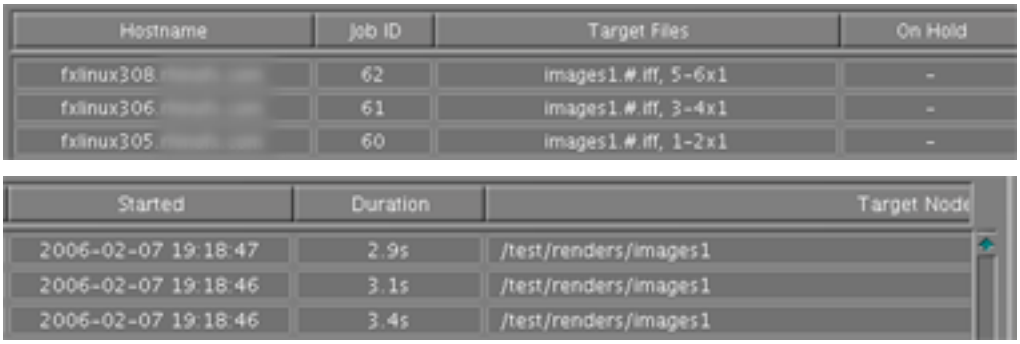
Preemption will cause a running job to terminate and return to the queue. This is distinct from how Kill Jobs works. When you terminate a job with Kill Jobs, the job will have the Failed state and will not attempt to run again. This is the option to use if there is something wrong with a job and you just wish to stop it. Preempt will terminated the job, but will leave it in the Preempted state, where it will attempt to run again. The Preempt job option should be used when a machine is needed for another job, but you wish the job that is being terminated to continue running later. The typical use for preemption is the case where a low-priority job group has been submitted to the queue and its jobs are using most of the machines. A high priority job is submitted, but it cannot get machines to run because of the low priority jobs that are running. The low priority jobs can be preempted, which will allow the high priority machines to take those machines. Once the high priority jobs have finished, the low priority ones will pick back up. Preempted jobs are marked with the pink color.

Right-click on one of the jobs listed in the slots view and select Kill Jobs. The job should disappear from the slot. If you return to the third tab (the job groups view), you should see one of the packets in the status bar has turned red. This represents the killed job. If you click on the job group and examine it in the Job Viewer, you will see that one of the rectangles is now red. It is possible to requeue this job. Right-click on the packet that you killed and select the Requeue option. This will create a new job group which will run the killed packed. You should now see a new job group in the list and when you click on it, you should see one packet, which is generating the frames from the job you killed.

## Job Groups with Multiple Jobs

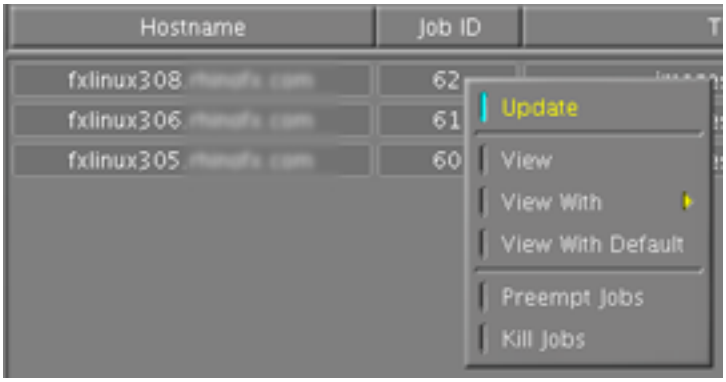
Once all the renders finish, go back to the Node Viewer and remove the files for both the (globals) node and the (images1) node. Press the (mouse3) button on the (images1) node and select the Queue menu item. This will cause both nodes to be queued in a single job group. If you go to the Job Browser and click on the job group, you'll see in the Job Viewer that each job for the (images1) node has another job displayed to the right of it. This job to the right is the (globals) job, and it is being displayed for each job because each job depends on it. It is only being run once, even though it appears multiple times in the Job Viewer.

Right-click on the grey rectangle that represents the whole group and select the Pause Jobs option. This will not effect any job which is currently running but will switch any job in the group in the Queued state into the Paused state. You'll notice that all the (images1) jobs have turned yellow, showing their new queue state. If you update the status in the Job Browser you'll see that most of the bar is now yellow, except the one small blue part that represents the (globals) node. Right-click on the group in the Job Browser and select Resume Jobs. This will put all the paused jobs back into the queued state. Clicking on the job group in the Job Browser is the same as clicking on the grey job group rectangle in the Job Viewer.

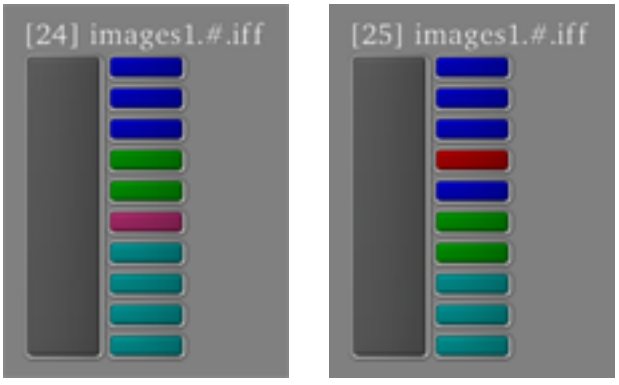



The left and right sides of the Slots tab (split for space reasons). This shows you which job is running on which machine and how long each job has been running.

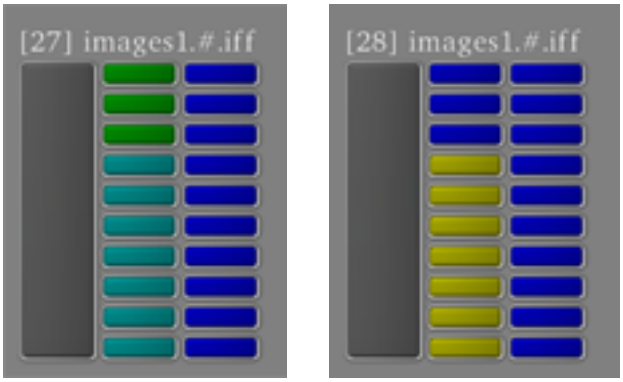

Preempted jobs show up as pink. Killed jobs are shown as red.




The Job Slots menu. The view options will only view the results of that particular job, not the entire job group.




A preempted job and a killed job in the Job Viewer.




Two rows of jobs, showing that the jobs on the left depend on the jobs on the right. In the second image, the yellow jobs have been paused. Selecting Resume Jobs will cause them to return to the Queued state.

## The Job Servers Tab

The Job Servers tab shows each machine in the queue and all the settings associated with each machine.

The first tab in the Job Browser shows the state of all the machines in the queue. Down the left hand side is a list of all the machines in the queue. Next to that is the status of the machine. There are four different statuses that machines can have. Users with the Queue Administrator privilege can adjust the status of a machine.

Enabled: Machines in this state are running the pljobmgr daemon and are actively looking for or running jobs. Any machine in the enabled state will be considered by the queue manager when it is dispatching jobs.

Disabled: Disabled machines are still running the pljobmgr daemon, but are not being considered for new jobs. This allows the machine to be used for other purposes (interactive work, perhaps) without shutting down the daemon, which will let a user to re-enable the daemon later. This is also a useful state when a machine that is currently running a job needs to be taken offline. Switching that machine to Disabled will permit the machine to finish the current job correctly, but will then prevent it from picking up any more jobs. Disabled is also the state that the Queue Manager assigns to machines that have been Hung for over [15] minutes, indicating an irreparable breach of network communications.

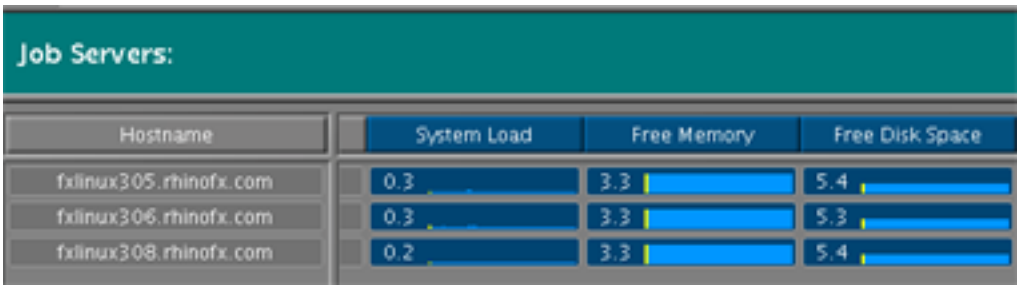
Hung: Hung is a state that the queue manager will assign to a machine. It is not useful for a user to set a machine to this status. A hung machine is one that was previously enabled, but which the queue manager can no longer communicate with. Several things can cause a job server to become hung. A breakdown in network communications will definitely cause this problem. It can also be caused if a job running on the machine consumes all of the machine's resources such that the pljobmgr daemon does not have access to CPU or memory or that the machine simply crashed. In any case, the queue manager will continue trying to contact the machine for [15] minutes: if the connection is successfully reestablished, then the machine returns to the Enabled state. Otherwise, the queue manager will change the state to Disabled.

Pipeline assumes that jobs on hung machines have continued to run, even after it switches to the Disabled state. While these jobs will not show up on the Machine tab or the Slots tab (since the Queue Manager cannot communicate with the machine in question for a status update), they will continue to be displayed as Running in the Jobs tab. This can, at time, be undesirable, especially when the machine has crashed and the job is clearly not running anymore. To solve this problem, Pipeline will mark all of the jobs currently 'running' on a machine that was disabled after being hung as Failed, when the server is switched to Shutdown.

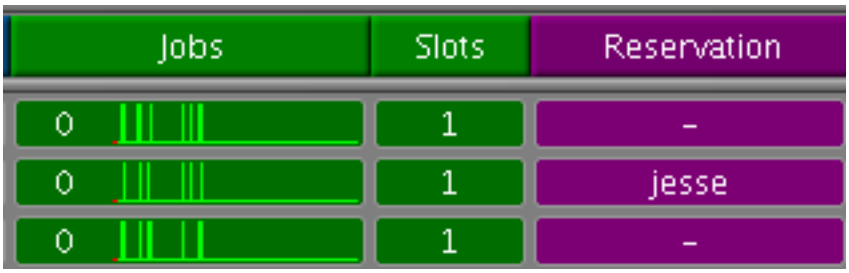
Shutdown: This state indicates that either the pljobmgr daemon is not running on the machine or that it has never been switched to enabled. By default, when Pipeline starts, all the machines on the queue are in the Shutdown state regardless of whether they are running the daemon. Switching a shutdown machine to Enabled or Disabled and hitting the apply button will cause the queue manager to attempt to contact the daemon on the machine. If it can successfully talk to the machine, then it will change its state. Otherwise it will remain set to Shutdown. If a machine which is currently set to Enabled or Disabled is set to Shutdown and the apply button is hit, then the queue manager will send a command to the machine to shutdown the daemon. The daemon will wait until all currently running jobs are finished and then shutdown. This does not cause the machine itself to shutdown.

If the machine needs to be shutdown immediately, rather than waiting for the jobs to finish, the following workflow is suggested. Switch the machine from Enabled to Disabled. Then switch to the Slots tab and Preempt all the jobs that are currently running on the machine. Once they are all dead, switch back to the Machine tab and switch the machine from Disabled to Shutdown. The daemon will then immediately shutdown.

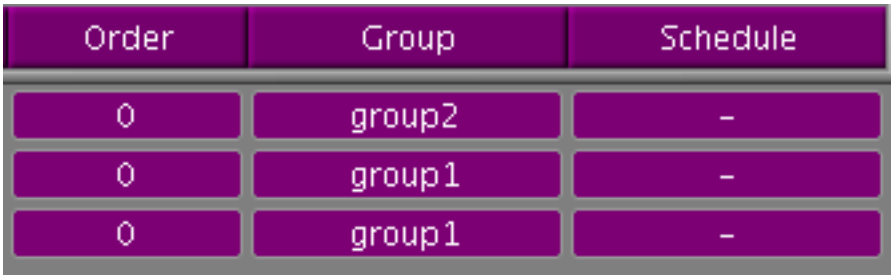

The far left side of the Job Servers window. This shows the machine status and the OS that the job server is running.




The next part of the Job Servers tab showing the dynamic stats of the machines.




The graph shows the number of jobs running on the machine over time. The graph is adjusted so that the top of the graph represents the number of slots. The slots field determines how many jobs can run on the machine. Finally, a name can be put in the reservation slot and only jobs submitted by that user will be run on that machine.




The far right of the Job Servers tab, showing the order and the the group and schedule the machine is currently on.

To the right of the machine status is the reservation field. Typing a user name into the reservation will reserve the machine for jobs submitted by that user. A normal user can change the reservation for the machine they are currently using. Users with the Queue Administrator privilege can change it for any machine.

Next to the order field are three graphs representing the realtime dynamic resources and a fourth graph displaying the number of jobs running on the machine. The first graph shows the CPU load, the second graph the amount of free memory, and the third the amount of free disk space. The jobs graph is normalized so that the maximum number of jobs a particular machine can run is the maximum of each graph. Immediately to the right of the jobs graph is the Slots field. The Slots field specifies the maximum number of jobs that a machine can run at once. Users with the Queue Administrator privilege can change how many jobs can run on a machine simply by switching this value and hitting the Apply button. This change is not retroactive. If a machine is currently running three jobs and the Slots field is switched to 1, it will continue running all three until they complete. However, the machine will not accept any new jobs, until it has an available slot.

To the right of the Slots field is the Order field. Users with the Queue Administrator privilege can change the Order of the machines here and then click the apply button. Machines with lower orders will be considered for jobs before machines with higher orders when jobs are being assigned to machines.

Finally are two drop down boxes which control which group and schedule the machines are on. Changing the group setting will change which selection keys and values the machines have. If the group is set to null (the "-" character) then the machine has no selection keys. The schedule setting will take control of the group field, assigning the machine a group based upon the settings assigned to the schedule. If a schedule is assigned to a machine, then the Group dropdown will be disabled and the Group column will display the Group that the schedule has currently assigned to that machine.

## The Job Servers Tab Menu

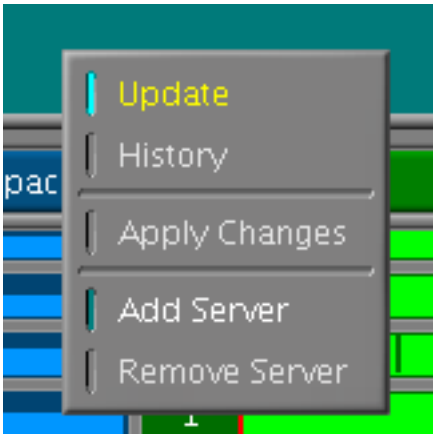
Right-clicking in the top of the panel will cause a menu to pop-up. The items in the menu are:

• Update: This will update the entire panel. All the graphs will be updated with the latest stats and any changes that have been made since the last update will be shown.

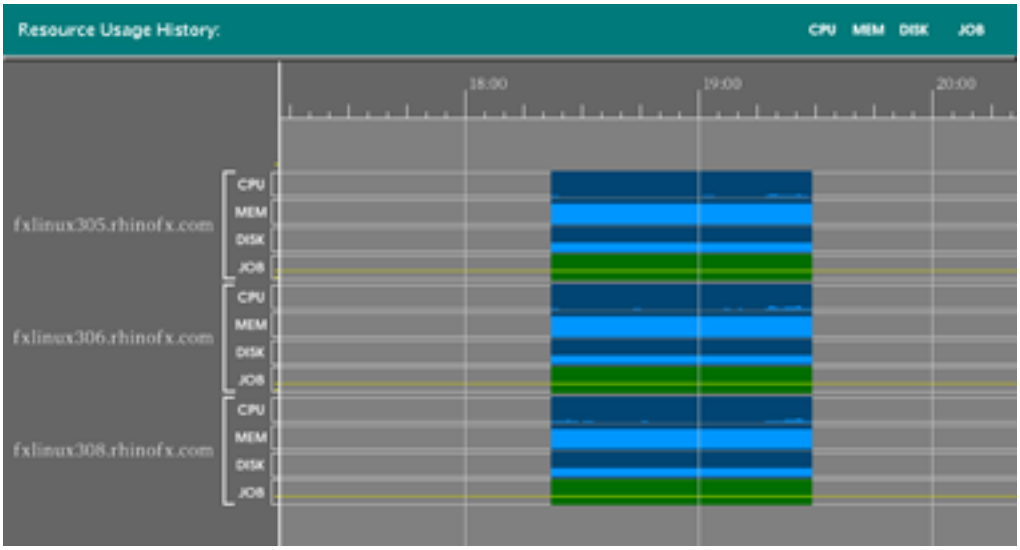
• History: This option will display the history dialog for all the machines that are selected on the left. The history window shows the System Load, the Free Memory, the Free Disk Space, and the number of jobs for each machine over a period of time. Pressing the (alt) key, (mouse2), and (mouse3) and dragging will zoom in and out allowing you to see more or less of the graphs. Pressing (alt) and (mouse2) and dragging will pan around allowing you to view different parts of the graphs. In the upper right corner are several icons that allow you to turn on and off the different parts of the history. Each different sort of data can be turned on and off.

• Apply Changes: This has the same effect as clicking on the icon in the upper right hand corner of the panel.

• Add Host: Typing in the name of a host will cause Pipeline to search for it on the network and then add it to the queue. You can type in a short hostname, an IP address, or the long hostname: Pipeline will always attempt to resolve whatever you give it to the long hostname. All added machines start in the Shutdown state. Only users with the Queue Administrator privilege can perform this function.




The menu in the Job Servers tab.




The Resource Usage History panel.

• Remove Host: This option is only available when the menu was activated by clicking on a machine name. Selecting this option will completely remove the host from Pipeline. [TODO: what happens when you remove a host currently running jobs?]. Only users with the Queue Administrator privilege can perform this function.

Clicking on any of the column headings will sort all the machines by that column. There are also several icons in the upper right hand corner of the left of the apply button. These toggle on and off the different columns in the Job Browser.

# Queue Configuration

## Setting Up License Keys

The Manage License Key dialog is used to add and remove license keys and configure licensing schemes.

User with either the Master Admin or the Queue Administrator privilege can set up license keys. The Manage License Keys dialog can be displayed by selecting the License Keys menu item found in the Admin submenu of the Main Menu. Down the left side of the dialog is a list of all the keys, with their properties stretching out to the right. To add a new license key, press the Add button at the bottom of the dialog. You will be prompted for a name for the key and brief description of what the key is. To remove a key, simply select it by pressing (mouse1) on its name and press the Remove Key button at the bottom of the dialog.

Once you've added a key, you can set its licensing scheme and the total number of licenses available under that scheme. The scheme that you select determines which columns are editable. The Available column is not editable and it displays the total number of licenses that are not currently being used. To see an updated value in that number, simply press the Update button at the bottom of the dialog. The different licensing schemes are discussed in the Queue Score Parameters section.

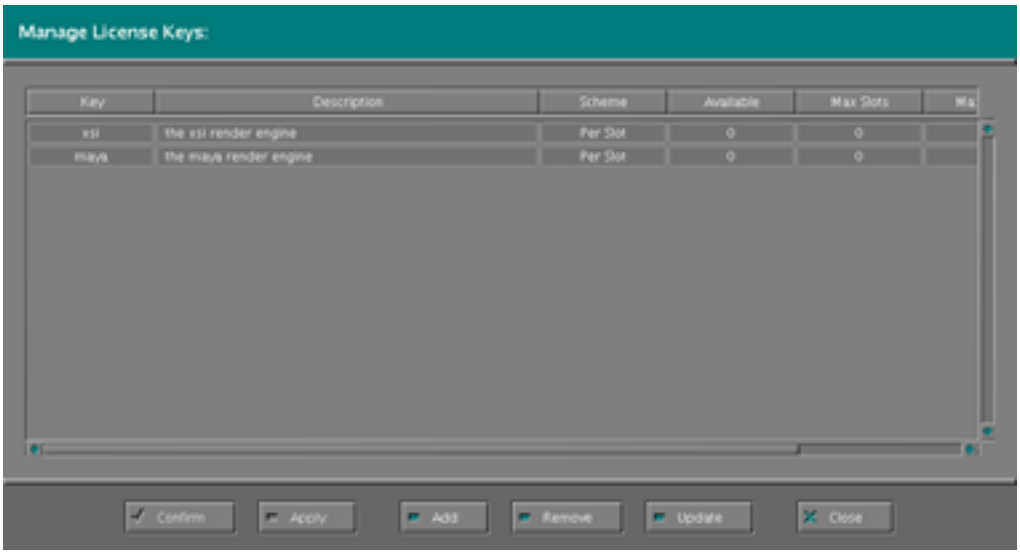
## Setting Up Selection Keys

The Manage Selection Key dialog is used to add and remove selection keys and configure groups and schedules of those selection keys.

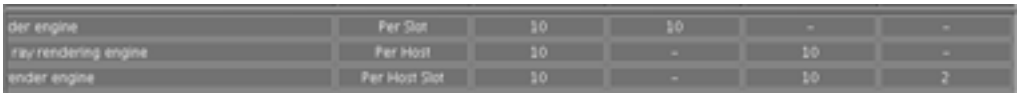
User with either the Master Admin or the Queue Administrator privilege can set up selection keys, groups and schedules. The Manage Selection Keys dialog can be displayed by selecting the Selection Keys menu item found in the Admin submenu of the Main Menu. The dialog is divided into three tabs. The first tab allows you to add and remove selection keys, the second tab is used to add keys to groups, and the third tab is used to create schedules for assigning those groups to machines in the queue.

In the first tab, the selection keys are listed in a column down the left hand side with a brief description of each key to its right. To add a key, press (mouse3) in the dialog and select the Add Key item. You will be prompted for a name for the key and brief description of what the key is used for. To remove a key, press (mouse3) on the name of the key you wish to remove and select the Remove Key item.

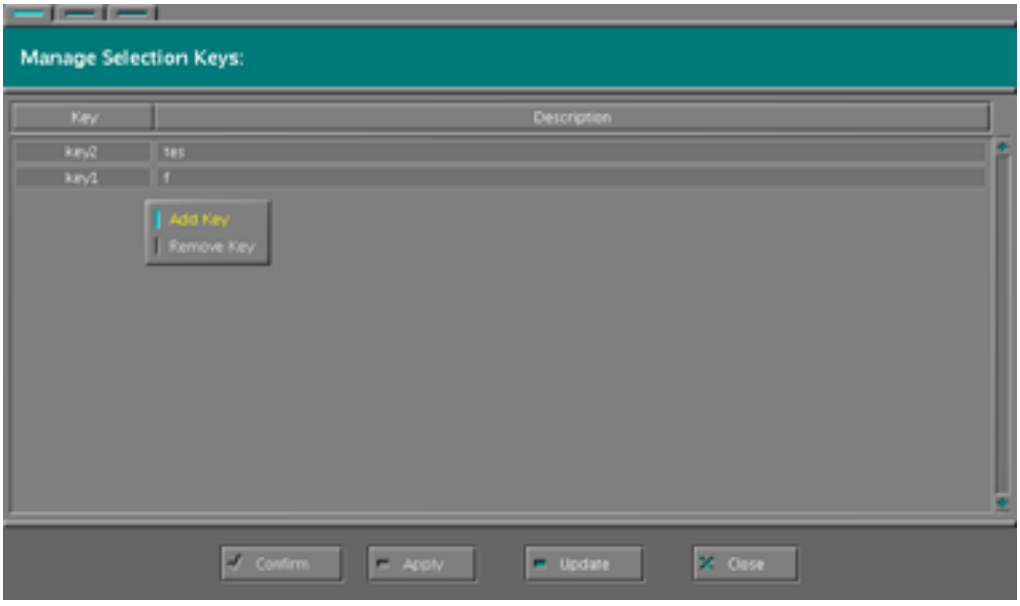
The next tab allows you to set up groups of selection keys. To add a group, press (mouse3) in the left hand column and select the Add Group option. To add a new group based on an existing group, press (mouse3) on the group you want to copy and select the Clone Group option. To remove an existing group, press (mouse3) on the group you want to remove and select the Remove Group option. The right hand side of the tab is a table, containing a value for each selection key for each group. Setting values here controls the priority of those selection keys for any machine that is in the group. The (-) value means that the group does not contain a value for that selection key and any job which uses that selection key will not be run on any machine in that group. A group where every key is set to (-) will only run jobs without any selection keys. A value in a field sets the priority of the key on that machine. Looking at the example on the right, a machine in group1 will favor jobs with key1




The Manage License Key dialog, showing the names of the keys listed down the left and keys settings to the right




The right hand side of the window, showing the configurations for the three different licensing schemes.




The first tab, showing the list of selection keys and the menu accessed by pressing (mouse3).

over jobs with key 2 over jobs with key3, while a machine in group3 will treat jobs with key1 and key2 the same, while ignoring any job with key3. Press the Apply button to save any changes you make to this tab without closing the dialog.

The last tab allows you to create schedule that will change what group a machine is in, depending on the time and date. To add a schedule, press (mouse3) in the left hand column and select the Add Schedule option. To add a new schedule based on an existing schedule, press (mouse3) on the schedule you want to copy and select the Clone Schedule option. To remove an existing schedule, press (mouse3) on the schedule you want to remove and select the Remove Schedule option.
Selection schedules are composed of a set of rules which define the selection group active for an interval of time. Selection rules are evaluated in a specific order. For a given point in time, the first rule which is active during that time will determine the selection group assigned to the job servers using the schedule. It is possible that more than one rule may be active at a given point in time, but the first active rule will always determine selection group. It is also possible for a rule to specify no selection group. In this case, the Group property of the associated job servers will also set to no group. Having no selection group is equivalent to a selection group with no selection keys set.

There are three types of selection rules:

• Default - This rule is always active and is typically the last rule in the schedule to define the selection group to use if no other rules are active.

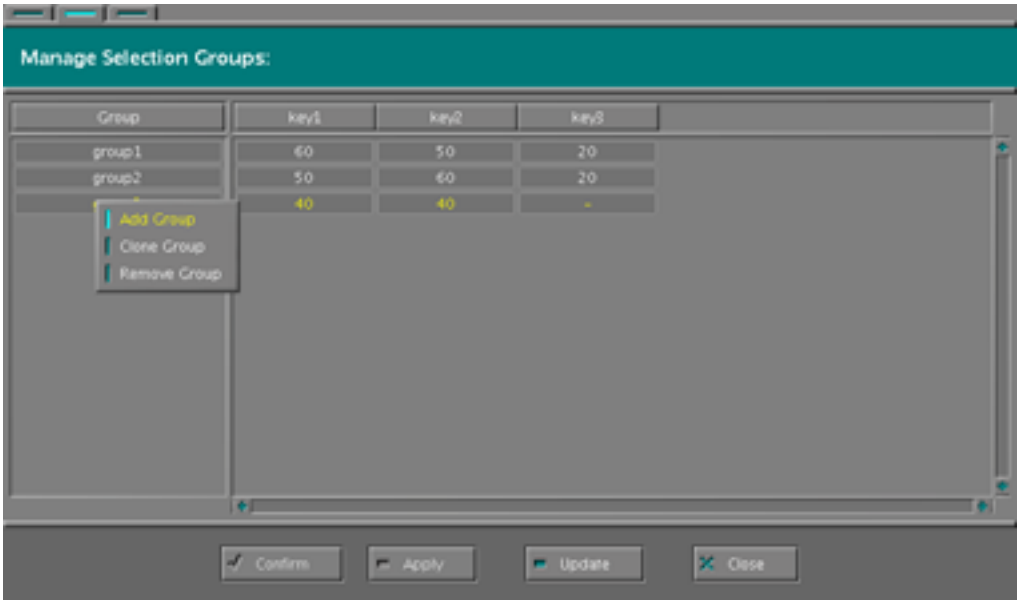
• Daily - This rule is active on one or more days of the week for an interval of time each day. This is typically used to schedule a selection group which should be used during working hours or other patterns of facility activity which are cyclic in nature and related to the day of the week.

• Specific - This rule is active for an interval of time on a single specific date. This can be used to schedule on-time changes related to holidays or special circumstances.

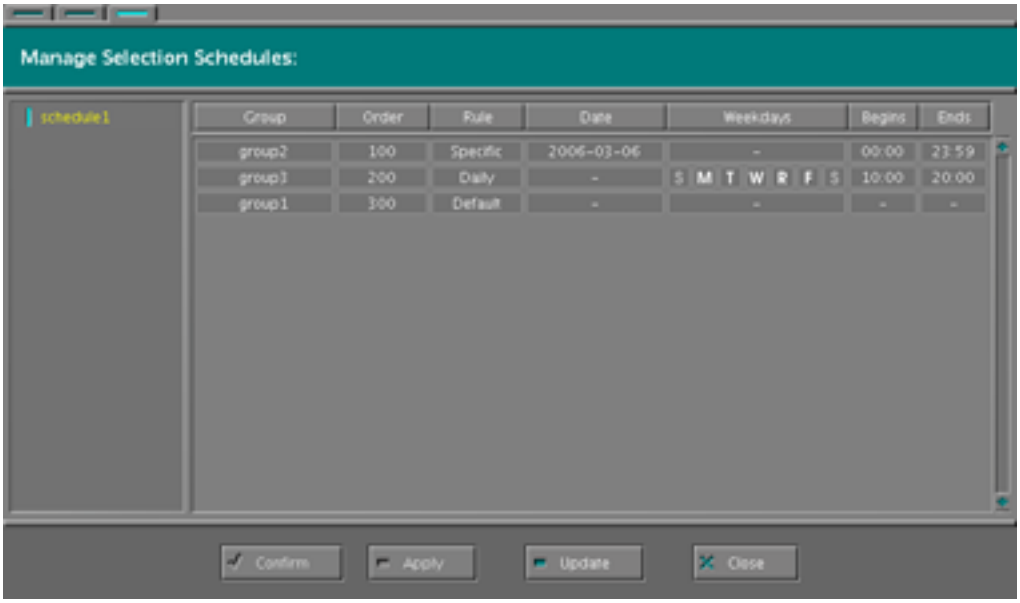
To add a new rule to a schedule, press (mouse3) on the right hand side of the tab and select the Add Rule item. To add a new rule based on an existing rule, press (mouse3) on the existing rule and select the Clone Rule option. To delete a rule, press (mouse3) on the rule and select the Remove Rule item. The first column in a rule is the Group field. This is a dropdown menu containing a list of all the selection groups and the null character. The next column is the Order. This parameter controls the order in which the rules are evaluated. The lower the order, the earlier it is evaluated. The next field is the Rule field, which contains a dropdown which lets you select the type of the rule. The next columns vary depending on what type you selected. If you selected Default, then all the remaining fields are blank. If you select Specific, then the Date field will be active, allowing you to enter a date and the Begins and Ends fields will also be active letting you set a start and end time for this rule. Finally, if you select Daily, the Weekdays option becomes active. Here you can click on the days of the week that you want the rule to be active for. You can also use the Begins and Ends fields to specify the time period on those days that the rule is active for. When a schedule is evaluated, Pipeline starts with the rule with the lowest Order and checks if the current time and date meet the conditions of the rule. If they do, then Pipeline assigns the group specified by the Group dropdown to any machine that is using the schedule. If the rule is not applicable, Pipeline continues to the next rule until it finds one which applies.

If you set a Default rule, it should always be the rule with the lowest Order. Any rule which ranks lower than the Default rule will never be evaluated. Specify rules should come before Daily rules, so that they override the everyday behavior.

Press the Apply button to save your changes without closing the dialog.




The Groups tab, showing the selection key values that are set in each group. Notice that group3 has a null value for key3.




The Schedule tab, showing the different rules that have been added. In the Daily rule, the days that the rule is active are highlighted.

# Appendix A: User Privileges

There are 5 different privilege levels in Pipeline, giving users different abilities to interact and configure with the program.

In order to control who has access to which settings and abilities Pipeline defines six different categories of user privileges. Every category has control over their own working area. Each category also has a varying level of control over the Pipeline environment, jobs in the queue, and other peoples working areas.

The most basic level is a normal user. Normal users can only interact with nodes in their own working area and can only control their own jobs in the queue. They can switch to other users' working areas to view what other users are doing, but have no editing capabilities in those areas. Similarly, they can view other users' jobs in the queue, but cannot affect them.

The next two groups of users privileges depends upon the idea of Work Groups of users. Using the Manage User Privileges dialog, users can be organized into named groups. Any user can be part of zero, one, or multiple groups. A user can be either a Manager or a Member of a Work Group. There are also two levels of user privileges called Node Manager and Queue Manager. These two settings, the privilege levels and the user groups, intersect to give some users the ability to edit the nodes and the queue jobs of other users. A user will have control over all the node operations of another if the user has the Node Manager privilege and is a Manager of a group which contains the other user as a Member. A user will have control over all the queue operations of another if the user has the Queue Manager privilege and is a Manager of a group which contains the other user as a Member.

A user with the Node Manager privilege is able to perform any of the commands in the Node Viewer and the node details panes that relate to nodes:

- Creating and Removing Working Areas
- Check-In, Check-Out, Lock and Evolve Nodes
- Link and Unlink Nodes
- Add and Remove Secondary Sequences
- Register, Clone, Export, Rename and Renumber Nodes
- Release Nodes and Add/Release Views
- Apply changes to nodes using the Node Details, Node Files and Node Links panels

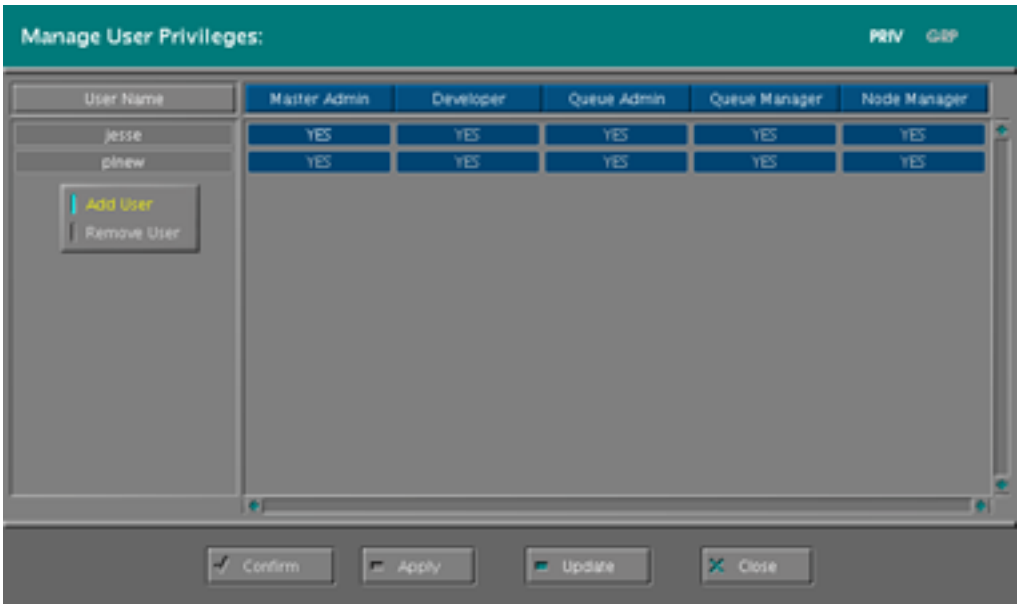
A user with the Queue Manager privilege are able to perform all the commands that effects jobs in the queue including:

- Queue, Pause, Resume, Preempt and Kill Jobs
- Remove Files

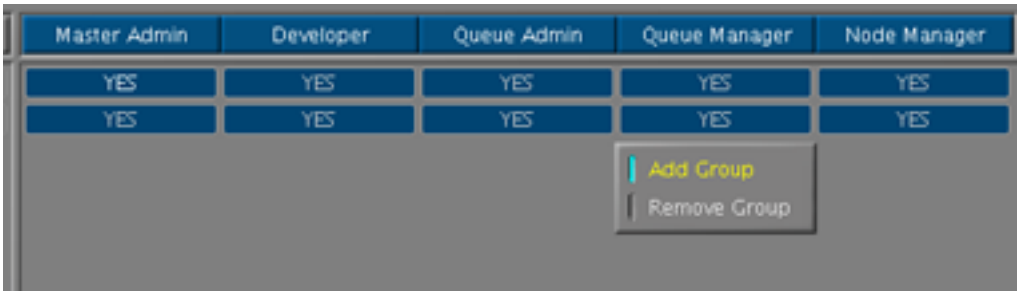
The Queue Administrator privilege gives a user the ability to:

- Add and remove job servers from the queue
- Adjust per-server status, order, slots, reservations, selection group and schedule
- Create and edit license keys and selection keys, groups and schedules
- Full control over all the jobs running in the queue.

A user with Developer privileges can install new plug-ins and modify existing plugins which are currently under development.




The Manage User Privileges dialog, showing the menu used to add and remove users. You can see the 5 different privilege types listed on the right.




The menu used on the right hand side of the dialog to add and remove groups.

User privileges can be adjusted in the Manage User Privilege dialog.

Finally, a user with the Master Admin privileges has all privileges of a Node Manager and Queue Manager for every user, as well Queue Administrator privileges and Developer privileges. In addition they also have access to all administration tools used to manage and configure Pipeline such as:

- Manage Work Groups and Privileges
- Archive, Offline and Restore Nodes
- Database Backup
- Manage Toolsets and Toolset Packages
- Delete Nodes

In order to adjust privileges for others, a user must have Master Admin privileges. The Manage User Privileges dialog can be found in the Admin submenu of the Main Menu.

The column on the left of the Manage User Privileges dialog lists all the users who have privileges assigned to them. In order to add a user to the list, press (mouse3) in that left hand column and select the Add User menu item. To remove a user, press (mouse3) on the name of the user and select the Remove User item.

To change the privileges for a user, simply click on the button that corresponds to the privilege that you want to add or remove. Some privileges are automatically added when you select another privilege. For example, selecting Master Admin will set all the other buttons to (YES). To the right of the privileges are the Work Groups. To add a group, press (mouse3) in the right hand side of the window and select the Add Group menu item. To remove a group, press (mouse3) anywhere in the group column and select the Remove Group menu item. For each user, you can set each group to either Manager, Member, or null. Press the Apply button to save your changes to the privileges.

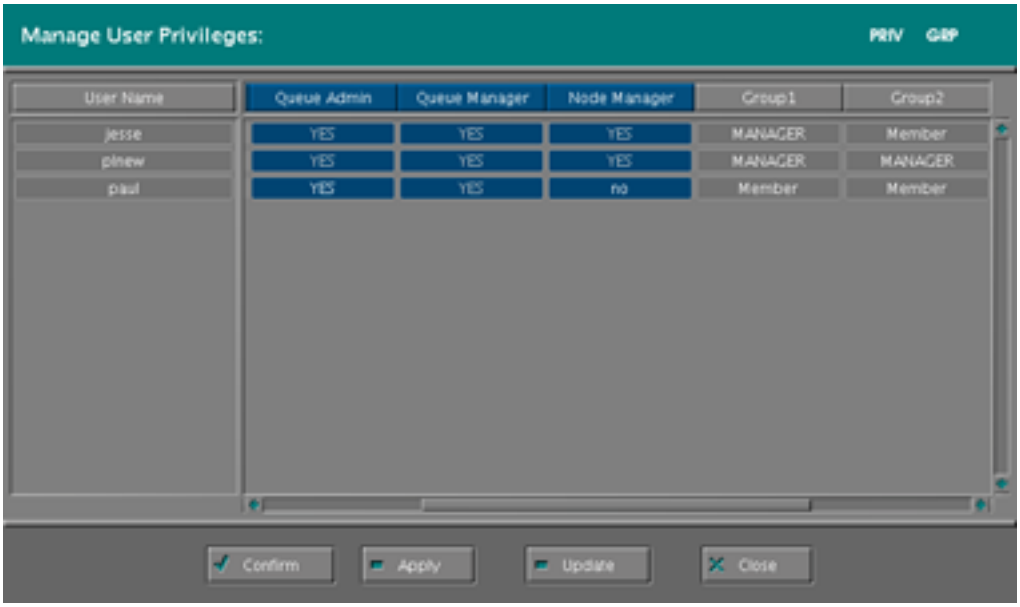
# Appendix B: Default Editors

The Default Editor dialog lets users specify which Editor is launched when the Edit with Default menu item is selected.

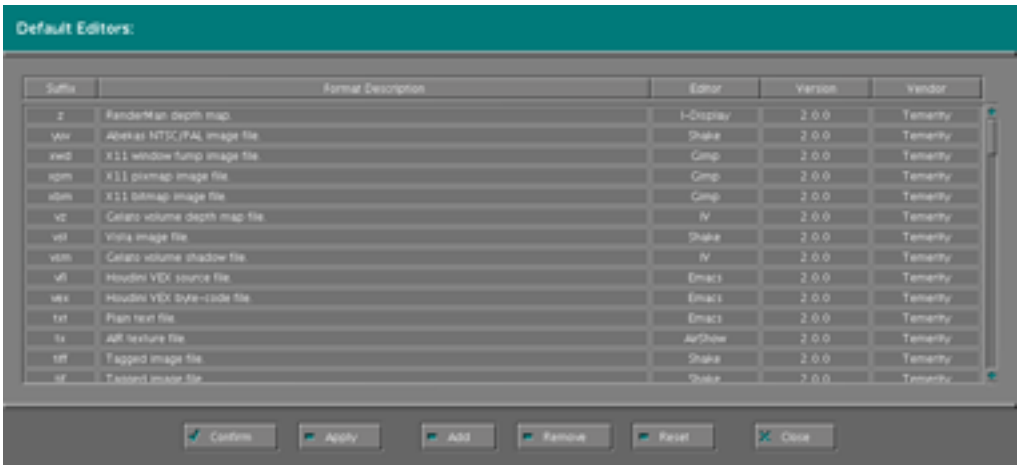
Default Editors are set on a per-user basis and control the Editor that you prefer for files with a certain extension. This editor is always available to you through the Edit With Default and View With Default menu items. Selecting one of those items from a menu will load up the target node or job in the editor specified in the Default Editors dialog. This functionality is important when you have file types that different users prefer to edit different ways. For example, different people may want to edit text files with emacs, GEdit, Kwrite, vim, or scite and it's obvious that not all of those can be set as the Editor assigned to the node. Having to hunt through the Edit With menu all the time is clearly less than satisfactory. But it is very simple to use the Edit With Default menu item or to create a hotkey for it.

To access the Default Editor dialog, select the Default Editors menu item from the main menu. The table in the dialog lists the suffix in the leftmost column and a brief description of what sort of file that suffix defines in the second column. The third column contains a dropdown menu that contains all of the Editors in the default toolset. Selecting an Editor from this list will set the Version and Vendor fields to the correct values for the selected plug-in. To add an extension to the list, simply press the Add button at the bottom of the dialog and enter the extension. You can then add in a description and select a default Editor for that file type.

The Default Editor is also what Pipeline will default to when you register an existing file as a new node. Pipeline will set the Editor in the Register dialog to whatever the default editor for the extension of the file being registered.




The dialog scrolled over to show two groups and users who are MANAGERS and Members of those groups.




The Default Editor dialog.

# Appendix C: Preferences

The Preferences dialog allows you to configure your Pipeline working environment.

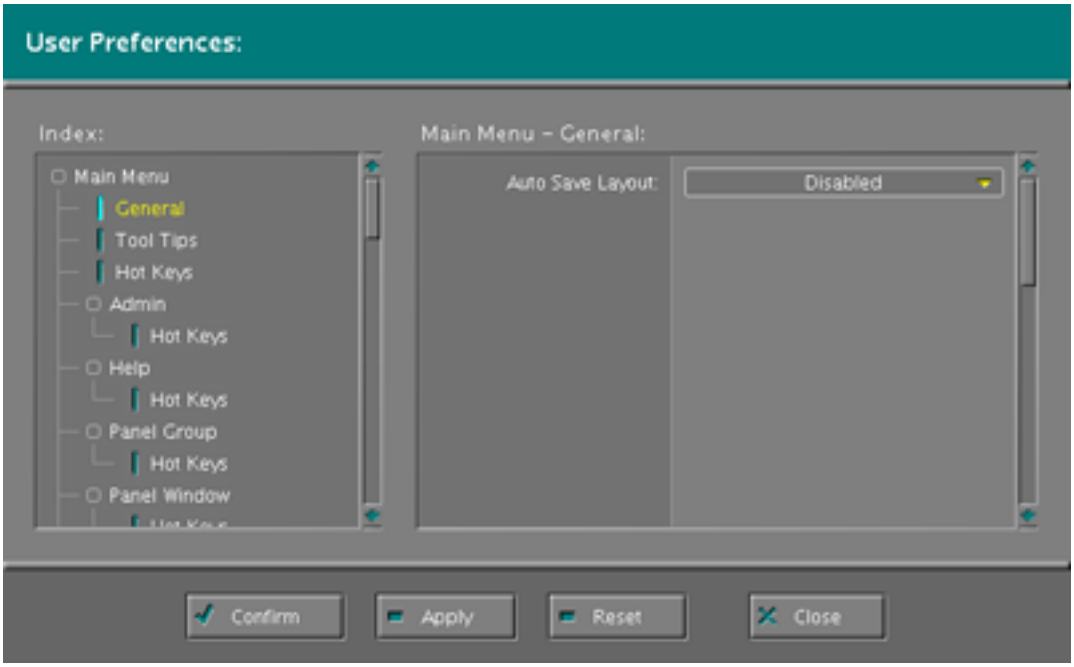
## Hotkeys

A number of the controls in this dialog involve hotkeys. Nearly every command in Pipeline has the potential to be a hotkey. Pipeline, by default, has a large number of hotkeys defined including some of those already discussed. Spacebar for status updates and Enter for Edit are the two most commonly used hotkeys. Hotkeys can be set in the Preference dialog by clicking on the field you wish to set a hotkey for and then pressing the key or combination of keys that you want to assign the command to. Pipeline will display the results of your input in the field. You can press the Apply button to save any changes that you've made.

It is important to remember that the same key can have different commands assigned to it not only in different panels, but also in different contexts. A hotkey will have a different meaning when it is pressed over an empty area of the Node Viewer than when it is pressed with the mouse pointer over a node. Pipeline considers these to be two different situations and lets you assign hotkeys to both. Since the Preferences dialog is broken down by panel type, it is easy to see what hotkeys are being used in a given context.

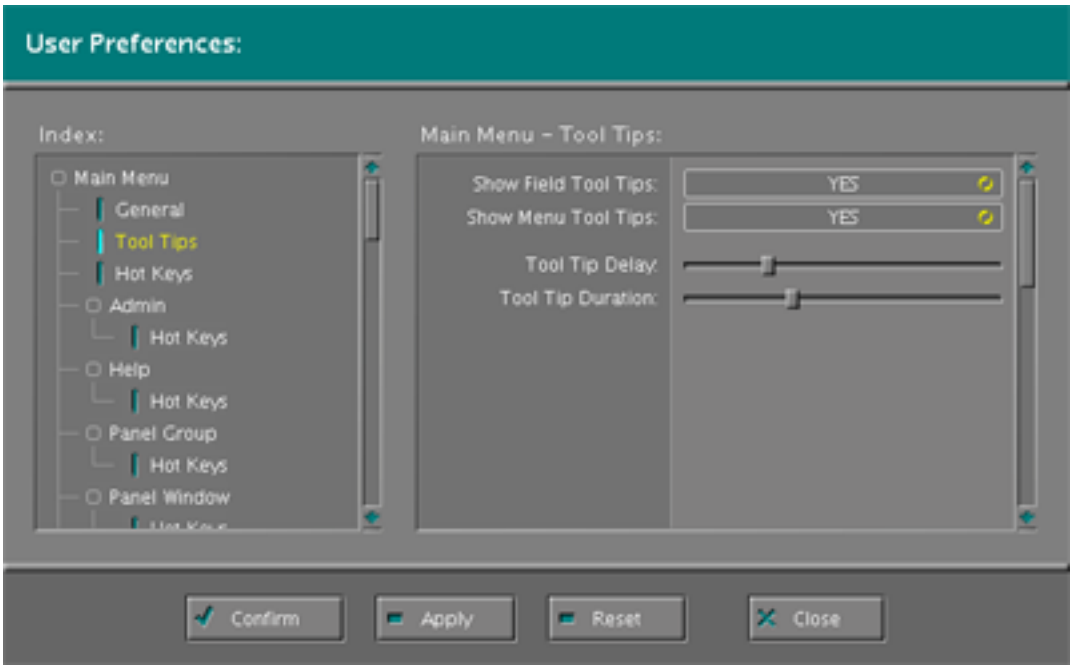
## Display Preferences

The rest of the preferences have to do with how Pipeline displays data, primarily in the Node Viewer. The different options are discussed below, with images showing the changes they can bring about.

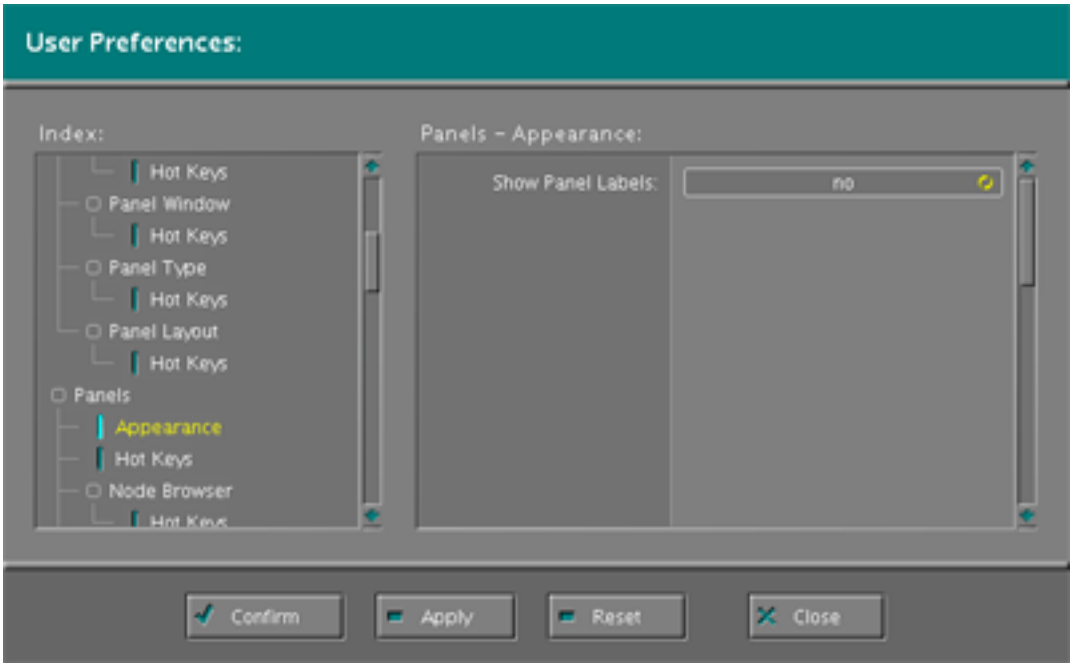


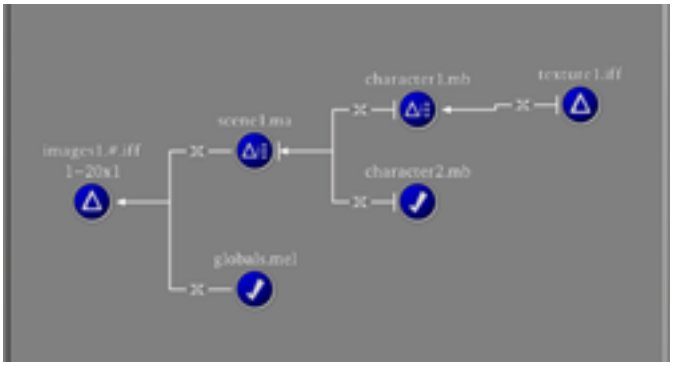

The User Preference dialog. Down the right hand side are a list of all the categories where you can adjust preferences. Any of the items which say Hot Keys are used to adjust the hot keys for the context they are displayed under. The other options are for primarily for controlling the display. The option listed here under the General category, however, has to do with saving layouts. If this option is set to yes, then every time you exist Pipeline, the layout will be saved and will then be reloaded when you restart Pipeline, allowing you to continue working right where you left off. When this is set to Disabled, Pipeline will not save your layout on exit and will load whatever layout you have specified as your default layout in the Manage Layouts dialog.

The Tool Tips menu controls if and when tool tips are displayed. If the Show Field Tool Tips button is set to (YES) then tool tips will be displayed when ever the mouse is held over a field, providing basic information about the value that field takes. The Show Menu Tool Tips will cause tool tips to appear in the Pipeline menus, briefly explaining a menu item. The two sliders control how long it takes for the tool tips to appear and how long they are displayed for.

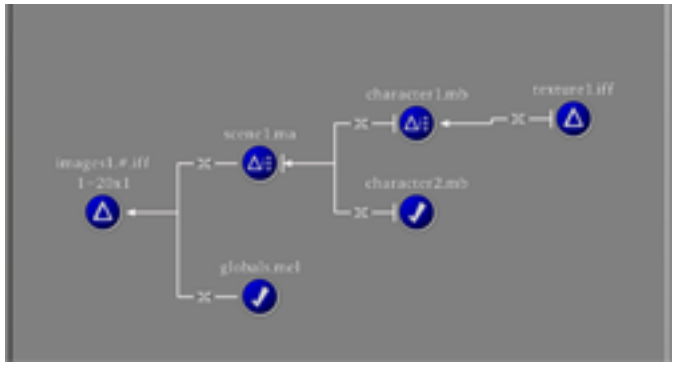



The Appearance tab has one control, the Show Panel Labels option. If this is set to (YES), then each Panel will have a bar at the top identifying what sort of panel it is. This can be useful for new users who are not comfortable with the different parts of the interface.

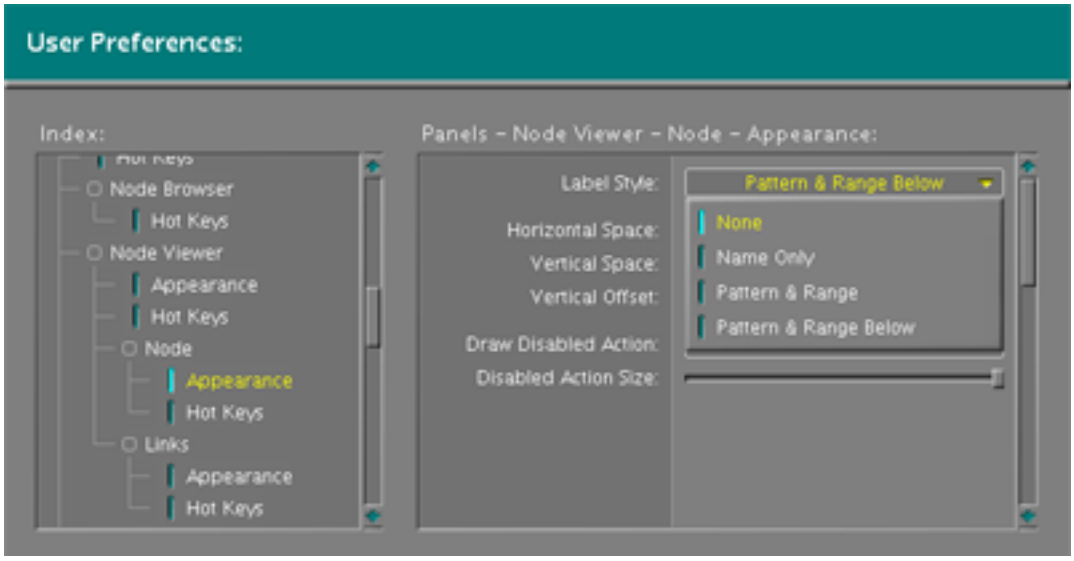




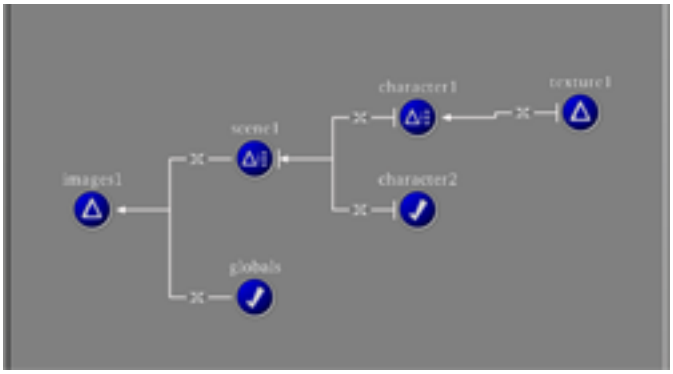

This is the standard Node Viewer panel,. with all the display options set to the defaults. This is the reference against which all the following examples can be compared.



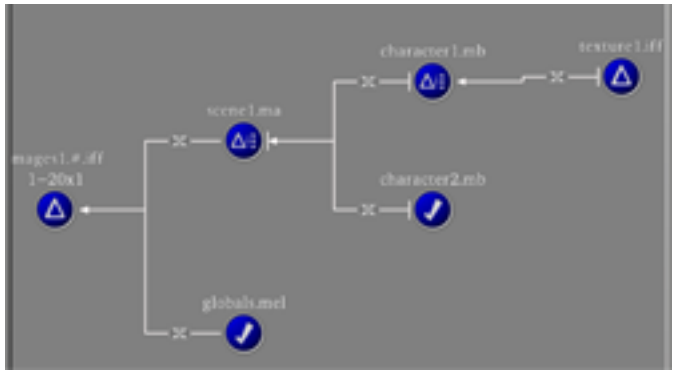

The same panel, now with the Panel Labels turned on.



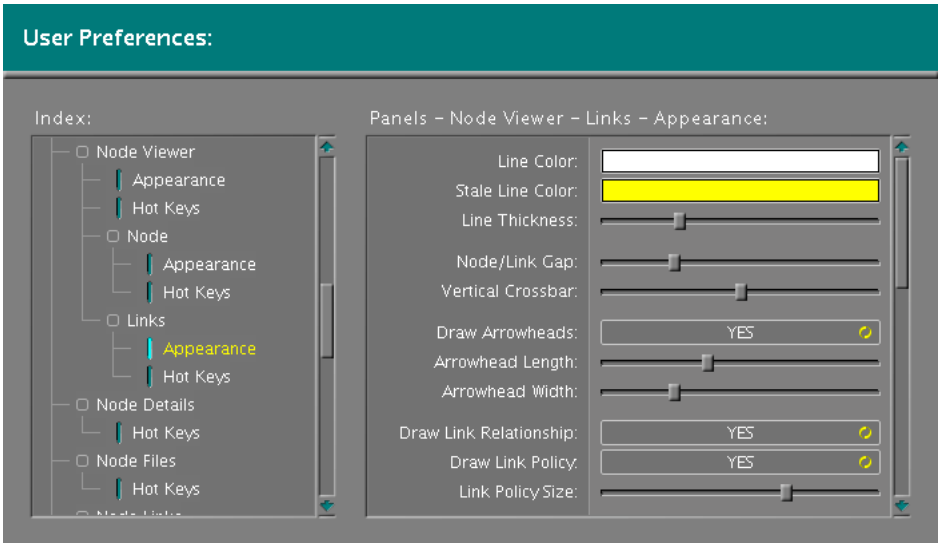

The Node Appearance preferences. The Label Style drop down allows you to tweak the text that is displayed with nodes. Setting it to Name will just display the name of the node, but not the frame range. Pattern & Range will display them in a single line, while the Pattern & Range Below option will display the range underneath the node name. None will turn off all the labels. The next three options control the spacing between nodes and how nodes are stagger vertically. The last two options control whether the Disabled Action indicator is drawn and if it is drawn, how big it is.



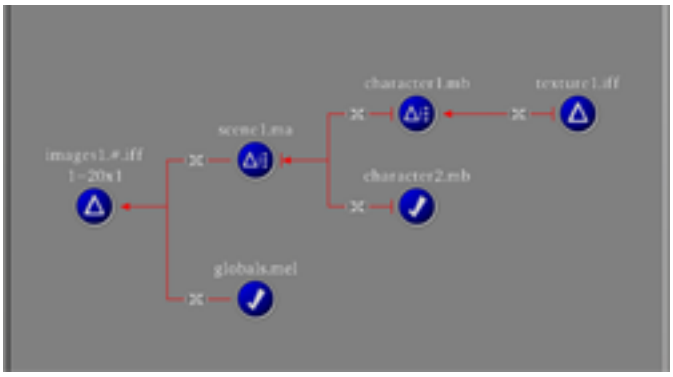

Nodes drawn with the Name Setting. No frame range info is displayed.



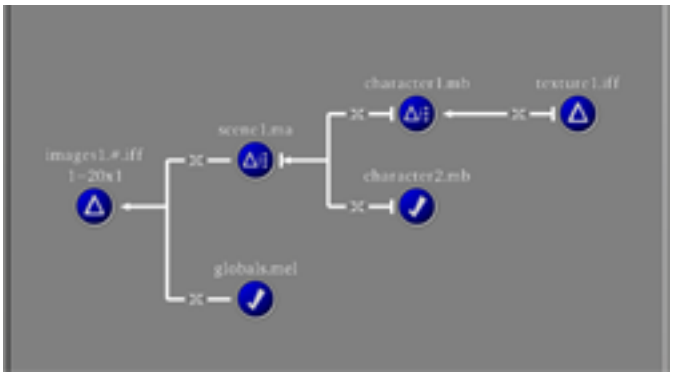

Increased horizontal and vertical spacing.



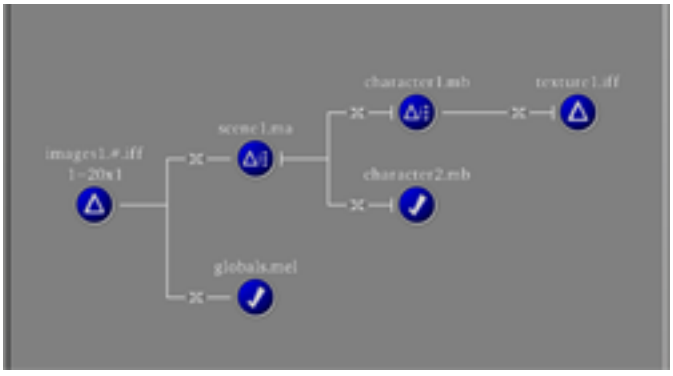

The Links Appearance preferences. These allow you to control how links are drawn in Pipeline. You can change the color of links and links denoting staleness, and set how thick the lines are drawn. You can turn off the arrows displaying the direction of the links and also turn off the symbols which show the different properties of the link. The symbol representing the Link Relationship can also be moved left or right using a slider



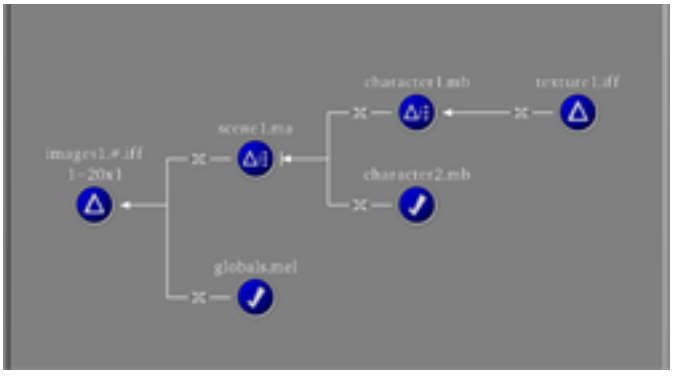

The link color has been changed to red.




The line thickness has been increased to the maximum.




The links are no longer displaying their arrows.




The Link Relationship indicator has been moved to the right (it's subtle).

# Appendix D: UI Visual Index

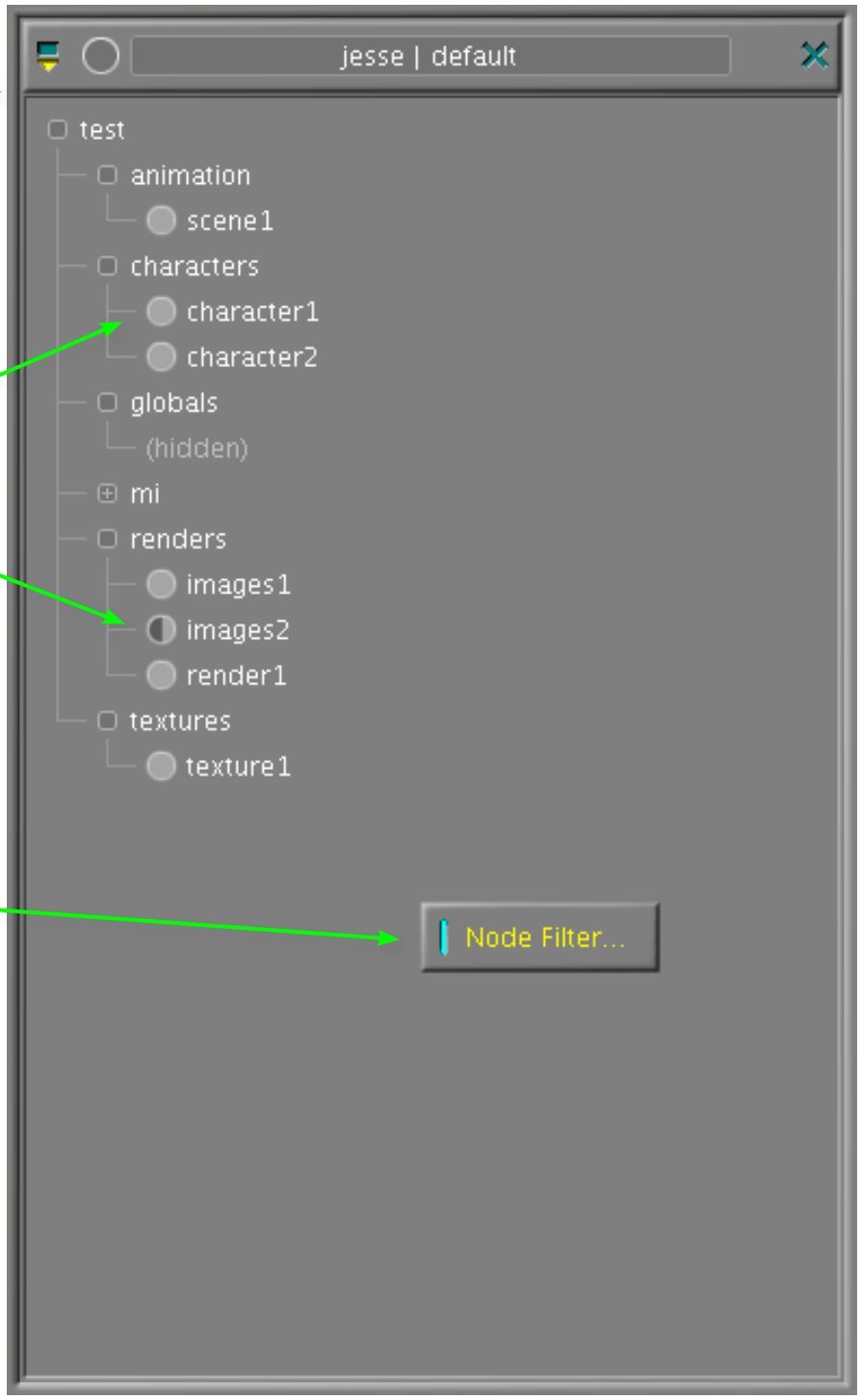
## Node Browser Panel

Basic information about this panel can be found in the *Node Browser Panel* section.

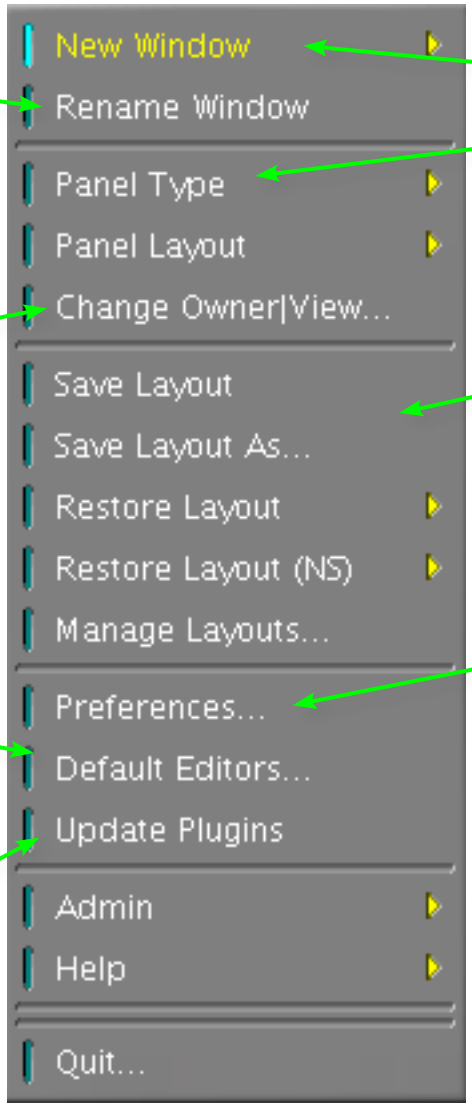
The icons to the left of node names indicate if the node is present in the current working area, other working areas, and the repository.

An explanation of what the symbols mean and how they can be used to filter the display can be found in the *Node Browser Symbols and Filtering* section.

Shows the Node Filter dialog for controlling which nodes are displayed in the panel.



# The Main Menu




Use this item to rename the current window. The benefits of window naming are discussed in the *Naming Windows* section.

The different types of panels that can be created using these two submenus are discussed in the *Panel Types* section.

The concept of working areas is addressed in the Revision Control Concepts section. Further discussion can be found in the *Multiple Working Areas* section followed by examples in the *Creating and Switching Working Areas* section.

Information on saving and restoring layouts can be found in the *Basic Layout Operations* section. A discussion about layout design can be found in the *Common Panel Layouts* section

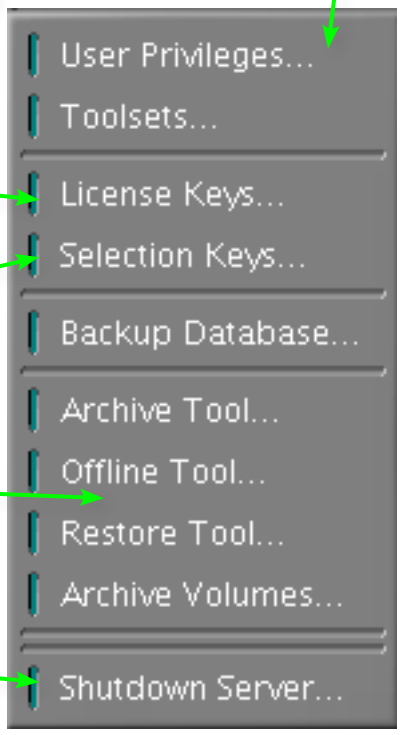
The Default Editors dialog is used to set the default editors for files based on their suffixes and is covered in *Appendix B*.

Shows the User Preferences dialog covered in *Appendix C*.

Refreshes the plugins for the graphical client, allowing the user to see any updates that have been made to the plugins.

Shows the User Privileges dialog described in *Appendix A*.

# The Admin Menu



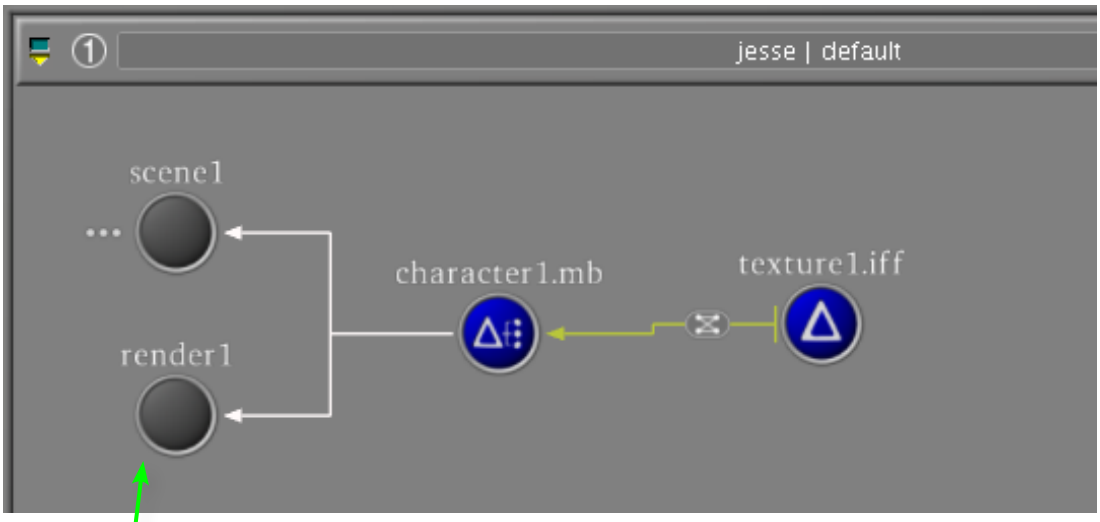

Shows the Selection Keys dialog discussed in the *Queue Score Parameters* and *Setting up Selection Keys* sections.

Shows the License Keys dialog discussed in the *Queue Score Parameters* and *Setting Up License Keys* sections.

These items display dialogs for managing the storage and retrieval of repository versions from offline storage.

Displays a dialog for shutdown all of the Pipeline servers. It has options for shutting down all the job servers and the plug-in server as well.

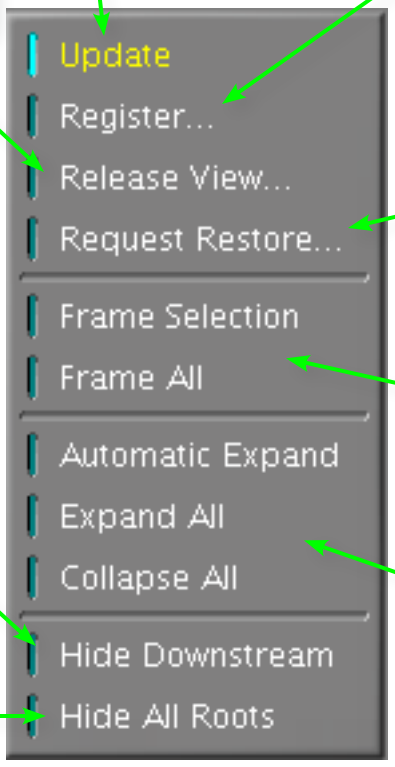
# Node Viewer Panel




Updates the Status in the Node Viewer and any linked panels.

The Register option is used to add new nodes to Pipeline. Information on it can be found in the *Basic Node Operations* and the *Links and Actions* section. The *File Sequences* section talks about valid node names.

This item is used to clean up an existing working area and is covered in the *Release View* section.




This option is used to file a request for offline files to be brought back online. See the *Offline Files* section for more information.

Downstream nodes can be made visible or invisible with this option the Node Viewer Menu. Downstream nodes are always displayed with the Undefined status . Information on this option can be found in the *Controlling the Node Displayed Section*.

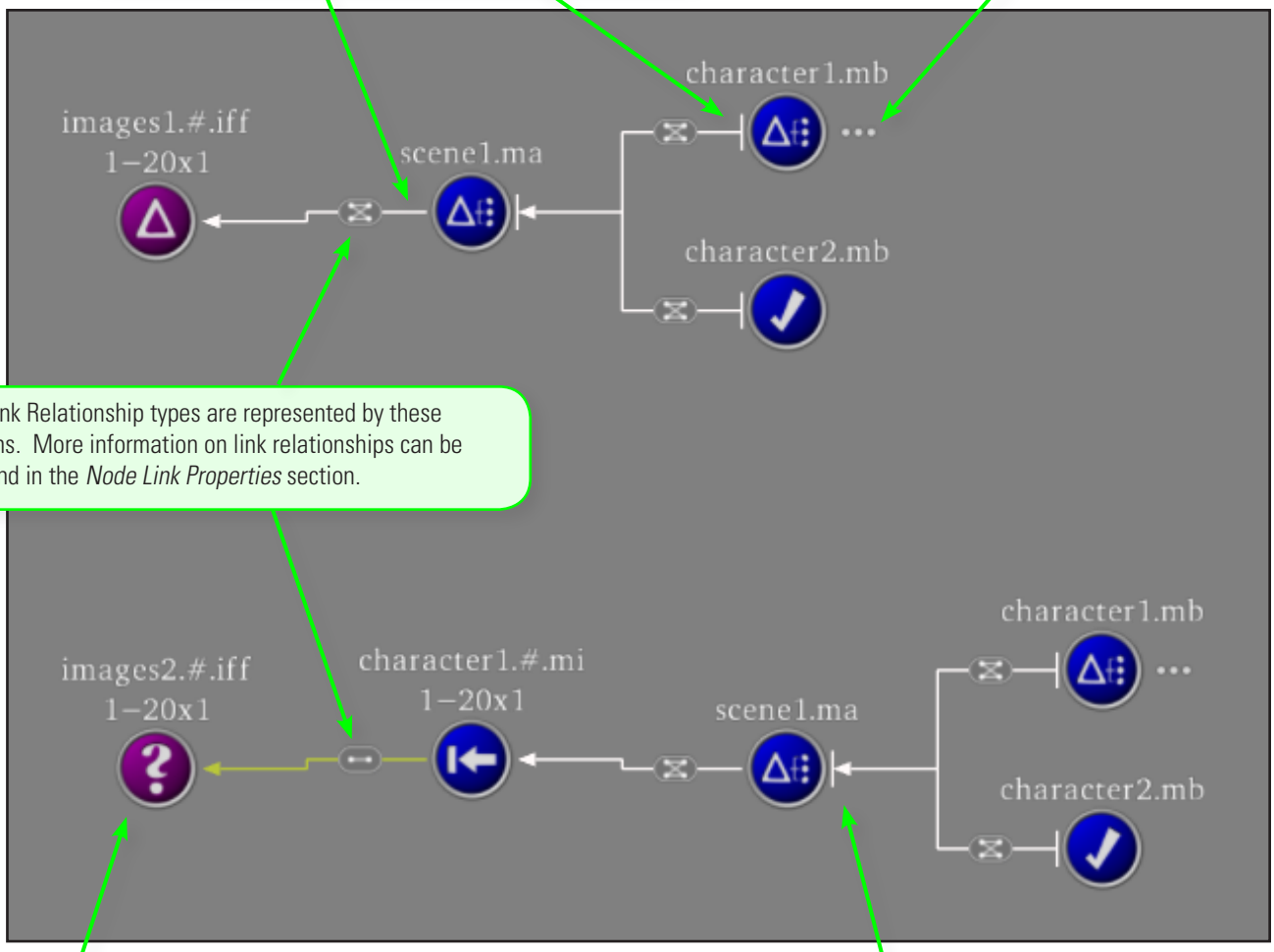
The framing of nodes controlled by these items are described in the *Viewer Navigation* section.

These items control complexity of node trees displayed and are covered in the *Collapsing/ Expanding Node Trees* section.

The Hide All Roots command is covered in the *Viewer Navigation* section.

A Dependency Link is shown between (images1) and (scene1). There is a Reference Link between (scene1) and (character1). The Reference link is differentiated by the vertical line appearing to the left of (character1). Information on links can be found in the *Node Links* section, the *Node Links Properties* section. Examples involving making links can be found in the *Reference Links and Source Parameters* section.

The ellipsis indicates a collapsed node tree. More information on collapsing and expanding node trees can be found in the *Collapsing/Expanding Node Trees* section.

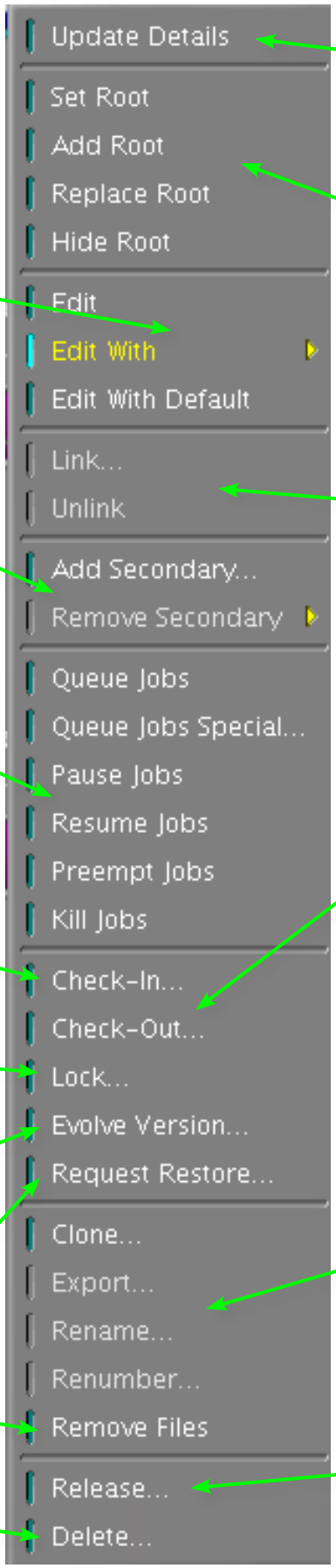



Link Relationship types are represented by these icons. More information on link relationships can be found in the *Node Link Properties* section.

The purple color indicates that the node is in the Stale state and the Action needs to be run again to regenerate the files. Information on Actions can be found in the *Regeneration Actions*, the *Actions*, and the *Leaf Nodes with Actions* sections. Information on the what the status colors and symbols mean can be found in the *Node Status* section. Specific information on what makes nodes Stale can be found in the *Queue State* section.

The vertical line to the right of the (scene1) node indicates a disabled Action. Details can be found in the *Disabled Action* section.

# Node Menu




Update all node detail panels to display information associated with the currently selected node.

These items control which nodes are visible in the Node Viewer. See the *Controlling the Node Displayed Section* for more info.

These items are used to launch applications to edit/view the data files associated with a node. Edit with Default uses the default editor set for the file suffix covered in *Appendix B*. Editors in general are covered in the *Editor* section.

Used to create and remove links. More information can be found in the *Node Links* section, the *Node Links Properties* section. Other examples involving links can be found in the *Reference Links and Source Parameters* section.

These items are used to control the secondary file sequences associated with a node. Details can be found in the *File Sequences* and *Adding and Removing Secondary File Sequences* sections.

These items are used to create new jobs and control existing jobs associated with the node. See the *Queue Jobs and Queue Jobs Special* sections for details about job creation. See the *Preemption* and the *Job Groups with Multiple Jobs* section for examples using these items.

Extracts a copies of nodes from the repository into the current working area. See the *Check-Out (Overwrite All)* and the *Check-Out (Keep Modified)* sections for details. Also see the *Frozen Nodes* section.

Commits a new version of the node to the repository. See the *Basic Node Operations* section for examples using Check-In. See the *Check-In Details* section for more information.

See the *Cloning* and *More Cloning* sections for details on the Clone menu item.

See the *Export* section for details on copying node properties.

See the *Renaming a Node* section for details on the Rename item.

The frame range of the file sequences associated with a node can be modified using the Renumber item. For details, see the *Renumbering Sequences* section.

Locks checked-out nodes. This item is covered by the *Node Locking* section.

See the *Resolving Nodes Conflicts* for more information and examples related to this item.

This item is used to request that currently offline files associated with the node be brought back online. See the *Offline Files* section for details.

See the *Removing Files* section for a discussion of when to use this option.

See the *Releasing Nodes* and *Releasing a Node* sections for more information on removing working copies of nodes with the Release item.

See the Deleting a Node section for information about permanent deletion of nodes.

# Node Details Panel

The Node Details Panel is where you interact with a node's metadata. The Panel is covered in detail in the *Node Details* section. Examples using the panel are in the *Regeneration Actions* section

Selects the execution environment detailed in the *Toolset* section.

Selects the Editor plugin used to edit/view the files associated with the node. See the *Editor* section for details.

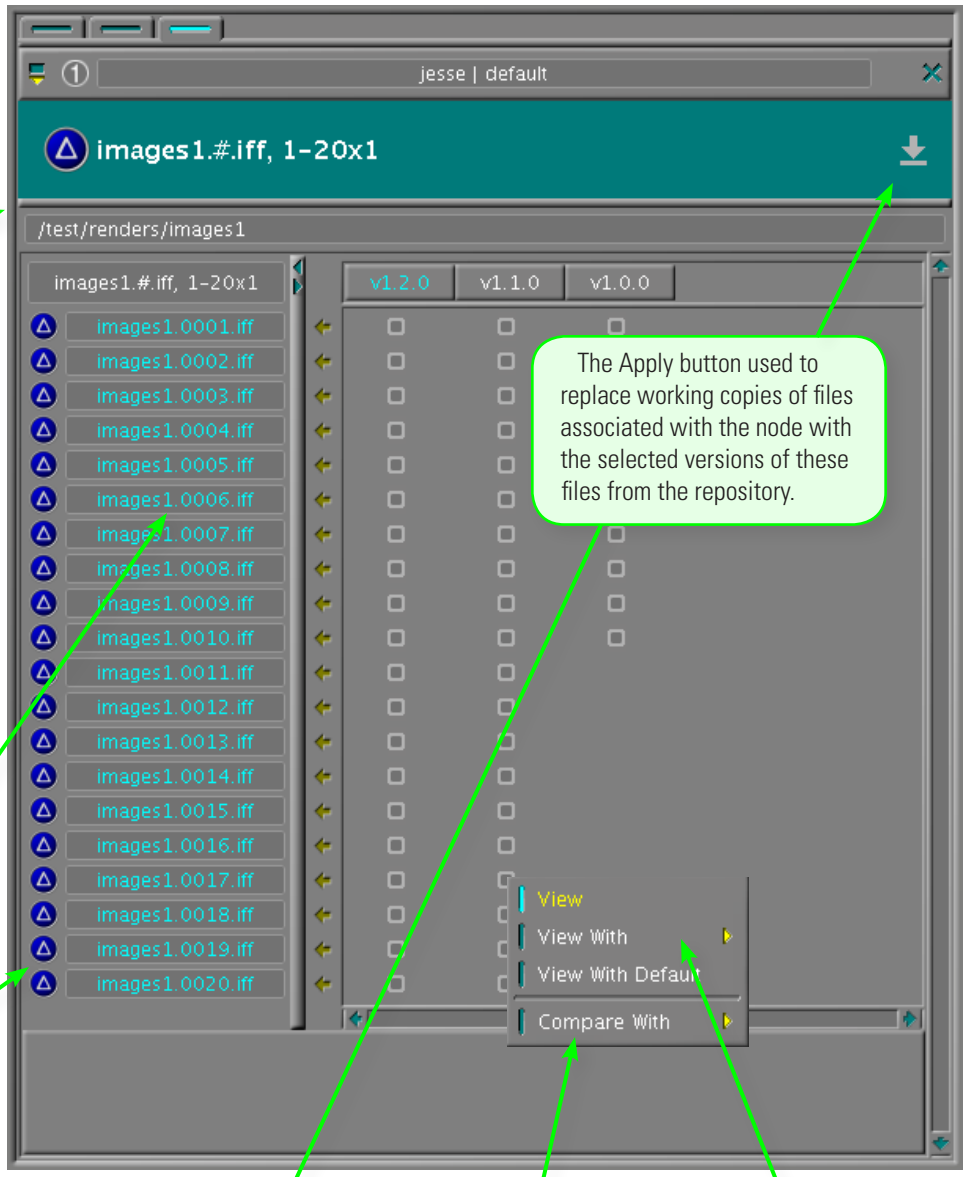
Selects the Action plugin and its parameters used to regenerate the files associated with the node. See the Action section for details. The Enabled parameter is discussed in the Disabled Action section. Source Parameters are covered in the Reference Links and Source Parameters section.

Setting Job Requirements, Selection Keys, and License Keys is covered in the *Queue Settings in the Job Details* section. An example of creating jobs from these requirements can be found in the *Job Submission Details* section.

# Node Files Panel

The Node Files panel allows you to interact with the individual files that make up a node. The panel is covered in detail in the *Node Files Panel* section and the *More Node Files Panel* section.

Editor's note: Show the selection of a few repository versions and some repository versions.

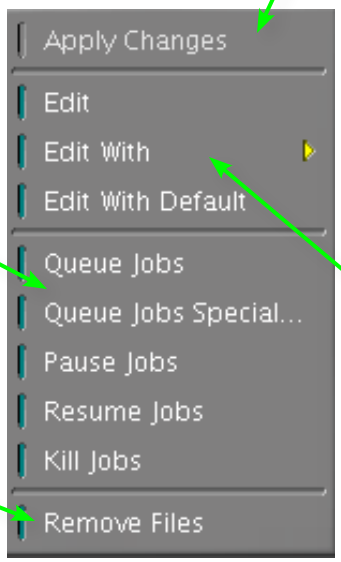



The Apply button used to replace working copies of files associated with the node with the selected versions of these files from the repository.

The names and states of the individual files which make up the files sequences associated with the node.

Information relating to queuing individual files can be found in the Batching When Queuing Individual Files.

See the *Queue Jobs and Queue Jobs Special* sections for general information about these queue related items for creating and controlling jobs associated with the node.

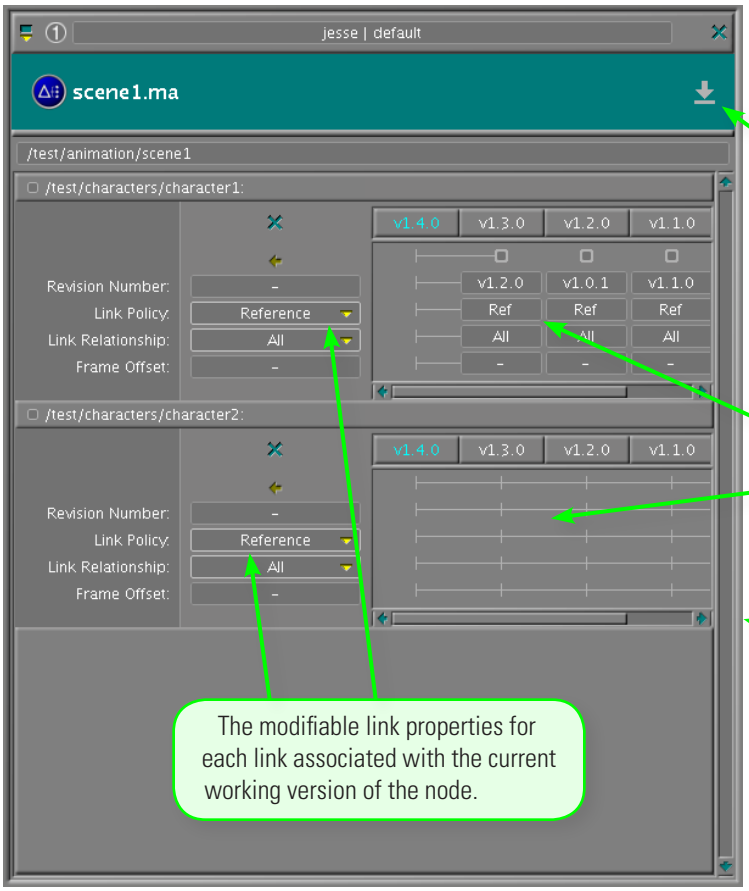



The Compare With item allows you to compare a repository version to the current working version using a Comparator plugin. See the *Comparators* section for details.

These items are used to launch applications to edit/view the selected individual data files associated with a node. Edit with Default uses the default editor set for the file suffix covered in *Appendix B*. Editor plugins in general are covered in the *Editor* section.

Remove Files allows a user to make a node stale by removing its files. See the *Removing Files* section for a discussion of when to use this operation.

# Node Links Panel




The Apply button used to modify the links of the current working version to match those of a selected checked-in version of the node.

The link properties for each of the checked-in versions of the node.

Displays the history of upstream links for all versions of the current node. The contents of this panel is explained in the *Node Links Panel* section and also in more detail in the *More Node Links Panel*.

The modifiable link properties for each link associated with the current working version of the node.

# Node History Panel

Shows the check-in log messages and other history information for each version of the current node. This panel is covered in detail in the *Node History* section.

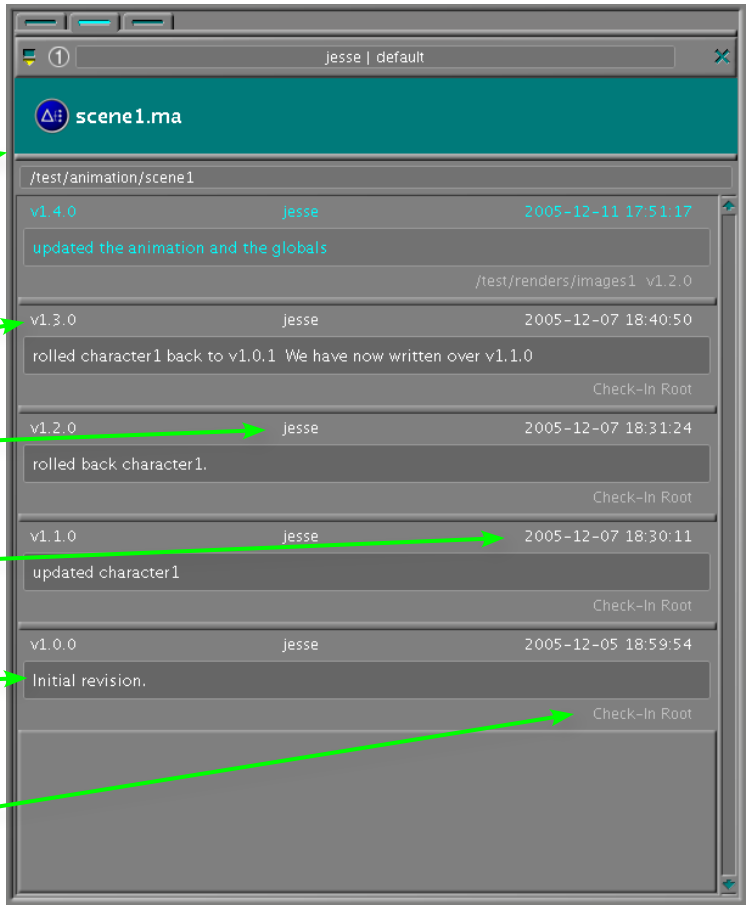
The version number.

The name of the artist which created the version.

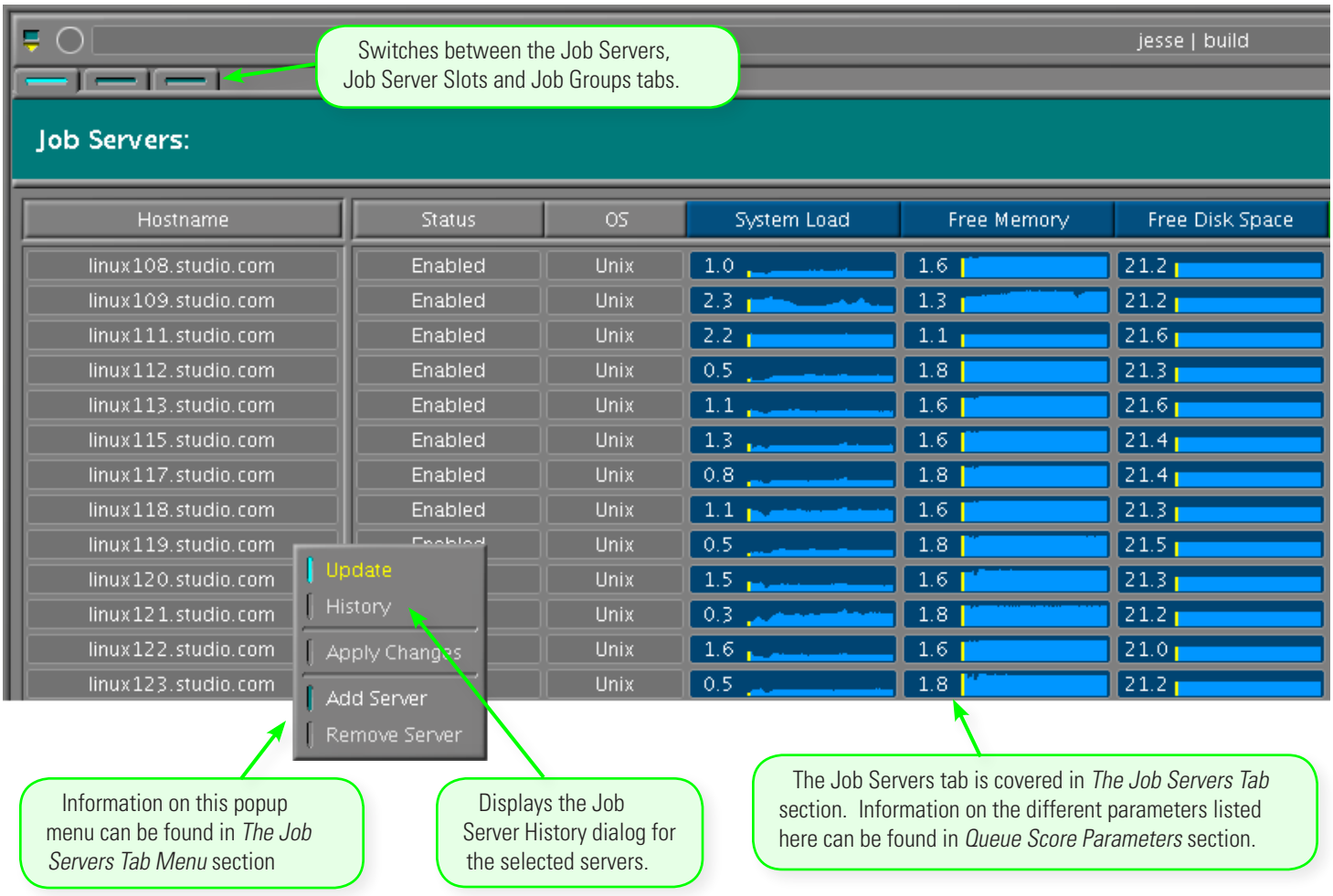
The date and time when the version was created.

The check-in log message explaining the nature of the changes since the last version.

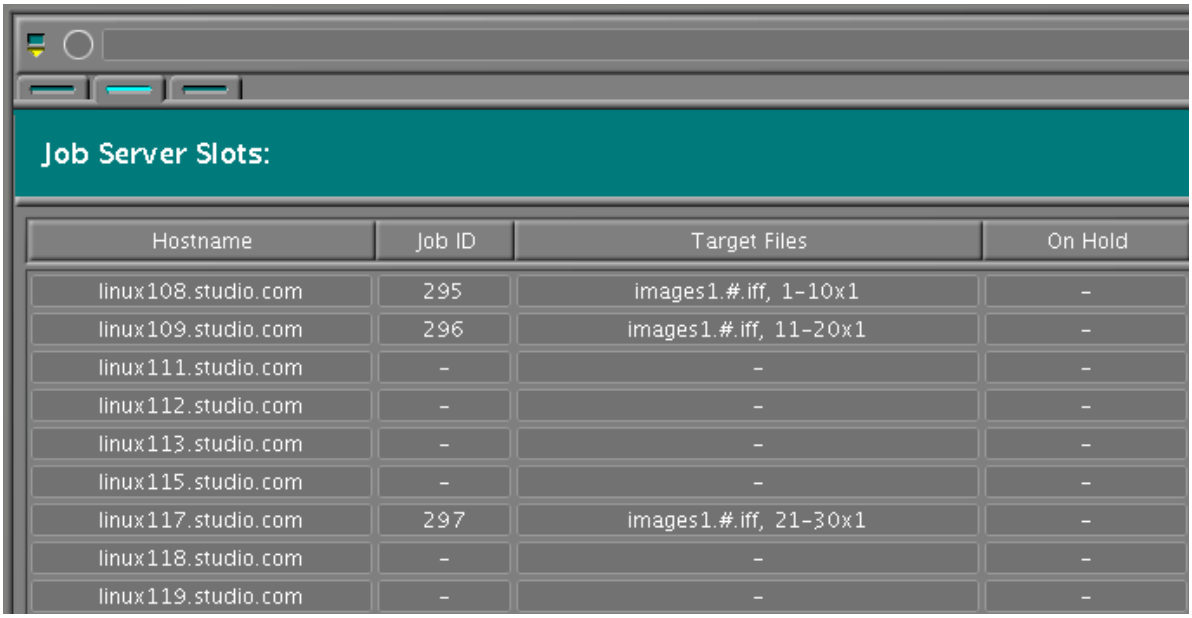
The root node of the check-in operation which created the node version.

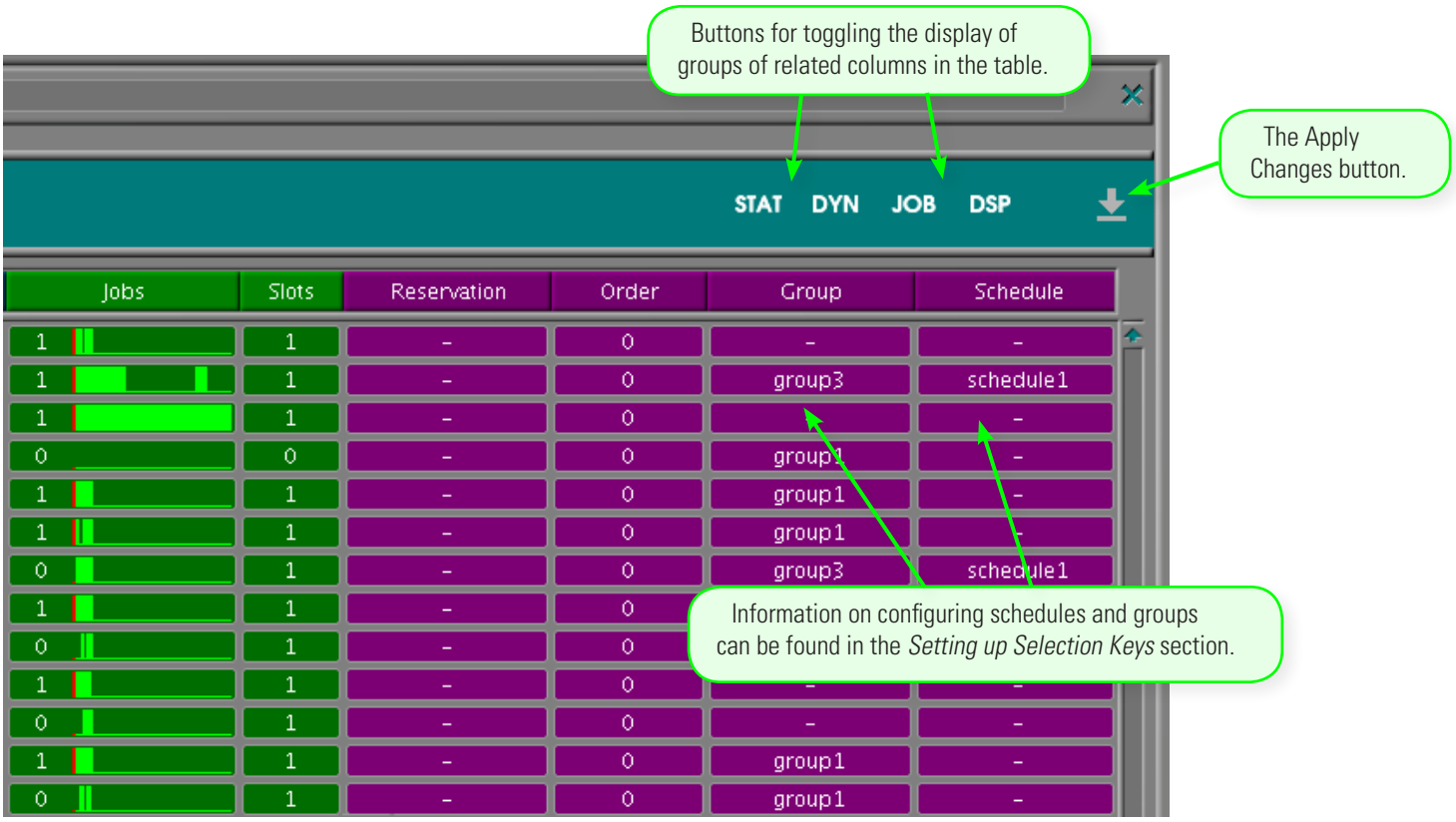


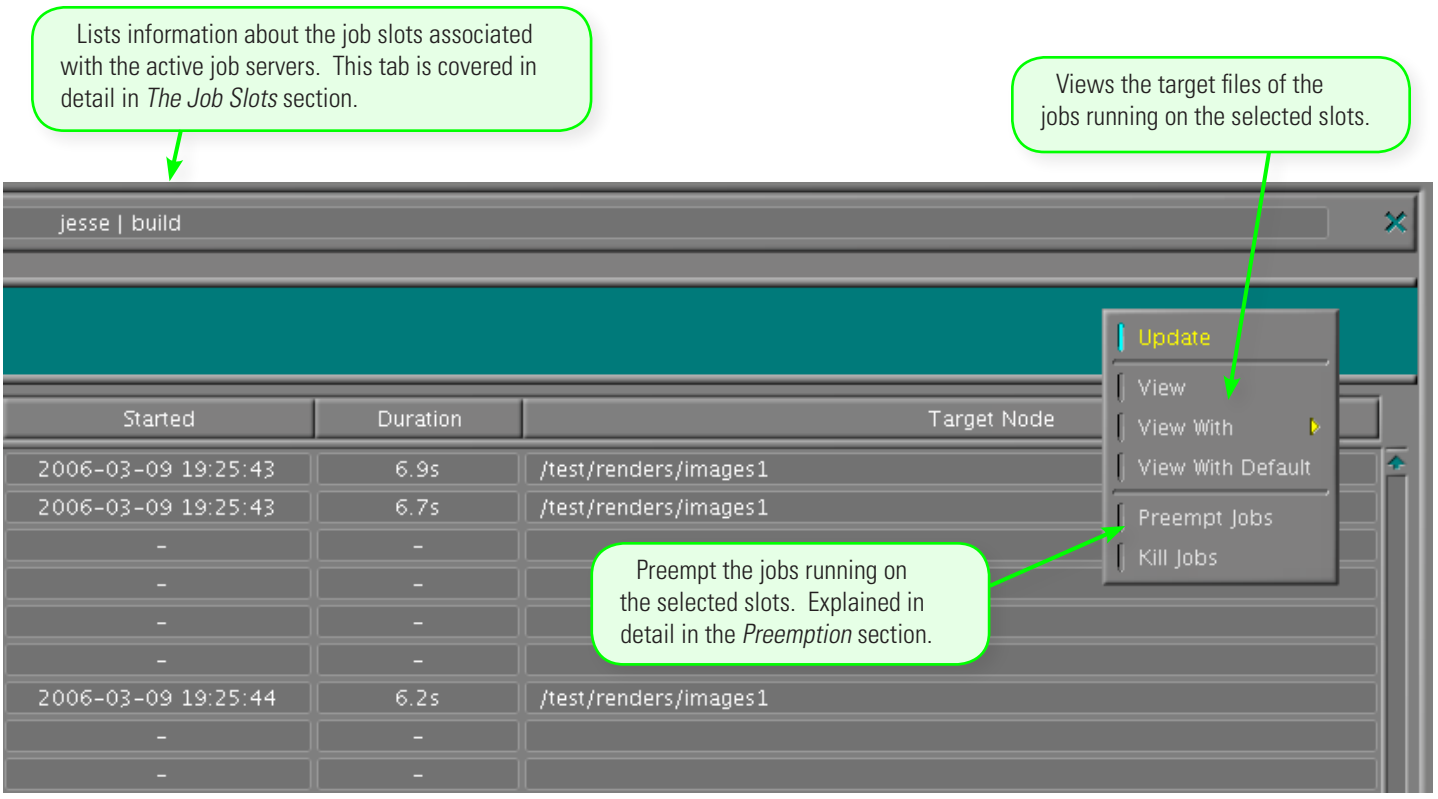
# Job Browser Panel: Job Servers Tab




# Job Browser Panel: Job Servers Slots Tab

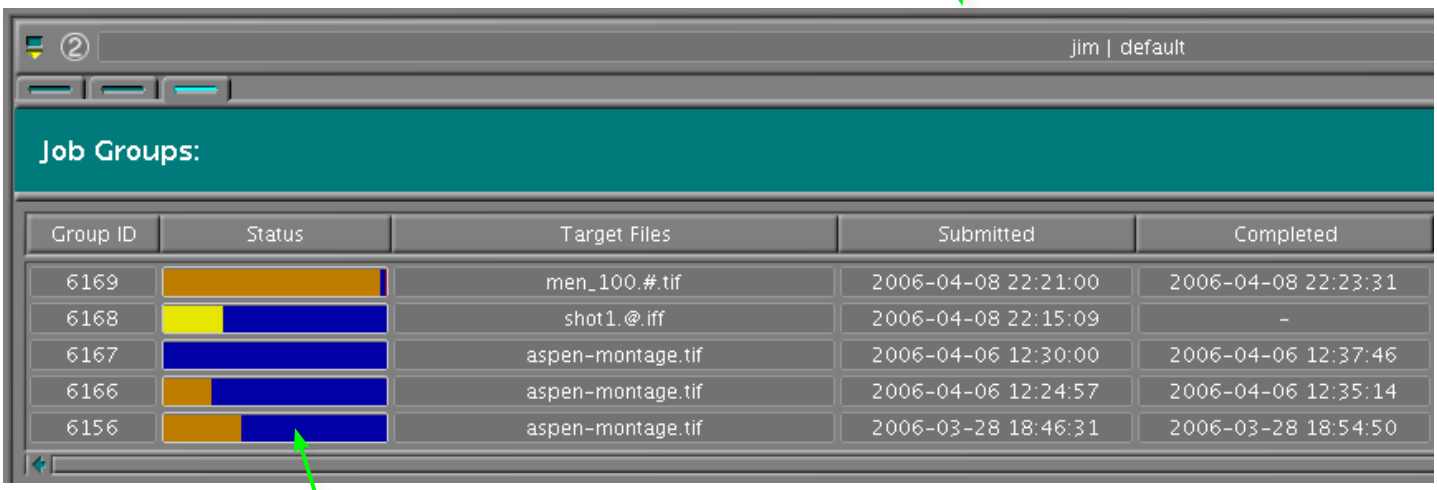




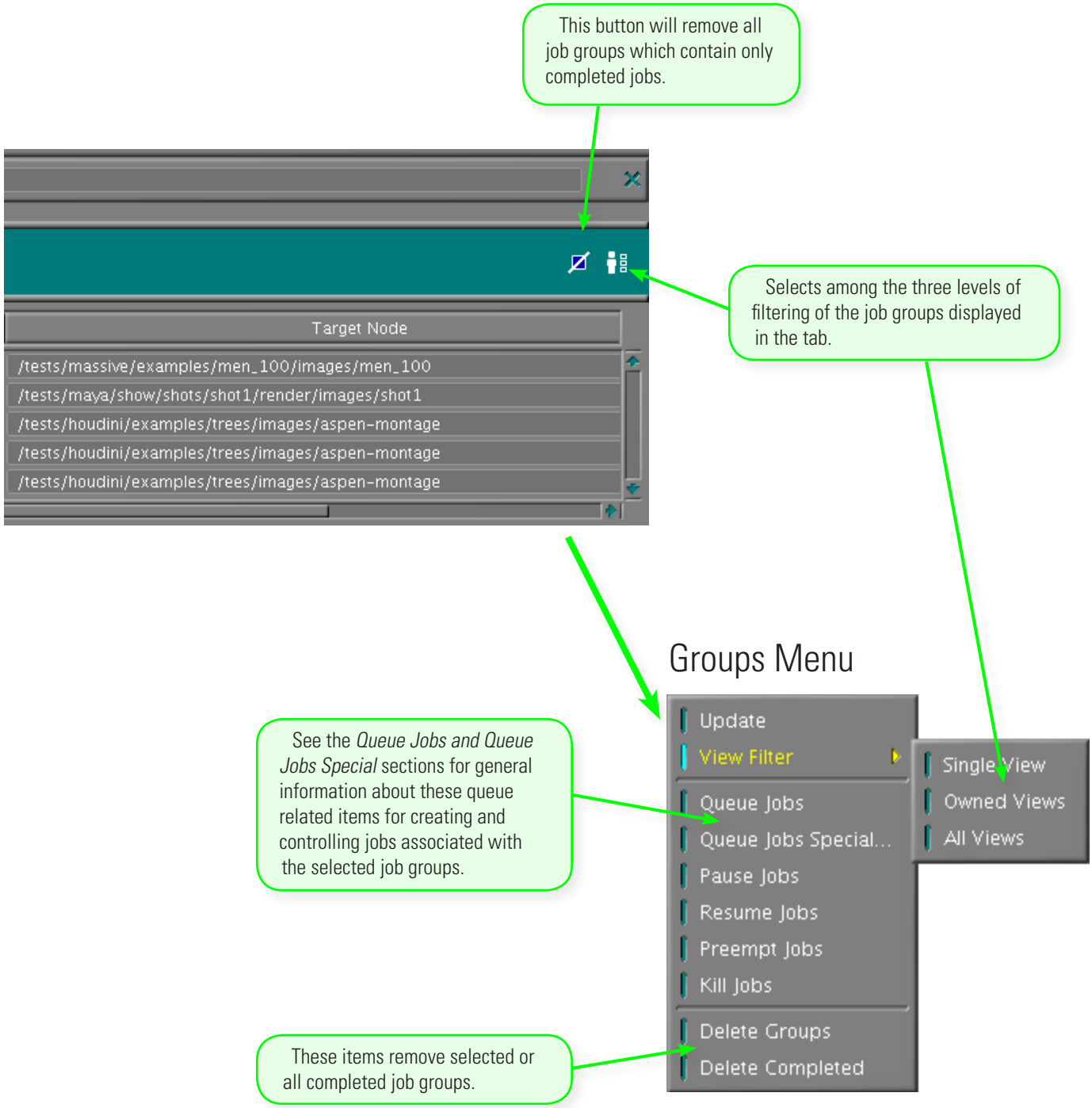



This tab lists information about the status of the currently queued job groups and is covered in *The Job Groups Tab* section.

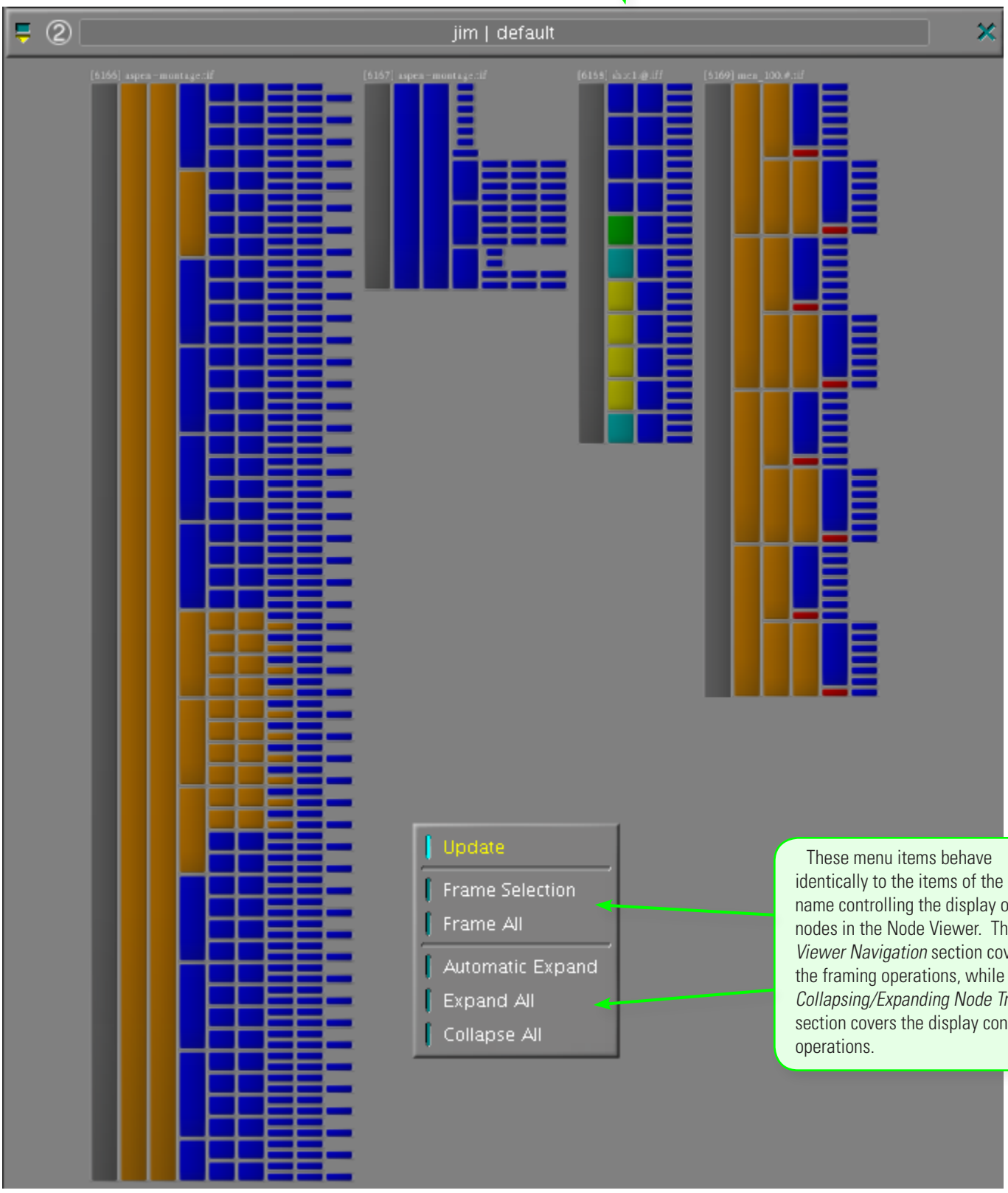
# Job Browser Panel: Job Group Tab




A summary of the queue status of the individual jobs which make of each job group. The meaning of the different queue state colors is covered in the *Queue State* section.

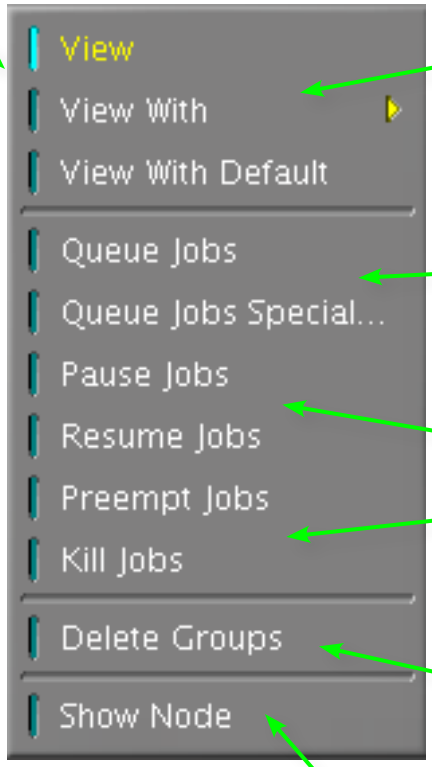


# Job Viewer Panel



This popup menu is displayed when the mouse is over a job or job group in the Job Viewer panel.

## Job Viewer Job/Group Menu




These items are used to launch applications to view the target files created by a job or job group. View with Default uses the default editor set for the file suffix covered in *Appendix B*. Editor plugins in general are covered in the Editor section.

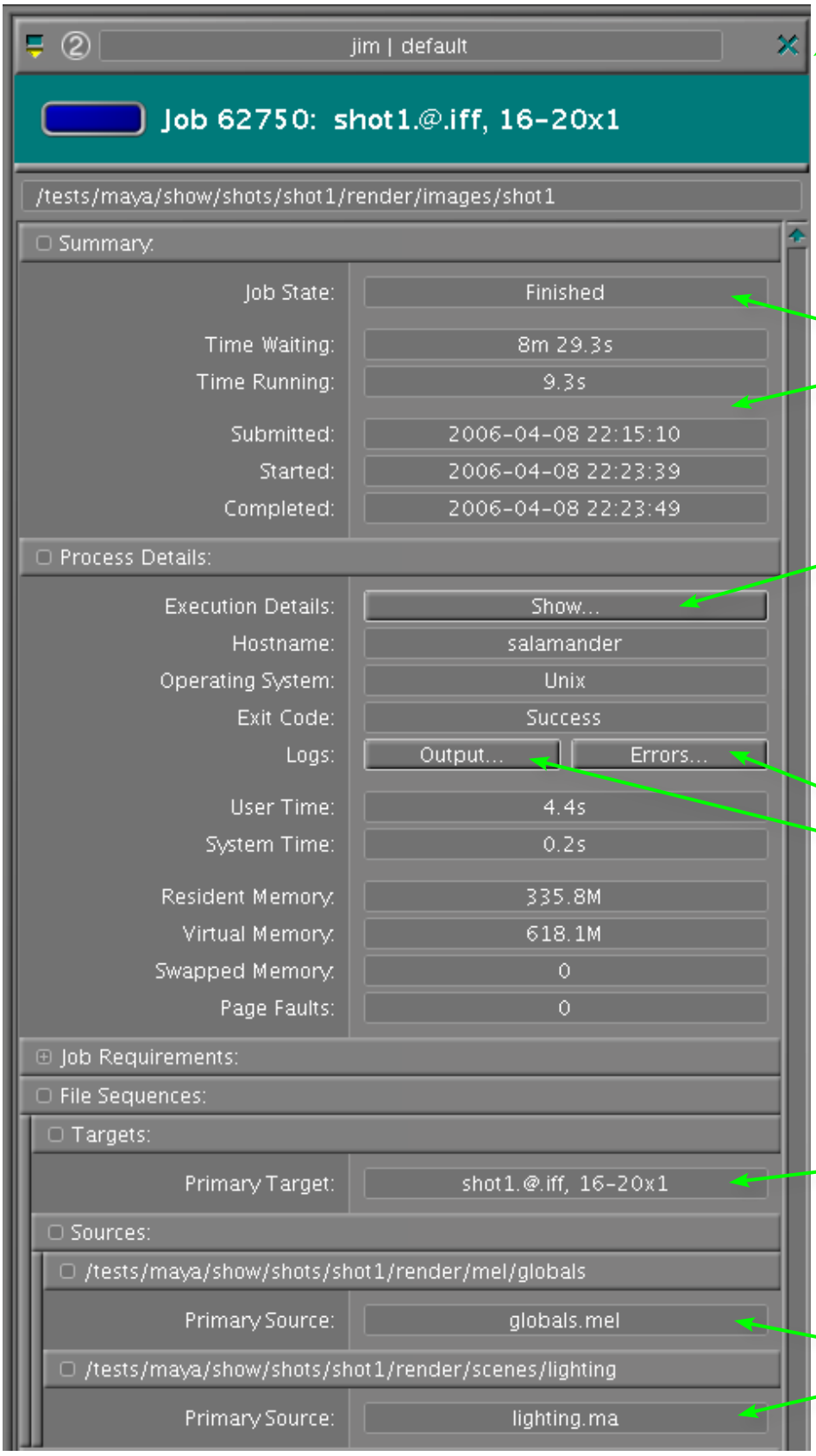
These two items are used to resubmit jobs to the Queue. See the *Queue Jobs and Queue Jobs Special* sections for details.

These items provide control over the selected jobs. See the *Preemption* and the *Job Groups with Multiple Jobs* section for examples.

Deletes the selected job groups for which all jobs have already completed.

Causes the node that this job is running for to be displayed in a linked Node Viewer, if one exists.

# Job Details Panel




This panel displays detailed information about the execution of an individual job. See *The Job Groups Tab* section for details.

The current queue state and overall timing statistics for the job.

Shows a dialog which displays the working directory, exact job command line and the values of all environmental variables under which the job was executed.

These two buttons display dialogs used to monitor the standard output and error files generated by the job.

The target files created by the successful execution of this job.

The source files used by this job in order to regenerate the target files.