SMEDGE

User Manual

Smedge 2018 Update 1

Table of Contents

Introduction

Overview of Smedge

Welcome to Smedge!

Smedge is a system that allows you to queue and distribute tasks across multiple computers. It is designed to be able to control anything that you could possibly imagine. If your task can be broken into parts, the rest can be done with Smedge. Currently, Smedge is geared around the queuing and distribution of image processing and rendering tools, but there is no limit to the possibilities. Smedge runs on Windows, Linux, and OS-X, and can seamlessly interconnect between all supported platforms. Any component can be run on any platform.

Smedge hides most of the complexity of configuring and managing the distribution, making it easy to get up and running without needing a lot of technical knowledge. At the same time, the underlying architecture allows nearly infinite customization in how the system operates, how it interacts with the processes that it controls, and how it interacts with your network and your user and file systems.

By default, when you start Smedge, you are actually starting 3 different components of the system that all interact with each other and with other Smedge components on your network. The **SmedgeMaster** is the process that manages the system. The **SmedgeEngine** is the process that actually performs the work. The **SmedgeGui** is the graphical interface that allows you to monitor and manage the system and the work that it executes. These components automatically find each other and integrate on every machine on your network.

As you run, Smedge will tune itself to your needs dynamically, using the Automatic Engine Mode. If you don't use the GUI for a while, it will disconnect itself, and shut itself down as a backup master, preferring to run as an engine, which uses much less of the CPU, memory and networking on the machine for Smedge overhead. Of course, you can manually configure the components directly, including integrating with your system's automatic process control, usually called "Services" or "daemons", allowing a machine to be available for doing work from the moment it boots up, even without any user logged into the machine's local console.

Smedge Terminology

Master

The Master is the central manager of Smedge. Being its own process, the Master is protected from problems that may crash other component applications or even other machines. As long as at least one Master is running somewhere on your network, everything will connect through it and you can use the system.

The Master keeps the complete current status of all system information saved to disk. If the Master goes down, all you have to do is restart it, and everything will pick right back up where it was. The Master also works redundantly across your network. Each machine that is running the SmedgeMaster process can take over as the Master for your system automatically if there is any problem. All of this is seamless and nearly invisible to the users.

Normally, all Smedge components should be able to find the Master automatically, but you can specify the actual machine address and TCP port, if you wish. For more information see Configuring the Connection. This is especially useful if you are trying to connect to a Master that is on a different subnet.

You can use the Shell applications, including the SmedgeGui program, to control many aspects of the Master's operation. In general, these sorts of commands are available only to Smedge Administrators. For more information, see the Administrative Control.

Engine

The Engine is the worker bee of the Smedge system. Any machine you want to perform work needs to have the SmedgeEngine process running on it. The Engine uses the Job information and the Module to actually perform the requested work, and then reports back to the Master about how it went. While most of the current Modules interface with third party tools operated by a command line, the Module could, in theory, do just about anything, including all of the processing itself, or using any form of inter-process communication to a third party process. All of this is handled identically and invisibly for you.

You can use the Shell applications, including the SmedgeGui program, to control many aspects of Engine operation.

Shells

Shell applications are what allow users to interface with the Smedge system. A Shell is actually a class of application, not a specific application. **SmedgeGui** is a Shell, as are **CheckFileSequence**, **Job**, and **Submit**.

The capabilities of Shells are virtually unlimited, but in general Shells will contain a sub-set of the features available in SmedgeGui. The Gui is designed as the main shell control for normal use. It allows you to see the current status of all Engines, Jobs and Work, and allows you to change everything, including the Master's operations. See the Operation chapter for complete details.

Jobs

A Job is the basic unit of Smedge processing. You create Jobs with enough information to know what to work on and how to divide it. The details of this information are managed by the specific Module that provides the Job type. Basic Job information includes the details required to know how to queue and prioritize Jobs in the system. Jobs can also be organized and prioritized by using Pools.

For information on how to submit Jobs, see Adding a Job, the Submit Job dialog reference, and the Submit Manual.

Modules

Modules are dynamically loadable plug-ins that extend Smedge functionality. The primary use of Modules is to provide the details of Job operations and interface. The Shell uses the Module to get the necessary information from the user, or display the information about the Job to the user in a manner that makes sense for the Job type. The Engine uses the Module to know how to actually perform the work. The Master can use the Module to know how to divide up Jobs into the work units that are sent to the Engines.

Work can be absolutely anything. Smedge is designed to be able to distribute anything that you can conceive of, as long as you can figure out how to break it into pieces. One common task for Smedge is to manage the rendering of a sequence of frames of animation. Smedge has several higher levels of built-in functionality to accommodate this sort of work, and can manage third-party applications that use a command line interface.

But you could do more. Companies that write rendering software could write a Smedge Module that actually performs the rendering directly. Companies that write simulation software can write a Smedge Module to distribute their simulation calculations.

Modules are not limited to just Job type implementation. A Module is simply a dynamically loaded extension, and can interact with any aspect of the Smedge system. Modules can include Messages, used for inter-process communication, Events and Event Handlers, used for triggering processing inside any Smedge component, and just about anything else you can think of. See API for more information.

Products

Product is the term for a Job type that you can create work for. Most Products are defined in the Modules, and a single Module may implement one or more Products. For example, the Lightwave.sx Module implements the Lightwave Product, the Maya.sx Module implements the Maya and the MentalRay for Maya Products, and the ProcessSequence.sx Module implements all Products defined in the Virtual Module files.

Pools

Pools allow you to create groups of Engines, and then use these groupings to define which Engines can work on which Jobs, and in what order. The Pool itself, is identified with a unique ID number, and a name string, which can contain any text you want. Engines maintain their own ordered list of the Pools they belong to. Each Engine can be a member of many Pools, but each Job can only be assigned to a single Pool.

Configuring the Pools works slightly different than you may be used to from other types of distribution management software. Instead of adding the Engine to the Pool, you add the Pool to the Engine. The reason for this is that each Engine prioritizes the Pools it works on, allowing you much more control over the prioritization of work from Pools. For example, if you have 2 Pools, "TV" and "Film," you can set half of your machines to give priority to "TV" Jobs and half to give priority to "Film" Jobs.

There are two pools that every machine is always a member of. First, each machine is also its own pool with only that machine as a member. This is always the highest priority pool, so work assigned directly to an individual machine will always go before any other work on that machine. You cannot disable this pool.

The other pool is the default "Whole System" pool. This pool includes every machine connected to the system. It is always the lowest priority pool , so any work assigned to another pool of which the Engine is a member will go before work assigned to the "Whole System" on that Engine.

You can disable the prioritization of pools, and you can disable the use of the "Whole System" pool altogether. See Controlling the Master Dispatching Options for more information. It is also now possible to assign a Job to multiple Pools. See Prioritizing Your Jobs for more information.

Advanced Concepts

Administration

Smedge has a special user role, called the "Administrator". This is the kind of user that is in charge of the system, as opposed to your general users who are primarily interested in queuing their Jobs. By default, there are no restrictions on what users can do, and any user can enter "Administrator Mode" in order to perform advanced system configuration or maintenance. You can password protect the Administrator Mode (see Changing the Administrator Password) and you can restrict the actions that users can perform.

Administration is maintained centrally by the Master. The system is a cooperative one, where component applications have to agree to play by the rules. This is only an issue if you use the API to create your own components, or download third-party components. Be sure that their use does not create security problems before you deploy them.

Options files

All options and settings in Smedge are stored in text based files. This means that options files are totally cross-platform and can be easily human read or modified if needed. No options or settings are stored in the Windows Registry or other proprietary binary systems.

Smedge also includes a system of multiple levels of options file loading for successive overriding of options. Options are first loaded from the Application directory. This gives you broad control over the entire system, and if you are running Smedge from a Shared Installation, a single file can control Smedge components on every machine that runs Smedge.

Next, files can be loaded from the Machine directory. The exact location of this directory is system dependent (on Windows, the default is C:\Documents and Settings\All Users\Application Data\Uberware\Component Name\). This gives machine-wide control over the component, no matter which user happens to be logged in or running the program.

Finally, files can be loaded from the User Directory. Again, the exact location of this directory is system dependent (on Windows, the default is C:\Documents and Settings*User Name*\Application Data\Uberware\Component Name\). Not all components will use all locations. For example, the Master and Engine do not use the User Directory for anything. If the files are not found in a directory, there is no error message, and no changes are made. Settings loaded from later directories will override those set in earlier directories.

As with most things in Smedge, the search locations for these files can be overridden, allowing you to specify a central location for default options that makes sense for your network.

Smedge is intelligent about its options. If you have set some options in a file in the Machine Directory and others in the User Directory, only options changed by the user will be saved back into the User Directory file. This means you can later change the file in the Machine Directory without worrying about whether an application has copied the old settings into the User Directory file, which would override your changes.

Restrictions

SmedgeGui is designed to be able to control any and every aspect of your Smedge network. However, there are times when you may want to limit what users can do. Smedge uses a system of named restrictions to perform this. By default, no restrictions are in place, but you can add them in the Master Options dialog. See Restricting What Users Can Do for more information.

The Master maintains a central list of restrictions in place, so you don't have to configure each client manually. No restrictions are in place when you are in Administrator Mode (see Administration). The restriction system is voluntary, which means that Smedge component applications need to check the restrictions themselves. This means that the restriction system is not a very secure security system, and should not be relied upon for that. If you need security, you will need to use the Smedge API and the security features of your OS and network components.

Distribution

Smedge is designed around the philosophy of "distribute anything," and works by having each type of job handle its own distribution of work to an Engine. Part of this philosophy carries over to how an Engine's resources are used. You can choose in Smedge to divide up an Engine based on the number of central processor cores installed, the amount of physical memory installed, or both at the same time. This gives you the flexibility to use your processing power as most suitable for your particular application. To configure this operation, set the Memory Distribution setting in the Master Options dialog.

For sequences being distributed, for example a render of a sequence of frames, Smedge provides some additional options for how the items from the sequence are distributed. Smedge allows for simple forward or reverse iterating through the sequence, but the default distribution mode is to send out a sampling of items from the entire sequence first, then go back and fill in the holes. This "sample mode" distribution allows you to get a better estimate of the total time a job will take to process (for example, if the render time is different at the end than at the beginning of a sequence), and gives you a chance to see any possible problems that may pop up later in a job without having to wait for the job to actually finish all work up to that point. You can set the distribution mode as an advanced Job parameter or as a default for a product.

Events

Events are notifications sent from the Smedge component applications at specific moments. Users can hook into these events in order to do extra processing or work. You can use the Event Comments attached to a Job or an Engine, the Herald Shell application or the API to access these events. You can even create custom Event types and put your Event Handlers into a Module that will hook directly into the operation of any Smedge component application.

API

Smedge is built on an open API, and you can download and use this API to create your own compiled Smedge components. The API is divided into two main sections: RLib, a cross-platform general purpose library, which provides the common functionality that programs require in a manner designed to be efficient, safe, and easily ported to different operating systems, and SmedgeLib, which provides the standard functionality of Smedge components, including the communication systems, the core Job functionality, and standard information needed to operate in the Smedge system. Smedge integrates with the wxWidgets cross-platform GUI library to implement the graphical applications. A third Smedge component library, wxSmedge, handles this integration.

The API is not included in the standard distribution, but is available without cost. Please contact info@uberware.net for more information.

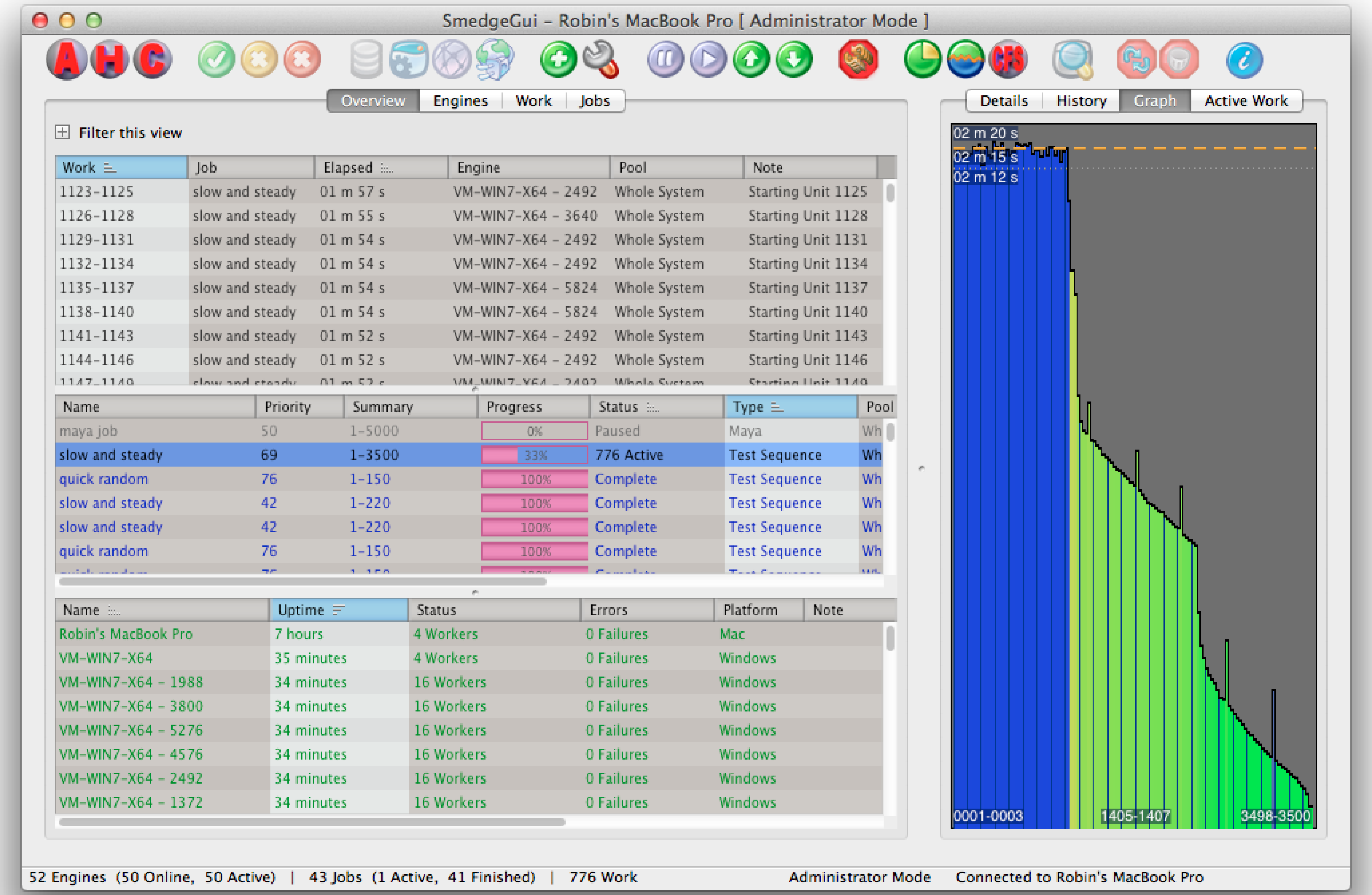
Security

Smedge is not designed as a secure system. There are simple security and restriction mechanisms in place (see Administration and Restrictions), these mechanisms are not very secure, and rely on cooperation on the part of component applications to be effective. You should not use Smedge as a system for securing your network from the outside, and if you need advanced security internally on your network, you may need to use the API combined with the security features of your OS and network components to create this security manually.

Because Smedge provides an easy interface for performing any imaginable task on multiple remote computers, the security of your systems can be compromised. The easiest way to reduce the risk of abuse of the system is to ensure that the SmedgeEngine process is being run by a user that has only the permissions to do what you need it to do. Using an administrator or root user account, can present a serious security risk, and should be avoided if possible.

Please feel free to contact info@uberware.net if you have any questions about security in Smedge.

SmedgeGui (pronounced "Smedge Gooey") is the primary interface for user interaction with the Smedge network. It can be used to submit and control Jobs, configure Engines, configure the Master, and monitor work.



Work	Job	Elapsed	Engine	Pool	Note
1123-1125	slow and steady	01 m 57 s	VM-WIN7-X64 - 2492	Whole System	Starting Unit 1125
1126-1128	slow and steady	01 m 55 s	VM-WIN7-X64 - 3640	Whole System	Starting Unit 1128
1129-1131	slow and steady	01 m 54 s	VM-WIN7-X64 - 2492	Whole System	Starting Unit 1131
1132-1134	slow and steady	01 m 54 s	VM-WIN7-X64 - 2492	Whole System	Starting Unit 1134
1135-1137	slow and steady	01 m 54 s	VM-WIN7-X64 - 5824	Whole System	Starting Unit 1137
1138-1140	slow and steady	01 m 54 s	VM-WIN7-X64 - 5824	Whole System	Starting Unit 1140
1141-1143	slow and steady	01 m 52 s	VM-WIN7-X64 - 2492	Whole System	Starting Unit 1143
1144-1146	slow and steady	01 m 52 s	VM-WIN7-X64 - 2492	Whole System	Starting Unit 1146
1147-1149	slow and steady	01 m 52 s	VM-WIN7-X64 - 2492	Whole System	Starting Unit 1149

Name	Priority	Summary	Progress	Status	Type	Pool
maya job	50	1-5000	0%	Paused	Maya	Wh
slow and steady	69	1-3500	33%	776 Active	Test Sequence	Wh
quick random	76	1-150	100%	Complete	Test Sequence	Wh
slow and steady	42	1-220	100%	Complete	Test Sequence	Wh
slow and steady	42	1-220	100%	Complete	Test Sequence	Wh
quick random	76	1-150	100%	Complete	Test Sequence	Wh
quick random	76	1-150	100%	Complete	Test Sequence	Wh

Name	Uptime	Status	Errors	Platform	Note
Robin's MacBook Pro	7 hours	4 Workers	0 Failures	Mac	
VM-WIN7-X64	35 minutes	4 Workers	0 Failures	Windows	
VM-WIN7-X64 - 1988	34 minutes	16 Workers	0 Failures	Windows	
VM-WIN7-X64 - 3800	34 minutes	16 Workers	0 Failures	Windows	
VM-WIN7-X64 - 5276	34 minutes	16 Workers	0 Failures	Windows	
VM-WIN7-X64 - 4576	34 minutes	16 Workers	0 Failures	Windows	
VM-WIN7-X64 - 2492	34 minutes	16 Workers	0 Failures	Windows	
VM-WIN7-X64 - 1372	34 minutes	16 Workers	0 Failures	Windows	

52 Engines (50 Online, 50 Active) | 43 Jobs (1 Active, 41 Finished) | 776 Work   Administrator Mode   Connected to Robin's MacBook Pro

Navigating the Interface

Views

The Smedge GUI is designed to present as much data as possible in a clean and organized fashion. The primary elements of the interface are the lists of data. These are grouped into tabs in the main window called “Views”. Each view is a tab with one or more lists of data, showing work, jobs, or engines. The lists are limited in what they display by a filter.

Every View can have a filter applied to limit the data displayed in its lists. Click the **Filter this view** text at the top of the view to open the filter controls. This user filter is applied on top of the view filter, which is configured when you create the view using the Customize Views command. The **Job Name** and **Creator** fields will filter the Job and Work lists, and the **Pool** field will filter all 3 lists. You can see an example of a filter in use on the **My Jobs** tab, which simply includes the **Creator** filter set by default to the current user name. The filters are not case sensitive, and will automatically work like wildcard searches, so a Job Name filter of “mov” will match Jobs named “My Movie” or “moving ball”.

You can choose 3 modes of operation for the string filters. Right click on the text field to access the option to choose how it works. “Exact Match” means that all characters, including space characters, are used to match the string. “Any Term” will divide the string using the space characters and look for any term in that string in the job name. For example, if you type “02 03” (with a space between the numbers), it will match “shot 02 version 01” “shot 03 version 05” and “shot 02 version 03”. In “All Terms” mode, the string will still be divided into terms using the space characters, but in order to match, all terms must exist in the matched name. In the example above “02 03” will only match the job “shot 02 version 03”.

You can sort the lists by any column by clicking on the header for that column. When you click on a column header, an icon shows up in the column header to let you know which column is the sort column, and if the sort is low-to-high or high-to-low. Clicking a second time on the column header toggles the sort order between these two states.

By right-clicking the header name of a column, you can bring up a context menu with controls for manipulating the displayed columns. The first item in this menu is disabled, and denotes the name of the column the menu controls/belongs to. Beneath that is a section with the organization commands, Move far left, move left, move right, and move far right, which moves the selected column in the fashion described. There is also a remove command to get rid of the current selected column.

The next section is again marked by a disabled item label, “Available Columns,” and indicates the start of column display toggling views. Here, each item is specific to the type of view list you are working on (Job, Engine, or Work), and are checked if its displayed, or not if it is removed. You can click an item to toggle its status causing new columns to appear on the far right of the view list. For Job List and Work List views, is a submenu item Custom Columns exist, which brings up a small GUI window to input your own column display. Here, you can specify the custom column's name, and the data it displays. This system uses the Job parameter substitution system. See the Administrator Manual for more information about the parameter substitution system and a complete list of all parameters that are available for every Product in Smedge.

Each view displays one, two or three lists of system data, about the Engines, Jobs, and Work currently in the system. Each list has its own context menu, offering a subset of commands that apply to those objects. You can access the context menu by right-clicking on one or more items in the list. It is important to remember that most commands can be applied to more than object at a time, if you select multiple objects. This can make it very easy to pause or resume a bunch of jobs at once, for example, by selecting all of the jobs you want to pause, then right clicking and selecting the **Pause Job** command. The commands in the main menu bar and the commands in the context menu work identically.

Engine List

This list shows basic information about Engines. It shows the engine's name, how long the machine has been running (technically how long it has been running the SmedgeEngine process, not the uptime of the machine itself), the status of that Engine, which operating system that machine is running, and any notations that have been added to the Engine's information. You can select one or more Engines from this list to see more detailed information about them, or to modify them.

Job List

This list shows basic information about Jobs. It shows the Job's name, a summary of the how the Job is being distributed, the priority of that Job in its Pool, the current status of the Job, which type of Job it is (which Product is being used), the Pool that the Job is assigned to, the name of the Job's creator, the time the Job was created, and any notations that have been added to the Job's information. You can select one or more Jobs to perform operations on, or to see more information about them.

Work List

This list shows basic information about currently running work. It shows the work element, the name of the parent Job, the elapsed time the work has been running, the Engine the work is running on, the pool, and the most recent notation attached to the work unit. You can select one or more Work Units to perform operations on, or to see more information about them.

Default Views

Overview presents a complete overview of the entire system, in a style based on the Smedge 2.x GUI design. Three lists are shown: the top is the list of currently active work, the middle is the entire list of Jobs, and the bottom is the list of all known Engines.

The second tab, **My Jobs**, shows the list of Jobs and Work for Jobs that you have submitted. This is useful in larger environments to help artists isolate their own Jobs in the network. You can also use the tabs to select a view that maximizes just one of the three lists, using the **Engines**, **Jobs**, and **Work** tabs to show just the appropriate list. No matter which view you are using, the lists work identically and show identical information. Besides using the tabs to select your list view, you can also use the commands in the View menu, or their keyboard shortcuts.

Custom Views

The customize tab is an optional view, restricted by the “Customize Views” restriction under administrator mode, which can be shown or hidden using the Customize Views command in the View menu. This view contains buttons to add new views, remove views, and restore the views to the default settings.

Selecting the “Add a new view” button adds a new view to the GUI view tabs, the dropdown menu to change views, and to the Custom Views tab itself, at the bottom of the list. Selecting the “Reset to Default” button removes all custom views and restores all default views that may have been reordered or deleted.

Beneath these buttons are a list of filter controls belonging to each view. Here, under the “View Name” field, you can change the name of a selected view. Under the 3 “List” checkbox items, you can toggle whether or not that view shows information for engines, jobs, and work. You can filter out items by included Job or Creator names, by pools, and/or by job status and products. As with the main list filters, you can choose how the string searching is done using the options “Exact Match” “Any Term” or “All Terms”.

Note, that these filter options are separate from the view's filter options, and get applied to the items before the view's filter gets applied on top of it. Beneath the filter controls are buttons to move the view up or down, which change the order the view tabs are located at along with their respective menu items and custom views list order. Lastly, the “Remove this View” button to the bottom right of each listed view in the Custom view tabs deletes and removes the given view from the GUI.

View Presets

In the View Manager panel, you can save a the current collection of views as a Preset. Presets are saved to the Master, and stored in the Master options, so once you have created a preset, that preset is available to all connected clients automatically. You can select a preset to use for your local GUI display from the drop-down list of presets. You can also specify a preset to be applied automatically to every node in the Administrator Options dialog box.

Info Pane

The second set of tabs towards the right of the main window is the Info Pane, which is designed to show more detailed information about the items you select in one of the lists. The Info Pane is an optional view, which can be shown or hidden using the Info Pane command in the View menu, or by using the shortcut key `Alt+0` (that’s the number zero, not the letter O).

The Info Pane includes extra details about selected objects, and related information that may be useful. Exactly what is shown depends on what is selected. If you select work, you can see a pane with more details about the work element itself, and about the job, as well as the Job History and the Job Graph views. If you select a job, you can see more details about the job, the Job History and Job Graph views for that job, and a list of the active work for that job. If you select an engine, you can see more details about the engine, the live status of that engine (if it is running), a list of the active work on that engine, and a list of the failures that have happened on that engine

Limiting the Visible Products

Smedge can control a very large number of rendering products, and with customizations that number can get very large with multiple versions that may be in use at your facility. You can reduce the clutter in the interface by hiding products from being displayed in the interface, making easier to find just the products that you are using.

Note that the Products are not removed or disabled from your network, and will be visible on other machines. This option only hides them from your display, and only when not in "Administrator Mode." To completely remove a Product from your network, you need to remove the Module that provides the Product from every machine on your network. You can use the Module Manager for this.

For more information, see the SmedgeGui Options reference.

Engine Mode

By default, when you start the GUI on a machine, three Smedge component processes are actually started: the Master, so this machine can control the system (or act as a backup if the Master should go offline); the Engine, so this machine can do work, and the GUI process itself. For a small network of workstations, this is fine, but as you grow and want to add more machines, the additional overhead of the full GUI and Master can start to degrade performance of the render machines, and cause undue load on the Master machine.

Smedge is highly configurable at the component level, with individual executables capable of being controlled as system services. However, if you are growing faster than your IT staff can keep up, Smedge will take it upon itself to try to make things better for you. If the GUI is not used for a period of time, it will automatically disengage itself from the Master to reduce its load. It will also automatically stop a redundant backup master running on the machine, if there is one, further improving machine and network performance.

You can control the time delay for entering Engine mode at a system level (in the Administrator Dialog), and at a local machine level (in the SmedgeGui Options). You can disable the automatic engine mode as well (a time delay of 0 will disable it). You can also manually enter Engine mode with a command in the System menu. If the GUI is closed while in Engine Mode, it will start in Engine mode the next time you start it.

Additionally, Smedge tries to be a little clever about entering Engine mode automatically. If your machine name with the word *render*, *blade*, *node*, *smedge*, or *farm* and the name includes a number, or if the machine name is exclusively numbers, then Smedge makes the assumption that this machine is supposed to be used for rendering only, and starts in Engine Mode the first time you start it.

You can return to normal operation at any time using the Connect command in the System menu. Also see the Administrator Manual for more information about components and direct operation of the SmedgeMaster and SmedgeEngine components as normal processes or as system services.

Job Control

The Submit Job Window

The Submit Job window is the primary interface for adding and editing Jobs in Smedge. It allows you to configure every possible setting for any type of Job, from a single, simple to use interface. It also gives you options for submitting the Job, submitting several similar Jobs, loading and saving Jobs to a file, repeating jobs, and more. For full information, see the Submit Job window reference.

Adding a new Job

There are many ways commonly used to add a Job from the GUI. In the Job menu there are three commands that are commonly used, New Job…, Load Job…, and Repeat Job….

New Job will open a new Submit Job window. This window is the standard interface for creating and editing Jobs with the GUI. Every detail about a Job can be configured in this window. The general categories of data are divided into separate tabs to keep things simple. By default, the editor will show the last Job you submitted, but you can also load and save Job data or use the customizable default system. See the Submit Job window reference for more information.

Load Job will allow you to browse for a previously saved Job file, and will immediately load the Job(s) described in the file into the system. You will not have a chance to change the Jobs before they are added if you use the Load Job menu command. Any dependencies (see Making a Job dependent on another Job) in the Job file will be maintained, even if the IDs of the jobs are changed as the file is loaded. This means that you can build up a complex hierarchy of dependent jobs in the file (or save such a hierarchy into a file) then just load the file over and over every time you want to submit the whole hierarchy of jobs. See Saving Jobs to a File for more information.

If you want to load a previously saved Job file into the Submit Job dialog box, you can use the Load button on the dialog itself. If the Job file contains more than one job, only the first (the one with the lowest original ID value alphabetically, not the first in the file) will be loaded and all other Jobs in the file will be ignored.

Repeat Job will bring up the Repeat Job window. This window allows you to select one of a number of previously submitted Jobs to repeat. When you double click a job in this list, Smedge GUI will open a new Submit Job window with a copy of the settings from the previously submitted Job. It does not submit the Job directly.

Copying and resubmitting Jobs

Another way to add Jobs is by copying existing Jobs in the system. If you select one or more Jobs in the Job list, the Copy Job menu command becomes available. This command will open the Submit Job window and will copy all of the settings from the selected Job. If you have multiple Jobs selected, multiple windows will be opened, one for each Job, each with a copy of one of the selected Jobs. The windows will submit brand new jobs; they will not change the parameters of the existing Jobs.

Another command that becomes available when you select Jobs is the Resubmit Job command. This will immediately resubmit the selected Job(s) as brand new Jobs. The selected Jobs will not change at all. Instead, Smedge GUI creates identical copies of the Jobs and submits them as new Jobs.

The Submit Job window itself offers a few different ways to add Jobs to the system. For more information, see the Submit Job window reference.

Finally, you can add Jobs to the system independent of the Smedge GUI program with any Shell application designed to do so. Another Shell application that ships with Smedge is the Submit program, which is a command line based tool for adding Jobs to the system. While some old-school users may enjoy submitting jobs manually from a command prompt, the real power of Submit is integration with other programs or scripts. For more information, see the Submit Shell reference.

Prioritizing your Jobs

The first way you can prioritize Jobs is to set the **Priority** value for the Job. The valid range for priority is from 0 – 100 and higher priority values will be dispatched first. Priority 0 Jobs will still be dispatched, unless you have configured the Master to consider priority 0 as paused. See the Master Options dialog for more information. To pause a Job use the Pause and Resume commands in the Job menu or check the **Paused** checkbox field on the Submit Job window.

Besides using the priority value, Jobs are also dispatched according to their **Pool**. Each Job can be assigned to only a single pool, but each Engine can have multiple pools that it belongs to, and the pools are arranged in a prioritized list. If Machine 1 is a member of

Pool A and Pool B, with Pool A as a higher priority, and Machine 2 is a member of both pools, but has Pool B as a higher priority, Jobs assigned to Pool A will have priority on Machine 1, and Jobs assigned to Pool B will have priority on Machine 2.

By default, the **Pool** value overrides the actual **Priority** value, though this can be disabled in the Master options. When two jobs have the same **Pool**, then the Priority comes into effect, to prioritize the Jobs. If both the Pool and the Priority are the same, there are two ways Smedge can use to determine how to dispatch Jobs. You can select "First in, First Out", where Jobs that were created earlier get priority over later ones, or you can choose "Round-Robin" style, where Smedge tries to evenly dispatch Jobs with the same priority levels, no matter when they were first created. To change this option, you need to set the Master Dispatch options in the Master Options dialog. You can now assign a Job to multiple pools. This will only work correctly if the Pool Prioritization has been disabled (see above). This way you can use Pools as simple groups of machines and assign jobs directly to those groups. To assign multiple pools, select the additional pools in the Custom Pool tab of the Submit Job dialog, or use the Pool ID with the -IncludeEngines flag when using the Submit command line tool.

Smedge can optionally use priority boosting. This lets you get a few frames rendered from a job that you have just submitted, even if there are other higher priority jobs in the queue. You can set a priority boost globally for all jobs in the Master Options dialog. You can set a global boost on the Distribution tab, or you can set a boost for all jobs of a specific product on the Product Settings tab. You can also set a boost number for a specific Job when you submit it. The job's value will override the Product value, which will override the global value. By default, priority boosting is off. Priority boosting works as if you added 100 to the Job's priority value. All other prioritizing systems work normally, so pools can affect this, and higher priority jobs will get boosted over lower priority jobs.

Getting information about a Job

A lot of basic information about a Job is visible right there in the Job List. However, to get more information, you can click on a Job. If the Info Pane is open, you will get three different tabs of information you can explore to get more detailed information about a particular Job.

The **Details** tab gives you an overview of the data that makes the Job up. The top part shows the basic common Job information, including how far along this Job is. Under that is a complete list of all the user configurable parameters for the Job. These fields are all read only, so you can see the information that makes the Job up, but you cannot accidentally change anything here. If you want to change Job information, you will need to open the Submit Job window for this Job (see below).

The **History** tab shows the Job's history. The top of the History pane shows you some basic statistics about the Job, and is followed by an expandable list of each work-unit that makes the Job up. Normally, only the most recent work unit history information is displayed, but you can expand a work unit to see every history element that makes that work unit up. This can be really useful for diagnosing errors, because Smedge always stores the reason for failures in the work unit history as a note. See Viewing a Job's History for more information.

The **Active Work** tab is a Work List that is filtered to only show work from the selected Job. For more information about what is shown here, see the Work List. Note that, since this is already in the Info Pane, clicking on work units here does not change the Info Pane. That would be confusing. However, other aspects of the Job list, specifically the right button context menu, work identically.

Viewing a Job's History

As noted above, you can view the Job History in one of the Info Panel tabs. You can also double click on the Job or select **View Job History** from the Job menu, and open the Job History in a separate window. The Window and the Info Panel work identically.

Here you can see some of the information that can be viewed in the Job History. You can also see the error messages that Smedge produces when work fails, to help you determine what is going wrong with your Jobs when things are not working properly. Error messages appear in red, making them easy to spot.

You can select individual work units from this list to re-queue. Select the units, and right click to open the context menu. The first command in the list is the "Requeue Selected Work" command, which will create a copy of the Job with only the work units you have selected.

Most work commands can also be accessed from the Job History view by right clicking on the elements. One exception is the viewing of the captured output. By default, the view output command will try to actually load the file from the filename displayed in the history. However, this path is relative to the Engine that actually performed the work. Make sure that the path to the log file indicated with the Log: entry in the work's history is a valid path on the computer you are using.

Modifying an existing Job

Once a Job is in the system, it is still quite configurable. The only thing you cannot change about any Job is the type of Job that it is. Jobs will allow or deny different changes to a job after it has been created, depending on the type of Job. Many Job changes will be passed along to any running work from that job. However, some types of changes cannot be made to a process that has already started (for example, changing the number of threads it will use), so not all change can be forwarded.

The most common changes you will want to make are to change the prioritization of your Job. You can select Increase Priority or Decrease Priority from the Job menu, to bump the priority number of the Job up or down by 1, respectively. You can also change the Pool that a Job is assigned to with the Pool sub-menu. You can pause or resume Jobs with the Pause Job and Resume Job commands also in this menu. Other Job menu commands that modify the existing Job are the Include and Exclude Engines commands, and the Set Note command are self descriptive. You can use the Set Parameter command to change any single parameter for a bunch of selected jobs at one time.

Another way to modify your job is to choose Edit Job from the Job menu. This will open up a Submit Job window with all of the settings from the Job. You can then go through and change any parameter of the Job just like before you actually submitted the Job. As long as the Job has not finished, this will modify the properties of the existing Job. If the Job has finished, this command will actually create a new copy of the Job and will submit it as a new Job.

Repeating a Job

Any Job currently in the system can be resubmitted, by selecting Resubmit Job in the Job Menu. This will make an identical copy of the Job and submit it as a new Job.

You can also access the Repeat Job dialog from the Job menu. This brings up a dialog with the last several Jobs that have been submitted. Double click a Job in this dialog, and you will open up the Submit Job window with the parameters all copied from the last time this Job was submitted. You can change any parameters you wish, and it won't affect any existing Jobs.

The Repeat Job dialog, when accessed in this manner, is a modeless window that will stay open as long as you wish. You can still use the entire interface or any open Submit Job windows event when this dialog is open. However, you can also access the Repeat Job dialog by clicking the Repeat button on the Submit Job window. In this case, the Repeat Job dialog will be modal. Selecting a job will reset the entire contents of the Submit Job dialog to the values of the Job you are requesting to repeat. Again, using the Repeat Job dialog will never affect any existing Jobs that have already been submitted into the system.

Submitting several similar jobs

You can use the **Submit Copy** button on the Submit Job window to submit the Job as it exists in the window and leave the window open. This allows you to then modify parameters of the Job, allowing easy submission of several similar jobs.

You can also use the Search and Replace dialog to modify the parameters in the Submit Job window. To access the Search and Replace dialog, click the **Replace…** button near the lower right corner of the Submit Job window. The text that you search for will be located in any field of the Job, including the Advanced parameters, and replaced with the text you supply. Using the Search and Replace dialog with the **Submit Copy** command allows very quick and easy batch submission of several similar Jobs at one time.

Remember that you can save one or more jobs into a single Job File (**.sj** file), including the dependencies (see Making a Job dependent on another Job) between the Jobs. You can use this feature to save out a set of jobs that you will run over and over. See Adding a Job and Saving Jobs to a File for more information.

Specifying Engines to work on a Job

The most obvious way to specify which Engines will work on a Job is by setting the Job's Pool. Only Engines that have this Pool as a member will be allowed to work on the Job. Remember that Pools also prioritize Jobs on the Engines.

Sometimes, though, you want to specifically include or exclude Engines from working on a Job regardless of the Pool setting. In this case, you can use the Include Engines and Exclude Engines dialogs. The Include Engines dialog allows you to specify additional Engines that may be outside the Job's Pool that are allowed to work on a Job. The Exclude Engines dialog allows you to specify Engines that should not work on this Job, even if they are in the Pool and available for work.

You can access these dialogs in two ways. From the Submit Job window, you can press the **Include Engines** or **Exclude Engines** buttons along the right side of the Window. Or, if you select one or more Jobs from the Job List, you can then choose the **Include Engines…** or **Exclude Engines…** commands from the Job menu.

Saving Jobs to a file

You can save jobs from the Job List by selecting them and choosing Save Job from the Job menu. All of the Jobs that you have selected will be saved into a single file. Any dependencies (see Making a Job dependent on another Job) between the Jobs will also be saved, and will be restored correctly every time you load this file, even if you load it multiple times. The ID's will be adjusted to always load the set and its dependencies correctly without affecting any other job currently in the system.

Job Files (.sj files) are simply INI format text files. Each section contains the information about the Job. Any job parameters that are missing will have whatever the default data for the Job is, and any parameters that don't make sense for the type will be ignored. Note that the ID is the section heading, but may be changed when the Job File is loaded. Dependent Jobs should put the same ID as the section heading for the Job upon which it depends. The type value must be the ID of the type – the loading process will not look up the type names or shortcuts.

Making a Job dependent on another Job

In the Submit Job window, you can select any Job in the system in the **Wait For** field. When a Job is set to wait for another Job, then no work from the Job will be dispatched until the "wait for" Job either has no more work to dispatch, or the Job has been deleted. This is useful when your dependent Job absolutely depends on the output of another Job, and you want to be absolutely sure that all work from the "wait for" job is finished. For example, a Job that composites elements into a final animation can be set to run only after a Job that renders the elements that are used in the composite.

If **Whole Job** is checked, then your Job will have to wait for the other Job to have completely finished, as above. If unchecked, work from your Job will go as soon as the equivalent work in the waited for job completes, even if the waited for job has more work to do.

For example, if you have to export from a source file on a single machine, but can render the exported objects in parallel, make the render job wait for the export job without the whole job so that as soon as a frame is exported, its render can start, even as the export job still has more frames to export. The export job starts running first, and as soon as the first frame is finished, the render job can send out that frame.

The partial job waiting system will only work if the two Job's distributors are compatible. For example, if both use ranges, or if both are Generic Script jobs. If the distributors are not compatible for partial job waiting, it will fall back to whole job waiting automatically.

Hooking into Work events

Job Event commands allow you to hook arbitrary command line actions into any of the Work related events on the Engines that do the work. For example, you could hook into the First work event to make sure that a network asset is available, or hook into the work finished event to post-process the result or delete a temporary file. Use the

The command line that is executed will have any of the variables in it substituted with data from the Job or Work. You should use the standard Smedge $(Name) parameter syntax to access this data. See the Administrator Manual for more information on the syntax and available parameters. You can assign these events as Job parameters, using the Event Commands tab of the Submit Job window to add these commands to your Job, or you can set global event commands for an Engine using the Event Commands tab of the Configure Engine window (available only in Administrator Mode). You can also add event command actions using the Herald, but these commands will always be executed asynchronously with the work, and will not affect the success status of any work.

There are two types of events that Smedge allows you to handle. Job related events are performed by the Master, and Work related events are performed by the Engine that is doing the work. Event commands are called either **Synchronously** or **Asynchronously**. Synchronous events will pause any further processing until the command completes, and a failure result from the command (i.e., if the executed process returns a non-zero result code) will cause the work to be marked as "Unsuccessful" and requeued, just like any other type of work failure. Asynchronous events will not delay Smedge processing and the result will be ignored.

See the following table for a complete list of the work events, and how they are run by either the Master or the Engine during the Job and Work life-cycles:

Job Events called by the Master

Event	*When called*	*Executes*
JobFirstStartedEvt	Sent the first time any work from this Job is assigned. Sent exactly once	**Synchronous**

Event	*When called*	*Executes*
	per Job on the Master. Note: if you restart the system, this event will be sent again.	
JobAssignWorkEvt	Sent when the Master has determined the next worker for an Engine, but before that work has been sent to the Engine. An unsuccessful result from this command will cause the Master to not send this work unit and try to pick another for the Engine.	**Synchronous**
JobFinishedEvt	Sent when all work from the Job has either finished or been permanently canceled.	**Asynchronous**
DeleteJobEvt	Sent when the Master is about to delete a Job from the system.	**Asynchronous**

Work Events called by the Engine

Event	*When called*	*Executes*
WorkAssignedEvt	Sent when the Master has assigned work to the Engine, but before the Engine has accepted the assignment.	**Synchronous**
FirstWorkEvt	Sent the first time work from this Job is started on this Engine. Sent exactly once per Job on each Engine that performs work from the Job	**Synchronous**
WorkStartedEvt	Sent when the Engine has accepted a work assignment but before the work has actually begun.	**Synchronous**
WorkParameterChangedEvt	Sent when the Engine has detected a change in a parameter for the active work.	**Asynchronous**
WorkPostExecuteEvt	Sent when the Work has finished executing but before the result is sent back to the Master	**Synchronous**
WorkPostExecuteSuccessfulEvt	Sent immediately after the WorkPostExecuteEvt only if the Work is about to finish successfully.	**Synchronous**
WorkPostExecuteUnsuccessfulEvt	Sent immediately after the WorkPostExecuteEvt only if the Work is about to finish unsuccessfully.	**Synchronous**
WorkFinishedEvt	Sent after the Work has finished and notification has been sent to the Master	**Asynchronous**
WorkFinishedSuccessfulEvt	Sent immediately after the WorkFinishedEvt only if the Work finished successfully	**Asynchronous**
WorkFinishedUnsuccessfulEvt	Sent immediately after the WorkFinishedEvt only if the Work finished unsuccessfully	**Asynchronous**

Event	*When called*	*Executes*
EngineCleanupEvt	Sent when all work from the Job has either finished or been permanently canceled.	**Asynchronous**

Custom Job commands

Different Job types can have different custom commands that are associated with the Product. For example, every Product designed to render a sequence of images includes several commands. You can browse the image sequences, or you can use the CheckFileSequences component application to verify that all of the files rendered correctly. You can also open a Shell explorer window that brings you to the scene file or the rendered image files.

Each Product could have its own custom commands. For more information about custom Job commands, see the Custom Job Commands chapter. You can also use the Status bar to see a short online help about the command. For custom Product types, contact the Module author or your Smedge Administrator.

Examining Failures

When work fails, the system will maintain a failure count in order to keep a failing job from taking up too many resources. Once you have fixed whatever was causing the problem, you may want to reset the failure counts so that work will pick up again from the Job that had previously failed.

You can see a list of the failures that an Engine has had by selecting that Engine in the list and looking in the Info Pane. The "Failures" tab will list all jobs that the Engine has failed on, grouped by Product. You can see how many failures each job had, and the list gives you easy access via a context menu to find the Job history or reset the failure counts for a Job or for an Engine. You can also copy and paste from the display, search it, and save the contents to a file.

You can reset the Job failure counts either by Job or by Engine. To reset failures by Job, select one or more Jobs in the Job List and select the "Reset Job Failure Counts" command from the Job menu (or from the context menu in the Job List). All error counts associated with work from these Jobs will be reset, and work will be dispatched normally. To reset failures by Engine, see Resetting failure counts in the Engine Control section.

You can adjust the count limits before work is no longer set by adjusting the Master's settings. See Controlling the Master's Dispatching Options.

Setting the “Creator”

By default, Jobs submitted from the SmedgeGui Shell will have the local machine name as the Job’s creator. You can override the string that is put in the “Creator” field in the SmedgeGui Product Options dialog. Access this dialog by selecting “SmedgeGui options” from the System menu.

Resubmitting some of the Work from a Job

You can resubmit individual work units from a Job using the Job History context menu. Select the work units you want to resubmit, and then choose “Requeue Selected Work” command from the context menu. For more information, see Viewing a Job’s History and the Job History window reference.

You can also use the “Check File Sequences” command custom Job command if it is available for the Job type. This command only works if the Job type supports it, and if the Job was able to detect any output file sequences.

Common Custom Job Parameters

Every Product can have its own attached custom parameters. Because Products are implemented in dynamically loaded Modules, and because users can create custom parameters using the Virtual Module system or the API, this manual cannot describe every possible parameter you may run into in your use of Smedge.

The sections below describe three basic Job implementations that are included in the core of Smedge. All of the Products currently included in the Smedge distribution will include the functionality of these three classes.

Process Jobs

Process Jobs allow Smedge to control a third party application by spawning and monitoring another process on the machine. It adds attributes and commands related to this functionality.

Setting Process Execution options

There are several parameters that are used to control how the process is actually launched. In any of these parameters you can use the $(Name) syntax to allow Smedge to use the variable substitution system to substitute the value from another Job parameter. The substitution is performed when the attribute is actually used, so it will use the current latest information in the Job when the value is actually used.

Executable	Allows you to specify the full path and filename of the executable that will be launched.
Start Directory	Allows you to specify the directory that will be set as the current directory when the spawned application starts up.
Log Path	This will specify where the Smedge captured output log files will be saved. If you do not specify a directory here, the files will be saved in the Engine's log folder.
Run Process Visibly	Allows you to specify if the spawned process is launched visibly or hidden.
Process Priority	Allows you to specify the relative priority that the child process will be spawned with. This is currently only available as an Engine Option, and cannot be overridden as a Job parameter.
Limit Memory Usage	Allows you to specify if the spawned process should have its memory usage capped by the operating system to the amount of RAM allocated to the job based on the memory distribution system. If the memory distribution system is not enabled, or if the Job's memory usage request is either for "all" or "none" of the memory, then no limit will be placed on the process. Note that this hard memory limit can cause unusual behavior from the work process itself should it actually hit that limit, and may produce unexpected results.

Viewing the output from the spawned process

When a Process Job is actually executing on the Engine, it will serve the output from the spawned process on a TCP port on the machine. In order to allow clients to know which port to connect to, the Process Job will send the output address to the Master, and make it known to any clients that care. Clients can then establish their own connection directly with the output server.

The address data is stored internally in the Job information, but is not publicly available through the SmedgeGui Shell.

Making sure the process has resources available

On Windows, Process Job implements a system that can allow you to ensure that network resources are available before the work is started. Use the **Resources** attribute to supply the list of resources you want to ensure are available. Smedge Render Jobs will automatically detect if your scene file is on a mapped network drive, and if so, it will ensure that the drive letter has been remapped before starting work. However, if your scene file references other files that are not on that same drive letter, the drive may not be available, and you may see file not found errors from your workers.

Resources are listed in a semicolon delimited list of mappings to validate. The format is:

Drive-letter`: = \\`*Server*`\`*SharedVolume* [`;` *Drive-letter*`: = \\`*Server*`\`*SharedVolume*...]

For example, if you have drive V mapped to the shared volume "Vault" on the File Server "Big_Fun", and W mapped to the shared volume "Work" on the File Server "Server", you would fill in the resources like this:

```
V: = \\Big_Fun\Vault ; W: = \\Server\Work
```

You can also specify a user name and password in the **User Name** and **Password** parameters that will be used to connect to the file server. If you leave these blank, the system will attempt to use the permissions of the account that SmedgeEngine is running under. When you run the Engine normally, this is the user currently logged into the console. For Services, you can specify this user account using the Service Control Panel.

Smedge allows you to configure different mapped drive resources as an option for each Product, and each Job. However, you can use the Mapped Drive Manager to quickly configure your required mapped drives for all Products on all Engines in once place. The Mapped Drive Manager is only available in Administrator Mode, and is accessed from the Engine menu. See the Mapped Drive Manager section in the reference chapter for more information.

Error detection

Process jobs can do basic error detection from the entire life of the spawned child process. These tests include monitoring the output from the process for error messages, checking the result code of the process, and watching for hung processes that are sitting idle instead of doing the work they are supposed to be doing.

You have a lot of control over error detection in Smedge, in order to get Jobs through that may be reporting errors:

Detect Errors in Output turns on or off the checking for error messages in the process output stream. The default is set in the Engine Options and can be overridden on a Job basis in the Advanced Info tab.

Error Start Strings defines the text that indicates an error message from the process. If a line starts with this text, and error detection is enabled (see above), then this line will trigger an error, causing the work to abort and be requeued. The default is set in the Engine Options and can be overridden on a Job basis in the Advanced Info tab. You can also set if the error start strings must be at the beginning of a line of output, or if they can be anywhere in the output line.

Error Ignore Strings allows you to define indicators in an error message that tell Smedge to ignore a line of output detected as an error. If error detection is enabled and a line contains an error indicator (see above), then if it also contains one of these strings, the error will be ignored by Smedge, and work will be allowed to finish normally. The default is set in the Engine Options and can be overridden on a Job basis in the Advanced Info tab.

Check Process Exit Code allows you to tell Smedge whether or not it should care about the result code from the work process when it finishes. Normally, most processes return a code of 0 to indicate success and a non-zero code to indicate an error. Sometimes, however, work may actually have succeeded and the process is returning a non-zero code for some other reason. Use this to disable this error check, and ignore a non-zero result as an error. The default is set in the Engine Options and can be overridden on a Job basis in the Advanced Info tab.

Minimum Successful Time allows you to specify an absolute minimum amount of time for a work unit to be considered successful. If set, and the process starts and finishes in less than this amount of time, then it is considered to have failed, and the work is requeued. The default time is set in the Engine Options dialog and can be overridden on a Job basis in the Avanced Info tab.

Idle Time Limit allows you to specify an absolute maximum amount of time that a process can sit idle, using no CPU time. This generally means the process is hung or stuck in some wait state from which it will never recover, so it is terminated and the work is requeued. The default time is set in the Engine Options dialog, and can be overridden on a Job basis in the Advanced Info tab.

Sequence Distributor

Jobs that use the Sequence Distributor allocate themselves by breaking up a sequence of numbers into small groups to be worked on. The idea of rendering a sequence of frames uses this concept, but you can use a Sequence Job to distribute anything that can be thought of as a sequence of numbers.

By default, sequences are distributed using the “Sample” algorithm. The first units distributed from a job correspond to the beginning, end, middle, and six other points evenly spaced in the range. After that, frames are distributed from the job filling in the rest of the range from the beginning. You can also submit Jobs in a normal front to back mode or in reverse order. Use the **Distribute Mode** parameter to determine how the sequence is distributed:

Forward	Items are distributed in ascending order
Reverse	Items are distributed in descending order
Sample	Items are distributed using the sampling algorithm

The default is set in the Product Options tab of the Configure Master dialog, and can be overridden on a Job basis in the Advanced Info tab.

Generating a sequence of items

Sequence Job uses the **Range** parameter to define the sequence that is to be worked on. The range can be specified using numbers, the comma character to signify "and", and the minus character to signify "through". Any whitespace will be removed. For example all of the following are valid ranges:

```
1-100

1, 2, 3, 4, 5

30-40,50-60,77

-10 - 10
```

Using this syntax it is possible to generate a range that includes duplicates (for example: 1-100,50-200). Smedge will detect and eliminate any duplicates when it is distributing the work.

Breaking the sequence up into chunks

The **Packet Size** parameter defines how the Sequence Job will break up the range into work. The packet size is specified as an integer value of the number of elements from the range that will be sent at one time. Using **Range** and **Packet Size** together you can determine how many work units will be sent and in what order.

The smaller your packet size, the better your system granularity is, to improve use of your available Engines. However, if the load time for your work is high, you may want to increase your packet size to decrease time spent loading.

If you set your packet size to a value greater than or equal to the number of elements in the range, then the entire Job will be sent as a single work unit to a single Engine.

Padding work names

By default, the name generated for the work units is the sub-range of the work, and the numbers are padded with zeros to four digits long. You can override this ability if you wish by creating a Master option for the Product. You can set Product options for the Master using the **Configure Master…** command from the System menu. You must be in Administrator Mode first.

This will open the Master Options dialog. On the Job Type Settings tab, you can see and modify the Master's options for every currently installed Product.

Render Jobs

Render Jobs allow the control of the rendering of a sequence of images. For example, rending a 3D animation from Maya, or a composite from Shake. This class includes several attributes and commands that aid in this process, and allow a general form of control for just about any sequence rendering system.

Basics

Render Jobs use the concept of a **Scene** file. This is a file that contains the data that you wish to render. Different products may have different names for the scene file, such as “project” or “workspace”. No matter the name, this is the file that you would send to render. For complete details about how the Scene parameter works for a specific Product, see the Administrator Manual.

The **Range** parameter supplied by Sequence Job, provides the frame range of the animation you want to render, and the **Packet Size** parameter will control how many frames from the Job are sent to each engine in a single Work unit.

Error Checking

Render Jobs provide common error detection systems for rendering sequences of images. Besides the basic process error detection provided by Process Job, Render Jobs can also check the rendered image files themselves, to make sure that they exist and exceed a required minimum size. It can also check the render process output stream for error messages that can indicate failure.

Image file checking uses the **Image Filename Format** attribute and the **Minimum Image Size** Engine option.. The **Image Filename Format** holds a formatting string that is used to determine the output image filenames. *This is not used by the actual rendering product to tell it where to put the images or what to name them.* Telling the Product where to put files is dependent upon each individual product, and is usually handled in a different place. Instead, this parameter is used only by Smedge only for doing the image detection test when the work has finished. Normally, you do not need to supply a value for this parameter, as Smedge will fill it in automatically. It is only there to override if Smedge cannot fill it in automatically or if Smedge gets the value confused for some reason. The **Image Filename Format** is available as an Advanced Job parameter. To modify it, open the Submit Job window, and click the Advanced button to open the Advanced Parameters dialog.

The **Minimum Image Size** option allows you to specify a minimum allowed file size. This is useful for detecting invalid image files that may be corrupt or incomplete. The value is a size in bytes. You can set the size only as an option for your Engines. Select one or more Engines from the Engine List and select “Configure Modules…” from the Engine menu or the context menu. In the Engine Product Options dialog, select the Product you wish to set the image size for, and scroll down until you see the **Minimum Image Size** field, then set the appropriate value. Remember to click the button with the red triangles in order to transfer the new value to the other selected Engines.

There are three additional parameters that you can use to configure the error detection. **Auto-detect Image Formats** turns on or off the automatic detection of the **Image Filename Format** parameter. **Check Rendered Files** turns on or off the actual frame file checking. These settings are "Bool Override" type options, which means that there is an on or off setting that is an Engine option (available through the Engine Product Options dialog), and a three-state parameter available as an Advanced Job parameter (available through the Advanced Parameters dialog). The three-state parameter allows you to specify an override value (on or off) or allow the Job to just use the Engine's default from the options. The default for all tests is to be on.

Checking and Viewing frame file sequences

Render Jobs deal with the idea of an output sequence of image files, and includes attributes and commands that enable you to check and view these sequences with your favorite frame viewer. This is handled through options for the Shell, which SmedgeGui passes to the Job to perform the work. To access these settings, select "SmedgeGui Options…" from the System menu.

The **View Frame command line** is the command line that will be executed when you use the "View Frame" command. The **View Sequence command line** is the command line that will be executed with the "View Sequence" command. See Checking and Viewing Rendered Image Files for more information about using these commands.

These command lines will have standard Smedge variable substitution performed before being executed. Any variables in the standard $(ParameterName) syntax will be substituted with the value from the Job or Work you are calling the command for. The special parameter name **SequenceName** will be created specially using the values of some of the other options in order to create a sequence name string that is correct for the sequence viewer you want to use.

The **Sequence Format Specifier** is a string that is put where the frame number goes. If you need to have a repeating character for the frame digit padding, you can use the **Repeat Format Specifier** to tell SmedgeGui to repeat the character for each digit of padding.

Finally, you can modify the command that is executed when you use the "Check File Sequences" command to start the Smedge Check File Sequences shell to explore the rendered image files. Normally, you should not mess with this, but it's there just in case.

Common Custom Job Commands

Every Product can have its own attached custom commands. Because Products are implemented in dynamically loaded Modules, and because users can create custom parameters using the Virtual Module system or the API, this manual cannot describe every possible command you may run into in your use of Smedge.

The sections below describe three basic Job implementations that are included in the core of Smedge. All of the Products currently included in the Smedge distribution will include the functionality of these three classes.

Process Jobs

Process Jobs allow Smedge to control a third party application by spawning and monitoring another process on the machine. It adds attributes and commands related to this process.

When a Process Job is running, the captured output from that process is available for viewing. Smedge does this by starting a TCP server for the output, and clients can connect to this server to get this output stream. You have two choices for making this connection:

View Captured Output	This command will request the entire history of the output when it connects. If there is a lot of output, it may take some time for all of it to transfer over the connection.
View Truncated Captured Output	This command will only request to view any new output from the time that the client connects and after. This will not get the back history.

Sequence Distributor

Jobs that use the Sequence Distributor allocate themselves by breaking up a sequence of numbers into small groups to be worked on. The idea of rendering a sequence of frames uses this concept, but you can use a Sequence Job to distribute anything that can be thought of as a sequence of numbers.

Sequence Distributor does not add any commands.

Render Jobs

Render Jobs allow the control of the rendering of an animation sequence. It includes several attributes and commands that aid in this process, and allow a general form of control for just about any sequence rendering system.

Checking and Viewing Rendered Image Files

For individual work units, Render Job adds a command that allows you to view the most recently detected rendered image file, using the **View Frame** command. The command line that will be spawned in response to this command is actually specified as a Shell option named **View Frame command line**.

For Jobs, you can use the **View Sequence** command to view a rendered image sequence. Because a single render can potentially produce a number of different frame sequences (depending on the operation of the third party tool itself), each different frame file sequence will be listed in a child menu, allowing you to pick which of the sequences you want to view. The command line that will be spawned in response to this command is actually specified as a Shell option named **View Sequence command line**.

Finally, Smedge includes the CheckFileSequence Shell program. This Shell is integrated with the Render Job class, allowing you to check any frame sequence. The CheckFileSequence program will actually check every expected frame file for existence, minimum size and consistency in size, and it allows you to easily resubmit an arbitrary subset of the original frame range. To access this tool, use the **Check File Sequences** command. The command line that will be spawned in response to this command is actually specified as a Shell option named **Check Sequences command line**.

Exploring the scene and image files

Render Jobs also supply two commands that allow you to explore in your OS shell the files associated with the Job. **Explore Scene Folder** will explore to the location where your scene file is. **Explore Images Folder** will explore to the location where the selected image sequence is.

Work Control

Pausing and Resuming work distribution

Smedge has the ability to "pause" a Job, which means that no work will be dispatched from that Job until you un-pause or "resume" it. Any currently outstanding work from that Job will continue and finish normally.

You can set a Job to be paused by checking the **Paused** check box on the Submit Job window for the Job. Alternatively, you can select one or more Jobs from the Job List, and select "Pause Job" from the Job menu (or context menu in the Job List) and all of the selected Jobs will be paused. If the Job was already paused, it will remain so.

To resume Jobs, you can similarly uncheck the **Paused** check box on the Submit Job window, or use the "Resume Job" command from the Job menu or context menu. If the Job was not paused when you resume it, it will remain un-paused.

Optionally, you can configure Smedge so that Jobs with a priority of 0 will be "paused". This is different than pausing a Job, which is actually stored separately in the Job's **Status** parameter. By default this behavior is off, meaning that Jobs with a priority of 0 will be worked on, with the lowest possible priority in the system. To enable this system, enable the "PriorityZeroIsPaused" option in the Master. You can do this from the Master Options dialog box.

Stopping work currently going

There are two ways you can stop work. You can select the work units you want to stop and signal them to abort, or you can select Engines that are working and disable them from performing work. For more information on disabling Engines, see Allowing Engines to Work and Allowing Engines to Work on Specific Products.

To stop specific work units, you can select the active workers in the Work List, and choose one of the three stop work commands from the Work menu. You have three choices for stopping work that is currently going. The difference between the three is how the work is handled after it has stopped. "Stop and Requeue" is the most basic way to stop work. The Master sends a request to the Engine to immediately stop working, and will reset the Work's status to queued, so that it will get dispatched again at the next opportunity.

"Stop and Divide" is a more advanced tool for requeuing work. If you have a job with a "Packe Size" of more than 1 frame, this command will first change the Job settings to set the "Packet Size" to 1, then stop the selected work units to requeue them divided up into single frame work units. This is useful when you have that one last packet that will take another hour to render all 10 frames, and you have 9 of your render farm machines sitting idle. Using this command will divide those last 10 frames across all 10 machines.

"Stop Permanently" will also tell the Master to request that the work is stopped immediately, but it will mark that work unit as permanently canceled. The work will never be dispatched again, and will count towards the completion of the Job.

"Stop and Submit as New Job" will also stop the work permanently as part of the original job, just like "Stop Permanently" does. But it will also create a new Job with just this work in it, and submit that to the system as a completely new and independent Job.

Custom work commands

Different Job types can have different custom commands that are associated with the Product. For example, every Product that spawns a child process on the Engine allows you to connect to the output from that process to view progress in real time.

Each Product could have its own custom commands. For more information about custom Job commands, see the Custom Job Commands chapter. You can also use the Status bar to see a short online help about the command. For custom Product types, contact the Module author or your Smedge Administrator.

Engine Control

Overview

SmedgeGui allows you to configure most engine settings for an Engine from a single tabbed interface. See the Configure Engines Window reference for complete information about the Engine editor and how you can use it to configure one or more Engines at once.

Common Engine Operations

Mapping network drive letters

Windows uses the concept of a drive letter as one way to access files. For distributed systems, it is common to use a file server that provides a shared network drive, and every Windows based client machine mounts that network drive on a drive letter, giving your end users access to the files as if the network drive was just another disk on the machine.

There are many ways to handle network drives in Smedge. For many users, especially on smaller networks, Smedge can automatically detect mapped network drives, and ensure that they are available, all without any user intervention. At most, you may need to configure your file server to allow the default Smedge user account access, or use a different user account when you install SmedgeEngine on your Engine machines. If that doesn't work for you, you can use the Mapped Drive Manager to easily and quickly configure the mapped drive settings for all Products on all currently online Engines. If you need further control, Products that allow mapping of drives provide ways for you to configure the resources manually, either by Job or as options for the Product on each Engine. See the Submit Job Window and the Configure Engines Window for more information.

Drive mappings can be set from any machine on any platform, but they will only be used by the SmedgeEngine process when running on a Windows based machine. Also, see

Adding a Note to an Engine

Notes attached to Engines are handled specially in SmedgeGui. Unlike every other Engine setting, you cannot modify an Engine's Note text through the Configure Engine dialog. Instead you must use the **Add Note** command from the Edit menu (or the Engine List context menu). An Engine's Note is displayed in the Configure Engine dialog for reference, but you cannot change it. To clear a note, you also use the **Add Note** command, and just leave the Note text blank.

Stopping Engines from doing Work

You can stop Engines from performing work by disabling them. Disabling an Engine simply stops it from taking on work. It does not release the license to perform work that the Engine may have already checked out from the Master. If you wish to free up a license, you must actually stop the SmedgeEngine process on the machine.

While you can use the to disable Engines, it is faster to simply select the Engines you want to enable or disable, and then choose **Disable Engine Immediately** or **Disable Engine Deferred** from the Engine menu or from the context menu of the Engine List. **Disable Engine Immediately** will modify the Engine setting, and request that the Engine immediately abort any work it is currently executed. The Work will be requeued for later processing. **Disable Engine Deferred** will modify the Engine setting, but will allow the Engine to finish processing any work it is currently working on. That work unit will finish normally and will be requeued according to its result status and post-processing tests. In either case, if the Engine is already disabled, calling this will have no effect on that Engine.

Enable Engine will set the Engine back to being able to process work. Work will be assigned to that Engine normally. If the Engine is already enabled, calling this will have no effect on that Engine.

Managing Products

By default, most Products are enabled on an Engine. There can be situations where you will want to change this. For example, if you have a small number of node-locked licenses for a renderer, you can either create a Pool with those machines, or you can simply disable that Product on the other Engines. You can use the Configure Engines Window to select which Products each Engine is allowed to work with.

You may also need to handle multiple versions of the Products. Most Smedge Modules have a system that allows you to configure multiple versions by simply creating or modifying an INI file. For more information about how you can customize the Products in Smedge, see the Administrator Manual.

Assigning Engines to Pools

Pools provide another way to organize and optionally to prioritize how your Engines will be assigned work. Engines can be assigned to any number of pools. When pool priority is enabled, the order in which the pools are assigned matters. Any work assigned to a higher priority pool will be dispatched to this engine before work from a lower priority pool, even if the first Job’s priority value is actually lower than the second. In other words, Pools trump the Job priority. Or, to think of it another way, Job priority only prioritizes Jobs in the same pool. When pool prioritization is disabled, only the Job priority will be used to order the jobs, and the pools will only be used to limit which machines can work on the job. If a Job is assigned to a Pool to which the Engine does not belong, then no work from this Job will ever be dispatched to this Engine.

Configuring the Pools works slightly different than you may be used to from other types of distribution management software. Instead of adding the Engine to the Pool, you add the Pool to the Engine. The reason for this is that each Engine prioritizes the Pools it works on, allowing you much more control over the prioritization of work from Pools. For example, if you have 2 Pools, "TV" and "Film," you can set half of your machines to give priority to "TV" Jobs and half to give priority to "Film" Jobs. Note that the order pools appear only matters if pool prioritization is enabled. If it is disabled, then the controls are all still available, but the order of the pools is simply ignored when distributing work.

To configure the Pools, you must select the Engine(s) you want to configure in the Engine List, then choose "Configure Pools" from the Engine menu or context menu in the Engine List. This opens the Pools tab of the Configure Pools dialog. See the reference section for more information about how this dialog box works. You can also use this dialog box to create, rename and delete pools from the system. Unlike changes you make to the Engine settings, creating, renaming and deleting pools are immediately broadcast to the entire system. For example, if you delete a pool, that pool will be immediately deleted from all machines, even if you cancel the Engine edits.

Setting Engine Options for Products

Most Products that you use with Smedge have options available to configure exactly how the work will be performed. SmedgeGui allows you to configure these options for any Engine from any machine on the system. It also gives you the ability to copy settings from one Engine to others, and to configure multiple Engines at the same time.

You can configure these options using the Configure Engines Window, using the Product Options tab (accessible from the **Configure Product Options** command in the Engine menu or the Engine List context menu). Exactly which options will be available for you to configure will depend on the Product you want to configure. For more information about what options are available and how they work, see the information about Product Parameters available in the Administrator Manual.

Many of these options are available both as Job parameters (usually under the Advanced Tab of the Submit Job window), and as Engine options as described above. In the case of any such parameter, if you supply a value for the Job parameter, that value will be used for that Job, and will override an Engine option. If you do not supply an override value as part of the Job, then the Engine's option value will be used. If neither is set, the Product will use its default value for this option or parameter where needed.

In general these options control overall behavior of the product being used to actually perform the work. Products also can have options for the display and interaction with SmedgeGui (and Shells in general) and options for the Master as well. You can modify the Shell options using the SmedgeGui Options dialog, and you can modify the Master options using the Product Settings tab of the Master Options dialog.

Configuring Several Engines at Once

Because Smedge controls many machines at the same time, and usually those machines are configured identically, Smedge is designed to allow you to configure many Engines at one time. Simply select all of the Engines you want to configure at once before opening the Configure Engines Window. Any changes you make are automatically propagated to all of the Engines you selected, without affecting other customized settings that may be different for each Engine.

You can also apply a customizable "default" configuration, customizable preset configurations you have previously saved, or load settings from an INI file, in order to quickly set all of the settings for every Engine at the same time. See the Configure Engines Window reference for more information.

Resetting Engine Failure Counts

When work fails, the system will maintain a failure count in order to keep a failing job from taking up too many resources. Once you have fixed whatever was causing the problem, you may want to reset the failure counts so that work will pick up again from the Job that had previously failed.

You can reset the Job failure counts either by Job or by Engine. To reset failures by Engine, select one or more Engines in the Engine list and select the “Reset Engine Failure Counts” command in the Engine menu (or from the context menu in the Engine List). All error counts associated with those particular Engines will be reset, and any work will be dispatched to those Engines normally from all available Jobs. To reset failures by Job, see Resetting failure counts in the Job Control section.

You can adjust the count limits before work is no longer set by adjusting the Master’s settings. See Controlling the Master’s Dispatching Options.

Administrative Control

Configuring the Connection

By default, Smedge components will attempt to locate the Master automatically. This system uses UDP broadcasts to locate the Master on a subnet. For almost all users, this system is adequate, and there is no need to manually configure your connection. In general, we recommend you leave the automatic system in place unless you have a specific purpose or reason for manually configuring your connection.

There are two common reasons why you may need to manually configure your connection. First, if the machine you are using is on a different subnet than your Master, the automatic location mechanism may not find the Master. This is because most routers do not send UDP broadcast traffic between subnets. In this case, you can configure the connection information that all Smedge component applications installed will use to find the Master. Second, if you have multiple separate networks running in parallel (for example, a production network, and a smaller test network for developing and debugging Modules), you may want to switch the network that you are viewing in SmedgeGui. In this case, you can configure the connection for the current user's preferences for the SmedgeGui program only.

Note that configuring the connection for the current user will override whatever connection was configured in the application folder. For more information, see the Configure Connection dialog reference.

Administrator Mode

Smedge has the ability to restrict access to portions of the interface. But in order to allow easy maintenance, Smedge has the concept of an "Administrator" (See Administration). Entering "Administrator Mode" will temporarily unlock every feature of the GUI, allowing full system administration. Any restrictions will also be ignored (See Restricting What Users Can Do).

To enter Administrator Mode, select "Administrator Mode" from the System menu. If there is a password set in the system, you will have to enter your password to continue. Once you are in Administrator Mode, you can tell by looking at the SmedgeGui window title bar. When you are in Administrator Mode, it will say "Administrator Mode" in the title.

Some commands, particularly anything that controls the Master or the system as a whole, are only available when you are in Administrator Mode.

Making Administrator Mode the Default

You can set the GUI to always start in Administrator Mode, so that you don't have to log in each time. If you have a password, you will not have to type it every time. You must be in Administrator Mode already to configure this.

Select "Administrator Options…" from the System → System Commands menu. Here you can specify if you want the GUI to start in Administrator Mode. You can also specify the Administrator Mode password, and customize the message that appears in the SmedgeGui About box.

Changing the Administrator Password

Select "Administrator Options…" from the System → System Commands menu. Here you can specify if you want the GUI to start in Administrator Mode. You can also specify the Administrator Mode password, and customize the message that appears in the SmedgeGui About box.

Configuring the Master

You can use the SmedgeGui program to control how the Master dispatches work. You must be in Administrator Mode to access this functionality. Select "Configure Master…" from the System menu, and the select the "Distribution" tab on the Master Options dialog. Here you can configure several options that the Master uses to dispatch work.

On the "Job Type Settings" tab, you can set up global restrictions on the number of work units for specific Products that will be dispatched. Select the product you wish to restrict, and then use the other controls to configure the limit you want.

Restricting What Users Can Do

You can restrict what control the SmedgeGui will give users by default. These restrictions are stored on the Master, and passed to the GUI when it connects to the Master. The Master relies on the Shell programs voluntarily following these restrictions, as the SmedgeGui program does. Note, however, that the command line Shell components currently ignore the restrictions and maintain full functionality no matter what restrictions you have set in place. Entering Administrator Mode will bypass any restrictions you have set.

You can set or remove restrictions in the "Restrictions" tab of the Master Options dialog. Restrictions are a text string based system. If the restriction name is listed, then that feature will be restricted when users start SmedgeGui in normal mode. To unrestrict an item, remove the restriction name from the list.

The names of the currently known restrictions are available in the dropdown list where you can type the names in. More restriction names may be added in the future or used by a custom Module, and there is no limit to either the number or format of the names. To see a full list of the restrictions available by default, see the "Restrictions" chapter in the Administrator Manual.

Smedge implements some restrictions by default to keep the interface a little cleaner. The current list of restrictions that are in place by default can be seen in the "Restrictions" chapter of the Administrator Manual.

Configuring automatic path translations

The Master can translate file and folder paths between the various platforms that it Smedge runs on. These translations are stored on the Master, and used by any client that connects to that Master. The Master sends the translations as part of the initial connection protocol, and then the client uses the current translations when reading any path data sent by the Master, or when requested as part of a Job parameter command (See "Parameters" in the Administrator Manual for more information on parameters and parameter commands).

Each translation set contains a path "root" for each of the platforms on which Smedge runs (currently, Linux, Macintosh OS-X, and Windows). If you only use two of the three platforms, you can leave the unneeded platform's field blank. You can add as many path translations as you wish. Translations are searched for in the order they appear, so you can have multiple translations for the same path. For example, both a drive letter and a UNC path on Windows can map to the same mount point on a Linux or Mac file system.

See the Path Translations Tab reference for more information.

Limit number of workers by creator

The Master can limit the number of outstanding workers assigned to users and their created jobs. You can set or remove limits in the "Creator Job Limits" tab of the Master Options dialog. Limits are applied to creators by names, and limited by the supplied number. Creators that are not listed in this limit list, have a default unlimited number of workers that can be assigned to them. To remove a limit on a creator / user, remove the creator name from the list. The user limit system uses Perl style regular expression matching to find which limit applies to a specific Job. This allows you a high level of flexibility in how you use the Creator string for the Job, and in how the limits can be applied.

See the Creator Job Limits Tab reference for more information.

Licensing Smedge

Smedge only requires a license to actually run renders. All other Smedge components can be used on as many machines as you wish. The license information is maintained by the Master, and licenses are encrypted using a hardware ID related to the network interface information of your system. In Administrator Mode, you can select "Submit License" from the System → System Commands menu. This will bring up the Submit License dialog.

The top control contains the ID string that is used to decrypt the license code. Make sure to send this to Überware licensing, or use this code on the Überware licensing web page in order to generate a license that will be able to be decrypted by your system. This number is based on information on the Master machine, so it does not matter from which machine you submit the license. However, if the networking hardware in your Master machine ever changes, or if you change which machine is running the Master, you will need to have a new license code generated. You can send a new request to Überware licensing, or you can use the Überware client home page, allowing you to regenerate your own license code whenever you wish.

You enter the code that you receive back in the bottom field. The code is sent to the Master, which will try to decrypt it. If the decryption succeeds, this new license information will replace any existing license information. You can always enter any license code that can be decrypted by your Master machine at any time. So, if you have a permanent license and get a temporary license code with a few extra machines, when that temporary code expires you can just re-enter the original permanent code and get your old license back.

You can submit a license from any machine on your system. Every machine will display the same license information. If the license is correctly installed, the GUI will show the license information in the about box. If the license installation fails, it will report the failure.

Smedge includes multiple types of licenses. For information about how licenses work and how they are actually distributed and managed, see the **Licensing** Chapter in the **Administrator Manual**.

System-wide exit

Smedge has the ability to stop itself across the entire system with a single request. You must be in Administrator Mode to execute this command. Select "System-wide Exit" from the System → System Commands menu. You will be asked to confirm the shutdown but you can check the box to not have to confirm this in the future.

The Gui will send the request to the Master, which then forwards it to every connected client. The Master will try to remain running until all the clients have stopped, at which point it will also then stop itself. It is very useful to quickly stop the whole system, for example when upgrading Smedge. However, there is no way to initiate a system startup, because the software would already have to be running in order to do so! However, this is usually a pretty easy task to accomplish with batch scripts or simple shell scripts. Ask your IT person or system administrator for more information.

This system only stops the Smedge componenet application processes that are running and connected, and aborts any work being managed by those processes. It does not stop any other processes on the machine, nor does it initial a hardware shut down. There is currently no mechanism in Smedge to manipulate hardware.

Removing offline Engines from the system

When Engines are offline, the Master will still maintain all of the last known information about them, and will send that information to any shell that cares. Offline Engines show up in the Engine List of the Main window. Because Engines are actually sorted by ID string, it is possible that the ID of an Engine may change (say if the options are deleted for example). This can leave a phantom offline Engine that you no longer need.

You can delete offline Engines from the system in two ways. First, you can select one or more offline Engines and select "Remove offline Engine" from the Engine menu. This will remove any of the selected Engines that are currently offline from the system. You can also simply select "Remove all offline Engines" from the Engine menu, which will remove every Engine currently offline from the system.

Any time an Engine comes online, it updates the Master with all of its current settings and information. If you previously removed an Engine, then start it up again, all of its information will be restored when it connects, just like if it was connecting for the first time.

Reference

Menus

System

Command	*Description*	*Restriction*
Connect	*(only available when not connected)* Try to establish a connection with the Smedge Master	
Reconnect	*(only available when connected to a Master)* Try to reconnect to the Smedge Master	
Refresh	*(only available when connected to a Master)* Refresh all current data from the Master	
Engine Mode	*(only available when connected to a Master)* Set the GUI into Engine mode to reduce overhead	
Configure Connection…	Configure how to connect to the Smedge Master	Configure Connection
SmedgeGui Options…	Set options for SmedgeGui operation and interaction with Modules	
Smedge Colors	Lets you select the color scheme you want for all of the Smedge GUI components.	
Reset all confirmation messages	All confirmation messages will be shown again.	

Command	*Description*	*Restriction*
Smedge Files Browse the Application Folder	Opens a local operating system window showing the contents of the Smedge program folder	
Smedge Files Browse the Utilities Folder	Opens a local operating system window showing the contents of the Smedge utilities folder	
Smedge Files Browse the Log Folder	Opens a local operating system window showing the contents of the Smedge log folder	
Smedge Files View the History.log	Opens a window that shows the History log from SmedgeGui. This is the log of the GUI operation.	
Smedge Files Browse the User Folder	Opens a local operating system window showing the contents of the Smedge user folder	
Smedge Files Browse the Machine Folder	Opens a local operating system window showing the contents of the Smedge machine folder	
Smedge Files Browse the TEMP Folder	Opens a local operating system window showing the contents of the system TEMP folder	
Start Aegis	Starts the Aegis Shell program	Process Control
Start Herald	Starts the Herald Shell program	Process Control
Start Conspectus	Starts the Conspectus Shell program	Process Control
Components Set System Default Component Startup	Allows you to set which components will start with SmedgeGui by default on all machines.	Process Control OR Core Process Control
Components Start the Engine Now	Tries to start the SmedgeEngine process on the local machine, if it is not already running. This makes the machine available to do work.	Process Control OR Core Process Control
Components Stop the Engine Now	Tries to stop the SmedgeEngine process on the local machine, if it is running. This stops the machine from being available to work and releases the license, if it has checked on out.	Process Control OR Core Process Control
Components Remove Engine Lock File	Tries to remove the SmedgeEngine.lock file that is used to prevent multiple instances of the Engine from starting on a machine at the same time. Try using this if the Engine refuses to start correctly	Process Control OR Core Process Control
Components Engine Service	Submenu for Engine service control.	Process Control OR Core Process Control
Components Engine Service Install the Engine as a service	Tries to install the SmedgeEngine component as a Service (or daemon) on the machine. You must have appropriate permission to install and control services or daemons on the machine.	Process Control OR Core Process Control

Command	*Description*	*Restriction*
Components Engine Service Start the Engine service	Tries to start the SmedgeEngine component as a Service (or daemon) on the machine. You must have appropriate permission to install and control services or daemons on the machine.	Process Control OR Core Process Control
Components Engine Service Stop the Engine service	Tries to stop the SmedgeEngine component as a Service (or daemon) on the machine. You must have appropriate permission to install and control services or daemons on the machine.	Process Control OR Core Process Control
Components Engine Service Remove the Engine Service	Tries to uninstall the SmedgeEngine component as a Service (or daemon) on the machine. You must have appropriate permission on the machine to perform the operations.	Process Control OR Core Process Control
Components Engine Control Service	Submenu for Engine Control service control.	Process Control OR Core Process Control
Components Engine Control Service Install the Engine Control service	Tries to install the Engine Shell component as a Service (or daemon) on the machine to speed up performance of the Engine shell. You must have appropriate permission to install and control services or daemons on the machine.	Process Control OR Core Process Control
Components Engine Control Service Start the Engine Control service	Tries to start the Engine Shell component as a Service (or daemon) on the machine. You must have appropriate permission to install and control services or daemons on the machine.	Process Control OR Core Process Control
Components Engine Control Service Stop the Engine Control service	Tries to stop the Engine Shell component as a Service (or daemon) on the machine. You must have appropriate permission to install and control services or daemons on the machine.	Process Control OR Core Process Control
Components Engine Control Service Remove the Engine Control Service	Tries to uninstall the Engine Shell component as a Service (or daemon) on the machine. You must have appropriate permission on the machine to perform the operations.	Process Control OR Core Process Control
Components Start the Master Now	Tries to start the SmedgeMaster process on the local machine, if it is not already running. This makes the machine available to manage the system if needed, and keeps a backup of the data if allowed.	Process Control OR Core Process Control
Components Stop the Master Now	Tries to stop the SmedgeMaster process on the local machine, if it is running. This stops the machine from being available to manage the system, and stops keeping the data synchronized locally.	Process Control OR Core Process Control

Command	*Description*	*Restriction*
Components Remove Master Lock File	Tries to remove the SmedgeMaster.lock file that is used to prevent multiple instances of the Master from starting on a machine at the same time. Try using this if the Master refuses to start correctly	Process Control OR Core Process Control
Components Master Service	Submenu for Master service control.	Process Control OR Core Process Control
Components Master Service Install the Master as a service	Tries to install the SmedgeMaster component as a Service (or daemon) on the machine. You must have appropriate permission to install and control services or daemons on the machine.	Process Control OR Core Process Control
Components Master Service Start the Master service	Tries to start the MasterEngine component as a Service (or daemon) on the machine. You must have appropriate permission to install and control services or daemons on the machine.	Process Control OR Core Process Control
Components Master Service Stop the Master service	Tries to stop the MasterEngine component as a Service (or daemon) on the machine. You must have appropriate permission to install and control services or daemons on the machine.	Process Control OR Core Process Control
Components Master Service Remove the Master Service	Tries to uninstall the MasterEngine component as a Service (or daemon) on the machine. You must have appropriate permission on the machine to perform the operations.	Process Control OR Core Process Control
Administrator Mode	*(only available in Normal Mode)* Enter Administrator Mode	
Normal Mode	*(only available in Administrator Mode)* Return to Normal Mode	
System Commands Administrator Options…	*(only available in Administrator Mode)* Configure Administrator options	
System Commands Module Manager…	*(only available in Administrator Mode)* Control Modules that will be used in the system.	
System Commands Configure Master…	*(only available in Administrator Mode)* Configure the Master distribution and product options. You can also configure the Restrictions here.	
System Commands Submit License…	*(only available in Administrator Mode)* Submit a new license to the Master	
System Commands Execute a Command…	*(only available in Administrator Mode)* Lets you manually run a command line on the Master outside of the Job queue	

Command	*Description*	*Restriction*
System Commands Dump all logs	*(only available in Administrator Mode)* Requests every connected Smedge client to dump every log for debugging	
System Commands View Master History.log	*(only available in Administrator Mode)* Opens a window that shows the History.log from SmedgeMaster. This is a log of the operation of the Master	
System Commands View License Report	*(only available in Administrator Mode)* Opens a window that shows the license report from SmedgeMaster. This is a report of the installed licenses and lists which type of license any engine may have and when point licenses expire.	
System Commands View Current Dispatch.log	*(only available in Administrator Mode)* Opens a window that shows the current Dispatch.log (a new log will be generated when you select the command to open the window). This is a log of the distribution loop , useful for debugging problems where Jobs don't distribute properly.	
System Commands Restart Dispatch Logging	*(only available in Administrator Mode)* Starts the dispatch log on the master going again without opening a new window with the View Current command above.	
System Commands Force Master to Resign	*(only available in Administrator Mode)* Forces the currently connected Master to resign its duties, allowing connection to a different Master or the Master to restart itself.	
System Commands System-wide Exit	*(only available in Administrator Mode)* Request a system-wide shutdown of all Smedge components	
Exit	Closes the SmedgeGui program	

Engine

Command	*Description*	*Restriction*
Enable Engine	Enables the selected Engine(s)	Edit Engine
Disable Engine Immediately	Disables the selected Engine(s), aborting any currently going work if needed.	Edit Engine
Disable Engine Deferred	Disables the selected Engine(s), but lets any currently going work finish normally	Edit Engine
Add Note…	Adds a note to the selected Engine(s). Opens the Add Note dialog.	Edit Engine

Command	*Description*	*Restriction*
Configure Engine Settings…	Change settings for the selected Engine(s). Opens the Configure Engines dialog.	Edit Engine
Configure Product Settings…	Configure the Module settings (Engine options). Opens the Engine Product Options dialog.	Edit Engine
Configure Pools…	Configure the Pools for the selected Engine(s). Opens the Configure Pools dialog.	Edit Engine
Configure Event Commands…	Configure the Event commands for the selected Engine(s). Opens the Configure Pools dialog.	Edit Engine
VNC to this Engine	Opens a VNC viewer that will try to connect to the selected Engine.	VNC
View the Engine History.log file	Opens a window to view the History.log file from the selected Engine. This is a log of the operation of the SmedgeEngine process.	
View Engine Dispatch Report	Opens a window to view the dispatch report for the selected Engine. This is useful for determining why an Engine is not taking on work without having to wade through the entire Dispatch.log file.	
Reset Engine Failure Counts	Reset the failure counts for the selected Engine(s). Jobs that had previously failed too many times on the Engine will be allowed to start work again.	Reset Failures
Remove offline Engine	*(only available in Administrator Mode)* Remove an offline Engine from the system.	
Remove all offline Engines	*(only available in Administrator Mode)* Remove any currently offline Engines from the system.	
Mapped Drive Wizard	*(only available in Administrator Mode)* Allows easy control over network drive remapping for all Products on all Engines.	
Machine Control Execute a Command...	*(only available in Administrator Mode)* Immediately execute a given command on all selected Engine(s)	
Machine Control Any License	*(only available in Administrator Mode)* Allow the machine to check out any license	
Machine Control Legacy License Only	*(only available in Administrator Mode)* Only allow the selected Engine(s) to check out a legacy license.	
Machine Control Point License Only	*(only available in Administrator Mode)* Only allow the selected Engine(s) to check out a point license	
Machine Control Stop All Smedge Processes	*(only available in Administrator Mode)* Stops all Smedge component applications on a specific node.	

Command	*Description*	*Restriction*
Machine Control Sleep the Machine	*(only available in Administrator Mode)* Forcibly sets the selected Engine(s) to standby/sleep mode	
Machine Control Wake the Machine	*(only available in Administrator Mode)* Sends a Wake-On-Lan packet to the selected Engine(s) to try and wake them from standby/sleep mode	
Machine Control Reboot The Machine	*(only available in Administrator Mode)* Forcibly reboots the selected Engine(s)	
Machine Control Shutdown the Machine	*(only available in Administrator Mode)* Forcibly shutdown (power off) the selected Engine(s)	

Job

Command	*Description*	*Restriction*
New Job…	Opens a brand new Submit Job window.	Submit Job
Load Job…	Loads Jobs from a saved Job file (.sj file). Loads the Jobs directly into the system.	Submit Job
Repeat Job…	Shows a list of recently submitted Jobs. Opens the Repeat Job window modelessly.	Submit Job
Pause Job	Pauses the selected Job(s)	Change Job
Resume Job	Resumes (un-pauses) the selected Job(s)	Change Job
Preempt Workers with Lower Priority	Stops any workers from Jobs that have a lower priority than the lowest priority Job you have selected. Note that this does not account for Pool prioritization, and can stop other users' Jobs.	Stop Work My Jobs Only
Stop Workers from Job	Stops and requeues all work from the selected Job(s).	Stop Work
Edit Job…	Edit parameters from the selected Job(s). Opens a Submit Job window for each selected Job.	Change Job
Set Parameter…	Edit a single parameter from the selected Job(s). Opens a small window with two text fields for you to enter the name of the parameter and the value to set it to.	Change Job
Increase Priority	Increases the priority of the selected Job(s) by 1.	Change Job
Decrease Priority	Decreases the priority of the selected Job(s) by 1.	Change Job
Set Pool	Allows you to set the Pool that the selected Job(s) are assigned to using a menu of all available Pools. You cannot select individual	Change Job

Command	*Description*	*Restriction*
	Engines as a Pool here. Use Edit Job to open a Submit Job window to do this.	
Add Note	Add a note to the selected Job(s). Opens the Add Note dialog.	Change Job
Include Engines…	Manually add Engines to work on this Job. Opens the Include Engines dialog.	Change Job Edit Engine
Exclude Engines…	Force specific Engines not to work on this Job. Opens the Exclude Engines dialog.	Change Job Edit Engine
View Job History	Show the full history of the selected Job(s). Opens the Job History window showing the history in its own window.	
View Job Graph	Show the graph of all work for the selected Job(s). Opens the Job Graph window showing the graph in its own window	
View Job Dispatch Report	Show the dispatch report for the selected Job(s). This allows you to see why specific Jobs are not being distributed without having to wade through the entire Dispatch.log from the Master.	
View Job Distributor Status	Show the distributor status window from the selected Job(s). This allows you to verify what work the distributor has sent and what has finished.	
Copy Job…	Opens new Submit Job window(s) with a copy of all of the settings from the selected Job(s).	Submit Job
Resubmit Job	Submit the selected Job(s) as new Job(s)	Submit Job
Reset Job Failure Counts	Reset the failure counts for the selected Job(s). Any Jobs that have stopped sending work because of too many failures will be able to send new work again.	Reset Failures
Execute a command	Allows you to immediately execute an arbitrary command on your local machine using data from the selected jobs. The command string you enter will be run one time for each selected job with that job's data used for the variable substitutions.	Execute Job Commands
Save Job…	Saves the selected Job(s) as a Smedge .SJ Job file. Any Job dependencies will be maintained as long as both the Job that is waiting and the Job it is waiting for are both included in the save.	
Mark Job as Finished	Permanently cancels the Job(s), so that no new work is started. Any currently executing work will be allowed to finish normally.	Change Job
Delete Job Immediately	Deletes the selected Job(s) from the system, immediately canceling any active work from the Job(s).	Delete Job

Command	*Description*	*Restriction*
Delete Job Deferred	Deletes the selected Job(s) from the system only after any currently active work finishes normally.	Delete Job
Delete Finished Jobs	*(only available in Administrator Mode)* Deletes all finished Jobs from the system.	
Archived Job Manager	*(only available in Administrator Mode)* Opens the Archived Job Manager window to allow you to view, restore or remove jobs that have been deleted from the system.	

Work

Command	*Description*	*Restriction*
Stop and Requeue	Stops the selected Work items and sets them to be requeued for later processing	Stop Work
Stop and Divide	Changes the Job's packet size to 1 before stopping the selected work units and requeueing their frames.	Stop Work OR Change Job
Stop Permanently	Stops the selected Work items permanently. The work status is set to Canceled	Stop Work
Stop and Submit as New Job	Stops the Work permanently, but then creates a brand new Job with just the work you stopped in it.	Stop Work
VNC to this Engine	Opens a VNC viewer that will try to connect to the Engine that is processing the selected Work.	VNC

View

The views available in the View menu depend on the customized views you may have configured.

Command	*Description*	*Restriction*
Overview	*Shows the **Overview** pane on the main Window. See Views.*	
My Jobs	*Shows the **My Jobs** pane on the main Window. See Views.*	
Engines	*Shows the **Engines** pane on the main window. See Views.*	
Jobs	*Shows the **Jobs** pane on the main window. See Views.*	
Work	*Shows the **Work** pane on the main window. See Views.*	

Command	*Description*	*Restriction*
Customize Views	Shows or hides the **Customize** control pane on the main window. See Views.	Customize Views
Windows	Sub menu that shows all open floating windows, allowing you to find and activate specific windows	
Windows Main Window	Brings the main window to the top of the window stack (only useful on Mac, where the menu bar is not attached to the window itself)	
Windows Collect All	Brings all open windows centered over the main window	
Windows Close All	Closes all child windows and modeless dialogs at one time	
Select Previous	Moves backward through your selection history	
Select Next	Moves forward through your selection history	
Info Pane	Shows or hides the Info panel	
Toolbar	Shows or hides the Toolbar	
Status Bar	Shows or hides the status bar	

Column

Command	*Description*	*Restriction*
|<< (Move Far Left)	Move the selected column to the far left of the view list	
< (Move Left)	Move the selected column one over to the left	
> (Move Right)	Move the selected column one over to the right	
>>| (Move Far Left)	Move the selected column to the far right of the view list	
Remove	Removes the selected column from being displayed	
Custom Columns Add Columns...	*(only Job and Work lists)* Brings up a GUI window to specify a custom display column	
Restore Defaults	Restores the columns displayed to the default set and order	
Available Column: Item	Shows or hides the named column item depending on whether its currently checked or not.	

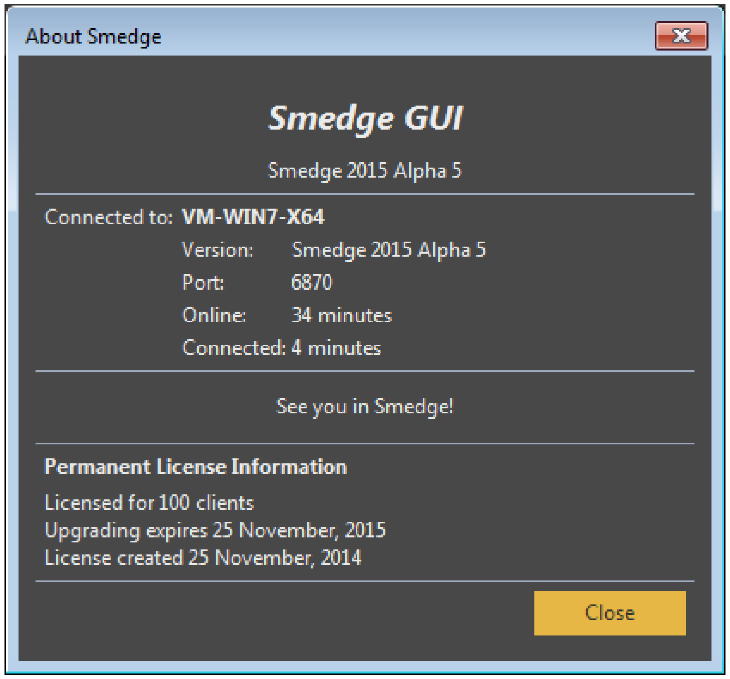
Toolbar

The toolbar can be shown from the **View** > **Toolbar** menu command. It provides shortcuts to some common menu commands:

Tool	*Menu Command*	*Restriction*
	System > Start Aegis	Process Control
	System > Start Herald	Process Control
	System > Start Conspectus	Process Control
	Engine > Enable Engine	Edit Engine
	Engine > Disable Engine Deferred	Edit Engine
	Engine > Disable Engine Immediately	Edit Engine
	Engine > Configure Engine Settings...	Edit Engine
	Engine > Configure Product Options...	Edit Engine
	Engine > Configure Pools...	Edit Engine
	Engine > Configure Event Commands	Edit Engine
	Job > New Job...	Submit Job
	Job > Edit Job...	Change Job
	Job > Pause Job	Change Job
	Job > Resume Job	Change Job
	Job > Increase Priority	Change Job
	Job > Decrease Priority	Change Job
	Job > Stop Workers from Job	Stop Work
	Job > View Job History	
	Job > View Job Graph	
	Job > Check File Sequences	
	Work > View Process Output	
	Work > Stop and Requeue	Stop Work
	Work > Stop Permanently	Stop Work
	View > Select Previous	
	View > Select Next	
	View > Info Pane	

Dialog Boxes

About SmedgeGui



Smedge 2015 Alpha 5

Connected to: **VM-WIN7-X64**

Version:	Smedge 2015 Alpha 5
Port:	6870
Online:	34 minutes
Connected:	4 minutes

See you in Smedge!

Permanent License Information

Licensed for 100 clients
Upgrading expires 25 November, 2015
License created 25 November, 2014

This dialog shows basic information about SmedgeGui and your Smedge system.

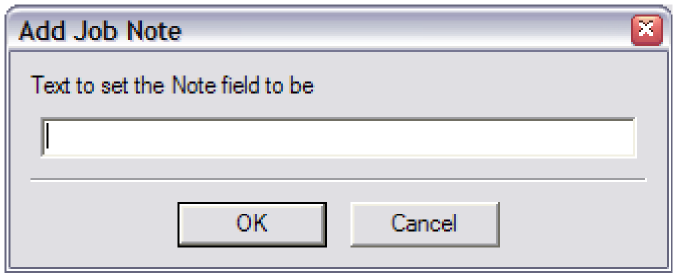
The top section shows the version information for SmedgeGui running on your computer.

The second section shows you information about the Master you are connected to. You can see the machine name and port you are connected to, how long it has been onine and how long you have been connected, and the version information of the Master.

The third section shows the System Message.

The fourth section shows the license information.

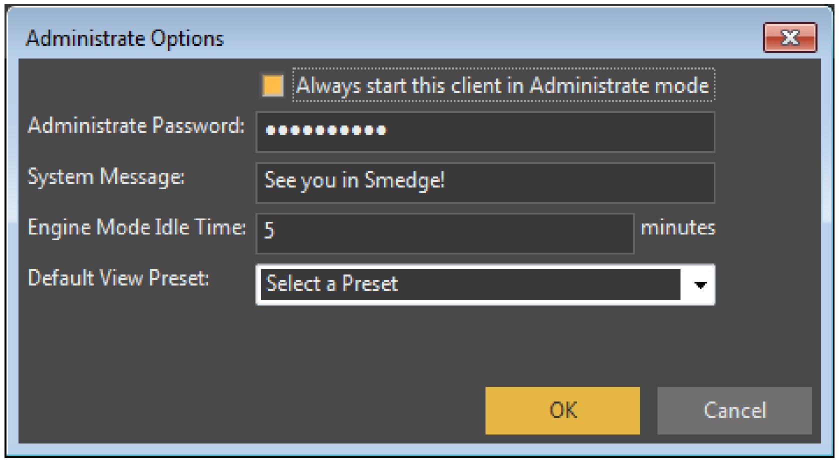
Add Note


Add Job Note

Text to set the Note field to be

Allows you to set the **Note** value for Engines. Put the desired note text in the text box, then press **OK** to submit the Change. If you press **Cancel**, no changes will be made.

Administrate Options


Administrate Options

Always start this client in Administrate mode

Administrate Password: ●●●●●●●●●●

System Message: See you in Smedge!

Engine Mode Idle Time: 5 minutes

Default View Preset: Select a Preset

You can configure some "Administrator" only properties here. The checkbox at the top, **Always start this client in Administrator Mode** is a SmedgeGui specific option. If checked, SmedgeGui will start up in Administrator Mode, if unchecked it will start in Normal Mode. See Adminstrator Mode for more information.

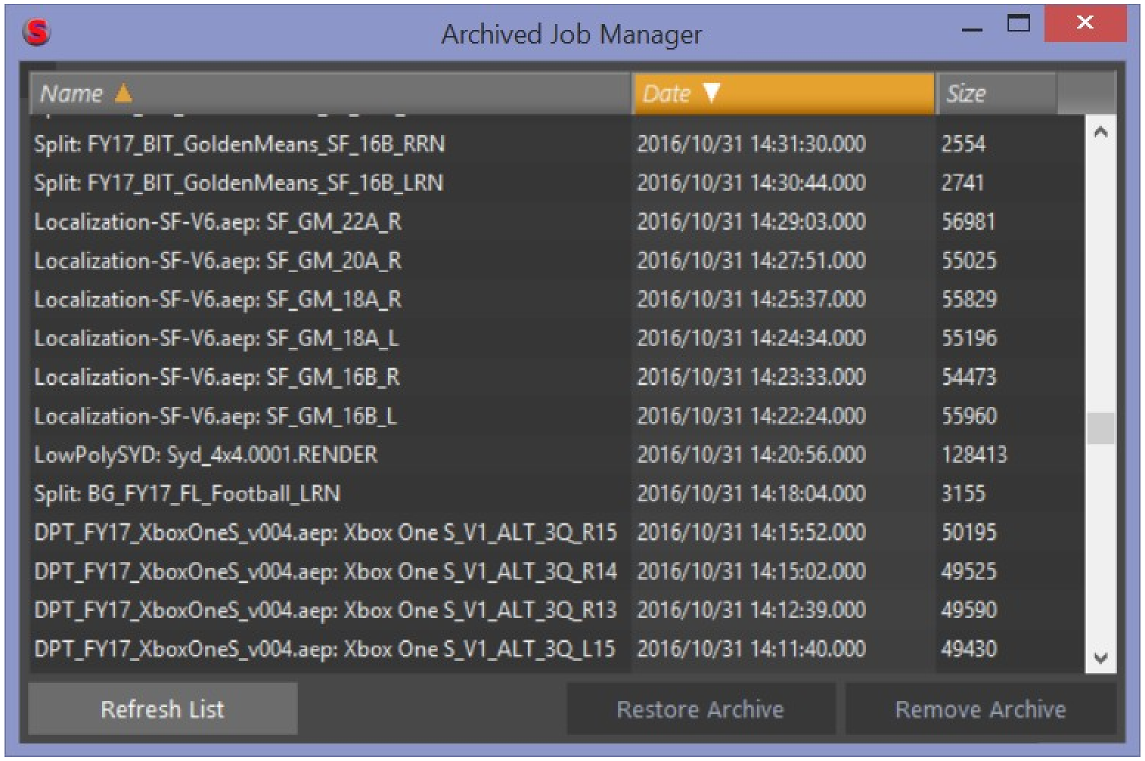
The **Adminstrate Password** can be changed here. Note that the password characters are always hidden. Make sure to remember your password. You cannot get the password from any Smedge option file, but if you need to reset it, you can delete the key "Administrate" under the section "Setup" in the SmedgeMaster.ini file. **Warning:** if you do this, you will have no password on Administrate Mode on the entire network. If you need this security, be sure to set a new password!

You can modify the system "message" here as well. This is a welcome message that is sent to every client when they connect to the Master. SmedgeGui displays this message in the About SmedgeGui dialog, accessed by selecting "About SmedgeGui" in the Help menu.

You can set the system-wide default timeout for connected GUIs to enter Engine Mode. The time you set here can be overridden in the SmedgeGui Options dialog for an individual machine. Set the time to 0 to disable Engine Mode by default.

You can select the default GUI view preset here as well. The preset you select will be automatically applied to all connected GUIs (if it has changed), overriding any customized GUIs that users may have configured themselves.

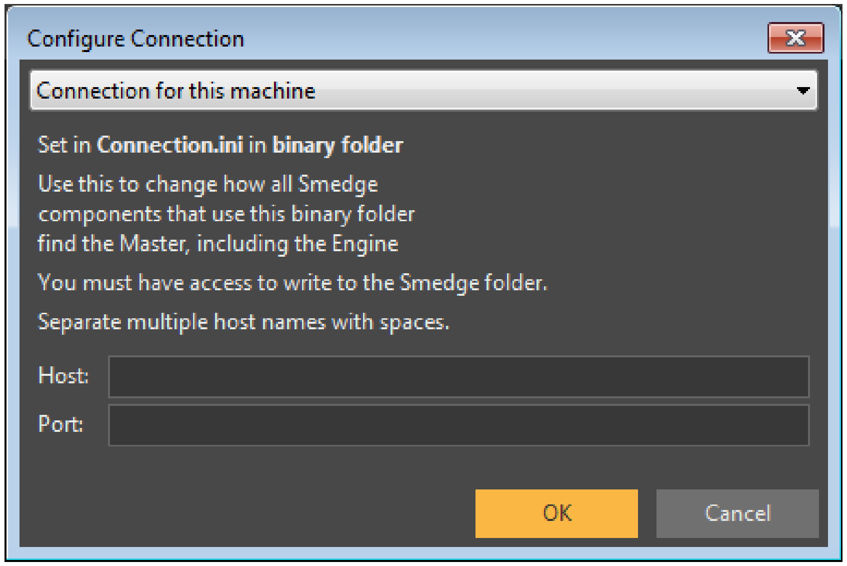
Archived Job Manager



Name	Date	Size
Split: FY17_BIT_GoldenMeans_SF_16B_RRN	2016/10/31 14:31:30.000	2554
Split: FY17_BIT_GoldenMeans_SF_16B_LRN	2016/10/31 14:30:44.000	2741
Localization-SF-V6.aep: SF_GM_22A_R	2016/10/31 14:29:03.000	56981
Localization-SF-V6.aep: SF_GM_20A_R	2016/10/31 14:27:51.000	55025
Localization-SF-V6.aep: SF_GM_18A_R	2016/10/31 14:25:37.000	55829
Localization-SF-V6.aep: SF_GM_18A_L	2016/10/31 14:24:34.000	55196
Localization-SF-V6.aep: SF_GM_16B_R	2016/10/31 14:23:33.000	54473
Localization-SF-V6.aep: SF_GM_16B_L	2016/10/31 14:22:24.000	55960
LowPolySYD: Syd_4x4.0001.RENDER	2016/10/31 14:20:56.000	128413
Split: BG_FY17_FL_Football_LRN	2016/10/31 14:18:04.000	3155
DPT_FY17_XboxOneS_v004.aep: Xbox One S_V1_ALT_3Q_R15	2016/10/31 14:15:52.000	50195
DPT_FY17_XboxOneS_v004.aep: Xbox One S_V1_ALT_3Q_R14	2016/10/31 14:15:02.000	49525
DPT_FY17_XboxOneS_v004.aep: Xbox One S_V1_ALT_3Q_R13	2016/10/31 14:12:39.000	49590
DPT_FY17_XboxOneS_v004.aep: Xbox One S_V1_ALT_3Q_L15	2016/10/31 14:11:40.000	49430

This dialog shows a list of all of the jobs you have deleted. The date is the date when you deleted the job, and the size is the size of the job and history data on disk. Select jobs here and use **Restore Archive** to restore the deleted data back to the system or use **Remove Archive** to remove it completely. Note that there is no way to undo a permanent removal. Use the **Refresh List** button to download the list of archived jobs again. Note that jobs deleted while the window is open will get added automatdically.

Configure Connection



Configure Connection

Connection for this machine

Set in **Connection.ini** in **binary folder**

Use this to change how all Smedge components that use this binary folder find the Master, including the Engine

You must have access to write to the Smedge folder.

Separate multiple host names with spaces.

Host:

Port:

This dialog allows you to configure basics about how your connect to the Master. It allows configuration of three different types of connection:

Your SmedgeGui Connection.

These controls allow you to set an address and port number that will only be used by the SmedgeGui program when you are logged into the computer. This will not affect how any other users will connect, nor will it affect any other Smedge component or any other machine.

Connection for this machine.

These controls allow you to set an address and port number in the Connection.ini file in the Smedge binary folder. This will affect how every Smedge component that starts from this binary folder will try to connect to the Master. If you are using a Local installation, this means every Smedge component on the local machine. If you are using a Shared Installation, this means every Smedge component on your entire network.

By default, Smedge communicates on port 6870. If you do not supply the port, it will use the default port. If you do not set a host, then you can use this to configure an alternate Master port but still use the automatic Master location system on this alternate port.

You do not need to supply host/port pairs for both the current user and for the binary folder. If you do supply both, the settings for the current user (on the **Your SmedgeGui Connection** page) will override the settings for the binary folder (on this page). To reset to the default behavior, make sure that all controls are blank, which will allow the automatic Master detection system to work. See Configuring the Connection for more information.

Client Listening Network.

This control allows you to specify the IP address of one of the network interfaces installed on your machine that will be used to establish the “listening” point for the client applications. This is useful if you have multiple network interfaces installed, and want to control which one is used for Smedge communication and control data.

For example, if you have an ethernet connection and a fiber optic connection to the file server, you may need to ensure that Smedge communication happens only on the ethernet network. Rather than trying to rely on the OS to pick the network to use, you can specify the address of the ethernet network, and ensure that the fiber network is reserved only for file server access.

Configure Engines

The Configure Engines window provides the central location for changing data associated with the Engine. You can use it to configure multiple engines at the same time, each with its own custom data, or sharing common data. This window is a “modeless” window. This means that you can open this window, and still manipulate the main window and the menu commands. You can also open multiple windows to configure multiple Engines or multiple sets of Engines at the same time.

The controls grouped in the ***Show Settings for Engine*** section show you which Engine you are currently viewing. If you selected more than one engine when you opened this dialog, every Engine will be available in the drop down list at the top. Directly underneath the list of names some basic information about the selected Engine will be displayed.

If you change the selected Engine, all of the information displayed will update to reflect the current settings for the specific Engine you selected.

When you change anything about the currently displayed Engine, that change will be automatically propagated to every Engine as you make it. To make a change only to the currently displayed Engine’s data without automatically propagating that change to all of the other Engines, uncheck the **Copy Changes to All** check box on the center-right of the window.

No changes are committed to the system until you press either **Apply** or **Apply and Close**.

Button	*Action*
Apply and Close	Applies all settings to all Engines and close the window.
Apply	Applies all settings to all Engines, but leaves the window open for further changes.
Cancel	Closes the window without applying any changes.
Copy Current to All	Copies the configuration of the currently displayed Engine to the editor buffer for every Engine you selected when you opened the dialog box. Note that the changes are not applied to any Engine until you click one of the **Apply** buttons. This button is not available if you are only editing one Engine.
Presets	Brings up a menu of actions related to the creation and application of previously configured Engine settings. You can save the currently displayed Engine's settings as a preset to apply, or you can choose one of the previously saved presets of settings to apply. Any settings loaded will be adjusted for every engine if

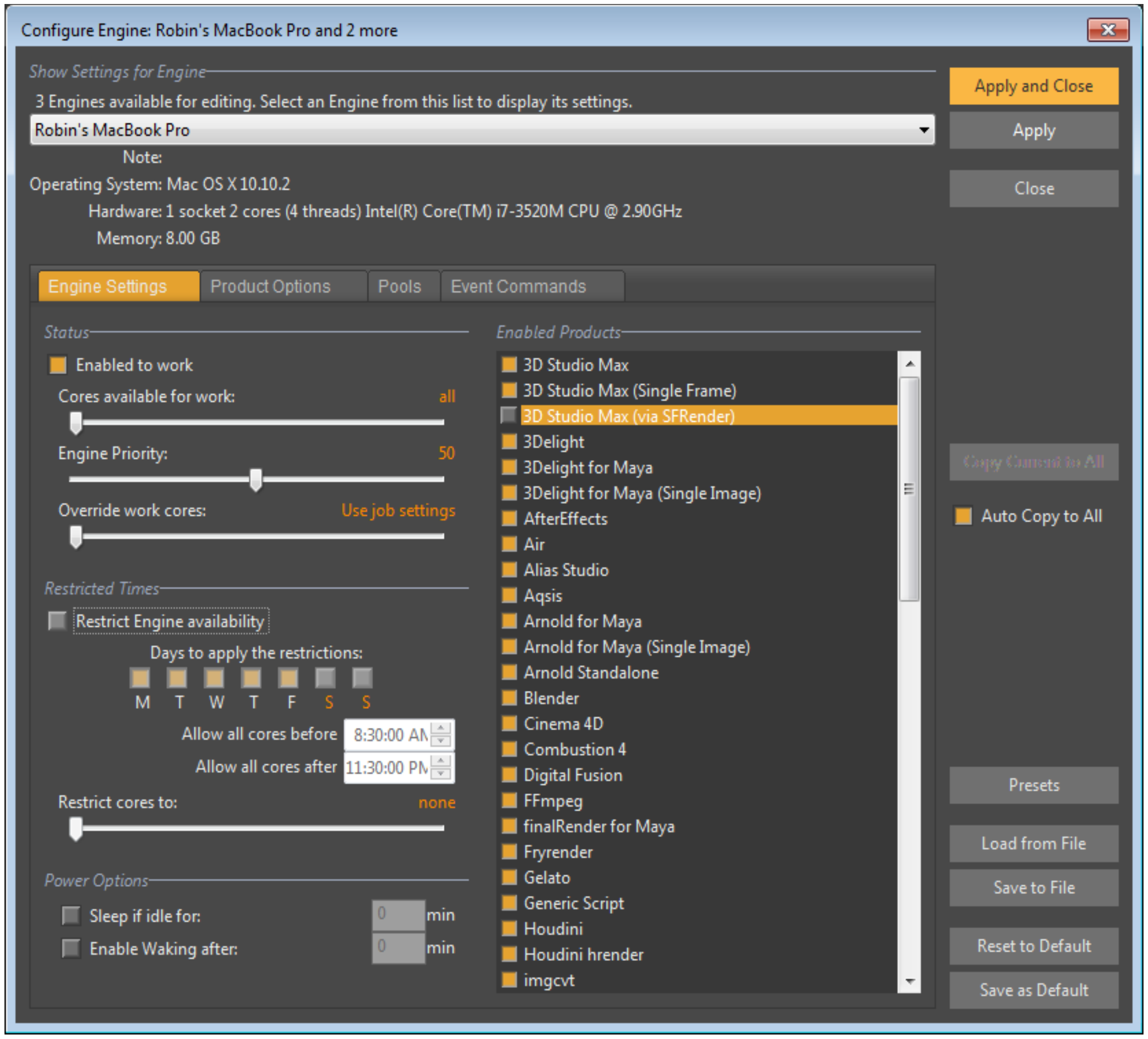
	the **Copy Changes to All** check box is checked, or only for the currently selected Engine if it is not. Presets are saved in the User's SmedgeGui preferences folder.
Load	Load Engine settings from an INI file. Any settings loaded will be adjusted for every engine if the **Copy Changes to All** check box is checked, or only for the currently selected Engine if it is not. If the INI file only changes some settings, the other settings will remain the same for each separate Engine.
Save	Save the currently displayed Engine's settings to an INI file for later loading.
Reset to Default	Loads all the settings from the current "default" file. Note that this file is saved for the current user on the current machine.
Save as Default	Save the currently displayed Engine's settings to a "default" file that will be loaded automatically when you press the **Reset to Default** button. Note that setting this default only changes this "default" file for the current user on the current machine. It specifically does not change the default Engine parameters that an Engine will use the very first time it starts up.

If you are editing a single Engine, the Engine's name will be displayed in the top section, but you will not be able to choose any other Engines to modify. Also, the **Copy Changes to All** check box will not be available. Otherwise, operation is identical.

The tabs classify the different types of information you want to view. You can automatically open the editor with any of your choice of tabs using the **Configure Engine Settings…**, **Configure Product Options…**, or **Configure Pools…** commands in the Engine Menu.

Engine Settings Tab

These are the basic operational settings for the Smedge's use of the Engine. In the ***Engine Details*** section, you can set some descriptive text about the machine you are configuring.



Configure Engine: Robin's MacBook Pro and 2 more

Show Settings for Engine
3 Engines available for editing. Select an Engine from this list to display its settings.
Robin's MacBook Pro
Note:
Operating System: Mac OS X 10.10.2
Hardware: 1 socket 2 cores (4 threads) Intel(R) Core(TM) i7-3520M CPU @ 2.90GHz
Memory: 8.00 GB

Engine Settings | Product Options | Pools | Event Commands

Status
Enabled to work
Cores available for work: all
Engine Priority: 50
Override work cores: Use job settings

Restricted Times
Restrict Engine availability
Days to apply the restrictions: M T W T F S S
Allow all cores before 8:30:00 AM
Allow all cores after 11:30:00 PM
Restrict cores to: none

Power Options
Sleep if idle for: 0 min
Enable Waking after: 0 min

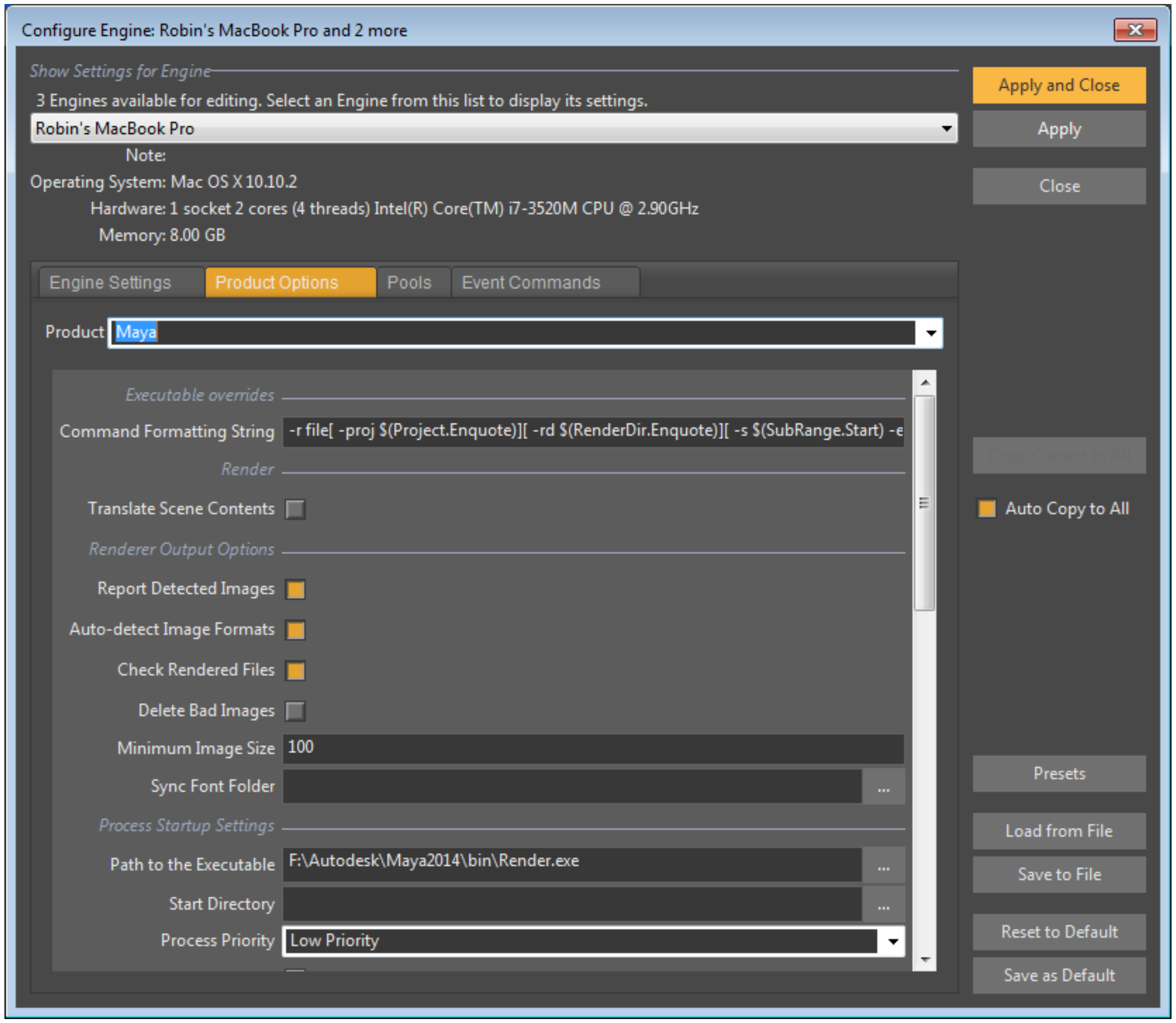
Enabled Products
3D Studio Max
3D Studio Max (Single Frame)
3D Studio Max (via SFRender)
3Delight
3Delight for Maya
3Delight for Maya (Single Image)
AfterEffects
Air
Alias Studio
Aqsis
Arnold for Maya
Arnold for Maya (Single Image)
Arnold Standalone
Blender
Cinema 4D
Combustion 4
Digital Fusion
FFmpeg
finalRender for Maya
Fryrender
Gelato
Generic Script
Houdini
Houdini hrender
imgcvt

The ***Status*** section controls the Engine's overall availability. The **Enabled to work** is a master switch that determines if work will be sent to the Engine or not. You can also configure how many cores will be made available for work sent to the Engine. Not every product will use every core, nor will every Product necessarily respect the value you set here. See Allowing Engines to Work for more information. The **Engine Priority** slider sets the relative priority this Engine has for getting work assigned from the Master. This priority value is used to determine the order that available Engines will be assigned work. If all other prioritizing factors are equal, Engines with a higher priority value will be assigned work before Engines with lower priority values. The **Override Work Cores** slider lets you specify if the Engine should override the value set in the Job's Processor core setting. You can force every job on the engine to be "one per engine", "one per core" or to use a custom number of cores.

The ***Restricted Times*** section allows you to set up a schedule for when this machine will be allowed to be used by Smedge for processing work. For example, if you have a workstation that you want to use to render at night, you can leave the SmedgeEngine service running on the machine all the time, and use this information to enable the machine to be used only during the hours you specify. Note that once work has started, Smedge will let it finish normally, even if that work is still going when the restriction kicks in. You can specify if you want to restrict some or all of the cores on the machine with the **Restrict cores to** slider. Restrictions will be ignored when the **Restrict Engine availability** check box is unchecked.

In the ***Power Options*** section, you can configure Smedge power management features. **Sleep if idle for** enables the ability for the Master to request the machine to put itself to sleep if it is not being used for rendering, after a certain number of minutes. The **Enable Waking after** option allows the Master to try to wake sleeping engines if more work is needed, but only if they have been asleep for a certain amount of time.

The ***Enabled Products*** tells Smedge which types of Jobs can be executed by this machine. All of the Products currently installed are listed. To allow the machine to work on Jobs of a given Product, make sure that the Product name is highlighted in the list. You can highlight multiple Products at one time using the selection modifier keys for your system (e.g., Shift and Control). See Allowing Engines to Work on Specific Products for more information.



Configure Engine: Robin's MacBook Pro and 2 more

Show Settings for Engine

3 Engines available for editing. Select an Engine from this list to display its settings.

Robin's MacBook Pro

Note:

Operating System: Mac OS X 10.10.2

Hardware: 1 socket 2 cores (4 threads) Intel(R) Core(TM) i7-3520M CPU @ 2.90GHz

Memory: 8.00 GB

Product: Maya

Executable overrides

Command Formatting String: -r file[-proj $(Project.Enquote)][-rd $(RenderDir.Enquote)][-s $(SubRange.Start) -e

Render

Translate Scene Contents ☐

Renderer Output Options

Report Detected Images ☑

Auto-detect Image Formats ☑

Check Rendered Files ☑

Delete Bad Images ☐

Minimum Image Size: 100

Sync Font Folder:

Process Startup Settings

Path to the Executable: F:\Autodesk\Maya2014\bin\Render.exe

Start Directory:

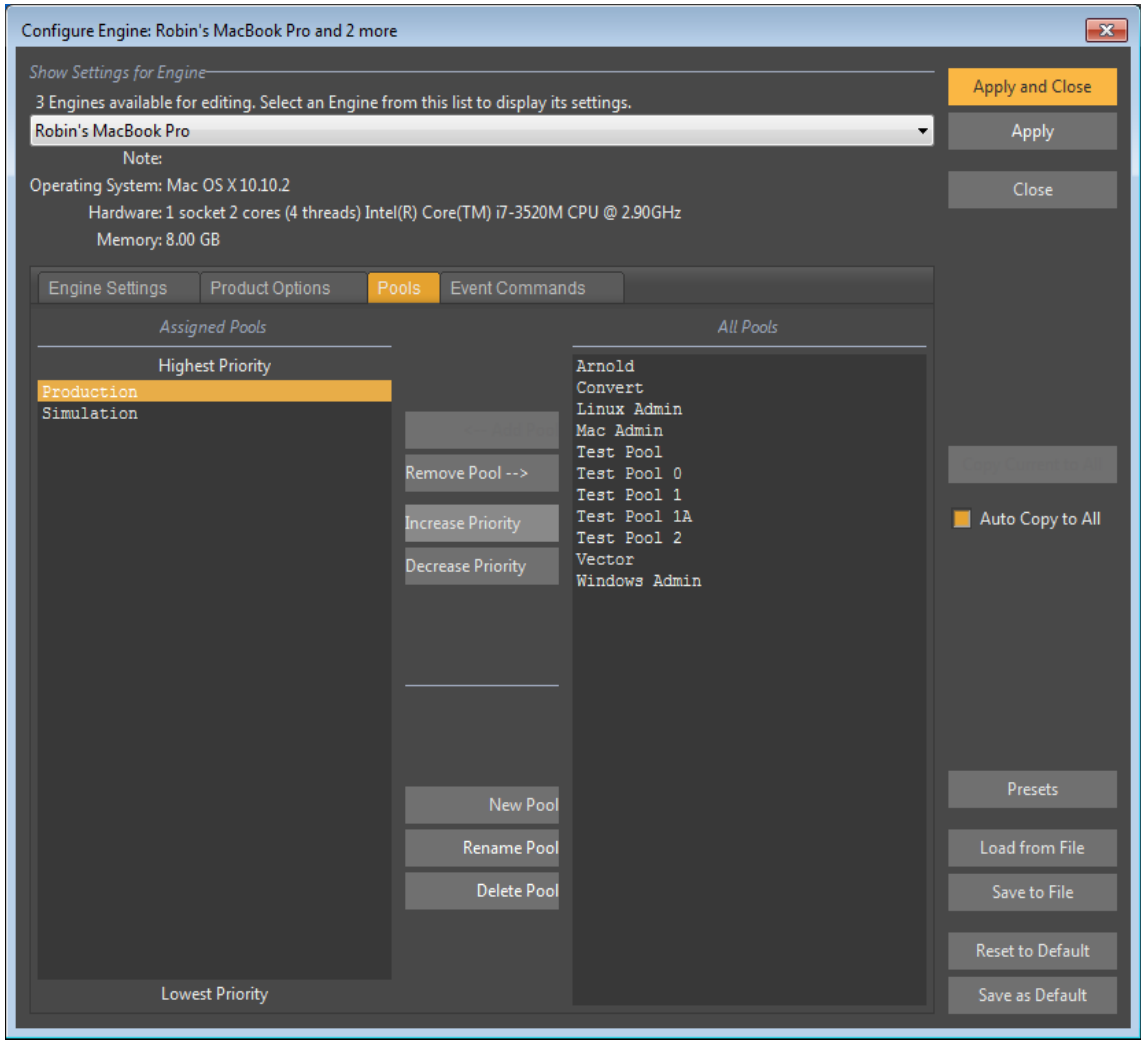
Process Priority: Low Priority

☑ Auto Copy to All

Product Options Tab

Every Product can have options that are available to configure details about how the Engine will perform work for that Product. The values displayed are for the currently selected Engine, and you can use the button with the red triangles to transfer the displayed settings to every other Engine. See Setting Engine Options for Products for more information about how Engines use these options.

The list of displayed options here will depend on the selected Product. You can get more information about many of these options in the Common Custom Job Parameters chapter, and in the Administrator Manual, or from the documentation that may have come with any third party Modules.


Configure Engine: Robin's MacBook Pro and 2 more

Show Settings for Engine

3 Engines available for editing. Select an Engine from this list to display its settings.

Robin's MacBook Pro

Note:

Operating System: Mac OS X 10.10.2

Hardware: 1 socket 2 cores (4 threads) Intel(R) Core(TM) i7-3520M CPU @ 2.90GHz

Memory: 8.00 GB

Assigned Pools

Highest Priority

```
Production
Simulation
```

Lowest Priority

All Pools

```
Arnold
Convert
Linux Admin
Mac Admin
Test Pool
Test Pool 0
Test Pool 1
Test Pool 1A
Test Pool 2
Vector
Windows Admin
```

Auto Copy to All

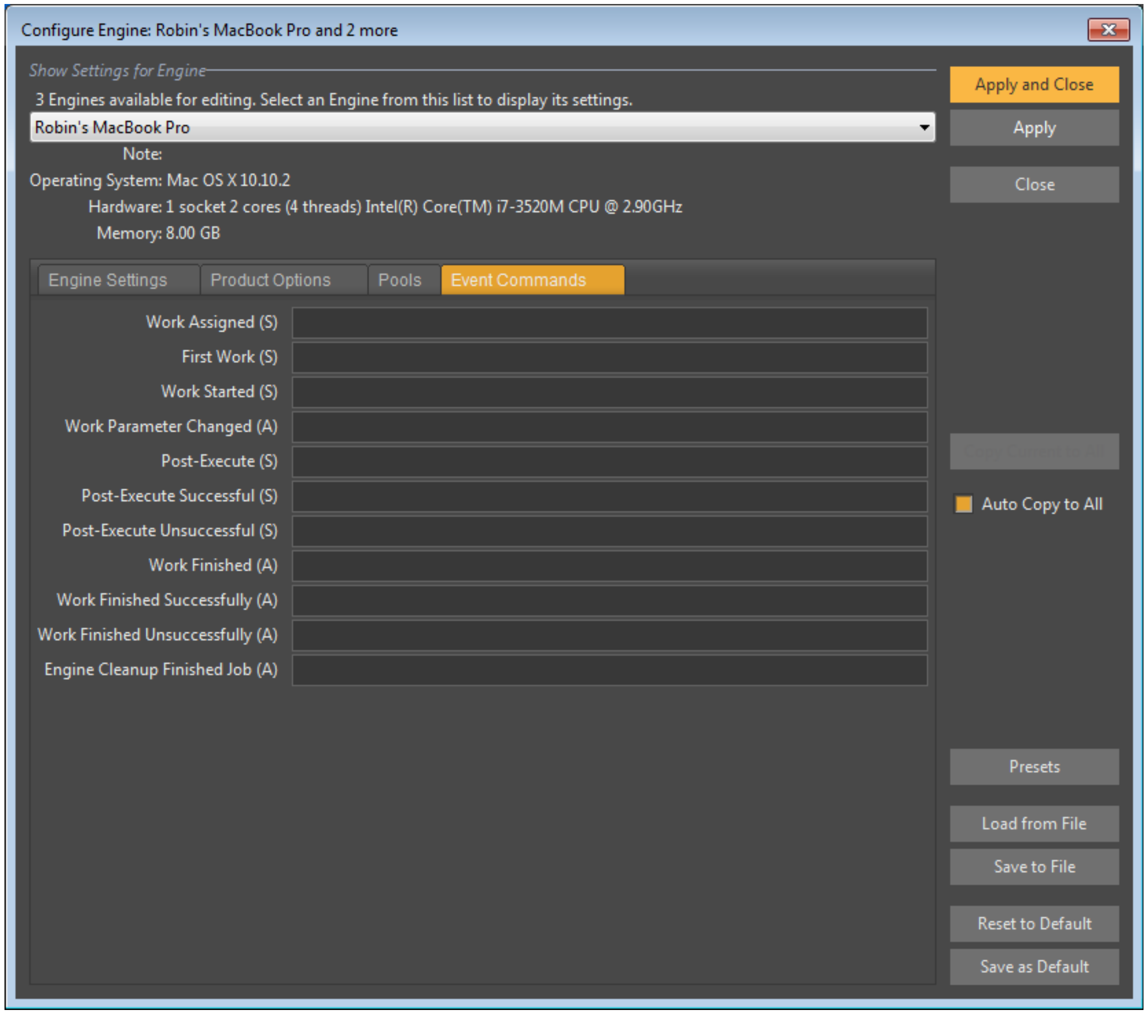
Configure Pools

This tab shows the available Pools that you can assign to Engines, and the currently selected Engine's prioritized list of Pools. In between the two lists of pools are several buttons that provide the operations you may want to perform. For more information about Pools, see the section on Pools and the section Assigning Engines to Pools.

The left side list is an ordered list of the Pools that the currently displayed Engine is a member of. The list is ordered with the highest priority Pool at the top and the lowest priority at the bottom. The right side list is a list of all the Pools in the system that the current Engine is not a member of. You can move pools from one list to the other with the ← **Add Pool** and **Remove Pool** → buttons. You can also change the relative priorities of the pools on the left with the **Increase Priority** and **Decrease Priority** buttons.

The bottom three buttons allow you to create new Pools, or rename or delete existing Pools. Unlike all other operations, the effects of using these buttons are immediate system-wide. Specifically, if you delete a Pool here, that Pool will be removed from the system and from the Engine information for every currently online Smedge component application. When you create a new Pool, that Pool will not be assigned to the Engine(s), and will only appear in the list of all Pools until you use one of the Add button commands.

If you have set a default or try to load an Engine file that contains Pool IDs not found in the system, those IDs will be ignored by the editor. The load process will never create missing Pools for you. If you cancel this editor, then all changes you make to pool assignment and ordering will be discarded. However, if you created, renamed, or deleted pools, those effects have already been processed and will not be undone.


Configure Engine: Robin's MacBook Pro and 2 more

Show Settings for Engine

3 Engines available for editing. Select an Engine from this list to display its settings.

Robin's MacBook Pro

Note:

Operating System: Mac OS X 10.10.2

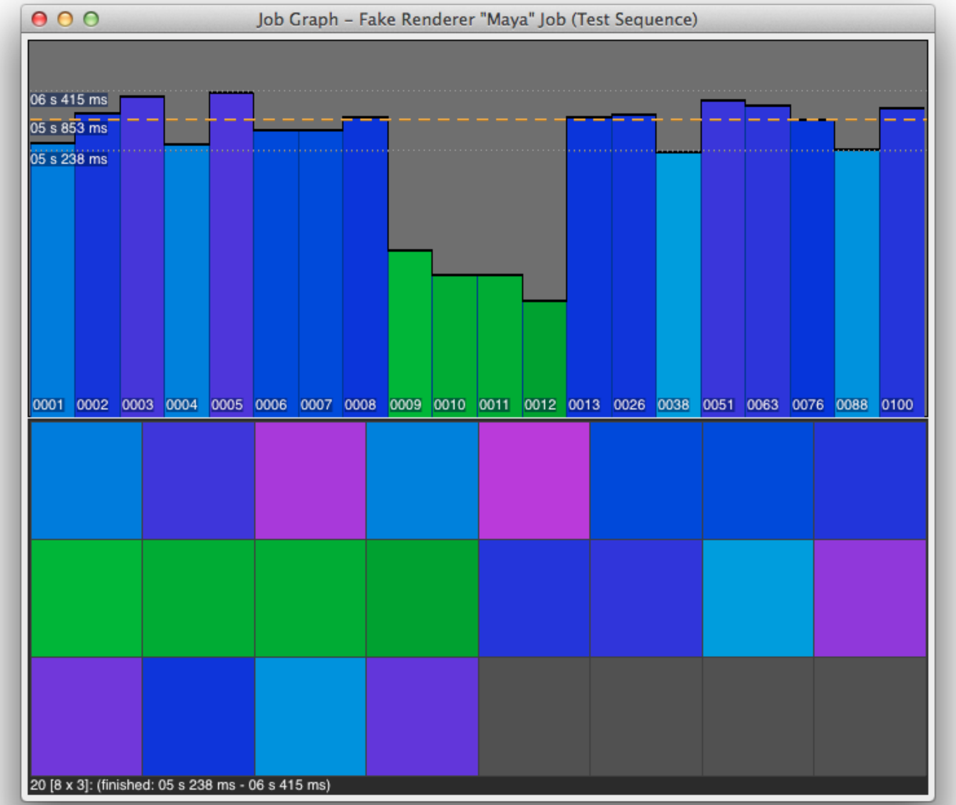
Hardware: 1 socket 2 cores (4 threads) Intel(R) Core(TM) i7-3520M CPU @ 2.90GHz

Memory: 8.00 GB

Event Commands

This tab allows you to configure global event commands that are run on the Engine at specific points during the work life-cycle. As with the Job Event commands, these commands will have the job data substituted into the command line you specify before processing using the $(Name) variable substitution system. See the Administrator Manual for more information about variable substitution and the parameter names that are available for each type of Job.

The commands marked with an "S" are executed synchronously with the normal work processing. A non-zero result code from the command will trigger a work failure. The "A" commands are executed asynchronously with the work, and the result is ignored by Smedge.



Job Graph

The Job Graph window shows two displays of your job's history to help you see more about what is going on with your job. The two graphs are the Histogram and the Heat Grid. They both can show the same types of data about your job, one of: the total time spent on each work unit, the amount of time spent on only the latest run of each work unit, or the number of tries for each work unit. The hist

You can select which display is shown for each graph by right clicking on the graph and choosing the appropriate menu option. Note that you can set each graph independently, and that your choice of what to display is remembered for each type of job you select. You can also hide or show each graph independently based on your preference. The histogram can be graph the time on a linear or logarithmic scale.

The histogram plots the work units on the X axis and the time or tries spent on the Y axis. It will also show you the shortest, longest, and average completion times or the maximum number of tries as a dashed line. The Heat Grid shows each work unit as a cell in a grid based on keeps the aspect ratio of the display.

Each work element is rendered with colors to help you see what is going on. Complete work will range from cyan for the shortest to blue for the average to purple for the longest. Running work ranges from green to yellow to white, and work that is pending but not active will show as dark red to red.

If there are more work elements than fit into a specific work item, the graph will average out the values for all of them. This gets a more accurate general picture but sacrifices details. You can scale the graph window bigger to try to see more detail.

From the Job Graph, you can also open up the Job History view for the same job, to get more detailed information about the work and the job statistics.

The Job Graph is also available as an Info Panel if you have the Info Pane visible and select a job or a work unit form a job.

Job History

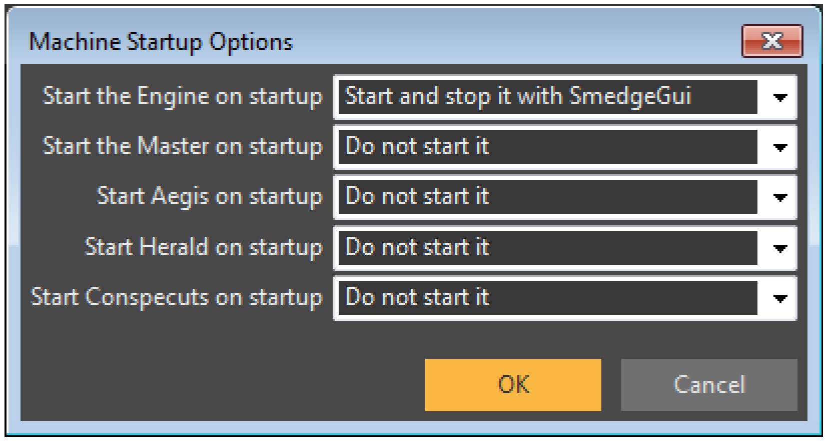
This window displays the Job history information. The top fields display some basic Job status information and estimated completion time (if the work is still going). More information about the Job history is available in the “Show More” section, with extra details about the elapsed and estimated times for the Job.

The list below shows the complete history of every work unit. In the closed position, just the final information (from the most recent attempt) is displayed. If you click on the Plus sign next to the item, you will expose the complete history of that packet, including all run attempts and every change in the work or in the work status for each attempt.

The History element list has a context menu that you can access by right clicking. From this menu, you can access most Job commands, and you can also open up the Job Graph window for a more visual overview of this information.

The menu also gives you options to switch if you want the top level history for a work unit to show you the total amount of time spent on that work for all tries, or only the amount of time for the current or latest try.

The Job History is also available as an Info Panel if you have the Info Pane visible, and select a Job or a work unit from a Job.


Machine Startup Options

Start the Engine on startup	Start and stop it with SmedgeGui
Start the Master on startup	Do not start it
Start Aegis on startup	Do not start it
Start Herald on startup	Do not start it
Start Conspecuts on startup	Do not start it

Machine Startup Options

This window allows you to configure which components will start with the GUI by default. When the GUI connects to a Master, it will ensure that components that you have requested to start are started, and any you have requested not to be started will be shut down, as needed.

For each component you have three possible options:

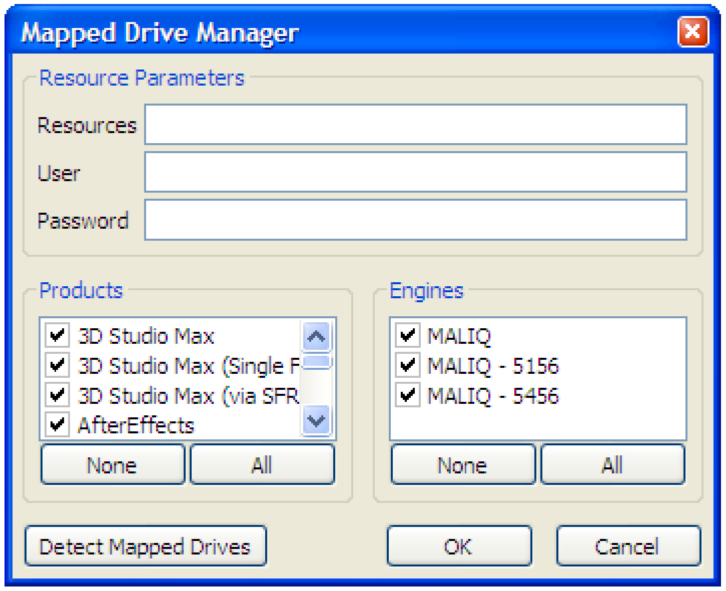
Do not start it	The component is not started with the GUI
Start and Stop it with SmedgeGui	The component is started with the GUI and stopped when you close the GUI
Start it to run until you log out	The component is started with the GUI but allowed to run until you shutdown, log out or explicitly kill the component process.

The options you set here are stored on the Master, and sent to each node when it connects. The GUI will then conform itself to the requested component process disposition. Note, however, that if the GUI starts the Master component and that instance of the Master is the active primary Master controlling the system (visible in the About dialog box), then it will ignore any requests not to start the Master component.

The GUI will remember the component start up details from the last time it connects. If you change the settings while the GUI is running on any machines, all machines will re-conform to the new requested settings, unless you have overridden which components to start on a specific machine in the SmedgeGui options dialog box.

The SmedgeGui options and the command line will always override the default settings from the Master.

Mapped Drive Wizard



✓ 3D Studio Max
✓ 3D Studio Max (Single F
✓ 3D Studio Max (via SFR
✓ AfterEffects

✓ MALIQ
✓ MALIQ - 5156
✓ MALIQ - 5456

The Mapped Drive Manager provides a convenient system to configure multiple mapped network drives for multiple products on multiple Engines all with just a few mouse clicks. Normally, Smedge can detect if your scene file is on a mapped network drive, and will automatically ensure that this drive letter is mapped. However, if your scene file references other files that are not on the same mapped drive, Smedge won't be able to detect the required mapped drive before the work is started, and you may get Work errors about missing files.

This dialog is a shortcut for configuring the Engine Product Options for every possible Product on every currently connected Engine. By setting the resources here, you can save yourself the time it takes to configure each Product manually. Any values you set here will override the values set for the Engine.

Resource Parameters. Fill in these fields with the resource remapping information. You can provide the **Resources**, **User** account, and **Password.** The format of the Resources field is the same as the **Resources** field in the Engine Product Options dialog. See Making sure the process has resources available for more information. The format is:

Drive-letter`: = \\`*Server*`\`*SharedVolume* [`;` *Drive-letter*`: = \\`*Server*`\`*SharedVolume*...]

For example, if you have drive V mapped to the shared volume "Vault" on the File Server "Big_Fun", and W mapped to the shared volume "Work" on the File Server "Server", you would fill in the resources like this:

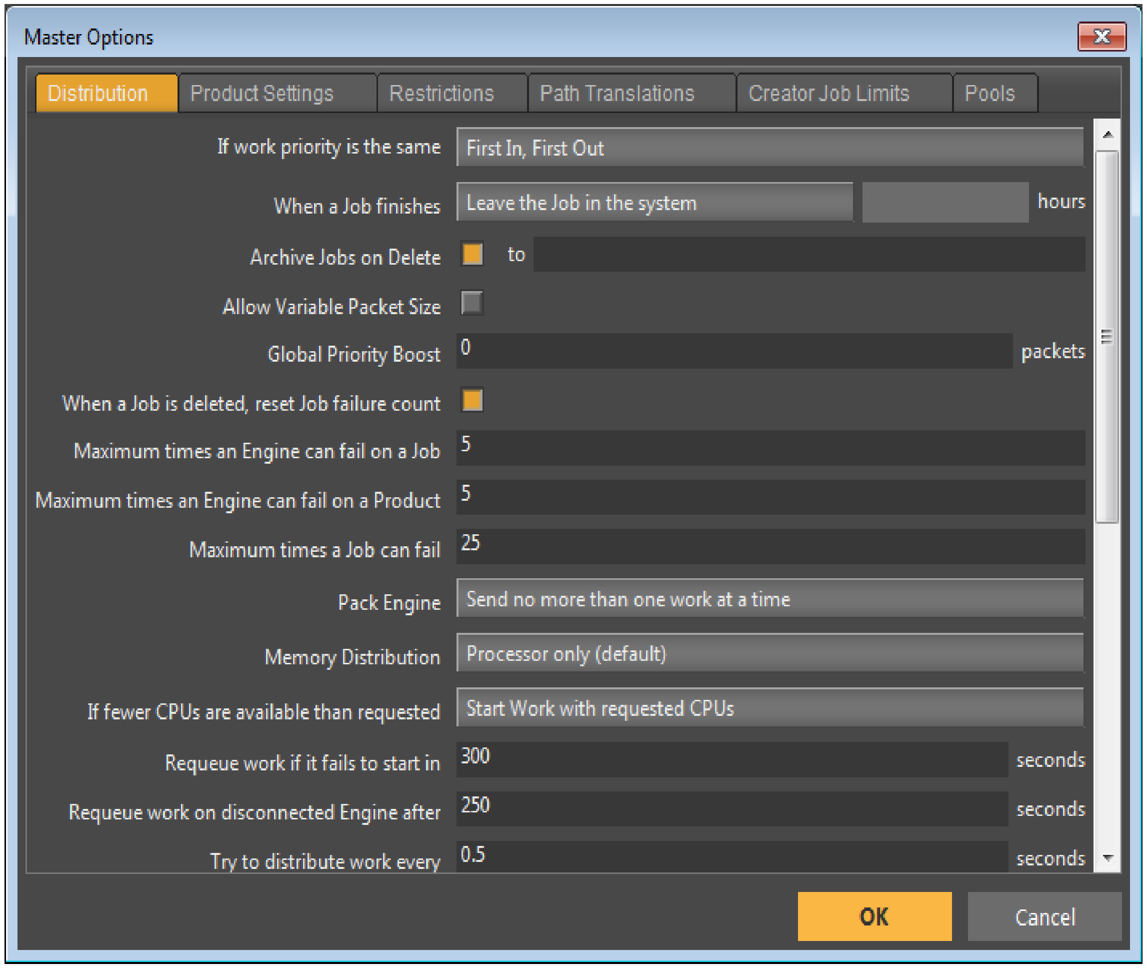
```
V: = \\Big_Fun\Vault ; W: = \\Server\Work
```

You can also provide the account name and password that are used to connect to the file server, if needed. (Note that any extra whitespace around the drive mapping information is ignored.)

Products. Specify which Products you want to configure the mapped drives for. Smedge can maintain different drives to remap for different Products. However, every product you check here will have the resources options set at one time. The settings in the **Resources** section will be set for every Product you have checked in this list. You can use the **None** and **All** buttons to quickly check or uncheck all Products in the list.

Engines. Specify which Engines you want to configure the mapped drives for. The settings in the **Resources** section will be set for every Product selected in the **Products** section for every Engine that is checked in this list. If an Engine is unchecked, no settings will be changed for that Engine. You can use the **None** and **All** buttons to quickly check or uncheck all the Engines in the list.

Detect Mapped Drives (only available on Windows). This button will examine the local machine for all the drives mapped, and generate a **Resources** string that will allow Smedge to map those drives.



Master Options

If work priority is the same	First In, First Out	
When a Job finishes	Leave the Job in the system	hours
Archive Jobs on Delete	✓ to	
Allow Variable Packet Size	☐	
Global Priority Boost	0	packets
When a Job is deleted, reset Job failure count	✓	
Maximum times an Engine can fail on a Job	5	
Maximum times an Engine can fail on a Product	5	
Maximum times a Job can fail	25	
Pack Engine	Send no more than one work at a time	
Memory Distribution	Processor only (default)	
If fewer CPUs are available than requested	Start Work with requested CPUs	
Requeue work if it fails to start in	300	seconds
Requeue work on disconnected Engine after	250	seconds
Try to distribute work every	0.5	seconds

Master Options

The Master Options dialog box allows you to configure the Master. There are three tabs which contain different types of options you can configure for the Master. You must be in Administrator Mode to access the Master Options dialog box.

Distribution Tab

This tab allows you to configure the details of how the Master will dispatch work.

If work priority is the same. Controls what happens when you have more than one Job with the same priority. Your choices are:

First In, First Out	The Job submitted first will get dispatched first.
Round Robin	All Jobs will get dispatched evenly.

When a Job finishes. Controls what happens when a Job finishes. Your choices are:

Leave the Job in the System	The Job and its history will stay in the system until a user deletes it using one of the Shells.
Delete the Job Immediately	As soon as the Job finishes, remove it and its history from the system.
Delete the Job After	Specify how many hours after the Job has finished for it to be automatically deleted.

Archive Jobs On Delete. If checked, the job and its history will be preserved in a compressed archive format after it has been deleted. You can use the Archive Job Manager to list, restore or remove these archived deleted jobs. By default, the archive folder will be the same folder as the live job data. To archive to a different folder, specify that folder in the field to the right of the checkbox.

Allow Variable Packet Size. If checked, packets are distributed to each machine depending on their assigned priority. The priority value, ranging from 1-100, is used here as a percentage modifier (from 1%-100%) to the PacketSize parameter defined by the sequence distributor. An engine with a priority of 100 will be given a packet of full 100% size, equivalent to what it would have been given if this item was not checked. An engine with a priority of 1 will be given a packet of minimal 1% size, and will be given a packet size far less than an engine with priority 100.

Global Priority Boost. Use this value to set the number of work units from any given job to boost in priority. This many work units will get run before any other jobs in the queue, allowing you to see a sample of work to ensure the job is reliable before waiting for it to render fully. This is the master priority boost level applied to all jobs. If you set a boost level in the Product Settings tab, that value will override this one. If you set a priority boost when you submit the job, the Job's value will override both the Product Settings value and this value.

When a Job is deleted, reset Job failure count. If checked, when a job is deleted, the job failure count belonging to that Job ID is automatically reset, thus allowing machines/engines that are blocked due to surpassing their job failure limits to be brought back to a working state. Otherwise, deleting a job will not reduce an engine's failure count belonging to that job, and will remain blocked until the job failure count is manually reset.

Maximum times an Engine can fail on a Job. This is the maximum number of times that Smedge will allow an Engine to fail on a single Job. Once an Engine has failed this many times on a single Job, that Engine will no longer attempt to execute work from that Job until a user resets the failure counts for the Engine or Job. See the sections on Resetting Job Failure Counts and Resetting Engine Failure Counts.

Maximum times an Engine can fail on a Product. Once an Engine has failed on the specified number of Jobs of the same Product, that Engine will no longer be allowed to work on Jobs from that Product. The product will automatically be deselected from the list of supported applications. To re-enable it, you will have to use the Engine Settings tab from the Configure Engines Window box.

Maximum Times a Job can Fail. This is the maximum number of times that Smedge will allow all combined Engines to fail on the same Job. Once a job has failed this many times, the job will no longer attempt to execute until a user requests to reset the failure count. See the sections on Resetting Job Failure Counts.

Pack Engine. Smedge 2018 and later allow selecting whether the distribution algorithm will fill up an engine with as many workers as it can take before moving on to the next one (the default behavior for Smedge 2016 and earlier). The choices are:

Send no more than one work at a time	If an engine gets assigned work, the dispatch algorithm will move on to the next engine for that iteration, for a more round-robin distribution of work to different nodes. If a node has more resources available and work is pending, it will get assigned on the next dispatch iteration. This is currently the default.
Send as many workers as the Engine can take	Enables packing as many workers on to the engine as it can, the way Smedge 2016 and earlier worked.

Memory Distribution. By default, Smedge distributes work using the cores on the machine. You can also configure it to distribute work based on the physical memory installed on the machine. This setting determines which criteria are used for the Job:

Processor only (default)	Only use the CPUs setting from the Job. The Engine's RAM is not considered. The work CPU parameter will have the number of CPUs assigned to the worker. The RAM parameter will be 0.
Memory only	Only use the RAM setting from the Job. The CPU count is not considered. The work CPU parameter will have the number of CPUs on the machine, and the RAM parameter will be the assigned RAM for the work.
Both Processor and Memory	Consider both the CPU count and RAM count from the Job. Note that either system can still be disabled on a Job basis by setting the appropriate value for the Job itself. Both resources are used to distribute work, and the parameter values for the work will be the values assigned to the work.

If fewer CPUs are available than requested. If your Engines have more than one CPU and you have Jobs with different settings for the number of CPUs to use, it is possible to have a situation where the next Job that is available for an Engine may be requesting to use more CPUs than the Engine currently has available (because other CPUs may be occupied with other work). Your choices are:

Do not start Work	Work from this Job will not be allowed to start, and the Master will find the next Job available for this machine to work on, or skip this machine if no more Jobs are available for it.
Start Work with Available CPUs	Work from this Job will be allowed to start, but the Work's CPUs value will be set to the number of CPUs that the Engine is reporting as available, which may be less than the number of CPUs that the Job is asking for.
Start Work with Requested CPUs	Work from this Job will be allowed to start, and the Work's CPUs value will be set to the number of requested CPUs that the Job specifies. This may be more than the number of CPUs that the Engine is reporting as available.

Some Products can use the CPUs setting to control the number of threads that the Work will use. For other Products, the CPUs value cannot be controlled, so it is used to allow Smedge to know how many threads that the Work will be using. Also, no matter what setting you have chosen here, Jobs that have their requested CPUs value set to "One per Engine" (which is a value of 0) will only ever allow a single Work unit to start on an Engine at one time.

Requeue work if it fails to start in. When the Master requests an Engine to start a work unit, if that Engine does not report back within this timeout period, the Master assumes that something went wrong, and re-queues the work for another Engine. If you find that workers are getting killed this way under load, you can try increasing this timeout. Alternatively, you can set it to 0 to disable this test.

Requeue work on disconnected Engine after. If an Engine that is currently working loses its connection to the Master, this is the timeout period for that work. If the Engine does not connect to the Master within this timeout period, the Work is considered "Lost" and will be re-queued for another Engine.

Try to distribute every. This is the main distribution loop period. Smaller values will improve the distribution speed, and decrease unused Engine time, but will also increase the amount of processor time that the Master process will use.

Output File Cleanup. By default, the Engine stores captured output locally in its log folder. To keep this data from slowly consuming the entire disk, old captured output is cleaned up after this many days. The default is 30 days. Note that if an Engine is not connected to the Master for an extended period of time, the value from the last Master will be used. If the Engine has never connected to a Master, it will use 30 days. If you override the value with the SmedgeEngine command line parameter, that value will be used regardless of whatever option is set on the Master.

Allow 'Whole System' Pool. If checked, all Engines are automatically included in a Pool called "Whole System." This is the lowest priority Pool for every machine. If unchecked, this automatic Pool is not available. Only Jobs assigned to one of the Pools specified in the Pools tab of the Configure Engines dialog for each Engine will be allowed to execute on that machine. In this case, the "Whole System" Pool ID corresponds to having no Pool, and any Jobs assigned to this null pool will never have work distributed.

Priority 0 is paused. If unchecked, setting a Job's priority value to 0 is just the lowest possible numeric priority. If checked, then any Job with a priority value of 0 will not have work distributed. Note that the "Paused" flag is always respected, no matter what the Job's numeric priority may be.

Pool Priority. This determines how the pools are prioritized. Options are:

Allow Engines to prioritize their pools

Always use the Job priority only, ignoring pool priority.

Master resets Job creation time. If checked, when a Job is created in the system, the Master will always set the Job's creation time using its own system clock. If unchecked, then the creation time that is specified in the Job by the Shell that is submitting it will be saved. That time may be based on a different system clock than the Master's, which can cause confusion if the clocks are not in sync.

Global Stagger Start. Allows you to specify that the Master will only start a given number of Jobs during a given period of time. This stagger start feature works for all Jobs in the system, no matter which Product they may be.

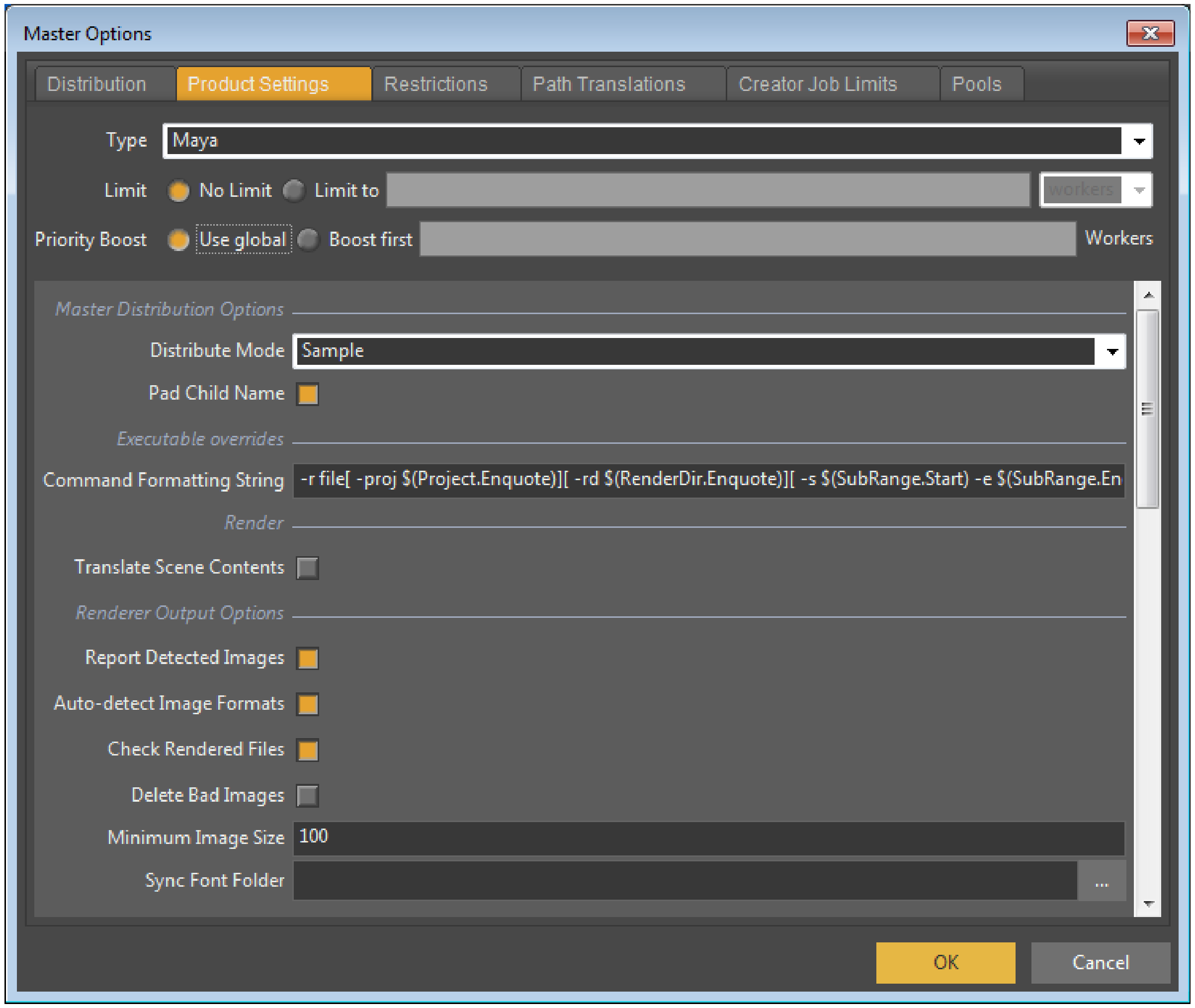
Wake-On-Lan Thread. Allows you to specify that the Master will only wake a given number of Engines during a given period of time, during the Wake-On-Lan feature for waking idle engines on new job submissions when there are not currently enough available engines. Note that if you disable the Wake-On-LAN thread, no engines will be woken, despite any settings you configure for the Engines.

Allow Mirrors. When this is checked, this Master will allow connections from other Masters on the same network that can work as redundant backup Masters called Mirrors. If disabled, then any Master that tries to connect will be denied and terminated on the remote machine. Use this if you have a problem with Mirrors taking over unexpectedly and altering the job queue.

Except Mirrors. This can manually add specific machines to be allowed to connect as Mirrors even if the general **Allow Mirrors** option is disabled. This way you can specify a group of redundant Masters for your network, and keep all other machines from ever accidentally taking over as the Master. This can be useful if you use SmedgeGui on a laptop that accesses more than one network.

Default Creator String. This determines how the "Creator" field should be implemented by shells by default. The choices are:

Current User Name	The name of the currently logged on user
Local Machine Name	The name of the computer
User@Machine	Both



Type: Maya

Limit: [✓ No Limit] Limit to ___ workers

Priority Boost: [✓ Use global] Boost first ___ Workers

Master Distribution Options

Distribute Mode: Sample

Pad Child Name: ✓

Executable overrides

Command Formatting String: -r file[-proj $(Project.Enquote)][-rd $(RenderDir.Enquote)][-s $(SubRange.Start) -e $(SubRange.En

Render

Translate Scene Contents: ☐

Renderer Output Options

Report Detected Images: ✓

Auto-detect Image Formats: ✓

Check Rendered Files: ✓

Delete Bad Images: ☐

Minimum Image Size: 100

Sync Font Folder:

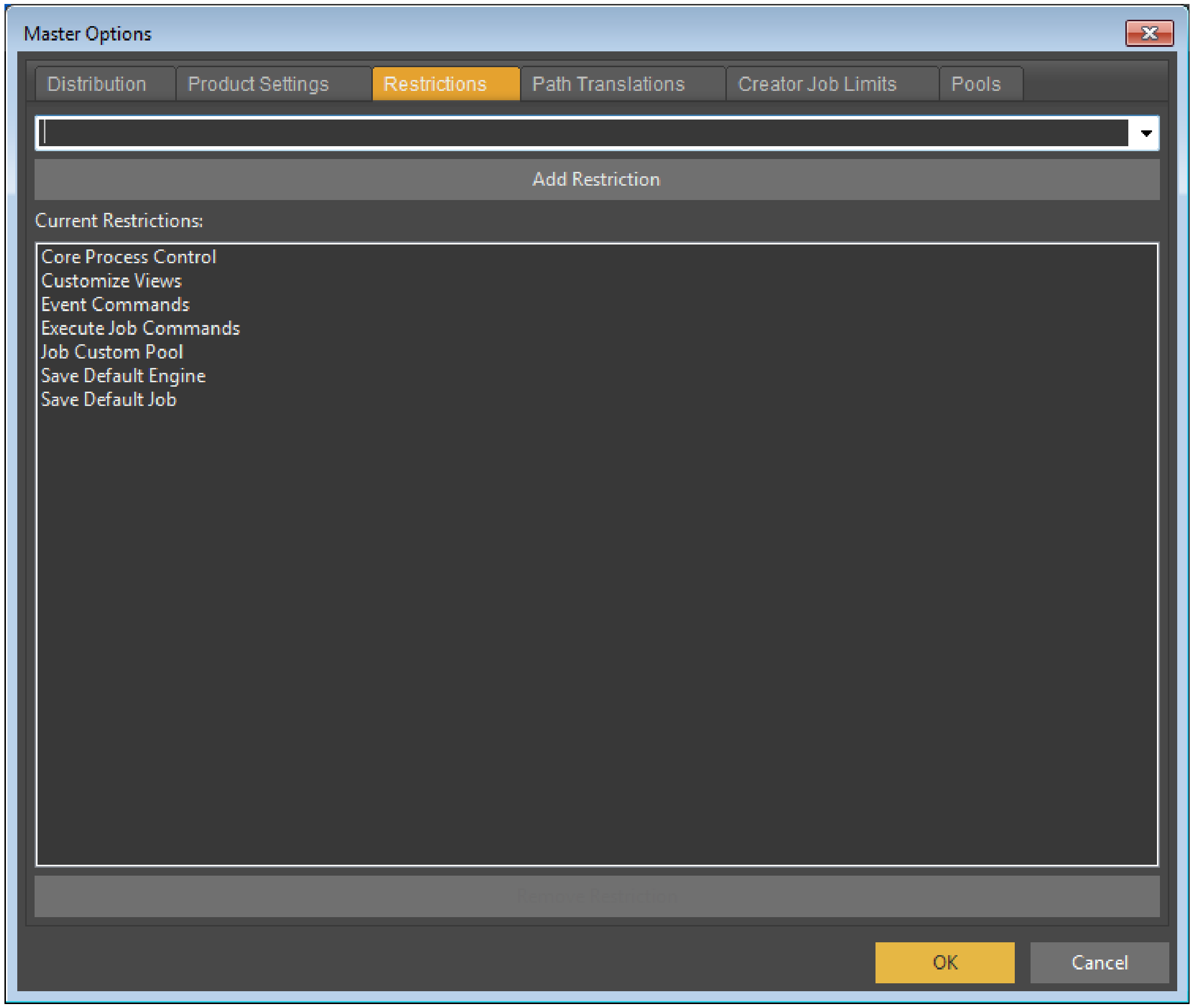
Product Settings Tab

This tab allows you to configure the options specific to the Products available on your system. The tab shows only the options for the currently selected product in the **Type** field.

The **Limit** field sets a global limit on this Product. No more workers of this Product type will be sent than the limit you provide here. If you set it to **No Limit**, then there will be no limit to the number of workers from a particular Product that can be working at the same time.

Any custom Master options for the Products will appear in the **Product Specific Options** section. Because custom Modules can include any number of custom parameters, we cannot describe all possible values you may see here. See the Common Custom Job Parameters chapter, or ask your system administrator for more information.

The current Engine Product Defaults are also displayed and editable here. These are the default values used by the Engines if any specific parameter has not been customized for that Engine. They are listed after the Master specific options, and can also be set using the Save Default button on the Configure Engine dialog box.

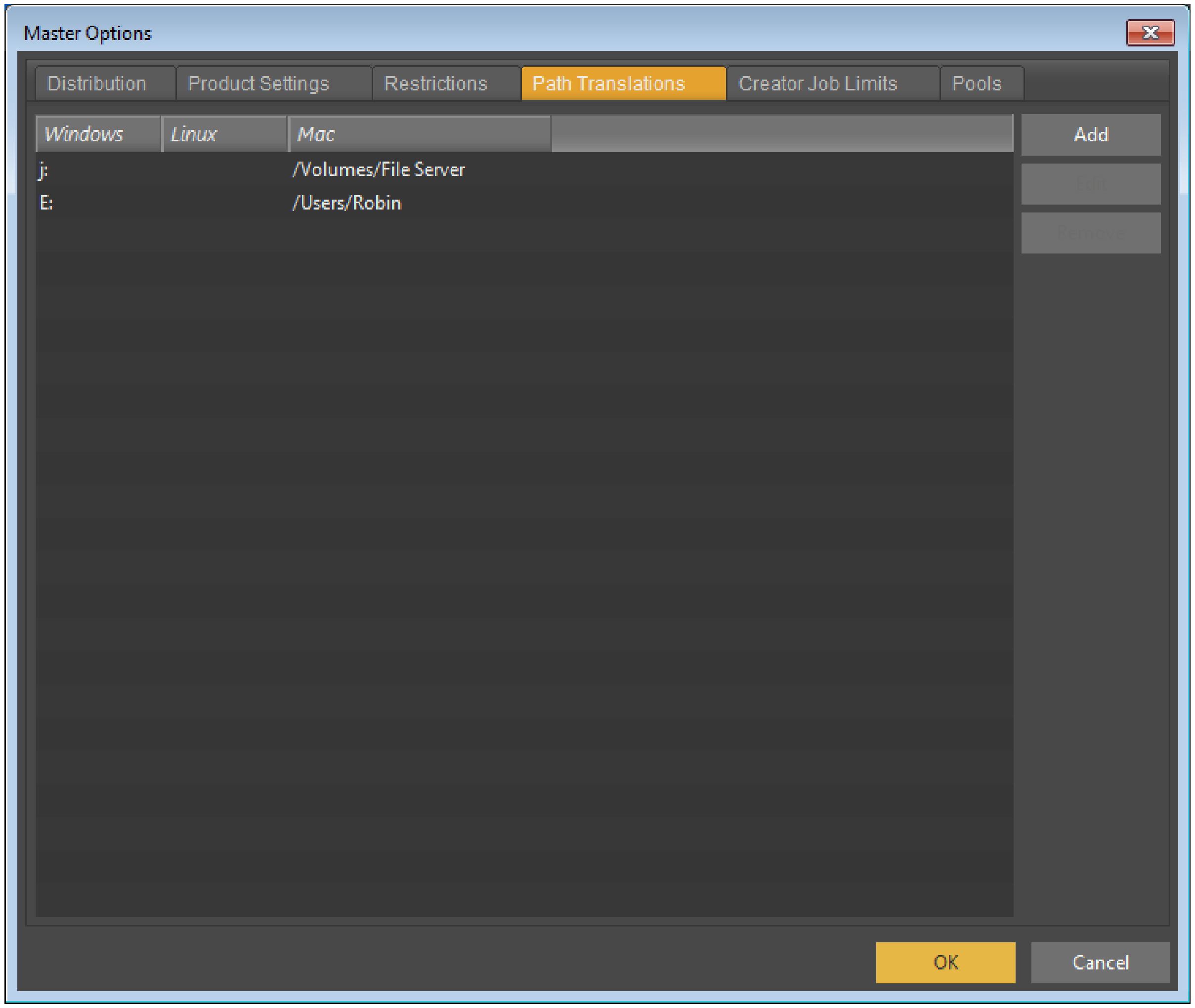


Core Process Control
Customize Views
Event Commands
Execute Job Commands
Job Custom Pool
Save Default Engine
Save Default Job

Restrictions Tab

This tab allows you to configure the Restrictions on SmedgeGui operation. You can type a restriction name, or select one from the drop-down list of predefined names, then click the **Add Restriction** button to add that restriction. To remove restrictions, select the restriction from the list of **CurrentRestrictions**, then click the **Remove Restriction** button. See Restricting What Users Can Do for more information.

To see a full list of the restrictions available by default, see the "Restrictions" chapter in the Administrator Manual. You can also type custom restrictions that may be implemented by Modules here.



Windows	Linux	Mac
j:		/Volumes/File Server
E:		/Users/Robin

Path Translations Tab

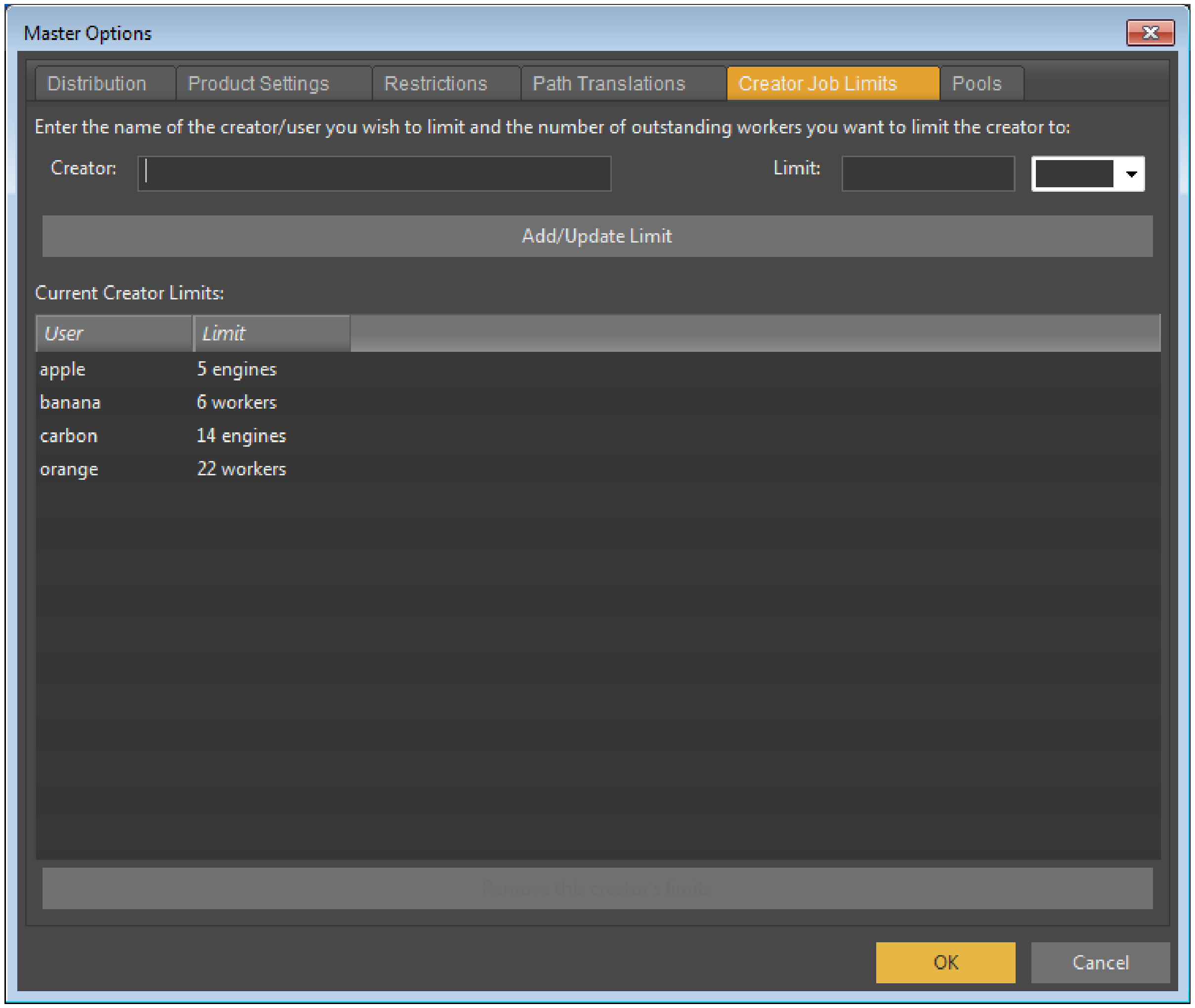
Here is where you can set up the automatic path translation system for cross-platform use. When you specify a file or directory name on one operating system, other operating systems may need a different name to find the same file or folder. Once you have told Smedge how to translate these paths, all path data is automatically translated between platforms as the work is performed.

Each translation set contains a path "root" (the first part of the file or path name) for each of the platforms on which Smedge runs The root of a translation is usually the mount point of the storage device. On Windows, this is either the UNC share name or the drive letter. On other systems, this is the folder where the shared disk is mounted.

The list shows the current translations in your system. Use the **Add** button to add a new translation. Set the unique root part of your filenames for each OS. Any part after the root will be left alone during the translation process. You can edit an existing translation by selecting it in the list and clicking the **Edit** button. To remove a translation, select it from the list and click the **Remove** button.

If you don't actually make use of every platform, you can leave the unneeded platform's field blank. You can add as many path translations as you wish. You can have multiple translations for the same path. For example, both a drive letter and a UNC path on Windows can map to the same mount point on a Linux or Mac file system.

Translations are searched for in the order they appear (top to bottom). In the example here, a Mac path that starts with "/Volumes/Work" will get translated to the drive letter "W:" on windows, because that translation appears before the UNC root "\\Server\Work" in the list of translations. Paths from Windows that start with either the drive letter "W:" or the UNC root "\\Server\Work" will get translated to the same path on Mac ("/Volumes/Work") or Linux ("/mnt/Work").



Enter the name of the creator/user you wish to limit and the number of outstanding workers you want to limit the creator to:

Creator:

Limit:

Current Creator Limits:

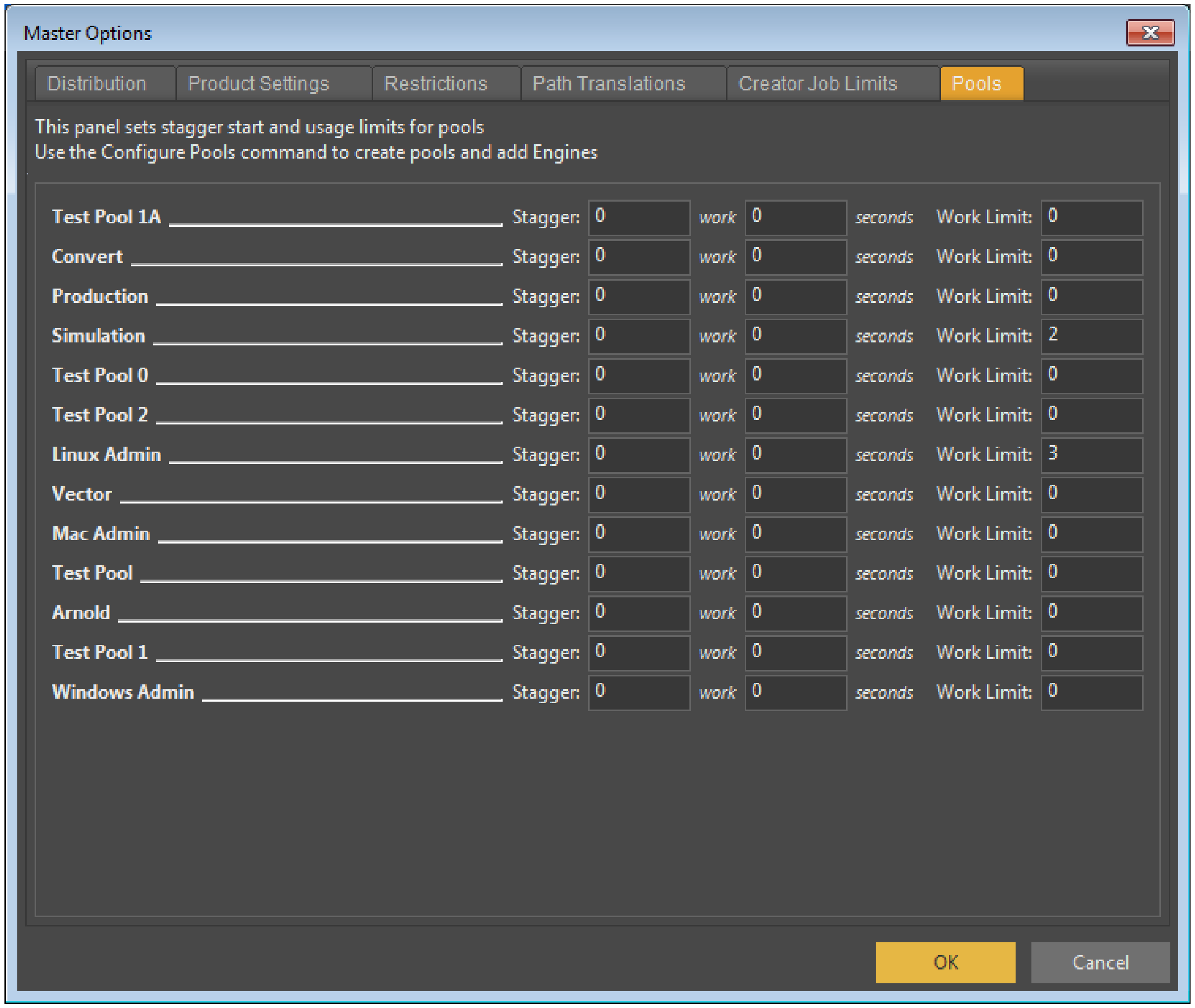
User	Limit
apple	5 engines
banana	6 workers
carbon	14 engines
orange	22 workers

Creator Job Limits Tab

This tab allows you to configure the limits on workers assigned to specified creator / users. Type the name of the user you wish to limit in the top left text control box, and enter the number of workers you wish to limit that user to in the top right text control box (integer value numbers only. Invalid entries will be ignored on add). Once these two entries are filled, click the **Add/Update Limit** button to add that limit to the specified user.

You can specify the creator string as a regular expression, allowing you to match a pattern of creator strings with a single entry in this table.

If the user already exists in the limit list, then that entry will be updated to the new value. You can click on an entry to enable to **Remove this creator's limits** button on the bottom. Clicking on this button will remove the entry for the highlighted creator / user, and remove its limit of workers. You can remove multiple creators from this list at once by highlighting more than one entry with a [ctrl]+click, or a click on first entry and [shift]+click on last entry to highlight all entries between. When more than one entry is highlighted and the **Remove this creator's limits** button is clicked, all entries highlighted will be removed at once.


Master Options

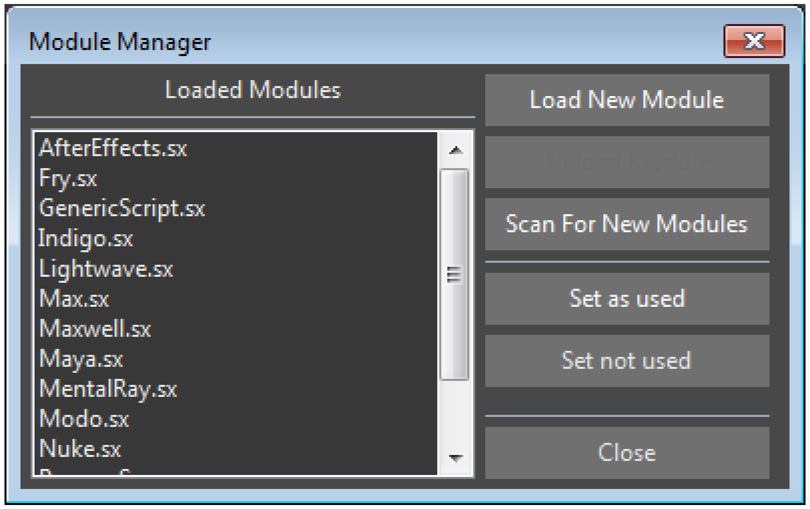
This panel sets stagger start and usage limits for pools
Use the Configure Pools command to create pools and add Engines

Test Pool 1A	Stagger:	0	work	0	seconds	Work Limit: 0
Convert	Stagger:	0	work	0	seconds	Work Limit: 0
Production	Stagger:	0	work	0	seconds	Work Limit: 0
Simulation	Stagger:	0	work	0	seconds	Work Limit: 2
Test Pool 0	Stagger:	0	work	0	seconds	Work Limit: 0
Test Pool 2	Stagger:	0	work	0	seconds	Work Limit: 0
Linux Admin	Stagger:	0	work	0	seconds	Work Limit: 3
Vector	Stagger:	0	work	0	seconds	Work Limit: 0
Mac Admin	Stagger:	0	work	0	seconds	Work Limit: 0
Test Pool	Stagger:	0	work	0	seconds	Work Limit: 0
Arnold	Stagger:	0	work	0	seconds	Work Limit: 0
Test Pool 1	Stagger:	0	work	0	seconds	Work Limit: 0
Windows Admin	Stagger:	0	work	0	seconds	Work Limit: 0

Pools Tab

This tab lets you configure individual work staggering and total work limits for each pool. You cannot create or manage pools and their members here, that should be done on the Pools tab of the Configure Engine window, because the pool ordering is handled per engine. However, once pools have been created, they will appear here and can have their extended settings configured here.

For each pool in your system, you will see two controls for the Staggering to define the number of workers to start in each period. If either number is set to zero here, staggering is disabled for the pool. You will see a third control in the row for that pool to configure the limit of the number of workers allowed for all jobs assigned to that pool.



Module Manager

The Module Manager allows you to dynamically load and unload Modules on the system, without having to shut down Smedge on your entire system. You can specify a module by name by using the **Load New Module** button, or select a loaded Module and use the **Unload Module** button to unload it. The **Scan for New Modules** button will request every Smedge component to re-scan the Modules folder for any new Modules to load.

Modules maintain a usage count. It could cause Smedge to crash if you unload a Module that is in use. This is especially important with Modules that provide Products. If there are any Jobs in the system with those Products, this could cause Smedge to crash. Just in case you want to override the usage count, you can use the **Set as used** button to increase the usage count, or **Set not used** to decrease it.

The Module Manager can only manage compiled Modules. Virtual Modules are handled by a specific compiled Module: ProcessSequence.sx. All Virtual Modules in the same folder as the ProcessSequence.sx module will be loaded or unloaded together when you load or unload the ProcessSequence.sx module. If you want to create separate sets of Virtual Modules, you must put them into different sub-folders of the Modules folder. We also recommend changing the filename of the ProcessSequence.sx Module itself in order to make it easier to spot in the Module Manager.

This can be especially useful for debugging your Virtual Modules. You may want to follow these steps:

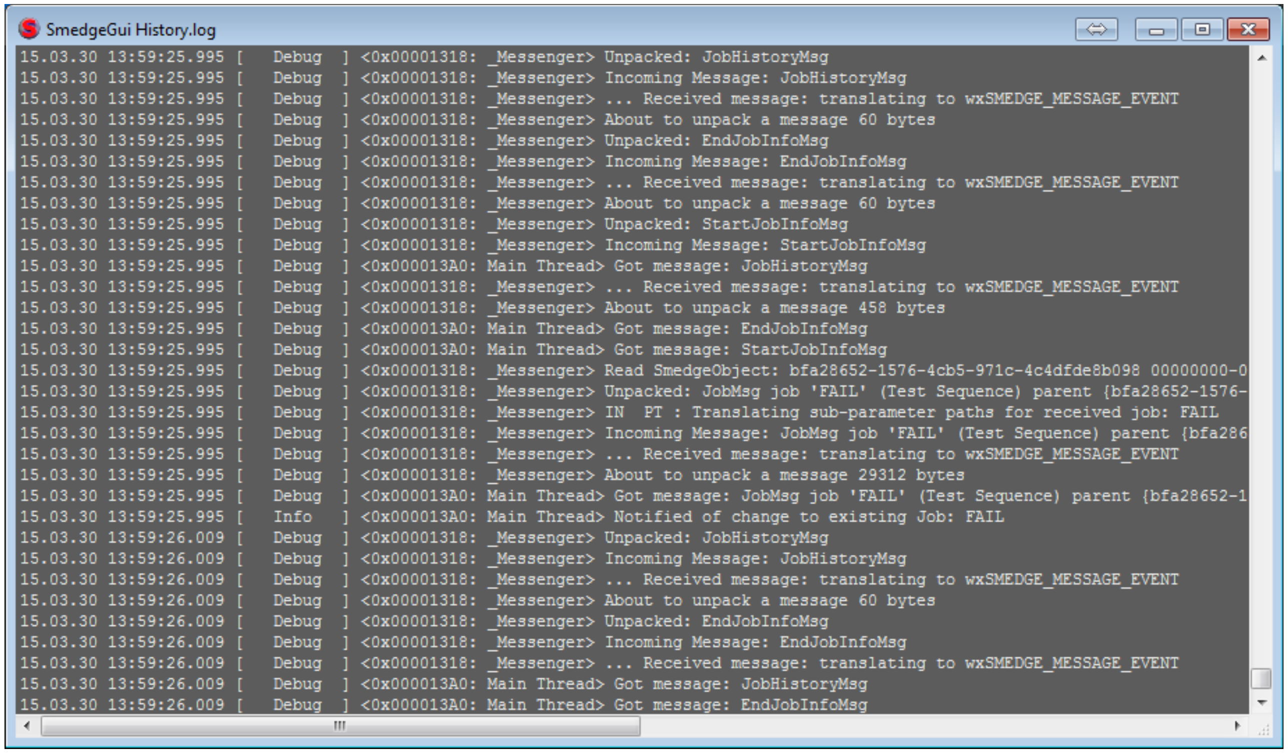
1. Create a sub-folder in your Modules folder

2. Copy **ProcessSequence.sx** into the new sub-folder

3. Rename **ProcessSequence.sx** to something more descriptive, like **DebugVirtualModules.sx**. Make sure you keep the **.sx** file extension, or the Module will not be loaded. Make sure that there are no other copies of the **ProcessSequence.sx** Module in the folder.

4. Create and debug your Virtual Module in the new sub-folder. You can use the Module Manager to load and unload the Virtual Module as you make changes. It will be listed with the new name that you provided in the previous step.

5. When your Virtual Module is ready, you can move it into the base folder, and unload and reload the primary **ProcessSequence.sx** Module.



```
15.03.30 13:59:25.995 [   Debug  ] <0x00001318: _Messenger> Unpacked: JobHistoryMsg
15.03.30 13:59:25.995 [   Debug  ] <0x00001318: _Messenger> Incoming Message: JobHistoryMsg
15.03.30 13:59:25.995 [   Debug  ] <0x00001318: _Messenger> ... Received message: translating to wxSMEDGE_MESSAGE_EVENT
15.03.30 13:59:25.995 [   Debug  ] <0x00001318: _Messenger> About to unpack a message 60 bytes
15.03.30 13:59:25.995 [   Debug  ] <0x00001318: _Messenger> Unpacked: EndJobInfoMsg
15.03.30 13:59:25.995 [   Debug  ] <0x00001318: _Messenger> Incoming Message: EndJobInfoMsg
15.03.30 13:59:25.995 [   Debug  ] <0x00001318: _Messenger> ... Received message: translating to wxSMEDGE_MESSAGE_EVENT
15.03.30 13:59:25.995 [   Debug  ] <0x00001318: _Messenger> About to unpack a message 60 bytes
15.03.30 13:59:25.995 [   Debug  ] <0x00001318: _Messenger> Unpacked: StartJobInfoMsg
15.03.30 13:59:25.995 [   Debug  ] <0x00001318: _Messenger> Incoming Message: StartJobInfoMsg
15.03.30 13:59:25.995 [   Debug  ] <0x000013A0: Main Thread> Got message: JobHistoryMsg
15.03.30 13:59:25.995 [   Debug  ] <0x00001318: _Messenger> ... Received message: translating to wxSMEDGE_MESSAGE_EVENT
15.03.30 13:59:25.995 [   Debug  ] <0x00001318: _Messenger> About to unpack a message 458 bytes
15.03.30 13:59:25.995 [   Debug  ] <0x000013A0: Main Thread> Got message: EndJobInfoMsg
15.03.30 13:59:25.995 [   Debug  ] <0x000013A0: Main Thread> Got message: StartJobInfoMsg
15.03.30 13:59:25.995 [   Debug  ] <0x00001318: _Messenger> Read SmedgeObject: bfa28652-1576-4cb5-971c-4c4dfde8b098 00000000-0
15.03.30 13:59:25.995 [   Debug  ] <0x00001318: _Messenger> Unpacked: JobMsg job 'FAIL' (Test Sequence) parent {bfa28652-1576-
15.03.30 13:59:25.995 [   Debug  ] <0x00001318: _Messenger> IN  PT : Translating sub-parameter paths for received job: FAIL
15.03.30 13:59:25.995 [   Debug  ] <0x00001318: _Messenger> Incoming Message: JobMsg job 'FAIL' (Test Sequence) parent {bfa286
15.03.30 13:59:25.995 [   Debug  ] <0x00001318: _Messenger> ... Received message: translating to wxSMEDGE_MESSAGE_EVENT
15.03.30 13:59:25.995 [   Debug  ] <0x00001318: _Messenger> About to unpack a message 29312 bytes
15.03.30 13:59:25.995 [   Debug  ] <0x000013A0: Main Thread> Got message: JobMsg job 'FAIL' (Test Sequence) parent {bfa28652-1
15.03.30 13:59:25.995 [   Info   ] <0x000013A0: Main Thread> Notified of change to existing Job: FAIL
15.03.30 13:59:26.009 [   Debug  ] <0x00001318: _Messenger> Unpacked: JobHistoryMsg
15.03.30 13:59:26.009 [   Debug  ] <0x00001318: _Messenger> Incoming Message: JobHistoryMsg
15.03.30 13:59:26.009 [   Debug  ] <0x00001318: _Messenger> ... Received message: translating to wxSMEDGE_MESSAGE_EVENT
15.03.30 13:59:26.009 [   Debug  ] <0x00001318: _Messenger> About to unpack a message 60 bytes
15.03.30 13:59:26.009 [   Debug  ] <0x00001318: _Messenger> Unpacked: EndJobInfoMsg
15.03.30 13:59:26.009 [   Debug  ] <0x00001318: _Messenger> Incoming Message: EndJobInfoMsg
15.03.30 13:59:26.009 [   Debug  ] <0x00001318: _Messenger> ... Received message: translating to wxSMEDGE_MESSAGE_EVENT
15.03.30 13:59:26.009 [   Debug  ] <0x000013A0: Main Thread> Got message: JobHistoryMsg
15.03.30 13:59:26.009 [   Debug  ] <0x000013A0: Main Thread> Got message: EndJobInfoMsg
```

Output Window

This window shows output that Smedge makes available. This includes output captured from the child processes that it spawns, and the Smedge log files.

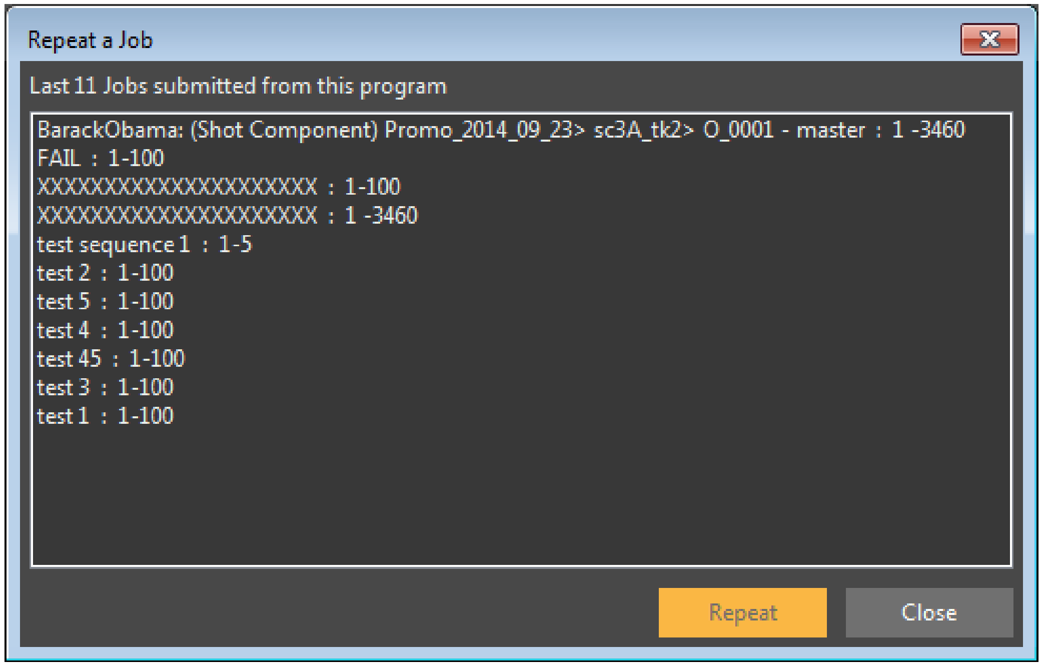
As you move the mouse over the window, it highlights the line under the mouse. You can use the mouse to select lines, which can be copied to the clipboard.

You can also control the window using the context menu. You can select all or no lines, copy the selection to the clipboard, or scroll to the top or bottom of the output this way. You can also use the keyboard to navigate the window, and to perform the menu operations.

If you have scrolled to the very bottom of the window, then the window will automatically continue to scroll when new lines are added. Otherwise, it will maintain your current scroll position as data is added to the end.

You can save the contents of the Output Widnow to a text file. Use the menu and select Save As... or press Control + S, then supply the name and path where you want to save.

You can also search for text in the window. Use the menu and select Search, or press Control + F. This will expand a search control where you can type in a search term, and search either up or down in the output for the next (or previous) occurrence of the given text. If the text is found, the line it is found on will be highlighted, with the search term further highlighted, and the line will be scrolled to the top of the display. You can choose if the search will be case sensitive or not.



```
BarackObama: (Shot Component) Promo_2014_09_23> sc3A_tk2> O_0001 - master : 1 -3460
FAIL : 1-100
XXXXXXXXXXXXXXXXXXXXXX : 1-100
XXXXXXXXXXXXXXXXXXXXXX : 1 -3460
test sequence 1 : 1-5
test 2 : 1-100
test 5 : 1-100
test 4 : 1-100
test 45 : 1-100
test 3 : 1-100
test 1 : 1-100
```

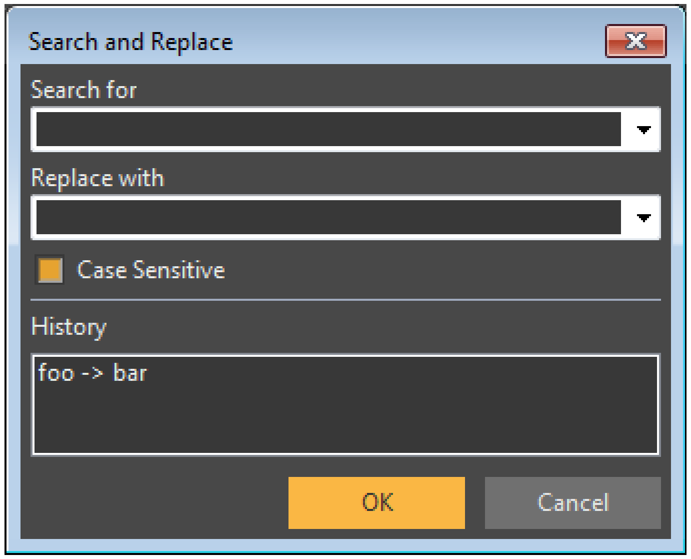
Repeat Job

This dialog allows you to repeat a recently submitted Job. The list will show recently submitted Jobs, and the summary information about that Job. You can select a Job and then click the **Repeat** button, or you can simply double click Job. What happens next depends on how you started the Repeat Job window.

You can start this dialog modelessly (meaning you can still use the SmedgeGui in any manner when it is open) by selecting "Repeat Job…" from the Job menu. In this case, repeating a Job will open a new Submit Job window with the settings from that Job.

You can also open this as a normal modal dialog if you click the **Repeat** button in the Submit Job window. In this case, the Submit Job window will be filled in with the settings from the job you repeated.

You can configure how many Jobs will be shown here. The default value is 20 Jobs. To change this number, choose "SmedgeGui Options…" from the System menu, and set the value in the SmedgeGui Product Options dialog.



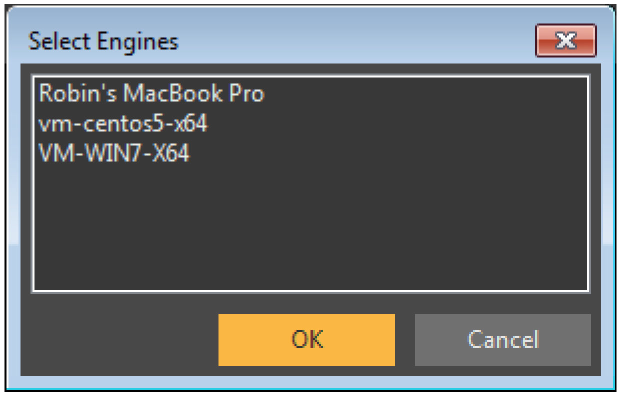
foo -> bar

Search and Replace

This dialog is opened when you click the **Replace** button on the Submit Job window. You can type any text into the **Search For** field, and it will be replace by the text in the **Replace With** field.

Both fields maintain an individual history of what you type in the field box. Click the button on the far right side of the field to open a drop-down list of previous values you have used. The dialog also maintains a history of the last search/replace pairs you have submitted. You can select one of these sets by clicking on it in the **History** list.

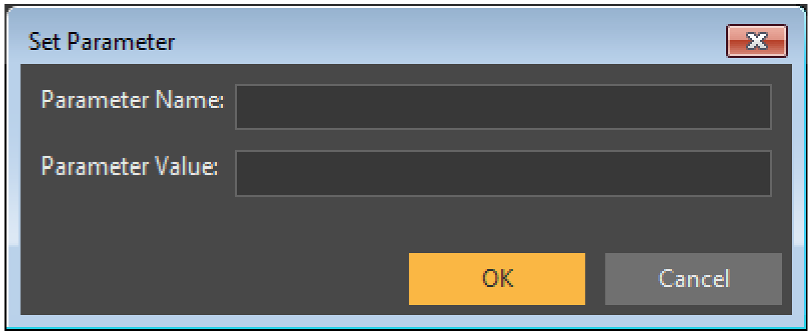
When you press **OK**, all text in any field of the Submit Job window will be replaced. Note that the auto-detect routines are not called when you use the Search and Replace dialog to change values.



Robin's MacBook Pro
vm-centos5-x64
VM-WIN7-X64

Select Engines

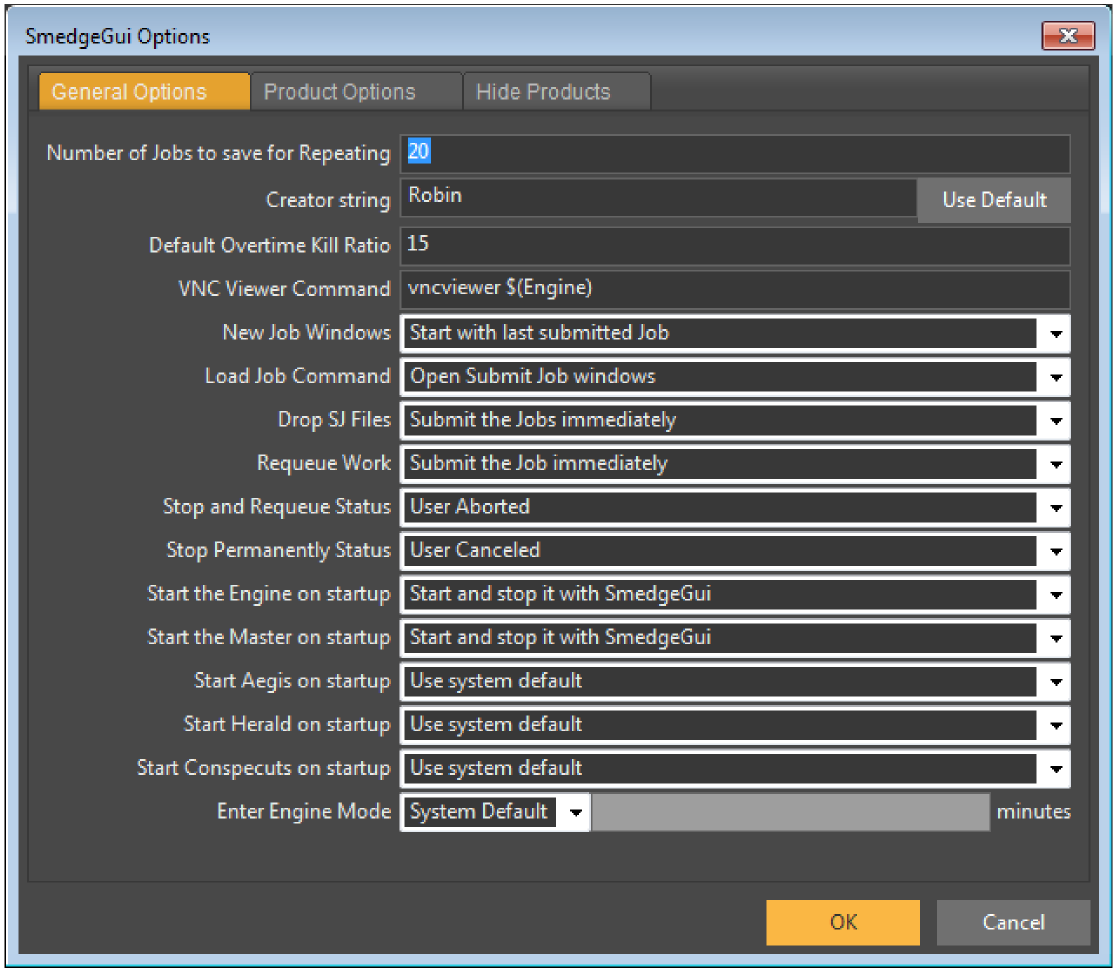
You can select Engines to include or exclude from a Job with this dialog. To select an Engine, highlight its name in the list. You can highlight multiple engines by pressing the Shift and Control keys while clicking on Engine names. When you are finished, press the **OK** button.



Set Parameter

This dialog allows you to set a job's parameter given the name of the parameter and the value you wish to set it to. This change will be applied to all jobs that are currently selected.

You can open this dialog by selecting job(s) and choosing the “Set Parameter...” option in either the job drop down menu or the right-clicked job context menu.



Number of Jobs to save for Repeating	20
Creator string	Robin
Default Overtime Kill Ratio	15
VNC Viewer Command	vncviewer $(Engine)
New Job Windows	Start with last submitted Job
Load Job Command	Open Submit Job windows
Drop SJ Files	Submit the Jobs immediately
Requeue Work	Submit the Job immediately
Stop and Requeue Status	User Aborted
Stop Permanently Status	User Canceled
Start the Engine on startup	Start and stop it with SmedgeGui
Start the Master on startup	Start and stop it with SmedgeGui
Start Aegis on startup	Use system default
Start Herald on startup	Use system default
Start Conspecuts on startup	Use system default
Enter Engine Mode	System Default minutes

SmedgeGui Options

This dialog is used to configure the SmedgeGui program and how it interacts with different Products. This dialog contains two tabs of options. The first tab, “General Options,” configures how SmedgeGui itself works. The second tab, “Product Options,” shows options available for customizing how SmedgeGui interacts with the options that the installed Products provide for Shells.

Number of Jobs to save for Repeating. SmedgeGui keeps a list of Jobs recently submitted by the current user through it. This option configures how many of these jobs are remembered in this list. Note that this list only applies to jobs submitted through SmedgeGui by the current user on the current machine. It is not a list of all jobs submitted to the system.

Creator String. This option allows you to override the “Creator” value for any Job submitted through SmedgeGui by the current user.

Default Overtime Kill Ratio. This option changes the default Overtime Kill ratio value for any Jobs submitted through SmedgeGui by the current user.

VNC View Command. This option specifies the command line that SmedgeGui will use to launch the remote Engine viewing program. SmedgeGui will substitute the Engine’s name where the text **$(Engine)** appears. You can also specify to use the IP address of the engine instead of the name with the text **$(Engine.IP)**. Note that the IP address is detected from the InfoPeer, so if that address is not valid, it will fall back to using the name only. If the path to your remove viewing executable program includes spaces, be sure to surround it in double quote marks.

New JobWindows. This option allows you to specify if the New Job window should be initialized with the information from the last Job you submitted, or if it should be initialized to the current default settings.

Load Job Command. This option allows you to specify what happens when you use the **Load Job** command in SmedgeGui. Your choices are to load the Job(s) immediately into the system, or to open Submit Job windows for every Job in the Job file.

Drop SJ Files. This option allows you to specify what happens when you drag and drop .SJ files onto the SmedgeGui window. Your choices are to load the Job(s) immediately into the system, or to open Submit Job windows for every Job in the Job file.

Requeue Work. This option allows you to specify what happens when you requeue work from a Job as a new Job. Your choices are to submit the Job immediately into the system, or to open the Submit Job window for the work you want to requeue.

Stop and Requeue Status. Use this to determine what status value is used when you stop and requeue work using the SmedgeGui menu commands. Different status may affect distribution and failure counting, or can be used for your own administrative purposes.

Stop Permanently Status. Use this to determine what status value is used when you stop work permanently using the SmedgeGui menu commands. Different status may affect distribution and failure counting, or can be used for your own administrative purposes.

Start the Engine on startup. This option allows you to configure if SmedgeGui should try to start the SmedgeEngine process on startup, in order to make the machine available for work as well as for monitoring the system. Your choices are to start and stop the Engine with the GUI, to start the Engine when the GUI starts, but leave it running if the GUI closes, or not start the Engine at all.

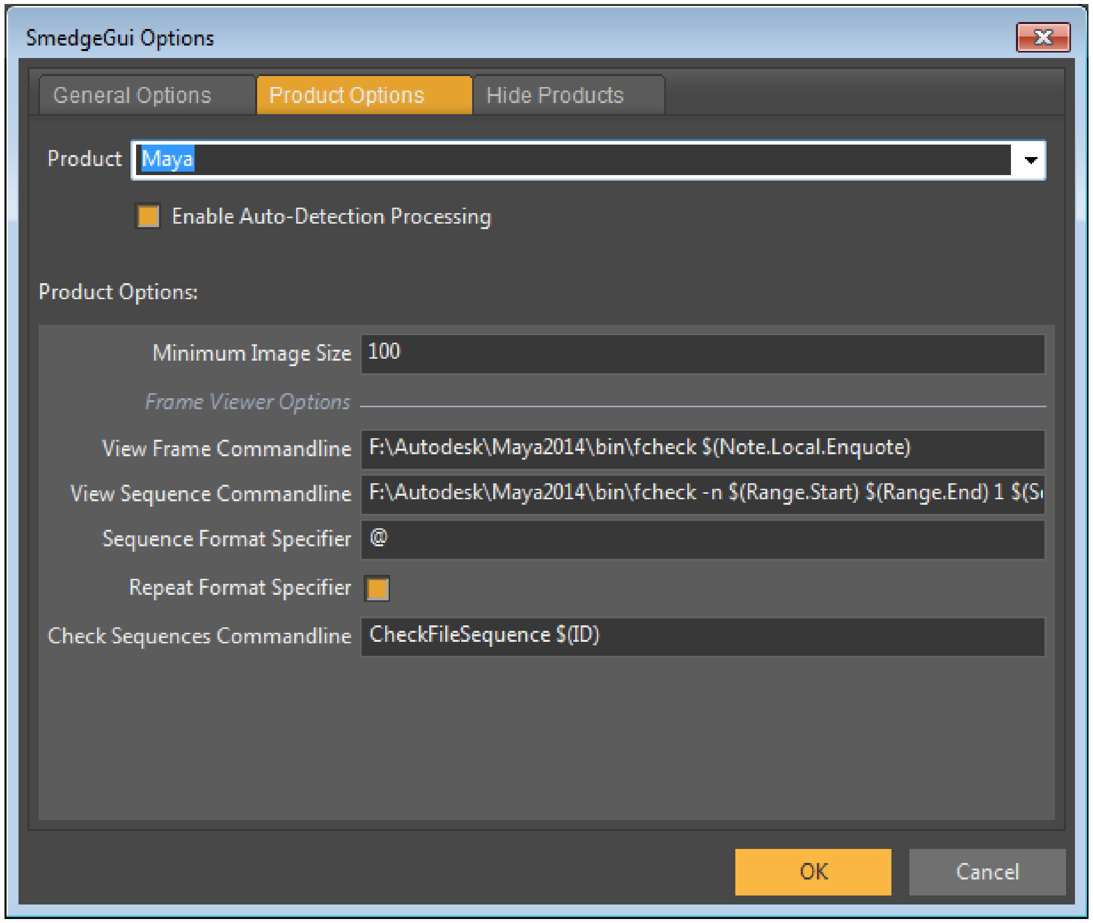
Start the Master on startup. This option allows you to configure if SmedgeGui should try to start the SmedgeMaster process on startup, in order to make the machine available to manage as well as monitor the system. Your choices are to start and stop the Master with the GUI, to start the Master when the GUI starts, but leave it running if the GUI closes, or not start the Master at all.

Start Aegis on startup. This option allows you to configure if SmedgeGui should try to start the Aegis shell process on startup. Your choices are to start and stop Aegis with the GUI, to start Aegis when the GUI starts, but leave it running if the GUI closes, or not start Aegis at all.

Start Herald on startup. This option allows you to configure if SmedgeGui should try to start the Herald shell process on startup. Your choices are to start and stop Herald with the GUI, to start Herald when the GUI starts, but leave it running if the GUI closes, or not start Herald at all.

Start Conspectus on startup. This option allows you to configure if SmedgeGui should try to start the Comspectus shell process on startup. Your choices are to start and stop Conspectus with the GUI, to start Conspectus when the GUI starts, but leave it running if the GUI closes, or not start Conspecuts at all.

Enter Engine Mode. Use this to customize how your GUI will enter Engine Mode, overriding the master option for the delay for entering Engine Mode automatically. You can set your GUI to use the System default, to never enter Engine Mode, or to select a custom time out for your machine that is independent of the system default.



Product: Maya

Enable Auto-Detection Processing

Product Options:

Minimum Image Size: 100

Frame Viewer Options

View Frame Commandline: F:\Autodesk\Maya2014\bin\fcheck $(Note.Local.Enquote)

View Sequence Commandline: F:\Autodesk\Maya2014\bin\fcheck -n $(Range.Start) $(Range.End) 1 $(S

Sequence Format Specifier: @

Repeat Format Specifier

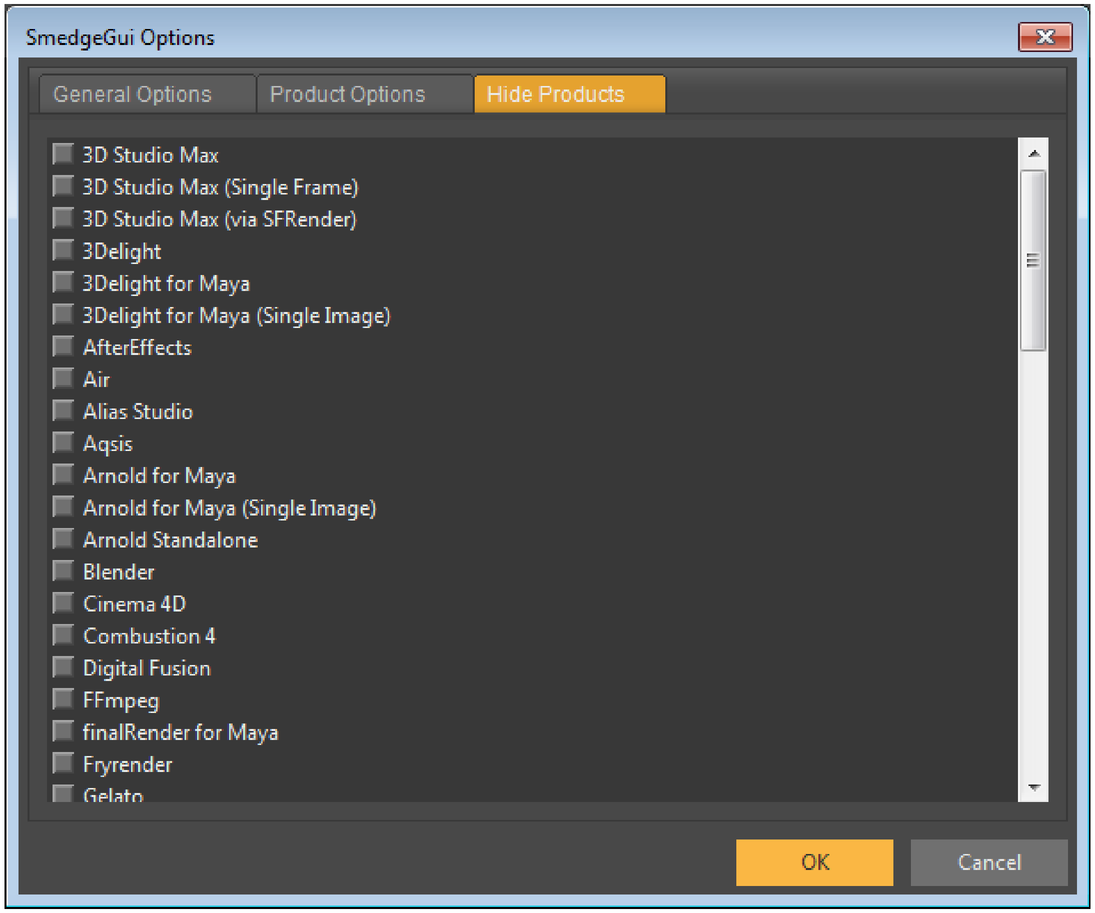
Check Sequences Commandline: CheckFileSequence $(ID)

Product Options

The second tab shows options available for the currently installed Products. What options appear and their default values will depend on the Product. Only the options for the currently selected Product are displayed in the Window. Select a different Product to see that Product's settings.

For each Product, you can disable the auto-detection processing system. Normally, when you are using the Submit Job window and you change a value, SmedgeGui will allow the Module to do some automatic processing to determine other attributes. Exactly what processing is done depends on the Module, but may involve reading through your scene files or other potentially slow processes that can make the GUI appear to hang. To disable this automatic processing, you can uncheck **Enable Auto-Detection Processing**.

The specific fields that are displayed in the bottom section will depend on the Product. We cannot describe every possible field that you may come across, but many of the common ones are described in the Common Custom Job Parameters chapter. Ask your system administrator for more information about the Products installed on your system.

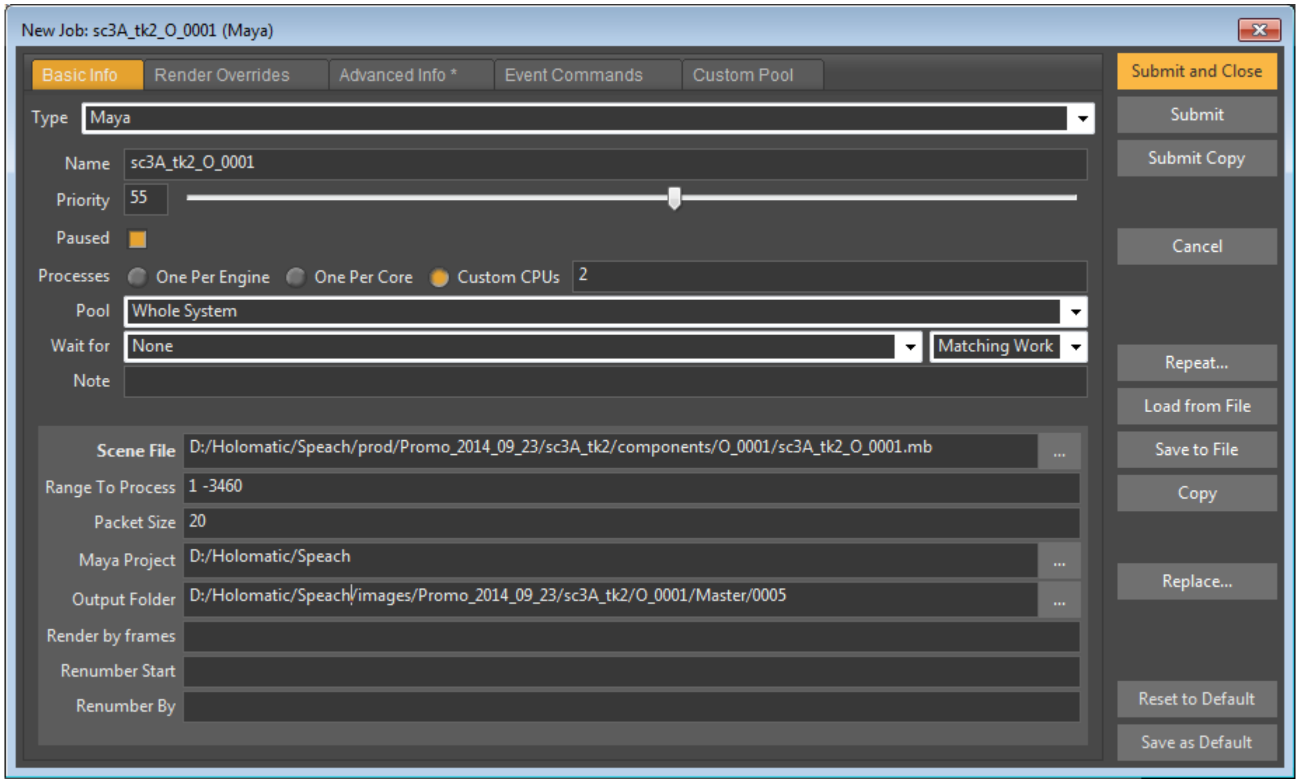


- 3D Studio Max
- 3D Studio Max (Single Frame)
- 3D Studio Max (via SFRender)
- 3Delight
- 3Delight for Maya
- 3Delight for Maya (Single Image)
- AfterEffects
- Air
- Alias Studio
- Aqsis
- Arnold for Maya
- Arnold for Maya (Single Image)
- Arnold Standalone
- Blender
- Cinema 4D
- Combustion 4
- Digital Fusion
- FFmpeg
- finalRender for Maya
- Fryrender
- Gelato

Hide Products

This tab allows you to hide which Products will be displayed in the GUI. This list will always show every possible Product that is installed on your system. Any products that are checked will be hidden, and any products that are unchecked will be displayed.

You can easily check or uncheck all products by right clicking on the list and choosing the command from the context menu.


New Job: sc3A_tk2_O_0001 (Maya)

Type: Maya
Name: sc3A_tk2_O_0001
Priority: 55
Paused: ✓
Processes: One Per Engine, One Per Core, [✓ Custom CPUs] 2
Pool: Whole System
Wait for: None — Matching Work
Note:
Scene File: D:/Holomatic/Speach/prod/Promo_2014_09_23/sc3A_tk2/components/O_0001/sc3A_tk2_O_0001.mb
Range To Process: 1 -3460
Packet Size: 20
Maya Project: D:/Holomatic/Speach
Output Folder: D:/Holomatic/Speach/images/Promo_2014_09_23/sc3A_tk2/O_0001/Master/0005
Render by frames:
Renumber Start:
Renumber By:

Submit Job

The Smedge GUI allows you to use a standardized interface to add Jobs, no matter what the specific data for that Job may be. To create a new Job select New Job from the Job menu. You can select if you want the New Job window to open with the last submitted Job, or with a set of Default settings. See SmedgeGui Product Options.

The Submit Job window is the primary interface for creating and configuring Jobs in SmedgeGui. The data is shown in a group of tabs, with some command buttons along the right side of the window. The Submit Job window is a "modeless" dialog box, which means that you can open multiple windows at the same time, and you can still access the main program window and menus with one or more Submit Job windows open.

Command Buttons

Submit (Update) and Close. For new Jobs, this button submits the Job to the system, and closes the Submit Job window. For existing Jobs this button is labeled with the text "Update", and sends the changes to the existing Job to the system, then closes the Submit Job window.

Submit (Update). For new Jobs, this button will submit the Job to the system. The window will then change its format to an existing Job editor. You can continue to make changes, which will be sent to the system when you click either the **Update and Close** or **Update** buttons. For an existing Job, clicking the **Update** button updates the Job being edited with the current values in the Submit

Job window. For both new and existing Jobs, clicking this button will send the Job information to the system, but will leave the Submit Job window open to make further changes to this Job. Note that once a Job has been submitted, you cannot change its type.

Submit Copy. Clicking this button will always submit a new Job to the system, using the current values of the Submit Job window for the new Job. After clicking this button, the window will remain open, and the dialog will not be linked to the just submitted Job. If the Submit Job window was originally opened as a New Job, you will still be able to change every parameter, including the Job type of the window, and the first two buttons will still say "Submit" instead of "Update".

Cancel. Cancel closes the Submit Job window without sending anything to the system. Note that if you previously used the **Submit (Update)** or **Submit Copy** buttons, any new Jobs or changes to existing Jobs that were previously sent will remain unchanged in the system.

Repeat. Clicking this opens the Repeat Job dialog box, which allows you to fill in the contents of the Submit Job window from a previously submitted Job.

Load. This loads the Job settings from a Saved Job file into the Submit Job window (if the file contains more than one Job, the Job with the lowest ID will be the one used).

Save. This allows you to save the current contents of the Submit Job window into a Saved Job file.

Copy. Clicking this will create a new Submit Job window with an exact copy of the current one. If you are currently editing an existing Job, the copy window will be in the state as if you created a new Job, so you will be able to change every field, including the Type. This is another way you can easily submit several similar jobs quickly.

Replace. Click this to open the Search and Replace dialog, allowing you to quickly change a text string in every parameter of your Job.

Reset to Default. Click this to set every field on every tab of the dialog to the current default settings.

Save As Default. Click this to customize the default settings that are used by the Reset to Default button. When clicked, the current settings on every tab of the dialog will be saved as the new default.

Basic Info

Type. Select which Product you are creating the Job for. You cannot change the Type of an existing Job.

Basic Job Info. These are attributes of the Job that are used by Smedge only. The values you specify here can configure how Smedge will distribute your Job, but changing these values will not change anything about how the work itself is performed.

Name. The Name you give your Job is completely up to you. There are no restrictions on what you can put here. Many products will attempt to fill in the name for you with other information from the Job. For example, a Maya Job will fill in the Name with the scene file name and the detected project when you change the Scene parameter.

Priority. Allows you to specify a relative ordering of Jobs. Priority can range from 0 to 100, with higher numbered Jobs having work distributed before lower numbered Jobs.

Paused. Allows you to determine if work will be allowed to be dispatched from the Job immediately upon submission. If you check the Paused box, no work will be dispatched until you Resume the Job. You can resume jobs with the Resume command in the Job menu.

Processes. Allows you to determine how Smedge will try to utilize the Engine machines' processors. One per Engine means that only a single worker from this Job is allowed to run on an Engine, no matter how many processors that Engine has. One per Processor means that the Engine will allow a Work Unit for every processor that the machine has. Or, you can specify a specific number of CPUs to request for the Job. The number you provide tells Smedge not how many workers to start, but how many of the available CPUs to claim for each Work Unit. Smedge will then start an appropriate number for the machine based on the Engine's available number of CPUs. If the Master's "Memory Distribution" mode is set to "Memory Only" then this line is hidden, and any jobs submitted will be set to use no processors. If a job is set to use no processors, either because CPU distribution is disabled or because the job has been specifically set that way, generally any work started this way will use all available cores on the machine, however different renderers may interpret this value differently.

Memory. Allows you to determine how Smedge will try to utilize the Engine machine's memory. None means that memory distribution is disabled for the job, and the processes count will be used to divide up the Engine's resources alone. All means that a worker from the machine will be assumed to use all available memory on the machine, essentially ensuring that only one worker from a job can run on a machine at one time (similar to "One per Engine" processor setting above). Alternatively, you can set a specific amount of memory to allocate for the work. If an engine has sufficient memory available, it will be allowed to try to start work from this job. If the Master's "Memory Distribution" mode is set to "Processor Only" then this line is hidden, and any jobs submitted will be set to use no memory. Note that, by default, Smedge does not actually place a physical limit on the memory that a process can allocate. This means that a job that may be set to use 200MB of memory per worker can actually have workers that allocate more than that, which can lead to overloading memory, resulting in slower renders if swap memory is available, or possible memory failures. For any process based jobs (almost everything), you can optionally configure Smedge to place a hard limit on the memory available to the process. This can avoid overloading memory and swapping, but may also lead to memory related work failures if the process needs more memory than was assigned to the Job. See the information on the LimitMemoryUsage parameter of ProcessJob in the Administrator Manual for more information.

Pool. Select a Pool to assign to this Job. Pools are another way you can prioritize Jobs, and are also used to define groups of Engines. Any Engine that is not a member of the selected Pool will never work on the Job, and other Engines will prioritize Jobs based on the ranking of the Pools to which they belong. See Pools for more information.

Wait For. Allows you to specify another Job upon which this Job will become dependent. All work from the selected Job must be finished, failed, or permanently cancelled before any work from the new Job will be allowed to start. Leave this at "None" to keep your Job free of dependencies. Be careful, because it is possible to make two Jobs dependent on each other, so neither will ever start!

If you check the **Whole Job** check box, then the Wait For Job will have to complete in its entirety (all work either finished or permanently canceled) before the second Job will start any work. Unchecked, the distributors will attempt to use "partial job waiting," where individual work from the waiting job can be sent out as soon as the corresponding work from the waited-for job has finished, even if the watied-for Job still has more work to do.

Note. Allows you to add any extra information you would like to the Job. Again, you can put anything you like here without restriction.

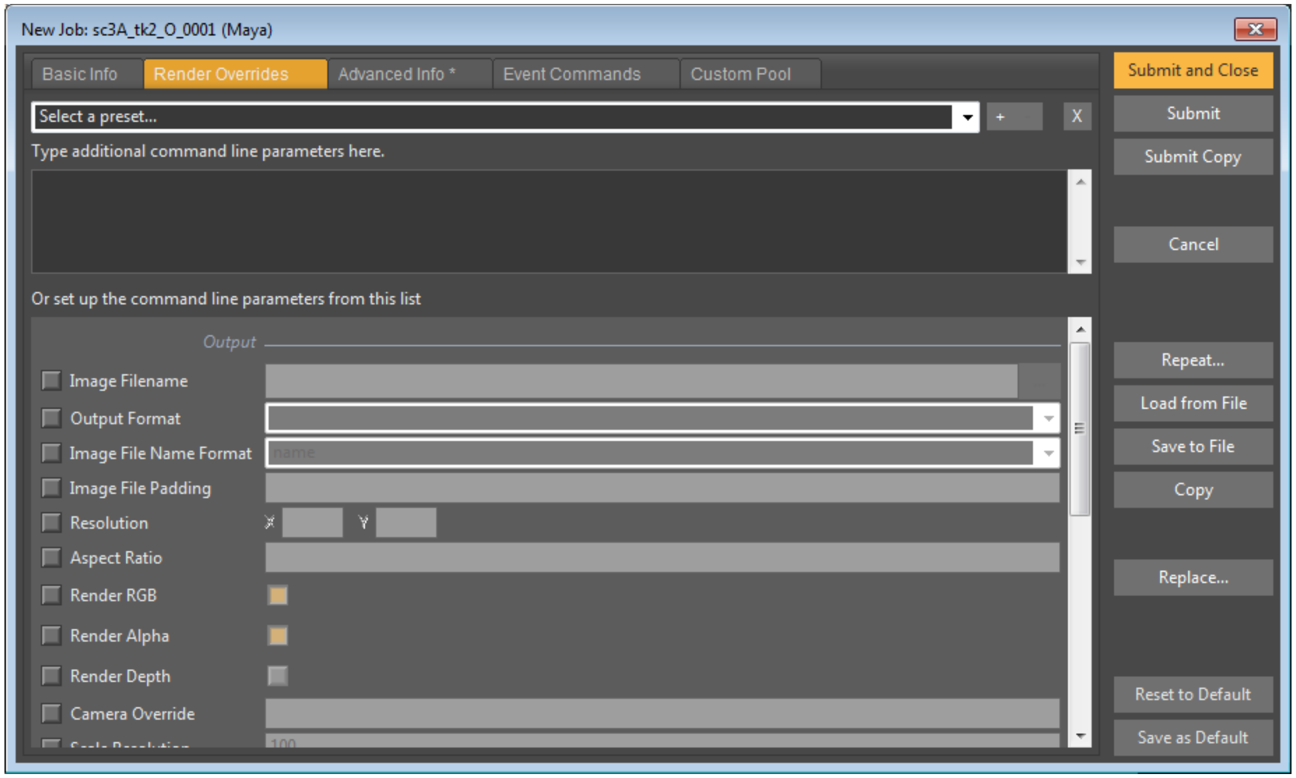
Type Specific Parameters. The contents of this area will change depending on the selected Product in the **Type** field at the top. Each Product can have its own specific parameters that are used to determine exactly how Jobs will function. For example, Jobs that are used for rendering animation sequences will generally have fields such as the scene file and the frame range here. Other types of Jobs may have completely different parameters. For more information about the fields that appear here, see the Administrator Manual or the documentation that comes with custom Modules you may have.

In this section, any fields that you must supply in order to distribute the Job are labeled with a **bold** label text. You must supply a value for these fields, or the Job will not distribute. Fields that are not bold can be left blank, or filled in with more data to configure the Job.

Finally, the right side of the dialog has the buttons. **OK** and **Cancel** are pretty self explanatory. **Submit Copy** will act just like OK, but will leave the dialog box open. This is one way to easily submit several similar jobs quickly.

Custom Tabs

Between the Basic Info and Advanced Info tabs, SmedgeGui will display a Tab with additional parameters that the Product makes available. For example, in the Maya Job shown above, there is a tab called "Render Overrides". This tab provides a bunch of optional overrides that you an pass to the Product as part of its operation. What tabs appear and what options they provide depends on each Product.



Type additional command line parameters here.

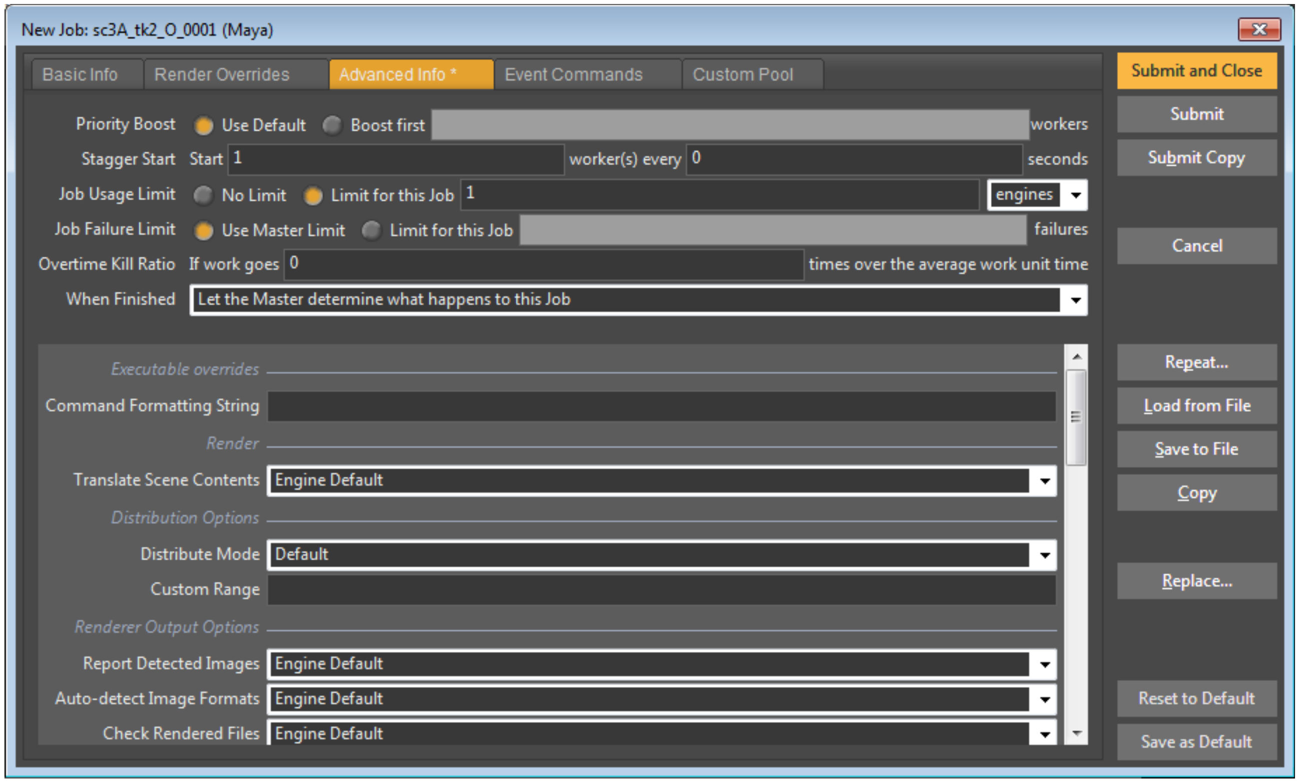
Or set up the command line parameters from this list

Note that these tabs actually correspond to a single parameter value for the Job, accessed with a single name. For example, the Maya parameters shown here are all combined into the $(Extra) parameter. You can see and modify the value of this Job parameter in the upper field (where it says "Type additional command line parameters here"). You can also manipulate the parameters individually with an intuitive graphic interface (where it says "Or set up the command line parameters from this list"). The Submit Job window will keep these two fields in sync as you edit them. You can make changes in either field at any time.

You can also create and use presets to automatically fill in the field with common values. For example, you can create a "Preview" preset that uses a lower resolution and lower quality anti-aliasing settings.

To use a preset, select its name from the list. To create a preset, set up the fields as you want, then click the button with the + character next to the list of presets. To remove a preset from the list, select that preset then click the button with the – character. Click the button with the **X** to clear the field and reset the entire tab.

Presets are stored for the current user on the local machine only. It is possible to make presets available to all users by adding them into a globally available SmedgeGui.ini file. For more information on doing this see the Administrator Manual.



New Job: sc3A_tk2_O_0001 (Maya)

Priority Boost [✓ Use Default] Boost first workers
Stagger Start Start 1 worker(s) every 0 seconds
Job Usage Limit No Limit [✓ Limit for this Job] 1 engines
Job Failure Limit [✓ Use Master Limit] Limit for this Job failures
Overtime Kill Ratio If work goes 0 times over the average work unit time
When Finished Let the Master determine what happens to this Job

Executable overrides
Command Formatting String

Render
Translate Scene Contents Engine Default

Distribution Options
Distribute Mode Default
Custom Range

Renderer Output Options
Report Detected Images Engine Default
Auto-detect Image Formats Engine Default
Check Rendered Files Engine Default

Advanced Parameters

The top section shows parameters that are available for all Jobs. You can use the **Stagger Start** information to keep too many Jobs from starting at the same time. This is useful, for instance, if your Job will access a network resource and you don't want to overload that resource when they all hit at once. You can set a limit of the number of work units that are allowed to start within a given time period. For example, if you want 2 workers to start then have a 30 second delay, then have the next 2 workers start, you would set the Stagger Start to: Start **2** worker(s) every **30** seconds. If you set the time delay to 0, then all workers will start immediately.

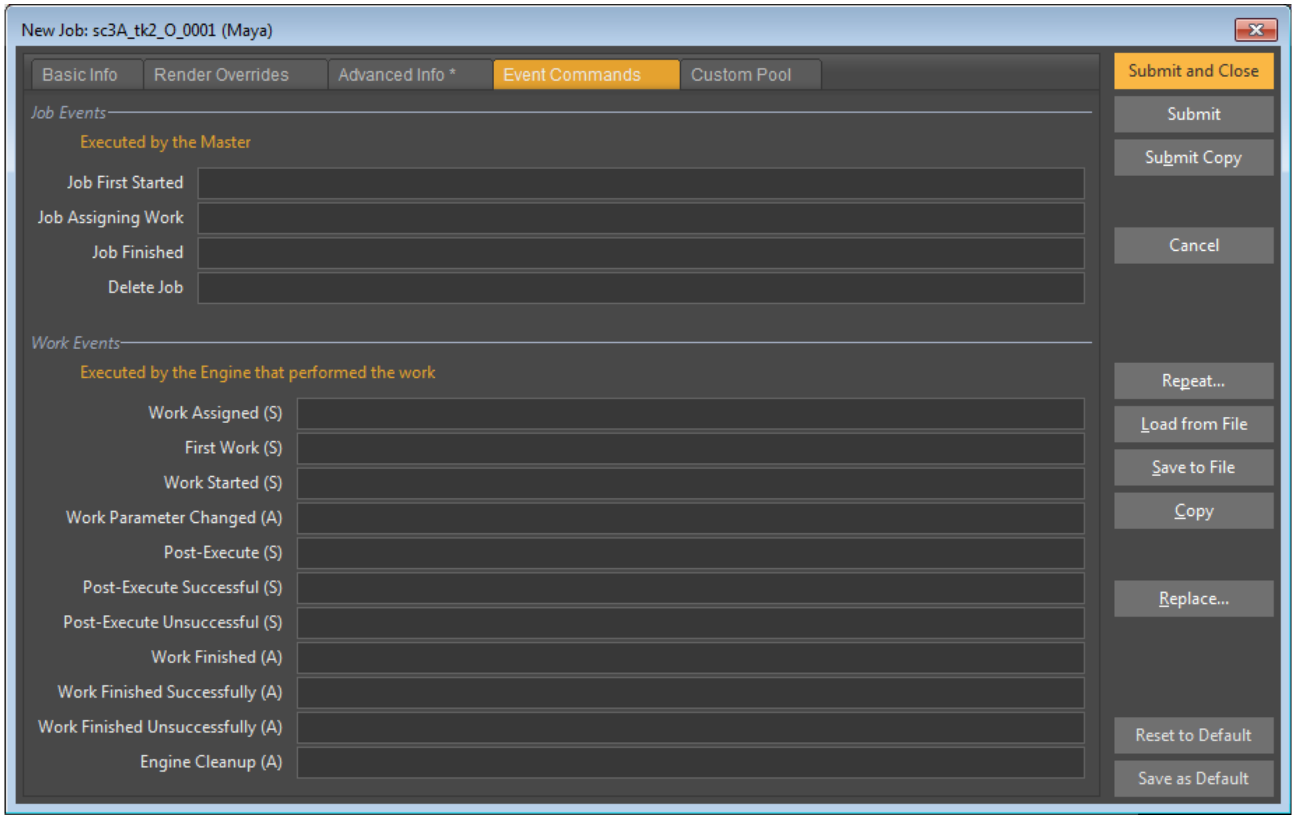
The **Job Usage Limit** allows you to specify a maximum number of workers from this Job that will be allowed to start at one time. The **Job Failure Limit** allows you to specify a custom failure count for the job that overrides the Master's Maximum Job Failure Limit. **Overtime Kill Ratio** allows you to specify when the Master should consider a worker as "hung" based on the ratio of how long it has been running to how long the average worker takes. A larger value will increase how long a process can remain "hung" before it is terminated.

You can override what happens to this Job when it is finished with the choices in the **When Finished** field. You have three choices:

Let the Master determine what happens to this Job	Uses the finished Job handling determined by the Master. You can configure this in the Configure Master dialog.
Keep this Job until it is manually deleted	This Job will remain in the system, no matter what the Master is normally configured to do, until someone manually deletes it.

Delete this Job immediately	This Job will be deleted from the system immediately upon completion (whether successful or not), no matter what the Master is normally configured to do.

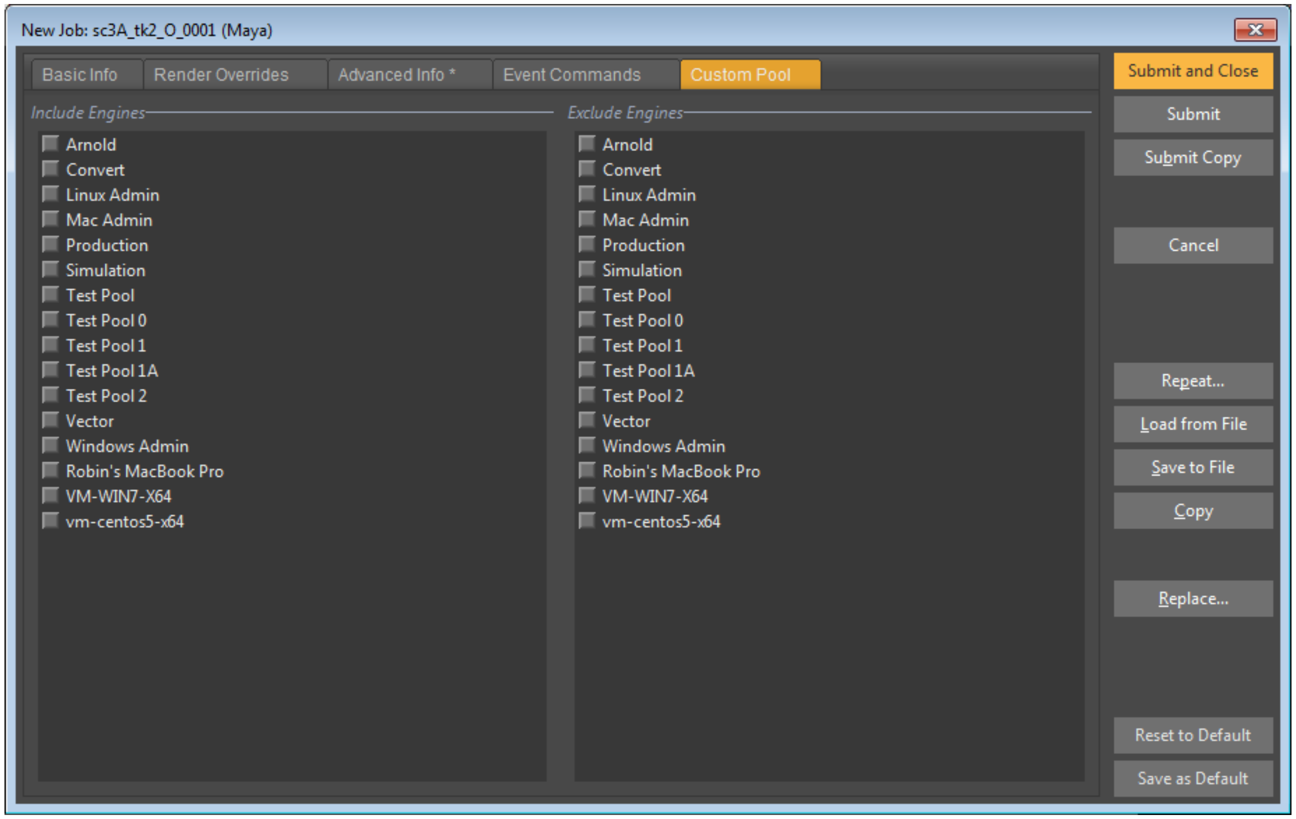
The bottom section shows the Product's custom advanced parameters. The exact list of parameters that will be displayed here will vary with each Product. For more information about the parameters included in most Smedge Modules, see the Common Custom Job Parameters chapter. Or ask your system administrator for more information about Products installed on your system.


New Job: sc3A_tk2_O_0001 (Maya)

Event Commands

Event Commands allow you to perform customized actions when things happen to the Job or its workers. There are two types of event commands, Job related events and Work related events. Job events are performed by the Master when overall Job actions occur. Work events are performed by the Engine for each work unit from a Job that the Engine works on.

In both cases, the program will execute the command listed in the appropriate field at the time of the event. The command lines will have standard Smedge variable substitution performed, using the $(Name) syntax. See Hooking into Job Events for more information on the events, or the Administrator Manual for more information on the $(Name) variable substitution syntax.



Custom Pool

Here you can customize the Pool of machines that will be used to perform work from this Job by manually including or excluding Pools and Engines. You can click on a Pool or an Engine and highlight its name to have it included or excluded. Note that if you have selected a Pool or an Engine in both lists, the exclude list will take priority and that Pool or Engine will be excluded from working on this Job.

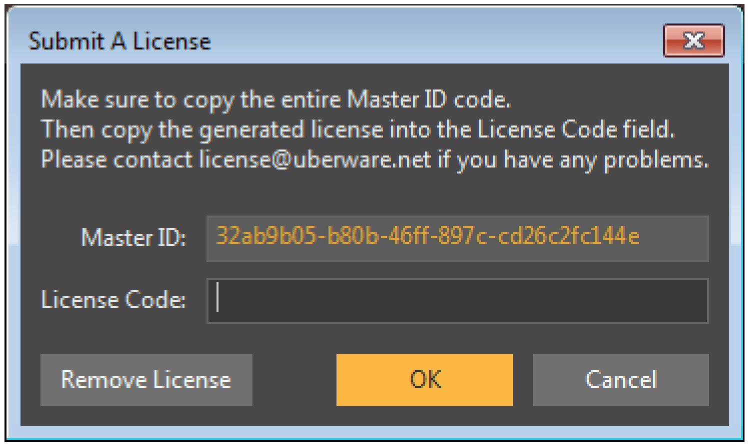
Any Pools you select in the **Include Engines** list will be used for distributing work from the Job in addition to the Pool assigned to the Job on the Basic Info tab. If you select the same pool here, it has no additional effect. It does not really make much sense to select pools to exclude from the Job, as they are already excluded by default.

Any Engines checked in the **Include Engines** list will be added to the Engines in the Pool assigned to this Job. If the Engine is already in the assigned Pool, checking it here has no effect.

You can highlight multiple Engines using the Shift and Control keys. Using the Shift key will select a range of Engines and using the Control key will select multiple individual Engines. Press the space bar to check or uncheck the selected items.

Note that, in general, it's best to avoid manually specifying Engines to include or exclude from working on the Job. Use your Pools and Engine settings to enable or disable certain types of work on specific Engines, as it is easier to understand and less likely to cause confusion about why an Engine is or is not working on a Job.

Multiple pools are only supported if you disable Pool Prioritization in the Master Options dialog.


Make sure to copy the entire Master ID code.
Then copy the generated license into the License Code field.
Please contact license@uberware.net if you have any problems.

Master ID: 32ab9b05-b80b-46ff-897c-cd26c2fc144e

License Code:

Submit License

Allows you to submit license information to the Master. You must be in Administrator Mode to access this dialog box.

The top control contains the ID string that is used to decrypt the license code. Make sure to send this to Überware licensing, or use this code on the Überware licensing web page in order to generate a license that will be able to be decrypted by your system. This number is based on information on the Master machine, so it does not matter from which machine you submit the license. However, if the networking hardware in your Master machine ever changes, or if you change which machine is running the Master, you will need to have a new license code generated. You can send a new request to Überware licensing, or you can use the Überware client home page, allowing you to regenerate your own license code whenever you wish.

You enter the code that you receive back in the bottom field. The code is sent to the Master, which will try to decrypt it. If the decryption succeeds, this new license information will replace any existing license information. You can always enter any license code that can be decrypted by your Master machine at any time. So, if you have a permanent license, and get a temporary license code with a few extra machines, when that temporary code expires you can just re-enter the original permanent code and get your old license back.

Use the **Remove License** button to request the Master to discard the last installed license code. The Master will revert to the default licensing (as of Smedge 2014, allowing up to 3 machines to render). This can be useful if you have an expired temporary license that you need to remove to return to the default licensing.

Be sure to contact Überware licensing if you have any questions.

Colors

You can control the colors of the Smedge interface to have it fit in with your work environment and match other tools you may use. Smedge includes four built-in color schemes:

Lighter:	High contrast	Dark type	Light background
Light:	Low contrast	Dark type	Light background
Dark:	Low contrast	Light type	Dark background
Darker:	High contrast	Light type	Dark background

Smedge also allows you to customize the colors for every part of the interface. Currently, you must manually adjust the Colors.ini file to make this work. Colors you can use (note that some of these are not currently used by any built-in controls, but the labels exist for future expansion):

WindowBG	The basic window background color
WindowBGLighter	A slightly lighter background color to accentuate certain areas
WindowBGDarker	A slightly darker background color to accentuate certain areas.
Text	The default text color
TextRed	Alternate text color - red
TextOrange	Alternate text color - orange
TextYellow	Alternate text color - yellow
TextGreen	Alternate text color - green
TextCyan	Alternate text color - cyan
TextBlue	Alternate text color - blue
TextIndigo	Alternate text color - indigo
TextViolet	Alternate text color - violet
TextDarkRed	Alternate text color - dark red
TextDarkOrange	Alternate text color - dark orange
TextDarkYellow	Alternate text color - dark yellow
TextDarkCyan	Alternate text color - dark cyan
TextDarkBlue	Alternate text color - dark blue
TextDarkIndigo	Alternate text color - dark indigo
TextDarkMagenta	Alternate text color - dark magenta
3DLight	Color for lighter part of “3D” type effects (buttons and bevels)

3DDark	Color for darker part of “3D” type effects (buttons and bevels)
ButtonFace	Color for the face of button controls
ButtonText	Color for the text on button controls
Selected	The “selected” color (affects lists, headers and some active controls)
Header	Color for the header labels for list controls

Herald

The **Herald** is a Smedge Shell component that allows you to configure notification actions in response to Job and Work events. For an overview of the Smedge system structure and how the component applications work and for information about Events, see Hooking into Work Events above.

The Herald sits unobtrusively in your System Tray and connects to the Master to listen for notifications about the various Events. You can manage a list of as many event notifications as you wish using its simple interface.

The actions available are

• Display a message on the screen.	The message can be variable substituted to allow you to display information about the specific Job or Work that triggered the notification. The message can optionally close itself after a specified number of seconds.
• Play a sound.	You can use any WAV sound file, or use the default gong sound.
• Send an email.	The To, From, Subject, and the message body can all contain Job parameter variables that will be substituted before the email is sent. You can also configure the mail server, port and a username and password, if needed.
• Execute a command.	The command will also be variable substituted, allowing a nearly unlimited ability to perform actions with the Job or Work information.
• Save the Job.	You can save to a standard .SJ file, or to a custom file format, specifying whether to overwrite or append to an existing file.

Notifications

There are four key pieces of information that make up a Notification:

- ✔ The Job filter
- ✔ The Event
- ✔ One-Time or Repeating flag
- ✔ The Action to perform

Each notification also can be individually enabled or disabled. Disabled notifications will not be triggered, even if Herald has Enable Notifications checked on the main window. Enabled notifications will not be triggered if the global Enable Notifications check box has been unchecked.

Job Filter

The Job filter allows you to limit which Jobs will trigger the notification. If you specify "All Jobs", then the filter is ignored. Otherwise, you can specify the internal name of any parameter, a comparison, and a value to compare with. For each notification that the Herald receives from the Master, it will then try to obtain the value of the specified parameter, and perform the comparison you request against the value you supply.

You can specify any parameter name. If the parameter name is not found, then the comparison will fail and that Job will be filtered out. The actual comparisons done are not case sensitive.

Event

Herald can notify about several different Job events and several different Work events. Job related events will have the parent Job object's data available for variable substitution, and Work related events will have the child Work object's data available. Currently, these events are available for notification:

Event	*Object*	*Description*
Job Created	Parent Job	A new Job has been created and added to the system.

Event	*Object*	*Description*
Job Updated	Parent Job	A Job has been changed
Job Paused	ParentJob	A Job was paused
Job Resumed	Parent Job	A paused Job was resumed
Job First Started	Parent Job	The first work unit from the Job has been started
Work Assigned	Child Work	A Work unit has been assigned to an Engine
Work Started	Child Work	A Work unit has started
Work Updated	Child Work	A Work unit's status or data has been changed
Work Finished	Child Work	A Work unit has finished
Work Finished Successful	Child Work	Follow on to Work Finished only for successful Work
Work Finished Unsuccessful	Child Work	Follow on to Work Finished only for unsuccessful Work
Job Finished	Parent Job	The last work from a Job has finished
Job Deleted	Parent Job	The Job has been removed from the system.

One-Time or Repeating

One-Time Notifications will be removed as soon as they are triggered. Repeating Notifications will remain until you remove them manually.

Action

The action determines what is going to happen when the specified Event for the filtered Job actually gets triggered. Each action has its own custom parameters that you can configure. Most of these parameters are run through the Smedge variable substitution mechanism, allowing you to use the standard $(***Name***.*Command*) syntax. The specific object that determines the values for the substitution depends on the type of Event. Parent Job events will use the parent Job’s data, and Work events will use the child Work unit’s data.

Interface

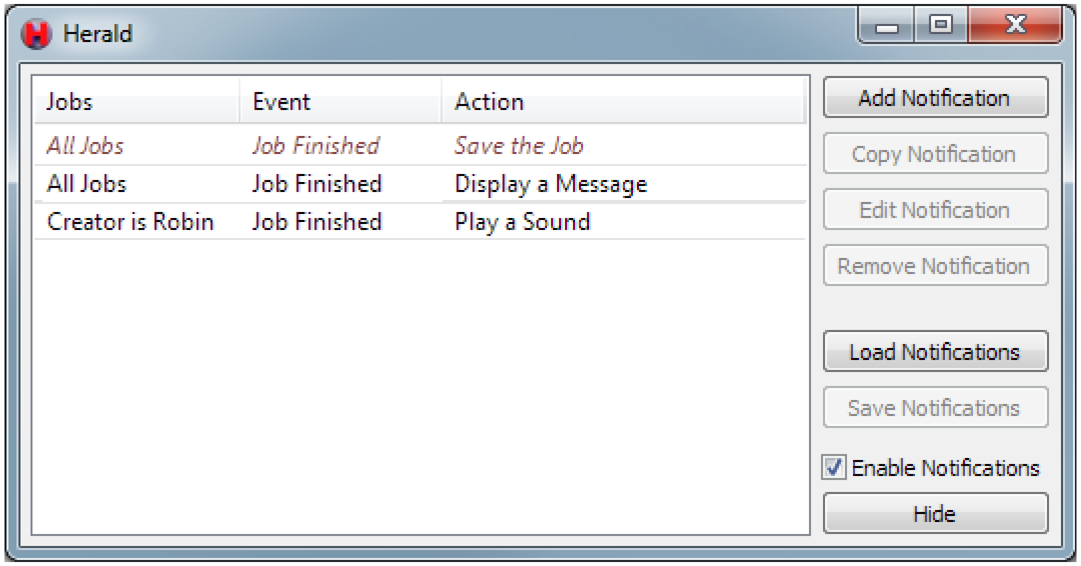
System Tray

Herald starts by default minimized to the System Tray area. You will see the little red "S" Smedge logo in the your system tray. You can access some commands by right-clicking on the S. Double clicking the icon or selecting "Show Notification Manager" will display the Notification Manager window, where you can add, edit or remove notifications.

You can temporarily enable or disable all notifications without actually shutting down the application. To do this, you can use the "Enable Notifications" command in the system tray menu. A check mark will appear next to this menu item when the notifications are enabled. When they are disabled, you can still manage notifications, but no notification actions will be performed when Job or Work events are sent from the Master.

The "Launch SmedgeGui" command will attempt to launch the SmedgeGui shell application that resides in the same folder as the Herald executable. Similarly, "Launch Aegis" will try to launch the Aegis shell application, which allows you to easily control your local Engine with 3 simple butons, and "Launch Conspectus" will try to launch the Conspectus shell application to get a graphical overview of the memory and CPU usage on all currently connected Engines.

Finally, you can stop the Herald process by selecting "Exit" from this menu.



Jobs	Event	Action
All Jobs	*Job Finished*	*Save the Job*
All Jobs	Job Finished	Display a Message
Creator is Robin	Job Finished	Play a Sound

Notification Manager Window

This window allows you to create, edit and modify the Notifications. Existing Notifications are listed here, showing the Job filter, the Event, and the type of Action.

Add Notification opens the Notification Editor dialog to create a new Notification.

Copy Notification will create a duplicate of a selected Notification.

Edit Notification opens the Notification Editor dialog to modify an existing Notification.

Remove Notification deletes the Notification.

Load Notifications loads notifications from a file and adds them to the action list.

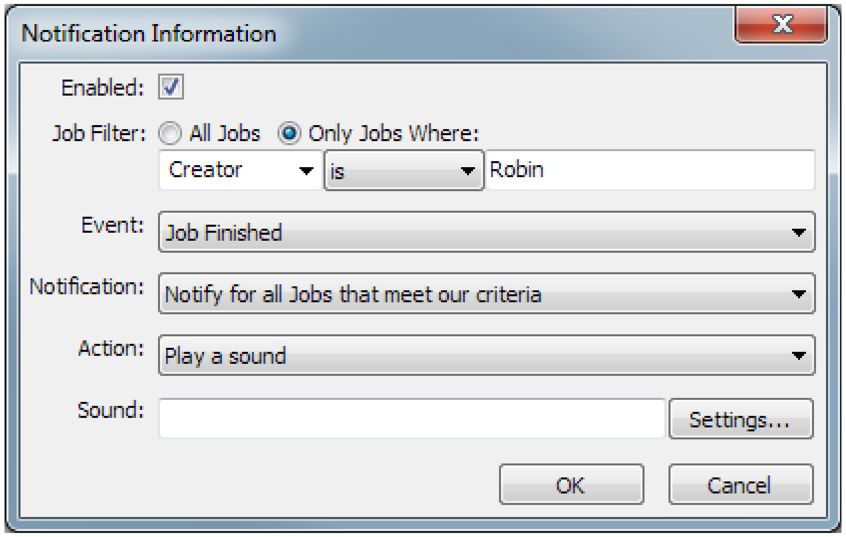
Save Notifications saves the selected notifications into a file for later use.

You can uncheck the **Enable Notificatons** box to temporarily disable all notifications. This is the same as unchecking the "Enable Notifications" menu item in the system tray. Note that this does not affect the individual notification enabled status. Notifications that have been individually disabled are shown in italicized red in the main window.

Hide will hide the window. Pressing the standard minimize button or command from the system menu will do the same thing. You can restore this window using the "Show Notification Manager" command from the system tray icon.

Notification Editor Dialog

Job Filter settings allow you to specify if the Notification should apply to all Jobs, or only to a subset of Jobs. If you choose to apply the Notification only to a subset of Jobs, you need to specify how the filter is applied. The first field allows you to specify the internal name of the Parameter you want to compare. You can type any parameter name you want here, but a list of some useful names is available via the drop-down portion menu.


Notification Information

Enabled: ✓

Job Filter: All Jobs [✓ Only Jobs Where:]

Creator | is | Robin

Event: Job Finished

Notification: Notify for all Jobs that meet our criteria

Action: Play a sound

Sound:

The next box allows you to specify the comparison that will be used. The comparison is case insensitive. Comparisons available:

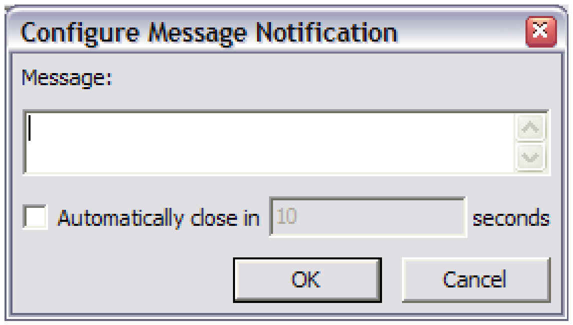
is	The parameter matches the value you supply exactly
is not	The parameter does not exactly match the value you supply
contains	The value you supply is found anywhere in the parameter's data
starts with	The value you supply is found at the beginning of the parameter's data
ends with	The value you supply is found at the end of the parameter's data

In the last field you put the value you want to filter with.

Event allows you to select which event the Notification is about. See the list of Events above for more information.

Action allows you to select which action the Notification will perform when it is triggered. Which ever action is selected here will modify the last field available, which is the most commonly changed. This parameter, and all of the other parameters for the action can also be changed by pressing the **Settings** button. This will bring up one of the following Configure dialog boxes, depending on which action you have selected.

Configure Message Dialog



You can type the text of your message into the main text control. To have the message automatically close after a period of time, click the **Automatically close** check box, and type the number of seconds you want the message to appear.

The **Message** can also be set directly from the Notification Editor dialog.

Configure Sound Dialog

You can specify a WAV sound file, or allow Herald to use the default. To specify a file, uncheck the **Use Default** check box, and type the path to the sound file into the text field. You can use the button with the three dots on it to open a open file dialog and browse for a file.

If you want to sample what the sound file sounds like, click the **Play Sound** button.

The Sound filename can also be set directly from the Notification Editor dialog. If you it is blank, then the default will be used.

Configure Email Dialog

Specify all of the fields that will create the email. You can use variable substitution on the **To**, **From**, **Subject**, and **Message** fields. You cannot use variable substitution on the **Host**, **Port**, **Username**, or **Password** fields.

You can specify multiple recipients, using commas or semicolons to separate the entries. Also, the To and From email addresses can be either just an email address or a nicely formatted name and address using the *name <email>* format.

You can also set the **To** field from the Notification Editor dialog.

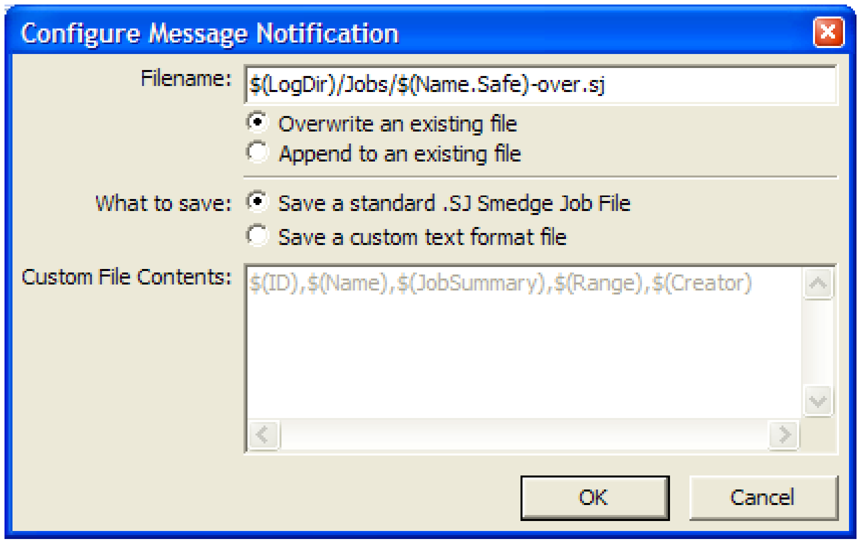
Note that Herald's email system is quite limited. It does not support SSL connections, and does not communicate with SMTP servers that use secure email protocols. You can get around these limitations by using a separate command line based SMTP client to send the actual message, and using the “Execute a Command” Action instead to pass the data you require to this client process.

Configure Command Dialog

Use the **Command** field to specify the full command line to execute. You can set the **Priority** at which the process is launched, and specify if it should be launched visibly or hidden using the **Visiblity** field. The Command will be variable substituted before it is run.

You can also set the **Command** directly from the Notification Editor dialog.

Configure Save Dialog



Filename: $(LogDir)/Jobs/$(Name.Safe)-over.sj

[✓ Overwrite an existing file] Append to an existing file

What to save: [✓ Save a standard .SJ Smedge Job File] Save a custom text format file

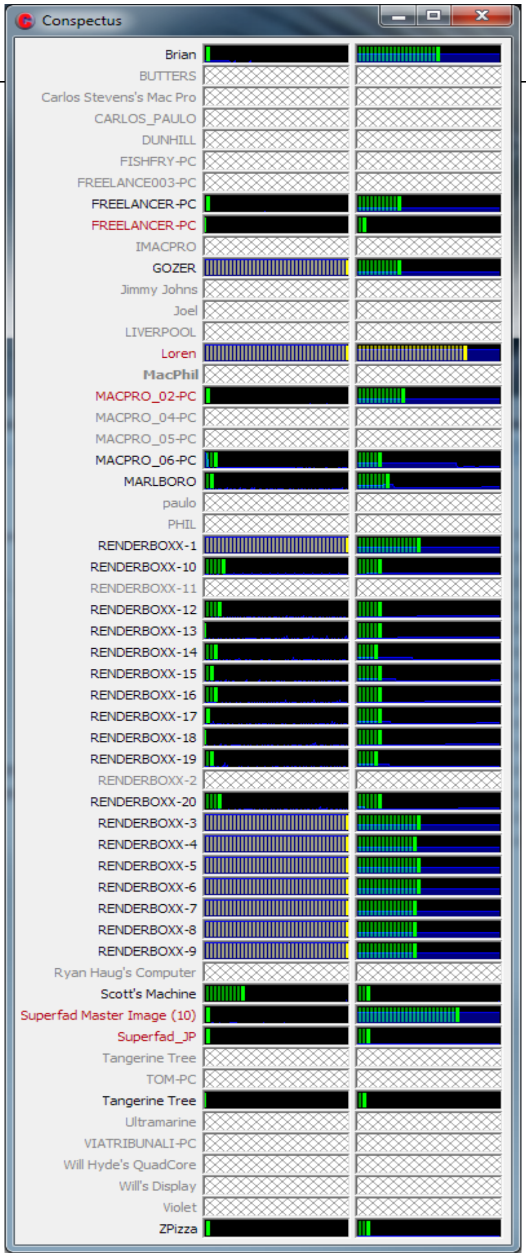
Custom File Contents: $(ID),$(Name),$(JobSummary),$(Range),$(Creator)

Set the filename and options in the **Filename** field. The filename string you pass will be variable expanded to generate the actual filename, and the folder that the file exists in will be created if it does not already exist. You can choose whether you want to overwrite an existing file or append to it.

Use the settings in the **What to Save** section to specify the details of the file that is saved. **Save a standard .SJ Smedge Job File** will save the INI format Smedge Job file, a text file format that includes every detail of the Job. .SJ Job files can be imported directly into Smedge using the SmedgeGui or the Submit shells. If you selected to “append” to the file, then any existing Jobs in the file are preserved, creating a single file that can submit every Job that has been saved in it.

Alternatively, you can use your own custom text format, selecting any parameters you wish using the standard $(Name.*command*) parameter substitution syntax. Herald will always add a newline character after whatever you put into this field when it saves the file. This can be used to create, for example, a CSV file with one line per event, or you can put a more complex structure in to suit whatever needs you may have for this file. Please note: this file is a text file. There is currently no way to save the Job information in a binary format. You must use the Smedge API if you want to be able to access the Job information as binary data.

Conspectus



Conspectus is a simple graphical view of the name, CPU status and memory status of every Engine in the system. It does not provide any control for the Engines.

Engines that are offline will show up with grayed out status controls. Those that are online, but disabled from doing work (in the Engine Settings) will show up with a red label.

The current Master will show up with its name in bold characters if that machine is also running the Engine (or ever has). When it's not connected, the window will be blank except for the text "Not connected."

If you put the mouse over an engine's display, you will see more information. Mouse over the Engine's name to see the hardware, operating system and memory information for the Engine. Mouse over the graph to get a specific number that it is representing for the processor or memory usage.

You can right click on any Engine in the list to access some commands. You can use this menu to enable the Engine, disable it deferred or disable it immediately. Conspectus will actually call the Engine command line shell to change the Engine state.

The context menu also allows you to specify an update frequency. You can specify an update period from 1 second to 1 hour. A longer update period will mean that the updates are less often and that the history will represent a correspondingly longer period of time. However note that longer times mean you have lower sampling resolution. The value shown will be the value at the time the sample is taken, not an average over the duration of the period.

CheckFileSequence

CheckFileSequence is a separate graphical shell application. SmedgeGui can start this tool using the “Check File Sequences” command in the Job menu or the Job context menu for Jobs that support it. You can also start it using the CFS toobar button if you have one or more jobs selected. If you have multiple Jobs selected it will start a separate instance for each Job.

CheckFileSequence looks for the rendered image files that Smedge detected from the output, and performs some basic validations on them. If the files are missing, the name shows up grayed out, and the time and size are missing. If the file exists but is under a minimum valid size, the file shows up in red. If the file is outside of the average file size for its neighboring frames, it shows up in orange, giving you a warning that this file may not be right.

If a Job has detected multiple different sequences being output by the renderer (e.g., render layers and passes, or multiple comps in a project file), then you can select which sequence you want to check from the list at the top of the window. Use the menu command **Job > Edit Custom Image Formats** to customize the file sequence specifiers.

By default, CheckFileSequence uses the Range set in the Job to determine which files to look for. It will also try to account for frame renumbering, if that is set in your Job. If you set the “Custom Range” parameter for your job (in the Advanced Info tab for most Products), it will use that range instead, and will ignore frame renumbering. You can change the Job's Custom Range directly by using the menu command **Job > Edit Custom Range**.

You can configure the verification settings and colors using the **File** > **Analysis Settings** dialog box. You can also either re-queue or delete bad, missing or custom selected frames from the interface, so you can easily fix just the frames that you need. You also have access to the custom Job commands for the Product. For most products, this includes the ability to view frames, view sequences, and explore the scene and images folders.

Aegis

Aegis is a very simple utility that gives you a GUI to control the local engine using the Engine command line shell. Its interface consists of three buttons that correspond to these three Engine commands:

Disable Now	`engine disable true`

This will disable the local engine and abort and re-queue any currently running work.

Disable Deferred	`engine disable`

This will disable the local engine but allow any currently running work to finish.

Enable	`engine enable`

This will enable the local engine immediately.

Aegis will also show you the current status of your engine along the top, including if there is currently any work going on the machine.

Command Line Shells

Smedge distributes with a suite of command line driven tools that allow complete access to the Smedge system from a command line or script.

Common options

All Smedge client applications are allowed to use the shared command line options. Unless otherwise specified, these options can be intermixed with the options specific to the application. See the "Common command line Options" section in the **Administrator Manual** for a full list of these additional parameters and their values.

ConfigureMaster

ConfigureMaster allows you to set the Master's options, limits, restrictions and more from a command prompt or script. The syntax for ConfigureMaster is:

```
ConfigureMaster
       -Option name value
       -SetUserLimit name value
       -RemoveUserLimit name
       -TypeOption type name value
       -Limit type limit
       -AddRestriction name
       -RemoveRestriction name
       -CopyFile file
       -GetMasterID
       -InstallLicense code
       -DumpLogs
       -SuperExit
       -GetPathTranslations
       -PathTranslation windows-root linux-root mac-root
       -RemovePathTranslation root
       [common-options]
```

Some options are designed to be able to be used multiple times to set multiple options at the same time. Others should only appear a single time on the command line for proper operation. ConfigureMaster tries to be somewhat smart about what you ask it to do, but of course it is possible to set up situations that could cause confusion or errors.

These options can be used multiple times on a single command line to set multiple options at once.

`-Option` *name value*	Set a distribution with the given name to the value you supply.
`-SetUserLimit` *name value*	Set a limit of outstanding workers that will be applied to the user of the given name, to the value that you supply. Note, for names with spaces, surround them in " ".
`-RemoveUserLimit` *name*	Remove the limit of outstanding workers that is applied on the user of the given name. You can optionally choose not to provide a name to remove all user limits.
`-TypeOption` *type name value*	Set an option for the given type with the given name to the value you supply.
`-Limit` *type limit*	Set a limit on the given type to the limit you specify
`-AddRestriction` *name*	Add a restriction with the given name

`-RemoveRestriction` *name*	Remove a restriction with the given name
`-PathTranslation` *windows linux mac*	Adds a path translation to the system. You must supply a value for all three possible roots for the translation, in the specified order (Windows, then Linux, then Mac). If you don't need a translation for a specific platform, you can use a blank root by using a pair of double quote marks for that root.
`-RemovePathTranslation` *root*	Removes a path translation from the system. You can use any of the platform roots for a specific translation set to remove that translation set (you do not have to use the local platform's root). If more than one translation is found with the same root, then all of those translations will be removed.

These options should only appear one time on the command line:

`-CopyFile` *file*	Sends all of the settings from the given INI file to the Master
`-DumpLogs`	Sends a request to have every client make a debug level log dump. These files contain a lot more information than the normal logs, which can be useful for Uberware support to help track down bugs or other issues.
`-GetMasterID`	Will show you the Master's ID code that is used to encrypt the license.
`-GetUserLimits`	Will show you current list of restrictions placed on user worker limits.
`-GetPathTranslations`	Will show you all current path translations
`-InstallLicense` *code*	Sends the given license code to the Master
`-Reboot`	Sends the request to reboot the machine the master is running on.
`-ReportLicenses`	Requests and displays the Master license report.
`-SuperExit`	Sends the request to perform a system-wide shut down of all Smedge components on every machine. Note that the components must have the Smedge messenger system enabled and be currently connected to the Master to receive the request.
`-Shutdown`	Sends the request to shut down the machine the master is running on.

Engine

Engine allows you to control the operation of any currently online Engine on your system. The syntax for Engine is:

```
Engine Command [parameter] [ID/Name…] [-regex expr…] [common-options]
```

Be sure to put any of the Common Options at the end of the command line. If you do not supply an ID, name or any regex patterns, Engine will try to operate on the Engine running on the local machine. If there is no Engine running on the local machine, or if an ID or name you supply does not refer to an actual running Engine, Engine will still succeed, though it may not have any effect. With the -regex flag, you can supply one or more regular expression patterns that match against the Engine name, ID, or Note to determine if the Engine is affected by the command.

Commands available:

The best way to configure Engines is using the Set command, which gives a lot of power and simplicity and uses JSON formatting to configure the settings.

```
Engine Set [Default] [type] object [type object …]
```

Set any arbitrary value or values with an object in JSON format. Quotes must be escaped. Optionally use the keyword **`Default`** after the command to set the values for the default engine.

Type before the object can be an ID, name or alias for a product to set the product options or default product options for that type. You can supply multiple product options from a single command.

To set options individually in the older way:

`Active`	Get the number of currently running workers.
`CheckOutLicense`	Tries to check out licenses for the Engine(s). Note that the Master will only check out as many licenses as it has available, so not all Engines may check out licenses.
`CopyFile` *file*	Copy all of the settings from an Existing Engine.ini file. This will not copy the Name, ID, Max CPUs, or hardware information strings. Any settings not in the file will be left at their current values, so you can use this to apply some settings to engines without changing any other settings about that Engine.

`Disable` [*boolean*]	Disables the Engine(s). Set the parameter to True to have any currently executing work aborted (and requeued) immediately. The parameter is optional: default is false, meaning that work will finish normally.
`DisableProduct` *product*	Disables processing of Jobs of the given Product on the Engine(s). Product can be a name, alias or ID for any known Product.
`DispatchLog`	Display the Dispatch Log of the selected Engine(s).
`DownloadFile` *file*	Requests to download the given file from the Engine. You can specify the file by path or by ID. The file must have been previously shared by the Engine. The file will be output to stdout, and the process will terminate when the whole file has downloaded.
`Enable`	Enables the Engine(s)
`EnableProduct` *product*	Enables processing of Jobs of the given Product on the Engine(s). Product can be a name, alias or ID for any known Product.
`Failures`	Shows the failure counts for the requested Engine(s).
`FollowFile` *file*	Requests to download the given file from the Engine. You can specify the file by path or by ID. The file must have been previously shared by the Engine. The file will be output to stdout, and the process will stay running so that as new data is added to the file on the Engine, that data is also downloaded.
`History`	Shows the history of the Engine
`List`	List all currently known Engines. (No IDs/Names required.)
`ProductOption` *Product OptionName Value*	
	Change option settings for a given product where *Product* is the name or ID of the specified product, *OptionName* is the CamelCase name of the specified option, and *Value* is the resulting value you wish to set to.
`Reboot`	Requests to reboot the given Engine(s).
`ReleaseLicense`	Requests for the Master to release licenses checked out to the selected Engine(s). This may be ignored if the Engines do not already have a license checked out.

`ResetFailures`	Resets all failure counts for the selected Engine(s)
`Run` *`<command>`*	Execute the given command string on selected Engine(s).
`SetNote` *note*	Sets the Note text for the Engine(s). In order to clear the note text, supply an empty set of double quotes ("") as the note value.
`SetProcs` *integer*	Set the number of processors to make available for work on the Engine(s). Parameter is the integer number of processors, and is a required.
`Shutdown`	Requests to shutdown the given Engine(s).
`Sleep`	Requests to put the given Engine(s) into a low power mode (sleep/standby).
`SleepOnIdle` *integer*	Sets the idle wait time that the engine will wait for before going to sleep (in minutes). If the wait time is set to 0, sleep on idle will be disabled for the engine(s).
`Stop`	Requests to stop all Smedge component processes on the given Engine(s)
`WakeOnLan`	Requests to send the wake up signal to the given Engine(s).
`Work`	Get a list of the IDs of the currently active work on the given Engine(s).

Job

Job allows you to get information about existing Jobs in your system, and modify their status, delete or resubmit them, or even to wait for completion. You can also get information about the installed products and their parameters. Be sure to put any common options at the end of the command line.

To get information about all jobs in the system:

```
Job GetProductInfo [Product] [common-options]
```

Product is a Name, ID, or Shortcut to a currently installed Product. This will list every parameter available for that Product with one line per parameter. The information displayed is:

Internal Name, Nice Name, Type, Default, Help

If you don't supply a Product, this will list all the available Products currently installed on the system, with one line per Product. The information displayed is:

Name, ID, Help

To get information about all Jobs in the system:

```
Job List [parameter ...] [-ANY ...] [-Creator [NOT] ...] [-Name [NOT] ...] [-Pool [NOT] ...]
         [-Status  [NOT] ...] [-Type [NOT] ...] [[-Line/-Tab/-Comma] [-Header] [common-options]
```

List one or more parameters by their internal name, and then only these parameters will be shown. If you use the spacial parameter name * (a single asterisk character), every available parameter will be shown for every available job. Otherwise, a default list of parameters will be shown. Note that the list of parameters is the same for every job if you supply no names or one or more parameters by name, but if you use the * parameter, different types of jobs may have more, fewer or different parameters.

The jobs listed can be filtered out into a smaller group by optionally adding on the switches: -Creator, -Name, -Pool, -Status, and/or -Type, to filter by those parameters. These switches take at least one argument, the name (or ID) of the value to filter jobs by. You can optionally add the keyword "NOT" before each argument to negate it and filter instead those jobs that do not have the provided values. For example, the command line `Job List -Name foo NOT bar "one two"` will filter out and list all jobs that have a name that includes "foo", does not include "bar", and has the string sequence "one two" somewhere in the name. In this example, the job "one twofoo" will be displayed, but the jobs "one foo two" and "one twofoo bar" will not be displayed.

By adding the optional switch [-ANY], you can change the behavior of the filter to display all jobs that have at least one of the provided values. By default, if you don't supply the [-ANY] switch, all provided values must be met by a job to be displayed. You can optionally add switch names as arguments to [-ANY] to apply this change to a specific switch. By default, if no arguments are added to any, the behavior is applied to all switches. For example, the command line `Job List -Any Name Type -Name foo bar -Type...` will filter out and display all jobs that have either "foo" or "bar" in its name, and has at least one of the provided types as its type, and all of the remaining switch argument values that you declare. Alternatively, the command `Job List -Name foo bar -Type ...` will filter out and display all jobs that have "foo" or "bar" in its name, or has a provided type as its type, or at least one of the other remaining switch agrument values that you provide.

Use –Line, –Tab, or –Comma to specify how the Jobs are formatted. –Line means one line per parameter. –Tab means to use a tab character to separate parameters. –Comma means to use a comma to separate parameters. –Header means to include parameter name information with the data. You can optionally specify which parameters you would like to show.

To get information about all Jobs in the system:

```
Job Count [-ANY ...]    [-Creator [NOT] ...] [-Name [NOT] ...]
          [-Pool [NOT] ...] [-Status  [NOT] ...] [-Type [NOT] ...]
```

This is similar to the command List, but does not take the optional beginning parameters, and instead of displaying the job information, this command counts how many jobs would be displayed by List (and thus, how many jobs pass the provided filter commands), and displays the resulting count. If no filter switches and arguments are supplied, then this command, [Job Count], will display the total number of jobs that currently exist.

To get information about specific Jobs

```
Job Info [Parameter-Name ...] ID [ID...] [-Line|-Tab|-Comma] [-Header] [common-options]
```

Lists information about specific Jobs. The optional ParameterName is the internal name of a parameter to show. If you don't specify one or more parameter names, then a default set of parameters will be shown. If a supplied name does not represent, Job will display:

```
Invalid Parameter
```

You can optionally specify if the list format will be one line per parameter, or one line per Job using tabs or commas to separate parameters. The default is "-Line". Optionally add -Header to include the parameter names as well as values.

To get the history from specific Jobs:

```
Job History ID [ID...] [-Dump [-Tab|-Comma]] [common-options]
```

Lists the History from specific Jobs. By default it lists the full history and statistics for the job in a human readable format, organized by the work name, run number and then element details, similar to the format of the history in the GUI.

If you use the -Dump flag, it will instead just output every history element associated with the job as a giant list, similar to how the command worked in Smedge 2012 and earlier. In this mode, you can optionally specify if the history element fields are separated with tabs or commas. The default is "-Tab".

To perform operations on specific Jobs:

```
Job Command [options] ID [ID...] [common-options]
```

Where *Command* is one of:

`Command` *Command-String*	Sends the custom named command to the requested Job(s). The same command is sent to all of the Jobs. You cannot currently supply any parameters to the command, so only commands that do not expect parameters will work as expected. If a Job does not understand the requested command the failure will be logged, but it will not affect operation or change the return code from the Job shell process itself.
`Delete [`*Boolean*`]`	Deletes the Job(s). Optionally, set the parameter to “True” to have any work currently executing stopped immediately. The default is “False” which will allow any currently executing work to finish normally.
`DeleteFinished`	Deletes all jobs that are finished. You do not need to supply an ID for this command.
`DispatchLog`	Prints out the dispatch report for sending work from the specified Job(s).
`FailedEngines`	Prints the IDs of the Engines that have failed on the specified Job(s).
`Failures`	Prints the failure counts for the specified Job(s).
`Pause`	Sets the status of the Job(s) to “Paused”. Work will not be distributed from the Job(s) while paused.
`Preempt [`*status*`]`	Stops all currently running work from the Job(s), and requeues that work. You can optionally supply a numerical status code to send to the stopped work. If you do not supply a status value, the system will use the default value that corresponds to “User Aborted” and the work will be requeued. Other status codes may have different effects, depending on the Product.

`ReportStatus`	Prints out the full status message for the distribution of the specified Job(s).
`ResetFailures`	Resets Job failure counts for the requested Job(s).
`Resubmit`	Resubmit the specified Job(s) as entirely new Job(s).
`Resume`	Sets the status of the Job(s) to “Pending” (if it’s not finished). Work can be distributed again.
`Save` *filename*	Save the specified Job(s) into a .SJ Smedge Job file. Any dependencies will be maintained, and all specified Jobs will be saved into a single file. You must provide a filename as the parameter.
`Status` *status*	Sets the Job's status to the value you supply.
`Usurp` [*priority*]	Stops any work from Jobs with lower priority than the lowest priority Job supplied, or lower than the optionally supplied priority value (within the range 1 – 100, inclusive).
`WaitFor`	The process will suspend until the specified Job(s) have finished or are deleted.

JobData

JobData allows you to directly read Job data from the Master disk database without having to be in network contact with the Master. The Master does not even have to be running, however, the JobData process must be able to directly access the disk files using only the operating system path to the file.

```
JobData file [parameter-name ...] [ -Separator separator]
```

You must supply the path to the .JOB file for the job file you want to examine. You can optionally supply one or more parameter names for the specific data parameters you want to retrieve. If you do not supply any parameter names, it reports the job status as a string value and the percentage of work complete. You can optionally supply the separator between returned values. The default separator is a single comma.

PoolManager

PoolManager allows you to display information about Pools, as well as create, rename and delete pools from the system. Note that it does not assign or remove Engines from a Pool. Instead, use the Engine shell to modify the Pools for a specific Engine. Be sure to put any common options at the end of the command line.

The syntax to use PoolManager for listing pools or pool members is:

```
PoolManager Command [ID/Name/Both] [ID/Name...] [common-options]
```

Where *Command* is one of:

`List`	List all available Pool names.
`ListMembers`	List all members of the given pool or pools. You can supply pools by ID or by Name. If you do not supply any pools, data about all pools will be listed.
`ListPools`	List all pools for the given Engine or Engines. You can supply engines by ID or by Name. If you do not supply any engines, data about all engines will be listed.

PoolManager will list the pools with the pool's ID, Name, or both. By default, it will list both. To supply a specific format, use one of these options to change how the data is displayed:

`ID`	Display only the ID of the Pool or Engine.
`Name`	Display only the Name of the Pool or Engine.
`Both`	Display both the ID and the Name of the Pool or Engine.

To create pools:

```
PoolManager Create ID/Name...  [common-options]
```

Each ID or name you supply will be used to create a new pool. If you create pools by ID, the pool name will be just the ID as a string.

To rename a pool:

```
        PoolManager Rename ID/Name New-Name [common-options]
```

You can rename a pool by supplying the original pool name or the pool ID, and a new name for that pool.

To delete pools:

```
        PoolManager Delete ID/Name...  [common-options]
```

Each ID or name you supply will be deleted from the system.

To signal for all engines within a pool to shutdown their machine:

```
        PoolManager Shutdown ID/Name...  [common-options]
```

Each ID or name you supply will have their contained engines/machines remotely shutdown.

To signal for all engines within a pool to reboot their machine:

```
        PoolManager Reboot ID/Name...  [common-options]
```

Each ID or name you supply will have their contained engines/machines remotely reboot.

Submit

Submit is the command line Shell that allows command line based submission of Jobs. While it is possible to use Submit interactively from a Command Prompt or shell window, the real power of Submit is the script mode, allowing you to integrate Smedge Job submission into other tools.

The syntax for Submit is:

```
Submit [Script] [-Type type-string] [-Paused] [-ParameterName value ...]
```

The parameter names, including the **Script** command are not case sensitive. This document uses capital letters, but they are not required. The *type-string* is the ID, name, or an alias for an installed Job type, and this search is also not case sensitive. Use the –Paused flag to indicate that you want the Job to be in the "paused" state when it is submitted. This means that no work will be allowed to be distributed from this Job until you resume it later. Without this switch, the Job will be submitted pending. In interactive mode, you will not have another opportunity to set this value for the Job.

Typing **Submit** as a command by itself will start Submit in Interactive mode. In this mode, Submit will prompt the user for any parameters that are not specified by the command line. The prompt will show the help text for the parameter, as well as the default value.

To use Submit in Script Mode, you must add the **Script** keyword. In script mode, you must supply enough parameters by command line to determine the Job type, and fill in any required parameters that do not have defaults. If you do not supply enough information, Submit aborts with a non-zero result code, and no Job is submitted. Other parameter names are included with the initial – character followed immediately by the name of the parameter you want to set, and then the value you want for that parameter.

The information about what ID and names will get a specific type is included in the section for that type in the Products chapter of the Administrator Manual. You can also get the complete list of the possible *ParameterNames* for each Product in the Administrator Manual. Modules from third party vendors may not be included. Ask your system administrator for more information about third party Modules that he or she may have installed on your system.

Submit allows you to pass parameters with spaces without having to quote them. These two are equivalent to Submit:

```
Submit -Name "Test Name" -Scene "c:\Documents and Settings\Robin\scenes\My Scene.mb"
Submit -Name Test Name -Scene c:\Documents and Settings\Robin\scenes\My Scene.mb
```

You can also use Submit to submit Jobs from .SJ Smedge Job files. The syntax for this is:

```
Submit –FromFile filename [-Paused [<yes/no>]] [-Creator [<creator name>]]
```

When submitting Jobs form a file, by default Smedge will preserve the “paused” status from the Job file, but will override the “Creator” value to be the current user. You can override both behaviors using the command line flags.

`-Creator [`*creator string*`]`	Override the “Creator”. If you supply a creator string value, that value will be used. If you leave off the creator string value, Submit will preserve the value that is saved in the Job file. If you do not add the `-Creator` flag at all, then the default behavior will be to that Submit will change the Creator to the user that started Submit.
`-Paused [yes/no]`	Override the “Paused” status. Optionally specify the status, yes for paused, no for resumed. Defaults to “yes”.
`-Update`	Preserves the Job IDs from the Job File. This allows you to edit existing Job(s) instead of adding new ones by specifying the Ids of the Jobs you wish to edit as the IDs in the Job file.

Finally, you can also use Submit to update a parameter to an existing job. The syntax for updating jobs is:

```
        Submit Update id parameter-name new-value
```

If a job with the given ID cannot be found, the update attempt will fail. If the job is found, the parameter with the given name will be updated with the value you specify. Note that if the new value contains spaces, you do not have to enclose it in quotes to have it used correctly. The entire rest of the Submit command line after the parameter name will be used as the value.

Getting Help

Although we strive to make setting up and managing a server farm as easy as possible, it is still a complicated task and there are a lot of options. If things aren't going well for you, there are several places you can get help.

Documentation

The documentation provides a lot of information about working with the product. Make sure to check the Administrator Manual or any custom Module manuals (like the "Using Smedge with After Effects" manual) for more details about advanced operation or specific product operation. The manuals are also available online on our web site, where you will always find the most up-to-date version:

http://www.uberware.net/smedge/docs.php

Frequently Asked Questions

There is a list of answers to common questions and basic problems available on our web site, in the list of Frequently Asked Questions. You can browse and search through it here:

http://www.uberware.net/faq.php

Remote Support

Uberware provides remote support and installation services. You can get more information here:

http://www.uberware.net/remote_support.php

Contact Us

You can contact Uberware technical support. Find the latest contact information on our web site here:

http://www.uberware.net/contact.php