SMEDGE

Administrator Manual

Smedge 2020

Table of Contents

ABOUT IDS 4

SMEDGE ENVIRONMENT VARIABLES 5

- Variables that control Smedge functionality 5
- Variables set for work processes 9

LICENSING 10

RESTRICTIONS 11

- Default Restrictions 12

AUTOMATIC SYSTEMS 13

- Automatic Redundant Master 13
- Automatic Master Location 15
- Automatic Engine Mode 16
- Automatic Engine Settings 17
- Automatic Executable Paths 18
- Automatic GUI Preset 19

RLIB INI FILE SYNTAX 20

- Alternate file locations 21
- Overloadable Options Files 22

.SJ JOB FILES 23

VARIABLE SUBSTITUTION 24

- Syntax 25
- Parameter Commands 26

COMMON PARAMETERS 32

- Job 33
- ProcessJob 39
- RenderJob 43
- RepeatMergeDistributor 45
- SequenceDistributor 47
- SliceDistributor 49

DYNAMIC PRODUCTS 50

- Product Editor GUI 50
- Command Line Product Control 54
- Classes 55

LEGACY DYNAMIC PRODUCTS 57

LEGACY MAYA PRODUCTS 58

LEGACY VIRTUAL MODULES 60

- Parameter Types 61
- Common Parameters 63
- Reference 65
- Example File 76

PRODUCT REFERENCE 82

- 3D Studio Max 83
- 3D Studio Max (Single Frame) 85

3Delight 86
3Delight for Maya 87
3Delight for Maya (Single Frame) 89
After Effects 92
Air 93
Aqsis 94
Alias 95
Arnold for Maya 96
Arnold for Maya (Single Frame) 98
Arnold Standalone 101
Blender 102
Cinema 4D 103
finalRender for Maya 104
FFmpeg 106
Fryrender 107
Fusion 108
Gelato 109
Generic Script 110
Houdini 111
Hrender 112
imgcvt 113
Indigo 114
Large File Transfer 115
Lightwave 116
Maxwell Light Simulator 117
Maya 119
Maya Hardware Renderer 121
Maya Lightmap Generator 123
Maya Software Renderer 124
Maya Software Renderer (Single Frame) 126
Maya to mental ray Exporter 129
Maya Vector Renderer 131
mental ray for Maya 133
mental ray for Maya (Single Frame) 135
mental ray Standalone 138
Mistika VR 139
Modo 140
Modo (Single Frame) 141
Nuke 143
Pixar RenderMan 144
REDline 145
Redshift for Maya 146
RenderMan for Maya 147
RenderMan for Maya (Single Frame) 149
Rendition 152
Thea 153
Turtle 154
Viz 155
V-Ray for Maya 156
V-Ray for Maya (Single Frame) 158
V-Ray Standalone 161
Vue 162
XSI 163

COMMON CLIENT SETTINGS 164
Common Client Command Line Options 164
Common INI File Options 168

SMEDGEMASTER REFERENCE 169
Command Line Interface 169
SmedgeMaster.ini options 171

SMEDGEENGINE REFERENCE 176

TIPS AND TRICKS 178

LEGAL 180

About IDs

Smedge uses an ID for nearly everything. These IDs are forms of a Universally Unique Identifier, or UUID. A UUID is a 16-byte (128-bit) number. The number of theoretically possible UUIDs is about 3×10^{38}. In its canonical form, a UUID consists of 32 hexadecimal digits. Smedge does not care if the hexadecimal letters (A-F) are upper or lower case. The digits are displayed in 5 groups separated by hyphens, in the form 8-4-4-4-12 for a total of 36 characters (32 digits and 4 hyphens). For example:

```
7582c8ad-cafd-4ac2-b5f0-a20101872527
```

Smedge uses the system run time libraries to generate these IDs. Any ID generated by Smedge is going to be unique to your system. You can also use other UUID generation tools, like those distributed with development tools or web pages for generating them, with confidence.

You can use the **`uidgen`** tool included in the Smedge distribution to generate IDs from a command prompt or script.

Variables that control Smedge functionality

These variables configure how Smedge itself operates. You must set these variables before starting a Smedge component process in order to have the value you have set recognized by the process.

SMEDGELIB — *API Only.* The base directory where the Smedge API is installed. For example, `C:\dev\Smedge`

SMEDGE_BIN — *Linux only*. The actual directory where the Smedge binaries are installed. Normally, the Smedge start scripts take care of this for you, but if you want to be able to override the installation folder (for example, if you want to test a new version without affecting the working installation), you can override the path to the binaries with the value of this variable.

SMEDGE_CLIENT_PRERESOLVE — Enables the Smedge 2014 and earlier address pre-resolution by the client. If you set this value to 1, the client will try to resolve its hostname to an IP address and use that for all communication. If disabled, the client will always use its hostname string instead. This can have performance impacts on some features if name resolution is slow for any reason. You can override the environment variable with the command line switch `-SetClientPreresolve`.

SMEDGE_DAEMON_PATH — *Mac daemon only.* Use this variable to override the location where the launchd plist file to control the daemon has been installed so that it can be properly shut down programmatically. If this variable is not set, the program will assume the plist file is in the default system daemon folder: `/Library/LaunchDaemons`.

SMEDGE_FIND_LOG — Allows you to find where the Smedge files are created. Set this variable to a path to a file path, and every Smedge component that starts on the machine will write the path where the log file is created to this file.

SMEDGE_MACHINE_FOLDER_BASE — Allows you to override the base folder where Smedge will read and write “machine” folder data. The component processes will still create their own folders under this folder,

so it can be useful for testing new versions or running parallel networks without affecting existing machine level data (from the Engine or Master, for example).

SMEDGE_MACHINE_LOGS Allows you to override the base folder where Smedge will write its own run-time log files. By default, this is in the local user directory for the user that started the component process. Using the environment variable will override the default, but using the command line flag `-LogFolder` will override the environment variable.

SMEDGE_MAX_THREAD_POOL Set this if you want to override the default maximum number of threads Smedge component processes can create for the main thread pool. The default maximum is 36 threads.

SMEDGE_MODULES This is a semicolon separated list of folders to look for Smedge .SX Modules. The folders will each be recursively searched by the Module Manager during startup.

SMEDGE_NO_WORK_ID_NOTE On the Master, set this to a non-zero integer to disable the feature in Smedge 2016 and later that sets the work unit note to the work unit ID when the work is created. Instead, the work will inherit the parent job's note when it is created. Set this variable to 0 or unset it to have the note changed to the ID.

SMEDGE_OPTIONS_PATH This is a semicolon separated list of folders to search when looking for INI files. The folders here will be searched before any default folders, but after any folders specified with the command line switches `-OptionsFolder` *folder*.

SMEDGE_PROC_AFFINITY *Windows* only. Set this to **true** to force the processor affinity to all available cores. This is needed on the 32 bit version running on machines with more than 32 cores. You can override the environment variable with the command line switch `-SetProcAffinity`.

SMEDGE_UMASK *Linux daemon only.* If you want to set a umask for the files created by Smedge when it is running as a daemon, you can set the umask value here. Smedge expects this value to be a 3 digit octal value corresponding to the bits you want umasked. For example: export SMEDGE_UMASK=022

SMEDGE_UNIQUE_ID This allows you to customize the unique ID value that is used to identify this machine in the system. If unset, Smedge will try to generate an ID that is tied to the hardware (derived from the primary network interface MAC address). If this environment value is set to a valid UID string, then the given value will be used as the ID of this machine. If the value of this variable points to a file with a valid 16 byte UID in it, the value loaded from the file will be used. If the value is set but is neither a valud UID string nor the name of

an existing and valid file with the 16 byte UID data, then a random UID will be generated. In this case, after generating the random ID, Smedge will try to save the generated ID as 16 byte ID data in the file named by the environment variable. Note that failure to save the ID will not result in Smedge failing to start. However, restarting the application will result in a new ID being generated. This can affect the ability for the machine to decrypt license codes, and may cause Engines to appear multiple times in the Engine list.

SMEDGE_USER

Linux daemon only. This is the user account you want the Smedge 3 daemon process to run as. ***Warning****:* if unset, the process will run as the user that started it. If you start it as root, then Smedge 3 Jobs will have potential root access to every machine on your network. **This could be very dangerous.**

SMEDGE_USER_FOLDER_BASE

Allows you to override the base folder where Smedge will read and write "user" folder data. The component processes will still create their own folders under this folder, so it can be useful for testing new versions or running parallel networks without affecting existing user level data (from the GUI, for example).

SMEDGE_WAKE_COMMAND

This is an optional command you can use to try to wake up engines over the network. By default, Smedge uses an internal system that tries to broadcast the wake request as a Wake-On-LAN UDP message. If you want to customize this operation, you can set this environment variable to a command string that will be used instead of the internal system. It understands the following parameters:

$(Broadcast*:A|B|C|subnet-mask*) A broadcast IP address derived from the $(IP) address

No options	`255.255.255.255`
A	xxx`.255.255.255`
B	xxx.xxx`.255.255`
C	xxx.xxx.xxx`.255`
subnet-mask	*the broadcast for the specified subnet*

if no IP address is known for the Engine, this will always have the LAN broadcast address of `255.255.255.255`.

$(Engine) Equivalent to $(Name)

$(ID) The ID of the Engine

$(IP) The last known IP address for the Engine, or the host name if there is no last known IP address.

$(MACAddress:*c*) The Engine's MAC address. The optional character c can be:

$(MACAddress)	`01-23-45-67-89-ab`
$(MACAddress:)	`0123456789ab`
$(MACAddress:-)	`01-23-45-67-89-ab`
$(MACAddress:.)	`01.23.45.67.89.ab`

$(Name) The name of the Engine

SMEDGE_WAKE_PORT

By default, Smedge sends its own wake broadcast on UDP using port 9. If you want to use a different port, specify the port number with this environment variable. Note that this variable is ignored if you use the SMEDGE_WAKE_COMMAND environment variable to override the wake system.

Variables set for work processes

SmedgeEngine will set some more environment variables when it starts up. These variables will be valid from the context of any operation performed by the SmedgeEngine process, or by any child process it starts. This includes any form of work, no matter what Job classes it may use to implement its functionality.

SMEDGE_ENGINE_ID	The ID of the Engine on which the work was started
SMEDGE_ENGINE_NAME	The name of the Engine on which the work was started.
SMEDGE_LOG_FOLDER	The Smedge machine logs folder for SmedgeEngine.
SMEDGE_PROGRAM_FILES	The location where the Smedge program files, including the SmedgeEngine executable, reside.

Smedge work that spawns a child process (anything that is derived from ProcessJob, technically) will create several additional environment variables that the work process can use to access information from the Smedge system without having to pass the data directly via the command line. See the ProcessJob parameters list for more information. These variables are only set for a child process started from ProcessJob::Execute. They are not available from the context of the SmedgeEngine process itself, unlike the variables above.

SMEDGE_COMMAND_LINE	The command line that was used to start the child process.
SMEDGE_JOB_ID	The ID of the parent Job
SMEDGE_JOB_NAME	The name of the parent Job.
SMEDGE_WORK_ID	The ID of the work unit.
SMEDGE_WORK_NAME	The name of the work unit
SMEDGE_WORK_PARAMETERS	Any custom parameters that you have exported from the ProcessJob $(EnvironmentParameters) parameter of the Job.

Products may set other customized variables required for normal operation.

Licensing

Smedge only requires a license for a machine that will be doing rendering. The master is in charge of distributing licenses, and will grant a license to an engine when it starts working. All of the other components, including the GUI, the Master, the command line tools and the supporting scripts and tools can be used on as many machines as you like without paying license costs.

There are two types of licenses available in Smedge: “Legacy” licenses and “Point” licenses. The legacy license allows unlimited work by an engine, and the point license is managed on a per-minute basis. You can mix both kinds of licenses in a single network. By default, engines can check out either type of license and will try for a legacy license first. You can customize the license request settings in the Configure Engine Settings dialog box, or from commands in the Engine menu or by command line.

Once an engine has checked out a license for one work unit started on it, more work can be sent to the engine without additional licenses required. While work is going, if an engine has a point license and the point expires, it will check out another point as long as it is still doing work. When the work finishes, if an engine has a legacy license, that license is released when the engine finishes working, but a point license will remain checked out to the engine until the point expires (one minute after it was granted).

Licenses can be installed using the Submit License dialog or the Configure Master command line shell. See the User Manual for more information on these tools. The Master will keep track of 3 different licenses: “permanent” legacy licenses (that never expire), “temporary” legacy licenses (which expire on a fixed date), and the per-minute points.

- If a temporary legacy license is installed and not expired, that count will be used for the legacy licenses available.
- If there is no temporary license, or the temporary license is expired, and if a permanent legacy license is installed, that permanent license count will be used, but only if the Engine connecting is running a version that is allowed before the support expiration date of the legacy permanent license.
- If there is no legacy license available, or if the Engines are running a later version than is allowed by the support expiration of the permanent legacy license, Smedge will still allow up to 4 machines to render on the legacy system at a minimum without additional licenses required.

Point codes expire 3 days after they are generated. Once you install a point code, you cannot install that code again. You can install the same legacy codes again if you need to. Codes must be generated using the current Master ID in order to be able to be properly installed on the Master.

For more information about license costs and options, please see our web site or contact license@uberware.net.

Restrictions

Smedge includes a restriction system that allows administrators to limit the SmedgeGui functionality for end users. For example, you may want to limit users to only being allowed to modify their own Jobs. The restriction system allows this.

You can configure Restrictions in the Configure Master dialog. For more information on how this window works and how to access it, see the SmedgeGui reference in the User Manual. Note that restrictions currently only affect SmedgeGui operation. Entering “Administrator Mode” will override any Restrictions set, and allow complete SmedgeGui functionality.

The following is a list of the currently available restrictions and what they do:

Change Job	Users cannot change any parameter of a job once it has been created.
Configure Connection	Users cannot change their local user preference for how SmedgeGui to connect to a Master .
Configure Connection.ini	Users cannot use SmedgeGui to configure the Connection.ini file for how any Smedge client application on the system can connect to the Master.
Core Process Control	Users cannot start, stop, install or remove the Master and Engine components from SmedgeGui.
Customize Views	Users cannot see or use the Customize Views menu item or GUI View Manager tab.
Delete Job	Users cannot delete Jobs from the system.
Edit Engine	Users cannot modify any Engine settings, Engine Product Options, or what Pools an Engine belongs to. Essentially, the Edit Engine window can no longer be accessed.
Event Commands	Users cannot see or modify the “Event Commands” tab of the Submit Job window or the Configure Engine windoow to set processes to be started during the Job life cycle.
Job Advanced Info	Users cannot see or modify the “Advanced Info” tab of the Submit Job window to set Advanced Job parameters.
Job Custom Pool	Users cannot see or modify the “Custom Pool” tab of the Submit Job window to customize which Engines can or cannot work on the Job.
Job Pool	Users cannot set or change the Pool to which the Job has been assigned.

Job Priority	Users cannot set or change the priority of any Job.
My Jobs Only	Users can only control or modify Jobs that they have created. This restriction will also remove the ability to customize the user name that is set as the Job "Creator", so that users cannot defeat the system just by changing the user name in the SmedgeGui options.
Process Control	Users cannot start any Smedge component processes, including Conspectus, Herald and Aegis, as well as the core processes. This restriction implies **Core Process Control**, even if that restriction has not been specifically set.
Reset Failures	Users cannot access the Reset failures commands to reset Job or Engine failure counts.
Save Default Engine	Users cannot save the "default" Engine.
Save Default Job	Users cannot save the "default" Job.
Stop Work	Users cannot stop work from any Job.
Submit Job	Users cannot submit new Jobs to the system using SmedgeGui.
VNC	Users cannot access the remote desktop viewing commands.

Default Restrictions

By default, Smedge starts with the following restrictions:

Core Process Control
Event Commands
Job Custom Pool
Save Default Engine
Save Default Job

Automatic Redundant Master

The central component of the Smedge system is the Master, which is in charge of keeping track of the Jobs and sending work out to the Engines as needed, as well as basic system operation. In client / server terminology, the Master component is the “server” part that the other components (the “clients”) connect to in order to send and receive data.

As the central manager of the system, this component is pretty important. It is essential that all of the machines on your network can establish a network connection to the Master in order to operate. At this time, you must connect directly to the primary Master, and you cannot use a redundant Master as a router. These are planned upgrades for a future release of Smedge.

Smedge does provide the ability to run the Master redundantly on several machines. The instances of the Master component that are running and not acting as the “primary” master, will instead act as “mirrors”. They automatically update all information and are ready to take over as the primary Master at any time, should the primary Master be stopped or go offline for any reason.

By default, Smedge handles this for you automatically when you start the SmedgeGui. The GUI will normally start an instance of the Master and an instance of the Engine with it. If this happens to be the first Master on your network, it will be the actual primary Master, controlling the whole system. If, however, a master is already running on the network, the redundant mirrors will simply copy all of the Master data and wait around to take over as needed.

Manually Configuring the Redundant Mirrors

As you add more machines, the overhead for keeping full copies of the data can get burdensome. And if your hardware and network are reliable enough, the redundancy can be unnecessary. You can disable the automatic redundant Master with the GUI with the following steps:

1. Choose System > Administrator Mode (if you're not already in Administrator Mode)
2. Choose System > Components > Set System Default Component Startup
3. Set the “Start the Master” option to “Do not start it”
4. Click OK

You may want to enable a few mirrors manually in order to provide the redundancy you need without overwhelming the system. You can manually configure the GUI to override the default component startup option set above in the SmedgeGui options dialog box on that machine. Again, you must be in Administrator Mode to access the option.

In more advanced installation, it makes more sense to run the SmedgeMaster component on the required nodes directly, and avoid using the GUI to reduce system overhead. The SmedgeMaster component can always be started directly by itself, in the standard ways for staring executables on each platform (e.g., on Windows, double click the SmedgeMaster.exe program file to start it, or use a shortcut, a batch file, or type the executable path into the Command Prompt, etc.).

The Master component process is also designed to be able to be run as a system service on all platforms, so you can run it with the system startup, pick the user account that you want to be running the process, and have your Smedge nodes available without even having to log into the console. Smedge includes tools in the Utilities folder of the program distribution that can aid in the installation of the services. Also see the SmedgeMaster reference, and your operating system reference for configuring and managing system services.

Automatic Master Location

If your entire Smedge network runs on a single subnet, and as long as there is no firewall blocking communication, Smedge will generally be able to automatically find the Master with no user interaction. However, if you want to use Smedge across multiple subnets, you will have to point Smedge to the Master. There are two ways you may want to do this, which are the two ways available in the Configure Connection dialog box.

In both cases, you can optionally specify one or more hosts and an alternate port for the Master. If nothing is set, the system will use the default automatic location system, which is a UDP broadcast on the subnet. If you specify one or more host names (or IP addresses), Each will be tried, in order. This will repeat indefinitely as long as the process is running and not connected to the master. You can add the special host name: * (a single asterisk character) to make an attempt using the automatic system.

If you specify a port, that port will be used to find the Master. Each instance of the Master only listens on a single port. Any client will only ever find and communicate with a Master using the same port. The default port is 6870. You can use alternate ports to set up unrelated Smedge networks, perhaps a maintenance network and a production network, on the same machines.

GUI Only (Not Engine or Master)

This sets the master GUI options for the current user account only. Any other components, even those started by the GUI itself, will not use the options set here. This is useful for quickly switching a GUI between separate, unrelated networks running on different ports without affecting other parts of the pipeline, such as submit or control scripts that are using the command line utilities.

Connection for this Machine

This tries to create or update the Connection.ini file in the Smedge program folder on the local machine. As such, any Smedge component that is started from that program folder will use the settings from this file (unless that process options have been specifically set to override the settings, using the GUI Only options). This is how you can specify at a machine level where to find the Master, and it will affect every component run from that folder, even if the GUI did not start the component directly. This will even affect any currently running components the next time they try to find the Master.

Once you have created a Connection.ini file, you can copy that file around, or put include it in a disk image or virtual machine. Also, see the chapter on the RLib INI File Syntax for information about staggered loading of INI files.

Automatic Engine Mode

To reduce the Smedge overhead when a machine is primarily being used for rendering and not as an active interface to the system, Smedge includes a feature called “Engine Mode” where the GUI disconnects itself from the Master, and shuts down any redundant backup Master running on the machine. In this mode, the machine runs with about the same amount of overhead as if you ran just the the SmedgeEngine component manually.

Automatic Engine Mode kicks in after a certain amount of time without any interaction in the GUI. The amount of time is configurable at the system level in the Administrator Options dialog box, and can also overridden for a specific machine in the SmedgeGui options. You can also manually enter Engine Mode using a menu command in the system menu.

If your computer's name includes one of the words: “render”, “node”, “blade”, or “smedge” and a number, or consists only of numbers, Smedge assumes that this machine is primarily meant to run as an Engine, and the GUI will start in Engine Mode by default the first time you start Smedge. You can always use the Connect menu command to run in normal mode.

For successive runs, GUI will run in the same mode as whatever the last mode it was in when it shut down. Note that the Master component is also stopped in Engine mode, unless your machine is the primary Master machine. When you restart a machine that was in Engine Mode, the Master component is not started with the GUI, and is only restarted when you try connecting if your machine was previously the Master when it last shut down.

In normal usage, this is not a problem. If you generally start the same machines up in the same modes, things will work consistently. If you end up with no Master on your network, you will need to manually start the Master on at least one machine. If that machine was not the master or an up-to-date mirror the last time it was running, you may have out of date job information. If you find this to be the case, you can stop that master and try starting it on a different machine that may have been up-to-date.

Automatic Engine Settings

The default Engine settings are designed to allow you to get up and running quickly with default installations of the software. However, Smedge allows extensive customization of the options for how various products work through the product options. The GUI and Engine command line shells both allow you to configure multiple engines at the same time, but you can only configure Engines that are currently connected to the Master.

You can override the defaults that will be used for any new Engines that you connect, so you do not have to manually configure them at all. To set the default, set up one engine as you want in the Configure Engine dialog box, then use the “Save as Default” button to have the settings saved as the new defaults. Any engines that are currently online will automatically take on these new default values for any options that were previously at the old default value. Any settings that had been customized from the defaults will remain. Additionally, any new Engines that come online will automatically download and use these default instead of the standard defaults that ship with the system.

You can also view and edit the Engine product default options in the Configure Master dialog, on the Product Options tab.

Automatic Executable Paths

Smedge requires the location of a third party executable program files in order to run jobs for most products. Because these executables are often located in predictable paths, Smedge is often able to find these executable automatically. If you have not set a path manually, and if no path option has been previously found, Smedge will try to find the executables when it starts up. This means that if you have installed and started Smedge, then install a rendering product like Maya, Smedge will find the Maya installation automatically.

Smedge will check no more than once per hour for a new installation. For the system to work, Smedge must know the common installation location. It cannot find executables outside of its known search paths. When making or customizing virtual modules, you can provide those paths yourself in a simple format in the Virtual Module INI file itself. See the Virtual Module reference section on FindLatestExecutable for more information.

If you set a custom path in the engine product options or engine default options, that path will be used first. If the given executable cannot be found, Smedge will fall back on the automatic location system to try to find the correct executable to use. You can see which executable is used at the top of the captured output from each work, and you can see if it is using a custom or an automatic path in the Engine's History.log file.

The system to find the latest executable is implemented in the API as a virtual method of the ProcessJob class, which can be overridden by the derived Job classes to customize how the search is handled.

On Windows, if you supply the default English program files folder (“C:\Program Files”) as part of the search , Smedge will automatically substitute the actual local program files folder, if you have customized its location or are using an alternate language version of Windows.

Automatic GUI Preset

Once you set up at least one preset with a set of views, you can select a preset that is applied by default to every GUI that connects to the system. Use the Administrator Options dialog to select the preset you want applied by default. Every GUI that connects will automatically apply this preset view set. If the ability for users to customize views is not restricted, they can still apply any other preset of views, or create or modify their own views. Any previous customized views will be lost when the new preset is applied, but all presets saved on the Master will still be available in the Customize Views manager.

RLib INI File Syntax

RLib INI files build on the basic section, key and value scheme. Section names and Key names are stored in a case insensitive manner, so watch out because “SECTION,” “Section,” and “section” will all refer to the same section.

Sections are delimited by text placed in square brackets:

```
[ Section Name ]
```

The square brackets will be removed, and any whitespace between the brackets and the text will be trimmed. Spaces are allowed inside the section name. A section name must be contained on a single line. If you have multiple sections with the same names, the values will be added to the same section, or will overwrite existing values with the same key.

Data is stored in sections in a key equals value format:

```
Key Name = Value
```

The equal sign character is the important token. Without this character, the data will fail to load correctly. Whitespace around the equal sign will be trimmed off, along with any whitespace at the end of the data. You can surround your value in quote marks, which will allow you to have actual whitespace as a value. If you do this, the quote marks will be removed. For example:

```
Space = " "
```

A line like this will set the value of a key called “Space” to be a single space character.

Unlike sections, keys can span multiple lines. Text that is found without an equal sign will be appended directly on the end of the last found value. No extra space is added.

You can insert comment lines by making the first non-whitespace character of the line be either the semicolon (**;**), pound sign (**#**), or single quote (**'**):

```
# This is a comment line!
; So is this!
' And this too!
```

RLib loads the whole file at once, and provides access to the data loaded through the IniFile class. This class will be documented more fully in the RLib API documentation. What this means, however, is that you cannot have multiple keys in the same section with the same name. The last one read will override any previous values.

Alternate file locations

In the Smedge cross-platform system library, files can use a system of alternate locations to allow you to create and change your default configurations more easily. The alternate file location system is built into the basic File management system of the cross-platform library that underlies all Smedge operation, but how it is used depends on your use of the Smedge API.

When you are trying to just open an file, if the file is not found in the directory specified, or if no directory is specified as part of the name, Smedge tries to find the file in other standard locations. The locations are searched in this order, and as soon as a file with the same name is found in any of those locations, that file is loaded, and the searching stops.

The folders searched are:

1. The folder specified in the file path, if any

2. The current user's options folder for the component application that is trying to use the file.
   On Windows[1]: **`C:\Users\`***User*`\AppData\Roaming\Uberware\`***Component*
   On Linux: **`~/.Smedge/`***Component*
   On Mac: **`~/Library/Smedge 3/`***Component*

3. The current machine's options folder for the component application that is trying to use the file.
   On Windows[2]: **`C:\ProgramData\Uberware\`***Component*
   On Linux: **`/etc/smedge3/`***Component*
   On Mac: **`/Users/Shared/Smedge 3/`***Component*

4. The Smedge 3 program folder.

5. Any folders specified with the `-OptionsFolder` *folder* command line flag. If more than one folder is specified, they are searched in the **reverse** of the order they appear after the command line flag.

6. Any folders specified in the `SMEDGE_OPTIONS_PATH` environment variable. If more than one folder is specified, they are searched in the **reverse** of the order they appear in the environment variable.

The first file found is the one read, and the search is stopped as soon as a readable file is found. Using the API, if the file is found in one of the alternate locations (any location from 2 through 6) it is possible to configure the file to copy the file from the alternate location to the originally specified path. See the API documentation and headers for more information.

[1] On Windows XP the path is:
`C:\Dcouments and Settings\`*User*`\Application Data\Uberware\`***Component*

[2] On Windows XP the path is:
`C:\Documents and Settings\All Users\Application Data\Uberware\`*Component*

Overloadable Options Files

The program options INI file has more advanced "overloading" functionality. Instead of loading the file only once, it actually loads the file multiple times from several locations, allowing you to specify common default option values that can be specifically overridden for a specific machine or user. This system is available to any file operation that uses the OptionsFile API system. Normally, this is limited to files that contain the options and settings for a component application.

This system reads the file in the following order:

1. Every folder specified in the `SMEDGE_OPTIONS_PATH` environment variable. If more than one folder is specified, they are searched in the order they appear in the environment variable.

2. Every folder specified with the `-OptionsFolder` *folder* command line flag. If more than one folder is specified, they are searched in the order they appear after the command line flag.

3. The Smedge 3 program folder

4. The current machine's options folder for the component application that is trying to use the file.
   - On Windows[3]: **`C:\ProgramData\Uberware\`***Component*
   - On Linux: **`/etc/smedge3/`***Component*
   - On Mac: **`/Users/Shared/Smedge 3/`***Component*

5. *For Shell components only:*
   The current user's options folder for the component application that is trying to use the file.
   - On Windows[4]: **`C:\Users\`***User***`\AppData\Roaming\Uberware\`***Component*
   - On Linux: **`~/.Smedge/`***Component*
   - On Mac: **`~/Library/Smedge 3/`***Component*

If a file with the same name exists in more than one of the locations, it will be read from each location where it is found. If an key is set in an earlier file, but not in a later file, then that key's value will be used as the "default" value. If the key exists in more than one file, then the file most recently read file will specify the key's value. Any values unspecified in any file read will use the built-in default values for whatever element of the Smedge system is trying to access the options (for example values specified for a Virtual Module will come from the Virtual Module definition file).

Note that the Master and Engine do not use the user's options folder for storing options. Master and Engine options are always stored for the machine as a whole, so location #4 is where the customized options are always stored.

[3] On Windows XP the path is:
`C:\Documents and Settings\All Users\Application Data\Uberware\`*Component*

[4] On Windows XP the path is:
`C:\Dcouments and Settings\`*User***`\Application Data\Uberware\`***Component*

.SJ Job Files

The Smedge Job (.SJ) job file is an RLib INI file that contains the information needed to specify everything about one or more Jobs. This file can be saved from SmedgeGui or using the Job command line shell, and can be used to submit or modify jobs using the Submit command line shell or by being loaded into SmedgeGui.

Each Job in the file is defined in its own section, where the section label is an ID for the Job. Note that when you submit jobs from .SJ files, some of the parameters may be forced to new values, including the ID itself. However, in the context of a single .SJ job file, the ID will be unique and can be used to build Job dependencies within the file.

Inside of the section, the Name = Value pairs specify all of the data for the Job. When you save a Job from one of the Smedge component applications, the file will have every known parameter for the type of job. However, when you create the files yourself, you do not need to include every single parameter. Any parameters you don't supply will have the system default value when you submit the file as a new job.

At the least you must supply the Type. Without a Type, Smedge does not know what type of Job (which Product) you are creating. In an SJ file, you must supply the Type as the ID. You cannot use the Product Name or Shortcuts.

For more information about the parameters available for each Product and which are required to get the Job to work correctly, see the Product Reference. The reference also shows the default type ID for all Products that are included with Smedge. Note that if you have customized Products using PSX files or Module INI files (see Dynamic Products, Maya Products, and Virtual Modules for more information), you can use the ID you specified to access the customized Products.

Variable Substitution

Smedge includes a powerful variable substitution system that is used in just about every component application and Module in some way. Most often, this system allows you to generate a command line to spawn a work process using data from the Job to replace variables in a standard command line syntax. However, the system is also used by the Herald to extract Job data when performing an action, and can be used for event commands attached to Jobs, and is used in many other places throughout the Smedge system.

The information in this section is primarily directed towards users that wish to create custom Virtual Modules. However, the parameter system is also used when you are creating Job Event commands, and is also commonly used in the Herald to allow access to information from the Job or Work that triggers the notification. For example, if you want to send an email when a Job finishes, you probably want to include the Job name in the email, and the Parameter system allows you to do that easily.

Note that the parameter and command names are not case sensitive.

Syntax

The syntax for accessing a parameter is to start with a dollar sign, then surround the name in parentheses. You can optionally attach one or more commands to perform on the parameter before it is substituted back into the resulting string. The full syntax is:

$(ParameterName.*Commands…*)

The entire text from the dollar sign to the closing parenthesis will be replaced with the actual value from the Job object for the parameter named. The value may come from one of many places depending on the settings of the Job parameter being accessed:

1. If the ParameterName names a Job parameter, and that parameter has a non-empty value for the specific Job being accessed, that value will be used.
2. If the parameter names a Product option for the application type accessing it (e.g. an Engine Option for access by the Engine), and that option has been configured in the options for that component application, that value will be used.
3. If the ParameterName names an environment variable, that variable's value will be used.
4. If there is a hard coded default value for the parameter, it will be used.
5. Otherwise, the variable (everything from the '$' through the last ')' character) will be removed.

You can access any parameter value using its internal name. Be aware that this process is recursive: if you access a parameter that has data that uses this syntax, that value will also be parsed and replaced with data before being replaced into the command line. Smedge will check for direct recursion, where the parameter name appears inside the parameter value itself. However, it is possible to set up a chain of two different parameters that become mutually recursive. This will crash Smedge. Avoid doing this.

You can set part of the variable substituted string to be dependent on a non-empty value found for a parameter by enclosing it in square brackets. The entire text inside the square brackets will be dropped if the parameter inside the brackets cannot be found or is an empty string. For example, say you have this:

```
[-left $(Left) ]
```

If a parameter named **Left** is found and has the value **100**, the block will be replaced with this text:

```
-left 100
```

If, however, the parameter **Left** cannot be found, or has is an empty string, the entire block will be left out of the formatted string.

See the Product Reference chapter for a complete list of all parameters that are available for every Product distributed with Smedge.

Parameter Commands

The optional commands will be processed after the value is determined and before it is substituted back into the string. Commands are denoted with a period and then the command name, one of the names in the table below. You can chain commands by adding another period and another command name. Commands are processed from left to right. All commands will be processed before the final result value is substituted back into the result value.

Some commands can take optional parameters. These parameters are separated from the command name by a colon character. For example, to adjust the padding size for the **pad** parameter command to be 6 digits of padding, you would use a command like this:

```
$(Frame.pad:6)
```

The characters in black are required syntax, `Frame` is the name of the parameter, `pad` is the command, and `6` is the parameter to the pad command. The entire structure is replaced by the 6 digit padded value of Frame.

If you need to include a period character into a parameter command's parameter, you can use the backslash to escape the period or surround the entire parameter command, including its own parameter, in quote marks.

These are the commands currently available. The command processing will happen regardless of the type of the parameter, so the result may be unexpected if you use it on the wrong type of parameter.

Command	*What it does*
+	Add a value to the current value.
-	Subtract a value from the current value.
*	Multiply the current value with a value.
/	Divide the current value with a value. If the divisor is 0, it will return the original value
i/ or /i	Same as / but uses integer math to do the division, so 5 / 2 = 2
%	Modulo of the current value divided by a value. If the divisor is 0, it will return the original value
^	Raise the current value by a power (X^Y). The parameter can be a number or it can be a string that is itself formatted as if it was "$(parameter)".
?	Conditional substitution. The current value is tested for boolean value. The parameter provides the alternates that are used depending on the result, separated by a | character.
=	Compares the current value with another and returns 1 if they are the same, and 0 if not.
!=	Compares the current value with another and returns 0 if they are the same, and 1 if not.
<>	Same as !=
<	Compares the current value with another and returns 1 if it is less, 0 if not. If it knows the parameter type, that type will be used, otherwise it will default to a string comparison.

Command	*What it does*
>	Compares the current value with another and returns 1 if it is greater, 0 if not. If it knows the parameter type, that type will be used, otherwise it will default to a string comparison.
<=	Compares the current value with another and returns 1 if it is less or equal, 0 if not. If it knows the parameter type, that type will be used, otherwise it will default to a string comparison.
>=	Compares the current value with another and returns 1 if it is greater or equal, 0 if not. If it knows the parameter type, that type will be used, otherwise it will default to a string comparison.
Abs	Return the absolute value of the current value.
Absolute	If the value refers to a file name or a relative path, it will make that path into the absolute path where the file will be created on your system.
Ciel	Rounds current value upwards to nearest integer: 3.3 → 4 -3.3 → -3
Contains	Checks if the current value contains the parameter value. Returns 1 if it does, and 0 if not.
ContainsOneOf	Checks if the current value contains one of a list of characters supplied in the parameter value. Returns 1 if it does, and 0 if not.
ContainsOneNotOf	Checks if the current value contains any characters other than those supplied in a list of characters in the parameter value. Returns 1 if it does, and 0 if not.
CopyLocally	Copies the file to a local temp folder and returns the path to the local copy. If the source file exists and either has not been copied, or is newer than the copy, it will be copied immediately upon the execution of the command. If the command is accessed as part of a parameter for a Job, all files copied will be deleted when the Job finishes. Otherwise the copied files will remain in the temp folder. The folder copied to is $TEMP/smedge3/LocalCopies/*job-id*.
CutRoot	Attempts to return everything except the root drive part of a full file path or directory
CutExtension	Attempts to return everything except the extension of a file path
Default	Replaces the current value of the given parameter with the default value for that parameter
Dequote	Makes sure that there are no double quotes around the value, even if there is a space in that value.
Display	*For "Choice" type parameters:* Replaces the current value of the given parameter with the user display string for the value type. For all other types, this command has no effect on the value.
Empty	1 if the current value is empty, 0 otherwise
End	Returns anything after the last – or , character, or the whole value if there is no – or ,
EndsWith	1 if the current value ends with the parameter, 0 otherwise.
Enquote	If there is a space character in the value, it will be surrounded by double quote marks, allowing the OS to treat it as a name with a space in it.
Extension	Attempts to return the extension part of a file path
File	Attempts to return the file name part of a full file path
FileSize	Size on disk (in bytes) of the file named by the current value
Floor	Rounds current value downwards to the nearest integer: 3.3 → 3 -3.3 → 4

Command	*What it does*
Format	Attempts to format the value as if it is an Rlib time value (count of milliseconds since year 0) By default it will format the value like 2013/09/14 06:35:13.041 You can customize the format by adding a format string with a :*string* using strftime formatting, with the addition of using %x for the milliseconds.
FormatDuration	Attempts to format the value as if it is a count of milliseconds specifying a duration of minutes and seconds to days and hours (similar to the formatting of the elapsed time in the GUI)
FormatSpan	Attempts to format the value as if it is a count of milliseconds using the following format specifiers: `%%` - Insert % sign `%D` - Total days in this span `%d` - Total days in this span (default `%2d`) `%H` - Total hours in this span `%h` - Hours in the current day (default `%02d`) `%M` - Total minutes in this span `%m` - Minutes in the current hour (default `%02d`) `%S` - Total seconds in this span `%s` - Seconds in the current minute (default `%02d`) `%X` - Total milliseconds in this span `%x` - Milliseconds in the current second (default `%03d`) You can insert an optional width and pad character between the % and the specifier: `%`[[0]*width*]*char* By default it will generate a time like: 0:01:34:12.205
Hex	Converts the value to a hexadecimal number
InternalSeparator	*For "Parameters" type parameters:* Replaces the current value of the given parameter with the internal separator string used by that parameter. For all other types, this command has no effect on the value.
IsDirectory	1 if the current value names an existing directory on disk, 0 otherwise
Left	Leftmost *N* characters, *N* supplied by parameter
LeftFirst	Leftmost characters up to the first occurrence of parameter
LeftFirstOneOf	Leftmost characters up to the first occurrence of any character in parameter list
LeftFirstNotOf	Leftmost characters up to the first occurrence of any character not in parameter list
LeftLast	Leftmost characters up to the last occurrence of parameter
LeftLastOneOf	Leftmost characters up to the last occurrence of any character in parameter list
LeftLastNotOf	Leftmost characters up to the last occurrence of any character not in parameter list
Length	Length of the current value (in characters)
Local	Checks the path to see if it can be converted to the local platform using the Path translation system.
MakeLower	Converts the value to all lowercase letters
MakeUpper	Converts the value to all uppercase letters
MatchRegex	1 if the current value matches the parameter regular expression, 0 otherwise.

Command	*What it does*
NiceName	Replaces the current value of the given parameter with the parameter's user readable name string
Pad	Converts the value to a padded, signed number. By default it will be padded to 4 digits, but you can override the pad size using the optional parameter command
PadUnsigned	Converts the value to a padded, unsigned number.
Path	Attempts to return the directory part of a full file path
Replace	Replaces characters in the string with other characters, using String::Replace()
ReplaceAny	Replaces any character from a list found in the string with other characters, using String::ReplaceAny()
ReplaceRegex	Replaces characters found using a regular expression match with other characters, using String::RegexReplace()
Right	Rightmost *N* characters, *N* supplied by parameter
RightFirst	Characters from the first occurrence of parameter to the end
RightFirstOneOf	Characters from the first occurrence of any character in parameter list to the end
RightFirstNotOf	Characters from the first occurrence of any character not in parameter list to the end
RightLast	Characters from the last occurrence of parameter to the end
RightLastOneOf	Characters from the last occurrence of any character in parameter list to the end
RightLastNotOf	Characters from the last occurrence of any character not in parameter list to the end
Root	Attempts to return the root drive part of a full file path or directory
Safe	Converts any characters that are unsafe for a file name into _ characters.
Separator	*For "Multi" or "Parameters" type parameters:* Replaces the current value of the given parameter with the separator string used by that parameter. For all other types, this command has no effect on the value.
Start	Returns anything before the first **–** or **,** character, or the whole value if there is no – or ,
StartsWith	1 if the current value starts with parameter, 0 otherwise.
SwapExtension	Replaces the file extension from the current value with the given parameter value
TranslateContents	This command will attempt to perform a text file copy and parse operation on the given parameter. If the parameter points to a readable file, and if you have configured path translations, the file will be read and copied to the job local folder. Each line in the file will be tested for a translatable root, and the path in that line will be translated. If the value is not a path to a readable file, if there are no path translations configured, or if an error occurs, the result value will be just the original value.
Trunc	Returns the whole number portion of a floating point value: 3.3 → 3 -3.3 → -3
Type	Replaces the current value of the given parameter with a string that labels the type of parameter this is
UnixEpoch	Converts a value from an Rlib time (milliseconds since year 0) to the standard unix time_t format (seconds since January 1, 1970).
WordUpper	Capitalizes only the first letter of each work (separated by spaces)

Additionally, you can specify a sub-field from a "Multi" or "Parameters" type parameter by name as a parameter command, and the value will be replaced by only the resulting value of that field or sub-parameter from the current full value. The names available for use this way will depend on the specific parameter you are accessing. Additionally, these commands can also be chained so that subsequent command will operate on that sub-field or sub-parameter value as if it was the original value used.

Examples:

To access the directory part of a scene parameter you could use this syntax:

```
$(Scene.Path)
```

To access the filename without the directory part, you could use this syntax:

```
$(Scene.File)
```

To access the start and end frames of a sub-range, you could use these:

```
$(SubRange.Start) $(SubRange.End)
```

This converts a plain filename into the absolute path to that file, and then puts quotes around it if that absolute path happens to have a space character anywhere in it:

```
$(Scene.Absolute.Enquote)
```

To get the Y value from the resolution sub-parameter of a Maya job's extra parameters as a 5 digit padded value:

```
$(Extra.-x.y.Pad:5)
```

To get the creation time for a job with custom formatting:

```
$(Created.Format:"Created %A, the %e day of %B, year of our lord %Y.")

→ Created Thursday, the 19 day of September, year of our lord 2013.
```

To do math (assume the $(CPUs) value is 4 and the $(PacketSize) value is 5)

```
$(CPUs.-:1)                          →    3
$(CPUs.+:PacketSize)                 →    9
$(CPUs.*:PacketSize.Pad)             →    0020
$(CPUs./:PacketSize.*:2)             →    1.6
$(CPUs./:"PacketSize.*:2")           →    0.4
$(PacketSize.%:CPUs)                 →    1
$(PacketSize.i/:CPUs)                →    1
$(CPUs.^:".+:7.-:PacketSize")        →    16
```

To use conditional substitution (using same values as above)

```
$(CPUs.?:CPUs|One per Engine)          →     4
$(CPUs.*:0.?:CPUs|One per Engine)      →     One per Engine
```

Compbine comparison and conditional with the ability to nest parameters for complex functionality:

```
$(Status.!=:7.?:|$(Scene.CutExtension)_$(SubRange.Start.+:1.Pad).$(Scene.Extension))
```

The blue command checks the value of the status variable. If it is any value other than 7 (the “successful” status value), this replaces the status value with 1, otherwise it becomes 0. Next, the red command uses the conditional command to choose between one of two options, separated by the red | character. The space between the : and the | is empty, so if a job was not successful, this would generate an empty string, but if it was successful, it will take the scene name, cut the extension, take the sub range, add one, and pad that to 4 digits, and add the scene extension, resulting in a new file name.

It is also possible to mix using quote marks and nested parameter names.

Common Parameters

The following tables present all of the parameters available in Smedge currently. Note that Smedge uses a "class hierachry" system to add Job functionality. See the reference for each specific Product to see which of these common classes are included for that Product. Authors of Compiled Modules, using the Smedge API, may not make use of the full class hierarchy, if they don't need some functionality, so some of these parameters may not be available for these custom Modules. When in doubt, check with the author of the custom Module. Also, check the Products chapter for more information about each specific Product that currently ships with Smedge.

Name is the parameter name you can use to access this parameter. This is the text that goes inside of the $(*Name*) syntax, or that you use as a parameter with the **Submit** commandline shell to submit jobs, in a –*Name* syntax (See the documentation for Submit in the User Manual for more information). **Type** is the type of value that Smedge expects for this parameter. See Parameter Types for more information about what the type means. **Get** means that you can use this parameter to get a value with the $(*Name*) syntax. **Set** means that you can set this value with the **Submit** command line shell (or programmatically with the Smedge API). **Meaning** gives a brief description of how the parameter will be used. **Default** is the value that will be filled in for the Job if you don't supply it at submission time. Parameters in red *must* be supplied when you submit the Job.

Job

The Job class provides the basic, common functionality for all jobs and workers. Every Job and Work unit has at least the Job information.

Name	*Type*	*Get*	*Set*	*Meaning*	*Default*
CPUs	Int	X	X	The number of CPUs/threads assigned to this Job	0
Created	Time	X	X	The time when the Job was created	*Set by Master*
Creator	Text	X	X	The Job's creator string value	*Set by Shell*
CurrentDate	Text	X		Gets the current date formatted as a string. You can optionally provide a formatting string for the date by putting a colon and then the formatting string before any parameter commands or the closing parenthesis. See the RLib Time API documentation for the time and date formatting syntax. The default is %d-%b-%y	
CurrentProcessID	Int	X		Gets the current process ID (assigned by the operating system) for the calling process.	
CurrentTime	Text	X		Gets the current time formatted as a string. You can optionally provide a formatting string for the date by putting a colon and then the formatting string before any parameter commands or the closing parenthesis. See the RLib Time API documentation for the time and date formatting syntax. The default is %H:%M:%S	
DeleteJobEvt	Text	X	X	The event command to execute when the Job is deleted. This command will have parameters expanded and then will be executed asynchronously by the Master.	
DispatchCPUs	Int	X	X	This is the count of CPUs that the work unit uses when using Processor based distribution. If the processor distribution is not enabled for the parent Job (CPUs = -1) this value will be 0 for the work, and CPUs will be the number of processors available on the Engine.	
EngineCleanupEvt	Text	X	X	The event command to execute on any engine that worked on this job when the job has finished completely. This command will have parameters expanded and then will be executed asynchronously on all Engines.	
Environment	Text	X		Access to environment variables. Specify what variable name after a colon to this parameter, like: $(Environment:HOME). As a shortcut, just just the colon: $(:HOME)	

Name	*Type*	*Get*	*Set*	*Meaning*	*Default*
ExcludeEngines	ID List	X	X	A comma separated list of Engine IDs to exclude from this Job. Any Engines listed here will never be allowed to work on this job.	
FailureLimit	Int	X	X	Override the maximum number of failures that this Job can have before no more work is distributed from it. Set to -1 to use the Master Maximum Job Failures limit.	-1
FinishDisposition	Choice	X	X	What should happen to the job when it finishes: **-1** = Let the Master determine what happens **0** = Keep the job until manually deleted **1** = Delete the Job immediately upon finishing	-1
FirstWorkEvt	Text	X	X	The event command to execute when the first work from this Job is started on an Engine. This command will have parameters expanded and then will be executed synchronously by the Engine.	
ID	ID	X		The Job's unique identification number	
IncludeEngines	ID List	X	X	A comma separated list of Engine IDs to include on this Job. If the Engine is listed here, it will be allowed to do work from this Job even if the Engine is not a member of the Job's Pool. You can also use this field to assign multieple Pool IDs for the Job.	
JobAssignWorkEvt	Text	X	X	The event command to execute when the Master is about to assign work to an Engine. This command will have parameters expanded and then will be executed synchronously by the Master.	
JobFinishedEvt	Text	X	X	The event command to execute when all work from a Job has finished or been permanently canceled. This command will have parameters expanded and then will be executed synchronously by the Master.	
JobFirstStartedEvt	Text	X	X	The event command to execute when the first work from this Job is started. This command will have parameters expanded and then will be executed synchronously by the Master.	
JobLocalFolder	Dir	X		The Job specific local temp folder where files are copied by the **CopyLocally** parameter command	

Name	*Type*	*Get*	*Set*	*Meaning*	*Default*
JobSummary	Text	X		This is a nicely formatted summary of the job for display-only. Note that the exact format of this value will depend on the type of Job (primarily on the type of distributor actually). See SequenceDistributor and SliceDistributor for examples of how this parameter is used.	
LogDir	Dir	X		The application's log directory	
MachineDir	Dir	X		The machine-wide options folder	
MachineName	Text	X		The name of the local computer	
MachineNumber	Int	X		A number hash based on the local computer's name (there is a very small, but non-zero chance that two different machines' names will hash to the same value).	
Name	Text	X	X	The Job's name	
Note	Text	X	X	The Job's note	
OvertimeKill	Float	X	X	If a worker goes this many times over the average time for workers from this Job, the worker will be timed out and requeued. (Set to 0 to disable).	15
Parent	ID	X		The Parent Job's identification number	
ParentName	Text	X		The Parent Job's name, if it can be found	
PercentDone	Float	X		The percentage complete of the Job. The value actually comes from the Job's distributor, so exactly how it is calculated varies by the different types of Jobs and their different distributors.	
Pool	ID	X	X	The Job's pool. Get will always return an ID, but when you set the value, you can use either the ID or the name of the pool.	Whole System
Priority	Int	X	X	The Job's priority value	50
PriorityBoost	Int	X	X	The number of workers from this Job to prioritize over other existing jobs. The Job's PriorityBoost value will override the value set for the Product in the Master options, which will override the global boost you can set for all jobs.	
ProductID	ID	X		The "Product" ID. For a Job, this is the Type. For Work, this will be the Type of the Job. It will always correspond to a known product ID from the list of Products or a custom Product you created.	
RAM	Int	X	X	The amount of memory assigned to this job	0
SmedgeDir	Dir	X		The Smedge program folder	
StaggerCount	Int	X	X	The stagger start count of workers to start at once	1

Name	*Type*	*Get*	*Set*	*Meaning*	*Default*
StaggerStart	Float	X	X	The number of seconds to stagger the worker starts (0 disables the stagger start system, so work starts as soon as possible)	0
Status	Int	X		The Job's current status code. Built in codes: **-1** = Paused **0** = Pending **1** = Assigned (*work only*) **2** = In progress **3** = Changed (*work only*) **4** = Work Aborted (*work only*) **5** = Work Unsuccessful (*work only*) **6** = Work Lost (*work only*) **7** = Complete **8** = Canceled **9** = Failed **10** = Timed out (*work only*) **11** = Restarted (*work only*) **12** = Orphaned (*work only*) **13** = Administrator Abort (*work only*) **14** = Administrator Cancel (*work only*)	
StatusAsString	Text	X		The Job's current status as in a human readable format	
SystemID	ID	X		The unique system ID.	
TempDir	Dir	X		The system's TEMP directory	
Type	ID[1]	X	X[2]	The Job's type identification number	
TypeString	Text	X		The Job's type's Name string	
UsageLimit	Int	X	X	The limit for this job to have outstanding	-1
UserDir	Dir	X		The user's options folder	
WaitForJobID	ID	X	X	The ID of another Job in the system that the current job must wait for. As long as a Job with the given ID exists and has pending work, no work from this job will start.	

[1] When setting the Type using the Submit shell, you can generally use any number of possible values, including the ID, the Product's Name, or any of the defined Shortcuts for that Product. See the Products chapter for a reference of all values you can use for every Product currently distributed with Smedge.

[2] Once a Job has been created, you cannot modify the Type.

Name	*Type*	*Get*	*Set*	*Meaning*	*Default*
WaitForWholeJob	Bool	X	X	If the ID of another job is supplied in **WaitForJobID**, this option determines if the whole job must be complete, or if the distributor is allowed to start sending work when the waited for job is still only partially complete. Exact implementation depends on the distributor.	Yes
WorkAssignedEvt	Text	X	X	The event command to execute when the Master has asked the Engine to start work from a Job. This command will have parameters expanded and then will be executed synchronously by the Engine.	
WorkEngine	ID	X	X	The ID of the Engine that is assigned to perform the Work. NULL if the work has never been assigned to an Engine or for Parent Jobs.	
WorkEngineName	Text	X		The name of the Engine that is assigned to perform the Work. Uses the application's GetEngineByID() method to get Engine data associated with the ID from **WorkEngine**. Returns "No Engine" for NULL IDs, and the ID as the name if the name cannot be found from the application.	
WorkFinished	Text		X	This is a special parameter that can be used to broadcast some information associated with the work finishing. Job classes use this internally to pass data about the work that finished back to the parent Job. Derived Job classes are free to use this in whatever manner they wish. By default it is simply ignored by the Job, but may be used by the standard distributors to help keep track of available work. See SequenceDistributor and SliceDistributor for examples of how this parameter is used.	
WorkFinishedEvt	Text	X	X	The event command to execute when the Engine has finished a work unit. This command will have parameters expanded and then will be executed asynchronously by the Engine	
WorkFinishedSuccessfulEvt	Text	X	X	Called immediately after **WorkFinishedEvt** for successful work. This command will have parameters expanded and then will be executed asynchronously by the Engine	
WorkFinishedUnsuccessfulEvt	Text	X	X	Called immediately after **WorkFinishedEvt** for unsuccessful work. This command will have parameters expanded and then will be executed asynchronously by the Engine.	

Name	*Type*	*Get*	*Set*	*Meaning*	*Default*
WorkParameter-ChangedEvt	Text	X	X	The event command to execute when the Engine has detected a change in a parameter for a currently executing work unit. This command will have parameters expanded and then will be executed asynchronously by the Engine.	
WorkPostExecuteEvt	Text	X	X	The event command to execute after the work executes but before the result is sent to the Master. This command will have parameters expanded and then will be executed synchronously by the Engine.	
WorkPostExecuteSuccess-fulEvt	Text	X	X	Called immediately after **WorkPostExecuteEvt** for successful work. This command will have parameters expanded and then will be executed synchronously by the Engine.	
WorkPostExecuteUnsuc-cessfulEvt	Text	X	X	Called immediately after **WorkPostExecuteEvt** for unsuccessful work. This command will have parameters expanded and then will be executed synchronously by the Engine.	
WorkStartedEvt	Text	X	X	The event command to execute when the Engine has accepted a work assignment and is about to start working. This command will have parameters expanded and then will be executed synchronously by the Engine.	

ProcessJob

ProcessJob provides the data and services necessary to have a Product that performs work by launching a separate process to do the work. This is how most Products work in Smedge. The alternative is to have the SmedgeEngine process actually do the work itself as code directly compiled into the Module. For example, the Large File Transfer job actually does not use a separate process to do the work, but simply does the requested operation directly in SmedgeEngine.

Name	*Type*	*Get*	*Set*	*Meaning*	*Default*
CheckReturnValue	Bool-Override	X	X	Allows you to bypass the check on the code returned from the spawned child process. When enabled, any non-zero return value is interpreted as an error, resulting in the work being requeued.	Engine Default (yes)
DetectErrors	Bool-Override	X	X	Determines if the process output is monitored for error messages	Engine Default (yes)
ElapsedProcessTime	Int	X		The total elapsed processor time (in milliseconds) of the process when it has finished (as reported by the OS). Only available after the process has finished.	
ElapsedRealTime	Int	X		The total elapsed real time (in milliseconds) of the process when it has finished (calculated: time at finish - time at start). Only available after the process has finished.	
EnvironmentParameters	Text	X	X	This value will be expanded with any data from the Job and exported to the environment as the **`SMEDGE_WORK_PARAMETERS`** environment variable. **Note:** If this value expands to an empty string, the environment variable will not be set (or unset)	
ErrorIgnores	TextList	X	X	A list of strings to look for in a line of output that has been detected as an error by the containing one of the ErrorStarts values. This allows you to specify errors that can be safely ignored by the system.	
ErrorStarts	TextList	X	X	A list of strings to look for in a line of output that can be used to detect that an error has occurred in the processing of a work unit. If any of the elements of this list is found in a line of output and none of the **ErrorIgnores** values are also found in that line, the work is assumed to have failed, and it will be immediately terminated and requeued for later processing.	

Name	*Type*	*Get*	*Set*	*Meaning*	*Default*
Executable	Path	X	X	The rendering executable program file. You normally won't need to access this yourself. It's handled automatically by ProcessSequence.	
IdleTimeout	Float	X	X	The number of seconds that a spawned child process is allowed be running without consuming any CPU resources before it is considered timed-out, and requeued.	300
LimitMemoryUsage	Bool-Override	X	X	Determines whether Smedge should set a hard limit on the amount of memory that the process can access based on the RAM value (see above). If memory distribution is enabled, and the job RAM value is a positive integer, and this value is on (either for the Job or if the job is at the Engine Default setting, for the Product options on the Engine), then the process will be limited to the number of megabytes of memory specified. If it attempts to use more than that, the process may fail.	Engine Default (no)
MinimumTime	Float	X	X	The minimum amount of time (in seconds) for a work run before it can be considered successful. If the process takes less time than this, it is assumed to have failed, even if no error message is reported and the result code from the process is 0.	-1 (not used)
OutputLogFile	File	X		The full path to the saved captured output file for the process. This path is relative to the Engine doing the work, and may not be correct on any other machine.	
OutputPath	Dir	X	X	The directory in which saved captured output files will be saved.	
OutputPeer	Text	X		The IP address and port where the work is serving the output from.	

ProcessPriority	Choice	X	X	One of:
				`Normal Priority`
				`Low Priority`
				`Idle Priority`
					Normal Priority

Password	Password	X	X	The password that will be used to gain access to requested resources, if needed.	
ReportIgnoredErrors	Bool-Override	X	X	Determines whether SmedgeEngine will report errors that it ignores back to the master as part of the job history	Engine Default (yes)
Resources	TextList	X	X	*Windows only.* A semicolon separated list of Drive=Share pairs that the work unit will try to ensure are available before starting work.	

Name	*Type*	*Get*	*Set*	*Meaning*	*Default*
ShowProcess	Bool-Override	X	X	Determines whether the SmedgeEngine should try to spawn the process visibly, instead of hidden from view. This may not have an effect if the spawned process does not have any kind of user interface, or if SmedgeEngine is running as a background process on the machine.	Engine Default (no)
StartupMessage	Text	X	X	If not blank, the work will be requeued if the startup message is not detected within the timeout period. See **StartupMessageType** for how this test is made.	
StartupMessageTimeout	Text	X	X	If set, the timeout in seconds for the work to wait for the startup message. If the message is not detected before this timeout, the work is requeued.	
StartupMessageType	Choice	X	X	How the startup message is detected. Choices are: **Match** Exact string match **Start** The output line starts with the message **End** The output line ends with the message **Contains** The output line contains the message **Regex** The output line matches the message as a regular expression (you may need to escape special characters)	Contains
SuccessMessage	Text	X	X	This is an alternate form of error detection. If a string is provided here, the work will only be considered successful if this string is detected in the output from the process. If this string is empty, no test is made. See **SuccessMessageType** for how this test is made.	
SuccessMessageTimeout	Text	X	X	A number of seconds after detecting the success message for the process to terminate. If it has not terminated on its own before this timeout, the process will be terminated but still considered successful.	
SuccessMessageType	Choice	X	X	How the success message is detected. Choices are: **Match** Exact string match **Start** The output line starts with the message **End** The output line ends with the message **Contains** The output line contains the message **Regex** The output line matches the message as a regular expression (you may need to escape special characters)	Contains

Name	*Type*	*Get*	*Set*	*Meaning*	*Default*
StartDirectory	Dir	X	X	The directory that will be made the currently active directory when the process is started.	
Username	Text	X	X	The username that will be used to gain access to requested resources, if needed.	
LastError	Text			The last error detected by jobs/work and assigned as the final note to the JobHistory on Failure	

RenderJob

RenderJob is designed around the concept of a Job that renders an image or image sequence from an image processing or generation tool. It is optimized for 3D design and animation and 2D compositing tools, but it can be used to manage most related systems as well, such as file conversions, cache export, simulations, etc. The terms may differ, but the concepts of these processes are similar, and can fit into this paradigm.

The key elements it provides are the concept of the "scene" that is being rendered, and the "images" it produces as a result. It also provides some common error detection associated with the operation of the process and the results it produces. The process will be spawned with parameters to process the "scene" using the distributor's sequencing information, and will make the "image" results available in the system, if it can.

Name	*Type*	*Get*	*Set*	*Meaning*	*Default*
CheckImages	Bool-Override	X	X	An option to have the Job try to validate any detected frame filenames before reporting a successful completion.	Engine Default (yes)
DeleteBadImages	Bool-Override	X	X	An option to have any image files that are detected but that fail the image check test (if enabled, see **CheckImages**) deleted at the end of the work unit.	Engine Default (no)
DetectImageFormat	Bool-Override	X	X	An option to have the Job try to automatically detect the image formats from detected rendered image filenames.	Engine Default (yes)
ImageDir	*Dir*	[7]	[7]	*If it exists in a derived class, this value will be prepended to any detected image filename if they are not already absolute. If this value is empty or cannot be found, RenderJob will try to prepend the start directory to any relative filenames. This parameter does not exist in this class, but will be used if it exists in your Virtual Module.*	[7]
ImageFile	File	X	X	Get will return the last detected image file, Set will append the given file to the list of detected files, and will try to adjust the **ImageFormat** and **ImageDir** if possible.	
ImageFileList	Text	X		Returns the entire list of detected image files, enquoted and separated by spaces. Useful, for example, for passing the entire list of image files to another application, for display or compositing.	

[7] Access and default depend on the implementation in the derived Product.

Name	*Type*	*Get*	*Set*	*Meaning*	*Default*
ImageFormat	TextList	X	X	A list of the filename path and formats. The format string is kept in printf format, where the frame range is replaced with a formatting string like %d. This parameter is used by Smedge Shells, but its value does not affect the actual rendering output. You must supply parameters specific to the individual Products to actually configure how the product will format the rendered frame files.	
Scene	File	X	X	The scene file that the user selected to process.	
SequenceName	Text	X		Always returns the actual text **`$(SequenceName)`** as a string. This is used internally by Shell programs as a placeholder for the view sequence command.	
TranslateSceneContents	Bool-Override	X	X	An option to have the job try to translate the scene content script for any paths it finds using the Smedge path translations that may have been configured. Note that this is disabled by default because it will likely fail with any binary format scene file. Products that use text format scene files will work. Other products use at your own risk.	Engine Default (no)

RepeatMergeDistributor

RepeatMergeDistributor is a Job Distributor. This is the component of a Job that the Master uses to actually divide the job up into work units to be distributed to the Engines. RepeatMergeDistributor distributes work as a number of separate repetitions of the same render that can then be merged together. This is used, for example, to control single frame rendering for the light simulation type renderers, like Maxwell. Each repetition of the render is given a unique seed value and then the resulting rendered frames from each machine are merged together to produce a higher quality result than can be achieved by a single machine in the same amount of time. The distributor requires that the Job type is based on RepeatMergeJob, as the Job and Distributor must work together to properly coordinate the work.

Name	*Type*	*Get*	*Set*	*Meaning*	*Default*
ClientL	ID	X	X	For merge work units, this will be the ID of the local client that has one of the files to be merged.	
ClientR	ID	X	X	For merge work units, this will be the ID of the remote client that has one of the files to be merged	
FileL	ID	X	X	For merge work units, this will be the ID of the local file to be merged.	
FileR	ID	X	X	For merge work units, this will be the ID of the remote file to be merged	
Frame	Int	X	X	The frame number for the work unit. Frame 0 is a special value used for “single frame” type jobs	
MergeExecutable	File	X		Engine option for the merge executable path	
Mode	Choice	X	X	Distribution mode for repeating frames. One of: `0` Dispatch all repetitions from each frame `1` Dispatch every frame once then repeat	1
Output	File	X	X	The full path and filename of the rendered image file	
RealFileL	File	X		For merge work units, this is the full path and filename of the local file to be merged	
RealFileR	File	X		For merge work units, this is the full path and filename of the remote file to be merged after it has been downloaded	
RealOutput	File	X		The full path and filename for the output for this work unit.	
Repeat	Uint	X	X	For parent jobs, this is the number of repetitions for each frame. For work jobs this is the repetition number for this particular work unit.	1

Name	*Type*	*Get*	*Set*	*Meaning*	*Default*
Seed	Uint	X	X	The unique seed value for a given work unit to ensure that it can be correctly merged (renderers may have their own name for this value, like CPU ID)	
WorkType	Choice	X	X	The type of work object that this is. One of: `0` Parent Job `1` Render Work `2` Merge Work	

Custom Commands

The RepeatMerge system also adds some additional commands that you can use to control the rendering process.

Stop and Merge This will stop a current running render, and allow it to be merged immediately at its current quality level. Note that this only affects the specific work unit(s) you have selected when you request the command, and won't stop any additional renders from starting in order to finish all of the requested repetitions.

Stop All Work and Merge This will stop all currently running renders and start the merge process for all of them. Additionally, no further renders will be started, even if there are outstanding repetitions that haven't been rendered yet. You can access this command by selecting any work unit from the Job, the Job itself, or any history element from the job.

SequenceDistributor

SequenceDistributor is a Job Distributor. This is the component of a Job that the Master uses to actually divide the job up into work units to be distributed to the Engines. SequenceDistributor distributes work as subsets from a range. In Smedge, this generally means a range of frames to render, and the subset is the number of frames that will be rendered at one time by one machine in one work unit. Note, however, that SequenceDistributor simply uses the set range you supply, and does not know or care about how that set range actually applies to the product in question.

Name	*Type*	*Get*	*Set*	*Meaning*	*Default*
ActualPacketSize	Int	X		In use, a work unit have fewer frames than the Job PacketSize requests. This parameter will return the actual number of frames in a specific work unit. This value will be between 1 and the PacketSize	
CustomRange	Text	X	X	A user customizable range value for the Job that is not used as part of the distribution of the Job. It is used by some Shells such as CheckFileSequence to allow you to override the range of frames to look for.	
CustomRangeCount	Int	X		The count of items in the CustomRange value (if any)	
DistributeMode	Choice	X	X	How the items will be distributed from the range: **0** = Default (determined by Master option) **1** = Forward **2** = Reverse **3** = Sample	0
JobSummary	Text	X		The range of the sequence.	
PacketSize	Int	X	X	The packet size used by the parent Job to break the work up. Note that there may be fewer frames in the range than the packet size allows, if the division came to the end of a sequence.	1
Range	Text	X	X	The entire range of the parent job. This can be a complex range string including both – and , to separate elements.	
RangeCount	Int	X		The count of items in the Range	
RenumberBy	*Text* [3]	[4]	[4]	*If it exists in a derived class, this value will be used to correctly calculate the renumbering. This parameter does not exist in this class, but will be used if it exists in your Virtual Module.*	[4]

[3] The type currently needs to be set to Text to work correctly, but should generally be used as an integer value

[4] Access and default depend on the implementation in the derived Product.

Name	*Type*	*Get*	*Set*	*Meaning*	*Default*
RenumberStart	*Text* [3]	[4]	[4]	*If it exists in a derived class, this value will be automatically updated for each worker to correctly allow you to renumber the sequence. This parameter does not exist in this class, but will be used if it exists in your Virtual Module.*	[4]
SequenceBy	*Text* [5]	[4]	[4]	*If it exists in a derived class, this value will be used to correctly calculate the renumbering. This parameter represents a different sequencing than just 1 at a time. For example, you may want to render every other frame from a sequence. To do so, add a parameter to your Job class with this name. This parameter does not exist in this class, but will be used if it exists in your Virtual Module.*	[4]
SubRange	Text	X		The entire range of the work, formatted as a string in **start-end** format. There is no space in the formatted range. If the work has only one frame, it will be returned the same as if you used **SubRange.Start** or **SubRange.End**.	
UserRange	Text	X		If the **CustomRange** is not empty, it returns that value, otherwise it returns the **Range** value.	
UserRangeCount	Int	X		The count of items in the range returned by **UserRange**	
WorkFinished	Text		X	The value should indicate the range of the work that has just finished.	

[5] The type currently needs to be set to Text to work correctly, but should generally be used as an integer or floating point value

SliceDistributor

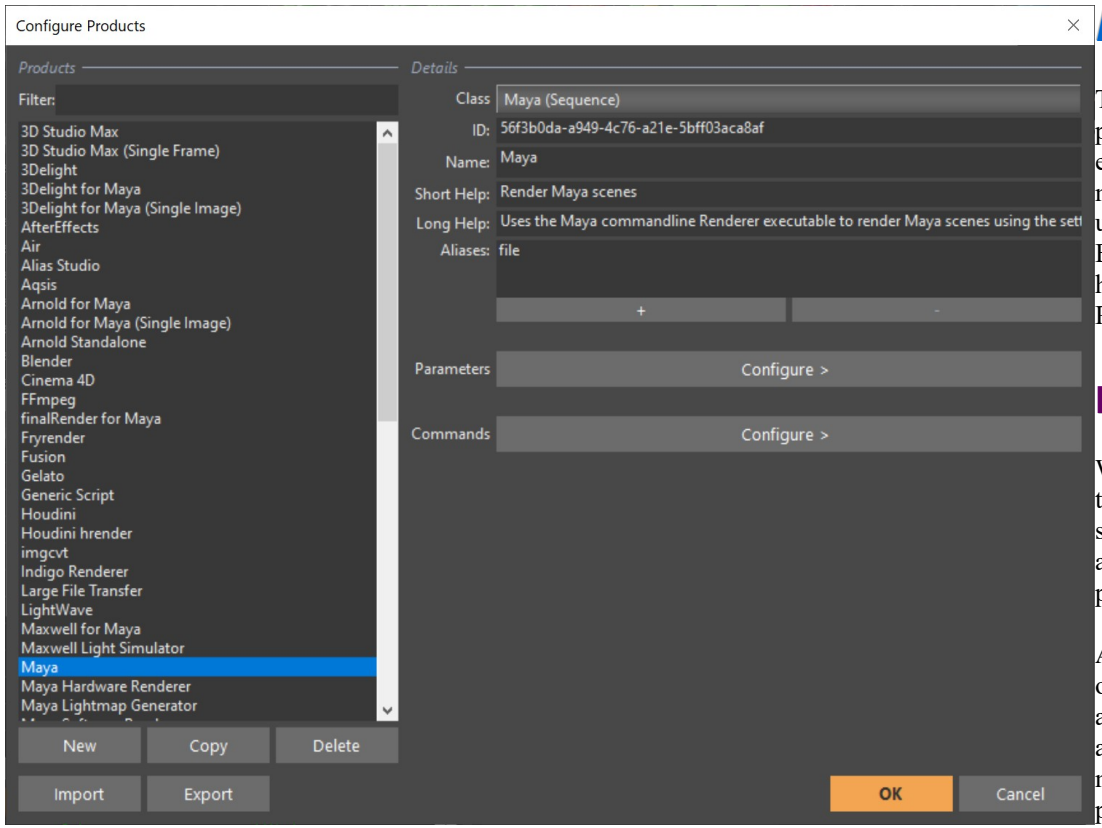
SliceDistributor is a Job Distributor. This is the component of a Job that the Master uses to actually divide the job up into work units to be distributed to the Engines. Slice distributor is designed to divide a single frame render into slices that are then combined back together into the final result. It distributes a worker for each slice, then a final worker that does the combination processing.

Name	*Type*	*Get*	*Set*	*Meaning*	*Default*
JobSummary	Text	X		The repetition of the child Work or the number of slices of the parent Job	
Slice	Int	X	X	The repetition number of a specific child Work	
Slices	Int	X	X	The number of slices to divide the parent Job into	
WorkFinished	Text		X	The value should indicate the repetition of the work that has just finished.	

Dynamic Products

Smedge 2020 and later include a full dynamic product system that allows you to create, edit, and remove products from the system while it is running, without any interruptions. You can access this system in the GUI using the SmedgeGui menu command System > System Commands > Configure Products, or you can use the ConfigureMaster command line tool to import, export, or remove products.

While Smedge does still support the legacy dynamic products systems, these are only useful when no other customizations have been made with the new system. When using the new system, the Master will store the products in the Data folder using a proprietary binary format. However, you can easily import and export products using a JSON file, allowing great flexibility for how you can use this sytem.



Product Editor GUI

The product editor interface has 3 modes, depending on what part of the product you are editing. You can edit the global product information, or information about one of the product's parameters or about one of its commands. Exactly what commands you have available and how they work depending on if you are editing Product, Parameter, or Command info.

Product

When you first start the editor, you will be in the Product editing mode. In this mode, you will see a list of all available products on the left, and selecting one will show you the detailed product settings for that product on the right.

Add, **Copy**, and **Delete** will allow you to create or remove products from the list. **Add** will make a generic new product, where **Copy** will clone all details of an existing product. **Delete** removes a product. Use **Import** and **Export** to get product information to and from a JSON file.

You can edit almost any of the product information. If you change the product name, you will see that change in the list on the left, but no changes are ever sent to the system until you press the OK button. Note that you cannot edit the ID field. ID's are generated for you dynamically when you generate new products, and this field is available for you to see and copy the value if needed.

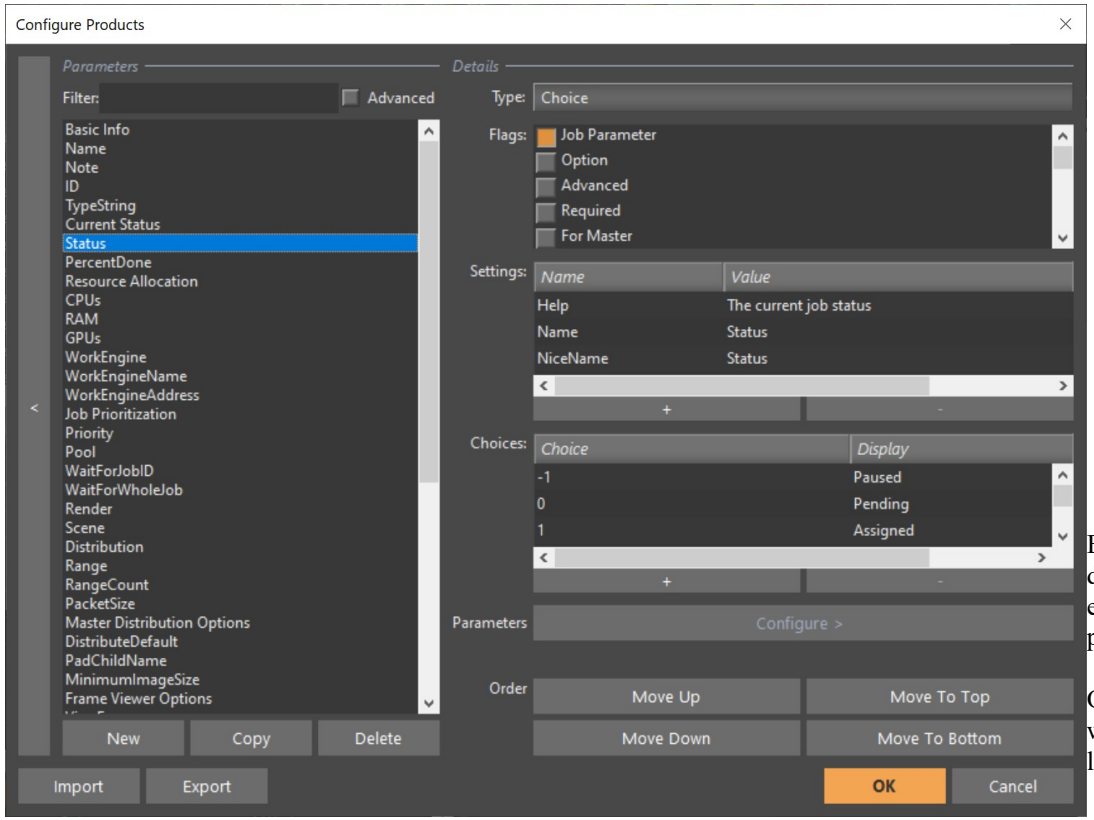
Class	The internal type of product. These correspond to the API classes that provide functionality and default configuration. The selection is a string name based on the values provided by SmedgeLib.
ID	The ID of this product. This value is read-only
Name	The name of this product.
Short Help	The short help string of this product.
Long Help	The long help string of this product.
Aliases	A list of aliases for this product. Use the + and – buttons to add or remove these
Parameters	Allows you to switch to edit the parameters for this product. Press the button to change mode.
Commands	Allows you to switch to edit the commands for this product. Press the button to change mode.

The class defines basic operation and will set up a different set of default parameters based on that operation. For a list of Classes and information about them, see Classes.

Parameters

(See next page for an image.) In this mode, you can see and configure all of the parameters associated with the product in the list on the left, and edit the details of any specific parameter with the controls on the right. **New** will create a new generic parameter, **Copy** will clone an existing parameter. **Delete** removes a parameter. Use the large button marked **<** on the left to go back to the product information, or up one level of nested parameters.

Type	Select what type of parameter this is.
Flags	Settings about how the parameter is displayed, used, and interpreted.
Settings	A key/value list of settings for this parameter.
Choices	A key/value list of choices this parameter can present and understand.



Parameters If the parameter can have sub-parameters, press this button to switch the Parameter being edited to allow configuring the sub-parameters.

Order The order parameters appear in the list also affects the order they appear throughout the interface. Use these buttons to re-order the parameter being edited.

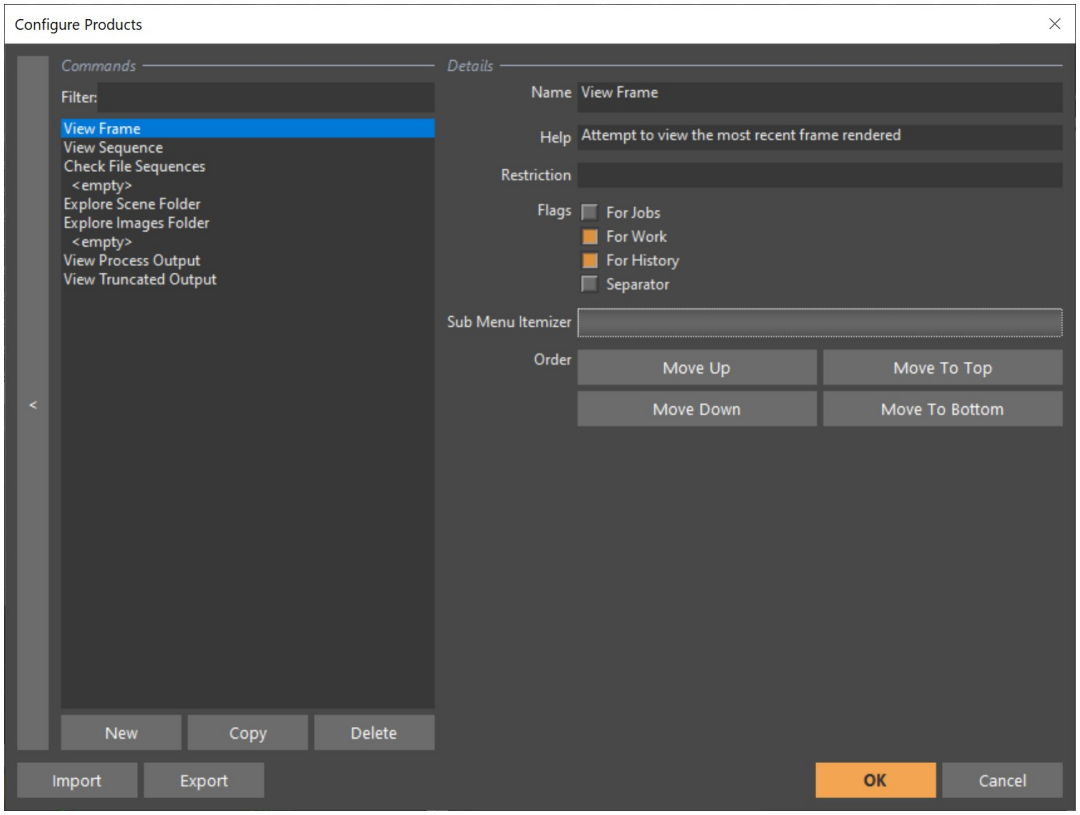
Each type of parameter will recognize and use different settings and choices. Only the "Parameters" type will allow you to add additional sub-parameters.

Often the simplest way to get what you want will be to copy an existing parameter that works like you want and modify it as needed.

Commands

(See next page for an image.) In this mode, you can see and configure all of the commands associated with the product in the list on the left, and edit the details of any specific command with the controls on the right. **New** will create a new generic command, **Copy** will clone an existing command. **Delete** removes a command. Use the large button marked < on the left to go back to the product information, or up one level of nested parameters.

Currently, all commands are built directly into the library and available by name, and the functionality depends on some of the settings being configured correctly. It is recommended that you currently do not modify the command name or sub menu itemizer.



Name The command name. This appears in the menu in the GUI, and is also the name used by the library to run commands.

Help A help string that can show up in the GUI in the status bar when the menu is open

Restriction Add a Restriction name here to have the command be restricted if that restriction is settings

Flags Determines which menus this command is available for, and if it is a separator

Sub Menu Itemizer Some commands can work on lists, and these can itemize the list items to show up in a sub menu.

Order The order commands appear in the list also affects the order they appear throughout the interface. Use these buttons to re-order the parameter being edited.

Command Line Product Control

You can use the ConfigureMaster command line tool to manipulate the Smedge products as well:

-ExportProduct *name path*

Use this to export a product to a path. The *name* can be the product name, ID, or any alias. The *path* points to a valid path where the product information will be saved. Product information is saved in JSON format with everything needed to recreate that product.

-ImportProduct *path*

Use this to import a product to add or change it. The *path* is a valid path to a JSON file with the product information.

-RemoveProduct *name*

Use this to remove a product. The *name* can be the product name, ID, or any alias.

Classes

Classes correspond to what existed as “Modules” prior to Smedge 2020. These are the API derived classes that implement the details of how best to operate for various rendering products. With the Smedge 2020 dynamic products, all of the former product module code has been brought into the library. This reduces the startup time for all Smedge components and allows you to make use of any of these product control systems in your dynamic products.

The most common class to use for a generic new rendering process will be “Virtual Module”. You will generally only need to use one of the other classes if you specifically want to support multiple versions of a specific renderer, or handle customizations or special plugins.

3DS Max (Sequence) Supports rendering an image sequence (animation) using 3D Studio Max. Includes extra parameters that can configure Smedge functionality, like copying the scene file locally to improve stability and performance on large networks.

3DS Max (Single) Supports rendering a single image (still frame) using 3D Studio Max. Includes extra parameters that can configure Smedge functionality, like copying the scene file locally to improve stability and performance on large networks.

After Effects Supports rendering an After Effects projects. Can handle rendering a project as a whole on multiple machines, or rendering individual frames from a comp in the project . Includes extra parameters that can configure Smedge functionality and more aggressive error detection to help recover from After Effects’ notorious stability issues.

Fry Supports rendering using Fry. Includes extra parameters that can configure Smedge functionality, like the repeat and merge settings to improve image quality with minimal file copying.

Generic Script Implements a generic command line functionality that can be used for running arbitrary commands. Exposes the command as a job parameter and includes special functionality regarding the range such that an empty range sends the command one time to every machine in the pool instead of as a single job like most other classes.

Indigo Supports rendering using Indigo. Includes extra parameters that can configure Smedge functionality, like the repeat and merge settings to improve image quality with minimal file copying.

Large File Transfer Supports queue and monitoring of large file transfers. Using these allows you to ensure that you don’t swamp a network if you have to copy a bunch of files around, as you can queue each one as a job. Adds fields and implements functionality to move or copy files.

Maxwell	Supports rendering using Maxwell Light Simulator. Includes extra parameters that can configure Smedge functionality, like the repeat and merge settings to improve image quality with minimal file copying.
Maya (Sequence)	Supports rendering an image sequence (animation) using Maya. Includes extra parameters that can configure Smedge functionality, and supports an extended output detection system to handle some of Maya's peculiarities.
Maya (Single)	Supports rendering a single image (still frame) using Maya. Includes extra parameters that can configure Smedge functionality, and supports an extended output detection system to handle some of Maya's peculiarities.
Modo (Sequence)	Supports rendering an image sequence (animation) using Modo. Includes extra parameters that can configure Smedge functionality and the specific way Modo has to communicate by sending commands.
Modo (Single)	Supports rendering a single image (still frame) using Modo. Includes extra parameters that can configure Smedge functionality and the specific way Modo has to communicate by sending commands.
Nuke	Supports rendering using Nuke. Includes extra parameters that can configure Smedge functionality and perform translations to allow cross-platform rendering of the same Nuke script.
Test	Implements the test modules. Includes extra parameters needed to implement and run test work, which does no actual work.
Thea	Supports rendering using Thea. Includes extra parameters that can configure Smedge functionality, like the repeat and merge settings to improve image quality with minimal file copying.
Virtual Module	This is the most flexible class, used to support pretty much any generic render-like process started by command line. Includes extra parameters to implement process control and output monitoring for image filenames. Most Smedge products are Virtual Module class.
imgcvt	Supports manipulating image sequences using the imgcvt tool included with Maya. Includes extra parameters that can configure and implement this functionality.
mental ray	Supports rendering using mental ray standalone.

Legacy Dynamic Products

Legacy dynamic products are still supported and are used for the initial product definitions. In general, it is easier and more reliable to use the new Dynamic Product system instead.

Several Modules use this simple Dynamic Product creation system to make it easy to support multiple versions of the software on your Engine at the same time. The Products are defined in an INI file with the same name as the Module file. For more information about the RLib INI file format, see the chapter on RLib INI File Syntax.

The Module will always create at least one product, with the ID given for that Product in the Products chapter. You can override the values that define that Product, and you can also add new Products. To override the existing Product values, just specify that Product's ID as the INI file section. For new Products, make sure to use a new unique ID. You can use the **`uidgen`** program included in the Smedge distribution to generate UIDs.

Products are defined by creating a new section in the file with the Product's ID. You can then optionally override any or all of the parameters that define the Product. Any parameters you do not override will use the default value that the Module provides for the default Product.

`[` *Product ID* `]`

`Name`	=	*Product Name*
`Alias`	=	*comma separated list of short cuts used for* `Submit -type`
`DefaultEnabled`	=	`yes` *or* `no`
`Executable`	=	*full path to the executable for this particular Product*
`ShortHelp`	=	*short help message*
`LongHelp`	=	*longer help message*
`OverrideDefaults`	=	*A semicolon separated list of {Name} = {Value} pairs that will change the default value of the named parameter for this Job type.*
`OverrideFlags`	=	*A semicolon separated list of {Name} = {Comma separated list of flags} pairs that will change the flags of the named parameter for this Job type.*

Currently this system is used by these Modues: AfterEffects.sx, Fry.sx, Lightwave.sx, Max.sx, Maxwell.sx, MentalRay.sx, Modo.sx, Nuke.sx, which supply the After Effects, Fry, Lightwave, 3D Studio Max (and related), Maxwell, MentalRay Standalone, Modo, and Nuke Products.

Legacy Maya Products

Legacy Maya dynamic products are still supported and are used for the initial product definitions. In general, it is easier and more reliable to use the new Dynamic Product system instead.

The Maya.sx Module supports a hybrid customization of its own, that provides a subset of the entire Virtual Module functionality provided by the Process Sequence.sx Module, that is specifically oriented towards using the Maya integrated renderers. As with Modules that support Dynamic Products, the Maya module will load the Product definitions from an INI file. The Maya Module also adds the ability to customize the options shown in the “Render Overrides” parameter, and other parameters that are specific to how Maya renders its scenes.

The Smedge Module definition file also allows you to override how the Maya module will look for detectable image files from the Maya output stream, and you can override the default values and flags for any other parameter provided by the Maya module.

Smedge ships with a Maya.ini file in the Modules folder that includes definitions of all of the renderers that are supported through Maya 2012, and several third party renderers that have been integrated in, like RenderMan for Maya and finalRender for Maya. You must supply the values shown in bold. It also includes default "single frame" products for several of the built in renderers that support region rendering by command line.

This is the definition of each product section in the Maya.ini file:

`[` *Product ID* `]`

`Name` = ***The name of the product***

`Shortcuts` = ***Comma separated list of alternate shortcut names***

`Help` = ***Short help text***

`LongHelp` = ***Long help text***

`CommandLine` = ***The command line parameters to send to the*** `Render.exe` ***executable***

`DefaultEnabled` = **`yes`** *or* **`no`**

`Extra` = *The list of extra parameter internal names*

`SingleImage` = **`yes`** *or* **`no`**

`ImageCheck` = *Semicolon separated list of image filename search tags in this format:*
`[`*Line Start Text*`] [/ [`*Image Filename Text End*`]]`
Either the Line Start Text *or* Image Filename Text End *parameters can be empty for any entry. The detected image file will start with the specified text, and end before the specified end text It will also automatically have whitespace trimmed and be dequoted.*

`ImageNotStart` = *Semicolon separated list of text strings that will be sued to exclude lines from the image detection system. Excludes any lines that starts with any of the given text strings.*

`ImageNotEnd` = *Semicolon separated list of text strings that will be used to exclude lines from the image detection system. Excludes any lines that end with any of the given text strings.*

`ErrorStarts` = *Semicolon separated list of text strings. If a line of output starts with one of these Smedge assumes that means an error has occurred. And aborts the work unit. If you do not include this value, it will default to the single entry* `Error:`

`ErrorIgnores` = *A semicolon separated list of text strings. If one of these strings is found in a line of output detected as an error because of an Error Start String, that line will be ignored, and the work will not be aborted or marked as failed.*

`CompositeCommand` = *Only used for single frame type renders. This specifies the default composite command for a customized single frame type render. By default, there is no composite command for any single frame Maya render type, but you can specify one in this file and it will be used by default so you don't have to manually add it to your job.*

The Parameters listed in the **Extra** field are also defined in the file. They are defined in the same format as Parameters are defined in Virtual Module files. See the Custom Parameters in the Virtual Module chapter for more information. The default Maya.ini file includes definitions for every parameter available for every renderer included with the current release of Maya, available for reuse as needed for each Product. For more information on what these parameters do, see the Maya software documentation.

Legacy Virtual Modules

Legacy virtual modules are still supported and are used for the initial product definitions. In general, it is easier and more reliable to use the new Dynamic Product system instead.

Smedge 3 includes a module that allows the creation of custom render types via a relatively simple text file. The Module is called **ProcessSequence.sx** and the text files used are called **Virtual Modules**. Virtual Modules are defined in a text file with a **.PSX** extension.

Virtual Modules are derived from the RenderJob class in SmedgeLib. RenderJob is designed to control a third party rendering application via a command line interface to render a sequence of frames. This lineage gives your virtual modules nearly all the functionality you need not only to start and control the rendering of a sequence of images, but to detect and examine the rendered image files, to make resources available at work execution time, and to capture and save the output from the spawned program. Virtual Modules can handle both single scene rendering and one scene file per frame rendering.

When a ProcessSequence Module is loaded, it scans the directory it is in for all files with the .PSX extension. Each .PSX file is designed to hold a single virtual Job type. Every file found by the module will be controlled by that module. If a type is loaded that has a conflicting ID with an already loaded or otherwise existing type, it will simply be ignored. You should only have one copy of the ProcessSequence module in a given directory. Otherwise, if multiple copies of the module are loaded, they will find all the same virtual modules without actually loading any of them.

Because ProcessSequence does not recurse directories, it could be useful for testing purposes to have a copy of the ProcessSequence module in a child directory with just the virtual module you are working on. This way, it will be run by a separate Module than any others. It would be a good idea to rename this module if you do this, which will make it much easier to dynamically load and unload.

PSX Files are simple text documents in an INI style. ProcessSequence uses RLib's IniFile class to handle the actual loading of these files. See the RLib INI File syntax section for more information. Note that ProcessSequence does not use the standard RLib alternate file location system to locate Virtual Module Files. It will only scan for files in the same directory as the compiled module file itself.

Parameter Types

Every parameter has a "type" that gives an indication of how this parameter is expected to be used. There are many types available, but not all types will make sense for all applications. The parameters are passed around as text strings, but they may not be stored that way. There may be some validation associated with some types when you set their value.

Type	*Elements*	*Description*
Alternate	Alternate	A combination of a Boolean value that will result in one of two names for the parameter. Useful for alternate switches for a toggle value where two different names are used for the choices. For example: [-blur/-noblur]
Bool	Bool	A true/false value, with customizable formatted value strings
BoolOverride	BoolOverride	Adds a third possible state to a Boolean value for the Job type: Engine Default. If the Job parameter is set to this value, then the Engine's value will be used. This type requires that the parameter is both a Job parameter and an Engine option to make any sense.
Choice	Choice	Select one of a list of strings. You can have a different value for display and for use as a formatted parameter.
Dir	Directory	A text field interpreted as a directory.
DirList	Dir	A list of directories separated by semicolons
File	File	A text field interpreted as a file. Has a prompt text and a filter. The prompt is displayed to the user by Shells as part of the request to find a file on the local system. The filter is a file selection filter like: `C++ Files (*.cpp, *.cxx)|*.cpp;*.cxx|{All Files}`. If no filter is supplied, the filter will default to: `{All Files}`. If the string includes the single filter `{All Files}` (case insensitive and including the {}s) an default for the local system should be added by Shells appropriate to the local platform. If the string includes the single filter `{Executable Files}` (case insensitive and including the {}s) Shells should add the appropriate filter for the local platform for locating executable files.
FileList	File	A list of files separated by semicolons
Float	Preset	Like **Text**, but interpreted to be a floating point number
ID	Preset	Like **Text**, but interpreted to be a UID in 8-4-4-4-12 hexadecimal format
Info	Common	Not a parameter, but an extra bit of textual information displayed with the parameters by the Shell.
Int	Preset	Like **Text**, but interpreted to be a signed integer
Multi	Multi	Allows a single parameter to be displayed as multiple fields. Display using a name for each field, and a custom separator string.
None	Common	No other controls are created. Useful for a switch in a list of Parameters

Type	*Elements*	*Description*
`Parameters`	Parameters	A text field that can allow a sub-selection of more parameters. The sub-parameters are determined using the separator and internal separator strings.
`Password`	Common	A text field with no presets, and Shells should obfuscate its display
`Separator`	Common	Not a parameter, but a visual separator in the Shell application that is displaying the parameters
`Text`	Preset	A text field, with optional presets to fill it in
`TextList`	Common	A list of text items separated by semicolons
`Time`	Preset	Like **Text**, but interpreted to be a time (in milliseconds since midnight, January 1, 1900)
`Uint`	Preset	Like **Text**, but interpreted to be an unsigned integer

Common Parameters

ProcessSequence is derived from RenderJob and includes all of the parameters from Job, ProcessJob, RenderJob, and SequenceDistributor. It also adds the following parameters. Note that, unlike the underlying classes or other compiled Modules, the default values for these parameters are defined as part of your Virtual Module definition file. There is no hard coded default for any parameter added by the ProcessSequence Module.

Name	*Type*	*Get*	*Set*	*Meaning*
ActualScene	Text	X		This is a standardized way to access the actual scene file or multiple scene files that are used for a single work operation. If the job is using a single scene file, this is identical to the $(Scene) parameter from RenderJob. However, if the job uses one file per frame, this will expand to the correctly formatted scene file or files that are being worked on in a single work unit. It uses the $(SceneNameFormat) and the $(SubRange) parameters to determine which files are returned.
CheckForMulitpleFiles	Bool			This is a Shell Option that configures whether the shell should try to detect multiple scene source fie sequences.
CommandString	Text	X	X	The string that is used to build the parameters that are passed to the rendering executable to allow it to do its job. Generally, this parameter will contain a string of other variables that will be substituted with other parameters for the work being executed. If set, the Job parameter value will override the Engine Product option. If neither is set, the default comes from the Virtual Module definition file. If no default is given, the work may not execute correctly
EnquoteActual	Bool	X	X	$(ActualScene) can automatically enquote scene files as needed. Because multiple files may be returned and those files are separated by spaces, it can produce a command line syntax error to use the **.Enquote** Parameter Command with that variable. You can use this option to disable the automatic enquoting, if needed.
HideSubrange	Bool	X	X	If Smedge detects that your scene is part of a scene file sequence, setting this will cause the Job variable substitution system to return an empty string for the work SubRange. This allows you to have a command line that will include a frame range only for single scene files.
ImageEndString	Text	X	X	If a line is checked for an image, this text marks the end of the line, and will be trimmed off from the detected filename. You do not need to include any whitespace, because that will be trimmed off automatically.

Name	*Type*	*Get*	*Set*	*Meaning*
ImageNotEndString	Text	X	X	If a line starts with the Image Start String, but ends with this text, it won't be considered a filename. This test is applied before the Image End String test.
ImageStartString	Text	X	X	When parsing the output, lines that start with this text are assumed to contain a rendered image filename. The filename is assumed to start at the first non-whitespace character after this text. If left empty, no filename detection will be performed.
MinimumNumberPadding	Int			This is a Shell Option that configures when searching for a scene file sequence, any numbers found must meet or exceed this many digits to be considered part of a sequence identifier.
SceneNameFormat	Text	X	X	For sequences of scene files, this is the formatting string that is used to create the correct filename for each frame. This value is normally automatically calculated by Smedge and should be blank if the entire scene is in a single scene file
SuccessTexts	Text List	X	X	Allows you to define text that will be searched for at the beginning of a line of output that would signify successful work. If there are no SuccessText entries, then the normal work success tests are done. However, if there are any entries in this section, then there must be at least one line of output that starts with one of the entries, or the work will be considered unsuccessful. You can specify as many of these as you need. Separate multiple entries with a semicolon. The text is not case sensitive.

Reference

The Virtual Module definition file must contain at least a **Module** section. This section contains the basic information that Smedge needs to identify and display the Product to users, and to keep it separate from every other Product. Additionally, you can provide data to create your own custom parameters. Custom parameters will be maintained as part of the Job data, and you can configure how they are interpreted and displayed to users.

The Module Section

This is the information that identifies the virtual Job type in the system. The entries in this section define the Product to the system, and provide many of the defaults for the common parameters that ProcessSequence adds (see Parameters above).

ID

This is a unique identifier for this Job type. Smedge uses the standard UUID (or GUID) format for this value. This is a 16 byte number that can be pretty well guaranteed to be unique in the known universe under most circumstances. You **must** provide an ID that is unique to your whole network, or you will have real problems. ProcessSequence will not allow virtual modules with conflicting IDs to load on a single application, but if you have differences between machines on the same network, you could experience erratic behavior or crash when data from one type of job is forced into a conflicting type.

ProcessSequence expects this number to be in the standard 8-4-4-4-12 hexadecimal format:

```
12345678-9ABC-DEF0-1234-56789ABCDEF0
```

Case does not matter. You can use RLib to generate this value (see the uidgen sample program for an example), or you can use your favorite UUID generating program (for example there is one that comes with the Microsoft Visual Studio compiler).

Command

Virtual Modules work by spawning a child process via a command-line interface. This element provides the actual command-line parameters passed to the executable. Because these modules actually inherit from the ProcessJob base class, you also have a parameter called **Executable**, which can be used to provide the rendering executable program.

What ProcessSequence normally does is build a command-line by getting the value of the **Executable** Job parameter, adding a space if needed, then formatting this command value on the end. But there are many other options. For example, users can override a default **Executable** as an option for the Engine. In this case, if the Job's parameter value is empty, the Engine's option value will be used. If

that, too, is empty, then the default set for the type will be used. If there is no default set, no text will be added to the actual command-line.

For the command string part, you can use a variable substitution system to get the actual value of any other Job parameter by name.

You can use any parameter defined in the Virtual Module file, or any parameter defined in the base classes.

Name

This is the nice, user-readable name for this Job type. It is displayed in the Shells for all Jobs of this type.

Shortcut

This is a comma separated list of alternate names that refer to this type. These may be useful as shortcuts for using command-line based Shells to interact with the Job type. For example, the submit program will use these as alternates names to determine what Job type you are trying to create.

Help

This is a short help string that Shells may display to users. In this field (and in this field only) you can use the special character sequence **\n** (a backward slash followed by a lowercase letter N) to signify a line break. If you need to have a literal \n in your string, double the backslash. Because \n is the only sequence that Smedge will look for, you do not need to double the backslashes in any other situation.

Details

A more detailed help string that Shells may display to users.

DefaultEnabled

You can determine if this type will be enabled by default when an Engine encounters it for the very first time. If you set this to a value that will be parsed as true (“True” “Yes” “On” or “1”), then an Engine will enable this Job type the first time it is loaded. Otherwise, the type must be manually enabled by users.

ErrorDetectPosition

Override the default error string detection position. Set to "0" to look at the beginning of the line or "1" to look anywhere in the line.

ImageStartText

You can supply one or more values here which the Virtual Module will use to try to detect rendered image filenames from the output. The work unit will watch the output from the process for a line that starts with one of these texts. The first non-whitespace character following whichever string is detected will be assumed to be the beginning of the filename. You can use parameters in this text, and it will be translated when the work starts, based on the current values of the work unit. Separate entries using the semicolon (**;**) character.

ImageEndText

You can supply one or more values that will be search for from the back of any line determined to be an image line using the **ImageStartText** values. If one of these strings is found, the filename will be assumed to end with the last non-whitespace character before this text. Whitespace is automatically trimmed from the parsed lines, so you don't need to add it here if you just want to trim off trailing whitespace. Separate entries using the semicolon (**;**) character.

ImageNotEndText

You can supply one or more value that will be searched for from the back of any line determined to be an image line using the **ImageStartText** values. If one of these strings is found, then any lines that start with the **ImageStartText** value that end with this value will be ignored as potential image filenames. Separate entries using the semicolon (**;**) character.

CheckForMultipleFiles

Virtual Modules can handle products that use either a single scene file for a sequence, or products that use a sequence of scene files, one for each rendered output that is produced. In order to enable the extra processing to detect multiple source scene sequences, set this value to **yes**. The system expects that the last set of digits in the filename are used to put the scene files into sequence. Some examples of sequenced filenames that Smedge will understand:

```
/Volumes/Work/Scene/Folder/Sequence012_ver2.0122.input
X:\Projects\Animation\File01115.input
/usr/data/sequence/0124
```

If you set this value to **no**, or if you do not supply this key in the Virtual Module file, then this processing is disabled, and the scene file will be assumed to be only a single file for all of the frames in the Job.

Custom Parameters

The real power of Virtual Modules comes in defining your own Parameters. You can then use these parameter types as part of your **Command** string and every other place where you need to access custom data. The parameters are accessed using the parameter syntax described above, but are defined in a simple INI format with a few lines of text.

There are actually two sections that you can use to define parameters: **PrimaryParameters** and **SecondaryParameters**. The only difference between the two is the order in which the parameters are added to the Job Type Information. "Primary" parameters are added to the info data before the underlying base classes, and "Secondary" parameters are added after. The order of the parameters has no meaning to the actual operation of the work, but may affect the order in which parameters are displayed in Shells.

The name you supply is the internal name of the parameter, which is the name that is used in parameter substitution. This internal name must not conflict with any other internal name for a parameter or a command in the virtual module, or any of its base classes. See above for a full list of the names already used. Because the parameter name is not generally shown to users (the "NiceName" value is shown instead), it is customary not to use spaces in parameter names. This helps differentiate between parameters and commands. Note that the NiceName, which is displayed to users, can be anything you wish, but it's probably a good idea not to let this name conflict with other names so that you don't confuse users.

The ProcessSequence module will iterate through all of the keys in these sections to determine what parameters are going to be added. The key name is ignored in this iteration, but can be used to determine the order in which parameters will be added, if you care. The iteration will go through the keys alphabetically (case insensitive, ASCII order). Remember this is using a text comparison, so a key of "10" will be parsed before a key of "2".

Each value will be the internal name of a parameter. ProcessSequence will expect that there is a section with a matching name that is used to actually load the parameter information. These sections contain the following elements:

Type

You must provide a type for the parameter. This is used to determine what other data is available in the Job Type Information for this parameter. It is also used by Shells to determine how to format, display or validate the data. See Parameter Types for a list of the types available.

Common Elements

All parameter types can contain these elements.

RealName This is an optional element that allows you to specify an alternate real (internal) name for the parameter. If no RealName is supplied, the real (internal) name for the parameter will come from the section name in the INI file. This element is useful when you need to ensure that parameter names are case sensitive, because INI files are not case sensitive in Smedge.

NiceName This is the name of the parameter that will be displayed to the user.

Help This is some help information that Shells may display to the user when interacting with this parameter.

Default This is the default value for this parameter type

Flags This is a comma separated list of the flags that affect how this parameter is used. You can have any combination of the following values:

Name	*Meaning*
`Parameter`	The parameter describes a value member of the Job object
`Required`	*Modifies **Parameter***. This parameter must have a value associated with it. Without a value associated, the parameter is ignored
`Advanced`	*Modifies **Parameter***. This member is considered an "advanced" parameter. Exact interpretation is left up to the Shell.
`Option`	The parameter describes an option for this Job type.
`Master`	*Modifies **Option***. This option is used by the Master. *Currently ignored…*
`Engine`	*Modifies **Option***. This option is used by the Engine.
`Shell`	*Modifies **Option***. This option is used by the Shells.
`NoOptionDefault`	The default is not used for this parameter when shown as an **Option**.
`NoParamDefault`	The default is not used for this parameter when shows as a **Parameter**.
`NoInputDisplay`	This parameter is not shown by Shells to input or change parameters.
`NoOutputDisplay`	This parameter is not shown by Shells to display Job parameters.

Preset Elements

For the types that can have presets (see the list of types above), all of the common elements are allowed, as well as these elements

Choices This is a comma separated list of preset choices that can be made available to the user. If items are supplied here, they will be made available as presets in a manner appropriate for the Shell. For example, a GUI may show them as entries in a combo box. But the value is not limited to one of these choices. You must use the **Choice** parameter type if you want that behavior.

Choice Elements

For the **Choice** type, all of the common elements are available, as well as these elements.

Choices	This is a comma separated list of the choices available. These will be the only allowed choices, and even the default must be one of these.

You can have each choice have a value that is displayed to the user that is actually different than the value that is substituted in the parameter substitution. Each choice is in a **DisplayName**:**ActualValue** format. For example:

```
Choices = Red:r, Green:g, Blue:b
```

This will create a choice that displays “Red,” “Green,” or “Blue” in a Shell, but that substitutes to **r**, **g**, or **b** in the command string. If you don’t put a colon into the choice string, the name and display name will be the same.

You should make sure that the default is one of the choices you have supplied. You must currently use the display name for the choice, not the actual value. If you don’t supply a default, the first choice will be selected by default.

Multi Elements

For the **Multi** type, all of the common elements are available, as well as these elements.

Fields	This is a comma separated list of the fields that make up this multi-value. The count of fields is determined by this list, but empty elements are allowed (i.e., no text between two commas).
Separator	This is the string that is used to break up the value into the fields, and that is inserted between each element when assembling the multiple fields back into a single value. If you don’t supply a value for this, the default, a single space character, will be used.

Bool Elements

For the **Bool** type, all of the common elements are available, as well as these elements

True	This is the text value that will be substituted for a true value. If not supplied, the system will use "Yes" by default.
False	This is the text value that will be substituted for a false value. If not supplied, the system will use "No" by default.

BoolOverride Elements

For the **BoolOverride** type, all of the **Bool** elements are available as well as this element

OverrideText The text value that is displayed for Job parameters in the third state (when the Engine option provides the actual value for work execution).

Alternate Elements

For the **Alternate** type, all of the common elements are available, as well as this element

Alternate This is the alternate name that is used. The Name value is used for “true” or “on” and this alternate value is used for “false” or “off”.

Dir Elements

For the **Dir** type, all of the elements of the **Presets** type are available, as well as these elements.

Prompt This is a prompt string that can be shown to users when they want the Shell to allow them to browse for the directory (or file).

File Elements

For the **File** type, all of the elements of the **Dir** type are available as well as these elements

Filter This is a filter string that can be used by the file selection dialog to make it easier to find files based on their extensions. The filter part will be a file selection filter like "C++ Files (*.cpp, *.cxx)|*.cpp;*.cxx|{All Files}". No filter will default to {All Files}. If the string includes the single filter "{All Files}" (case insensitive and including the {}s) the appropriate actual filter should be added by Shells for the platform they are running on. If the string includes the single filter "{Executable Files}" (case insensitive and including the {}s) the appropriate actual filter should be added by Shells for the platform they are running on.

Parameters Elements

For the **Parameters** type, all of the common elements are available, as well as these elements.

InternalSeparator This is the string that is inserted in between a child parameter name (real name) and its value if that child parameter is being added to the full parameter string. Defaults to a single space if not supplied in a parameter definition.

Parameters This is a comma separated list of the parameter names that make up this list of parameters. This allows you to create a group of parameters accessed with a single parameter name. SmedgeGui displays this parameter type as a separate tab in the Submit Job window. (See the User Manual for more information about this feature.)

Separator This string is inserted between the name (real name) of the parameter and its child parameters string. Defaults to a single space if not supplied in a parameter definition.

First you give each child parameter the name of the actual commandline switch that it uses. Then you create these parameters to work in a manner that allows users to easily set and select which extra parameters they want to add to their commandline. The easiest way to understand this is to see it in action. Check out the **Extras** parameter in the following example PXS file, and check out how it works in the SmedgeGui Shell application.

Custom Commands

As with parameters, you can add custom commands to your Virtual Module. Currently, these commands are limited to executing a commandline (after performing parameter substitution on it, of course!). Just like with the parameters, you can add your commands in either the **PrimaryCommands** section or the **SecondaryCommands** section. The only difference is that the "Primary" commands are added before commands from the base classes, and "Secondary" commands are added after. The order of the commands is not important to Smedge, but may affect the order they are displayed in Shells.

Each command must have a name that does not conflict with any parameter or command either defined in the Virtual Module file or from any base class. Because Commands names are usually displayed directly to the user, the convention is to use spaces in command names. Unlike Parameters, commands do not have a separate display name.

Command Elements

Commands have three elements.

Command This is the actual command-line that will be substituted and executed when this command is triggered by a user.

Help This is a string that Shells can display to let the user know what this command is going to do

RestrictionName This is a string that Shells can use as part of the voluntary restriction system in Smedge. If you give your command a restriction name, then you can use the Configure Master dialog box to set a restriction on that command or not, using your custom name. Any restricted command will be unavailable at run time if the RestrictionName is set as restricted, unless the application is running as an administrator.

Flags This is a comma separated list of flags, which can be any combination of the following values:

`ForParent` This command applies to parent Jobs

`ForChild` This command applies to child Workers

`Separator` A placeholder to insert a separator between commands. Shells should interpret this in a manner appropriate to how that Shell displays commands to users.

Other sections

ErrorText

This section allows you to define text that will be searched for at the beginning of a line of output that would signify an error in the work unit. Smedge will then assume that the work unit has failed and will requeue it.

You can specify as many of these as you need. The key name does not matter, but must be unique for each string you want to search for. The text search is case sensitive.

SuccessText

This section allows you to define text that will be searched for at the beginning of a line of output that would signify successful work. This section will take precedence over any other error detection systems. If there are entries in this section, then there must be at least one line of output that starts with one of the entries, or the work will always be considered unsuccessful. However, if there are no entries in this section or if this section is missing from your Virtual Module file, then the normal error detection systems will be used.

You can specify as many of these as you need. The key name does not matter, but must be unique for each string you want to search for. The text search is case sensitive.

OverrideDefaults

This section allows you to override the default values for any parameter. Since you can just provide the default for custom parameters defined in the file, this really only makes sense to override defaults for parameters from the base classes. Use the parameter name as the key and supply your new default value.

Note that parameter defaults will still follow the defined flag for allowing display of the default value (the **NoOptionDefault** and **NoParamDefault** flags are not modified). See Custom Parameters for more information.

OverrideFilters

This section allows you to customize the filter strings for any File type parameters defined in ProcessSequence or any of its base classes. The key should contain the internal name of the File parameter you want to modify the filter for, and the value specifies the new filter string value.

OverrideFlags

This section allows you to customize the flags for any parameter defined in ProcessSequence or any of its base classes. The key should contain the internal name of the parameter you want to modify the flags for, and the value specifies a comma separated list of the flags you want for that parameter. The flags are replaced with the flags you specify, so be sure to specify all of the flags that the parameter will require or you may find that things don't work as you expect them to. See Parameter Flags for a list of all the flags available and what they mean.

AutoDetect

This section allows you to configure a basic auto-detect ability to set the value of one or more other parameters when a parameter value is changed by a user in a Shell. This system is not as sophisticated as the functionality you would have if you used a compiled module, because compiled modules give you full access to the Smedge API and whatever 3rd party libraries you link into your module. However, it does give a basic functionality using the variable command syntax described in the Command section.

The key is the name of the parameter that triggers the auto-detect change. When this parameter is changed by the user, the parameters listed in the value will then be changed as well, using the current values of any parameters requested. The basic syntax is ***Value*** **=** ***Change****[, Change…]*. Each ***Change*** is in a ***ChangeValue*** **=** ***ChangeTo*** format. For the changes, the **ChangeValue** is the name of the parameter that you want to automatically modify. No variable substitution is performed on this string. The **ChangeTo** string will be variable substituted.

This syntax can be somewhat complicated. It is also additionally complicated by the use of the = character in the Value, which means that you should not break these lines up onto multiple lines in your Virtual Module file. Examine this sample:

```
Scene = Name = [$(Project): ]$(Scene.File)
```

The *Value* that triggers the auto-detect is **Scene**. When the **Scene** parameter is changed by the user, the *Change* that will be executed is "Name = [$(Project):]$(Scene.File)". This auto-detect has only one Change. The *ChangeValue* is **Name**, which means that when the **Scene** parameter is modified, the **Name** parameter will then also be automatically modified with the *ChangeValue* of "[$(Project):]$(Scene.File)". The current value of the **Project** parameter and the new value of the **Scene** parameter will be run through the variable substitution system to calculate the actual value that will then be assigned to the **Name** parameter.

FindLatestExecutable

This section sets up the criteria by which Smedge can find the latest executable automatically when jobs for this product are run. The latest executable search requires 3 parameters for each platform you want to be able to search on: "root" has the folders you want to search in, "base" has the base filenames to search for versions, and "exe" is the executable name to search for in the each base folder version found. You can define separate criteria for each platform (Windows, Linux and Mac).

For example, see the Houdini.psx settings:

```
WindowsRoots = C:\Program Files\Side Effects Software, C:\Program Files, C:\Program Files
(x86)\Side Effects Software, C:\Program Files (x86)
WindowsBases = Houdini
WindowsExe   = bin\mantra
MacRoots     = /Library/Frameworks
MacBases     = Houdini.framework/Versions/
MacExe       = Resources/bin/mantra
LinuxRoots   = /opt
LinuxBases   = hfs
LinuxExe     = bin/hrender
```

On Mac, this will search in the folder `/Library/Frameworks`, searching for each version found with a base folder name like: `Houdini.framework/Versions/*` that has, inside of it, the executable in `Resources/bin/mantra`. Using these criteria, the highest version number found in any of the root folders is used.

Example File

This is a simple example Virtual Module file that shows the basic structure and syntax. It does not actually support any particular renderer, but is useful as an example of creating the file. Note that any comments (in green and starting with a ; at the beginning of the line) are totally optional, and whitespace in and around the section and key strings is ignored. See the RLib INI syntax reference for more information.

Also be sure to look at the Virtual Module files that are included in the Smedge distribution to see other examples of what you can do with a Virtual Module, and how to accomplish it.

```
; ProcessSequence.psx

;
; This is a sample virtual Module file for a ProcessSequence.sx Module.
;
; Smedge 3
; Copyright (c) 2004-2005 Überware

; The Module section includes the basic module information

       [ Module ]

; Every Job type in Smedge needs its own unique ID. Make sure to modify this
; ID for each new type. Note that Jobs and options will store this type as
; well, so if you change this ID, you will lose any saved options, and be
; unable to use any saved jobs that rely on it. You can use the included
; commandline too uidgen.exe to create new IDs, or any other tool that will
; create a Globally Unique ID.

ID                 = 46D1CA19-FD89-4ad5-9A43-676A28764C3D

; The actual commandline is generated by assembling the executable and command
; fields and doing a member name substitution on any special parameters
; marked. Parameters are marked with the notation $(name). Any parameters from
; ProcessSequenceJob or its base classes is available for use, as well as any
; custom parameters defined in this file. For a complete list of parameters
; available, see the documentation for the Job derived classes: SequenceJob,
; ProcessJob, and RenderJob (more may be available in later versions of Smedge).
;
; The most common parameters you will use will be:
;
; $(Scene)          - The scene the user can specify for the Job
; $(SubRange)       - The full range of the work formatted into a x-y string (no spaces)
; $(SubRange.Start) - Just the start of the range of the work
; $(SubRange.End)   - Just the end of the range of the work
; $(CPUs)           - The number of CPUs that are assigned to this job.
;
; If you want to have a portion of the commandline conditional on a non-empty
; parameter value, you can surround that portion with square brackets: []
;
```

```
; The Command gives the defaults for these values.
; All values are overridable at both the Job and Engine levels.

Command             = [-proj $(Project) ]-start $(SubRange.Start) -end $(SubRange.End)
                      -format $(Format) [$(Extra) ]$(Scene)


; Because ProcessSequence is derived from the RenderJob base class, it
; automatically gets all of the functionality of a Rendering app, including
; automatic frame name detection and verification. To enable this, you can
; provide output parsing strings here for detecting the filenames from the
; output of the spawned process. If the detected names are not absolute
; paths, the Engine will look for a parameter called ImageDir and prepend
; that value, or, if that value is empty or cannot be found, the
; StartDirectory value will be prepended.

; Look for lines that with this text. The first non-white spcace text
; following this text is assumed to be the start of the filename. You can use
; parameters in these fields, and they will be translated when the Work is
; started


ImageStartText      = Rendering:

; Trim this text from the end of the line. Whitespace will be trimmed
; autmoatically, so you don't need to add it here.

ImageEndText        = .


; You can specify text that indicates that lines which may start with the
; ImageStartText string may not actually be images. If the line ends with
; this text, it will not be processed as an image. This test is done before
; the ImageEndText is checked and removed.

ImageNotEndText     =

; The rest of the fields allow you to customize how users interact with your
; custom type

; This is the name displayed to the user for this type of Job

Name                = My Job Type

; This is a shortcut you can use to create jobs of this type, say by a
; commandline. Values can be separated by commas. Whitespace surrounding values
; will be ignored.

Shortcut            = myjobtype, myjob

; This is a short help string that some Shells may display to the user about this type

Help                = This is a custom ProcessSequenceJob type

; This is a more extended help text that some Shells may display to the user about this type

Details             = There are no Shells that currently display this information, but
                      we're working on that. In the mean time, here's a nice long string of
                      information to get you started.

; This allows you to determine if this type should be enabled by default on
; an Engine that has never seen it before.

DefaultEnabled      = No
```

```
; The PrimaryParameters section allows you to add custom parameter for the
; type. These parameters are added before any parameters from the underlying
; base classes, so they will generally appear above them in Shells. The order
; of parameters is only important to Shells. You can order your parameters by
; using a number for the key name. If two parameters have the same number, the
; last one read will be the one used.
;
; Parameters must have a unique name. This name is used internally to access
; the parameter value, and is used as a section name in this file to describe
; all of the parameter information details. The parameter name must not
; conflict with any other parameter or command, either in this virtual Module
; file or in any of the underlying base classes. The parameter name is not
; generally shown to users, so the general convention is to not use spaces in
; parameter names, and use spaces in command names.


    [ PrimaryParameters ]


1                   = Format


; The SecondaryParameters section is the same as the PrimaryParameters, except
; that these parameters are added to the Job type information after any
; parameters from the underlying base classes. This means that they will
; appear below the underlying parameters in Shells. The order of parameters is
; only important to Shells. The names of the parameters must not conflict with
; any other parameters or commands, either in this file or in any of the
; underlying base classes. The parameter name is not generally shown to users,
; so the general convention is to not use spaces in parameter names, and use
; spaces in command names.


    [ SecondaryParameters ]

1                   = Extra
2                   = Project


; The PrimaryCommands section allows you to add custom commands to the type.
; Currently commands are limited to executing an external program via a
; commandline. As with the parameters, Primary commands are added to the list
; before the underlying job classes. The only affect this has is the order of
; display of commands in Shells. Commands can be ordered by a number in the key
; name. The Command name must not conflict with any other parameters or
; commands. A command's name is also what is displayed to a user, so the
; general convention is to use spaces in Command names an no spaces in
; Parameter names.


    [ PrimaryCommands ]

1                   = View Format Info
2                   = Separator


; The SecondaryCommands is just like the PrimaryCommands, except that these
; commands are added after the underlying Job base classes.


    [ SecondaryCommands ]

; The ErrorText section allows you to define text that will be searched for
; at the beginning of a line of output that would signify an error that Smedge
```

```
; should assume would signify a failed work unit. You can specify as many of
; these as you need, and the key name does not matter (but must be unique for
; each different value you need). The text is case sensitive.

    [ ErrorText ]

1                   = ERROR:

; The SuccessText section allows you to define text that will be searched for
; at the beginning of a line of output that would signify successful work. If
; there are no SuccessText entries, then the normal work success tests are
; done. However, if there are any entries in this section, then there must be
; at least one line of output that starts with one of the entries, or the work
; will be considered unsuccessful. You can specify as many of these as you need
; and the key name does not matter (but must be unique for each different value
; you need). The text is case sensitive.

    [ SuccessText ]


; The OverrideDefaults section allows you to override the default values for
; any parameter. Since you can just provide defaults for custom parameter types
; in their definition, this really only makes sense to override defaults for
; parameters from the base classes. Use the parameter name as the key and
; provide your new default.
;
; Note that parameter defaults will still follow the defined flag for allowing
; display of a default for a Job parameter or an Engine/Shell option. See the
; documentation of ParameterInfo for more information.


    [ OverrideDefaults ]

Executable          = MyProgram
StartDirectory      = $(Project)/images


; The AutoDetect section allows your virtual module to perform a basic
; auto-detect ability to set the value of one or more other parameters
; of a job based on the values of other parameters. This is not as cool as
; what you get for doing a compiled module, where you have full access to the
; APIs for both Smedge and whatever 3rd party libraries you want to link to
; your module (e.g., the Maya module, if you wanted to actually open and work
; with Maya files in the AutoDetect logic.)
;
; The keys are in a Value = change[, change ...] sequence. The Value is the
; parameter that has just been modified. Each change in the list is also in
; a value=change syntax. The value here is the name of the parameter you
; want to modify, and the change is a string that will be run through the
; FormatStringWithParameters function, like the commandline. Use $(Name)
; syntax to access other job values

    [ AutoDetect ]

Scene               = Name = [$(Project): ]$(Scene.File)

; Now we have a section for each parameter and command to actually define the
; functionality. The name of these sections must correspond tothe names used
; above.


;
;               Parameters
;
```

```
    [ Project ]

; This is an example of a Parameter definition. You must provide at least a
; type and a name, and a flag if this is a Job object parameter or an option
; for the type. See the documentation for ParameterInfo for full details
; and requirements.

NiceName            = Project
Type                = Dir
Help                = This is the project base directory
Default             =
Flags               = Parameter

; This is specific to the Dir type:

Prompt              = Select the base project directory


     [ Format ]

NiceName            = Scene Format
Type                = Text
Help                = This is the custom type format. You must provide a valid format, or select one from the list
Default             = ascii
Flags               = Parameter, Required

; This is specific to the Text type (technically to all PresetsParameterInfo types):

Choices             = ascii, binary


    [ Extra ]

NiceName            = Extra Parameters
Type                = Parameters
Help                = Extra commandline parameters you can pass to the executable.
Flags               = Parameter, Advanced

; This is specific to the Parameters type:
; These are all of the sub-parameters that are used to generate this value

Parameters          = -im, -x, -alpha, X


    [ -im ]


NiceName            = Image Name
Type                = Text
Help                = The base filename for the images
Flags               = Required


    [ -x ]


NiceName            = Resolution
Type                = Multi
Help                = The X and Y resolution in pixels
Flags               = Required

; These are specific to the Multi type:
Fields              = X, Y
; Notice the whitespace is made part of the separator by using quotes. The
; quote marks will be removed when the file is read, but the spaces inside
```

```
; will be left as part of the value of Separator.
Separator           = " -y "


    [ -alpha ]


NiceName            = Render Alpha
Type                = Bool
Help                = Include the alpha channel in the rendered image file
Flags               = Required


    [ X ]

RealName            = -x
NiceName            = Capital X
Type                = Bool
Help                = This parameter should appear as a capital X separate from the lower case x for the Resolution
Flags               = Required

;
;               Commands
;


    [ View Format Info ]

; This is an example of a command section

Command             = $(Executable) -viewFormatInfo $(Format)
Help                = Try to view the completed frame from the renderer
Flags               = ForChild, ForParent


    [ Separator ]

Flags               = Separator
```

Product Reference

This is a reference for all Products that currently ship with Smedge. The following information provides the Smedge default Products from the suite of Job Modules included with Smedge. Many of these Modules support dynamic creation of additional Products, and your system may include different or additional Products. See the chapters on Dynamic Products, Maya Products, and Virtual Modules, or talk to your system administrator for more information about custom or third party Modules you may have installed.

3D Studio Max

This Module is designed to control the 3ds max command line renderer included in versions 6.0 and later. This is a Compiled Module. It includes all of the parameters from Job, ProcessJob, RenderJob, and SequenceDistributor. The Max.sx Module can be customized to support multiple versions of Max with the Dynamic Products configuration system, with one extension: The "SplitFrame" key is a Boolean value that determines if the customized Product is a sequence renderer or a single frame renderer (see 3D Studio Max (SingleFrame))

Copying the scene locally

Max has some issues when used in larger networks. Autodesk has confirmed that the Max program code itself can cause failures reading the scene file when a large number of nodes try to read the same file simultaneously. To work around this problem, the Max.sx Module has a system to copy the scene file to the local Engine's drive before starting the max renderer. In so doing, it can also generate a path file with all of the directories and subdirectories of from the directory in which the original scene file is located. This way, any textures can be referenced relative to the scene file and can still be accessed even if the scene file is copied to a different folder. If you have specified a path file for the job, the paths from that file are also read and added to the generated path file.

This system is implemented internally in the Module. You can use the **CopyLocally** parameter of a Job or option for an Engine to enable or disable this behavior. All Max Products (both single and multi-frame) can use this system. By default, it is not enabled. See the Job Parameter Command CopyLocally for more information about the local copy system that is used to implement this behavior.

General Information

Default ID	`A4F726B0-A613-4f2b-94FB-42C812857459`

Type Name and Shortcuts	`3D Studio Max`
	`Max`
	`Max6`

Parameters

Name	*Type*	*Get*	*Set*	*Meaning*	*Default*
CopyLocally	BoolOverride	X	X	Enables the Max scene file to be copied locally. See Copying the scene locally.	Engine Default (no)

Name	*Type*	*Get*	*Set*	*Meaning*	*Default*
Extra	Parameters	X	X	Extra parameters you want to send on the command line. Note: In order to be sure that the parameters are passed properly, enclose this entire list of parameters in quote marks. If you have only 1 parameter, you still need to enclose it in quote marks, and include a space somewhere in the string. (See bug database for more information.)	
LocalPathFile	BoolOverride	X	X	Enables the generation of the local path file (the MXP file) when the copy locally system is enabled. The local MXP file will not be generated if this is not on, and the renderer will only look for textures in the default locations and any locations specified in the original MXP file.	Engine Default (yes)
NthFrame	Text	X	X	Allows you to change the number of frames rendered in a sequence. Make sure that your Packet Size is set to a whole number multiple of this value. If blank, it will default to 1	
PathFile	File	X	X	A path configuration file (.mxp)	
TimeLimits	Parameters	X	X	Max time limits	
WorkPath	Dir	X	X	Root location for job data folders	

3D Studio Max (Single Frame)

This Module is designed to control the 3ds max command line renderer included in versions 6.0 and later to split a single frame across multiple machines. This is a Compiled Module. It includes all of the parameters from Job, ProcessJob,RenderJob, and SliceDistributor. The Max.sx Module can be customized to support multiple versions of Max with the Dynamic Products configuration system, with one extension: The "SplitFrame" key is a Boolean value that determines if the customized Product is a sequence renderer or a single frame renderer (see 3D Studio Max)

Single Frame Max Products can also make use of the system for Copying the scene locally.

General Information

Default ID	`e59d00dd-3c2e-4994-b665-ca5fdcbc92ed`

Type Name and Shortcuts	`3D Studio Max (Single Frame)`
	`Max single`
	`Max1`

Parameters

Name	*Type*	*Get*	*Set*	*Meaning*	*Default*
CopyLocally	BoolOverride	X	X	Enables the Max scene file to be copied locally. See Copying the scene locally.	
Extra	Parameters	X	X	Extra parameters you want to send on the command line. Note: In order to be sure that the parameters are passed properly, enclose this entire list of parameters in quote marks. If you have only 1 parameter, you still need to enclose it in quote marks, and include a space somewhere in the string. (See bug database for more information.)	
Overlap	Int	X	X	The amount of overlapping pixels that border each strip	10
PathFile	File	X	X	A path configuration file (.mxp)	
TimeLimits	Parameters	X	X	Max time limits	
WorkPath	Dir	X	X	Root location for job data folders	

3Delight

This Module is designed to control 3Delight renderer. This is a Virtual Module. It includes all of the parameters from ProcessSequence.

General Information

Default ID	`7582c8ad-cafd-4ac2-b5f0-a20101872527`
Type Name and Shortcuts	`3Delight`

Parameters

Name	*Type*	*Get*	*Set*	*Meaning*	*Default*
Extra	Parameters	X	X	Extra parameters you want to send on the command-line. Note: In order to be sure that the parameters are passed properly, enclose this entire list of parameters in quote marks. If you have only 1 parameter, you still need to enclose it in quote marks, and include a space somewhere in the string. (See bug database for more information.)	

3Delight for Maya

The **Maya.sx** compiled Module now includes the ability to set up customized products for different renderers accessible through the Maya command line interface. Each Product is a section in the Maya.ini file which is in the Module folder. You can add your own or customize the existing ones in a similar manner to how you create Virtual Modules.

The **RenderMan for Maya** Product is designed to control the 3Delight for Maya plug-in for Maya. This product is included in the Compiled Module **Maya.sx**. This will override any Render Layer renderer settings in your Maya 7 file.

The RenderMan for Maya Product includes all of the parameters from Job, ProcessJob, RenderJob, and SequenceDistributor.

General Information

Default ID	`4b877075-b0d8-44a5-89dd-d62e557bff45`
Type Name and Shortcuts	`3delight4m` `3d4m`

Parameters

Name	*Type*	*Get*	*Set*	*Meaning*	*Default*
Extra	Parameters	X	X	Extra parameters you want to send on the command-line. Note: In order to be sure that the parameters are passed properly, enclose this entire list of parameters in quote marks. If you have only 1 parameter, you still need to enclose it in quote marks, and include a space somewhere in the string. (See bug database for more information.)	
Project	Dir	X	X	The Maya project to use for rendering. If left blank, no project will be passed to the renderer.	
RenderDir	Dir	X	X	Override the output folder	
RenumberBy	Text	X	X	The increment value for renumbering the frame files. Blank defaults to 1	
RenumberStart	Text	X	X	The Job can be renumbered starting at this value. If left blank, the file frame numbers will correspond to the frames rendered	

Name	*Type*	*Get*	*Set*	*Meaning*	*Default*
SequenceBy	Text	X	X	Allows you to change the number of frames rendered in a sequence, using the **-b** flag. Make sure that your PacketSize is set to a whole number multiple of this value. If blank, defaults to 1	

3Delight for Maya (Single Frame)

The **Maya.sx** compiled Module now includes the ability to set up customized products for different renderers accessible through the Maya command line interface. Each Product is a section in the Maya.ini file which is in the Module folder. You can add your own or customize the existing ones in a similar manner to how you create Virtual Modules.

The **RenderMan for Maya (Single Frame)** Product is designed to control the 3Delight for Maya plug-in for Maya. This product is included in the Compiled Module **Maya.sx**. Using this product will override any Render Layer renderer settings in your Maya file.

The RenderMan for Maya (Single Frame) Product includes all of the parameters from Job, ProcessJob, RenderJob, and SliceDistributor. It adds the SingleFrame option in the product definition in order to provide the attributes used for splitting single frames

General Information

Default ID	`51C0B8BB-D6C5-4c7f-AC01-C827B3F1D9E1`

Type Name and Shortcuts	`3d4m-1`
	`3Delight for Maya Single Image`
	`3Delight for Maya Single`
	`3delight4m-1`

Parameters

Name	*Type*	*Get*	*Set*	*Meaning*	*Default*
CompositeCommand	Text	X	X	This is the command that can be used to re-assemble the slices. No default is provided because Maya does not include a system that can composite every possible type of image that it can create. You can supply your own command, using job variable substitution to find and place the slices. If this is blank when the stitch work is ready to be sent, no automated composition of the image slices is done.	
Extra	Parameters	X	X	Extra parameters you want to send on the command-line. Note: In order to be sure that the parameters are passed properly, enclose this entire list of parameters in quote marks. If you have only 1 parameter, you still need to enclose it in quote marks, and include a space somewhere in the string. (See bug database for more information.)	

Name	*Type*	*Get*	*Set*	*Meaning*	*Default*
Project	Dir	X	X	The Maya project to use for rendering. If left blank, no project will be passed to the renderer.	
RegionEnd	Multi: •X •Y	X		The top, right corner of the current slice in pixels	
RegionEndPercent	Multi: •X •Y	X		The top, right corner of the current slice as a percentage (floating point between 0 and 1). *Note:* This value is currently only correctly calculated if the RenderRegion is the entire image size.	
RegionName	Text	X	X	The naming convention for each slice. You can reference other parameters in this text, and they will be properly expanded at render time.	Slice_ $(RegionStart.X)- $(RegionStart.Y)
RegionSize	Multi: •X •Y	X		The size of the current slice, in pixels.	
RegionSizePercent	Multi: •X •Y	X		The size of the current slice, in pixels.	
RegionStart	Multi: •X •Y	X		The bottom, left corner of the current slice in pixels	
RegionStartPercent	Multi: •X •Y	X		The bottom, left corner of the current slice as a percentage (floating point between 0 and 1). *Note:* This value is currently only correctly calculated if the RenderRegion is the entire image size.	
RenderDir	Dir	X	X	Override the output folder	
RenderRegion	Multi: •Left •Right •Bottom •Top	X	X	This is the total sub-region of the full image size that will be divided up to other machines. To render the full image size, use 0 for Left and Bottom, and the image width and height for Right and Top, respectively. Fields: Left, Right, Bottom, Top Separator: " "	
SliceImage	Multi: •Horizontal •Vertical	X	X	The number of sections, horizontally and vertically, to divide the image into.	

Name	*Type*	*Get*	*Set*	*Meaning*	*Default*
SequenceBy	Text	X	X	Allows you to change the number of frames rendered in a sequence, using the **-b** flag. Make sure that your PacketSize is set to a whole number multiple of this value. If blank, defaults to 1	
SliceOverlap	Int	X	X	The number of pixels of overlap between the slices. Useful for reducing edge artifacts on the slices.	50
Slices	Text	X		The total number of slices from the image that will be rendered. Equal to: $(SliceImage.Horizontal) x $(SliceImage.Vertical)	

After Effects

This Module is designed to control the AfterEffects 6.x or later commandline renderer. This is a Compiled Module. It includes all of the parameters from Job, ProcessJob, RenderJob, and SequenceDistributor.

This Module supports the creation of Dynamic Products with a simple INI file format. See the Dynamic Products chapter for more information.

Using Smedge with After Effects

After Effects works differently than most other rendering systems. Because of the variety of ways that you can configure and use AE to render, see the supplemental *Using Smedge with After Effects* manual included in the documentation folder of the Smedge distribution.

Note that you cannot currently use Smedge to control After Effects rendering to a single movie file on multiple machines. You will need to render to an image sequence of some kind, which you can then convert to a movie file if you need.

General Information

Default ID	`ef7f0373-3542-4d1d-80d0-bb8599fa4c63`
Type Name and Shortcuts	`After Effects` `AE`

Parameters

Name	*Type*	*Get*	*Set*	*Meaning*	*Default*
Comp	Text	X	X	The name of the comp in the scene file to render	
Extra	Text	X	X	Extra commandline parameters you wish to pass to the renderer.	

Air

The **Air** Product allows you control the Air renderer. This is a Virtual Module. It includes all of the parameters from ProcessSequence.

General Information

Default ID	`1ee7351d-6d52-48f7-9bb8-df232d2faca4`
Type Name and Shortcuts	`Air`

Parameters

Name	*Type*	*Get*	*Set*	*Meaning*	*Default*
Extra	Parameters	X	X	Extra parameters you want to send on the command-line. Note: In order to be sure that the parameters are passed properly, enclose this entire list of parameters in quote marks. If you have only 1 parameter, you still need to enclose it in quote marks, and include a space somewhere in the string. (See bug database for more information.)	

Aqsis

The **Aqsis** Product allows you control the Aqsis command line renderer. This is a Virtual Module. It includes all of the parameters from ProcessSequence.

General Information

Default ID	89f86e8d-578e-4859-96b0-102153b79a32
Type Name and Shortcuts	Aqsis

Parameters

Name	*Type*	*Get*	*Set*	*Meaning*	*Default*
Crop	Multi	X	X	Define a crop window. Only the portion of the image inside the specified region will be rendered. The coordinates are in screen space, so a value of 0.0 is at the top resp. left and a value of 1.0 is at the right resp. bottom, irrespective of the actual resolution. Using this option is equivalent to the RIB command CropWindow x1 x2 y1 y2.	
Extra	Parameters	X	X	Extra parameters you want to send on the command-line. Note: In order to be sure that the parameters are passed properly, enclose this entire list of parameters in quote marks. If you have only 1 parameter, you still need to enclose it in quote marks, and include a space somewhere in the string. (See bug database for more information.)	
Res	Multi	X	X	Set the output image pixel resolution regardless of what is specified in the RIB.	

Alias

This Module is designed to control the Alias Studio commandline renderers. This is a Virtual Module. It includes all of the parameters from ProcessSequence.

General Information

Default ID	`374e7b28-645c-4385-a3Cb-c5fc7f45ac11`

Type Name and Shortcuts	`Alias Studio`
	`Alias`
	`Studio`

Parameters

Name	*Type*	*Get*	*Set*	*Meaning*	*Default*
Product	Choice	X	X	Which Alias renderer to use for the Job. Your choices are: Renderer RayTracer PowerCaster PowerTracer	PowerTracer

Arnold for Maya

The **Maya.sx** compiled Module now includes the ability to set up customized products for different renderers accessible through the Maya command line interface. Each Product is a section in the Maya.ini file which is in the Module folder. You can add your own or customize the existing ones in a similar manner to how you create Virtual Modules.

The **Arnold for Maya** Product is designed to control the Arnold for Maya plugin for Maya. This product is included in the Compiled Module **Maya.sx**. Using this product will override any Render Layer renderer settings in your Maya file.

The mental ray for Maya Product includes all of the parameters from Job, ProcessJob, RenderJob, and SequenceDistributor.

General Information

Default ID	`833c1fa9-fc0b-46da-920a-c4b74b92d5c1`

Type Name and Shortcuts	`Arnold for Maya`
	`a4m`
	`arnold4m`

Parameters

Name	*Type*	*Get*	*Set*	*Meaning*	*Default*
Extra	Parameters	X	X	Extra parameters you want to send on the command-line. Note: In order to be sure that the parameters are passed properly, enclose this entire list of parameters in quote marks. If you have only 1 parameter, you still need to enclose it in quote marks, and include a space somewhere in the string. (See bug database for more information.)	

Name	*Type*	*Get*	*Set*	*Meaning*	*Default*
mrVerbosity	Choice	X	X	This controls the amount of verbosity of the mental ray renderer (if it is being used by the scene file). You can choose from: "" - Use default verbosity "0" - No messages "1" - Fatal Errors Only "3" - Errors and Warnings Only "5" - Progress (including image filenames) Note you must use the highest verbosity for Smedge to detect the image filenames, but this can slow down renders that have a lot of output	
Project	Dir	X	X	The Maya project to use for rendering. If left blank, no project will be passed to the renderer.	
RenderDir	Dir	X	X	Override the output folder	
RenumberBy	Text	X	X	The increment value for renumbering the frame files. Blank defaults to 1	
RenumberStart	Text	X	X	The Job can be renumbered starting at this value. If left blank, the file frame numbers will correspond to the frames rendered	
SequenceBy	Text	X	X	Allows you to change the number of frames rendered in a sequence, using the **-b** flag. Make sure that your PacketSize is set to a whole number multiple of this value. If blank, defaults to 1	

Arnold for Maya (Single Frame)

The **Maya.sx** compiled Module now includes the ability to set up customized products for different renderers accessible through the Maya command line interface. Each Product is a section in the Maya.ini file which is in the Module folder. You can add your own or customize the existing ones in a similar manner to how you create Virtual Modules.

The **Arnold for Maya (Single Frame)** Product is designed to control the Arnold for Maya plug in for Maya. This product is included in the Compiled Module **Maya.sx**. Using this product will override any Render Layer renderer settings in your Maya file.

The Arnold for Maya (Single Frame) Product includes all of the parameters from Job, ProcessJob, RenderJob, and SliceDistributor. It adds the SingleFrame option in the product definition in order to provide the attributes used for splitting single frames

General Information

Default ID	`1ac408ad-ef9f-438f-94b3-55ffb719a1b1`

Type Name and Shortcuts	`Arnold for Maya Single Image`
	`a4m-1`
	`arnold4m-1`

Parameters

Name	*Type*	*Get*	*Set*	*Meaning*	*Default*
CompositeCommand	Text	X	X	This is the command that can be used to re-assemble the slices. No default is provided because Maya does not include a system that can composite every possible type of image that it can create. You can supply your own command, using job variable substitution to find and place the slices. If this is blank when the stitch work is ready to be sent, no automated composition of the image slices is done.	
Extra	Parameters	X	X	Extra parameters you want to send on the command-line. Note: In order to be sure that the parameters are passed properly, enclose this entire list of parameters in quote marks. If you have only 1 parameter, you still need to enclose it in quote marks, and include a space somewhere in the string. (See bug database for more information.)	

Name	*Type*	*Get*	*Set*	*Meaning*	*Default*
Project	Dir	X	X	The Maya project to use for rendering. If left blank, no project will be passed to the renderer.	
RegionEnd	Multi: •X •Y	X		The top, right corner of the current slice in pixels	
RegionEndPercent	Multi: •X •Y	X		The top, right corner of the current slice as a percentage (floating point between 0 and 1). *Note:* This value is currently only correctly calculated if the RenderRegion is the entire image size.	
RegionName	Text	X	X	The naming convention for each slice. You can reference other parameters in this text, and they will be properly expanded at render time.	Slice_ $(RegionStart.X)- $(RegionStart.Y)
RegionSize	Multi: •X •Y	X		The size of the current slice, in pixels.	
RegionSizePercent	Multi: •X •Y	X		The size of the current slice, in pixels.	
RegionStart	Multi: •X •Y	X		The bottom, left corner of the current slice in pixels	
RegionStartPercent	Multi: •X •Y	X		The bottom, left corner of the current slice as a percentage (floating point between 0 and 1). *Note:* This value is currently only correctly calculated if the RenderRegion is the entire image size.	
RenderDir	Dir	X	X	Override the output folder	
RenderRegion	Multi: •Left •Right •Bottom •Top	X	X	This is the total sub-region of the full image size that will be divided up to other machines. To render the full image size, use 0 for Left and Bottom, and the image width and height for Right and Top, respectively. Fields: Left, Right, Bottom, Top Separator: " "	
SliceImage	Multi: •Horizontal •Vertical	X	X	The number of sections, horizontally and vertically, to divide the image into.	

Name	*Type*	*Get*	*Set*	*Meaning*	*Default*
SequenceBy	Text	X	X	Allows you to change the number of frames rendered in a sequence, using the **-b** flag. Make sure that your PacketSize is set to a whole number multiple of this value. If blank, defaults to 1	
SliceOverlap	Int	X	X	The number of pixels of overlap between the slices. Useful for reducing edge artifacts on the slices.	50
Slices	Text	X		The total number of slices from the image that will be rendered. Equal to: $(SliceImage.Horizontal) x $(SliceImage.Vertical)	

Arnold Standalone

This Module is designed to control the Arnold standalone renderer (kick). This is a Virtual Module. It includes all of the parameters from ProcessSequence.

General Information

Default ID	A44B6595-1CA3-4657-8852-B5ADEE8FCED7
Type Name and Shortcuts	Arnold

Parameters

Name	*Type*	*Get*	*Set*	*Meaning*	*Default*
Extra	Parameters	X	X	Extra parameters you want to send on the command-line. Note: In order to be sure that the parameters are passed properly, enclose this entire list of parameters in quote marks. If you have only 1 parameter, you still need to enclose it in quote marks, and include a space somewhere in the string. (See bug database for more information.)	
Output	File	X	X	Override the output filename	
Verbose	Choice	X	X	Set the verbosity level of the renderer: 0 – 6	2

Blender

This Module is designed to control the Blender commandline renderer. This is a Virtual Module. It includes all of the parameters from ProcessSequence.

General Information

Default ID	`75316b6c-2510-44f1-b15d-d6a8f23a4c3f`
Type Name and Shortcuts	`Blender`

Parameters

Name	*Type*	*Get*	*Set*	*Meaning*	*Default*
Extra	Parameters	X	X	Extra parameters you want to send on the command-line. Note: In order to be sure that the parameters are passed properly, enclose this entire list of parameters in quote marks. If you have only 1 parameter, you still need to enclose it in quote marks, and include a space somewhere in the string. (See bug database for more information.)	

Cinema 4D

This Module is designed to control the Cinema 4D commandline renderer. This is a Virtual Module. It includes all of the parameters from ProcessSequence.

General Information

Default ID	`986184cd-04c1-451f-af66-f2947e405434`
Type Name and Shortcuts	`Cinema 4D` `C4d`

Parameters

Name	*Type*	*Get*	*Set*	*Meaning*	*Default*
Extra	Parameters	X	X	Extra parameters you want to send on the command-line. Note: In order to be sure that the parameters are passed properly, enclose this entire list of parameters in quote marks. If you have only 1 parameter, you still need to enclose it in quote marks, and include a space somewhere in the string. (See bug database for more information.)	

finalRender for Maya

The **Maya.sx** compiled Module now includes the ability to set up customized products for different renderers accessible through the Maya command line interface. Each Product is a section in the Maya.ini file which is in the Module folder. You can add your own or customize the existing ones in a similar manner to how you create Virtual Modules.

The **finalRender for Maya** Product allows you to tell Maya to render using the Maya Vector renderer. This will override any Render Layer renderer settings in your Maya 7 file.

The finalRender for Maya Product includes all of the parameters from Job, ProcessJob, RenderJob, and SequenceDistributor.

General Information

Default ID	`a9bdf783-810b-46ce-885e-90d37f2e4128`

Type Name and Shortcuts	`finalRender for Maya`
	`fr4m`
	`fr`

Parameters

Name	*Type*	*Get*	*Set*	*Meaning*	*Default*
Extra	Parameters	X	X	Extra parameters you want to send on the command-line. Note: In order to be sure that the parameters are passed properly, enclose this entire list of parameters in quote marks. If you have only 1 parameter, you still need to enclose it in quote marks, and include a space somewhere in the string. (See bug database for more information.)	
Project	Dir	X	X	The Maya project to use for rendering. If left blank, no project will be passed to the renderer.	
RenderDir	Dir	X	X	Override the output folder	
RenumberBy	Text	X	X	The increment value for renumbering the frame files. Blank defaults to 1	
RenumberStart	Text	X	X	The Job can be renumbered starting at this value. If left blank, the file frame numbers will correspond to the frames rendered	

Name	*Type*	*Get*	*Set*	*Meaning*	*Default*
SequenceBy	Text	X	X	Allows you to change the number of frames rendered in a sequence, using the **-b** flag. Make sure that your PacketSize is set to a whole number multiple of this value. If blank, defaults to 1	

FFmpeg

This Module is designed to control the FFmpeg renderer. This is a Virtual Module. It includes all of the parameters from ProcessSequence.

General Information

Default ID	`5fe84ba1-cbbf-49fd-a244-2732bfbdfb71`
Type Name and Shortcuts	`FFmpeg`

Parameters

Name	*Type*	*Get*	*Set*	*Meaning*	*Default*
GlobalOptions	Parameters	X	X	Extra parameters you want to send on the command line, which apply to the global settings of FFmpeg.	
InfileOptions	Parameters	X	X	Extra parameters you want to send on the command line, which apply to the input file settings of FFmpeg.	
OutfileOptions	Parameters	X	X	Extra parameters you want to send on the command line,which apply to the output file settings of FFmpeg.	
OutputFile	File	X	X	Specifies the full path and format string or file name used for the output file passed into Ffmpeg.	

Fryrender

This Module is designed to control the Fryrender command line renderer. This is a Compiled Module. It includes all of the parameters from Job, ProcessJob, RenderJob, and RepeatMergeDistributor.

Fryrender/dsimerge or Frycmd

The Fry.sx Module can handle the command line operation for both Fryrender using dsimerge to merge the DSI files together (version 1.0), and for Frycmd, which can also be used to merge DSI files (version 1.5). If you set the command line to the Fryrender executable, the command line will automatically be created with the extra flags required, and DSI merging will use the dsimerge command. If you set it to the Frycmd executable, that executable will be used for both the rendering and merging DSI files.

General Information

Default ID	`26BD6DFF-1F01-4CD3-A326-94722F132039`
Type Name and Shortcuts	`Fryrender` `Fry`

Parameters

Name	*Type*	*Get*	*Set*	*Meaning*	*Default*
Extra	Parameters	X	X	Extra parameters you want to send on the command-line. Note: In order to be sure that the parameters are passed properly, enclose this entire list of parameters in quote marks. If you have only 1 parameter, you still need to enclose it in quote marks, and include a space somewhere in the string. (See bug database for more information.)	
ImageOutput	File	X	X	Specifies the full path and name of the image file. The file name can refer to any of the multiple graphic formats supported.	

Fusion

This Module is designed to control the Alias Studio commandline renderers. This is a Virtual Module. It includes all of the parameters from ProcessSequence.

General Information

Default ID	`aefec858-eeb1-4e67-8b75-d147e0b60a0b`
Type Name and Shortcuts	`Fusion`

Parameters

Fusion currently does not add any more parameters than what is supplied by ProcessSequence.

Gelato

This Module is designed to control the Gelato renderer. This is a Virtual Module. It includes all of the parameters from ProcessSequence.

General Information

Default ID	`0e304886-b40b-4471-8e08-3f431442c3e1`
Type Name and Shortcuts	`Gelato`

Parameters

Name	*Type*	*Get*	*Set*	*Meaning*	*Default*
Extra	Parameters	X	X	Extra parameters you want to send on the command line. Note: In order to be sure that the parameters are passed properly, enclose this entire list of parameters in quote marks. If you have only 1 parameter, you still need to enclose it in quote marks, and include a space somewhere in the string. (See bug database for more information.)	

Generic Script

This Module is designed to allow the distribution of any command line. You can use it to distribute a range based sequence, such as a rendered animation sequence, or you can use it to distribute a command to execute exactly one time on every Engine that is a member of the assigned Pool. If you supply a **Range** value, the Master will use that range to break up the distribution just like every other range based Product. However, if you leave the **Range** blank, the work will be evenly distributed to every Engine in the pool. When you submit a Generic Script Job with **Submit**, you will still need to supply a –Range flag to the **Submit** command line, but you can leave its value blank. **This is different than other Products.**

Generic Script is a Compiled Module. It includes all of the parameters from Job, ProcessJob, and SequenceDistributor.

General Information

Default ID	`2c0ad30d-5432-44f3-8ab9-5d09d08e2955`

Type Name and Shortcuts	`Generic Script`
	`Generic`
	`Script`

Parameters

Name	*Type*			*Meaning*	*Default*
Command	Text	X	X	The command line to execute. The Engine will perform standard Smedge style variable substitution before launching the command. You can access other Job parameters using the **$(***Name***)** syntax.	

Houdini

This Module is designed to control the Houdini mantra command line renderer. This is a Virtual Module. It includes all of the parameters from ProcessSequence.

General Information

Default ID	`044da801-d94b-41bd-9861-ec429c7ee6e4`
Type Name and Shortcuts	`Houdini` `Mantra`

Parameters

Name	*Type*	*Get*	*Set*	*Meaning*	*Default*
Extra	Parameters	X	X	Extra parameters you want to send on the command-line. Note: In order to be sure that the parameters are passed properly, enclose this entire list of parameters in quote marks. If you have only 1 parameter, you still need to enclose it in quote marks, and include a space somewhere in the string. (See bug database for more information.)	
PythonFilter	File	X	X	This is the path to a Python script file that is used to do filtering of the Houdini IFD Files before rendering.	

Hrender

This Module is designed to control the Houdini hrender script command line renderer. This is a Virtual Module. It includes all of the parameters from ProcessSequence.

General Information

Default ID	`da3b4fc2-6a1c-439f-87d9-6fb1a9aad7de`
Type Name and Shortcuts	`Houdini Hrender` `Hrender`

Parameters

Name	*Type*	*Get*	*Set*	*Meaning*	*Default*
COP	Text	X	X	The COP name passed with the -c flag	
Driver	Text	X	X	The Output Driver passed with the -d flag	
Extra	Parameters	X	X	Extra parameters you want to send on the command-line. Note: In order to be sure that the parameters are passed properly, enclose this entire list of parameters in quote marks. If you have only 1 parameter, you still need to enclose it in quote marks, and include a space somewhere in the string. (See bug database for more information.)	
LicenseToken	Choice	X	X	Allows the renderer to use a graphics or non-graphics license token.	
RenderBy	Text	X	X	Allows you to set up the step frame to skip rendering every frame.	

imgcvt

This Module allows you to control the imgcvt image conversion program that is included with Maya. It is supplied by the Maya.sx compiled module, and includes all of the parameters from Job, ProcessJob, RenderJob, and SequenceDistributor. Note that this module is hard coded in the Maya module code, and is not configured by the Maya.ini file.

When using the GUI, if you select an existing frame file for the InSequence or OutSequence parameters, it will automatically convert that image to a format specifier.

General Information

Default ID	`fdae3386-7a20-460d-9795-a1adf0f8a841`
Type Name and Shortcuts	`imgcvt`

Parameters

Name	*Type*	*Get*	*Set*	*Meaning*	*Default*
InSequence	File	X	X	The input sequence	
Options	Text	X	X	Additional command line options	
OutSequence	File	X	X	The output sequence	

Indigo

This Module is designed to control the Indigo command line renderer. This is a Compiled Module. It includes all of the parameters from Job, ProcessJob, RenderJob, and RepeatMergeDistributor.

General Information

Default ID	`2602738b-7176-4d8d-aa5c-34ac1dce620d`
Type Name and Shortcuts	`Indigo`

Parameters

Name	*Type*	*Get*	*Set*	*Meaning*	*Default*
ImageOutput	File	X	X	Specifies the full path and name of the image file. The file name can refer to any of the multiple graphic formats supported.	
Settings	Parameters	X	X	Extra parameters you want to send on the command-line. Note: In order to be sure that the parameters are passed properly, enclose this entire list of parameters in quote marks. If you have only 1 parameter, you still need to enclose it in quote marks, and include a space somewhere in the string. (See bug database for more information.)	

Large File Transfer

This Module allows you to queue file operations in order to reduce your network load. File operations are queued separate from other Smedge workers, but can still be limited in the number of simultaneous operations allowed. This is a Compiled Module. It includes all of the parameters from Job.

General Information

Default ID	`3e732d37-a865-45f3-b2b2-3624db07ce2d`

Type Name and Shortcuts	`Large File Transfer`
	`lft`
	`copy`
	`move`

Parameters

Name	*Type*	*Get*	*Set*	*Meaning*	*Default*
Operation	Choice	X	X	The type of operation to perform. Choices are: 0 = Move/Rename a file 1 = Copy a file	1
Overwrite	Bool	X	X	Allow the target file to be overwritten if it exists	Yes
Source	File	X	X	The source file	
Target	File	X	X	The target file	

Lightwave

This Module is designed to control the Lightwave command line renderer. This is a Compiled Module. It includes all of the parameters from Job, ProcessJob, RenderJob, and SequenceDistributor.

This Module supports the creation of Dynamic Products with a simple INI file format. See the Dynamic Products chapter for more information.

General Information

Default ID	`0c09288e-7241-439c-af2b-b9954c61fb6d`

Type Name and Shortcuts	`Lightwave`
	`LW`
	`LWSN`

Parameters

Name	*Type*	*Get*	*Set*	*Meaning*	*Default*
ConfigFile	Dir	X	X	The full path to the Lightwave config files to use for rendering.	
ContentDir	Dir	X	X	The path to the content directory to use for rendering.	
SequenceBy	Text	X	X	Allows you to change the number of frames rendered in a sequence. Make sure that your **PacketSize** is set to a whole number multiple of this value. If blank, defaults to 1	

Maxwell Light Simulator

This Module is designed to control the Maxwell Light Simulator command line renderer. This is a Compiled Module. It includes all of the parameters from Job, ProcessJob, RenderJob, and RepeatMergeDistributor.

Maxwell Versions

The command line syntax and functionality of the rendering has changed significantly in various versions of Maxwell. Smedge currently supports the syntax for Maxwell 2.0 and later.

General Information

Default ID	`644d0701-8027-48e0-8bf8-ea8851a519f2`

Type Name and Shortcuts	`Maxwell Light Simulator`
	`Maxwell`
	`mx`
	`mxcl`

Parameters

Name	*Type*	*Get*	*Set*	*Meaning*	*Default*
Display	Choice	X	X	Select the options for displaying the Maxwell program while a render is running. (Note that you cannot display an interface when SmedgeEngine is running as a background Service). 0: Node license (always hidden) 1: Hidden (uses GUI license) 2: Console View (uses GUI license) 2: Full GUI display (uses GUI license)	0

Name	*Type*	*Get*	*Set*	*Meaning*	*Default*
Extra	Parameters	X	X	Extra parameters you want to send on the command line. Note: In order to be sure that the parameters are passed properly, enclose this entire list of parameters in quote marks. If you have only 1 parameter, you still need to enclose it in quote marks, and include a space somewhere in the string. (See bug database for more information.) Note that there are some extra parameters that are only allowed for Maxwell 2.0+. Smedge does not verify that you have specified flags in this parameter that are allowed for the command format you are using. Be sure to check that you are not using flags that are not allowed, or you will have work units that fail.	
ImageOutput	File	X	X	Specifies the full path and name of the image file. The file name can refer to any of the multiple graphic formats supported. In case of sequences, the output files will be numbered with a 4-digit suffix.	
Verbose	Choice	X	X	Sets the verbosity level from the renderer. One of: 0: None 1: Errors 2: Warnings 3: Info 4: All	3

Maya

The **Maya.sx** compiled Module now includes the ability to set up customized products for different renderers accessible through the Maya command line interface. Each Product is a section in the Maya.ini file which is in the Module folder. You can add your own or customize the existing ones in a similar manner to how you create Virtual Modules.

The default Maya product is designed for use with Maya 7.0 and later. It can handle the extended Render Layers feature new in Maya 7, which allows you to assign different renderers for each layer. The downside of this new system is that you have less control of the render settings via the Extra Parameters you can supply to the command line. If you are using Maya 6, or if you want more command line controls, you can use the **Maya Software** Product.

The Maya Product includes all of the parameters from Job, ProcessJob, RenderJob, and SequenceDistributor.

General Information

Default ID	`56f3b0da-a949-4c76-a21e-5bff03aca8af`
Type Name and Shortcuts	`Maya` `file`

Parameters

Name	*Type*	*Get*	*Set*	*Meaning*	*Default*
Extra	Parameters	X	X	Extra parameters you want to send on the command-line. Note: In order to be sure that the parameters are passed properly, enclose this entire list of parameters in quote marks. If you have only 1 parameter, you still need to enclose it in quote marks, and include a space somewhere in the string. (See bug database for more information.)	

Name	*Type*	*Get*	*Set*	*Meaning*	*Default*
mrVerbosity	Choice	X	X	This controls the amount of verbosity of the mental ray renderer (if it is being used by the scene file). You can choose from: "" - Use default verbosity "0" - No messages "1" - Fatal Errors Only "3" - Errors and Warnings Only "5" - Progress (including image filenames) Note you must use the highest verbosity for Smedge to detect the image filenames, but this can slow down renders that have a lot of output	
Project	Dir	X	X	The Maya project to use for rendering. If left blank, no project will be passed to the renderer.	
RenderDir	Dir	X	X	Override the output folder	
RenumberBy	Text	X	X	The increment value for renumbering the frame files. Blank defaults to 1	
RenumberStart	Text	X	X	The Job can be renumbered starting at this value. If left blank, the file frame numbers will correspond to the frames rendered	
SequenceBy	Text	X	X	Allows you to change the number of frames rendered in a sequence, using the **-b** flag. Make sure that your PacketSize is set to a whole number multiple of this value. If blank, defaults to 1	

Maya Hardware Renderer

The **Maya.sx** compiled Module now includes the ability to set up customized products for different renderers accessible through the Maya command line interface. Each Product is a section in the Maya.ini file which is in the Module folder. You can add your own or customize the existing ones in a similar manner to how you create Virtual Modules.

The **Maya Hardware** Product allows you to tell Maya to render using the Maya Hardware renderer. Note that hardware rendering is actually dependent upon the hardware installed on your Engines. Using this product will override any Render Layer renderer settings in your Maya 7 file.

The Maya Hardware Renderer Product includes all of the parameters from Job, ProcessJob, RenderJob, and SequenceDistributor.

General Information

Default ID	`bbf04770-cb4b-40b7-8d2d-ebee98c3a767`

Type Name and Shortcuts	`Maya Hardware Renderer`
	`Maya Hardware`
	`hw`

Parameters

Name	*Type*	*Get*	*Set*	*Meaning*	*Default*
Extra	Parameters	X	X	Extra parameters you want to send on the command-line. Note: In order to be sure that the parameters are passed properly, enclose this entire list of parameters in quote marks. If you have only 1 parameter, you still need to enclose it in quote marks, and include a space somewhere in the string. (See bug database for more information.)	
Project	Dir	X	X	The Maya project to use for rendering. If left blank, no project will be passed to the renderer.	
RenderDir	Dir	X	X	Override the output folder	
RenumberBy	Text	X	X	The increment value for renumbering the frame files. Blank defaults to 1	
RenumberStart	Text	X	X	The Job can be renumbered starting at this value. If left blank, the file frame numbers will correspond to the frames rendered	

Name	*Type*	*Get*	*Set*	*Meaning*	*Default*
SequenceBy	Text	X	X	Allows you to change the number of frames rendered in a sequence, using the **-b** flag. Make sure that your PacketSize is set to a whole number multiple of this value. If blank, defaults to 1	

Maya Lightmap Generator

The **Maya.sx** compiled Module now includes the ability to set up customized products for different renderers accessible through the Maya command line interface. Each Product is a section in the Maya.ini file which is in the Module folder. You can add your own or customize the existing ones in a similar manner to how you create Virtual Modules.

The **Maya Lightmap Generator** Product allows you to tell Maya to render using the metnal ray lightmap renderer. Using this product will override any Render Layer renderer settings in your Maya 7 file.

The Maya Product includes all of the parameters from Job, ProcessJob, RenderJob, and SequenceDistributor. **Note:** Currently the Maya lightmap renderer does not support overriding the frame range to render. You can only send this type of Job as a single whole work unit for the entire scene.

General Information

Default ID	`cb4dcb75-6330-47df-af47-cae6fe573389`

Type Name and Shortcuts	`Maya Lightmap Generator`
	`Lightmap`
	`lm`

Parameters

Name	*Type*	*Get*	*Set*	*Meaning*	*Default*
Extra	Parameters	X	X	Extra parameters you want to send on the command-line. Note: In order to be sure that the parameters are passed properly, enclose this entire list of parameters in quote marks. If you have only 1 parameter, you still need to enclose it in quote marks, and include a space somewhere in the string. (See bug database for more information.)	
Project	Dir	X	X	The Maya project to use for rendering. If left blank, no project will be passed to the renderer.	
RenderDir	Dir	X	X	Override the output folder	

Maya Software Renderer

The **Maya.sx** compiled Module now includes the ability to set up customized products for different renderers accessible through the Maya command line interface. Each Product is a section in the Maya.ini file which is in the Module folder. You can add your own or customize the existing ones in a similar manner to how you create Virtual Modules.

The **Maya Software** product is what used to be the normal Maya Product in Smedge 3 version 1.0.x. It actually forces the Maya command line to use the Maya Software renderer. If you are using Maya 6, you will need to use this Product if you want to render with the Maya software renderer, because the **Maya** product uses a render product switch that is not available in Maya 6. You can also use this product with Maya 7 or later if you want to force Maya to use the Maya Software renderer, or if you want the extended command line control over the render quality that you can get with the specific render products.

The Maya Software Renderer Product includes all of the parameters from Job, ProcessJob, RenderJob, and SequenceDistributor.

General Information

Default ID	`44d7ee77-ffe5-4f45-abe0-8aced1a7fae7`

Type Name and Shortcuts	`Maya Software Renderer`
	`Maya Software`
	`sw`

Parameters

Name	*Type*	*Get*	*Set*	*Meaning*	*Default*
Extra	Parameters	X	X	Extra parameters you want to send on the command-line. Note: In order to be sure that the parameters are passed properly, enclose this entire list of parameters in quote marks. If you have only 1 parameter, you still need to enclose it in quote marks, and include a space somewhere in the string. (See bug database for more information.)	
Project	Dir	X	X	The Maya project to use for rendering. If left blank, no project will be passed to the renderer.	
RenderDir	Dir	X	X	Override the output folder	
RenumberBy	Text	X	X	The increment value for renumbering the frame files. Blank defaults to 1	

Name	*Type*	*Get*	*Set*	*Meaning*	*Default*
RenumberStart	Text	X	X	The Job can be renumbered starting at this value. If left blank, the file frame numbers will correspond to the frames rendered	
SequenceBy	Text	X	X	Allows you to change the number of frames rendered in a sequence, using the **-b** flag. Make sure that your PacketSize is set to a whole number multiple of this value. If blank, defaults to 1	

Maya Software Renderer (Single Frame)

The **Maya.sx** compiled Module now includes the ability to set up customized products for different renderers accessible through the Maya command line interface. Each Product is a section in the Maya.ini file which is in the Module folder. You can add your own or customize the existing ones in a similar manner to how you create Virtual Modules.

The **Maya Software Renderer (Single Frame)** product is what used to be the normal Maya Product in Smedge 3 version 1.0.x. It actually forces the Maya command line to use the Maya Software renderer. If you are using Maya 6, you will need to use this Product if you want to render with the Maya software renderer, because the **Maya** product uses a render product switch that is not available in Maya 6. You can also use this product with Maya 7 or later if you want to force Maya to use the Maya Software renderer, or if you want the extended command line control over the render quality that you can get with the specific render products.

The Maya Software Renderer (Single Frame) Product includes all of the parameters from Job, ProcessJob, RenderJob, and SliceDistributor. It adds the SingleFrame option in the product definition in order to provide the attributes used for splitting single frames

General Information

Default ID	`2B2C5C75-6AA8-457f-B72D-3CAE698C7772`

Type Name and Shortcuts	`sw-1`
	`Maya Software Renderer Single Image`
	`Maya Software Single Image`
	`Maya Software Single`

Parameters

Name	*Type*	*Get*	*Set*	*Meaning*	*Default*
CompositeCommand	Text	X	X	This is the command that can be used to re-assemble the slices. No default is provided because Maya does not include a system that can composite every possible type of image that it can create. You can supply your own command, using job variable substitution to find and place the slices. If this is blank when the stitch work is ready to be sent, no automated composition of the image slices is done.	

Name	*Type*	*Get*	*Set*	*Meaning*	*Default*
Extra	Parameters	X	X	Extra parameters you want to send on the command line. Note: In order to be sure that the parameters are passed properly, enclose this entire list of parameters in quote marks. If you have only 1 parameter, you still need to enclose it in quote marks, and include a space somewhere in the string. (See bug database for more information.)	
Project	Dir	X	X	The Maya project to use for rendering. If left blank, no project will be passed to the renderer.	
RegionEnd	Multi: •X •Y	X		The top, right corner of the current slice in pixels	
RegionEndPercent	Multi: •X •Y	X		The top, right corner of the current slice as a percentage (floating point between 0 and 1). *Note:* This value is currently only correctly calculated if the RenderRegion is the entire image size.	
RegionName	Text	X	X	The naming convention for each slice. You can reference other parameters in this text, and they will be properly expanded at render time.	Slice_ $(RegionStart.X)- $(RegionStart.Y)
RegionSize	Multi: •X •Y	X		The size of the current slice, in pixels.	
RegionSizePercent	Multi: •X •Y	X		The size of the current slice, in pixels.	
RegionStart	Multi: •X •Y	X		The bottom, left corner of the current slice in pixels	
RegionStartPercent	Multi: •X •Y	X		The bottom, left corner of the current slice as a percentage (floating point between 0 and 1). *Note:* This value is currently only correctly calculated if the RenderRegion is the entire image size.	
RenderDir	Dir	X	X	Override the output folder	

Name	*Type*	*Get*	*Set*	*Meaning*	*Default*
RenderRegion	Multi: •Left •Right •Bottom •Top	X	X	This is the total sub-region of the full image size that will be divided up to other machines. To render the full image size, use 0 for Left and Bottom, and the image width and height for Right and Top, respectively. Fields: Left, Right, Bottom, Top Separator: " "	
SliceImage	Multi: •Horizontal •Vertical	X	X	The number of sections, horizontally and vertically, to divide the image into.	
SequenceBy	Text	X	X	Allows you to change the number of frames rendered in a sequence, using the **-b** flag. Make sure that your PacketSize is set to a whole number multiple of this value. If blank, defaults to 1	
SliceOverlap	Int	X	X	The number of pixels of overlap between the slices. Useful for reducing edge artifacts on the slices.	50
Slices	Text	X		The total number of slices from the image that will be rendered. Equal to: $(SliceImage.Horizontal) x $(SliceImage.Vertical)	

Maya to mental ray Exporter

The **Maya.sx** compiled Module now includes the ability to set up customized products for different renderers accessible through the Maya command line interface. Each Product is a section in the Maya.ini file which is in the Module folder. You can add your own or customize the existing ones in a similar manner to how you create Virtual Modules.

The **Maya to mental ray Exporter** Product allows you to tell Maya to convert the Maya scene to .mi files for use with the mental ray standalone renderer.

The Maya to mental ray Exporter Product includes all of the parameters from Job, ProcessJob, RenderJob, and SequenceDistributor.

General Information

Default ID	`17e64628-b48c-4464-86d6-3f3389758db9`

Type Name and Shortcuts	`Maya to mental ray Exporter`
	`MentalRay Exporter`
	`exporter`
	`mi`

Parameters

Name	*Type*	*Get*	*Set*	*Meaning*	*Default*
Extra	Parameters	X	X	Extra parameters you want to send on the command-line. Note: In order to be sure that the parameters are passed properly, enclose this entire list of parameters in quote marks. If you have only 1 parameter, you still need to enclose it in quote marks, and include a space somewhere in the string. (See bug database for more information.)	
Project	Dir	X	X	The Maya project to use for rendering. If left blank, no project will be passed to the renderer.	
RenderDir	Dir	X	X	Override the output folder	
RenumberBy	Text	X	X	The increment value for renumbering the frame files. Blank defaults to 1	
RenumberStart	Text	X	X	The Job can be renumbered starting at this value. If left blank, the file frame numbers will correspond to the frames rendered	

Name	*Type*	*Get*	*Set*	*Meaning*	*Default*
SequenceBy	Text	X	X	Allows you to change the number of frames rendered in a sequence, using the **-b** flag. Make sure that your PacketSize is set to a whole number multiple of this value. If blank, defaults to 1	

Maya Vector Renderer

The **Maya.sx** compiled Module now includes the ability to set up customized products for different renderers accessible through the Maya command line interface. Each Product is a section in the Maya.ini file which is in the Module folder. You can add your own or customize the existing ones in a similar manner to how you create Virtual Modules.

The **Maya Vector Renderer** Product allows you to tell Maya to render using the Maya Vector renderer. Using this product will override any Render Layer renderer settings in your Maya 7 file.

The Maya Vector Renderer Product includes all of the parameters from Job, ProcessJob, RenderJob, and SequenceDistributor.

General Information

Default ID	`fb5da2e3-22bf-41f9-adc4-e00fb6fbeed2`

Type Name and Shortcuts	`Maya Vector Renderer`
	`Maya Vector`
	`vr`

Parameters

Name	*Type*	*Get*	*Set*	*Meaning*	*Default*
Extra	Parameters	X	X	Extra parameters you want to send on the command-line. Note: In order to be sure that the parameters are passed properly, enclose this entire list of parameters in quote marks. If you have only 1 parameter, you still need to enclose it in quote marks, and include a space somewhere in the string. (See bug database for more information.)	
Project	Dir	X	X	The Maya project to use for rendering. If left blank, no project will be passed to the renderer.	
RenderDir	Dir	X	X	Override the output folder	
RenumberBy	Text	X	X	The increment value for renumbering the frame files. Blank defaults to 1	
RenumberStart	Text	X	X	The Job can be renumbered starting at this value. If left blank, the file frame numbers will correspond to the frames rendered	

Name	*Type*	*Get*	*Set*	*Meaning*	*Default*
SequenceBy	Text	X	X	Allows you to change the number of frames rendered in a sequence, using the **-b** flag. Make sure that your PacketSize is set to a whole number multiple of this value. If blank, defaults to 1	

mental ray for Maya

The **Maya.sx** compiled Module now includes the ability to set up customized products for different renderers accessible through the Maya command line interface. Each Product is a section in the Maya.ini file which is in the Module folder. You can add your own or customize the existing ones in a similar manner to how you create Virtual Modules.

The **mental ray for Maya** Product is designed to control the mental ray for Maya command line renderer included in versions Maya 6.0 and later. This product is included in the Compiled Module **Maya.sx**. Using this product will override any Render Layer renderer settings in your Maya file.

The mental ray for Maya Product includes all of the parameters from Job, ProcessJob, RenderJob, and SequenceDistributor.

General Information

Default ID	`43a9184b-2b5e-4abc-958c-2aa244a36019`

Type Name and Shortcuts	`mental ray for Maya`
	`MentalRay for Maya`
	`MR4M`

Parameters

Name	*Type*	*Get*	*Set*	*Meaning*	*Default*
Extra	Parameters	X	X	Extra parameters you want to send on the command-line. Note: In order to be sure that the parameters are passed properly, enclose this entire list of parameters in quote marks. If you have only 1 parameter, you still need to enclose it in quote marks, and include a space somewhere in the string. (See bug database for more information.)	

Name	*Type*	*Get*	*Set*	*Meaning*	*Default*
mrVerbosity	Choice	X	X	This controls the amount of verbosity of the mental ray renderer (if it is being used by the scene file). You can choose from: "" - Use default verbosity "0" - No messages "1" - Fatal Errors Only "3" - Errors and Warnings Only "5" - Progress (including image filenames) Note you must use the highest verbosity for Smedge to detect the image filenames, but this can slow down renders that have a lot of output	
Project	Dir	X	X	The Maya project to use for rendering. If left blank, no project will be passed to the renderer.	
RenderDir	Dir	X	X	Override the output folder	
RenumberBy	Text	X	X	The increment value for renumbering the frame files. Blank defaults to 1	
RenumberStart	Text	X	X	The Job can be renumbered starting at this value. If left blank, the file frame numbers will correspond to the frames rendered	
SequenceBy	Text	X	X	Allows you to change the number of frames rendered in a sequence, using the **-b** flag. Make sure that your PacketSize is set to a whole number multiple of this value. If blank, defaults to 1	

mental ray for Maya (Single Frame)

The **Maya.sx** compiled Module now includes the ability to set up customized products for different renderers accessible through the Maya command line interface. Each Product is a section in the Maya.ini file which is in the Module folder. You can add your own or customize the existing ones in a similar manner to how you create Virtual Modules.

The **mental ray for Maya (Single Frame)** Product is designed to control the mental ray for Maya command line renderer included in versions Maya 6.0 and later. This product is included in the Compiled Module **Maya.sx**. Using this product will override any Render Layer renderer settings in your Maya file.

The mental ray for Maya (Single Frame) Product includes all of the parameters from Job, ProcessJob, RenderJob, and SliceDistributor. It adds the SingleFrame option in the product definition in order to provide the attributes used for splitting single frames

General Information

Default ID	`99BCA4F7-1973-454E-99E4-A60871473997`

Type Name and Shortcuts	`mr4m-1`
	`mental ray for Maya Single Image`
	`MentalRay for Maya Single Image`
	`mental ray for Maya Single`
	`MentalRay for Maya Single`

Parameters

Name	*Type*	*Get*	*Set*	*Meaning*	*Default*
CompositeCommand	Text	X	X	This is the command that can be used to re-assemble the slices. No default is provided because Maya does not include a system that can composite every possible type of image that it can create. You can supply your own command, using job variable substitution to find and place the slices. If this is blank when the stitch work is ready to be sent, no automated composition of the image slices is done.	

Name	*Type*	*Get*	*Set*	*Meaning*	*Default*
Extra	Parameters	X	X	Extra parameters you want to send on the command-line. Note: In order to be sure that the parameters are passed properly, enclose this entire list of parameters in quote marks. If you have only 1 parameter, you still need to enclose it in quote marks, and include a space somewhere in the string. (See bug database for more information.)	
Project	Dir	X	X	The Maya project to use for rendering. If left blank, no project will be passed to the renderer.	
RegionEnd	Multi: •X •Y	X		The top, right corner of the current slice in pixels	
RegionEndPercent	Multi: •X •Y	X		The top, right corner of the current slice as a percentage (floating point between 0 and 1). *Note:* This value is currently only correctly calculated if the RenderRegion is the entire image size.	
RegionName	Text	X	X	The naming convention for each slice. You can reference other parameters in this text, and they will be properly expanded at render time.	Slice_ $(RegionStart.X)- $(RegionStart.Y)
RegionSize	Multi: •X •Y	X		The size of the current slice, in pixels.	
RegionSizePercent	Multi: •X •Y	X		The size of the current slice, in pixels.	
RegionStart	Multi: •X •Y	X		The bottom, left corner of the current slice in pixels	
RegionStartPercent	Multi: •X •Y	X		The bottom, left corner of the current slice as a percentage (floating point between 0 and 1). *Note:* This value is currently only correctly calculated if the RenderRegion is the entire image size.	
RenderDir	Dir	X	X	Override the output folder	

Name	*Type*	*Get*	*Set*	*Meaning*	*Default*
RenderRegion	Multi: •Left •Right •Bottom •Top	X	X	This is the total sub-region of the full image size that will be divided up to other machines. To render the full image size, use 0 for Left and Bottom, and the image width and height for Right and Top, respectively. Fields: Left, Right, Bottom, Top Separator: " "	
SliceImage	Multi: •Horizontal •Vertical	X	X	The number of sections, horizontally and vertically, to divide the image into.	
SequenceBy	Text	X	X	Allows you to change the number of frames rendered in a sequence, using the **-b** flag. Make sure that your PacketSize is set to a whole number multiple of this value. If blank, defaults to 1	
SliceOverlap	Int	X	X	The number of pixels of overlap between the slices. Useful for reducing edge artifacts on the slices.	50
Slices	Text	X		The total number of slices from the image that will be rendered. Equal to: $(SliceImage.Horizontal) x $(SliceImage.Vertical)	

mental ray Standalone

The **MentalRay Standalone** Product allows you control the mental ray standalone renderer. This is a Compiled Module. It includes all of the parameters from Job, ProcessJob, RenderJob, and SequenceDistributor.

This Module supports the creation of Dynamic Products with a simple INI file format. See the Dynamic Products chapter for more information.

General Information

Default ID	`bfe924c1-6d48-422c-bf71-a1305ef75437`

Type Name and Shortcuts	`MentalRay Standalone`
	`MR`
	`Ray`

Parameters

Name	*Type*	*Get*	*Set*	*Meaning*	*Default*
ActualScene	Text	X		Returns the full path to the actual scene file or files that will be handled in this work unit. For single .mi file scenes, it's the same as Scene, but for multiple .mi files, it will return a space separated list of files with the actual frames substituted in.	
Extra	Parameters	X	X	Extra parameters you want to send on the command-line. Note: In order to be sure that the parameters are passed properly, enclose this entire list of parameters in quote marks. If you have only 1 parameter, you still need to enclose it in quote marks, and include a space somewhere in the string. (See bug database for more information.)	
SceneNameFormat	Text	X	X	For sequences of .mi files, this is the formatting string that is used to determine the scene file names. Uses standard printf formatting. Leave blank if you have the entire scene in a single .mi file.	

Mistika VR

The **Mistika VR** Product allows you control the Mistika VR renderer. This is a Virtual Module. It includes all of the parameters from ProcessSequence.

General Information

Default ID	`08e71f90-2a07-431b-a820-f629d1f847fe`

Type Name and Shortcuts	`Mistika VR`
	`MistikaVR`
	`Mistika-VR`

Parameters

Mistika VR does not add any additional parameters.

Modo

This Module is designed to control Modo via the modo_cl command line interface. It sends a series of commands to open the scene and start the render using the input pipe, and includes the ability to add custom operations. It cannot currently detect the rendered image filenames directly. This is a Compiled Module. It includes all of the parameters from Job, ProcessJob, RenderJob, and SequenceDistributor.

This Module supports the creation of Dynamic Products with a simple INI file format. See the Dynamic Products chapter for more information.

General Information

Default ID	`4C5D8735-1D15-4BA0-9F38-B80A034A0FC0`
Type Name and Shortcuts	`Modo`

Parameters

Name	*Type*	*Get*	*Set*	*Meaning*	*Default*
Extra	Parameters	X	X	Extra parameters you want to send on the command-line. Note: In order to be sure that the parameters are passed properly, enclose this entire list of parameters in quote marks. If you have only 1 parameter, you still need to enclose it in quote marks, and include a space somewhere in the string. (See bug database for more information.)	
PassGroup	Text	X	X	Set the render pass group name to render. If empty, all passes are rendered.	
RenderOutputPath	Dir	X	X	Override the path where the rendered image files will be saved.	

Modo (Single Frame)

This Module is designed to control Modo via the modo_cl command line interface. It sends a series of commands to open the scene and start the render using the input pipe, and includes the ability to add custom operations. It cannot currently detect the rendered image filenames directly. This is a Compiled Module. It includes all of the parameters from Job, ProcessJob, RenderJob, and SliceDistributor.

This Module supports the creation of Dynamic Products with a simple INI file format. See the Dynamic Products chapter for more information.

This product allows single frame rendering by rendering regions of the frame on separate machines. It is also possible to configure the job to reassemble the regions into a single frame file if you have a tool to do so that can be controlled by a command line.

General Information

Default ID	`3fe20756-7028-4ca6-abfb-592eafbfad8d`
Type Name and Shortcuts	`Modo (Single Frame)`

Parameters

Name	*Type*	*Get*	*Set*	*Meaning*	*Default*
CompositeCommand	Text	X	X	This is the command that can be used to re-assemble the slices. No default is provided because Maya does not include a system that can composite every possible type of image that it can create. You can supply your own command, using job variable substitution to find and place the slices. If this is blank when the stitch work is ready to be sent, no automated composition of the image slices is done.	
Extra	Parameters	X	X	Extra parameters you want to send on the command-line. Note: In order to be sure that the parameters are passed properly, enclose this entire list of parameters in quote marks. If you have only 1 parameter, you still need to enclose it in quote marks, and include a space somewhere in the string. (See bug database for more information.)	
Overlap	Float	X	X	The amount of overlap of the slices, as a percentage value	0.01

Name	*Type*	*Get*	*Set*	*Meaning*	*Default*
PassGroup	Text	X	X	Set the render pass group name to render. If empty, all passes are rendered.	
RenderOutputPath	Dir	X	X	Override the path where the rendered image files will be saved.	
SliceEnd	Multi	X		The bottom right corner of the slice as a percentage of the image size (0 to 1) formatted as Right x Bottom	
SliceImage	Multi	X	X	Set the number of slices in the horizontal and vertical dimensions separately. Use a space to separate the values. The total number of slices is calculated by multiplying these two values together	
SliceName	Text	X	X	Set the naming convention for the slices. Default: `Slice_$(SliceStart.X)-$(SliceStart.Y)`	*(see left)*
SliceStart	Multi	X		The top left corner of the slice as a percentage of the image image size (0 to 1) formatted as Left x Top	

Nuke

This Module is designed to control Nuke via the commandline renderer. This is a Compiled Module. It includes all of the parameters from Job, ProcessJob, RenderJob, and SequenceDistributor.

This Module supports the creation of Dynamic Products with a simple INI file format. See the Dynamic Products chapter for more information.

Note that the automatic scene translation system is now part of the RenderJob base class. You can find more information about the parameters used to access and configure this system in the section on the RenderJob common parameters.

General Information

Default ID	`dfd3ea42-d630-4326-a33b-9b9125f9b7ea`
Type Name and Shortcuts	`Nuke`

Parameters

Name	*Type*	*Get*	*Set*	*Meaning*	*Default*
argv	Text	X	X	Anything here can be used by [argv n] expressions to provide changing arguments to the script. Each must start with a non-digit to avoid confusion with the frame ranges. See the Nuke command line documentation for more information.	
Extra	Parameters	X	X	Extra parameters you want to send on the command-line. Note: In order to be sure that the parameters are passed properly, enclose this entire list of parameters in quote marks. If you have only 1 parameter, you still need to enclose it in quote marks, and include a space somewhere in the string. (See bug database for more information.)	
InteractiveLicense	Bool	X	X	Use an interactive license instead of a render license	No
Verbose	Bool-Override	X	X	Turn on verbose output from Nuke, used to detect errors and rendered image files	Engine Default (yes)
WriteNode	Text	X	X	An optional write node to render. If you do not supply a value for this parameter, all write nodes in the Nuke scene file will be rendered.	

Pixar RenderMan

The **Pixar RenderMan** Product is designed to control the prman commandline renderer. This is a Virtual Module. It includes all of the parameters from ProcessSequence.

The PRMan Product handles sending sequences of RIB files to the prman standalone renderer. It uses the RIBFile parameter to determine how to locate the files in the sequence. The RIBFile parameter itself is a formatting parameter that uses the RIBBase folder parameter as a starting point, and can format an entire valid RIB filename to pass to the renderer.

Using the RIBFile parameter in the command line allows Smedge to perform in-file translation on the paths inside the RIB file itself, allowing cross platform rendering using the Smedge built-in mechanisms. The default commandline will copy the RIB file to the local temp folder at render time, translating paths internally, and run off that copy of the file. The files are cleaned up when the job finishes automatically.

General Information

Default ID	`cbf2137f-a69a-4f81-a826-e7d4140fe06b`

Type Name and Shortcuts	`Pixar RenderMan`
	`RenderMan`
	`prman`

Parameters

Name	*Type*	*Get*	*Set*	*Meaning*
Extra	Parameters	X	X	Extra parameters you want to send on the commandline. Note: In order to be sure that the parameters are passed properly, enclose this entire list of parameters in quote marks. If you have only 1 parameter, you still need to enclose it in quote marks, and include a space somewhere in the string. (See bug database for more information.)
RibBase	Dir	X	X	The base folder with the RIB folder sequence.
RibFile	Text	X	X	The formatting string for generating the full path to each RIB file. Default: `$(RibBase)/$(SubRange.Start.Pad)/$(SubRange.Start.Pad).rib`

REDline

This Module is designed to control the REDline RED camera image conversion utility. This is a Virtual Module. It includes all of the parameters from ProcessSequence.

General Information

Default ID	`f7696060-645c-4bd9-93d6-4dbe07c91610`
Type Name and Shortcuts	`REDline`

Parameters

Name	*Type*	*Get*	*Set*	*Meaning*	*Default*
Extra	Parameters	X	X	Extra parameters you want to send on the command line that control the settings for the resulting converted output files.	
LeftEye	File	X	X	Left Eye Input (for stereo conversion instead of a single camera conversion that uses the Scene parameter)	
Output	Text	X	X	The base name for the output image files from the conversion process.	
OutputDir	Dir	X	X	The folder where the converted output files are saved	
RenumberStart	Text	X	X	Optional number for offsetting the frame number in the converted output files.	
RightEye		X	X	Right Eye Input (for stereo conversion instead of a single camera conversion that uses the Scene parameter)	
Verbosity	Choice	X	X	Controls the output verbosity of the convert process. Options are "" (default) "--verbose" (verbose) "--errorsOnly" (errors only) and "--silent" (silent)	

Redshift for Maya

The **Maya.sx** compiled Module now includes the ability to set up customized products for different renderers accessible through the Maya command line interface. Each Product is a section in the Maya.ini file which is in the Module folder. You can add your own or customize the existing ones in a similar manner to how you create Virtual Modules.

The **Redshift for Maya** Product is designed to control the RedShift for Maya plug-in for Maya. This product is included in the Compiled Module **Maya.sx**. This will override any Render Layer renderer settings in your Maya 7 file.

The Redshift for Maya Product includes all of the parameters from Job, ProcessJob, RenderJob., and SequenceDistributor

General Information

Default ID	`959015f0-8857-4f04-b6d1-1c3007b0b11c`
Type Name and Shortcuts	`Redshift for Maya`

Parameters

Name	*Type*	*Get*	*Set*	*Meaning*	*Default*
Extra	Parameters	X	X	Extra parameters you want to send on the command-line. Note: In order to be sure that the parameters are passed properly, enclose this entire list of parameters in quote marks. If you have only 1 parameter, you still need to enclose it in quote marks, and include a space somewhere in the string. (See bug database for more information.)	
Project	Dir	X	X	The Maya project to use for rendering. If left blank, no project will be passed to the renderer.	
RenderDir	Dir	X	X	Override the output folder	
RenumberBy	Text	X	X	The increment value for renumbering the frame files. Blank defaults to 1	
RenumberStart	Text	X	X	The Job can be renumbered starting at this value. If left blank, the file frame numbers will correspond to the frames rendered	
SequenceBy	Text	X	X	Allows you to change the number of frames rendered in a sequence, using the **-b** flag. Make sure that your PacketSize is set to a whole number multiple of this value. If blank, defaults to 1	

RenderMan for Maya

The **Maya.sx** compiled Module now includes the ability to set up customized products for different renderers accessible through the Maya command line interface. Each Product is a section in the Maya.ini file which is in the Module folder. You can add your own or customize the existing ones in a similar manner to how you create Virtual Modules.

The **RenderMan for Maya** Product is designed to control the RenderMan for Maya plug-in for Maya. This product is included in the Compiled Module **Maya.sx**. This will override any Render Layer renderer settings in your Maya 7 file.

The RenderMan for Maya Product includes all of the parameters from Job, ProcessJob, RenderJob., and SequenceDistributor

General Information

Default ID	`6e5b4220-63ad-4a1b-b6a6-a7ccaa4ecf1a`
Type Name and Shortcuts	`RenderMan for Maya` `rm4m`

Parameters

Name	*Type*	*Get*	*Set*	*Meaning*	*Default*
Extra	Parameters	X	X	Extra parameters you want to send on the command-line. Note: In order to be sure that the parameters are passed properly, enclose this entire list of parameters in quote marks. If you have only 1 parameter, you still need to enclose it in quote marks, and include a space somewhere in the string. (See bug database for more information.)	
Project	Dir	X	X	The Maya project to use for rendering. If left blank, no project will be passed to the renderer.	
RenderDir	Dir	X	X	Override the output folder	
RenumberBy	Text	X	X	The increment value for renumbering the frame files. Blank defaults to 1	
RenumberStart	Text	X	X	The Job can be renumbered starting at this value. If left blank, the file frame numbers will correspond to the frames rendered	

Name	*Type*	*Get*	*Set*	*Meaning*	*Default*
SequenceBy	Text	X	X	Allows you to change the number of frames rendered in a sequence, using the **-b** flag. Make sure that your PacketSize is set to a whole number multiple of this value. If blank, defaults to 1	

RenderMan for Maya (Single Frame)

The **Maya.sx** compiled Module now includes the ability to set up customized products for different renderers accessible through the Maya command line interface. Each Product is a section in the Maya.ini file which is in the Module folder. You can add your own or customize the existing ones in a similar manner to how you create Virtual Modules.

The **RenderMan for Maya (Single Frame)** Product is designed to control the RenderMan for Maya plug-in for Maya. This product is included in the Compiled Module **Maya.sx**. Using this product will override any Render Layer renderer settings in your Maya file.

The RenderMan for Maya (Single Frame) Product includes all of the parameters from Job, ProcessJob, RenderJob, and SliceDistributor. It adds the SingleFrame option in the product definition in order to provide the attributes used for splitting single frames

General Information

Default ID	`92E88EB6-EFD3-4bfa-BBBD-00A79219F849`

Type Name and Shortcuts	`rm4m-1`
	`RenderMan for Maya Single Image`
	`RenderMan for Maya Single`

Parameters

Name	*Type*	*Get*	*Set*	*Meaning*	*Default*
CompositeCommand	Text	X	X	This is the command that can be used to re-assemble the slices. No default is provided because Maya does not include a system that can composite every possible type of image that it can create. You can supply your own command, using job variable substitution to find and place the slices. If this is blank when the stitch work is ready to be sent, no automated composition of the image slices is done.	
Extra	Parameters	X	X	Extra parameters you want to send on the command-line. Note: In order to be sure that the parameters are passed properly, enclose this entire list of parameters in quote marks. If you have only 1 parameter, you still need to enclose it in quote marks, and include a space somewhere in the string. (See bug database for more information.)	

Name	*Type*	*Get*	*Set*	*Meaning*	*Default*
Project	Dir	X	X	The Maya project to use for rendering. If left blank, no project will be passed to the renderer.	
RegionEnd	Multi: •X •Y	X		The top, right corner of the current slice in pixels	
RegionEndPercent	Multi: •X •Y	X		The top, right corner of the current slice as a percentage (floating point between 0 and 1). *Note:* This value is currently only correctly calculated if the RenderRegion is the entire image size.	
RegionName	Text	X	X	The naming convention for each slice. You can reference other parameters in this text, and they will be properly expanded at render time.	Slice_ $(RegionStart.X)- $(RegionStart.Y)
RegionSize	Multi: •X •Y	X		The size of the current slice, in pixels.	
RegionSizePercent	Multi: •X •Y	X		The size of the current slice, in pixels.	
RegionStart	Multi: •X •Y	X		The bottom, left corner of the current slice in pixels	
RegionStartPercent	Multi: •X •Y	X		The bottom, left corner of the current slice as a percentage (floating point between 0 and 1). *Note:* This value is currently only correctly calculated if the RenderRegion is the entire image size.	
RenderDir	Dir	X	X	Override the output folder	
RenderRegion	Multi: •Left •Right •Bottom •Top	X	X	This is the total sub-region of the full image size that will be divided up to other machines. To render the full image size, use 0 for Left and Bottom, and the image width and height for Right and Top, respectively. Fields: Left, Right, Bottom, Top Separator: " "	
SliceImage	Multi: •Horizontal •Vertical	X	X	The number of sections, horizontally and vertically, to divide the image into.	

Name	*Type*	*Get*	*Set*	*Meaning*	*Default*
SequenceBy	Text	X	X	Allows you to change the number of frames rendered in a sequence, using the **-b** flag. Make sure that your PacketSize is set to a whole number multiple of this value. If blank, defaults to 1	
SliceOverlap	Int	X	X	The number of pixels of overlap between the slices. Useful for reducing edge artifacts on the slices.	50
Slices	Text	X		The total number of slices from the image that will be rendered. Equal to: $(SliceImage.Horizontal) x $(SliceImage.Vertical)	

Rendition

The **Rendition** Product is designed to control the Rendition commandline renderer. This is a Virtual Module. It includes all of the parameters from ProcessSequence.

General Information

Default ID	`6cecdad0-061a-43e7-b36e-5a5f20431dcb`
Type Name and Shortcuts	`Rendition`

Parameters

Name	*Type*	*Get*	*Set*	*Meaning*	*Default*
Extra	Parameters	X	X	Extra parameters you want to send on the command-line. Note: In order to be sure that the parameters are passed properly, enclose this entire list of parameters in quote marks. If you have only 1 parameter, you still need to enclose it in quote marks, and include a space somewhere in the string. (See bug database for more information.)	
FileName	File	X	X	The filename of the rendered images	

Thea

This Module is designed to control the Thea command line renderer. This is a Compiled Module. It includes all of the parameters from Job, ProcessJob, RenderJob, and RepeatMergeDistributor.

General Information

Default ID	`EE01E431-A313-4315-80E1-27BA0493D2F6`
Type Name and Shortcuts	`Thea`

Parameters

Name	*Type*	*Get*	*Set*	*Meaning*	*Default*
ImageOutput	File	X	X	Specifies the full path and name of the image file. The file name can refer to any of the multiple graphic formats supported.	
MergeCommand	String	X	X	Use this to define the command line for merging files	
RenderCommand	String	X	X	Use this to define the command line for rendering	
Settings	Parameters	X	X	Extra parameters you want to send on the command-line. Note: In order to be sure that the parameters are passed properly, enclose this entire list of parameters in quote marks. If you have only 1 parameter, you still need to enclose it in quote marks, and include a space somewhere in the string. (See bug database for more information.)	

Turtle

This Module is designed to control the Turtle for Maya commandline renderer. This is a Virtual Module. It includes all of the parameters from ProcessSequence.

General Information

Default ID	`d472685f-d229-4d9b-9e75-270d7851ea5c`
Type Name and Shortcuts	`Turtle for Maya` `turtle`

Parameters

Name	*Type*	*Get*	*Set*	*Meaning*	*Default*
Extra	Parameters	X	X	Extra parameters you want to send on the command-line. Note: In order to be sure that the parameters are passed properly, enclose this entire list of parameters in quote marks. If you have only 1 parameter, you still need to enclose it in quote marks, and include a space somewhere in the string. (See bug database for more information.)	
SequenceBy	Text	X	X	Allows you to change the number of frames rendered in a sequence, using the **-frameStep** flag. Make sure that your PacketSize is set to a whole number multiple of this value. If blank, defaults to 1	

Viz

This Module is designed to control the Viz commandline renderer. This is a Virtual Module. It includes all of the parameters from ProcessSequence.

General Information

Default ID	d57cf46d-3df8-4239-a163-79f2e27951bf
Type Name and Shortcuts	Viz

Parameters

Name	*Type*	*Get*	*Set*	*Meaning*	*Default*
CommandFile	File	X	X	You can specify a file with your parameters instead of adding them manually using this option. If you use standard Smedge $(Name) syntax, you can have other parameters from the Job substituted into this filename when the commandline is actually generated	
Extra	Parameters	X	X	Extra parameters you want to send on the command-line. Note: In order to be sure that the parameters are passed properly, enclose this entire list of parameters in quote marks. If you have only 1 parameter, you still need to enclose it in quote marks, and include a space somewhere in the string. (See bug database for more information.)	
SequenceBy	Text	X	X	Allows you to change the number of frames rendered in a sequence, using the **-nthFrame** flag. Make sure that your PacketSize is set to a whole number multiple of this value. If blank, defaults to 1	

V-Ray for Maya

The **Maya.sx** compiled Module now includes the ability to set up customized products for different renderers accessible through the Maya command line interface. Each Product is a section in the Maya.ini file which is in the Module folder. You can add your own or customize the existing ones in a similar manner to how you create Virtual Modules.

The **V-Ray for Maya** Product is designed to control the V-Ray for Maya plug-in for Maya. This product is included in the Compiled Module **Maya.sx**. This will override any Render Layer renderer settings in your Maya 7 file.

The V-Ray for Maya Product includes all of the parameters from Job, ProcessJob, RenderJob, and SequenceDistributor.

General Information

Default ID	`31beca7c-49ea-4c14-93e9-4d0d7d225175`
Type Name and Shortcuts	`VRay for Maya` `vr4m`

Parameters

Name	*Type*	*Get*	*Set*	*Meaning*	*Default*
Extra	Parameters	X	X	Extra parameters you want to send on the command-line. Note: In order to be sure that the parameters are passed properly, enclose this entire list of parameters in quote marks. If you have only 1 parameter, you still need to enclose it in quote marks, and include a space somewhere in the string. (See bug database for more information.)	
Project	Dir	X	X	The Maya project to use for rendering. If left blank, no project will be passed to the renderer.	
RenderDir	Dir	X	X	Override the output folder	
RenumberBy	Text	X	X	The increment value for renumbering the frame files. Blank defaults to 1	
RenumberStart	Text	X	X	The Job can be renumbered starting at this value. If left blank, the file frame numbers will correspond to the frames rendered	

Name	*Type*	*Get*	*Set*	*Meaning*	*Default*
SequenceBy	Text	X	X	Allows you to change the number of frames rendered in a sequence, using the **-b** flag. Make sure that your PacketSize is set to a whole number multiple of this value. If blank, defaults to 1	

V-Ray for Maya (Single Frame)

The **Maya.sx** compiled Module now includes the ability to set up customized products for different renderers accessible through the Maya command line interface. Each Product is a section in the Maya.ini file which is in the Module folder. You can add your own or customize the existing ones in a similar manner to how you create Virtual Modules.

The **V-Ray for Maya (Single Frame)** Product is designed to control the V-Ray for Maya plug-in for Maya. This product is included in the Compiled Module **Maya.sx**. Using this product will override any Render Layer renderer settings in your Maya file.

The V-Ray for Maya (Single Frame) Product includes all of the parameters from Job, ProcessJob, RenderJob, and SliceDistributor. It adds the SingleFrame option in the product definition in order to provide the attributes used for splitting single frames

General Information

Default ID	`51C0B8BB-D6C5-4c7f-AC01-C827B3F1D9E1`

Type Name and Shortcuts	`vr4m-1`
	`V-Ray for Maya Single Image`
	`Vray for Maya Single Image`
	`V-Ray for Maya Single`
	`Vray for Maya Single`

Parameters

Name	*Type*	*Get*	*Set*	*Meaning*	*Default*
CompositeCommand	Text	X	X	This is the command that can be used to re-assemble the slices. No default is provided because Maya does not include a system that can composite every possible type of image that it can create. You can supply your own command, using job variable substitution to find and place the slices. If this is blank when the stitch work is ready to be sent, no automated composition of the image slices is done.	

Name	*Type*	*Get*	*Set*	*Meaning*	*Default*
Extra	Parameters	X	X	Extra parameters you want to send on the command-line. Note: In order to be sure that the parameters are passed properly, enclose this entire list of parameters in quote marks. If you have only 1 parameter, you still need to enclose it in quote marks, and include a space somewhere in the string. (See bug database for more information.)	
Project	Dir	X	X	The Maya project to use for rendering. If left blank, no project will be passed to the renderer.	
RegionEnd	Multi: •X •Y	X		The top, right corner of the current slice in pixels	
RegionEndPercent	Multi: •X •Y	X		The top, right corner of the current slice as a percentage (floating point between 0 and 1). *Note:* This value is currently only correctly calculated if the RenderRegion is the entire image size.	
RegionName	Text	X	X	The naming convention for each slice. You can reference other parameters in this text, and they will be properly expanded at render time.	Slice_ $(RegionStart.X)- $(RegionStart.Y)
RegionSize	Multi: •X •Y	X		The size of the current slice, in pixels.	
RegionSizePercent	Multi: •X •Y	X		The size of the current slice, in pixels.	
RegionStart	Multi: •X •Y	X		The bottom, left corner of the current slice in pixels	
RegionStartPercent	Multi: •X •Y	X		The bottom, left corner of the current slice as a percentage (floating point between 0 and 1). *Note:* This value is currently only correctly calculated if the RenderRegion is the entire image size.	
RenderDir	Dir	X	X	Override the output folder	

Name	*Type*	*Get*	*Set*	*Meaning*	*Default*
RenderRegion	Multi: •Left •Right •Bottom •Top	X	X	This is the total sub-region of the full image size that will be divided up to other machines. To render the full image size, use 0 for Left and Bottom, and the image width and height for Right and Top, respectively. Fields: Left, Right, Bottom, Top Separator: " "	
SliceImage	Multi: •Horizontal •Vertical	X	X	The number of sections, horizontally and vertically, to divide the image into.	
SequenceBy	Text	X	X	Allows you to change the number of frames rendered in a sequence, using the **-b** flag. Make sure that your PacketSize is set to a whole number multiple of this value. If blank, defaults to 1	
SliceOverlap	Int	X	X	The number of pixels of overlap between the slices. Useful for reducing edge artifacts on the slices.	50
Slices	Text	X		The total number of slices from the image that will be rendered. Equal to: $(SliceImage.Horizontal) x $(SliceImage.Vertical)	

V-Ray Standalone

This Module is designed to control the V-Ray standalone command line renderer. This is a Virtual Module. It includes all of the parameters from ProcessSequence.

General Information

Default ID	`f703ae62-8fbb-425f-9681-f4a22624480f`

Type Name and Shortcuts	`V-Ray Standalone`
	`V-Ray`
	`vray`

Parameters

Name	*Type*	*Get*	*Set*	*Meaning*	*Default*
AutoClose	Bool-Override	X	X	If **Display** is enabled, this tells the renderer to automatically close the V-Ray Frame Buffer when the render is complete.	Engine Default (Yes)
Display	Bool-Override	X	X	If enabled, this tells the renderer to open the V-Ray Frame Buffer and show the progress of the rendering in the window.	Engine Default (No)
Extra	Parameters	X	X	Extra parameters you want to send on the command-line. Note: In order to be sure that the parameters are passed properly, enclose this entire list of parameters in quote marks. If you have only 1 parameter, you still need to enclose it in quote marks, and include a space somewhere in the string. (See bug database for more information.)	
RT	Parameters	X	X	Extra parameters relating to the V-Ray RT rendering engine that you want to send on the commandline. Note: In order to be sure that the parameters are passed properly, enclose this entire list of parameters in quote marks. If you have only 1 parameter, you still need to enclose it in quote marks, and include a space somewhere in the string.	
SetFocus	Bool-Override	X	X	If **Display** is enabled, this tells the renderer to set the focus to the V-Ray Frame Buffer window during the render.	

Vue

This Module is designed to control the Vue renderbull command line renderer. This is a Virtual Module. It includes all of the parameters from ProcessSequence.

General Information

Default ID	`b88c0489-128b-4e4f-9d38-f25133857745`
Type Name and Shortcuts	`VUE Renderbull` `Vue`

Parameters

Name	*Type*	*Get*	*Set*	*Meaning*	*Default*
Output	Dir	X	X	The path to the rendered image files. You cannot override the filename, and there should be no slash at the end of this value.	

XSI

This Module is designed to control the XSI command line renderer. This is a Virtual Module. It includes all of the parameters from ProcessSequence.

General Information

Default ID	`b3a204c8-da3c-408b-b2c0-2d417e531ae2`
Type Name and Shortcuts	`XSI` `xsibatch`

Parameters

Name	*Type*	*Get*	*Set*	*Meaning*	*Default*
Extra	Parameters	X	X	Extra parameters you want to send on the command-line. Note: In order to be sure that the parameters are passed properly, enclose this entire list of parameters in quote marks. If you have only 1 parameter, you still need to enclose it in quote marks, and include a space somewhere in the string. (See bug database for more information.)	
Echo	Parameters	X	X	Custom parameters that can be set to control XSI to mental ray exporting. If you have only 1 parameter, you still need to enclose it in quote marks, and include a space somewhere in the string. (See bug database for more information.)	

Common Client Command Line Options

All Smedge client applications (everything except the Master) have a shared set of common command line options available:

`--release-date`	Show the release date of this version of Smedge in YYYY-MM-DD format.
`--release-date-ms`	Show the release date of this version of Smedge in milliseconds since Jan-1 year 0.
`--release-date-timet`	Show the release date of this version of Smedge as a Unix time_t value.
`--version`	Show the version of Smedge in human readable format.
`--version-dec`	Show the version of Smedge as a 32 bit decimal value
`--version-hex`	Show the version of Smedge as a 32 bit hexidecimal value.
`-ClientInterface` *interface-address*	Use this to override which network interface the client component will use for communication. The interface-address is the IP address assigned to the network interface. Using this command line flag will override the value set in the Connection.ini file (if any), which overrides the value set in the options for that component, which overrides the default, which is to let the OS pick the first available interface.
`-ClientPort` *port*	Override the port the client uses to listen. This flag overrides the setting from the `-ClientInterface` option, if supplied, and also from the Connection.ini file (if any), which overrides the value set in the options for that component, which overrides the default, which is to let the OS pick an ephemeral port.
`-ConnectTimeout` *float*	If no connection is established after the specified number of floating point seconds, the program will automatically terminate itself. The default is 0. Any value less than or equal to 0 will disable this time out, and the program will continue to wait for a connection indefinitely.
`-Heartbeat` *[float]*	Enables logging of the internal status of all threads in the component process to a heartbeat log file. Optionally supply a timeout period in seconds for the logging. If you don't supply a timeout, it will report every 15 seconds.

`-KeepAlive` *float*	Override the “keep alive” communication timeout for the client. Supply a timeout in floating point seconds. Be careful not to use different settings for the master than the clients, or else you will have communication issues and clients may go offline randomly.
`-LogCleanup` *seconds size*	Overrides the History.log file cleanup system. In order to keep the Smedge operational log files from filling up your hard drive, they are periodically culled. This flag allows you to override the period and the maximum log file size. Period is specified in seconds (default is 3600 or 1 hour), and size is specified in megabytes (default is 1 MB).
`-LogDisplayLevel` *integer*	This flag is deprecated and ignored. Smedge no longer logs to the display because of threading issues that can cause it to hang.
`-LogFileLevel` *integer*	Every application saves a log file with everything it does. This switch controls what level of information is saved in that file. Valid values are 0 – 6, where higher numbers are more detailed information. The default file level is 5.
`-LogFolder` *base-folder*	Override the machine Log folder, the base folder in which this application will create its logs.
`-LogMessages`	If set, the client will create additional log files that report every message sent and received by the component.
`-LogWithPID`	Add this flag to have the Smedge log files saved in a sub-folder with the process ID of the component process. This is useful when you have a command that may be executed several times on the same machine, even simultaneously (for example as event handler commands) to ensure that each instance of the process can independently save its log without interfering with any other instance of the process. This makes it easier to find and fix problems with these commands. Note that these logs are not cleaned up, so you should ensure that you clean up the file eventually to avoid consuming excessive amounts of disk space over time.
`-LogSubFolder` *name*	Requests for Smedge to store the log files in a sub-folder of the normal log folder that the process will use. Note that the string will be made into a single safe subfolder name.
`-LostClient` *float*	Override the timout for determing if a client has gone offline without notification. Be careful to keep this value consistent for all clients or you may notice communicaton issues and clients disconnecting seemingly at random.
`-Master` *master[:port]*	Override the automatic Master location system with a specific machine name and optional port.

`-MasterPort` *port*	Override the Master port only. (Default port is 6870)
`-ModulePath` *directory* [*directory ...*]	Add additional folders to scan for Modules. You can specify as many folders as you wish. If the directory path has a space anywhere in it, you should enclose the whole path in quotes.
`-OptionsFolder` *folder [folder]*	Add additional folders to scan for Options files. You can specify as many folders as you wish. If the directory path has a space anywhere in it, you should enclose the whole path in quotes. Added folders will be searched in the order given, and before any default folders.
`-ReportStatistics` *[seconds]*	Enables the statistics reporting functionality, which periodically dumps a log of internal memory usage, status of various queues and lists, and snapshot of every thread's current call stack. You can optionally specify a reporting period (in seconds) or it will run every 900 seconds (15 minutes) by default if you don't supply a timeout.
`-SendLogDump` *name|id*	Use this to request a specific component process running on a machine to initiate a log dump. You can specify the component by the name supplied to that component with the `-ListenForShutdown` command line flag, or by its process ID number (as reported by the operating system). The component you are using to send the request does not have to be the same component you are requesting to dump logs from, and all components, even the graphical ones, will respond to the request by dumping logs. The process you start with this command line option will immediately terminate after sending the dump request to the other process (regardless of whether the other component is running or receives the request).
`-SendShutdown` *name|id*	Use this to request a specific component process running on a machine to terminate itself. You can specify the component by the name supplied to that component with the `-ListenForShutdown` command line flag, or by its process ID number (as reported by the operating system). The component you are using to send the request does not have to be the same component you are requesting to terminate, and all components, even the graphical ones, will respond to the request by terminating. The process you start with this command line option will immediately terminate after sending the dump request to the other process (regardless of whether the other component is running or receives the request).
`-SendToProc` *name|id command*	Use this to send a custom request to a specific component process.You can specify the component by the name supplied to that component with the `-ListenForShutdown` command line flag, or by its process ID number (as reported by the operating system). The component you are using to send the request does not have to be the same component you are sending the request to. The process you start with

this command line option will immediately terminate after sending the request to the other process regardless of whether the other component is running or properly responds to the request. Known commands:

`DUMP`	Requests the process to dump its logs files
`HEARTBEAT`	Dump the current call stack locations for all threads
`IPFLUSH`	Force the process to flush its IP address name cache
`REPORT`	Report process usage and performance statistics
`STOP`	Shutdown the process

`-SetClientPreresolve` *bool*

Enables the client name to IP address pre-resolve system. This can speed up operation in certain environments. This command line switch will override the environment variable `SMEDGE_CLIENT_PRERESOLVE`.

`-SetProcAffinity` *bool*

Force Smedge to force the processor affinity to all available cores. This is required when using 32 bit Smedge on a machine with more than 32 cores. This command line switch will override the environment variable `SMEDGE_PROC_AFFINITY`. Note that this system is only functional on Windows currently.

`-SmedgeOptionsFile` *name*

Override the name of the options file to use for the process. By default, most components use an options file with the same name as the component name.

`-WaitForPID` *pid*

Requests that the component process wait for another process (specified by the process ID you supply) to complete before starting up.

Common INI File Options

All Smedge components understand the following options. You can set these options at various levels to override the behavior of Smedge while still allowing user customization. See the Rlib INI File Format chapter for more information.

```
    [ Communication ]

; The timeout in seconds for a client to establish a connection after trying to make contact
; default is 5 seconds
ClientConnectTimeout = float

; The interface for the client to listen on.
; Default is blank, which will attempt to bind to all available TCP interfaces.
ClientInterface = string

; The port that Clients will listen on for messages from the Master
; Default is 0, allowing the OS to choose a port for us
ClientPort = integer

; The amount of time in seconds to wait for a reply after broadcasting for a Master.
; The default is 1 second
LocateMasterTimeout = float

; Where to find the Master
; Separate multiple hosts with a space, and use * to add the automatic system into the list
; Default is blank, using automatic systems to find the Master
Master = string

; The TCP and UDP port that the Master will listen on for clients
; Default is 6870
MasterPort = integer

; The timeout during shutdowns for final communication and notifications to be cleared
; Default is 5 seconds
ShutdownTimeout = float


    [ Setup ]

; A semicolon separated list of paths to search for Smedge Modules at startup in addition to the
; default paths searched by the library.
; Default is blank, using only the system paths.
ModulePaths = directory[;directory...]
```

SmedgeMaster Reference

SmedgeMaster is the program that performs the role of the "Master" in a Smedge environment. Normally, you don't have to interact with this process. Sometimes, however, it is useful to configure its operation. For example, if you are debugging a Virtual Module, you may want to isolate yourself from your active system.

Command Line Interface

SmedgeMaster will respect most of the common command line parameters above. Additionally, it recognizes the following:

`-AllowMultiple`	Disable the single instance check, to allow multiple instances of SmedgeMaster to run on the same machine. **Warning:** be sure to also specify an alternate database location or data corruption or crashes can result.
`-Daemon`	Use this to notify Smedge on startup that it is starting as a Service or daemon. This should not be used directly, but should be part of the command line that starts the Service. `-Service` has the same effect.
`-Description` *text*	*Windows only.* Used with –InstallService. This sets the "Description" text in the Service database.
`-InstallService`	*Windows only.* Tells SmedgeMaster to install itself as a service with the system.
`-License` *license_code*	Override the license code that will be used by SmedgeMaster to allow Engines to work.
`-MasterDatabase` *directory*	Provide an alternate path for the Master to save its data.
`-MasterPort` *port*	Override the TCP port that the Master uses for communication. (Default port is 6870.)
`-MaxJobFailures` *integer*	Override the maximum allowed Job failure count

`-MaxTypeFailures` *integer*	Override the maximum allowed Product Type failure cont.
`-Name` *text*	*Windows only.* Used with –InstallService. This sets the “Name”of the Service in the Service database.
`-NoDefaultEngine`	Disables the customizable default Engine system. Only the hard-coded defaults built into the system will be available.
`-NoSleep`	Disables the power saving system in the Master. When present, no machines will be put to sleep, no matter what the settings and time delays are.
`-NoUDP`	Disables the automatic Master location responder thread. When present, the Master will not listen on the network for clients sending out requests to locate the master automatically. In order for your clients to find this master, you will have to specify the master information to the client, either with a Connection.ini, options, or command line flags.
`-Password` *text*	*Windows only.* Used with –InstallService. This sets the password for the account that the Service will run as.
`-RemoveService`	*Windows only.*Tells SmedgeMaster to remove itself from the service database.
`-Service`	Use this to notify Smedge on startup that it is starting as a Service or daemon. This should not be used directly, but should be part of the command line that starts the Service. `-Daemon` has the same effect.
`-StartService`	*Windows only.* Asks the Service Control Manager to start the SmedgeMaster service.
`-StopService`	*Windows only.* Asks the Service Control Manager to stop the SmedgeMaster service.
`-User` *name*	*Windows only.* Used with –InstallService. This sets the user account that the Service will run as.
`-Wait` [*seconds*]	*Windows only*. Used with –StartService and –StopService. This requests the process to wait for the Service process to report itself as fully started or shut down before returning. If you leave off this flag, the process will return immediately after requesting the Service to close, though the operation you requested may not yet have completed. You can optionally supply a maximum number of seconds to wait. If this timeout expires before the Service has reported its status, the service may not have

finished the requested operation. If you do not supply a timeout, the default timeout period is 30 seconds.

SmedgeMaster.ini options

These are all of the possible configuration values that SmedgeMaster will read from its options files. SmedgeMaster will first try to load these options from a copy of SmedgeMaster.ini in the Program folder. It will then load additional settings from a copy of SmedgeMaster.ini in the "machine" folder (on Windows this is usually `C:\Documents and Settings\All Users\Application Data\Uberware\SmedgeMaster`, on mac, this is `/Users/Shared/Smedge/SmedgeMaster`, and on Linux it is `/etc/smedge3/SmedgeMaster`), overriding any settings that may have been previously loaded. The Master will then save any settings that are configured dynamically while the program is running in the machine folder.

Many Master settings can be configured directly with the Shells. More advanced settings must be manually added to the file. You should only mess with the more advanced settings under the advisement of Uberware technical support, or if you really know what you are doing. Changing the values inappropriately can cause the Master to fail to operate normally.

Note that any changes you manually to the SmedgeMaster.ini file in the Machine folder while the SmedgeMaster process is running will be lost. In order to make changes while the process is running, you must use a Shell program that can interact with the Master, such as SmedgeGui or ConfigureMaster. ConfigureMaster includes an option to copy all settings from an INI file, which would allow you to set any setting listed here while the program is running. Changing some options will require you to restart the SmedgeMaster process to have it use the new value.

```
    [ Options ]

; This stores an encrypted form of the Administrator password.
Administrate = value

; Limits the number of repeated exceptions the main thread can encounter before aborting.
; Default is 25
ExceptionLimit = integer

; The license code is stored here
License = license code

; Allow you to change the folder where the Master will save its data on disk
MasterDatabase = directory

; Change the default message displayed in the SmedgeGui about box
WelcomeMessage = text to display
```

```
    [ Communication ]

; Amount of time to search for a Master until becoming the primary Master. Note that mirrors will use a multi-
ple
; of this amount based on their priority in the set of Mirrors to determine which will become primary Master
; when the current Master goes offline.
; Default is 5 seconds
BecomeMasterTimeout = float

; How often to look for lost clients (clients with no contact in the LostClientTimeout period).
; Default is every 7.5 seconds.
CheckForLostClients = float

; How often for the primary Master to check for other primary Masters on the network. This is the maximum time
; between checks. The minimum time is 5 seconds, and it grows exponentially to this maximum value.
; Default is 60 seconds
CheckForSplitNetwork = float

; Specify the network interface to use for communication by the Master.
ClientInterface = interface-address


; The amount of time with no communication from a client to send a KeepAliveMsg to validate the client.
; Default is 45 second
KeepAliveTimeout = float

; The amount of time after broadcasting for a duplicate master to wait for a response.
; Default is 1 second
LocateMasterTimeout = float

; If a client has not reported in after this amount of time since their last report, the client is considered
; lost If the client was an Engine, any work it was executing will be re-queued.
; Default is 5 KeepAliveTimeout periods
LostClientTimeout = float

; Interface that the Master will use to listen for clients.
; Default is blank (listening on all available interfaces)
Master = text

; TCP and UDP ports that the Master will use to listen for clients.
; Default is port 6870
MasterPort = port

; The maximum number of messages for the Master to process from a single client at one time.
; Higher values run faster and lower values improve parallel communication.
; Default is 100.
MaxReadAtOnce = integer

; The maximum number of messages for the Master to send to a single client at one time.
; Higher values run faster and lower values improve parallel communication.
; Default is 5000.
MaxSendAtOnce = integer

; Timeout period during shutdown to wait for final communication notifications to be sent.
; Default is 10 seconds
```

```
ShutdownTimeout = float

; Number of updates to cache for mirrors that reconnect to avoid getting the whole database refreshed.
; Default is 1500
UpdateCacheSize = integer


    [ Distribution ]

; If true, Jobs with a packet size greater than 1 can get smaller packets on machines with lower priority
AllowDynamicPacketSize = boolean

; Allows work to be distributed with fewer than the requested number of CPUs
AllowFewerCPUs = boolean


; The state of the Master mirror system.
; 0:    No mirrors allowed - Note any GUI that has started SmedgeMaster will stop it in this case
; 1:    All mirrors allowed
; 2:    Auto mirrors allowed (default) - A dynamic set of mirrors is kept online by the GUI
AllowMirrors = integer

; Tells the Master to automatically delete a finished Job as soon as it finishes
AutoDelete = boolean

; Sets the default "creator" string generation. Accepted values:
; 0: the current user name
; 1: the local machine name
; 2: user@machine
CreatorMode = integer

; Delays distribution of work until this many seconds after the last license is checked out.
; Allows time for Engines to connect and report any work that may have completed while disconnected from the
; Master before the Master starts redistributing that work, thinking it lost.
; Default is 15.0 seconds.
DelayDistribution = float

; Tells the Master to automatically delete work after the given number of hours. Ignored if AutoDelete is true.
; Default is 0, which means that the Jobs are never automatically deleted.
DeleteAfter = integer

; Amount of time to sleep between successive iterations of the distribution loop.
; Larger numbers use less CPU. Smaller numbers improve distribution performance.
; Default is .5 seconds.
DistributeFrequency = float

; A comma separated list of the names of computers that are allowed to be mirrors if AllowMirrors is false.
ExceptMirrors = text

; Distribution mode. 'true' is First-In, First-Out; 'false' is Round-Robin.
; Default is 'true'
FIFO = boolean

; The maximum number of work units to distribute during the global stagger start interval for all Products
```

```
; Default is 0, which means there is no limit, and no global stagger start.
GlobalStaggerCount = integer

; The global stagger start period, in seconds. Once this timeout has expired, more work can be started, up to
; the limit set in the GlobalStaggerCount above.
; Default is 0.0, which means there is no limit, and no global stagger start.
GlobalStaggerDelay = float
; If the master should use pool prioritization of work (false) or ignore pool priority (true)
; default is false
IgnorePoolPriority = boolean

; The master will release the license for the engine when that engine is disabled
; default is true (old behavior was is if it were false)
LicenseMode = boolean

; Tells the Master to dump the dispatch loop details into the Dispatch.log file. This value will be reset
; to false at the end of a dispatch loop in which it has been set to true.
LogDispatch = boolean

; Time after which dispatch logging is disabled (seconds).
; Default is 15 seconds
LogDispatchTime = boolean

; How long after an Engine disconnects until it is considered "lost" and re-queued.
; If the Engine re-connects before this period, its work is not lost.
; Default timeout is 150 seconds
LostWorkTimeout = integer

; Maximum number of times an Engine is allowed to fail on a single Job.
; Default is 5
MaxJobFailures = integer

; Maximum number of failures any Job can have in total before no more work is sent from that Job to any Engine
; default is 25
MaxTotalJobFailures = integer

; Maximum number of Jobs of the same Product an Engine is allowed to fail before that Product is disabled.
; Default is 5.
MaxTypeFailures = integer

; Memory Distribution mode:
; 0 = Processors Only (default - like old versions)
; 1 = Memory only
; 2 = Both Processors and Memory
MemoryDistribution = integer

; Minimum elapsed work time to count as part of the average work time for a Job.
; Default is 5.0 seconds.
MinimumTimeForAverage = float

; Tells the Master to automatically delete finished work as soon as it finishes
OutputFileCleanup = integer

; Allows work to be distributed to an Engine with the requested number of CPUs, even if the Engine does not
have
; that many CPUs actually available for work
```

```
OverloadEngine = boolean

; If true, no work will be distributed from Jobs with a "Priority" of 0.
; If false, priority 0 is just the lowest possible priority
PriorityZeroIsPaused = boolean


; If true, the Master will always reset the Job Creation time to the current local system time when the Job is
; created. If false, the Master will keep the Job creation time that the Shell submitting the Job has set the
; as the Creation time
ResetCreationTime = boolean

; A number of seconds for a work unit to start. If the Engine does not report the work started within this
; timeout period, the Master will assume there was some problem, and re-queue the work.
; Set this to 0 to disable this test.
StartWorkTimeout = integer

; If true, all Engines are part of the "Whole System" pool by default.
; If false, there is no "Whole System" pool.
UseWholeSystem = boolean


    [ Limit ]

; Entries in this section set the limit for the total number of workers from a given type that are allowed
; to be outstanding at one time. The Syntax is <type> = <limit>. The default for a type if it is not provided
; is -1, or no limit.

typeid-string = integer


    [ Restrictions ]

; Entries in this section establish the restrictions on SmedgeGui's functionality.
; The syntax is <restriction> = <Boolean>. If the Boolean value is true, then the restriction is in place.
; If the Boolean value is false, or if the restriction name does not appear in this section at all,
; the functionality is available to all SmedgeGui users

restriction-name = boolean
```

SmedgeEngine is the program that performs the work in a Smedge environment. Normally, you don't have to interact with this process. Sometimes, however, it is useful to configure its operation. SmedgeEngine will respect most of the common command line parameters above. Additionally, it recognizes the following:

`-AllowMultiple` [*name*]	Runs the Engine in a virtual test mode, ignoring single instance check, machine ID, name, and the number of cores. The ID will be generated uniquely for the run. If you supply the optional name parameter, that is the machine name that will be used, otherwise, it will generate one with the process ID. The virtual machines always have 64 cores available. ***Use this option with caution.*** It is for simulating larger machines and networks when developing custom Products and components.
`-Daemon`	Use this to notify Smedge on startup that it is starting as a Service or daemon. This should not be used directly, but should be part of the command line that starts the Service. `-Service` has the same effect.
`-DeleteOutputAfter` *days*	Override the amount of time that the Engine will keep captured output files in the local log folder. This command line flag will override the Master option for this Engine. If not set, the Engine will either use the value set by the last connected Master, or the default (30 days) if the Engine has never been connected.
`-Description` *text*	*Windows only.* Used with –InstallService. This sets the “Description” text in the Service database.
`-InstallService`	*Windows only.* Tells SmedgeEngine to install itself as a service with the system.
`-Name` *text*	*Windows only.* Used with –InstallService. This sets the “Name”of the Service in the Service database.
`-Password` *text*	*Windows only.* Used with –InstallService. This sets the password for the account that the Service will run as.
`-RemoveService`	*Windows only.* Tells SmedgeEngine to remove itself from the service database.

`-Service`	Use this to notify Smedge on startup that it is starting as a Service or daemon. This should not be used directly, but should be part of the command line that starts the Service. `-Daemon` has the same effect.
`-StartService`	*Windows only.* Asks the Service Control Manager to start the SmedgeEngine service.
`-StopService`	*Windows only.* Asks the Service Control Manager to stop the SmedgeEngine service.
`-User` *name*	*Windows only.* Used with –InstallService. This sets the user account that the Service will run as.
`-Wait` [*seconds*]	*Windows only*. Used with –StartService and –StopService. This requests the process to wait for the Service process to report itself as fully started or shut down before returning. If you leave off this flag, the process will return immediately after requesting the Service to close, and the operation you requested may not yet have completed. You can optionally supply a maximum number of seconds to wait. If this timeout expires before the Service has reported its status, the service may not have finished the requested operation. If you do not supply a timeout, the default timeout period is 30 seconds.

Using Engine to control the local machine

The new commandline shell Engine, included in Smedge 3 version 1.5.0, allows you to control any Engine with simple commands via a command prompt window, or a script. You can use this in your own network to integrate more extensive control of Smedge.

Engine allows you to specify one or more Engines, either by name or by their ID, to have the command operate on those Engines. However, it also allows you to leave off the Engine to control, in which case it tries to control an Engine running on the local machine.

This means you can create two handy shortcuts to allow users to enable or disable the Engine running on their local machine. Create a shortcut and make the commandline:

```
Engine Enable
```

(You may need to put the full path to Engine.exe if it is not in your PATH environment variable). Whenever a user launches this short-cut, it will try to enable the Engine running on their local machine. Similarly you can make another shortcut with the commandline:

```
Engine Disable
```

This will allow them to disable their machine. You can optionally specify whether the disable is immediate or delayed (immediate will attempt to terminate any currently going work, and delayed will allow it to finish). Delayed is the default if you don't specify anything, but to specify immediate make your commandline:

```
Engine Disable true
```

On Windows, if you want to use Engine Enable as part of a Log-Off script, you may want to use the Engine service mode. This mode enables the Engine shell to send the requests asynchronously, which means that when a user logs off, she doesn’t have to wait for the Engine shell to connect and successfully send the request. Instead, the request is queued to be sent by a service, and the log off process can proceed more quickly. To enable this service, you must install the Engine shell as a service on the machine.

Check out the Engine help by typing **`Engine`** with no parameters into the commandline.

Maintain complex Job hierarchies in a Job file

Job Files (**.sj** files) can contain any number of Jobs. When the Job File is loaded, any dependencies between jobs within the file will be correctly maintained when the file is loaded, every time it is loaded, even though the jobs will have new IDs each time.

This is useful for creating a complex render that includes preparation or post-processing steps. For example, you could have a 3-D render, then run through a composite, then make a QuickTime movie from the composite frames, making sure that each job waits for the previous to be complete before beginning.

Because the IDs are changed each time, you can repeatedly submit the same file over and over and a new set of jobs will be created each time. This is another way you can "batch submit" a bunch of Jobs at once, even if you don't need to maintain dependencies between them.

This software uses Boost C++ libraries, distributed under the Boost License:

Boost Software License - Version 1.0 - August 17th, 2003

Permission is hereby granted, free of charge, to any person or organization obtaining a copy of the software and accompanying documentation covered by this license (the "Software") to use, reproduce, display, distribute, execute, and transmit the Software, and to prepare derivative works of the Software, and to permit third-parties to whom the Software is furnished to do so, all subject to the following:

The copyright notices in the Software and this entire statement, including the above license grant, this restriction and the following disclaimer, must be included in all copies of the Software, in whole or in part, and all derivative works of the Software, unless such copies or derivative works are solely in the form of machine-executable object code generated by a source language processor.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON-INFRINGEMENT. IN NO EVENT SHALL THE COPYRIGHT HOLDERS OR ANYONE DISTRIBUTING THE SOFTWARE BE LIABLE FOR ANY DAMAGES OR OTHER LIABILITY, WHETHER IN CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

This software uses the Portable Hardware Locality (hwloc) library, distributed under the **New BSD license**:

```
Copyright © 2009 CNRS
Copyright © 2009 inria.  All rights reserved.
Copyright © 2009 Université Bordeaux 1
Copyright © 2009 Cisco Systems, Inc.  All rights reserved.
See COPYING in top-level directory.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions
are met:
1. Redistributions of source code must retain the above copyright
   notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright
   notice, this list of conditions and the following disclaimer in the
   documentation and/or other materials provided with the distribution.
3. The name of the author may not be used to endorse or promote products
   derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE AUTHOR ``AS IS'' AND ANY EXPRESS OR
IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES
OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED.
IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR ANY DIRECT, INDIRECT,
INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
(INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF
THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
```

This software uses liblzh:

```
Copyright (c) 2000-2009 Marc Alexander Lehmann <schmorp@schmorp.de>

Redistribution and use in source and binary forms, with or without modifica-
tion, are permitted provided that the following conditions are met:

  1.  Redistributions of source code must retain the above copyright notice,
      this list of conditions and the following disclaimer.

  2.  Redistributions in binary form must reproduce the above copyright
      notice, this list of conditions and the following disclaimer in the
      documentation and/or other materials provided with the distribution.

THIS SOFTWARE IS PROVIDED BY THE AUTHOR ``AS IS'' AND ANY EXPRESS OR IMPLIED
WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MER-
CHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED.  IN NO
EVENT SHALL THE AUTHOR BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPE-
CIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS;
OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY,
WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTH-
ERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
OF THE POSSIBILITY OF SUCH DAMAGE.
```

This software uses wxWidgets:

```
                wxWindows Library Licence, Version 3.1
                ======================================

  Copyright (C) 1998-2005 Julian Smart, Robert Roebling et al

  Everyone is permitted to copy and distribute verbatim copies
  of this licence document, but changing it is not allowed.

                        WXWINDOWS LIBRARY LICENCE
     TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

  This library is free software; you can redistribute it and/or modify it
  under the terms of the GNU Library General Public Licence as published by
  the Free Software Foundation; either version 2 of the Licence, or (at
  your option) any later version.

  This library is distributed in the hope that it will be useful, but
  WITHOUT ANY WARRANTY; without even the implied warranty of
  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU Library
  General Public Licence for more details.

  You should have received a copy of the GNU Library General Public Licence
  along with this software, usually in a file named COPYING.LIB.  If not,
  write to the Free Software Foundation, Inc., 59 Temple Place, Suite 330,
  Boston, MA 02111-1307 USA.

  EXCEPTION NOTICE

  1. As a special exception, the copyright holders of this library give
  permission for additional uses of the text contained in this release of
  the library as licenced under the wxWindows Library Licence, applying
  either version 3.1 of the Licence, or (at your option) any later version of
  the Licence as published by the copyright holders of version
  3.1 of the Licence document.

  2. The exception is that you may use, copy, link, modify and distribute
  under your own terms, binary object code versions of works based
  on the Library.
```

This software uses ZeroMQ, which is released under the GNU Lesser General Public License:

GNU GENERAL PUBLIC LICENSE

Version 3, 29 June 2007

Copyright © 2007 Free Software Foundation, Inc. <http://fsf.org/>

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

Preamble

The GNU General Public License is a free, copyleft license for software and other kinds of works.

The licenses for most software and other practical works are designed to take away your freedom to share and change the works. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change all versions of a program--to make sure it remains free software for all its users. We, the Free Software Foundation, use the GNU General Public License for most of our software; it applies also to any other work released this way by its authors. You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for them if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs, and that you know you can do these things.

To protect your rights, we need to prevent others from denying you these rights or asking you to surrender the rights. Therefore, you have certain responsibilities if you distribute copies of the software, or if you modify it: responsibilities to respect the freedom of others.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must pass on to the recipients the same freedoms that you received. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

Developers that use the GNU GPL protect your rights with two steps: (1) assert copyright on the software, and (2) offer you this License giving you legal permission to copy, distribute and/or modify it.

For the developers' and authors' protection, the GPL clearly explains that there is no warranty for this free software. For both users' and authors' sake, the GPL requires that modified versions be marked as changed, so that their problems will not be attributed erroneously to authors of previous versions.

Some devices are designed to deny users access to install or run modified versions of the software inside them, although the manufacturer can do so. This is fundamentally incompatible with the aim of protecting users' freedom to change the software. The systematic pattern of such abuse occurs in the area of products for individuals to use, which is precisely where it is most unacceptable. Therefore, we have designed this version of the GPL to prohibit the practice for those products. If such problems arise substantially in other domains, we stand ready to extend this provision to those domains in future versions of the GPL, as needed to protect the freedom of users.

Finally, every program is threatened constantly by software patents. States should not allow patents to restrict development and use of software on general-purpose computers, but in those that do, we wish to avoid the special danger that patents applied to a free program could make it effectively proprietary. To prevent this, the GPL assures that patents cannot be used to render the program non-free.

The precise terms and conditions for copying, distribution and modification follow.

TERMS AND CONDITIONS

0. Definitions.

"This License" refers to version 3 of the GNU General Public License.

"Copyright" also means copyright-like laws that apply to other kinds of works, such as semiconductor masks.

"The Program" refers to any copyrightable work licensed under this License. Each licensee is addressed as "you". "Licensees" and "recipients" may be individuals or organizations.

To "modify" a work means to copy from or adapt all or part of the work in a fashion requiring copyright permission, other than the making of an exact copy. The resulting work is called a "modified version" of the earlier work or a work "based on" the earlier work.

A "covered work" means either the unmodified Program or a work based on the Program.

To "propagate" a work means to do anything with it that, without permission, would make you directly or secondarily liable for infringement under applicable copyright law, except executing it on a computer or modifying a private copy. Propagation includes copying, distribution (with or without modification), making available to the public, and in some countries other activities as well.

To "convey" a work means any kind of propagation that enables other parties to make or receive copies. Mere interaction with a user through a computer network, with no transfer of a copy, is not conveying.

An interactive user interface displays "Appropriate Legal Notices" to the extent that it includes a convenient and prominently visible feature that (1) displays an appropriate copyright notice, and (2) tells the user that there is no warranty for the work (except to the extent that warranties are provided), that licensees may convey the work under this License, and how to view a copy of this License. If the interface presents a list of user commands or options, such as a menu, a prominent item in the list meets this criterion.

1. Source Code.
The "source code" for a work means the preferred form of the work for making modifications to it. "Object code" means any non-source form of a work.

A "Standard Interface" means an interface that either is an official standard defined by a recognized standards body, or, in the case of interfaces specified for a particular programming language, one that is widely used among developers working in that language.

The "System Libraries" of an executable work include anything, other than the work as a whole, that (a) is included in the normal form of packaging a Major Component, but which is not part of that Major Component, and (b) serves only to enable use of the work with that Major Component, or to implement a Standard Interface for which an implementation is available to the public in source code form. A "Major Component", in this context, means a major essential component (kernel, window system, and so on) of the specific operating system (if any) on which the executable work runs, or a compiler used to produce the work, or an object code interpreter used to run it.

The "Corresponding Source" for a work in object code form means all the source code needed to generate, install, and (for an executable work) run the object code and to modify the work, including scripts to control those activities. However, it does not include the work's System Libraries, or general-purpose tools or generally available free programs which are used unmodified in performing those activities but which are not part of the work. For example, Corresponding Source includes interface definition files associated with source files for the work, and the source code for shared libraries and dynamically linked subprograms that the work is specifically designed to require, such as by intimate data communication or control flow between those subprograms and other parts of the work.

The Corresponding Source need not include anything that users can regenerate automatically from other parts of the Corresponding Source.

The Corresponding Source for a work in source code form is that same work.

2. Basic Permissions.
All rights granted under this License are granted for the term of copyright on the Program, and are irrevocable provided the stated conditions are met. This License explicitly affirms your unlimited permission to run the unmodified Program. The output

from running a covered work is covered by this License only if the output, given its content, constitutes a covered work. This License acknowledges your rights of fair use or other equivalent, as provided by copyright law.

You may make, run and propagate covered works that you do not convey, without conditions so long as your license otherwise remains in force. You may convey covered works to others for the sole purpose of having them make modifications exclusively for you, or provide you with facilities for running those works, provided that you comply with the terms of this License in conveying all material for which you do not control copyright. Those thus making or running the covered works for you must do so exclusively on your behalf, under your direction and control, on terms that prohibit them from making any copies of your copyrighted material outside their relationship with you.

Conveying under any other circumstances is permitted solely under the conditions stated below. Sublicensing is not allowed; section 10 makes it unnecessary.

3. Protecting Users' Legal Rights From Anti-Circumvention Law.
No covered work shall be deemed part of an effective technological measure under any applicable law fulfilling obligations under article 11 of the WIPO copyright treaty adopted on 20 December 1996, or similar laws prohibiting or restricting circumvention of such measures.

When you convey a covered work, you waive any legal power to forbid circumvention of technological measures to the extent such circumvention is effected by exercising rights under this License with respect to the covered work, and you disclaim any intention to limit operation or modification of the work as a means of enforcing, against the work's users, your or third parties' legal rights to forbid circumvention of technological measures.

4. Conveying Verbatim Copies.
You may convey verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice; keep intact all notices stating that this License and any non-permissive terms added in accord with section 7 apply to the code; keep intact all notices of the absence of any warranty; and give all recipients a copy of this License along with the Program.

You may charge any price or no price for each copy that you convey, and you may offer support or warranty protection for a fee.

5. Conveying Modified Source Versions.
You may convey a work based on the Program, or the modifications to produce it from the Program, in the form of source code under the terms of section 4, provided that you also meet all of these conditions:

a) The work must carry prominent notices stating that you modified it, and giving a relevant date.
b) The work must carry prominent notices stating that it is released under this License and any conditions added under section 7. This requirement modifies the requirement in section 4 to “keep intact all notices”.
c) You must license the entire work, as a whole, under this License to anyone who comes into possession of a copy. This License will therefore apply, along with any applicable section 7 additional terms, to the whole of the work, and all its parts, regardless of how they are packaged. This License gives no permission to license the work in any other way, but it does not invalidate such permission if you have separately received it.
d) If the work has interactive user interfaces, each must display Appropriate Legal Notices; however, if the Program has interactive interfaces that do not display Appropriate Legal Notices, your work need not make them do so.
A compilation of a covered work with other separate and independent works, which are not by their nature extensions of the covered work, and which are not combined with it such as to form a larger program, in or on a volume of a storage or distribution medium, is called an “aggregate” if the compilation and its resulting copyright are not used to limit the access or legal rights of the compilation's users beyond what the individual works permit. Inclusion of a covered work in an aggregate does not cause this License to apply to the other parts of the aggregate.

6. Conveying Non-Source Forms.
You may convey a covered work in object code form under the terms of sections 4 and 5, provided that you also convey the machine-readable Corresponding Source under the terms of this License, in one of these ways:

a) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by the Corresponding Source fixed on a durable physical medium customarily used for software interchange.
b) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by a written offer, valid for at least three years and valid for as long as you offer spare parts or customer support for that prod-

uct model, to give anyone who possesses the object code either (1) a copy of the Corresponding Source for all the software in the product that is covered by this License, on a durable physical medium customarily used for software interchange, for a price no more than your reasonable cost of physically performing this conveying of source, or (2) access to copy the Corresponding Source from a network server at no charge.
c) Convey individual copies of the object code with a copy of the written offer to provide the Corresponding Source. This alternative is allowed only occasionally and noncommercially, and only if you received the object code with such an offer, in accord with subsection 6b.
d) Convey the object code by offering access from a designated place (gratis or for a charge), and offer equivalent access to the Corresponding Source in the same way through the same place at no further charge. You need not require recipients to copy the Corresponding Source along with the object code. If the place to copy the object code is a network server, the Corresponding Source may be on a different server (operated by you or a third party) that supports equivalent copying facilities, provided you maintain clear directions next to the object code saying where to find the Corresponding Source. Regardless of what server hosts the Corresponding Source, you remain obligated to ensure that it is available for as long as needed to satisfy these requirements.
e) Convey the object code using peer-to-peer transmission, provided you inform other peers where the object code and Corresponding Source of the work are being offered to the general public at no charge under subsection 6d.
A separable portion of the object code, whose source code is excluded from the Corresponding Source as a System Library, need not be included in conveying the object code work.

A “User Product” is either (1) a “consumer product”, which means any tangible personal property which is normally used for personal, family, or household purposes, or (2) anything designed or sold for incorporation into a dwelling. In determining whether a product is a consumer product, doubtful cases shall be resolved in favor of coverage. For a particular product received by a particular user, “normally used” refers to a typical or common use of that class of product, regardless of the status of the particular user or of the way in which the particular user actually uses, or expects or is expected to use, the product. A product is a consumer product regardless of whether the product has substantial commercial, industrial or non-consumer uses, unless such uses represent the only significant mode of use of the product.

“Installation Information” for a User Product means any methods, procedures, authorization keys, or other information required to install and execute modified versions of a covered work in that User Product from a modified version of its Corresponding Source. The information must suffice to ensure that the continued functioning of the modified object code is in no case prevented or interfered with solely because modification has been made.

If you convey an object code work under this section in, or with, or specifically for use in, a User Product, and the conveying occurs as part of a transaction in which the right of possession and use of the User Product is transferred to the recipient in perpetuity or for a fixed term (regardless of how the transaction is characterized), the Corresponding Source conveyed under this section must be accompanied by the Installation Information. But this requirement does not apply if neither you nor any third party retains the ability to install modified object code on the User Product (for example, the work has been installed in ROM).

The requirement to provide Installation Information does not include a requirement to continue to provide support service, warranty, or updates for a work that has been modified or installed by the recipient, or for the User Product in which it has been modified or installed. Access to a network may be denied when the modification itself materially and adversely affects the operation of the network or violates the rules and protocols for communication across the network.

Corresponding Source conveyed, and Installation Information provided, in accord with this section must be in a format that is publicly documented (and with an implementation available to the public in source code form), and must require no special password or key for unpacking, reading or copying.

7. Additional Terms.
“Additional permissions” are terms that supplement the terms of this License by making exceptions from one or more of its conditions. Additional permissions that are applicable to the entire Program shall be treated as though they were included in this License, to the extent that they are valid under applicable law. If additional permissions apply only to part of the Program, that part may be used separately under those permissions, but the entire Program remains governed by this License without regard to the additional permissions.

When you convey a copy of a covered work, you may at your option remove any additional permissions from that copy, or from any part of it. (Additional permissions may be written to require their own removal in certain cases when you modify the work.) You

may place additional permissions on material, added by you to a covered work, for which you have or can give appropriate copyright permission.

Notwithstanding any other provision of this License, for material you add to a covered work, you may (if authorized by the copyright holders of that material) supplement the terms of this License with terms:

a) Disclaiming warranty or limiting liability differently from the terms of sections 15 and 16 of this License; or
b) Requiring preservation of specified reasonable legal notices or author attributions in that material or in the Appropriate Legal Notices displayed by works containing it; or
c) Prohibiting misrepresentation of the origin of that material, or requiring that modified versions of such material be marked in reasonable ways as different from the original version; or
d) Limiting the use for publicity purposes of names of licensors or authors of the material; or
e) Declining to grant rights under trademark law for use of some trade names, trademarks, or service marks; or
f) Requiring indemnification of licensors and authors of that material by anyone who conveys the material (or modified versions of it) with contractual assumptions of liability to the recipient, for any liability that these contractual assumptions directly impose on those licensors and authors.
All other non-permissive additional terms are considered "further restrictions" within the meaning of section 10. If the Program as you received it, or any part of it, contains a notice stating that it is governed by this License along with a term that is a further restriction, you may remove that term. If a license document contains a further restriction but permits relicensing or conveying under this License, you may add to a covered work material governed by the terms of that license document, provided that the further restriction does not survive such relicensing or conveying.

If you add terms to a covered work in accord with this section, you must place, in the relevant source files, a statement of the additional terms that apply to those files, or a notice indicating where to find the applicable terms.

Additional terms, permissive or non-permissive, may be stated in the form of a separately written license, or stated as exceptions; the above requirements apply either way.

8. Termination.
You may not propagate or modify a covered work except as expressly provided under this License. Any attempt otherwise to propagate or modify it is void, and will automatically terminate your rights under this License (including any patent licenses granted under the third paragraph of section 11).

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, you do not qualify to receive new licenses for the same material under section 10.

9. Acceptance Not Required for Having Copies.
You are not required to accept this License in order to receive or run a copy of the Program. Ancillary propagation of a covered work occurring solely as a consequence of using peer-to-peer transmission to receive a copy likewise does not require acceptance. However, nothing other than this License grants you permission to propagate or modify any covered work. These actions infringe copyright if you do not accept this License. Therefore, by modifying or propagating a covered work, you indicate your acceptance of this License to do so.

10. Automatic Licensing of Downstream Recipients.
Each time you convey a covered work, the recipient automatically receives a license from the original licensors, to run, modify and propagate that work, subject to this License. You are not responsible for enforcing compliance by third parties with this License.

An “entity transaction” is a transaction transferring control of an organization, or substantially all assets of one, or subdividing an organization, or merging organizations. If propagation of a covered work results from an entity transaction, each party to that transaction who receives a copy of the work also receives whatever licenses to the work the party's predecessor in interest had or could give under the previous paragraph, plus a right to possession of the Corresponding Source of the work from the predecessor in interest, if the predecessor has it or can get it with reasonable efforts.

You may not impose any further restrictions on the exercise of the rights granted or affirmed under this License. For example, you may not impose a license fee, royalty, or other charge for exercise of rights granted under this License, and you may not initiate litigation (including a cross-claim or counterclaim in a lawsuit) alleging that any patent claim is infringed by making, using, selling, offering for sale, or importing the Program or any portion of it.

11. Patents.
A “contributor” is a copyright holder who authorizes use under this License of the Program or a work on which the Program is based. The work thus licensed is called the contributor's “contributor version”.

A contributor's “essential patent claims” are all patent claims owned or controlled by the contributor, whether already acquired or hereafter acquired, that would be infringed by some manner, permitted by this License, of making, using, or selling its contributor version, but do not include claims that would be infringed only as a consequence of further modification of the contributor version. For purposes of this definition, “control” includes the right to grant patent sublicenses in a manner consistent with the requirements of this License.

Each contributor grants you a non-exclusive, worldwide, royalty-free patent license under the contributor's essential patent claims, to make, use, sell, offer for sale, import and otherwise run, modify and propagate the contents of its contributor version.

In the following three paragraphs, a “patent license” is any express agreement or commitment, however denominated, not to enforce a patent (such as an express permission to practice a patent or covenant not to sue for patent infringement). To “grant” such a patent license to a party means to make such an agreement or commitment not to enforce a patent against the party.

If you convey a covered work, knowingly relying on a patent license, and the Corresponding Source of the work is not available for anyone to copy, free of charge and under the terms of this License, through a publicly available network server or other readily accessible means, then you must either (1) cause the Corresponding Source to be so available, or (2) arrange to deprive yourself of the benefit of the patent license for this particular work, or (3) arrange, in a manner consistent with the requirements of this License, to extend the patent license to downstream recipients. “Knowingly relying” means you have actual knowledge that, but for the patent license, your conveying the covered work in a country, or your recipient's use of the covered work in a country, would infringe one or more identifiable patents in that country that you have reason to believe are valid.

If, pursuant to or in connection with a single transaction or arrangement, you convey, or propagate by procuring conveyance of, a covered work, and grant a patent license to some of the parties receiving the covered work authorizing them to use, propagate, modify or convey a specific copy of the covered work, then the patent license you grant is automatically extended to all recipients of the covered work and works based on it.

A patent license is “discriminatory” if it does not include within the scope of its coverage, prohibits the exercise of, or is conditioned on the non-exercise of one or more of the rights that are specifically granted under this License. You may not convey a covered work if you are a party to an arrangement with a third party that is in the business of distributing software, under which you make payment to the third party based on the extent of your activity of conveying the work, and under which the third party grants, to any of the parties who would receive the covered work from you, a discriminatory patent license (a) in connection with copies of the covered work conveyed by you (or copies made from those copies), or (b) primarily for and in connection with specific products or compilations that contain the covered work, unless you entered into that arrangement, or that patent license was granted, prior to 28 March 2007.

Nothing in this License shall be construed as excluding or limiting any implied license or other defenses to infringement that may otherwise be available to you under applicable patent law.

12. No Surrender of Others' Freedom.
If conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot convey a covered work so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not convey it at

all. For example, if you agree to terms that obligate you to collect a royalty for further conveying from those to whom you convey the Program, the only way you could satisfy both those terms and this License would be to refrain entirely from conveying the Program.

13. Use with the GNU Affero General Public License.
Notwithstanding any other provision of this License, you have permission to link or combine any covered work with a work licensed under version 3 of the GNU Affero General Public License into a single combined work, and to convey the resulting work. The terms of this License will continue to apply to the part which is the covered work, but the special requirements of the GNU Affero General Public License, section 13, concerning interaction through a network will apply to the combination as such.

14. Revised Versions of this License.
The Free Software Foundation may publish revised and/or new versions of the GNU General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies that a certain numbered version of the GNU General Public License "or any later version" applies to it, you have the option of following the terms and conditions either of that numbered version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of the GNU General Public License, you may choose any version ever published by the Free Software Foundation.

If the Program specifies that a proxy can decide which future versions of the GNU General Public License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Program.

Later license versions may give you additional or different permissions. However, no additional obligations are imposed on any author or copyright holder as a result of your choosing to follow a later version.

15. Disclaimer of Warranty.
THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. Limitation of Liability.
IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

17. Interpretation of Sections 15 and 16.
If the disclaimer of warranty and limitation of liability provided above cannot be given local legal effect according to their terms, reviewing courts shall apply local law that most closely approximates an absolute waiver of all civil liability in connection with the Program, unless a warranty or assumption of liability accompanies a copy of the Program in return for a fee.

END OF TERMS AND CONDITIONS

GNU LESSER GENERAL PUBLIC LICENSE

Version 3, 29 June 2007

Copyright © 2007 Free Software Foundation, Inc. <http://fsf.org/>

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

This version of the GNU Lesser General Public License incorporates the terms and conditions of version 3 of the GNU General Public License, supplemented by the additional permissions listed below.

0. Additional Definitions.
As used herein, “this License” refers to version 3 of the GNU Lesser General Public License, and the “GNU GPL” refers to version 3 of the GNU General Public License.

“The Library” refers to a covered work governed by this License, other than an Application or a Combined Work as defined below.

An “Application” is any work that makes use of an interface provided by the Library, but which is not otherwise based on the Library. Defining a subclass of a class defined by the Library is deemed a mode of using an interface provided by the Library.

A “Combined Work” is a work produced by combining or linking an Application with the Library. The particular version of the Library with which the Combined Work was made is also called the “Linked Version”.

The “Minimal Corresponding Source” for a Combined Work means the Corresponding Source for the Combined Work, excluding any source code for portions of the Combined Work that, considered in isolation, are based on the Application, and not on the Linked Version.

The “Corresponding Application Code” for a Combined Work means the object code and/or source code for the Application, including any data and utility programs needed for reproducing the Combined Work from the Application, but excluding the System Libraries of the Combined Work.

1. Exception to Section 3 of the GNU GPL.
You may convey a covered work under sections 3 and 4 of this License without being bound by section 3 of the GNU GPL.

2. Conveying Modified Versions.
If you modify a copy of the Library, and, in your modifications, a facility refers to a function or data to be supplied by an Application that uses the facility (other than as an argument passed when the facility is invoked), then you may convey a copy of the modified version:

a) under this License, provided that you make a good faith effort to ensure that, in the event an Application does not supply the function or data, the facility still operates, and performs whatever part of its purpose remains meaningful, or
b) under the GNU GPL, with none of the additional permissions of this License applicable to that copy.
3. Object Code Incorporating Material from Library Header Files.
The object code form of an Application may incorporate material from a header file that is part of the Library. You may convey such object code under terms of your choice, provided that, if the incorporated material is not limited to numerical parameters, data structure layouts and accessors, or small macros, inline functions and templates (ten or fewer lines in length), you do both of the following:

a) Give prominent notice with each copy of the object code that the Library is used in it and that the Library and its use are covered by this License.
b) Accompany the object code with a copy of the GNU GPL and this license document.
4. Combined Works.
You may convey a Combined Work under terms of your choice that, taken together, effectively do not restrict modification of the portions of the Library contained in the Combined Work and reverse engineering for debugging such modifications, if you also do each of the following:

a) Give prominent notice with each copy of the Combined Work that the Library is used in it and that the Library and its use are covered by this License.
b) Accompany the Combined Work with a copy of the GNU GPL and this license document.
c) For a Combined Work that displays copyright notices during execution, include the copyright notice for the Library among these notices, as well as a reference directing the user to the copies of the GNU GPL and this license document.
d) Do one of the following:
0) Convey the Minimal Corresponding Source under the terms of this License, and the Corresponding Application Code in a form suitable for, and under terms that permit, the user to recombine or relink the Application with a modified version of the Linked Version to produce a modified Combined Work, in the manner specified by section 6 of the GNU GPL for conveying Corresponding Source.
1) Use a suitable shared library mechanism for linking with the Library. A suitable mechanism is one that (a) uses at run time a copy of the Library already present on the user's computer system, and (b) will operate properly with a modified version of the Library that is interface-compatible with the Linked Version.

e) Provide Installation Information, but only if you would otherwise be required to provide such information under section 6 of the GNU GPL, and only to the extent that such information is necessary to install and execute a modified version of the Combined Work produced by recombining or relinking the Application with a modified version of the Linked Version. (If you use option 4d0, the Installation Information must accompany the Minimal Corresponding Source and Corresponding Application Code. If you use option 4d1, you must provide the Installation Information in the manner specified by section 6 of the GNU GPL for conveying Corresponding Source.)
5. Combined Libraries.
You may place library facilities that are a work based on the Library side by side in a single library together with other library facilities that are not Applications and are not covered by this License, and convey such a combined library under terms of your choice, if you do both of the following:

a) Accompany the combined library with a copy of the same work based on the Library, uncombined with any other library facilities, conveyed under the terms of this License.
b) Give prominent notice with the combined library that part of it is a work based on the Library, and explaining where to find the accompanying uncombined form of the same work.
6. Revised Versions of the GNU Lesser General Public License.
The Free Software Foundation may publish revised and/or new versions of the GNU Lesser General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Library as you received it specifies that a certain numbered version of the GNU Lesser General Public License “or any later version” applies to it, you have the option of following the terms and conditions either of that published version or of any later version published by the Free Software Foundation. If the Library as you received it does not specify a version number of the GNU Lesser General Public License, you may choose any version of the GNU Lesser General Public License ever published by the Free Software Foundation.

If the Library as you received it specifies that a proxy can decide whether future versions of the GNU Lesser General Public License shall apply, that proxy's public statement of acceptance of any version is permanent authorization for you to choose that version for the Library.