SMEDGE

What's New in Smedge

Smedge 2011 Update 1

Table of Contents

Update 1

New Modo Product

There is a new compiled Module to control the Modo renderer. The Module uses the modo_cl and sends command via input pipe currently. It can detect errors from the renderer but not filenames.. The Module will try to find the latest executable installed on your system at the first startup, and it can be customized with multiple versions using the standard Dynamic Product control via a Modo.ini file, if you wish.

Maya Single Frame rendering

The Maya module now has full-featured single frame rendering product support. The system is designed to be able to divide up a sub-region of a large image into slices, horizontal, vertical or both, with a specified amount of overlap. The image slices are named to make it understandable how to reassemble them, with a customizable naming convention. It does not include a re-assembly step by default, but there is a customizable parameter to allow you to include this if you have a system available in your pipeline.

By default, the Maya.ini file adds single frame products for the “Maya Software Renderer”, “mental ray for Maya”, “VRay for Maya”, “RenderMan for Maya”, and “3Delight for Maya”. You can add custom single frame type Maya products by including the key “SingleFrame = yes” in the product definition section of the Maya.ini file. There are several new parameters for the single frame type products to access the various attributes of single frame type jobs, which can be seen in the Administrator Manual.

New 3Delight for Maya Product

The default Maya.ini file now includes support for the 3Delight plugin renderer for Maya with new Product definitions.

Improved Maya submit scripts

The smedgeRender.mel script has been completely revamped. Now, just source the script, and it self-installs itself into the render menu. There is a new interface with full customization, but the script can also be called directly without the interface as well. There is a lot of control in the script, including the ability to download the pools.

There is also a new op_SmedgeRender.mel script that provides a similar level of control to integrate submitting to Smedge with the openPipline MEL script system. To install, just copy the script to the addons folder of your openPipeline installation.

Fixed support for detecting frames rendered with mental ray for Maya 2012

mental ray for Maya changed the way that image filenames are output in the process output text. The change was subtle,but it was enough to break the way Smedge was looking for the image filenames. The default Maya.ini file has been modified to correctly detect the filenames in both Maya 2012 with the new format, and the older format in all previous versions.

Also improved on the image detection definitions to ensure that Maya products can detect any specified image filename. Before, it was possible to set up an arrangement of image detection names that could cause the image detection to fail. This was most noticeable with “Maya” product renders that were rendering a file with the Vray for Maya plugin renderer set in the render globals, which were not properly detecting the rendered image file names.

3D Studio Max Module now properly expands all parameters in the command line

It was not recursively expaning the parametes: $(Extra) $(TimeLimits) $(WorkPath) or $(PathFile), but now it does.

Virtual Module products were not correcty returning $(ImageNotEndText)

The compiled module now returns the correct value for the parameter.

The GUI will now report if license installation was successful or failed

The master will report back to the client that installs a license. The GUI will use the report to notify the user if the installation was successful or failed. Failure includes a message that can help determine why the license failed to install.

New Master option to disable Pool prioritization

In the Master Options dialog box, on the Distribution tab, there is now a new option to disable Job prioritization on an Engine based on the order of the Pools. Pool prioritization is enabled by default, but if disabled, the order of the pools in the list of pools an Engine belongs to will be ignored. When disabled, only the Jobs' priority values are used to prioritize jobs (along with the creation time or the number of currently going work units, for FIFO and Round-Robin distribution, respectively). This can make use of pools simpler, as the pools are only used to group machines that will work on the job.

Submit Shell was ignoring -FinishDisposition

Now you can correctly set the finish disposition of a job with using the Submit command line Shell in script mode and specifying the parameters on the command line. Before it was ignoring this parameter and would always submit jobs with the default FinishDisposition. Submit will also now hide any parameters marked with the NoInputDisplay flag when running in interactive mode.

Engine Shell can run as a service/daemon on all platforms

You can now install the Engine command line shell component as a service on all platforms. Running as a service allows the use of the Engine shell to run without waiting for the result, which allows it to run much faster. This is especially useful in log-on or log-off script usage to enable or disable the engine without holding up the process.

The service/daemon installation scripts in the Utilities folder of each platform have been updated to support installation, control and removal of the service, called the "Engine Control" service. There are also new commands in the SmedgeGui Components sub-menu of the System menu to install and remove the service.

The Engine List command now lists all of the parameters from each Engine.

New Engine Shell commands: DownloadFile, FollowFile, DownloadLog, FollowLog

These four new commands are available in the Engine command line shell to download files from Engines using the File Server system. For the download file commands, the file to download can be supplied either as a path or as the unique ID assigned by the engine to the file. The file is downloaded and printed to stdout..

`DownloadFile` *file*	Downloads the given file by ID or path.
`FollowFile` *file*	Downloads the file and stays running waiting for more data to be written to the file.
`DownloadLog`	Downloads the engine's log file
`FollowLog`	Downloads the log file and stays running for more

All File and Dir type sub-parameters in a ParametersPI type parameter are translated

Before, because the paths were embedded inside of a single parameter, the translation system couldn't find them, and they were not translated between platforms. This could make it difficult to use some overrides across platforms. Now, the path translation system breaks down the parameters and translates any File or Dir type sub-parameters as it does with normal parameters.

New ProcessJob Parameters $(ElapsedRealTime) and $(ElapsedProcessTime)

These read-only parameters are available after the work has finished (for example in the work “post execute” and “finish” events).

ElapsedRealTime	This is replaced with the total number of milliseconds of real time that elapsed between when the process started and when it finished
ElapsedProcessTime	This is replaced with the total number of milliseconds of processor usage that the process accumulated, as reported by the operating system.

Changes to a Job now are forwarded to active work from that Job

Before, if you changed a parameter of a job with active work, the parameter value was not also updated in any work units that were running before the change was made. Now, changes to job parameters are also forwarded to the engines with any active work from that job, and the work units have the parameters updated as well, if possible. This is useful, for example, for adding error ignore strings or changing a timeout value without losing time for the currently running work that still had the old values.

There is a new ParameterInfo “NoParentUpdate” flag to disable a parameter from being updated in an active work unit when the job is updated. For example, the packet size of the work unit is not updated, because once the work process has been started, it will finish the original requested packet size. You can add the “NoParentUpdate” flag to custom parameter definitions in Virtual Modules and Custom Products.

wxBarHistoryCtrl tooltip now includes average and peak displays

The graph control use for showing the memory and CPU usage on the nodes has a tooltip if you hover the mouse over the control. Besides the current vale and the display limits, the tooltip now includes the average and the peak values (along with the last peak time). It also now allows custom text before and after the value text, which Conspectus uses to show more info about the available CPUs and memory on each machine. Also, the graphs in the Engine Status info pane no longer start with random numbers, which looked weird and screwed up the average and peak.

New command line flag -LogMessages

Understood by all Smedge component processes, if supplied on the command line, this flag will indicate to the communication system to create log files for all messages that are sent and received. This is for debugging purposes, and will slow down Smedge operation if it is specified. If you specify the flag to a component that can start other components, the presence of the flag will be passed along to those components as well. The clients will create two files in the Log folder, one for messages sent and one for messages received. The Master will create a sub folder for each client with a sent and received log file for each.

Components started with a custom MasterPort will listen for shutdown with a custom name

For exmple, you can now start a second instance of the GUI with a different master port, and you can control the engine with that alternate port without affecting any component processes that are running with the default port, so you won't accidentally interrupt work on the main network as you start and stop a test network on an alternate port.

Updating a Job would reset its history and cause work to be repeated

This bug was introduced in Smedge 2011 by changes made to improve stability. The history data was not being copied correctly, and would work while in memory, but any change to the job would reset it, and the job would start over.

Engines did not always send all of their current options on connection to the Master

This could cause the display of the current values of Engine settings to be incorrect in the GUI. Now the options are correctly sent always.

Command line arguments were not always parsed correctly on Linux

The wrapper scripts could fail to correctly parse the command line, causing the command to fail with certain input.

Info stream can go out of range if a machine's virtual memory size changes

Now the info stream will always update the ranges as well as the current values to ensure that the memory display is correct even if the machine changes the amount of virtual memory available while the SmedgeEngine process is running.

The AlwaysDefault parameter flag was being ignored

Now it is not ignored.

Memory Based Engine Loading

Smedge now allows you to distribute work to Engines based on the amount of memory that the work uses instead of, or in addition to, the number of CPU cores it uses. Memory distribution works by dividing up the amount of available physical memory by the requested memory size specified in the Job. For example, if a machine has 16 GB of RAM installed, and you set the memory request to 4 GB, then Smedge will allow up to 4 workers from this job on this machine.

By default, memory distribution is disabled, making Smedge work the way it always has. To enable memory distribution, you must configure your Master to allow it. Choose System > System Commands > Configure Master and set the **Memory Distribution** setting to either "Memory only" or "Both Processor and Memory". The former will use only memory distribution, and will ignore the Processes setting for the Job. Using both allows you to specify both a requested CPU count and a requested memory utilization

You can also set Smedge to actually place a hard limit on the amount of memory that a worker process is allowed to use. You can set this either Job by Job in the Advanced Info tab of the Submit Job window, or you can set it as the default behavior for a product in the Engine Product Options. By default, this hard limit is not enabled, which means that processes may exceed the requested amount of RAM you set for a job. This can impact performance if a worker uses more memory than expected, which can cause a machine to either start using swap memory, which is incredibly slow, or to have memory failures. On the other hand, if the limit is enabled and a worker hits the limit, this may also lead to memory failures for the work.

Engine event commands

You can now configure global event command handlers on an engine, for every job. These event commands are configured in the Configure Engine window, on the new Event Commands tab. These commands work similar to the Work related event commands attached to specific Jobs. However, these commands are triggered for all jobs that the Engine works on.

As with the Job event commands (and unlike Herald event notifications), the commands can be synchronous to the work operation, and can affect the result of the work if they report a failure. The Engine will also run the command through the variable substitution system, using the data from the work that triggers the event.

Engine event commands are also hooked into the default Engine system, so you can set up default event handlers for your system that can be customized for each Engine. The Event Commands tab is hidden for non-administrator users if the new "Event Commands" restriction is in place (see below).

New Restrictions

Smedge has added several new system restrictions:

Core Process Control	If set, the options for controlling the Master and Engine components are not available in the System menu for non-administrators. The SmedgeGui options are also hidden. Note that users may still be able to control those components through the operating system depending on the permissions configured on your system.
Event Commands	If set, the “Event Commands” tab of the Submit Job window and the Configure Engine window will be hidden for non-administrators
Job Advanced Info	If set, the “Advanced Info” tab of the Submit Job window will be hidden for non-administrators
Job Custom Pool	If set, the “Custom Pool” tab of the Submit Job window will be hidden for non-administrators
Job Pool	If set, the Pool will not be able to be changed for non-administrators by the Submit Job window or from the Job menu or the Job list context menu.
Save Default Engine	If set, the button to save the “default” engine is hidden for non-administrators.
Save Default Job	If set, the button to save the “default” job is hidden for non-administrators.

By default, the system starts with these restrictions in place automatically: **Core Process Control**, **Job Custom Pool**, **Job Event Commands**, **Save Default Engine**, **Save Default Job**. You can configure restrictions on the “Restrictions” tab of the Configure Master window.

Pixar RenderMan support

Support for rendering directly with prman is now available by way of a new Virtual Module. The Virtual Module is slightly more limited in some ways than a compiled module would be, but it is also easier to customize to a specific pipeline for end users.

The current setup makes some assumptions about the structure of your RIB files, based on the structure created by the RIB export MEL command included in RenderMan Pro Studio 3. The details of the folder structure can be modified while the system is online, to quickly adjust the command for any specific folder structure without having to restart the system, even while jobs are queued and other work is running.

Centralized Component Control

Smedge now allows you to configure which components will start with SmedgeGui by default. When the GUI connects to the Master, one of the first messages it receives specifies which components to start simultaneously. Any components requested to be started that are not yet started will be started at this time, and any components that were previously started and that have been requested not to be started will be automatically shut down. You can use this to help enforce a single Master system on your network, even for machines that have a fresh installation and no options set yet, or to specify that Aegis always starts and stops with SmedgeGui, to give users an easy way to control their own machine.

If you configure some components to start (or not start) manually in the SmedgeGui options, then this configuration will override the system default configuration. Additionally, Smedge will remember the last known Master default startup configuration, and will use that the next time you start SmedgeGui. Note that if you configure the default not to start the Master using SmedgeGui on the Master machine itself, it will ignore your request, and leave the Master running without changing any options about starting the Master.

Integration with Growl

Smedge on Mac includes a simple AppleScript that you can use to integrate Smedge with Growl to give very Mac friendly notifications. The script is included in the Utilities folder inside of the Smedge application bundle. To access it you can either use the menu command **System** > **Smedge Files** > **Browse the Utilities Folder**, or you can use the Finder to browse for it using the “View Package Contents” command. The command line syntax is:

```
osascript /Applications/Smedge.app/Contents/MacOS/Utilities/SmedgeGrowl.applescript [Title] Message
```

You can adjust the path to the script if you installed Smedge in a different location. The two parameters are an optional title string and a message string to display. You should surround the parameters with quote marks if you want to put a space into them. If you do not supply a title string, it will use the default title “Smedge Notification”. Of course, for it to work correctly, you must have Growl installed on your system.

Interface

New work re-queuing operation: Stop and Divide

You can use this command to perform two operations at one time. First, it changes the Job's Packet Size to 1, then it stops any selected work units and requeues them. This is useful when you have a large packet size job finishing up on a few machines, and no other work in the queue, so machines are sitting idle. Select work units that you want to divide up, then select this command, and the work will be divided up and re-dispatched, making better use of your resources.

Search in the Output Window

You can now search for text in the output window, which is used to display process output and log files. You can search up or down from the current position, case sensitive or not. The window will scroll to display a found item, and will highlight it for easy identification. You can also now save the contents of an output window to a text file at any time. To access the search function, press Control + F, and to save the data press Control + S, or you can access both commands from the context menu.

Any windows that open off-screen will be moved to be visible

If your arrangement of monitors has changed since you last ran Smedge on a machine, the windows will now figure out that they are positioning themselves offscreen and will move back to a visible monitor. If you ever find that a window is showing up offscreen, delete the user preferences file for the component when it is not running.

View Engine History.log is available in main Engine menu

The command was previously only available in the Engine List context menu. Now you can access it in the Engine menu of the main window as well. This command allows you to view the complete Smedge log file from the selected Engines. Note that the Engine must be online to access its history file. The files are not kept on the machine that views them, but are instead downloaded from the Engine on demand.

Allow VNC to offline Engines

You can now use the VNC commands to attempt to access a remote machine, even if the machine is offline. Because the offline status only indicates any Smedge component processes running on a machine, it's entirely possible that the actual machine could be up and running, with a remote access system enabled. So, as long as the engine is known of, you can access the VNC commands. Remember that, even though it is named VNC, you can use any remote access system by setting the VNC command line in the SmedgeGui Options dialog box.

Herald Notifications can be individually enabled or disabled

Besides the global enable/disable switch for Herald, you can now individually enable notifications. At the top of the Notification Information window is a check box to enable or disable that notification. If a notification is disabled, it will not be triggered, even if the notifications are enabled globally. If notifications are disabled globally, no notification will be triggered, even if the notification has been individually enabled. Disabled notifications are listed in red in the Herald window.

Conspectus can now enable and disable Engines

Right click on the conspectus to bring up the new menu system. You can enable or disable the engine with or without immediately killing work directly from Conspectus. Also in this menu you can set the update interval for getting the status from your network. You can select an update interval from intervals of 1, 2, 5, 10, 15, or 30 seconds or 1, 2, 5, 10, 15, 30 or 60 minutes.

You may also notice that hovering the mouse over the name of an Engine shows you some information about the machine, and you can see the exact CPU and memory count by hovering the mouse over the graphs. The range for the CPU is from 0 to 100% of the total processor usage on the machine, counting all cores reported by the operating system. The range for the memory graph is from 0 to the total virtual memory on the machine, as counted by Smedge (virtual memory = physical memory + swap). The graph will be green if the used memory is less than the physical memory on the machine.

CheckFileSequence can now configure the file scan parameters

CheckFieSequence has two new commands in the Job menu: Edit Custom Range allows you to set a different frame range to scan than the range specified in the Range parameter of the Job, using the new parameters CustomRange and UserRange (see What's New in the API). This menu command will actually change the CustomRange for the Job to the range you specify. CFS now uses the UserRange to find the range of files. If you have set a CustomRange, that value will be used. Otherwise, the original Range will be used.

The second new command is Edit Image Formats. This opens a window that lists and allows you to manipulate the list of image formats for the Job. You can add, change, or remove formats using this window, to force CFS to search for a specific set of files. This will also directly change the ImageFormat parameter of the job. Note that this parameter may be changed automatically by Smedge if it detects a new image sequence, even after you may have edited the formats yourself here.

ConfigureMaster now list and manage path translations

You can now list, add and remove path translations from a command line using ConfigureMaster. The new commands available are:

`-GetPathTranslations`	Lists the current path translations. If you combine this with either -PathTranslation or -RemovePathTrhanslation, this command will list the translations as they are before the requested operations.
`-PathTranslation` *Windows-root Linux-Root Mac-root*	Adds a path translation. You must provide all three roots for a translation. If you do not need one of the roots, pass an empty string with double quote marks for the platform you do not need. You can add multiple translations at the same time by adding this command multiple times.
`-RemovePathTranslation` *Root*	Removes a path translation. You can supply any one of the roots to delete the translation. It does not have to be the local platform root. If the root is found in more than one translation, all of them will be removed. You can remove multiple translations by adding this command multiple times.

PoolManager can now create, rename and delete Pools

You can now create and manage the overall pool set from a command line using the PoolManager. The new commands available are:

`PoolManager Create` *Name* [*options*]	Creates a pool with the given name. Will return the ID
`PoolManager Rename` *ID/Name New-Name* [*options*]	Renames the given pool. The original name can be supplied as either a name or an ID
`PoolManager Delete` *ID/Name*	Deletes a pool. The pool can be supplied by ID or by name.

Pool members are managed using the Engine command line shell to add, remove or reorder the pools, because each engine prioritizes the pools separately.

VNC to the Engine from the Job History list context menu

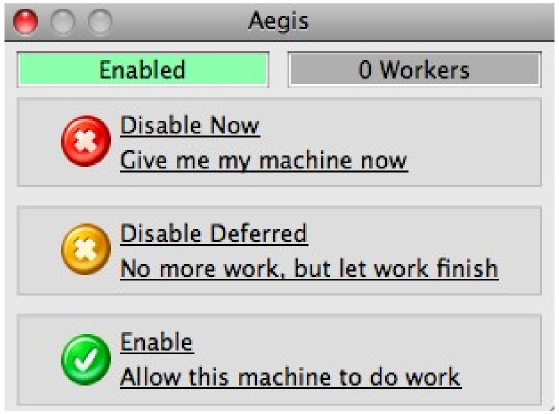
Using the context menu in the History list, you can now VNC to the engine listed for a work unit, or for an individual work history element. This command uses the same path to the viewer that the Engine menu VNC commands use, and is available whether or not the Engine is online.

You can see failures on each machine in the Engine list

The list of Engines on the main interface now shows the count of failures that an Engine has had. This makes it easier to see why an Engine may not be taking on work, or if an Engine has configuration errors that cause it to fail where others succeed. There is also a new Engine InfoPane that lists more details about the errors, including the type of job and the name of the job, along with a count of the errors that the Engine had on each specific Job. You can reset Job or Engine failure counts and easily see the entire Job History for a job that has failures.

Aegis shows the Engine status and work count

Along the top of the Aegis window are two status displays. The left shows the Engine status, color coded for easy recognition. The right shows the count of active work.



Modules

Maya Module disables Autodesk crash reporting by default

The REPORTERROR environment variable controls Autodesk crash reporting. By setting this variable to 0, it is possible to disable the crash reporting system. Because the crash reporting requires human interaction to complete, this can pile up crash reporters on render machines that are not being monitored. To avoid this, Smedge now sets this variable by default, unless it has already been set manually by you, or you enable crash reporting as a Product Option for an Engine.

AfterEffects Module catches missing layer dependencies as errors

The AfterEffects Module will now monitor the output stream for a message that starts with:

```
INFO:The following layer dependencies are missing
```

If such a line is found, Smedge immediately marks the work as failed and aborts it. You can remove this line from the list of error strings if you want to ignore these errors, or you can add a keyword to the Error Ignore Strings that allows you to ignore specific warning messages that you may not care about.

AfterEffects handles Unix path output

AfterEffects CS5 on Macs now outputs the path information using a Unix style path instead of the older Mac style path used on previous versions. Because Smedge can handle multiple versions of AE, it will now check if the path is already in Unix format. Any path in Unix format is left as is, and any path in the Mac format is run through the Mac path translation system (part of the operating system), to convert it to a Unix format for Smedge use. All Smedge components currently use only the Unix style paths on Macs.

mental ray standalone looks for fatal errors

The mental ray standalone Module now looks for error messages that begin with the keyword “fatal:” as well as “error:” in order to catch more errors from the renderer. You can always change the error detection settings yourself by Engine option or on a Job by Job basis. Also, be sure to check out the new cumulative nature of the error detection strings, mentioned in the What's New API chapter.

Nuke module now detects errors in the output

The Nuke module will look for error messages reported by the render process. Smedge will look for the keyword “ERROR:” anywhere in the output line, and all of the standard Smedge error detection processing will take place if an error is found. As with all other ProcessJob based products, you can adjust or disable the error detection system for Nuke jobs on a Job by Job basis, in the Advanced Info tab of the Submit Job window, or as a Product Option for the Engine.

Cinema 4D now detects missing file as a failure

Cinema 4D only indicates a file not found error with the output message:

```
Document not found
```

This message is the only indication from Cinema 4D that the scene file was not correctly found, so that message has been added to the Cinema4D virtual module default error detection strings so that this is now correctly detected by Smedge to report the error.

Maxwell failed renders did not requeue correctly

The Maxwell Module could cause erratic Engine behavior after rendering and merging failures. The primary symptom was that the Job would get hung up and would never complete. It could also cause an Engine to stop taking on other work, and could lead to Jobs that could not be easily removed, and it could cause the Job shell to report the wrong status during merge operations. The underlying issues involved a small bug in the communication of Maxwell job data that would lead to corrupted data, and some incorrect handling of failure conditions downloading an MXI before merging.

New Job Parameter Commands

The parameter command system has been integrated with the parameter information data, when accessing job data using the parameter substitution system with a proper Job object. When accessing the parameter substitution system for other data, for example, non-job data parameters or when using the job data table system instead, these commands will have no effect and will be silently ignored.

For all parameter types, the following commands are available:

Default	Gets the parameter's default value (overrides the current job value)
NiceName	Gets the parameter's display name (overrides the current job value)
Type	Gets the parameter's type as a string (overrides the current job value)

For ParameterInfo::Multi parameters (the parameters that have multiple fields in the submit job window), these commands are available:

Separator	Gets the separator string for this parameter (overrides the current job value)
Field-Name	Gets the sub-portion of the current job value that corresponds to the given field name

For ParameterInfo::Choice parameters (the parameters that have a pop-down menu), these commands are available:

Display	Gets the display string value for the current job value

For ParameterInfo::Parameters parameters (the parameters that have their own tab with sub-parameters), these commands are available:

Separator	Gets the separator string for this parameter (overrides the current job value)
InternalSeparator	Gets the internal separator string for this parameter (overrides the current job value)
Parameter-Name	Gets the sub-portion of the current job value that corresponds to the given sub-parameter. Any successive commands will be based on the parameter information for this sub-parameter

For example, you can get the Y pixel override value of a Maya job in a padded format using this: **`$(Extra.-x.y.pad)`**. The first command is '-x' to get the Resolution sub-parameter of the Extra parameter. The second command is 'y' to get the Y field from this sub-parameter, which is itself a multi parameter. The last command is Pad, which pads the value to four digits with leading zeros.

New RenderJob Parameter: InitImageFormat

The detected image format parameter is always initialized to an empty list by the Master when the job is created, even if it is supplied as part of the submit process. In order to allow manual creation of the image format string when you submit a job, there is a new RenderJob parameter InitImageFormat, which allows you to set the initial image format specifier.

InitImageFormat has the same syntax as the ImageFormat parameter, but it is only read by the system at job submission time. At that point, it's value is copied to the ImageFormat parameter. Any further changes made to InitImageFormat will not be reflected in the ImageFormat parameter.

See the Administrator Manual for more information about the RenderJob parameters.

New SequenceDistributor Parameters: CustomRange and UserRange

To allow users to customize the range of files that Smedge can detect from a job, there is now a new SequenceDistributor parameter CustomRange. This parameter allows you to specify a custom range for the job that is not used for distribution as part of the sequence. For example, CheckFileSequence uses this to allow you to search for a customized range of rendered image files. There is also a new read-only parameter UserRange that allows you to get the CustomRange, if it has been set, or to return the Range if the custom range had not been set.

New standardized component control system

The API now provides a single standard control system for starting component processes. It is accessed via the static members of the Components class. The header file is `smedge/Components.h`.

Error Start strings and Error Ignore strings are now inclusive

If you set error strings for detecting or ignoring errors in the output text in the Job advanced parameters, these strings will now be added to any strings set for the Engine's Product options. The old behavior was to that any strings you set for the job would completely override the strings set for the Engine. This could mean that errors that may have been detected or correctly ignored before may not be handled correctly. The new behavior makes it easy to just add an extra string to search for or ignore for a job without affecting the normal error detection system.

Added Thread::WaitForStartup

Added a standardized WaitForStartup method to the Thread class. This allows a thread to be notified when another thread has finished its startup and is about to start its main execution method. Each thread has its own Trigger object that provides the synchronization, created when the thread object is created. The trigger is signaled after the thread has finished its startup, and is left signaled, no matter how many other threads wait on it.

Application failure logs to stderr

If a component process fails to start, Executable logs information about the failure and where the Smedge startup log file is to help figure out what went wrong.

Other Optimizations

Memory performance improvements

The memory subsystem has been optimized to increase performance and reduce system load from the Smedge component processes.

CheckFileSequence now checks in the background

CFS file checking now runs as a background operation, keeping the interface running smoothly while still providing reasonable data refresh rates.

FileServer startup is significantly faster

The file server system does bookkeeping during component process startup that has been optimized to greatly reduce the startup time, especially on the Engine, which can potentially keep track of a lot of files from the process output from its work.

No longer confirms close when receiving a log-off event from the OS.

The GUI component processes now no longer ask for confirmation when receiving s sytem log-off event notification from the system. The rationale is that the system itself would have already confirmed the logoff or shut down with the user, so why bother asking about it again for Smedge?

Bug Fixes

Possible lock-up or crash points have been fixed

Several possible causes of process deadlock or crash have been found and fixed in the including a possible component crash or hang in messenger, a possible Master freeze accessing Engine information, a possible Master crash when an Engine connects, a possible GUI crash deleting work, and a crash if you tried to send an empty string to the Log system. Also fixed is an issue that could cause the Job data to get corrupted causing communication to the clients to be interrupted, and causing a possible unplanned Master switch.

Master and Engine not start on Linux without the Machine folder

The single instance locking mechanism tries to create the file lock in the machine folder, to make it lock for all users on the machine at the same time. However, if the machine folder was missing and could not be created (for example because the user does not have permission), the process would fail to start. Now, if the single instance lock file cannot be created in the machine folder, Smedge falls back to using the temp folder. This will at least make the system run a single instance for the user, and possibly also other users if there is a shared temp folder for the system.

Round-Robin Distribution is more even handed

The round-robin distribution mode has been fixed. Previously, it would actually distribute work in a least-recently-dispatched mode, which would bias the distribution to slower jobs.

Default Engine could get set to have no enabled Products

Something would get set incorrectly in the default Engine file, and machines that had the default applied, either manually through using the default system in the Configure Engine window, or automatically when they connected, could get all products unset, resulting in no work being done by the machine. The chances of this happening were increased when the Master switched between machines, which could happen unexpectedly because of some of the crash and hangs (fixed above).

Now, when the Master loads the default file, if it is set to have no enabled products for any reason, the Master will instead use the hard coded defaults, because a default engine that is not able to do any work is sort of useless.

ProcesJob removes spaces in entire output log path now

When creating the output log files, any spaces in the full path to the file are now converted to an underscore. Previously, it would remove spaces in the output file itself, but not in the parent folder of that output file.

Job list sorting by Status was not consistent

Sorting the Job list by status is now consistent and correctly accounts for running work. Status order should be (high to low):

Finished (failed)
Finished (canceled)
Finished
Working (order by number of workers)
Working (paused) (order by number of workers)
Queued
Paused

List selection did not always update correctly

If new elements were added to the list between the last time you selected something and the attempt to select something new, the Info Pane did not get correctly updated to the new selection. Also, deselecting one of many selected items in a list caused the entire selection to be cleared.

Delete Finished Jobs command was always visible in the Job list menu

The command is only supposed to be available in Administrator Mode, but it was always getting displayed.

Machine CPU and Memory information stream could get hung up

The real-time processor and memory status indications could get hung up on the Engine that was being monitored. The system has been redesigned to use UDP communication to avoid possible networking/thread interactions with long term connections, and it is now able to automatically recover a connection if it detects a loss of communication. The display should be more reliable in both the GUI Engine Status info panel, and the Conspectus.

CheckFileSequence would not always startup correctly

There was a race condition in the startup sequence that could cause CFS to connect to the Master but never access the Job information. This would lead to the the program running, but not displaying anything, and no way to force it to try again. The race condition has been fixed so that it will start up correctly and start its processing when it receives the Job information from the Master.

CheckFileSequence could not be quit by the menu on Mac

The Quit menu handler was not working correctly, and the only way to close the program was to click the close button on the window.

Conspectus sorted incorrectly if several machines had the same name

If you happened to have two machines with the same name on a network it would cause Conspectus to sort the Engines incorrectly. This has been fixed, in case you are even in a situation like that for any reason.

Smedge got the wrong number of CPUs on 64 bit Windows

A bug in the 32 bit Windows libraries on x64 Windows would not report more than 32 cores in a machine. Smedge now checks for and works around this situation, getting the correct count of cores on these machines.

FileFinder fails to find a folder with a valid search string

A trailing directory delimiter on a filename causes the Windows file searching API to fail to find a path, even if that path exists. The FileFinder system in Smedge now removes any trailing directory delimiter to ensure that if you are searching for a folder, it can be found correctly.

An access violation from an errant MayaJob object could hang work

Through unusual conditions, it was possible for a Maya job object in the Master's data to get corrupted, and thereafter could cause other workers to fail to start correctly because of a problem the corrupted job triggered during the other job's startup processing. Smedge now greatly reduces the chances of the Maya job getting hooked into other job's startup processing, and should allow a clean recovery from a similar type of error should it occur again.