SMEDGE

What's New in Smedge

Smedge 2012

Table of Contents

INTRODUCTION 3

OPERATION 4

- Wake-on-LAN
- Remote machine control
- Variable packet size
- Automatic font synchronization
- New system to translate the contents of scene files
- Limit jobs per creator
- New maximum job failures limit
- Processor time and peak memory usage are now reported in the Job History
- New error detection test: Minimum memory usage at start-up
- Average work time calculations by the Master now normalize for the packet size
- You can now disable reporting of detected images
- Ignored error messages are reported in the Job History
- New option to automatically reset job failures when a job is deleted
- Extensive performance optimizations throughout

INTERFACE 8

- Customizable GUI window and improved filtering
- Controllable Display columns in VIewLists
- Improved GUI component process control for Master and Engine running normally and as services (and lock files)
- Individual dispatch reporting for specific Jobs and Engines
- Job shell list command can now filter the results and can report a count as well as listing the matched jobs
- Engine shell can set product options
- New Submit shell built-in Pool “My Computer”
- Submit shell can update individual parameters for a job
- Conspectus window can scroll if there are more engines than fit on the window
- You can now reset failure counts for offline engines

MODULES 11

- New Product: V-Ray Standalone
- New Product: imgcvt
- Maya Module correctly detects new image output formats in Maya 2012
- Maya submit script improvements
- After Effects integration script improved
- Improved error detection for Cinema 4D renders
- Cinema 4D submit script
- 3D Studio Max submit script
- Nuke submit script
- Modo Module shows additional statistics and fixes

PLATFORM SPECIFIC CHANGES 13

- RPM installer for Linux
- Linux and Mac have more appropriate file permissions
- Mac now has version info available
- Work around for bug in Mac runtime libraries that could cause hangs writing to the log files

API 14

- New SequenceDistributor parameter $(ActualPacketSize)
- New ProcessJob error controls
- New RenderJob Parameters
- Job class includes a scoped helper class to disable automatic parameter transformations
- New scoped Increment/Decrement template classes
- Updated to wxWidgets 2.8.12

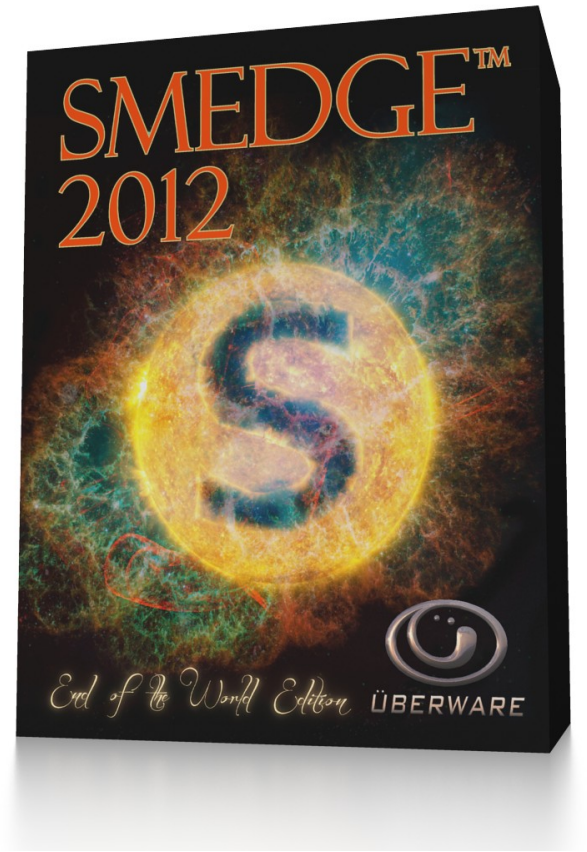
MAJOR BUG FIXES 16

- Fixed several possible crash and hang points
- Master was ignoring the -Database flag
- Master did not always honor requests to see the Dispatch.log file
- Mirror Masters ignored the request to shutdown correctly if the Mirror option was not set in the Master Options
- Mirror Masters did not always persistently mirror the original primary master
- Clients could refuse to connect to the Master
- Engine did not always update settings correctly when editing engine settings
- Editing a Job that is in progress could cause rendering of already completed frames
- GUI sometimes did not update the Info Panel when you selected an item
- GUI will not show percent done at 100% if the job is not yet finished
- Virtual Modules now correctly translate File and Dir type parameters

Introduction

Welcome to Smedge 2012!

The latest version of Smedge provides a whole new level of performance and reliability, and gives you more control over your rendering workflow than ever before.



Smedge 2012 does not communicate with prior versions of Smedge. We strongly recommend that you update your whole network at one time to avoid strange any problems with old versions left running.

Because the data structures have also changed to allow for the new features, like Wake-on-LAN and automatic scene file content translation, we also recommend that you back up any old data before upgrading. While the data can be upgraded automatically, if you ever want to go back to the old version of Smedge, the upgraded data will not be successfully read by the old versions.

As always, if you have any questions about new features or our future development plans, we encourage you to contact us.

Thanks, and see you in Smedge!

Wake-on-LAN

Smedge now includes a sophisticated system to allow you to reduce your farm's power consumption using the “Wake on LAN” technology built into most modern machines. After a period of idle time, Engines can now be set to automatically put themselves into their low power “sleep”mode. Then, when new jobs are submitted, the Master will wake up as many nodes as it needs to get the work complete. You can configure the timeout period for the machines to be put to sleep in the Engine Settings dialog.

The ability to wake machines is also available both from the GUI and using the Engine command line shell

Remote machine control

There are now several commands available in the Engine menu that allow you to shutdown or restart a machine, or put a machine into low power “sleep” mode. These commands all use a simple script system (batch files on Windows, and shell scripts on Linux/OS-X) that allow you to customize exactly how it works, and even configure special permissions that may not be available to the Engine itself. Additionally, there is a new command to force all Smedge component processes on a machine to shut down, making it possible to cleanly stop all Smedge on a remote machine (say if you want to perform an upgrade or configuration change) without having to go to the machine and find and kill those processes manually.

These commands are all available in the Engine menu, and are only available in “Administrator Mode.” The commands to sleep, restart or shutdown the machine do require that the Engine component is running and connected on the machine, however the commands to wake a machine up or to stop all Smedge component processes are available even if the engine is offline or disconnected (though they may have no effect if the configuration is incorrect or if the processes cannot communicate with the Master). These commands are also now available from the Engine command line shell.

Variable packet size

You can now set the Master to vary the packet size based on the Engine's “Priority” value. The system uses the priority of each engine as a percentage of the given packet size, allowing you to set up your machines to get more or less consistent render times even on widely varied hardware.

For example, if you set your Job packet size to 20 frames, and you have your fast machines set to priority 100 and a few slow machines set to priority 50, then the fast machines get the full 20 frame packet but the slow machines get 10 frame packets.

When this system is enabled, the Job History displays only the total and average packet times and does not attempt to determine “per frame” times (since each machine is rendering different numbers of frames).

Automatic font synchronization

Smedge can now automatically ensure that fonts from a specified folder are all correctly installed and available on a machine before starting work. This system is available for any product in Smedge, but is primarily useful for programs like After Effects that can make direct use of system fonts as part of their operation. The fonts are installed only if they are not available, and will be removed from the system after they are no longer needed, to ensure that your system does not get bogged down with an excessive number of fonts over time.

New system to translate the contents of scene files

There is a new Parameter Command called “TranslateContents”. If the value of the parameter is a valid path to a file, and if there are path translations set up on your Master, using this command will make a copy of the file in the Job local folder. Each line in the file will be searched for any translatable roots, and if found it will translate the path as it writes the file to the local folder. If the parameter doesn't correspond to a readable file, or if there are no translations configured, the command has no effect and the parameter value is untouched.

You can enable products to automatically use the TranslateContents command for the Scene parameter, using the new “TranslateSceneContents” parameter, set in Job's Advanced Info dialog under Render options. When enabled, any time you access the $(Scene) parameter, it will be translated locally (if it has not already been) and

Limit jobs per creator

You can now set up a limit to how many workers are allowed from jobs with a specific Creator. This is another option for ensuring that your farm resources can be shared, without having to set up special Pools for every possible user. These limits are set in the Configure Master dialog box.

New maximum job failures limit

Smedge can now limit how many times a single job is allowed to fail before no more attempts are made to send out work from the Job. Before, you could limit how many times specific Engines were allowed to fail on a Job (and on a type of Job), but the Job itself may get passed around to every machine in your network. This could cause jobs to accrue hundreds of errors before they finally failed on every possible engine, and could cause delays in getting to other work caused by the failing job. Now, you can set a maximum number of failures for the Job itself regardless of the Engine failures, to keep these failing jobs from wasting too much time. This option is set with the other failure limits in the Configure Master dialog box.

Processor time and peak memory usage are now reported in the Job History

Smedge previously monitored the processor usage and total elapsed real time of a Job, but did not make that information readily available without using the API. Now, that information is directly reported in the Job's history. Additionally, Smedge now also monitors the process' peak memory usage and reports that as well. These systems all work consistently on all platforms.

Note that it is possible on the Unix based platforms (Linux and OS-X) for 3rd party applications to defeat this system. For example, After Effects essentially launches its own render daemon, and completely disconnects itself from the traditional Unix based process hierarchy. There is no way for Smedge to use only operating system resources to monitor these kinds of processes, so you may see incorrect statistics reported for AE renders on Macs until Adobe changes how their command line renderer works or provides some kind of public interface to find the spawned “daemon” process.

New error detection test: Minimum memory usage at start-up

Related to the memory monitoring system, Smedge can now ensure that a render process exceeds a certain amount of memory usage within a certain period of time after the process starts. This is another way to catch processes that can occasionally hit infinite loops during their initialization phase. For example, 3D Studio Max can occasionally hang when starting up, using CPU time but never actually starting the render. This system can now be used to detect these conditions, triggering an error and forcing the work to be restarted.

Average work time calculations by the Master now normalize for the packet size

When the Master is monitoring for packets that are out of the range of the average packet time, it will now normalize the time based on the packet size, so that 1 frame packet at the end does not get incorrectly detected as an error because it is so much slower than the other packets that are rendering 10 frames at a time.

You can now disable reporting of detected images

If you are not using the Smedge image detection system for anything in particular, you can now disable the reporting of the detected image files from the Engine back to the Master. The Engine will still do its normal error processing on the image files it detects, but the filenames will not appear in the job history. This can speed up the master and the GUI processing, especially if a render generates a lot of different images, but some Smedge features that use the image file names will no longer be available. You can set this option in the Configure Master dialog box.

Ignored error messages are reported in the Job History

When Smedge finds an error string that is set to be ignored, it now reports that message in the job history. This is useful so you can see if there are errors in the scene that may be being ignored but that should not. Also, if an error message is ignored, you can now optionally set the engine to

also ignore the process exit code. This is useful for renders that may produce benign errors that are correctly ignored, but that cause the rendering process to return an “unsuccessful” exit code anyway. If this option is enabled (in the Advanced Job Info) then the process exit code is only ignored if an error string is detected but ignored. If no error messages are detected, then the process exit code test works normally. There is also an option to disable reporting of the ignored errors, if you don't want to use the extra memory and bandwidth to report them in the history.

New option to automatically reset job failures when a job is deleted

In the Configure Master dialog you can now set the Master to automatically reset all failures from a job that gets deleted. Without this, once you delete a job the only way to reset the failures from that job is to reset the engine failure counts on every engine that failed on that job.

Extensive performance optimizations throughout

All Smedge components have been thoroughly profiled and optimized to make the system much more responsive and scalable.

Customizable GUI window and improved filtering

The views available in the GUI are now completely customizable. By using the menu command View > Customize Views, you get a new tab that allows you to create your own custom tabs in the interface. You can configure the default “filters” that are used for the views, and give them your own names, as well as change the order in which the views appear. There is also an option for removing views and for restoring the default views. For each view, you can select which list controls will be shown and use the filters to customize which items will populate the lists.

There are also more filtering options available now. Besides the name, createor and pool, you can now filter based on the job status and active products, and you can specify if you want to filter based on all of the options or any of them (AND or OR the filter items). The filtering you set up for your custom view here does not affect the filter that is also available at the top of each view itself, so you can still further filter each custom view to find specific items in each list.

See the User Manual for more information on the customization options and how it all works.

Controllable Display columns in VIewLists

You can now control the columns displayed in each view's Engine, Job, and Work ViewLists. You can right click the header / name of each column to bring up a context menu with some manipulation options. You can move the column you selected by right-clicking, left, right, to the extreme left, or to the extreme right. You can remove a column from being displayed by clicking on “Remove” or clicking on the name of that column if its checked as being displayed. If the column name in the menu is not checked, you can check it to bring that column display up. Theres a restore defaults option to reset the column display to what smedge originally displays. Finally, theres a Custom columns option where you can define yourself, whats displayed with a parameter command string, in the form “$(ParameterName).”

Improved GUI component process control for Master and Engine running normally and as services (and lock files)

The Components submenu of the System menu is now more clear and gives you more options for controlling the Master and Engine componetns, both as normal processes and as services. There is also a new command available to attempt to remove the “lock file” for these components. This is helpful when something has left a stale lock file that is blocking the component from starting correctly.

Individual dispatch reporting for specific Jobs and Engines

You can now get an individual dispatch report for specific Engines or Jobs to find out why that Engine is not starting work or that Job is not being dispatched. This can make it a lot easier to see what is going on, especially if you have a very large number of jobs or engines, and don't want to have to deal with the entire dispatch log, which can get quite large. To get the report, select an Engine or Job and choose the appropriate View Dispatch Report command from either the Job or Engine menu. These commands are also available in the context menus for the Job and Engine lists.

Job shell list command can now filter the results and can report a count as well as listing the matched jobs

The Job command line Shell executable has new options available when listing jobs. You can now supply similar filtering as the GUI list controls, to limit the results displayed to the jobs that fit your criteria. You can filter based on the name, creator, pool, status and type, with options to negate the filtering (i.e., all jobs except those named “foo”), and you can set up if the filtering requires for all items to be matched or any one of them.

Engine shell can set product options

There is a new Engine shell command that allows you to set individual product options by command line. For example, if you want to update the path to the executable for a specific product, you can supply the product name or ID, then the option name (e.g. “Executable”) and the value for the new option. Using the shell you can set the option for multiple machine simultaneously just like any other Engine operation.

New Submit shell built-in Pool “My Computer”

You can supply the pool name “My Computer” to the Submit shell, and it will set the pool for the Job it creates to the ID of the local machine. This is the machine that is running Submit, so this is most useful for scripts that integrate with a third party application, allowing end users to easily submit jobs that run only on their local machine, without them requiring to set up a pool manually or know the ID of their local machine.

Submit shell can update individual parameters for a job

You can now use Submit to change individual parameters for existing Jobs in the system. The new command is “Update”, after which you must supply the ID of the job to update, the name of the parameter to change, and the new value.

Conspectus window can scroll if there are more engines than fit on the window

Instead of becoming impossible to read, when there are more engines than can fit in the window, Conspectus will now make its display scroll so you can see every engine at a reasonable size.

You can now reset failure counts for offline engines

Because the failure counts are maintained by the Master, there is no reason not to allow resetting failures for Engines that are not actually online, so now you can.

New Product: V-Ray Standalone

There is a new Virtual Module to control the V-Ray standalone renderer.

New Product: imgcvt

There is a new Product defined by the Maya.sx module to control the imgcvt image conversion tool that is included in all Maya distributions. This module can spread the conversion of an image sequence across a farm making it very fast to convert from one format to another. It will also automatically detect the sequence specifiers and range of a sequence when you browse for files in the Smedge GUI.

Maya Module correctly detects new image output formats in Maya 2012

mental ray changed how it outputs the filenames in Maya 2012, and the Maya module has been modified in order to correctly detect image files from both Maya 2012 and earlier versions.

Maya submit script improvements

The smedgeRender.mel script that allows you to submit jobs to Smedge from directly inside of Maya has been greatly improved. It now provides nearly identical control as using the Smedge Gui to manually submit a job. It also now allows you to configure the path to the Submit executable yourself with a nice GUI interface in Maya instead of having to edit the script manually. You can download pools and select which pool to use, and it gives some advanced options, like first copying the scene file to the Maya renderScenes folder in your project to avoid possible issues as you keep working. Plus it now automatically integrates itself into the Render menu in the Maya GUI when you source the script.

The openPipeline version of the script is now also included in the Smedge distribution. This script is called op_SmedgeRender.mel and provides the same level of integration as the normal smedgeRender script, but is designed to work with the openPipeline project management system. To use this script, you should put it into the “addons” folder of your openPipeline installation, and it will be automatically available when you start openPipeline.

After Effects integration script improved

The After Effects submit script has also been significantly expanded. It is now designed to create a dockable panel that you can use as part of your AE interface to submit your project to Smedge. It allows you to download the pools from the master and select a pool, as well as configure several

other options for how the job will render. You can also now use a simple graphical UI from inside of AE to set up the path to the Submit executable, so you no longer need to manually edit the script file to get it to work correctly.

Improved error detection for Cinema 4D renders

Several new possible error messages were added to the Cinema 4D virtual module file to help aid in detecting when renders fail with Cinema 4D. Unfortunately the C4D command line renderer is quite terse, so even with the new messages it can be a bit of a challenge to figure out what is actually wrong, but at least more of the failures will be correctly detected as failures now.

Cinema 4D submit script

There is a new, very simple, submit script for submitting jobs to Smedge from inside of the Cinema 4D interface. This script is still quite simple, and you must manually specify the path to the Submit executable in the script for it to work correctly. It does not currently provide any interface.

3D Studio Max submit script

There is a new, very simple, submit script for submitting jobs to Smedge from inside of the 3D Studio Max interface. This script is still quite simple, and you must manually specify the path to the Submit executable in the script for it to work correctly. It does not currently provide any interface.

Nuke submit script

There is a new simple submit script for submitting jobs to Smedge from inside of the Nuke interface. This script is still quite simple, and you must manually specify the path to the Submit executable in the script for it to work correctly. The script provides a very simple interface for suppplying some of the basic job parameters when you submit the job.

Modo Module shows additional statistics and fixes

Modo module now shows extended output information. It has been tested with Modo 601 and works fine. There is also a fix that allows you to correctly override the output path for your Modo renders.

Platform Specific Changes

RPM installer for Linux

Both the 32 bit and 64 bit builds of Smedge for Linux are now available as RPM files. The RPM installs Smedge into a standard location (`/opt/smedge`), and creates the machine folder (`/etc/smedge`) with appropriate permissions. The RPMs also handle the required dependencies for the Smedge component applications. Of course you can still download Smedge as a compressed TAR archive, and install it in a custom manner if you wish, but you will need to make the machine folder manually, and you will need to install any required packages yourself.

Linux and Mac have more appropriate file permissions

Though not technically a problem, the Unix based distributions should have more appropriate permissions on the component files. Only the executable files will have the executable bits set.

Mac now has version info available

The Mac release now displays the Smedge version information when you use the Get Info command in the Finder on the Smedge application icon.

Work around for bug in Mac runtime libraries that could cause hangs writing to the log files

Smedge now works around a bug in the Mac runtime libraries that could lead to a component hanging if it tried to write too much data to a file. The problem still exists (it is a bug in either the low level OS or the C runtime libraries provided by Apple), but Smedge should stumble into it only extremely rarely now.

New SequenceDistributor parameter $(ActualPacketSize)

This new parameter in SequenceDistributor will report the actual number of frames in a packet, as opposed to the **$(PacketSize)** parameter, which is the maximum number of frames in a single packet. Where **$(PacketSize)** comes from the parent job and will be the same for all workers, this parameter will vary depending on how many frames are actually in a specific work unit.

New ProcessJob error controls

ProcessJob has added some more advanced error handling controls. When an error is ignored, by default it now reports the ignored error in the Job History. You can use the **$(ReportIgnoredErrors)** parameter to view or adjust this behavior. Ignored errors will also now trigger Smedge not to examine the process exit code, as well, which is controlled by the parameter **$(ExtendIgnoredErrors)**.

ProcessJob also now monitors the child process for the memory and processor usage, which it can report in the Job History. (See here for more informaiton.) Though the values are not reported until the end of the work, Smedge can now monitor the child process to ensure that it exceeds a specifiable amount of memory usage within a certain period of time. This is useful for catching failed renders that may be caused by infinite loops in the rendering process at the start of rendering. (For example, 3DS Max can have this issue, and Smedge has some low level defaults that should catch this kind of failure most of the time). This can speed up detection of renders that will hang indefinitely without having to wait for the render to exceed the Overtime Kill ratio. The parameters that control this are **$(StartupMemoryJob)** to set it as a Job parameter and **$(StartupMemoryEngine)** to set it as an engine product option. The value is formatted as: *memory-in-mb/seconds*.

And you can access the result processor and memory usage events at the end of the work using **$(ElapsedRealTime)**, **$(ElapsedProcessTime)**, and **$(PeakMemoryUsage)**.

New RenderJob Parameters

The new RenderJob functionality has associated new parameters to support it. To set or query if the image files are reported to the Master, use the **$(ReportImages)** parameter. To set or query if the automatic scene contents translation system is enabled, use the **$(TranslateSceneContents)** parameter. To set or query the font synchronization folder, use **$(FontSyncFolder)**.

Job class includes a scoped helper class to disable automatic parameter transformations

The new scene translation system is implemented using an automatic transformation applied to the parameter value when it is queried. In order to allow normal operation (where the raw value is returned), the API now provides a simple scoped class you can implement to disable the automatic

transformation from being applied. Any future automatic transformations that may be added will automatically be part of this system, future-proofing any code that makes use of these systems.

New scoped Increment/Decrement template classes

Several scoped classes have been handled to better handle incrementing or decrementing values in an exception safe manner.

Updated to wxWidgets 2.8.12

Smedge now uses wxWidgets version 2.8.12 (the current latest in the “stable” development branch). For best results, you should use this same version of wxWidgets if you want to create your own graphical components that make use of the API.

Major Bug Fixes

Fixed several possible crash and hang points

Crashes are a fact of life, but several causes of crashes and hangs in the Smedge code have been detected and resolved or avoided. This version of Smedge should be far more reliable, even under the heaviest of loads. With a system with this much open operation and this much dependance on the reliability of third-party applications, of course it is impossible to avoid all possible ways that you can crash the machine, but the code is greatly improved and Smedge itself should be the cause of these problems vary rarely.

Master was ignoring the -Database flag

The Master was ignoring the request for an alternate database location using the -Database flag.

Master did not always honor requests to see the Dispatch.log file

The Dispatch log was not always generated and there could be a delay between the request to view it and the actual generation of the file that would cause the GUI to report that no log was available. Now the log is always generated and available, though the contents are only actually filled in when a request comes to view them (to avoid the performance penalty of logging the dispatch loop). Note that the new Job and Engine dispatch reporting system uses the same logging messages as this log, but filters only the messages appropriate for specific Jobs and Engines, so it is probably a more useful tool than parsing the entire dispatch report, which can be quite verbose.

Mirror Masters ignored the request to shutdown correctly if the Mirror option was not set in the Master Options

This could cause component processes not to shut down correctly, leading to strange behavior if there was a problem with a master, and difficulty in updating Smedge on a machine (since program files were still in use). This should no longer happen.

Mirror Masters did not always persistently mirror the original primary master

Some of the Master data was not being correctly persisted by the mirror instances of the Master on other nodes. They would work correctly when taking over duties as the primary Master, but after a restart, some data could be lost, including Engine settings, license codes, and Pools. These issues have been corrected and mirrors should be complete accurate copies of all data both when taking over while running and after restarting.

Clients could refuse to connect to the Master

It was possible for the communication system to get into an invalid state where the client thought it was connected but the Master did not agree. The conditions that would lead to this have been corrected, so communciation is more reliable and you should not get the random engines that drop out or the random GUI processes that refuse to connect to the master.

Engine did not always update settings correctly when editing engine settings

Some engine settings were not being correctly updated, due to an issue with the “default” engine system. This is now corrected, so all engine settings are correctly updated.

Editing a Job that is in progress could cause rendering of already completed frames

There was a bug in the way the job distributor data was being managed that would cause all status information to be discarded when a change was made to a job parameter. This would cause any completed progress to be ignored and could lead to repeated rendering of a job.

GUI sometimes did not update the Info Panel when you selected an item

The Info Panel could sometimes get out of sync with the selection in the list.

GUI will not show percent done at 100% if the job is not yet finished

This was a rounding error. If a Job was at a percentage complete of 95.5% or higher, this was being automatically rounded to 100%. This looked weird (100% but not complete?) so it is adjusted now to never exceed 99% if the job is not complete.

Virtual Modules now correctly translate File and Dir type parameters

Fixed an issue with the Virtual Module system that caused custom parameters of type Dir or File not to be translated correctly.