TELESTREAM

Admin Guide 5.3

Note on License

The accompanying Software is licensed and may not be distributed without written permission.

Disclaimer

The contents of this document are subject to revision without notice due to continued progress in methodology, design, and manufacturing. Telestream shall have no liability for any error or damages of any kind resulting from the use of this document and/or software.

The Software may contain errors and is not designed or intended for use in on-line facilities, aircraft navigation or communications systems, air traffic control, direct life support machines, or weapons systems ("High Risk Activities") in which the failure of the Software would lead directly to death, personal injury or severe physical or environmental damage. You represent and warrant to Telestream that you will not use, distribute, or license the Software for High Risk Activities.

Export Regulations. Software, including technical data, is subject to Swedish export control laws, and its associated regulations, and may be subject to export or import regulations in other countries. You agree to comply strictly with all such regulations and acknowledge that you have the responsibility to obtain licenses to export, re-export, or import Software.

Copyright Statement

This document is part of the software product and, as such, is part of the license agreement governing the software. So are any other parts of the software product, such as packaging and distribution media.

The information in this document may be changed without prior notice and does not represent a commitment on the part of Telestream.

Trademarks and Patents

- Episode is a registered trademark of Telestream, Inc.
- UNIX is a registered trademark of UNIX System Laboratories, Inc.
- Apple is a trademark of Apple Computer, Inc., registered in the U.S. and other countries.
- QuickTime is a trademark of Apple Computer, Inc., registered in the U.S. and other countries.
- Windows Media is a trademark of Microsoft Inc., registered in the U.S. and other countries.
- RealNetworks, RealAudio, and RealVideo are either registered trademarks or trademarks of RealNetworks, Inc. in the United States and/or other countries.

All other trademarks are the property of their respective owners.

MPEG-4 AAC

"Supply of this Implementation of MPEG-4 AAC technology does not convey a license nor imply any right to use this Implementation in any finished end-user or ready-to-use final product. An independent license for such use is required."

MP3

This software contains code from LAME, `http://lame.sourceforge.net/`. "Supply of this product does not convey a license nor imply any right to distribute content created with this product in revenue-generating broadcast systems (terrestrial, satellite, cable and/or other networks.), streaming applications (via Internet, Intranets, and/or other networks), other content distribution systems (pay audio or audio-on-demand applications and the like) or on physical media (compact discs, digital versatile discs, semiconductor chips, hard drives, memory cards and the like). An independent license for such use is required. For details, please visit `http://mp3licensing.com/`."

OGG Vorbis

This software contains code that is ©2010, Xiph.Org Foundation. "THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "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 FOUNDATION OR CONTRIBUTORS 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."

PCRE

PCRE is a library of functions to support regular expressions whose syntax and semantics are as close as possible to those of the Perl 5 language.

Release 7 of PCRE is distributed under the terms of the "BSD" licence, as specified below. The documentation for PCRE, supplied in the "doc" directory, is distributed under the same terms as the software itself.

The basic library functions are written in C and are freestanding. Also included in the distribution is a set of C++ wrapper functions.

The basic library functions

Written by:	Philip Hazel
Email local part:	`ph10`
Email domain:	`cam.ac.uk`

University of Cambridge Computing Service, Cambridge, England.

The C++ wrapper functions

Contributed by: Google Inc.

The "BSD" licence

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

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- 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.
- Neither the name of the University of Cambridge nor the name of Google Inc. nor the names of their contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "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 COPYRIGHT OWNER OR CONTRIBUTORS 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.

Disclaimer of Warranty on Software

You expressly acknowledge and agree that use of the Software is at your sole risk. The Software and related documentation are provided "AS IS" and without warranty of any kind and Licensor and the third party suppliers EXPRESSLY DISCLAIM ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. NEITHER LICENSOR NOR ANY THIRD PARTY SUPPLIER WARRANT THAT THE FUNCTIONS CONTAINED IN THE SOFTWARE WILL MEET YOUR REQUIREMENTS, OR THAT THE OPERATION OF THE SOFTWARE WILL BE UNINTERRUPTED OR ERROR-FREE. FURTHERMORE, THE TERMS OF THIS DISCLAIMER AND LIMITATION OF LIABILITY BELOW DO NOT AFFECT OR PREJUDICE THE STATUTORY RIGHTS OF A CONSUMER ACQUIRING THE SOFTWARE OTHERWISE THAN IN THE COURSE OF A BUSINESS, NEITHER DO THEY LIMIT OR EXCLUDE ANY LIABILITY FOR DEATH OR PERSONAL INJURY CAUSED BY NEGLIGENCE.

Limitation of Liability

LICENSOR AND THE THIRD PARTY SUPPLIERS EXPRESSLY DISCLAIMS ALL LIABILITY FOR DAMAGES, WHATEVER THEIR CAUSE, INCLUDING DIRECT OR INDIRECT DAMAGE, SUCH AS CONSEQUENTIAL OR BUSINESS DAMAGE, AMONGST OTHERS CAUSED BY THE NON-FUNCTIONING OR MALFUNCTIONING OF THE SOFTWARE. SHOULD LICENSOR OR THE THIRD PARTY SUPPLIERS IN ANY WAY BE LIABLE FOR DAMAGES, EITHER AS PER THE TERMS OF THIS LICENSE OR OTHERWISE, THEN THIS LIABILITY WILL IN NO EVENT EXCEED THE AMOUNT PAID BY YOU FOR THE SOFTWARE. SOME JURISDICTIONS DO NOT ALLOW THE LIMITATION OF INCIDENTAL OR CONSEQUENTIAL DAMAGES SO THIS LIMITATION MAY NOT APPLY TO YOU.

Contents

1 About **Episode Engine**

Episode Engine is intended for transcoding of media material. It will automatically retrieve files from designated sources, transcode them and place the result in output folders for further processing by the next component in your workflow. You can install and run **Episode Engine** either on a single workstation or in a computing cluster to increase your throughput.

With the **Engine Admin** graphical interface you set up input monitoring, prioritise sources, and monitor the progress of transcoding.

Episode Engine comes with a large number of predefined transcoding settings. The distribution also includes the client application **Episode Encoder**, which lets you create new settings for your particular needs.

Episode Engine Pro supports additional functionality:

Episode Engine Pro

Split-and-stitch Increase your cluster throughput even further by splitting large media files into smaller pieces that can be encoded in parallel and then stitched together.

Episode Engine High Availability Option Augment your cluster with a backup server that takes over if the primary server fails for any reason.

SNMP Use the Simple Network Management Protocol to monitor **Episode Engine Pro**.

The **Episode Engine Pro** logotype in the margin indicates features, topics, or procedures that are specific to **Episode Engine Pro**.

Episode Engine is a Universal Application, running on both PowerPC and Intel Macs.

This Administrator's Guide describes how to install and configure **Episode Engine**. For running **Episode Engine** we refer you to the ***Episode Engine*** *User Guide*, which is included in the same distribution.

The reader is assumed to be a system administrator experienced with Mac OS X.

If you have a current maintenance and support agreement with Telestream, you can contact our support team according to the instructions at `http://www.telestream.net/telestream-support/episode-engine/support.htm`. The **Episode Engine Software Development Kit** requires a separate support agreement.

Note than Telestream does not provide support for any third-party hardware or software.

1.1 Document conventions

NOTE

Paragraphs marked like this highlight items of particular importance for the proper function of the software.

TIP

Paragraphs marked like this highlight procedures that can save time or produce particularly good results.

Paragraphs marked like this warn about features which may cause loss of data or failed execution if used incorrectly.

Document references, both internal and external, are shown in italics. Example:
See chapter 2 *Before You Install.*

Literature references are given as numbers in brackets with the full reference in the Bibliography. Example:
See [2].

Directory names, file names, code examples, and prompts, are shown in plain typewriter type. Example:
The file `printer.ppd` can be found in `/etc/cups/ppd/`.

The names of interface components are given in **bold**. Example:
Adjust the time limit with the **Time limit** slider. Select **Quit** from the **Episode Engine** drop-down menu.

Keys to be pressed on the keyboard are displayed in bold typewriter type. Example:
Press **`Return`** to select the GUI installation.

Examples of extended dialogue will include the shell `prompt>` .

Command syntax is described in Backus-Naur form.

Copy-pasting from the manual is not guaranteed to work, as the text contains formatting information which may not be accepted by the target application.

2 Prerequisites

2.1 Hardware requirements

2.1.1 Minimum hardware configuration

This is the minimum configuration you need to be able to perform encoding at all:

Processor	1.33 GHz G5 or any Intel Mac
Memory	1 GiB
Hard drive	20 GiB, 5400 rpm

2.1.2 Recommended hardware configuration

For the best possible performance we suggest the following configuration for each node:

Processor	Dual 2.66 GHz G5 or Intel Mac
Memory	4 GiB or more is recommended
Hard drive	750 GiB, 7200 rpm or more recommended

For a cluster installation you should connect the cluster nodes using Gigabit Ethernet instead of the normal Ethernet 10/100 standard. To do so, you will need a Gigabit Ethernet switch, otherwise the speed of your cluster will be limited by the 10/100 standard Ethernet switch.

2.2 Software requirements

Episode Engine 5.3.2 requires Mac OS X, version 10.4 or later, and QuickTime™, version 7 or later. QuickTime adds support for various input formats.

Episode Engine High Availability Option requires Mac OS X Server.

You also need **Episode Encoder**, which is included in the distribution.

3 Installing

Episode Engine is delivered either on CD-ROM or as a disk image over the Internet. The distribution contains all necessary installation scripts and manuals.

You have two alternative installers, one graphical, one command-line-based. Choose whichever suits your preferences best.

3.1 Configuring a cluster

Episode Engine can be run on a single computer, but can easily be scaled up to a cluster of multiple computers (nodes) to increase throughput. Note that you cannot combine **Episode Engine** with **Episode Engine Pro** in the same cluster.

There are some issues that have to be considered if you are installing on a cluster:

- Making data and applications available to all nodes.

  All cluster nodes had to be able to access the source files and the application files. Appendix A, *Example configurations* contains several examples of shared storage configurations. Source files should always be placed on the fastest available storage.

  You should also have identical directory structures on all nodes, so that the processes are able to find the right files on their *local* storage.

  A consequence is that the user and group you install **Episode Engine** to run as must have the same IDs on all nodes.

- Load balancing.

  If you have a small cluster (2–3 nodes), you can easily allow encoding on the controller node. Larger clusters require a separate controller node in order for it to keep up.

- Maintaining the installation.

  It will simplify configuration and maintenance if you don't have too imaginative names for your cluster nodes, so that you, e g, don't have to be unsure of whether you remembered to update all nodes. Using the same administrator name and password on all nodes and allowing **ssh** remote login also simplifies administration while also being a security risk; you will have to decide on the appropriate risk level for your organisation.

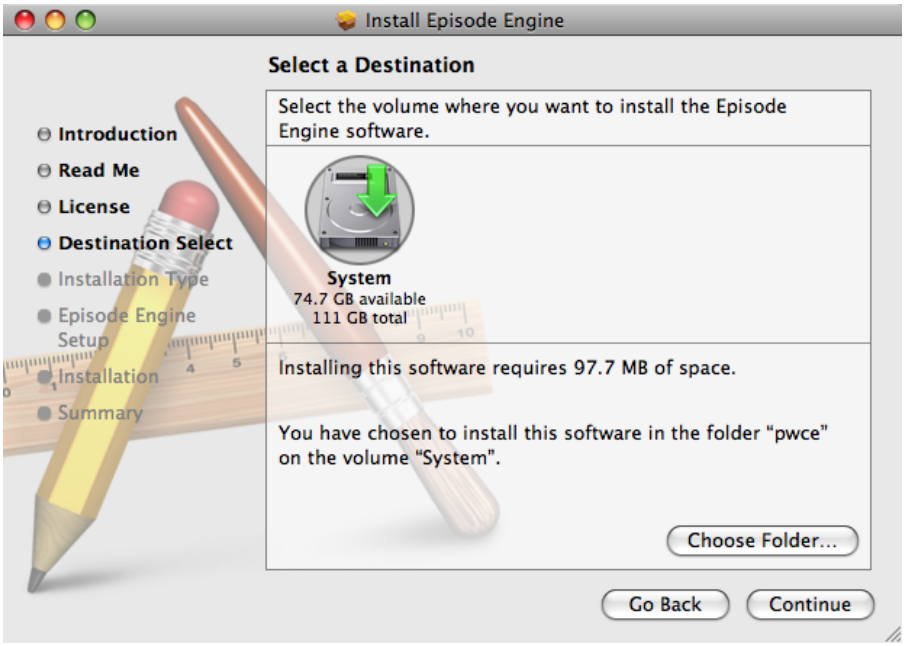
Once you have decided on your configuration, start your installation on the controller node and then install on each of the encoder nodes.

3.2 Graphical installer

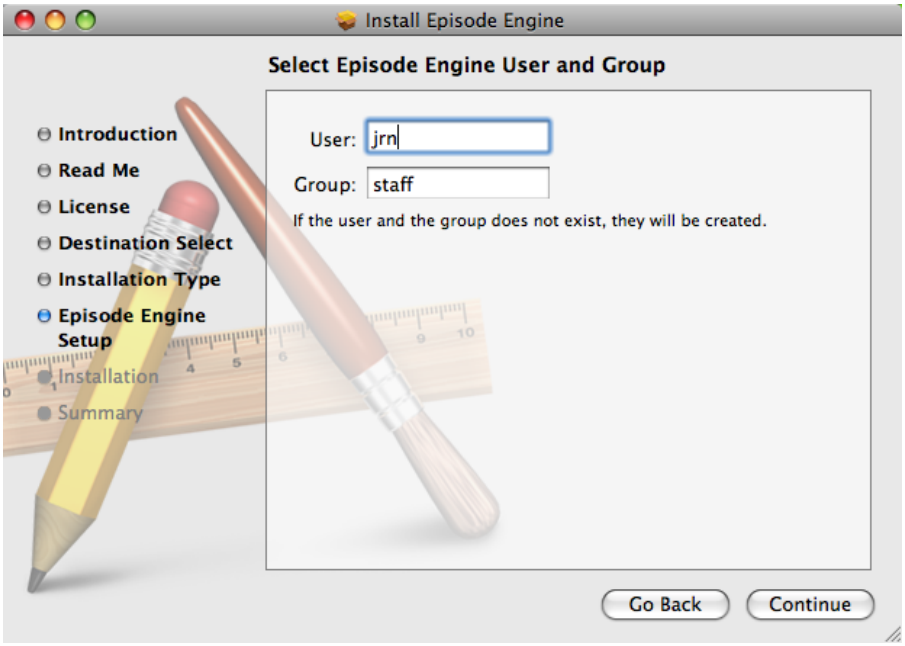
If you do not have any earlier installation and no special configuration needs the graphical installer will be the easiest to use. Double-click `Episode Engine.pkg` to start an installation guide for a simple installation.

Read the information displayed in the installation guide and follow the instructions to install **Episode Engine**. Since some parts of the installation require root or admin privileges you will be prompted to enter an administrator password to authenticate the installation.

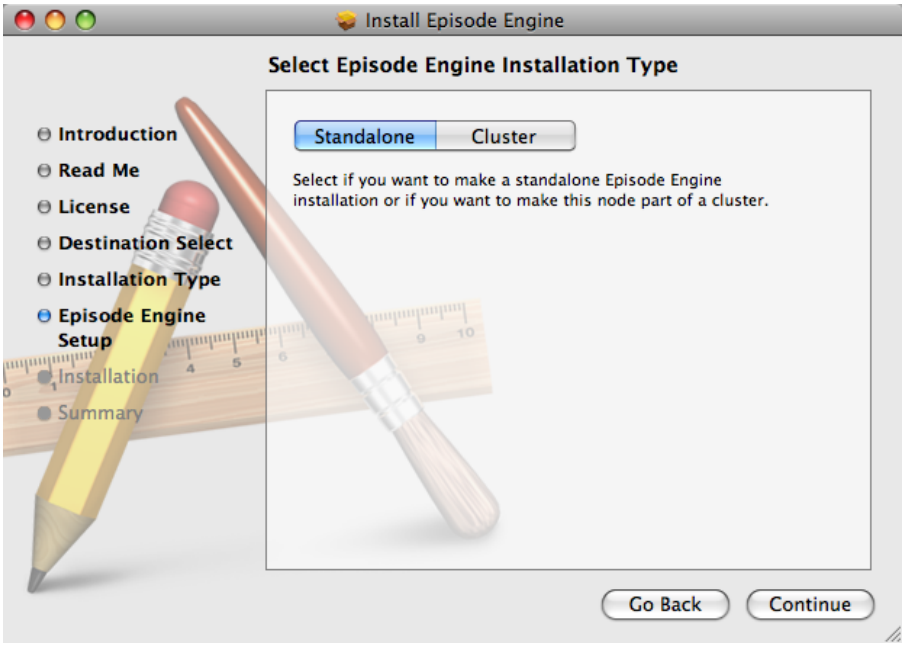
After the preliminaries, you get to choose what disk to install the software on. The folder `pwce` is to be understood as `/usr/local/pwce`, but you can choose any other suitable folder.



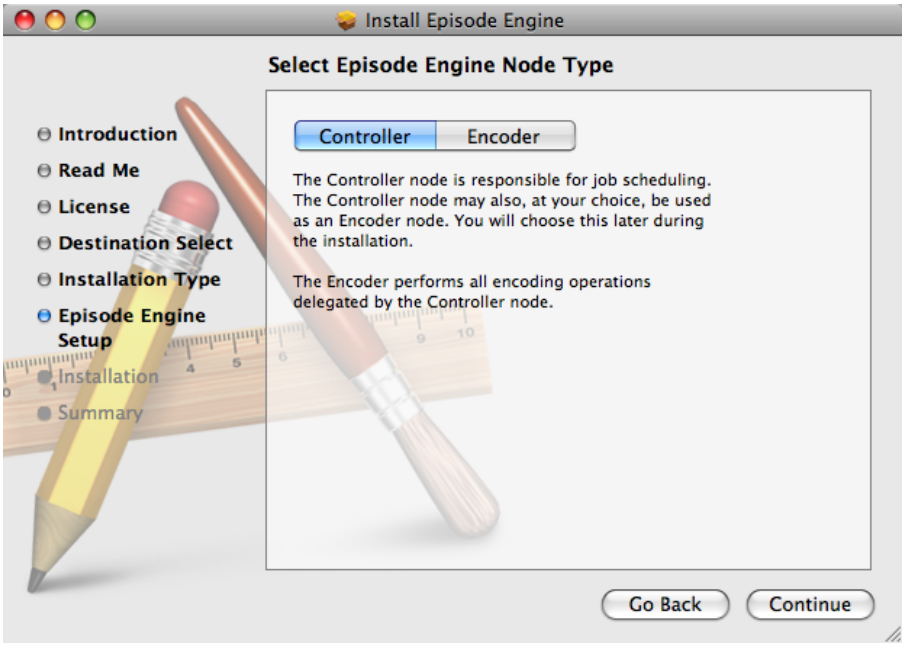
You will then be asked to determine which user and group **Episode Engine** will run as. The default options are your own user and group. This means that **Episode Engine** will see the same volumes and folders that you do. You may prefer to let **Episode Engine** run as a separate user, in which case you should enter suitable user and group names, these will be created by the installer.



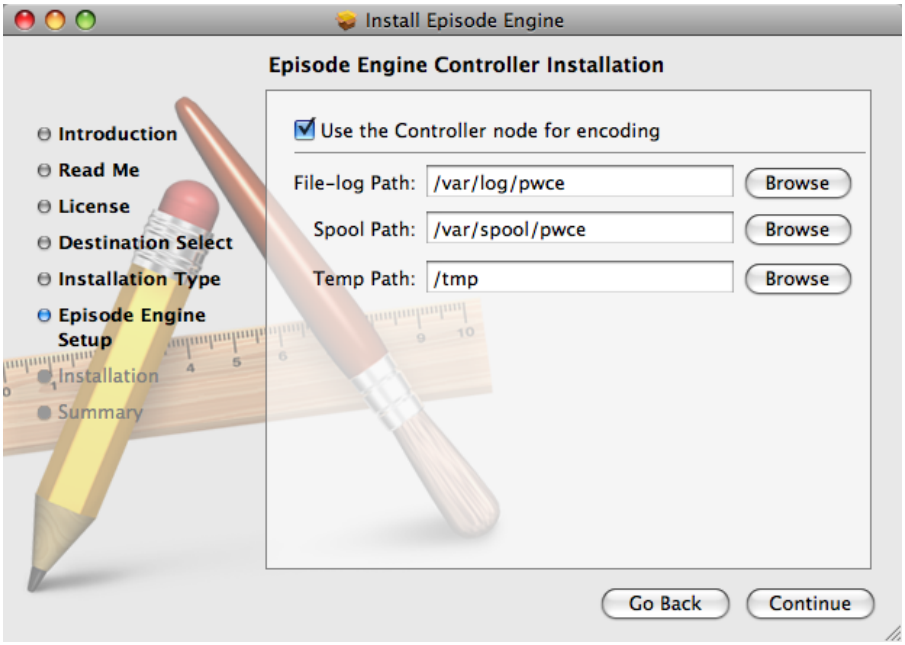
If you want to do a default installation, select the **Standalone** option. Installation will then proceed with no further questions. After installation **System Preferences** will be launched to let you adjust your configuration, see section 4.1, *System Preferences*. In particular you may want to set a password for **Engine Admin** and adjust the watch folder paths.



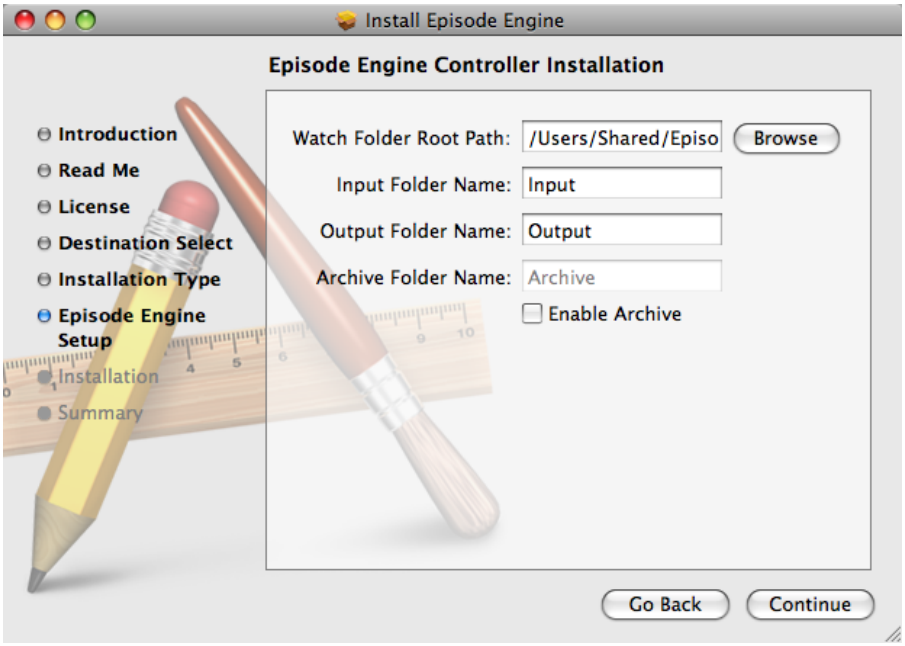
If you need to configure the installation and/or if you are installing on a cluster, select the **Cluster** option. You will then be asked if the current installation is for a controller or encoder, select the appropriate alternative.



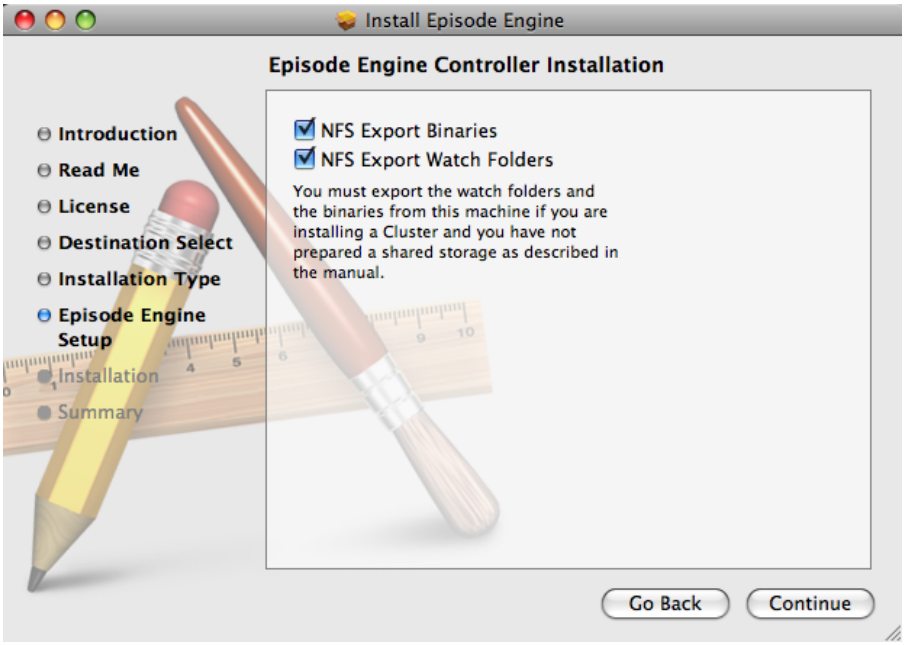
Installing a controller node you are requested to set up the paths to directories used by **Episode Engine**. The default is that the controller node is also used for encoding, but if your controller node is expected to be heavily loaded you may uncheck this option.



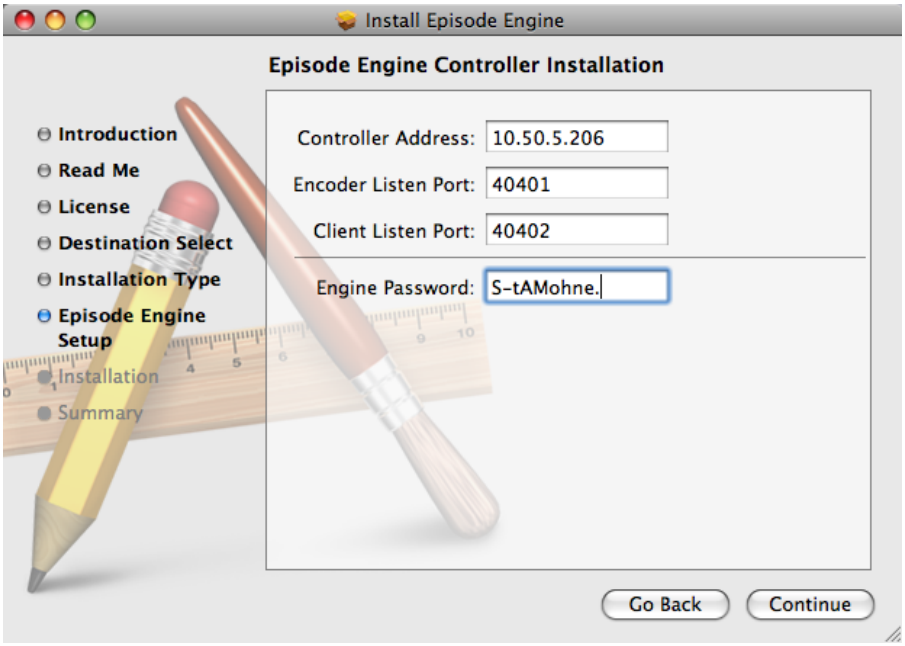
Next, you are requested to set up the watch folders. **Episode Engine** can receive files from different sources, but the classic method is to pick up sources files when they are placed in the watch folders. Normally the sources files are deleted from the watch folders when transcoding has finished successfully, but you can check **Enable Archive** to have the source files archived in a separate folder.



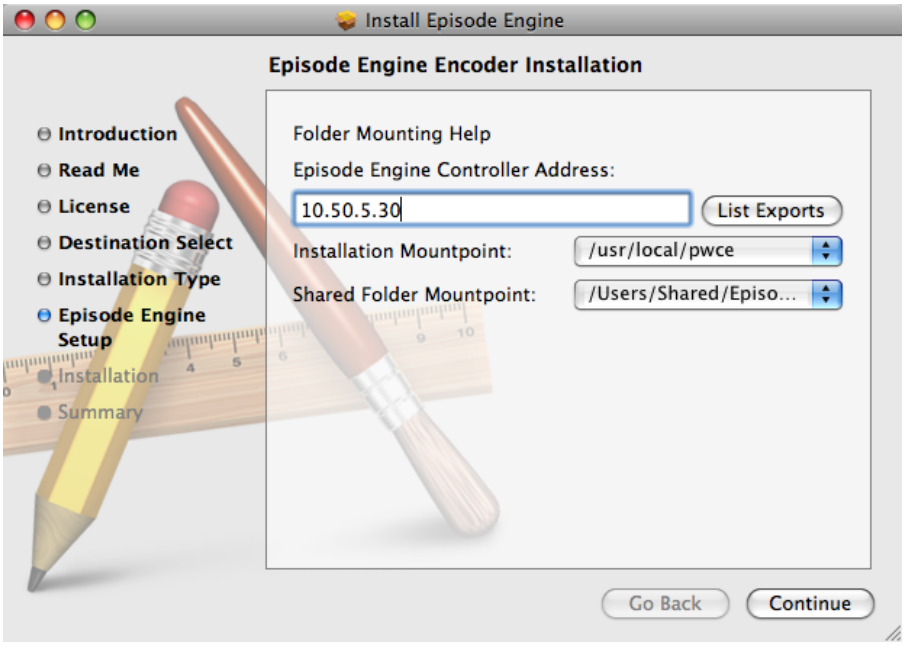
If you have not installed the software on shared storage, you must export the watch folders and binaries to the encoders.



Finally you should set up the address and communication ports where the encoders will communicate with the controller.



The encoders will mount their binaries from the controller. When you enter the address of the controller and press **List Exports**, the installer will query the controller for where the software and watch folders are located and then mount them.



On the encoders you need to manually start the processes with the command **`sudo /usr/local/pwce/script/enginectl start`** after installation.

3.3 Controller installation

3.3.1 Uninstallation

If you already have **Episode Engine** installed, you are recommended to uninstall the previous installation in order to avoid conflicts between different versions of installed files. If the installer script detects an earlier installation it will suggest uninstallation but you can explictly uninstall by double-clicking `Uninstall.command` in the distribution to launch a **Terminal** window running the script. You will be prompted for an administrator password.

```
EPISODE ENGINE UNINSTALLER
    (c) Copyright 2009 Telestream Inc.

  This will start the removal of Episode Engine. Proceed [Y/n]? Yes
Old plist file found and removed.
  Do you want to remove the Episode Engine Controller installation [Y/n]? Yes

  Removing Episode Engine Controller installation in /usr/local/pwce..
    Install directory removed.
    Prefpane removed.
    Preference list removed.
    Startup item removed.
  Do you want to remove the spool directory /var/spool/pwce [y/N]? No
  Do you want to remove the log directory /var/log/pwce [y/N]? No
    User pwce removed.
    Group pwce removed.
    Old installation receipt removed.
    Installation receipt removed.

Uninstallation completed successfully.
```

When the script has finished it will exit the shell and you can close the **Terminal** window.

3.3.2 Installation

To install, double-click `Command Line Installer.command` to launch a **Terminal** running the script. Each section of the installer will let you go back to the beginning of that section and adjust any settings you made in error.

```
Episode Engine INSTALLER
(c) Copyright 2009 Telestream Inc.


This is the Episode Engine installation.
You will be guided through the steps necessary to install this software.
Thank you for purchasing Episode Engine.

Press any key to review the license agreement, then press space,
or use the arrow keys, to scroll the text.

LICENSE AGREEMENT

Please take a moment to configure your Episode Engine installation.
```

```
NOTE: Default values, or the capitalized letter, within brackets are selected
by pressing ENTER at the prompts.
```

If you have an earlier installation, the installation path will already exist, so do not worry about warnings about existing paths.

```
INSTALLATION PATH

  If the path does not exist, it will be created.
  Installation path [/usr/local/pwce]: /usr/local/pwce

USER & GROUP

  If the user and the group does not exist,
  they will be created later during the installation.
  Episode Engine user [jrn]: jrn
  Episode Engine group [staff]: staff
```

The default is to install **Episode Engine** owned and run as yourself, but in the interest of security it may be better to let it run with its own user and group.

```
NODE TYPE

  1. Controller node
     The Controller node, previously called the Master node, is responsible for
     job scheduling. The Controller node may also, at your choice, be used as
     an Encoder node.

     NOTE: If you want to perform a stand alone installation,
     choose 1 for Controller. Also make sure to answer yes to the question
     later about using the Controller for encoding.

  2. Encoder node
     The Encoder, previously called the Slave node, performs all encoding
     operations delegated by the Controller node.

Select a node type for this machine [1]: 1

Is the above correct [Y/n]? Yes

Running preinstall script.. ....

CONTROLLER
  Do you want to use the Controller node for encoding [Y/n]? Yes
```

Even if you are running a cluster, you can use your controller node for encoding if it fulfills the requirements in chapter 2, *Prerequisites*.

```
EPISODE ENGINE PATHS
  Installation path is set to /usr/local/pwce.
  File-log path [/var/log/pwce]: /var/log/pwce
  Temporary path [/tmp]: /tmp
  Spool path [/var/spool/pwce]: /var/spool/pwce

  Is the above correct [Y/n]? Yes
```

The default installation directory is `/usr/local/pwce/`. This is not normally visible in the **Finder**, which may be a problem if you prefer not to use a terminal to interact with the files (rarely needed in any case). In this case, select some other

suitable directory, we suggest a subdirectory of `/Users/Shared/`. If needed, the path settings can be adjusted later in the **System Preferences**.

Episode Engine creates temporary files when performing two-pass encoding of Flash 8 or Windows Media output as well as when creating "Fast start" QuickTime files. If these output formats are often used, the temp directory should be on as fast a disk as possible, as it will be required to move large amounts of data to and from processing. The space required when writing two-pass temp files is $1.5 \cdot \textit{output width} \cdot \textit{output height} \cdot \textit{framerate} \cdot \textit{duration}$ bytes. The space required for "Fast start" temp files is slightly larger than the output file.

Episode Engine can receive files from different sources, but the classic method is to pick up sources files when they are placed in special watch folders. You set up their locations here. **Episode Engine** must have write access as root to watch folders, you can therefore not access watch folders through e g AFP.

Since **Episode Engine** runs as root, there will be no check for space remaining on the temp and output devices, therefore you have to make sure that ample space is available, or you may risk filling the disk completely, which is difficult to recover from.

```
WATCH FOLDER SETUP
  Watch folder root path [/Users/Shared/Episode Engine]: /Users/Shared/Episode Engine
  Input watch folder name [Input]: Input
  Output watch folder name [Output]: Output
  Archive folder name [Archive]: Archive
  Enable input material archive [y/N]? Yes

  Is the above correct [Y/n]? Yes
```

If you already have set up file sharing between the controller and the encoders, you should answer **n** to the questions about NFS export, if not, this is where file sharing is set up.

```
NFS-EXPORTING FROM THIS NODE
  NOTE: You must export the watch folders and the binaries from this machine if
  you are installing a Cluster and you have not prepared a shared storage as
  described in the manual.

  Do you want to NFS-export the watch folders from this node [y/N]? Yes
  Do you want to NFS-export the binaries from this node [y/N]? Yes

  Is the above correct [Y/n]? Yes
```

You have to supply an IP address or hostname for your controller node. You have the option of performing coding on the controller node as well, if you consider it powerful enough. If you have a stand-alone installation and use DHCP you may prefer to use 127.0.0.1 for the controller IP address in order to avoid problems with changing addresses.

```
COMMUNICATION
  Controller IP address [10.50.5.222]: 10.50.5.222

  Is the above correct [Y/n]? Yes
```

You should set up a password for client applications such as **Engine Admin** to access the server.

```
PERMISSIONS AND SECURITY
  Engine Admin password [anonymous]: ennnsynen

  Is the above correct [Y/n]? Yes

Updating intermediate settings file..
Running postinstall script..
Please wait while installer is copying files...
```

The copying of files from the package to the appropriate folders may take a few minutes.

```
NOTE: Your old configuration files were backed up and can be found in /usr/local/pwce/
```

You can give the path to a license file here, or if you prefer to use a file browser to locate it, you can answer **n** to installing a license file and instead use the **System Preferences** as shown in section 3.6, *Installing a license*.

```
LICENSE
  No Episode Engine license is installed. Do you want to install a license now [Y/n]?
  Full path to license file (type 'skip' to skip): /Users/jrn/Desktop/licenses.xml
License successfully installed.
  Do you want to start Episode Engine now [Y/n]? Yes
Episode Engine started.

NOTE: To make additional configuration of your Episode Engine installation
please open the Episode Engine preference pane which has been installed in
System Preferences. Here you can also manage your Episode Engine licenses.

Episode Engine successfully installed.
```

If you did a stand-alone installation, skip ahead to section 3.6, *Installing a license*, otherwise proceed with installing on all encoder nodes.

3.4 Encoder installation

The installation procedure for an encoder node is mainly the same as for a controller node, with the differences noted below. As for the controller node, you are recommended to first uninstall any previous installation.

```
Episode Engine INSTALLER
(c) Copyright 2009 Telestream Inc.


This is the Episode Engine installation.
You will be guided through the steps necessary to install this software.
Thank you for purchasing Episode Engine.

Press any key to review the license agreement, then press space,
or use the arrow keys, to scroll the text.

LICENSE AGREEMENT
```

```
Please take a moment to configure your Episode Engine installation.

NOTE: Default values, or the capitalized letter, within brackets are selected
by pressing ENTER at the prompts.


INSTALLATION PATH

  If the path does not exist, it will be created.
  Installation path [/usr/local/pwce]: /usr/local/pwce

USER & GROUP

  If the user and the group does not exist, they will be created.
  Episode Engine user [jrn]: jrn
  Episode Engine group [staff]: staff


NODE TYPE

  1. Controller node
     The Controller node, previously called the Master node, is responsible for
     job scheduling. The Controller node may also, at your choice, be used as
     an Encoder node.

     NOTE: If you want to perform a Standalone installation,
     choose 1 for Controller. Also make sure to answer yes to the question
     later about using the Controller for encoding.

  2. Encoder node
     The Encoder, previously called the Slave node, performs all encoding
     operations delegated by the Controller node.

Select a node type for this machine [1]: 2

Is the above correct [Y/n]? Yes
```

In this case we configure the encoder to NFS-mount the software directories as well as the watch folder directories from the controller node.

```
Running preinstall script......done

WARNING: The installation path /usr/local/pwce does not exist.
WARNING: The slave assumes this to be mounted from the master node or from the shared
  Do you need assistance mounting the installation directory [Y/n]? Yes

INSTALLATION DIR MOUNTING
Remote NFS-server IP-address: 10.50.5.200
  Listing NFS-shares at 10.50.5.200..

Exports list on 10.50.5.200:
/usr/local/pwce                     Everyone
/Users/Shared/Episode Engine    Everyone
Remote NFS-server binary export name [/usr/local/pwce]: /usr/local/pwce
    /usr/local/pwce successfully mounted from 10.50.5.200.

Do you want to make this NFS-share automatically remounted at reboot [Y/n]? Yes
Binaries are now automounted upon reboot.
Running postinstall script..

  Do you want to start Episode Engine now [Y/n]? Yes
```

```
Episode Engine started.

Episode Engine successfully installed.
```

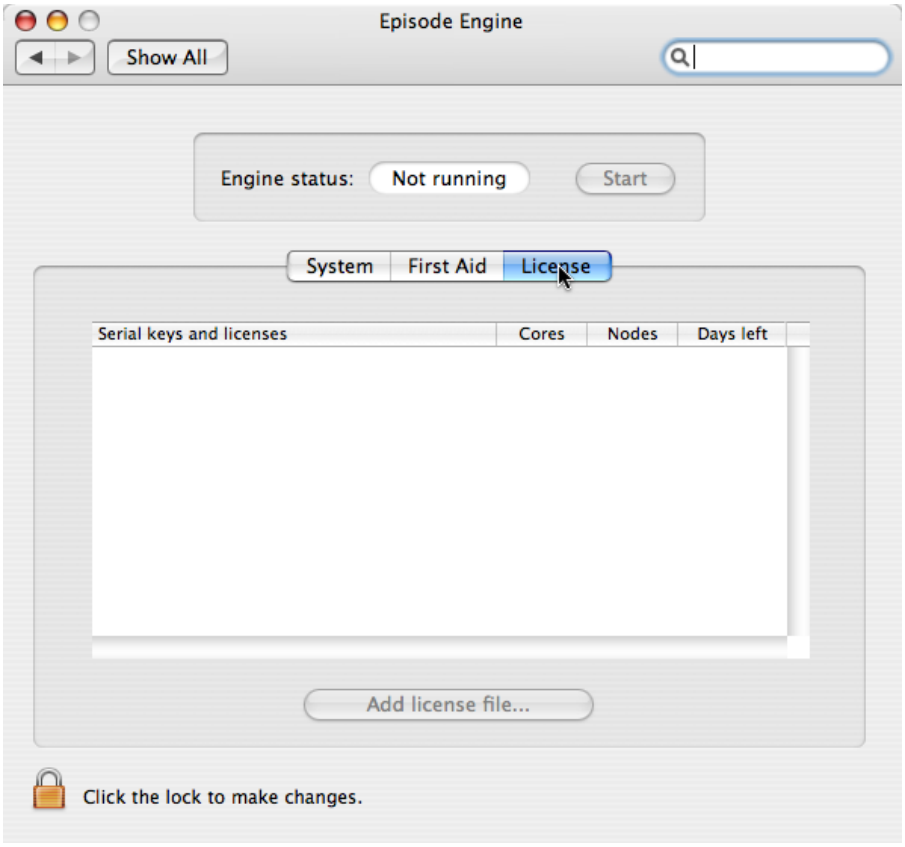
3.5 Installing plugins

Episode Engine can be extended with plugins that retrieve input data from various sources, such as **ftp** servers, VTRs, DV cameras, etc. **Episode Engine** comes with file monitor, FTP monitor, SMB/CIFS monitor and image sequence monitor plugins pre-installed. Additional plugins may be downloaded from the Telestream website. When you have downloaded and uncompressed a plugin, you must place it in either `/Library/Application Support/Telestream/Plugins`, `~/Library/Application Support/Telestream/Plugins`, or the directory given in the plugin−path field in your `engine.conf` file (see appendix B, *`engine.conf`*). **Episode Engine** will now be extended with the new acquisition methods. The ***Episode Engine*** *User Guide* describes how to use the Telestream-developed plugins.

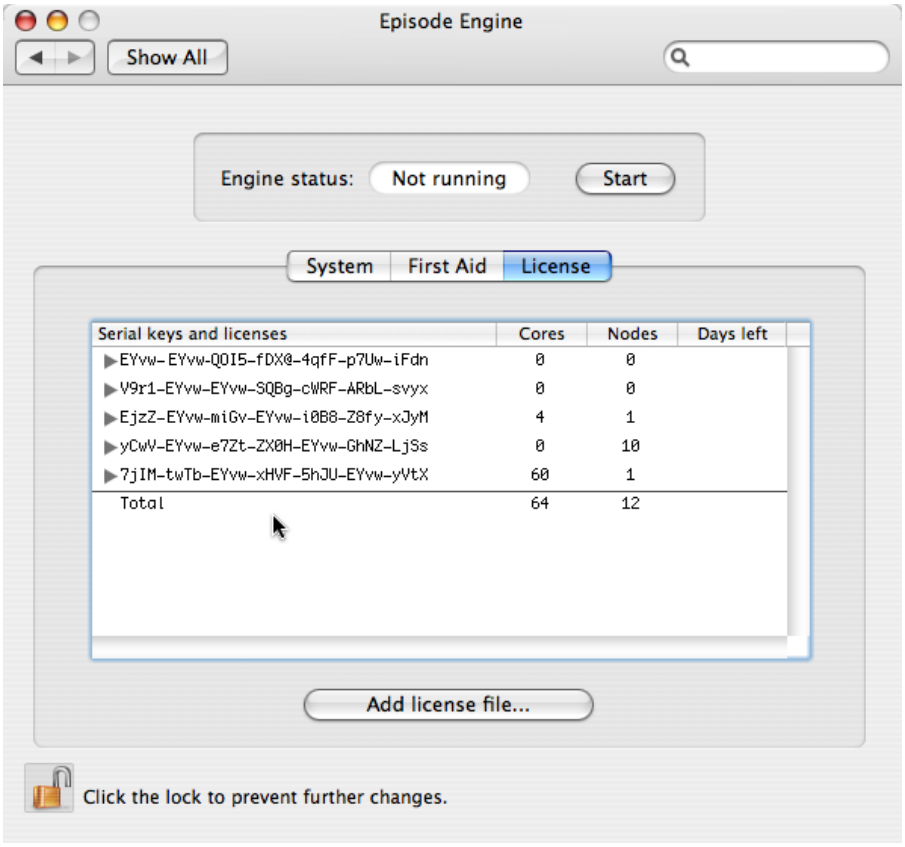
3.6 Installing a license

If you used the graphical installer or did not give a license to the installer script you need to install a license key for **Episode Engine**.

To install the license key open the **System Preferences**, select **Episode Engine** and its **License** tab. Click the lock and enter an administrator password.



Press the **Add license file...** button in the **License** tab and browse to a suitable `licenses.xml` file. Once you have selected a valid license file, the **License** tab will show your serial key and how many CPUs are covered by the serial key. You can expand the license information by clicking on the triangle icon which will show for how long the license is valid. Expanding further you can see exactly what features are included in your license. You can extend your copy of **Episode Engine** with additional features and add their licenses, which will be shown along with your old licenses.



3.7 Installing **Engine Admin**

In addition to the installation scripts the installation medium contains the `Engine Admin` program. Drag this to the `Applications` folder on any computer(s) where you want to monitor the progress of encoding jobs in **Episode Engine**.

4 Configuring **Episode Engine**

The preferred way to configure **Episode Engine** is to use **System Preferences**, but it may be useful to be able to edit the configuration parameters manually when needed. Global parameters are stored in `/usr/local/pwce/etc/engine.conf`, its format is described in appendix B, *`engine.conf`*. Parameters specific to the controller and encoder nodes are stored in `svc-ctrl/services.conf` and `svc-encoder/services.conf`, respectively. Their format is described in appendix C, *`services.conf`*.

NOTE

If you are upgrading from an older version the previous versions of your `etc/engine.conf`, `svc-ctrl/services.conf` `svc-encoder/services.conf` will be saved with a `.old` suffix in each respective directory. The old settings will not be used for the new configuration files, but you can manually edit the new files to conform to your old settings.

4.1 System Preferences

The **Episode Engine** System Preferences window is divided into the tabs **System**, **First Aid** and **License**.

Episode Engine Pro

Episode Engine Pro also has an **SNMP** tab that becomes visible when a valid license file has been added.

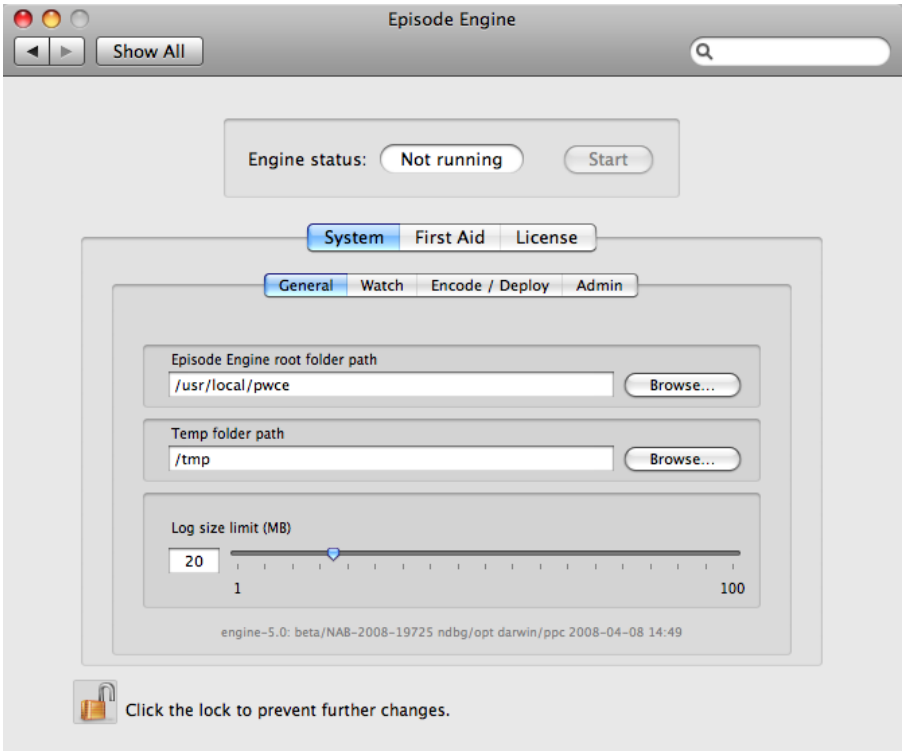
4.1.1 System

The **System** tab is in turn divided into subtabs: **General**, **Watch**, **Encode / Deploy**, and **Admin**.

Episode Engine Pro

Episode Engine Pro also has a **Split'n'Stitch** subtab.

General



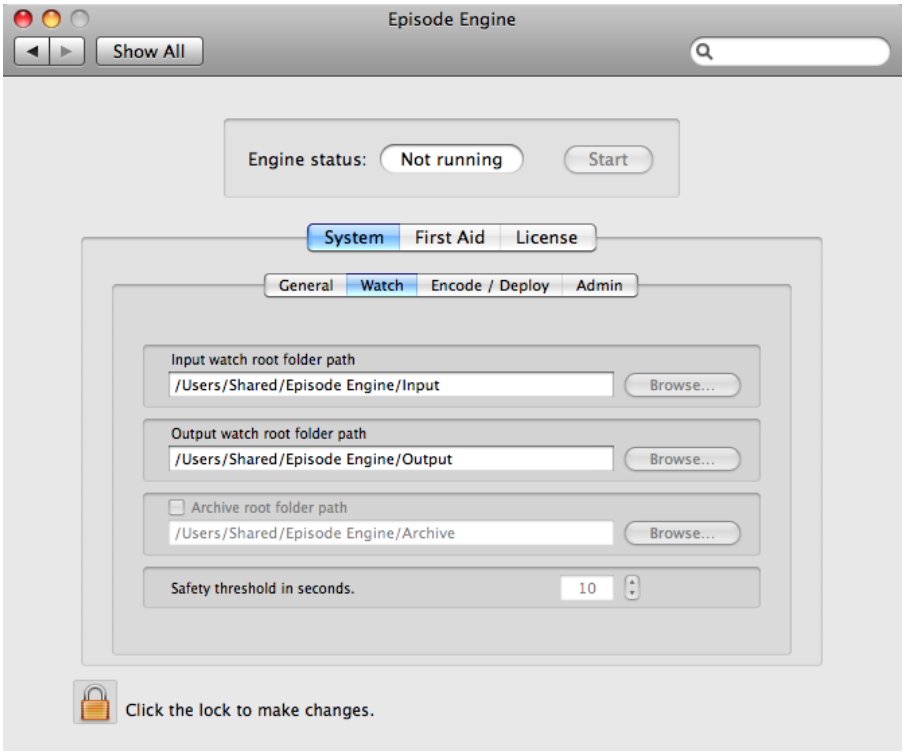
Episode engine root folder path The root folder path tells **Engine Admin** where **Episode Engine** is located. The default **Episode Engine** location is `/usr/local/pwce`. If you change the location of **Episode Engine** you will have to change the path in the preferences. Press the **Browse...** button and set the new path.

Temp folder path The temp folder path is the directory that is used for the temporary files created by **Episode Engine**.

Log size limit (MB) During processing an event log will be written to `/var/log/pwce/pwce.log` (default location). The size of the log file is limited by the field filelog−filesize in the `/usr/local/pwce/etc/engine.conf` configuration file, see appendix B, *`engine.conf`*. If the log file grows larger than this size, it will be renamed `pwce.1.log` and a new empty `pwce.log` created. Any pre-existing `pwce.1.log` will have been renamed `pwce.2.log` and so on, up to a maximum number of log files which is determined by the field filelog−num−rotation in the configuration file. **Log size limit (MB)** is the product of filelog−filesize and filelog−num−rotation and is thus the total size of all log files.

The syntax of the log file can be found in appendix D, *The Log File*.

Watch



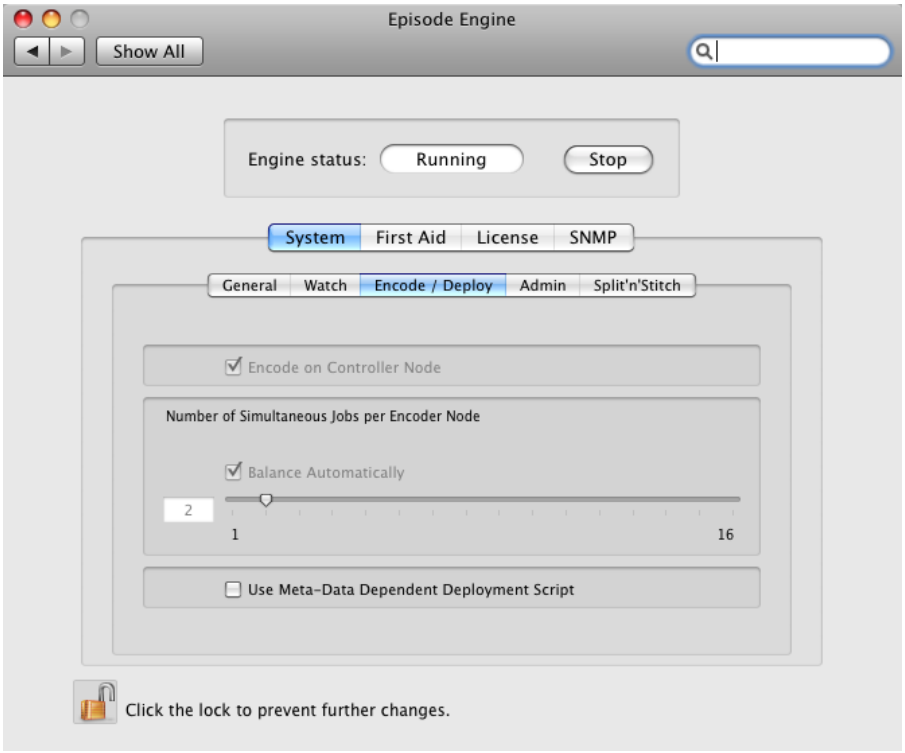
Input watch root folder path The input watch folder location. You create your watch folders in the input watch folder. The default location is `/Users/Shared/Compression Engine/Input`. To change the location of your input watch folder, create a folder named, e g, `WatchIN` in a location where the user `pwce` has read and write access and use the **Browse...** button to set the new path to this folder.

Output watch root folder path The output watch folder location. The encoded material is placed in subfolders of this folder. The default location is `/Users/Shared/Compression Engine/Output`. To change the location of your output watch folder, create a folder named, e g, `WatchOUT` in a location where the user `pwce` has read and write access and use the **Browse...** button to set the path to this folder.

Archive root folder path The default behaviour of **Episode Engine** is that source files that have been transcoded are deleted from their input watch folder, but you may wish to keep your source files. In this case, check the box and enter a value for a Archive root folder. Just as for the output watch folders, transcoded files will be placed in subfolders of the root folder, with the same name as the input watch folder.

Safety threshold in seconds. When **Episode Engine** finds a new file in a watch folder, an external process may still be writing to it. Thus the watcher will check again after a safety threshold. If the file has not changed during that time, it is assumed that it is safe to start reading from it.

Encode / Deploy

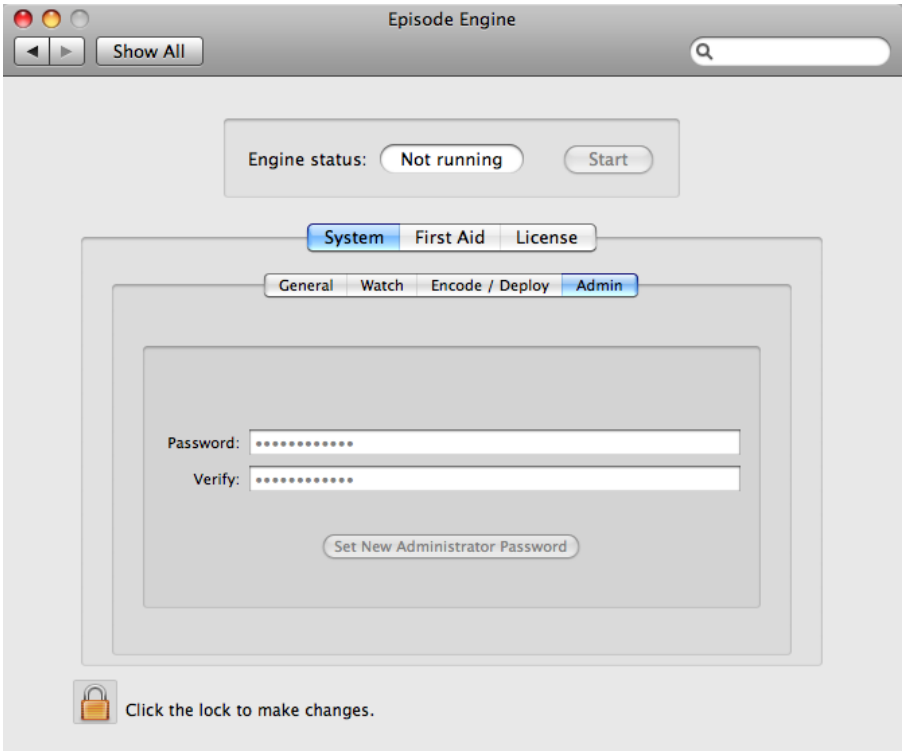


Encode on Controller node turns on and off encoding on the controller node. It is set to whatever value you specified during installation; if you find that encoding slows down the controller you can turn it off, if you have excess capacity you can turn it on.

The slider **Number of Simultaneous Jobs per Encoder Node** sets the number of jobs that can be encoded simultaneously on each node in the range 1–16 jobs. You are recommended to check **Balance Automatically** to let **Episode Engine** estimate an appropriate value for each individual node, but you may set a value manually if needed.

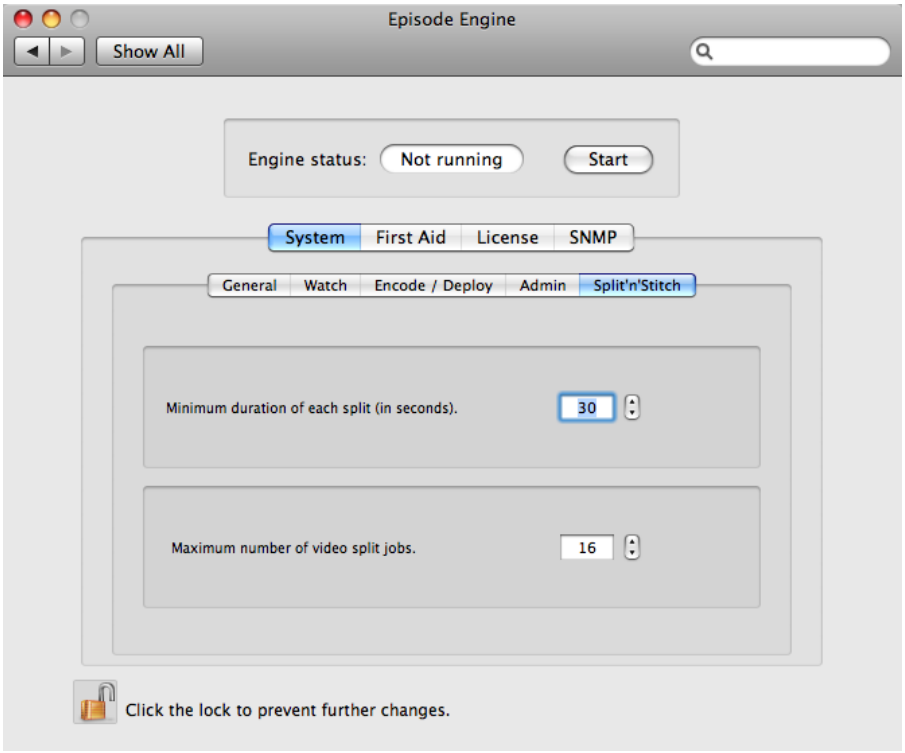
Checking **Use Meta-Data Dependent Deployment Script** enables the script `/usr/local/pwce/evt/00_job_Deployment_Telestream`. This will look for special keywords in the metadata associated with a transcoding job and perform standard actions in response to the values of the keyword parameters. See the chapter *Engine tab* in the ***Episode Encoder*** *User Guide* for further details on how to use this feature.

Admin



The transcoding processes are password protected so that only authorised users can access them with **Engine Admin** and other clients. At installation the system has the default password `anonymous`, but we recommend that you change this password to something of your own choice. Enter your new password in the two fields **Password** and **Verify** and then press the button **Set New Administrator Password** to set the password. Note that you can only set the password while **Episode Engine** is stopped.

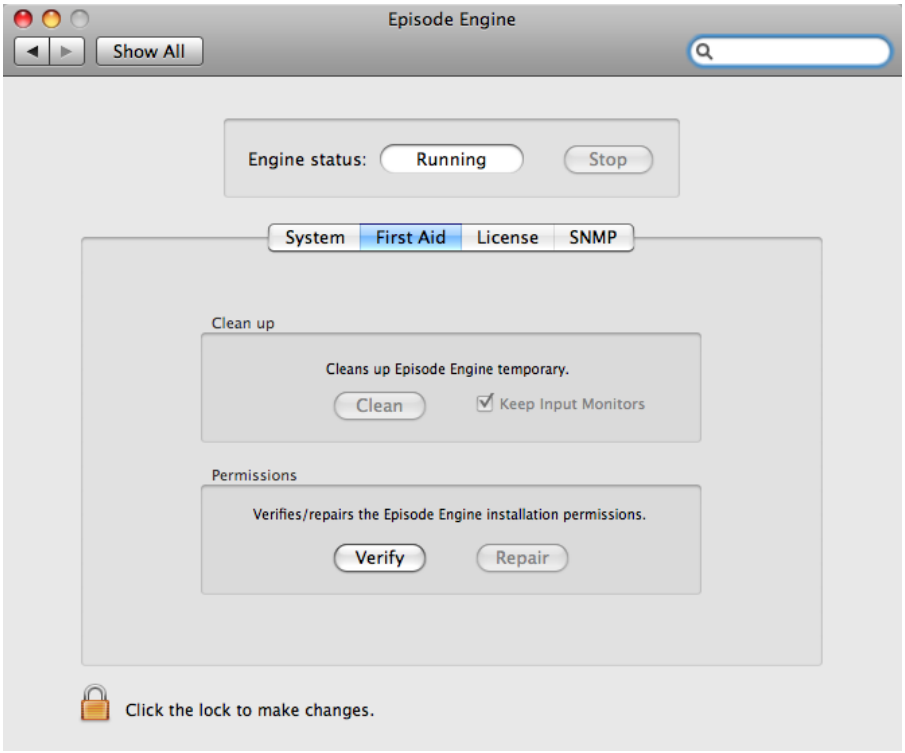
Split'n'Stitch



Episode Engine Pro

With the split-and-stitch function, source data is split into smaller pieces that are distributed over the CPUs in your cluster. If a video file is split into too many segments relative to the amount of material in each segment, there will be no speed-up advantage, since time is lost in the splitting, moving and recombination, so if you find that the automatic splitting algorithms do not give optimal splits on your cluster, you can reconfigure splitting. The **Minimum duration of each split (in seconds).** value lets you set a limit on how short segments a video file will be split into. **Maximum number of video split jobs.** determines how many segments a video file will be split into. (Note that audio tracks are never split, but are transcoded in a single thread.)

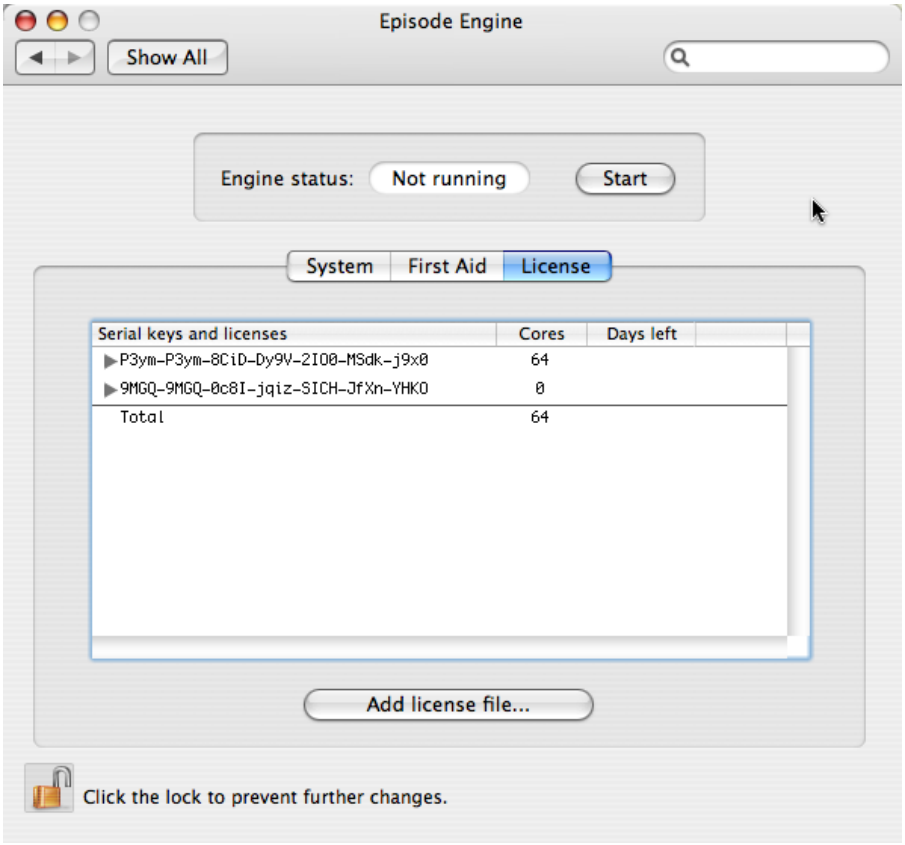
4.1.2 First Aid



In rare circumstances working files may become corrupted. In that case, stop **Episode Engine** and press **Clean** in the **First Aid** tab. This will remove all jobs from the queue and all working files from the temp area. The job history will also be emptied. Checking **Keep Input Monitors** will retain any input monitors. (See the ***Episode Engine*** *User Guide* for a discussion of input monitors.)

Manually managing the directories used by **Episode Engine** may change ownership of files and directories so that the user you installed as no longer can read and/or write files as needed. If you suspect this to be the case you can press the **Verify** button in the **First Aid** tab to check that all directories have the required permissions. If a problem is indicated, press **Repair** to correct the faulty permissions. Note that watch folders are *not* checked by the verify process.

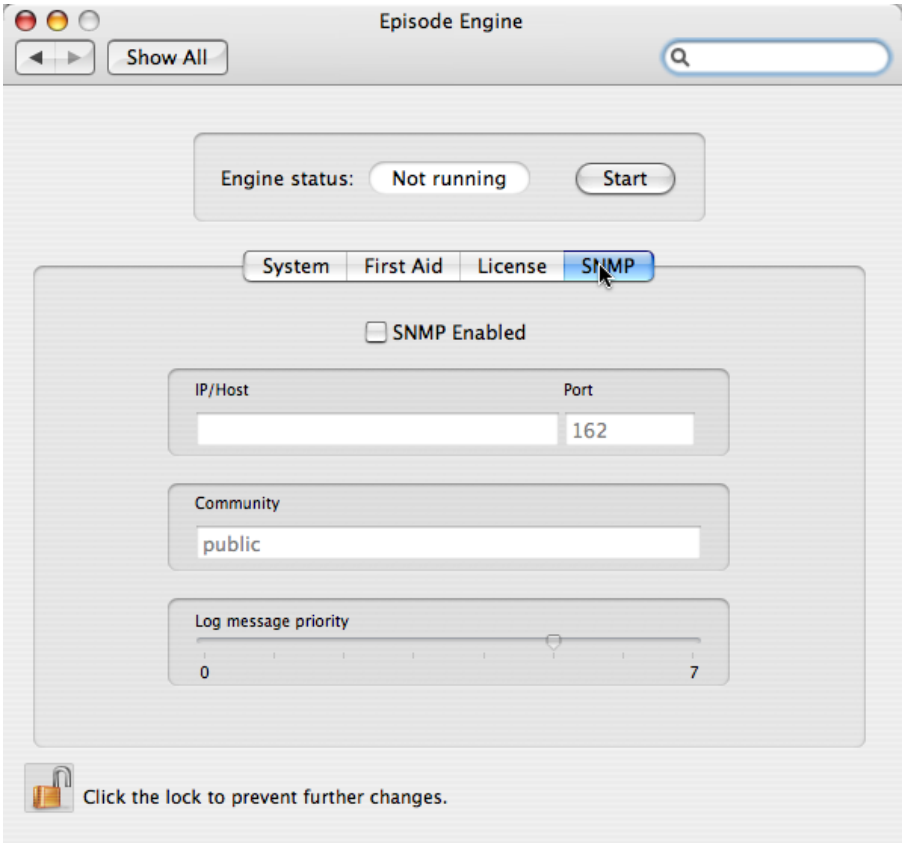
4.1.3 License



The **License** tab is where you handle your license files. The first time you install **Episode Engine** you must provide the system with a valid license file. If you have a time-limited license it is an easy matter to upgrade to a new license file. Either press the **Add license file…** button to find your license file with the file browser or drag the license file from the desktop onto the tab. New licenses will be added to any existing ones.

You can remove licenses by Right-/Ctrl-clicking on a license and selecting **Remove serial key** in the context menu.

4.1.4 SNMP



Episode Engine Pro

SNMP (Simple Network Management Protocol) is used to monitor networks and applications on networks. You can configure a device or application to send messages to an SNMP management station, where they can be displayed in various ways by suitable applications. You can read more about SNMP in e g [1].

Checking the box **SNMP Enabled** lets you monitor the state of **Episode Engine** via SNMP v2c Notifications.

In the fields **IP/Host** and **Port** you set the host that is to receive the log messages on the given port. In the **Community** field you set which SNMP community (group of managed units) is allowed to see the messages.

In **Log message priority** you set the severity cutoff of the messages you will receive (a low number means only more urgent messages will be transmitted, a higher number that less important events also will be reported).

If enabled, your management station will receive log messages in the format described in appendix D, *The Log File*.

4.2 Storage depots

Episode Engine can receive job submissions from various clients. In order to achieve this, **Episode Engine** has to inform the clients of available *storage depots*, storage volumes that are accessible to both **Episode Engine** and the client,

allowing the exchange of media files.

The storage depots made available by your installation are defined in `/usr/local/pwce/etc/depots.conf` which is created during installation on the controller node. It is an XML file describing the storage depots, their mount points and how they can be reached.

```
<?xml version="1.0" encoding="UTF−8"?>
<depots>
  <depot>
    <name>Default</name>
    <mount−point>/Users/Shared/Episode Engine</mount−point>
    <sub−dir>Depot</sub−dir>
    <url>nfs://server.company.com/Users/Shared/Episode Engine</url>
  </depot>
</depots>
```

The file contains a list of <depot> items, each describing a storage depot. <name> gives the identifier of the depot. <mount−point> is the directory where **Episode Engine** has mounted the depot. <sub−dir> gives a subdirectory of the mount point that a client is allowed to access. <url> gives a URL through which the depot can be accessed by a client; there may be multiple <url> tags indicating different access paths to the same depot.

4.3 Links to files

You can save on disk space by using links instead of physically copying the file to be encoded to the input watch folder, but perhaps more importantly, avoiding the copying of large files to the encoder nodes can decrease encoding time dramatically. However, this causes some additional complications in clusters where linked files must be visible to all nodes.

There are three methods of creating links under OS X: 1. hard links, which are pointers to a file under a different name, 2. symbolic links, which contain a reference to the name of the original file, and 3. **Finder** aliases, which are similar to symbolic links, but are updated if the original file is moved.

Hard links can only refer to files on the same volume, so a link on local storage cannot refer to a file on shared storage.

Symbolic links just contain the name of the referenced file, so they can be used cross-device. This however also means that all named directories have to exist on all nodes, with the same permission settings. If you set up your cluster with identical directory structures on all nodes as recommended in section 3.1, *Configuring a cluster* this should automatically be the case.

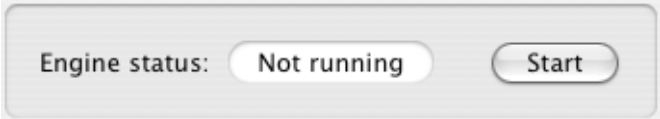
You can get technical details on links on the UNIX manual page ln(1) and on symbolic links in symlink(7). A more extended discussion can be found in e g [2]. **Finder** aliases are explained in the section *Creating and using aliases* in the **Mac Help**.

NOTE

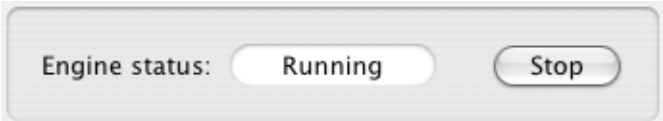
Make sure the volumes mount automatically if the machines are restarted or the setup won't work after a power failure or controlled restart.

4.4 Starting the processes

The last thing you need to do to be able to encode, is to start the **Episode Engine** processes. You can do this in **System Preferences** by clicking the **Start** button.



It will then change to a **Stop** button, with which you can stop the processes.



Alternatively you can use the command-line interface `enginectl` to start, stop and control the processes. By default this is installed as `/usr/local/pwce/script/enginectl`. `enginectl` takes one of four arguments: **`start`** and **`stop`** start respectively stop the **Episode Engine** processes; **`restart`** restarts already running processes and thus forces input of any changed configuration parameters; **`status`** will tell if the processes are running or not. You have to run `enginectl` as administrator.

NOTE

QuickTime, on which **Episode Engine** is dependent, requires someone to be logged on each node running **Episode Engine**.

Once you have started **Episode Engine** you will have the following processes on your computer(s): **pwanalyzer** (input file analyzer for split-and-stitch programmer's interface), **pwdwatch** (creator of dynamic input monitors), **pwevent** (action event supervisor), **pwmon** (monitor process), **pwwatch** (job supervisor), **queen** (job dispatcher), **vinculum** (job receiver), and a number of **drone**s (the actual workers). Note that by default a controller node will also perform encoding and therefore will run **vinculum** and **drone** processes.

The processes communicate over TCP/IP. Should the default port numbers interfere with other applications in your cluster you can change them to other values in `/usr/local/pwce/etc/engine.conf`, `/usr/local/pwce/svc-ctrl/services.conf`, and `/usr/local/pwce/svc-encoder/services.conf` as explained in appendix B, *`engine.conf`* and appendix C, *`services.conf`*.

In addition to these processes a start-up item is added to the system. When you start **Episode Engine** from the **System Preferences** you also activate the start-up item. This means that **Episode Engine** will start automatically after a reboot of your computer. If you stop **Episode Engine** from **System Preferences** you will de-activate the start-up item.

5 Finding and solving problems

While the installation scripts do their best to set up so that things work out of the box, your particular installation may run into problems either immediately or later on. Often the problem is fairly easy to locate and fix if you work methodically and record what happens when you test various things. If you cannot solve the problem on your own, please contact our support staff at `support@telestream.net` and tell them what the symptoms are, what you have tested in order to isolate the problem and what your computing environment is like.

The rest of this chapter covers how you can verify that your installation is OK, how to locate the cause of any problems and finally, how you can modify your installation if you need to do so later on.

5.1 Test transcoding

After installation you can do a test transcoding to verify that everything is OK.

Start by making sure that **Episode Engine** is running (with **System Preferences**).

Download the file `http://www.telestream.net/download-files/episode/4-2/verify.zip`. Double-click it to unpack it, creating a folder `Verify` in your download folder. `Verify` contains a source media file named `sample.mov`, four settings files named `FL8_Widescreen_1280x720.setting`, `WMV9_640x480.setting`, `H264_240x180.setting`, and `3g_64kbit_stream_meta.setting`. Finally there is a metadata definition file `sample.mov.inmeta` (read more about metadata in the User Guide). Assuming that your watch folders are in the default location, create a folder `/Users/Shared/Episode Engine/Input/Test`. Copy the settings files and the metadata file to this folder, followed by `sample.mov`.

Four output media files and a metadata file should appear in `/Users/Shared/Episode Engine/Output/Test/`. If this does not happen, you have a problem in your installation and should proceed to the next section to identify the cause of the problem.

5.2 Simple things to check

The typical problem symptom is that encoding does not proceed. Often you can diagnose the problem just with the information in **Engine Admin**.

5.2.1 Are all clients up?

In a healthy system you should in the **Connected Clients** tab see **Event Action Daemon**, **Analyzer**, **Watcher**, programDynamic Watcher as well as your own **Engine Admin**. If you only see **Engine Admin**, the IP address of the controller is probably incorrect. Check the files `/usr/local/pwce/svc-ctrl/services.conf` and `/usr/local/pwce/etc/engine.conf` (in a default installation) for the controller address, correct it as needed and restart **Episode Engine**.

5.2.2 Are all nodes up?

If you are running **Episode Engine** in a multi-node cluster, the tab **Connected Nodes** shows the controller and all encoding nodes. If an encoding node is not visible in the list, log in on it and check the files `/usr/local/pwce/svc-encoder/services.conf` and `/usr/local/pwce/etc/engine.conf` that they have the correct address to the controller. If not, correct as needed and restart **Episode Engine** on the encoder.

5.3 Troubleshooting

Some common problems are covered here. If the described symptoms do not match your problem, or the proposed solutions do not solve your problem, contact our helpdesk at `support@telestream.net`.

5.3.1 The controller node shuts down immediately after start

1. Check that you have a valid license file. If you have graphical access to the controller node (by sitting directly by it or through **Remote Desktop**), you easily see this in **System Preferences**. If you don't have a GUI available, check that the file `/usr/local/pwce/etc/licenses.xml` exists and has read permission set for the `pwce` user (assuming installation in the default directory). If the license file looks OK, continue with the next test:

2. Run the start script in verbose mode with **`/usr/local/pwce/script/enginectl -v start`**. The output from a healthy system looks like the example below:

   ```
   enginectl: Episode Engine started
   Telestream Process Monitor (Sep 17 2009)
   starting service: 'pwdwatch:run', pid = 11782
   starting service: 'pwevent:run', pid = 11783
   starting service: 'pwwatch:run', pid = 11784
   starting service: 'queen:run', pid = 11785
   starting service: 'vinculum:run', pid = 11786
   ```

   If there are problems, this will be reflected in the output. In the example output below the **queen** process fails and all other services fail in consequence. The exitcode is generated by the shell (**bash** in this case)—`126` implies that

the script is present but cannot be executed, so checking the permissions of the file would be a good idea.

```
Telestream Process Monitor (Sep 17 2009)
starting service: 'pwdwatch:run', pid = 11808
starting service: 'pwevent:run', pid = 11809
starting service: 'pwwatch:run', pid = 11810
starting service: 'queen:run', pid = 11811
starting service: 'vinculum:run', pid = 11812
service terminated: 'queen:run' pid = 11811, status = 126
service exited with exit code. 'queen:run' pid = 11811, exitcode = 126
starting service: 'queen:run', pid = 11820
enginectl: Episode Engine started
service terminated: 'queen:run' pid = 11820, status = 126
service exited with exit code. 'queen:run' pid = 11820, exitcode = 126
error: service 'queen:run' reached signal limit and has been marked as dead
terminating services..
service 'pwwatch' has terminated
service 'pwevent' has terminated
service 'vinculum' has terminated
service 'pwdwatch' has terminated
```

If the **enginectl** output does not help, proceed to the next test:

3. The `/usr/local/pwce/svc-ctrl/` directory contains several subdirectories, one for each service run on the node. Each directory contains a script named `run`. On startup of **Episode Engine**, **enginectl** runs the script **pwmon** which in turn scans the service directories and runs each `run` script. To get debug output from a script, create a file called `log` in the same directory. The `log` file should contain a single line containing the path to a file where the debug output will be written, e g, `/tmp/queen.log`.

5.3.2 An encoder node shuts down immediately after start

1. Run the start script in verbose mode with **`/usr/local/pwce/script/enginectl -v start`**. The output from a healthy system looks like the example below:

```
enginectl: Episode Engine slave started
Telestream Process Monitor (Sep 17 2009)
starting service: 'vinculum:run', pid = 12573
```

In the example below the svc-encoder directory is unreadable and process startup fails:

```
Telestream Process Monitor (Sep 17 2009)
Permission denied : failed processing using readdir
enginectl: Episode Engine slave started
Permission denied : failed processing using readdir
```

If the **enginectl** output does not help, proceed to the next test:

2. The `/usr/local/pwce/svc-encoder/` directory contains several subdirectories, one for each service run on the node (normally only **vinculum**). Each directory contains a script named `run`. On startup of **Episode Engine**, **enginectl** runs the script **pwmon** which in turn scans the service directories

and runs each `run` script. To get debug output from a script, create a file called `log` in the same directory. The `log` file should contain a single line containing the path to a file where the debug output will be written, e g, `/tmp/queen.log`.

5.3.3 The encoder nodes cannot connect to the controller node

1. Check that the file `/usr/local/pwce/svc-encoder/services.conf` on the controller node specifies the correct IP address to the controller node.
2. See if the log file `/var/log/pwce/pwce.log` contains any useful information.
3. The `/usr/local/pwce/svc-ctrl/` directory contains several subdirectories, one for each service run on the node. Each directory contains a script named `run`. On startup of **Episode Engine**, **enginectl** runs the script **pwmon** which in turn scans the service directories and runs each `run` script. To get debug output from a script, create a file called `log` in the same directory. The `log` file should contain a single line containing the name of a file where the debug output will be written.

5.3.4 All jobs fail

- Check that the settings file works in **Episode Encoder**.
- Check that the watch folders are mounted correctly and are accessible from the encoder nodes.
- Check that root processes have root permissions in the mounted folder. If you are mounting directories over NFS the `/etc/exports` file should have the directive **`-maproot=0`** for the mounted directories.
- Check if you are attempting to use symbolic links over the network. For them to work you need to set up your file system according to the example in section A.5, *Enabling symbolic links*.
- Check that the pwwatch process has 755 permissions set. If you have graphical access to the controller node (through **Remote Desktop**) you can verify and repair that in **System Preferences**.

  In a terminal, **`ls -l /usr/local/pwce/bin/pwwatch`** should return `-rwxr-xr-x 1 root wheel` *`etc`*. If not, correct the settings with the command **`sudo chmod 755 /usr/local/pwce/bin/pwwatch`**

5.3.5 The controller node has a local IP address/hostname that differs from how the encoder nodes will reach it

Assuming default installation, the file `/usr/local/pwce/svc-encoder/services.conf` is a symbolic link to `/usr/local/pwce/svc-ctrl/services.conf`. Delete the link, copy `svc-ctrl/services.conf` to `svc-encoder/` and fill in the correct controller node info there.

5.4 Frequently asked questions

5.4.1 How do I change the installation directory?

If you want to reinstall **Episode Engine** in a different directory, but you have added scripts or made other changes to the configuration that you do not want lost, you can do as follows:

1. Stop **Episode Engine**.
2. Move the installation directory to the desired point.
3. Make sure that the user **`pwce`** has read, write and execute access to the new directory.
4. Edit the `etc/engine.conf`, changing all references to the installation directory to the new directory.
5. Restart **Episode Engine**.

5.4.2 How do I disable encoding on the controller node?

Turn off encoding in the **Encode** subtab of the **System** tab of **Episode Engine** in **System Preferences**.

5.4.3 How do I run the **pwwatch** process on the node with the watch folders?

If you have a setup where the watch folders are stored on a file server separate from both controller and encoding nodes, it is a good idea to run the **pwwatch** on the file server as it will discover files added to the watch folders faster than if it runs on the controller.

Start by disabling **pwwatch** on the controller, either by deleting the `svc-ctrl/pwwatch` directory or renaming the `run` script in it. Restart **Episode Engine** on the controller. Then perform a controller node installation on the file server according to the instructions in section 3.3, *Controller installation*. You do not have to worry about a license, but make sure that you use the same path to the watch folders as all other nodes do.

You have to remove all services except **pwwatch**, so delete the directories `svc-ctrl/pwdwatch`, `svc-ctrl/pwevent`, `svc-ctrl/queen`, and `svc-ctrl/vinculum` on the file server. Then start **pwwatch** on the file server with the command **`/usr/local/pwce/script/enginectl start`**. It should connect to the controller node and start submitting jobs as soon as any file is added to a watch folder.

Note that since you have a separate installation on the file server, configuration changes on the controller will have to be manually duplicated on the file server.

5.4.4 How do I run event action scripts on a dedicated node?

If your event-action scripts do a lot of processing this may slow down encoding, so you may prefer to use a dedicated node for script execution. This is essentially similar to the problem in section 5.4.3, *How do I run the **pwwatch** process on the node with the watch folders?*, but instead of **pwwatch**, you set up for execution of the **pwevent** process on the script execution node.

5.4.5 Why is **Engine Admin** so slow to start up?

On startup **Engine Admin** receives the job history from **queen**. If you submit very many jobs per unit of time the job history will be long and it will take **Engine Admin** some time to read this list. **queen** in turn keeps a record of the job history in the spool directory.

If you feel the time and space required to keep track of job history is excessive you can edit the `engine.conf` file and set the history−expiration−time to a smaller value (see appendix B, *`engine.conf`*).

5.5 Useful commands

There are a number of common operations which are most easily performed from the command line and we will look at them here. Most require you to run them as **`root`**, so either perform a **`su`** first or imagine a **`sudo`** in front of each command. The examples are shown in the **bash** shell.

5.5.1 Define a remote mount point for NFS mount requests

Assuming you have a directory `/my/shared/folder` and want to export it via NFS, you need to add the following line to `/etc/exports` (creating the file if it does not exist):

```
/my/shared/folder -maproot=0
```

To start exporting the directory, start by making sure at least one **nfsd-server** is running.

```
prompt> ps -aux | grep nfsd-server
```

If no **nfsd-server** was running, start some up

```
prompt> nfsd -t -u -n 6
```

Then, check in a similar manner to see if any **mountd** process is running. If so, restart it.

```
prompt> killall -HUP mountd
```

If not, start up a process.

```
prompt> mountd
```

You can verify that the intended directory is exported.

```
prompt> showmount -e
```

5.5.2 NFS-mounting a remote directory

Assume the directory in the previous example is exported from file server called `server.comp.com` and that you wish to mount it on another machine.

Either of

```
prompt> mount server.comp.com:/my/shared/path /my/shared/path
```

or

```
prompt> mount_nfs server.comp.com:/my/shared/path /my/shared/path
```

will perform the mounting. Note that the remote volume will not be mounted on reboot. For this you need to set up automount as shown in the next example.

You can verify that the intended volume is mounted.

```
prompt> mount
```

5.5.3 Automounting a volume

This example sets up automounting of the default engine root folder from a file server with the IP address `10.1.1.1`.

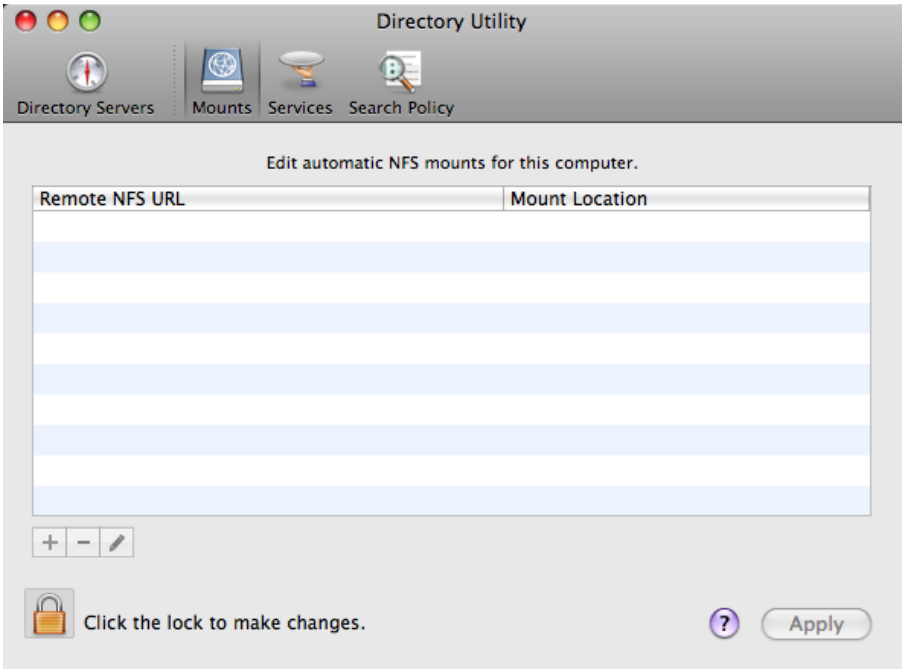
Automounting procedures are different between OS X 10.4 and 10.5, we give both versions.

10.4 version OS X 10.4 uses the **niutil** command line application to handle automounts.

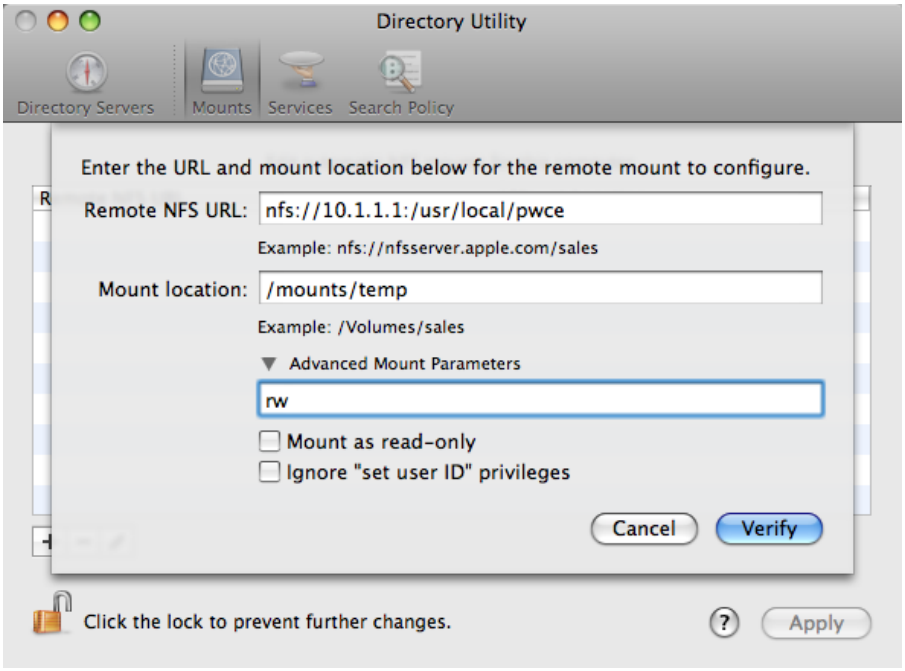
```
prompt> niutil -create . /mounts
prompt> niutil -create . /mounts/temp
prompt> niutil -createprop . /mounts/temp type "nfs"
prompt> niutil -createprop . /mounts/temp dir "/usr/local/pwce"
prompt> niutil -createprop . /mounts/temp opts "rw"
prompt> niutil -createprop . /mounts/temp name "10.1.1.1:/usr/local/pwce"
```

10.5 version Under OS X 10.5 you can use the **dscl** command line application, but the **Directory Utility** application may be more convenient.

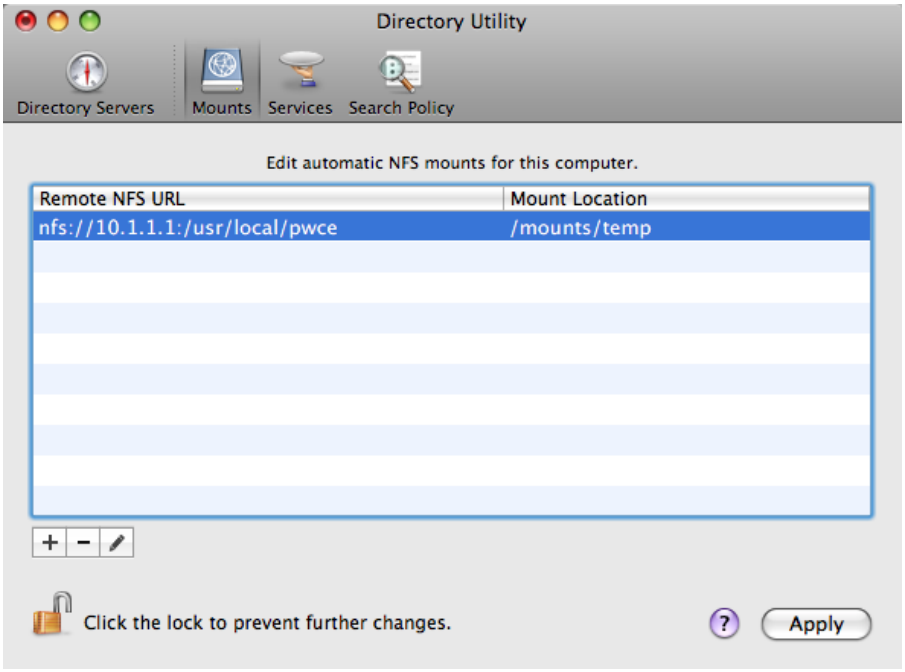
Select the **Mounts** tab.



Press **+** to add an automount record. Fill in the appropriate values on the form sheet.



Directory Utility will check that your input is correct and if that is the case, will display the new values.



5.5.4 Unpacking archives inside packages

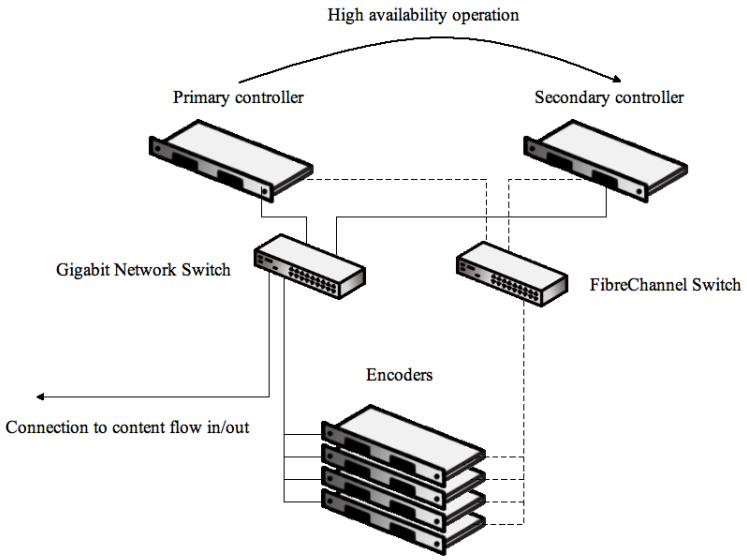
`.pkg` files are in fact directory structures and contained files can be accessed individually. You can use **ls** to peer inside packages. If a package `X.pkg` contains a **pax** archive called `Archive.pax.gz`, the contents of that archive can be unpacked with

```
prompt> pax -zrf X.pkg/Contents/Archive.pax.gz
```

6 High availability operation

Episode Engine Pro

If you require uninterrupted operation you can augment your cluster with **Episode Engine High Availability Option**. With this option you set up two controllers to run in parallel and monitor each other. If the primary controller fails, the secondary controller takes over, letting encoding continue without interruption while you take action to repair or replace the failed server. When the primary server is up again you can reinstate the primary controller as the running controller.



6.1 Installation

Episode Engine High Availability Option is delivered on CD-ROM or over the Internet. The distribution includes a license file enabling the functionality.

6.1.1 **Episode Engine** installation

If you are just now setting up your **Episode Engine** cluster, you have to first install the base **Episode Engine** software before installing **Episode Engine High Availability Option**.

If you are upgrading from a previous version of **Episode Engine High Availability Option**, we strongly recommend that you uninstall the old version first (see section 6.5, *Uninstallation*). Your old settings will be cached, so that you can reuse them when reinstalling.

Episode Engine High Availability Option requires external shared storage, which is not specified by the default installation of **Episode Engine**, so if you currently have a configuration without shared storage, you must uninstall **Episode Engine** and reinstall it configured for external shared storage.

Start by creating and exporting suitable directories on your external storage. We recommend Xsan for shared storage, so in the following examples we will assume that your Xsan installation exports the unit `/Xsan`.

Then run the **Episode Engine** installation process, first on the primary server, then on the secondary server. You can use either the graphical installer or the command-line installer, but in the example below we will use the command-line installer.

```
Episode Engine INSTALLER
(c) Copyright 2009 Telestream Inc.


This is the Episode Engine installation.
You will be guided through the steps necessary to install this software.
Thank you for purchasing Episode Engine.

Press any key to review the license agreement, then press space,
or use the arrow keys, to scroll the text.

LICENSE AGREEMENT

Please take a moment to configure your Episode Engine installation.

NOTE: Default values, or the capitalized letter, within brackets are selected
by pressing ENTER at the prompts.

INSTALLATION PATH

  If the path does not exist, it will be created.
  Installation path [/usr/local/pwce]: /Xsan/EpisodeEngine/engine/bin
```

The installer will create any necessary subdirectories, so we can give installation paths below `/Xsan`.

```
USER & GROUP

  If the user and the group does not exist,
  they will be created later during the installation.
  Episode Engine user [jrn]: jrn
  Episode Engine group [staff]: staff


NODE TYPE

  1. Controller node
     The Controller node, previously called the Master node, is responsible for
     job scheduling. The Controller node may also, at your choice, be used as
     an Encoder node.
```

```
      NOTE: If you want to perform a stand alone installation,
      choose 1 for Controller. Also make sure to answer yes to the question
      later about using the Controller for encoding.

  2. Encoder
     The Encoder, previously called the Slave node, performs all encoding
     operations delegated by the Controller node.

Select a node type for this machine [1]: 1

Is the above correct [Y/n]? Yes

Running preinstall script.. ....

CONTROLLER
  Do you want to use the Controller node for encoding [Y/n]? Yes
```

Even if you are running a cluster, you can use your controller node for encoding if it fulfils the requirements in chapter 2, *Prerequisites*.

```
EPISODE ENGINE PATHS
  Installation path is set to /Xsan/EpisodeEngine/engine/bin.
  File-log path [/var/log/pwce]: /Xsan/EpisodeEngine/engine/log
  Temporary path [/tmp]: /Xsan/EpisodeEngine/engine/tmp
  Spool path [/var/spool/pwce]: /Xsan/EpisodeEngine/engine/spool

  Is the above correct [Y/n]? Yes

WATCH FOLDER SETUP
  Watch folder root path [/Xsan/EpisodeEngine]: /Xsan/EpisodeEngine/Media
  Input watch folder name [Input]: Input
  Output watch folder name [Output]: Output
  Archive folder name [Archive]: Archive
  Enable input material archive [y/N]? Yes

  Is the above correct [Y/n]? Yes

NFS-EXPORTING FROM THIS NODE
  NOTE: You must export the watch folders and the binaries from this machine if
  you are installing a Cluster and you have not prepared a shared storage as
  described in the manual.

  Do you want to NFS-export the watch folders from this node [y/N]? No
  Do you want to NFS-export the binaries from this node [y/N]? No

  Is the above correct [Y/n]? Yes

COMMUNICATION
  Controller IP address [10.50.5.1]: 10.50.5.1

  Is the above correct [Y/n]? Yes

PERMISSIONS AND SECURITY
  Engine Admin password [anonymous]: ennnsynen

  Is the above correct [Y/n]? Yes

Updating intermediate settings file..
Running postinstall script..
Please wait while installer is copying files.......
```

```
NOTE: Your old configuration files were backed up and can be found in /usr/local/pwce/

LICENSE
  No Episode Engine license is installed. Do you want to install a license now [Y/n]?
  Full path to license file type 'skip' to skip: /Users/jrn/Desktop/ha-license.xml
License successfully installed.

  Do you want to start Episode Engine now [Y/n]? Yes
Episode Engine started.

NOTE: To make additional configuration of your Episode Engine installation
please open the Episode Engine preference pane which has been installed in
System Preferences. Here you can also manage your Episode Engine licenses.

Episode Engine successfully installed.
```

Note that if you have a separate **Episode Engine High Availability Option** licence, you will get an error message if you attempt to start **Episode Engine** before you have installed the **Episode Engine High Availability Option** licence. You should therefore instead open **System Preferences** and add the **Episode Engine High Availability Option** licence to the license pane as explained in section 3.6, *Installing a license*.

When **Episode Engine** is running, start **Engine Admin** and check that you can connect to the server. If everything goes well, stop **Episode Engine** in **System Preferences**.

Do the same installation and test on the secondary server. Note that the secondary server shall also be configured as a controller node. You will get warnings that the directories already exist, this is not a problem. You will also not need to install a license, since that already has been done.

6.1.2 **Episode Engine High Availability Option** installation

Once you have made sure that the **Episode Engine** installation works well on both the primary and secondary server, proceed to install **Episode Engine High Availability Option** on both nodes.

Before installing you need to decide on the following:

- Primary and secondary server Ethernet addresses. In the example below we use 10.50.5.1 and 10.50.5.2, respectively. These must be static so that they do not change upon reboot, reconnection to the network, etc. If your servers have multiple network interfaces you must make sure that you use the interfaces that are on the same subnet.
- A virtual IP address to point to the currently active controller. This is the address that encoder nodes and clients such as **Engine Admin** must connect to. In the example below we use 10.50.5.100.
- An administrator notification email address. In the example below we use `admin@yourcompany.com`.

On the primary server

The software disk image contains the script `Install High Availability.command`. Double-click it to run the installation script as shown below.

```
prompt> sudo ./INSTALLER install

LICENSE AGREEMENT

EPISODE ENGINE FAILOVER MODULE INSTALLATION SCRIPT
                        (c) Copyright 2009 Telestream Inc.

Which role is this server having [primary/secondary]? : primary
```

Note that you have to type **`primary`** and **`secondary`** in full.

```
FILES AND DIRECTORIES
  Failover module installation dir [/usr/local/pwha]: <ENTER>
  Is the above correct [Y/n]?
```

Note that the software is to be stored locally and not on the shared storage.

```
NETWORK CONFIGURATION
- Ethernet configuration
  Primary server ethernet IP address  : 10.50.5.1
    Available Ethernet interfaces on system:
        en0
        en1
  Primary server Ethernet interface   : en0
    Ethernet netmask is 255.255.255.0
    Default gateway set to 10.50.5.1
  Secondary server ethernet IP address: 10.50.5.2
  Virtual IP address                  : 10.50.5.100
  Is the above correct [Y/n]? <ENTER>

NOTIFICATIONS
- Email recipients
  Administrator e-mail address        : admin@yourcompany.com
  Is the above correct [Y/n]? <ENTER>

INSTALLATION PROCESS
  Installing primary failover server.........done.
  Making primary failover server master....done.

Primary failover server successfully installed and configured.
```

The script copies and modifies all the required files. It also starts all the needed services to put the primary server in the controller role when installation is complete. Verify the state of the primary server by opening the **System Preferences** panel for **Episode Engine** and selecting the **Failover** tab.

NOTE

If you are re-installing the software, your earlier settings will have been cached and you will be given the option to reuse them when you start the installation script.

On the secondary server

The installation process is very similar to the one conducted at the primary server.

```
prompt> sudo ./INSTALLER install
EPISODE ENGINE FAILOVER MODULE INSTALLATION SCRIPT
                        (c) Copyright 2009 Telestream Inc.

Which role is this server having [primary/secondary]? : secondary

FILES AND DIRECTORIES
  Failover module installation dir [/usr/local/pwha]: <ENTER>
  Is the above correct [Y/n]?  <ENTER>

REMOTE CONFIGURATION SETUP
  Primary server ethernet IP address  : 10.50.5.1
  Is the above correct [Y/n]?  <ENTER>

Fetching configuration from primary server..done.

    Virtual IP Address is 10.50.5.100
    Primary server IP address is 10.50.5.1
    Secondary server IP address is 10.50.5.2

    Secondary server Ethernet interface is en0
    Ethernet netmask is 255.255.255.0
    Default gateway set to 10.50.5.1

INSTALLATION PROCESS
  Installing secondary failover server.............done.

Secondary failover server successfully installed and configured.
```

As seen above the script fetches configuration information from the primary server. The script copies and modifies all required files. It also starts all the needed services to put the secondary server in the standby role when installation is complete.

Once installation is complete on both servers, you need to restart the clients on all encoding nodes. Log in on each client and execute

```
prompt> sudo /usr/local/pwce/script/enginectl restart
```

Now open the **Episode Engine System Preferences** pane on one of the servers. Select the **Failover** tab to check the functions.

TIP

You may wish to use the secondary failover server as an encoder as well. This requires the **vinculum** process to run on the secondary node. Given the same virtual IP address as above and default install path, execute on the secondary node

```
prompt> launchctl submit -l encoder -- /usr/local/pwce/bin/vinculum 10.50.5.100
```

This will also keep the **vinculum** process alive if you reboot the node.

6.2 System Preferences

You control the operation of the servers from the **System Preferences**. The **Episode Engine** pane contains a **Failover** tab which at the top shows the state of the servers. A star (☆) indicates which of the servers currently is the controller. On the primary server you can use the **Switch master to** button to manually switch to the other server. Note that if the primary server has gone down and been restarted, you have to do a manual switch-over to return control to the primary server. You can also do this switch-over on the command line: run **`/usr/local/pwha/script/failover release`** on the primary server to relinquish control to the secondary; run **`/usr/local/pwha/script/failover acquire`** on the primary server to return control to it.

The pulse icon ( ) shows that the server is running. If the server goes down the icon will change to a red pulse ( ).

The icons in the fields **Primary server status** and **Secondary server status** are shown in grey to indicate that the following conditions hold:

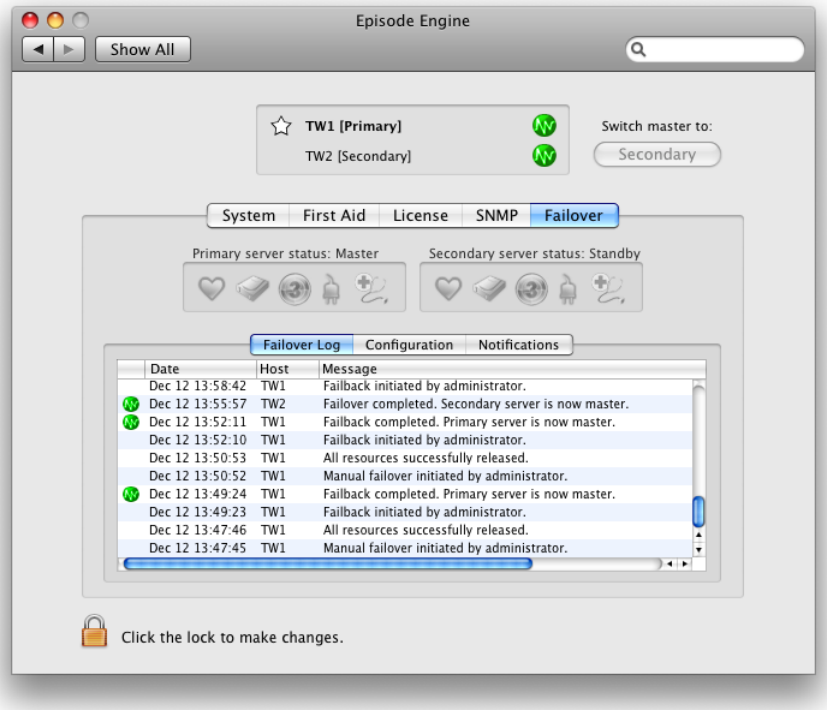
- The heartbeat process on the primary server and the heartbeat receiver process on the secondary server, respectively, are running.
- The shared storage is readable.
- Only one server (normally the primary) is running the **Episode Engine** processes.
- The preference pane has contact with its server.

The **pwhad** process is running.

If a given condition does not hold, its corresponding icon will turn red.

Below the status area are three additional tabs for configuration information:

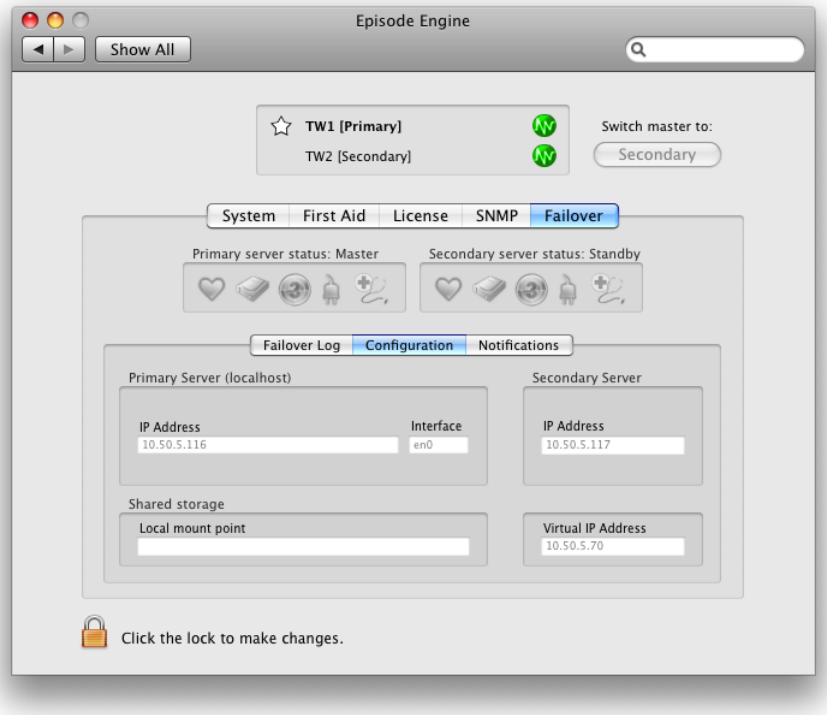
6.2.1 Failover log



The **Failover log** tab shows a running log of events in the servers. Events related to the status checks above will be marked with similar icons.

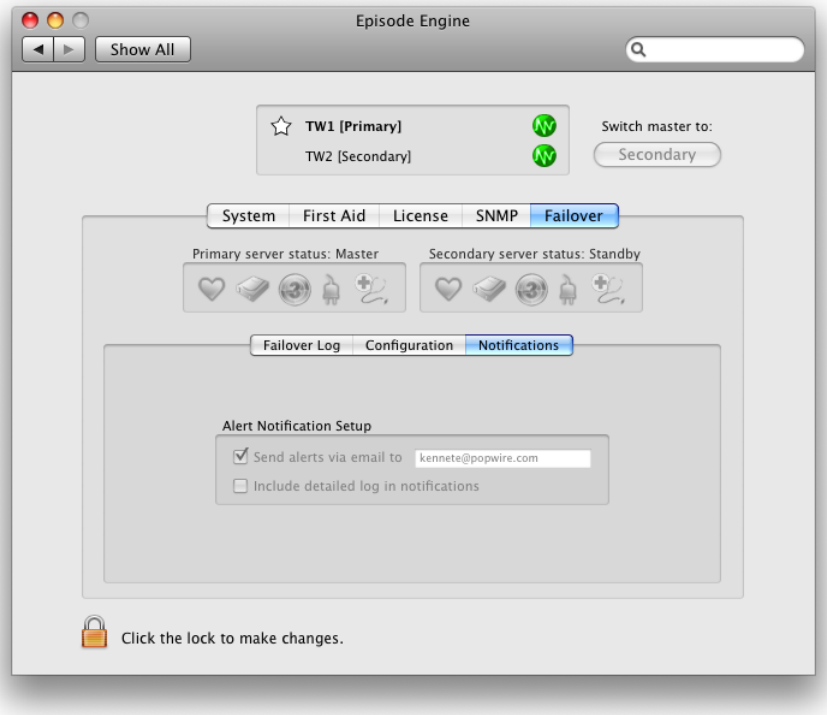
The log messages will also be written to the files `/usr/local/pwha/log/event.log` and `/usr/local/pwha/log/pwha.log` (default paths) on both servers.

6.2.2 Configuration



The **Configuration** tab lets you check and configure the network settings of the servers and the shared storage setup.

6.2.3 Notifications

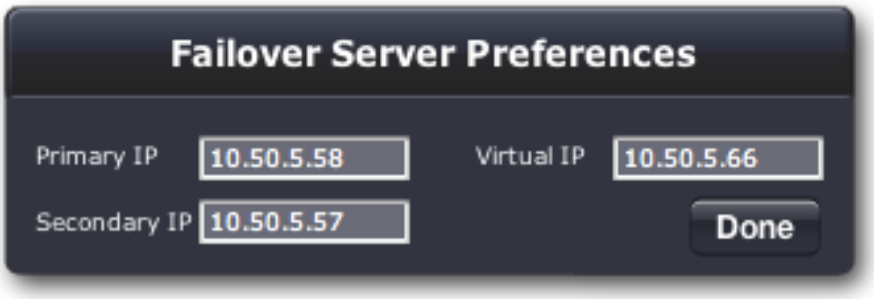


In the **Notifications** tab you can adjust the sending of alerts. You have three levels of notifications: none, simple alerts by email to a given email address, and email alerts with a copy of the Failover log in the message.

6.3 Status widget

In the distribution you will also find `FailoverStatus.wdgt`, which you can add to your **Dashboard** to show the status of the servers.

Clicking on the round button at lower right turns the widget over so that you can enter the addresses of the servers.



6.4 Processes

Episode Engine High Availability Option will add processes on both controller and encoder to those created by **Episode Engine Pro**: **heartbeatd** runs on the primary server and **failoverd** on the secondary server. If the secondary server no longer receives heartbeats it will take over as controller and optionally send a notification to the administrator as explained in section 6.2.3, *Notifications*. **pwhad** keeps the status information in the **System Preferences** updated.

6.5 Uninstallation

If you have to uninstall **Episode Engine**, you should first uninstall **Episode Engine High Availability Option**. You do this by running the **`INSTALLER`** script, but with the argument **`remove`**:

```
prompt> sudo ./INSTALLER remove
EPISODE ENGINE FAILOVER MODULE REMOVAL SCRIPT
                          (c) Copyright 2009 Telestream Inc.

Primary failover server installation found in /usr/local/pwha.

Uninstalling..done.

Uninstallation successfully completed.
```

Appendix A Example configurations

You can set up a **Episode Engine** cluster in many different ways. Below are shown a number of example configurations that you can use as a guide for your own installation. Each will first describe how directories are laid out over the cluster nodes and then go through the commands necessary to achieve that configuration. We will use nodenames like `controller.comp.com` and `encoder.comp.com` in the installation examples.

A.1 Controller node exports its local disk through NFS

In the default configuration the controller node exports its physical disk through NFS to the encoder nodes.

A.1.1 Controller node

We assume that the software is installed in the default directories.

Watch folders

The input watch folders are located in `/Users/Shared/Episode Engine/Input/`, the output folders are placed in `/Users/Shared/Episode Engine/Output/`.

Installation directories

```
/usr/local/pwce
├── bin
├── component
├── etc
├── evt
├── lib
├── script
├── svc-ctrl
└── svc-encoder
```

NFS export

The watch folders and software root directory are exported through NFS. `/etc/exports` contains the lines

```
/Users/Shared/Episode\ Engine -maproot=0
/usr/local/pwce -maproot=0
```

Other installed files and directories

Preference pane `/Library/PreferencePanes/Episode Engine.prefPane`

Preferences file `/Library/Preferences/com.popwire.preferences.engine.plist`

Startup item `/Library/StartupItems/EpisodeEngine`
- `EpisodeEngine`
- `StartupParameters.plist`

Machine start variable To indicate that **Episode Engine** should be started on "machine start" the file `/etc/hostconfig` should contain the line

```
PWCESERVER=-YES-
```

Temporary file directory `/tmp/`

Spool directory `/var/spool/pwce/`

Log directory `/var/log/pwce/`

Processes running on startup

pwmon Monitors all other processes or "services" that it starts from the `svc` directory.

queen Controls the system and schedules jobs.

pwwatch Monitors watch folders, synchronises dependency files and creates jobs.

pwdwatch Dynamically creates monitors for any acquisition plugins.

pwevent Listens to events sent by **queen** and executes corresponding user defined scripts.

If you choose to use controller node as encoder as well, the following will also run here.

vinculum Receives and processes encoding requests and starts one or more sub-processes to do the actual encoding work (called "drones").

A.1.2 Encoder nodes

The encoder nodes all share the same configuration.

Watch folder mount point `/Users/Shared/Episode Engine`

Installation root mount point `/usr/local/pwce/`

Preferences file `/Library/Preferences/com.popwire.preferences.engine.plist`

Startup item `/Library/StartupItems/EpisodeEngine`

```
EpisodeEngine
StartupParameters.plist
```

Machine start variable To indicate that **Episode Engine** should be started on "machine start" the file `/etc/hostconfig` should contain the line

```
PWCESERVER=-YES-
```

Processes running on startup

vinculum Receives and processes encoding requests and starts one or more subprocesses to do the actual encoding work (called "drones").

A.1.3 Installation

The installation instructions below assume that you are running on a diffent machine than the one intended to be the controller node and have to use the command line interface.
Log in on the controller and copy the installation package and the license file to it.

```
home> ssh username@controller.comp.com
controller> cd ~/Desktop
controller> scp -r username@home.comp.com:~/Desktop/Episode\ Engine.pkg .
controller scp username@home.comp.com:~/Desktop/licenses.xml .
```

Run the installation inside the package and answer the questions.

```
controller> sudo ./Episode\ Engine.pkg/Contents/Resources/install.command
```

If you did not give the location of the license file to the installation script, you can copy it to the correct destination manually.

```
controller> sudo cp licenses.xml /usr/local/pwce/etc
```

This completes the installation. Now, start **Episode Engine**.

```
controller> sudo /usr/local/pwce/script/enginectl start
```

If any of the processes report an error and quit, all processes are killed. So give the processes a few seconds to start and then verify their health.

```
controller> sudo /usr/local/pwce/script/enginectl status
```

Now, log out from the controller and continue the installation on all encoder nodes.

```
home> ssh username@encoder.comp.com
encoder> cd ~/Desktop
encoder> scp -r username@home.comp.com:~/Desktop/Episode\ Engine.pkg .
```

While you are recommended to let the installation script do it for you, you can create the necessary mount points before installation.

```
encoder> mkdir /Users/Shared/Episode\ Engine
encoder> mkdir -p /usr/local/pwce
```

Run the installation script

```
encoder> sudo ./Episode\ Engine.pkg/Contents/Resources/install.command
```

Either start the processes from the installation script, or start them manually, just as for the controller.

A.2 Watch folders on a file server, software on the controller

A dedicated machine with a large disk serves as shared storage but the software is installed on the controller node, which accordingly has to export this directory.

A.2.1 File server

Watch folders

The input watch folders are located in `/Users/Shared/Episode Engine/Input/`, the output folders are placed in `/Users/Shared/Episode Engine/Output/`.

NFS export

The watch folders are exported through NFS. `/etc/exports` contains the line

```
/Users/Shared/Episode\ Engine -maproot=0
```

A.2.2 Controller node

Watch folder mount point

```
/Users/Shared/Episode Engine/
```

Installation directories

```
/usr/local/pwce
├── bin
├── component
├── etc
├── evt
├── lib
├── script
├── svc-ctrl
└── svc-encoder
```

NFS export

The software root directory is exported through NFS. `/etc/exports` contains the line

```
/usr/local/pwce -maproot=0
```

Other installed files and directories

Preference pane `/Library/PreferencePanes/Episode Engine.prefPane`

Preferences file `/Library/Preferences/com.popwire.preferences.engine.plist`

Startup item `/Library/StartupItems/EpisodeEngine`
- `EpisodeEngine`
- `StartupParameters.plist`

Machine start variable To indicate that **Episode Engine** should be started on "machine start" the file `/etc/hostconfig` should contain the line

```
PWCESERVER=-YES-
```

Temporary file directory `/tmp/`

Spool directory `/var/spool/pwce/`

Log directory `/var/log/pwce/`

Processes running on startup

pwmon Monitors all other processes or "services" that it starts from the `svc` directory.

queen Controls the system and schedules jobs.

pwwatch Monitors watch folders, synchronises dependency files and creates jobs.

pwdwatch Dynamically creates monitors for any acquisition plugins.

pwevent Listens to events sent by **queen** and executes corresponding user defined scripts.

If you choose to use controller node as encoder as well, the following will also run here.

vinculum Receives and processes encoding requests and starts one or more sub-processes to do the actual encoding work (called "drones").

A.2.3 Encoder nodes

The encoder nodes all share the same configuration.

Watch folder mount point `/Users/Shared/Episode Engine`

Installation root mount point `/usr/local/pwce/`

Preferences file `/Library/Preferences/com.popwire.preferences.engine.plist`

Startup item `/Library/StartupItems/EpisodeEngine`
- `EpisodeEngine`
- `StartupParameters.plist`

Machine start variable To indicate that **Episode Engine** should be started on "machine start" the file `/etc/hostconfig` should contain the line

```
PWCESERVER=-YES-
```

Processes running on startup

vinculum Receives and processes encoding requests and starts one or more subprocesses to do the actual encoding work (called "drones").

A.2.4 Installation

Start by logging in on the file server.

```
home> ssh username@server.comp.com
```

Create the root watch folder. (The input/output folders will be created by the installation script.)

```
server> mkdir /Users/Shared/Episode\ Engine
```

The next step is to set up NFS export of the root watch folder. Edit the file `/etc/exports` to add the line

```
/Users/Shared/Episode\ Engine -maproot=0
```

This line defines a remote mount point for NFS mount requests. The **`-maproot=0`** directive ensures that root processes may act as root on the mount, which is necessary since the **pwwatch** and **drone** processes have to be able to do anything necessary to the files used by **Episode Engine**.
To start exporting the directory, follow the instructions in section 5.5.1, *Define a remote mount point for NFS mount requests* or just reboot the file server.
Next you need to set up things on the controller node. As before, we assume that we are restricted to the command line interface.

```
home> ssh username@controller.comp.com
controller> cd ~/Desktop
controller> scp -r username@home.comp.com:~/Desktop/Episode\ Engine.pkg .
controller> scp username@home.comp.com:~/Desktop/licenses.xml .
```

Create the root watch folder so you can mount it before installation.

```
controller> mkdir /Users/Shared/Episode\ Engine
```

Then mount the watch folder root directory so the installation script can create the actual input and output folders.

```
controller> sudo mount server.comp.com:/Users/Shared/Episode\ Engine /Users/Shared/Epi
```

When running the installation script, you need to indicate that the controller node is exporting only the software directories.

```
controller> sudo ./Episode\ Engine.pkg/Contents/Resources/install.command
...
NFS-EXPORTING FROM THIS NODE
  NOTE: You must export the watch folders and the binaries from this machine if
  you are installing a Cluster and you have not prepared a shared storage as
  described in the manual.

  Do you want to NFS-export the watch folders from this node [y/N]? No
  Do you want to NFS-export the binaries from this node [y/N]? Yes

  Is the above correct [Y/n]? Yes
...
```

If you did not give the location of the license file to the installation script, you can copy it to the correct destination manually.

```
controller> sudo cp licenses.xml /usr/local/pwce/etc
```

This completes the installation. Now, start **Episode Engine**.

```
controller> sudo /usr/local/pwce/script/enginectl start
```

If any of the processes report an error and quit, all processes are killed. So give the processes a few seconds to start and then verify their health.

```
controller> sudo /usr/local/pwce/script/enginectl status
```

Now, log out from the controller and continue the installation on all encoder nodes.

```
home> ssh username@encoder.comp.com
encoder> cd ~/Desktop
encoder> scp -r username@home.comp.com:~/Desktop/Episode\ Engine.pkg .
```

While you are recommended to let the installation script do it for you, you can create the necessary mount points before installation.

```
encoder> mkdir /Users/Shared/Episode\ Engine
encoder> mkdir -p /usr/local/pwce
```

Run the installation script

```
encoder> sudo ./Episode\ Engine.pkg/Contents/Resources/install.command
```

Either start the processes from the installation script, or start them manually, just as for the controller.

A.3 Watch folders and software on a file server

A file server with a large disk serves as shared storage with both watch folders and software installation.

A.3.1 File server

Watch folders

The input watch folders are located in `/PWCE/watch/Input/`, the output folders are placed in `/PWCE/watch/Output/`.

Installation directories

```
/PWCE/pwce
├── bin
├── component
├── etc
├── evt
├── lib
├── script
├── svc-ctrl
└── svc-encoder
```

NFS export

The directories are exported through NFS. `/etc/exports` contains the line

```
/PWCE -maproot=0
```

A.3.2 Controller node

Watch folder mount point

`/PWCE/`

Other installed files and directories

Preference pane `/Library/PreferencePanes/Episode Engine.prefPane`

Preferences file `/Library/Preferences/com.popwire.preferences.engine.plist`

Startup item `/Library/StartupItems/EpisodeEngine`

```
EpisodeEngine
StartupParameters.plist
```

Machine start variable To indicate that **Episode Engine** should be started on "machine start" the file `/etc/hostconfig` should contain the line

```
PWCESERVER=-YES-
```

Temporary file directory `/tmp/`

Spool directory `/var/spool/pwce/`

Log directory `/var/log/pwce/`

Processes running on startup

pwmon Monitors all other processes or "services" that it starts from the `svc` directory.

queen Controls the system and schedules jobs.

pwwatch Monitors watch folders, synchronises dependency files and creates jobs.

pwdwatch Dynamically creates monitors for any acquisition plugins.

pwevent Listens to events sent by **queen** and executes corresponding user defined scripts.

If you choose to use controller node as encoder as well, the following will also run here.

vinculum Receives and processes encoding requests and starts one or more sub-processes to do the actual encoding work (called "drones").

A.3.3 Encoder nodes

The encoder, or encoder, nodes all share the same configuration.

Watch folder mount point `/PWCE`

Installation root mount point `/PWCE`

Preferences file `/Library/Preferences/com.popwire.preferences.engine.plist`

Startup item `/Library/StartupItems/EpisodeEngine`

```
EpisodeEngine
StartupParameters.plist
```

Machine start variable To indicate that **Episode Engine** should be started on "machine start" the file `/etc/hostconfig` should contain the line

```
PWCESERVER=-YES-
```

Processes running on startup

vinculum Receives and processes encoding requests and starts one or more subprocesses to do the actual encoding work (called "drones").

A.3.4 Installation

Start by logging in on the file server.

```
home> ssh username@server.comp.com
```

Create the installation root folder.

```
server> mkdir /PWCE
```

The next step is to set up NFS export of the installation root folder. Edit the file `/etc/exports` to add the line

```
/PWCE -maproot=0
```

This line defines a remote mount point for NFS mount requests. The **`-maproot=0`** directive ensures that root processes may act as root on the mount, which is necessary since the **pwwatch** and **drone** processes have to be able to do anything necessary to the files used by **Episode Engine**.
To start exporting the directory, follow the instructions in section 5.5.1, *Define a remote mount point for NFS mount requests* or just reboot the file server.
Next you copy all needed files to the file server.

```
server> cd ~/Desktop
server> scp -r username@home.comp.com:~/Desktop/Episode\ Engine.pkg .
server> scp username@home.comp.com:~/Desktop/licenses.xml .
```

You may wish to install the **pwwatch** program on the server node in order to increase performance as explained in section 5.4.3, *How do I run the **pwwatch** process on the node with the watch folders?*.
Next, log out from the file server and log in on the controller node in order to install the software.

```
home> ssh username@controller.comp.com
controller> sudo mkdir /PWCE
controller> sudo mount server.comp.com:/PWCE /PWCE
```

You should set up automounting of the volume, as described in section 5.5.3, *Automounting a volume*.
When running the installation script, you need change a few responses from the defaults.

```
controller> sudo ./Episode\ Engine.pkg/Contents/Resources/install.command
...
INSTALLATION PATH

  If the path does not exist, it will be created.
  Installation path [/usr/local/pwce]: /PWCE/pwce

USER & GROUP

  If the user and the group does not exist, they will be created.
  Episode Engine user [pwce]: pwce
  Episode Engine group [pwce]: pwce
...
WATCH FOLDER SETUP
  Watch folder root path [/Users/Shared/Episode Engine]: /PWCE/watch
  Input watch folder name [Input]: Input
  Output watch folder name [Output]: Output
  Archive folder name [Archive]: Archive
  Enable input material archive [y/N]? Yes

  Is the above correct [Y/n]? Yes
...
NFS-EXPORTING FROM THIS NODE
  NOTE: You must export the watch folders and the binaries from this machine if
  you are installing a Cluster and you have not prepared a shared storage as
  described in the manual.

  Do you want to NFS-export the watch folders from this node [y/N]? No
  Do you want to NFS-export the binaries from this node [y/N]? No

  Is the above correct [Y/n]? Yes
...
```

If you did not give the location of the license file to the installation script, you can copy it to the correct destination manually.

```
controller> sudo cp licenses.xml /usr/local/pwce/etc
```

This completes the installation. Now, start **Episode Engine**.

```
controller> sudo /usr/local/pwce/script/enginectl start
```

If any of the processes report an error and quit, all processes are killed. So give the processes a few seconds to start and then verify their health.

```
controller> sudo /usr/local/pwce/script/enginectl status
```

Now, log out from the controller and continue the installation on all encoder nodes. Most of the steps are the same, but using the encoder installer script instead.

```
home> ssh username@encoder.comp.com
encoder> mkdir /PWCE
encoder> mount server.comp.com:/PWCE /PWCE
```

You will need to set up automounting here too.
Run the installation script

```
encoder> sudo ./Episode\ Engine.pkg/Contents/Resources/install.command
...
INSTALLATION PATH

  If the path does not exist, it will be created.
  Installation path [/usr/local/pwce]: /PWCE
```

Either start the processes from the installation script, or start them manually, just as for the controller.

A.4 All nodes access a Storage Area Network

All nodes can mount a SAN, Xsan, XRaid or similar disk array, which looks like a local disk to all nodes.

A.4.1 Shared storage

Watch folders

The input watch folders are located in `/Volumes/shared/watch/in/`, the output folders are placed in `/Volumes/shared/watch/out/`.

Installation directories

```
/Volumes/shared/pwce
├── bin
├── component
├── etc
├── evt
├── lib
├── script
├── svc-ctrl
└── svc-encoder
```

A.4.2 Controller node

Preference pane `/Library/PreferencePanes/Episode Engine.prefPane`

Preferences file `/Library/Preferences/com.popwire.preferences.engine.plist`

Startup item `/Library/StartupItems/EpisodeEngine`

```
├── EpisodeEngine
└── StartupParameters.plist
```

Machine start variable To indicate that **Episode Engine** should be started on "machine start" the file `/etc/hostconfig` should contain the line

```
PWCESERVER=-YES-
```

Temporary file directory `/tmp/` (Should not be placed on the shared storage as this increases the number of accesses to the storage, thus slowing down processing of other files.)

Spool directory `/Volumes/shared/pwce/spool/` (*Must* be placed on the shared storage.)

Log directory `/var/log/pwce/` (Could be placed on the shared storage.)

Processes running on startup

pwmon Monitors all other processes or "services" that it starts from the `svc` directory.

queen Controls the system and schedules jobs.

pwwatch Monitors watch folders, synchronises dependency files and creates jobs.

pwdwatch Dynamically creates monitors for any acquisition plugins.

pwevent Listens to events sent by **queen** and executes corresponding user defined scripts.

If you choose to use controller node as encoder as well, the following will also run here.

vinculum Receives and processes encoding requests and starts one or more subprocesses to do the actual encoding work (called "drones").

A.4.3 Encoder nodes

Preferences file `/Library/Preferences/com.popwire.preferences.engine.plist`

Startup item `/Library/StartupItems/EpisodeEngine`

```
EpisodeEngine
StartupParameters.plist
```

Machine start variable To indicate that **Episode Engine** should be started on "machine start" the file `/etc/hostconfig` should contain the line

```
PWCESERVER=-YES-
```

Processes running on startup

vinculum Receives and processes encoding requests and starts one or more subprocesses to do the actual encoding work (called "drones").

A.4.4 Installation

Start by logging in on the file server and copying all needed files to the file server.

```
home> ssh username@server.comp.com
server> cd ~/Desktop
server> scp -r username@home.comp.com:~/Desktop/Episode\ Engine.pkg .
server> scp username@home.comp.com:~/Desktop/licenses.xml .
```

Now, log out from the file server and log in on the controller node in order to install the software. When running the installation script, you need change a few responses from the defaults.

```
home> ssh username@controller.comp.com
controller> sudo ./Episode\ Engine.pkg/Contents/Resources/install.command
...
INSTALLATION PATH

  If the path does not exist, it will be created.
  Installation path [/usr/local/pwce]: /Volumes/shared/pwce

USER & GROUP

  If the user and the group does not exist, they will be created.
  Episode Engine user [pwce]: pwce
  Episode Engine group [pwce]: pwce
...
WATCH FOLDER SETUP
  Watch folder root path [/Users/Shared/Episode Engine]: /Volumes/shared/watch
  Input watch folder name [Input]: Input
  Output watch folder name [Output]: Output
  Archive folder name [Archive]: Archive
  Enable input material archive [y/N]? Yes

  Is the above correct [Y/n]? Yes
...
NFS-EXPORTING FROM THIS NODE
  NOTE: You must export the watch folders and the binaries from this machine if
  you are installing a Cluster and you have not prepared a shared storage as
  described in the manual.

  Do you want to NFS-export the watch folders from this node [y/N]? No
  Do you want to NFS-export the binaries from this node [y/N]? No

  Is the above correct [Y/n]? Yes
...
```

If you did not give the location of the license file to the installation script, you can copy it to the correct destination manually.

```
controller> sudo cp licenses.xml /usr/local/pwce/etc
```

This completes the installation. Now, start **Episode Engine**.

```
controller> sudo /usr/local/pwce/script/enginectl start
```

If any of the processes report an error and quit, all processes are killed. So give the processes a few seconds to start and then verify their health.

```
controller> sudo /usr/local/pwce/script/enginectl status
```

Now, log out from the controller and continue the installation on all encoder nodes. Most of the steps are the same, but using the encoder installer script instead.

```
home> ssh username@encoder.comp.com
encoder> sudo ./Episode\ Engine.pkg/Contents/Resources/install.command
...
INSTALLATION PATH

  If the path does not exist, it will be created.
  Installation path [/usr/local/pwce]: /Volumes/shared/pwce
```

Either start the processes from the installation script, or start them manually, just as for the controller.

A.5 Enabling symbolic links

You have to work a bit harder to be able to use symbolic links across a network. This is important, e g, when saving QuickTime reference files from **Final Cut Pro** into a watch folder on another machine.
Assume that we have a setup similar to that in section A.1, *Controller node exports its local disk through NFS* and a **Final Cut Pro** user on a separate machine. All the other setup is equal to the earlier example, but the FCP user has to export the source material in the directory `/Users/Shared/source/fcp/` by adding the line

```
/Users/Shared/source/fcp
```

to the `/etc/exports` file. Controller and encoder nodes have to mount this volume.

Appendix B `engine.conf`

`/usr/local/pwce/etc/engine.conf` contains global settings for **Episode Engine**. It is an XML-format file, so you can edit the file with any text editor of your choice, as long as you save the file as pure text.
The following example shows a typical configuration file with explanatory comments inserted.

```
<?xml version="1.0" encoding="UTF−8"?>
<!DOCTYPE sys−conf SYSTEM "sys−conf.dtd">
<sys−conf>
<!−−
        This is a generated xml file.
        Any comments added to this file will be destroyed.
−−>
  <common>
  <!−− The following clauses define the paths to various files used by Episode Engine
       . @ indicates the directory containing this file. −−>
    <app−path>@/../bin</app−path>
    <component−path>@/../component</component−path>
    <!−− In addition to the <plugin−path>, Episode Engine will look for plugins in
         /Library/Application Support/Telestream/Plugins/ and
         /Library/Application Support/Telestream/Plugins/. −−>
    <plugin−path>@/../plugin</plugin−path>
    <dtd−path>@/../etc</dtd−path>
    <license−path>@/../etc</license−path>

    <!−− The <logging> clauses set up the log files. −−>
    <logging>

      <!−− The log file size limit is given in kibibytes. −−>
      <filelog−filesize>4096</filelog−filesize>

      <!−− This is the number of log files, as explained in section 4.1.1, General. −−>
      <filelog−num−rotation>5</filelog−num−rotation>

      <!−− The log files will be stored in this directory. −−>
      <filelog−path>/var/log/pwce</filelog−path>

      <!−− This is the types of events which will be logged, from 0 (most critical
           events) to 7 (least critical events); 8 will turn off logging. See the syslog(3)
           manual page for additional information on priority levels. −−>
      <filelog−priority>6</filelog−priority>

      <!−− The messages that will be written to the system log, filtered as above. −−>
      <syslog−priority>4</syslog−priority>

    </logging>
```

```
    <!-- This is the address of the server. -->
    <public-address>127.0.0.1</public-address>

    <!-- This is the port on which a vinculum listens to messages from the queen, as
         explained in section 4.4, Starting the
         processes. -->
    <queen-vinculum-port>40401</queen-vinculum-port>

    <!-- This is where temporary files are stored. -->
    <temp-path>/tmp</temp-path>

  </common>

  <!-- The settings from here on are only relevant for clusters, but will be present in
       all configuration files. -->
  <drone>
    <!-- This is the interval in seconds with which status updates are sent to listening
         clients. "0" will turn off updates. -->
    <progress-interval>2</progress-interval>

    <!-- This sets the high- and low-water marks for the number of frames that are
         buffered between components in the transcoding chain. -->
    <queue-limit>
      <high>16</high>
      <low>8</low>
    </queue-limit>
  </drone>

  <!-- The queen delegates the jobs via the vinculums, as explained in section 4.4,
       Starting the processes. -->
  <queen>

    <!-- This is the port on which the Queen listens to messages. -->
    <client-listen-port>40402</client-listen-port>

    <!-- The monitor password is encrypted. -->
    <monitor-password>aef3a7835277a28da831005c2ae3b919e2076a62</monitor-password>

    <spool-path>/var/spool/pwce</spool-path>
    <shared-settings-path>/Users/Shared/Episode Engine/Settings</shared-settings
-path>

    <!-- This is the time in seconds that the queen retains a job history, so that clients
         can ask about prior events.
    <history-expiration-time>86400</history-expiration-time>

    <vinculums>
      <max-job-retry>2</max-job-retry>
      <num-jobs>2</num-jobs>
    </vinculums>
  </queen>

</sys-conf>
```

Appendix C `services.conf`

There are two `services.conf` files, one for the controller node(s), by default stored in `/usr/local/pwce/svc-ctrl/`, and one for the encoder nodes, by default stored in `/usr/local/pwce/svc-encoder/`. They both contain the same data, simple assignments of variables. A typical file is shown below.

```
IP_ADDRESS=10.50.5.222
CLIENT_PORT=40402
ENGINE_USER="pwce"
ENGINE_GROUP="pwce"
ENGINE_ROOT="/usr/local/pwce"
WATCH_INPUT="/Users/Shared/Episode Engine/Input"
WATCH_OUTPUT="/Users/Shared/Episode Engine/Output"
WATCH_ARCHIVE="/Users/Shared/Episode Engine/Archive"
ENABLE_ARCHIVE="NO"
SAFE_DELAY=10
MAX_SPLIT_COUNT=16
MIN_SPLIT_TIME=60
```

IP_ADRESS is the address of the controller. ENABLE_ARCHIVE is either "YES" or "NO", determining whether transcoded source files are archived or not (in the WATCH_ARCHIVE directory). SAFE_DELAY is the number of seconds a file has to remain unchanged for the folder watcher to assume that it is safe to start using the file.

Episode Engine Pro

MAX_SPLIT_COUNT is the maximum number of pieces a split-and-stitched video file is divided into. MIN_SPLIT_TIME is the minimum length of a split video clip.

Appendix D The Log File

The log file (by default `/var/log/pwce/pwce.log`) logs the parameters for each job and its results. The output is line-based. Each line starts with a time stamp followed by either `ENGINE`, `CLIENT`, `NODE`, `MONITOR` or `JOB`.
`ENGINE` lines give information about the actions of **Episode Engine** itself.

⟨line⟩ ⟶ ⟨time⟩ `ENGINE` ⟨id⟩ `START '` ⟨host⟩ `'`
⟨line⟩ ⟶ ⟨time⟩ `ENGINE` ⟨id⟩ `STOP`
⟨line⟩ ⟶ ⟨time⟩ `ENGINE` ⟨id⟩ ⟨message⟩

`START` and `STOP` messages indicate that **Episode Engine** has started and stopped, respectively. ⟨id⟩ is the process id of the started process.
Example:

```
Wed May  3 10:46:27 2006 ENGINE 20469 START '3c273.i.popwire.com'
```

`CLIENT` lines are emitted by the **watcher** as it prepares for processing jobs.

⟨line⟩ ⟶ ⟨time⟩ `CLIENT` ⟨id⟩ `START '` ⟨host⟩ `'` ⟨id⟩
⟨line⟩ ⟶ ⟨time⟩ `CLIENT` ⟨id⟩ `STOP`
⟨line⟩ ⟶ ⟨time⟩ `CLIENT` ⟨id⟩ ⟨message⟩

⟨host⟩ is the machine on which the client has started.
Examples:

```
Wed May  3 10:46:31 2006 CLIENT 1 START 'localhost' 666
Wed May  3 10:46:31 2006 CLIENT 1 INFO 'client is a job supervisor'
Wed May  3 10:46:31 2006 CLIENT 1 INFO
'input root directory: /Users/Shared/Episode Engine/Input'
Wed May  3 10:46:31 2006 CLIENT 1 INFO
'output root directory: /Users/Shared/Episode Engine/Output'
```

`NODE` lines indicate the state of the machines under the control of **Episode Engine**. The lines are similar to `ENGINE` lines:

⟨line⟩ ⟶ ⟨time⟩ `NODE` ⟨id⟩ `START'` ⟨host⟩ `'`
⟨line⟩ ⟶ ⟨time⟩ `NODE` ⟨id⟩ `STOP`
⟨line⟩ ⟶ ⟨time⟩ `NODE` ⟨id⟩ ⟨message⟩

Example:

```
Wed May  3 10:46:28 2006 NODE 0 START 'localhost'
```

`MONITOR` lines indicate that an input monitor has been created, edited, deleted, started or stopped.

⟨line⟩ ⟶ ⟨time⟩ `MONITOR` ⟨id⟩ `NEW'` ⟨name⟩ `'` ⟨job-prio⟩ `'` ⟨config⟩ `'` ⟨num-settings⟩
⟨line⟩ ⟶ ⟨time⟩ `MONITOR` ⟨id⟩ `DELETE`
⟨line⟩ ⟶ ⟨time⟩ `MONITOR` ⟨id⟩ `UPDATE'` ⟨name⟩ `'` ⟨job-prio⟩ `'` ⟨config⟩ `'` ⟨num-settings⟩
⟨line⟩ ⟶ ⟨time⟩ `MONITOR` ⟨id⟩ `START`
⟨line⟩ ⟶ ⟨time⟩ `MONITOR` ⟨id⟩ `STOP` ⟨reason⟩
⟨line⟩ ⟶ ⟨time⟩ `MONITOR` ⟨message⟩

Examples:

```
Thu Oct 11 10:27:29 2007 MONITOR 1 NEW 'Input Folder' 500
'file:/Volumes/Velox Barnardi/Users/kai/My media files;interval=2&delay=10' 1
Thu Oct 11 10:27:32 2007 MONITOR 1 START
Thu Oct 11 10:36:28 2007 MONITOR 1 STOP abort
```

`JOB` lines indicate the progress of compression jobs and have correspondingly more output alternatives:

⟨line⟩ ⟶ ⟨time⟩ `JOB` ⟨id⟩ `CREATE '` ⟨setting-name⟩ `'` ⟨id⟩ ⟨prio⟩ ⟨time⟩
⟨line⟩ ⟶ ⟨time⟩ `JOB` ⟨id⟩ `QUEUE` ⟨prio⟩ ⟨queue-index⟩ ⟨queue-total⟩
⟨line⟩ ⟶ ⟨time⟩ `JOB` ⟨id⟩ `START` ⟨node-id⟩
⟨line⟩ ⟶ ⟨time⟩ `JOB` ⟨id⟩ `STOP` ⟨reason⟩
⟨line⟩ ⟶ ⟨time⟩ `JOB` ⟨id⟩ `DROP` ⟨reason⟩
⟨line⟩ ⟶ ⟨time⟩ `JOB` ⟨id⟩ `TIME` ⟨realtime⟩ ⟨user-cpu-time⟩ ⟨system-cpu-time⟩
⟨line⟩ ⟶ ⟨time⟩ `JOB` ⟨id⟩ `OUTPUT-URLS'` ⟨url⟩ `'` [⟨urls⟩]
⟨urls⟩ ⟶ (*ε* | `,␣'` ⟨url⟩ `'` ⟨urls⟩)
⟨line⟩ ⟶ ⟨time⟩ `JOB` ⟨id⟩ `INPUT-INFO '` ⟨infile⟩ `'` ⟨info⟩
⟨line⟩ ⟶ ⟨time⟩ `JOB` ⟨id⟩ `OUTPUT-INFO '` ⟨outfile⟩ `'` ⟨info⟩
⟨info⟩ ⟶ `(` ⟨component-name⟩ `,` ⟨component-fourcc⟩ `,` ⟨size⟩ `,` ⟨duration⟩ `,` ⟨bitrate⟩
[`,` `(` ⟨id⟩ `,` ⟨type-fourcc⟩ `,` ⟨format-fourcc⟩ `,` ⟨starttime⟩ `,` ⟨duration⟩ `,` ⟨bitrate⟩ `,` ⟨width⟩ `,` ⟨height⟩ `,` ⟨framerate⟩ `)`]
[`,` `(` ⟨id⟩ `,` ⟨type-fourcc⟩ `,` ⟨format-fourcc⟩ `,` ⟨starttime⟩ `,` ⟨duration⟩ `,` ⟨bitrate⟩ `,` ⟨channels⟩ `,` ⟨bits/sample⟩ `,` ⟨samplerate⟩ `)`]
[`,` `(` ⟨id⟩ `,` ⟨type-fourcc⟩ `,` ⟨format-fourcc⟩ `,` ⟨starttime⟩ `,` ⟨duration⟩ `,` ⟨bitrate⟩ `,` ⟨payload⟩ `,` `(` ⟨fmtp-parm⟩ {`,` ⟨fmtp-parm⟩} `)` `)`]
`)`
⟨reason⟩ ⟶ (`no-start` | `bad-com` | `bad-job` | `fail` | `crash` | `lost` | `cancel` | `finish`)
⟨line⟩ ⟶ ⟨time⟩ `JOB` ⟨id⟩ `ARCHIVE` ⟨timestamp⟩ ⟨num-archived⟩
⟨line⟩ ⟶ ⟨time⟩ `JOB` ⟨id⟩ `EXPIRE` ⟨num-archived⟩

The unit for the `TIME` parameters is seconds. ⟨fourcc⟩ four-character codec codes are listed at `http://www.fourcc.org/fcccodec.htm`. ⟨timestamp⟩ is a UNIX timestamp.
Examples:

```
Mon May  8 20:55:46 2006 JOB 0 CREATE
'(watch) Test/3g_64kbit_qcif_dl.setting/CIMG1406.AVI'
1147112356 255 1147114546
Mon May  8 20:55:46 2006 JOB 0 QUEUE 255 0 1
Mon May  8 20:55:46 2006 JOB 0 START 0
Mon May  8 20:55:49 2006 JOB 0 INPUT-INFO
'file:///Users/Shared/Episode Engine/Input/Test/.workdir/1147112356#3g_64kbit_qcif_dl.
(avi_, 1606018, 10.53, 26554.90,
(0, vide, mjpa, 525609216, 27128216.00, 0.00, 10.53, 320, 240, 14.72),
(1, audi, pc8U, 84250, 64000.00, 0.00, 10.53, 1, 8, 8000.00))
Mon May  8 20:55:49 2006 JOB 1 INPUT-INFO
'file:///Users/Shared/Episode Engine/Input/Test/.workdir/1147112357#mp4_medium.setting
(avi_, 1606018, 10.53, 26554.90,
(0, vide, mjpa, 525609216, 27128216.00, 0.00, 10.53, 320, 240, 14.72),
```

```
(1, audi, pc8U, 84250, 64000.00, 0.00, 10.53, 1, 8, 8000.00))
Mon May  8 20:55:57 2006 JOB 0 OUTPUT-INFO
'file:///Users/Shared/Episode Engine/Output/Test/.workdir/1147112356#CIMG1406-3g_64kbi
(moov, 84613, 10.46, 63.18,
(0, vide, mp4v, 61129, 45.00, 0.00, 10.60, 176, 144, 7.36),
(1, audi, mp4a, 20894, 15.00, 0.00, 10.50, 1, 16, 22050.00))
Mon May  8 20:55:57 2006 JOB 0 STOP finish
Mon May  8 20:55:57 2006 JOB 0 TIME 10.70 2.29 0.65
Mon May  8 20:55:57 2006 JOB 0 DROP finish
```

All lines can output informational messages at different severity levels. What messages are output is determined by the `filelog-priority` field in the `engine.conf` file, see appendix B, *`engine.conf`*.

⟨message⟩ ⟶ `DEBUG '` ⟨messagetext⟩ `'`
⟨message⟩ ⟶ `INFO '` ⟨messagetext⟩ `'`
⟨message⟩ ⟶ `NOTICE '` ⟨messagetext⟩ `'`
⟨message⟩ ⟶ `WARNING '` ⟨messagetext⟩ `'`
⟨message⟩ ⟶ `ERROR '` ⟨messagetext⟩ `'`
⟨message⟩ ⟶ `CRITICAL '` ⟨messagetext⟩ `'`
⟨message⟩ ⟶ `ALERT '` ⟨messagetext⟩ `'`
⟨message⟩ ⟶ `EMERGENCY '` ⟨messagetext⟩ `'`

Bibliography

[1] Douglas R. Mauro and Kevin J. Schmidt. *Essential SNMP*. O'Reilly, 2005.

[2] Dave Taylor. *Learning Unix for Mac OS X Tiger*. O'Reilly, 2005.

Index