TELESTREAM

Admin Guide 5.1

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.

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 ©2009, 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 this document

This Administrator's Guide describes how to install and configure **Episode Podcast**.

1.1 Audience for this manual

The reader is assumed to be a system administrator with experience of Mac OS X Server.

1.2 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 **Show Log** from the **Window** 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.

2 Overview of **Episode Podcast**

Episode Podcast is an application that seamlessly plugs into Apple **Podcast Producer**, greatly extending the range of supported input and output formats as well as allowing a number of options to process your media. **Episode Podcast** lets you publish your media in leading video and audio formats, including Windows Media, Flash 8, MP3 and RealMedia. It also lets you use the Apple **Podcast Capture** application to submit files in any popular format for processing in your **Podcast Producer** workflow.

Episode Podcast comes with the companion application **Episode** for managing the settings that determine how the input is transcoded. **Episode** contains over 200 predefined transcoding settings and an easy-to-use interface for creating bespoke settings, giving you complete control over the output.

You can increase throughput by adding transcoding nodes and purchasing additional **Episode Podcast** licenses. You can upgrade your **Episode Podcast** license to **Episode Podcast Pro** which will add a number of professional file formats. See appendix B, *Supported formats* for all formats supported by **Episode Podcast**.

3 Installing and configuring

3.1 System requirements

Episode Podcast requires OS X Server 10.5 or later with Apple **Podcast Producer** installed. Consult the *Mac OS X Server Podcast Producer Administration* manual on the setup and configuration of the base system.

Episode Podcast will only run on Intel Macs.

3.2 Installing

Episode Podcast is delivered as a disk image. It contains:

- a package installer with the **EpisodePodcast** software,
- a folder with this Administrator's Guide, and the **Episode** User's Guide,
- a folder with example workflow bundles that you can modify for your own needs,
- the **Episode** application, for creating transcoding settings.

You should also have received your licenses in a separately sent file named `licenses`.

1. Double-click the package installer. (This requires you to get administrator privileges.) **Episode Podcast** will automatically be placed on shared storage by **Podcast Producer** and thus be visible to all nodes in your cluster.
2. Move the file `licenses` to `/Library/PodcastProducer/Resources/Tools/EpisodePodcast/etc/`.
3. Move the **Episode** application to your `/Applications` directory
4. Optionally, move any or all example workflows to `/Library/PodcastProducer/Workflows/`.

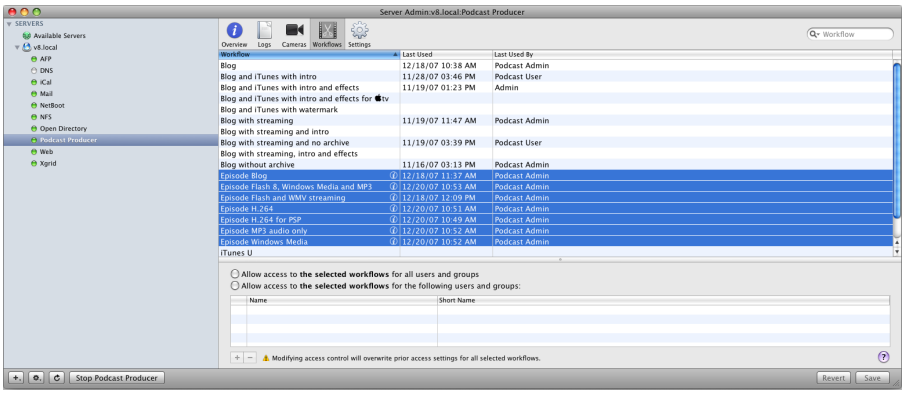
Check your installation by running e g the **Episode Windows Media** workflow. This should create a WMV version of your input media, in the output directory you have designated in **Server Admin**.

As your throughput requirements increase, you may purchase and install additional licenses to enable additional concurrent transcoding jobs. The number of concurrent jobs is determined by the license and not by the number of transcoding nodes.

3.3 Creating workflows

Podcast Producer uses a *workflow* to define how input is to be processed and the output stored. A workflow is a bundle containing several files. Here we will only cover those relevant for **Episode Podcast** and refer you to the *Mac OS X Server Podcast Producer Administration* manual for the full details.

System-defined workflow bundles are stored in `/System/Library/PodcastProducer/Workflows/`; these may be overwritten by system updates. Workflows you create yourself should therefore be stored in `/Library/PodcastProducer/Workflows/`. You can check what workflows you have with the administration tool **Server Admin**.



To use **Episode Podcast**, we first need to go through how it performs transcodings.

3.3.1 Settings

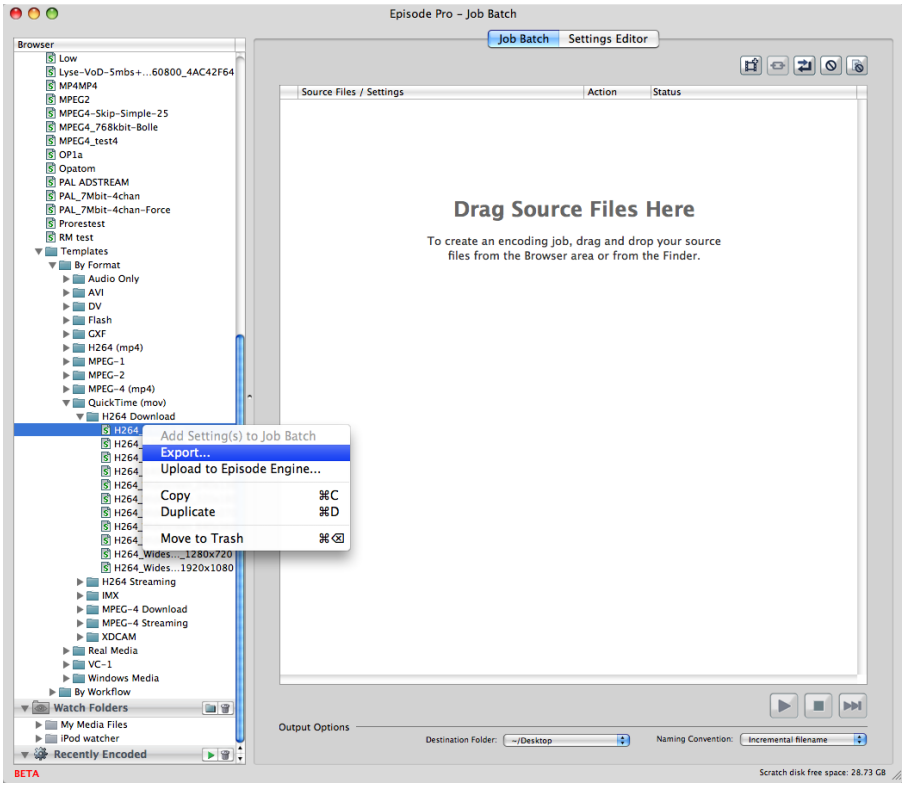
Episode Podcast takes a source file and applies a set of transformations, called *settings*, to transcode the source and create an output file. Settings define an output format and codecs for video and audio. In addition they can contain *filters* for resizing, resampling, colour adjustment and other such operations. Settings are not dependent on the input format but will automatically adapt.

A large number of *templates*, predefined settings files, are supplied with **Episode**. The templates cover the most common formats and needs but if you have special needs, you can create bespoke settings files. See the **Episode** manual for details on creating settings and what transformations can be applied to the input media.

To use a settings file in **Episode Podcast**, you *export* settings from **Episode** by Right-/Ctrl-clicking a settings file in the **Compression Settings** browser and choosing **Export...** in the context menu. Make sure that you select **Episode Podcast** format in the Save dialogue.

Note that modifying the workflow bundles requires administrator privileges, so unless you are running as an administrator, you must therefore first save the settings file to a folder you have write permission in and from there move the file to the intended workflow structure with **Finder** or **Terminal** so that you can authenticate as an administrator.

Note that unless you have a separate license for **Episode**, it will be running in demo mode with limited encoding possibilities, as encoding is intended to be done by **Episode Podcast**.



Additional features

The settings can also set up the source media to be enhanced with additional features: *Intros* and *outros* (also known as *bumpers* and *trailers*) are video sequences attached respectively before and after the main video. *Watermarks* are images or animations added on top of the main video. *Metadata* is information on the contents of the video.

Adding intros, outros and watermarks is as easy as specifying their use in a settings file and then giving their paths as arguments when calling **Episode Podcast**. Using metadata is more complex, since you usually want to modify (parts of) the data for each podcast you create, so see section 3.4, *Metadata* for an overview of how metadata work.

3.3.2 Creating a new workflow

When creating workflows of your own the easiest is to copy an existing workflow (we recommend the example workflows shipped with **Episode Podcast**, see appendix A, *Example workflows*) and edit it according to your needs. In order to use **Episode Podcast** in your local workflow you must edit `Contents/Resources/template.plist`, an XML file which defines all operations that are to be performed on the source material.

`template.plist` contains a section taskSpecifications which is simply command lines in XML form. These define each of the operations that will take a source file from your camera, transcode it and place the output in a suitable location for viewers to view or download. To use **Episode Podcast** you make calls to it here with appropriate arguments.

EpisodePodcast arguments

EpisodePodcast options can be given either in a short form with a dash and a single character, or in a long form with two dashes and a word, either then followed by the parameter(s), as shown below. Equals signs (=) after an option word are optional.

-a *<string1>* *<string2>* | --replace *<string1>* *<string2>* *<string1>* is replaced by *<string2>* in the settings file. As the settings file syntax may change between versions this argument should be used with some care. Some potentially useful strings to replace are:

!SRCFILE! The absolute filename of the input file.

!DSTPATH! The absolute path to the output file.

!NAME! The basename of the input file. (Used when constructing the name of the output file.)

!SETTING! The basename of the settings file. (Used when constructing the name of the output file.)

You can also use this to add metadata to the output file, as explained in section 3.4, *Metadata*.

-b *<prefix>* | --basedir *<prefix>* The <*prefix*> will be used as a path prefix for all *relative* path arguments. Example:

```
EpisodePodcast --basedir /tmp --input file.mov \
--encoder /scratch/WMV.setting
```

will look for `/tmp/file.mov` and `/scratch/WMW.setting`.

-f *<file>* | --file *<file>* <*file*> is the `engine.conf` file. It is used by **Episode Podcast** for certain internal configuration parameters. The <*file*> path is not affected by the --basedir argument. You should normally never need to use this argument.

-h | --help Print out a list of all supported arguments.

-k *<file>* | --watermark *<file>* <*file*> is a watermark file. Watermark files can be BMP, GIF, JPEG, QuickTime, Targa or TIFF files. GIF animations and QuickTime videos can be used for animated watermarks. See the **Episode** manual for a fuller discussion of how to use watermarks. If a watermark has been specified in the settings file this argument is mandatory.

-l *<low>* *<high>* | --limit *<low>* *<high>* The internal buffers in the encoding process should be limited to avoid excessive memory use. <*low*> and <*high*> are the low- and high-water marks for the number of image frames that are allowed to be buffered between individual encoding stages. The default values are 4 and 8, respectively; only under exceptional circumstances should you change these values.

-n *<file>* | --input *<file>* <*file*> is the input file. This argument is mandatory.

-o *<file>* | --output *<file>* <*file*> is the output file. If no output file is specified the output will be placed in the `Desktop` directory with a name constructed from the basename of the input file and the basename of the settings file.

-p | --preflight Check that all input files are correct, but do not perform the actual transcoding. You should normally never need to use this argument.

-w *<time>* | --wait *<time>* Wait <*time*> seconds before exiting after transcoding is finished. You should normally never need to use this argument.

-x *<file>* | --encoder *<file>* <*file*> is the settings file. This argument is mandatory.

-y *<file>* | --intro *<file>* <*file*> is an intro video file (see section 3.3.1, *Additional features*). If an intro has been specified in the settings file this argument is mandatory.

-z *<file>* | --outro *<file>* <*file*> is an outro video file (see section 3.3.1, *Additional features*). If an outro has been specified in the settings file this argument is mandatory.

QuickTime Player requires that MPEG-4 files have the extension `.mp4` for it to be able to play them. Make sure that your **`--output`** argument uses the correct extension.

An example workflow

In the following example we take a source file and convert it to a Flash 8 video, suitable for publishing to the web, and then put a copy of it in an archive. We use a settings file stored in the `Contents/Resources/Encodings/` folder in the workflow bundle, thus making it possible to move the workflow as one unit with no external files to keep track of. If you have watermark and intro/outro files that will be used for multiple encodings it is similarly convenient to place them in respectively `Contents/Resources/Images/` and `Contents/Resources/Movies/`.

The two tasks in this workflow correspond to the command line calls

```
/Library/PodcastProducer/Resources/Tools/EpisodePodcast/bin/EpisodePodcast \
  --encoder=/Library/PodcastProducer/Workflows/Flash8/Contents/Resources/\
  Encodings/FLV_FL8_large.setting --input=inputfile \
  --output=/Users/Shared/outputfile.flv
/usr/bin/pcastaction archive --basedir=BaseDirectory \
  --file=/Users/Shared/outputfile.flv --title=Title \
  --format=archive --archive_folder=ArchiveRoot \
  --use_day_folders=YES --create_folder=YES
```

```
<?xml version="1.0" encoding="UTF−8"?>
<!DOCTYPE plist PUBLIC "−//Apple Computer//DTD PLIST 1.0//EN"
    "http://www.apple.com/DTDs/PropertyList−1.0.dtd">
<plist version="1.0">
<array>
  <dict>
    <!−− The following are configuration parameters that you should not edit. −−>
    <key>artConditions</key>
    <dict>
      <key>0</key>
      <dict>
        <key>artEqual</key>
        <string>1</string>
      </dict>
    </dict>
    <key>artSpecifications</key>
    <dict>
      <key>0</key>
      <dict>
        <key>artArguments</key>
        <array>
          <string>$$Shared Filesystem$$</string>
          <string>$$Server UUID$$</string>
        </array>
        <key>artPath</key>
        <string>$$Workflow Resource Path$$/Tools/art.rb</string>
      </dict>
    </dict>
    <key>name</key>
    <string>$$Xgrid Job Name$$</string>
    <key>notificationEmail</key>
    <string>$$Administrator Email Address$$</string>
    <!−− The following section defines the actual processing steps. This is where
        you define your Episode Podcast tasks. −−>
    <key>taskSpecifications</key>
    <dict>
```

```
<!−− encode_episode is a name that serves to identify this task and can be
     chosen arbitrarily. −−>
<key>encode_episode</key>
<dict>
  <key>command</key>
  <string>$$Global Resource
      Path$$/Tools/EpisodePodcast/bin/EpisodePodcast</string>
  <key>arguments</key>
  <array>
    <!−− It is practical to have the settings file inside the workflow bundle, but
         you can place it elsewhere and modify this argument accordingly. This
         particular settings file converts the source file to Flash 8 format. −−>
    <string>−−encoder=$$Workflow Resource
         Path$$/Encodings/FLV_FL8_large.setting</string>
    <!−− The name of the source file is constructed from resource strings set by
         Podcast
         Producer. −−>
    <string>−−input=$$Content File Basename$$$$Content File
         Extension$$</string>
    <!−− We place the output file in /Users/Shared with the name given for
         this transcoding and the extension .flv, as it is a Flash 8 video file.
         −−>
    <string>−−output=/Users/Shared/$$Content File Basename$$.flv</string>
  </array>
</dict> <!−− End of encode_episode −−>
<!−− You can have any number of tasks in taskSpecifications. By default they
     will be executed in parallel, but often one task is dependent on the output of
     another and we therefore have to specify what other tasks have to be
     finished before starting on a task. This is done with the dependsOnTasks
     clause. The order tasks are written in taskSpecifications has no influence on
     the order they are executed in. −−>
<key>archive</key> <!−− Second task −−>
<dict>
  <key>command</key>
  <string>/usr/bin/pcastaction</string>
  <key>arguments</key>
  <array>
    <string>archive</string>
    <string>−−basedir=$$Base Directory$$</string>
    <string>−−file=/Users/Shared/$$Content File Basename$$.flv</string>
    <string>−−title=$$Title$$</string>
    <string>−−format=archive</string>
    <string>−−archive_folder=$$Archive Root$$</string>
    <string>−−use_day_folders=YES</string>
    <string>−−create_folder=YES</string>
  </array>
  <!−− archive cannot execute before encode_episode has finished. You can list
       any number of other tasks in the array. −−>
  <key>dependsOnTasks</key>
  <array>
    <string>encode_episode</string>
  </array>
</dict> <!−− End of archive −−>
</dict> <!−− End of taskSpecifications −−>
<key>userRequirements</key>
<array>
  <string>Title</string>
```

```
        <string>Description</string>
      </array>
    </dict>
</array>
</plist>
```

A single-argument option **`--file=source.mov`** can be translated into XML as <string>−−file=source.mov</string> or <string>−−file</string><string>source.mov</string>. For the options that take two arguments, **`--limit`** and **`--replace`**, you *must* have the two arguments in separate <string> clauses for them to be properly interpreted. In other words, you can write either

```
<string>−−replace=!AUTHOR!</string>
<string>$$Author$$</string>
```

or

```
<string>−−replace</string>
<string>!AUTHOR!</string>
<string>$$Author$$</string>
```

but you may *not* put all arguments in a single <string> clause.

3.4 Metadata

Metadata contain information about the media file and its contents, such as title, producer, performer etc. **Episode** lets you place metadata in the settings file, but this information will then be constant for all files transcoded with that settings file. Instead you can put placeholders for metadata in the settings file and then use `template.plist` for replacing them with specific values.

The **Podcast Producer** administration tool **Server Admin** lets you add and edit resources. These resources can then be accessed in the `template.plist` file as $$ResourceName$$. You can use these resources to set metadata in the output files as described below.

Using your favourite editor, add a metadata template similar to the following to your settings file:

```
<meta−data meta−file="no">
    <meta−group type="movie">
        <meta name="album" value="!ALBUM!"/>
        <meta name="artist" value="!ARTIST!"/>
        <meta name="author" value="!AUTHOR!"/>
        <meta name="comment" value="!COMMENT!"/>
        <meta name="copyright" value="!COPYRIGHT!"/>
        <meta name="creation−date" value="!CREATION−DATE!"/>
        <meta name="description" value="!DESCRIPTION!"/>
        <meta name="director" value="!DIRECTOR!"/>
        <meta name="disclaimer" value="!DISCLAIMER!"/>
        <meta name="edit−date" value="!EDIT−DATE!"/>
        <meta name="genre" value="!GENRE!"/>
        <meta name="host" value="!HOST!"/>
        <meta name="information" value="!INFORMATION!"/>
        <meta name="original−source" value="!ORIGINAL−SOURCE!"/>
```

```
    <meta name="performers" value="!PERFORMERS!"/>
    <meta name="producer" value="!PRODUCER!"/>
    <meta name="software" value="!SOFTWARE!"/>
    <meta name="title" value="!TITLE!"/>
    <meta name="writer" value="!WRITER!"/>
  </meta-group>
</meta-data>
```

You do not need to use all of these fields, but use those that are relevant for your needs. The !NAME! parts are placeholders, the values of which you can then replace in the call to **EpisodePodcast** in `template.plist`, as follows:

```
<key>taskSpecifications</key>
<dict>
  <key>encode_episode</key>
  <dict>
    <key>command</key>
    <string>$$Global Resource
        Path$$/Tools/EpisodePodcast/bin/EpisodePodcast</string>
    <key>arguments</key>
    <array>
      <string>--encoder=$$Workflow Resource
          Path$$/Encodings/h264_Large.setting</string>
      <string>--input=$$Content File Basename$$$$Content File
          Extension$$</string>
      <string>--output=/Users/Shared/$$Content File Basename$$.mov</string>
      <!-- The --replace will replace the placeholder with the value of the
          resource. $$Title$$ is defined by default, but you can add any number
          of local resources. Note that you have to give the second argument in a
          separate <string> clause. -->
      <string>--replace=!TITLE!</string> <string>$$Title$$</string>
      <!-- Any number of --replace arguments here... -->
    </array>
  </dict>
</dict>
```

This example assumes that you use **Server Admin** to edit the resources whenever you need to change them; a more sophisticated approach would be to use a script that lets the user enter metadata information at the time of recording and then call another script in taskSpecifications to read those metadata and use them to construct the actual call to **EpisodePodcast**.

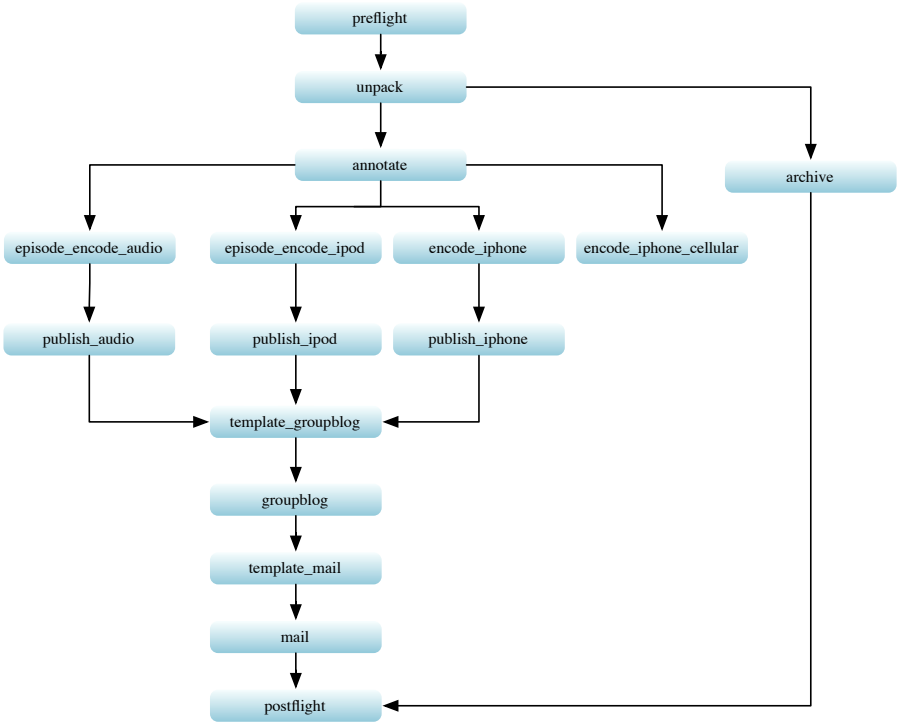
Appendix A Example workflows

With the distribution you get two example workflows which you may use directly or modify to adapt them to your needs.

We will here give graphical overviews of the workflows as an aid in reading the code.

A.1 Episode Blog

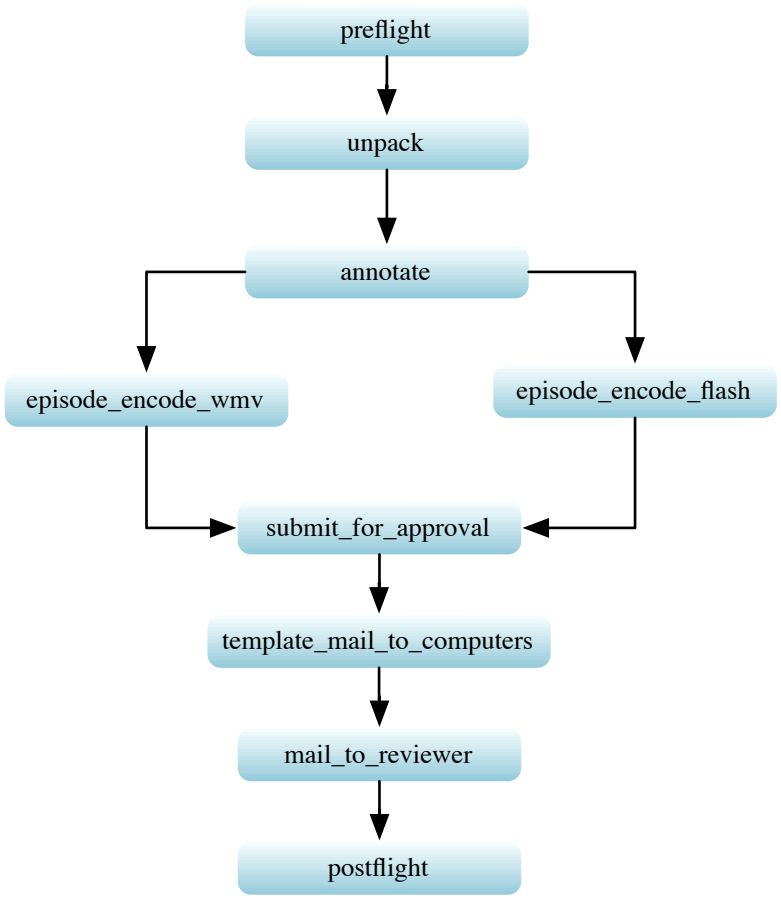
The Episode Blog workflow is based on `/System/Library/PodcastProducer/Workflows/Blog.pwf`, modified to create its output with **Episode Podcast**. It archives and annotates the input file. `episode_encode_audio` creates a 192 Kib/s MP3 output file, `episode_encode_ipod` creates an H.264 video file with AAC audio, suited for replay on an iPod Video. `groupblog` places all output files on a group blog and `mail` email to the blog members on how to access the files.



A.2 Episode Flash and WMV Streaming

The Episode Flash and WMV Streaming workflow is based on `/System/Library/PodcastProducer/Workflows/Blog with streaming and no archive.pwf` and `/System/Library/PodcastProducer/Workflows/Submit for Approval.pwf`, modified to create two output files with **Episode Podcast**. `episode_encode_wmv` creates a 512 Kib/s streaming Windows Media Video 9 file, `episode_encode_flash` creates a 512 Kib/s streaming Flash 8 file. The input will be annotated. `submit_for_approval` places the output in the `Approval` folder on the shared file system for approval by reviewers and `mail_to_reviewer` sends email to the reviewers to inform them of this. To finally put the files up for streaming has then to be done manually or through a second workflow.

Note that neither WMV-9 nor Flash 8 is supported by the QuickTime Streaming Server, you will have to place the output files on a Windows Media Streaming Server and a Flash Server, respectively.



Appendix B Supported formats

The following media formats and codecs are supported by **Episode Podcast**:

3GPP (.3gp)

The 3GPP (3rd Generation Partnership Project) video format is based on QuickTime and intended for mobile phones.

Supported codecs: AAC, AMR NB, H.263, H.264, MPEG-4.

Restrictions: H.264 High Profile is input only. Multi-Bit Rate is output only.

Pro adds: H.264 High Profile support for output. HE-AAC, but for output only.

3GPP2 (.3g2)

3GPP2 (3rd Generation Partnership Project 2) video format is based on QuickTime and intended for mobile phones.

Supported codecs: AAC, AMR NB, EVRC, H.263, H.264, MPEG-4, QCELP.

Restrictions: EVRC, H.264 High Profile, QCELP are input only.

Pro adds: EVRC, H.264 High Profile, QCELP support for output. HE-AAC, but for output only.

3GPP2 (EZMovie) (.3g2)

KDDI have developed the EZMovie version of 3GPP2, which allows a distributor to limit how many times a file is played.

Supported codecs: AAC, AMR NB, EVRC, H.263, H.264, MPEG-4, QCELP.

Restrictions: EZMovie is output only.

Pro adds: EZMovie is only available in Pro.

ADTS (.aac)

Audio Data Transport Stream is a wrapper format for AAC-encoded audio files.

Supported codecs: AAC.

Restrictions: HE-AAC not supported for input.

AIFF (`.aif`)

The Audio Interchange File Format was developed by Apple.

Supported codecs: PCM.

AMC (`.amc`)

AMC (Adaptive Modulation and Coding) is a 3GPP variant. KDDI have developed the EZMovie version of AMC, which allows a distributor to limit how many times a file is played.

Supported codecs: MPEG-4, QCELP

Restrictions: EZMovie files with distribution restrictions are only supported for output.

Pro adds: EZMovie only available in Pro.

AMR (`.amr`)

AMR (Adaptive Multi-Rate) is an audio format.

Supported codecs: AMR Narrowband.

Restrictions: Input only.

Pro adds: AMR support for output.

ATSC A/52 (`.a52`)

ATSC A/52 is an audio format compatible with Dolby AC3.

Supported codecs: ATSC A/52.

AVI (`.avi`)

AVI is Microsoft's wrapper format that encapsulates other video compression standards.

Supported codecs: DV25, DVCPRO25, DVCPRO50, MJPEG, MP3, MPEG-4 (DivX, XViD, FMP4), PCM, RGB16 (555), RGB16 (556), RGB24, RGB32, UYVY, Windows RGB, YCbCr 4:2:0, Y8, YUY2, YV16, YVU16, YVU9, YV12.

Restrictions: DivX, DVCPRO25, DVCPRO50, FMP4, MJPEG, Windows RGB, XviD, YCbCr only supported for input. DivX and XviD input requires a third-party QuickTime plugin.

Pro adds: DVCPRO25, DVCPRO50, Windows RGB, YCbCr supported for output.

DV (`.dv`)

Format for Digital Video; a recording format. This fileformat cannot handle separate timecode tracks like the `.mov` fileformat. iMovie uses this fileformat.

Supported codecs: DV25, DVCPRO25, DVCPRO50.

Restrictions: DVCPRO25 and DVCPRO50 only supported for input.

Pro adds: DVCPRO25 and DVCPRO50 supported for output.

Flash Video (`.flv`)

The Adobe Flash video format.

Supported codecs: H.263, MP3, VP6.

Shockwave Flash (`.swf`)

The Adobe Shockwave Flash format is a multimedia wrapper format which can contain video and audio data.

Supported codecs: H.263, MP3, VP6.

Restrictions: Only audio and video data are supported.

GXF (`.gxf`)

GXF (General eXchange Format) is an interchange format for archival storage and data networks developed by Grass Valley. The GXF format is only used as a transfer format, the receiving server will convert the file to an appropriate internal format. Although GXF can contain several formats like DV and JPEG streams, it is mainly used with MPEG-2. A timecode track can be added to the file.

Supported codecs: MPEG-2, PCM.

Restrictions: Input only.

Pro adds: GXF supported for output.

MP3 (`.mp3`)

A part of the MPEG-1 standard; the full name of this standard is MPEG-1 Audio Layer III. MP3 is a common standard for audio and music compression.

Supported codecs: Lame MP3

MPEG Elementary Stream (`.m1a, .m1v, .m2v, .mpg`)

An MPEG Elementary stream contains a single medium, audio or video, and can be contained in a Program Stream.

Supported codecs: AES, MPEG Audio, MPEG-1, MPEG-2

Restrictions: AES is input only.

MPEG Program Stream (`.mpg`)

An MPEG Program Stream contains Elementary Streams. While the Elementary Streams could be placed sequentially, they are typically interleaved (muxed). Program Streams are intended for reliable media such as DVD or SVCD.

Supported codecs: AAC, AES, ATSC A/52, H.264, MPEG Audio, MPEG-1, MPEG-2, MPEG-4, PCM

Restrictions: AES, H.264 High Profile is input only.

Pro adds: H.264 High Profile supported for output.

MPEG Transport Stream (`.m2t, .ts`)

An MPEG Transport Stream is intended for broadcast media where packets may be lost and viewers have to be able to enter a transmission in mid-stream. Elementary streams are interleaved (muxed) on the Transport Stream.

Supported codecs: AAC, AES, ATSC A/52, H.264, HDV, MP3, MPEG Audio, MPEG-1, MPEG-2, MPEG-4, PCM.

Restrictions: Only supported for input.

Pro adds: Supported for output, except for AES.

MPEG-4 (`.m4a, .m4b, .m4v, .mp4`)

The MPEG standard most commonly in use today, encapsulated by most modern video applications in one aspect or another. The `.m4a`, `.m4b`, and `.m4v` versions are adapted for iPods as audio, audiobook and video specialisations, respectively.

Supported codecs: AAC, H.264, MPEG-4.

Restrictions: H.264 High Profile is input only.

Pro adds: H.264 High Profile and HE-AAC supported for output.

MXF (`.mxf`)

The Material eXchange Format is a wrapper standard intended to better support metadata for media files so that they can be easier kept track of in an environment where media are transmitted, edited and stored entirely digitally. There is currently no implementation that covers the full standard, so interoperability has to be tested for each case. More information is available at `http://www.mxf.info/`.

Supported codecs: AES, BWF, D-10/IMX, DNxHD, DV25, DVCPRO25, DVCPRO50, JPEG2000, MPEG-2, Omneon, PCM, QuickTime codecs, XDCam HD.

Restrictions: D-10/IMX, DNxHD, JPEG2000, Omneon, XDCam HD are input only.

Pro adds: MXF XDCam, D-10/IMX, XDCam HD supported for output.

OGG (`.ogg`)

Ogg is an open media wrapper format designed for efficient streaming and manipulation. More information is available at `http://Xiph.Org`.

Supported codecs: Vorbis.

Restrictions: Output only.

PSP (`.mp4`)

PlayStation Portable can play MPEG-4 files, but requires that they be named `M4Vxxxxx.mp4`, where `xxxxx` is five decimal digits, and stored in the directory `E:\MP_ROOT\100MNV01` on the PSP.

Supported codecs: AAC, H.264, MPEG-4.

QuickTime™ (`.mov`)

Apple's movie file format. This is an umbrella format that encapsulates other video compression standards as well as a few of its own.

Supported codecs: µLaw 2:1, AAC, aLaw 2:1, AMR NB, Apple Animation, Apple Component, Apple GSM, Apple Intermediate Format, Apple Lossless, Apple Prores, Apple Video, *Avid*, *Avid DNxHD*, *Avid DV*, *Avid Meridien*, Blackmagic, Cinepak, D-10/IMX, DV25, DVCPRO25, DVCPRO50, *DVCPRO100*, H.261, H.263, H.264, HDV, IMA, Mace 3:1, Mace 6:1, *Media 100*, MJPEG, MP3, MPEG-4, PCM, Pixlet, Qdesign, *RED*, RGB, RGB16 (555), RGB16 (556), RGB24, RGB32, Sheer Video, Sorenson Video 1, 3, Targa Cine YUV, XDCAM HD, YCbCr (YUV), UYVY, Y8, YUY2, YV16, YVU16, YVU9, YV12. Codecs in *italics* require third party plugins.

Restrictions: Only AAC, AMR NB, Apple Video, Cinepak, DV25, H.261, H.263, H.264, MP3, MPEG-4, PCM, Pixlet, RGB, RGB16 (555), RGB16 (556), RGB24, RGB32, Sorenson Video 1, Sorenson Video 3 supported for output. QuickTime reference files are only supported for input. Timecodes are not supported by the native QuickTime importer—this mainly affects reference files.

Pro adds: µLaw 2:1, aLaw 2:1, Apple Animation, Apple Component, Apple GSM, Apple Intermediate Format, Apple Lossless, Apple Prores, Apple Video, *Avid*, *Avid DNxHD*, *Avid DV*, *Avid Meridien*, Blackmagic, D-10/IMX, DVCPRO25, DVCPRO50, *DVCPRO100*, HDV, HE-AAC, IMA, Mace 3:1, Mace 6:1, *Media 100*, MJPEG, Qdesign, Sheer Video, Targa Cine YUV, XDCAM HD, YCbCr (YUV), UYVY, Y8, YUY2, YV16, YVU16, YVU9, YV12 also supported for output. While any installed QuickTime codec plugins will be used, we do not guarantee full functionality of nor offer helpline support for any such third party QuickTime components.

Wave (`.wav`)

Microsoft's basic audio format.

Supported codecs: PCM.

Windows Media (`.wma, .wmv`)

The Windows Media encoder creates files in Windows Media format, a proprietary format currently playable in **Windows Media player**, **VLC**, and, with the help of the **Flip4Mac Windows Media Components for QuickTime**, in **QuickTime Player**.

Supported codecs: Intellistream, VC-1, Windows Media, Windows Media MBR, WMA Pro, WMA Standard.

Restrictions: Only the largest stream is read from Intellistream multi-bit rate files. VC-1 files require the **Flip4Mac Windows Media Components for QuickTime** for input.

Index