FlipFactory®

Using FlipFactory's Conform Engine Option

This App Note applies to FlipFactory versions 6.0 & later

Part No. 74-0233-00

Introduction

FlipFactory provides a powerful platform for *conform* editing MPEG1 or MPEG2 video, by combining multiple source files (optionally clipped) into a continuous output sequence, transcoding them as necessary to meet your media workflow requirements.

FlipFactory enables you to effectively *clip and compose* content by *conforming* high-quality source clips to the desired composite clip, based on edit points you provide. FlipFactory's *Conform Engine* uses the edit points to clip a single file or combine two or more video files or subsections of files together, with optional external audio, into a single video file with one video layer and up to eight audio layers.

Multi-track audio is valuable in providing a voice-over track for the conformed clip. Note that this feature is not intended to be a full-featured editing tool, but rather, an automated system for conforming multiple source clips into a single output clip – ideal for news, rough-cut editing, or quickly assembling training and other time-sensitive content for broadcast or publication.

Obtaining Conform Engine for FlipFactory

To use FlipFactory's Conform Engine, you must purchase a Conform Engine license for your FlipFactory server. To obtain the license, contact Telestream Sales at license@telestream.net or in the U.S., call 530-470-1300.

Note: *You can only use the Conform Engine through the FlipFactory SDK, which is available at no charge from Telestream. To obtain the FlipFactory SDK, contact the SDK Support Team: sdk@telestream.net.*

Conform Engine Features

The Conform Engine supports MPEG1 and MPEG2 video, and uses EDL lists in specific XML format to perform its task.

Supported Input Files

Video files submitted to the Conform Engine must be in Program Stream, System Stream, or Elementary Stream format with MPEG1 or MPEG2 Standard Definition video. Audio files must be Program Stream with MPEG1 Layer 2 audio, or WAV files. If you have source files in other formats, you must first create a factory to convert them to MPEG2 Program Stream, and then submit a second, Conform Engine job, to assemble them into a single file.

Note: *There is no limit to the number of audio files, provided you are not attempting to utilize all of these files at any one time. However, the number of audio channels the Conform Engine may emit is limited to eight.*

Note: *Conform Engine only supports 1 MPEG1 layer 2 audio stream in a program stream or system stream file. If you have a file with multiple audio streams you will only be able to obtain the first stream.*

EDL Requirements and Conform XML Files

The Conform engine requires in and out marks – an EDL list – for each clip submitted in a job. FlipFactory uses an Edit Decision List (EDL) in the form of a FlipFactory Conform XML file (using FlipFactory's MDML protocol), to control the Conform Engine for each job.

Note: *You can use FlipFactory to produce a frame-accurate, QuickTime proxy file of each video file you are submitting, and use this proxy to produce your mark-in / mark-out points. EDLs may also be generated from MAM systems. Or, you can use any other editor or method required to generate or obtain the EDL data.*

The EDL data must be converted to MDML-compliant Conform XML files, which are submitted to FlipFactory for processing. Each Conform file also includes video specifications, mark-in/mark-out points, video and audio layer definitions, and reference links to each of the source clips which are to be conformed.

You can create a Conform file with EDL data to trim the head and tail of a single clip, or you can create a Conform file with EDL data and references to multiple clips and multiple audio files. You can combine multiple complete files using the Conform Engine as well, by supplying the in and out mark values which represent the entire clip.

In addition to creating Conform files for submission, you can also create Submit XML files, which either reference external Conform files or contain Conform XML content. These Submit XML files are intended for submission to FlipFactory via a custom FlipFactory submit application, using the FlipFactory SDK.

Submitting Conform Engine Jobs

You can submit Conform Engine Jobs in the same manner that you submit other media files to FlipFactory for processing. However, unlike typical FlipFactory jobs where the media file itself is submitted, in Conform Engine jobs, you submit the Conform XML file, which *references* the media files you're submitting. You can set up a factory for manual submission of Conform files using the Job Submit window in the FlipFactory console, or you can add a local or network folder to your Conform Engine factory and submit Conform files via the monitor.

Conform files must have the suffix *.composite*. All files with the *.composite* extension are processed by the Conform Engine. Conform files without the *.composite* suffix will not be recognized or processed.

Processing Conform Engine Jobs

When a factory processes MPEG1 or MPEG2 video and optional audio via a Conform file, it outputs the audio clips ordinally, according to the specifications of the Conform file. If there are separate audio files, each audio layer is a parallel track in time, and is output in parallel with the video to the transcode engine, to produce the new output file according to the product specifications of the factory.

You can set up your Conform Engine factory to flip (transcode) your conformed source into an output file in the same format as your source, or you can flip your source into a different format.

Note: *If the media referenced by the Conform file does not correspond to the specification of the video in the Conform XML file (for example, the video is specified as 720x480 @ 29.97 fps, but is in fact 720 x 576 @ 25 fps); then the particular piece of media that does not adhere to these properties will be rendered as black with no audio. In most cases, your video specifications should match the input video, unless you are deliberately producing filler.*

Conform Engine XML File Requirements

Before creating your submit application, you should examine a sample Conform XML file. Specific elements are highlighted for your review.The complete MDML format and all attributes and elements required to create conform messages and submit messages is described in the SDK.

Example Conform XML File

```
<?xml version="1.0" encoding="UTF-8"?>
<composite uuid="F37D4246-154F-414b-8D5B-6C69C3805200" name="MPEG2 Program Stream" duration="00:00:40.0">
    <setup>
        <frame-width name="Frame Width" unit="pixels" min="80" max="720">720</frame-width>
        <frame-height name="Frame Height" unit="pixels" min="60" max="512">480</frame-height>
        <frame-rate name="Frame Rate" unit="frames/second">29.97</frame-rate>
        <channels name="Audio Channels" unit="channels">5</channels>
        <sample-rate name="Audio Sample Rate" unit="Hz">48000</sample-rate>
    </setup>
    <file uuid="{FDE6E47F-96C5-40F4-8142-6FC59DCC378D}" path="F:\Video Media\MPEG\demo.mpg" url="file://F:\Video Media\MPEG\demo.mpg" name="demo.mpg"/>
    <file uuid="{FDE6E47F-96C5-40F4-8142-6FC59DCC3788}" path="C:\SpongeBob\Media\kellogs.wav" url="file://C:\SpongeBob\Media\kellogs.wav" name="kellogs.wav"/>
    <file uuid="{FDE6E47F-96C5-40F4-8142-6FC59DCC3789}" path="C:\SpongeBob\Media\ftd1Channel.wav" url="file://C:\SpongeBob\Media\ftd1Channel.wav" name="ftd1Channel.wav"/>
    <layer type="movie">
        <container type="movie" name="MPEG2 Program Stream" extensions="mpg" description="MPEG2 program Stream Format" mime-type="video/mpeg"/>
        <setup>
            <channels name="Audio Channels" type="exclusive" unit="channels">2<output1 name="Output Channel 1" type="numeric" min="1" max="8" mod="1" unit="channels">1</output1>
            <output2 name="Output Channel 2" type="numeric" min="1" max="8" mod="1" unit="channels">2</output2>
            </channels>
        </setup>
        <edit sequence="1" in="00:00:05.0" out="00:00:15.0" uuid="{FDE6E47F-96C5-40F4-8142-6FC59DCC378D}"/>
        <edit sequence="2" in="00:00:00.0" out="00:00:10.0" uuid="{FDE3347F-96C5-40F4-8142-6FC522CC3781}"/>
        <edit sequence="3" in="00:00:10.0" out="00:00:30.0" uuid="{FDE6E47F-96C5-40F4-8142-6FC59DCC378D}"/>
    </layer>
    <layer type="audio">
        <container type="audio" name="Uncompressed Audio" extensions="wav" description="Microsoft WAV Format" mime-type="audio/x-wav"/>
        <setup>
            <channels name="Audio Channels" type="exclusive" unit="channels">2<output1 name="Output Channel 1" type="numeric" min="1" max="8" mod="1" unit="channels">3</output1>
            <output2 name="Output Channel 2" type="numeric" min="1" max="8" mod="1" unit="channels">4</output2>
            </channels>
        </setup>
        <edit sequence="1" in="00:00:00.0" out="00:00:20.0" uuid="{FDE6E47F-96C5-40F4-8142-6FC59DCC3788}"/>
        <edit sequence="2" in="00:00:00.0" out="00:00:03.0" uuid="{FDE3347F-96C5-40F4-8142-6FC522CC3781}"/>
        <edit sequence="3" in="00:00:00.0" out="00:00:17.0" uuid="{FDE6E47F-96C5-40F4-8142-6FC59DCC3788}"/>
    </layer>
    <layer type="audio">
        <container type="audio" name="Uncompressed Audio" extensions="wav" description="Microsoft WAV Format" mime-type="audio/x-wav"/>
```

```
        <setup>
            <channels name="Audio Channels" type="exclusive" unit="channels">1<output1 name="Output
Channel 1" type="numeric" min="1" max="8" mod="1" unit="channels">5</output1>
            </channels>
        </setup>
        <edit sequence="1" in="00:00:00.0" out="00:00:07.0" uuid="{FDE6E47F-96C5-40F4-8142-
6FC59DCC3789}"/>
        <edit sequence="2" in="00:00:00.0" out="00:00:20.0" uuid="{FDE3347F-96C5-40F4-8142-
6FC522CC3781}"/>
        <edit sequence="3" in="00:00:00.0" out="00:00:13.0" uuid="{FDE6E47F-96C5-40F4-8142-
6FC59DCC3789}"/>
    </layer>
</composite>
```

The single input file referenced in this example document is specified in the *file* element: *F:\video media\mpeg\demo.mpg*. You can specify any number of input files, and each should be referenced in this manner, using the file element. Each file element must have a unique UUID attribute, so that it can be referenced later.

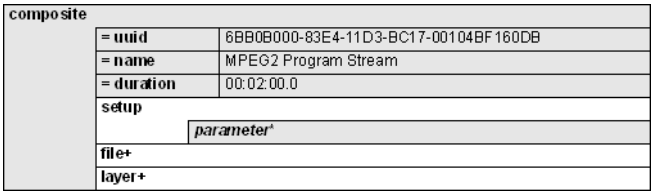
The *layer* element contains the description of how the clip specified in the *setup* element is to be constructed. The layer has 2 *edit* elements, which contain a cut of the specific portion of the input file that is desired. The first cut is from time 00:00:05.0 to 00:00:15.0 (a total of 10 seconds); the second cut is from time 00:00:10.0 to 00:00:30.0 (for a total of 20 seconds). Together, these combine to create a clip of duration 30 seconds (which is specified in the composite element).

The Composite Element

The main, *composite* element is a description of the output of the Conform Engine. The UUID identifies the Conform component in FlipFactory. In this example, the output has a total duration of 30 seconds, and the properties (specified in the *setup* sub-element) of 720x480 @ 29.97, with 2 channels of audio, sampled at 48000Hz.

The composite element is used to describe the media content or program file that is produced by editing together multiple source clips. A composite element may contain multiple audio and/or video layers with each layer described by a corresponding layer element.

Figure 1. The Composite element and attributes.



The setup element contains a list of parameters that define the output of the conform engine, before decoding. In most cases, it should specifically replicate the incoming video – if not, black is output. For example, image width and height, number of audio channels, audio sample rate, etc.

The file element(s) describe the format and location of each file used within the composite. The UUID must be created by you, and it must be unique. The same UUID is used later in the message to identify the edit sequence of the same file.

Composite content is produced by a specific composite decoder component. Each decoder component has unique capabilities typically related to the number or type of layers that can be produced or the type of media files that can be included in the composite. The composite element is also used to describe the capabilities of the component.

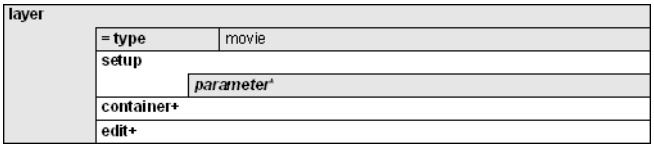
Attributes	Description
uuid	A globally unique identifier associated with composite component

name	A brief descriptive name for this composite component.
duration	A standard time value representing the total duration of the composite.

The Layer Element

The layer element describes a particular audio, video or movie (video and audio) layer within a composite program.

Figure 2. Layer element and attributes.



The container element(s) define the type of media content files that are allowed within the layer.

The setup element contains a list of parameters that define the behavior of the layer, for example, image width and height, number of audio channels, audio sample rate, etc.

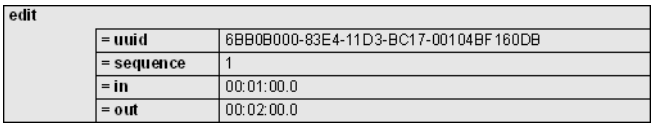
The edit elements determine the actual media content within the layer.

Attributes	Description
type	Describes the type of media content allowed within the layer from a set of enumerations: video Layer contains only video samples. audio Layer contains only audio samples. movie Layer contains video and audio samples

The Edit Element

The edit element describes a single piece of media content within a layer. This element determines the portion of the original media that is used and the position of the media with respect to other edits in the same layer.

Figure 3. Edit element and attributes.



The edit/@uuid attribute is a reference to the source media and should contain the uuid value of the corresponding file element. The absence of a uuid reference indicates a fill or spacer edit which is generally replaced with black (or silence) in the final composite.

Attributes	Description
uuid	Unique identifier of the corresponding media file used for this edit
sequence	number representing the order of this edit with respect to other edits in the same layer
in	Standard time value representing the first sample of the edit (inclusive)
out	Standard time value representing the last sample of the edit (exclusive).

Creating an App to Submit Conform Jobs

If you want to automate Conform job submission to FlipFactory, you can use the FlipFactory SDK to create a custom job submission application. The FlipFactory SDK provides the tools you need to produce FlipFactory MDML messages for job submission, or to integrate the capabilities of FlipFactory into a digital media application or workflow.

Note: *The FlipFactory SDK is available at no charge from Telestream. To obtain the FlipFactory SDK, contact the SDK Support Team: sdk@telestream.net.*

You can write your own job submit application in any language or system that can create an XML message and submit it to FlipFactory. Submitting a FlipFactory job requires collecting information about the input files and other parameters, the factory, and then formatting an MDML message (a specific format of XML for FlipFactory) and sending it to the appropriate factory.

You can write the application to produce the required Conform files and submit jobs manually, or you can trigger the application to submit jobs based on other systems, creating your own Conform automatic submission component.

Writing your Conform Job Submission Application

Use the FlipFactory SDK Developer Reference (Chapter 2, Submit) to learn about the format of MDML messages you need to construct for submission. After constructing the proper MDML message for your job, you can submit it to FlipFactory using various methods (HTTP, Java RMI, SQL, or Email).

Copyright and Trademark Notice

Limited Warranty and Disclaimers

Telestream, Inc. warrants to you, as the original licensee only, that the software you licensed will perform as stated below for a period of one (1) year from the date of purchase of the software by you:

The software will operate in substantial conformance with its specifications as set forth in the applicable product user's guide/published specifications/product description. Telestream does not warrant that operation of the software will be uninterrupted or error-free, will meet your requirements, or that software errors will be corrected. Telestream's sole liability under Section 1 of this Limited Warranty shall be to use reasonable commercial efforts to bring the Software's performance into substantial conformance with the specifications in the applicable product user's guide/ published specifications/product description.

FlipFactory has been designed for professionals skilled in the art of digital media transformation and workflow automation, to facilitate the automation of complex media operations and workflow that require a multitude of input and output media formats, delivery to numerous types of media devices and file systems, and notification of media systems including broadcast automation systems and media asset management systems.

The FlipFactory architecture and user interface is designed to provide maximum flexibility in the setup and configuration of these complex media transformations and workflow. In providing this high degree of flexibility, it is possible for media transformation and workflow processes to be configured that are impractical, likely to result in unexpected or unintended results, or beyond the limits of FlipFactory to perform satisfactorily. Additionally, FlipFactory may be executed on a platform that lacks the performance or capacity to perform the media transformations and workflow you've configured, which is your responsibility to specify. Telestream has chosen to implement FlipFactory to provide the greatest flexibility without limiting its functionality to only those transformations and workflow that are known with certainty to be within its performance capabilities, including those limits imposed by the platform upon which you have installed FlipFactory.

Therefore, you acknowledge that you may create transformations and workflow that are impractical or beyond your FlipFactory installation's limits, and Telestream does not warrant that each transformation or workflow you specify or use will complete without error.

Limitations of Warranties. EXCEPT AS EXPRESSLY SET FORTH IN SECTION 1 ABOVE, NO OTHER WARRANTY, EXPRESS OR IMPLIED, IS MADE WITH RESPECT TO THE SOFTWARE, INCLUDING WITHOUT LIMITATION ANY IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, NON-INFRINGEMENT OF THIRD PARTY RIGHTS AND THOSE ARISING FROM A COURSE OF DEALING OR USAGE OF TRADE. NO WARRANTY IS MADE THAT USE OF THE SOFTWARE WILL BE ERROR FREE OR UNINTERRUPTED, THAT ANY ERRORS OR DEFECTS IN THE LICENSED MATERIALS WILL BE CORRECTED, OR THAT THE SOFTWARE'S FUNCTIONALITY WILL MEET YOUR REQUIREMENTS.

July, 2010 **Part No. 74-0233-00**