Flip64 CML
Developer Guide

Flip64 CML Developer Guide

Vantage 8.0
ComponentPac 8.0.8

December 2020 **302698**

Copyrights and Trademark Notices

PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER 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.

Microsoft. Microsoft, Windows NT|2000|XP|XP Professional|Server 2003|Server 2008 |Server 2012, Windows 7, Windows 8, Windows 10, Media Player, Media Encoder, .Net, Internet Explorer, SQL Server 2005|2008|Server 2012, and Windows Media Technologies are trademarks of Microsoft Corporation.

NLOG, MIT, Apache, Google. NLog open source code used in this product under MIT License and Apache License is copyright © 2014-2016 by Google, Inc., © 2016 by Stabzs, © 2015 by Hiro, Sjoerd Tieleman, © 2016 by Denis Pushkarev, © 2015 by Dash Industry Forum. All rights reserved.

SharpSSH2. SharpSSH2 Copyright (c) 2008, Ryan Faircloth. All rights reserved. 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 Diversified Sales and Service, Inc. nor the names of its 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.

Swagger. Licensed from SmartBear.

Telerik. RadControls for ASP.NET AJAX copyright Telerik All rights reserved.

VoiceAge. This product is manufactured by Telestream under license from VoiceAge Corporation.

x264 LLC. The product is manufactured by Telestream under license from x264 LLC.




MPEG Disclaimers

MPEGLA MPEG2 Patent

ANY USE OF THIS PRODUCT IN ANY MANNER OTHER THAN PERSONAL USE THAT COMPLIES WITH THE MPEG-2 STANDARD FOR ENCODING VIDEO INFORMATION FOR PACKAGED MEDIA IS EXPRESSLY PROHIBITED WITHOUT A LICENSE UNDER APPLICABLE PATENTS IN THE MPEG-2 PATENT PORTFOLIO, WHICH LICENSE IS AVAILABLE FROM MPEG LA, LLC, 4600 S. Ulster Street, Suite 400, Denver, Colorado 80237 U.S.A.

MPEGLA MPEG4 VISUAL

THIS PRODUCT IS LICENSED UNDER THE MPEG-4 VISUAL PATENT PORTFOLIO LICENSE FOR THE PERSONAL AND NON-COMMERCIAL USE OF A CONSUMER FOR (i) ENCODING VIDEO IN COMPLIANCE WITH THE MPEG-4 VISUAL STANDARD ("MPEG-4 VIDEO") AND/OR (ii) DECODING MPEG-4 VIDEO THAT WAS ENCODED BY A CONSUMER ENGAGED IN A PERSONAL AND NON-COMMERCIAL ACTIVITY AND/OR WAS OBTAINED FROM A VIDEO PROVIDER LICENSE IS GRANTED OR SHALL BE IMPLIED FOR ANY OTHER USE. ADDITIONAL INFORMATION INCLUDING THAT RELATING TO PROMOTIONAL, INTERNAL AND COMMERCIAL USES AND LICENSING MAY BE OBTAINED FROM MPEG LA, LLC. SEE HTTP://WWW.MPEGLA.COM.

MPEGLA AVC

THIS PRODUCT IS LICENSED UNDER THE AVC PATENT PORTFOLIO LICENSE FOR THE PERSONAL USE OF A CONSUMER OR OTHER USES IN WHICH IT DOES NOT RECEIVE REMUNERATION TO (i) ENCODE VIDEO IN COMPLIANCE WITH THE AVC STANDARD ("AVC VIDEO") AND/OR (ii) DECODE AVC VIDEO THAT WAS ENCODED BY A CONSUMER ENGAGED IN A PERSONAL ACTIVITY AND/OR WAS OBTAINED FROM A VIDEO PROVIDER LICENSED TO PROVIDE AVC VIDEO. NO LICENSE IS GRANTED OR SHALL BE IMPLIED FOR ANY OTHER USE. ADDITIONAL INFORMATION MAY BE OBTAINED FROM MPEG LA, L.L.C. SEE HTTP://WWW.MPEGLA.COM.

MPEG4 SYSTEMS

THIS PRODUCT IS LICENSED UNDER THE MPEG-4 SYSTEMS PATENT PORTFOLIO LICENSE FOR ENCODING IN COMPLIANCE WITH THE MPEG-4 SYSTEMS STANDARD, EXCEPT THAT AN ADDITIONAL LICENSE AND PAYMENT OF ROYALTIES ARE NECESSARY FOR ENCODING IN CONNECTION WITH (i) DATA STORED OR REPLICATED IN PHYSICAL MEDIA WHICH IS PAID FOR ON A TITLE BY TITLE BASIS AND/OR (ii) DATA WHICH IS PAID FOR ON A TITLE BY TITLE BASIS AND IS TRANSMITTED TO AN END USER FOR PERMANENT STORAGE AND/OR USE. SUCH ADDITIONAL LICENSE MAY BE OBTAINED FROM MPEG LA, LLC. SEE HTTP://WWW.MPEGLA.COM FOR ADDITIONAL DETAILS.

Limited Warranty and Disclaimers

Telestream, LLC (the Company) warrants to the original registered end user that the product will perform as stated below for a period of one (1) year from the date of shipment from factory:

Hardware and Media—The Product hardware components, if any, including equipment supplied but not manufactured by the Company but NOT including any third party equipment that has been substituted by the Distributor for such equipment (the "Hardware"), will be free from defects in materials and workmanship under normal operating conditions and use.

Warranty Remedies

Your sole remedies under this limited warranty are as follows:

Hardware and Media—The Company will either repair or replace (at its option) any defective Hardware component or part, or Software Media, with new or like new Hardware components or Software Media. Components may not be necessarily the same, but will be of equivalent operation and quality.

Software Updates

Except as may be provided in a separate agreement between Telestream and You, if any, Telestream is under no obligation to maintain or support the Software and Telestream has no obligation to furnish you with any further assistance, technical support, documentation, software, update, upgrades, or information of any nature or kind.

Restrictions and Conditions of Limited Warranty

This Limited Warranty will be void and of no force and effect if (i) Product Hardware or Software Media, or any part thereof, is damaged due to abuse, misuse, alteration, neglect, or shipping, or as a result of service or modification by a party other than the Company, or (ii) Software is modified without the written consent of the Company.

Limitations of Warranties

THE EXPRESS WARRANTIES SET FORTH IN THIS AGREEMENT ARE IN LIEU OF ALL OTHER WARRANTIES, EXPRESS OR IMPLIED, INCLUDING, WITHOUT LIMITATION, ANY WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. No oral or written information or advice given by the Company, its distributors, dealers or agents, shall increase the scope of this Limited Warranty or create any new warranties.

Geographical Limitation of Warranty—This limited warranty is valid only within the country in which the Product is purchased/licensed.

Limitations on Remedies—YOUR EXCLUSIVE REMEDIES, AND THE ENTIRE LIABILITY OF TELESTREAM, LLC WITH RESPECT TO THE PRODUCT, SHALL BE AS STATED IN THIS LIMITED WARRANTY. Your sole and exclusive remedy for any and all breaches of any Limited Warranty by the Company shall be the recovery of reasonable damages which, in the aggregate, shall not exceed the total amount of the combined license fee and purchase price paid by you for the Product.

Damages

TELESTREAM, LLC SHALL NOT BE LIABLE TO YOU FOR ANY DAMAGES, INCLUDING ANY LOST PROFITS, LOST SAVINGS, OR OTHER INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF YOUR USE OR INABILITY TO USE THE PRODUCT, OR THE BREACH OF ANY EXPRESS OR IMPLIED WARRANTY, EVEN IF THE COMPANY HAS BEEN ADVISED OF THE POSSIBILITY OF THOSE DAMAGES, OR ANY REMEDY PROVIDED FAILS OF ITS ESSENTIAL PURPOSE.

Further information regarding this limited warranty may be obtained by writing:
Telestream, LLC
848 Gold Flat Road
Nevada City, CA 95959 USA

You can call Telestream during U. S. business hours via telephone at (530) 470-1300.

Regulatory Compliance

Electromagnetic Emissions: FCC Class A, EN 55022 Class A, EN 61000-3-2/-3-3, CISPR 22 Class A

Electromagnetic Immunity: EN 55024/CISPR 24, (EN 61000-4-2, EN 61000-4-3, EN 61000-4-4, EN 61000-4-5, EN 61000-4-6, EN 61000-4-8, EN 61000-4-11)

Safety: CSA/EN/IEC/UL 60950-1 Compliant, UL or CSA Listed (USA and Canada), CE Marking (Europe)

California Best Management Practices Regulations for Perchlorate Materials:
This Perchlorate warning applies only to products containing CR (Manganese Dioxide) Lithium coin cells. Perchlorate Material-special handling may apply. See www.dtsc.ca.gov/hazardouswaste/perchlorate.

Contacting Telestream

To obtain product information, technical support, or provide comments on this guide, contact us using our Website, email, or phone number as listed below.

Resource	Contact Information
Vantage Technical Support	Web Site: http://www.telestream.net/telestream-support/Vantage/support.htm Support Email: support@telestream.net Enterprise Telephone Support: U. S. Toll Free: (877) 257-6245 U. S. from outside U.S.: (530) 470-2036 Europe | Middle East | Africa | Asia | Pacific: +49 228 280 9141 Terms and times of support services vary, per the terms of your current service contract with Telestream.
Vantage Information, Assistance, FAQs, Forums, & Upgrades	Web Site: http://www.telestream.net/telestream-support/Vantage/support.htm Support Email: support@telestream.net
Telestream, LLC	Web Site: www.telestream.net Sales and Marketing Email: info@telestream.net Telestream, LLC 848 Gold Flat Road, Suite 1 Nevada City, CA USA 95959
International Distributor Support	Web Site: www.telestream.net See the Telestream Web site for your regional authorized Telestream distributor.
Telestream Technical Writers	Email: techwriter@telestream.net If you have comments or suggestions about improving this document, or other Telestream documents—or if you've discovered an error or omission, please email us.

Contents

Flip64 CML Overview

Flip64 Composition Markup Language (CML) is a specialized dialect of the Transcode family of CML, which describes media in a manner suitable for stitching media using the Flip64 transcoder in Vantage workflows. Flip64 is ideally suited for automating repetitive media generation tasks in a production environment.

The purpose of this guide is to help producers, editors, operators, and others involved in media processing learn how to create an edit decision lists (EDL) using Flip64 Composition Markup Language to automatically create simple compositions from same-format media sources or trim and concatenate—*stitch*—clips together to generate new media to your specifications, directly in Flip64 workflows.

Note: Users familiar with CML will recognize Flip64 CML fundamentally as a dialect of Post Producer CML, with variations designed specifically for media stitching in Flip64.

Learning and using Flip64 CML requires familiarity with both Vantage and the Flip64 action. Because the language is implemented in XML format, a basic understanding of XML is also helpful.

In this chapter, you'll learn what Flip64 CML is designed to do, and how it works.

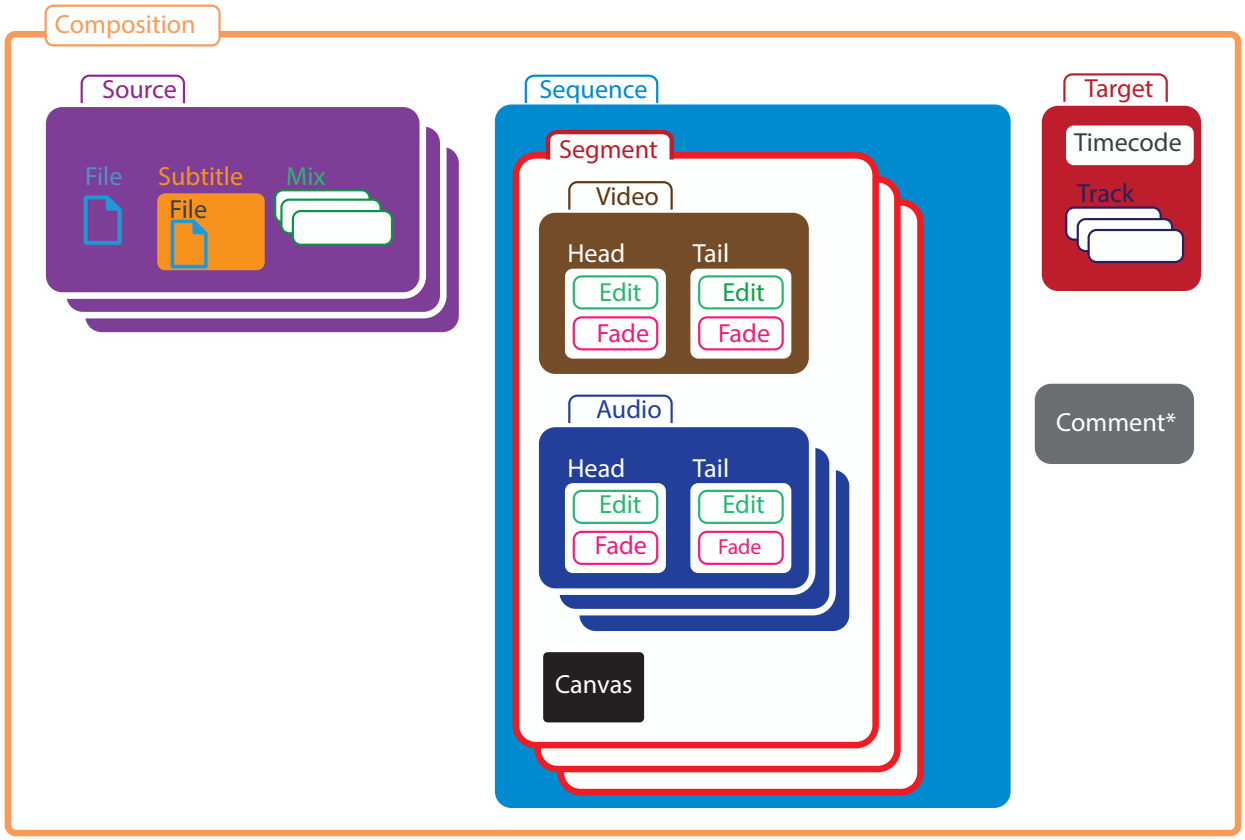
Introduction

Flip64 Composition Markup Language (CML) is a media definition language designed specifically for utilizing Telestream's Flip64 transcoder in Vantage to stitch media clips—in transcoding or direct-conversion workflows. In addition, some basic audio processing typically associated with stitching is provided.

Stitching media is an efficient way to connect multiple, sequential input files using a Flip64-based Vantage workflow to produce a single file. Stitching with Flip64 CML is ideal for combining short clips, removing black sections, extracting sub-clips, stitching program segments together, or adding sponsorship (or black frames) in the middle of a clip. You can also use stitching for adding bumpers or trailers (or both), without resorting to a non-linear editor (NLE).

A typical application is to create a thirty-minute program with a bumper (remapping audio if required), three segments with ads and a trailer, and submitting them to a workflow that combines them to produce an MPEG-2 production output file.

Flip64 CML uses XML syntax. Each of its elements identifies a particular aspect of media, along with attributes to configure it as required. The elements function as building blocks, enabling you to create and organize the components to describe the media you want to generate. This diagram illustrates the relationship between the element. Click on any element to learn about it:



Stacked elements indicates multiple instances are permitted.

Stitching is *only* performed on media of the same resolution and frame rate, where Flip64 typically performs a direct-convert on the CML output. Sources of the same resolution and frame rate that are encoded with different codecs or formats may also be stitched and then transcoded in Flip64.

Note: Flip64 CML processing should be approached from a design and implementation perspective as a separate, pre-processing step, performed directly in Flip64 before transcoding the media generated by the CML pre-processor. When Flip64 ingests CML, it parses it to generate (conform) the specified media for processing by the Flip64 transcoder —just as if the dynamically-generated input media had been ingested directly from a pre-existing file. Thus, in the context of this guide, the term *output* represents the dynamically-generated output from the CML pre-processor; not the output of Flip64 itself. The output of the CML pre-processor is *input* only in the context of the Flip64 transcoder.

How Flip64 CML Works

Flip64 enables you to stitch media clips together and perform other related tasks using Flip64 CML. These topics introduce the key aspects of Flip64 CML stitching:

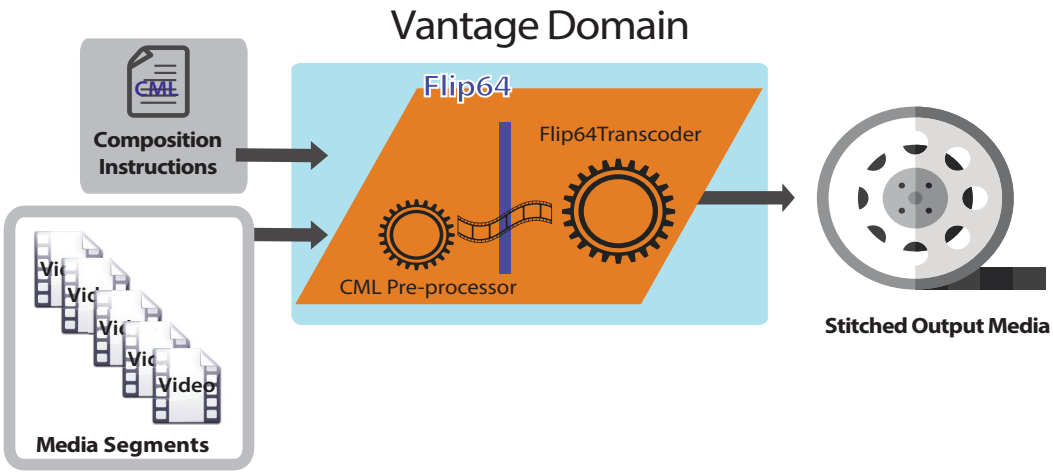
- Processing Video and Adding Gaps
- Audio Processing
- Processing VBI, VANC, and Captions
- Controlling Timecode

Processing Video and Adding Gaps

When Flip64 ingests a CML file, it parses the elements and generates new media from the specified source files (or clips) and optionally, auto-generated black frames, based on the media instructions (commonly called a *composition*) contained in the Flip64 CML file. Stitching is performed prior to Flip64 transcoding that occurs.

Gaps or slate—spaces between clips—are specified in CML, using the *Canvas* element.

Flip64 next transcodes or direct-converts the media generated per the CML, generating the final output file per the Flip64 settings. This process is the functional equivalent of submitting the generated output of the CML directly to Flip64, instead of the CML file.



When direct converting, Vantage doesn't decode the source video. Instead, it passes the frames directly to the output file. With long-GOP media, Vantage creates a new GOP if the original GOP is broken. Otherwise, the original video essences are stitched without re-encoding. This is an extremely fast operation, and doesn't degrade video quality. To accomplish this, the Flip64 action is configured with a Direct Convert profile for the format being encoded. For example, three SD MPEG-2 files can be stitched with their original video essences entirely preserved, with new frames encoded only at stitch points as needed to repair broken GOPs, into one new SD MPEG-2 file.

Vantage supports video re-wrapping (direct convert) and transcoding; both performed via the Flip64 action in your workflow. (Only uncompressed audio is direct-converted; compressed audio is always decoded and re-encoded.)

Alternatively, Vantage can stitch input files together while encoding the video into a different format, in the same workflow. This allows you to encode the segments into any format supported by Vantage, as configured in the workflow's Flip64 action. For example, three SD MPEG-2 files can be stitched, and the media then re-encoded as a QuickTime file, or a Material eXchange Format (MXF) file.

Audio Processing

To utilize audio, you define the audio sources and then add them on a clip-by-clip basis. Finally, you organize channels into tracks for output.

Here are the typical steps:

1. First, you create sources—specifying the file whose audio you want to use. This is often the same file as the video stream. You may have audio-only files or files from which you only use the audio. In each source, you identify which channels to use and specify their order, which may be different than their order in the source file.
2. You assign sources to segments to arrange them in sequence on the timeline.
3. Finally, you organize all channels into tracks for processing by Flip64.

Note: For a comprehensive explanation, see *Basic Audio Processing*.

Sources in CML serve to identify the file containing the audio you plan to use, along with channel selection and order. Each segment you create in your sequence can include audio sources, along with edits, on the sequence's timeline. With a sequence's channels organized into tracks, the audio is added to the output file with the video.

Processing VBI, VANC, and Captions

During stitching, vertical blanking interval (VBI), vertical ancillary data (VANC), and captions are passed from source files to the output file when supported. To preserve blanking data, both the input file decoder and the encoder you use in your workflow must support the required type of blanking data for the media format you're processing. Most encoders support blanking data when enabled in the encoder configuration. Flip64 CML processing does not support all possible types of blanking data for all possible input file types, but many of the most commonly used combinations are supported.

When you specify gaps in the output file, black video is generated along silent audio data, and some form of blanking data: timecodes, captions, VANC, and VBI. When creating gaps, an empty VANC payload is provided. If the format is NTSC, a null caption packet is also produced. If the source has VBI lines (SD material only) blank or black VBI data is produced. If the sources are NTSC, null closed caption lines are synthesized onto line 21 in field 1 and field 2.

Controlling Timecode

You can optionally set the output media's starting timecode, which overrides source timecode. The timecode is then incremented throughout the stitched output file.

Flip64 CML Requirements and Limitations

The following requirements and limitations apply to Flip64CML processing:

- CML files must have a CML extension (for example, MyFlip64CMLfile.cml) to be recognized as a CML file and processed properly in Flip64.
- Flip64 only processes files on Windows (NTFS) platforms—using a drive letter or UNC path. If your workflow is watching for new files on a server other than a Windows server, you must add a Copy or other transport action to locate the file on a Windows server accessible to the Transcode service processing it.
- Flip64 must be configured with one Auto Input. Other types of Inputs cause the job to fail. Auto Input supports one video stream, one compressed audio track or multiple uncompressed audio tracks, and one ANC stream. For input files that contain multiple audio tracks, Auto Input consolidates the tracks into a single audio track and stacks the channels ordinally, index 1.
- Stitching is supported for media with the same video frame size and rates. You cannot stitch media with mixed frame sizes or frame rates. If you have mixed media, process the media through another workflow to bring your frame size and rate into conformance with your planned output specifications.
- Flip64 CML supports a single layer of media. It does not support overlays or compositing. If you require overlays or compositing (or other forms of media than video and audio), use a Post Producer Conform workflow with Post Producer CML.
- Audio V-fade is supported, but cross-fade is not.
- Compressed audio is decompressed (decoded) prior to presenting it to Flip64 for processing; direct-conversion of compressed audio is not supported. For instance, if an MP4 source has H264 video and AAC audio, the video can be direct-converted, but the audio can't be. Instead, the compressed audio is decoded in preparation for re-encoding by Flip64.
- Black segment insertion in Direct Convert mode is only supported in these formats: MPEG2 | H264 | H265 | XDCAM | IMX sources. See *Canvas*. Black segment insertion during conversion does not have format limits.
- Audio-only CML output is not supported.

Basic Audio Processing

The purpose of this topic is to familiarize you with the process of how the CML pre-processor processes audio, and the Flip64 CML elements associated with audio processing, by way of an example.

Note: The discussions in this topic focus on audio processing in Flip64 CML. Many elements have options not audio-related; refer to the reference topic for details.

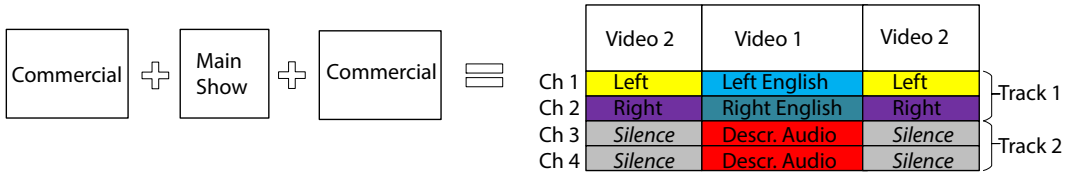
This example illustrates audio processing associated with TV show production.

In this example, you'll learn how to:

- Select audio channels from various source files and rearrange them as required for your output, when the audio is arranged differently than in the input files.
- Add audio to the clips on your timeline, as you create your output
- Define each track in the CML output for use in Flip64.

This show master has two stereo tracks—Spanish and English— which is combined with a commercial which has one stereo track, using only the show's English track for this production. The commercial is placed at the front and the back of the show. A descriptive audio (a single channel track) is also included for sight-impaired viewers.

Here is a visual depiction of the timeline and the desired structure of the output media for subsequent Flip64 ingest:



To define this media, follow these steps:

1. Specify the source assets, arranging audio channels as required.
2. Create a sequence of segments, to create the timeline.
3. Organize the sequence's audio into tracks.

The CML output—the resulting media—is passed to Flip64 for transcoding as required.

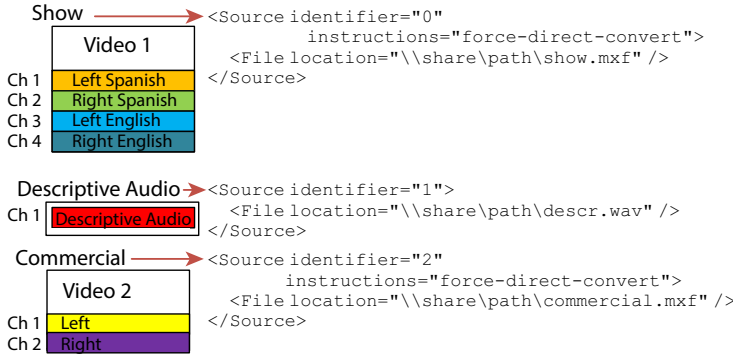
Specifying the Source Assets

Recall that a *Source* (each of which has a unique identifier) specifies a media file to be used in *Segment*s that comprise your *Sequence*. By default, the video, and all audio channels (in order) are used. If you require a subset of channels and/or require them in a different order, you must specify them in and their order. For each unique set of channels and/or order from the same file required, you create another *Source*.

Three source files are utilized in this example:

- The show master, with Spanish and English tracks
- The descriptive audio for the show
- The commercial being aired.

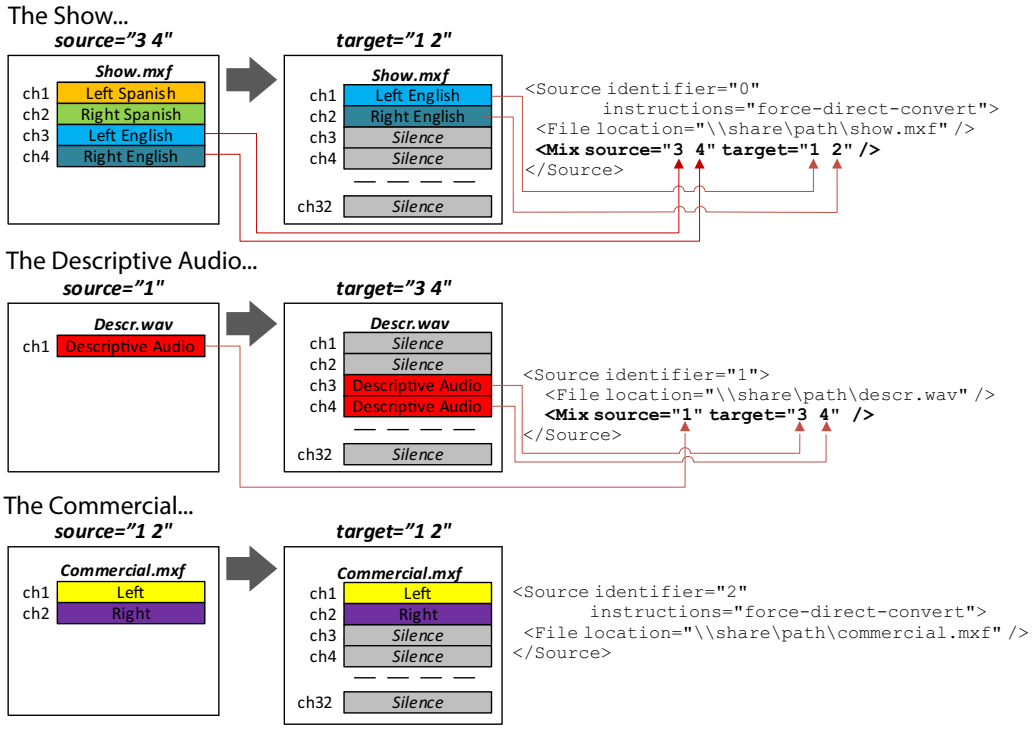
This graphic depicts each file's media, and the CML snippet that identifies the file:



Note that each *Source* has a unique integer ID, so you can add this *Source* to a *Segment*. Also, the video in *Source* 0 and 2 has direct-convert instructions, but the descriptive audio in *Source* 1—not having any video—lacks that instruction.

Selecting and Rearranging Audio Channels

However, the *Source* element is not yet complete. Video is included automatically without specification, but because you can't use the audio exactly as it exists in the sources, you need to identify the specific channels you're going to use and their new channel order. To do that, add *Mix* elements as shown here:



From *Source* 0, the *Mix* element specifies that only channels 3 and 4 should be utilized—as channel 1 and 2—in any *Segment* where this *Source* is used.

From *Source* 1, the *Mix* element specifies that the only channel—channel 1—should be utilized, but presented as channels 3 and 4—in any *Segment* where this *Source* is used. Thus, we have a dual mono track presented, from a single source channel.

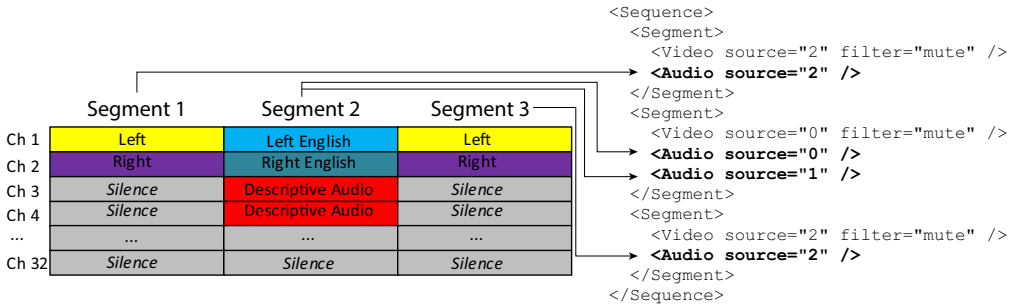
In *Source* 2, notice that there is no *Mix* element. By default, if all audio should be used, in the same channel order, no *Mix* is required. You could specify it (here, as `<Mix source="1 2" target="1 2" />`), but it's redundant.

Adding Sources to Segments on the Timeline

Now, you can create the *Segment*s that comprise your output (your *Sequence*) and specify the video and audio sources to use.

As you can see, the *Sequence* is comprised of three *Segments*. The *Segment* elements must be arranged in order, from top to bottom, to specify which *Segment* follows other

Segments on the timeline. *Segments*, unlike some other elements, do not have identities; their position in the CML dictates their position on the *Sequence* timeline.



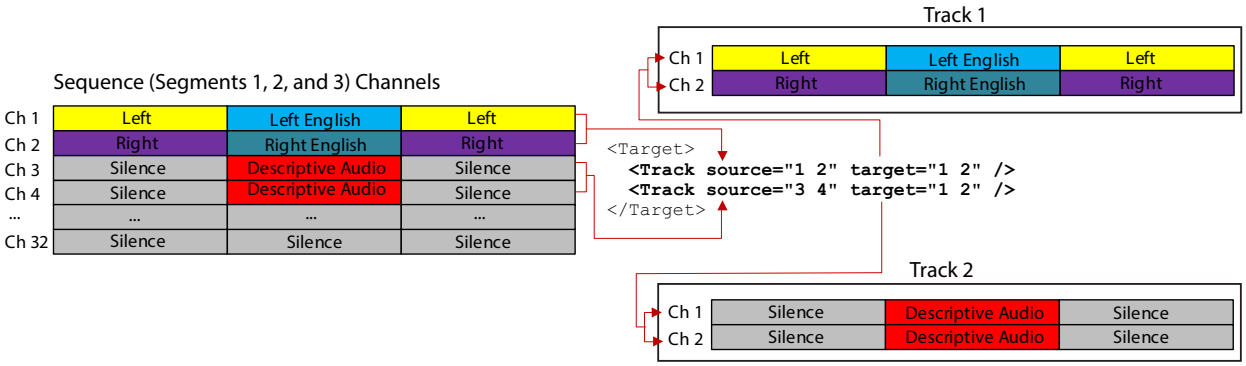
Next, for each *Segment*, add *Video* and *Audio* elements, as shown:

- The first *Segment* utilizes both the video and audio from *Source* 2—the commercial.
- The second *Segment* utilizes the video and audio from *Source* 0—the show—as well as the descriptive audio from *Source* 1.
- The third *Segment* repeats the first segment (*Source* 2)—the commercial again.

Organizing Channels into Tracks

In this last step, you organize all of channels at the *Sequence* level on the entire timeline into output tracks, for utilization in Flip64.

First, create one *Target*, and provide *Timecode* information.



Next, you create one *Track* in the *Target* for each track you want to present to Flip64. Referring to the *Target* CML code in the above illustration:

- The first *Track* (track 1) is comprised of the sequence's channels 1 and 2 (*source*), and they are placed in channels 1 and 2 (*target*).
- The second *Track* (track 2) is comprised of channels 3 and 4 (the *source* attribute), and they are placed in the same position in the output track: channels 1 and 2 (the *target* attribute).

(Timecode is dropped from the *Target* for clarity in this illustration.)

Conclusion

You've just defined a show with a commercial front and back where the show's audio is an English stereo track plus a descriptive, dual-mono audio track for use by sight-impaired viewers and the commercial's audio in its original form, using Flip64 CML,.

CML

```
<Composition xmlns="Telestream.Soa.Facility.Playlist">
  <Source identifier="0" instructions="force-direct-convert">
    <File location="\\share\path\show.mxf" />
    <Mix source="3 4" target="1 2" />
  </Source>
  <Source identifier="1">
    <File location="\\share\path\descr.wav" />
    <Mix source="1" target="3 4" />
  </Source>
  <Source identifier="2" instructions="force-direct-convert">
    <File location="\\share\path\commercial.mxf" />
  </Source>
  <Sequence>
    <Segment>
      <Video source="2" filter="mute" />
      <Audio source="2" />
    </Segment>
    <Segment>
      <Video source="0" filter="mute" />
      <Audio source="0" />
      <Audio source="1" />
    </Segment>
    <Segment>
      <Video source="2" filter="mute" />
      <Audio source="2" />
    </Segment>
  </Sequence>
  <Target>
    <Timecode type="start" time="01:00:00;00@29.97" />
    <Track source="1 2" target="1 2" />
    <Track source="3 4" target="1 2" />
  </Target>
</Composition>
```

Troubleshooting

During the design phase, it's typical to discover bugs in your CML. One excellent way of isolating the problem is to use the CML *Comment* element or XML comments <!-- and --> to surround portions of your CML to prevent it from executing.

As you design, develop and process Flip64 CML, you may encounter problems. Here are some common problems you may encounter:

- XML formatting—make sure that your CML conforms to XML syntax.
- Ignored elements and attributes—elements and attributes that are misspelled or capitalized incorrectly are ignored. Make sure keywords are spelled and capitalized correctly as well.
- Smart quotes—be sure to use straight double quotes to surround values.
- Segments missing or out of order—Segments are placed on the Sequence timeline, based on their ordinal position in the CML—their output order is not explicitly stated by an attribute value. Make sure your elements are spelled and capitalized correctly. Make sure your paths are correct.
- Cannot access files referenced in the CML—Only files located on Windows servers and accessible to the Vantage Transcode service can be used.
- CML vs Flip64—Determine whether the error is originating from the CML or in Flip64 directly.
- Audio channel problems—make sure that you are mixing the audio correctly and your Target is configured correctly, and Preserve Original Audio Tracks is enabled/disabled in Flip64 as required (see *Mix* for details).
- Verify stitching and audio—consider using Telestream Switch to review your output, verifying stitching and audio meters to check the audio on each of the channels and you don't have any A/V sync issues. Also verify your captions.

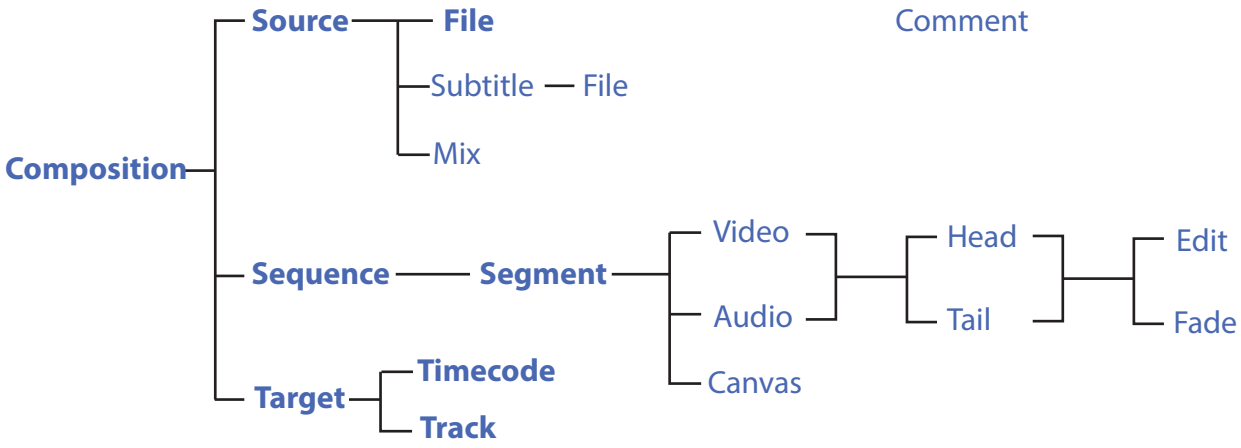
Flip64 CML Reference

The purpose of this chapter is to identify all elements in the Flip64 Composition Markup Language and their relationship, and describe each element in detail, including its attributes, along with simple examples to illustrate typical usage.

Flip64 CML Elements in Alphabetic Order

Flip64 CML Hierarchy Map

This tree structure illustrates the hierarchical relationship between the Flip64 CML elements, expressed as a tree. (Click on any element to display its reference topic.)



Elements in bold text represent elements which are required in a composition. *File* is required in *Source*, but optional in *Subtitle*.

Comment is an optional child of all elements. *Comments* are unique; you can use them in any element, and as many as you require.

Target, *Timecode*, and *Track* are required in any CML that includes *Audio*. However, if you are producing video-only CML, *Track* is not used.

Flip64 CML Implementation Details

These topics provide information about creating Flip64 CML files for processing in Flip64 workflows.

Case Sensitivity is Critical

XML is case-sensitive.

CML elements must be entered in title case—with the first letter in upper-case—for example, `Composition`, not `COMPOSITION` or `composition`—to be valid. If you don't capitalize an element (or you misspell it), it is ignored—and no error is displayed.

Although XML attributes are also by convention title cased, attributes in CML are all lower case.

Specifying File References in Flip64 CML

Files referenced in Flip64 CML must be located on Windows servers, because Flip64 only accesses files on Windows (NTFS) platforms. If your workflow is monitoring a server with a different operating system (HTTP, FTP, S3, etc.) for new files, you must provide a Copy or other transport action in the workflow to relocate the file on a Windows server, and it must be accessible to the Transcode service processing it.

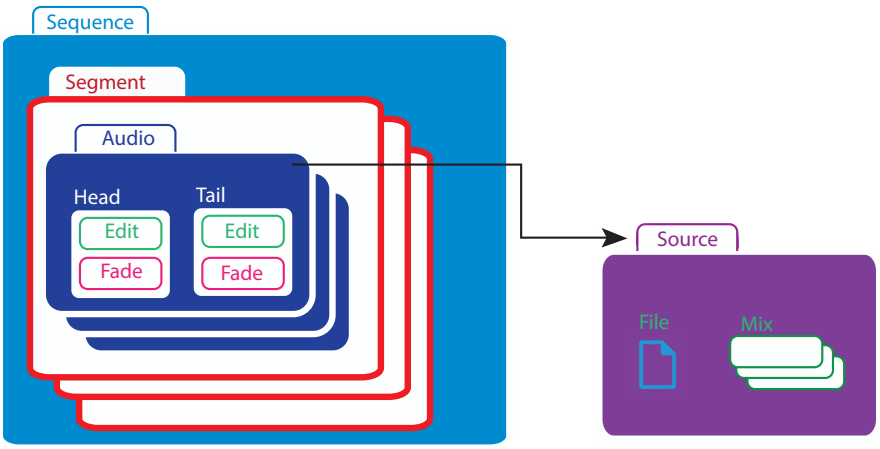
References to (potentially relocated) files may be specified using a Windows drive letter or a UNC path.

In the examples in this guide, files are typically shown as a UNC path (for example: *share* *path**myvideo.mov*) to reinforce the reliable use of shares to access network--based files by services operating in the Vantage domain.

Audio

Flip64 CML Elements in Alphabetic Order | Flip64 CML Hierarchy Map

Audio is a material element. You add one or more *Audio* elements to a *Segment* to include the audio channels specified in its *Source* (as arranged in its *Mix* elements, if present) as depicted here:



An *Audio* element identifies the *Source* via its *identifier*. By adding *Audio* to a *Segment,* you are placing the *Source's* channel(s) onto the timeline (accounting for optional trim points) in this *Segment*.

Note: Audio output from the CML pre-processor is always uncompressed 24-bit PCM. If you plan to utilize Flip64 to encode/re-organize the final output audio, use a *Target* in the CML to define a single audio *Track* (which will be used as a source in the Flip64 action). However, if you plan to configure the audio directly in the CML, you should define one or more *Tracks* as required. Then, configure the Flip64 action to propagate these tracks directly to the output by enabling *Preserve Original Audio Tracks* and using the Direct Convert codec.

You can optionally trim the *Audio* using *Head* and *Tail*.

In order to generate audio in the media, you must also utilize a *Target*. Without a *Target*, no audio is present in the output.

Note: For a comprehensive explanation of managing audio , see *Basic Audio Processing*.

See also *Video*.

Child Elements

Material items may be identified by two temporal parts: the beginning (*Head*) and the end (*Tail*), for the purpose of specifying the utilized length of the material by an *Edit* and—for *Audio* only—applying a *Fade*.

One each may be added to an *Audio* element:

- *Head*
- *Tail*

Attributes

Name	Description
source (required)	Integer value corresponding to the *Source* element's *identifier* value, to identify the file whose audio is to be used in this *Segment* of the output. Example: `<Audio source = "6" />`

Example

In this CML snippet, a *Source* > *File* (`6C_Audio_Overlay.mov`) is identified so that its media can be utilized in the *Segment*. The *Audio* in the *Segment* (`<Audio source="1">`) specifies that the audio from *Source* 1 (as configured by *Mix*) is included.

Note: This file has six channels of audio, but only the first two are utilized. However, all six are listed with the last four commented out for clarity, and possible use in other CMLs.

```
<Source identifier="1">
  <File location="\\share\path\6C_Audio_Overlay.mov" />
  <Mix source="1" target="1" />
  <Mix source="2" target="2" />
  <!-- Do not use tracks 2 and 3 in this production
  <Mix source="3" target="3" />
  <Mix source="4" target="4" />
  <Mix source="5" target="5" />
  <Mix source="6" target="6" />
  -->
</Source>
<Sequence>
  <Segment>
    <Video source="1" filter="mute" />
    <Audio source="1" />
...
```

Canvas

Flip64 CML Elements in Alphabetic Order | Flip64 CML Hierarchy Map

The *Canvas* element defines a visual rectangle (slate) the same size as the output frame, in a *Segment*. The *Canvas* element is a material element, which is auto-generated by Flip64, along with silent audio as appropriate, for the duration specified.

You can add black frames to output for headers and trailers, or to create gaps between clips for local ad insertion, for example. Black is the only color supported—you can't specify other colors.

However, unlike *Audio* and *Video*, you can't *Edit* or *Fade* a *Canvas*.

Segment insertion in direct convert mode is only supported in these independent frame formats: MPEG2 | H264 | H265 | XDCAM | IMX.

A *Canvas* element must be in its own *Segment* and may not be combined with any other material elements.

There are no child elements in *Canvas*.

Attributes

Name	Description
duration	The amount of time to display the canvas. Default: 0. If you are specifying a relative timecode for video with a timecode track, use these formats: HH:MM:SS:FF@FPS | HH:MM:SS:FF | HH:MM:SS;FF If you are specifying an absolute timecode or timestamp value, you can also use these formats if the material has a frame rate; otherwise you must use one that can be converted to a time: HH:MM:SS.sss | HH:MM:SS:FF@FPS Time is measured on a 24-hour clock. Time may include milliseconds (*sss*); frames (:|;FF) as DF (;) or NDF (:), and optional frame rate (FF@FPS), as an integer, decimal number, or rational (for example, 30000/1001). If FPS is not specified, it is inferred from the associated source. Rates supported: 23.976, 24, 25, 29.97, 30, 50, 59.94. Timecode references are applied to the timecode track specified in the associated Source element's file. Example: `<Canvas duration = "00:00:05.500" />`

Example

This *Segment* illustrates the typical use of a *Canvas* element. When you have a *Canvas* segment, you can only include *Canvas*—you can't add any other material.

```
...
<Segment>
  <Canvas duration="00:00:18.700" />
</Segment>
...
```

Comment

Flip64 CML Elements in Alphabetic Order | Flip64 CML Hierarchy Map

The optional *Comment* element can be placed in any element in the composition. The *Comment* element can span multiple lines.

The purpose of a *Comment* is to enable you to pass information (for example, a key-pair value or other metadata) into a workflow for use in the workflow, or to provide documentation for the composition. You can use *Comment* to provide in-line documentation to explain the purpose of the various elements of the Composition, which aid in development, troubleshooting, debugging and maintenance.

There are no child elements or attributes in a *Comment* element. You can place any text in the comment element that are valid value characters in XML; except that you can't place XML syntax in a *Comment*.

Note: You can't use *Comment* as a surrounding element for other XML elements that you want to disable. The workflow will fail with an error: "The <XYZ> start tag... does not match end Comment tag."
The *Comment* element is not an HTML | XML comment. To disable multi-line portions of the CML, you can use the multi-line XML comment elements <!-- and -->.

Example

In this *Sequence*, the comments (in bold) contain key pair values, which can be extracted in the Flip64 action or used elsewhere in the workflow as necessary.

```
<Composition xmlns="Telestream.Soa.Facility.Playlist">
  <Source identifier="1">
    <File location="\\share\path\My_TV_Show_Promo.mov" />
  </Source>
  <Sequence>
    <Segment>
      <Comment>TRP ID = urn:uuid:06a4-e204-407b-ac8b</Comment>
      <Comment>TRP Hash = emtXD7kMraxeTojicE6Ofva2HAc</Comment>
      <Comment>TRP Size = 224957607</Comment>
      <Video source="1" filter="mute" />
    </Segment>
  </Sequence>
...
```

To extract values from *Comment* elements, the best practice is to create variables and a metadata label in the Management Console. Then, use Metadata and Compute actions to extract the values, parse them as needed, and update a label with the variables. The label is automatically passed downstream where other actions can extract the appropriate values from the label as needed.

Composition

Flip64 CML Elements in Alphabetic Order | Flip64 CML Hierarchy Map

The *Composition* element identifies this XML-formatted text as a Composition object. It is the root element of every composition; one *Composition* element is required. The *Composition* element is the container for all other CML elements.

Child Elements

- *Source* (one or more; one required)
- *Sequence* (one; required)
- *Target* (one, required)

Attributes

Name	Description
created (optional)	String, a date/time stamp, for informational purposes only; not used in Vantage.
name (optional)	String, a logical or practical name for this composition XML file for informational purposes only; not used in Vantage.
version (optional)	String; a version number for informational purposes only; not used in Vantage.
xmlns	String; specifies utilized name spaces. Must include this namespace: xmlns="`Telestream.Soa.Facility.Playlist`". Omission results in an error. When using XML prefixes in CML, a namespace for each prefix must be defined.

Example

This example demonstrates the basic functionality of a *Composition*: a *Sequence* with a single *Segment* which includes *Video* and *Audio,* and a *Target*.

```
<Composition xmlns="Telestream.Soa.Facility.Playlist">
  <Source identifier="1">
    <File location="\\share\path\My_TV_Show_Promo.mov" />
  </Source>
  <Sequence>
    <Segment>
      <Video source="1" filter="mute" />
      <Audio source="1">
    </Segment>
  </Sequence>
  <Target>
    <Timecode type="source" />
    <Track source="1 2" target="1 2" />
  </Target>
</Composition>
```

Edit

Flip64 CML Elements in Alphabetic Order | Flip64 CML Hierarchy Map

The optional *Edit* element can be used in the *Head* and *Tail* of *Video* and *Audio*. The purpose of an *Edit* element is to create a smaller clip from a longer source—by specifying a mark-in or mark-out point—for this instance of the material. Used in the *Head*, it is the mark-in point; in the *Tail*, the mark-out point.

By default, the mark-in point of material is the beginning of the file, the mark-out point is the end of the file. Effectively, the *Edit* element moves the beginning or end of the clip from its default position to the point specified.

In any given instance (*Segment*) of the material, you can provide an *Edit* in either the *Head* or the *Tail*, or in both. Providing an *Edit* only in the *Head* results in a clip running from the *Head's Edit* point to the end of the file. Providing an *Edit* only in the *Tail* results in a clip running from the beginning of the clip to the *Edit* point in the *Tail*. In *both*, you have a clip trimmed at both the beginning and ending of the file.

Note: It is important to note that *Head Edits* are inclusive; *Tail Edits* are exclusive. That is, the first frame at the specified time in the *Head Edit* is *included* in the clip. However, the first frame at the specified time in the *Tail Edit* is *excluded*.
Also, *Edit* elements in *Video* only affect the video stream; to match the audio simultaneously, you must also apply the same *Edit* to the audio stream.

Bear in mind that a *Segment* is always as long as its longest material element. Thus, if you clip the *Head* of a *Video* but not the *Audio* (or vice versa) that references the same source, it may lose synchronization (lip sync).

Conversely, if you clip the *Tail* of a *Video* but not its *Audio*, the video is padded with black to match the *Audio* clip length. Conversely, if you clip the *Audio* but not the *Video*, the *Audio* is padded with silence.

There are no child elements in an *Edit* element.

Attributes

The combination of the *mode* and *time* attributes enables you to specify edit points in any manner you choose, depending on your media and your requirements.

Name	Description
mode (required)	Specifies how time or timecode (*absolute* and *relative* only) is applied to a head or tail. Keywords: *absolute* | *relative* | *duration*. *absolute*—used when a timecode is present in the source; identifies the edit point relative to the timecode. If you supply a timecode but the source lacks a timecode, it is converted to a time value and utilized. *relative*—specifies an edit point measured from the beginning (in a *Head*) or ending (in a *Tail*) of the source, irrespective of the timecode (if present). *duration*— specifies an edit point measured as a length of time, from the beginning (in a *Head*) or the ending (in a *Tail*) of the source. **Note:** You can't use *duration* in both *Head* and *Tail* in the same instance of material. Example: `<Edit mode = "relative" time = "00:10:00" />`

time (optional)	Specifies (by time or timecode) the location of the edit point. Drop frame, non-drop frame and time references are valid.
	A time is not valid when mode is set to *absolute*. A timecode is not valid when mode is set to *duration*.
	Examples:
	`<Edit mode = "absolute" time = "01:00:10" />`
	`<Edit mode = "absolute" time = "01:27;10" />`
	`<Edit mode = "relative" time = "00:00:10.000" />`
	`<Edit mode = "duration" time = "00:00:10.600" />`

Example

This *Segment* illustrates trimming the *Video's Tail* to a 7 second duration, using *Edit*.

Note: The `<Edit mode="relative" time="00:00:00.000" />` element is present for clarity. However, since it is *relative* (to the actual source, regardless of the timeline)—and the *time* is zero—the beginning (*Head*) of the *Video* is not modified.

```
<Segment>
  <Video source="1" filter="mute">
    <Head>
      <Edit mode="relative" time="00:00:00.000" />
    </Head>
    <Tail>
      <Edit mode="relative" time="00:00:07.000" />
    </Tail>
  </Video>
</Segment>
```

Fade

Flip64 CML Elements in Alphabetic Order | Flip64 CML Hierarchy Map

The *Fade* element applies a change in intensity to an *Audio* source, based on duration and shape. *Fade* is only valid for *Audio* material. You can't apply *Fade* to Dolby E audio.

The *Fade* element may be used in the *Head* and *Tail*; one is permitted. In a *Tail*, *Fade* is a fade out—it decreases in intensity; in a *Head*, a fade in—increases in intensity.

For each *Fade*, you can control the duration, and specify a shape.

Note: The duration of a *Fade* (or pair of *Fades*) can't be longer than the duration of the element to which it is being applied. For example, if you fade a *Head* and *Tail* for five seconds each on an eight-second audio clip, it will fail.

There are no child elements in a *Fade* element.

Attributes

Name	Description

duration	The amount of time to perform the fade. Default: 0.
	When specifying a relative timecode for video with a timecode track, use these formats:
	HH:MM:SS:FF@FPS | HH:MM:SS:FF | HH:MM:SS;FF
	If you are specifying an absolute timecode or time value, you can also use these formats if the material has a framerate; otherwise you must use one that can be converted to a time:
	HH:MM:SS.sss | HH:MM:SS:FF@FPS
	Time is measured on a 24-hour clock. Time may include milliseconds (*sss*); frames (:|;FF) as DF (;) or NDF (:), and optional frame rate (FF@FPS), as an integer, decimal number, or rational (for example, 30000/1001). If FPS is not specified, it is inferred from the associated source.
	Rates supported: 23.976, 24, 25, 29.97, 30, 50, 59.94.
	Timecode references are applied to the timecode track specified in the associated Source element's file.
	Examples:
	`<Fade duration = "00:00:05.500" />`
	`<Fade duration = "00:00:05;14@29.97" />`

Name	Description
shape (optional)	Keyword; specifies the shape of the interpolation. Keywords: *linear* | *s-curve* | *positive-3dB* | *negative-3dB* Default: *linear*. Example: `<Fade duration = "00:00:00.500" shape = "linear" />` *linear* (default)—Maintains a constant rate of change over the length of the fade. *s-curve*—Eases in and out of a fade with the midpoint at 0 dB. *positive-3dB*—Starts quickly; slowly tapers toward the end. *negative-3dB*—Starts slowly; quickly tapers toward the end.

Example

In this *Segment* code snippet, the *Head* and *Tail* of the *Audio* has a 500ms *Fade*. However, the *Head Fade* and *Tail Fade* use different shapes.

```
<Segment>
  <Video source="1" filter="mute">
  <Audio source="1">
    <Head>
      <Edit mode="relative" time="00:01:00.000" />
      <Fade duration="00:00:00.500" shape="positive-3dB" />
    </Head>
    <Tail>
      <Fade duration="00:00:00.500" shape="negative-3dB" />
    </Tail>
  </Audio>
</Segment>
```

File

Flip64 CML Elements in Alphabetic Order | Flip64 CML Hierarchy Map

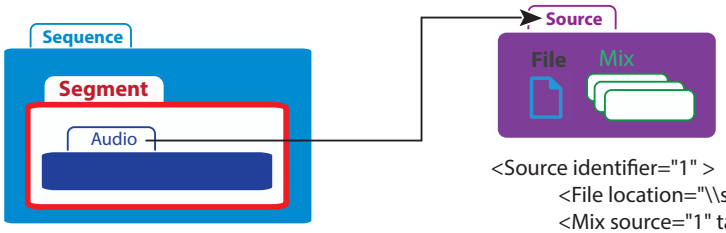
The *File* element specifies a fully-qualified path to a file included in a given *Source*. The path identifies the file so that its media may be utilized in a given *Segment*, by a *Video* or *Audio* element (or by a *Subtitle* element).

At least one *File* element is required in a *Source*. Multiple files are allowed—when you want to associate channels from separate files in a single source. You must identify each file whose media you plan to utilize in order to access it.

When utilized in a *Subtitle*, the *File* element is optional, and one is permitted. If subtitles are embedded, no *File* should be referenced.

Note: Files referenced in Flip64 CML must be on Windows servers, because Flip64 only accesses files on Windows (NTFS) platforms. If your workflow is watching for new files on a server with a different operating system (HTTP, FTP, S3, etc.), you must add a Copy or other transport action to relocate the file on a Windows server accessible to the Transcode service processing it on behalf of Flip64.

The association between a *Segment* and its *Source* is the *Video* or *Audio* element's source attribute in a *Segment* which matches the *Source* element's identifier attribute, as depicted here:




```
<Sequence>
  <Segment>
    <Video source="1" filter="mute" />
    <Audio source="1" />
  </Segment>
</Sequence>
```

```
<Source identifier="1" >
        <File location="\\share\Media\3StereoTracks.mxf" />
        <Mix source="1" target="3" />
        <Mix source="2" target="4" />
        <Mix source="3" target="5" />
        <Mix source="4" target="6" />
        <Mix source="5" target="1" />
        <Mix source="6" target="2" />
</Source>
```

Here, the *Audio* references *Source* `identifier 1`, where the *File* is `\\share\Media\3StereoTracks.mxf`, and six *Mixes* are specified to re-arrange the three stereo pairs as required for output: Now, this *Segment's Audio* is the set (and order) of channels specified by *Mix* elements in the *Source's File* element.

There are no child elements in a *File* element.

Attributes

This attribute is required for a *File* element.

Name	Description
location	Specifies the fully-qualified path to a file on a Windows (NTFS) server, including file name. The location must be accessible to the Transcode service which is executing the Flip64 action. You may use a drive letter or a UNC path. Locations may be specified as: • *Drive Letter*—`D:\pathname\filename.ext` • *UNC Path*—`\\share\path\filename.ext` Shares are recommended to ensure access by Vantage services, especially in an array where the service that executes an action may change from job to job.

Example

This example illustrates using a *File* element to specify two different *Source* files. Typically, one is specified, but in this example, audio tracks from two different files are required in the output. The *Mix* instructions are ignored for this example.

```
<Composition>
  <Source identifier="1" instructions="no-direct-convert">
    <File location="\\share\path\Mystic_River_Music.mov" />
    ...
  </Source>
  <Source identifier="2">
    <File location="\\share\path\Mystic_River_Voice_Over.mov" />
    ...
  </Source>
...
```

Head

Flip64 CML Elements in Alphabetic Order | Flip64 CML Hierarchy Map

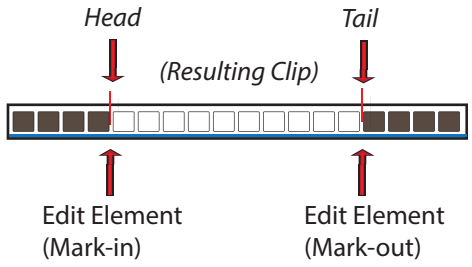
The *Head* element is the point (the first frame) in a *Segment* of *Audio* and *Video* material at the beginning (or mark-in point, when using an *Edit*).

The corresponding *Tail* element is the point (or frame) on a media element at the end (or mark-out point, when using an *Edit*) of the instance. *Tails*—as well as *Heads*—by default have no duration; they represent either edge of the material.

The purpose of adding *Head* to *Audio* is to perform effects—clipping and/or fading—*Edit* changes the start time (typically in sync with video); *Fade* controls the audio intensity. In *Video*, clipping is also supported, but fading is not. Otherwise, the *Head* is not required.

In this example, *Head* and *Tail* have no *Edit* elements; thus the *Head* is the beginning timecode; and *Tail* is the end timecode of the material:

In this example, *Head* and *Tail* each have *Edit* elements; thus the *Head* is equal to the timecode of its *Edit*; *Tail* is also equal to its *Edit*, and the resulting clip is all that exists in the output media:



There are no attributes in a *Head* element.

See also *Tail*.

Child Elements

One each of these elements may be optionally added to a *Head* or *Tail* element:

- *Edit* (*Audio* and *Video*)
- *Fade* (*Audio* only)

Example

In this *Segment*, the media has both video and audio. The *Video* is clipped in the beginning (*Head* > *Edit*), along with the *Audio*. The *Segment* starts at 3 seconds into the clip, with a short *Fade* in and plays to the original end, since there is no *Tail* > *Edit*.

```
<Segment>
  <Video source="1" filter="mute" >
    <Head>
      <Edit mode="relative" time="00:00:03.000" />
    </Head>
  </Video>
  <Audio source="1">
    <Head>
      <Edit mode="relative" time="00:00:03.000" />
      <Fade duration="00:00:00.020" shape="linear" />
    </Head>
  </Audio>
</Segment>
```

Mix

Flip64 CML Elements in Alphabetic Order | Flip64 CML Hierarchy Map

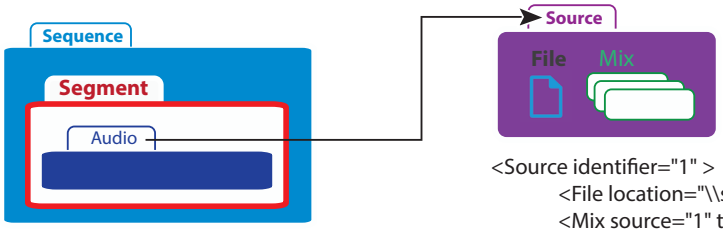
You apply one or more *Mix* elements to a *Source* to identify which audio channels to utilize from the associated *File*, and how to order them (which may be identical or re-arranged).

You should use a *Mix* element when you want to utilize only certain channels from a file and/or reorganize them. By default, if you do not specify a *Mix*, all channels in the file are utilized, in order. For video-only output, do not use *Mix*.

You can optionally adjust the audio gain and also apply a phase shift on a channel-by-channel basis. You can also create silent channels.

Note: By default, if there are no *Mix* elements in a *Source*, then all channels in the associated *File* are placed into the *Segment* in the same order. Thus, if you have two or more *Audio* elements —but no *Mix* elements in the specified *Source*, each succeeding *Audio* automatically overlays the previous channels with its own.

Use *Mix* elements to select specific channels and optionally, re-order them:




```
<Sequence>
  <Segment>
    <Video source="1" filter="mute" />
    <Audio source="1" />
  </Segment>
</Sequence>
```

```
<Source identifier="1" >
        <File location="\\share\Media\3StereoTracks.mxf" />
        <Mix source="1" target="3" />
        <Mix source="2" target="4" />
        <Mix source="3" target="5" />
        <Mix source="4" target="6" />
        <Mix source="5" target="1" />
        <Mix source="6" target="2" />
</Source>
```

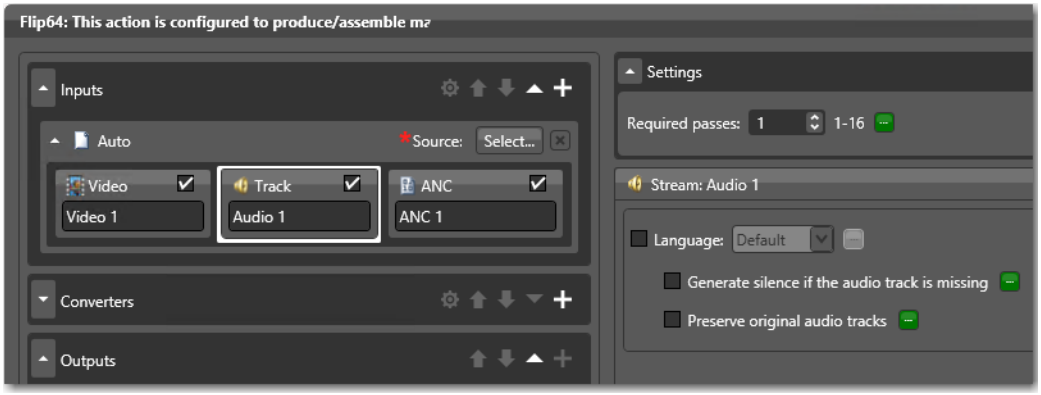
Here, the *Audio* references *Source* `identifier 1`, where the *File* is `\\share\Media\3StereoTracks.mxf`. Six *Mixes* are specified to re-arrange the three stereo pairs as required for output. Now, this *Segment's* audio is the set of channels specified by *Mix* elements in the *Source's File*, as ordered in the *Segment*.

Note: For a comprehensive example of how to manage audio in Flip64 CML, see *Basic Audio Processing*.

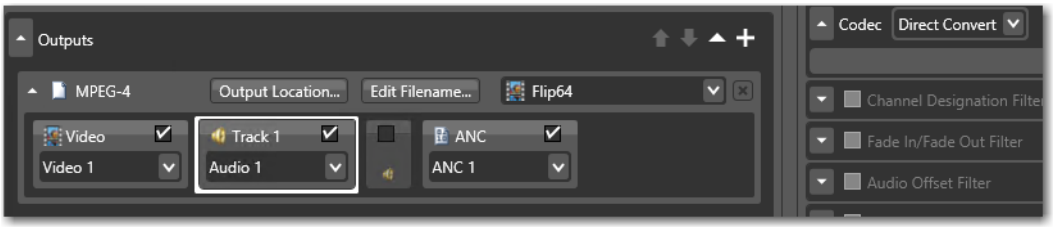
Note: Audio output from the CML pre-processor is always uncompressed 24-bit PCM. If you plan to utilize Flip64 to encode/re-organize the final output audio, use a *Target* in the CML to define a single audio *Track* (which will be used as a source in the Flip64 action). However, if you plan to configure the audio directly in the CML, you should define one or more *Tracks* as required. Then, configure the Flip64 action to propagate

these tracks directly to the output by enabling *Preserve Original Audio Tracks* and using the Direct Convert codec.

The *Preserve Original Audio Tracks* control in the Flip64 inspector dialog is displayed in the Auto Input's audio track component:



Select the *Direct Convert* Codec in your output Track components:



There are no child elements in a *Mix* element.

Attributes

Name	Description

level (optional)	A real value (default: 1.0), specifying the audio level in unit gain, as a percent of the current value or in decibels (dB). When no unit of measure is provided, the value is the multiplier where 1 (100%) is no change; a value less than 1 reduces the current value by that percentage and a value greater than 1 increases the value in similar manner. If you set level to 0, the output is silence.
	When specifying dB, where 0 is no change, a negative value reduces the current value by the specified amount; a positive value increases it by that amount.
	Examples:
	`<Mix level=".707"... />`
	Here, the current value is multiplied by 0.707, thus reducing the value relatively, by 29.3 percent.
	`<Mix level="3dB"... />`
	Here, 3 is added to the current value, raising the value absolutely, by 3 decibels.

phase (optional)	A numeric value specifying the phase shift in degrees. (The degree symbol is not required.) A negative number rotates clockwise, and a positive number rotates counter-clockwise. The default value is 0.0 degrees. Phase may be used to implement surround sound matrix encoding (+/-90°) or to correct phase inversions (+/-180°). Example: `<Mix phase="-90"... />`

source (required)	Integer; one or more (separated by a space) specifies the channel(s) (beginning at 1) or other key word to utilize from the source file.
	Keywords: *1* through *32* | *All*
	1...*32*—the channel number or numbers to utilize.
	All (default)—use to specify that all source channel(s) should be utilized in order, based on (and limited by) the target channels.
	Single source channel example: `<Mix source="1" target="5" />`
	Here, channel 1 from the source is placed into channel 5.
	All source and target example: `<Mix source="All" target="All" />`
	Here, all source channels (up to 32) in order, are placed into the same target channels.
	You can optionally, define multiple channels in a single *Mix*, by separating each with a space, as shown here:
	Example: `<Mix source="1 2" target="5 6" />`
	This is functionally equivalent to:
	`<Mix source="1" target="5" />`
	`<Mix source="2" target="6" />`

Name	Description
target (required)	Integer; one or more (separated by a space) specifies the ordinal index—channel number—of the corresponding channel(s) specified in the *source* attribute. Keywords: *1* through *32* | *All* *1*...*32*—the channel number or numbers to utilize. *All* (default)—Specifies to present the specified source channel(s) in order, exactly as specified, repeating as necessary (if there are less than 32 source channels) to fill all 32 channels. Example: `<Mix source="3" target "1" level=".9"/>` In this example, channel 3 in the source is identified as channel 1. All Target Example: `<Mix source="1 2" target="All" />` Here, presents source channels 1 and 2 in order, into successive, repeating target channels, resulting in 1 2 1 2 1 2... for 16 pairs in the 32 channel map.

Examples

In this example Source snippet, the source MXF file has six channels comprising three stereo tracks of various languages, which must be re-ordered for use in *Segments*. The first track is placed in track 3 (channels 5 and 6), the second track is moved to track 1, and the third track is moved to track 2—all using channel identifiers; *Source* does not operate on tracks.

```
<Composition "xmlns="Telestream.Soa.Facility.Playlist">
  <Source identifier="0" >
    <File location="\\share\media\3LangTracks.mxf" />
    <Mix source="1" target="5" />
    <Mix source="2" target="6" />
    <Mix source="3" target="1" />
    <Mix source="4" target="2" />
    <Mix source="5" target="3" />
    <Mix source="6" target="4" />
  </Source>
```

In this second snippet, the *Source* identifier=1 file has a Dolby 5.1 track and one stereo track. The *Source* `identifier="2"` MXF file has three stereo tracks. Only the stereo track from *Source* `identifier="1"` and the third track of *Source* `identifier="2"` are placed into a *Segment*.

```
<Composition "xmlns="Telestream.Soa.Facility.Playlist">
  <Source identifier="0" instructions="no-direct-convert">
    <File location="\\share\media\video.mxf" />
    <Subtitle preserve708="false" />
  </Source>
  <Source identifier="1" >
    <File location="\\share\Media\Dolby_and_Stereo.mxf" />
    <Comment>Dolby 5.1 track unused</Comment>
    <Mix source="7" target="1" />
    <Mix source="8" target="2" />
  </Source>
```

```
<Source identifier="2" >
  <File location="\\share\Media\3StereoTracks.mxf" />
  <Comment>English and French tracks unused</Comment>
  <Comment>Track 3 Portuguese </Comment>
  <Mix source="5" target="3" />
  <Mix source="6" target="4" />
</Source>
<Sequence>
  <Segment>
    <Video source="0" filter="mute" </Video>
    <Audio source="1" </Audio>
    <Audio source="2" </Audio>
  </Segment>
</Sequence>
```

Segment

Flip64 CML Elements in Alphabetic Order | Flip64 CML Hierarchy Map

At least one *Segment* must be included in a *Sequence*; multiple are permitted. A *Segment* is comprised of material items (*Video* | *Audio* | *Canvas*). Each *Segment* may have one *Video* stream and up to 32 channels in one or more *Audio* elements. Alternatively, it may contain *Canvas*, in which case *Video* and *Audio* may not be included.All material in a *Segment* is played simultaneously (has the same timecode).

Segment insertion in direct convert mode is only supported in these independent frame formats: MPEG2 | H264 | H265 | XDCAM | IMX.

The duration of a *Segment* is determined by the duration of the longest material item.

Segments organize their material ordinally along the timeline relative to other *Segments*. The order of *Segment* elements dictates the organization of clips.

There are no attributes in a *Segment* element.

Child Elements

At least one material must be present in a *Segment*. *Audio* and *Video* may be present together; if *Canvas* is present, no other material is permitted.

- *Audio*
- *Video*
- *Canvas*

Example

This *Segment* snippet illustrates the typical use of a *Segment*; video and audio are presented from the same source. Notice the required `mute` in *Video* for correct processing in Flip64. The video and audio are both trimmed using the same values:

```
<Segment>
  <Video source="0" filter="mute">
    <Head>
      <Edit mode="absolute" time="01:00:00;03@29.97" />
    </Head>
    <Tail>
      <Edit mode="absolute" time="01:00:05;25@29.97" />
    </Tail>
  </Video>
  <Audio source="0">
    <Head>
      <Edit mode="absolute" time="01:00:00;03@29.97" />
    </Head>
    <Tail>
      <Edit mode="absolute" time="01:00:05;25@29.97" />
    </Tail>
  </Audio>
</Segment>
```

Sequence

Flip64 CML Elements in Alphabetic Order | Flip64 CML Hierarchy Map

A *Sequence* is used to organize media (material elements) on an independent timeline, using one or more *Segment* elements, which contain the media streams. It's often used to join multiple material elements ordinally.

One *Sequence* element is required in a *Composition*. There are no attributes in a *Sequence*.

Child Elements

One or more *Segment*s must be added to a *Sequence* element.

Example

In this example, the *Sequence* contains three *Segments*: the first has a *Video* and an *Audio* stream, the second is black (*Canvas*), and the third is a *Video* and *Audio* segment.

Note: The `<Edit mode="relative" time="00:00:00.000" />` element is placed here for clarity. However, since it is *relative* (to the actual source, regardless of the timeline)—and the *time* is zero—the beginning (*Head*) of the *Video* and *Audio* is NOT modified.

```
<Sequence>
  <Segment>
    <Video source="0" filter="mute">
      <Head>
        <Edit mode="relative" time="00:00:00.000" />
      </Head>
    </Video>
    <Audio source="0">
      <Head>
        <Edit mode="relative" time="00:00:00.000" />
        <Fade duration="00:00:00.020" shape="linear" />
      </Head>
      <Tail>
        <Fade duration="00:00:00.020" shape="linear" />
      </Tail>
    </Audio>
  </Segment>
  <Segment>
    <Canvas duration="00:00:10.010" />
  </Segment>
  <Segment>
    <Video source="1" filter="mute">
      <Head>
        <Edit mode="relative" time="00:00:00.000" />
      </Head>
      <Tail />
    </Video>
```

```
    <Audio source="1">
      <Head>
        <Edit mode="relative" time="00:00:00.000" />
        <Fade duration="00:00:00.020" shape="linear" />
      </Head>
      <Tail>
        <Fade duration="00:00:00.020" shape="linear" />
      </Tail>
    </Audio>
  </Segment>
</Sequence>
...
```

Source

Flip64 CML Elements in Alphabetic Order | Flip64 CML Hierarchy Map

A *Source* specifies a media file to be used in *Segments* that comprise your *Sequence*. Each *Source* includes a unique identifier, the file, and optionally, audio channel selection and organization, and subtitle information.

The unique identifier is used to include the *Source* in *Segments*, for placement in the output. *Source* elements may be in any order.

The video, and all audio channels (in order) are available. If you plan to use the video, you must specify how to process it, via the *instruction* attribute.

By default, all audio channels are usable, in the order in the file. If you require a subset of audio channels and/or require the audio channels in a different order, you must specify the channels and channel order in a *Mix*. Each unique set of audio channels and/or order requires a different *Source*. You may specify multiple *Source*s using the same *File*.

You can also control captioning propagation.

At least one *Source* must be defined in your *Composition*; one or more *Source* elements may be included in a given *Segment*. *Source* elements by convention (but not required) are typically listed first in a *Composition*.

Child Elements

Each of these elements are child elements to a *Source*:

- *File*—one required for each *Source* element
- *Mix* elements are optional
- *Subtitle*—One permitted; optional.

Attributes

Name	Description
identifier	An integer value, which must be unique among all *Source* elements. Example: `<Source identifier="1">` This identifier value is used in the *source* attribute of *Video* | *Audio* elements to reference and utilize media from this file.

Name	Description
instructions (optional)	Keywords which specify whether the *Source* file's video should or should not be direct-converted for ingest by the Flip64 transcoder. This attribute parallels the encoder setting in the Flip64 action. This attribute is ignored for audio-only Source. Keywords: *force-direct-convert* | *no-direct-convert* Example: `<Source identifier="1" intructions="force-direct-convert">` The keyword *force-direct-convert* is required in all *Source* instances that utilize video, when the video encoder in Flip64 is set to *Direct Convert*. The video stream is not transcoded; it is passed through without decoding into Flip64. However, stitching clips (*Edits*) of temporally-compressed video (MPEG-2|x265 etc.) results in edge re-encoding to repair GOPs. If Flip64 is configured to re-encode video, *no-direct-convert* may optionally be specified for clarity.

Examples

This example illustrates the typical use of a *Source* element to identify the file providing the *Video* in a *Segment*, including *Subtitle* processing. In this example, there is no audio in the output:

```
<Source identifier="1">
  <File location="\\share\path\My_TV_Show_Promo.mov" />
  <Subtitle preserve708="false" />
</Source>
<Sequence>
  <Segment>
    <Video source="1" filter="mute" />
  </Segment>
</Sequence>
```

The second example illustrates the typical use of a *Source* element to identify the file providing *Audio* from a separate file. The *Audio* has been added to the *Segment*. (*Target* element not shown, but required):

Note: The `<Edit mode="relative" time="00:00:00.000" />` element is placed here for clarity. However, since it is *relative* (to the actual source, regardless of the timeline)—and the *time* is zero—the beginning (*Head*) of the *Audio* is NOT modified.

```
<Source identifier="1">
  <File location="\\share\path\TV_Show.mov" />
  <Subtitle preserve708="false" />
</Source>
<Source identifier="2">
  <File location="\\share\path\6MonoTracks.mxf" />
  <Mix source="1" target="1" />
  <Mix source="2" target="2" />
```

```
    <Mix source="3" target="3" />
    <Mix source="4" target="4" />
    <Mix source="5" target="5" />
    <Mix source="6" target="6" />
  </Source>
  <Sequence>
    <Segment>
      <Video source="1" filter="mute" />
      <Audio source="2">
        <Head>
          <Edit mode="relative" time="00:00:00.000" />
          <Fade duration="00:00:00.020" shape="linear" />
        </Head>
        <Tail>
          <Fade duration="00:00:00.020" shape="linear" />
        </Tail>
      </Audio>
    </Segment>
```

Subtitle

Flip64 CML Elements in Alphabetic Order | Flip64 CML Hierarchy Map

Optionally, you can add one *Subtitle* element to a *Source*. The purpose of the *Subtitle* element is to propagate closed captions for the specified file on a source-by-source basis.

Without specifying a separate SCC file via a *File* element in *Subtitle*, you use embedded captions in this *Source*'s file to propagate subtitles. You can also optionally preserve 708 captions; see the *preserve708* attribute, following.

Note: *Subtitle* enables subtitle processing. If subtitles are present in the source, they are propagated to Flip64. Whether Flip64 will propagate them to its output depends on how Flip64 is configured. If Flip64 is configured to propagate subtitles—but *Subtitle* is not specified in the CML—they will not be propagated; it would be the same as processing a media file that doesn't contain subtitles.

The *Subtitle* element optionally uses a *File* element to identify a separate—side car—closed caption (.scc) file. If the media and the SCC file are not already timecode-aligned, they can be aligned in *Subtitle* directly, using the *offset* attribute.

SCC file supported frame rates: 23.976, 29.970, and 59.94 fps.

See also *File*.

Child Elements

One *File* can be added to a *Subtitle* element, to utilize embedded captions from a side car file instead of the embedded captions in the *Source* file itself.

Examples

In this example, the SCC file has a start timecode one hour later than the media file, which must be reconciled. In addition, the media is edited to have an in point one minute into the clip, which must also be taken into consideration.

Here are the details:

- Media start timecode 00:00:00;00
- SCC file start timecode 01:00:00;00
- In point edit 00:01:00;00 (<Edit time="00:01:00;00" />)
- Offset 01:00:00;00
- Result 01:01:00;00

The SCC file's offset value is added to the timecode of the media file's in point before it is applied to the SCC file. The result is that the edit point in the SCC file specifies the frame that is one minute into the SCC file, so that it aligns with the media:

00:01:00;00 + 01:00:00;00 = 01:01:00;00

In a second example, the media file has a one hour start time, but the SCC file starts at zero. The media file also has the in point one minute into the clip, as in the first example.

Here are the details:

- Media start timecode 01:00:00;00
- SCC file start timecode 00:00:00;00
- In point edit 01:01:00;00 (<Edit time="00:01:00;00" />)
- Offset 23:00:00;00
- Result 00:01:00;00

In this example, a 23-hour offset is added to the specified in point to create a matching in point in the SCC file at 0 hours (as in a 24-hour clock, where 2400 is actually 0000 hours):

01:01:00;00 + 23:00:00;00 = 00:01:00;00

Attributes

These attributes may be applied to a *Subtitle* element.

Name	Description
offset (optional)	The amount of time required to align non-matching timecode references of source media and the closed caption file. The offset value is added to the media timecode reference to adjust for the difference. The offset has a range of 24 hours. Default: `"00:00:00.000"`. If you are specifying a relative timecode for video with a timecode track, use these formats: HH:MM:SS:FF@FPS | HH:MM:SS:FF | HH:MM:SS;FF If you are specifying an absolute timecode or time value, you can also use these formats if the material has a framerate; otherwise you must use one that can be converted to a time: HH:MM:SS.sss | HH:MM:SS:FF@FPS Time is measured on a 24-hour clock. Time may include milliseconds (*sss*); frames (:|;FF) as DF (;) or NDF (:), and optional frame rate (FF@FPS), as an integer, decimal number, or rational (for example, 30000/1001). If FPS is not specified, it is inferred from the associated source. Rates supported: 23.976, 24, 25, 29.97, 30, 50, 59.94. Timecode references are applied to the timecode track specified in the associated Source element's file. Example: `<Subtitle offset = "01:43:05.333"... />`

Name	Description
preserve708 (optional)	Controls whether 708 captions are propagated or not, on a per-source basis. However, if there are no 708 captions in the source, 608 captions are transcoded, regardless of this setting. Default: `preserve708="false"`. If preserve708 is set to *false* or not specified, then the source 708 caption track (if present) is overridden and the output 708 caption track is transcoded from the 608 caption track. You can modify this behavior by setting preserve708 to true: `<Subtitle preserve708="true" />` to retain any 708 captions in the source. In this case, if a 708 caption track is present in the source, then the 708 caption track is propagated in the output without changes.

Examples

Here are two examples—one using sidecar file-based captions, and one using embedded captions from the *Source* file itself.

Basic Subtitle with a Sidecar SCC File

```
<Composition>
  <Source identifier="1">
    <File location="\\share\path\Chronicle_St_Lucia_TC.mov" />
    <Subtitle>
      <File location="\\share\path\Chronicle_St_Lucia_TC.scc" />
    </Subtitle>
  </Source>
  <Sequence>
    <Segment>
      <Video source="1" filter="mute" />
    </Segment>
...
```

Propagating Embedded Captions

Another aspect of the *Subtitle* element is that it will propagate embedded captions from the source file if no file attribute is set, as shown in this *Source*:

```
  <Source identifier="1">
    <File location="\\share\path\bxcy_420.mxf" />
    <Subtitle />
  </Source>
```

Tail

Flip64 CML Elements in Alphabetic Order | Flip64 CML Hierarchy Map

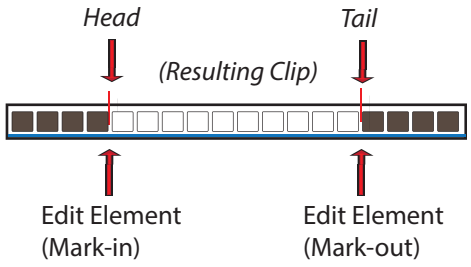
The *Tail* element is the point (the last frame + 1) in a *Segment* of *Audio* and *Video* material at the end (or mark-out point, when using an *Edit*).

The corresponding *Head* element is the point (or frame) at the beginning (or mark-in point, when using an *Edit*) of the instance. *Tails*—as well as *Heads*—have no duration; they represent either edge of the material.

The purpose of adding a *Tail* to *Audio* is to perform effects—clipping and/or fading—*Edit* changes the end time (typically in sync with video); *Fade* controls the audio intensity. In *Video*, clipping is also supported, but fading is not. Otherwise, the *Tail* is not required.

In this example, *Head* and *Tail* have no *Edit* elements; thus the *Head* is the beginning timecode; then *Tail* is the end timecode of the material:

In this example, *Head* and *Tail* each have *Edit* elements; thus the *Head* is equal to the timecode of its *Edit*; *Tail* is also equal to its *Edit*, and the resulting clip is all that exists in the output media:



There are no attributes in a *Tail* element.

See also the corollary element, *Head*.

Child Elements

One each of these elements may be optionally added to a *Head* or *Tail* element:

- *Edit* (*Audio* and *Video*)
- *Fade* (*Audio* only)

Example

In this example, the *Segment* is 30 seconds long, starting at the beginning of the clip (since there is no *Head* > *Edit*), with an audio fade out.

```
<Segment>
  <Video source="1" filter="mute" >
    <Tail>
      <Edit mode="relative" time="00:00:30.000" />
    </Tail>
  </Video>
  <Audio source="1">
    <Tail>
      <Edit mode="relative" time="00:00:30.000" />
      <Fade duration="00:00:00.020" shape="linear" />
    </Tail>
  </Audio>
</Segment>
```

Target

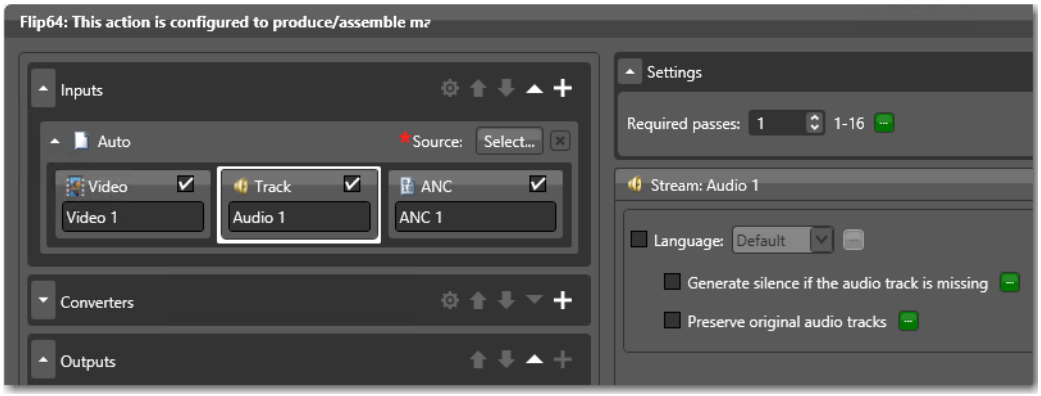
Flip64 CML Elements in Alphabetic Order | Flip64 CML Hierarchy Map

One *Target* element is required in a *Composition* when the output media contains audio. If you are creating video-only output, you should only include *Target* to override the timecode.

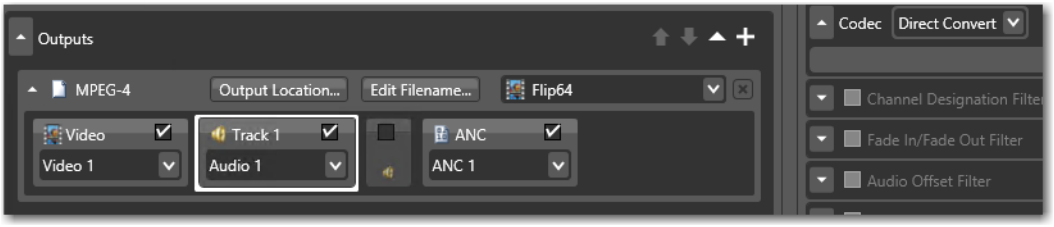
The *Target* element utilizes *Track* elements to organize the *Sequence's* collective set of audio channels into tracks for Flip64, and also to control the output timecode via the *Timecode* element.

Note: Audio output from the CML pre-processor is always uncompressed 24-bit PCM. If you plan to utilize Flip64 to encode/re-organize the final output audio, use a *Target* in the CML to define a single audio *Track* (which will be used as a source in the Flip64 action). However, if you plan to configure the audio directly in the CML, you should define one or more *Tracks* as required. Then, configure the Flip64 action to propagate these tracks directly to the output by enabling *Preserve Original Audio Tracks* and using the Direct Convert codec.

The *Preserve Original Audio Tracks* control in the Flip64 inspector dialog is displayed in the Auto Input's audio track component:



Select the *Direct Convert* Codec in your output Track components:



There are no attributes in a *Target*.

Child Elements

- *Track* (required when media includes audio; multiple permitted. For video-only media, do not include.)
- *Timecode* (optional; one permitted)

Example

In this composition, the media generated by the CML pre-processor contains two stereo tracks:

```
<Composition "xmlns="Telestream.Soa.Facility.Playlist">
  <Source identifier="0" instructions="no-direct-convert">
    <File location="\\share\Media\video.mxf" />
    <Subtitle preserve708="false" />
  </Source>
  <Source identifier="1" >
    <File location="\\share\Media\Dolby_and_Stereo.mxf" />
    <Mix source="7" target="1" />
    <Mix source="8" target="2" />
  </Source>
  <Source identifier="2" >
    <File location="\\share\Media\2StereoTracks.mxf" />
    <Mix source="5" target="3" />
    <Mix source="6" target="4" />
  </Source>
  <Sequence>
    <Segment>
      <Video source="0" filter="mute" </Video>
      <Audio source="1" </Audio>
      <Audio source="2" </Audio>
    </Segment>
  </Sequence>
  <Target>
    <Timecode type="source" />
    <Track source="1 2" target="1 2" />
    <Track source="3 4" target="1 2" />
  </Target>
</Composition>
```

Timecode

Flip64 CML Elements in Alphabetic Order | Flip64 CML Hierarchy Map

One *Timecode* is optional in a *Target*; you use it to specify the timecode of the generated media. You can override the source timecode or you can propagate the source timecode to the output.

When *Timecode* is used, the Timecode filter in Flip64 should be disabled or set to *Use Source Timecode*. In this manner, the *Timecode* specification is propagated directly to the final Flip64 output.

There are no child elements of *Timecode*.

See also*Target*.

Attributes

These attributes must be applied to a *Timecode* element.

Name	Description
type	Keyword; specifies the type of timecode: *source* | *start*. The *source* keyword propagates the existing timecode from the source into the output without changing its value. Example: `<Timecode type="source" />` The *start* keyword forces an override of the time code if it exists; you must specify the *time* attribute and supply the starting timecode value. Example: `<Timecode type="start" time="01:00:00;00@29.97" />`
time	Specifies the starting timecode to utilize, when Timecode type = start is specified. If you specify a relative timecode for video with a timecode track, use these formats: HH:MM:SS:FF@FPS | HH:MM:SS:FF | HH:MM:SS;FF If you specify an absolute timecode or timestamp value, you can also use these formats if the material has a frame rate; otherwise you must use one that can be converted to a time: HH:MM:SS.sss | HH:MM:SS:FF@FPS Time is measured on a 24-hour clock. Time may include milliseconds (*sss*); frames (:|;FF) as DF (;) or NDF (:), and optional frame rate (FF@FPS), as an integer, decimal number, or rational (for example, 30000/1001). If FPS is not specified, it is inferred from the associated source. Rates supported: 23.976, 24, 25, 29.97, 30, 50, 59.94. Timecode references are applied to the timecode track specified in the associated Source element's file. Example: `<Timecode type="start" time="01:00:00;00@29.97"/>`

Example

In this *Target* element, the timecode for the output is specified, ignoring and overriding any timecode that may exist in the input.

```
<Target>
  <Timecode type="start" type="01:00:00;00@29.97" />
  <Track source="1 2" target="1 2" />
  <Track source="3 4" target="1 2" />
  <Track source="5 6" target="1 2" />
</Target>
```

Track

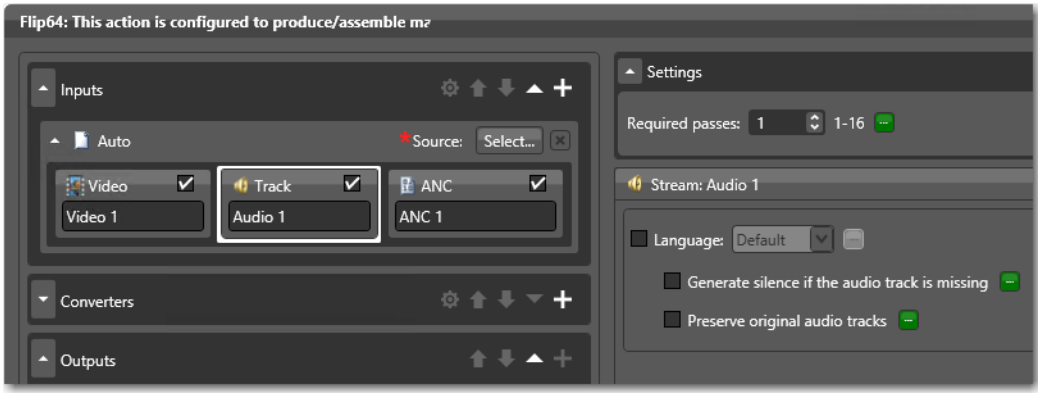
Flip64 CML Elements in Alphabetic Order | Flip64 CML Hierarchy Map

One or more *Track* elements must be included in a *Target* for media that includes audio. For video-only output, *Track* is irrelevant and should be omitted.

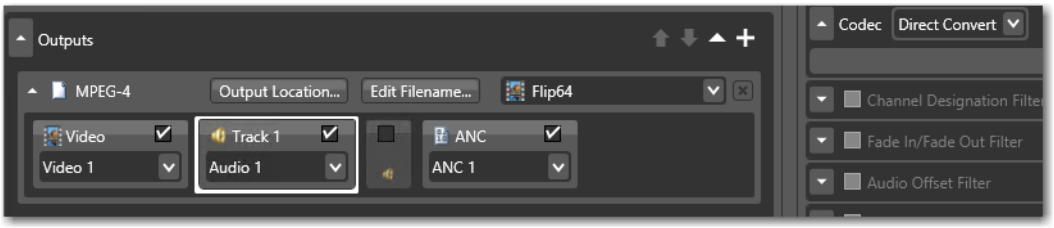
Each *Track* element included in the *Target* creates a new audio track in the output, using the specified channel(s) in the *Sequence*. *Tracks* must be arranged ordinally in the CML so that they progress in logical, sequential order, beginning at 1.

Note: Audio output from the CML pre-processor is always uncompressed 24-bit PCM. If you plan to utilize Flip64 to encode/re-organize the final output audio, use a *Target* in the CML to define a single audio *Track* (which will be used as a source in the Flip64 action). However, if you plan to configure the audio directly in the CML, you should define one or more *Tracks* as required. Then, configure the Flip64 action to propagate these tracks directly to the output by enabling *Preserve Original Audio Tracks* and using the Direct Convert codec.

The *Preserve Original Audio Tracks* control in the Flip64 inspector dialog is displayed in the Auto Input's audio track component:



Select the *Direct Convert* Codec in your output Track components:



There are no child elements of *Track*.

See also *Mix* and *Target*.

Attributes

These attributes are required. The *source* and *target* attributes act in pairs: for each channel in the *Sequence*, there should be a corresponding channel for this *Track*.

Name	Description
source	Integer; one or more (separated by a space), that identifies each channel in the Sequence that comprises this *Track*. Example: `<Track source = "3 4" target = "1 2" />` Here, this track is comprised of the *Sequence's* channels 3 and 4.
target	Integer, one or more (separated by a space), to identify the ordinal sequence—channel number—of the specified output channel(s) in this *Track*. Example: `<Track source = "3 4" target = "1 2" />` Here, this track has two channels—1 and 2. (which are channels 3 and 4 in the *Source*).

Example

In this example, the MXF file has video and six audio tracks, one channel each. The *Mix* elements define how the channels are placed on the timeline when this *Source* is referenced in a *Segment*. The *Target* specifies three tracks to be presented for processing in Flip64, per the three *Track* elements. Each track is a stereo pair of channels in logical order: 1-2 and 3-4, and 5-6, placed into channels 1 and 2 of each track.

```
<Composition "xmlns="Telestream.Soa.Facility.Playlist">
  <Source identifier="1" instructions="no-direct-convert">
    <File location="\\share\Media\6MonoTracks.mxf" />
    <Mix source="1" target="1" />
    <Mix source="2" target="2" />
    <Mix source="3" target="3" />
    <Mix source="4" target="4" />
    <Mix source="5" target="5" />
    <Mix source="6" target="6" />
  </Source>
  <Sequence>
    <Segment>
      <Video source="0" filter="mute">
      <Audio source="0" />
    </Segment>
  </Sequence>
  <Target>
    <Timecode type="source" />
    <Track source="1 2" target="1 2" />
    <Track source="3 4" target="1 2" />
    <Track source="5 6" target="1 2" />
  </Target>
</Composition>
```

Three stereo tracks are presented to Flip64. as PCM. *Preserve Original Audio Tracks* should be enabled in Flip64 because you are creating multiple tracks.

Note: Audio output from the CML pre-processor is always uncompressed 24-bit PCM. If you plan to utilize Flip64 to encode/re-organize the final output audio, use a *Target* in the CML to define a single audio *Track* (which will be used as a source in the Flip64 action). However, if you plan to configure the audio directly in the CML, you should define one or more *Tracks* as required. Then, configure the Flip64 action to propagate these tracks directly to the output by enabling *Preserve Original Audio Tracks* and using the Direct Convert codec.

Video

Flip64 CML Elements in Alphabetic Order | Flip64 CML Hierarchy Map

A *Video* element is required in a *Segment* in order to include video in the output. The *Video* element is a material element that defines the video stream from a file identified by a specific *Source*, by using the *source* attribute (required).

By adding *Video* and/or *Audio* to a *Segment,* you are placing the corresponding source(s) on the *Sequence*'s timeline (accounting for optional trim points).

One *Video* element is allowed per *Segment*.

Video that includes audio in the same file MUST include the `filter = "mute"` attribute, to mute the associated audio if it is present in the source, regardless of the source of the *Video* and *Audio*. *Audio* must be identified separately to include it in the *Segment*; it is not included automatically.

See also *Audio*.

Child Elements

One each of these elements may be added to apply a *Video* element:

- *Head*
- *Tail*

Attributes

These attributes are required in a *Video* element.

Name	Description
filter	Keyword. *Video* that includes *Audio* MUST include the *filter* attribute to insure proper processing. This mutes the associated audio if it is present in the source. Keyword: *mute*. Example: `<Video source="0" filter = "mute" />`
source	Integer, corresponding to the *Source*'s *identifier* attribute value, to identify the file that contains this video stream.

Example

In this *Video*, the input file (not shown) has both video and audio; thus the *Video* element requires a *filter* attribute.

This example illustrates clipping the *Video* and the *Audio*; unclipped material in a *Segment* is always padded to the full length of the clip unless you trim all material to the same length, since a *Segment's* length is always the duration of the longest material.

```
<Segment>
  <Video source="1" filter="mute" >
    <Head>
      <Edit mode="relative" time="00:00:03.000" />
    </Head>
    <Tail>
      <Edit mode="relative" time="00:00:30.000" />
    </Tail>
  </Video>
  <Audio source="1">
    <Head>
      <Edit mode="relative" time="00:00:03.000" />
      <Fade duration="00:00:00.020" shape="linear" />
    </Head>
    <Tail>
      <Edit mode="relative" time="00:00:30.000" />
      <Fade duration="00:00:00.020" shape="linear" />
    </Tail>
  </Audio>
</Segment>
```