Transcode CML
Developer Guide


# Transcode CML Developer Guide

## Vantage 8.0

## Flip64 ComponentPac 8.0.8
## IPTV Flip ComponentPac 8.0.5
## Multiscreen Flip ComponentPac 8.0.7

telestream

**October 2022** **328005**

# Copyrights and Trademark Notices

Copyright © 2022 Telestream, LLC and its Affiliates. All rights reserved. No part of this publication may be reproduced, transmitted, transcribed, altered, or translated into any languages without written permission of Telestream, LLC. Information and specifications in this document are subject to change without notice and do not represent a commitment on the part of Telestream. Specifications subject to change without notice.

Telestream, CaptionMaker, Cerify, DIVA, Episode, Flip4Mac, FlipFactory, Flip Player, GraphicsFactory, Kumulate, Lightspeed, MetaFlip, Post Producer, ScreenFlow, Switch, Tempo, TrafficManager, Vantage, VOD Producer, and Wirecast are registered trademarks and Aurora, ContentAgent, Cricket, e-Captioning, Inspector, iQ, iVMS, iVMS ASM, MacCaption, Pipeline, Sentry, Surveyor, Vantage Cloud Port, CaptureVU, FlexVU, PRISM, Sentry, Stay Genlock, Aurora, and Vidchecker are trademarks of Telestream, LLC and its Affiliates. All other trademarks are the property of their respective owners.

**Adobe**. Adobe® HTTP Dynamic Streaming Copyright © 2014 Adobe Systems. All rights reserved.

**Apple**. QuickTime, MacOS X, and Safari are trademarks of Apple, Inc. Bonjour, the Bonjour logo, and the Bonjour symbol are trademarks of Apple, Inc.

**Avid**. Portions of this product Copyright 2012 Avid Technology, Inc.

**CoreOS**. Developers of ETCD.

**Dolby**. Dolby and the double-D symbol are registered trademarks of Dolby Laboratories Licensing Corporation.

**Fraunhofer IIS and Thomson Multimedia**. MPEG Layer-3 audio coding technology licensed from Fraunhofer IIS and Thomson Multimedia.

**Google**. VP6 and VP8 Copyright Google Inc. 2014 All rights reserved.

**MainConcept**. MainConcept is a registered trademark of MainConcept LLC and MainConcept AG. Copyright 2004 MainConcept Multimedia Technologies.

**Manzanita**. Manzanita is a registered trademark of Manzanita Systems, Inc.

**MCW**. HEVC Decoding software licensed from MCW.

**MediaInfo**. Copyright © 2002-2013 MediaArea.net SARL. All rights reserved.

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 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|Server 2016|Server 2019, Windows 7, Windows 8, Windows 10, Media Player, Media Encoder, .Net, Internet Explorer, SQL Server 2005|2008|2012|2016|2019, 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.

**Xceed**. The Software is Copyright ©1994-2012 Xceed Software Inc., all rights reserved.

**ZLIB**. Copyright (C) 1995-2013 Jean-loup Gailly and Mark Adler.


Other brands, product names, and company names are trademarks of their respective holders, and are used for identification purpose only.

# 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<br>Support Email: support@telestream.net<br>Enterprise Telephone Support:<br>U. S. Toll Free: (877) 257-6245<br>U. S. from outside U.S.: (530) 470-2036<br>Europe \| Middle East \| Africa \| Asia \| Pacific:<br>+49 228 280 9141<br>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<br>Support Email: support@telestream.net |
| Telestream, LLC | Web Site: www.telestream.net<br>Sales and Marketing Email: info@telestream.net<br>Telestream, LLC<br>848 Gold Flat Road, Suite 1<br>Nevada City, CA USA 95959 |
| International Distributor Support | Web Site: www.telestream.net<br>See the Telestream Web site for your regional authorized Telestream distributor. |
| Telestream Technical Writers | Email: techwriter@telestream.net<br>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


## Transcode CML Overview 13

Introduction 15
Licensing Requirements 16
CML-enabled TMF Transcode Actions 17
- Flip64 17
- Multiscreen Flip 17
- .IPTV Flip 17

How Transcode CML Works 18
- Stitching Media 18
  - Stitching Video in Direct-Convert Mode 19
  - Stitching Video in Transcoding Mode 19
- Audio Processing 19
  - Processing Single-track Audio 20
  - Processing Multi-track Audio in Flip64 20
  - Processing Multi-track Audio in IPTV Flip and Multiscreen Flip 20
- Processing VBI, VANC, and Captions 20
- Controlling Timecode 21

Transcode CML Requirements and Limitations 22
Troubleshooting 23

## Developing Compositions 25

Approaches to Composition Development 26
- Skills and Tools Required for the Programming Approach 26
- Skills and Tools Required for the Manual Approach 26

Developer Resources 27
Media Characteristics Affect Composition Design 28
Composition File Requirements 28
- Case Sensitivity is Critical 28

Specifying Unit Designators in Attributes 29
The Basic Format of a Composition 31
Resolving Errors in CML Processing Jobs 33
Commenting CML Files 34

## Transcode CML Reference 35

Transcode CML Elements 35
Transcode CML Hierarchy Map 36
Transcode CML Implementation Details 37
- Case Sensitivity is Critical 37
- Specifying File References in Transcode CML 37

Audio 38
- Child Elements 39
- Attributes 39
- Example 39

Canvas 40
- Attributes 40
- Example 41

Comment 42
- Example 42

Composition 43
- Child Elements 43
- Attributes 43
- Example 43

Edit 44
- Attributes 44
- Example 45

Fade 46
- Attributes 46
- Example 46

File 48
- Attributes 49
- Example 49

Head 50
- Child Elements 50
- Example 51

Mix 52
- Attributes 54
- Examples 56

Segment 57
- Child Elements 57
- Example 57

Sequence 58
- Child Elements 58
- Example 58

Source 60
- Child Elements 60
- Attributes 61
- Examples 61

Subtitle 63
- Child Elements 63
- Examples 63

Attributes 65
Examples 65
Basic Subtitle with a Sidecar SCC File 66
Propagating Embedded Captions 66
Tail 67
Child Elements 67
Example 68
Target 69
Configuring Audio Tracks in Flip64 69
Configuring Tracks in CML 69
Child Elements 70
Example 70
Timecode 72
Attributes 72
Example 73
Track 74
Attributes 75
Flip64 Example 75
IPTV Flip | Multiscreen Flip Example 76
Video 78
Child Elements 78
Attributes 78
Example 79

## Prototype Applications 81

Mapping and Mixing Audio 82
Audio Configuration Details 82
Example 1 83
Example 1 83
Example 83
Specifying Avid OPAtom Files in a Segment 86
Processing Multi-Track Audio CML for Flip64 87
Specifying the Source Assets 88
Selecting and Rearranging Audio Channels 90
Adding Sources to Segments on the Timeline 90
Organizing Channels into Tracks 91
Conclusion 92
CML 92

# Transcode CML Overview

The purpose of this guide is to help video producers, editors, operators, and other professionals involved in media processing learn how to create edit decision lists (EDLs) using Telestream's *Transcode Composition Markup Language* (Transcode CML) to automatically create simple compositions, by trimming and concatenating—*stitching*—clips together, directly in Vantage's Telestream Media Framework (TMF) workflows using Flip64 | IPTV Flip | Multiscreen Flip transcoding actions.

Learning and using Transcode CML requires familiarity with both Vantage and the Flip64 | IPTV Flip | Multiscreen Flip transcoding actions. Because the language is implemented in XML format, a basic understanding of XML is also helpful.

In this chapter, you'll learn what Transcode CML is designed to do, how it works, and how you can implement it in your Flip64 | Multiscreen Flip | IPTV Flip workflows.


- Introduction
- Licensing Requirements
- CML-enabled TMF Transcode Actions
- How Transcode CML Works
- Transcode CML Requirements and Limitations
- Troubleshooting


**Note:** This document applies to the version of Vantage identified in the title and later, unless superseded by a newer publication. You should use the latest publication for your version of Vantage. Visit https://www.telestream.net/telestream-support/vantage/support.htm to obtain current publications. Visit https://www.telestream.net/telestream-support/vantage/help.htm to obtain publications for previous versions.

Users familiar with Post Producer CML may recognize Transcode CML as similar. However, Transcode CML is designed specifically for media stitching in Telestream Media Framework actions, and is not intended for use in Post Producer workflows.

Post Producer CML is much broader in scope, and is only intended for processing by Post Producer actions—those in Workflow Designer's Edit category. Post Producer CML is a fully-featured media composition engine. It automatically creates complex

compositions that conform to a pre-defined timeline from various media sources. For example, it can flatten IMF and DPP packages. Driven by CML, Post Producer can trim, segment, stitch and clip, overlay video, transition, cross fade, overlay audio, route audio, insert graphics, etc.

---

**Note:** This guide is written for video professionals who are familiar with using Vantage. To implement media processing applications in Vantage, you should know how to use Vantage Workflow Designer to create and configure workflows and submit jobs. If you aren't familiar with Vantage, Telestream suggests that you review the *Vantage User's Guide* and *Vantage Domain Management Guide* as required.

---

# Introduction

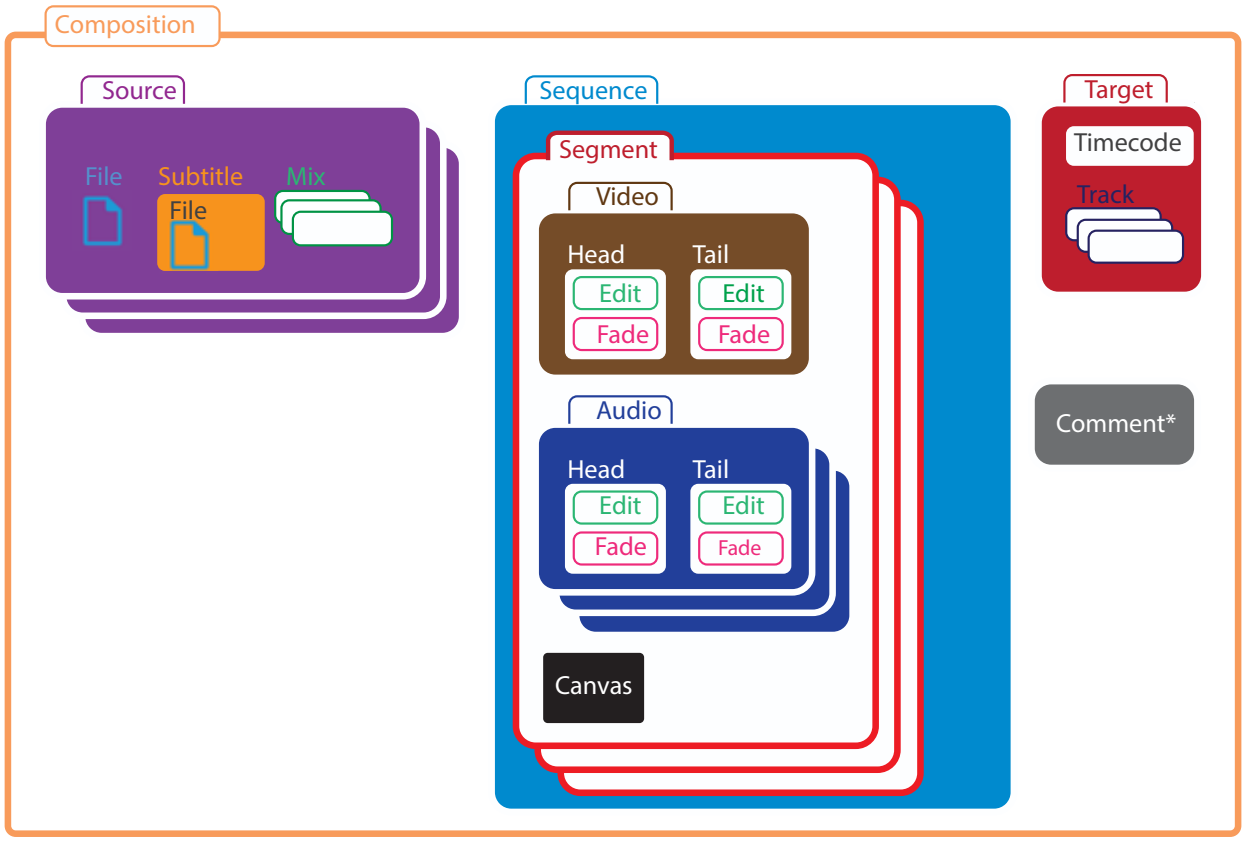
Transcode CML is Telestream's media definition language—designed specifically for utilizing the Telestream Media Framework (TMF) Flip64 | Multiscreen Flip | IPTV Flip transcoders in Vantage to stitch media clips as part of the transcoding process—in transcoding or direct-conversion workflows.

In transcoding workflows, you can stitch media of different formats (with the same frame size and frame rate) to generate new media; in direct-conversion workflows you can assemble media from same-format sources, to meet your specifications.

Audio is always presented as uncompressed PCM; some basic audio processing typically associated with stitching is provided.

Transcode CML is hierarchical, and uses XML syntax. Each of its elements identifies a particular aspect of media, along with attributes to conform it as required. The elements function as building blocks, enabling you to create and optionally trim, and organize the media you want to generate on a timeline.

This diagram illustrates the relationship between elements. Click on any element for details:




Stacked elements indicates that multiple instances are permitted in a composition.

Transcode CML describes media in a manner suitable for automated content assembly to segment, trim, and stitch like-media (whose frame size and rate is identical)—using supported TMF transcoding actions in Vantage workflows. Transcode CML is ideally suited for automating repetitive media generation tasks in a production environment.

Common parlance referring to processes for connecting media from separate files includes stitching, tagging, trimming, timing, clipping, segmenting, cut-downs, adding stringers, and adding a header/footer. In the context of the use of CML, this guide generally uses the term *stitching*.

The term *composition* in the context of CML refers not only to the XML file that contains the CML instructions, but often, the output of a transcoding action that ingested the composition as an XML file, and produced (conformed) a media composition defined by the CML itself.

Stitching media connects multiple, sequential input files/segments to produce a single file—a composition. Stitching with Transcode 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). Gaps or slate—spaces between clips—can also be specified, using the *Canvas* element.

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

Transcode CML processing should be approached from a design and implementation perspective as a separate, CML pre-processing step, performed directly in TMF transcoder actions before transcoding the media generated by the CML pre-processor. When a TMF transcoder ingests CML, it engages the CML pre-processor to generate (conform) the specified media for processing by the transcoder —just as if the dynamically-generated media had been ingested directly from the file.

Thus, in the context of this guide, the term *output* generally represents the output from the CML pre-processor; not the output of the TMF transcoder action itself. The output of the CML pre-processor is *input* only in the context of the TMF transcoder.

# Licensing Requirements

Transcode CML can be utilized in Flip64, Multiscreen Flip, and IPTV Flip actions in Vantage workflows, which require the Transcode Pro license. No other license is required. For details, see *Contacting Telestream* for customer service for licensing.

# CML-enabled TMF Transcode Actions

Transcode CML can be ingested and processed in TMF transcoding actions—Flip64 | Multiscreen Flip | IPTV Flip actions—operating on-premises or in Cloud mode.

## Flip64

Flip64 is Telestream's broadcast-centric, high-performance 64-bit transcoder. Flip64 can be driven by CML to create simple compositions or trim content to your specification. It automatically creates simple compositions from like media sources or trims and stitches content without requiring a separate NLE or Post Producer. It is ideal for automating repetitive tasks in your production environment.

## Multiscreen Flip

The Multiscreen Flip action combines the quality of x264 H.264, GPU acceleration, and complete workflow automation for multi-screen encoding. Multiscreen CML automatically creates simple compositions or trims/stitches content and encode/package ABR for OTT.

Multiscreen workflows can automate the entire process of creating adaptive bitrate packages, including content ingest, transcoding, packaging, encryption, delivery and notification.

## .IPTV Flip

IPTV VOD offers a complete solution to automate transcoding for IPTV and Cable VOD production. The IPTV VOD Flip action enables you to achieve the highest possible quality at the lowest bit rates. With GPU accelerated transcoding, integration of x264 H.264 and x265 HEVC encoding technology and Manzanita Transport Stream multiplexing. You can use CML to automatically create simple compositions or trim/stitch content for VOD and IPTV deliveries.

# How Transcode CML Works

TMF transcoders enable you to stitch media clips together and perform other related tasks using Transcode CML. These topics introduce the key aspects of Transcode CML:

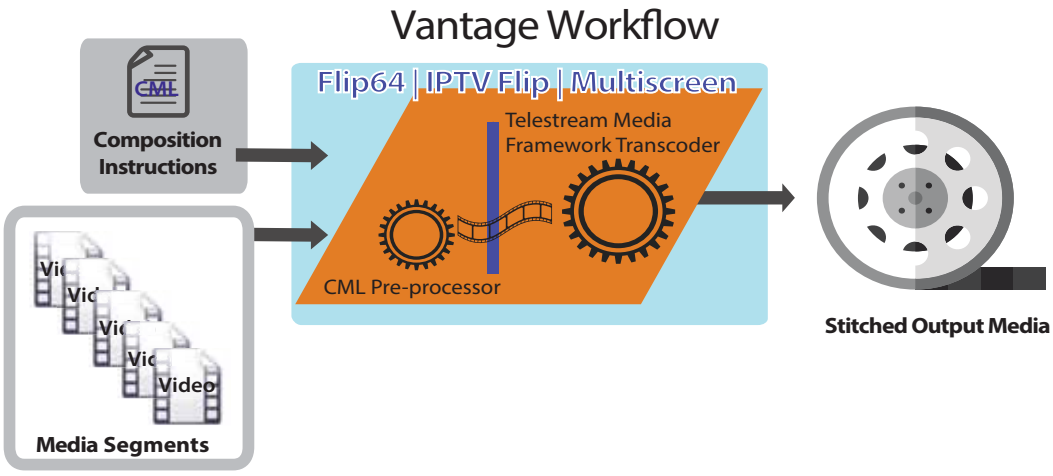
- Stitching Media
- Audio Processing
- Processing VBI, VANC, and Captions
- Controlling Timecode

## Stitching Media

When you are designing a workflow to process CML, the Input component in the transcoder action must be set to *Auto*. No other Input components are supported. The Audio is represented as a single channel.

When a TMF transcoder ingests a CML file, it passes it directly to the CML preprocessor for conforming the media.

The CML preprocessor parses each element and generates (conforms) new media from the specified source files (or clips) and optionally, auto-generated black frames, as described in the CML.




In transcode workflows, uncompressed baseband video is presented; in direct-convert mode, original compressed frames are presented. The media segments in the CML must be of the same resolution—same frame size (width and height)— and frame rate. Audio is always presented as uncompressed PCM.

Stitching is performed prior to transcoding; the transcoder is presented with a conformed media stream, just as if it had been presented as an actual media file, instead of a definition in CML comprised of multiple files/segments.

The TMF transcoder encodes or direct-converts the conformed media stream from the CML preprocessor. This process is the functional equivalent of submitting the

generated output of the CML directly to the TMF transcoder, instead of the CML file. The action generates the final output file per the transcoder action's configuration.

### Stitching Video in Direct-Convert Mode

When you are processing source media in a TMF transcode action configured to direct-convert the media, in addition to being the same frame size and rate, each of the video segments must have been compressed with the same codec; for example, H.265. The CML must also be configured to operate in direct-convert mode, presenting the video as compressed frames to the transcoder.

---

**Note:** If you submit media in CML of differing formats to a workflow where the transcode action is configured with a direct-convert profile, the job will fail.

---

Stitching video may require edge-frame re-encoding to repair the stream, but this is internal to the CML pre-processor. With long-GOP media (temporally-compressed video (MPEG-2 | x264 | x265 and derivatives, etc.), the preprocessor creates a new GOP if the original GOP is broken. Otherwise, the original video essences are stitched without decoding and re-encoding. This is an extremely fast operation, and doesn't degrade video quality. This occurs when stitching trimmed clips (*Edits*) and <Head> or/and <Tail> are present and not aligned with the first/last frame (thus, you have not specified the entire clip), results in edge re-encoding to repair GOPs.)

For example, three SD MPEG-2 files are stitched—their original video essences entirely preserved—with new frames re-encoded only at stitch points as needed to repair broken GOPs, into one new SD MPEG-2 file. No decoding or re-encoding occurs in the transcoder.

### Stitching Video in Transcoding Mode

When you are processing media in a TMF transcode action configured to encode the video in a different format, the frames must be the same frame size and rate, but the video segments may have been compressed with different codecs. This allows you to include segments in any format supported by Vantage. The CML must also be configured to operate in no-direct-convert mode, presenting the video as uncompressed frames to the action.

When configured to transcode the video, the CML preprocessor decodes the source video to uncompressed baseband frames. It passes the uncompressed frames to the action for compression, as configured. For example, three files with different formats of video—two in H.264 and one in MPEG-2—are stitched, and the baseband media is then re-encoded into H.265.

## Audio Processing

Unlike video, compressed audio is alway decompressed to PCM, stitched along with the video, and passed into the transcoder as uncompressed PCM audio, irrespective of the video mode (direct-convert or no-direct-convert).

When you are processing audio in Flip64, you can optionally use Transcode CML to organize audio into one or more tracks, with channels as appropriate.

The CML can be also configured to produce a single track with multiple channels. Single-track audio can be processed in all three actions: Flip64 | Multiscreen Flip | IPTV Flip. For a comprehensive discussion, see *Processing Multi-Track Audio CML for Flip64*.

### Processing Single-track Audio

Single-track audio is passed channel-by-channel into the transcoder as uncompressed PCM. In the transcoding action, you can perform any audio processing required that is supported by the action.

### Processing Multi-track Audio in Flip64

Processing audio in Flip64 offers more flexibility than in IPTV Flip and Multiscreen Flip. When you configure your CML with multi-track audio, you configure Flip64 to preserve the original audio tracks, presenting audio channels in their original track organization. This eliminates the need to configure Flip64 for the same result.

---

**Note:** When you configure Flip64 to preserve original audio tracks, Flip64 simply passes them through to the output, in PCM format. You must also configure your output tracks for Direct Convert. You must also define how you want to organize your tracks and channels directly in CML. Flip64 passes the input directly to the output, untouched.

---

### Processing Multi-track Audio in IPTV Flip and Multiscreen Flip

When processing multi-track audio in IPTV Flip and Multiscreen Flip, all of the channels are organized into a single track in the CML and sent to the transcoder. You can either configure your output audio for direct convert—where you will have one track with the same number and order of channels in PCM format in your output—or you can organize your channels into new tracks as required (organizing, mixing, labeling, setting audio levels, etc) for output, by configuring the action to perform these tasks.

---

**Note:** When you are processing multi-track audio in IPTV Flip and Multiscreen Flip, you can only define one track in CML. Tracks are configured directly in the action.

---

## 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.

Transcode 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.

SCC files can be associated with CML as a separate file, for processing as supported.

## Controlling Timecode

You can optionally specify the output media's starting timecode in CML using the Timecode element, which overrides source timecode. The timecode is then incremented throughout the stitched output file.

# Transcode CML Requirements and Limitations

The following requirements and limitations apply to Transcode CML processing:

- CML files must have a *.cml file extension (for example, *MyStitchCMLfile.cml*) to be recognized as a CML file and processed properly by the transcoding action.
- Transcoding actions only process files stored 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 relocate the file on a Windows server accessible to the Transcode service processing it.

All transcoding actions must be configured with one Auto Input. Other types of Inputs cause the job to fail. Auto Input supports one video stream, one audio track, and one ANC stream. This applies even if you are performing multi-track audio processing directly in CML for Flip64 workflows (see *Processing Multi-Track Audio CML for Flip64*).

- Stitching is only 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 size/rate media, process the media through another workflow to bring your frame size and rate into conformance with your planned output specifications.
- Transcode 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) to PCM prior to presenting it to the transcoder for processing; direct-conversion of compressed audio is not supported. For instance, if an MP4 source has H.264 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 if required.
- 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 is not supported.

# 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 Transcode 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 or GLIM 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.

# Developing Compositions

This chapter provides referential information—practical, detailed guidelines for various aspects of creating and developing composition files.

**Note:** Transcode Composition Markup Language is a dialect of CML. Though the CML dialects are similar, Transcode CML Language is designed specifically for Flip64 | Multiscreen Flip | IPTV Flip workflows.

As a CML programmer, you may be writing a program to automatically generate (or otherwise utilize) a composition or you may simply be manually creating or editing compositions for submission. Regardless of your background, skills, or methods, these topics are designed to help you in developing CML compositions.

## Topics


- Approaches to Composition Development
- Developer Resources
- Media Characteristics Affect Composition Design
- Composition File Requirements
- Specifying Unit Designators in Attributes
- The Basic Format of a Composition
- Resolving Errors in CML Processing Jobs
- Commenting CML Files

# Approaches to Composition Development

CML is written in XML schema that provides the vocabulary to describe video clips and their temporal relationships on a video timeline, in a manner suitable for stitching clips together and conforming the specified input files into an output media file that adheres to the general description.

There are two different approaches to creating compositions: write a program that generates compositions or manually create them in a text editor. The benefit of using a program to generate compositions is that you're relieved of the requirement to hand-craft valid, complex CML. With a properly designed program, you're assured of producing a valid composition. Programs are worth the effort when compositions must be generated in high volume, making manual editing the more expensive proposition.

However, you can also create a composition manually, using an XML editor or text editor. The benefit of this approach is that you don't have to be a programmer.

---

**Note:** Neither approach eliminates the need to understand Composition Markup Language. In both cases, there is a common requirement: you need to know how to properly construct a composition to meet your media generation requirements. If you don't understand CML, you'll have a difficult time creating the composition you intend without a lot of trial and error.

---

The approach you take depends on several factors:

- Technical skills—that you have, or are available to you
- Scale of effort—do you need a few compositions, or lots of them
- Intended use—whether this a testing/learning experience or a production task.

## Skills and Tools Required for the Programming Approach

If you are planning to write a program to generate a composition, you'll need the following skills and tools:

- Composition Markup Language for the dialect you are using
- XML
- C# or other .Net CLI language
- Microsoft Visual Studio (or other .Net development tool).

## Skills and Tools Required for the Manual Approach

If you are planning to write a composition manually, you'll need these skills and tools:

- Composition Markup Language for the dialect you are using
- XML
- XML or text editor program

# Developer Resources

In order to learn about CML and apply it to your video processing requirements, refer to the following products and documents:

*Vantage SDK*—The Vantage SDK is an indispensable part of CML development. It contains the .Net class libraries, C# code examples, and documents that you need to develop programs to create and generate compositions, and to submit files (such as compositions and media descriptor files) to Vantage workflows.

Registered users can download the Vantage SDK by logging into the Registered Vantage Customer section on the Telestream Web site and following the steps.

*Composition Class Library Help*—A CHM file in the SDK for programmers who are creating a composition or utilizing compositions for other purposes. This reference describes the Vantage Composition class library (referred to as the Playlist library), which implements the Composition Markup Language.

---

**Note:** *Transcode CML Developer's Guide*—The guide you're reading now.There may be XML errors in your properly-formed CML file that are permitted. See *Resolving Errors in CML Processing Jobs* for more details.

---

# Media Characteristics Affect Composition Design

Compositions can't be created in a vacuum. For any composition to be practical and useful, some assumptions must be made.

For any composition you want to create, in addition to the design of your intended output, you should at a minimum, know:

- Input container type and media format
- The intended length of output (60 seconds, 3 minutes, etc.)
- Input and output video formats (SD / HD 720, 1080, etc.)
- Input and output audio track layout: 5.1 | stereo | Dolby E, track count and layout

The specifications of your media directly affect the media modifications you plan to perform in the CML to generate the required output.

# Composition File Requirements

Compositions must be written in XML, and saved as a text file. It is common to name XML files with the .xml suffix. By convention, Telestream often names composition files using a *.cml* file suffix, to identify it as a Composition Markup Language file. The fact is, you can name a composition file with any suffix you want, and submit it to a Vantage workflow for processing. The suffix is immaterial in this context.

## 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 mis-spell it), it is ignored—and no error is displayed.

Although attributes are also by convention in upper case, most attributes in CML are all lower case. Be sure to use the case indicated in the element's definition, in *Composition Markup Language Reference*.

# Specifying Unit Designators in Attributes

Whenever you specify a dimensional, volumetric, or other value-oriented number in an attribute (for example, *<Crop top="30px" left="0" bottom="-2px" right="100%" />*), you can identify the logical unit designator that applies to it.

---

**Note:** Per XML syntax rules, all attribute values must be enclosed in straight (not smart) double quotes *(*" and "*)*. Use of smart quotes causes the workflow to fail. This applies to all value types: strings, integers and other numbers, and keywords. For example, *<Crop top="30px"... />*.

---

If you specify a value without a unit designator, it is *always* treated as a ratio.

The unit designator you use should immediately follow the value, with no intervening space. For example, *top="30px"*.

The table below describes each unit designator permitted in CML.

| Unit | Designator | Description |
|---|---|---|
| Percent | % | Use percent (%) to define the ratio of the attribute value compared to the whole it is referencing (for example, right edge). You can use percent to describe most value types—for example, an increase in volume, or the width of an output frame.<br>Integers and decimals are permitted.<br>In some attributes, the value may be negative. For others, it must be in the range 0-100%. Or, the value may be over 100%, as in the case of Rotation: 200% is equal to 720 degrees, or two complete rotations. |
| Decibels | dB | Use Decibels (dB) to define audio volume values. |
| Degrees | ° | Use degrees (°) to specify the amount of rotation. In some attributes, it may be in the range 0 to 360; in others (such as Rotate), you may specify an unlimited amount of rotations—for example, 720 for 2 turns, or 3600 (for 10 turns). You might also be able to specify a negative degree: *phase="-90°"*, for example. |
| Pixels | px | Use pixels to define values relating to dimensions or points in raster images, input and output frames. In some cases, you may use negative pixels. Pixels are always expressed as integer values. |

| Unit | Designator | Description |
|---|---|---|
| Points | pt | Use points (approx. 1/72 of an inch) to define values relating to the size of text, as in a Title element. Integers and decimals are permitted. |
| Fraction | {} | There is no designator for a fraction. Use fractions when it is more accurate, easier (or better understood by other readers) to express the proper value. Because a fraction is a mathematic expression, it must be enclosed in braces. For more details, see *Using Nicknames, Expressions, Variables, and Constants in Compositions*. |
| Ratio | | Like fractions, there is also no designator for a ratio. Use ratio in the same manner as percent: to define the ratio of the attribute value to the whole being referenced—for example, right edge. A ratio may be expressed as an integer, a decimal number, or a rational number {5/10} which must be presented as an expression, in braces. Ratio is effectively a percentage value, with the decimal moved 2 places left. For example, 100% = 1.00.<br>In some attributes, the range is 0 to 1. In other cases, the value may be greater than 1. |

# The Basic Format of a Composition

Now that you conceptually understand the major elements of a composition and their relationship to each other, lets turn to the task of implementing them in Composition Markup Language—by creating a CML file.

Here, a simple composition is presented. Note that all attribute values are in straight double quotes—an XML requirement. Also, note the UNC path in the *File* element—the most effective way to access files in a multi-server environment hosting a typical Vantage domain.

```
<Composition>
  <Source identifier="0" instructions="no-direct-convert">
    <File location="\\footage\Media\6MonoTracks1.mxf"/>
    <Mix source="1" target="1" level="0" phase="0°" />
    <Mix source="1" target="2" level="0" phase="0°" />
    <Subtitle preserve708="true" />
  </Source>
  <Source identifier="1" instructions="no-direct-convert">
    <File location="\\footage\Media\6MonoTracks2.mxf" />
    <Subtitle preserve708="true" />
  </Source>
  <Sequence>
    <Segment>
      <Video source="0" filter="mute">
        <Head>
          <Edit mode="absolute" time="00:00:00.000" />
        </Head>
        <Tail />
      </Video>
      <Audio source="0">
        <Head>
          <Edit mode="absolute" 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>
  <Target>
    <Timecode type="start" time="01:02:03:04@29.97" />
    <Track source="1 2" target="1 2" />
  </Target>
</Composition>
```

This example illustrates the proper format and organization of the basic composition elements.

A composition always starts with an XML element, as required per XML standards, and has one Composition element. A composition has the following elements:

- Inside the Composition element is a Source element, to identify the source file for the Video. By convention they are placed before any Sequence elements to improve readability.

- A Sequence element. (One or more sequences are permitted).
- Segment elements in the Sequence, where the video is utilized. As with sequences, multiple segments are permitted.
- Head and Tail elements corresponding to segment mark in and mark out points.

This example composition produces an output file in the format specified. This is the foundation you'll build on to create  conformed media files.

# Resolving Errors in CML Processing Jobs

When you submit a CML file to a workflow, it may fail for the following reasons:

- *Invalid XML*—you may have created a file that does not follow XML standards.
- *Invalid CML*—you may have created a file that does not conform to Composition Markup Language requirements for dialect you are using.
- *Errors in values*—you may have an attribute value that is not correct.
- *Unsupported source material*—it may be necessary to transcode source material using Flip64 or other transcoding action to convert it to a supported format, prior to using it in a Flip64 | Multiscreen Flip | IPTV Flip workflow (see *Supported Formats for Stitching* .
- *Invalid path or filenames*—paths or file names may be incorrect or inaccessible.

When a CML file cannot be processed by a transcoding action, you typically see an error in Workflow Designer's Job Status tab:

If the composition is valid, verify that the values you supply for each attribute are correct (keywords/range) for the media being processed.

---

**Note:** If you provide an attribute that isn't valid (for example, you add *duration*, but it's in the wrong element, or you mis-spell *align* as *allign*), the attribute is ignored. If you provide a value that is of the wrong type (for example, providing a string where an integer is required), the action will fail. Or, if you provide a keyword that is not permitted or incorrectly capitalized, the action will fail.

---

# Commenting CML Files

XML permits comment lines. Telestream recommends that you document your media specifications directly in the CML, so that you and others are cognizant of them as the composition is studied, used, or modified.

Optionally, provide other design or runtime (for example, transcode action configuration requirements) details including the author, and a change history.

Here's an example code snippet, with comments in green:

```
...
</Source>
<!-- Stereo audio in video source and 10 VOs map source mono -->
<!-- track to a track in the output file. Channel map must be -->
<!-- set in the Compose action in the Vantage workflow. -->
  <Sequence layer="1">
    <Segment>
      <Video align="head" adjust="edge" fill="none" source="1"
layer="0">
       ...
```

Using comments can make understanding, modifying, and maintaining your compositions much easier. You can also use the CML *Comment* element for documenting your CML.

# Transcode CML Reference

The purpose of this chapter is to identify each element in Transcode Composition Markup Language and their relationships. Each element is described in detail, including its attributes, along with simple examples to illustrate typical usage.

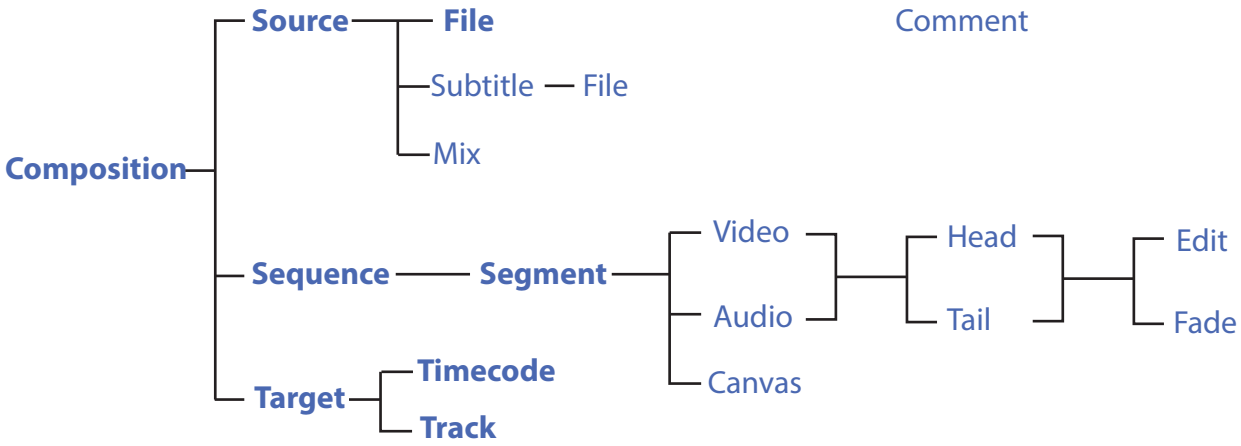

- Transcode CML Elements
- Transcode CML Hierarchy Map
- Transcode CML Implementation Details


## Transcode CML Elements


- Audio
- Canvas
- Comment
- Composition
- Edit
- Fade
- File
- Head
- Mix
- Segment
- Sequence
- Source
- Subtitle
- Tail
- Target
- Timecode
- Track
- Video

# Transcode CML Hierarchy Map

This tree structure illustrates the hierarchical relationship between the Transcode CML elements, expressed as a tree. Click on any element to display its 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 use 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.

# Transcode CML Implementation Details

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


- Case Sensitivity is Critical
- Specifying File References in Transcode CML


## Case Sensitivity is Critical

XML is case-sensitive.

CML elements must be entered in title case (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 Transcode CML

Files referenced in Transcode CML must be located on Windows servers, because TMF transcode actions 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.

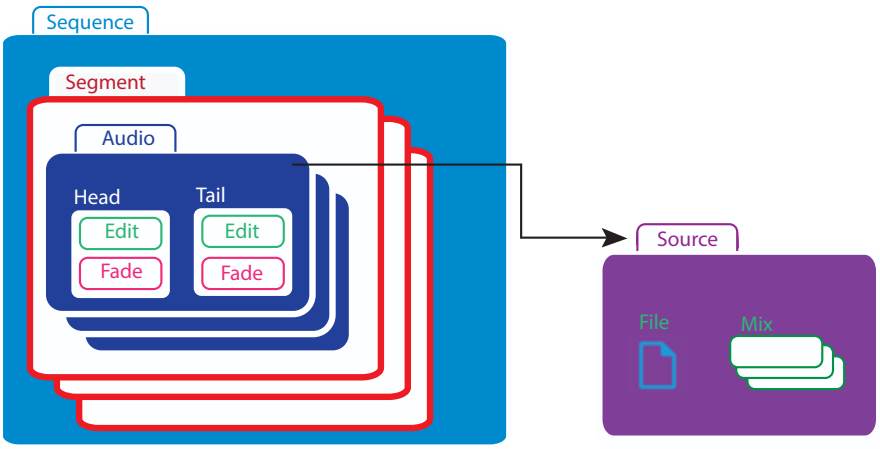
References to (potentially relocated) files may be specified using a Windows drive letter or a UNC path.

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

Transcode CML Elements | Transcode CML Hierarchy Map

*Audio* is a material element. You add up to 32 *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*.

*Video* that includes audio in the same file MUST include the `filter="mute"` attribute, to mute the associated audio, 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.

You can optionally trim the *Audio* using *Head* and *Tail* with an *Edit* element.

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

---

**Note:** Audio output from the CML pre-processor is always uncompressed 24-bit PCM. If you plan to utilize a transcode action to encode/re-organize the final output audio, use a *Target* in the CML to define a single *Track* to be used as a source in the action and configure tracks and channels in the transcoder action to meet your requirements.
In IPTV Flip and Multiscreen Flip, you can only define a single track. For Flip64 only, if you plan to define one or more *Tracks* as required directly in the CML. Then, configure the Flip64 action to propagate these tracks directly to output by enabling *Preserve Original Audio Tracks* and set the audio codec to Direct Convert.
For a comprehensive explanation of managing audio, see *Processing Multi-Track Audio CML for Flip64*.

---

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) | String; the 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.<br>Example: `<Audio source = "Audio6" />` |

## 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 channels 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" />
  </Segment>
```

# Canvas

Transcode CML Elements | Transcode 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, along with silent audio as appropriate, for the duration specified. Still-frame images are not supported.

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 | Timecode; the amount of time to display the canvas. Default: 0.<br>If you are specifying a relative timecode for video with a timecode track, use these formats:<br>HH:MM:SS:FF@FPS \| HH:MM:SS:FF \| HH:MM:SS;FF<br>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:<br>HH:MM:SS.sss \| HH:MM:SS:FF@FPS<br>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.<br>Rates supported: 23.976, 24, 25, 29.97, 30, 50, 59.94.<br>Timecode references are applied to the timecode track specified in the associated Source element's file.<br>Example:<br>`<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

Transcode CML Elements | Transcode 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 document the composition or to pass information (for example, a key-pair value or other metadata) directly into a workflow for use in the workflow. You can use *Comment* to provide in-line documentation that explains the purpose of the various elements of the Composition, which aid in development, troubleshooting, debugging and maintenance.

Another valuable use of *Comment* is to use it during debugging, by commenting out certain elements to simplify the CML and narrow down the target of problem you're having.

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 TMF transcode actions 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 use a Metadata action to extract the values, parse them as needed, and update variables. The variables are automatically passed downstream where other actions can utilize the variables as needed.

# Composition

Transcode CML Elements | Transcode 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; optional for video-only compositions).

## 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.<br>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

Transcode CML Elements | Transcode 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.

To make all material the same length within a *Segment*, the pre-processor generates silent audio/blank video only at the end of the shorter material, based on the *Edit* configuration. There is no element to explicitly insert black/silence at the beginning of *Video* or *Audio* in a *Segment*.

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*.<br>Keywords: *absolute* \| *relative* \| *duration*.<br>*absolute* (default)—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.<br>*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).<br>*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.<br>**Note:** You can't use *duration* in both *Head* and *Tail* in the same instance of material. If you use *duration* in one, do not include it in the other.<br>Examples:<br>`<Edit mode = "relative" time = "00:10:00" />`<br>`<Edit mode = "duration" time = "00:00:30" />` |
| time (optional) | Specifies (by time or timecode) the location of the edit point. Drop frame, non-drop frame and time references are valid.<br>The time attribute is only valid when Edit mode is set to *relative*. It should not be used swhen Edit mode is set to *absolute* or *duration*.<br>Examples:<br>`<Edit mode = "relative" time = "01:00:10" />`<br>`<Edit mode = "relative" time = "01:27;10" />`<br>`<Edit mode = "relative" time = "00:00:10.000" />`<br>`<Edit mode = "relative" time = "01:00:00;03@29.97" />` |

# Example

This *Segment* illustrates trimming the *Video 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

Transcode CML Elements | Transcode CML Hierarchy Map

The *Fade* element applies a change in intensity to an *Audio* source, based on duration. *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; the shape is linear: it maintains a constant rate of change over the duration of the fade.

---

**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.<br>When specifying a relative timecode for video with a timecode track, use these formats:<br>HH:MM:SS:FF@FPS \| HH:MM:SS:FF \| HH:MM:SS;FF<br>If you are specifying an absolute timecode or time 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:<br>HH:MM:SS.sss \| HH:MM:SS:FF@FPS<br>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.<br>Rates supported: 23.976, 24, 25, 29.97, 30, 50, 59.94.<br>Timecode references are applied to the timecode track specified in the associated Source element's file.<br>Examples:<br>`<Fade duration = "00:00:05.500" />`<br>`<Fade duration = "00:00:05;14@29.97" />` |

## Example

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

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

# File

Transcode CML Elements | Transcode 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.

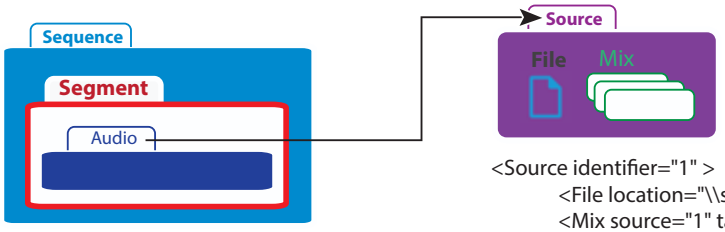
When utilized in a *Subtitle*, the *File* element is optional; one is permitted. If subtitles are embedded, no *File* should be referenced.

---

**Note:** Files referenced in Transcode CML must be on Windows servers, because TMF transcode actions 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 TMF transcode actions.

---

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.

For alternate methods of defining OPAtom MXF files in a *Source*, see *Specifying Avid OPAtom Files in a Segment*.

## 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 TMF transcode action. You may use a drive letter or a UNC path.<br>Locations may be specified as:<br>• *Drive Letter*—`D:\pathname\filename.ext`<br>• *UNC Path*—`\\share\path\filename.ext`<br>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

Transcode CML Elements | Transcode CML Hierarchy Map

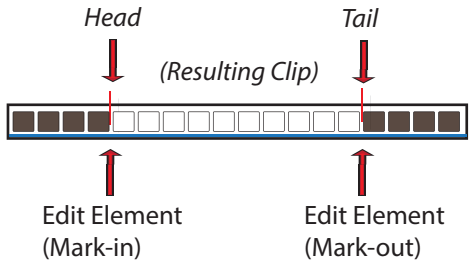
The *Head* element is the point (the first frame) in a *Segment* of *Audio* and *Video* material at the beginning (or mark-in point, when using an *Edit*).

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

Transcode CML Elements | Transcode 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*. A *Target* is required to present audio to the transcode action.

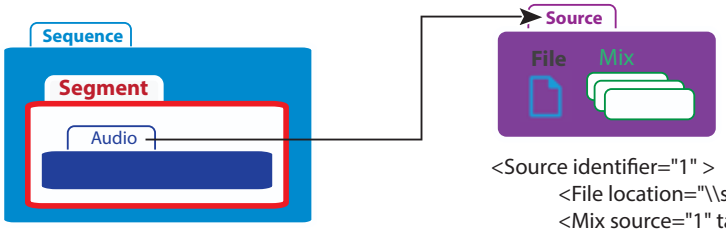
You can optionally adjust the audio gain and also apply a phase shift on a channel-by-channel basis. You can also create silent channels or set channels to silent.

---

**Note:** If there are no *Mix* elements in a *Source*, then all channels in the associated *File* are placed in the in-memory map and into the *Segment* in the same order, beginning at 1. Thus, if you have two or more *Audio* elements in a *Segment* —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 Transcode CML, see *Processing Multi-Track Audio CML for Flip64*.

---

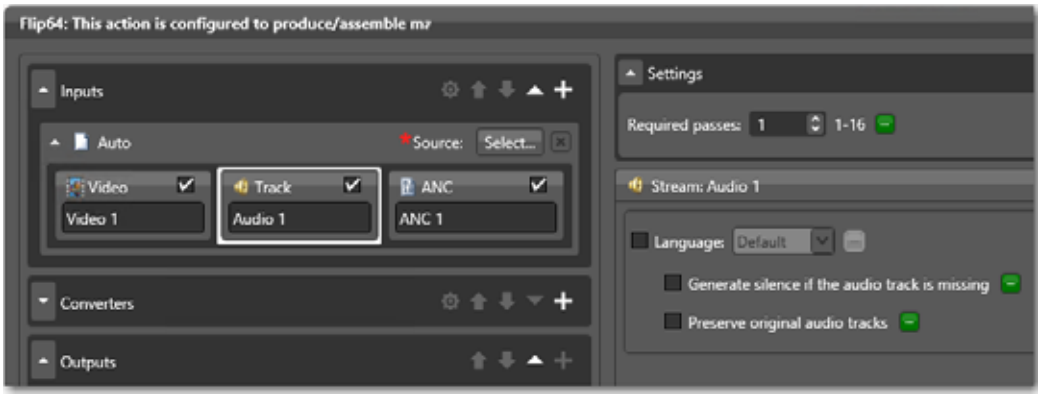
---

**Note:** Audio output from the CML pre-processor is always uncompressed 24-bit PCM. If you plan to utilize a transcode action to encode/re-organize the final output audio,

use a *Target* in the CML to define a single *Track* to be used as a source in the action and configure tracks and channels in the transcoder action to meet your requirements.
In IPTV Flip and Multiscreen Flip, you can only define a single track. For Flip64 only, if you plan to define one or more *Tracks* as required directly in the CML. Then, configure the Flip64 action to propagate these tracks directly to output by enabling *Preserve Original Audio Tracks* and set the audio codec to Direct Convert.
For a comprehensive explanation of managing audio, see *Processing Multi-Track Audio CML for Flip64*.

---

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) | Real \| Integer (default: 1.0); a numeric value specifying the audio level in unit gain, as a percent of the current audio level or in decibels (by adding the unit of measure keyword *dB* to the numeric value).<br>When no unit of measure is provided, the audio level is the multiplier, where 1 (100%) is no change; a value less than 1 reduces the current audio level by that percentage and a value greater than 1 increases the audio level in the similar manner. If you set level to 0, the output is thus silence.<br>When specifying *dB*, where 0 is no change, a negative value reduces the current audio level by the specified decibel value; a positive value increases it by that amount.<br>Examples:<br>`<Mix level = ".707"... />`<br>Here, the current audio level is multiplied by 0.707, thus reducing the value relatively, by 29.3 percent.<br>`<Mix level = "3dB"... />`<br>Here, 3 decibels is added to the current audio level, raising the value absolutely, by 3 decibels. |
| phase (optional) | Real \| Integer; 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.<br>The default value is 0.0 degrees.<br>Phase may be used to implement surround sound matrix encoding (+/-90°) or to correct phase inversions (+/-180°).<br>Example: `<Mix phase = "-90"... />` |

| Name | Description |
|---|---|
| source (required) | Keyword; one or more integers (separated by a space) specifies the channel(s) (beginning at 1) or other key word to utilize from the source file.<br>Keywords: *1* through *32* \| *All*<br>*1...32*—the channel number or numbers to utilize.<br>*All* (default)—use to specify that all source channel(s) should be utilized in order, based on (and limited by) the target channels.<br>Single source channel example:<br>`<Mix source = "1" target = "5" />`<br>You can optionally, define multiple channels in a single *Mix*, by separating each with a space, as shown here:<br>Example: `<Mix source = "1 2" target = "5 6" />`<br>This is functionally equivalent to:<br>`<Mix source = "1" target = "5" />`<br>`<Mix source = "2" target = "6" />`<br>Here, channel 1 source is placed into channel 5 and 2 is placed in 6.<br>All source channel / limited target example:<br>`<Mix source = "All" target = "1 2" />`<br>Here, channel 1 and 2 from the source is placed into channel 1 and 2, since it is the limiting attribute. Other source channels are ignored—they are not propagated.<br>Specific source / all target example:<br>`<Mix source = "1 2" target = "All" />`<br>Here, the specific source channels 1 and 2 are placed repetitively into all of the target channels: (\|1\|2\|1\|2\|1\|2\|….\|1\|2\| <br>All source and target example:<br>`<Mix source = "All" target = "All" />`<br>Here, all source channels (up to 32) in order, are placed into the same target channels. |
| target (required) | Integer; one or more value (separated by a space) specifies the ordinal index—channel number—of the corresponding channel(s) specified in the *source* attribute.<br>Keywords: *1* through *32* \| *All*<br>*1...32*—the channel number or numbers to utilize.<br>*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.<br>Example: `<Mix source = "3" target "1" level = ".9"/>`<br>In this example, channel 3 in the source is identified as channel 1.<br>All Target Example: `<Mix source = "1 2" target = "All" />`<br>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

Transcode CML Elements | Transcode CML Hierarchy Map

At least one *Segment* must be included in a *Sequence*; multiple are permitted. Each *Segment* represents one clip on a video timeline and 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) and the duration is determined by the duration of the longest material item.

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

*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 a typical use of a *Segment*; video and audio are from the same source. Notice the required *mute* attribute in *Video* for correct processing in TMF transcode actions. The video and audio are 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

Transcode CML Elements | Transcode 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

Transcode CML Elements | Transcode 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 itself, and optionally, audio channel selection and organization, and subtitle information.

The media file must be in a media container and contain video and/or audio. Transcode CML does not support still frame images as a *Source*, although you can auto-generate slates using *Canvas*.

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

If you plan to use the video stream, you must specify how to process it, via the *instructions* attribute.

By default, all audio channels are available, 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*. *Tracks* are defined in the *Target*. 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 | A string value, which must be unique among all *Source* elements. By convention, it usually is an index, beginning at 1. You could also add descriptive text for clarity.<br>Examples:<br>`<Source identifier = "1">`<br>`<Source identifier = "video1">`<br>This identifier value is used in the *source* attribute of *Video* \| *Audio* elements to reference and utilize media from this file. |
| ignore-user-data | Boolean; keywords *true* \| *false* (default). Sets the priority of caption extraction for sources with HEVC/H265, AVC/H264 and MPEG-2 video essence. When set to *true*, captions in the video essence are ignored and are extracted from other available locations (SMPTE 436 ancillary data in MXF sources, caption tracks in QuickTime sources, etc.). When set to false or omitted, captions in the video essence have priority over all other locations. |
| instructions (optional) | Keywords: *force-direct-convert* \| *no-direct-convert* (default: *no-direct-convert)* which specifies whether the *Source* file's video should or should not be direct-converted for ingest by the transcoder. If you do not supply the instructions attribute, the default setting (*no-direct-convert*) is used.<br>This attribute is ignored for audio-only source material.<br>The keyword must agree with the transcoder configuration: If you are transcoding the video, you may specify *no-direct-convert* (for clarity), eliminate the instructions attribute, or set it to nil (""). If you are direct-converting the video, you must specify *force-direct-convert*.<br>The keyword *force-direct-convert* is required in all *Source* instances that utilize video, when the transcoder action is direct-converting video.<br>Flip64 specifies direct-conversion explicitly: the Output > Video > Codec setting is *Direct Convert*. However, IPT VOD and Multiscreen Flip actions direct-convert implicitly, by not supplying a Transcoder component.<br>If your transcoder configuration conflicts with this setting, the job will fail with an error.<br>Example: `<Source identifier = "1" instructions = "force-direct-convert">` must be used when the transcoder is configured to direct-convert the video. |

## 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 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

Transcode CML Elements | Transcode 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 the TMF transcode action. Whether the transcode action propagates them to output depends on how it is configured. If it 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) | Timecode; 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.<br>Default: `"00:00:00.000"`.<br>If you are specifying a relative timecode for video with a timecode track, use these formats:<br>HH:MM:SS:FF@FPS \| HH:MM:SS:FF \| HH:MM:SS;FF<br>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:<br>HH:MM:SS.sss \| HH:MM:SS:FF@FPS<br>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.<br>Rates supported: 23.976, 24, 25, 29.97, 30, 50, 59.94.<br>Timecode references are applied to the timecode track specified in the associated Source element's file.<br>Example: `<Subtitle offset = "01:43:05.333"... />` |
| preserve708 (optional) | Boolean; keywords true \| false 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.<br>Default: `preserve708 = "false"`.<br>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.<br>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

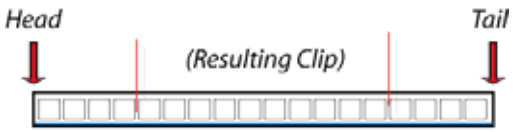
Transcode CML Elements | Transcode CML Hierarchy Map

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*).

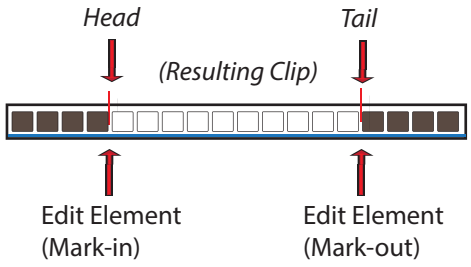
The corresponding *Head* element is the point (or frame) at the beginning (or mark-in point, when using an *Edit*) of the instance. *Tails*—as well as *Heads*—have no duration; they represent either edge of the material.

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 audio of the same duration and an audio fade.

```
<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

Transcode CML Elements | Transcode CML Hierarchy Map

One *Target* element is required in a *Composition* to include audio in the output media. If you are creating video-only output, you should only include *Target* when you want to override the timecode, and it should not include *Tracks*.

The *Target* element utilizes *Track* elements to organize the *Sequence's* collective set of audio channels into tracks for processing in TMF transcode actions, 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 a transcode action to encode/re-organize the final output audio, use a *Target* in the CML to define a single *Track* to be used as a source in the action and configure tracks and channels in the transcoder action to meet your requirements.
In IPTV Flip and Multiscreen Flip, you can only define a single track. For Flip64 only, if you plan to define one or more *Tracks* as required directly in the CML. Then, configure the Flip64 action to propagate these tracks directly to output by enabling *Preserve Original Audio Tracks* and set the audio codec to Direct Convert.
For a comprehensive explanation of managing audio, see *Processing Multi-Track Audio CML for Flip64*.

---

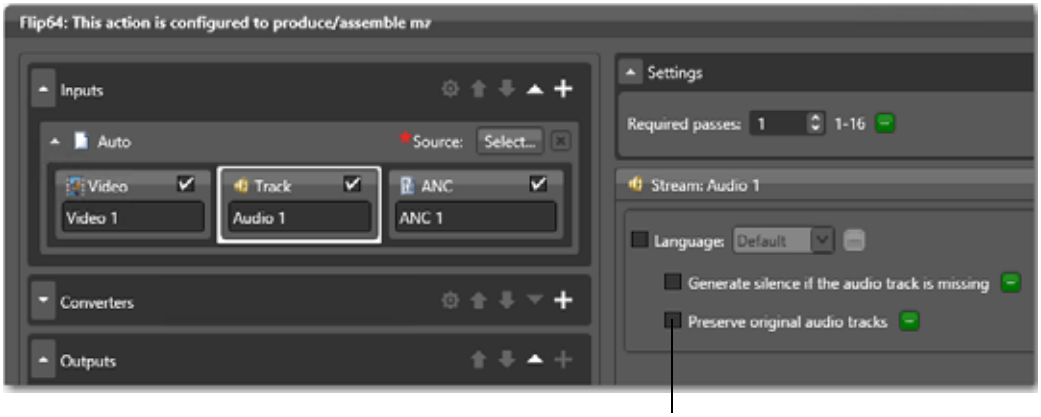
## Configuring Audio Tracks in Flip64

If you plan to organize or re-organize tracks and channels in Flip64 instead of performing the same task directly in CML, a *Target* element is required in the CML which define a single audio *Track*. Configure it to contain all of the channels present in all tracks that you want to utilize in the output.

## Configuring Tracks in CML

However, if you plan to configure the audio directly in the CML (*Processing Multi-Track Audio CML for Flip64*), 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.

In the Flip64 action, uncheck *Preserve Original Audio Tracks*, which is located in the Input component's track:




Check Preserve original audio tracks

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


There are no attributes in a *Target*.

## Child Elements

- *Track*—required to conform media that includes audio; multiple permitted. For video-only media, do not include.
- *Timecode*—required to override the timecode if present; one permitted. If *Timecode* is not specified, source timecode is propagated, which is the functional equivalent of specifying *Timecode* with "source".

## 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

Transcode CML Elements | Transcode 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 TMF transcode actions should be disabled or set to *Use Source Timecode*. In this manner, the *Timecode* specification is propagated directly to the final 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*.<br>The *source* keyword propagates the existing timecode from the source into the output without changing its value.<br>Example: `<Timecode type = "source" />`<br>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.<br>Example: `<Timecode type = "start" time = "01:00:00;00@29.97" />` |
| time | Specifies the starting timecode to utilize, when Timecode `type = "start"` is specified.<br>If you specify a relative timecode for video with a timecode track, use these formats:<br>HH:MM:SS:FF@FPS \| HH:MM:SS:FF \| HH:MM:SS;FF<br>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:<br>HH:MM:SS.sss \| HH:MM:SS:FF@FPS<br>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.<br>Rates supported: 23.976, 24, 25, 29.97, 30, 50, 59.94.<br>Timecode references are applied to the timecode track specified in the associated Source element's file.<br>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

Transcode CML Elements | Transcode CML Hierarchy Map

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

For IPTV and Multiscreen workflows, one, and for Flip64, one or more *Track* elements must be included in a *Target* to specify each audio track in the output. For video-only output, *Track* is irrelevant and should be omitted.

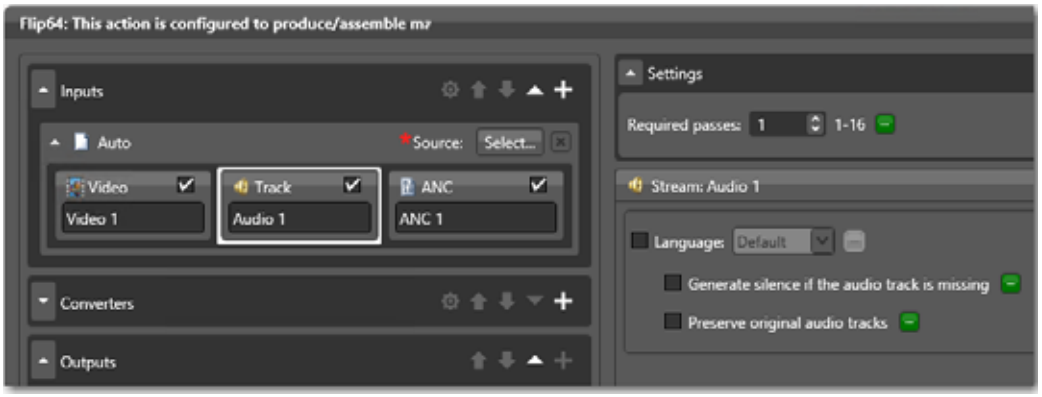
If you skip a channel in the target attribute, the skipped channels are created with silence.

---

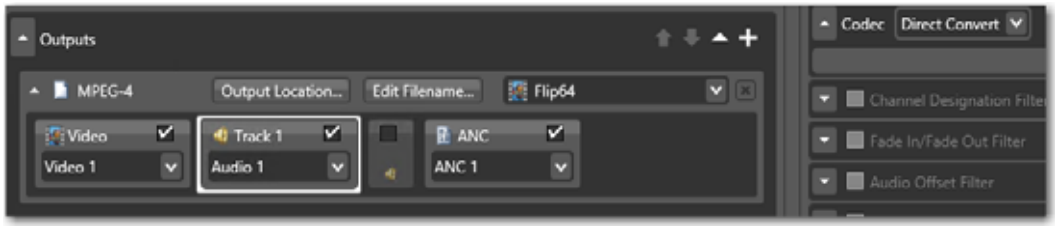
**Note:** Audio output from the CML pre-processor is always uncompressed 24-bit PCM. If you plan to utilize a transcode action to encode/re-organize the final output audio, use a *Target* in the CML to define a single *Track* to be used as a source in the action and configure tracks and channels in the transcoder action to meet your requirements.
In IPTV Flip and Multiscreen Flip, you can only define a single track. For Flip64 only, if you plan to define one or more *Tracks* as required directly in the CML. Then, configure the Flip64 action to propagate these tracks directly to output by enabling *Preserve Original Audio Tracks* and set the audio codec to Direct Convert.
For a comprehensive explanation of managing audio, see *Processing Multi-Track Audio CML for Flip64*.

---

Multiple tracks may be mixed directly in CML for processing in Flip64, but you must enable the *Preserve Original Audio Tracks* control in the Flip64 > Auto Input > Track dialog:



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 integers (1...32, separated by a space), that identifies each channel in the *Sequence* that comprises this *Track*, base 1. Source values must be unique.<br>Example: `<Track source = "3 4" target = "1 2" />`<br>Here, this track is comprised of the *Sequence's* channels 3 and 4. |
| target | Integer; one or more integers (1...32, separated by a space), to identify the ordinal sequence—channel number—of the specified output channel(s) in this *Track*, base 1. Track values must be unique.<br>Example: `<Track source = "3 4" target = "1 2" />`<br>Here, this track has two channels—1 and 2 (which are channels 3 and 4 in the *Source*). |

## Flip64 Example

**Note:** This example only applies to Flip64, because you can configure it to preserve original audio tracks, as defined by the *Target* element's *Track* elements. If you submit this to Multiscreen Flip or IPTV Flip, only the first *Track* is recognized.

In this Flip64 example, the MXF file has video and six audio tracks, one channel each. The *Mix* elements define how the channels are ordered 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. *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 a transcode action to encode/re-organize the final output audio, use a *Target* in the CML to define a single *Track* to be used as a source in the action and configure tracks and channels in the transcoder action to meet your requirements.
In IPTV Flip and Multiscreen Flip, you can only define a single track. For Flip64 only, if you plan to define one or more *Tracks* as required directly in the CML. Then, configure the Flip64 action to propagate these tracks directly to output by enabling *Preserve Original Audio Tracks* and set the audio codec to Direct Convert.
For a comprehensive explanation of managing audio, see *Processing Multi-Track Audio CML for Flip64*.

---

## IPTV Flip | Multiscreen Flip Example

In this example, the MXF file has video and six audio tracks, one channel each. The *Mix* elements define how the channels are ordered when this *Source* is referenced in a *Segment*. The *Target* specifies one track to be presented for processing in IPTV Flip | Multiscreen Flip. The track contains 6 channels in logical order: 1-2 and 3-4, and 5-6.

```
<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 3 4 5 6" target = "1 2 3 4 5 6" />
  </Target>
</Composition>
```

Three stereo tracks are presented to the transcoder action in a single track. If you want multiple tracks in your output from the transcoder action, add a Converter > Audio Mixer and configure it as required

---

**Note:** Audio output from the CML pre-processor is always uncompressed 24-bit PCM. If you plan to utilize a transcode action to encode/re-organize the final output audio, use a *Target* in the CML to define a single *Track* to be used as a source in the action and configure tracks and channels in the transcoder action to meet your requirements.
In IPTV Flip and Multiscreen Flip, you can only define a single track. For Flip64 only, if you plan to define one or more *Tracks* as required directly in the CML. Then, configure the Flip64 action to propagate these tracks directly to output by enabling *Preserve Original Audio Tracks* and set the audio codec to Direct Convert.
For a comprehensive explanation of managing audio, see *Processing Multi-Track Audio CML for Flip64*.

---

# Video

Transcode CML Elements | Transcode 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). ANC, if present, is automatically included in the output, without a specific reference.

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).

To make all material the same length within a *Segment*, the pre-processor generates silent audio/blank video at the end of the shorter material, based on the *Edit* configuration. There is no element to explicitly insert black/silence at the beginning of *Video* or *Audio* in a *Segment*.

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

---

**Note:** *Video* that includes audio in the same file MUST include the `filter="mute"` attribute, to mute the associated audio, 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.<br>Keyword: *mute*.<br>Example: `<Video source = ”0” filter = "mute" />` |
| source | String; corresponding to the *Source*'s *identifier* attribute value, to identify the file that contains this video stream.<br>Example: `<Video source = ”video1” filter = "mute" />` |

# 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>
```

# Prototype Applications

This chapter provides examples of compositions and associated workflows that may be used in a wide range of Flip64 applications.

- Mapping and Mixing Audio
- Specifying Avid OPAtom Files in a Segment
- Processing Multi-Track Audio CML for Flip64
- Building a Syndicated Show

All of the examples in this guide are complete compositions with associated workflows. To test them, you can copy and paste the example into a text editor. Change the references to the media files you are supplying, and save the file. Then, submit it to a suitable workflow.

A library of composition examples and media generated by the composition are provided to illustrate various features of Post Producer. To access these examples, go to the www.telestream.net Web site and log in as an authorized user.

Click on the Post Producer link and select the CML examples tab to view each CML example, which includes the CML file, the exported workflow it can be processed with, source media, and a sample proxy video clip that was produced from it.

# Mapping and Mixing Audio

One of the key features associated with clipping, or conforming EDLs, is the ability to define, map, and mix the associated audio as required for your output. Transcode CML enables you to map and mix the audio in preparation for output by Flip64.

This example CML illustrates how to map and mix audio. Here are the details:

- Sources are defined in a single OP1a MXF file: an MXF with one video track, 8 mono audio tracks and a 436M ANC track.
- This CML utilizes a set of OP-atom sources from a simple Avid export: 1 video OP-atom source, 8 mono audio OP-atom sources, and 1 ANC OP-atom source.
- Direct-convert mode is enabled (`<Source instructions="force-direct-convert"...`) as required, because the Flip64 action in the workflow is also set to direct convert the *Video*.
- Captions processing is enabled via *Subtitle* so that the captions are passed through to the output.
- There are three segments in this sample:
  - Trimmed main video
  - Black slate (Canvas)
  - Trimmed black slate
- In/Out- points are specified via Head and Tail with Edit elements.
- Output timecode is overridden—`<Timecode time="01:00:00;00@29.97" />`.
- 2 audio tracks are created:
  - source channels 1- 6
  - source channels 7, 8

## Audio Configuration Details

Mix elements are used to map input audio tracks to output audio tracks. This layout must be referenced in the Target element to configure the output tracks. In this example, the Mix has 8 channels:

- Channel 1 is sourced from channels 1 in [audio_opa_1.mxf], ..., channels 8 is sourced from channel 1 in [audio_opa_8.mxf]
- Source channels in [<Track source="x x"/>] refer to "in-memory" audio layout created with <Mix> instructions.
- Target channels in [<Track target="x x"/>] specify where "in-memory" channels should be placed within an output track.
- Channel indexes specified in [<Track source="x x"/>] and [<Track target="x x"/>] must be unique. The following configs are illegal:
- [<Track source="1 1 2 3 4"/>]
- [<Track target="1 2 3 3 4 4"/>]

### Example 1

<Track source="1 2" target="5 6" /> will create a 6 channel output track with empty channels 1-4 and channels 5 and 6 having the audio from in-memory channels 1 and 2.

### Example 1

Let's assume that we have the following channels lineup in a source clip: [5.1 (ch1 ch2 ch3 ch4 ch5 ch6)] + [English Mono (ch7)] + [Spanish Mono (ch8)] and want to create these 3 output tracks: [

- An English Dual Mono track
- A Spanish Dual Mono track
- 5.1 Surround Sound track

We will need to:

1. Create an in-memory layout using <Mix> where [source="x"] is the channel index within the source and [target="x"] is the corresponding channel index "in-memory":

<Mix source="1" target="5" />

<Mix source="2" target="6" />

<Mix source="3" target="7" />

<Mix source="4" target="8" />

<Mix source="5" target="9" />

<Mix source="6" target="10" />

<Mix source="7" target="1" />

<Mix source="7" target="2" />

<Mix source="8" target="3" />

<Mix source="8" target="4" />

2. Create output tracks where [source="x x"] refers to "in-memory" channels and [target="x x"] specifies where to place [source="x x"] channels within an output track

<Track source="1 2" target="1 2" />

<Track source="3 4" target="1 2" />

<Track source="5 6 7 8 9 10" target="1 2 3 4 5 6" />

## Example

```
<Composition xmlns="Telestream.Soa.Facility.Playlist">
  <Source identifier="1" instructions="force-direct-convert">
    <File location="c:\footage\video_opa.mxf" />
    <File location="c:\footage\audio_opa_1.mxf" />
    <File location="c:\footage\audio_opa_2.mxf" />
    <File location="c:\footage\audio_opa_3.mxf" />
```

```
    <File location="c:\footage\audio_opa_4.mxf" />
    <File location="c:\footage\audio_opa_5.mxf" />
    <File location="c:\footage\audio_opa_6.mxf" />
    <File location="c:\footage\audio_opa_7.mxf" />
    <File location="c:\footage\audio_opa_8.mxf" />
    <File location="c:\footage\anc_opa_8.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"/>
    <Mix source="7" target="7"/>
    <Mix source="8" target="8"/>
    <Subtitle preserve708="false" />
  </Source>
  <Sequence>
    <Segment>
      <Video source="1" filter="mute">
        <Head>
          <Edit mode="absolute" time="00:01:00.060" />
        </Head>
        <Tail>
          <Edit mode="absolute" time="00:02:00.053" />
        </Tail>
      </Video>
      <Audio source="1">
        <Head>
          <Edit mode="absolute" time="00:01:00.060" />
        </Head>
        <Tail>
          <Edit mode="absolute" time="00:02:00.053" />
        </Tail>
      </Audio>
    </Segment>
    <Segment>
      <Canvas layer="0" duration="00:00:05.000" />
    </Segment>
    <Segment>
      <Video source="1" filter="mute">
        <Head>
          <Edit mode="absolute" time="00:03:00.046" />
        </Head>
        <Tail>
          <Edit mode="absolute" time="00:04:00.039" />
        </Tail>
      </Video>
      <Audio source="1">
        <Head>
          <Edit mode="absolute" time="00:03:00.046" />
        </Head>
        <Tail>
          <Edit mode="absolute" time="00:04:00.039" />
        </Tail>
      </Audio>
    </Segment>
  </Sequence>
```

```
  <Target>
    <Timecode time="01:00:00;00@29.97" />
    <Track source="1 2 3 4 5 6" target="1 2 3 4 5 6" />
    <Track source="7 8" target="1 2" />
  </Target>
</Composition>
```

# Specifying Avid OPAtom Files in a Segment

This method of specifying a *Source* is a special case for a set of separated OPAtom MXF files that are treated as a single source. This construct is functionally the same as specifying a single, self-contained OP1a MXF file that includes video, audio, and 436M ANC metadata.

ANC, if present, is automatically included in the output, without a specific reference.

---

**Note:** This method of specifying a *Source* should not be used with other formats.

---

```
<Source identifier="1" instructions="force-direct-convert">
  <Comment> Video stream (note the file name video~)</Source </
Comment>
  <File location="c:\footage\video_opa.mxf" />
  <Comment> 8 audio streams (note the file name audio~) </Comment>
  <File location="c:\footage\audio_opa_1.mxf" />
  <File location="c:\footage\audio_opa_2.mxf" />
  <File location="c:\footage\audio_opa_3.mxf" />
  <File location="c:\footage\audio_opa_4.mxf" />
  <File location="c:\footage\audio_opa_5.mxf" />
  <File location="c:\footage\audio_opa_6.mxf" />
  <File location="c:\footage\audio_opa_7.mxf" />
  <File location="c:\footage\audio_opa_8.mxf" />
  <Comment> 1 ANC stream (note the file name anc~) </Comment>
  <File location="c:\footage\anc_opa_8.mxf" />
</Source>
```

This *Source* definition is functionally the same as:

```
<Source identifier="1" instructions="force-direct-convert">
  <File location="c:\footage\OP1a_all.mxf" />
</Source>
```

# Processing Multi-Track Audio CML for Flip64

In most cases, audio is presented as a single track to Flip64, Multiscreen Flip and IPTV Flip, and any track and channel mixing is performed directly in the action—it is not defined in CML.

The purpose of this topic is to describe an alternative method of defining audio directly in CML and using the CML preprocessor to produce a single tack or multiple tracks for use in Fip64 action workflows only, where the Flip64 action is configured to preserve original audio tracks and is set for direct-convert transcoding.

Here, you'll learn how the CML pre-processor processes audio, and the Transcode CML elements associated with audio processing, by way of an example.

---

**Note:** This topic focuses on ingesting multi-track audio sources and re-arranging the audio to create a single track or multiple new tracks using Transcode CML in Flip64. Multiscreen Flip and IPTV Flip always ingest single track audio and must perform any mixing directly in the action.
Regardless of using CML to mix audio or performing it directly in Flip64, Multiscreen Flip or IPTV Flip, you must always specify you Input as Auto.

---

When you have multiple tracks (which may have multiple channels) in your source, you can configure the CML to combine all channels into one or more tracks, ordinally. That is, if Track 1 has 2 channels, then the first channel in track 2 becomes channel 3, etc. You can also present the audio to the transcode action without any changes, preserving each track and its channels in the source.

To utilize the audio and re-configure tracks in CML, you must define the audio sources and then add them on a clip-by-clip basis. Finally, you organize channels into one or more tracks for output.

Commonly, the requirement is to create one track. This example creates two tracks, but you use the same principles to create a single track.

Here are the typical steps:

1. First, in CML you define your 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. Next, you assign sources to segments to arrange them in a sequence on the timeline.
3. Finally, you organize all channels into one or more tracks prior to presenting the output to the Flip64 action.

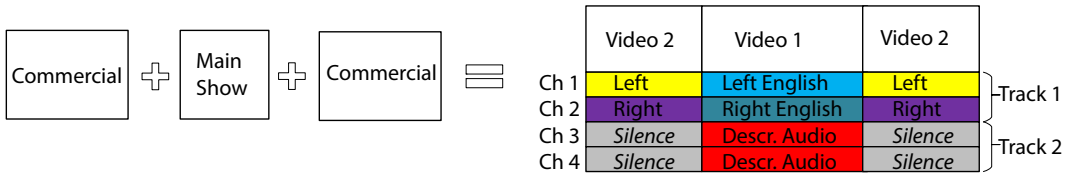
This example illustrates audio processing typically associated with TV show production.

In this example, you'll learn how to:

- Select audio channels from various source files and rearrange them in CML as required for your output, when the audio is arranged differently 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 structure of the media output for processing by the Flip64 transcoder:




To define the media you want produced in CML, follow these steps:

1. Specify the assets to be used for the media (*Specifying the Source Assets*) .
2. Select and arrange the audio channels as required (*Selecting and Rearranging Audio Channels*).
3. Create a sequence of segments to create the timeline (*Adding Sources to Segments on the Timeline*).
4. Organize the sequence's audio into tracks (*Organizing Channels into Tracks*).

The CML output—the conformed, stitched media—is passed to the transcoder for transcoding as required to generate the correct output media as specified.

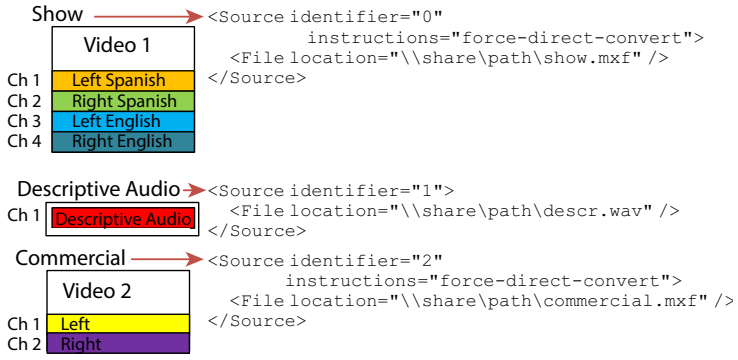
## Specifying the Source Assets

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, in a single track) are used in Flip64. If you require multiple tracks, a subset of channels and/or require them in a different order, you may specify them in CML. 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.

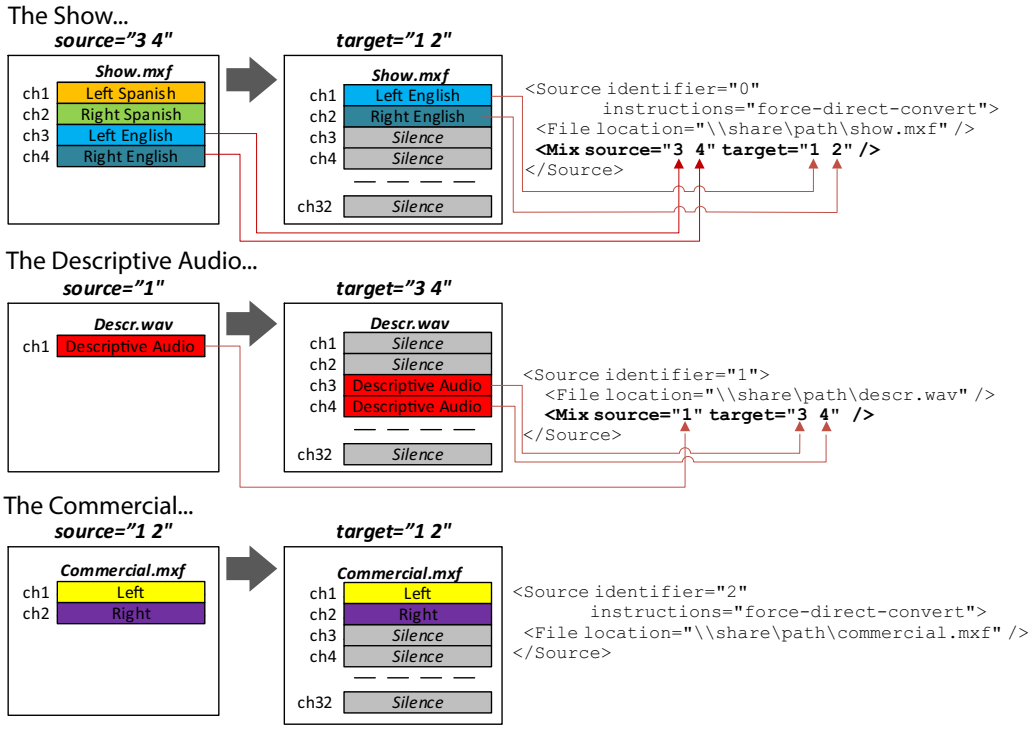
This graphic depicts each file's video and audio, and the CML Source element that identifies the file:




Note that each *Source* has a unique ID, so you can use the *Source* in 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. When you are using multiple tracks in Flip64, you must set the instructions to *force-direct-convert*. The audio can not be encoded in the same action in this situation.

# Selecting and Rearranging Audio Channels

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, along with their new channel order. To do that, add *Mix* elements as shown here:




From *Source* 0 (the show itself), the *Mix* element specifies that only channels 3 and 4 should be utilized—as channel 1 and 2 in the output—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.

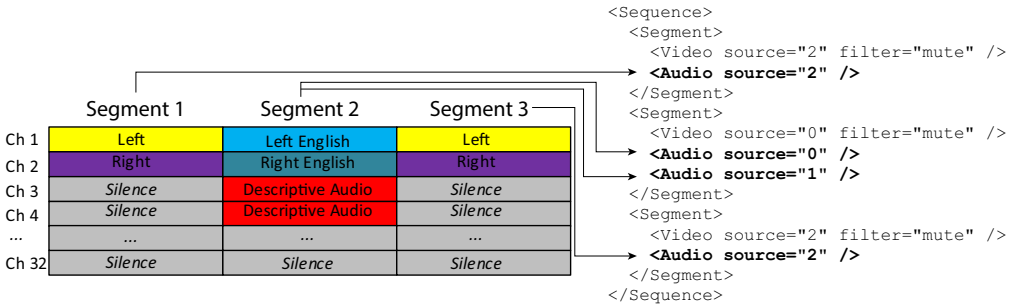
In *Source* 2, notice that there is no *Mix* element. By default, if all audio should be used, in the same channel order, no *Mix* is required. You could specify it (here, as `<Mix source="1 2" target="1 2" />`), but it's redundant.

# 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.



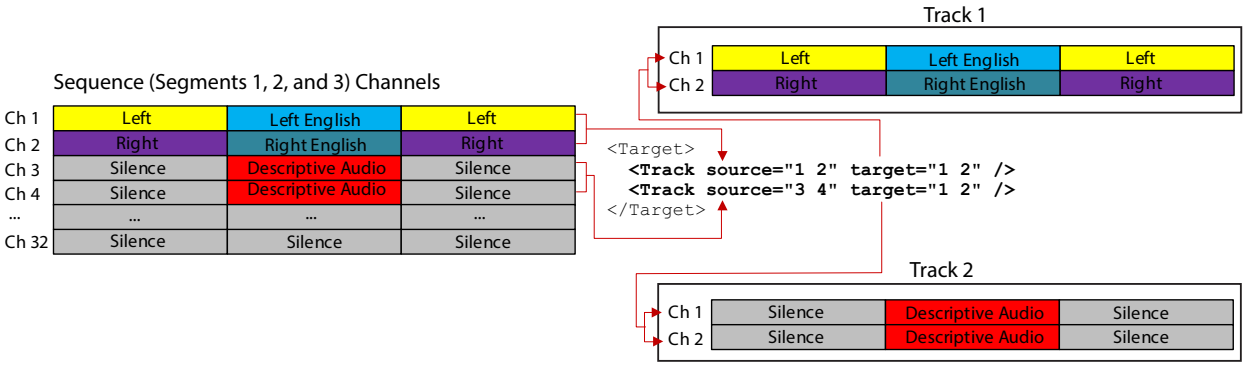

Next, for each *Segment*, add *Video* and *Audio* elements, as shown:

- 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 Transcode CML. Here's the 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>
```