©

Post Producer CML Developer's Guide

Vantage 8.0 + ComponentPac 8.0.0.276247 and later

telestream

October 2022 328005

Contacting Telestream

Resource	Contact Information
Vantage Technical Support	Web Site: http://www.telestream.net/telestream-support/vantage/support.htm Support Email: support@telestream.net Enterprise Telephone Support: U. S. Toll Free: (877) 257-6245 U. S. from outside U.S.: (530) 470-2036 Europe | Middle East | Africa | Asia | Pacific: +49 228 280 9141 Terms and times of support services vary, per the terms of your current service contract with Telestream.
Telestream, LLC	Web Site: www.telestream.net Sales and Marketing Email: info@telestream.net Telestream, LLC 848 Gold Flat Road, Suite 1 Nevada City, CA USA 95959
International Distributor Support	Web Site: www.telestream.net See the Telestream Web site for your regional authorized Telestream distributor.
Telestream Technical Writers	Email: techwriter@telestream.net Share comments about this or other Telestream documents.

Copyrights and Trademark Notices

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.

Contents

Composition Markup Language Reference 103

Post Producer Overview

Post Producer is a programmatic method of performing simple video editing, often automating work commonly performed in an NLE. The purpose of this guide is to help producers, editors, operators, and others involved in promotion production learn how to use Post Producer in Vantage to automate repetitive editing and encoding of assets.

Note: This guide is written for video professionals who are familiar with using Vantage. To use Post Producer effectively in Vantage, you should know how to create workflows and submit jobs. If you aren't familiar with Vantage, we suggest that you review the Vantage User Guide.
This guide includes examples in XML format. A basic understanding of XML is helpful, but alternate methods of Post Producer composition creation are also noted.

Learning to use Post Producer to effectively design, develop, and implement compositions that describe media involves a variety of skills, depending on the preferred method for creating automation compositions.

Familiarity with Vantage is essential. The language of Post Producer—Composition Markup Language (CML)—is a powerful and comprehensive media descriptor language. Telestream recommends that you read the first four chapters in sequence before you begin working with compositions. The remaining chapters are reference material you can turn to when you need information.

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

Topics

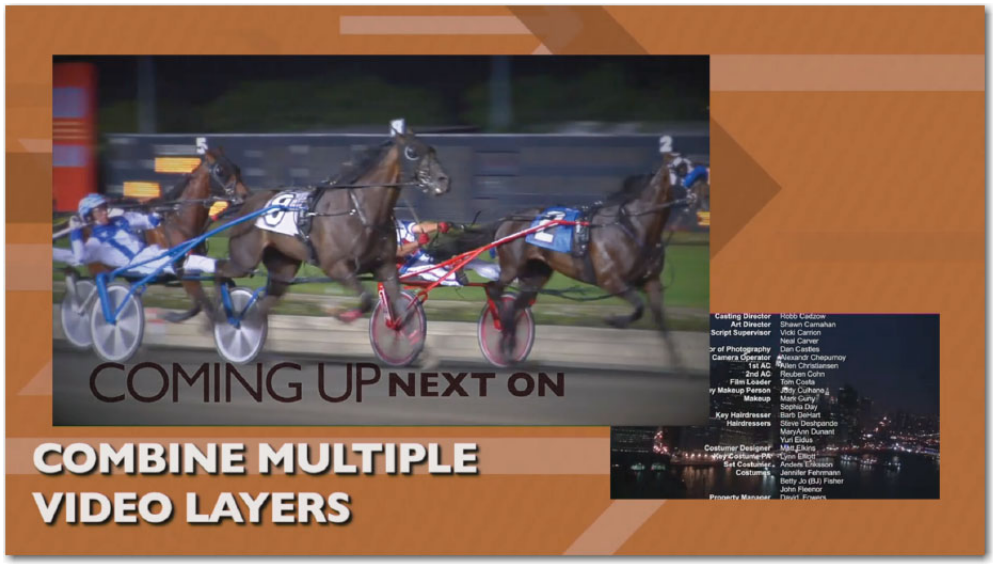
Note: A companion guide, the Post Producer Cookbook, illustrates typical applications for Post Producer, and includes CML examples.

Introduction

Post Producer™ is software for Vantage—Telestream's transcoding and workflow management platform—that automates high-volume, repetitive media production processes. It accomplishes this by assembling often-complex, multi-layered video from common/re-usable mezzanine media sources, based on a media design template.

Post Producer reduces promo production costs by eliminating operator-intensive labor in the edit bay, and by using automated workflows to perform repetitive tasks such as adding bumpers or ad inserts, graphics, audio tracks, lower-third tag lines, etc.

Automated production of iterative content such as promos and regional ads, also increases efficiency—allowing you to scale up your distribution capacity. This is accomplished by employing Vantage's comprehensive workflow design and execution capabilities, job management, and encoding scalability, including accelerated encoding on GPU-accelerated Telestream Lightspeed servers.



Typical applications include:

- Broadcast, cable, and IPTV VOD and promo production and insertion
- Automated news and weather report assembly
- International localization
- Assembly and insertion of bumpers, branding graphics, legal notifications, etc.—often referred to as *bag & tag* operations.

Post Producer is implemented as an optional, licensed feature integrated directly in Vantage as actions—which enables you to build and execute Post Producer workflows.

Post Producer allows you to automatically generate multi-layered video, mixing different resolutions, formats, and frame rates. You can also simultaneously composite video, title text, images, audio, content advisories, and subtitles.

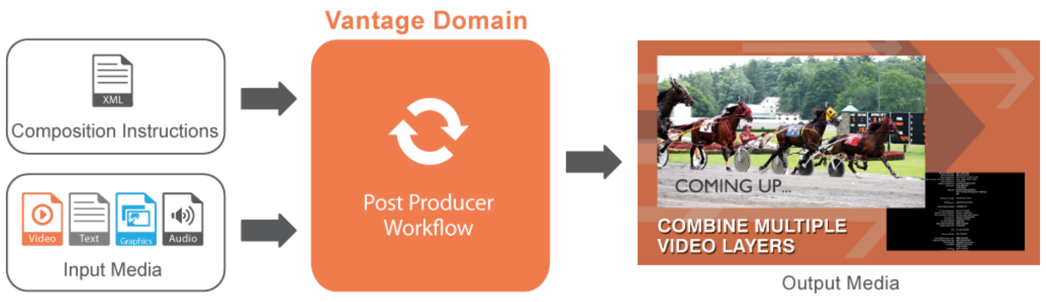
Post Producer renders media from the source media file identified in a media design template, including title text, images, audio, content advisories, and subtitles, into a single media file, producing unlimited variations of a promo in any Vantage-supported format/container and media format you require.

How Post Producer Works

Post Producer generates new media based on a media design template written in Composition Markup Language (CML)—a comprehensive set of media composition instructions (commonly called a *composition*) in XML format, created by you—using the media resources you specify, as input.

During the process, Post Producer applies display properties to the input media. It composes the video (including audio), with graphics and text overlays in exactly the proper arrangement (both spatially and temporally), as specified in the composition.

Finally, it renders a media file from your input media that conforms to the instructions.



Note: Composition Markup Language does not describe *how* to encode media—it only describes the media itself—including the material in the media, its location and duration, and its layout on the frame, plus certain visual effects.

Creating a media file using Post Producer involves these steps:

1. Creating a composition (a CML file) to describe the media you want to create.
2. Updating the CML with the file-based media resources to use for the specific media you want to generate.
3. Generating the media by processing the CML file in a specially-configured Vantage workflow.

Creating Post Producer Compositions (Templates)

There are several ways to create compositions for Post Producer.

- *Using non-linear editors*—Use an NLE to create a project that includes the desired timing and sequences, add variables to elements that are to be replaced by automation, and export the project as an Apple Final Cut Pro file into a Post Producer watch folder for conversion to CML. For details on using non-linear editors to create compositions, see *Converting NLE Projects and Media Descriptor Files to Compositions*.
- *Developing applications using the Post Producer API*—Develop custom applications that utilize the Post Producer API (Application Programming Interface) and automatically create variable compositions from external media management and editing systems. For details, see *Sample Composition Program*.

  Telestream provides a Post-Producer developer's section on its Web site, which includes downloads of an SDK (Software Developer's Kit) that provide information, frameworks, and examples for software developers to create custom interfaces to work with Vantage and Post Producer.

  For more information, browse the Post Producer page in the Customer Center on the Telestream Web site. (Viewing this page requires a valid Telestream user name and password, which you can create online if you need one.)
- *Creating compositions manually or editing existing compositions*—Create compositions manually, using Post Producer CML (Composition Markup Language) in an XML editor. You can also edit existing composition files using a text editor to change media locations and timing events.

  The power of Post Producer's CML is that it can be created by developers who possess an understanding of the XML language, with virtually unlimited options for manipulating the production of rich compositions to drive powerful, automated workflows for content creation and distribution. For a full description of creating compositions, see *Developing Compositions*.
- *Using Workflow Portal to process playlists of clips*—Use Vantage’s Workflow Portal with a CML configuration to select clips from a Vantage catalog, add or edit metadata, and submit the auto-generated CML to a Vantage workflow.

  Workflow Portal—an easy-to-use operator’s program—enables operator-intervention processing, where you can assemble (stitch) short form clips together to create longer form clips—highlight reels, news segments for VOD, long form shows, etc. While Post Producer compositions can be created by other methods to perform this work, Workflow Portal makes it easy for both technical and non-technical users to assemble content from a quickly assembled playlist.

Note: Workflow Portal also provides an EDL configuration, with similar functionality. The difference between the two is that EDL configurations can only submit jobs to a Flip action workflow. The CML configuration submits jobs to Conform action workflows, which utilize the Telestream Media Framework transcoder, with more features, greater transcoding flexibility, and higher efficiency.

Related Guides

The following guides include a book of practical examples, and guides for other CML-related Vantage products which you might find of value. These guides are posted in the Vantage section of the Telestream web site.

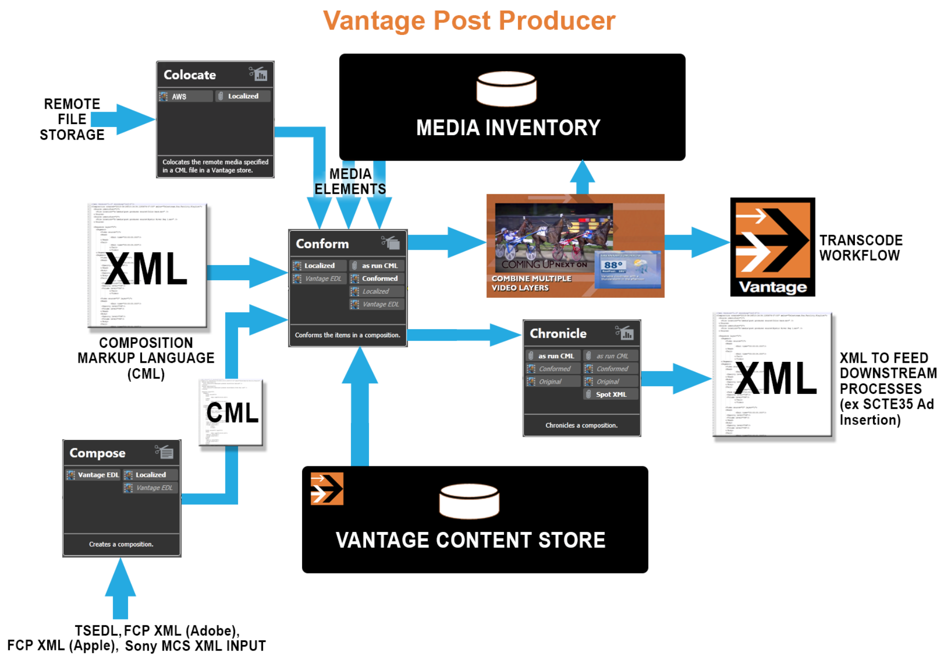
- *Post Producer Cookbook*—provides in-depth examples and explanations of how to use CML in a variety of applications.
- *Creating DPP Packages in Vantage*—this app note describes DPP Packager, which automates the production of Digital Production Partnership (DPP)-compliant television programme packages, ready for broadcast or distribution, using Post Producer CML.
- *VOD Producer | DAI User Guide*—helps Vantage administrators, workflow designers, and operators set up and configure VOD Producer and create VOD ingest and publication workflows using Post Producer CML to generate CableLabs 1.1-compliant VOD assets and deliver them to multi-channel video programming distributors.
- *Transcode CML Developer Guide*—describes how to create edit decision lists (EDLs) using Transcode Composition Markup Language to automatically create simple compositions from same-format media sources or trim and concatenate—*stitch*—clips together to generate new media to your specifications, directly in Flip64 | IPTV Flip | Multiscreen Flip workflows.
- *Tempo CML Developer Guide*—Tempo™ CML is designed specifically for controlling Telestream's Tempo re-timing, transcoding, and workflow management system in Vantage, which automates the process of re-timing media assets. Tempo CML re-times media by automating assembly of media segments that have been designated for re-timing together with segments that are marked as exempt from re-timing to produce a complete asset that meets overall re-timing goals.
- *Timed Text CML Developer Guide*—Timed Text CML is ideally suited for automating repetitive, caption file stitching tasks in a production environment. This guide describes how to use Timed Text CML to automatically concatenate—*stitch*—separate captions together to generate a new caption | subtitle file that corresponds to the media clip specifications, directly in Timed Text Conform workflows.

Using Post Producer in Vantage

This chapter describes how Post Producer is implemented in Vantage. In this chapter you'll learn about Post Producer actions, review a prototypical Post Producer workflow, and review how to submit Post Producer jobs in various ways to generate media.

In addition, you'll learn how to use Post Producer workflows in operator-intervention applications such as assembling playlists, and creating DPP-compliant packages.

Topics

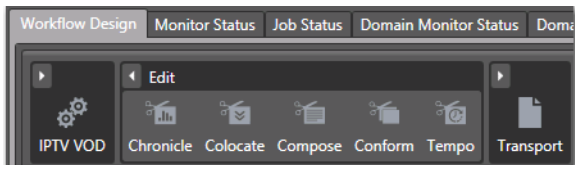


Post Producer Actions

Post Producer is implemented in Vantage via actions, which you can use to build Post Producer workflows. To generate promos and other media from source media as specified by a Post Producer CML file, you process it in a Post Producer workflow that contains a Conform action.

The Tempo action is used to re-encode media while shortening or lengthening the run time of media. For higher encoding performance, the these two actions can be executed on a Lightspeed server. The Tempo action can only process Tempo CML files.

All Post Producer actions are executed by the Vantage Edit Service, and are displayed as a group in Workflow Designer in the Edit category.



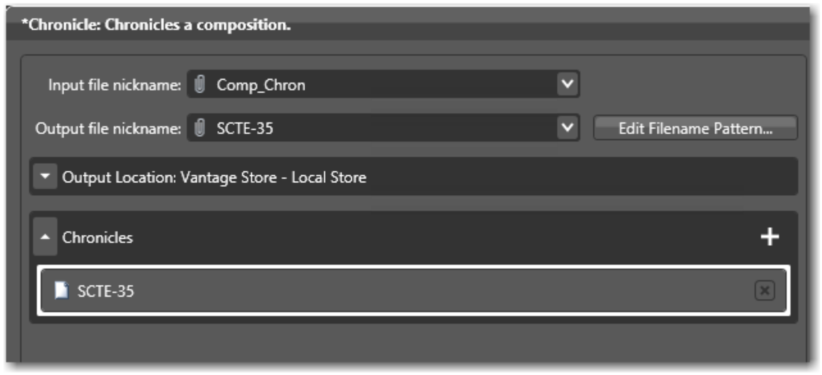
The remaining actions—Compose, Colocate, and Chronicle—are Post Producer actions that provide utility features. They are described later in this chapter.

Note: Like all Vantage actions, you must configure them for use in a specific workflow before you can run jobs through them. For configuration details on each action, see the man page for the selected action in Workflow Designer.

- Chronicle Action
- Colocate Action
- Compose Action
- Conform Action
- Tempo Action

Chronicle Action

The *Chronicle action* is a utilitarian action which transforms *As Run CML files* that have been produced from a Conform action to other formats for use with other processes. This image depicts the Chronicle inspector:


*Chronicle: Chronicles a composition.

Input file nickname: Comp_Chron

Output file nickname: SCTE-35

Output Location: Vantage Store - Local Store

Chronicles

SCTE-35

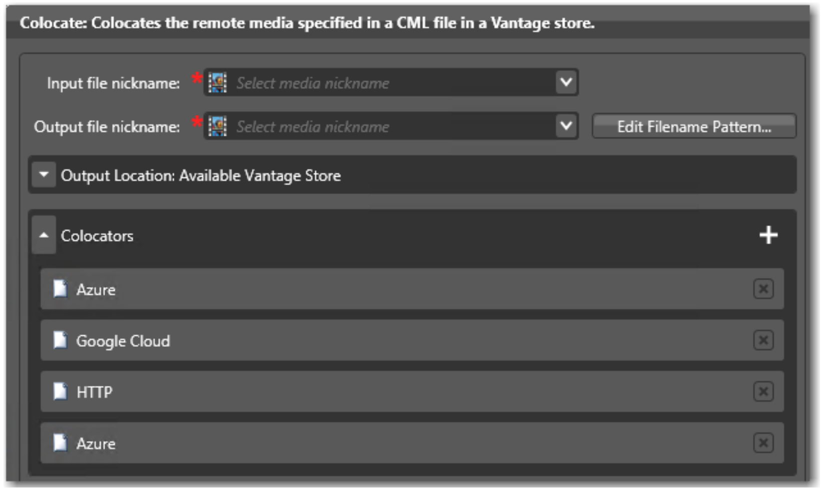
Note: An *As Run CML file* is one that has been produced by a Conform or Tempo action after the file has been conformed (that is, the media file has been produced), and unresolved timecode values are replaced with actual start times for each segment marked as requiring a key frame. Variables have also been fully resolved. To generate an As Run CML file, check the Generate Composition Chronicle checkbox in the Conform action and configure its settings.

You can use the Chronicle action to generate Manzanita-specific SCTE35 files or a Canoe CSV file, for digital program insertion applications. The output file is provided a nickname, for further use in downstream actions.

Colocate Action

The purpose of the *Colocate action* is to ingest CML files with remote file references to source media, localize the files as necessary and then convert the file references into Windows file paths in a copy of the CML.

Here is the Colocate inspector:



Colocate: Colocates the remote media specified in a CML file in a Vantage store.

Input file nickname: * Select media nickname

Output file nickname: * Select media nickname

Colocate operates on non-Windows local paths, S3 Amazon AWS paths, and HTTP server paths. For files on S3 and HTTP servers, you supply credentials in the Colocator component as appropriate, and the files are copied to the specified Vantage-accessible location. For local non-Windows files (those on a Mac OS X platform), the path is translated into a Windows-accessible (share or mapped driver-letter) path without relocating the file.

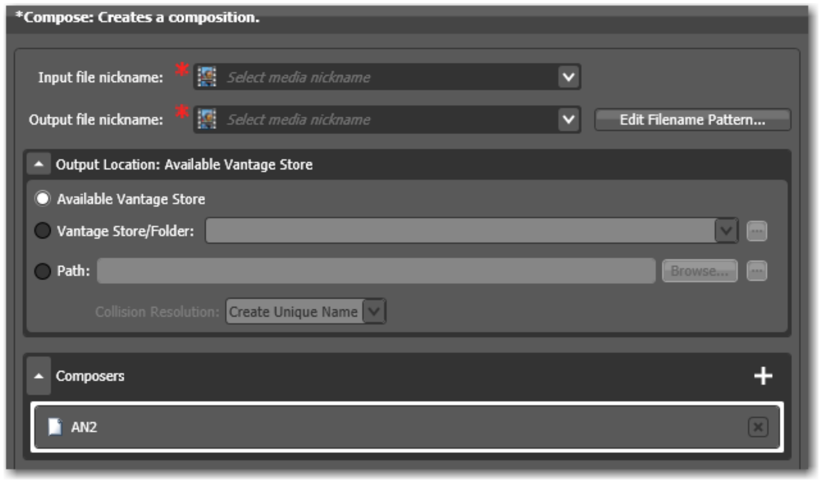
After the files have been localized and the CML has been converted, it can be submitted to the Conform or Tempo action to compose the media. You could create one workflow with both a Colocate and a Conform or Tempo action, or you could create two workflows—one to localize the media and convert the file (Colocate), and the second with a Conform or Tempo action, to conform and transcode it.

Note: HTTPS paths are not supported.

Add one or more components as required (for Local Path | S3 | HTTP files) and configure them with your server credentials.

Compose Action

The *Compose action* converts media descriptor files into a Post Producer CML file and also generates Tempo CML files. To use the action, you specify an input file and output file by nickname, specify the output storage location, and select a composer to perform conversion, based on the input file type. This is the Compose action inspector:



Converting Files to Post Producer CML

Media descriptor file formats that you can ingest and convert to Post Producer compositions (CML files) for utilization in Post Producer workflows include:

- *CML*—modifies CML files submitted to it for various purposes.
- *EDL*—converts specifically-formatted EDL files from a Harris Velocity system.
- *Final Cut Pro 7 XML*—converts Final Cut Pro 7 XMEML files from any system that generates or exports this file format.
- *GraphicsFactory*—converts GraphicsFactory template files for processing by Conform workflows instead of Flip workflows.
- *IMF*—converts an IMF file to a Composition for processing by a Conform workflow.
- *QuickTime Reference*—converts QuickTime reference files, typically created by exporting a sequence from an Avid system.
- *Simple AAF*—converts Simple AAF files, typically created by exporting a sequence from an Avid system.
- *Telestream TSEDL files*—converts Telestream TSEDL files.
- *Text File List*—converts text files to a Composition for processing by a Conform workflow.
- *Trigger*—converts trigger files; enables you to add triggers (similar to the EDL Composer) in which secondary events can trigger events based on ancillary timecode, including pre-roll and post-roll clips, letterbox settings, closed captions and subtitles, and center cuts.

Once a file has been converted to a CML file (composition), it can optionally be further edited to add features before processing it. The resulting Composition can be submitted to the Conform action to compose media. You could create one workflow with both a Compose and a Conform action, or you could create two workflows—one to convert the file (Compose), and the second (with a Conform action) to conform it.

Generating Tempo CML Files

Unlike the other options which define the ingested file for conversion, the Tempo Composer generates a Tempo CML file, optionally using an Analysis Result file from an upstream Analysis action, for ingest and processing by a Tempo action or Tempo-based workflow.

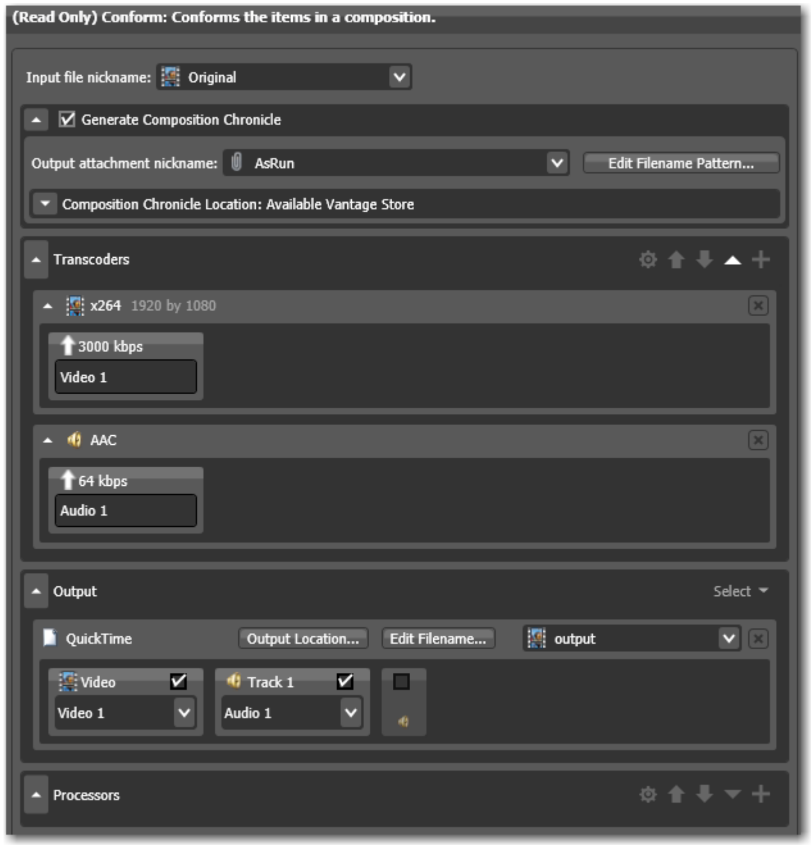
For details on converting media descriptor files to CML, see *Converting NLE Projects and Media Descriptor Files to Compositions*.

Conform Action

The *Conform action* is the primary Post Producer action. It assembles (*conforms*) and encodes source video, audio, titles, and still image (raster or bitmap) files, based on specifications contained in the Post Producer CML file that started the job, and then transcodes (or direct-converts) into an output file, based on the container and codec configuration you have specified in the Conform action's inspector.

Note: Workflow Portal also supports EDL processing (using the Flip action), with similar functionality. However, the primary difference—and advantage—of using a CML workflow is that you can reference clips with mixed essences. The EDL configuration requires that all clips have the same frame rate, frame size and video codec; CML does not have this restriction. Composition processing also improves VANC and caption processing.

The process of creating media from a set of input media files using CML-based instructions is referred to as *conforming*. The input files and instructions for conforming the output file are all defined in the Composition XML file. Here is the Conform action inspector:


(Read Only) Conform: Conforms the items in a composition.

Input file nickname: Original

☑ Generate Composition Chronicle

Output attachment nickname: AsRun

Composition Chronicle Location: Available Vantage Store

Transcoders

x264 1920 by 1080 — 3000 kbps — Video 1

AAC — 64 kbps — Audio 1

Output

QuickTime — output

Video ☑ Video 1

Track 1 ☑ Audio 1

Processors

The Conform action can only process source video and audio in certain formats. If your input video and audio is not in a supported format, you can transcode it into one of the supported formats prior to processing it in the Conform action. You can create a separate workflow to perform this task, or flip it right in the same workflow.

Note: For a list of supported formats, see *Input Media Formats*, and *Graphic File Input Formats*. For supported output formats, see *Output Video Formats*.

The Conform action, which utilizes the Telestream Media Framework transcoder, with more features, greater transcoding flexibility and efficiency, can encode media in various essences, into these formats: MP4, MXF OP1a, QuickTime MOV, and TIFO.

The Conform action can also apply filters as specified by you, along with the media and container to generate during transcoding. If you require output media in formats other than those directly created by the Conform action, you can pass the media on to Flip64, Multiscreen Flip, or IPTV Flip encoding actions in this or another workflow for processing.

You can configure the Conform action with both a video and audio transcoder to produce both streams in your output or you can produce video-only output. However, you can not use the Conform action to produce audio-only output.

Note: You can optionally add Video Processing Library (VPL) templates that can be applied to specific media files. VPL templates can be configured to process input media with specific decoding requirements, such as frame rate conversion, Telecine or inverse Telecine, or complex audio mapping. Templates are created by adding a Processor, configuring, and naming it. The template can then be referenced in the source configuration section of a composition, and used to perform decoding as specified when the composition is processed in a workflow with a Conform action.

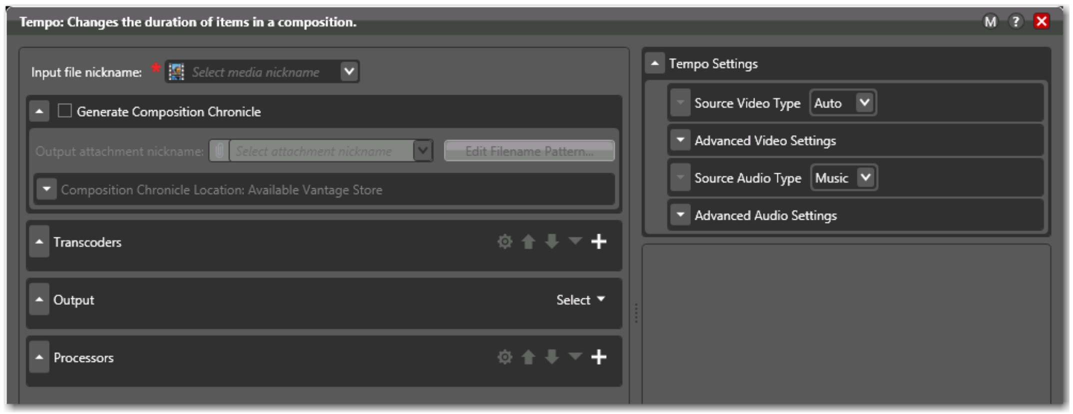
Tempo Action

The Tempo action is a special-purpose action which re-times file-based content and adjusts the running time of shows and segments while maintaining high program quality. The Tempo action accomplishes this by interpolating the desired time change over the duration of the content—adjusting total elapsed time without any perceived loss of video or audio quality, including captions.

The Tempo action produces newly-timed media from the source media file specified in Tempo CML, applying option filters as appropriate, and encoding the output video in the specified format (as defined in the ingested CML file).

Note: Supported audio and video input formats and output formats that Tempo can generate can be viewed on the Tempo data sheet, available on the Telestream web site. The video may be ready for distribution or you can process it with a Flip, Multiscreen, or IPTV VOD action for further transcoding and packaging prior to distribution.

This is the Tempo action inspector:



Timecode propagation is supported in TIFO and QuickTime containers, but not in Program Stream or MP4 containers. Tempo supports closed captioning, content advisory, Active Format Description, and copy generation management (CGMS-A).

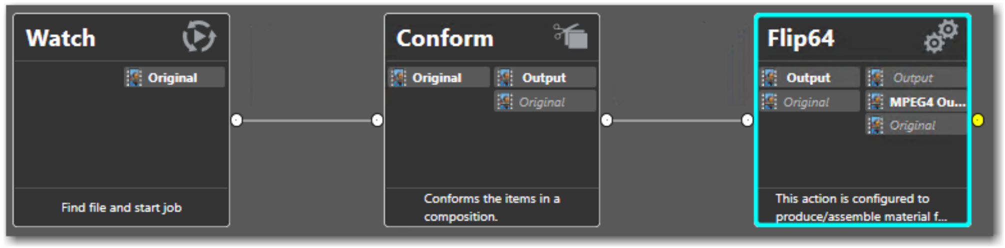
Note: For full details on setting up and using Tempo, see these guides, which are available in the Vantage section of the Telestream web site: The Tempo User Guide provides an overview of what Tempo is, how to create Tempo workflows and compositions, and how to use the Tempo Portal to produce re-timed media assets. The Tempo Developer's Guide describes the requirements for producing Tempo CML compositions outside of Tempo Portal.

Typical Post Producer Conforming Workflow

The type of workflow you'll typically use repeatedly to produce your promos with Post Producer is a conforming workflow.

Conforming workflows can be very simple: the basic workflow involves a Watch, and a Conform—to composite the media as described in the CML, and encode it to one of the supported formats. (For a complete list, see *Supported Post Producer Formats*.)

Here is a basic Post Producer conforming workflow—this example adds a Flip64 action after the Conform, to also create a proxy for review and approval.



Of course, if the format of the media from the Conform action is not suitable for your production requirements, you can add another Flip64, or a Multiscreen Flip or IPTV Flip action for further encoding to the workflow. In short—a Post Producer conforming workflow is no different than any other workflow in terms of flexibility or application—except that it uses a Conform action to generate (conform) the media—which you can use as the starting point for other processing, as required for your video production.

Note: For a comprehensive review of typical Post Producer workflows for a variety of applications, see the separate Post Producer Cookbook.

Submitting Compositions to Workflows

You can submit a CML file to a Post Producer conforming workflow (a workflow with a Conform action) just like you submit media files to other workflows. Here are the ways that you can submit CML files for processing:

- Dropping the file into a folder monitored by a Watch action-based workflow.
- Dropping a workorder file (which contains jobs to submit) into a folder monitored by a Workorder action-based workflow. See *Batch Processing Jobs with Workorders*.
- Manually submitting the file to a Receive action-based workflow in Workflow Designer or other Vantage client program.
- Programmatically generating and/or submitting the CML to a workflow using the Vantage SDK. See *Submitting Compositions Using the SDK*.
- Using Workflow Portal with a CML configuration to select media, automatically create a CML-based playlist and submit it. See *Assembling and Processing Playlists with Workflow Portal*.

Batch Processing Jobs with Workorders

You can submit multiple jobs at one time, using a workorder. Workorders can only be processed by workflows that start with a Workorder action.

The purpose of the Workorder action is to enable automated submission of multiple jobs in batches. Workorder files are a simpler, automated submission alternative to job submission via the Vantage SDK.

A Workorder file is a comma-separated text file containing one or more work orders—job descriptions. Each Post Producer job is comprised of a fully-qualified reference to input files (CML files, media files, NLE project files and media descriptor files, etc.), optional attachments, and variables—all the input needed for the workflow performing the processing. At least one workorder scheme must be created (either directly in the Workorder action or in the Vantage Management Console) in order to parse the workorder schema and submit the jobs. Workorder files must conform to the schema specified in the selected workorder scheme.

You can create workorder files manually (in an Excel spreadsheet, or a text editor of some kind, for example). However, ideally a program or system component will dynamically and automatically create workorder files and deliver them to the target share for submission and processing.

For details on creating workorder schemes, refer to the Domain Management Guide.

Submitting Compositions Using the SDK

You can submit compositions to a workflow for processing using methods in the SDK. For any SDK-based submission, the workflow must start with a Receive action.

Methods you can use include *SubmitFile*, where the CML is an XML file, located on a system that is accessible by the Vantage service that reads the file. You can also submit a composition using *SubmitItems*, where the composition is an object held in memory.

Note: When using *SubmitItems*, the maximum URL data size is 64K. If you exceed this size, the method fails. To keep your URL under the size limit, consider eliminating default values and comments, if any. Alternatively, write the composition to a file, and use *SubmitFile* (or other file-based submit method) to submit the composition to the workflow.

For information on using the Vantage SDK, see *Developer Resources*.

Assembling and Processing Playlists with Workflow Portal

In addition to submitting compositions you've created directly to a Flip64 (or other) action workflow, you can also use Workflow Portal to create a playlist formatted as a CML file and submit it to a Post Producer conforming workflow (a workflow with a Conform action).

In this situation, the Conform action (instead of the Flip64 action) processes the compositions you submit, and renders a single media file as output.

Note: Create CML (Composition Markup Language) configurations can only be used in the Windows client.

Workflow Portal also supports EDL processing (using the Flip action), with similar functionality. However, the primary difference—and advantage—of using a CML workflow is that you can reference clips with mixed essences. The EDL configuration requires that all clips have the same frame rate, frame size and video codec; CML does not have this restriction. Composition processing also improves VANC and caption processing.

Workflow Portal configurations are created in the Management Console. For details, see the *Configuring Vantage Workflow Portal Operation* chapter in the *Domain Management Guide*. For details on using Workflow Portal to process CML-based playlists, see the *Vantage User Guide, Using Workflow Portal* chapter.

Creating DPP Packages with Workflow Portal

You can use Vantage to create a media processing system which produces Digital Production Partnership (DPP), Version 4.1-compliant television programme packages, ready for broadcast or distribution.

DPP package production in Vantage provides the following features:

- *DPP Workflow Portal*—enables operators to create DPP package processing jobs, marking parts in a proxy to generate a DPP timeline including slate, bars, and black as a CML file, enter DPP metadata, and submit the job to DPP package publishing workflows. (Workflow Portal has also been implemented as a Web application, but it does not support DPP configurations.)
- *Post Producer*—renders the CML with the programme's video and metadata to create video with a DPP-compliant timeline, including slate, bars, and tone.
- *Transcode Pro*—encodes DPP programme files, automatically inserting DPP metadata, creating a DPP package in AS-11 MXF format with appropriate audio and video encoding.
- *Vantage Analysis tools*—validates the metadata in the DPP metadata XML files, ensuring valid formatting and compliance.
- *Delivery Connectors*—enables automatic delivery or deployment to archiving, delivery, and publishing systems, including Aspera and Signiant delivery protocol options (HTTP, S3, and FTP are standard capabilities).

Note: For full details on setting up and using DPP processing applications in Vantage, please refer to the *Workflow Portal Guide*, and the *Creating DPP Packages in Vantage* app note, which is published on the Telestream Web site in the Vantage Options section. Also refer to Creating DPP Configurations in the Using Workflow Portal chapter of the *Vantage User Guide*.

Composition Overview

The purpose of this chapter is to introduce you to the concept of a Post Producer composition. This chapter describes (conceptually) how to create and use compositions, and the major components of a composition, and their relationship with each other. Of course, at this introductory stage some details are left out, and there is little mention of Composition Markup Language (CML)—the XML schema used to implement a composition. The Post Producer CML language is covered in detail in succeeding chapters.

Note: It is important to note that CML is frame-accurate, not field-accurate. Thus, timecodes that you use to specify a temporal property must not contain an asterisk (denoting 2nd field position)—they will be ignored, and your output media probably will not be what you expected.

To understand how to describe the behavior and the look and feel of your intended media in a composition, it is important to understand the building blocks in CML. Creating templates with an NLE does not require a deep understanding of compositions, but if you plan to write a program to auto-generate compositions or manually create a composition in a text editor, this conceptual understanding is vital to your success.

Topics

Creating a Composition

Compositions (the design template) must be written in a specific schema—Composition Markup Language (*CML* for short). CML is designed to make describing media (and thus providing the composition instructions) as easy, flexible, and comprehensive as possible. Compositions are written as XML files.

Because Post Producer is intended for repetitively producing media files with the same structural components from different sets of input media (hourly weather updates, for example), the composition must specific enough to comprehensively describe the output media characteristics, but general enough to generate the media from any given set of input media provided.

When designing the composition, you won't typically know in advance the file names of the input media resources. All you'll know is that you need a baseline video, and you want a logo, plus lower-thirds text, for example. So you have to leave these elements undefined—or, just identify a sample set of test input media for development.

You typically create CML files using a computer program that you have developed, or you can create them manually in an XML or text editor.

Note: Before you can create CML files (or write programs that create CML files), you must have a solid understanding of its elements and topology. For details, see *Developing Compositions*.

You can create CML files in the following ways:

- Develop a program using the Post Producer Class Library in the Vantage SDK
- Using an XML or text editor
- In Vantage, use a Compose action workflow to convert a media descriptor file in another format to CML format.

Each of these methods are described in detail later in the guide.

Note: While creating Composition files manually is feasible, it is not usually the preferred practice. The best practice is to generate compositions programmatically.

Specifying the Input Media in the Composition

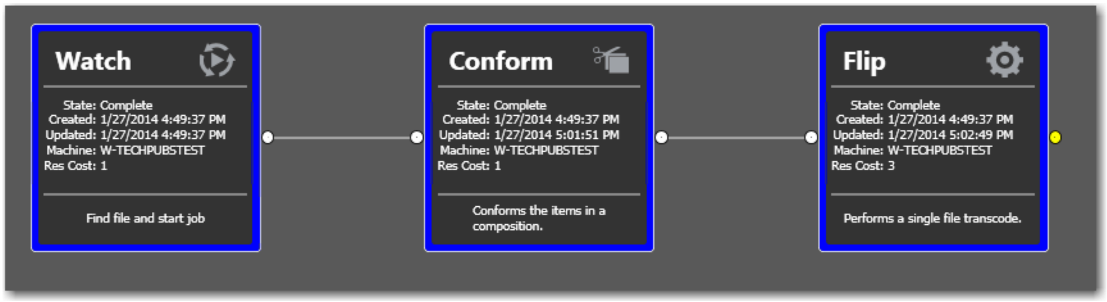
Each time you submit a job (for testing or for production), you modify the composition file to identify the path to each input media file you require—creating a ready-to-run composition—and then you can submit it to a Vantage workflow to process the job and generate the output.

Before you submit a composition file to generate a new output file, update it with the correct input files. While you can update it manually, you can also automate this process in various ways in production. For example, you could use a nickname that is replaced with a run-time path and file name. Or, you could implement your composition generator application to supply the correct file name just prior to submission.

Alternatively, you could design a composition with generic or functional filenames: *baseline_video.mov*, *network_logo.png*, *backtimed_credits.mov*, etc. Then, you would design your system to rename the input files for each job to these generic names.

Generating Media from a Composition

In Vantage you can create media from a composition by processing it with a Post Producer workflow. You first create a Post Producer workflow—one which contains a Conform action (or other Post Producer transcoding action, such as Tempo). This workflow assembles and encodes the input media files—the video, audio, text, and still images you've identified in the ready-to-run composition—into an output file.



Watch
State: Complete
Created: 1/27/2014 4:49:37 PM
Updated: 1/27/2014 4:49:37 PM
Machine: W-TECHPUBSTEST
Res Cost: 1
Find file and start job

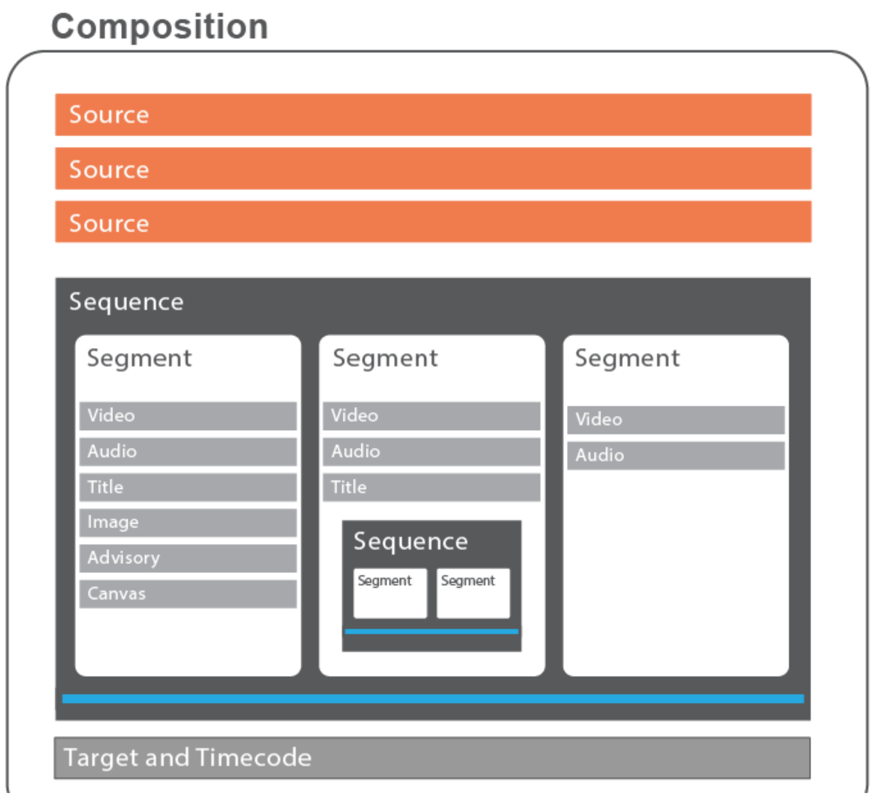
Conform
State: Complete
Created: 1/27/2014 4:49:37 PM
Updated: 1/27/2014 5:01:51 PM
Machine: W-TECHPUBSTEST
Res Cost: 1
Conforms the items in a composition.

Flip
State: Complete
Created: 1/27/2014 4:49:37 PM
Updated: 1/27/2014 5:02:49 PM
Machine: W-TECHPUBSTEST
Res Cost: 3
Performs a single file transcode.

To generate your media, you submit your ready-to-run composition file to the workflow. Details are described in the next section. When the job is done, Post Producer has automatically created your promo—based on the composition's design template—from the source media you identified.

Composition Object Hierarchy

As illustrated below, a composition is a collection of multi-level objects, working together to create a hierarchical, temporal, and spatial arrangement of media.



Referring to the illustration above, note that each composition as a minimum, has these components:

Sources—These are simply pointers to input media files—nothing more than fully-qualified paths to the media files.

Sequences—An independent timeline; a collection of one or more segments, played out sequentially on the timeline.

Segments—A layered collection of material (and optionally, other sequences) that is played simultaneously.

Material—The media—video, audio, titles, content advisories, canvases, and images—each with unique properties and effects, and spatial and temporal arrangements.

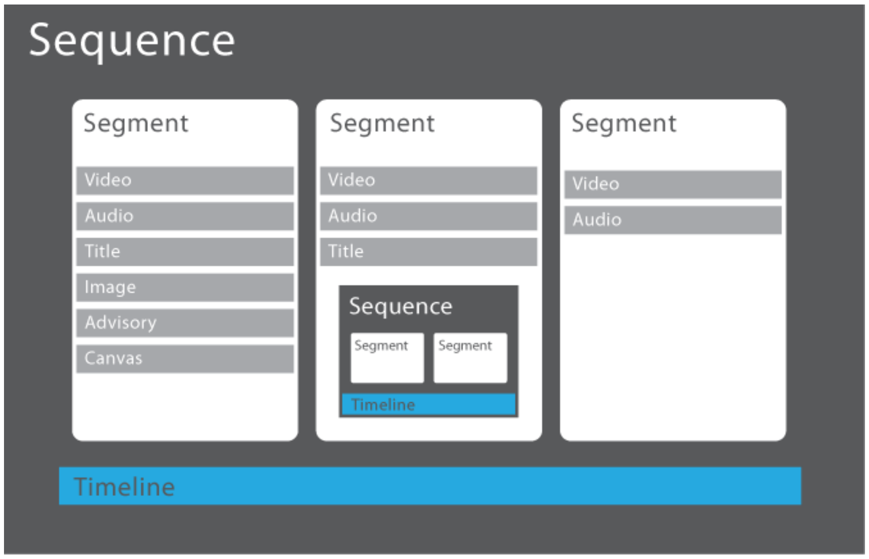
Target—optionally, timecode properties.

A composition also contains media sources, which identify the media files of the media that are utilized in the composition.

Sequences, segments, and material (and their effects) are described in detail in the following topics.

Sequences

A sequence defines a set of segments, which play sequentially, one after the other, to completion. At least one sequence is required, in order to hold segments.The total length of a sequence is the sum of the length of its segments. Sequences can be placed in the composition, or inside segments.



The temporal position (start time) of each segment in the sequence is determined by the accumulated duration of the preceding segments in that sequence.

All sequences in a composition begin playing concurrently. Likewise, sequences in a segment begin at the same moment—when that segment begins.

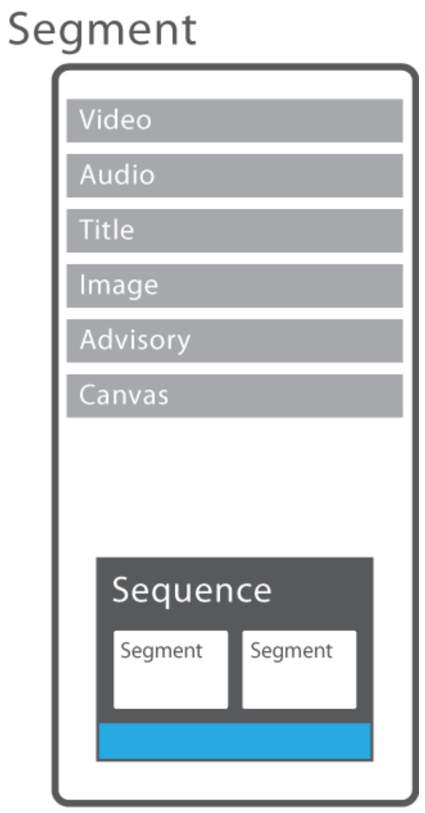
You change the moment when a sequence plays by offsetting it with a time value.You use the layer property of a sequence to control visual layering (compositing): which sequence can obscure the visual material in other sequences. Thus, visual material in a sequence with a lower layer value may be obscured by visual material in sequences with a higher layer value.

The material in a given sequence is played back independently of material in all other sequences. Thus, you should add an additional sequence when you want an additional, independent timeline upon which to present material.

The duration of a sequence is equal to the accumulated duration of all the segments it contains. The longest sequence in the composition determines the duration of the composition.

Segments

Segments define a set of material that is played back concurrently. At least one segment is required in each sequence. Multiple segments play consecutively. Segments are dynamic, bounded time periods—the ordinal (ordered) building blocks of a sequence. Each segment has a duration—and the duration of each segment in the sequence determines the total duration of the sequence.



Segments create time boundaries on the enclosing sequence's timeline; the material within a segment can not cross the segment's boundary. You add additional segments to a sequence when you want to increase the timeline of the sequence.

The start time of a given segment is determined by the accumulated duration of the preceding segments. Like a sequence, you can change the moment when material begins playing by offsetting the item with a time value.

Each segment also includes a reference to the source media file, and may include mark-in and mark-out -point to specify the portion of the media to play back.

Material

Material items are the building blocks of your media. You can provide the following types of material in segments:

- *Video*—instances of the video or video clips referenced by sources
- *Audio*—instances of audio in a file, ignoring video (if present)
- *Images*—instances of raster images, or image sequences
- *Content Advisories*—instances of advisory graphics and metadata
- *Titles*—instances of text, burned in on the frame
- *Canvases*—instances of black rectangles, used for fades

While all material has common characteristics, each type also has a unique set of properties. Just like sequences, all material items in a segment play concurrently—at a specific temporal position in the timeline of the sequence—beginning at the segment time. To alter the moment when a specific instance of material is presented, you provide a time-based offset.

Also like sequences, all material items within a segment are layered from back to front visually. Thus, visual items in a lower layer within the segment may be obscured by visual items in a higher layer.

Visual and Auditory Effects

Visual effects can be applied to material. You can also apply these effects over time, creating animations and audio fades. Effects can also be combined. (Volume adjustment is also an effect, which can be applied to audio material.)

You can apply the following effects to material:

- *Opacity*—controls the transparency of visual material, by layer
- *Scaling*—controls the size of the visual material in the frame
- *Translation*—controls the location of the visual material in the frame
- *Rotation*—controls the Z axis of the visual material
- *Volume*—controls the loudness of audio material.

Using these effects effectively requires a solid understanding of each effect, as well as how to identify various temporal portions of the media.

Note: The application of material and application of effects in your composition is really where the rubber hits the road. The ability to implement the material in the manner you desire is the key to creating functional compositions. For a comprehensive, detailed discussion of working with material, see *Working with Material*.

Developing Compositions

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

Note: Post Producer Composition Markup Language is a dialect of CML. Though the CML dialects are similar, Post Producer CML is designed specifically for Conform 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

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

Aside from using an NLE to generate CML, there are two different approaches to creating compositions: write a program that generates compositions or manually create them in a text editor. Telestream recommends generally, that you use the Playlist .Net Class Library and develop your compositions in supported .Net languages. 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 or learn how to use the Playlist Class Library.

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
- Telestream's Playlist Class Library
- 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.

Post Producer CML Developer's Guide—The guide you're reading now.

Composition.xsd—The specification for a valid composition. You can use this file to validate your compositions before submitting them. It is helpful when you are testing a composition generator program during development, and especially when you are hand-coding a composition. You can also review the XSD in a text editor to determine the exact rules for each element and attribute.

Note: 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. Integers and decimals are permitted. 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. 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="1">
    <File location="\\share\path\my_promo.mov" />
  </Source>
  <Sequence layer="1">
    <Segment>
      <Video align="head" adjust="edge" fill="none" source="1"
layer="0" />
    </Segment>
  </Sequence>
</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. One or more Source elements—you can optionally add one or more as necessary. 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 in the Conform action. This is the foundation you'll build on to create complex, high-quality highlights, promos, and other types of media files.

We suggest you copy and paste this into a file, save it as XML, and change the location attribute on the File element to point to a suitable file that your Vantage server has access to. Then, submit the XML file to a Conform workflow and view the output.

Validating a Composition

To validate a composition, you can use a variety of tools (Notepad++, Microsoft Visual Studio, Oxygen, etc.) You can also validate your composition using tools provided by the Worldwide Web Consortium at: http://www.w3schools.com/xml/xml_validator.asp.

The validation file to use is named *Composition.xsd*, located in the Vantage SDK.

Note: There may be XML errors in your properly-formed CML file that are permitted. See *Resolving Errors in CML Processing Jobs* for more details.

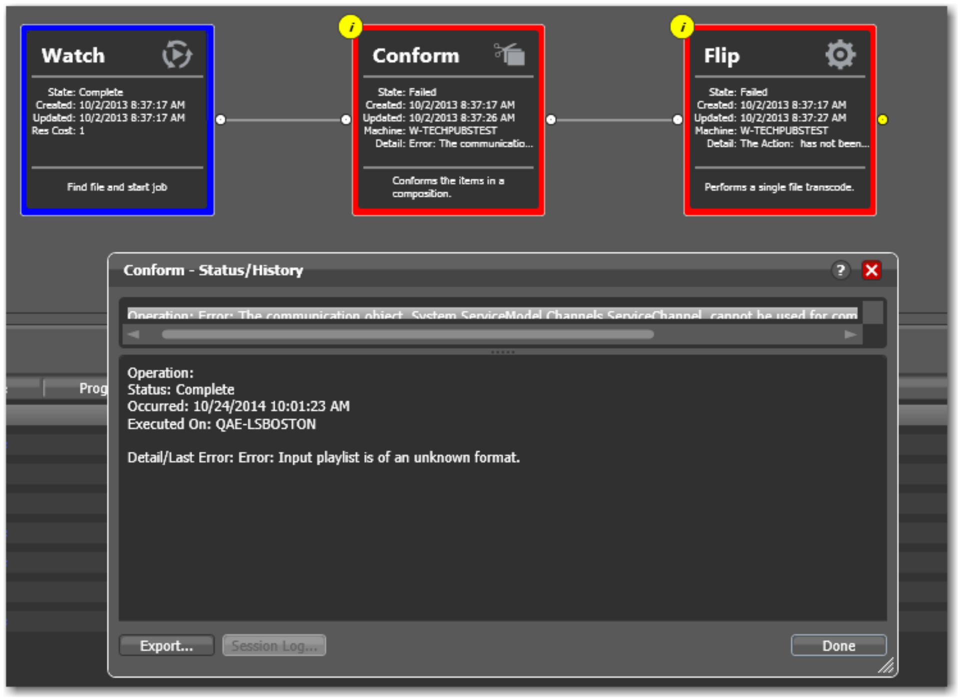
It is a good habit to validate your composition frequently as you develop it—regardless of how you create it. If you submit a composition to a workflow with XML or CML errors, it will fail.

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 Post Producer (see *Supported Post Producer Formats*).
- *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:



```
State: Complete
Created: 10/2/2013 8:37:17 AM
Updated: 10/2/2013 8:37:17 AM
Res Cost: 1

Find file and start job
```

```
State: Failed
Created: 10/2/2013 8:37:17 AM
Updated: 10/2/2013 8:37:26 AM
Machine: W-TECHPUBSTEST
Detail: Error: The communicatio...

Conforms the items in a composition.
```

```
State: Failed
Created: 10/2/2013 8:37:17 AM
Updated: 10/2/2013 8:37:27 AM
Machine: W-TECHPUBSTEST
Detail: The Action: has not been...

Performs a single file transcode.
```

```
Operation: Error: The communication object, System.ServiceModel.Channels.ServiceChannel, cannot be used for com

Operation:
Status: Complete
Occurred: 10/24/2014 10:01:23 AM
Executed On: QAE-LSBOSTON

Detail/Last Error: Error: Input playlist is of an unknown format.
```

To resolve the issue, you should validate your composition (see *Validating a Composition*). 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.

Using Nicknames, Expressions, Variables, and Constants in Compositions

In any attribute value in CML (text surrounded by double quotes), you can utilize nicknames, expressions, variables, and constants, which you place in braces { }.

The use of nicknames, expressions, and variables allows you to design general-purpose compositions, and dynamically supply appropriate values at run-time—on a job-by-job basis. This makes them more broadly applicable and thus, more practical—you'll usually end up creating fewer, more flexible compositions.

For example, you could create a filename as: {$$filename}.{$#Original}.{37*24}.mov, where the first expression is a variable, the second is a media item identified by its nickname, and the third is an expression that will be evaluated.

Evaluation is performed by the Conform action during execution to resolve nicknames, expressions, variables, and constants to the final string value (literal) before the composition is processed. All values are ultimately presented as strings, of course.

Type	**Format**
Nickname	{$#Nickname} (preferred) | {#Nickname} for compatibility
Expression	{expression}—see *Expression Examples*
Variable	{$$VariableName}
Constant	{%%Constant}

- Nicknames
- Nickname Examples
- Expressions
- Expression Examples
- Variables
- Variable Examples
- Constants

Nicknames

Vantage actions usually reference files with a descriptive string—a *nickname*—to make it easier to design and implement workflows. You can think of a nickname as a logical reference; a variable for file names, which can be different for each job. For example, you can use the string *Original* as the nickname for the file ingested by a Watch action, or *Vantage Proxy* for a QuickTime, low-res file produced by a Flip action. Vantage supplies several; you can also add your own and use them anytime.

A nickname in a composition takes the form *{$#Nickname}*.

Note: To maintain backward compatibility, a single # is permitted unless its value has space characters in it, or any characters (-) that could be interpreted as operators.

A nickname can also be used in place of the actual file path and name in a composition. During processing, the runtime value of the nickname is substituted. In order to use nicknames, an upstream action must have either ingested or produced the file, and assigned it the associated nickname.

The use of nicknames creates more flexibility in the design of compositions. For example, you could have one composition that works in three different workflows, each processing video for a different output size, or a different language. In each workflow, you could use an Associate action to identify the correct file for the workflow.

Nickname Examples

In the example code snippet below, the File element references a fully-qualified path and file by nickname: `{$#Promo}`.

```
<Composition>
  <Source identifier="1">
    <File location="{$#Promo}" />
  </Source>
  <Sequence layer="1">
    <Segment>
      <Video align="head" adjust="edge" fill="none" source="1" 
layer="0" />
    </Segment>
  </Sequence>
</Composition>
```

In this example, the composition is configured to process a file nicknamed *Promo*. The CML file is stored permanently in a workflow's watch folder, and as you drop media in, a job is submitted for processing.

A Watch (or other origin) action ingests media to be processed by the composition shown above, and assigns it the nickname *Promo*. An Associate action ingests the composition for this job. The Associate action's file match pattern should be configured to pick up the specific CML you want to be processed.

Note: You could also create a media file upstream, and pass it in as *Promo*, instead of using the Watch action to ingest a pre-existing media file.

Expressions

You can use arithmetic expressions to calculate the value of an attribute or element string value. Expressions can—and often do—contain variables. (Using expressions alone—without variables—may not be very practical—though it does accurately present the expression's logic.)

Expressions are always surrounded by braces {}.

The use of expressions enables you to create more flexible and powerful compositions. For example, you can use expressions to dynamically calculate curtains or pillars, or frame sizes at run time.

Supported operations are:

- Addition (+)
- Subtraction (-)
- Division (/)
- Multiplication (*).

Expressions can include parentheses to support precedence.

Expressions that use only integers return an integer presented as a string. Expressions that involve real numbers return a real number presented as a string, even if the decimal portion is zero.

Expression Examples

Here are some expression examples shown in the context of XML tags:

`<... "{4 * (1+3)}" />` is evaluated as `<..."16" />`

`<... "{4 * 4.0}" />` is evaluated as `<..."16.0" />`

`<... "{4 / 3}" />` is evaluated as `<..."1.33333333" />`

`<Translation x="{40+40}%" />` is evaluated as `<Translation x="80%" />`.

You can also populate variables in upstream actions and pass them downstream. For example, an Analysis action can obtain timecode information to set the tail of a clip:

```
<Tail>
  <Edit mode="duration" time="{$TC_duration - $TC_offset}"/>
</Tail>
```

Variables

You can also use variables in attributes and elements, to provide a run-time value during processing, rather than at design time. The variables may be of any type, as appropriate: integers, strings, timecodes, etc. You can also use variables in expressions, to supply run-time values.

Variables in a composition are strings preceded by the reserved characters $$, and surrounded by braces {}. For example, *{$$FrameWidth}*.

Note: To maintain backward compatibility, a single $ character is also permitted unless the variable's value has space characters in it, or any characters that could be handled as operators. For example: a string such as *16/9* that is intended to be a literal string value and not the numeric representation of *1.77778*.

If the value of the variable will be computed or derived from analysis, the value must be specified in a previous action in the workflow. If a static value will be used for the workflow, then the variable simply needs to be declared with a default value at design time, either in the ConformTempoFlip64 action (right-click on the action and select Add Variable) or by declaring the variable and the default value directly in the Vantage Management Console.

Variables can also be used in supported non-linear editors by renaming clips on the timeline with a variable that will ultimately be used to replace those media elements with alternative content.

Variable Examples

Here are some examples of using various types of variables in compositions.

Numeric Variables

In this example, you are supplying an SD source file which contains NTSC IMX video (720 pixels by 512 lines), and you want to display it in HD aspect ratio. You use the Crop element to remove vertical blanking, and the Mask element to convert the source to the 16:9 aspect ratio by adding a 2:9 (0.222) curtain to each side of the image.

```
<Composition version="1.0"
xmlns="Telestream.Soa.Facility.Playlist">
  <Source identifier="1">
    <File location="\\share\path\Source_video.mxf" />
    <Crop top="{$$VBI_Top}px" left="0" bottom="-{$$VBI_Bottom}px"
right="100%" />
    <Mask top="0" left="{$$2_9_Curtain}" bottom="100%" right="-
{$$2_9_Curtain}" />
  </Source>
  <Sequence layer="1">
    <Segment>
      <Video align="head" adjust="edge" fill="none" source="1"
layer="0"/>
    </Segment>
  </Sequence>
</Composition>
```

In an action upstream of the Conform action in your workflow, you'll add three variables: *VBI_Top* and *VBI_Bottom*, and *2_9_Curtain*, and then assign the values: 30, 2, and 0.222222, respectively. By using variables, these values can be determined at runtime—not design time—on a job-by-job basis.

Channel Numbers as Variables

In this example, the audio channel numbers in the Mix elements (in bold, below) are identified only by variables, whose values are set at runtime in the workflow. The audio channels are specified with hard-coded target channel maps and mix levels.

```
<Composition version="1.0"
xmlns="Telestream.Soa.Facility.Playlist">
  <Source identifier="1">
    <File location="{$#Original}" />
    <Mix source="{$$AudioChannelLeft}" target="1" level=".9"/>
    <Mix source="{$$AudioChannelRight}" target="2" level=".9"/>
  </Source>
  <Sequence layer="0">
    <Segment>
      <Video align="head" adjust="edge" fill="none" source="1"
layer="0" />
    </Segment>
  </Sequence>
</Composition>
```

String Variables

You can also provide text string variables, and optionally, concatenate strings. For example:

```
<Title align="head" adjust="edge"... >Watch {$$Show_Name} tonight
at {$$Show_Time}</Title>
```

In your workflow, in an appropriate action upstream of the ConformTempo action, you'll add two variables: *Show_Name* and *Show_Time*, and assign the string values "What's Up, Doc?" and "7:00 PM Eastern", respectively.

Note: If the action does not explicitly provide the variable you require, you can always right-click the action and select Add Variables to provide any variable you want.

Constants

Constants are reserved strings pre-pended by the characters %%, and surrounded by braces: {}.

LOCAL_MEDIA_FILES is a constant which identifies a Vantage-supplied set of PNG files stored in a specific location during Vantage domain installation. You cannot change the value of this constant.

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.

Generating CML As Run Files

As Run CML files can be generated by Conform actions, when configured to do so.

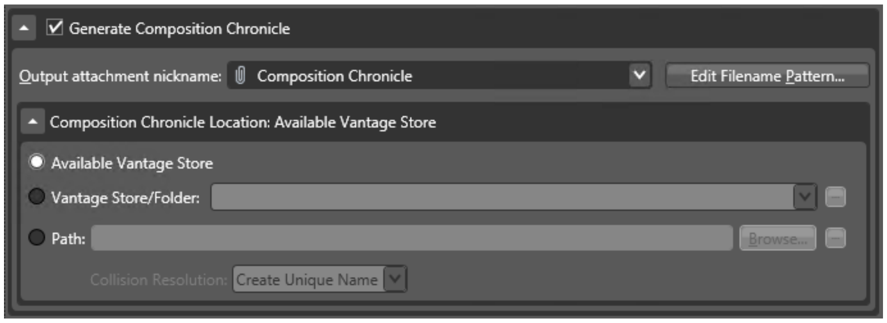
An As Run CML file is a replica of the input CML file, with many attributes updated to indicate how options were processed (for example, images) and actual values determined at runtime.

One application of As Run files is to review the file to determine how elements were actually processed—images, for example.

Another application is to use a Conform action to generate an MPEG4 file from MPEG video with AAC audio and create an As Run CML file (with actual values in timecode attributes, for example), for use in Black Arrow SCTE-35 ad insertion systems.

Another application is to generate a CML file that you can then modify manually.

To produce a Composition Chronicle CML file, check the Generate Composition Chronicle option in the inspector, specify an Output Attachment Nickname for the file, and specify the Vantage Store or output path where the file will be placed when generated.


☑ Generate Composition Chronicle

Output attachment nickname: Composition Chronicle

Composition Chronicle Location: Available Vantage Store

● Available Vantage Store

○ Vantage Store/Folder:

○ Path:

Collision Resolution: Create Unique Name

Converting NLE Projects and Media Descriptor Files to Compositions

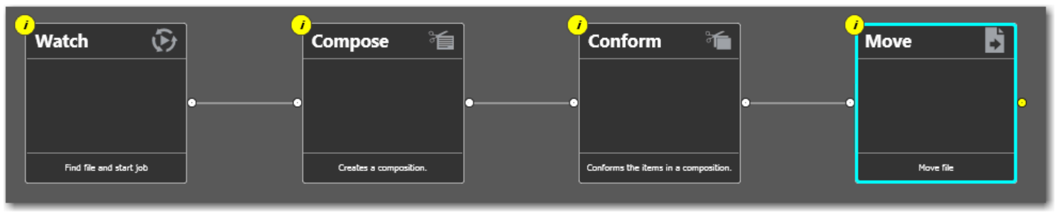
This chapter describes how to export projects in supported non-linear editors and convert them into CML templates for use in Post Producer workflows. In this chapter, you'll also learn about converting other media descriptor files into CML templates.

Topics

Converting NLE Projects into Compositions

One of the easiest methods for creating Post Producer templates is by integrating variables into NLE timeline elements in your project. You can export the project as XML and process it in a Post Producer workflow utilizing a Compose action to convert the project to a composition. The composition can then be processed by a Conform action (in the same or other workflow) to generate your media. Using this method requires only expertise in the NLE, and requires little or no understanding of XML or Composition Markup Language.

The typical Vantage NLE project conversion workflow starts with a Watch or Receive action to ingest the project/media descriptor file and start the job, as shown below.



Use a Watch action when you want to submit media manually or by adding the file to a target folder. Use a Receive action when you want to submit the file programmatically, using the Vantage SDK.

The Compose action (configured with a specific *composer* for the type of file you're converting—for example, Final Cut Pro or Simple AAF) parses the file and converts it a Post Producer composition (CML) file. It passes the CML file to a Conform action for media encoding, and then to optional file-operations actions for delivery of the output media file to its intended destination. (You could also implement the Conform action and other tasks in a second, separate workflow.)

Post Producer can convert basic transitions such as cuts and fades from an NLE. However, dissolves, page turns, and similar kinds of transitions need to be rendered out to a media file, and the file added to the CML for stitching in the Conform action workflow.

NLE Feature Support

The following table provides information about which editors and versions are supported, plus details about supported features in each editor.

Editor & Version	Export Format	Variable Support	Transition Support	Max Layers	Mixed Format Content Support
Adobe Premiere Pro CC for Mac OS	FCP7-XML (XMEML)	Yes, via Rename Clip	Cuts, Fades, Cross Fades	10	All formats supported by Post Producer
Adobe Premiere CC for Windows	FCP7-XML (XMEML)	Yes, via Rename Clip	Cuts, Fades, Cross Fades	10	All formats supported by Post Producer
Apple Final Cut Pro 7	FCP7-XML (XMEML)	Yes, via Rename Clip	Cuts, Fades, Cross Fades	10	All formats supported by Post Producer
Avid Media Composer 7 and newer for Mac OS	Simple AAF	No	Cuts	1	No
Avid Media Composer 7 and newer for Windows	Simple AAF	No	Cuts	1	No

Post Producer Titler Engine is an optional replacement for the built-in Premiere title tool.

A practical example of using an NLE involves creating a template for show promo spots. The clips in the CML can be tagged with tokens for variable substitution in Post Producer.

For, example, you can add a *DayofWeek* variable to create specific spots, each for play at different times. Each spot uses a preset bed created in an NLE with graphic overlays and audio overlays for a show announced as playing on "Tuesday" or "Tomorrow", or "Tonight". The workflow populates the variables pulled from the sequence XML, such that three files are created from one job: "Tuesday" (with a graphic overlay and audio dub for Tuesday,) Tomorrow (with matching graphic overlay and audio) and Tonight (also with matching graphic overlay and audio.)

The Final Cut Pro 7 XML Interchange Format was designed by Apple to describe every element in an NLE project in a human-readable, XML-based format. Final Cut Pro and Adobe Premiere Pro CC can export this format to Post Producer (through a watch folder or direct import) which converts this file to a CML file which includes the elements that are required for automating production.

Note: Post Producer only supports Final Cut Pro 7 XML files as source, which is used by Apple Final Cut Pro v7 and current versions of Adobe Premiere Pro CC. Final Cut Pro X uses a different schema that is not supported by Post Producer. See Creating Compositions from Final Cut Pro 7 XML.

Since the Vantage server and services are running as a user in a Windows domain, when you are using files from Mac OS X, the system should be set up to log in as a user on the Windows domain as well. If media is shared directly from the OS X volume, that folder must be set up as a Samba share, allowing the Vantage domain services full access to this share. Leaving the OS X system set up as a workstation will result in permissions issues when Vantage tries to access the files.

Note: If you are creating files on a Mac OS volume and the paths in the composition are not UNC paths, you must pre-process the composition using the Colocate action to convert the paths to the share that identifies the file directory before you can process the composition in Conform. See *Colocate Action* and *File* for details.

Submitting NLE Projects and Media Descriptor Files to Workflows

You can submit files from NLE projects and certain media descriptor files to a Post Producer workflow with a Compose action just like you submit media files to other workflows. Media descriptor files that you can convert include QuickTime Reference, EDL, TSEDL, and Trigger files.

Here are the ways that you can submit NLE projects and media descriptor files for conversion to CML:

- Dropping the file into a folder monitored by a Watch action-based workflow.
- Dropping a workorder file (which contains jobs to submit) into a folder monitored by a Workorder action-based workflow. See *Batch Processing Jobs with Workorders*.
- Manually submitting the file to a workflow in Workflow Designer.
- Using Workflow Portal to modify metadata and supply variable values on a job-by-job basis, submitting the file to target workflows starting with a Receive action.
- Programmatically generating and/or submitting the file to a Receive action-based workflow via a program using the Vantage SDK. See *Submitting Compositions Using the SDK*.

Creating Compositions from Avid Sequences

The Simple AAF Composer enables you to convert a Simple AAF file (typically exported from Avid Media Composer or Newscutter) into a CML file. The Avid AAF file format defines how collections of media files and ancillary data are referenced an Avid system.

Avid Simplified AAF files are handled as media source types in Vantage. AAF files can be exported from Media Composer and dropped into Vantage watch folders or submitted by Vantage SDK-based programs to a Receive-based workflow for processing.

The Post Producer Compose action configured with a Simple AAF composer parses the Avid Simplified AAF file into CML, and the Conform action transcodes the video as directed by the CML, processing audio and data track (D-Track) MXF Op-Atom media referenced in the file ingested from ISIS storage.

During processing, you can specify if all media should be tagged for direct-covert as appropriate. You can also supply a source folder string (usually, a UNC path to a share) to replace all local file path strings up to the file name, resulting in network-accessible file paths to your source media.

Note: If the source media files aren't accessible to the Vantage domain server executing the workflow, the Conform action will fail. Typically, the media folder is set up as a Windows share, with read privileges for the Vantage services using the files.

Exporting Simplified AAF Files

This topic describes how to export an AAF file and submit it to a Vantage Compose-based workflow for conversion to a composition.

The Link To Effects Mixdown export setting allows you to mix down both audio and video effects so that the exported AAF references only master clips. This is useful for workflows in Vantage that can ingest the exported AAF media for further encoding.

When the video is mixed down, if a segment of the video is an existing master clip or filler, a reference to that segment is added to the new sequence. If the segment is a transition or effect, a video mix-down occurs which creates a new master clip. The new mixed-down master clip is added to the project bin and a reference is added to the new sequence.

For the audio mix-down, the editing application can limit the number of tracks included in the exported sequence to the first 2, 4, 8, or 16 tracks. If an audio track contains at least one effect or gain change, the entire track is mixed down to a new master clip. The new mixed-down master clip is added to the project bin and a reference is added to the new sequence.

To perform a simplified AAF Export:

1. In the Project Window, click the Settings tab and select the Export Link to Effects Mixdown setting. If the setting does not appear, click the User Profile Selection menu and select Update User Profiles.The new setting appears in the settings list.
2. Select the sequence you want to export as a simplified AAF.
3. Select File > Export.
4. Click Options to display the Export Settings dialog.
5. Select AAF from the Export As menu.
6. Leave these options enabled:
   - Video/Data Details: Mixdown Video/Effects to V1.
   - Audio Details: Flatten Audio Tracks that Contain Effects.
7. Select the number of audio tracks to include in the sequence.
8. Select the Media Destination Drive where you want to save any newly created media. (This should be a drive that Vantage has access to.)
9. Click Save.
10. Select a location for the AAF file, name the file and click Save. The AAF file is saved to the selected server and folder that is being monitored by the Vantage workflow.

Creating Compositions from Final Cut Pro 7 XML

The Compose action's Final Cut Pro 7 Composer supports the XMEML standard, used by Final Cut 7 and the current version of Premiere Pro (but not Final Cut X).

- Vantage only supports media essences that Post Producer can decode. Thus, not all essences that can be used within Final Cut 7 or Premiere Pro can be used in Post Producer workflows.
- Post Producer workflows must have access to referenced media. Typically, the media should be stored on Windows shares that are accessible to Vantage. The folder location can be set globally in the Compose action in a Vantage workflow. You can use this method when all media is in a single folder. The file:// references in the XML export are not used, as they generally reference the localhost volume.
- The Colocate action Local Path component can also be configured to process a CML file using pattern search and replace from, so that multiple media locations can be used. For example, the Mac OS path can be set up for access via SMB and \\*localhost_drive**this_share*\ can be replaced by \\*network_server**this_share*\ or by a letter-drive it has been mapped to.
- Closed Caption propagation can be set globally in the FCP XML Composer in the Compose action for a given XML source.

Limits on Final Cut Pro 7 XML Features

The Final Cut Pro 7 Composer converts Final Cut Pro 7 XML files to Post Producer compositions, with certain limits:

- One bumper and one trailer can be specified in the Final Cut Pro 7 Composer.
- There is limited support for transitions. For example, all dissolve transitions are transformed to overlapping fades in CML.
- Internally-generated video elements (blacks/whites/bars) are not supported. You have to generate media files for these in the resolution/scan type, audio channel configuration, and frame rate you need, and use them as if they were regular media files on a file system.
- Most audio cross fades are supported.
- Dip to color transitions are supported. A canvas with the color specified in the source XML as RGBA will be inserted in the CML, and fades will be set in the overlaying video.
- There is limited support for image manipulation. Static opacity, scaling and positioning is supported, but cropping, rotation, or any kind of animated control is not.
- Support for static clip volume (keyframes are ignored).

- Support for audio gain filter.
- Overlays can be transparent PNG files. PSD single layer files with a straight matte should be used for transparency support.
- To enable media file substitution, a clip in the FCP sequence can be renamed as a variable— {$$<clip name>}—and a value can be set for the variable within the Vantage workflow, or via a job in a Workorder file. For details, see *Final Cut Pro 7 XML File Workflows*.
- Vantage does not support CML translation of text/titles.

Exporting Projects from Final Cut Pro

This topic describes how to create a Post Producer composition from a sequence exported from Final Cut Pro as a Final Cut Pro (XMEML) file.

Note: Final Cut Pro Version 7 is required to export sequences compatible for conversion to Post Producer compositions.

In Final Cut Pro:

1. Create the project as you normally would, with video, audio, and graphical elements and transitions.
2. Identify the elements that you want to replace with a different video or audio clip, or graphic file each iteration of this production and rename the element, on the timeline, using a variable pattern, as described immediately following.
3. Export the project as a Final Cut Pro 7 XML file. Save this file to a watch folder that is monitored by your Vantage workflow.

In *Final Cut Pro 7 XML File Workflows*, two prototype workflows are examined. Each is designed to ingest a sequence created in Final Cut Pro 7 or Premiere and exported as Final Cut Pro 7 XML. The sequence simply overlays a graphic over a media file with video and audio.

Exporting Projects from Adobe Premiere Pro

This topic describes how to create a Post Producer composition from a sequence exported from Adobe Premiere.

Note: Adobe Premiere Pro CC for Mac OS X or Windows is required to export sequences compatible for conversion to Post Producer compositions.

In Premiere:

1. Create the project as you normally would, with video, audio, and graphical elements and transitions.

2. Identify the elements that you want to replace with a different video or audio clip, or graphic file each iteration of this production and rename the element, on the timeline, using a variable pattern, as described immediately following.
3. Export the project as a Final Cut Pro 7 XML file (File > Export > Final Cut Pro 7 XML). Save this file to a watch folder that is monitored by your workflow or to a folder that can be accessed from Vantage, for manual submission in Workflow Designer.

In *Final Cut Pro 7 XML File Workflows*, two prototype workflows are examined. Each is designed to ingest a sequence created in Final Cut Pro 7 or Premiere and exported as Final Cut Pro 7 XML. The sequence simply overlays a graphic over a media file with video and audio.

Final Cut Pro 7 XML File Workflows

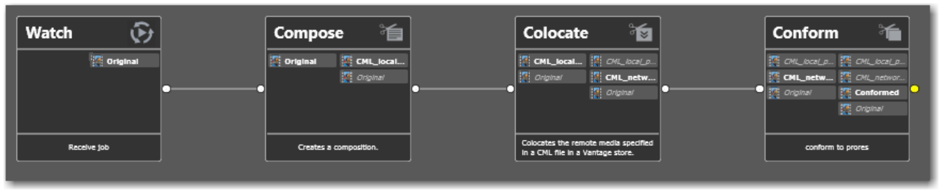
Two prototype workflows are examined. Each is designed to ingest a sequence created in Final Cut Pro 7 or Adobe Premiere and exported as Final Cut Pro 7 XML. The sequence simply overlays a graphic over a media file with video and audio.

The first workflow uses variables in the Composition; in the second, they are submitted externally.

- Converting FCP7 XML Files to a Composition with Variables
- Converting FCP7 XML Files to a Composition and Processing Them with a Workorder

Converting FCP7 XML Files to a Composition with Variables

This example workflow converts an exported Final Cut Pro 7 XML file to CML, populating a schedule variable directly in the Compose action to specify the name of the schedule graphic file to use at run time. The Colocate action can be configured with the Local Path options to convert Mac OS local paths to media files into share paths, so that the Conform action can access all of the required input media to generate the output media file.

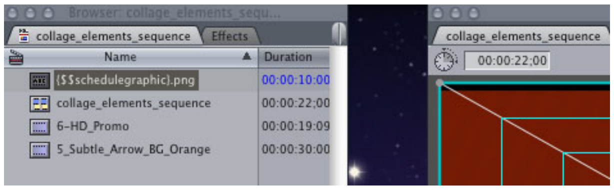


- *Watch action*—enables an operator to submit media files to an ingest folder: the CML file that specifies the clips in the program, and the associated metadata file.
  - *Media Files Nickname—Original*

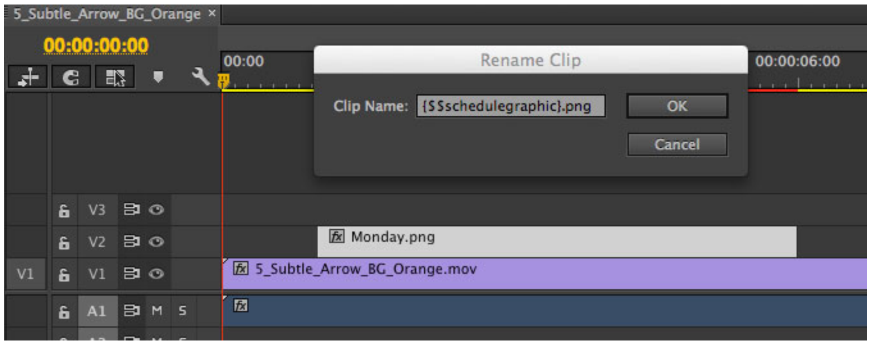
- *Compose action*—converts the exported Final Cut Pro 7 XML file into a CML file for processing in the Conform action.

  To specify the overlay file name at run time so it can be changed on every job, do two things:

  – In Final Cut or Premiere, use the Post Producer tokenization scheme—{$$<Variable Name>}.ext—where a graphic that is intended to be replaced on each job is renamed with a variable name in brackets. The extension is required for images to be properly handled in the transformation to CML. For example, *{$$schedulegraphic}.png*:


Browser: collage_elements_sequ...
collage_elements_sequence
Effects
collage_elements_sequence
00:00:22:00

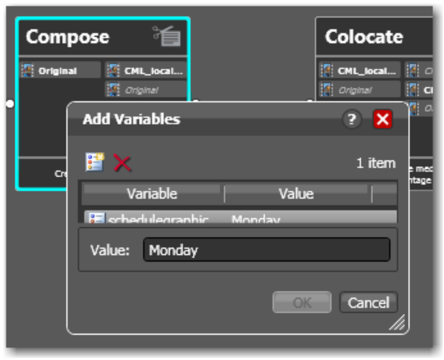
Name	Duration
{$$schedulegraphic}.png	00:00:10:00
collage_elements_sequence	00:00:22:00
6-HD_Promo	00:00:19:09
5_Subtle_Arrow_BG_Orange	00:00:30:00



Clip Name: {$$schedulegraphic}.png

  – In the Compose action, add a text variable with the same name (in this case, *schedulegraphic*), and set its value to the base name of the file you want to use (Monday—to identify the file as *Monday.png*). To rename a clip in the Premiere timeline, control click the clip and select Rename from the context menu.

Note: Note that for each tokenized value for media specified in the sequence, the workflow must assign a correct value to this variable, or the job will fail.



Colocate action—with the Local Path option, used as required to make Mac OS-located media files accessible to Vantage. You supply Mac OS source path string and target UNC share path strings. Colocate finds and replaces the source path strings with the corresponding share path strings in the CML file before passing it downstream.

This is required because the Final Cut Pro 7 XML file refers to local paths to media, even though the media in the sequence is located on a share (which it must be). The path replacement string (Target path) uses a share that is accessible to Vantage. The Prepend Target Path to locations without a Source Path option is also enabled, to create a fully-qualified path for the tokenized media, which does not have any path specified in the CML created by the Compose action.

Note: Instead of using a Colocate action in these examples, a single source folder can be specified in the Compose action for use by both Final Cut XML Composer and Vantage. The advantage of the Colocate action is that multiple local path colocators can be set up for multiple file locations.

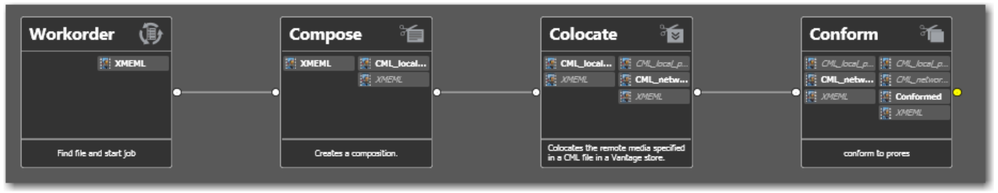
- *Conform action*—renders the media files (clips) referred to in the converted CML file into an output media file, as specified by you.

The use of this workflow requires that you manually change the value of any tokenized media variables each time you run it. While it is not typically convenient to do so, it provides a clear example of the use of tokens to use specified files at run time. For a more efficient method of using tokens, see the next workflow.

Note: You can also create an operator-driven workflow to implement job-by-job file names or other metadata. The target workflow must start with a Receive action. You should set up a Browse Windows configuration for Workflow Portal and specify the variable to update for each job you submit.

Converting FCP7 XML Files to a Composition and Processing Them with a Workorder

When using tokenized media, the more common scenario is to use a workorder so that multiple variable substitutions can be used automatically. The XMEML_compose_to_HD_from_workorder workflow provides this example.



The Workorder action uses a specific workorder configuration to process workorder files. (Workorder configurations specify the record layout, and are created in the Management Console.)

The Workorder action ingests a CSV file with a path to media and variables, instead of a typical source file. In this case, CSV file has a path to the Final Cut Pro 7 XML file, which is the source file for the Compose action, and a text variable which is used to provide a value for the tokenized graphic file from the XML source.

Here are two example jobs specified in the workorder:

```
\\Media\Source\collage_elements_sequence.xml,Monday
```

```
\\Media\Source\collage_elements_sequence.xml,Tuesday
```

The workorder populates the Media Nickname *XMEML* used in the workflow with the path to the source file (the first value), and the variable from XML/CML {$$schedulegraphic} with the value of the second value of the workorder (Monday, Tuesday.)

The workorder action processes the workorder file and submits two jobs, one for each row, with an graphic overlay for the day of the week.

Note: For full details on the use of workorders, see topics in the User Guide, Domain Management Guide, and context help topics in the Workorder action inspector.

Creating Compositions from EDL Files

The EDL Composer in the Compose action converts an EDL file from a Harris Velocity broadcast server system into a CML file.

When converting the EDL file for use in Post Producer, you can specify the frame rate for the media (if you do not, the Conform action by default uses the frame rate in the source media).

During conversion, you can manually add bumpers and trailers, perform centercuts and letterbox formatting during SD/HD conversion, and specify how to handle closed captioning and subtitles.

Creating Compositions from QuickTime Reference Files

The QuickTime Reference File Composer in the Compose action allows you to convert a QuickTime Reference file (typically created by exporting a sequence from an Avid system) into a CML file.

Note: This conversion should not be performed if closed captions are required, since the QuickTime reference file itself will not contain captions.

Creating Compositions from Trigger Files

The Trigger Composer is designed to use the SCTE trigger event types and times to create CML that brackets the program start, end, and ad spots within the show, producing a log of the ad timing in ad replacement workflows (Canoe, OTT, etc.).

During conversion, you can also enable insertion of keyframe segments, manually add bumpers and trailers, perform centercuts and letterbox formatting during SD/HD conversion, and specify how to handle closed captioning and subtitles.

You can create compositions from trigger files in two modes:

Ancillary Time Code Mode

When your trigger source is ancillary timecode, you submit the MXF file, and the Compose action uses the SCTE-104 triggers in the 436M track of the MXF file to create stream-conditioned VOD output with a Chronicle action that logs ad insertion points.

SCTE Triggers Mode

When your trigger source is SCTE triggers you submit a DAI Analysis Trigger XML file instead of a video file. You use an Analysis action configured with the Dynamic Ad Insertion Analysis component to generate the DAI Analysis Trigger XML file. The DAI component extracts the SCTE 104 and 35 formatted trigger data from the source file submitted to the Analysis action.

Creating Compositions from Telestream EDL (TSEDL) Files

The Telestream TSEDL Composer in the Compose action allows you to convert a Telestream TSEDL file into a CML file.

This composer should be used when you are creating TSEDL files manually or programmatically, with EDL content that conforms to the Telestream TSEDL specification.

When converting a TSEDL file for use in Post Producer, you can specify whether the in and out points are mark elements (and specify a frame rate) or they are represented as timecode elements.

Preparing Mezzanine CML Files for Conforming

The Telestream CML Composer in the Compose action is a general-purpose utility which modifies CML files submitted to it for various purposes, before submitting it to Conform for processing:

- You can add Letterbox or Centercut (by adding Crop/Mask elements) to compositions submitted from Workflow Portal before processing by the Conform action.
- You can automatically resolve variables and nicknames in an input CML from a VOD Portal submission, to generate an output CML capable of being rendered and encoded in a Conform action.
- You can enable closed captions and subtitles.

The CML Composer enables features or functionality to easily be integrated into complex, multi-step workflows that would otherwise be more difficult to achieve.

Creating Compositions from GraphicsFactory Templates

Telestream GraphicsFactory® is software integrated in Vantage that automates graphics insertion in file-based workflows. GraphicsFactory Jobs are submitted to a Vantage, Flip-based workflow (with GraphicsFactory filters enabled) for encoding using Workflow Portal.

The GraphicsFactory Composer enables Vantage customers who are using GraphicsFactory, to convert GraphicsFactory template files into a composition (CML) file so that it can be submitted and processed by the Conform action without using Workflow Portal and to meet processing requirements that exceed the capability of Flip-based workflows.

Creating Compositions from IMF Files

The IMF Composer enables you to convert an IMF file into a composition (CML) file for utilization in Post Producer.

IMF is a specification for an interoperable set of master files and associated metadata to enable standard interchange and automated creation of downstream distribution packages. IMF Composition packages include several different files (PKL, CPL (Composition Play Lists), media, etc.).

When a CPL file in the package is submitted to a Compose workflow, the IMF Composer assembles the sources and composition based on the data in that file and converts it to CML in preparation for submission to a Conform-based workflow.

Creating Compositions for Tempo Processing

The Tempo Composer enables you to submit jobs to Tempo-based workflows without requiring the use of the Workflow Portal, thus streamlining your processes. This is useful when, for example, no manual editing is required because you are using the Black and Slate/Spot analyzers in the Analyze action to identify segments for you.

The typical workflow framework in this application includes the Analyze action with Black and Slate/Spot analyzers, and a Compose action with the Tempo Composer, then on to the Tempo action for processing.

The Tempo Composer ingests the results of the Analyze action and the media file and generates a composition ready for processing by Tempo.

The Tempo Composer can generate various editions of CML for three types of applications:

Single Segment Mode

Single segment mode is designed to process commercials and normalize them by dropping leading slate and leading and trailing black, creating a single output segment.

CSV Mode

In CSV mode, you can supply (automatically from a media asset system for example, or manually) a CSV file describing the segments and produce a composition.

Typical

This mode provides the most flexible method of creating a composition that meets your processing requirements and provides the most settings and options.

Sample Composition Program

This chapter illustrates the use of the Vantage SDK to write a program that produces a CML file, highlighting the basic process.

Note: The Post-Producer developer's section on the Telestream site includes downloads of an SDK (Software Developer's Kit) that provide information, frameworks, and examples for software developers to create custom interfaces to work with Vantage and Post Producer.

Sample C# Composition Creation Program

This sample program illustrates the basics of using the Playlist class to produce a composition and save it as a file. This C# console program was developed in Microsoft Visual Studio 2012 using .NET 4.5.

Note the Composition methods which are used to create a new composition and populate it with CML elements and attributes.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using Telestream.Soa.Facility.Playlist;
namespace CompositionExample
{
  class Program
  {
    static void Main(string[] args)
    {
      // create new composition and add version and date
      Composition composition = new Composition ();
      composition.Version = new Version (1, 0);
      composition.Created = DateTimeOffset.MinValue;
      // create and add a source
      Source source1 = new Source ();
      source1.Identifier = 1;
      source1.Files.Add
        (new File("\\\\share\\path\\Segment_01.mov"));
```

```
            composition.Sources.Add (source1);
            // Create and add a new sequence
            Sequence sequence1 = new Sequence();
            sequence1.Layer = 1;
            composition.Sequences.Add (sequence1);
            // Create and add a new segment
            Segment segment1 = new Segment();
            sequence1.Segments.Add(segment1);
            // create and add a new video element
            Video video1 = new Video();
            video1.Source = 1;
            video1.Layer = 0;
            video1.Align = EdgeAlign.Head;
            video1.Fill = FillMethod.None;
            video1.Adjust = PartAdjust.Edge;
            segment1.Material.Add(video1);
            // create a title
            Title title1 = new Title();
            title1.Value = ("Title Element Text");
            title1.Duration = new Time(0, 0, 0, 5, true);
            segment1.Material.Add(title1);
            // write the composition to a file for processing
            composition.Serialize
              (@"\\share\path\Sample_Composition.cml");
        }
    }
}
```

Working with Material

The purpose of this chapter is to help you learn the best approach to working with material in the context of a composition: accessing source files, creating clips, composing media, applying visual effect, and aligning it on the timeline, for example.

Topics

Note: All of the examples in this chapter are complete compositions. To use them, copy and paste the example into a text editor. Change the references to the media files you are supplying (note the design specs as comments at the top of each composition), and save the file as a text file.
Then, submit it to a Post Producer conforming workflow, using Workflow Designer.

Introduction to Material

Post Producer supports six types of material that you can use in a composition; each defined by a specific element:

- *Video*—instances of temporal, frame-based video
- *Audio*—instances of audio, ignoring the video stream (if present)
- *Image*—instances of raster images, or image sequences
- *Advisory*—instances of advisory graphics and metadata
- *Title*—instances of text, rendered on the frame
- *Canvas*—instances of black rectangles, used for fades.

Material is arranged in sequences and segments within the timeline, along with formatting instructions: montage ordering, composition layering, audio mixing and mapping, etc.

Material Attributes

Each element of material includes attributes that describe it in relation to the other material, and also describe its properties. There are fundamentally two types of attributes in CML:

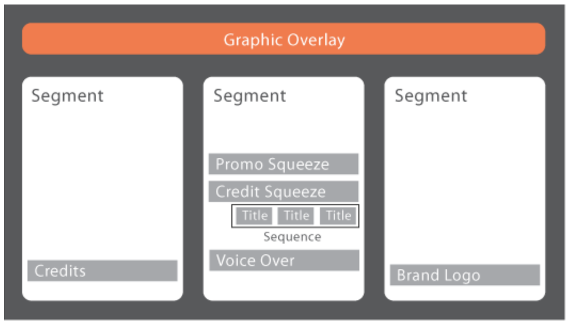
Spatial Attributes—these are attributes that describe characteristics of the material and its relationship to other material.

Transformational Attributes—attributes that describe the behavior and timing of the material: size, location, opacity, rotation. and audio volume.

The application of material elements is illustrated in the following graphic.

Story Boarding

It's often helpful to draw out your composition graphically (in an NLE or a graphics program, or on paper), to help you visually organize the material and determine specifically what you are trying to achieve in the CML. Here is a typical Composition story board and sample frame:



On the left, is a designer's story board of each element in the composition, arranged temporally, with three individual segments. Each segment is implemented when you want to divide the composition into distinct temporal segments, in which you can independently present media. On the right is a sample frame of the resulting video. Notice that this story board does not illustrate spatial arrangement; only temporal arrangement of material.

Story boarding helps you identify each segment and sequence, and each instance of material in every segment. Once you know these details, you can create your composition and begin adding attributes to control how each item of material should be presented.

Sample CML

The story board (or other form of design) must be implemented in CML (converted to a composition), and processed in a Conform workflow to generate the output media. Here is a very simple composition, with one instance of video, in one segment, within one sequence.

```
<?xml version="1.0" encoding="utf-8"?>
<Composition xmlns="Telestream.Soa.Facility.Playlist">
<!-- Basic CML Sample DUR=NA, AR=16x9, res. 1920x1080 -->
<!-- Input: 1080i Apple ProRes 25 FPS 16b 48k stereo -->
<!-- Output: 1080i x264 -->
<!-- Conform Workflow: Conform 1920x1080 x264 AAC + Flip Proxy -->
  <Source identifier="1">
    <File location="\\share\path\source_video.mov" />
  </Source>
  <Sequence layer="1">
    <Segment>
      <Video adjust="edge" fill="none" source="1" layer="0" />
    </Segment>
  </Sequence>
</Composition>
```

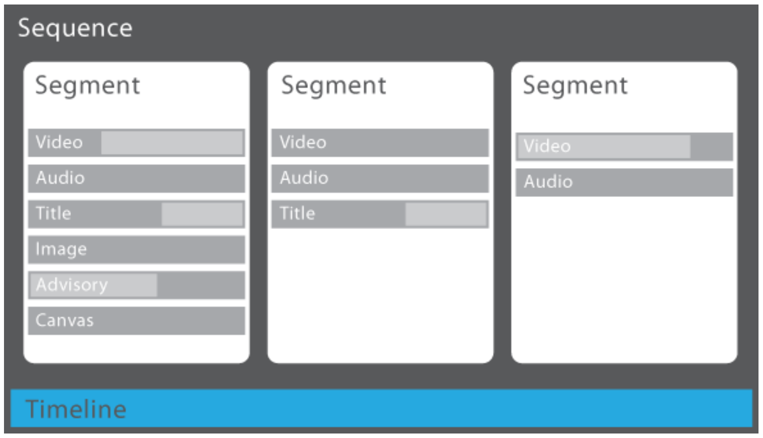
This is about as basic as it gets, and it illustrates how all these elements fit together.

Adding Material to a Composition

To add material (or a portion of it) to your composition, you add its element to the *Segment* where you want it to play, as you saw in the CML example above.

Note: For specifications of each element and its attributes, click on the link to view its reference topic in the *Composition Markup Language Reference* chapter.

Recall that each segment identifies a specific portion of the composition's timeline, based on its ordinal position in the sequence and segments play sequentially in their order in the Composition:



Also recall that each sequence creates an independently-functioning timeline within a composition—the point of a sequence is to provide you a new, independent timeline upon which you can place segments of material and play them independently of any other material in the composition.

Certain material is stored in files: Video, audio, and images. Post Producer also supplies Advisories with USTV images. For other rating systems, you must supply your own.

Canvas and Title material (which creates formatted titles from text) does not use files.

- Specifying Files for Video and Audio Material
- Specifying Files for Image Material

Specifying Files for Video and Audio Material

For each instance of video and audio that you use in a composition, you need to add a *Video* or *Audio* element to the segment. You can add as many video and/or audio elements as you need, and any number of them can create new instances of media from the same file. You can also create clip instances, by setting in and out points.

You also need a corresponding *Source* (a child of *Composition*), to identify the source file. By default, if a video source contains audio, it also plays when the video is played.

For convenient access, place *Source* elements as the first elements in the composition.

Post Producer allows you to combine source files with differing frame rates and Telecine cadences, plus SD, HD, television and 23.98 sources and re-encode them as specified in your output media specifications.

All video and audio files must be in a supported format that the Conform action can process. A file may contain the media essence, or it may reference additional files containing media essences (for example, a QuickTime reference movie). For a list of supported input formats, see *Supported Post Producer Formats*. If your video and audio files can't be decoded by the Conform action, you can add a Flip action to the workflow upstream of the Conform action (or create a separate workflow) to transcode the source file into a supported format.

Note: The Vantage Edit service (there may be more than one running in your domain) that executes the Conform action in the job must have access to each file. Because files are typically not stored on the same server as the Vantage service that processes them, Telestream recommends using shares to reference files. For details on dealing with file access by Vantage services, see the Domain Management Guide. You can use the Tempo Action to copy non-local source material to a location where all actions in your workflow can access it.

Here are the steps:

1. Add a *Video* or *Audio* to a *Segment*.
2. Add a *Source* to the *Fade* (usually at the top, where it's easy to access).
3. Add an identifier attribute to the video or audio element with a unique integer value in the composition.
4. Add a *Image* to the Source element, supplying a location attribute with a fully-qualified file path and the file name.
5. Finally, add a source attribute to the video or audio element, using the same integer value as the Source element's identifier attribute.

Video Example

In this example, one source file is specified, and one instance of it is used in a segment.

```
<?xml version="1.0" encoding="utf-8"?>
<Composition xmlns="Telestream.Soa.Facility.Playlist">
<!-- Basic CML Sample DUR=NA, AR=16x9, res. 1920x1080 -->
<!-- Input: 1080i Apple ProRes 25 FPS 16b 48k stereo -->
<!-- Output: 1080i x264 -->
<!-- Conform Workflow: Conform 1920x1080 x264 AAC + Flip Proxy -->
  <Source identifier="1">
    <File location="\\share\path\source_video.mov" />
  </Source>
  <Sequence layer="1">
    <Segment>
      <Video adjust="edge" fill="none" source="1" layer="0" />
    </Segment>
  </Sequence>
</Composition>
```

Why not just reference the file directly in the Video or Audio element? Because it is common to use multiple segments of material from a given source (for example, a master file). Also, in a source, you can perform basic modifications in one place: cropping and masking, audio mixing and routing, etc.

Audio Example

In this example, four source files are specified; one for video and three for audio. The CML creates a video file, inserting various language tracks on channels 3, 4, and 5.

```
<?xml version="1.0" encoding="utf-8"?>
<Composition xmlns="Telestream.Soa.Facility.Playlist">
<!-- Audio DUR=NA, AR=16x9, res. 1920x1080, DolbyE 5.1+2 -->
<!-- Input: 1080i -->
<!-- Output: 1080i x264 -->
<!-- Conform Workflow: Basic Conform 1920x1080 x264 AAC + Flip
Proxy -->
  <Source identifier="1">
    <File location="\\share\path\Amsterdam_promotion.mp4"/>
  </Source>
  <Source identifier="2">
    <File location="\\share\path\english.m4a" />
    <Mix source="1" target="3" level=".9"/>
  </Source>
  <Source identifier="3">
    <File location="\\share\path\French.wav"/>
    <Mix source="1" target="4" level=".9"/>
  </Source>
  <Source identifier="4">
    <File location="\\share\path\italian.wav" />
    <Mix source="1" target="5" level=".9"/>
  </Source>
  <Sequence layer="1">
    <Segment>
      <Video align="head" adjust="edge" fill="none" source="1"
layer="0">
        ...
      </Video>
      <Audio align="head" source="2" offset="00:00:00.50" />
      <Audio align="head" source="3" offset="00:00:00.50" />
      <Audio align="head" source="4" offset="00:00:00.50" />
    </Segment>
  </Sequence>
</Composition>
```

This audio is mixed or routed, as appropriate, so that viewers of different languages can hear the audio channel of their native language.

Specifying Files for Image Material

When you add an *Image* element (either a single image, or an image sequence) to a segment, the element directly references the file or files via the *location* attribute, as illustrated below (in bold).

```
<?xml version="1.0" encoding="utf-8"?>
<Composition xmlns="Telestream.Soa.Facility.Playlist">
  <Sequence layer="1">
    <Segment>
      <Image align="tail" adjust="body" fill="none" layer="2"
location="\\share\path\Arrow_Title-PNGs\myBrandLogo.png"
frames="72" layout="none">
        <Head>
          <Fade duration="00:00:01.000" />
        </Head>
        <Tail>
          <Fade duration="00:00:01.000" />
        </Tail>
      </Image>
    </Segment>
  </Sequence>
</Composition>
```

When you add a single image, you need to adjust its properties to play in a certain manner for a given duration. Likewise, you can supply an image sequence by referencing to the first file of a given pattern in a folder of sequential images.

To specify a file or file sequence for image material:

1. Add an *Image* to a *Segment*.
2. Add a *location* attribute to the Image element, which references the image (or the first image in the sequence).

If you are implementing an image sequence, file names must include a numeric sequence number (and no other numeric values), and there must not be a break in the sequence. Specify the first filename to use in the sequence. A break in the sequence causes the sequence to stop at the break.

You can use a wildcard asterisk (*) to solve two problems in your file list:

- If a break occurs in the sequence and you do not want it to stop, use a wildcard. For example: A series of files named `Banjo_470.dpx` (where 470 is the sequence number pattern) should be referenced with a wild card, such as `Banjo_*.dpx`. The directory is filtered to include all files that fit the criteria and sorted in ascending order.
- If the file names have additional numeric values that are constant, use a wildcard. For example: A series of files named `Banjo_25fps_470.dpx` (where 470 is the sequence number pattern) should be referenced with a wild card, such as `Banjo_25fps_*.dpx`. The directory is filtered to include all files that fit the criteria and sorted in ascending order (breaks are ignored when a wild card is used.)

For full details, see *Image*.

Specifying Files for Advisory Material

For each instance of an advisory that you plan to use in your output, you insert an *Advisory* in the correct segment.

If you are using the USTV rating system, default images (the larger ones) are supplied for each combination of ratings and reasons, so you don't need to reference a file. For other rating systems, you need to supply your own image and reference it in the Advisory element using the location attribute.

Here is an example of applying a USTV advisory to source 1, the introductory segment:

```
<?xml version="1.0" encoding="utf-8"?>
<Composition xmlns="Telestream.Soa.Facility.Playlist">
<!-- Advisory Example DUR=10s., AR=16x9, res. 1920x1080 -->
<!-- All PNG files: large USTV Ratings Bugs -->
<!-- Input: 1080i -->
<!-- Output: 1080i x264 -->
<!-- Conform Workflow: Conform 1920x1080 x264 AAC + Flip Proxy -->
  <Source identifier="1">
    <File location="\\share\path\source_video_intro.mov" />
  </Source>
  <Sequence layer="1">
    <Segment>
      <Video adjust="edge" fill="none" source="1" layer="0" />
      <Advisory align="head" adjust="body" fill="none" layer="1"
duration="00:00:05.000" type="USTV" rating="TV-Y" />
    </Segment>
  </Sequence>
  <Source identifier="2">
    <File location="\\share\path\source_video_main.mov" />
  </Source>
...
</Composition>
```

Extracting Clips from Master Media

The purpose of this topic is to explain how to perform frame-accurate editing—extracting clips from master files at run-time—directly in your compositions.

Note: CML is *frame*-accurate; not *field*-accurate. Thus, timecodes with an asterisk (denoting the 2nd field boundary) are not supported—don't put timecodes with an asterisk in CML.

Much of the video that you use as source material in Post Producer applications is longer than you require for a given promo. For example, you may want a 30-second clip of a movie that's about to air. Or, you may want to create a squeeze-back video and run the credits of the last movie that aired, in an Up Next promo. Thus, you need a way to create clips.

There are two practical ways to create video and audio clips for use in a composition:

- *Create clips externally, then reference each clip for use in a composition*
- *Create clips dynamically from master media directly in a composition*

To use clips that have been created externally, you simply add a Source element, and then reference the media as appropriate (*Adding Material to a Composition*).

However, you may have a master file—a full-length movie, for example—from which you want to identify and extract several smaller clips for your highlights composition.

As an alternative, Composition Markup Language also enables you to dynamically create clips from master media —producing the clips at run time by extracting the segment with mark-in and mark-out points—using *Head* and *Tail* elements.

Overview

To extract a clip (create an instance of media) from a master file, you need to:

- Identify the source (using a Source element) that you want to extract clips from.
- Set mark in and mark out points by adding *Head* and *Tail* elements with a *Fade* to the *Audio* or *Video* element on the media that define the portion you want to use.

Here are the CML elements that you'll use when defining a clip:

- Heads and Tails—the point in the media which defines the mark-in (*Head*) and mark-out (*Tail*) point.
- Edit—when used in a Head, the mark-in point of the clip. When used in a Tail, the mark-out point of the clip.

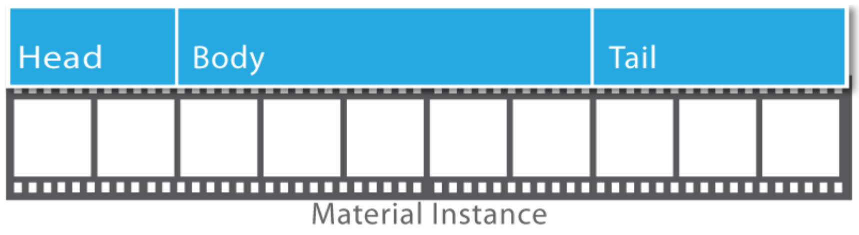
Together, the Head and Tail elements with an Edit thus define the clip of material to be extracted from the source file.

Note: It is important to note that Head Edits are inclusive; Tail Edits are exclusive. That is, the first frame after the specified time in the Head Edit is *included* in the clip. However, the first frame after the specified time in the Tail Edit is *excluded*.

Heads, Tails, and Bodies

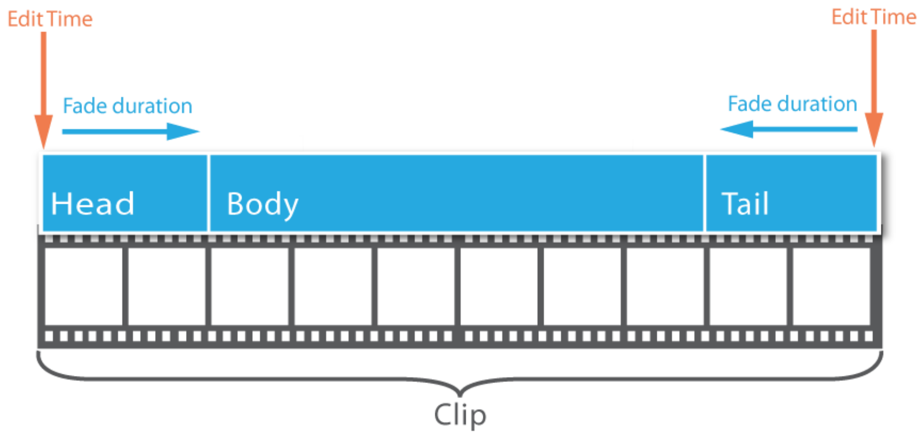
To crop media from a master file to create a clip, or to apply effects to your material, you need to understand how to use and implement heads, tails, and bodies. (If you don't have effects and you don't need to clip the instance, there's no point in having a Head, Tail, or Body.)

First, lets examine a graphic illustration of these components of an instance of media (a Video or Audio element): Heads, Body, and Tail are points on material:



The Head, Body, and Tail are temporal locations (mark-in and mark-out points) on the timeline of the material instance—but they're not depicted that way here. Instead, they are depicted as having a duration, which is usually the easiest way to think of using them, because in most cases, Edits are implemented with Fades, which are fundamental to applying effects over a series of frames—a period of time.

The mark-in point of the Head is the left edge—by default the beginning timestamp or timecode of the material. But, by adding an Edit element to a Head, you can set a mark-in point to trim the beginning of the clip. Likewise, you can set a mark-out point in the Tail, to trim it from the end of the clip. (The mark-out point of the tail is of course the right edge—the mirror opposite of the Head.) Without an Edit, the mark-out point of the tail is the end of the clip. Edit elements can only be added to a Head and a Tail.



By definition, a Head can not begin (have a time) before the beginning of the instance. It can be later (with an Edit), and it can have an effect duration (Fade), extending to the right (forward) along the instance. Similarly, a Tail can not be later than the end of the instance, but it can be earlier with an Edit, and it can also have a Fade, extending in this instance backward along the instance.

The Body's point (and duration) is not explicit—it's implicit—controlled entirely by how you set up the Head and Tail with Edit and Fade elements.

Note: You can also add a Fade element, to control each effect's duration. The Fade element is what gives a Head, Body, and Tail its length, so to speak. Using a Fade is describe in detail in *Applying Effects to Material*.

Example

Here is a simple example, with bold text identifying the operative elements:

```
<?xml version="1.0" encoding="utf-8"?>
<Composition xmlns="Telestream.Soa.Facility.Playlist">
<!-- Up Next Promo DUR=1 min,16x9, 1920x1080, DolbyE 5.1+2 -->
<!-- All PNG files: 600 x 420 -->
<!-- Input: 1080i -->
<!-- Output: 1080i x264 -->
<!-- Conforming Workflow: 1080 VOD -->
  <Source identifier="1">
    <File location="\\share\path\clip1.mov" />
  </Source>
  <Source identifier="2">
    <File location="\\share\path\master.mov" />
  </Source>
  <Sequence layer="0">
    <Segment>
      <Video source="1" layer="0" />
      <Video source="2" layer="1" align="head" adjust="edge"
fill="none" >
        <Head>
          <Edit time="00:00:04.000" />
        </Head>
        <Tail>
          <Edit time="00:00:22.200" />
        </Tail>
      </Video>
    </Segment>
  </Sequence>
</Composition>
```

The Source elements in this composition define the media files to be used in the composition. The *identifier* value in each source is what you use in the Video element's *source* attribute to reference the correct file.

The first Source points to *clip1.mov*, a clip which has been clipped and saved as a file outside of Post Producer, for use in this composition as-is. The second Source points to *master.mov*, a master file from which we want to extract a clip dynamically.

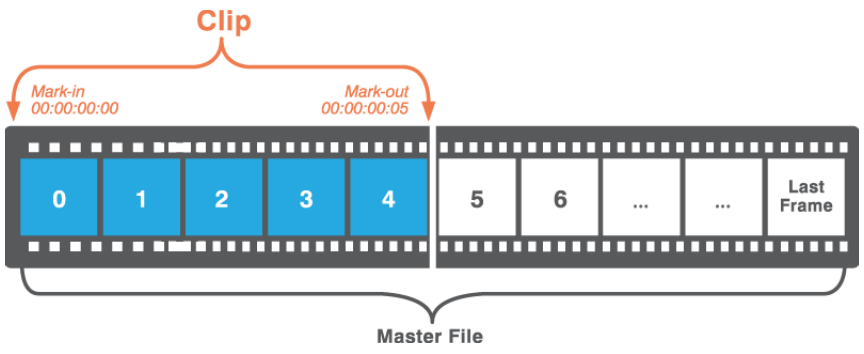
Here's how the second Video element is modified to create a clip dynamically:

1. Add a Head element to define the mark-in point on your master media file.
2. Add an Edit to the Head and specify the mark-in time.
3. Add a Tail element to define the mark-out point on your master media file.
4. Add an Edit to the Tail and specify the mark-out time.

Specifying Mark Times

In both the Head and the Tail, you specify the time by adding an Edit element and configuring it. The Edit element has two attributes—*time* and *mode*. Together, these two attributes enable you to accurately identify the mark-in and mark-out point in various situations.

Note: The time of the mark-in point on the master file is *inclusive*; the mark-out point is *exclusive*—it references the point in time immediately following the last frame to be included—the first frame to be excluded. For example, you specify mark-in 00:00:00:00 (the first frame in the clip) and mark-out 00:00:00:05 (the 6th frame in the clip), to create a clip with 5 frames (0, 1, 2, 3, and 4), as illustrated in the figure below.



Using the Mode Attribute

The mode attribute has 3 keyword values:

- *absolute* (default)—used when a timecode is present in the source; identifies the edit point based on the timecode. If you supply a timecode but the source lacks a timecode, it is converted to a timestamp value and utilized.
- *relative*—specifies an edit point relative to the beginning (in a Head) or ending (in a Tail) of the source, ignoring the timecode (if present).
- *duration*— specifies an edit point measured as a length of time, from the end (in a Head) or the beginning (in a Tail) of the source. (You can't use duration in both Head and Tail on the same instance of media.)

Using the Time Attribute

You can specify the *time* value using either a timestamp or a timecode, as appropriate:

- *Timestamp Notation*—is always specified with seconds as a decimal number, in the pattern HH:MM:SS.sss, with seconds specified to millisecond (one thousandth of a second) accuracy, generally using 3 digits (there is no requirement to use 3 digits).
- *Timecode Notation*— is always specified in the patterns (HH:MM:SS:FF | HH:MM:SS;FF | HH:MM:SS:FF@FPS), optionally using a drop timecode indicator (;) for NTSC timecodes. (Timecodes with an asterisk (denoting the 2nd field boundary) are not supported. CML is *frame*-accurate, not *field*-accurate. Do not put timecodes

with an asterisk in CML.) Best practice is to specify a frame rate when using time-code notation.

Note: You should always use non-drop-frame timecode, unless the timecode in the source is drop-frame timecode. Then you should specify your timecode as drop-frame timecode (using a semicolon between the second and frame value) as well.

Timecode mode should only be specified as *absolute* in an Edit point when timecode actually exists in the source.

When you specify a timestamp, Post Producer ignores the timecode in the source media, even if it exists. The timestamp always represents the temporal distance from the beginning of the media, whether there is timecode in the source media or not.

When you specify a timecode, Post Producer uses the timecode in the source material, if it exists—the time represented by your timecode is relative to the timecode in the file. Thus, it must be consistent with the type of timecode in the source. For example, if your media has a timecode of 01:00:00:00, your timecode must be specified relative to that timecode.

Timestamp and Timecode Examples

The following example illustrates how to use *mode* and *time* attributes in Edit elements.

Head Edit in Absolute Mode

```
<Head>
  <Edit mode="absolute" time="01:00:02:12@29.97"/>
</Head>
```

This material must contain this timecode, or the job will fail. Assuming that the timecode begins at 01:00:00, the mark-in for this video is set to 2 seconds, 12 frames.

Head Edit in Relative Mode

```
<Head>
  <Edit mode="relative" time="00:00:02:12@29.97"/>
</Head>
```

This material may or may not have a timecode; it is ignored if present. The mark-in for this video is set to 2 seconds, 12 frames from the beginning of the source.

```
<Head>
  <Edit mode="relative" time="00:00:02.500"/>
</Head>
```

This material may or may not have a timecode; it is ignored if present. The mark-in for this video is set to 2.500 seconds from the beginning of the source.

Head Edit in Duration Mode

```
<Head>
  <Edit mode="duration" time="00:00:54.200"/>
</Head>
```

In *duration* mode, you can specify a timestamp or a timecode value. In this example, the Head's mark-in point is set at 54.200 seconds back from the end of the source.

Tail Edit in Absolute Mode

```
<Tail>
  <Edit mode="absolute" time="01:00:30:00@29.97"/>
</Tail>
```

This material must have a timecode, or this will fail. Assuming that the timecode begins at 01:00:00, the mark-out for this video is set to 30 seconds from the beginning.

Tail Edit with Relative Mode

```
<Tail>
  <Edit mode="relative" time="00:00:02:12@29.97"/>
</Tail>
```

In a Tail element, when specifying a timecode, Edit sets the mark out point at 2 sec, 12 frames from the end of the source.

```
<Tail>
  <Edit mode="relative" time="00:00:02.500"/>
</Tail>
```

In a Tail element, when specifying a time, Edit sets the mark out point at 2.500 seconds from the end of the source.

Tail Edit with Duration Mode

```
<Tail>
  <Edit mode="duration" time="00:00:30.000"/>
</Tail>
```

In *duration* mode, you can specify a timestamp or you can use a timecode value.

The point is specified from the opposite end. For example, the mark-out point in this example is set at 30.000 seconds from the beginning of the source or the Head element, if specified. Thus, the instance of material will run for (have a duration of) 30.000 seconds from the Head.

Cropping and Masking

Sometimes, you need to crop or mask your video in a segment. For example, you might want to use SD material in an HD promo, and thus, need to crop and mask it.

You can perform cropping and masking directly in a composition. To do so, you add a *Crop* and/or *Mask* to the *Source* that identifies your source video file, and configure the Crop to suit your input frame size and the mask your output target aspect ratio and/or output frame size requirements.

Calculating the New Frame Size

Both elements have top, left, right, and bottom attributes. For maximum flexibility, you can supply integer or real values, identified as a fraction (no designator), pixels (px) or percent (%). If you do not supply a unit designator (% or px), the value is a ratio (for example, 0.222 = 222/1000 or 22.2%) of the frame.

The origin X and Y coordinates are the top left corner—0%, 0% or 0,0 pixels. The bottom right corner is indicated by 100%, 100% or X,Y pixels, where X and Y are the input frame's pixel dimensions.

Cropping Details

You perform cropping (make the frame smaller) to remove material from your input frame *before* creating an output frame with a mask. Cropping is performed on the input frame.

The left and top edge origins are at 0, 0 pixels; the right and bottom edge origins are the dimensional width and height of the frame.

Top and left adjustments values are in relation to the origin; bottom and right adjustments are relative to the bottom right corner. Positive values are downward or to the right; negative values are upward or to the left.

Thus, the top and left values should be positive to reduce the frame size; the bottom and right values should be negative to reduce the frame size.

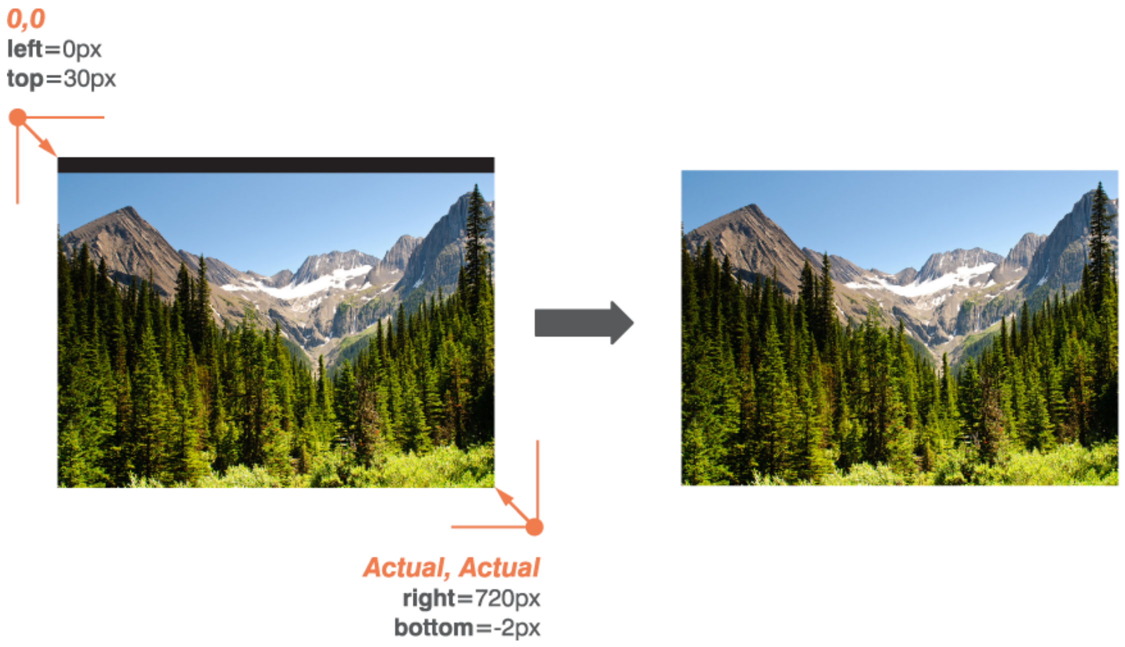
For example, to obtain a centercut from 1080i source for 4x3 SD output, you could specify:

```
<Crop top="0%" left="12.5%" bottom="100%" right="-12.5%" />
```

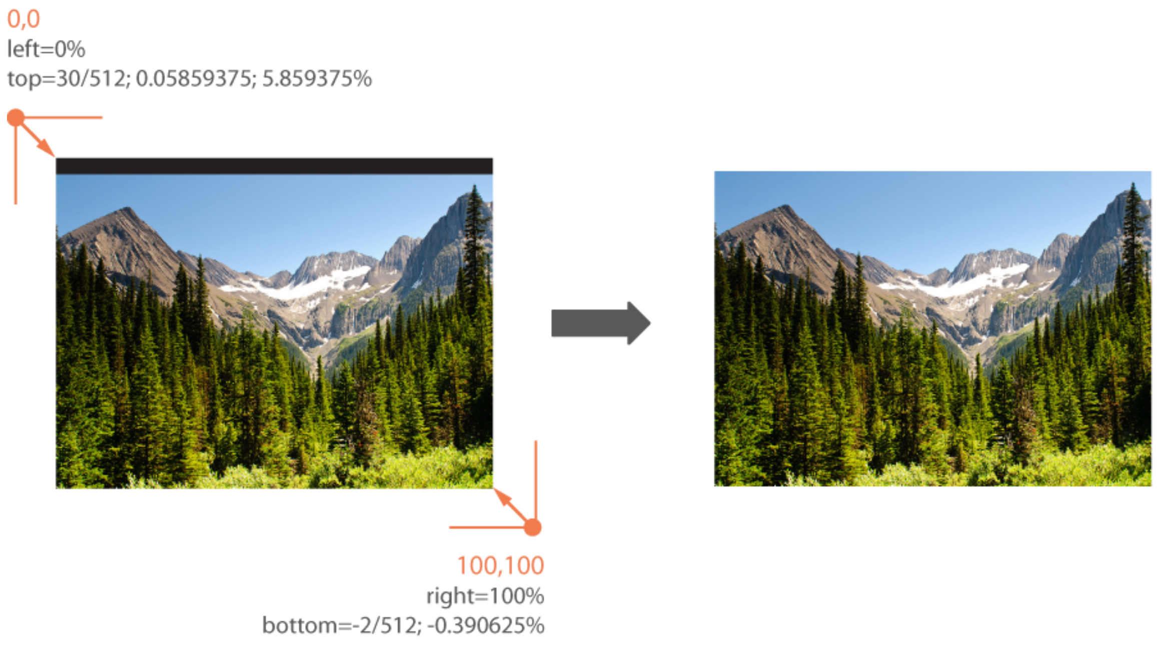
or

```
<Crop top="0px" left="240px" bottom="1080px" right="-240px" />
```

Here is an example of specifying Crop dimensions in pixels:



In this example, VBI is being stripped from SD source using pixels: 30 pixels from the top, 2 pixels from the bottom.



As shown in this example, percent, fractions and ratios can be used.

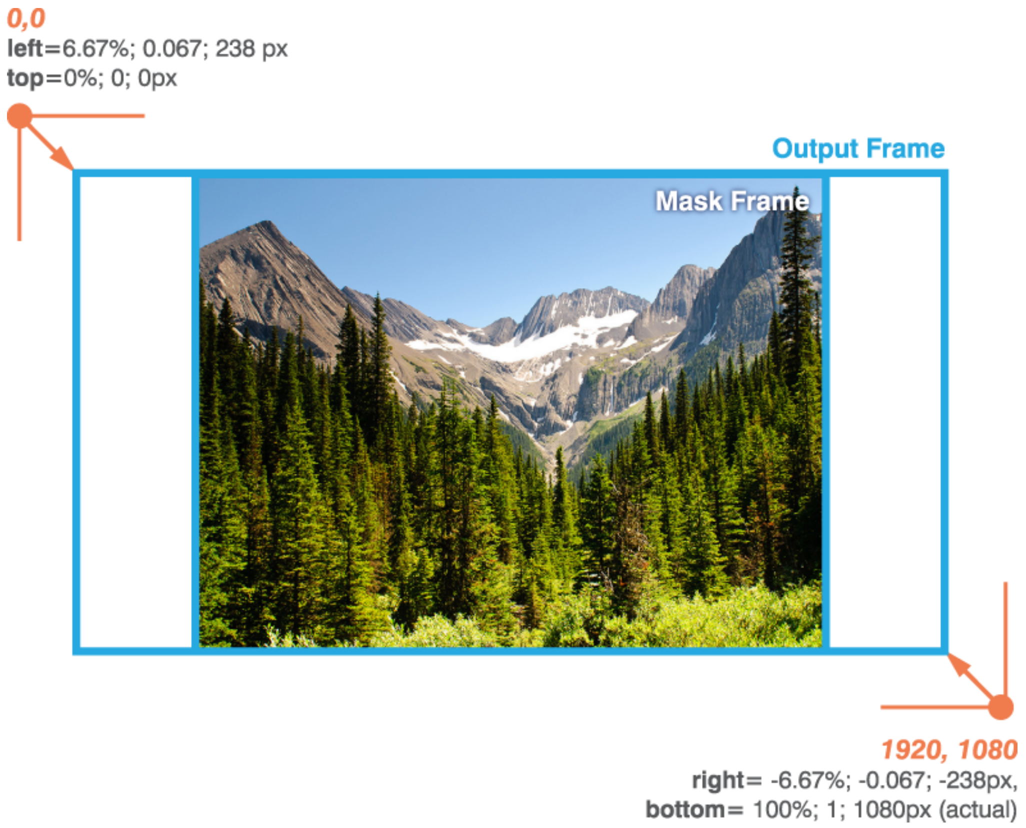
Base your crop values on the increase downward and to the right from 0 for top and left values, and the decrease to the left and upward from 100% or N (dimension) pixels for bottom and right. (The bottom and right dimensions obviously may change from job to job, based on the frame size of your source media.)

Masking Details

Masking on the other hand, is performed to implement an active rectangle inside the output frame, relative to the output frame size—often to preserve aspect ratio.

Note: Masking is usually applied only when you have mixed media—converting input media of one aspect ratio to output media of another aspect ratio.

When scaling occurs (for example, from SD to HD), the input frame (after optional cropping) is scaled into the masked (active) region (if specified) and display correctly. This frame illustrates a typical masking application.


0,0
left=6.67%; 0.067; 238 px
top=0%; 0; 0px
Output Frame
Mask Frame
1920, 1080
right= -6.67%; -0.067; -238px,
bottom= 100%; 1; 1080px (actual)

You also base your mask values on the difference from 0 for top and left, and from 100% or N (dimension) pixels for right and bottom dimensions of the output frame.

Note: By using percent as your unit of measurement on a mask, you can make various HD-aspect ratio output files, without changing the mask specification.

When masking, the default fill color for either pillars or bars is black. To change the color, use the *color* attribute (as shown below in the example). Keyword values are *black* | *white* | *ARGB value*, where only opaque colors (alpha = FF) are supported.

To specify a custom fill color, specify an ARGB value indicated by the # sign, followed by 4 pairs of hex values: Alpha, Red, Green, and Blue. The Alpha value FF is fully-opaque, and only opaque colors are supported.

Example

This example converts an SD source file to an HD target aspect ratio. The source contains an NTSC IMX video track (720 pixels by 512 lines). Use the Crop element to remove the vertical blanking lines, and then use the Mask element to convert the source to a 1080 output frame with a 16:9 target aspect ratio by adding a 2:9 (0.222) curtain to each side of the image.

```
<?xml version="1.0" encoding="utf-8"?>
<Composition version="1.0"
xmlns="Telestream.Soa.Facility.Playlist">
<!-- Name: Crop and Mask Demo -->
<!-- Output: 1080i x264 w/DolbyE -->
  <Source identifier="1">
    <File location="\\share\path\1_A2_Show_ONLY_NO-GFX2.mov" />
    <Crop top="30px" left="0" bottom="-2px" right="100%" />
    <Mask top="0" left="0.222222" bottom="100%" right="-0.222222"
color="#FFA2A4A6" />
  </Source>
  <Sequence layer="1">
    <Segment>
      <Video adjust="edge" fill="none" source="1" layer="0" />
    </Segment>
  </Sequence>
</Composition>
```

The Crop element removes 30 lines (pixels) from the top of the source frame, none from the left edge, 2 pixels from the bottom, and no material from the right.

This mask leaves the top edge intact; while removing 22/100 (22%) of the pixels from the left edge. When encoding 1080 HD, this creates a mask boundary 238 pixels in from the left, filling the left pillar with the fill color. The bottom edge is also left intact. The mask also removes 22% of the pixels from the right edge, creates another mask boundary 238 pixels in from the right as well. The region bounded by this rectangle is now a 4:3 ratio, so that the SD image can be scaled proportionally into the new HD frame and presently accurately, at a larger size.

Applying Effects to Material

One of the most powerful (and flexible) features of Post Producer is its ability to manipulate media as it is being played out. You can create dissolves, fades and dips to color, animations, and horizontal and vertical wipes by applying the following effects to material (*Advisory* / *Canvas* / *Image* / *Title* / *Video* / *Audio*) elements:

- *Opacity*—when you want to change the relative opaqueness material, you add an Opacity element to it. For implementation details, see *Segment*.
- *Scaling*—when you want to change the display dimensions of material, you add a Scaling element to it. For implementation details, see *Segment*.
- *Translation*—when you want to move the material around in the frame, you add a Translation element to it. For implementation details, see *Translation*.
- *Rotation*—when you want to rotate material on the Z axis, you add a Rotation element to it. For implementation details, see *Rotation*.
- *Volume*—For audio material (or video that has audio in it), you can adjust the volume. For implementation details, see *Volume*.

Each effect has its own element. However, you don't apply these effects directly to a material element (a Video, for example). Instead, you apply it (as appropriate), to portions of the material, to achieve animation effects and smooth changes over time.

In Post Producer, these portions are referred to as Head, Body, and Tail. By identifying each temporal part of the media (and specifying its mark-in or mark-out point and duration)—and applying effects as appropriate to each portion, you can apply effects in almost limitless ways, including fades and animation.

Controlling Material on the Timeline

The functions we discuss in this topic are all about time: they all relate to the temporal location and playout of material (Advisory | Canvas | Image | Video | Audio | Title) in a segment. Each function is specified by a given attribute on the material element, and they are all inter-related.

These functions *only come into play* when the duration of the material is less than the duration of the segment.

These functions provide a flexible method to define the behavior of overlay material so that it is consistently applied, regardless of the duration of the underlying content. For example, you can consistently display a branding bug (or an advisory bug) on underlying video of arbitrary duration. Or, you can squeeze back a credit roll, and cause them to always end 2 seconds before the end of the promo, regardless of their length.

Note: In practice, most of these situations involve visual material; audio is treated similarly of course.

There are several aspects that come into play when modifying material temporally. Each aspect is identified by a specific attribute you can add to the element:

- *Specifying the Duration of Material on the Timeline*—How long to play the material
- *Offsetting Material on the Timeline*—When to start (or stop) the playout of the material
- *Aligning Material on the Timeline*—Which point to anchor so the other end floats depending on duration
- *Adjusting Material on the Timeline*—What to adjust—the body or the edge (using *adjust*)
- *Modifying the Display of Material on the Timeline*—What portion of the material to replay

Specifying the Duration of Material on the Timeline

The duration of temporal material (video and audio) is specified by the length of the material in the instance. Thus, the duration attribute is only used for material that does not have a native length, or duration: Image, Advisory, Canvas, and Title.

Duration specifies the amount of time to display the material, and the default value is 0.

Frame-accurate time can be specified using a time (represent as timestamp—SS.sss, or a timecode—SS|FF), with these patterns: *HH:MM:SS* | *HH:MM:SS.sss* | HH:MM:SS:FF | HH:MM:SS:FF@FPS.

Rates supported: 23.976, 24, 25, 29.97, 30, 50, 59.94.

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.

Timecode references are applied to the timecode track specified in the associated Source element's file.

When using DF timecode, the first 2 (29.97 fps) or 4 (59.94 fps) timecode numbers are dropped every minute except minutes divisible by 10. DF timecode is supported for frame rates of 29.97 and 59.94 frames per second.

Examples:

```
<Advisory duration = "00:00:05.500"... />
<Advisory duration = "00:00:05;14@29.97"... />
```

Offsetting Material on the Timeline

By default, all material (Image, Title, Video, etc.) instances in a segment begin at the same time. To change the time when a given instance begins to play, add an *offset* attribute, and supply a time value.

For example: offset="00:00:10.000" causes this material to start playing 10.000 seconds after the segment begins, provided that the align is set to Head.

Note: You can use a timestamp or timecode value, based on the media and your requirements. For implementation details, see the offset attribute description in any material element.

Aligning Material on the Timeline

When the duration of the material is less than the duration of the segment, you can add an *align* attribute to your material element, and assign the appropriate keyword to pin (anchor) the material to the timeline of the segment. When you pin (or anchor) a point, you prevent it from being altered, when the length of the segment changes from job to job. By default, align = head.

These the keywords that apply to an align attribute:

head (default)—Anchors the beginning of the material to the start of the segment.

tail—Anchors the end of the material to the end of the segment.

both—Anchors the center of the material at the center of the segment, so that the Head and Tail are equidistant from the beginning and end of the segment.

Note: If an offset is used in this situation, it is also applied to both ends.

Tail Alignment Example

In this example, you are creating a one minute What's Next promo on a movie channel, and you want to show a preview which may run up to 30 seconds. You always want the preview to end two seconds before the end of the promo, leaving the beginning of the clip (regardless of its length) unspecified:

Example: `<Video align="tail" offset="00:00:02.000"... />`

Using this code, the video is aligned at the end (the tail) of the clip, and it is offset from the total segment duration by 2 seconds from the end. The beginning timecode is not specified: it is calculated by the Conform action when the job runs, based on the exact length of the clip.

Both Alignment Example

Continuing the What's Next promo example, you need to add an Advisory to the promo. You always want advisory bug to start two seconds after the start, and 2 seconds before the end of the promo.

Example: `<Advisory align="both" offset="00:00:02.000"... />`

Using this code, the advisory bug is aligned at both ends of the clip, and it is offset from the total segment duration by 2 seconds on each the end.

Adjusting Material on the Timeline

Use the *adjust* attribute to specify which portion of the material should be extended, using the keywords: *edge* | *body*. For example, using the *adjust* attribute, you can consistently display a branding bug on underlying video of arbitrary duration.

Here are the keywords you can apply to adjust:

edge (default)—The material is adjusted by extending the edge that is *not* identified by the *align* property and is applied to the duration of the head and tail of the material.

body—The material is adjusted by extending the body of the material. It is applied to the end of the body that is *not* identified by the *align* property.

When you adjust an item, you indicate whether you want to extend the duration of the body, or of the edge. When you adjust the body, the head and tail duration remain fixed; when you adjust the edge, the edge that is not anchored (see Align) is the one that is extended.

Example

In this example, 4 PNG files are played out one after the other by using the offset attribute. With an image (which has no temporal value), you have to specify how long to play the image (in this case, 9 seconds), and what part of the material to adjust—the body.

```
<Segment>
  <Image align="head" adjust="body" fill="loop" layer="2"
location="\\share\path\SAV_1700_1_0001_curCon.png" frames="1"
duration="00:00:09.000" layout="stretch" />
  <Image align="head" adjust="body" fill="loop" layer="3"
location="\\share\path\SAV_1700_2_0001_curCon.png" frames="1"
duration="00:00:09.000" layout="stretch" offset="00:00:10.000"/>
  <Image align="head" adjust="body" fill="loop" layer="4"
location="\\share\path\SAV_1700_3_0001_iconmap.png" frames="1"
duration="00:00:09.000" layout="stretch" offset="00:00:20.000"/>
```

```
  <Image align="head" adjust="body" fill="loop" layer="5"
location="\\share\path\SAV_1700_4_0001_fiveday.png" frames="1"
duration="00:00:09.000" layout="stretch" offset="00:00:30.000"/>
</Segment>
```

Modifying the Display of Material on the Timeline

By default, material is only played once per instance in a segment. When the length of the material is less than the duration of the segment, the use of the *fill* attribute determines if and how material continues to play until the end of the segment.

There are three keywords you can use:

none (default)—do not extend the material. (In theory, it actually plays to the end, as empty: no visual or auditory content.)

hold—continue playing the first or last frame to play the material the entire duration of the segment. If *align="head"*, the last (or only) frame is repeated. If *align="tail"*, the first (or only) frame is repeated.

loop—play the media repeatedly for the duration of the segment.

Of course, for temporal (multi-frame) media (video and audio), this plays out differently than for non-temporal (single-frame) media (Image, Advisory, Title, and Canvas).

Composition Markup Language Reference

This chapter describes each element in Post Producer Composition Markup Language, and its attributes. The elements are listed below in alphabetic order. The CML hierarchy map shows each element in relation to other elements.

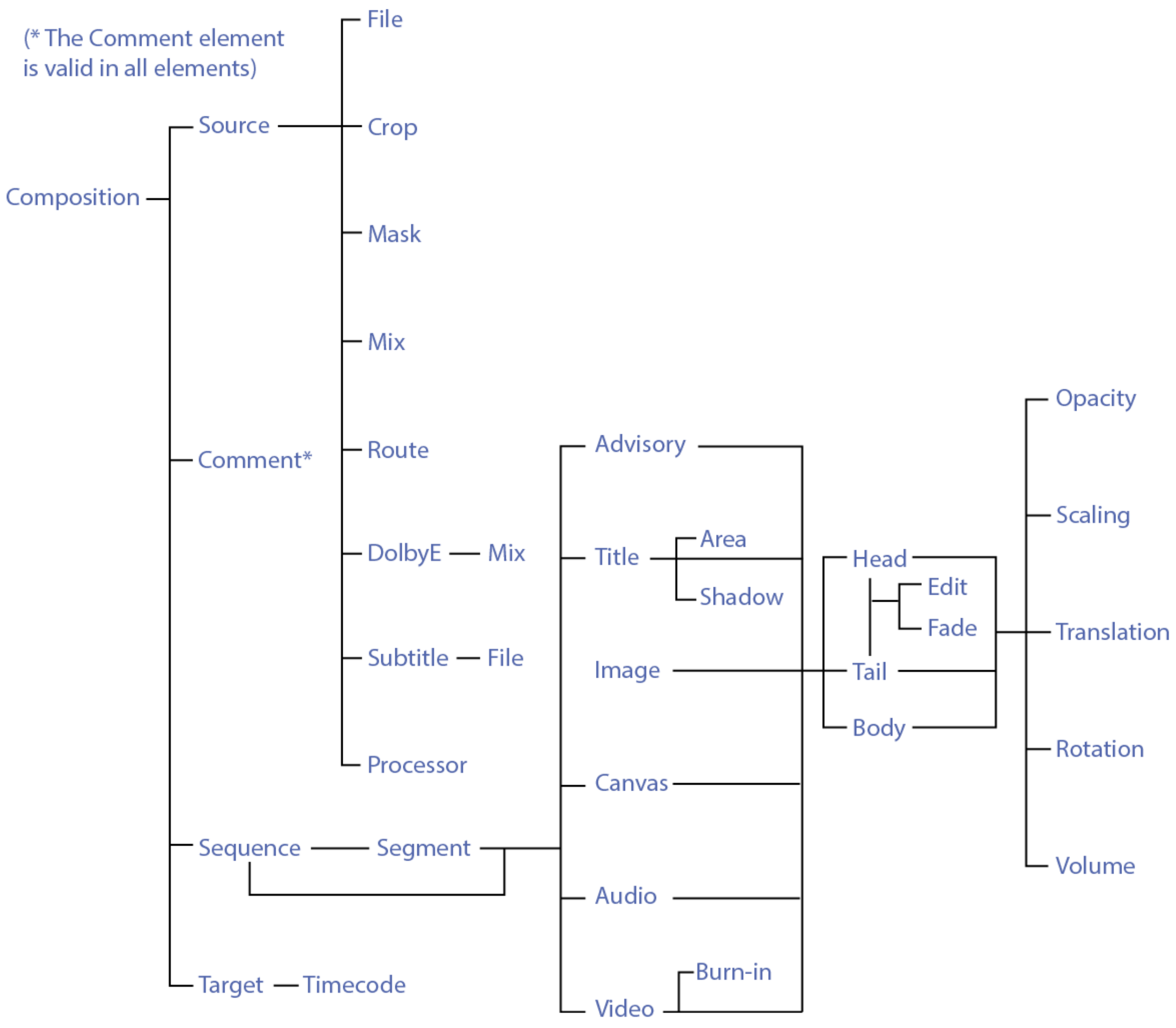
Topics

CML Elements

- Advisory
- Area
- Audio
- Body
- Burn-in
- Canvas
- Comment
- Composition
- Crop
- DolbyE
- Edit
- Fade
- File
- Head
- Image
- Mask
- Mix
- Opacity
- Processor
- Rotation
- Route
- Scaling
- Segment
- Sequence
- Shadow
- Source
- Subtitle
- Tail
- Target
- Timecode
- Title
- Translation
- Video
- Volume

CML Hierarchy Map

This tree structure illustrates the relationship between each of the elements in Composition Markup Language.



Overview

These topics provide general information about Composition Markup Language.

- Case Sensitivity is Critical in CML
- Using Dimensions in CML
- Specifying Files in CML

Case Sensitivity is Critical in CML

XML is case-sensitive. CML elements must be entered as title case to be recognized by Post Producer. If you don't enter the element in upper case 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 definition.

Using Dimensions in CML

CML is designed to be rational, practical, and flexible in its expression of values and units of measure You can use the following designators for various types of material:

- Frame/image dimensions may be designated in:
  - ratio—default; no designator
  - percent—using the *%* designator
  - pixels—using the *px* designator
- Audio volume level:
  - ratio—default; no designator
  - percent—using the *%* designator
  - decibels—using the *db* designator
- Rotations:
  - ratio—default; no designator
  - percent—using the *%* designator
  - degrees—using the ° designator
- Text: points—using the *pt* designator

Although it is valid XML to apply any unit designator to any attribute value, not all make sense. Follow these guidelines for the material you're using.

Specifying Files in CML

File references may by specified in any manner appropriate for accessing media on a given platform: Windows drive letters, UNC paths, URLs, etc. In these examples, files are always shown as a UNC path (for example: \\share *path**myvideo.mov*) to reinforce the reliable use of shares to access files in a distributed Vantage domain.

Advisory

CML Elements | CML Hierarchy Map

To create a content advisory, you add an optional *Advisory* element to a *Segment*. The *Advisory* element is a material element which defines the content rating for the segment. The content advisory metadata may be encoded into the video for use by the V-chip in the television or set top box. The optional advisory graphic must be placed and scaled on each segment where you want it to display. (Typically, the first segment displays a larger advisory bug than the others.)

The *Advisory* element sets the V-chip encoding value for the entire composition, unless additional *Advisory* elements are added in later segments. Each succeeding *Advisory* sets the V-chip encoding for the remainder of the composition.

Note: Material items may logically be divided into three temporal parts: the beginning (*Head*), the middle (*Body*), and the end (*Tail*) for the purpose of changing the presentation of the material with the effect elements (*Volume*, *Scaling*, *Translation*, *Rotation*, *Fade*), and applying a *Fade* to define the temporal length of the part.

The *Advisory* element adds two types of information:

- Metadata that identifies the rating system, content rating, and parental advisories. It is typically transmitted in the vertical blanking interval or ANC (using XDS protocol).
- A graphic overlay that displays the content rating. To display the graphic, you must specify the time in the *duration* attribute. The size, position and presentation of the graphic may be controlled using effect elements—*Opacity*, *Scaling*, *Translation*, and *Rotation*.

  USTV only: System-supplied images (the larger set) are displayed, based on the ratings you supply. These may be replaced (or supplied when using other rating systems) using the *location* attribute.

Note: To specify USTV advisory metadata insertion without creating a visual overlay, you omit all attributes except the *type* and *rating* attributes in the *Advisory* element. See *Example With Visual Overlay*, below.
To insert advisory metadata using XDS, you *must enable* the Content Advisory filter in the Transcoder of the Conform action and configure the repeat rate. You can also optionally configure the filter to specify alternate content advisory graphics.

- Child Elements
- Attributes
- Rating Systems Values
- CELR Ratings
- CFLR Ratings
- USTV Ratings

- Enabling the Advisory Filter in Workflow Designer
- Example With Visual Overlay
- Example Without Visual Overlay

Child Elements

One each of these elements may be added to apply effects or alignment to the graphic:

- *Head*
- *Body*
- *Tail*

Attributes

The *adjust*, *align*, and *fill* attributes are closely related. Together, they specify how the visual material is arranged along the timeline in the segment.

Name	Description
adjust (optional)	When the visual material's duration is less than the duration of the segment, the material may be extended within the segment. This provides a flexible method to define the behavior of overlay material so that it is consistently applied, regardless of the duration of the underlying material. For example, using the *adjust* attribute, you can consistently display a branding bug on underlying video of any duration. There are 3 things to consider when modifying material: 1. What part to adjust—the body or the edge (using *adjust*) 2. Which end to adjust from (based on *align* attribute) 3. How to display—hold, loop, none (based on *fill* attribute). Keywords: *edge* (default) | *body* *edge* (default)—The material is adjusted by extending the edge that is *not* identified by the *align* property and is applied to the duration of the head and tail of the material. *body*—The material is adjusted by extending the body of the material. It is applied to the end of the body that is *not* identified by the *align* property. Example: `<Advisory adjust = "body"... />`

Name	Description
align (optional)	Specifies which end (or both) of the visual material to anchor to the timeline of the segment. Keywords: *head* | *tail* | *both*. Example: `<Advisory align = "tail"... />` *head* (default)—Anchors the beginning of the material to the beginning of the segment. *tail*—Anchors the end of the material to the end of the segment. *both*—Anchors the center of the material at the center of the segment, so that the Head and Tail are equi-distant from the beginning and end of the segment. **Note**: If an *offset* is used in this situation, it is also applied to both ends.
customtype (optional)	String which specifies the rating system being referenced. Example: `<Advisory customtype = "MyRatingSystem"... />` Used when the Content Advisory filter is enabled and a custom ratings graphic is being employed. This string is concatenated into the fully-qualified path and filename of the graphic.
customrating (optional)	String which specifies the name of the rating overlay file indicating the rating. Example: `<Advisory customrating = "Great.png"... />` Used when the Content Advisory filter is enabled and a custom ratings graphic is being employed. This string is concatenated into the fully-qualified path and filename of the graphic.

Name	**Description**

duration (optional)	The amount of time to display an *Advisory* image. Default: 0.
	Do not include a duration or layer attribute if you do not want to display a visual overlay.
	If you are specifying a relative timecode for video with a timecode track, use these formats:
	HH:MM:SS:FF@RR | HH:MM:SS:FF | HH:MM:SS;FF
	If you are specifying an absolute timecode or time value, you can also use these formats if the material has a frame rate; otherwise you must use one that can be converted to a time:
	HH:MM:SS.sss | HH:MM:SS:FF@RR
	Time is measured on a 24-hour clock. Time may include milliseconds (*sss*); frames (:|;FF) as DF (;) or NDF (:), and optional frame rate (FF@FPS), as an integer, decimal number, or rational (for example, 30000/1001). If FPS is not specified, it is inferred from the associated source.
	Rates supported: 23.976, 24, 25, 29.97, 30, 50, 59.94.
	Timecode references are applied to the timecode track specified in the associated Source element's file.
	When using DF timecode, the first 2 (29.97 fps) or 4 (59.94 fps) timecode numbers are dropped every minute except minutes divisible by 10. DF timecode is supported for frame rates of 29.97 and 59.94 frames per second.
	Examples:
	`<Advisory duration = "00:00:05.500"... />`
	`<Advisory duration = "00:00:05;14@29.97"... />`

fill (optional)	When the duration of the visual material is less than the length of the segment, the fill attribute determines if and how the item continues to play: Keywords: *none* | *hold* | *loop* Example: `<Advisory fill = "loop"... />` There are 3 things to consider when modifying the material: 1. What to adjust—the body or the edge (using *adjust*) 2. Which end to adjust from (based on *align* attribute) 3. How to adjust—hold, loop, none (based on *fill* attribute) *none* (default)—extend with empty media (no visual or auditory material). From a practical perspective, no extension. *hold*—continue playing the first or last frame the entire duration of the segment. If *align* = *"head"*, the last frame is repeated. If *align* = *"tail"*, the first frame is repeated. *loop*—play media repeatedly for the *duration* of the segment. See *duration*, above.

Name	**Description**
layer (optional)	An integer value describing the ordinal position in the composite of all visual material in this segment. Minimum value: 0, default: 0. Do not include a duration or layer attribute if you do not want to display an overlay. Example: `<Advisory layer = "10"... />` Material with a higher layer value obscures material with a lower layer value in the same segment.
location (optional)	Specifies a fully-qualified path to the advisory image, including file name. The location may be specified as: • *Windows*—D:\pathname\filename.ext • *UNC*—\\share\path\filename.ext • *URL*—http://SierraStudio.com/media/Vane_logo.png • *URL*—s3://aws/bucket/folder/moms_on_strike.mpg"/> CML with URL paths must be processed by a Compose action to localize the file and produce a copy of the CML to process in Post Producer transcoding actions. Example: `<Advisory location = "\\Pluto\media\RatingSystems\USTV\PG13.png" />` The location must be accessible to the Vantage Playlist service which is executing the Conform action. For USTV ratings, a default image is supplied, based on the rating(s) you specify. For all other rating systems (or to override the USTV rating system graphics), supply the fully-qualified path to the image. Supported formats: PNG|JPEG|BMP|TGA|PSD|TIF.

Name	Description
offset (optional)	Specifies a time value which changes the location of the beginning or end of the material, relative to the segment. The offset is applied to the end of the content that is specified by the align property. If `align = "both"` the offset is applied at the defined value to both the head and tail of the material. Default: 0. If you are specifying a relative timecode for video with a timecode track, use these formats: HH:MM:SS:FF@RR | HH:MM:SS:FF | HH:MM:SS;FF If you are specifying an absolute timecode or time value, you can also use these formats if the material has a frame rate; otherwise you must use one that can be converted to a time: HH:MM:SS.sss | HH:MM:SS:FF@RR Time is measured on a 24-hour clock. Time may include milliseconds (*sss*); frames (:|;FF) as DF (;) or NDF (:), and optional frame rate (FF@FPS), as an integer, decimal number, or rational (for example, 30000/1001). If FPS is not specified, it is inferred from the associated source. Rates supported: 23.976, 24, 25, 29.97, 30, 50, 59.94. Timecode references are applied to the timecode track specified in the associated Source element's file. When using DF timecode, the first 2 (29.97 fps) or 4 (59.94 fps) timecode numbers are dropped every minute except minutes divisible by 10. DF timecode is supported for frame rates of 29.97 and 59.94 frames per second. Example: `<Advisory duration = "00:00:05.000"... />`
rating (required)	Keyword, specifies the rating, plus the reason or reasons. Used to insert content rating in XDS. One keyword from the *Ratings Descriptors* table. Default: Not Rated—specify as an empty string. Optionally, add one or more rating reasons from the *Rating Reasons* table, separated by spaces. Example: `<Advisory type = "USTV" rating = "PG-13 V" />` You may specify multiple ratings and reasons. If the combination is not logical, no graphic file is displayed, even if the element is configured to do so.
type (required)	Keyword, specifies the rating body. Used to insert content rating in XDS. One keyword from the *Rating Systems* table. Default: MPAA. Example: `<Advisory type = "USTV" rating = "TV-Y7 FV" />`

Rating Systems Values

These tables provides keywords for various types of metrics related to rating systems and their values.

Rating Systems

This table describes the string value to use in the *type* attribute to identify the rating system you are using.

Keyword	Rating System
CELR	Canada English Language Rating
CFLR	Canada French Language Rating
MPAA	Motion Picture Association of America
USTV	U.S. TV Parental Guidelines

Note: Graphics are supplied for USTV by Advisory. Graphics for other rating systems must be supplied by the content producer, and must be applied using Image.

Ratings Descriptors

This table describes the rating titles from various rating systems and their keyword, which you supply in the *rating* attribute.

Rating Keyword	Rating Title
"" (Empty String)	Not rated
G	General audiences (MPAA, CELR, CFLR)
PG	Parental guidance suggested (MPAA, CELR)
PG-13	Parental strongly cautioned (MPAA)
R	Restricted (MPAA)
NC-17	No one under 17 admitted (MPAA)
X	Explicit (MPAA)
TV-Y	Directed to all children (USTV)
TV-Y7	Directed to children age 7 and up (USTV)
TV-G	General audiences (USTV)
TV-PG	Parental guidance suggested (USTV)

Rating Keyword	Rating Title
TV-14	Parent strongly cautioned (USTV)
TV-MA	Mature audience only (USTV)

Rating Reasons

This table describes the rating reasons (which may only apply to one or more ratings). You may use one, or multiple keywords from this list with the Rating value to form a logical rating. For example, *rating = "PG-13 L V"*.

Keyword	Rating Description
NR	Not rated
FV	Fantasy violence (USTV)
V	Violence (USTV)
L	Adult language (USTV)
S	Sexual situations (USTV)
D	Suggestive dialog (USTV)
E	Exempt (CELR, CFLR)
C	Children (CELR)
C8+	Children age 8 and up (CELR)
8+	Age 8 and up (CFLR)
13+	Age 13 and up (CFLR)
14+	Age 14 and up (CFLR)
16+	Age 16 and up (CFLR)
18+	Age 18 and up (CFLR)

CELR Ratings

E | C | C8+ | G | 14+ | 18+

CFLR Ratings

E | G | 8+ | 13+ | 16+ | 18+

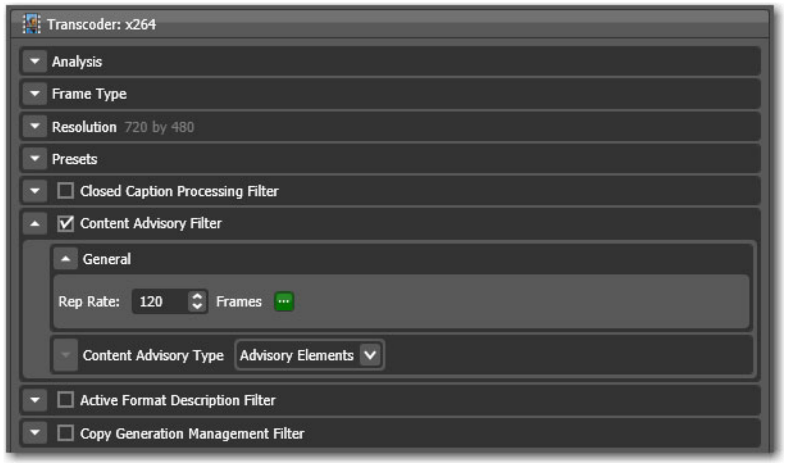
USTV Ratings

TV-G	TV-PG D	TV-PG V S D	TV-14 V S	TV-14 V L S D
TV-Y	TV-PG V L	TV-PG L S D	TV-14 V D	TV-MA
TV-Y7	TV-PG V S	TV-PG V L S D	TV-14 L S	TV-MA V
TV-Y7 FV	TV-PG V D	TV-14	TV-14 L D	TV-MA L
TV-PG	TV-PG L S	TV-14 V	TV-14 S D	TV-MA S
TV-PG V	TV-PG L D	TV-14 L	TV-14 V L S	TV-MA V L
TV-PG L	TV-PG S D	TV-14 S	TV-14 V L D	TV-MA V S
TV-PG S	TV-PG V L S	TV-14 D	TV-14 V S D	TV-MA L S
	TV-PG V L D	TV-14 V L	TV-14 L S D	TV-MA V L S

Enabling the Advisory Filter in Workflow Designer

To display the Content Advisory Filter editor panel, open the Conform action's inspector. Add a video transcoder if you haven't already done so, and click on the transcoder panel to display its controls on the right.

Configure the Advisory Filter to Implement a Parental Advisory:


Transcoder: x264

Analysis

Frame Type

Resolution 720 by 480

Presets

☐ Closed Caption Processing Filter

☑ Content Advisory Filter

General

Rep Rate: 120 Frames

Content Advisory Type: Advisory Elements

☐ Active Format Description Filter

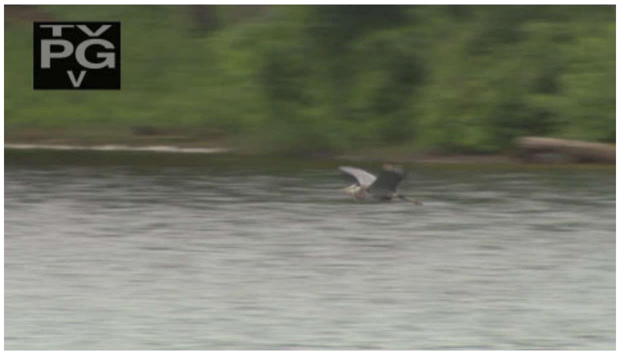
☐ Copy Generation Management Filter

Click to open the Content Advisory Filter editor panel and set the repeat rate, as shown.

Note: For a practical example of implementing an advisory, see Adding a Content Advisory in the Post Producer Cookbook.

Example With Visual Overlay

This example illustrates how to implement a typical advisory graphic in a segment, using the *Advisory* element, as shown here:



In the code below, the USTV advisory graphic has a head, body, and tail, so that a half-second fade can be applied, and the graphic positioned 5 percent down and to the right of the origin of the frame, with a total duration of 15 seconds.

Also note that the volume was set to 0 in the tail so that the Fade would also modify the audio volume.

Composition

```
...
  </Segment>
...
    <Advisory align = "head" adjust = "edge" layer = "1" type =
"USTV" rating = "TV-PG V" duration = "00:00:15.00" >
      <Head>
        <Translation x = "5%" y = "5%" />
        <Opacity level = "0%"/>
        <Fade duration = "00:00:00.500" />
      </Head>

      <Body>
        <Opacity level = "100%" />
        <Translation x = "5%" y = "5%" />
      </Body>
      <Tail>
        <Translation x = "5%" y = "5%" />
        <Opacity level = "0%"/>
        <Fade duration = "00:00:00.500"/>
        <Volume level = "0%"/>
      </Tail>
    </Advisory>
  </Segment>...
```

Example Without Visual Overlay

This example illustrates how to implement a Parental Advisory without displaying any graphic overlay, using the *Advisory* element.

In the code below, the *Advisory* element only has a type and rating. The *align*, *adjust*, *layer*, and *duration* attributes have been removed—these all affect the visual image. *Head*, *Body*, and *Tail*, likewise.

Composition

```
...
  </Segment>
...
    <Advisory type = "USTV" rating = "TV-PG V" >
    </Advisory>
  </Segment>...
```

Area

CML Elements | CML Hierarchy Map

One *Area* element should be added to a *Title* to define the location and size of the rectangle in which to render the title text. Without it, the default area is the same as the output frame, and the title text is rendered in the vertical center of the frame, on the left edge. Though the *Area* element is optional, in most cases an *Area* element is a practical necessity.

You can not define an area off-screen (outside the output frame). To place it offscreen for a slide, add a *Head*, *Body*, and *Tail* and use *Translation* to animate it.

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

Note: For a practical example of using titles with an Area, see Adding Titles in the Post Producer Cookbook.

See also *Title*, *Shadow*.

Attributes

These attributes are relative to origin of output frame at 0,0—the upper, left corner.

The effect of attributes is easier to predict by also locating the title text in the area at the top left corner using the vertical-align and horizontal-align settings.

You can specify a unit designator of pixels (px) or a ratio (no designator) or percent (%). For example, specifying *bottom=90%* places the bottom edge 10% above the bottom of the frame.

Note: If you use pixels, be aware that the area definition is directly correlated to the output frame—which is defined in the Conform action—beyond the scope of the Composition. Thus, if you design for HD output, but submit the composition to an SD Conform workflow, the area won't be located correctly. Use percent or ratio to specify the definition relative to the output frame regardless of its size.

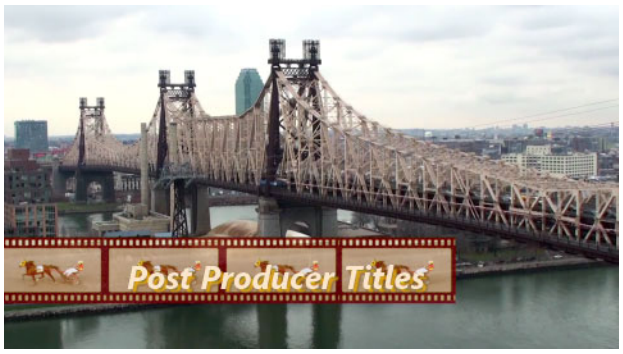
If you specify a value without a unit designator, it is treated as a ratio. (The only functional difference between a ratio and a percent is the location of the decimal point.) The ratio may be expressed as an integer, a decimal number, or a rational number {5/10} and must be presented as an expression, in braces. The ratio is the relationship of the edge to the output frame in the appropriate dimension.

Positive (on-screen) and negative (off-screen) values may be used to specify the area.

Name	Description
bottom (optional)	Specifies the distance from the top of the frame to the bottom edge of the area, as pixels (px), percent (%), or ratio (no unit designator). Default: 1. (The bottom of the output frame.) Example: `<Area bottom = "480px"... />` specifies the bottom edge of the area as 480 pixels below the top edge of the output frame.
left (optional)	Specifies the distance from the left edge of the frame to the left edge of the area, as pixels (px), percent (%), or ratio (no unit designator). Default: 0. (The left edge of the frame.) Example: `<Area left = "33.3%"... />` specifies the left edge of the area as 1/3 of the way across the frame from the left edge of the output frame.
right (optional)	Specifies the distance from the left edge of the frame to the right edge of the area, as pixels (px), percent (%), or ratio (no unit designator). Default: 1. (The right edge o f the frame.) Example: `<Area right = "100%"... />` specifies the right edge of the area identical to the right edge of the output frame.
top (optional)	Specifies the distance from the top of the frame to the top edge of the area, as pixels (px), percent (%), or ratio (no unit designator). Default: 0. (The top of the frame.) Example: `<Area top = "0"... />` specifies the top edge of the area identical to the top edge of the output frame.

Example

In this example, an Area is added to the *Title* element, and specified as a rectangle positioned directly over the race frames, in which to display the title text, as shown in the figure below.



Note that in this code snippet, the *Area* attribute values are specified as a percentage of the frame size, not in pixels.

Composition

```
...
<Title align = "head" adjust = "edge" fill = "none" layer = "2"
duration = "00:00:04.00" offset = "00:00:01.000" font =
"Helvetica" size = "50pt" style = "italic" weight = "bold"
    foreground-color = "bisque" background-color = "transparent"
wrap = "false" horizontal-align = "left" vertical-align = "middle"
layout = "stretch">
  <Area top = "70%" left = "20%" bottom = "90%" right = "80%" />
  <Shadow color = "gold" softness = "10%" vertical-offset = "6%"
horizontal-offset = "6%" />
  Post Producer Titles
</Title>
...
```

Audio

CML Elements | CML Hierarchy Map

You can add one or more *Audio* elements to a *Segment*. The *Audio* element is a material element, which adds audio material from the file identified by a *Source*.

Note: Material items may logically be divided into three temporal parts: the beginning (*Head*), the middle (*Body*), and the end (*Tail*) for the purpose of changing the presentation of the material with the effect elements (*Volume*, *Scaling*, *Translation*, *Rotation*, *Fade*), and applying a *Fade* to define the temporal length of the part.

Child Elements

One each of these elements may be added to apply effects or alignment:

- *Head*
- *Body*
- *Tail*

Attributes

Name	Description
adjust (optional)	When the material's duration is less than the duration of the segment, the material may be extended within the segment. This provides a flexible method to define the behavior of overlay material so that it is consistently applied, regardless of the duration of the underlying material. For example, using the *adjust* attribute, you can consistently display a branding bug on underlying video of any duration. There are 3 things to consider when modifying material: 1. What part to adjust—the body or the edge (using *adjust*) 2. Which end to adjust from (based on *align* attribute) 3. How to display—hold, loop, none (based on *fill* attribute). Keywords: *edge* (default) | *body* *edge* (default)—The material is adjusted by extending the edge that is *not* identified by the *align* property and is applied to the duration of the head and tail of the material. *body*—The material is adjusted by extending the body of the material. It is applied to the end of the body that is *not* identified by the *align* property. Example: `<Audio adjust = "body"... />`

Name	**Description**
align (optional)	Specifies which end (or both) of the material to anchor to the timeline of the segment. Keywords: *head* | *tail* | *both.* Example: `<Audio align = "tail"... />` *head* (default)—Anchors the beginning of the material to the beginning of the segment. *tail*—Anchors the end of the material to the end of the segment. *both*—Anchors the center of the material at the center of the segment, so that the *Head* and *Tail* are equi-distant from the beginning and end of the segment. **Note**: If an offset is used in this situation, it is also applied to both ends.
fill (optional)	When the duration of the material is less than the length of the segment, the fill attribute determines if and how the item continues to play: Keywords: *none* | *hold* | *loop* Example: `<Audio fill = "loop"... />` *none* (default)—extend with empty media (no visual or auditory material). *hold*—continue playing the first or last frame to play the material the entire duration of the segment. If *align* = *"head"*, the last frame is repeated. If *align* = *"tail"*, the first frame is repeated. *loop*—play the media repeatedly for the duration of the segment.

Name	Description
offset (optional)	The amount of time to wait from the beginning of the segment to play the material. Default: 0. If you are specifying a relative timecode for video with a timecode track, use these formats: HH:MM:SS:FF@RR | HH:MM:SS:FF | HH:MM:SS;FF If you are specifying an absolute timecode or a timestamp value, you can also use these formats if the material has a frame rate; otherwise you must use one that can be converted to a time: HH:MM:SS.sss | HH:MM:SS:FF@RR Time is measured on a 24-hour clock. Time may include milliseconds (*sss*); frames (:|;FF) as DF (;) or NDF (:), and optional frame rate (FF@FPS), as an integer, decimal number, or rational (for example, 30000/1001). If FPS is not specified, it is inferred from the associated source. Rates supported: 23.976, 24, 25, 29.97, 30, 50, 59.94. Timecode references are applied to the timecode track specified in the associated Source element's file. When using DF timecode, the first 2 (29.97 fps) or 4 (59.94 fps) timecode numbers are dropped every minute except minutes divisible by 10. DF timecode is supported for frame rates of 29.97 and 59.94 frames per second. Example: `<Audio offset = "00:00:05.000"... />`
source (required)	Integer value corresponding to the Source element's *identifier* value, to identify the file to be used in this instance. Example: `<Audio source = "6"... />`

Example

This represents a typical *Audio* element in a segment. In this code snippet—from the Promo composition—an audio source file (`6C_Audio_Overlay.mov`) is specified as Source 6, and the *Audio* element references it (`source = "6"`), to play the audio during the segment in which it is located.

Composition

```
...
  <Source identifier = "6">
    <File location = "\\share\path\6C_Audio_Overlay.mov" />
  </Source>
...
  <Sequence layer="1">...
    <Segment>
...
      <Audio align = "head" adjust = "edge" fill = "none" source = 
"6" layer = "0">
      </Audio>
```

Body

CML Elements | CML Hierarchy Map

The *Body* element is the temporal portion (duration) of a material item identified by the beginning of the material (or the head, if defined, and the end of the material (or the tail, if defined).

The *Body* (like the *Head* and the *Tail* may contain effects elements (opacity, scaling translation, rotation, and volume) that control the audio and video composition. Post Producer interpolates between the values set in the body and those in the *Head* and *Tail* to create transition and video effects.

There are no attributes in a *Body* element.

See *Working with Material* for a description of the relationship between *Head*, *Body*, and *Tail* parameters and timing.

See also *Head*, *Tail*.

Child Elements

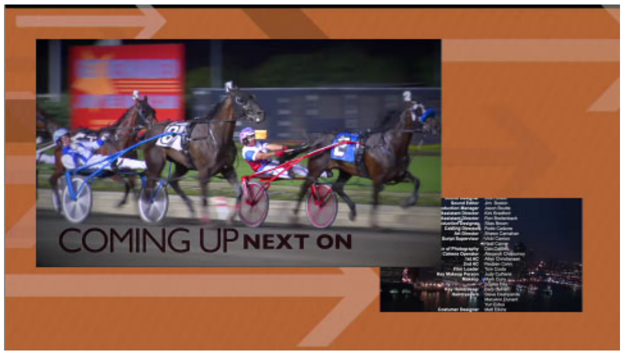
One each of these effects elements may be optionally added to a *Head*, *Body*, or *Tail* element:

- *Opacity*
- *Scaling*
- *Translation*
- *Rotation*
- *Volume*

Example

In this example, a video clip is superimposed on a background to highlight the upcoming show. A sample frame from an HD input source is shown here for reference.

The inset video fades in and out (using *Head* and *Tail*), is positioned at 5% over and 10% down on the frame, and displays on 66% of the frame. The *Body* element for this video merely continues the positioning and size of the inset for the duration. *Body* elements often continue positioning and size for the duration.



Composition

```
...
<Video align = "head" adjust = "body" fill = "hold" source = "5"
layer = "2">
  <!--               HD Promo                      -->
  <!--   hold the last frame until summary done.  -->
  <!--                                             -->
  <Head>
    <Edit time = "00:00:00.000" />
    <Scaling x = "66%" y = "66%" />
    <Translation x = "5%" y = "10%" />
  </Head>
  <Body>
    <Scaling x = "66%" y = "66%" />
    <Translation x = "5%" y = "10%" />
  </Body>
  <Tail>
    <Edit time = "01:00:26:23" />
    <Opacity level = "0"/>
    <Scaling x = "66%" y = "66%" />
    <Translation x = "5%" y = "10%" />
    <Fade duration = "00:00:05.000"/>
  </Tail>
</Video>
...
```

Burn-in

CML Elements | CML Hierarchy Map

The *Burn-in* element is used to render external text strings onto frames of the conformed output media. *Burn-in* can be used to render timecode or text from caption files onto the video.

When the *source* attribute is set to the keyword *captions,* the *Burn-in* element is used to process a captions file and render the captions into the conformed output media. When set to *timecode*, *Burn-in* renders the source file's timecode on the output frames.

When rendering captions, the *Burn-in* element uses the specified SCC or STL file identified as a Source in the CML (identified by a File element and connected via the Source identifier and Video source attribute pairs). Burn-in follows caption placement per the captions file, along with EIA 608 values for size, color, and font.

Child Elements

The *Translation* element may be added to a *Burn-in* element to animate the text over time.

Attributes

Name	Description
background-color (optional)	Specifies the background color for the caption text region that is painted. SCC captions default: *black*. STL captions default: *transparent*. Keywords: *transparent* | *<supported color keywords>* | *ARGB* The supported colors are defined in .NET. For details, see Color Structure. *ARGB*—The RGB + Alpha color, in #AARRGGBB format, where the high-order byte (AA) contains the alpha channel and the remaining bytes contain color components for red, green and blue, respectively. The alpha channel specifies the opacity of the pixel, where a value of 00 represents a pixel that is fully transparent and a value of FF is fully opaque. For example, #00000000 creates a totally transparent background, and #77000000" is a translucent black.
font (optional)	The family name of the font, which must be present on the system where the Vantage Edit service executing the Conform action is hosted. Default: Helvetica. For details, see .NET Font Family. For composition file encoding requirements for non-Roman fonts, see *Composition File Requirements*.

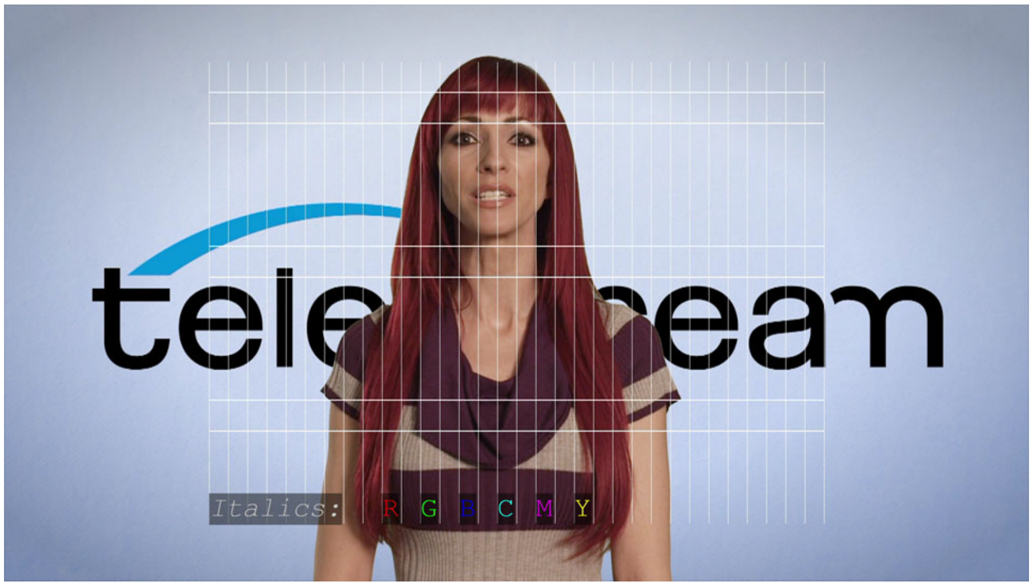
Name	Description
foreground-color (optional)	Specifies the text color for the caption text that is painted. Default: *black*. Keywords: *<supported color keywords>* | *ARGB* The supported colors are defined in .NET. For details, see Color Structure. *ARGB*—The RGB + Alpha color, in #AARRGGBB format, where the high-order byte (AA) contains the alpha channel and the remaining bytes contain color components for red, green and blue, respectively. The alpha channel specifies the opacity of the pixel, where a value of 00 represents a pixel that is fully transparent and a value of FF is fully opaque.
size (optional)	Applies only to Burn-in. An integer value describing the nominal height of the font in points (pt). Default: 36 points. Example: `<Burn-in size = "72pt"... />`
source (required)	Keyword value specifying the type of source. Keywords: *captions* | *timecode* Example: `<Burn-in source = "timecode"... />`
string-background-color (optional)	Specifies the background color for each caption character that is painted. Default: *black*. Keywords: *transparent* | *<supported color keywords>* | *ARGB* The supported colors are defined in .NET. For details, see Color Structure. *ARGB*—The RGB + Alpha color, in #AARRGGBB format, where the high-order byte (AA) contains the alpha channel and the remaining bytes contain color components for red, green and blue, respectively. The alpha channel specifies the opacity of the pixel, where a value of 00 represents a pixel that is fully transparent and a value of FF is fully opaque.
style (optional)	Specifies the style in which the font face is rendered. Default: *normal*. Keywords: *normal* | *italic* | *underline* | *strikeout* | *overline* Example: `<Burn-in style = "italic"... />`
weight (optional)	Specifies the thickness of the stroke of each character. Default: *normal*. Keywords: *normal* | *bold* | *light* | *medium* | *heavy* Example: `<Burn-in weight = "medium"... />`

Examples

In this example, an SCC file of sample caption text is burned into a media file, using this CML snippet:

```
<Video source = "1">
  <Burn-in source = "captions" background-color = "#00000000"
                string-background-color = "#77000000" />
</Video>
```

Here is a sample frame of burned-in caption text.



In this example, the caption's location/style/color is determined by the 608 data. Optionally, the caption font, size, style, weight, and foreground color, background color and string background color can be set from an attribute for the Burn-in element.

Composition

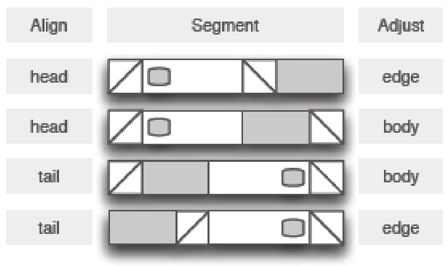
```
<Composition xmlns = "Telestream.Soa.Facility.Playlist">
  <Source identifier = "1">
    <Subtitle>
      <File location = "\\share\spanish.scc" />
    </Subtitle>
    <File location = "\\share\CC_CHI002_2398.mov" />
  </Source>
  <Sequence>
    <Segment>
      <Video>
        <Burn-in source = "captions" background-color = "green"
                     string-background-color = "green" />
      </Video>
    </Segment>
  </Sequence>
</Composition>
```

Canvas

CML Elements | CML Hierarchy Map

The optional *Canvas* element adds a visual rectangle at the same size as the output frame, to a *Segment*. The *Canvas* element is a material element, which is intended as a filler, and is rendered in black. It is used as content fill or as a background layer when fading in and out or dipping to black.

Align and *Adjust* attributes work together on material expansion. For a visual representation of the effect of adjust and align attributes when using a Fade element, see the illustration below.



There are no child elements in *Canvas*.

Attributes

Name	**Description**
adjust (optional)	When the material's duration is less than the duration of the segment, the material may be extended within the segment. This provides a flexible method to define the behavior of overlay material so that it is consistently applied, regardless of the duration of the underlying material. For example, using the *adjust* attribute, you can consistently display a branding bug on underlying video of any duration. There are 3 things to consider when modifying material: 1. What part to adjust—the body or the edge (using *adjust*) 2. Which end to adjust from (based on *align* attribute) 3. How to display—hold, loop, none (based on *fill* attribute). Keywords: *edge* (default) | *body* *edge* (default)—The material is adjusted by extending the edge that is *not* identified by the *align* property and is applied to the duration of the material's head and tail. *body*—The material is adjusted by extending the body of the material. It is applied to the end of the body that is *not* identified by the *align* property. Example: `<Canvas adjust = "body"... />`

Name	Description
align (optional)	Specifies which end (or both) of the material to anchor to the timeline of the segment. Keywords: *head* | *tail* | *both*. Example: `<Canvas align = "tail"... />` *head* (default)—Anchors the beginning of the material to the beginning of the segment. *tail*—Anchors the end of the material to the end of the segment. *both*—Anchors the center of the material at the center of the segment, so that the *Head* and *Tail* are equi-distant from the beginning and end of the segment. **Note**: If an *offset* is used in this situation, it is also applied to both ends.
Background (optional) **Note:** The Background attribute has an upper case B. Lower case instances are incorrect and ignored.	Specifies the color that the area is painted. Default: *black*. Keywords: *transparent* | *black* | *white* | *ARGB* The supported colors are defined in .NET. For details, see Color Structure. *ARGB*—The RGB + Alpha color, in 0xAARRGGBB format, where the high-order byte (AA) contains the alpha channel and the remaining bytes contain color components for red, green and blue, respectively. The alpha channel specifies the opacity of the pixel, where a value of 0x00 represents a pixel that is fully transparent and a value of 0xFF is fully opaque.

Name	Description

duration (optional)	The amount of time to display the canvas. Default: 0.
	If you are specifying a relative timecode for video with a timecode track, use these formats:
	HH:MM:SS:FF@RR | HH:MM:SS:FF | HH:MM:SS;FF
	If you are specifying an absolute timecode or timestamp value, you can also use these formats if the material has a framerate; otherwise you must use one that can be converted to a time:
	HH:MM:SS.sss | HH:MM:SS:FF@RR
	Time is measured on a 24-hour clock. Time may include milliseconds (*sss*); frames (:|;FF) as DF (;) or NDF (:), and optional frame rate (FF@FPS), as an integer, decimal number, or rational (for example, 30000/1001). If FPS is not specified, it is inferred from the associated source.
	Rates supported: 23.976, 24, 25, 29.97, 30, 50, 59.94.
	Timecode references are applied to the timecode track specified in the associated Source element's file.
	When using DF timecode, the first 2 (29.97 fps) or 4 (59.94 fps) timecode numbers are dropped every minute except minutes divisible by 10. DF timecode is supported for frame rates of 29.97 and 59.94 frames per second.
	Examples:
	`<Canvas duration = "00:00:05.500"... />`
	`<Canvas duration = "00:00:05;14@29.97"... />`

fill (optional)	When the duration of the material is less than the length of the segment, the fill attribute determines if and how the item continues to play: Keywords: *none* | *hold* | *loop*. Example: `<Canvas fill = "loop"... />` *none* (default)—extend with empty media (no visual or auditory material). *hold*—continue playing the first or last frame to play the material the entire duration of the segment. If *align* = *"head"*, the last frame is repeated. If *align* = *"tail"*, the first frame is repeated. *loop*—play the media repeatedly for the duration of the segment.
layer (optional)	An integer value describing the ordinal position in the composite of all material in this segment. Default: 0. Example: `<Canvas layer = "10"... />` Material with a lower layer value may be obscured by material with a higher layer value in the same segment. When used in *Canvas*, the value may be higher or lower than the video (or other material) you are fading with, because in either case one becomes fully transparent.

Name	Description
offset (optional)	The amount of time to wait from the beginning of the segment, to display the material. Default: 0. If you are specifying a relative timecode for video with a timecode track, use these formats: HH:MM:SS:FF@RR | HH:MM:SS:FF | HH:MM:SS;FF If you are specifying an absolute timecode or timestamp value, you can also use these formats if the material has a framerate; otherwise you must use one that can be converted to a time: HH:MM:SS.sss | HH:MM:SS:FF@RR Time is measured on a 24-hour clock. Time may include milliseconds (*sss*); frames (:|;FF) as DF (;) or NDF (:), and optional frame rate (FF@FPS), as an integer, decimal number, or rational (for example, 30000/1001). If FPS is not specified, it is inferred from the associated source. Rates supported: 23.976, 24, 25, 29.97, 30, 50, 59.94. Timecode references are applied to the timecode track specified in the associated Source element's file. When using DF timecode, the first 2 (29.97 fps) or 4 (59.94 fps) timecode numbers are dropped every minute except minutes divisible by 10. DF timecode is supported for frame rates of 29.97 and 59.94 frames per second. Example: `<Canvas offset = "00:00:05.000"... />`

Example

This example illustrates the typical use of a *Canvas* element in a *Segment* (shown in bold).

Composition

```
...
<Segment>
  <Canvas align = "head" adjust = "body" duration = "00:00:18.700"
Background = "black" layer = "0" />
...
```

Comment

CML Elements | CML Hierarchy Map

The optional *Comment* element can be placed in any element in the composition. The purpose of a *Comment* element is to enable you to pass information (for example, a key-pair value) through a workflow for use downstream such as an as-run chronicle output from Conform or to explain the composition.

Note: The Comment element is not in HTML format, and it is not an XML comment.

There are no child elements or attributes in a *Comment* element. You can place any text in the comment element that is valid XML.

Example

In this example, the comments contain key pair values, which can be passed through to a Chronicle action or extracted elsewhere as necessary.

Composition

```
<Segment>
  <Canvas align = "head" adjust = "body" duration = "00:00:18.700" 
Background = "black" layer = "0" />
     <Comment>TRP ID = urn:uuid:06a4-e204-407b-ac8b </Comment>
    <Comment>TRP Hash = emtXD7kMraxeTojicE6Ofva2HAc</Comment>
    <Comment>TRP Size = 224957607</Comment>
</Segment>
```

Composition

CML Elements | CML Hierarchy Map

The *Composition* element identifies this XML 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

These elements may be added to a *Composition* element:

- *Source* (one or more)
- *Sequence* (one or more)
- *Target* (one, optional).

Attributes

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

Example

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

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

Crop

CML Elements | CML Hierarchy Map

To crop the frames of source video, you add a *Crop* element to the *Source* element that identifies the instance. (The *Crop* element is optional; one may be added per *Source*.) All Video instances that reference a *Source* with a *Crop* element inherit the media as specified—that is, cropped.

The *Crop* element creates a new rectangle of the input frame that is smaller than the original. All pixels that lie outside the crop rectangle are discarded from the source, based on your configured dimensions.

For example, you can use *Crop* to:

- Produce a centercut HD 16:9 to SD 4:3
- Crop 32 lines off the top of IMX video to remove VBI lines
- Make an aspect ratio adjustment.

The crop frame is defined relative to the dimensions of the source frame. Positive values originate from the top, left corner. Negative values originate from the bottom, right corner.

For example, to remove 25% of a frame evenly from left and right, you could specify:

```
<Crop left = "12.5%" right = "-12.5%"... />
```

or

```
<Crop left = "12.5%" right = "87.5%"... />
```

The result is the same.

You could also crop 32 pixels off the top and 2 pixels off the bottom of a 640 x 480 frame in either manner:

```
<Crop top = "32px" bottom = "-2px"... />
```

or

```
<Crop top = "32px" bottom = "478px"... />
```

The dimensions may be specified in pixels, or as a ratio, fraction, or percent.

The cropped input frame is resized (if necessary) to fit the output frame size (or mask, if specified) before encoding. Cropping is frequently used in conjunction with masking. With a mask, you'll end up with pillars or bars, depending on how the mask ratio differs from the frame ratio.

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

See also *Mask*.

Attributes

Each of these attributes may be specified by a units designator: pixels (px), ratio, fraction, or percent (%). If you don't specify a designator, the value is treated as a ratio of the frame dimension. For example, *bottom="-0.222"* is the same as *bottom="-22.2%".*

Positive values originate from the top, left corner. Negative values originate from the bottom, right corner. Values may be integers or decimal numbers.

Name	Description
bottom (optional)	Specifies the amount of material to remove upward from the bottom edge of the input frame. Default: *0*. Example: `<Crop bottom = "-2px"... />`
left (optional)	Specifies the amount of material to remove from the left edge of the input frame. Default: *0*. Example: `<Crop left = "0.125"... />`
right (optional)	Specifies the amount of material to remove from the right edge of the input frame. Default: *0*. Example: `<Crop right = "-12.5%"... />`
top (optional)	Specifies the amount of material to remove from the top edge of the input frame. Default: *0*. Example: `<Crop top = "32px"... />`

Examples

Two examples describe using a *Crop* element.

- Center Cutting an HD Frame with Pillars
- Center Cutting an HD Frame for an SD Frame

Center Cutting an HD Frame with Pillars

In this example, a center cut is applied to 16x9, 1920 by 1080 input to produce 4x3, 1440 by 1080 output by removing 480 pixels from the width of the image. (25% total; 12.5% from each side).

Input HD Frame: 1920x1080

Output HD Frame: 1440x1080

To remove the 480 pixels, you can configure *Crop* by percent, ratio, or by pixels:

Composition

```
<Crop left = "12.5%" right = "-12.5%"/>
<Crop left = "0.125" right = "-0.125"/>
<Crop left = "240px" right = "-240px"/>
```

Center Cutting an HD Frame for an SD Frame

In the 2nd example, you want to center cut a 16 x 9, 1920 x1080 input to down-convert to 4 x 3, 720 x 540 output, also by removing 480 pixels from the width of the image:

- Input HD frame: 1920 x 1080
- Output SD frame: 720 x 540, specified in the encoder.

To remove the 480 pixels, you can configure Crop in the same manner as the first example. In both cases, it is necessary to remove 480 pixels from source, 240 from the left and right of the original frame. In the first case, no scaling is performed on the output after cropping. In the second case, the image is scaled to 1/2 in the X and Y directions, by virtue of the frame size set in the encoder.

Composition

```
<?xml version = "1.0" encoding = "utf-8"?>
<Composition xmlns = "Telestream.Soa.Facility.Playlist">
  <Source identifier = "1">
    <File location = "\\share\path\1920x1080.mov"/>
  <Crop left = "240px" right = "-240px" />
<!-- center cut 4x3 cropping for any resolution output-->
  </Source>
    <Sequence layer = "0">
      <Segment>
        <Video source = "1" >
        <Head>
          <Edit time = "00:00:00.000"/>
        </Head>
        <Tail>
          <Edit time = "00:00:05.000"/>
        </Tail>
      </Video>
    </Segment>
  </Sequence>
</Composition>
```

Summary

Cropping parameters are applied to the input frame and the resulting frame is placed within the output frame, re-rendering to fit, if necessary.

In both cases, it is necessary to remove 480 pixels from source, 240 from the left and right of the original frame. In the first case, no scaling is performed on the output after cropping, in the second case, the image is scaled to 1/2 in the x and y directions.

Therefore, specifying the cropping values in pixels produces exactly the same result in both cases. Crop values specified as a percent are relative to the output resolution. 240 pixels is 16.17% of 1440 and 33.34% of 720.

DolbyE

CML Elements | CML Hierarchy Map

The *DolbyE* element specifies the audio track layout of a source file that contains compressed, Dolby E audio channels to be decoded and remapped or remixed to the output. You add a *DolbyE* element to a *Source* when you want to re-arrange the channels in the output media.

The *Mix* element should be added to the *DolbyE* element to decode and remap the audio channels. The *Route* element is used to perform Dolby E pass through.

Dolby E source must be one of these program layouts for processing in Post Producer:

• 5.1+2	Surround plus one stereo pair
• 2+2+2+2	Four stereo pairs
• 5.1	5.1 Surround
• 7.1	7.1 Surround.

When using Dolby E audio, it is necessary to:

- Silence the WAV pair that contains Dolby E audio. This is accomplished by mixing it to channels 1 and 2 at zero volume. Failure to do this will result in output with static.
- Specify the Dolby E program configuration that is contained in the source.
- Map and mix the desired tracks that will be contained in the output.

DolbyE Program Channel Configurations

In the table below, each Dolby E program is identified in the column on the left. On the right, each channel (from 1 through 8) is identified by its logical segment of the program. (-1 indicates the channel is unused in this program configuration.)

For example, if your input is a 4+2+2 Dolby E program, then channels 1,2,5, and 6 comprise the 4 segment; channels 3 and 7 comprise the first 2 segment, and channels 4 and 8 comprise the second or last 2 segment.

Program	Index by Channel
5.1 + 2	0, 0, 0, 1, 0, 0, 0, 1
5.1	0, 0, 0, 0, 0, 0,-1,-1
7.1	0, 0, 0, 0, 0, 0, 0, 0

Child Elements

One or more *Mix* or *Route* elements may be added to a *DolbyE* element.

Attributes

Name	Description
channels (required)	Specifies the PCM WAV stereo pair that contains Dolby E audio, in the form N N. The two values should be separated with a space. Keywords: *None* | *1... 32* | *All* Example: `<DolbyE channels = "1 2"... />` *None*—use to specify no channels. *1...32*—The channel pair, separated by spaces. *All*—use to specify all channels.
program (required)	Specifies the channel layout of the compressed Dolby E audio, by keyword. Keywords: *5.1+2* | *4+2+2* | *5.1* | *7.1* Example: `<DolbyE program = "5.1"... />`

Example

This example illustrates mapping a 5.1 Dolby E stereo pair source to a stereo pair in the output file.

Composition

```
<Composition xmlns = "Telestream.Soa.Facility.Playlist">
  <Source identifier = "1">
    <File location = "\\share\Dolby_E\dolbye_audio.GXF" />
    <!-- Mute track 1 and 2 in prep for decoding -->
    <Mix source = "1 2" target = "1 2" level = "0" />
    <DolbyE channels = "3 4" program = "5.1">
      <!-- Mix Left (1) > Stereo Left (1) -->
      <Mix source = "1" target = "1" />
      <!-- Mix Right (4) to Stereo Right (2) -->
      <Mix source = "4" target = "2" />
      <!-- Mix Center (2) to Stereo Left & Right (2) at -3dB -->
      <Mix source = "2" target = "1 2" level = "0.707107" />
      <!-- Mix Left Rear (3) to Stereo Left (1) at -3dB -->
      <Mix source = "3" target = "1" level = "0.707107" />
      <!-- Mix Right Rear (6) to Stereo Right (2) at -3dB -->
      <Mix source = "6" target = "2" level = "0.707107" />
    </DolbyE>
  </Source>
  <Sequence>
    <Segment>
      <Video source = "1" />
    </Segment>
  </Sequence>
</Composition>
```

Edit

CML Elements | CML Hierarchy Map

The optional *Edit* element is used in the *Head* and the *Tail* of material: *Advisory*, *Image*, *Title*, *Canvas*, or *Media*. An *Edit* element creates a clip from a longer source—by specifying a mark-in and/or mark-out point—for the new 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 ending of the clip from its default position (the beginning or ending of the file) to the point specified.

In any given instance, 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 after the specified time in the *Head Edit* is *included* in the clip. However, the first frame after the specified time in the *Tail Edit* is *excluded*.

Post Producer supports back-timing edits from either end. For example, you may want to roll movie credits in a What's Next promo. So, you align the video at the tail (`align = "tail"`), so that it ends at the same time as the segment. Next, create an *Edit* in the *Head* and place a mark-in at the credits cut using `mode = "duration"`, and set the time equal to the length of the credit roll.

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

Note: For a detailed explanation of using *Edit* to create clips from a master file, see *Extracting Clips from Master Media*.

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.

Attribute Name	**Description**
mode (required)	Specifies how time or timecode (*absolute* and *relative* only) is applied to a *Head* or *Tail*. Keywords: *absolute* | *relative* | *duration*. *absolute* (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. *relative*—specifies an edit point measured from the beginning (in a Head) or ending (in a *Tail*) of the source, irrespective of the timecode (if present). *duration*— specifies an edit point measured as a length of time, from the end (in a *Head*) or the beginning (in a *Tail*) of the source. **Note:** You can't use *duration* in both *Head* and *Tail* in the same instance of material. Example: `<Edit mode = "relative" time = "00:10:00" />`

time (optional)	Specifies (by time or timecode) the location of the edit point. Drop frame, non-drop frame and time references are valid.
	The time element is only valid when mode is set to *relative*. It is not valid when mode is set to *absolute* or *duration*.
	Examples:
	`<Edit mode = "relative" time = "01:00:10" />`
	`<Edit mode = "relative" time = "01:27;10" />`
	`<Edit mode = "relative" time = "00:00:10.000" />`
	`<Edit mode = "relative" time = "01:00:00;03@29.97" />`

Examples

This example illustrates trimming the *Video Tail* to a duration of 7 seconds.

Composition

```
...
<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

CML Elements | CML Hierarchy Map

The optional *Fade* element adds a duration, and smoothly interpolates between the effects (*Opacity*, *Scaling*, *Transition*, *Rotation*, *Volume*) elements specified for the *Head* and *Tail* elements and those in the *Body*. The interpolation occurs over the specified duration.

It is common to think of the term *fade* as diminishing video or audio material; in the context of CML, the term is used generally, to increase or decrease *any effect*, blending it gradually and harmoniously with the properties of the *Body*, based on the duration and shape specified.

The Fade element may be used in the *Head* and *Tail*.

If you are using *Fade* for effects other than *Volume* or *Opacity* (that is, you do not want to fade these), you must supply a volume or opacity level explicitly, because in *Head* and *Tail*, the default level for both is 0%.

Note: The duration of a fade (or group of fades) can not be longer than the duration of the element to which the fade is being applied. For example, if you fade a Head and Tail 4 seconds each on a 6-second image or clip, it will fail.

There are no child elements in a Fade element.

See also *Opacity*, *Scaling*, *Translation*, *Rotation*, *Volume*.

Attributes

Name	Description

duration (optional)	The amount of time to perform the fade. Default: 0.
	If you are specifying a relative timecode for video with a timecode track, use these formats:
	HH:MM:SS:FF@RR | HH:MM:SS:FF | HH:MM:SS;FF
	If you are specifying an absolute timecode or time value, you can also use these formats if the material has a framerate; otherwise you must use one that can be converted to a time:
	HH:MM:SS.sss | HH:MM:SS:FF@RR
	Time is measured on a 24-hour clock. Time may include milliseconds (*sss*); frames (:|;FF) as DF (;) or NDF (:), and optional frame rate (FF@FPS), as an integer, decimal number, or rational (for example, 30000/1001). If FPS is not specified, it is inferred from the associated source.
	Rates supported: 23.976, 24, 25, 29.97, 30, 50, 59.94.
	Timecode references are applied to the timecode track specified in the associated Source element's file.
	When using DF timecode, the first 2 (29.97 fps) or 4 (59.94 fps) timecode numbers are dropped every minute except minutes divisible by 10. DF timecode is supported for frame rates of 29.97 and 59.94 frames per second.
	Examples:
	`<Fade duration = "00:00:05.500"... />`
	`<Fade duration = "00:00:05;14@29.97"... />`

motion-blur (optional)	A real or integer value (range 0-10), indicating the number of intermediate steps between each frame used to generate the blur. The faster you move material using the visual effects elements (*Rotate*, *Translate*, *Scale*, *Opacity*), the larger the number should be, to smooth out DVE motion effects. For instance, if the image moves 20 pixels per frame, and you set a value of 10 (the maximum) then the blur will be generated by averaging 10 frames, each 2 pixels apart. **Note**: The higher the value, the longer transcoding will take.

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

Example

In this example, the *Head* and *Tail* of the image have a one-second fade, during which the effects in each—*Scaling*, *Translation*, and *Opacity*—are applied.

Composition

```
...
<Sequence layer = "1">
  <Segment>
    <Image align = "both" adjust = "body" fill = "loop" layer = "1"
location = "\\share\path\vantage-logo small.png" frames = "1"
duration = "00:00:15.000" offset = "00:00:02.000" layout = "none">
      <Head>
        <Scaling x = "30%" y = "30%"/>
        <Translation x = "80%" y = "70%" />
        <Opacity level = "0%"/>
        <Fade duration = "00:00:01.000"/>
      </Head>
      <Body>
        <Scaling x = "30%" y = "30%"/>
        <Translation x = "80%" y = "70%" />
        <Opacity level = "100%"/>
      </Body>
      <Tail>
        <Scaling x = "30%" y = "30%"/>
        <Translation x = "80%" y = "70%" />
        <Opacity level = "0%"/>
        <Fade duration = "00:00:01.000"/>
      </Tail>
    </Image>
  </Segment>
</Sequence>
...
```

File

CML Elements | CML Hierarchy Map

The *File* element adds a fully-qualified path to a file identified by a *Source* or *Subtitle*. The path identifies a file to be utilized in a *Video* or *Audio* element.

The file must be in a valid Post Producer format (*Supported Post Producer Formats*).

The connection between the two elements is that the source value in a *Video* or *Audio* element matches the identifier value in the *Source* element.

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

Attributes

Name	**Description**
location (optional)	Specifies the fully-qualified path to a file, including file name. The location must be accessible to the Vantage Playlist service which is executing the Conform action. Locations that can be processed by Conform are specified as: • *Windows*—D:\pathname\filename.ext • *UNC*—\\share\path\filename.ext Locations that must be pre-processed by Colocate before processing in Conform are specified as: • *URL*—http://WebServerDNSName/Folders/filename.ext • *URL*—s3://bucket/folder/filename.ext"/> (this syntax is a CML construct for use in Vantage only). • *Non-accessible Local Paths*—network-accessible AFP|SMB|CIFS paths that are not Vantage accessible. URL paths and local paths that you enter in compositions must be processed by a Colocate action before processing. Files in URL paths are localized (using credentials you supply in Colocate Colocators), and the URL paths in the CML are replaced by the new, localized path where the file was saved. Local paths to files that are not accessible by Vantage (for example, files created by NewBlueFX, Avid, Final Cut or other Mac OS applications) must be replaced (using a simple string find and replace function) by a share path that IS accessible to Vantage. Colocate produces a copy of the original CML with the new paths, to process in Post Producer workflows.

Note: For more information on Mac OS file processing, see *Colocate Action* and *Creating Compositions from Final Cut Pro 7 XML*. For details on configuring the Colocate action, review the Colocate action man pages.

Example

This example illustrates using a *File* element to specify a file, accessed in the *Video* element via its *Source* identifier value.

Composition

```
<Composition>
  <Source identifier = "1">
    <File location = "\\share\path\Mystic River Seg 1.mov" />
  </Source>
  <Source identifier = "2">
    <File location = "\\share\path\3_FTD_Start at_16.21.mov" />
    <Mask left = "12.5%" right = "87.5%" top = "0%" bottom =
"100%"/>
  </Source>
...
```

Note that the *Segment's* video clip is referenced as *source 1*. It points to the *Source* element with an identifier attribute of value 1. The source 1 attribute in the *Segment* ties the clip to the file specified in the *Source* element with identifier 1.

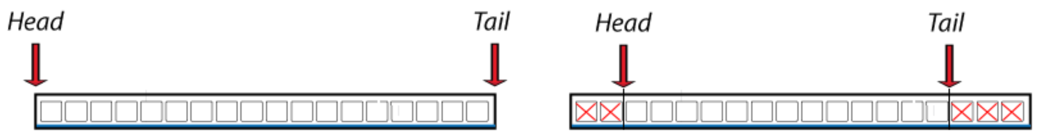
Head

CML Elements | CML Hierarchy Map

The *Head* element is the point (the first frame) of the material on *Advisory*, *Image*, *Title*, *Audio*, and *Video* elements at the beginning (or mark-in point, when using an *Edit*). *Heads* (and *Tails*) by default have no duration; they can represent either edge of the material.

The *Head* (like the *Body* and the *Tail*) may contain effects elements (*Opacity*, *Scaling*, *Translation*, *Rotation*, and *Volume*) that control the audio and video composition. If you do not supply a *Fade* duration, the effects change instantaneously with the time transition from the *Head* to the *Body*—the *Head* effects will not be applied to any video, because the *Head* and *Body* begin at the same moment.

The *Head* is either the beginning of the material or the *Head's Edit* value, if present:



You define a *Head* when you want to add effects on the beginning of the material.

There are no attributes in a *Head* element.

See also *Body*, *Tail*, *Edit*.

Child Elements

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

- *Edit*
- *Fade*

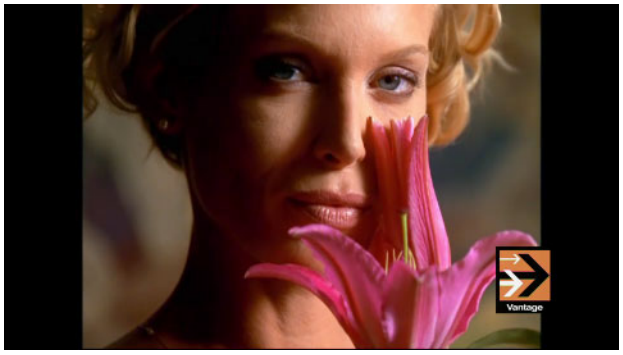
One each of these effects elements may be optionally added to a *Head*, *Body*, or *Tail* element:

- *Opacity*
- *Scaling*
- *Translation*
- *Rotation*
- *Volume*

Example

In this example, the *Head* of the image is scaled and translated to the lower right corner of the frame. Opacity is set to 0 (fully transparent for the 1 second fade.

Here, the *Head* is used to size, position an image, and apply affects:



Composition

```
...
<Image align = "both" adjust = "body" fill = "loop" layer = "1"
location = "\\share\path\vantage-logo small.png" frames = "1"
duration = "00:00:15.000" offset = "00:00:02.000" layout = "none">
  <Head>
    <Scaling x = "30%" y = "30%"/>
    <Translation x = "80%" y = "70%" />
    <Opacity level = "0%"/>
    <Fade duration = "00:00:01.000"/>
  </Head>
  <Body>
    <Scaling x = "30%" y = "30%"/>
    <Translation x = "80%" y = "70%" />
    <Opacity level = "100%"/>
  </Body>
  <Tail>
    <Scaling x = "30%" y = "30%"/>
    <Translation x = "80%" y = "70%" />
    <Opacity level = "0%"/>
    <Fade duration = "00:00:01.000"/>
  </Tail>
</Image>
...
```

Note that the body and tail have identical scaling and translation—resulting in the image displaying at exactly the same size and location for the duration.

Image

CML Elements | CML Hierarchy Map

The *Image* element is a material element which adds a raster graphic image (or a sequence of images) to a Segment. Unlike *Video* elements, an *Image* element references source files directly, via the *location* attribute.

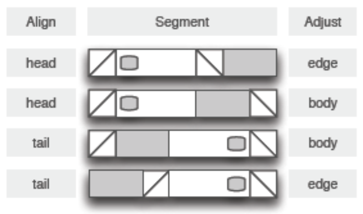
You can add a single image, or you can add an image sequence—a series of pictures—that constitutes a video clip when conformed. For example, you may have a 3-second, cartooned series of images to portray cloudy weather in a weather update promo.

Images are typically used in two ways:

- As an overlay (branding bug, rendered title, etc., applied to an upper layer; typically smaller (or resized) so that the underlying layers are visible. Or, the image may contain an alpha channel to achieve the same goal.
- To produce full frame video at the same resolution as the output frame.

Note: Material items may logically be divided into three temporal parts: the beginning (*Head*), the middle (*Body*), and the end (*Tail*) for the purpose of changing the presentation of the material with the effect elements (*Volume*, *Scaling*, *Translation*, *Rotation*, *Fade*), and applying a *Fade* to define the temporal length of the part.

For a visual representation of the effect of *adjust* and *align* attributes on material that is temporally shorter than the segment itself, see the illustration below, depicting a material element with a *Head*, *Body*, and *Tail*. This graphic illustrates how *align* and *adjust* attributes work together on material expansion:



In the first row, adjustment is on the edge (which could be either the head or tail), and with alignment on the head, the entire clip is started and played normally, at the beginning of the segment.

In the second row, adjustment is on the body. Thus, the head and body (forcing the head) are aligned at the beginning, but the tail is right-aligned to the segment end and the body is stretched.

In the third row, the reverse is depicted, by aligning on the tail.

In the fourth row, the reverse of the first row is depicted, also by aligning on the tail.

Implementing Image Sequences

To implement an image sequence to create video in your output, you should add all of the images to a given directory, each named with a numerically increasing pattern. In the Image element's location attribute, you only specify the file name of the image (*PartlyCloudy0001.png*, for example) that starts the sequence. Based on its based name at the start, Post Producer will attempt to determine the rest of the files in the sequence. Post Producer (without the presence of a wild card character) assumes all files in the sequence start with the same base name without numerical characters except for the sequence, and are sequentially numbered—you can't have any gaps: *PartlyCloudy0002*, *PartlyCloudy0003*, etc.

File names must include a numeric sequence number, and there must not be a break in the sequence. Specify the first filename to use in the sequence. A break in the sequence causes the sequence to stop.

You can use a wildcard asterisk (*) to solve two problems in your file list:

- If a break occurs in the sequence and you do not want it to stop, you must use a wildcard to identify the numeric string to use as the sequence. For example: A series of files named `Banjo_470.dpx` (where 470 is the sequence number pattern) should be referenced with a wild card, such as `Banjo_*.dpx`. The directory is filtered to include all files that fit the criteria and sorted in ascending order.
- If the file names have additional numeric values that are constant, you must use a wildcard to identify the numeric string to use as the sequence. For example: A series of files named `Banjo_25fps_470.dpx` (where 470 is the sequence number pattern) should be referenced with a wild card, such as `Banjo_25fps_*.dpx`. The directory is filtered to include all files that fit the criteria and sorted in ascending order (breaks are ignored when a wild card is used.)

Child Elements

One each of these elements may be added to apply effects or alignment:

- *Head*
- *Body*
- *Tail*

Attributes

Note: The adjust and align properties rows are identical for Advisory, Audio, Canvas, Image, and Title. If you change the definition, copy and paste to all these elements.

The following attributes may be applied to an *Image* element.

Name	Description
adjust (optional)	When the material's duration is less than the duration of the segment, the material may be extended within the segment. This provides a flexible method to define the behavior of overlay material so that it is consistently applied, regardless of the duration of the underlying material. For example, using the *adjust* attribute, you can consistently display a branding bug on underlying video of any duration. There are 3 things to consider when modifying material: 1. What part to adjust—the body or the edge (using *adjust*) 2. Which end to adjust from (based on *align* attribute) 3. How to display—hold, loop, none (based on *fill* attribute). Keywords: *edge* (default) | *body* *edge* (default)—The material is adjusted by extending the edge that is *not* identified by the *align* property and is applied to the duration of the head and tail of the material. *body*—The material is adjusted by extending the body of the material. It is applied to the end of the body that is *not* identified by the *align* property. Example: `<Image adjust = "body"... />`
align (optional)	Specifies which end (or both) of the material to anchor to the timeline of the segment, when the duration of the material is less than the duration of the segment. Keywords: *head* | *tail* | *both.* Example: `<Image align = "tail"... />` *head* (default)—Anchors the beginning of the material to the beginning of the segment. *tail*—Anchors the end of the material to the end of the segment. *both*—Anchors the center of the material at the center of the segment, so that the Head and Tail are equidistant from the beginning and end of the segment. **Note**: If an offset is used in this situation, it is also applied to both ends.

Name	Description

duration (optional)	The amount of time to display the image. Default: 0.
	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:
	Time is measured on a 24-hour clock. Time may include milliseconds (*sss*); frames (:|;FF) as DF (;) or NDF (:), and optional frame rate (FF@FPS), as an integer, decimal number, or rational (for example, 30000/1001). If FPS is not specified, it is inferred from the associated source.
	Rates supported: 23.976, 24, 25, 29.97, 30, 50, 59.94.
	Timecode references are applied to the timecode track specified in the associated Source element's file.
	When using DF timecode, the first 2 (29.97 fps) or 4 (59.94 fps) timecode numbers are dropped every minute except minutes divisible by 10. DF timecode is supported for frame rates of 29.97 and 59.94 frames per second.
	Examples:
	`<Image duration = "00:00:05.500"... />`
	`<Image duration = "00:00:05;14@29.97"... />`

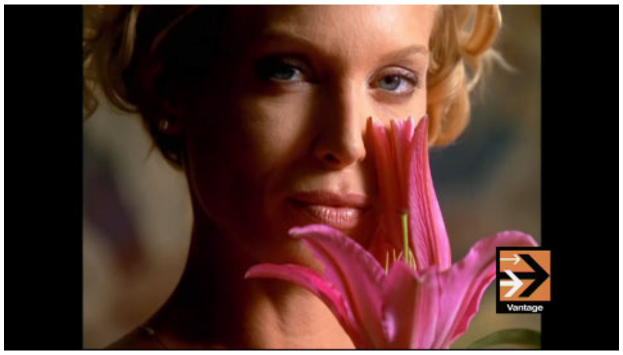
fill (optional)	When the duration of the material is less than the length of the segment, the fill attribute determines if and how the item continues to play: Keywords: *none* | *hold* | *loop*. Example: `<Image fill = "loop"... />` *none* (default)—extend with empty media (no visual or auditory material). *hold*—continue playing the first or last frame to play the material the entire duration of the segment. If *align* = *"head"*, the last frame is repeated. If *align* = *"tail"*, the first frame is repeated. *loop*—play the media repeatedly for the duration of the segment.
frames (optional)	An integer (default=1), which specifies the number of frames in the image sequence. Example: `<Image frames = "60"... />` If there are more images in the folder than you want to use in a given instance (Image), you can specify the first frame you want to use (by file name) in the *location* attribute, and then limit the number to use in the sequence with the *frames* attribute. You should use a numerical file naming sequence that Post Producer can discern (for example 00001.png, 0002.png, etc.,) so that your sequence will be successful. For templates, however, there may be an unknown number of frames; in that case only the numerical sequence is used.

Name	**Description**
instructions (optional)	A keyword specifying that during the conform process, the image is scaled and stretched to the correct display aspect ratio, along with the video. By default, when compositing overlays on output video with non-square display aspect ratios (for example, thin raster broadcast formats including XDCAM 35 and CableLabs SD at 528x480) the overlay is composited pixel for pixel, resulting in a distorted overlay. To adjust the image as appropriate for the output, use this attribute. Keyword: *aspect-ratio* Example: `<Image instructions = "aspect-ratio" />`
layer (optional)	An integer value describing the ordinal position in the composite of all material in this segment. Default: 0. Example: `<Image layer = "3"... />` Material with a lower layer value may be obscured by material with a higher layer value in the same segment. You should supply a layer value higher than the video layer—by default, layer is 0; thus, the video will obscure the graphic.
layout (optional)	Specifies the method used to alter the image to the target dimensions prior to applying any *Scaling* effects. Default: *none*. Keywords: *none* | *center* | *tile* | *zoom* | *fill* | *stretch*. Example: `<Image layout = "zoom"... />` *none*—The image is top-left aligned to the target area. Any unused target area is masked. Usually, when you overlay an image using `layout = none`, you are layering a smaller image on top of the main video/image. If you are not applying any effects (scaling, rotation etc.), the image is composited at the source image size unless you set the compose attribute to *output-size*. If effects are applied, the compositing is performed at the output size. The size used by the conforming process is reported back to you using the compose attribute. *center*—The image is centered within the target area while maintaining the source size and aspect ratio. Any unused target area is masked. *tile*—The image is tiled across the target area maintaining the source size and aspect ratio. Any unused source area is cropped. *zoom*—The image is resized to fill the closest dimension of the target area while maintaining the source aspect ratio. Any unused target area is masked. This results in under-fill when the target and material are not of the same aspect ratio. *fill*—The image is resized to fill both dimensions of the target area while maintaining the source aspect ratio. Any unused source area is cropped. This results in overfill when the target and material are not of the same aspect ratio. *stretch*—The image is resized to completely fill the target area.

Name	Description
location (optional)	Specifies the fully-qualified path to the image (or first image) file, including file name. The location may be specified as: • *Local Drive Letter*—D:\pathname\filename.ext • *UNC*—\\share\path\filename.ext • *URL*—http://SierraStudio.com/media/Vane_logo.png • *URL*—s3://aws/bucket/folder/moms_on_strike.mpg"/> CML with URL paths must be processed by a Compose action to localize the file and produce a copy of the CML to process in Post Producer transcoding actions. The location must be accessible to the Vantage Playlist service which is executing the Conform action. Example: `<Image location = "\\share\path\PartlyCloudy0001.png"... />` Supported formats: PNG|JPEG|BMP|TGA|PSD|TIF.
offset (optional)	The amount of time to wait from the beginning of the segment, to display the material. Default: 0. If you are specifying a relative timecode for video with a timecode track, use these formats: HH:MM:SS:FF@RR | HH:MM:SS:FF | HH:MM:SS;FF If you are specifying an absolute timecode or time value, you can also use these formats if the material has a framerate; otherwise you must use one that can be converted to a time: HH:MM:SS.sss | HH:MM:SS:FF@RR Time is measured on a 24-hour clock. Time may include milliseconds (*sss*); frames (:|;FF) as DF (;) or NDF (:), and optional frame rate (FF@FPS), as an integer, decimal number, or rational (for example, 30000/1001). If FPS is not specified, it is inferred from the associated source. Rates supported: 23.976, 24, 25, 29.97, 30, 50, 59.94. Timecode references are applied to the timecode track specified in the associated Source element's file. When using DF timecode, the first 2 (29.97 fps) or 4 (59.94 fps) timecode numbers are dropped every minute except minutes divisible by 10. DF timecode is supported for frame rates of 29.97 and 59.94 frames per second. Example: `<Image offset = "00:00:05.000"... />`

Example—Single Image

This example illustrates using an *Image* element in a composition to display a small Vantage logo (in a raster graphic file) in the lower right corner of the video for 15 seconds, with fade effects.



Composition

```
...
  <Sequence layer = "1">
    <Segment>
      <Image align = "both" adjust = "body" fill = "loop" layer =
"1" location = "\\share\path\vantage-logo-small.png" frames = "1"
duration = "00:00:15.000" offset = "00:00:02.000" layout = "none">
        <Head>
          <Scaling x = "30%" y = "30%"/>
          <Translation x = "80%" y = "70%" />
          <Opacity level = "0%"/>
          <Fade duration = "00:00:01.000"/>
        </Head>
        <Body>
          <Scaling x = "30%" y = "30%"/>
          <Translation x = "80%" y = "70%" />
          <Opacity level = "100%"/>
        </Body>
        <Tail>
          <Scaling x = "30%" y = "30%"/>
          <Translation x = "80%" y = "70%" />
          <Opacity level = "0%"/>
          <Fade duration = "00:00:01.000"/>
        </Tail>
      </Image>
    </Segment>
  </Sequence>
...
```

Example—Image Sequence

This example illustrates the use of an image sequence to create a lower third title with an image sequence:

Composition

```
...
<Image align = "head" adjust = "body" fill = "none" layer = "1"
location = "\\share\path\frame-000000.png" frames = "132" duration
= "00:00:00.000" layout = "stretch">
</Image>
...
```

The first file of the image sequence is identified with a count of the total image sequence (using the *frames* attribute). The *duration* of the entire sequence can be set to a desired value. Specify 00:00:00:00 to output one frame per image, at the frame rate of the output video.

The *fill* attribute may be used to extend the duration of the image sequence by looping the entire sequence or replaying the final frame.

Adjusting the *Edge* prevents the application of any effects (opacity, scale, translation and rotation). If this is desired, then adjust the *Body*.

Example—Image Sequence with Wildcard

In this example, a wild card (`*_iah.jpg`) is used in the *Image* element's file name reference (bold):

CML

```
<?xml version = "1.0" encoding = "utf-8"?>
<Composition xmlns = "Telestream.Soa.Facility.Playlist">
  <Source identifier = "0">
    <File location = "\\share\path\Radar.wav" />
  </Source>
  <Sequence>
    <Segment>
      <Audio fill = "hold" source = "0">
        <Head>
          <Edit mode = "relative" time = "00:00:00.000" />
        </Head>
        <Tail>
          <Edit mode = "relative" time = "00:00:10.000" />
        </Tail>
      </Audio>
      <Image fill = "loop" location =
"\\share\path\Wx_Image_Sequence\*_iah.jpg" frames = "35" duration
= "00:00:03.917" />
    </Segment>
  </Sequence>
</Composition>
```

The settings for implementing image sequences:

1. The first image of the sequence.
2. The total number of images in the sequence.
3. The length in time of the sequence. This may be specified or you can use the default of frame count multiplied by the output frame rate.

For example, you can provide 40 frames and then specify a 4-second duration—Post Producer plays each file the proportionate number of frames for 4 seconds, to play it out correctly. If you don't specify a duration, the frames are played out at the output frame rate specified in the encoder settings in the Conform action.

Note: For a practical example of an Image sequence, see Basic Image Overlays in the Post Producer Cookbook.

Mask

CML Elements | CML Hierarchy Map

To specify a *Mask* frame—a rectangle within the output frame in which to present material—add a *Mask* element to the *Source* element that identifies the instance. (The *Mask* element is optional; one may be added per *Source*.)

The *Mask* element resizes the source frame (which may have already been cropped) to fill the *Mask* frame, not the output frame. The space beyond the mask (typically pillars or bars) is filled with the fill color specified.

Mask is commonly used for filling space when converting content of 4x3 aspect ratio to output with a 16x9 aspect ratio. The *Mask* element is also commonly used in conjunction with a *Crop* element.

The *Mask* frame is defined relative to the dimensions of the output frame. Positive values originate from the top, left corner. Negative values originate from the bottom, right corner.

The *Mask* values can be specified as a ratio (no designator), fraction (in {}), percent (%), or pixels (px). Pixels are useful if the output resolution is constant for all output. Percent (or ratio) is useful if the aspect ratio remains constant, but the resolution is likely to change.

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

See also *Crop*.

Attributes

These attribute values may be specified by a units designator: pixels (px), ratio, fraction 9 (in {}), or percent (%). If you do not specify a designator, you are specifying a ratio. For example, *bottom="0.222"* is the same as specifying *bottom="22.2%"*. Values may be integers or decimal numbers.

Name	Description
bottom (optional)	Specifies the bottom edge of the frame. Example: `<Mask bottom = "480px"... />`
color (optional)	Keywords: *transparent* | *black* | *white* | *ARGB value* The default keyword is *black*. Example: `<Mask color = "#FFA2A4A6"... />` The supported colors are defined in .NET For details, see the Color Structure. *ARGB*—The RGB + Alpha color, in AARRGGBB format, where the high-order byte (AA) contains the alpha channel and the remaining bytes contain color components for red, green and blue, respectively. Only opaque colors (alpha = FF) are supported.

Name	Description
left (optional)	Specifies the left edge of the output frame. Example: `<Mask left = "0"... />`
right (optional)	Specifies the right edge of the output frame. Example: `<Mask right = "-12.5%"... />`
top (optional)	Specifies the top edge of the output frame. Example: `<Mask top = "0px"... />`

Mix

CML Elements | CML Hierarchy Map

You apply a *Mix* element to a *Source* element to map and mix audio channels from the source file into the output file. For example, a single master file with 10 stereo language tracks might be used in 10 separate source definitions in the composition. Each language-specific *Source* instance would contain the same video, with a different set of audio channels that are moved to the output file using the *Mix* element.

Both mixing and routing may require that you set up and configure the Channel Map filter in the Conform action.

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

Note: For a practical example of typical audio mixing applications, see Mixing Audio in the Composition Cookbook.

See also *Route*.

Attributes

Name	Description

level (optional)	A numeric value (default: 1), specifying the level in unit gain, in percent (% (default)) or decibels (dB).
	Examples:
	`<Mix level = ".707"... />`
	`<Mix level = "3dB"... />`

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

Name	Description
source (required)	Specifies the channel number (as an integer) or other channel identifier key word, of the input audio track or tracks. Keywords: *None* | *1* through *32* | *All* Example: `<Mix source = "1 2" target = "1 2"/>` Combine multiple channels by separating them with a space. You can also specify *None* or *All*, as defined below. *None*—use to specify no channels. *1...32*—the channel number. *All* (default)—use to specify all channels.
target (required)	Specifies the channel number (as an integer) or other channel identifier key word, of the output audio track or tracks. Keywords: *None* | *1* through *32* | *All* Example: `<Mix source = "3" target "1" level = ".9"/>` Combine multiple channels by separating them with a space. You can also specify *None* or *All*, as defined below. *None*—use to specify no channels. *1...32*—the channel number. *All* (default)—use to specify all channels.

Example

This example illustrates how to mix audio.

It is necessary to map the channels to the output for wrapping of the final encoded output. Metadata labels for channels (Left/Right), language (French, German) are defined in the Channel Map filter and the stream settings of the Conform action.

Composition

```
<?xml version = "1.0" encoding = "utf-8"?>
<Composition xmlns = "Telestream.Soa.Facility.Playlist">
  <Source identifier = "1">
    <File location = "\\share\path\Master.mov"/>
    <Mix source = "1 2" target = "1 2" level = "0%"/>
  </Source>
  <Source identifier = "2">
<!-- English -->
    <File location = "\\share\path\Master.mov"/>
    <Mix source = "1 2" target = "1 2" level = "0.707"/>
    <Mix source = "3" target = "1" level = ".9"/>
    <Mix source = "3" target = "2" level = ".9"/>
  </Source>
  <Source identifier = "3">
  <!-- French -->
```

```
    <File location = "\\share\path\Master.mov" />
    <Mix source = "1 2" target = "3 4" level = "0.707"/>
    <Mix source = "4" target = "3" level = ".9"/>
    <Mix source = "4" target = "4" level = ".9"/>
  </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 workflow. -->
  <Sequence layer = "1">
    <Segment>
      <Video source = "1"/>
      <Audio align = "head" source = "2" />
      <Audio align = "head" source = "3" />
      <Audio align = "head" source = "4" />
      <Audio align = "head" source = "5" />
      <Audio align = "head" source = "6" />
      <Audio align = "head" source = "7" />
      <Audio align = "head" source = "8" />
      <Audio align = "head" source = "9" />
      <Audio align = "head" source = "10" />
      <Audio align = "head" source = "11" />
    </Segment>
  </Sequence>
</Composition>
```

Opacity

CML Elements | CML Hierarchy Map

One *Opacity* element may optionally be added to a *Body*, *Head*, or *Tail*.

The *Opacity* element specifies the degree to which items behind it (as indicated by the layer attribute) it can be viewed—its transparency.

For practical purposes, the default *Opacity* value in a *Body* element is 100%, while in the *Head* and *Tail*, it is 0%. This makes it easy to apply a *Fade*, and its duration must be greater than 0. If you are using *Fade* for effects other than fading the opacity to 0 (that is, you do not want to fade the opacity—or you want a partial fade), you must supply an opacity level explicitly.

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

See also *Volume*, *Scaling*, *Translation*, *Rotation*, *Fade*.

Attributes

This attribute must be applied to an *Opacity* element.

Name	Description
level (required)	An integer or decimal number representing the relative opaqueness (where 0 is fully transparent and 100 is fully opaque), expressed as a ratio (no designator) or percent (%) of the original material. Example: `<Opacity level=”50%” />` Default (in *Head* and *Tail* elements): 0%. Default (in *Body* elements): 100%.

Example

In this example, a Vantage logo is placed over a video, and *Opacity* is used to fade the logo in and out.

By setting *Opacity* to zero in the *Head* and *Tail*, and 100% in the *Body*, the logo fades in and out over the course of the video.

Composition

```
...
<Sequence layer = "1">
  <Segment>
    <Image align = "both" adjust = "body" fill = "loop" layer = "1"
location = "\\share\path\vantage-logo-small.png" frames = "1"
duration = "00:00:15.000" offset = "00:00:02.000" layout = "none">
      <Head>
        <Scaling x = "30%" y = "30%"/>
        <Translation x = "80%" y = "70%" />
        <Opacity level = "0%"/>
        <Fade duration = "00:00:01.000"/>
      </Head>
      <Body>
        <Scaling x = "30%" y = "30%"/>
        <Translation x = "80%" y = "70%" />
        <Opacity level = "100%"/>
      </Body>
      <Tail>
        <Scaling x = "30%" y = "30%"/>
        <Translation x = "80%" y = "70%" />
        <Opacity level = "0%"/>
        <Fade duration = "00:00:01.000"/>
      </Tail>
    </Image>
  </Segment>
</Sequence>
...
```

Processor

CML Elements | CML Hierarchy Map

The *Processor* element is a named reference to a specific set of Conform or Tempo action processor settings, which are created, named, and configured directly in the Conform or Tempo action(s) in the target workflow. Processors enable extended or customized media processing on files.

Processors can be added to an action and configured to perform special processing, on a file-by-file basis. For example, you can add a Video Processor to an input file that has specific decoding requirements for audio and video, such as frame rate conversion, Telecine or inverse Telecine, and other pre-processing filters. You can also add an Advanced Title Processor to add animated text and transitions to a composition, such as rendering of 2D/3D animated title sequences. The Advanced Title Processor uses the Titler Engine which implements NewBlueFX Titler Pro. Processors are implemented as appropriate in the Conform and Tempo actions and may not be available in both.

To use a processor in a given file (or job, in a single-file composition), you must specify the processor (for example, *CS_601_BeforeGamma*) by name in the Source of your composition, as shown in the example below.

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

Attributes

Name	Description
template (required)	The name of the processor template (as defined in the target Conform or Tempo action) to use for applying the specialized processing to this file. Example: `<Processor template = "Telecine" />`

Example

This example illustrates the use of a *Processor* element, which specifies how a Telecine file should be decoded when processed by a Conform action, by specifying *Telecine*, which contains the decoding and pre-processing requirements specified in the Conform action.

Composition

```
<Source identifier = "1">
  <File location = "\\share\path\speed_racer.mov" />
  <Processor template = "Telecine" />
</Source>
```

Rotation

CML Elements | CML Hierarchy Map

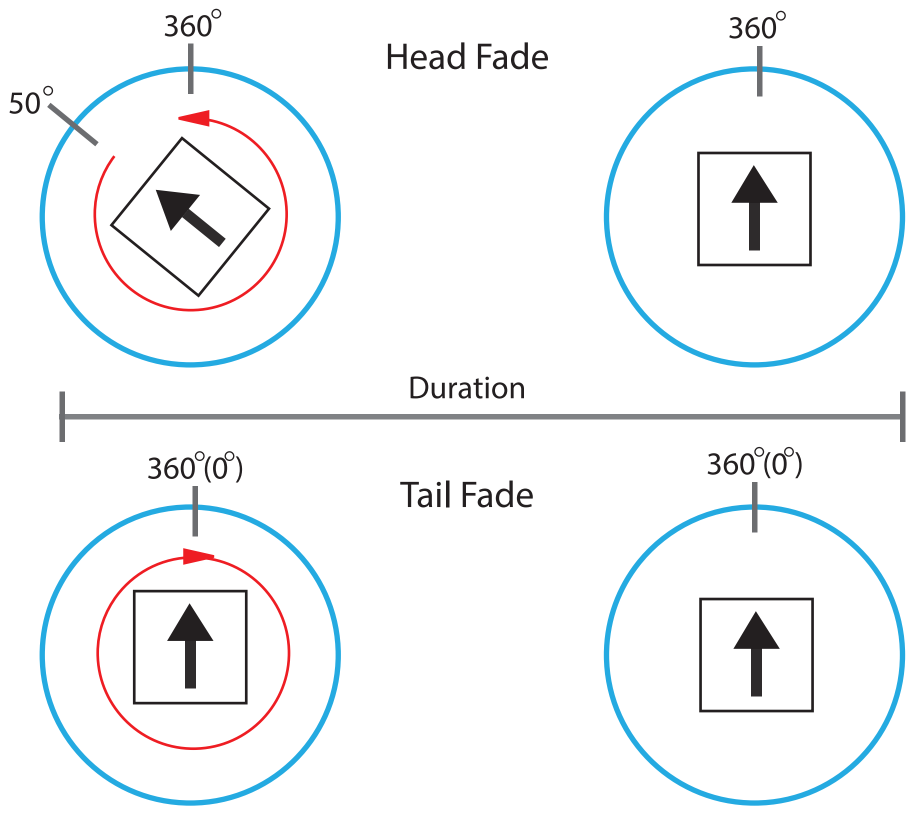
One *Rotation* element may optionally be added to a *Head*, *Body*, or *Tail*. The material (any visual element) rotates as specified, during the Head or Tail duration. It does not rotate during the Body duration, because a Fade is not permitted in a Body. During a Head Fade, the material displays at the specified degree in the Head and then rotates to the specified degree in the Body; during a Tail Fade, it rotates from the specified degree in the Body to the specified degree in the Tail.

Note: Although the Z value represents a point on the circle, a value greater than 360 requires additional trips around to achieve the same point. Thus, 720 is straight up just as 360 is, but with two trips around required, to arrive at the same point. Thus Z is simultaneously used as a point and as a distance.

You can use Rotation to rotate material on the Z axis for the duration of the fade. A relative increase in the rotation value rotates the material counter clockwise, a relative decrease in the rotation value rotates the material clockwise.

In this illustration, an image of an arrow pointing up is rotated in the head and tail: 50 to 360, and then from 360 to 0.

Here is a depiction of rotating an arrow with Tail 50 Body 360 Tail 0



In the Head (Z value = 50), the arrow is presented at 50 degrees and rotated counterclockwise to 360 degrees (Body Z value = 360). In the Tail (Z value = 0), the arrow is presented at 360 degrees and rotated clockwise one full turn back to 0 degrees.

For Rotation to be of practical value the *Fade* element must be present in the *Head* or *Tail* element, and its duration must be greater than 0.

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

See also *Volume*, *Opacity*, *Scaling*, *Translation*.

Calculating Rotation Distance and Direction

To determine the rotation distance (the number of turns or amount of a partial turn), and the direction of the spin, use this formula and rule:

The rotation value = Target – Source. A negative rotation value rotates material clockwise; a positive rotation value rotates material counterclockwise.

In a Head fade, the Head Z value is the source; the target is the Body Z value. In a Tail fade, the Body Z value is the source; the Tail Z value is the target.

Examples

Implementing a Rotation may be difficult to grasp in theory. Here are some examples to help you understand it in practice.

Counter-Clockwise Rotation

At the beginning of a video, you want a Head transition where the material rotates twice counter-clockwise. You specify a positive Body Z value of 720 greater than the Head Z value. For example: Head Z value = 0 and Body Z value = 720. 720 - 0 = 720. Thus, the material rotates 2 full turns from 0 degrees, counter-clockwise.

Clockwise Rotation

At the end of the video, you want a Tail transition where the material rotates twice clockwise. You specify a Tail Z value of 720 less than the Body Z value. Tail Z value = 720 and Body Z value = 0. 0 - 720 = -720. Thus, the material rotates 2 full turns from 0 degrees, clockwise.

Random Rotations

These examples are presented primarily as math problems, just to drill home how to apply the rule:

Scenario One

Head = 50; Body = 360

360 - 50 = 310. Thus, the image rotates counterclockwise from the presentation point of 50 degree (NW) to 360 (which is the same as the 0 point).

Scenario Two

Head Z = 50; Body = 0

0 - 50 = -50. Thus the image rotates clockwise from 50 to 0.

Scenario Three

Body 360; Tail = 0

0 – 360 = -360; thus the image rotate clockwise 360 degrees—one full turn.

Scenario Four

Body 360, Tail 360

360 – 360 = 0; the image does not rotate.

Scenario Five

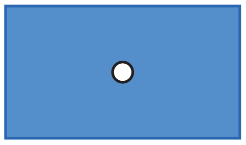
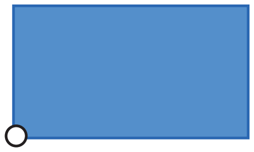
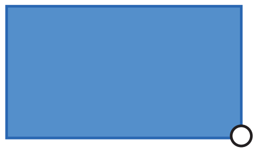
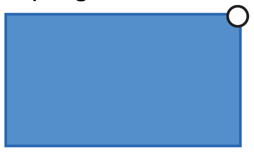
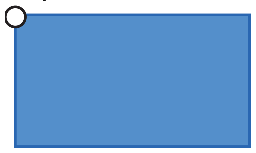
Body -360, Tail -360

-360 - 360 = - 720 = thus the image rotates clockwise 2 full turns

Attributes

The z attribute is required in a *Rotation* element.

Name	Description
z	The degree of rotation on the Z axis. Negative values rotate clockwise; positive values rotate counter-clockwise. Default: 0 degrees. The unit of measure is in degrees. Example: `<Rotation z = "360"... />`

Name	Description
center (optional)	Keyword: the point on the image that is the center of rotation; the point about which the image moves when the rotation occurs. Default: top-left. center | bottom-left | bottom-right | top-right | top-left | top-right. Example: `<Rotation center = "bottom-left"... />` center:  bottom-left:  bottom-right:  top-right:  top-left: 

Example

This example illustrates the application of a typical set of effects on an image, including scaling, translation, and opacity.

Composition

```
...
<Image align = "head" adjust = "edge" fill = "loop" layer = "1"
location = "\\share\path\alpha channel.png" duration =
"00:00:04.000" frames = "1" layout = "none" >
  <Head>
    <Fade duration = "00:00:02.000"/>
    <Scaling x = "100%" y = "100%" />
    <Rotation z = "0" center = "center"  />
    <Opacity level = "100%" />
</Head>
  <Body>
    <Scaling x = "20%" y = "100%" />
    <Rotation z = "20" center = "center" />
    <Opacity level = "100%" />
  </Body>
  <Tail>
    <Scaling x = "100%" y = "100%" />
    <Rotation z = "0" center = "center" />
    <Opacity level = "100%" />
    <Fade duration = "00:00:02.000"/>
  </Tail>
</Image>
...
```

Route

CML Elements | CML Hierarchy Map

You add a *Route* element to a *Source* to map audio tracks from the input file in the associated *File* to other tracks in the output file. Unlike the *Mix* element, the *Route* element does not decode the audio.

Note: When routing tracks other than 1 or 2, tracks 1 and 2 must explicitly be mixed at a level of 0 (`<Mix source = "1 2" target = "1 2" level = "0"/>`) in addition to your intended *Route* elements.

Both mixing and routing may require that you set up and configure the Channel Map filter in the Conform action.

The *source* and *target* attributes are required in a *Route* element.

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

Note: For a practical example of typical audio routing applications, see Routing Audio Without Decoding in the Post Producer Cookbook.

See also *Mix*.

Attributes

Name	Description
source (required)	Specifies the channel number (as an integer) of the input audio track. Default: "". Keywords: null | *1* through *32* | *All* Example: `<Route source = "1 2"... />` Combine multiple channels by separating them with a space. You can also specify "" (empty string) or *All*, as defined below. null (`""-empty string) (default)`)—use to specify no channels. *1...32*—the channel number itself. May be combined, separated by spaces. *All*—use to specify all channels.

Name	Description
target (required)	Specifies the channel number (as an integer) of the output audio track. Default: "". Keywords: *none* | *1* through *32* | *All* Example: `<Route target = "1"... />` Combine multiple channels by separating them with a space. You can also specify *none*— "" (empty string) or *All*: *none*—use to specify no channels. *1...32*—the channel number itself. May be combined, separated by spaces. *All*—use to specify all channels.
order (optional)	If multiple Route elements are present, specifies (by an integer) the order of this route relative to other routes targeting the same channels. Default: 0. Example: `<Route order = "4"... />` This value determines the order in which routes to the same target channel are resolved. The route with the lowest number takes precedence.

Example

This example illustrates how audio in track 5 and 6 are placed in output track 1 and 2 without decoding the audio, by utilizing a *Route* element.

Composition

```
...
<Source identifier = "2"> <!-- English -->
  <File location = "\\share\path\Master_DolbyE_Stereo_German.mov"/
>
  <Route source = "5 6" target = "1 2"/>
</Source>
...
```

Scaling

CML Elements | CML Hierarchy Map

One *Scaling* element may be added optionally to a *Body*, *Head*, or *Tail*. By default, if the source is Video, it is scaled to 100 percent of the target frame size. If it is an Image, it is scaled according to the *Image*'s layout attribute keyword: zoom or stretch, for example.

Add a *Scaling* element when you want to change the display dimensions of visual material. Adjustment is made from the origin, at the upper left corner.

You can adjust the vertical and horizontal dimensions independently, by ratio, percent or pixels. Values greater than the original size will stretch the size of the image beyond the boundaries of the target frame size, and smaller values will shrink it. For example, x=20% will reduce the horizontal display rectangle to 1/5 of the target frame size.

If you don't supply a *Scaling* element, will be stretched if necessary to fit in the target frame size.

For this effect to be of practical value in a *Head* or *Tail* element, the *Fade* element must be present, and its duration must be greater than 0.

The *Scaling* element is often used in conjunction with the *Translation*. Use a *Fade* to create animation.

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

See also *Volume*, *Opacity*, *Translation*, *Rotation*.

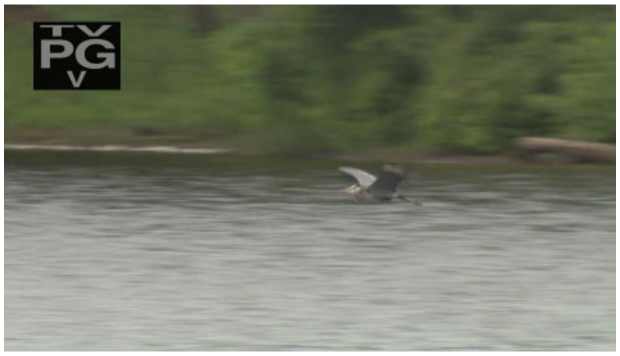
Attributes

Although both x and y are optional, if neither were supplied, the *Scaling* element as defined would be of no practical value.

Name	**Description**
x (optional)	Specifies the change in the horizontal dimension of the display rectangle of the visual material, as pixels (px), percent (%), or ratio (no unit designator). Example: `<Scaling x = "200%" y = "200%"/>`
y (optional)	Specifies the change in the vertical dimension of the display rectangle of the visual material, as pixels (px), percent (%), or ratio (no unit designator). Example: `<Scaling x = "2" y = "2"/>`

Example

In this example an *Advisory* image (the default image—thus, not specified) is scaled to twice its original size in the output frame. A percent is used here; but a ratio (of 2) could also be used. The result is the same in both cases.



Composition

```
...
<Video align = "head" adjust = "edge" source = "1" layer = "0" >
  <Head>
    <Edit time = "00:00:30.00" />
    <Scaling x = "100%" y = "100%" />
    <Translation x = "0%" y = "0%" />
    <Opacity level = "100%" />
    <Fade duration = "00:00:05.00" />
  </Head>
  <Body>
    <Translation x = "0" y = "0%" />
    <Scaling x = "50%" y = "100%" />
    <Opacity level = "100%" />
  </Body>
  <Tail>
    <Edit time = "00:00:45.00" />
    <Opacity level = "100%" />
    <Scaling x = "0%" y = "100%" />
    <Fade duration = "00:00:05.00" />
  </Tail>
</Video>
...
```

If you were to use a pixel designator (*px*), you'd have less flexibility in uniformly applying this attribute to images of varying sizes: Adding 10 pixels to a 100 pixel image would increase its size by 10 percent, but adding 10 pixels to a 200 pixel image would only increase it by 5 percent.

Note also, that when using a ratio or a percent, the use of the same but negative value does not have the directly opposite effect. For example: To double the size of an image, you would increase it by a ratio of 2 or 200%; but to reduce the size of an image by half, you would decrease it by a ratio of 0.5, or 50%.

Segment

CML Elements | CML Hierarchy Map

One or more *Segment* elements may be added to a *Sequence*. A *Segment* is comprised of material items (video, audio, titles, graphics, etc.). A *Segment* may also contain one or more *Sequences*, which allows a single material item (for example, a *Title*) to span a sequence of other material items.

The duration of a *Segment* is determined by the duration of the longest item (including any offset applied to that item).

Child Elements

Optionally, one each of these material elements may be added to a *Segment*:

- *Advisory*
- *Title*
- *Image*
- *Audio*
- *Video*
- *Canvas*

Optionally, a *Sequence* may also be added to a *Segment* to create an independent timeline for additional segments. The *Sequence* and *Segment* elements are the only recursive elements in CML.

Attributes

Name	Description
identifier (optional)	A user-assigned SCTE-35 ID for this segment. May be a description or an ad insertion ID, for example.

Name	Description
instructions (optional)—Conform action only	Keywords applied to segments for stream conditioning. They specify how the Conform action should insert I-frames for VOD DAI, regarding placement of keyframes and splice points to support Canoe/Black Arrow dynamic ad insertion specifications. Multiple keywords permitted, each separated by a space. Chronicle instructions (define in the row below) are ignored in Conform; they are passed through Conform into the As Run CML output, which is generated by Conform when the Generate Composition Chronicle option is enabled in the inspector. Keywords: *clear-captions* | *keyframe* | *splice* | *splice-end* Example: `<Segment instructions = "keyframe splice"/>`. *clear-captions*—Clears buffered captions from the previous segment, preventing them from being displayed in the current segment, in case an out point was set in the CML before the embedded captions are cleared automatically. This is useful when changing source files or removing content between segments. *keyframe*—instructs the encoder to insert an IDR frame; the MPEG-2 encoder to add black frames, to complete the GOP of the CURRENT segment. This should be used when creating splices of any kind. *splice*—indicates a splice point at the beginning of this segment; adds a SCTE-35 dynamic ad flag to the output stream. This flag is tagged with an ID that is passed with the SCTE-35 location to the transport stream output. *splice-end*—adds a SCTE-35 dynamic ad flag to the output stream. This flag is tagged with an ID that is passed with the SCTE-35 location to the transport stream output. The CSV file generated by the Chronicle action will list separate stop and start points at the start and end of each segment. DAI-enabled systems will skip the content segment.
instructions (optional)—Chronicle Action Only (for use in VOD | DAI applications)	These keywords define the behavior of the Chronicle action for proper alignment of the fields when creating the output CSV file. Multiple keywords permitted, each separated by a space. Keywords: *avail* | *local* | *break* | *next* | *leadblack* | *midroll* | *postroll* | *preroll* | *midroll_local* | *postroll_local* | *preroll_local* *avail* | *local*—flags for local spots; either term may be used. *break*—flag for generic break insertion to be made at the specified time. *next*—flag to set the end keyframe time for a splice-end instruction. *leadblack*—flag for Chronicle action to set the segment as the lead black segment. The following keywords for national and local ads set flags to be processed by a Chronicle action in a workflow, passed to the Chronicle action as part of VOD Producer DAI workflows: *midroll* | *midroll_local* | *postroll* | *postroll_local* | *preroll* | *preroll_local*—Keyword alignment specified in Canoe CSV output from the Chronicle action.

Name	Description
time (As Run)	As Run attribute. This is the starting time of the segment in the output, after conforming the composition. This value is set by the Conform action and inserted in the As Run composition, and should not be set by you.
transcode (As Run)	As Run attribute. This attribute advises if this segment was direct converted or not. This value is set by the Conform action and inserted in the As Run composition, and should not be set by you.

Example

This example illustrates the use of a *Segment* in a composition: simply as a container for one or more instance of media.

Composition

```
...
<Sequence layer = "1">
  <Segment>
    <Image align = "both" adjust = "body" fill = "loop" layer = "1"
location = "\\share\path\vantage-logo-small.png" frames = "1"
duration = "00:00:15.000" offset = "00:00:02.000" layout = "none">
      <Head>
        <Scaling x = "30%" y = "30%"/>
        <Translation x = "80%" y = "70%" />
        <Opacity level = "0%"/>
        <Fade duration = "00:00:01.000"/>
      </Head>
      <Body>
        <Scaling x = "30%" y = "30%"/>
        <Translation x = "80%" y = "70%" />
        <Opacity level = "100%"/>
      </Body>
      <Tail>
        <Scaling x = "30%" y = "30%"/>
        <Translation x = "80%" y = "70%" />
        <Opacity level = "0%"/>
        <Fade duration = "00:00:01.000"/>
      </Tail>
    </Image>
  </Segment>
</Sequence>
...
```

Sequence

CML Elements | CML Hierarchy Map

You can add one or more *Sequence* elements to a *Composition*, or as sub-sequences, to a *Segment*. You add a *Sequence* when you need to establish a separate, independent timeline in your composition, upon which to place segments of material.

Child Elements

One or more Segments may be added to a *Sequence* element. *Sequence* and *Segment* elements are the only recursive elements in CML.

Attributes

The layer attribute may be applied to a *Sequence* element.

Name	Description
layer (optional)	An integer value describing the ordinal position in the composite of all material in this segment. Default: 0. Example: `<Sequence layer = "10"... />` Material in a lower layer sequence may be obscured by material in a higher layer sequence.

Example

A *Sequence* is always used to organize media on an independent timeline, with one or more *Segment* elements, which contain the media. It's often used to combine multiple material elements. In this example, the graphic overlay spans two video clips within the segment by creating a complex video clip using the *Sequence* element.

Composition

```
...
<Composition created = "2013-06-26T10:16:34.1293676-07:00" xmlns =
"Telestream.Soa.Facility.Playlist">
  <Source identifier = "1">
    <File location = "\\share\path\Mystic River Seg 1.mov" />
  </Source>
  <Source identifier = "2">
    <File location = "\\share\path\3_FTD_Start at_16.21.mov" />
    <Mask left = "12.5%" right = "87.5%" top = "0%" bottom =
"100%"/>
  </Source>
  <Sequence layer = "1">
    <Segment>
```

```
      <Image align = "both" adjust = "body" fill = "loop" layer =
"1" location = "\\share\path\vantage-logo small.png" frames = "1"
duration = "00:00:15.000" offset = "00:00:02.000" layout = "none">
        <Head>
          <Scaling x = "30%" y = "30%"/>
          <Translation x = "80%" y = "70%" />
          <Opacity level = "0%"/>
          <Fade duration = "00:00:01.000"/>
        </Head>
        <Body>
          <Scaling x = "30%" y = "30%"/>
          <Translation x = "80%" y = "70%" />
          <Opacity level = "100%"/>
        </Body>
        <Tail>
          <Scaling x = "30%" y = "30%"/>
          <Translation x = "80%" y = "70%" />
          <Opacity level = "0%"/>
          <Fade duration = "00:00:01.000"/>
        </Tail>
      </Image>
    </Segment>
  </Sequence>
  <Sequence layer = "0">
    <Segment>
      <Video source = "1" >
        <Tail>
          <Edit time = "00:00:8.933"/>
        </Tail>
      </Video>
    </Segment>
    <Segment key-frame-align = "true">
      <Video source = "2" >
        <Tail>
          <Edit time = "00:00:10.000"/>
        </Tail>
      </Video>
    </Segment>
  </Sequence>
</Composition>
...
```

Shadow

CML Elements | CML Hierarchy Map

One *Shadow* element can optionally be added to a *Title* to apply a shading effect to the title text. The shadow is a replication of the *Title* text using the same attributes, but offsetting the text, applying a different color, and adjusting opacity through the softness attribute.

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

Note: For a practical example of adding a shadow to a title, see Adding Titles in the Composition Cookbook.

See also *Title*, *Area*.

Attributes

Name	Description
color (optional)	This is the color that the shadow is rendered in. Keywords: *transparent* | *black* | *white* | <color> | *ARGB* Default: *black*. Supported color names are defined in .NET. See Color Structure. Example: `<Shadow color = "10"... />` *ARGB*—The RGB + Alpha color, in 0xAARRGGBB format, where the high-order byte (AA) contains the alpha channel and the remaining bytes contain color components for red, green and blue, respectively. The alpha channel specifies the opacity of the pixel, where a value of 0x00 represents a pixel that is fully transparent and a value of 0xFF is fully opaque.
horizontal-offset (optional)	Specifies the horizontal offset of the shadow, as pixels (px), percent (%), or ratio (no unit designator). Positive numbers adjust the shadow text to the right. Default: 4 px. Example: `<Shadow horizontal-offset = "10px"... />`
softness (optional)	Specifies the relative blurriness of the shadow as a ratio or percent. Larger numbers increase softness. Default: 0.5 (50%). Example: `<Shadow softness = "10%"... />` The value may be an integer or real number to 3 digits, as a ratio (no unit designator) or percent (%).

Name	Description
vertical-offset (optional)	Specifies the vertical offset of the shadow, as pixels (px), percent (%), or ratio (no unit designator). Positive numbers adjust the shadow text downward. Default: 4 px. Example: `<Shadow vertical-offset = "20px"... />`

Example

In this example, a *Title* is applied, with every attribute, including an *Area* and *Shadow* element (*Shadow* highlighted).

Composition

```
...
<Title align = "head" adjust = "edge" fill = "none" layer = "2"
duration = "00:00:04.00" offset = "00:00:01.000" font =
"Helvetica" size = "50pt" style = "italic" weight = "bold"
foreground-color = "bisque" background-color = "transparent" wrap
= "false" horizontal-align = "left" vertical-align = "middle"
layout = "stretch">
  <Area top = "70%" left = "20%" bottom = "90%" right = "80%" />
  <Shadow color = "gold" softness = "10%" vertical-offset = "6%"
horizontal-offset = "6%" />
  Post Producer Titles
</Title>
...
```

Note: The value of the *Title* is not specified in an attribute. It is specified as the text of the element.

Source

CML Elements | CML Hierarchy Map

One or more *Source* elements may be added to a *Composition*. *Source* elements by convention are the first elements in a composition, because it is helpful to know the file you're targeting, and it is common to change the file names each time a job is run.

You add one *Source* element for each file you want to identify, to be used as material in a *Video*, *Audio*, and *Image* element.

Child Elements

These elements may be added to a *Source* element.

- *File*
- *Crop*
- *Mask*
- *Mix*
- *Route*
- *DolbyE*
- *Subtitle*
- *Processor*

Attributes

Name	Description
identifier (required)	An integer value, which must be unique among all *Source* elements. Example: `<Source identifier = "1">` This identifier value is used in the *source* attribute of *Canvas*, *Image*, *Title*, *Audio*, and *Advisory* elements to reference this specific file.
instructions (optional)	Keywords which specify that the file must or must not be direct-converted by the Conform action. This attribute overrides the Allow Direct Convert option in the Conform action. Keyword: *force-direct-convert* | *no-direct-convert* Example: `<Source identifier = "1" instructions = "no-direct-convert">` In media with mixed footage—where you may not want otherwise-matched, or nearly-matched footage to be direct-converted—but the enabled Allow Direct Convert option would allow it, you can override the sanity check in the CML, specifying that Conform should *not* direct-convert the source, by specifying the keyword *no-direct-convert*.

Name	Description
timecode (optional)	Keyword to specify which type of timecode to read from the file. Default: *none*. Example: `<Source timecode = "vitc1">` Keywords: *none* | *ltc* | *vitc1* | *vitc2* *none*—timecode is derived from the source file container (for example: Header Timecode.) If the source does not have header timecode, the Conform action assumes that the first frame starts at 00:00:00:00. *ltc*—use timecode flagged as longitudinal timecode *vitc1*—use timecode flagged as VITC line 1 *vitc2*—use timecode flagged as VITC line 2 **Note**: VITC timecodes are not the same as timecode from VBI.

Example

This example illustrates the typical use of a *Source* element to reference an input file being used in a video instance in one of the segments in the composition.

Composition

```
<?xml version = "1.0" encoding = "utf-8"?>
<Composition>
  <Source identifier = "1">
    <File location = "\\share\path\1_A2_Show_ONLY_NO-GFX2.mov" />
  </Source>
  <Sequence layer = "0">
    <Segment>
...
      <Video align = "head" adjust = "edge" fill = "none" source =
"1" layer = "0">
        <Head>
          <Edit time = "00:00:00.000" />
        </Head>
        <Tail>
          <Edit time = "00:00:07.000" />
        </Tail>
      </Video>
    </Segment>
  </Sequence>
</Composition>
```

Subtitle

CML Elements | CML Hierarchy Map

Optionally, you can add one or more *Subtitle* elements to a *Source*. The *Subtitle* element optionally uses a *File* element to associate the source media file with a closed caption (.scc) file. If the media and the SCC file are not already timecode-aligned, they can be aligned in the composition directly, using the *offset* parameter.

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

Only one subtitle file can be played at a time; when multiple files (and subtitle files) are present, use the *order* attribute to specify the subtitle file to use.

Without specifying an SCC file, you can also process embedded captions to propagate subtitles. You can also optionally preserve 708 captions; see the *preserve708* attribute, below.

To process captions in either case (file-based or embedded,) the Conform action in the workflow that processes the CML source file must have the Closed Captions filter enabled, and it must use a frame rate that supports closed captions (608: 29.97, 708: 23.976, 29.97 or 59.94 fps.)

If the Conform action is configured to encode the media in a QuickTime container, the 608 and/or 708 tracks must be enabled for the container.

See also *File*.

Child Elements

One *File* can be added to a *Subtitle* element.

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 1 minute into the clip, which must also be taken into consideration.

• 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 files' 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 1 minute into the clip, as in the first example.

• 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

Name	Description
offset (optional)	The amount of time required to align non-matching timecode references of source media and the closed caption file. The offset value is added to the media timecode reference to adjust for the difference. The offset has a range of 24 hours. Default: `"00:00:00.000"`. If you are specifying a relative timecode for video with a timecode track, use these formats: HH:MM:SS:FF@RR | HH:MM:SS:FF | HH:MM:SS;FF If you are specifying an absolute timecode or time value, you can also use these formats if the material has a framerate; otherwise you must use one that can be converted to a time: HH:MM:SS.sss | HH:MM:SS:FF@RR Time is measured on a 24-hour clock. Time may include milliseconds (*sss*); frames (:|;FF) as DF (;) or NDF (:), and optional frame rate (FF@FPS), as an integer, decimal number, or rational (for example, 30000/1001). If FPS is not specified, it is inferred from the associated source. Rates supported: 23.976, 24, 25, 29.97, 30, 50, 59.94. Timecode references are applied to the timecode track specified in the associated Source element's file. When using DF timecode, the first 2 (29.97 fps) or 4 (59.94 fps) timecode numbers are dropped every minute except minutes divisible by 10. DF timecode is supported for frame rates of 29.97 and 59.94 frames per second. Example: `<Subtitle offset = "01:43:05.333"... />`
order (optional)	Integer (default: 0), which specifies the rank of this subtitle in relation to other subtitle files, if present. Only the highest-ranking subtitle file is actually played out.

Name	Description
preserve708	When propagating captions, the default *behavior* (false) is to enable propagation per source with the Subtitle element. Assuming the Conform action is configured appropriately for captions creation, with the caption filter enabled and caption tracks added as needed, the use of this attribute will up-convert 608 captions in the source to make 708 captions. This behavior can be modified with an attribute preserve708 set to true: `<Subtitle preserve708 = "true" />` will keep any 708 captions in the source. If there are no 708 captions in the source, 608 captions will be up-converted regardless.

Examples

Here are two examples—using file-based captions, and using embedded captions.

Basic Subtitle with an SCC File

In this example the subtitle file (*Chronicle St Lucia TC.scc*) is assigned order number 1 (there's only one in this example):

Composition

```
<?xml version = "1.0" encoding = "utf-8"?>
<Composition xmlns = "Telestream.Soa.Facility.Playlist">
  <Source identifier = "1">
    <File location = "\\share\path\Chronicle St. Lucia TC.mov" />
    <Subtitle order = "1">
      <File location = "\\share\path\Chronicle St. Lucia TC.scc" />
    </Subtitle>
  </Source>
  <Sequence>
    <Segment>
      <Video source = "1" />
    </Segment>
  </Sequence>
</Composition>
```

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.

For example:

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

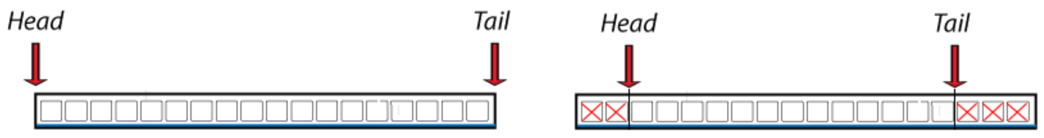
Tail

CML Elements | CML Hierarchy Map

The *Tail* element is the point (or frame) on an *Advisory*, *Image*, *Title*, or *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 can represent either edge of the material.

However, the *Tail* (like the *Head* and the *Body*) may contain effects elements (*Opacity*, *Scaling*, *Translation*, *Rotation*, and *Volume*) that control the *Audio* and *Video* in a composition. If you do not supply a *fade* and *duration*, the effects change instantaneously with the time transition from the *Body* to the *Tail*—the *Tail* effects will not be applied to any video, because the end of the *Body* and *Tail* are at the same moment—the end of the material.

The *Tail* is the end of the material or the *Tail's Edit* value, if present:



You define a *Tail* when you want to perform effects on the end of the material.

There are no attributes in a *Tail* element.

See also *Head*, *Body*.

Child Elements

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

- *Edit*
- *Fade*

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

- *Opacity*
- *Scaling*
- *Translation*
- *Rotation*
- *Volume*

Example

In this example, the video is trimmed by 4 seconds at the end, by adding an *Edit* to the *Tail*.

Composition

```
...
<Source identifier = "1">
  <File location = "\\share\path\Color bars.mov" />
</Source>
<Source identifier = "2">
  <File location = "\\share\path\Mystic River Seg 1.mov" />
</Source>
<Sequence layer = "0">
  <Segment>
    <Video source = "1">
      <Head>
        <Edit time = "00:00:00.000"/>
      </Head>
      <Tail>
        <Edit time = "00:00:04.000"/>
      </Tail>
    </Video>
  </Segment>
...
```

Target

CML Elements | CML Hierarchy Map

Optionally, one Target element may be added to a composition. You should add a *Target* when you want the timecode properties to be specified directly in the composition. Otherwise, they are determined at run time, when the composition is conformed—based on the encoder and settings configured in the Conform action.

You add a *Timecode* element to a *Target* element and specify the *type* and the *time* attributes.

Child Elements

One *Timecode* can be added to a *Target* element.

Composition

```
...
  <Target>
    <Timecode type = "source" time = "01:00:00;00@29.97"/>
  </Target>
...
```

Timecode

CML Elements | CML Hierarchy Map

One *Timecode* can be added to a *Target*. When you want to control the timecode of the output directly in CML, you add a *Timecode* element to a *Target* element and specify the *type* and the *time* attributes.

If a timecode must be set for the output without specifying it directly in the CML, specify it in the Override Timecode value in Conform action's Timecode filter. The timecode value can be set statically directly in Conform or it can be bound to a variable in an upstream action and that variable assigned to the Override Timecode value in Conform.

In addition, in the Conform action Transcoder, you must check the Timecode Processing Filter and select Use CML Target Time Code in Override.

There are no child elements of *Timecode*.

See also*Target*.

Attributes

Name	Description
type (required)	Specifies the type of timecode as a keyword: *source*. Example: `<Timecode type = "source" time = "01:00:00;00@29.97"/>`
time (required)	Specifies the starting timecode to utilize. If you specify a relative timecode for video with a timecode track, use these formats: HH:MM:SS:FF@RR | HH:MM:SS:FF | HH:MM:SS;FF If you specify an absolute timecode or timestamp value, you can also use these formats if the material has a frame rate; otherwise you must use one that can be converted to a time: HH:MM:SS.sss | HH:MM:SS:FF@RR Time is measured on a 24-hour clock. Time may include milliseconds (*sss*); frames (:|;FF) as DF (;) or NDF (:), and optional frame rate (FF@FPS), as an integer, decimal number, or rational (for example, 30000/1001). If FPS is not specified, it is inferred from the associated source. Rates supported: 23.976, 24, 25, 29.97, 30, 50, 59.94. Timecode references are applied to the timecode track specified in the associated Source element's file. When using DF timecode, the first 2 (29.97 fps) or 4 (59.94 fps) timecode numbers are dropped every minute except minutes divisible by 10. DF timecode is supported for frame rates of 29.97 and 59.94 frames per second. Example: `<Timecode type = "source" time = "01:00:00;00@29.97"/>`

Composition

```
...
  <Target>
    <Timecode type = "source" time = "01:00:00;00@29.97"/>
  </Target>
...
```

Title

CML Elements | CML Hierarchy Map

When you want to render text in the video frame, add a *Title* element to the segment. There may be multiple *Title* elements per segment.

The *Title* element is a material element that provides a text string, with attributes for optional formatting. You supply the text string as the value of the element.
For example: `<Title>Up Next...</Title>`.

When you render text by the use of the Title element, you have to know the file's character encoding format. For Roman character sets, you can use ANSI or UTF-8 encoding; for 2-byte non-Roman fonts (Cyrillic or Chinese, for example), you must save the file as UCS-2 or UTF-16 little-endian.

The font family being rendered must be present in the operating system of the Vantage server where the Vantage Edit service executing the Conform action is hosted. For details on text rendering and the use of font families, see the *Translation* element.

Note: Material items may logically be divided into three temporal parts: the beginning (*Head*), the middle (*Body*), and the end (*Tail*) for the purpose of changing the presentation of the material with effect elements (*Volume*, *Scaling*, *Translation*, *Rotation*, *Fade*), and applying a *Fade* to define the temporal length of the part.

The *Title* element has two optional child elements: *Area* and *Shadow*. The *Area* element enables you to position the title text exactly where you want it rendered on the frame. The *Shadow* element enables you to add a shadow to the text.

Note: For a practical example of adding a title, see Adding Titles in the Post Producer Cookbook.

See also *Area*, *Shadow*.

Child Elements

These optional elements may be added to control the title text:

- *Area*—specifies the rectangle in which to render the text.
- *Shadow*—specifies an optional shadow on the text.

These elements may be added to apply effects or alignment:

- *Head*
- *Body*
- *Tail*

Attributes

Name	Description
adjust (optional)	When the material's duration is less than the duration of the segment, the material may be extended within the segment. This provides a flexible method to define the behavior of overlay material so that it is consistently applied, regardless of the duration of the underlying material. For example, using the *adjust* attribute, you can consistently display a branding bug on underlying video of any duration. There are 3 things to consider when modifying material: 1. What part to adjust—the body or the edge (using *adjust*) 2. Which end to adjust from (based on *align* attribute) 3. How to display—hold, loop, none (based on *fill* attribute). Keywords: *edge* (default) | *body* *edge* (default)—The material is adjusted by extending the edge that is *not* identified by the *align* property and is applied to the duration of the head and tail of the material. *body*—The material is adjusted by extending the body of the material. It is applied to the end of the body that is *not* identified by the *align* property. Example: `<Title adjust = "body"... />`
align (optional)	Specifies which end (or both) of the material to anchor to the timeline of the segment. Keywords: *head* | *tail* | *both.* Example: `<Title align = "tail"... />` *head* (default)—Anchors the beginning of the material to the beginning of the segment. *tail*—Anchors the end of the material to the end of the segment. *both*—Anchors the center of the material at the center of the segment, so that the *Head* and *Tail* are equi-distant from the beginning and end of the segment. **Note**: If an offset is used in this situation, it is also applied to both ends.

Name	**Description**
rendering-hint (optional)	Specifies the hinting method used to render the font face. Default: *auto*. Keywords: *auto* | *system-default* | *single-bit-per-pixel-grid-fit* | *single-bit-per-pixel* | *anti-alias-grid-fit* | *anti-alias* | *clear-type-grid-fit* *auto or no specification (default)*—Automatically select the best rendering mode. This may not always produce best results, especially when *background-color* is set to *transparent*. *system-default*—Each character is drawn using its glyph bitmap, with the system default rendering hint. *single-bit-per-pixel-grid-fit*—Each character is drawn using its glyph bitmap. Hinting is used to improve character appearance on stems and curvature. *single-bit-per-pixel*—Each character is drawn using its glyph bitmap. Hinting is not used. *anti-alias-grid-fit*—Each character is drawn using its anti-aliased glyph bitmap with hinting. Improved quality due to anti-aliasing, but at a higher performance cost. *anti-alias*—Each character is drawn using its anti-aliased glyph bitmap without hinting. Improved quality due to anti-aliasing. Stem width differences may be noticeable because hinting is turned off. *clear-type-grid-fit*—The highest quality setting. Each character is drawn using its glyph ClearType bitmap with hinting. The highest quality setting. Used to take advantage of ClearType font features. Example: `<Title anti-alias = "clear-type-grid-fit"... />`
background-color (optional)	Specifies the color that the area is painted, over which the text is rendered. Default: *transparent*. Keywords: *transparent* | *black* | *white* | *ARGB* The supported colors are defined in .NET. For details, see Color Structure. *ARGB*—The RGB + Alpha color, in 0xAARRGGBB format, where the high-order byte (AA) contains the alpha channel and the remaining bytes contain color components for red, green and blue, respectively. The alpha channel specifies the opacity of the pixel, where a value of 0x00 represents a pixel that is fully transparent and a value of 0xFF is fully opaque.

Name	Description

duration (required)	The amount of time to display the *Title*. Default: 0.
	If you are specifying a relative timecode for video with a timecode track, use these formats:
	HH:MM:SS:FF@RR | HH:MM:SS:FF | HH:MM:SS;FF
	If you are specifying an absolute timecode or timestamp value, you can also use these formats if the material has a frame rate; otherwise you must use one that can be converted to a time:
	HH:MM:SS.sss | HH:MM:SS:FF@RR
	Time is measured on a 24-hour clock. Time may include milliseconds (*sss*); frames (:|;FF) as DF (;) or NDF (:), and optional frame rate (FF@FPS), as an integer, decimal number, or rational (for example, 30000/1001). If FPS is not specified, it is inferred from the associated source.
	Rates supported: 23.976, 24, 25, 29.97, 30, 50, 59.94.
	Timecode references are applied to the timecode track specified in the associated Source element's file.
	When using DF timecode, the first 2 (29.97 fps) or 4 (59.94 fps) timecode numbers are dropped every minute except minutes divisible by 10. DF timecode is supported for frame rates of 29.97 and 59.94 frames per second.
	Examples:
	`<Title duration = "00:00:05.500"... />`
	`<Title duration = "00:00:05;14@29.97"... />`

fill (optional)	When the duration of the material is less than the length of the segment, the fill attribute determines if and how the item continues to play: Keywords: *none* | *hold* | *loop*. Example: `<Title fill = "loop"... />` *none* (default)—extend with empty media (no visual or auditory material). *hold*—continue playing the first or last frame to play the material the entire duration of the segment. If *align* = *"head"*, the last frame is repeated. If *align* = *"tail"*, the first frame is repeated. *loop*—play the media repeatedly for the duration of the segment.
font (optional)	The family name of the font, which must be present on the system where the Vantage Edit service executing the Conform action is hosted. Default: Helvetica. For details, see .NET Font Family. For composition file encoding requirements for non-Roman fonts, see *Composition File Requirements*.

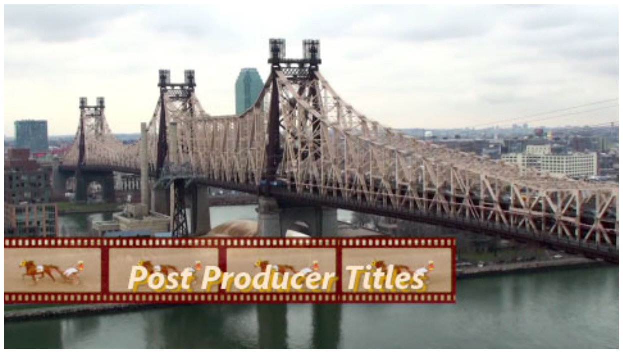
Name	**Description**
foreground-color (optional)	Keyword or ARGB value to define the color that the font should be rendered in. Default: white. Keywords: *black* | *transparent* | *white* Example: `<Title foreground-color = "black"... />` The supported colors are defined in .NET. For details, see Color Structure. *ARGB*—The RGB + Alpha color, in 0xAARRGGBB format, where the high-order byte (AA) contains the alpha channel and the remaining bytes contain color components for red, green and blue, respectively. The alpha channel specifies the opacity of the pixel, where a value of 0x00 represents a pixel that is fully transparent and a value of 0xFF is fully opaque.
horizontal-align (optional)	A keyword describing how the text is rendered on the horizontal plane, relative to the area. Default: left. Keywords: *left* | *right* | *center* | *justify* Example: `<Title horizontal-align = "right"... />`
instructions (optional)	A keyword specifying that during the conform process, the title is scaled and stretched to the correct display aspect ratio, along with the video. By default, when compositing overlays on output video with non-square display aspect ratios (for example, thin raster broadcast formats including XDCAM 35 and CableLabs SD at 528x480) the overlay is composited pixel for pixel, resulting in a distorted overlay. To adjust the title as appropriate for the output, use this attribute. Keyword: *aspect-ratio* Example: `<Title instructions = "aspect-ratio" />`
layer (required)	An integer value describing the ordinal position in the composite of all material in this segment. Default: 0. Example: `<Title layer = "10"... />` Material with a lower layer value may be obscured by material with a higher layer value in the same segment. You should supply a layer value higher than the video layer—by default, layer is 0; thus, the video will obscure the graphic.

Name	Description
offset (optional)	Specifies the amount of time to wait from the beginning of the segment, to display the material. Default 0. If you are specifying a relative timecode for video with a timecode track, use these formats: HH:MM:SS:FF@RR | HH:MM:SS:FF | HH:MM:SS;FF If you are specifying an absolute timecode or timestamp value, you can also use these formats if the material has a framerate; otherwise you must use one that can be converted to a time: HH:MM:SS.sss | HH:MM:SS:FF@RR Time is measured on a 24-hour clock. Time may include milliseconds (*sss*); frames (:|;FF) as DF (;) or NDF (:), and optional frame rate (FF@FPS), as an integer, decimal number, or rational (for example, 30000/1001). If FPS is not specified, it is inferred from the associated source. Rates supported: 23.976, 24, 25, 29.97, 30, 50, 59.94. Timecode references are applied to the timecode track specified in the associated Source element's file. When using DF timecode, the first 2 (29.97 fps) or 4 (59.94 fps) timecode numbers are dropped every minute except minutes divisible by 10. DF timecode is supported for frame rates of 29.97 and 59.94 frames per second. Example: `<Title offset = "00:00:05.000"... />`
size (optional)	An integer value describing the nominal height of the font in points (pt). Default: 36 points. Example: `<Title size = "72pt"... />`
style (optional)	Specifies the style in which the font face is rendered. Default: *normal*. Keywords: *normal* | *italic* | *underline* | *strikeout* | *overline* Example: `<Title style = "italic"... />`
vertical-align (optional)	Specifies where the text is rendered, relative to the area. Default: *top*. Keywords: *top* | *middle* | *bottom* Example: `<Title vertical-align = "bottom"... />`
weight (optional)	Specifies the thickness of the stroke of each character. Default: *normal*. Keywords: *normal* | *bold* | *light* | *medium* | *heavy* Example: `<Title weight = "medium"... />`
wrap (optional)	Specifies how text is treated when it extends beyond the area: wrap it or not. Default: false. Boolean: (*true* | *false*) Example: `<Title wrap = "false"... />`

Example

In this example, a *Title* is added to a *Segment*, illustrating every attribute, plus an *Area* and *Shadow* element.

This frame shows the use of the *Area* Element to Position the text on the frame.



Composition

Updated 9/3/14 \\Phobos\Source Media\Post Producer\web_examples\CML\lower-third-with-text.cml

```
...
<Segment>
  <Title align = "head" adjust = "edge" fill = "none" layer = "2"
duration = "00:00:04.00" offset = "00:00:01.000" font =
"Helvetica" size = "50pt" style = "italic" weight = "bold"
foreground-color = "bisque" background-color = "transparent" wrap
= "false" horizontal-align = "left" vertical-align = "middle"
layout = "stretch">
    <Area top = "70%" left = "20%" bottom = "90%" right = "80%" />
    <Shadow color = "gold" softness = "10%" vertical-offset = "6%"
horizontal-offset = "6%" />
Post Producer Titles
  </Title>
...
```

Translation

CML Elements | CML Hierarchy Map

One *Translation* element may optionally be added to a *Body*, *Head*, or *Tail*.

Add a *Translation* element when you want to animate visual material along the X/Y axes over time. Positive values move the frame to the right/bottom. Negative values move the frame to the left/top.

The Head Translation X/Y values determine the original placement of the frame; the Body X/Y values specifies that target point that the frame is moved to during the Head duration. Similarly, the Tail X/Y values are the target that the frame is moved to from the Head X/Y source, during the Tail duration.

For this effect to be of practical value, the Fade element must be present in a *Head* or *Tail* element, and its duration must be greater than 0. The animation occurs during the Head and Tail duration, not during the Body duration.

The *Translation* element is often used in conjunction with the *Scaling*. See *Rotation* for animating material on the Z axis.

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

See also *Volume*, *Opacity*, *Scaling*, *Rotation*.

Attributes

Although both of these attributes are optional, supplying neither would be impractical, and result in no change to the location.

Name	Description
x (optional)	Specifies the horizontal translation relative to the left edge of the frame, as a ratio (no designator), a percent (%), or as an absolute position in pixels (px). Default: 0. Example: `<Translation y = "9%" x = "16%"/>`
y (optional)	Specifies the vertical translation relative to the top edge of the frame, as a ratio (no designator), a percent (%), or an absolute position in pixels (px). Default: 0. Example: `<Translation y = "9%" x = "16%"/>`

Example

In this example, *Video* 1 is placed at 0,0 in the *Head*, and translated to 50,50 (the top-left corner is placed dead-center on the frame) in the *Body*, over a 5-second duration.

Composition

```
...
<Video source = "1" layer = "1" >
  <Head>
```

```
    <Edit time = "00:00:00.00"/>
    <Opacity level = "50%" />
    <Scaling x = "100%" y = "100%" />
    <Translation x = "0%" y = "0%" />
    <Fade duration = "00:00:05.00" />
  </Head>
  <Body>
    <Opacity level = "100%" />
    <Scaling x = "50%" y = "50%" />
    <Translation x = "50%" y = "50%" />
    <Volume level = "0%"/>
  </Body>
  <Tail>
    <Edit time = "00:00:08.00"/>
  </Tail>
<Video>
...
```

Video

CML Elements | CML Hierarchy Map

The *Video* element is a material element that adds video (with *Audio*, if present) to a *Segment*, from a file identified by a specific *Source*, using the *source* attribute (required).

A segment's length is defined by the duration of the longest material element within the segment. A primary *Video* element (usually at layer 0) typically defines the intended length; other image, title or video overlays that are shorter in length (or the same) are superimposed on the underlying video.

Picture-in-picture effects can be created by scaling and translating video clips on upper layers within the segment.

Note: Material items may logically be divided into three temporal parts: the beginning (*Head*), the middle (*Body*), and the end (*Tail*) for the purpose of changing the presentation of the material with the effect elements (*Volume*, *Scaling*, *Translation*, *Rotation*, *Fade*), and applying a *Fade* to define the temporal length of the part.

Child Elements

One each of these elements may be added to apply effects or alignment:

- *Head*
- *Body*
- *Tail*
- *Burn-in*

Attributes

The *adjust*, *align*, and *fill* attributes are closely related. Together, they specify how the material is aligned, and how it is adjusted along the timeline in the segment.

Name	Description
adjust (optional)	When the material's duration is less than the duration of the segment, the material may be extended within the segment. This provides a flexible method to define the behavior of overlay material so that it is consistently applied, regardless of the duration of the underlying material. For example, using the *adjust* attribute, you can consistently display a branding bug on underlying video of any duration. There are 3 things to consider when modifying material: 1. What part to adjust—the body or the edge (using *adjust*) 2. Which end to adjust from (based on *align* attribute) 3. How to display—hold, loop, none (based on *fill* attribute). Keywords: *edge* (default) | *body* *edge* (default)—The material is adjusted by extending the edge that is *not* identified by the *align* property and is applied to the duration of the head and tail of the material. *body*—The material is adjusted by extending the body of the material. It is applied to the end of the body that is *not* identified by the *align* property. Example: `<Video adjust = "body"... />`
align (optional)	Specifies which end (or both) of the material to anchor to the timeline of the segment. Keywords: *head* | *tail* | *both.* Example: `<Video align = "tail"... />` *head* (default)—Anchors the beginning of the material to the beginning of the segment. *tail*—Anchors the end of the material to the end of the segment. *both*—Anchors the center of the material at the center of the segment, so that the Head and Tail are equi-distant from the beginning and end of the segment. **Note**: If an *offset* is used in this situation, it is also applied to both ends.

Name	Description
fill (optional)	Keyword. When the duration of the material is less than the length of the segment, the fill attribute determines if and how the item continues to play: Keywords: *none* (default) | *hold* | *loop* Example: `<Video fill = "loop"... />` There are 3 things to consider when modifying the material: 1. What to adjust—the body or the edge (using *adjust*) 2. Which end to adjust from (based on *align* attribute) 3. How to adjust—hold, loop, none (based on *fill* attribute) *none* (default)—extend with empty media (no visual or auditory material). From a practical perspective, no extension. *hold*—continue playing the first or last frame the entire duration of the segment. If *align* = *"head"*, the last frame is repeated. If *align* = *"tail"*, the first frame is repeated. *loop*—play the media repeatedly for the duration of the segment.

fill-duration (optional, applies to fill only)	Time. When *fill="loop"* is specified, the *fill-duration* loops the video for a duration which may be shorter or longer than the video Source. Drop frame and non-drop frame references are valid.
	Examples:
	`<Video fill = "loop" fill-duration = "00:00:10" />`
	`<Video fill = "loop" fill-duration = "00:27;10" />`
	`<Video fill = "loop" fill-duration = "00:00:10.000" />`
	`<Video fill = "loop" fill-duration = "00:00:15;03@29.97" />`
	In this example, the time value is longer than the Source: The Source is a 3-second video overlay, and *fill-duration* is set to 8 seconds. The generated media loops the Source 2 times (for a total of 6 seconds), and then plays for 2 seconds and stops.
	`<Video fill = "loop" fill-duration = "00:00:08" source = "2" layer = "1">`
	In another example, the time value is shorter than the *Source* (no tail time is required): The Source is a 30 second video overlay, and the *fill-duration* is set to 15 seconds. In this case, the generated media plays the video for 15 seconds then stops.
	`<Video fill = "loop" fill-duration = "00:00:15" source = "2" layer = "1">`

filter (optional)	Keyword. When a *Video Source* is used in a *Segment*, the *Audio* is implicitly processed with the video. You can apply this attribute and keyword to mute the associated audio for Video source. Keyword: *mute*. Example: `<Video filter = "mute"... />`

Name	Description
layer (optional)	Integer. Describes the ordinal position in the composite of all material in the segment. Default: 0. Visual material with a higher layer value obscures material with a lower layer value in the same segment. Example: `<Video layer = "10"... />`
offset (optional)	Specifies a time value which changes the location of the beginning or end of the material, relative to the segment. The offset is applied to the end of the content that is specified by the align property. If align = "both" the offset is applied at the defined value to both the head and tail of the material. Default: 0. If you are specifying a relative timecode for video with a timecode track, use these formats: HH:MM:SS:FF@RR | HH:MM:SS:FF | HH:MM:SS;FF If you are specifying an absolute timecode or time value, you can also use these formats if the material has a frame rate; otherwise you must use one that can be converted to a time: HH:MM:SS.sss | HH:MM:SS:FF@RR Time is measured on a 24-hour clock. Time may include milliseconds (*sss*); frames (:|;FF) as DF (;) or NDF (:), and optional frame rate (FF@FPS), as an integer, decimal number, or rational (for example, 30000/1001). If FPS is not specified, it is inferred from the associated source. Rates supported: 23.976, 24, 25, 29.97, 30, 50, 59.94. Timecode references are applied to the timecode track specified in the associated Source element's file. When using DF timecode, the first 2 (29.97 fps) or 4 (59.94 fps) timecode numbers are dropped every minute except minutes divisible by 10. DF timecode is supported for frame rates of 29.97 and 59.94 frames per second. Example: `<Video offset = "00:10:00"... />`

Name	Description
rate (optional)	Specifies the frame rate of the video within a composition, relative to its original speed. May be specified as a percent (% units designator), or as a ratio (no units designator.) You can not specify an absolute speed. Used for changing the duration of a video clip, such as shortening the time required to roll credits in a promo. When the original rate is changed, volume (if any) is set to 0. Example: `<Video rate = "250%"... />` In this example, this video is played at 2.5 times the original speed. **Note:** The modified rate also applies to any effects applied to the video element. For example, slowing a clip to 50% doubles the duration of fade-ins and fade-outs.
source (required)	Integer, corresponding to the Source element's identifier value, to identify the file to be used in this instance.

Examples

Example 1

This CML snippet illustrates the extensive use of a video clip in a segment.

This example is the second segment in the sequence; it plays both video 1 and 2, and cross-fades the two, by using *Edit*, *Opacity*, and *Fade* elements in concert with each other.

```
...
<Segment>
  <Video source = "1" layer = "0">
    <Head>
      <Edit time = "00:00:04.000"/>
      <Opacity level = "100%"/>
    </Head>
    <Body>
      <Opacity level = "100%"/>
    </Body>
    <Tail>
      <Edit time = "00:00:05.000"/>
      <Fade duration = "00:00:01.000"/>
      <Opacity level = "0%"/>
      <Volume level = "0%"/>
    </Tail>
  </Video>
  <Video source = "2" layer = "1">
    <Head>
      <Edit time = "00:00:00.000"/>
      <Opacity level = "0%"/>
      <Volume level = "0%"/>
    </Head>
    <Body>
      <Opacity level = "0%"/>
      <Volume level = "0%"/>
    </Body>
    <Tail>
      <Edit time = "00:00:01.000"/>
      <Opacity level = "100%"/>
      <Volume level = "100%"/>
      <Fade duration = "00:00:01.000"/>
    </Tail>
  </Video>
</Segment>
...
```

Example 2

This example illustrates the use of cross-fades.

```
<Composition created = "2013-06-26T10:16:34.1293676-07:00" xmlns =
"Telestream.Soa.Facility.Playlist">
  <Source identifier = "1">
    <File location =
"\\Server\Share\1080i_2997_source\video_10MIN_SAMPLE.mov" />
  </Source>
  <Source identifier = "2">
    <File location =
"\\Server\Share\1080i_2997_source\PRORES_1080i_29.97.mov" />
  </Source>
  <Sequence layer = "0">
<!--source 1 in point at 40 seconds and play for 5 seconds until
out point at 45 seconds -->
    <Segment>
      <Video source = "1" layer = "0">
        <Head>
          <Edit time = "00:00:40.000"/>
        </Head>
        <Tail>
          <Edit time = "00:00:45.000"/>
        </Tail>
      <Video>
    </Segment>
    <!-- source 1 in point at 45 seconds at 100% video for seamless
playback from first segment -->
```

```
    <Segment>
      <Video source = "1" layer = "0">
        <Head>
          <Edit time = "00:00:45.000"/>
          <Opacity level = "100%"/>
        </Head>
        <Body>
         <Opacity level = "100%"/>
        </Body>
        <!-- source 1 out point at 55 seconds, fade to 0% video -
crossfade setup of layer 0 here, layer 1 below will become visible
as this layer fades to 0 and next layer begins fading from 0 -->
        <Tail>
          <Edit time = "00:00:55.000"/>
         <Fade duration = "00:00:05.000"/>
         <Opacity level = "0%"/>
         <Volume level = "0%"/>
        </Tail>
      </Video>
      <Video source = "2" layer = "1">
    <!-- source 2 in point at 35 seconds at 0% video on layer 1 -
this plays clear on top of layer 0 for 5 seconds, at which point
the transition begins. This video block total duration must match
the duration of the previous video block in this segment. -->
        <Head>
          <Edit time = "00:00:35.000"/>
         <Opacity level = "0%"/>
         <Volume level = "0%"/>
        </Head>
        <Body>
           <Opacity level = "0%"/>
           <Volume level = "0%"/>
        </Body>
    <!-- source 2 out point at 45 seconds at 100% video fades from
clear to 100% video on top of layer 0 beginning at 5 seconds into
the out point - crossfade from layer 0 to layer 1 occurs here, this
fade duration should match fade duration of the previous video
block in this segment - for equal transition times -->
      <Tail>
          <Edit time = "00:00:45.000"/>
         <Opacity level = "100%"/>
         <Volume level = "100%"/>
         <Fade duration = "00:00:05.000"/>
        </Tail>
      </Video>
    </Segment>
  <!-- source 2 in point at 45 seconds at 100% video for seamless
playback from first segment of layer 1 - continue playing at 100%
video here -->
    <Segment>
      <Video source = "2" layer = "1">
        <Head>
          <Edit time = "00:00:45.000"/>
          <Opacity level = "100%"/>
          <Volume level = "100%"/>
          </Head>
        <Tail>
```

```
          <Edit time = "00:00:55.000"/>
        </Tail>
      </Video>
    </Segment>
  </Sequence>
</Composition>
```

Example 3

In this example, Source 2 (the credit roll) is resized, repositioned, and sped up to play in half the time of its original duration.

When rate is employed, frames are added to or removed from the clip to achieve the desired duration, at the frame rate specified for the entire composition. For example:

- Source 2 is 2 seconds long with a frame rate of 15 FPS for a total of 30 frames
- The composition output frame rate is 59.94 FPS
- The rate setting is =200% for this source.

```
<Composition xmlns = "Telestream.Soa.Facility.Playlist">
  <Source identifier = "1">
    <File location = "\\share\path\Background.mov" />
  </Source>
  <Source identifier = "2">
    <File location = "\\share\path\creditroll.mov" />
  </Source>
  <Sequence>
    <Segment>
      <Video align = "head" adjust = "edge" fill = "none" source =
"1" layer  = "0" />
      <Video align = "head" adjust = "edge" fill = "none" source =
"2" layer = "1" rate = "200%">
        <Head>
          <Scaling x = "30%" y = "30%"/>
          <Translation x = "60%" y = "60%"/>
        </Head>
        <Body>
          <Scaling x = "30%" y = "30%"/>
          <Translation x = "60%" y = "60%"/>
        </Body>
        <Tail>
          <Scaling x = "30%" y = "30%"/>
          <Translation x = "60%" y = "60%"/>
        </Tail>
      </Video>
    </Segment>
  </Sequence>
</Composition>
```

During encoding, the source is converted to 120 frames (2 seconds at 60 FPS) and the rate setting is applied to reduce the total frame count to 60 so that it plays for 1 second: 50% of the original duration.

Volume

CML Elements | CML Hierarchy Map

The *Volume* element enables you to adjust the volume of audio in certain material.

One *Volume* element may optionally be added to a *Body*, *Head*, or *Tail* in a *Video* or an *Audio* element.

The default *level* in a *Body* element is 100%, while in a *Head* or *Tail*, it is 0%, making it easier to apply fades. For this effect to work, the *Fade* element must be present, and its duration must be greater than 0. If you are using *Fade* for effects other than volume (that is, you do not want to fade the volume), you must supply a *Volume level* explicitly.

You can adjust volume by percent, or by decibels. Positive values apply a gain, negative values attenuate the audio. The level is applied equally to all channels targeted by default or by a *Mix*. (This element does not effect any channels targeted by a *Route*.)

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

See also *Opacity*, *Scaling*, *Translation*, *Rotation*.

Attributes

Name	Description
level (optional)	A positive or negative integer or decimal number representing the volume level as a ratio (no designator), percent (%) or in decibels (dB) of the original material. Default (in Head and Tail elements): 0%. Default (in Body elements): 100%. Example: `<Volume level=”30%” />`

Example

This example applies a 1-second audio fade-in and fade-out to the beginning and end of the clip.

Notice that in the *Head*, a *Volume* at 0% is specified, but in the *Tail*, it is not specified. Because `Volume level = "0%"` is the default value for a *Head* and *Tail*, you’re not required to provide it—it still works as intended—fading the tail to zero audio volume.

Composition

```
...
<Segment>
  <Canvas align = "head" adjust = "body" duration = "00:00:18.700"
Background = "black" layer = "0" />
  <Video source = "1" layer = "1">
    <Head>
      <Edit time = "00:00:50.300" />
      <Opacity level = "0%" />
      <Volume level = "0%" />
      <Fade duration = "00:00:00.500"/>
    </Head>
    <Body>
      <Opacity level = "100%" />
      <Volume level = "100%" />
    </Body>
    <Tail>
      <Edit time = "00:01:09.00" />
      <Opacity level = "0%" />
      <Fade duration = "00:00:00.500"/>
    </Tail>
  </Video>
...
```

Supported Post Producer Formats

This chapter identifies the supported input formats for both video and audio, and images, and the supported output formats for the Conform action.

Topics

Input Media Formats

The following tables list the video and audio formats that the Conform action can process, by container.

- Video Formats
- Audio Formats

Video Formats

Container	DV[1]	IMX	MPEG2[2]	ProRes & ProRes 4:4:4:4	DNxHD	J2K	H.264	AVCI	QT Anim.
TIFO	■	■	■	■	■				
AVI	■								
MOV[3]	■	■	■	■	■				■
MP4							■		
GXF	■	■	■			■		■	
LXF			■						
MPEG2 PS			■						
MPEG2 TS			■						
MXF OP-1a	■		■		■	■		■	
MXF AS02[4]	■		■			■		■	
MXF AS11								■	

1. DV includes DV25, DVCPro25, DV50/DVCPro50, and DVCProHD.
2. MPEG2 includes Long-GOP and I-Frame only formats.
3. QuickTime formats include alpha channel support.
4. Files produced by Omneon Media Subsystem (6.1) and Amberfin ICR (7.8) were tested.

Audio Formats

Container	PCM	MPEG	302M	331M	AAC	AC3[1]	Dolby E
TIFO	■					■	■
AC3						■	
AVI	■						
LXF	■						
MOV	■				■		■
MP4	■				■		■
GXF	■						■
MPEG2 PS		■					
MPEG2 TS	■	■	■		■	■	■
MXF OP-1a	■			■			■
MXF AS02[2]	■						■
MXF XDCAM HD	■						
MXF AS11	■					■	■
WAVE	■						■

1. AC3 includes Dolby Digital and Dolby Digital Plus (E-AC3)
2. MPEG2 includes long GOP, I-Frame, XDCAM, XDCAM HD

Graphic File Input Formats

The supported raster image formats are:

- BMP
- DPX
- JPEG
- TIFF
- PNG
- TGA
- PSD—Single-layer Adobe Photoshop files with transparency

Alpha channels are supported.

Output Video Formats

The conform action is generally intended to create mezzanine format outputs to be re-wrapped or transcoded downstream, so some media generated directly from Conform may not play correctly in a particular player. For example, the QuickTime Player has issues playing mp4 output directly from Conform. Switch has issues playing some DNxHD output, and Conform will encode essences that are not supported in QuickTime player, such as XDCAM and MPEG. Ultimately, the final production output should be generated by a downstream transcode action.

The Conform action produces video files in these container types:

- TIFO
- MP4
- MOV

The conform action can encode video in the following formats:

- AVCI
- ProRes
- DNxHD (not supported in MP4)
- DNxHR
- JPEG2000
- MPEG
- x264
- Uncompressed
- XDCAM

Notes

- QuickTime and MP4 containers require that the Closed Captions filter for the Transcoder in Conform is explicitly enabled.
- The QuickTime container requires specific tracks to be enabled in its container settings: timecode, 608 captions, and 708 captions (which also includes other VANC oriented metadata such as CGMSA, VCHIP, AFD and Timecode).
- The MP4 container does not support metadata by itself, but some essences (ProRes, MPEG, x264, and XDCAM as of PP6.3.3CD) will embed metadata.
- Uncompressed and JPEG2000 support captions and metadata output in TIFO and MOV containers.
- No metadata in JPEG2000/TIFO in TIFO unless filters are explicitly enabled is a known issue.

Output Audio Formats

The following table lists the audio formats that the Conform action can produce, and their containers.

Container	PCM	AC3	AAC	MPEG1 Layer 2
TIFO	■	■	■	■
MP4	■	■	■	■
MOV	■	■	■	■

DolbyE Program and Channel Specifications

The table below identifies each Dolby E program ID, the sequence of programs, and the channel positions for the programs.

Program	Sequence	Channel Positions							
0	5.1+2	0L	0C	0Ls	1L	0R	0LFE	0Rs	1R
1	5.1+1+1	0L	0C	0Ls	1C	0R	0LFE	0Rs	2C
10	4+4	0L	0C	1L	1C	0R	0S	1R	1S
11	4+2+2	0L	0C	1L	2L	0R	0S	1R	2R
100	4+2+1+1	0L	0C	1L	2C	0R	0S	1R	3C
101	4+1+1+1+1	0L	0C	1C	3C	0R	0S	2C	4C
110	2+2+2+2	0L	1L	2L	3L	0R	1R	2R	3R
111	2+2+2+1+1	0L	1L	2L	3C	0R	1R	2R	4C
1000	2+2+1+1+1+1	0L	1L	2C	4C	0R	1R	3C	5C
1001	2+1+1+1+1+1+1	0L	1C	3C	5C	0R	2C	4C	6C
1010	1+1+1+1+1+1+1+1	0C	2C	4C	6C	1C	3C	5C	7C
1011	5.1	0L	0C	0Ls	0R	0LFE	0Rs		
1100	4+2	0L	0C	1L	0R	0S	1R		
1101	4+1+1	0L	0C	1C	0R	0S	2C		
1110	2+2+2	0L	1L	2L	0R	1R	2R		
1111	2+2+1+1	0L	1L	2C	0R	1R	3C		
10000	2+1+1+1+1	0L	1C	3C	0R	2C	4C		
10001	1+1+1+1+1+1	0C	2C	4C	1C	3C	5C		
10010	4	0L	0C	0R	0S				
10011	2+2	0L	1L	0R	1R				
10100	2+1+1	0L	1C	0R	2C				
10101	1+1+1+1	0C	2C	1C	3C				
10110	7.1	0L	0C	0Ls	0Bsl	0R	0LFE	0Rs	0Bsr
10111	7.1 screen	0L	0C	0Ls	0Lc	0R	0LFE	0Rs	0Rc